EQUIVALENTS NOTES

Introduction

In order to prevent orthographical variations from figuring in the collation of a set of manuscript transcriptions, it is necessary to standardise scribal word-forms through a process known as normalisation or regularisation. A *scribal form* is the particular spelling of a word as written by a scribe, whereas the corresponding *conventional form* is the spelling the same word would have in a standard edition. A conventional form should always be derivable from a word found in a standard lexicon, using the normal rules of word formation. (I am indebted to Dr Shelly Harrison of the Linguistics Department of the University of Western Australia for his help with this terminology.)

The standard edition referred to in this thesis is the fourth edition of the United Bible Societies' *Greek New Testament* (UBS4). The standard lexicon is the *Greek-English lexicon of the New Testament and other early Christian literature* by Bauer, Arndt, Gingrich, and Danker (BAGD). Occasionally, other lexicons (e.g., Liddell and Scott, Lampe) have been consulted where scribal forms do not have corresponding entries in BAGD.

I have carried out normalisation by means of an equivalents table that consists of two columns. The first column is a list of over 2000 scribal forms compiled by a program that found every word in the transcriptions which did not appear in the text of the standard edition of Hebrews. The second column is a list of corresponding conventional forms which I have specified. Prior to collation, every instance of a manuscript word (or, sometimes, words) which occurs in the first column is replaced by the corresponding entry in the second column. That is, every entry in the first column is replaced with its specified equivalent on a global basis:

ADIKEIAIS	ADIKIAIS
ADIKIAN	ADIKIAN
AFLARGUROS	AFILARGUROS
ANASTRES	ANASTAUROUNTES
AQETHSEIS	AQETHSEIS

DUNATH	DUNATH
EPEUX	IEREUS
KE	KE

The list is sorted according to the order of Roman letters. As a result, words that begin with the letters *chi*, *phi*, *gamma*, *theta*, *omega*, and *zeta* are not sorted according to the order of Greek letters. This is because these letters are represented by Roman letters that do not have equivalent positions to the Greek letters they represent.

Some scribal forms represent normal patterns of spelling transformation as exemplified in Gignac's 1975 volume on phonology. For example, ADIKEIAIS is obtained from the conventional ADIKIAIS by EI/I interchange. The corresponding entry in the equivalents table is:

ADIKEIAIS	ADIKIAIS

Scribal contractions and shorthand forms are expanded:

ANASTRES	ANASTAUROUNTES

Scribal errors are resolved by specifying the correct conventional forms. In some cases this is straightforward:

AFLARGUROS	AFILARGUROS

In more difficult cases, it is not possible to be certain of the word which suffered scribal corruption. When this occurs, I have attempted to choose the most likely candidate:

EPEUX	IEREUS

Sometimes there is more than one possible interpretation of a scribal form. For example, KE may stand for KAI or KURIE. There is no alternative but to leave such forms as they stand:

KE	KE

Scribal forms are frequently conventional forms as well. Such a scribal form can be a *genuine textual variant* — one which is different to the standard text but grammatically appropriate in its context. It is not necessary to specify a replacement for a genuine textual variant, and it may be removed from the table. Alternatively, it may be retained. If retained, its replacement must be the word itself:

ADIKIAN ADIKIAN

A scribal form may be a *spurious textual variant* — one which Greek grammar does not permit in its context. A spurious variant cannot be replaced with the form that would be appropriate to the context, as this practice could produce unpredictable results. This is because every occurrence of the scribal form would be replaced with the specified equivalent — which would have to be a different word — regardless of context. The probability of inadvertent replacement with an inappropriate word is small, as the only words that can be affected are those which do not occur in the standard text. However, prudence demands a procedure which minimises the potential for error. Therefore, a spurious textual variants is not replaced with its locally appropriate conventional form but is treated in the same way as a genuine textual variant — it is specified as its own replacement:

AQETHSEIS AQETHSEIS

This word occurs in U16 at Heb 7.18. If its equivalent were specified to be the appropriate word for this context, AQETHSIS, then AQETHSEIS would be replaced with AQETHSIS wherever it appeared in a transcription, prior to collation. In the process, any legitimate occurrences of AQETHSEIS would be replaced with AQETHSIS — an undesirable result.

Another example of spurious textual variation is found in U75-1 at Heb 10.1:

DUNATH DUNATH

The context requires DUNATAI, not DUNATH. Even though this is most

probably a case of H/AI interchange, to specify DUNATAI as equivalent to DUNATH would cause all valid occurrences of DUNATH to be replaced with DUNATAI.

Spurious textual variants can be spelling variants or scribal errors. The contracted form shown above is a case in point. Whereas the correct word for the context is ajnastaurou'nta", the scribe has mistakenly written the nominative case.

The collation output will contain spurious textual variants as a consequence of this prohibition upon global replacement by a different word for any scribal form which happens to be a conventional form. For this reason, a list of such variants has been accumulated during the construction of the equivalents table. With the list in hand, the offending variants can be easily identified and eliminated from further consideration if necessary.

Spurious variants may creep into the collation output by another route. If a word which is contextually inappropriate somewhere among the transcriptions appears in the standard text then it will not be listed in the first column of the equivalents table and will escape detection. Under a global approach to normalisation, the only completely reliable means of isolating such words is to check every word in the collation output against each of the transcriptions: an onerous task.

From a programming perspective, the global approach is easier to implement because the location of replacements does not need to be considered. Unfortunately, this approach also creates the previously mentioned problems, and can have other undesirable consequences if not used with caution. For example, one could fearlessly regularise KS to KURIOS on a global basis, but replacing KE with KURIE should only be done on a case by case basis because some scribes spell KAI as KE. A local replacement strategy is therefore superior, and is precisely that employed in Dr Robinson's interactive *Collate* program. (*Collate* also allows the specification of global replacements.) If not for my wish to perform multivariate analysis on spelling variants, *Collate* would have been ideal for the task at hand. It already incorporates a facility for the output of genuine textual variations in a form suitable for multivariate

analysis.

Returning to the equivalents table, I have added a third column which gives short descriptions of the nature of the variation in the first column, and a fourth column which gives a reference to the first occurrence of the listed scribal form that was encountered by the listing program. A fifth column gives translations of the scribal forms, usually for genuine textual variants alone. All but the first two columns are ignored for normalisation purposes.

Descriptive terms of the third column include: 'illegible' for transcribed words which incorporate question marks standing for illegible letters; 'punctuation' for alternative punctuation marks; 'division' for alternative word divisions; 'X/Y' for orthographical variations in which the scribal form represented by 'X' replaces the conventional form represented by 'Y'; 'error' for words which are erroneous, usually through scribal mistakes; 'optional' for words with optional conventional forms; 'variant' for genuine textual variants; 'spurious' for spurious textual variants; 'contraction' for scribal contractions and compendia; 'numeral' for letters used to represent numbers; 'iota adscript' for the addition of an *iota* adscript; and 'movable *nu*' for the omission of *nu* in dative plurals ending in *iota*, and third person verbs ending in *iota* or *epsilon*.

Particular descriptions may contain combinations of these terms. In the case of spurious variants, the description includes the likely true cause of variation in brackets. Complex spelling variations may be separated into their components, especially where diaeresis is concerned (e.g., J/EI -> J/I, I/EI). An 'ambiguous' category is reserved for scribal forms which have more than one possible interpretation. For example, KE can mean KAI, as it does at Heb 1.1 in P12, or KURIE, as it does at most other places.

The potential for misinterpretation of scribal forms is ever present. Whereas an unambiguous table of equivalents is required in order to normalise the transcriptions, many of the decisions upon which the table is based are not clear-cut. There are many instances where alternative interpretations of a scribal form exist. By way of illustration, the variation EGKATALIPONTES, which occurs at Heb 10.25 in U1, may be a

strong aorist form or simply a spelling variation of EGKATALEIPONTES (through EI/I interchange).

Ambiguous cases like this one have generally been classified as textual variations so that they will be reproduced in the collation. In this way, others will have the opportunity to consider whether or not they are genuine variants. An exception to this approach is made for the frequent interchange of infinitive -AI and imperative -E endings. Whereas the infinitive can function as an imperative ('the imperatival infinitive': BDF, 1961, §389), the more likely cause of the confusion is AI/E interchange, which is the most common type after EI/I interchange (Gignac, 1975, 192). Therefore, variations of this kind are generally classified as spurious — their cause being attributed to orthographical interchange.

Another potentially vexatious issue concerns whether a given scribal form is grammatically permissible in its context. Such problems can present a challenge which can hardly be overcome by anyone but a native koinhv speaker. Lacking such expertise, I have employed various helps in an attempt to make reasonable choices in ambiguous cases. The most frequently consulted reference works include Wachtel and Witte's *Das Neue Testament auf Papyrus* 2/2, Gignac's two volume *Grammar of the Greek papyri of the Roman and Byzantine periods*, Bagster's *Analytical lexicon*, the Friberg's *Analytical Greek New Testament*, and standard lexical and grammatical works such as BAGD and BDF. Other works consulted are found in the bibliography below. My supervisor, Dr Moore, provided valuable guidance in cases which I found particularly challenging. I am most grateful for his assistance in these places. The relevant entries in the following notes are marked by the initials [RKM]. (Dr Moore should not be blamed for the conclusions at which I have subsequently arrived.)

Less straightforward equivalents table entries are briefly discussed below, along with every case of spurious textual variation which I have been able to identify. Each discussion is preceded by a reference line which gives:
(1) a scribal form;
(2) the specified conventional equivalent;
(3) a short description of the mode of variation; and

(4) a reference to the place where the scribal form *first* occurs in my transcriptions (as detected by the program which produced the list of variations from the standard text).

The notes use the lunate *sigma* (ç) instead of the conventional forms of *sigma* (s, ") when reference is made to scribal forms. The reader is directed to a corresponding transcription note when more relevant information is to be found there. Where English translations of particular passages are given, the wording is often adapted from the *Revised Standard Version* (1946).

Notes

ADELFHN	ADELFHN	variant	U142 8.11

Perhaps a woman is responsible for this?

AFEREIN	AFAIREIN	E/AI	U15 10.4

AFEREIN could be a spelling variant of AFAIREIN or AFELEIN. The first possibility has been chosen because E/AI interchange is more common than R/L interchange.

AFERIN	AFAIREIN	E/AI, I/EI	U1-0 10.4

The original scribe of Codex Sinaiticus changed AFELIN to AFERIN. AFERIN could be a spelling variation of AFAIREIN or AFELEIN (AFELIN being assumed to be a spelling variant of AFELEIN). In my opinion, a scribe would be more likely to make a textual rather than an orthographical change to his or her own work. Therefore, AFERIN has been classified as a spelling variant of AFAIREIN. This, in turn, makes it a textual variant of AFELEIN. (Tischendorf seems to have been surprised at this alteration. See the corresponding transcription note.) [RKM]

AFESEIS	AFESEIS	spurious (EI/I)	P46 9.22

This is probably a case of EI/I interchange. The feminine noun should

agree in number with the singular verb. As AFESEIS is a conventional form, the appropriate substitution (AFESIS) cannot be made.

AFOMOIWMENOS AFWMOIWMENOS O/W P46 7.3

This could be a variant spelling of the present passive participle, ajfomoioumevno".

AF' EIREIN AFAIREIN division, error U150 10.4

A compound verb with the ajf' prefix would require a rough breathing on the verbal stem. The lexicons do not list any verbs beginning with eiJr- which could give rise to the inflected form ajfeirein.

AF' OMOIOUMENOSAFOMOIOUMENOS division, variant U150 7.3

ajfomoioumevno" is the present passive participle. The usual reading, ajfwmoiwmevno", is the perfect passive. Wachtel and Witte (1994) do not list this as a variant reading.

AF' OMOIWMENOS AFWMOIWMENOS division, O/W U75 7.3

The same comment applies here as in the entry for AFOMOIWMENOS above.

AISTHRIA AISQHTHRIA error U278 5.14

This could be a noun based on aji?ssw *move rapidly* but it is more likely to be a scribal error.

AKOUSHTAI AKOUSHTAI spurious (AI/E) U150 4.7

As AKOUSHTAI is a conventional form it cannot be replaced with the appropriate word for this context, which is AKOUSHTE. This is most probably a spelling variation due to AI/E spelling interchange.

ANALOGISASQAI ANALOGISASQAI spurious (AI/E) P13 12.3

As the imperative (-SASQE) and not the infinitive is appropriate, this is likely to be a spelling variation due to AI/E interchange.

ANAMIMNHSKESQAI ANAMIMNHSKESQAI spurious (AI/E) P46 10.32

The imperative (-ESQE) is required.

ANASTREFESQE ANASTREFESQE variant U2 13.18

This has been classified as a textual variant and not as a case of E/AI interchange because: (1) ANASTREFESQE can makes sense here: *in all things wishing you conduct yourselves well*; and (2) U2 has a habit of substituting AI for E, not vice versa.

ANASTRES ANASTAUROUNTES contraction, spurious (error) P46 6.6

This is classified as a spurious variant because the accusative is required. If the nominative was read, PARADEIGMATIZONTAS would have to be nominative as well.

ANTEKATESTHTE ANTIKATESTHTE optional U142 12.4

Compound verbs with two prefixed prepositions sometimes take an augment after each (Black, 1988, 82).

ANTIKATESTHTAI ANTIKATESTHTE AI/E U25 12.4

There is is no such conventional form as ejsthtai .

APEDOTO APEDETO optional M2815 12.16

Both are third person singular strong aorist indicative middle forms of ajpodivdwmi. Kubo (1975, 322) gives -OTO and Guillemette (1986, 37)

gives -ETO.

APEDWTO	APEDETO	W/O, optional	U150 12.16

This is a case of W/O interchange with APEDOTO. The conventional equivalent of APEDOTO is APEDETO.

APIQHSASEIN	APEIQHSASIN	I/EI, EI/I	P13 3.18

The participle form excludes the possibility of an infinitive ending. Note that this scribal form exhibits in the same word two opposite directions of EI/I interchange relative to the conventional spelling.

APOBALHTAI	APOBALHTAI	variant	U151 10.35

The third-person, singular, subjunctive, middle (-HTAI) can be read in this context: *Therefore, he should not throw away your confidence*. Wachtel and Witte (1994) omit this variation.

APODEKATOIN	APODEKATOUN	error	P46 7.5

The normal vowel contraction rule O + EI = OI appears to have been employed rather than the exception to the rule which should be used for the infinitive (Wenham, 1965, 240).

APOLEIPETE	APOLEIPETE	spurious (E/AI)	U142 4.6

This has been classified as spurious because an impossible construction results from replacing APOLEIPETAI with APOLEIPETE: *Since, therefore, you remain [for] some to enter into it*.

APOLUSIN	APOLUSIN	variant	P46 11.35

It is possible to read APOLUSIN for APOLUTRWSIN: *not having accepted release*.

APWLIA	APWLEIA	I/EI, spurious (error)	U1-3 10.39

The accusative is required.

AQETHSEIS	AQETHSEIS	spurious (error)	U16 7.18

The singular noun is required.

ARWN	AARWN	error	P46 7.11

This variation has been classified as an erroneous spelling because P46 normally has AARWN.

ARCIEREAN	ARCIEREA	error	U4 3.1

According to Gignac (1977, 85), the accusative singular is exclusively -ea.

ASFALHN	ASFALH	optional	U2 6.19

This accusative singular of s-stem masc./fem. adjectives occasionally ends in -n (Gignac,1977, 135).

ASPASASQAI	ASPASASQAI	spurious (AI/E)	U1 13.24

The substitution of -AI for -E in imperatives is unusual for this manuscript (in Hebrews at least), suggesting that this may be a case of the infinitive functioning as an imperative (see BDF, 1961, §389).

ASQENEIA	ASQENEIA	variant	U44 5.2

The dative is permissible here: *since he himself is encompassed by infirmity*.

BEBAIOUSQE	BEBAIOUSQE	spurious (error)	U1 13.9

This is more likely to be a scribal error than a spelling variation. It is one of the rare instances of this scribe replacing the infinitive with an imperative. A likely cause is the close antecedence of PARAFERESQE.

BLEPETAI	BLEPETAI	spurious (AI/E)	P13 3.12

The passive indicative is not grammatically sound in this context: *You are seen, brothers, lest there be in any of you an evil unbelieving heart.*

COLH	COLH	variant	P46 12.15

P46 has EN COLH (or ENCOLH) instead of ENOCLH. Katz has conjectured ejn colh'/ for ejnoclh/' based on Dt 29.17 LXX (BAGD, 1979, 883). This would result in the following: *Looking diligently lest any lack the grace of God — lest any bitter root spring up in gall — and by this many be defiled*. It is quite possible that ENCOLH is an error in P46 caused by the scribe inadvertently transposing the *omicron* and *chi*.

CRONIEI	CRONISEI	optional	TR 10.37

The third person singular future of cronivzw can be spelled croniei' (BDF, 1961, §74(1)).

CWREIS	CWREIS	spurious (EI/I)	U2 10.28

Although the scribe of U2 uses this spelling consistently, normalisation cannot be performed because CWREIS is a conventional form.

DEDEKATWTE	DEDEKATWTAI	E/AI	U142 7.9

The reduplicated prefix and the *tau* in the suffix rule out the possibility of this word being anything but DEDEKATWTAI.

DEHSIS	DEHSIS	spurious (error)	U3 5.7

The accusative is required at Heb 5.7.

DEICQENTAN	DEICQENTA	optional	U6 8.5

Gignac (1977, 45) writes in relation to third declension endings that, 'The

acc. sg. of consonant stems very frequently ends in -an'.

DH	DH	spurious (H/EI)	U150 2.1

Replacing dei' with dhv results in the loss of the main verb from this clause. The variation is probably due to H/EI interchange.

DIAKOUONTES	DIAKOUONTES	variant	U6 6.10

From diakouvw, *give someone a hearing*.

DIALEGETE	DIALEGETAI	E/AI	U2 12.5

This is a deponent verb so it cannot take the -ETE suffix.

DIAMENHS	DIAMENHS	variant	U151 1.11

The substitution of the subjunctive is grammatically permissible even if it is not theologically sound: *They shall perish but you should remain*. This variation is likely to be due to H/EI interchange. The same interchange is the likely cause of the variation in U150 noted in the entry for DH above. Both U150 and U151 are now located at Patmos.

DIAQHKOS	DIAQHKHS	error	U142 10.29

This error seems to have occurred by haplography. (See the corresponding transcription note.)

DIHNEGKES	DIHNEKES	G/-	P46 10.12

The insertion or omission of medial nasals before stops was a common phenomenon in Roman and Byzantine times (Gignac, 1975, 119).

DIKAIOSUNH	DIKAIOSUNH	variant	U6 11.33

It is permissible to have this dative rather than the accusative here.

DIWKETAI DIWKETAI spurious (AI/E) U15 12.14

The sentence does not make sense with a passive verb.

DOKEIDE DOKEITE D/T U6 10.29

Reading DOKEI DE results in nonsense. (See the corresponding transcription note.)

DOKEIN DOKEIN spurious (error) U150 12.11

Reading the infinitive (*to seem*) instead of the third person singular (*seems*) results in nonsense: *but discipline for the present to seem not to be joyous but grievous*. In any case, the infinitive is negated with mhv, not ouj (BDF, 1961, §426). [RKM]

DOKEITAI DOKEITAI spurious (AI/E) P46 10.29

This passage does not make sense with the third person passive rather than the second person active verb: *Of how much worse a punishment <u>it is thought</u> he shall be counted worthy?*

DUNAMEIN DUNAMIN EI/I U6 11.11

The present infinitive of dunamovw is dunamou'n.

DUNAMIS DUNAMIS spurious (I/EI) P46 6.5

The accusative is required here.

DUNANTE DUNANTAI E/AI U2 10.1

The -TE suffix cannot occur with a deponent verb which does not have *theta* in the suffix.

EAUTOS EAUTOS spurious (error) P46-1 6.6

The reflexive pronoun cannot take the nominative case. (Others transcribe this word as EAUTOIS. See the corresponding transcription note.)

EBEBAIWQHI	EBEBAIWQH	error	U142 2.3

The aorist indicative passive does not take the *iota* adscript.

ECETAI	ECETAI	spurious (AI/E)	P13 10.36

This clause does not make sense if e[cete is replaced with e[cetai. [RKM]

EECONTES	ECONTES	error	U18 4.14

See the corresponding transcription note.

EFUGAN	EFUGON	optional	U6 12.25

'The endings of the first aorist are very frequently substituted for those of the second aorist' (Gignac, 1977, 335).

EGEIRE	EGEIRE	spurious (E/AI)	U2 11.19

This passage does not make sense if the infinitive (EGEIRAI) is replaced with the imperative (EGEIRE). The variation is probably due to E/AI interchange.

EGKAINISTAI	EGKEKAINISTAI	error	U142 9.18

Whereas this has been classified as erroneous, it may be an optional spelling. Gignac (1977, 242) writes, 'Reduplication is sometimes omitted in compounds and in simple verbs or is sometimes replaced by the syllabic augment.' [RKM]

EGKATALIPONTES	EGKATALIPONTES	variant	U1 10.25

This variation may be the strong aorist of EGKATALEIPONTES or a case

of I/EI interchange. I have elected to classify it as a textual variation.

EGKLEIYOUSIN	EKLEIYOUSIN	error	U2 1.12

There is no such word as kleivpw which could give rise to this scribal form. It is conceivable that the verb klevptw could produce a word such as ejgklevyousin but klevptw is not listed as taking the ejg- prefix (see Liddell and Scott).

EICAMEN	EICOMEN	optional	P46 12.9

This is an example of the transfer of first aorist endings to the imperfect (Gignac, 1977, 332).

EICAN	EICON	optional	P46 11.15

The same comment applies here as for the preceding entry.

EIDESOUSIN	EIDHSOUSIN	E/H	U6 8.11

The -sousin ending restricts the range of possible inflections to the third person future indicative, the conventional form of which is eijdhvsousin. [RKM]

EIDOUSIN	EIDHSOUSIN	error	U3 8.11

I have classified this as a spelling error (due to haplography). It only occurs in the first hand of U3 and has been subsequently altered to eijdhvsousin. It could be a spelling variation of eijdovsin (dative participle) or eijdw'sin (subjunctive), but only the second of these permits a sensible construction: *because all should know me*. [RKM]

EILASKESQE	ILASKESQE	EI/I, spurious (E/AI)	U2 2.17

The imperative does not make sense here.

EIPA	EIPON	optional	U142 3.10

ei|pon sometimes takes first aorist endings (BAGD, 1979, 226).

EISFERETE	EISFERETE	spurious (E/AI)	U243 13.11

The imperative does not make sense here.

EKCUSIAS	EKCUSEWS	error, variant	P46 9.22

The usual word here is the first declension aiJmatekcusiva". P46 has the analogous ekcuçiaç, whereas the correct form is the third declension e[kcusew" (BAGD, 1975, 247).

EKDIKHSEIS	EKDIKHSEIS	variant	U1 10.30

The nominative plural is permissible here. This could be a spelling variation of ejkdivkhsi" brought about by EI/I interchange.

EKFEUXOMAI	EKFEUXOMAI	spurious (error)	U142 2.3

The plural verb is required to agree with the preceding plural pronoun.

EKFEUXWMEQA	EKFEUXWMEQA	spurious (W/O)	P46 2.3

This is not likely to be a future subjunctive (cf. BDF, 1961, §28; Westcott and Hort, 1881b, Appendix, 172; and Robertson, 1923, 324). It is probably due to W/O interchange instead.

EKLELHSQAI	EKLELHSQAI	spurious (AI/E)	P13 12.5

It would be necessary to have a preceding auxilliary verb or another infinitive in the immediate context for the infinitive ejklevlhsqai to make sense here. [RKM]

EKLELUSQE	EKLELUSQE	spurious (error)	U18 12.5

This word should be ejklevlhsqe *you have completely forgotten*. The substitution of ejklevlusqe results in nonsense: *you have been wearied the exhortation which he addresses to you as sons.* The scribe may have been influenced by ejkluvou further along in the same verse.

EK' LELHSQAI	EKLELHSQAI	division, spurious (AI/E)	U150 12.5

See the comment for EKLELHSQAI, above.

ELABWN	ELABON	W/O	U25 11.36

The augment rules out the possibility of this being a strong aorist participle.

ELALH	ELALEI	H/EI	U142 4.8

Guillemette (1986, 131) lists ejlavlh as an alternative spelling of ejlavlei.

ELATTWMENON	HLATTWMENON	E/H	U142 2.9

This word may be a present participle. However, one would expect the spelling to be ejlattoumevnon. The following liquid (i.e., L) provides the right conditions for E/H interchange at the beginning of the word (Gignac, 1975, 243).

ELATTWSAS	ELATTWSAS	variant	U6 2.7

The nominative participle makes sense in this context: *[God] making him somewhat inferior to the angels for a short time*. [RKM]

ELEWS	ELEOS	W/O	U142 8.12

The adverb derived from e[leo" is ejlehmovnhw" (Liddell and Scott, 1968, 531). [RKM]

EM	EN	M/N	P13 11.34

This *nu* is assimilated to *mu*, being conditioned by the following *pi* of polevmw/.

EMAPEDIA	AMEMPTOS	error	U6 8.7

Tischendorf (1852, 593) reads emapedia whereas Wachtel and Witte (1994) have emapeaia. Neither is a word. (See the corresponding transcription note.)

EMELLEN	HMELLEN	optional	U151 11.8

This word may be spelled with *epsilon* or *eta* (BAGD, 1979, 500).

EMISANTO	EKOMISANTO	error	U6 11.39

It is possible that this word was intended to be ejmivshsanto (*they were loved less*) rather than ejkomivsanto (*they received*). However, reading ejmivshsanto reduces the verse to nonsense.

EMMESW	EN MESW	optional	U25 2.12

This word sometimes substitutes for ejn mevsw/ (BAGD,1979, 255).

ENDOKIMASAN	EDOKIMASAN	error	U6-2 3.9

According to Tischendorf, the second corrector of U6 failed to delete the *nu* when correcting this word. (See the corresponding transcription note.)

ENDOTERON	ENDOTERON	variant	U142 6.19

This word means *innermost* (Lampe, 1961, 468).

ENEDEIXASQAI	ENEDEIXASQE	AI/E	U1-1 6.10

This word occurs at Heb 6.10 in both U1 and U6. The agreement of these manuscripts suggests that an infinitive may have been intended.

However, the *epsilon* augment would not be expected if that were the case.

ENERGEIS ENERGEIS spurious (EI/H) U150 4.12

As the second person singular verb does not make sense in the context, this word was probably intended to be the adjective ejnerghv".

ENESTOTA ENESTHKOTA error U142 9.9

This is unlikely to be a spelling variation of ejnestw'ta. The preceding article requires a masculine singular here, not a neuter plural.

ENKATALEIPW EGKATALEIPW N/G, spurious (EI/I) P46 13.5

The emphatic negative future requires the aorist subjunctive ejgkatalivpw to go with the double negative ouj mhv.

ENNUWN ENNOIWN U/OI P13 4.12

According to Gignac (1975, 197), U/OI interchange is the third most common mode after EI/I and AI/E .

ENOCLEI ENOCLEI spurious (EI/H) U151 12.15

The present indicative ejnoclei' makes sense in the context. However, the negative mhv requires the subjunctive ejnoclh'/.

ENPEGMATWN EMPAIGMWN N/M, E/AI, error U6 11.36

The scribe appears to have followed the pattern of a third declension neuter noun such as o[noma.

ENTOLH ENTOLH spurious (error) U3 7.16

The genitive, ejntolh'", is required here.

EPAKOHN EPAKOHN variant U150 5.8

This word does not appear in BAGD, Lampe, or Liddell and Scott. Liddell and Scott (1968, 605) does list ejpakouov", ovn (*attentive to*). I have treated ejpakohvn as an analogue of uJpakohvn and, therefore, translate it as *attentiveness*.

EPEUX IEREUS error P46 5.6

Zuntz (1953, 253) calls this variation an *error mirus* — a surprising error. Royse (1981, 301-3) surveys various theories put forward to explain its genesis, including the view that ejpeux, which does not occur in the lexicons, may be related to a word such as ejpeuvch, which means *kneeler*. He concludes that, 'All of this is to say that we truly have here an error mirus.'

It seems more likely that the scribe (or a predecessor) made an error of sight — seeing EPEUX for IEREUS in the exemplar — than that IEREUS was intentionally replaced with an otherwise unattested word.

EPHGGELLETAI EPAGGELLETAI error U150 12.26

This word is erroneous because the tenses in which the *alpha* lengthens to *eta* should also be spelled with a single *lambda*. It is unlikely to be a spelling variation because the interchange of *alpha* and *eta* only occurs sporadically according to Gignac (1975, 286). Likely candidates for the correct form of the word are ejpaggelletai (*it is promised*), ejpaggellete (*you (pl.) promised*), and the usual reading ejphggeltai (*he has promised*). Of these, what is written is most closely conformed to ejpaggelletai. The third person verb fits the context better than the second person ejpaggellete. However, ejpaggellete is found in U48.

The choice comes down to deciding whether two scribes each had a single variation relative to ejpaggelletai — one a mistake and the other a spelling transformation — or whether one scribe had two variations from

ejpaggellete. In my opinion, with the balance so evenly poised, the contextually superior possibility is the most probable: that is, ejpaggelletai. Consequently, ejphggelletai in U150 is classified as an erroneous version of ejpaggelletai, and ejpaggellete in U48 is classified as a spelling variation of ejpaggelletai.

EPIGNWSIAN	EPIGNWSIN	error	U1 10.26

The scribe originally wrote THS EPIGNWSIAN. The ending indicates the accusative case but the definite article is genitive. The lexicons do not list any words which could give rise to an accusative of the form ejpignwsian.

EPIKALEISQE	EPIKALEISQE	spurious (E/AI)	U150 11.16

ejpikalei'sqe (*you (pl.) call*) does not make sense in the context.

EPILAMBANETE	EPILAMBANETE	spurious (E/AI)	U2 2.16

The imperative does not fit the context.

EPILANQANESQAI	EPILANQANESQAI	variant	U142 13.2

The infinitive makes sense here.

EPILEIYI	EPILEIYEI	I/EI	U1 11.32

A verb is required and not a noun such as ejpileiyin.

EPILHNYI	EPILEIYEI	H/EI, N/-	U6 11.32

This appears to be an example of the insertion of a nasal before a stop (see Gignac, 1975, 118).

EPISKOPOUNTAS	EPISKOPOUNTAS	variant	P46 12.15

This is part of a reading which is peculiar to P46. It is possible to make sense of it provided that the preceding contraction k—"—is interpreted as the accusative plural kuvriou".

EPITREPEI	EPITREPEI	variant	U150 6.3

This spelling may be due to EI/H interchange rather than an inflectional change. I regard it to be an inflectional change because the same word occurs in U20 and U25.

ERANTHSEN	ERANTISEN	optional, H/I	U18 9.19

This may be spelled with one or two *rhos*. The double *rho* spelling conforms to the rule that aspirated *rhos* should be doubled if shifted from the initial position through inflection. However, 'The doubling cannot be carried out in the NT without doing strong violence to the oldest MSS' (BDF, 1961, §11(1)). According to Gignac (1977, 246), the variation is purely orthographical and the *rho* is usually not doubled. Moulton and Howard (1928, 193) write, 'the single r prevails over the double.' Westcott and Hort (1881b, Appendix, 163) write, 'In most cases verbs beginning with r do not double the r after the initial ej of the augmented tenses… Usually the evidence for the single r is overwhelming'.

It is difficult to say whether the single or double *rho* spelling should be printed in standard editions. The oldest manuscript evidence is for the single *rho* whereas the rule, which the eloquent author of Hebrews might have been inclined to follow, is for the double *rho*. I have opted for the single *rho* spelling of BAGD (1979, 734), which is also used in UBS4 and Westcott and Hort. NA27 has the double *rho* spelling: the only instance of a spelling difference between NA27 and UBS4 which I have noticed.

ERANTISMENOI	RERANTISMENOI	optional	U75-1 10.22

This spelling is peculiar to U75, although a number of other MSS have ejrrantismenoi. These were legitimate spellings during the Roman and Byzantine periods (Gignac, 1977, 246). I have adopted rJerantismenoi

as the conventional form. BAGD (1979, 734) and BDF (1961, §68) support this spelling, while Westcott and Hort, UBS4, and NA27 print it. According to Moulton and Howard (1928, 192-3), this verb displays analogical reduplication (rJer-). They continue on to say, 'Verbs in rJ. usually redupl[icate] ejrr… The rr was ultimately made single, to resemble other augments.'

ERCHTE	ERCHTAI	E/AI	U6 13.23

The -HTE suffix cannot coexist with this deponent verb stem. The corresponding aorist passive would have the strong aorist stem: hjluqhte.

ERHMIAS	ERHMIAS	variant	U48 11.38

This apparent variant may be due to a typographical error in Heath's transcription of U48. (See the corresponding transcription note.)

ERRANTISMENOI	RERANTISMENOI	optional	TR 10.22

See the note for ERANTISMENOI, above.

ESWMAI	ESOMAI	W/O	U150 1.5

This has been classified as a spelling variation as it is unlikely to be a future subjunctive. (See the entry for EKFEUXWMEQA above.)

EUAGGELISMENOI	EUHGGELISMENOI	optional	U142 4.2

I have classified this as an optional spelling because Gignac (1977, 242) writes 'Reduplication is occasionally omitted in compounds'. The present middle/passive participle would be eujaggelizomevnoi. Rather than being an optional spelling, this may be an error or a case of A/H interchange. This mode of interchange is rare. I have not found any examples listed in Gignac (1975), although Robertson (1923, 184) gives a few. [RKM]

EUARESTEITE	EUARESTEITE	spurious (E/AI)	U6 13.16

The singular subject makes it impossible to construct this sentence in a grammatically sound manner with the plural eujarestei'te.

EUARESTITAI	EUARESTEITAI	I/EI	U1 13.16

This could, instead, be equivalent to eujaresth'tai through I/H interchange.

EUDOKHSEI	EUDOKHSEI	spurious (error)	U44 10.38

This is the third person singular future indicative active, giving the translation: *my soul shall not take pleasure in him*. [RKM]

EUDOKIEI	EUDOKEI	error	U18 10.38

I have classified this scribal form as an erroneous transcription of eujdokei'. Lampe (1961, 563) lists eujdokiavw as an optional form of eujdokevw used for the sake of metre. However, the corresponding inflection would be eujdokia/', not eujdokiei'. [RKM]

EUHGGELISQENTES	EUAGGELISQENTES	error	U16 4.6

This has been classified as an erroneous rather than an optional spelling even though it occurs in a number of manuscripts. The lengthening of *alpha* to *eta* only occurs in the perfect, in which case the -qente" suffix would be incorrect. The perfect eujhggelismevnoi in 4.2 has probably influenced the scribes here.

EUHRESTHKENAI	EUARESTHKENAI	optional	M2815 11.5

Either spelling is acceptable, although 'Indirect compounds with euj- tend to augment a following short vowel' (BDF, 1961, §69(4)). Along with the superior manuscript attestation, this suggests that eujhresthkevnai (which is the form printed in BAGD) should be

adopted for the standard editions.

EULOGEITE EULOGEITE spurious (E/AI) U150 7.7

The singular to; e[latton requires a singular verb.

EULOGITE EULOGEITE I/EI, spurious (E/AI) U6 7.7

See the entry for EULOGEITE.

EURISKETO HURISKETO optional M2815 11.5

According to Gignac (1977, 240), 'Verbs beginning with eu- usually retain the eu- unaugmented, but the regular augmented forms in hu- are found increasingly frequently in later Roman and Byzantine documents.' In this case, however, the majority of earlier MSS have huJrivsketo.

EUROMENOS EURAMENOS optional U142 9.12

The first aorist suffix is well attested in the middle voice of this verb (BDF, 1961, §81(3)).

EYHLAFHMENW EYHLAFHMENW variant U44 12.18

I have taken this to be the perfect passive participle: *For you have not come to what has been touched*. Verbs beginning with *psi* reduplicate by prefixing *epsilon*, as exemplified by the perfect of ywriavw (BAGD,1979,894). [RKM]

EZHTEITW EZHTEITO W/O U75 8.7

The O/W interchange can take place in the -w verb suffix according to Gignac (1975, 276).

FENOMENON FAINOMENON E/AI, spurious (O/W) U25 11.3

The context requires a genitive plural participle.

FERESQE	FERESQE	spurious (E/AI)	U2 9.16

Reading the imperative reduces the sentence to nonsense. This scribal form is probably due to E/AI interchange in the infinitive suffix.

FHSEI	FHSIN	EI/I, movable *nu*	P46 8.5

This occurs due to a corruption in P46. (See the corresponding transcription note and the entry for GRA, below.) Given that this is an inflection of fhmiv and that the EI/I spelling change is relatively common, I regard fhsivn to be the most likely equivalent.

FOBHQHS	FOBHQHS	variant	U151 11.27

Whereas this is probably a case of H/EI interchange, the aorist passive subjunctive fobhqh/"' can make sense in the context: *By faith Moses left Egypt that you should not be fearful [of] the anger of the king.*

FULAKAIS	FULAKAIS	spurious (error)	P46 11.36

This clause does not really make sense if the genitive singular is replaced with the dative plural:

(1) genitive singular: *and others received a trial of mockings and scourgings, moreover of bonds and of imprisonment*;

(2) dative plural: *and others received a trial of mockings and scourgings, moreover of bonds and by imprisonments*.

FWNHN	FWNHN	variant	P46 12.19

Whereas a dative is expected here, the accusative is permissible.

GEGENHKA	GEGENNHKA	error	U142 5.5

This scribal form is probably a spelling error. 'We find in the papyri and

inscriptions of the Hellenistic age … a tendency to double and a counter-tendency to drop one of the elements in a double' (Moulton and Howard, 1928, 101). According to Gignac (1977, 300), both gevgona and gegevnhmai are used for the perfect of givnomai. Competing -k- and root perfects are found in some verbs (297), but I have not found any references to a -k- perfect of givnomai such as gegevnhka. [RKM]

GEGONATAI	GEGONATE	AI/E	U150 5.11

The perfect passive would be gegevnhtai.

GEGUMNASMENHS	GEGUMNASMENHS	variant	U20 12.11

This has been classified as a variant, even though the resultant sentence is somewhat flawed. It is acceptable for the participle to refer to aujth'" instead of toi'". However, this makes it necessary for toi'" to play a role that would be better filled by aujtoi'". [RKM]

GENAMENOS	GENOMENOS	optional	U44 11.24

There is sporadic confusion of first and second aorist endings in this verb (BDF, 1961, §81(3); Gignac, 1977, 344).

GENAMENWN	GENOMENWN	optional	P46 9.11

See the preceding comment.

GENEISQAI	GENHSQE	EI/H, AI/E	U25 6.12

As no conventional forms have an -hsqai suffix, it is safe to say that this scribal form is a spelling variation of gevnhsqe, provided that -ei- is a spelling variant of -h- and not -e-.

GENHSQAI	GENHSQE	AI/E	U16 6.12

The infinitive would end with -esqai.

GENHTE	GENHTAI	E/AI	U6 2.17

As this is a deponent verb, a *theta* should precede a genuine -hte suffix.

GEUSAMENOS	GEUSAMENOS	spurious	U278 6.4

The accusative is required here.

GEUSHTE	GEUSHTAI	E/AI	U15 2.9

See the entry for GENHTE.

GEWRGEITE	GEWRGEITE	variant	U6 6.7

The usual reading is, *and produces useful vegetation for the ones on whose account it is also being cultivated* (gewrgei'tai). The clause also makes sense with gewrgei'te: *and produces useful vegetation for the ones on whose account you also cultivate*.

GHRASKWN	GHRASKWN	spurious (W/O)	U151 8.13

The context requires a neuter participle.

GINWSKON	GINWSKONTES	error	U1 10.34

The singular neuter participle does not fit the context.

GNWQEI	GNWQI	EI/I	U6 8.11

This could only be the irregular aorist imperative gnw'qi.

GNWQH	GNWQI	H/I	U150 8.11

The aorist subjuctive passive would be gnwvsqh/.

GRA	GAR	error	P46 8.5

The scribe of P46 wrote GRAFHSEI, but it is hard to say what is meant by it. Sanders and Kenyon regard GRA as a corruption of GAR, and so divide what is written as GRA FHSEI. (See the corresponding transcription note.) Taking this approach leaves what the scribe meant by fhsei to be explained. It may be fhsivn with EI/I spelling interchange and the movable *nu* omitted. Alternatively, the scribe may have been thinking of some inflection of gravfw. Wachtel and Witte (1994) print the reading as a single word, grafhsei. According to Liddell and Scott (1968, 360), the passive future can be written as grafhvsomai. Perhaps, by analogy, the scribe of P46 regarded grafhvsei rather than gravyw as the future active. Whereas neither explanation is without problems, I am inclined to settle for the former one as being the least improbable.

GRAYWN	GRAYWN	variant	U75-1 10.16

This clause makes sense if gravywn is read instead of ejpigravyw: *and having written them upon their minds*. [RKM]

HBOULHQHS	EBOULHQHS	optional, variant	U16 10.8

This variant results from dividing the single word HBOULHQHSAN found in Sanders' transcription into two words, HBOULHQHS AN. (See the corresponding transcription note.) The resulting sentence may be translated, *You expressed a [conditional] wish above, saying that you neither desired nor took pleasure in sacrifices and offerings, burnt offerings, and sacrifices for sins*. hjboulhvqh" is an optional spelling of ejboulhvqh" (BAGD, 1979, 146). [RKM]

HCON	HCON	spurious (H/EI)	U150 11.15

The clause does not make sense if the verb ei\con is replaced with this noun.

HDUNASQHSAN	HDUNHQHSAN	optional	P13 3.19

These are alternative spellings of the same word, with hjdunavsqhsan being from the Ionic dialect (Moulton and Howard, 1928, 234).

HLAT'TWSAS	HLATTWSAS	punctuation	P46 2.7

A mark has been placed between the two *taus* to distinguish them from a *pi*. For want of a better category, this has been classified as a punctuation variant.

HQHNAI	GENHQHNAI	error	U150 5.5

See the corresponding transcription note.

HRGASANTO	EIRGASANTO	optional	P13 11.33

According to Gignac (1977, 236), the augment of this verb fluctuates between ei- and h-, with both forms having some historical justification.

HRHNIKON	EIRHNIKON	error	P13 12.11

This may be a case of H/EI interchange rather than an error, although its subsequent correction suggests that it was not a spelling variation.

HTHS	HTIS	H/I	U150 8.6

The accenting suggests that this is a spelling variation of h{ti". (See the corresponding transcription note.)

HUDOKHSAS	EUDOKHSAS	optional	U15 10.6

The augmented form of verbs beginning with eu- varies between hu- and eu-, with the former preferred in classical Attic and the latter being dominant in New Testament times (BDF, 1961, §67(1)).

HUHRESTHKENAI	EUARESTHKENAI	error	P13 11.5

The initial *eta* is doubtful. If the scribe did write hu-, then the alteration

to eu- shows that it was regarded as erroneous. Accordingly, I have classified this as a spelling error, even though it may have been a valid double-augment. I have adopted the unaugmented eujaresthkevnai as the conventional spelling. (See the note for EUHRESTHKENAI, above.)

HULOGHKEN	EULOGHKEN	optional	U6 7.6

As verbs beginning with eu- can be augmented or unaugmented, the same may be said for reduplication (Gignac, 1977, 232, n. 1).

HULOGHSEN	EULOGHSEN	optional	U150 11.20

See the note for HUDOKHSAS above.

IAI	KAI	error	U4 6.9

See the corresponding transcription note.

IAQEI	IAQH	iota adscript, E/H	U151 12.13

I have classified this as E/H rather than EI/H spelling interchange because I have assumed that the final *iota* is an adscript. *Epsilon* can substitute for *eta* in the final position (Gignac, 1975, 243).

IDON	IDON	spurious (I/EI)	U151 11.23

Replacing ei\don with the neuter participle ijdovn results in nonsense: *By faith Moses, when he was born, was hidden for three months by his parents because [the little child] seeing that the little child was beautiful, and they were not afraid of the king's edict*. [RKM]

IHS	IHSOUS	contraction	P46 10.10

The other manuscripts have Ôlhsou' Cristou' at 10.10.

ILASGHRION	ILASTHRION	error	U3-3 9.5

The scribe who re-inked Codex Vaticanus only partially retraced the *tau* to produce a *gamma.* (See the corresponding transcription note.)

IREI	OREI	error	U142 12.18

This is probably the result of a scribal slip. (See the corresponding transcription note.)

ISAK	ISAAK	optional	U6 11.9

The spelling Isavk is consistently used in P46 and D (BAGD, 1979, 380).

ISASIN	ISASIN	variant	P46 9.6

The initial *iota* is uncertain. (See the corresponding transcription note.) Even with the *iota*, this variation borders on being nonsensical: *These things having thus been prepared, the priests always know in the first tent, performing their duties*.

JDEN	EIDEN	J/I, I/EI, error	U75 11.23

I have taken this to be a spelling variation of ei\den (*he saw*). It is classified as an error because the sentence requires a plural verb here. It is possible that JDEN represents another word. Liddell and Scott (1968, 817) list ijdevn, but note that it is of uncertain meaning.

IDOSAN	EIDON	I/EI, optional	U278 3.9

Ei[dosan is an optional spelling of ei\don.

JDON	IDON	J/I, spurious (I/EI)	U150 11.23

See the entry for IDON above.

JEQAE	IEFQAE	J/I, error	P46 11.32

This word has been corrected by a subsequent scribe.

JEREAN	IEREA	J/I, error	U20 10.21

A *nu* has mistakenly been added to the accusative form, which should be iJereva. According to Gignac (1977, 85), 'The acc. sg. is exclusively -ea.'

JKEISIAS	IKESIAS	J/I, EI/E, variant	U16 5.7

Examples of the EI/E interchange before *sigma* are given in Gignac (1975, 256).

JSAK	ISAAK	J/I, optional	P13 11.9

See the comment for ISAK above.

JSCUI	ISCUJ	J/I, I/J, variant	U2 9.17

The sentence remains viable if the verb ijscuvei is replaced with the dative noun ijscuvi>: *since [it is] never <u>with force</u> as long as the one who made it is alive*. This could be a spurious variation brought about by I/EI interchange in the last syllable. Wachtel and Witte (1994) do not list this variation. [RKM]

JSCUJ	ISCUJ	J/I, variant	U75 9.17

The same applies as for the preceding entry. The diaeresis on the last syllable makes it less probable that this word is a spurious variation due to I/EI interchange.

KALIN	KALIN	spurious (I/EI)	U6 2.11

This is likely to be a spelling variation of the infinitive, but must be classified as a spurious variation because kalivn could be a word in its own right. Liddell and Scott (1968, 867) give kaliv" as equivalent to

skevparnon (*axe*) on the authority of the *Lexicographus* of Hesychius. Replacing the infinitive with this noun results in nonsense. [RKM]

KAMHTAI	KAMHTE	AI/E	U150 12.3

This could be the second aorist passive subjunctive kavmhtai, giving the reading, *so that he may not grow weary*.

KAQARIZESQE	KAQARIZESQE	spurious (E/AI)	U75 9.23

The sentence does not make sense if the infinitive is replaced with the imperative.

KAQARIZETE	KAQARIZETE	variant	U2 9.22

This clause can be read with kaqarivzete: *under the Law you (pl.) purify almost everything with blood*. Even though this is not likely to be the intended meaning, it does make sense.

KAQE	KAQ	error	P46 6.13

According to Liddell and Scott (1968, 851), kavqe occurs as the last word of a sentence in Hesychius, although its meaning is not given. It may be an imperative based on kavqw, which Liddell and Scott (857) give as a barbarism of kaqivzw. If so, it does not make sense in this context: *he swore by himself, 'Sit.'* A more plausible explanation is that the scribe wrote the *epsilon* of the following eJautou' twice (Wachtel and Witte, 1994, 290).

KAQ' JSTATE	KAQISTATE	J/I, division, spurious	U150 8.3

Replacing the indicative with the imperative results in nonsense: *For appoint every high priest to offer gifts and sacrifices*. The same scribe spells this word correctly at 5.1.

KARKINHS	SARKINHS	error	U3 7.16

See the corresponding transcription note.

KATABALOMENOI	KATABALOMENOI	variant	U25 6.1

The single *lambda* indicates a second aorist participle.

KATADEILON	KATADHLON	EI/H	U2 7.15

There could be an adjective katadeilon (*cowardly*), but no such word is listed in Liddell and Scott.

KATAFEUGONTES	KATAFUGONTES	optional	M2815 6.18

I have classified this as an optional spelling because the word is derived from katafeuvgw and a number of manuscripts have the spelling katafeugovnte". Gignac says that the eu > u transition is probably due to scribal error (1975, 230).

KATAKAIETE	KATAKAIETE	spurious (E/AI)	U243 13.11

Replacing katakaivetai (*[they are] burned*) with katakaivete (*you (pl.) burned*) in this context results in nonsense.

KATANISKON	KATANALISKON	error	U56 12.29

See the corresponding transcription note.

KATANOHSATAI	KATANOHSATE	AI/E	U150 3.1

The suffix -satai does not occur in conventional forms.

KATASCOMEN	KATASCWMEN	O/W	U150 3.6

This can only be a spelling variation of katascwmen. The second aorist indicative would take the augment, resulting in katescomen.

KATASKEUASMENWN KATESKEUASMENWN error U151 9.6

The perfect participle suffix requires reduplication, so kata- should be kate-.

KATASKEUAZETE KATASKEUAZETE spurious (E/AI) U2 3.4

Reading kataskeuavzete for kataskeuavzetai results in nonsense: *for every house you (pl.) build by someone.*

KATEPAUSES KATEPAUSES spurious (error) P13 4.4

This clause does not make sense with katevpause": *And, O God, you were resting on the seventh day from all his works*.

KATHRTISO KATHRTISW O/W U151 10.5

The same comment applies here as in the entry for EZHTEITW above.

KATHRTISQE KATHRTISQE spurious (E/AI) U2 11.3

The clause does not make sense if kathrtivsqe is read instead of kathrtivsqai: *By faith we understand you (pl.) have been created the worlds*.

KATHRTISTAI KATHRTISTAI spurious (T/Q) P46 11.3

Reading kathrtivstai instead of kathrtivsqai results in nonsense: *By faith we understand it has been been created the worlds*. Similar examples of T/Q interchange are given by Gignac (1975, 87).

KE KE ambiguous M2815 1.10

This could be kaiv or kuvrie.

KEKAQERISMENOUS KEKAQARISMENOUS E/A U2 10.2

This A/E interchange before *rho* is noted in BDF (1961, §29(1)).

KEKLHMEMOI KEKLHMENOI error P46 9.15

I have classified this as an error because the replacement of *nu* with *mu* in a medial position is rare (Gignac, 1975, 119).

KEKOIMWMENOUS KEKOIMHMENOUS W/H, variant U278 9.13

Gignac does not give any examples of the H > W change, but does give one example of the W > H transformation (1975, 293). I think that this is a variant spelling because the present passive participle is spelled koimwvmenou". On the other hand, it could be an error of the type made by scribes who have fallen asleep.

KENHN KENHN variant U1 8.13

This may be a spurious variation caused by E/AI interchange. However, the fact that the same scribe wrote kainhn at 8.8 and kainhç at 9.15, indicates that he or she was not prone to this spelling change. Consequently, I have classified this as a genuine variant. It gives the following reading which cannot be rejected as grammatically incorrect, even though it defies the logic of the broader context: *By saying 'empty', he has made the first obsolete*.

KENHS KENHS spurious (E/AI) U6 9.15

The first hand of U6 wrote kenhn at 8.8 and 8.13, and kenhç at 9.15. Therefore, it is safe to assume that this is a spurious variation. Any other manuscripts which use the *epsilon* spelling of any of the inflections of this word must be individually checked to see whether they do so consistently. If they do, then the variation is probably spurious. If not, it may be a genuine variation, as in the preceding case.

KLHQHSETE	KLHQHSETAI	E/AI	U150 11.1

The -qhsete suffix does not occur in conventional forms.

KOKINOU	KOKKINOU	K/KK	U150 9.19

This may be no more than an error. Gignac (1975, 160) gives examples of the kk > k change.

KOMHSHSQAI	KOMISHSQE	H/I, AI/E	U25 10.36

The -shsqai suffix does not occur in conventional forms.

KOMISHSQAI	KOMISHSQE	AI/E	U15 10.36

The preceding comment applies here as well.

KOMISQHTE	KOMISQHTE	variant	U56 10.36

The passive voice of komivzw can mean *be sent* (Lampe, 1961, 767). The punctuation of the two manuscripts in which this variation occurs isolates komivsqhte th;n ejpaggelivan from the surrounding text: *You have been sent the good news*. [RKM]

Note on kreivss-, kreivtt-

Normalisation of scribal forms of kreivsswn and kreivttwn presents a problem. The standard text prints kreivss- in some places (e.g., 6.9), and kreivtt- in others. As inflections of one stem may appear where the standard text prints the corresponding inflection of the other, the potential exists for the normalisation step to produce apparent textual variation where there is no actual variation. This is because the collation program treats words which differ after the normalisation step as distinct. For this reason, care should be taken to exclude from subsequent analysis those cases in which a scribal form of kreivss- occurs where the standard

text has kreivtt-, or in which a scribal form of kreivtt- occurs where the standard text has kreivss-.

KREITON	KREITTON	error	U48 12.2

I cannot determine whether this is an error or a different spelling as the word does not occur elsewhere in Hebrews for this manuscript. The single *tau* spelling is not common among the transcribed manuscripts. Its only other occurrence is in M2815, where it seems to be an error. (That scribe normally uses the double *tau* spelling.) Accordingly, I have classified KREITON as an error, even though Gignac (1975, 161) gives a few examples of the tt > t spelling transformation.

KREITONOS	KREITTONOS	error	M2815 11.16

See the preceding comment.

KREITTONO	KREITTONOS	error	U1-2 8.6

See the corresponding transcription note.

KREITTONOSIN	KREITTONOS	error	U1 8.6

See the corresponding transcription note.

KREITTW	KREITTW	spurious (optional)	U150 1.4

This is probably an optional spelling of kreivttwn in which the *nu* has been omitted because the following word begins with a stop (cf. Gignac, 1975, 111). However, it must be classified as a spurious variant because kreivttw can be an accusative plural neuter (Guillemette, 1986, 247; Gignac, 1977, 151, note 7; BDF, 1961, §47(2)). The nominative singular masculine is required here. [RKM]

KRISEIS	KRISEIS	variant	U2 9.27

It is necessary to classify this as a variant because the nominative plural krivsei" makes sense (*and after this, judgements*). However, it is more likely to be a spelling variant because the scribe of U2 was in the habit of writing -eiç for the first person singular of such nouns. (See KTISEIS below.) Wachtel and Witte are probably right not to list this variation (1994, 325). [RKM]

KRISH	KRITH	error	U6 12.23

I have not found any reference to a mode of spelling variation in which a single *sigma* is interchanged with a single *tau*.

KRIT'TON	KREITTON	I/EI, punctuation	P46 11.40

Rather than being punctuation, this mark distinguishes the double *tau* from *pi*.

KRITIKOO	KRITIKOS	error	U3-3 4.12

See the corresponding transcription note.

KRITTW	KREITTW	I/EI, spurious (optional)	U6 1.4

See the entry for KREITTW above.

KTISEIS	KTISEIS	spurious (EI/I)	U2 4.13

The context requires a singular noun.

KUKLOQEN	KUKLOQEN	variant	P46 11.30

The sentence is grammatically acceptable with the adverb kuklovqen instead of the participle kuklwqevnta, even if it does violate the author's intention: *By faith the walls of Jericho fell from all sides for seven days*. Wachtel and Witte (1994) write '*sic*', which indicates that they do not regard this as a true variant. [RKM]

KUKLWQEN KUKLOQEN W/O, variant U150 11.30

The same applies here as in the preceding entry. According to Liddell and Scott (1968, 1006), this spelling was condemned by Theognostus in his *Canones*. Wachtel and Witte (1994) do not list U150 as supporting kuklovqen. [RKM]

LALALOUNTA LALOUNTA error U4 12.25

See the corresponding transcription note.

LALOUNTWN LALOUNTWN error U44 12.24

This variation, which only occurs in U44, has been classified as an error because there is no genitive plural antecedent for LALOUNTWN. (Punctuation separates KREITTON LALOUNTWN PARA TON ABEL from the surrounding text in this manuscript.)

LALEISQE LALEISQE spurious (E/AI) U2 2.3

The passive imperative does not make sense in the context.

LALHQHSHS LALHQEISHS H/EI U150 9.19

The -qhsh" suffix does not occur in conventional forms.

LAMBANON LAMBANON spurious (O/W) U150 7.9

The context requires a masculine participle.

LATREUEN LATREUEIN E/EI U18 9.14

The -en suffix does not normally occur without an augment.

LEGESQE LEGESQE spurious (E/AI) U2 3.15

The passive imperative does not make sense in the context.

LEGETE	LEGETE	spurious (E/AI)	U150 7.13

Reading levgete instead of levgetai results in nonsense: *For the one of whom these things you are said [is] of another tribe*.

LEGON	LEGON	spurious (O/W)	U25 9.20

The masculine participle, levgwn, is always used when introducing a quotation.

LEITOURGON	LEITOURGON	variant	U20 10.11

It is possible to read the noun leitourgovn instead of the participle leitourgw'n: *And every priest has established daily a minister, offering the same sacrifices frequently*. Wachtel and Witte (1994) do not list this. It may well be a spurious variation caused by O/W interchange. [RKM]

LELOUMENOI	LELOUSMENOI	optional	M2815 10.22

The spelling is optional, with *sigma* omitted in Attic (BDF, 1961, §70 (3)).

LEUEIS	LEUI	EI/I, optional	U1-3 7.9

Leuiv also occurs in the declined forms Leuiv" (nom.), Leuiv (gen.), and Leuivn (acc.) (BDF, 1961, §55(1)).

MACAIRAS	MACAIRHS	optional	M2815 11.34

This noun may be declined macaivrh", -rh/ or macaivra", -ra/ (BAGD, 1979, 496).

MAKROQUMOUNTAS	MAKROQUMOUNTAS	error	U6

6.12

This seems to be a scribal error. (See the corresponding transcription note.) The accusative plural participle is not appropriate to the context. [RKM]

MAKROU	MAKROU	variant	U56 8.11

Wachtel and Witte (1994) do not list this reading of the first hand of U56.

MARTUREITE	MARTUREITE	spurious (E/AI)	U1-1 7.17

The immediate context makes it difficult to read marturei'te in place of marturei'tai: *For you (pl.) testify that you (sing.) are a priest for ever in the order of Melchizedek*.

MEGA	MEGAN	error	U150 4.14

The adjective should be masculine to agree with ajrciereva. This scribal form is found in a number of manuscripts and is probably due to assimilation to the endings of the preceding and following words.

MEIMISQAI	MIMEISQAI	EI/I, I/EI, spurious (AI/E)	U6 13.7

See the comment for MIMEISQAI below.

MEINEI	MEINH	EI/H	U243 12.27

The *iota* in the first syllable rules out the possibility that this could be a present or future active, unless EI/E exchange has taken place as well.

MEMARTUREITAI	MEMARTURHTAI	EI/H	U151 11.5

The -etai prefix (which becomes -eitai here) does not occur with the reduplicated prefix in conventional forms.

MEMARTURHTE	MEMARTURHTAI	E/AI	U6 11.5

The reduplicated prefix eliminates the possibility of an -hte suffix.

MERISMOI	MERISMOI	spurious (error)	U16 2.4

The dative is required, not the nominative. [RKM]

METABALEIN	METALABEIN	error	P46 12.10

This is probably an error caused by scribal inversion. While metabavllw is a word, it only occurs in the middle voice (BAGD, 1979, 510).

METAQESEIS	METAQESEIS	spurious (EI/I)	U2 7.12

The context requires a singular noun.

METECON	METECON	spurious (O/W)	U151 5.13

The context requires a masculine participle.

METETEQHKEN	METEQHKEN	error	P46 11.5

This is a scribal error, probably due to dittography. The closest conventional form would be the perfect metatevqeiken.

MHPO	MHPW	O/W	U142 9.8

See the corresponding transcription note.

MHPWS	MHPWS	variant	U6 9.8

This reading makes sense provided that mhvpw" is understood to be introducing an indirect question (cf. BAGD, 1979, 519): *By this the Holy Spirit indicates that perhaps the way into the sanctuary has been revealed while the first tent is still standing*. It may be no more than a spurious variation due to the addition of a movable *sigma* (cf. Gignac,

1975, 126). [RKM]

MI	MH	I/H	P13 10.17

Gignac (1975, 236) lists miv as a spelling variant of mhv.

MIMEISQAI	MIMEISQAI	spurious (AI/E)	U1 13.7

The imperative is appropriate in this context, not the infinitive. [RKM]

MIMESQAI	MIMEISQAI	E/EI, spurious (AI/E)	U150 13.7

See the comment for MIMEISQAI.

MIMISQAI	MIMEISQAI	I/EI, spurious (AI/E)	U16 13.7

See the comment for MIMEISQAI.

MIMNHSKEI	MIMNHSKEI	spurious (EI/H)	U150 2.6

Deponent verbs do not take the -ei suffix. [RKM]

MIMNHSKESQAI	MIMNHSKESQAI	spurious (AI/E)	U1 13.3

The context requires the imperative, not the infinitive. [RKM]

MONOGENHN	MONOGENH	optional	U150 11.17

In adjectives of the third declension, the accusative singular masculine-feminine occasionally ends in *nu* (Gignac, 1977, 135).

NENOMOQETHTE	NENOMOQETHTAI	E/AI	U6 7.11

The reduplicated prefix rules out the possibility of this being anything but a spelling variation.

ODEN	OQEN	D/Q	U6 9.18

This mode of interchange is rare but not unknown (Gignac, 1975, 96).

ODWN	ODWN	spurious (error)	M2815 9.8

The scribe, apparently under the influence of the two preceding genitive plurals, has added the genitive plural ending where an accusative singular is required.

OGKWN	OGKWN	spurious (error)	P46 12.1

Reading o[gkwn instead of o[gkon results in nonsense: *since we are surrounded by so great a cloud of witnesses of weights, let us lay everything aside*. While this could be a spelling variation caused by O/W interchange, it is more likely to be a scribal error caused by repeating the genitive plural ending of the preceding word.

OLEQREUWN	OLOQREUWN	optional	P46 11.28

See BDF (1961) §32(1).

OLIGORH	OLIGWRH	O/W, spurious (H/EI)	U20 12.5

The clause violates the context if the subjunctive is read instead of the imperative: *And you have completely forgotten the exhortation which he addresses to you as sons – My son, he should not despise the discipline of the Lord; do not lose heart when being corrected by him*. (The second person passive subjunctive does not work either.)

OLIGWRH	OLIGWRH	spurious (H/EI)	U151 12.5

The same applies as for the preceding entry.

OLOQREUWS	OLOQREUWN	error	U56 11.28

The adverb would be ojlevqriw" and the perfect participle would be wjloqreukwv". This may be a case of interchange of -" and -n (cf.

Gignac, 1975, 131-2).

OPW	OUPW	O/OU	U48 12.4

Accented ou is frequently exchanged with o in the initial position (Gignac, 1975, 211-212).

ORESEI	ORESIN	EI/I, movable *nu*	U6 11.38

This may be an error instead of a spelling variation.

ORESN	ORESIN	error	U6-3 11.38

See the corresponding transcription note.

OSFRUOS	OSFUOS	error	U6 7.5

Perhaps the scribe was thinking of o[sfrhsew" when he wrote this.

OUDEI	OUDEIS	error	P46 12.14

The first hand wrote OUDEI, which may be interpreted in a number of ways:

(1) A variation of oujdeiv", *no one*, caused by error or omission of movable *sigma* (cf. Gignac, 1975, 125). A corrector has added the *sigma* to give oujdeiv", which is the usual reading. However, it is inappropriate here if the following KS stands for kuvrio" because o[yetai would then have two competing subjects.

(2) A spelling variation of oujdev (cf. Gignac, 1975, 257, section 1e). This gives a reading that is grammatically compatible with kuvrio": *without which not even a lord will see*. However, this mode of spelling transformation (ei > e) is rare and appears not to occur in P46.

(3) ouj dei', *it is not necessary*. This would require an associated

infinitive, which is lacking.

(4) oujdæ eij. This results in nonsense: *without which not even if a lord will see*.

Of these, only the second one is acceptable if KS stands for kuvrio". But the first is acceptable if the contraction stands for the accusative plural kuvriou". Then a sentence which includes oujdeiv" and ends with the accusative plural ejpiskopou'nte" (as indicated in the papyrus by an added reading mark) can be translated: *without which no one shall see overseeing lords*. The third and fourth possibilities are still grammatically unsound if kuvriou" is read.

While neither of the first two explanations is particularly satisfying, I regard the one which assumes that the first hand omitted the *sigma* from oujdeiv" and that the contraction stands for kuvriou" as the least implausible, mainly because the ei > e spelling transformation is rare in P46. [RKM]

OUPWS	OUPWS	variant	U142 12.4

The *sigma* is not certain. (See the transcription note.) According to Liddell and Scott (1968, 1272), ou[pw" is an adverb meaning *not at all*. Wachtel and Witte (1994) do not note this reading.

OUQ	OUD	Q/D	P46 9.18

This spelling variation is caused by assimilation to the following rough breathing (Gignac, 1975, 97).

PAIDEA	PAIDEIA	error	U142 12.11

This has been classified as an error rather than a spelling variation because it has been altered to paideia by a later hand.

PALAIWKEN	PEPALAIWKEN	error	U150 8.13

This may be better classified as an optional spelling as reduplication is occasionally omitted in simple verbs (Gignac, 1977, 242).

PANTONTOS	PANTOS	error	P13 2.15

Apparently, the scribe wrote panton by mistake then supplied the correct suffix without striking out the erroneous letters.

PARABASEIS	PARABASEIS	spurious (EI/I)	U2 2.2

The context requires a singular noun.

PARADECETE	PARADECETAI	E/AI	U2 12.6

This cannot take the -ete suffix as it is a deponent verb.

PARADEIGMATIZONTES	PARADEIGMATIZONTES	spurious (error)	U6-2 6.6

The context requires an accusative plural participle. There is no antecedent for a nominative plural. [RKM]

PARADIGMATIZONTES	PARADEIGMATIZONTES	I/EI, spurious (error)	U6 6.6

The same applies here as in the preceding entry.

PARAFERESQAI	PARAFERESQAI	spurious (AI/E)	U2 13.9

The imperative is appropriate here, not the infinitive.

PARAITHSESQE	PARAITHSESQE	spurious (E/H)	U44 12.25

This is unlikely to be a genuine variation (i.e. the future indicative passive of paraitevomai) because mhv and not ouj is used as the associated negative particle.

PARAKALEITAI PARAKALEITAI spurious (AI/E) U151 3.13

The indicative passive is not appropriate in conjunction with the reflexive pronoun.

PARAPESONTOS PARAPESONTOS spurious (error) U6 6.6

The context requires the accusative participle. The scribe appears to have written parapevsonto" under the influence of the preceding mevllonto" aijw'no". It is unlikely that parapevsonto" can be taken to qualify aijw'no", given the unnatural word order and interposed kaiv. [RKM]

PARARRUWMEN PARARUWMEN optional M2815 2.1

BAGD (1979, 621) specifies the single *rho* spelling for the subjunctive. Robertson (1923, 212) writes, 'Pararuw'men (Heb. 2:1) is read by all the pre-Syrian classes.'

PAREMBOLAIS PAREMBOLAIS variant M2815 11.34

The clause makes sense if the dative is read instead of the accusative: *they put to flight by foreign armies*. [RKM]

PARETHSAMENOI PARAITHSAMENOI E/AI U6 12.25

As this word is an aorist participle, the *epsilon* is unlikely to be the result of augmentation or reduplication.

PARETHSHSQAI PARAITHSHSQE E/AI, AI/E U6 12.25

This scribal form contains two cases of E/AI spelling interchange, both tending away from the conventional spelling but in opposite directions (one has AI > E and the other has E > AI). The *epsilon* can only be a spelling variation because the augmented or reduplicated form, if it were possible, would be parh/-. The -sqai ending is never preceded by *eta* in

conventional forms.

PARETHSHSQE PARAITHSHSQE E/AI U6-1 12.25

The *epsilon* can only be a spelling variation. (See the preceding entry.)

PARHTHSAN PARHTHSANTO error U4 12.19

It seems that the active ending has mistakenly been appended to a deponent verb.

PARITHSHSQE PARAITHSHSQE error U25 12.25

Whereas this could be a case of I/AI interchange (cf. Gignac, 1975, 259), I have classified it as a spelling error because the scribe included the *alpha* in a similar word derived from the same root (paraithsavmenoi) a few words later.

PEDEUON PAIDEUON E/AI, spurious (error) U1-x 12.10

A corrector has deleted the initial epsilon of epedeuon. (See the corresponding transcription note.) The sentence collapses if the usual reading (ejpaivdeuon) is altered to a participle (paivdeuon). [RKM]

PEFANERETAI PEFANERWTAI E/W U142 9.26

The reduplicated prefix requires that this word be the perfect pefanevrwtai. Gignac gives examples of E/W interchange (1975, 292).

PEFANERWSAI PEFANERWSAI spurious (error) U6 9.8

The second person perfect does not make sense in this context: *By this the Holy Spirit indicates that you have not yet been revealed the way into the sanctuary*.

PEFANERWTE PEFANERWTAI E/AI U6 9.26

The reduplicated prefix rules out the possibility of the conventional form having the suffix -te.

PEIQESQAI	PEIQESQAI	spurious (AI/E)	U6 13.17

The context requires the imperative, not the infinitive.

PEISTI	PISTEI	EI/I, I/EI	U1 11.17

The scribe of Codex Sinaiticus regularly wrote piçti for pistei, but wrote peiçti on two occasions.

PELEWKEN	PEPALAIWKEN	error, E/A, E/AI	U6 8.13

See the comment relating to PALAIWKEN above. Gignac (1975, 279) gives examples of the E/A interchange.

PEPEIRAMENON	PEPEIRAMENON	variant	M2815 4.15

This word is a perfect participle of the middle voice peiravomai, and can be translated as *having been tempted (in his interest)*. The alternative reading (pepeirasmevnon) is a perfect participle from peiravzw. According to Dobson (1989, 228), the -zw ending signifies intensive or causative action. In consequence, pepeirasmevnon may be translated as *having been sorely tempted*.

PEPOIQA	PEPOIQA	variant	U150 13.18

Reading pevpoiqa for peiqovmeqa makes sense provided that the second person pronoun is supplied: *for I have persuaded [you] that we have a clear conscience*.

PEPOIQAMEN	PEPOIQAMEN	variant	M2815 13.18

Given that the second person pronoun is supplied, the clause may be

translated as follows if pepoivqamen is read in place of peiqovmeqa: *for we have persuaded [you] that we have a clear conscience*.

PEPOQEN	PEPONQEN	error	P46 2.18

The omission of medial nasals before stops is very frequent (Gignac, 1975, 116).

PEPWQAMEN	PEPOIQAMEN	W/OI	U4-2 13.18

Gignac (1975, 294) gives an example of this spelling change.

PERAN	PERAN	spurious (E/EI)	U48 11.36

Reading pevran instead of pei'ran results in nonsense: *others received mocking and scourging across*.

PERH	PERI	H/I	U4-2 13.11

Gignac (1975, 241) gives the analogous example of chrhv for ceiriv.

PERIKEITE	PERIKEITAI	E/AI	U150 5.2

No derivatives of the verb kei'mai have the form kei'te.

PERISSOTERONTERON	PERISSOTERON	error	U56 7.15

The scribe inadvertently wrote -teron twice. (See the corresponding transcription note.)

PETH	PESH	error	P13 4.11

Perhaps the scribe began to write pevthtai: *Let us strive to enter that rest, that no one <u>fly</u> by the same sort of disobedience*.

PIQESQAI	PEIQESQAI	I/EI, spurious (AI/E)	U16 13.17

See the entry for PEIQESQAI above.

PLAKAIS	PLAKES	AI/E	U20 9.4

The preceding article is nominative plural. The dative plural of this noun would be plaxivn.

PNEUMATWS	PNEUMASI	error	U18 12.23

See the corresponding transcription note.

PODIWN	PODWN	error	P13 10.13

Perhaps the scribe was thinking of the Latin *podium*?

POIHHSAI	POIHSAI	error	U142 13.21

See the corresponding transcription note.

POIHSH	POIHSH	variant	U150 13.6

This clause makes sense if the subjunctive is read: *I will not be afraid; what might a man do to me?* Wachtel and Witte (1994) do not list this variation.

POIHSIS	POIHSIS	spurious (I/EI)	U1 8.5

The clause does not make sense if the verb is replaced with a noun: *See performance everything according to the pattern*. The scribe probably meant the word to be the second person singular future (conventional form poihvsei"), but could have been thinking of the subjunctive (poihvsh/").

POLLA	POLLA	variant	U6 9.26

The accusative can function as an adverb (BAGD, 1979, 688).

PROELHLUQATAI PROELHLUQATE AI/E, variant U48 12.18

This scribal form is derived from proevrcomai (*go forward*) and not prosevrcomai (*approach*). It must be a spelling variation of proelhluvqate because conventionally formed perfects do not end in -atai. Wachtel and Witte (1994) do not list this variant.

PROHNEGKEN PROHNEGKEN variant U142 9.14

This is derived from profevrw (*bring before*), not prosfevrw (*offer*). Wachtel and Witte (1994) have not listed this variant.

PROQESEIS PROQESEIS spurious (EI/I) U2 9.2

The context requires a singular noun.

PROSAGOUSHS PROSAGOUSHS variant U6 7.18

This variant makes sense in its local context, but does not reflect the wider argument: *On the other hand, an approaching commandment is set aside because of its weakness and uselessness*.

PROSBLEYAMENOI PROSBLEYAMENOI spurious (error) P46 11.40

The genitive absolute construction requires that this participle be in the genitive case. If prosbleyamevnou were substituted for the normal probleyamevnou, this clause would read: *since God had looked towards something better for us*.

PROSDEXASQE PROSDEXASQE variant P46 10.34

The imperative makes sense here: *Accept with joy, knowing yourselves to have a better possession and an abiding one*.

PROSEDEXASQAI PROSEDEXASQE AI/E U151 10.34

The augment rules out the possibility of this being an infinitive.

PROSELHLUQATAI PROSELHLUQATE AI/E U25 12.18

Conventionally formed perfects do not take the -atai ending.

PROSENEGKEI PROSENEGKH optional U151 8.3

The strong aorist stem and lack of an augment make it likely that this is an aorist subjunctive, especially since 'Forms of the indicative are frequently substituted for those of the subjunctive' (Gignac, 1977, 358). Guillemette (1986, 141) lists ejnegkei' as a variant form of the infinitive, so it is possible that this scribal form is an infinitive. Strangely enough, the RSV uses an infinitive here: *hence it is necessary for this priest also to have something to offer*.

PROSENEIKAS PROSENEGKAS error P13 10.12

See the corresponding transcription note.

PROSEUCESQAI PROSEUCESQAI spurious (AI/E) U16 13.18

The imperative is required by the context, not the infinitive.

PROSHNEKKEN PROSHNEGKEN K/G P13 11.4

The *kappa* in question is uncertain. (See the corresponding transcription note.) Gignac (1975, 171) gives examples of assimilation of -gk- to -kk-.

PROSWKTEISA PROSWCQISA K/C, T/Q P13 3.10

The *tau* is uncertain. (See the corresponding transcription note.) Examples of T/Q interchange are given in Gignac (1975, 87).

PROSWPOU PROSWPOU spurious (error) P46 9.24

The context requires a dative, not the genitive.

PROTEROI	PROTEROI	variant	U18 4.6

The clause remains grammatically intact if provteron is replaced with provteroi: *and the first ones to receive the good news failed to enter because of disobedience*.

PROTERWN	PROTERWN	spurious (W/O)	U151 10.32

The genitive is not appropriate to the context. The neuter, provteron, (which acts as an adjective) is required.

PRWTOTOKEIAS	PRWTOTOKOUS	optional, variant	P46 12.16

The scribe of P46 is alone in writing taç prwtotokeiaç instead of the usual ta; prwtotovkia eJautou'. The correct accusative plural feminine form is prwtovtokou", but 'Many adjectives which elsewhere in Greek are adjectives of only two terminations -o", -on, … tend to have a distinct feminine' (Gignac, 1977, 105). The clause remains grammatically sound with this reading: *who for a single meal sold <u>the firstborn females</u>*.

QARROUNTES	QARROUNTES	spurious (error)	U4 13.6

The context requires the accusative case rather than the nominative.

QATTON	QASSON	TT/SS, variant	U150 7.23

qa'sson (Attic qa'tton) is the neuter comparative form of tacuv" (Liddell and Scott, 1968, 1762-3). This clause may then be translated, *through sooner being hindered from carrying on*.

QEMELIOTHTA	TELEIOTHTA	error	P46 6.1

The scribe appears to have conflated teleiovthta with the following

qemevlion to arrive at qemeliothta. The scribe might have been thinking of the noun qemeliwthv", ou', oJ (*founder*), but the preceding feminine article is against this.

QERISMOIS	QERISMOIS	variant	U1 2.4

The clause makes sense if qerismoi'" is read instead of merismoi'": *and by harvests of the Holy Spirit*. Wachtel and Witte (1994) do not list this variant.

QEWREITAI	QEWREITAI	spurious (AI/E)	U75 7.4

Reading qewrei'tai instead of qewrei'te results in nonsense: *It is considered how great this man was, to whom even Abraham gave a tenth of the spoils*.

QHSJASTHRIW	QUSIASTHRIW	J/I, error	M2815 7.13

There is no such noun as qhsiasthrion.

QIGEI	QIGH	EI/H	U18 11.28

The strong aorist stem qig- does not take the -ei suffix in conventional forms.

RHMATOS	RHMATOS	error	P46 6.5

P46 has the unique wording KAI KALON GEUSAMENOUS Q(EO)U RHMATOS' DUNAMIS TE MELLONTOS AIWNOS' here, with the reading marks having been inserted by a later hand. This is grammatically unsound because the adjective kalovn is left without a noun once the accusative rJh'ma is replaced with the genitive rJhvmato". The word cannot be interpreted as a noun substitute because it would then be in competition with qeou' rJhvmato" as the object of geusamevnou". The error is probably due to the scribe unconsciously changing the case of rJh'ma under the influence of the preceding qeou'.

SALEIM	SALHM	EI/H	M2815 7.1

This could be a spelling variant of Salivm.

SALHN	SALHM	N/M	U6 7.1

See the corresponding transcription note.

SAMMATISMOS	SABBATISMOS	M/B, M/B	P13 4.9

The letters in question are uncertain in P13. (See the corresponding transcription note.) Gignac (1975, 71) says that M/B interchange is rare.

SAMYW	SAMYWN	optional	P13 11.32

A final nasal is sometimes omitted in transliterated words (BDF, 1961, §39(8)).

SEIWN	SEIWN	spurious (EI/I)	P46 12.22

The noun Siwn (*Zion*) is appropriate here, not the participle seivwn (*shaking*).

SKLHRUNETE	SKLHRUNETE	variant	U6 3.15

'The present tense imperative negated indicates that an action in progress is to cease or else that one is not from time to time to do some action' (Hewett, 1986, 192-3). The corresponding clause is accordingly translated, *do not any longer harden your hearts as in the rebellion*.

SKLHRUNHTAI	SKLHRUNHTAI	spurious (AI/E)	U1 4.7

Reading sklhruvnhtai instead of sklhruvnhte results in a grammatically flawed sentence: *Let it not be hardened your hearts*.

SKOTEI	SKOTEI	variant	P46 12.18

This noun exhibits metaplasm: skovto", -ou", tov, and skovto", -ou, oJ both occur. (See BDF, 1961, §51(2).)

SPONDON	SPODON	N/-, error	U6 9.13

The insertion of a medial nasal before a dental stop is a known spelling variation (Gignac, 1975, 119, note 5). It is possible that the scribe was thinking of spondhvn (*drink offering*), but the *omicron* in the ending of the scribal form is against this. The nominative spodov" rather than the accusative spodovn is required here.

SPONDOS	SPODOS	N/-	U6-1 9.13

The preceding entry addresses this spelling variation.

STOMA	STOMA	variant	U6 11.33

This is grammatically acceptable if understood as a distributive singular (BDF, 1961, §140): *they stopped the mouth[s] of lions*.

SUGKEKRAMENOS	SUGKEKERASMENOS	optional, variant	TR 4.2
SUNKEKERASMENOS	SUGKEKERASMENOS	N/G, variant	U1 4.2

The perfect passive participle may be spelled sugkekerasmevno" or sugkekramevno" (BAGD, 1979, 773). The nominative singular gives: *but the message of the report did not benefit them, it not being united to the hearers by faith*.

SUGKEKRAMENOUS	SUGKEKERASMENOUS	optional	M2815 4.2
SUNKEKERASMENOUS	SUGKEKERASMENOUS	N/G	P13 4.2
SUNKEKRAMENOUS	SUGKEKERASMENOUS	N/G,	

optional U150 4.2

The spelling is optional, as noted in the preceding entry. The accusative plural participle gives: *but the message of the report did not benefit them, they not being united to the hearers by faith*.

SUMFERWN SUMFERWN spurious (W/O) U2 12.10

The context requires the neuter participle.

SUMFURON SUMFERON U/E U142 12.10

Gignac gives examples of U/E spelling interchange but notes that it mainly occurs in unaccented syllables (1975, 273-4).

SUN OUN error U75 4.11

See the corresponding transcription note.

SUNEPAQHSATAI SUNEPAQHSATE AI/E U150 10.34

Only -aw verbs can have the -atai suffix.

SUNEPIMARTUROUNTES SUNEPIMARTUROUNTES
variant P46 2.4

The nominative plural participle makes sense in this context: *and it was attested to us by those who heard him, [they] bearing God's witness by signs and wonders…*

SUNTHSEN SUNHNTHSEN error U4 7.10

See the corresponding transcription note.

SWTERON ESWTERON error U4 6.19

Perhaps the scribe was thinking of swthrivan?

SWZEI	SWZEI	variant	P46 7.25

This variation makes sense provided that there is a pause after to; pantele;": *Consequently he saves completely*. A reading mark has been placed at precisely this place in P46.

TAI	TAIS	-/S	U150 12.3

Instead of classifying this as an error, I have classified it as a spelling variation due to the omission of *sigma*. The difficulty of pronouncing *psi* after a sibilant provides a motive for the omission. Gignac (1975, 124) says that final *sigma* is frequently omitted, regardless of the following sound.

TAUTAIS	TAUTAIS	variant	P46 9.23

Reading the dative with parav changes the meaning somewhat: *it was necessary for the heavenly things themselves to be purified with better sacrifices <u>beside these</u>*.

TAXI	TAXIN	-/N	U151 6.20

Final *nu* is sometimes omitted before a liquid (Gignac, 1975, 112).

TEICEI	TEICEI	spurious (EI/H)	U150 11.30

The context requires the nominative plural.

TELEIOTHN	TELEIWTHN	O/W	U151 12.2

The accusative of teleiovth" is teleiovthta.

TELEIWSEIS	TELEIWSEIS	spurious (EI/I)	U2 7.11

teleiw'sei" is the second person future active of teleiovw. The context requires the noun teleivwsi".

TESSARAKONTA	TESSERAKONTA	optional	M2815 3.10

The earliest witnesses and UBS4 have the Ionic-Hellenistic tesseravkonta, but tessaravkonta is almost certainly the original spelling (BAGD, 1979, 813; Moulton and Howard, 1928, 67). Perhaps the spelling of UBS4 at this point should be changed?

TETEUCE	TETUCEN	movable *nu*, optional	M2815 8.6
TETEUCEN	TETUCEN	optional	TR 8.6
TETUCE	TETUCEN	movable *nu*	U1 8.6
TETUCHKE	TETUCEN	movable *nu*, optional	U44 8.6
TETUCIKE	TETUCEN	I/H, optional	U25 8.6

The perfect of tugcavnw can have two forms: the Attic tetuvchka or the Ionic tevteuca, which is also spelt tevtuca (BAGD, 1979, 829). Moulton and Howard (1928, 262) indicate that the tevtuca spelling is not older than the Hellenistic age. UBS4 prints tevtucen whereas BAGD and BDF (§101, tugcavnein) seem to prefer tevteucen. A review of the spelling of this word in UBS4 may be warranted.

THSEWS	KTISEWS	error	U20 9.11

See the corresponding transcription note.

TOS	TON	error	P46 11.7

See the corresponding transcription note.

TOSOUTWN	TOSOUTWN	variant	P46 1.4

This makes sense provided that a noun such as *kind* is understood to stand later in the clause: *having become of a so much superior [kind] relative to the angels as the name he has obtained is more excellent than theirs*.

TRAGW	TRAGW	spurious (error)	U3-3 9.12

This error is due to a late scribe not retracing a final *nu* superscript. (See the corresponding transcription note.) The context requires a genitive plural.

TRIMHNOS	TRIMHNOS	error	P46 11.23

This should be accusative. The error has not been corrected.

UPEMEINATAI	UPEMEINATE	AI/E	U25 10.32

The -atai suffix can only occur in -aw verbs.

UPOMEINATE	UPEMEINATE	error	U6 12.7

This may be a morphological variant rather than a spelling error. Gignac (1977, 223) writes, 'The syllabic augment is occasionaly omitted in compound verbs whose prefix ends in a vowel.'

UPOMENETAI	UPOMENETAI	variant	P13 12.7

The third person singular passive makes sense here: *It is endured for discipline*. Wachtel and Witte (1994) do not list this variation in their apparatus, implying that they regard it as a spelling variation. (This may well be the case.)

UPOSTASSEWS	UPOSTASEWS	SS/S	U4 3.14

See the corresponding transcription note.

UPOSTEILHTE	UPOSTEILHTAI	E/AI	U20 10.38

The aorist middle does not take the -hte suffix in conventional forms.

UPOSTILHTE	UPOSTEILHTAI	I/EI, E/AI	U6-0 10.38

The preceding comment applies here as well.

USTRERON	USTRERON	variant	U142 10.17

See the corresponding transcription note.

V	UMWN	error	P46 13.25

This is more of a deliberate contraction than an error. The scribe of P46 seems to have been in a hurry to finish, as is evident from the number of contractions and omissions in the last page of Hebrews.

VPEIKESQAI	UPEIKESQAI	V/U, spurious (AI/E)	U16 13.17

The passive imperative ujpeivkesqe fits the context (*obey those leading you and be submitted*), but the passive infinitive ujpeivkesqai does not.

VPEMEINAI	UPEMEINAI	V/U, error	U6-1 12.2

The second hand of U6 altered uJpemine to uJpemeinai. (See the corresponding transcription note.) This scribal form has been classified as an error because the -einai ending only occurs in the infinitive of mi verbs which have a root ending in *epsilon*, such as tivqhmi.

VPETAXA	UPETAXAS	V/U, error	U1 2.8

I have classified this as an error because a *sigma* has been added, perhaps by the first hand. (See the corresponding transcription note.) This may be a genuine case of the omission of final *sigma*. Such an omission can take place before a word beginning with a vowel (Gignac, 1975, 125).

VPOMENETAI	UPOMENETAI	V/U, variant	U2 12.7

See the note for UPOMENETAI above.

VPOMONHN	UPOMONHN	V/U, variant	P46 10.36

It is grammatically permissible to have the accusative uJpomonh'n, although the result violates the author's intention: *For you have endurance deficit, so that you may do the will of God and receive what is promised*. The accusative ending was subsequently changed to the genitive. Wachtel and Witte (1994) have not included this alteration in their transcription. (See the corresponding transcription note.)

VPOSGASEWS	UPOSTASEWS	V/U, error	U3-3 3.14

See the corresponding transcription note.

WFEILEI	OFEILEI	W/O	U20 5.3

This may be a spelling variant of wjfelei', in which case the sentence would be translated, *and because of this he helps to offer for his own sins as well as for those of the people*.

WMOIWQHNAI	OMOIWQHNAI	W/O	U2 2.17

This may actually be an augmented form. The augment is sometimes transferred to tenses which do not normally take it (Gignac, 1977, 234).

WMOSAI	OMOSAI	W/O	U20 6.13

The same applies here as in the preceding entry.

WMWSEN	WMOSEN	W/O	U243 3.18

The aorist subjunctive would be w[mosh/.

WRKWMOSIAS	ORKWMOSIAS	error	U2 7.21

I have classified this as an erroneous spelling because this scribe normally writes oJrkw-. (Compare 7.20 and 7.21 of the corresponding transcription.) The sound change may have been intentional, possibly being conditioned by the preceding meq.

XENESANTES	XENISANTES	E/I	U48 13.2

Analogous spelling changes are recorded by Gignac (1975, 253-4) under the heading 'i > e before a sibilant'.

YEUSASQE	YEUSASQE	spurious (E/AI)	U2 6.18

The context requires the infinitive yeuvsasqai.

ZEI	ZEI	variant	U25 9.17

This is more likely to be a spelling variation of zh/' than a genuine textual variant. I have classified it as a textual variation because the result is grammatically sound: *For a will takes effect only at death, since it is not in force as long as the one who made it boils*. No doubt, Wachtel and Witte (1994) are right to omit this variation.

ZEIN	ZHN	EI/H	U25 2.15

The present infinitive of zavw is normally written zh'n.

ZW	ZW	variant	U4 4.12

This sentence still makes sense if zw'n is replaced by zw': *For I, the word of God, live and [am] sharper than any two-edged sword*.

ZWON	ZWON	spurious (O/W)	U6 13.11

The context requires that this be a genitive plural.

Note on punctuation equivalents

According to Robertson's summary of the punctuation practices of New Testament manuscripts (1923, 241-5), the paragraph was variously marked by the following devices:
(1) a blank space within the text;
(2) an enlarged or projecting letters at the paragraph division or in the

following line;
(3) a *paragraphos* (_), *diplé* (>), or *coronis* ( ) in the left margin; or
(4) by a combination of these devices.

Paragraph division is represented by the paragraph symbol (¶ = ASCII 166) in my transcriptions.

The system of punctuation attributed to Aristophanes of Byzantium (260 BCE) consisted of three points. The high point, or stigmh; teleiva (˘), was equivalent to a full stop. The low point, or uJpostigmhv (.), was equivalent to a semicolon. The medial point or stigmh; mevsh (≥) was equivalent to a comma. (Robertson (1923, 242, note 5), says that there is disagreement on the identification of the medial point with the comma.) This situation gradually changed until the high point became equivalent to a semicolon, and the low point became equivalent to a full stop. The medial point vanished, to be replaced by a comma from the ninth century. The interrogation mark, or ejrwthmatikovn (;), appeared around the same time.

I used distinct symbols to transcribe the medial stop (≥ = ASCII 179), medial comma (≤ = ASCII 178), and high point (˘ = ASCII 249). The apostrophe (') was employed for marks signifying elision, word division following non-Greek names, and reading marks. The ellipsis (… = ASCII 201) was used to represent spaces that might indicate sense-pauses. Any other marks that had a ready equivalent on a standard ASCII keyboard (e.g., low point and low comma) were transcribed by the corresponding keystrokes. The *coronis* that signifies crasis was not treated as punctuation, but was transcribed as a smooth breathing by means of tags.

It is desirable to normalise the punctuation of the transcriptions so that any correlation between the punctuation in different manuscripts can be measured. The potential for error caused by a global replacement strategy is particularly acute in this context, mainly because the function of marks such as high points and low points can vary between manuscripts. In modern editions, the high point corresponds to a short pause and the low point indicates a full stop. The opposite is true in a

manuscript such as Codex Vaticanus (Robertson, 1923, 242).

Due to this ambiguity, it is not possible to specify a set of universally valid equivalents if a distinction is made between the values of different punctuation marks. For this reason, I have given a single punctuation mark (the full stop) as the equivalent of every punctuation mark found in the witnesses, including UBS4.

The following notes give further details concerning some of the punctuation marks listed in the equivalents table:

+	.	punctuation	U122-1 5.10

These symbols have been added by a corrector and often occur in conjunction with other punctuation marks. They are probably better classified as lectionary markers.

:	.	punctuation	M2815 12.23

This mark occurs frequently in P13. When compared with the punctuation of UBS4, this mark coincides with a comma 34 times, and with a full stop exactly the same number of times.

…	.	punctuation	M2815 10.15

The ellipsis ,which marks unusually large spaces between transcribed words, does not necessarily coincide with a sense-pause.

|	.	newline	UBS4

Some manuscripts use colometrical arrangement to indicate pauses in sense. Treating the line division symbol as a punctuation mark allows such manuscripts to be included in an investigation of sense-pause correlation.

References

The Analytical Greek lexicon. n.d. London: Samuel Bagster and Sons.

Bauer, Walter. 1979. *A Greek-English lexicon of the New Testament and other early Christian literature*. Trans. William F. Arndt and F. Wilbur Gingrich. 2nd rev. and augmented ed. Ed. F. Wilbur Gingrich and Frederick W. Danker. Chicago: University of Chicago Press.

Black, David Alan. 1988. *Linguistics for students of New Testament Greek: a survey of basic concepts and applications.* Grand Rapids: Baker Book House.

Blass, Friedrich and Albert Debrunner. 1961. *A Greek grammar of the New Testament and other early Christian literature: a translation and revision of the ninth-tenth edition incorporating supplementary notes of A. Debrunner†*. Trans. and ed. Robert W. Funk. Chicago: University of Chicago Press.

Dobson, John H. 1989. *Learn New Testament Greek.* Grand Rapids: Baker Book House.

The Englishman's Greek New Testament giving the Greek text of Stephens 1550, with the various readings of the editions of Elzevir 1624, Griesbach, Lachmann, Tischendorf, Tregelles, Alford, and Wordsworth: together with an interlinear literal translation and the Authorised Version of 1611. 1896. Third ed. London: Samuel Bagster and Sons.

Friberg, Barbara and Timothy (eds.). 1981. *Analytical Greek New Testament*. Grand Rapids: Baker Book House.

Gignac, Francis Thomas. [1975]. *A grammar of the Greek papyri of the Roman and Byzantine periods*. Vol. 1. *Phonology*. Testi e documenti per lo studio dell'antichità, 55-1. Milan: Istituto Editoriale Cisalpino - La Goliardica. (The date is that given in the preface.)

Gignac, Francis Thomas. [1977]. *A grammar of the Greek papyri of the Roman and Byzantine periods*. Vol. 2. *Morphology*. Testi e documenti per lo studio dell'antichità, 55-2. Milan: Istituto Editoriale Cisalpino - La Goliardica. (The date is that given in the preface.)

Guillemette, Pierre. 1986. *The Greek New Testament analysed*. Kitchener: Herald Press.

Hewett, James Allen. 1986. *New Testament Greek: a beginning and intermediate grammar*. Peabody: Hendrickson.

Kubo, Sakae. 1975. *A reader's Greek-English lexicon of the New Testament and a beginner's guide for the translation of New Testament Greek*. Andrews University monographs, 4. Grand Rapids: Zondervan.

Lampe, G. W. H. (ed.). 1961. *A patristic Greek lexicon*. Oxford: Clarendon Press.

Liddell, Henry George and Robert Scott. 1968. *A Greek-English lexicon*. Rev. and augmented ed. with a supplement. Oxford: Clarendon Press.

Moore, Richard K. 1992. *New Testament Greek*. 5th corrected ed. Perth: Murdoch University.

Moulton, James Hope and Wilbert Francis Howard. 1928. *A grammar of New Testament Greek*. Vol. 2. *Accidence and word-formation with an appendix on Semitisms in the New Testament.* Edinburgh: T. & T. Clark.

Robertson, A. T. 1923. *A grammar of the Greek New Testament in the light of historical research*. 4th ed. Nashville: Broadman Press.

Revised Standard Version. 1946. New York: Thomas Nelson & Sons.

Royse, James Ronald. 1981. 'Scribal habits in early Greek New Testament papyri'. Dissertation (ThD). Graduate Theological Union.

Wachtel, Klaus and Klaus Witte. 1994. *Das Neue Testament auf Papyrus*. Vol. 2. *Die Paulinischen Briefe*. Part 2. *Gal, Eph, Phil, Kol, 1 u. 2 Thess, 1 u. 2 Tim, Tit, Phlm, Hebr*. Arbeiten zur neutestamentlichen Textforschung, 22. Berlin: Walter de Gruyter.

Wenham, J. W. 1965. *The elements of New Testament Greek*. Cambridge: Cambridge University Press.

Westcott, Brooke Foss and Fenton John Anthony Hort. 1881b. *The New Testament in the original Greek*. Vol. 2. *Introduction* [and] *Appendix*. Cambridge: Macmillan. Repr. 1974. Graz: Akademische Druck.

Zuntz, Günther. 1953. *The text of the epistles: a disquisition upon the corpus Paulinum*. Schweich Lectures of 1946. London: British Academy.

MAP NOTES

Identical maps

Identical maps occur whenever two or more principal witnesses have the same data matrix. Witnesses with identical maps are listed here, along with the dimensions of their corresponding data matrices:

Identical textual maps

728 units x 12 witnesses:
P46, U2, U6, U18, U25, U56, U142, U150, U151, M2815, TR, UBS4

71 units x 31 witnesses:
M81, M104, M256, M365, M1739, M1881, M1962, M2127, Attridge, Bover, Kilpatrick, Merk, NA25, Souter, Tasker, Vogels, von Soden, von Tischendorf, Westcott and Hort

64 units x 35 witnesses:
Byzantine, Georgian-1, Georgian-2, Chrysostom

58 units x 35 witnesses:
Itala-b, Itala-comp

Identical spelling maps

317 units x 12 witnesses:
P46, U2, U6, U18, U25, U56, U142, U150, U151, M2815, TR, UBS4

Unmapped witnesses

The probability that a classical scaling map gives a true representation of the overall configuration of a set of witnesses decreases as the number of units upon which it is based decreases. The relative positions of distant witnesses are virtually meaningless in a map that is derived from only five units. Nevertheless, such a map can be expected to identify the closest neighbours of a particular witness with a reasonable degree of certainty because the probability of accidental agreement among five

binary variables is still only 1/32 (~0.03). The probability of random agreement makes it unsafe to draw any conclusions from a resemblance coefficient calculated from less than five units. The following witnesses are not mapped because they are represented by less than five units:

Unmapped textual witnesses

P12, P13-2, P89, U1s-x, U2-0, U2-2, U16-0, U20-0, U56-1, U75-2, U75s-2, U75s-3, U75s-x, U122-0, U122-2, U142-2, U150-0, U151-0, U227, U228, U228-0, U243-x, M2815-0

Unmapped spelling witnesses

P12, P13-2, P13-x, P17, P89, U1s-x, U2-0, U2-2, U4-0, U4-x, U6-0, U16-0, U18-1, U20-0, U25-0, U44-2, U56-1, U75-2, U75-x, U75s-2, U75s-3, U75s-x, U122-0, U122-2, U122-3, U142-0, U142-2, U142-3, U150-0, U151-0, U227, U228, U228-0, U243-x, U252, M2815-0, EI-H

Textual and spelling maps have not been produced for P46-3 and P46-4 because their alterations concern paraphernalia such as reading marks and page numbering.

MAPS

Textual maps

962-1116

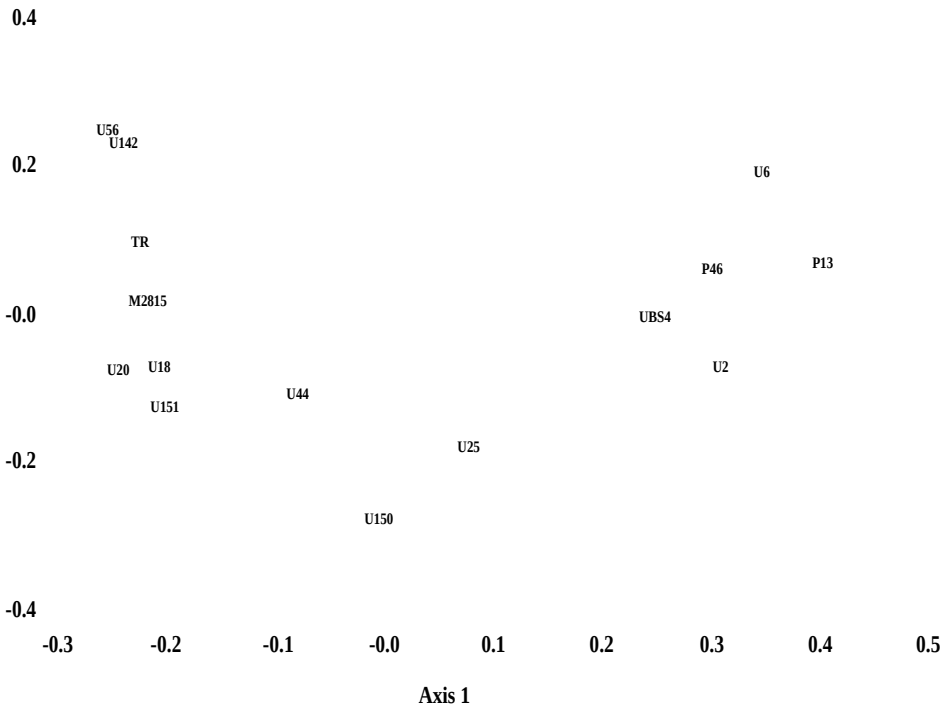
Spelling maps

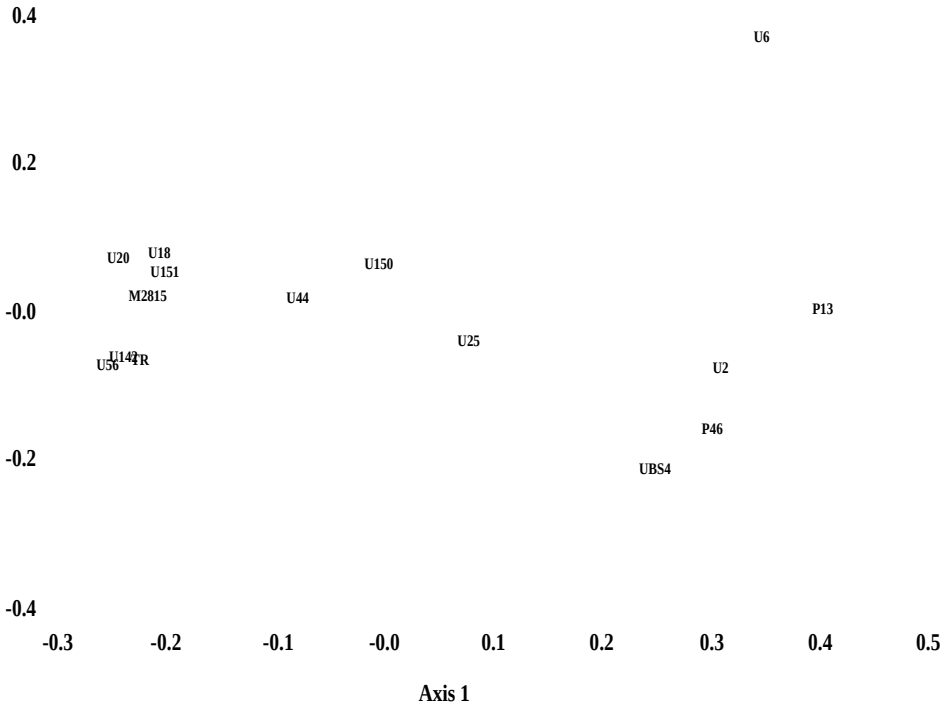
1117-1187

Other maps

1188-1194

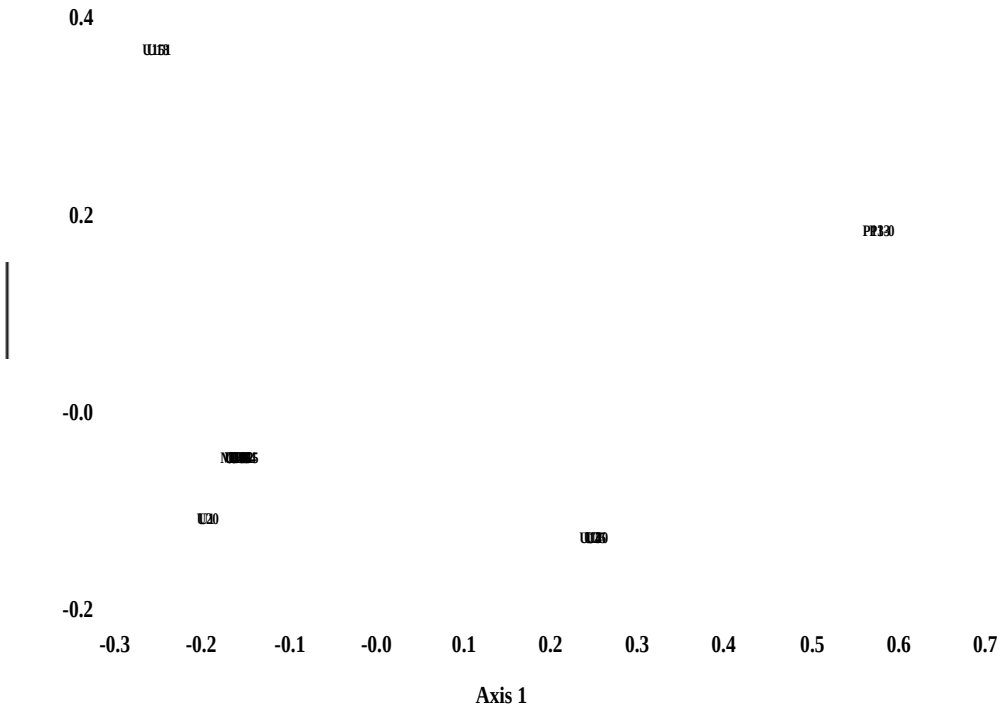
P13 (Oxyrhynchus, 300?)

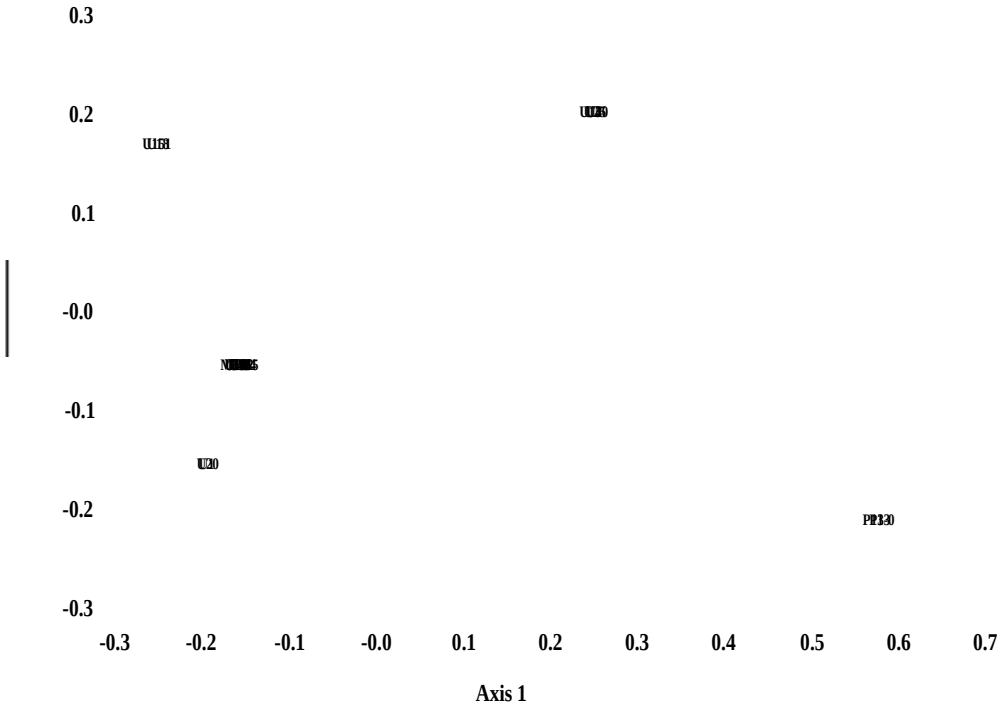




159 units; 35%, 13%, 10%

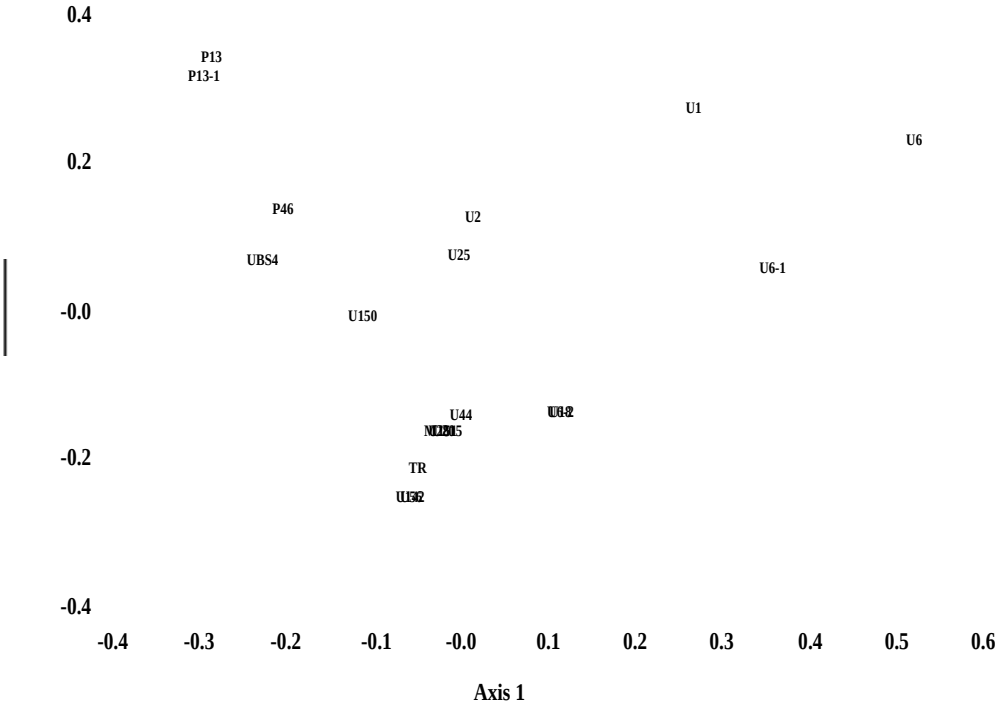
P13-0

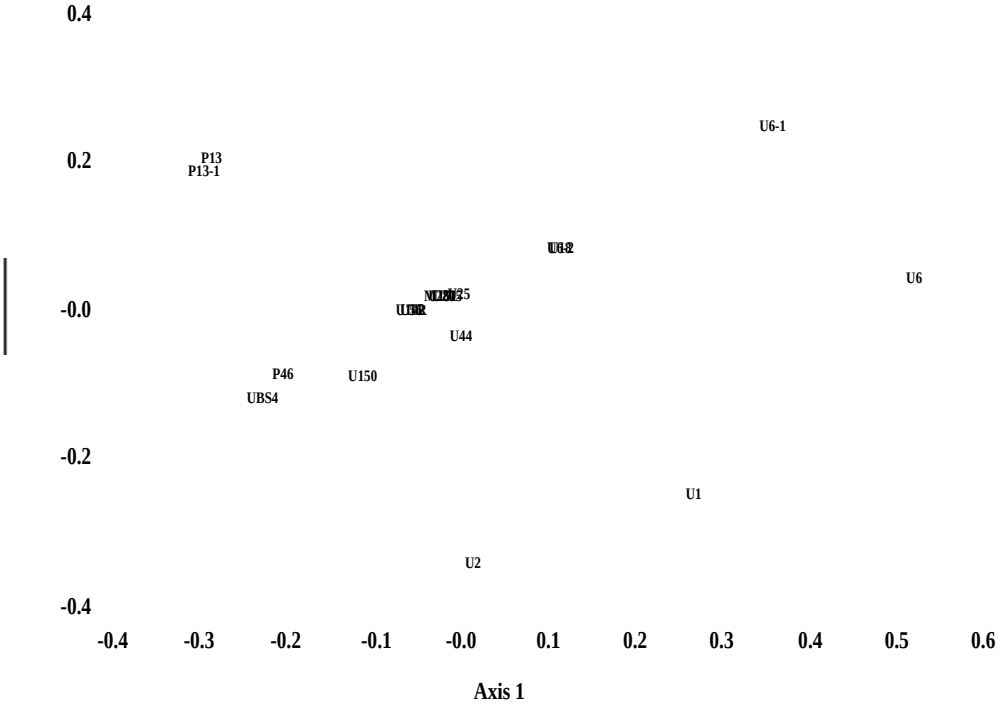




8 units; 57%, 19%, 16%

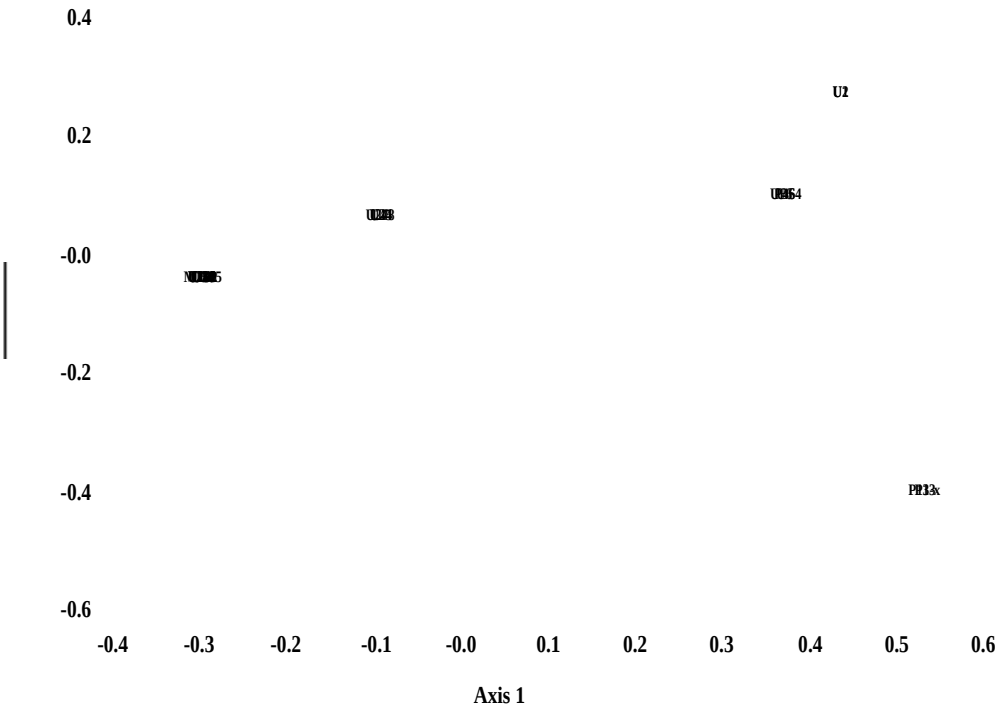
P13-1

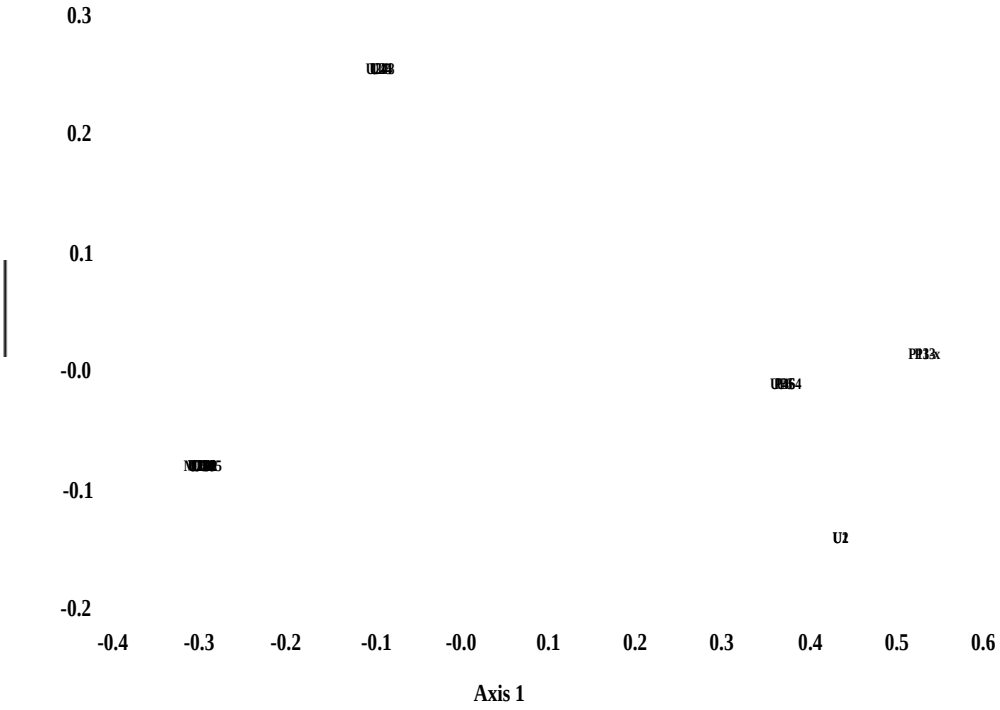




36 units; 29%, 26%, 14%

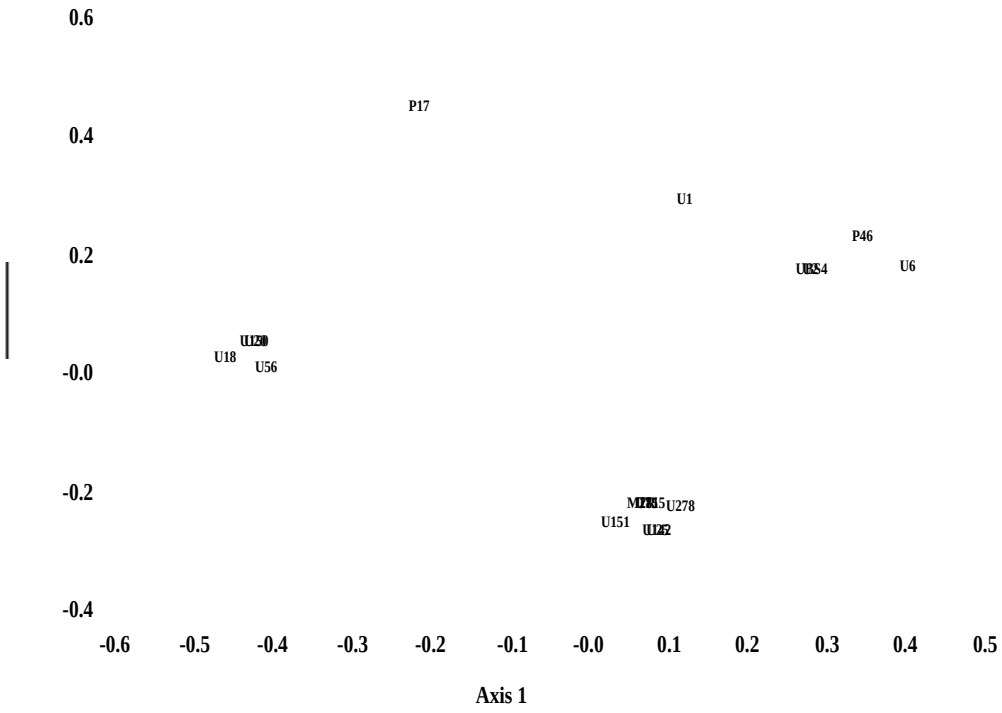
P13-x

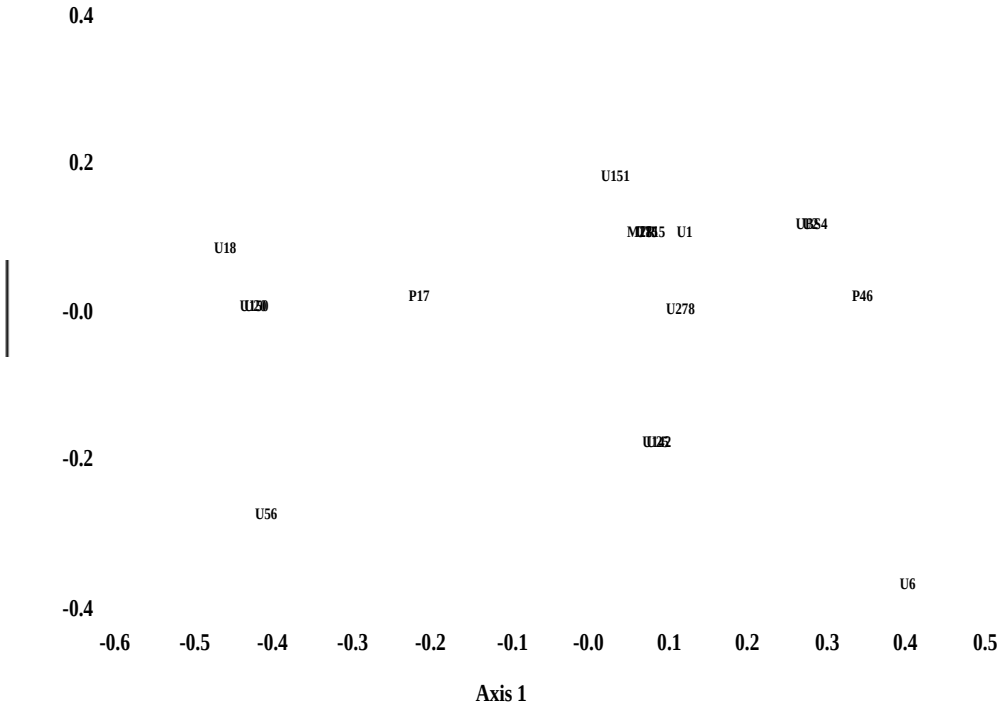




6 units; 68%, 16%, 11%

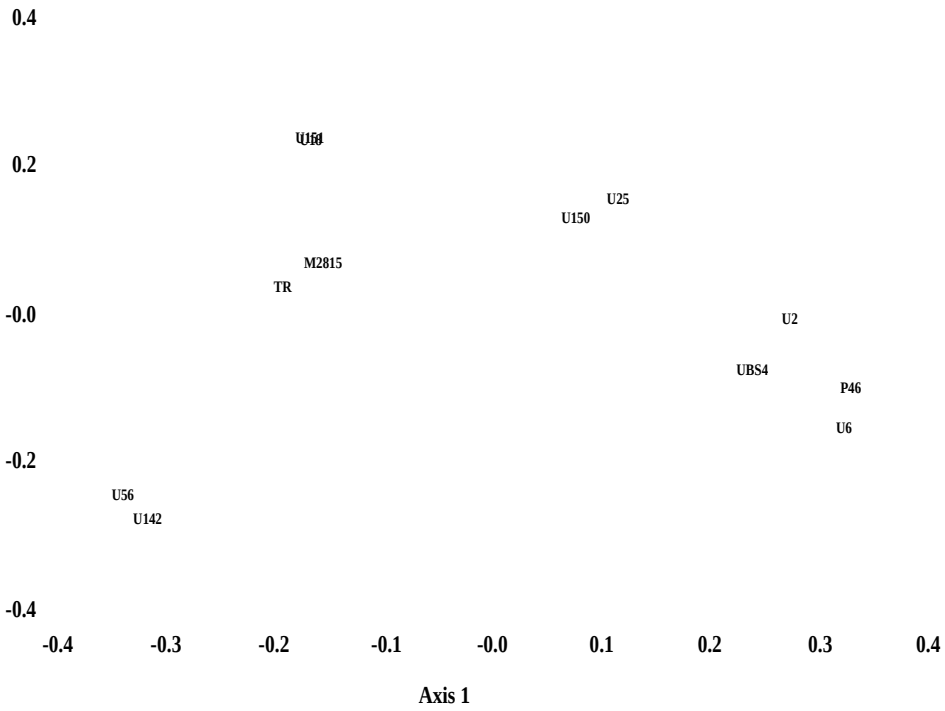
P17 (Oxyrhynchus, 350?)

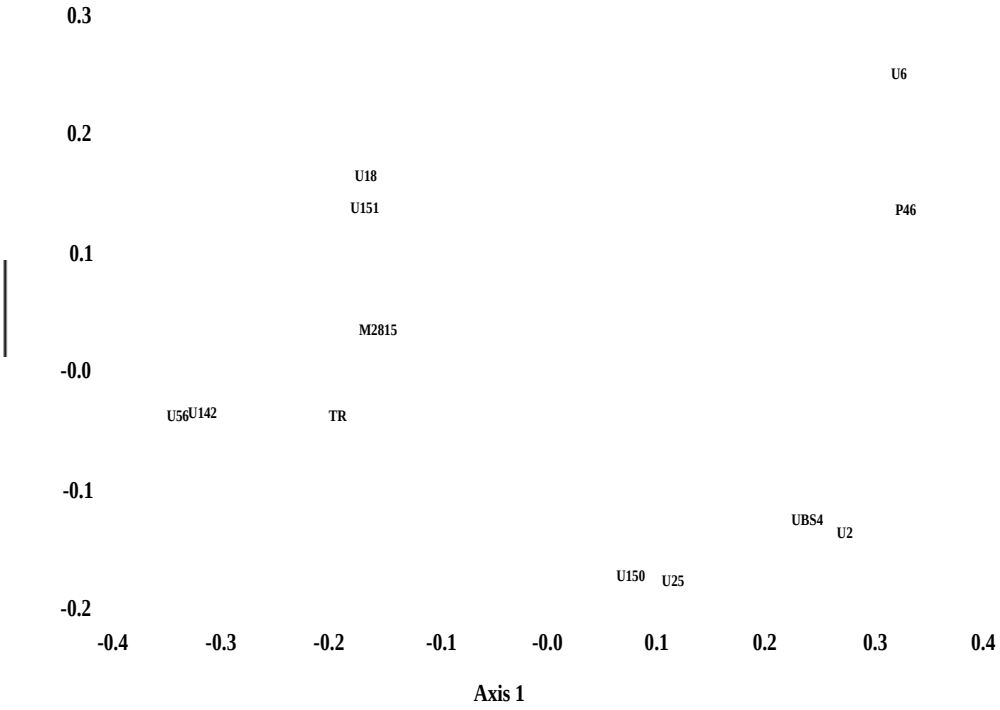




9 units; 43%, 28%, 13%

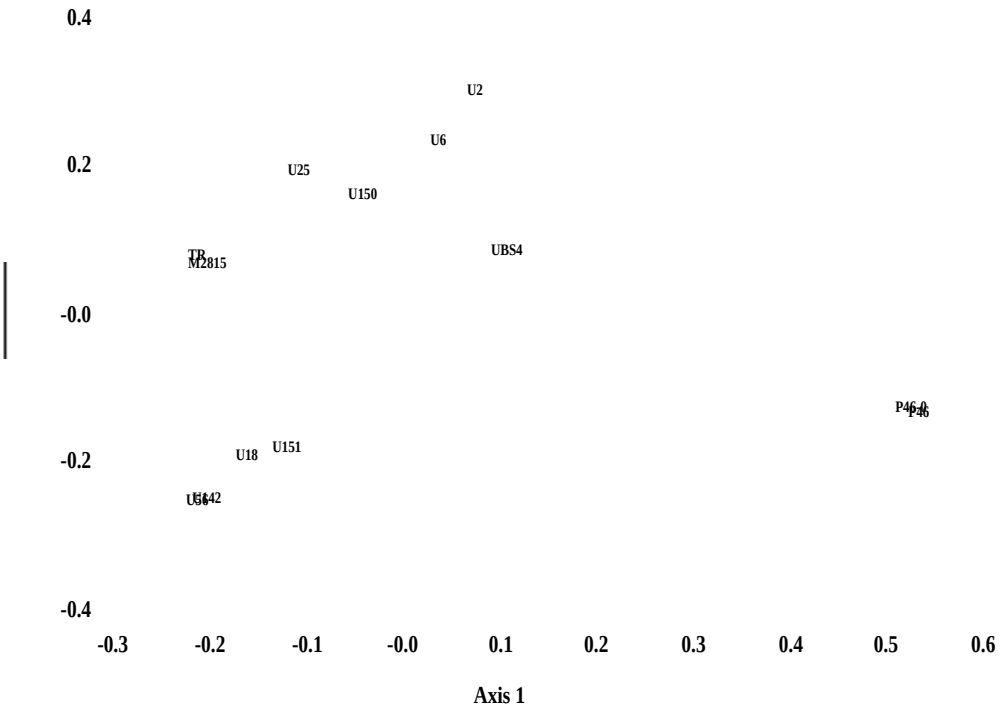
P46 (Fayyum, 200?)

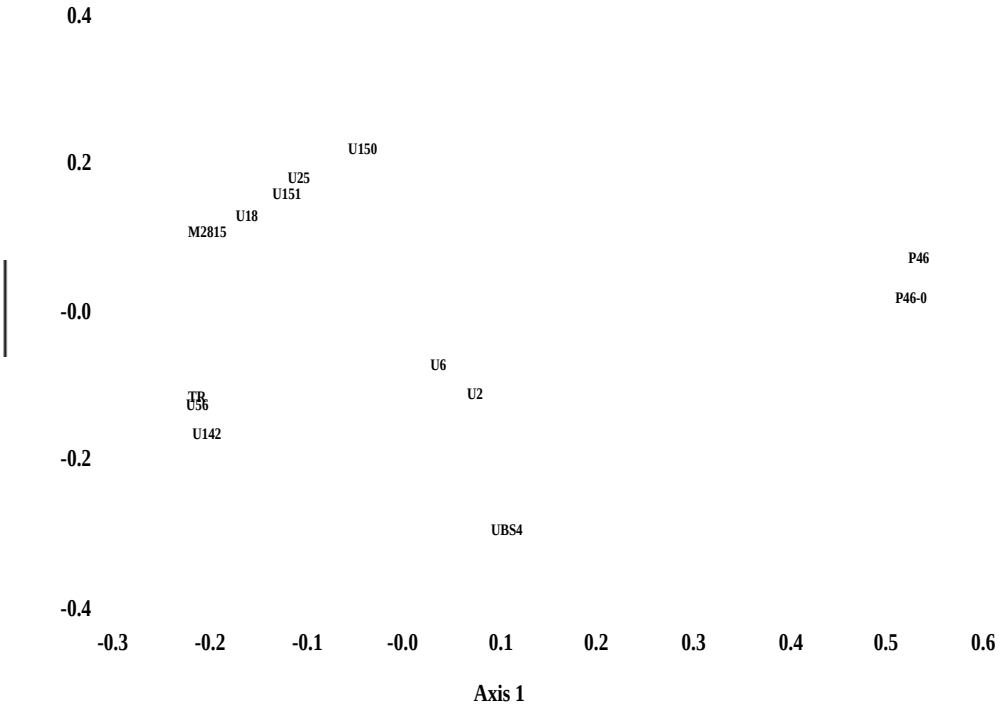




728 units; 32%, 15%, 10%

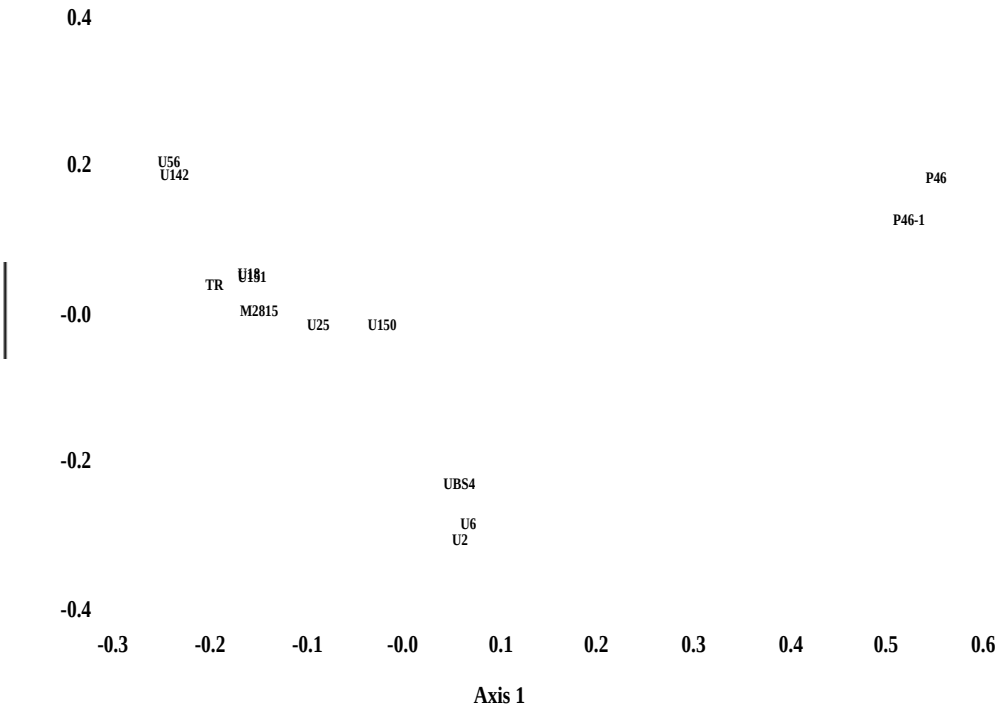
P46-0

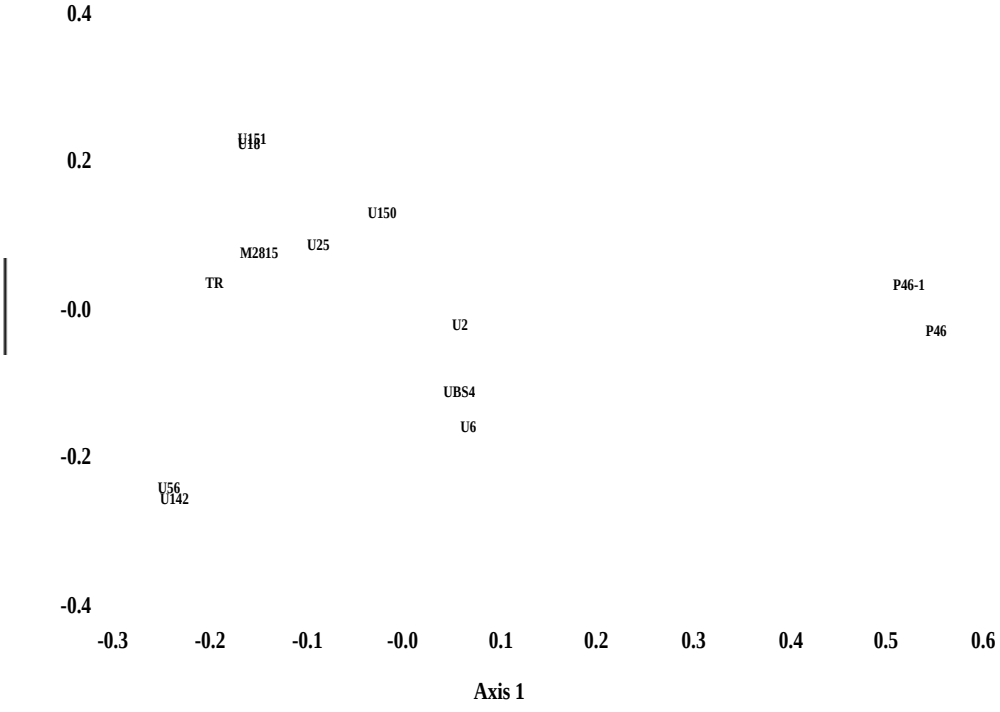




119 units; 34%, 19%, 13%

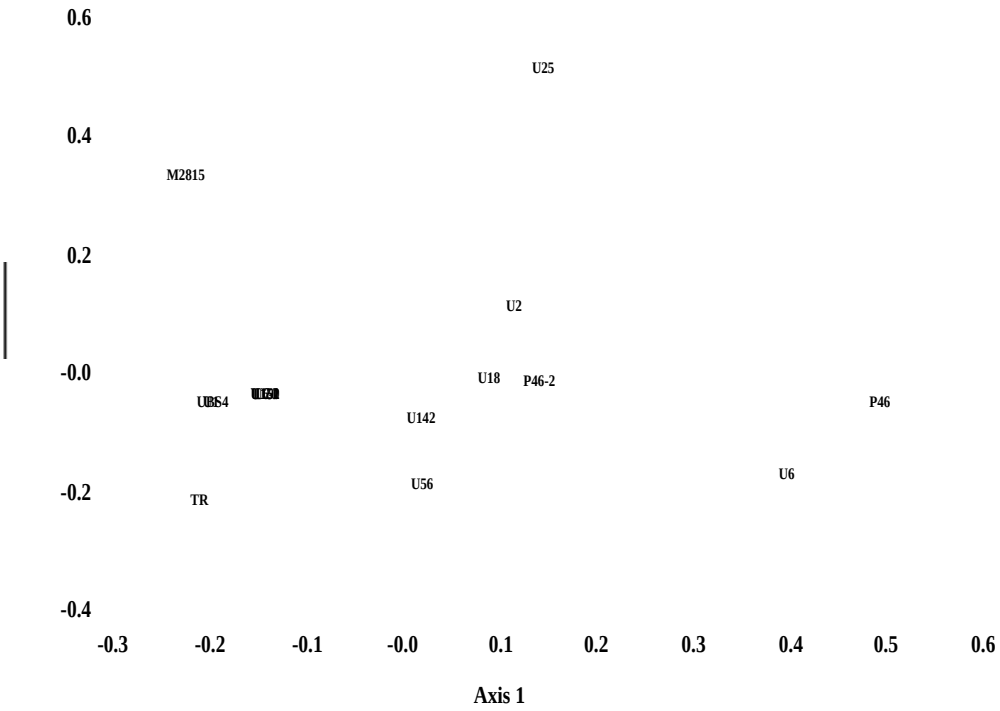
P46-1

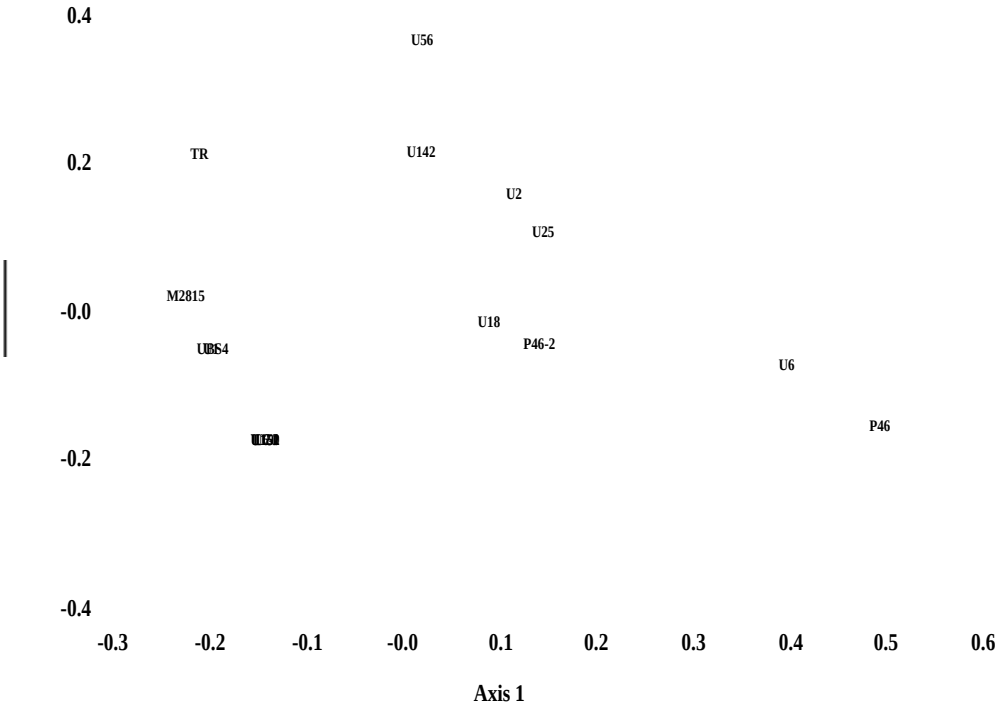




173 units; 37%, 16%, 13%

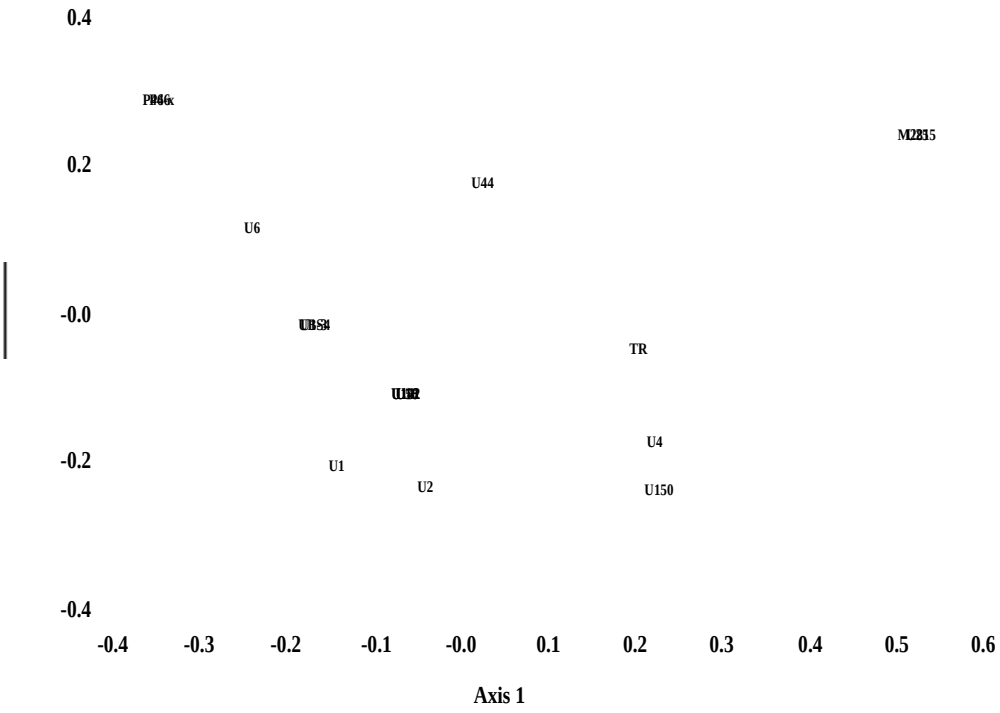
P46-2

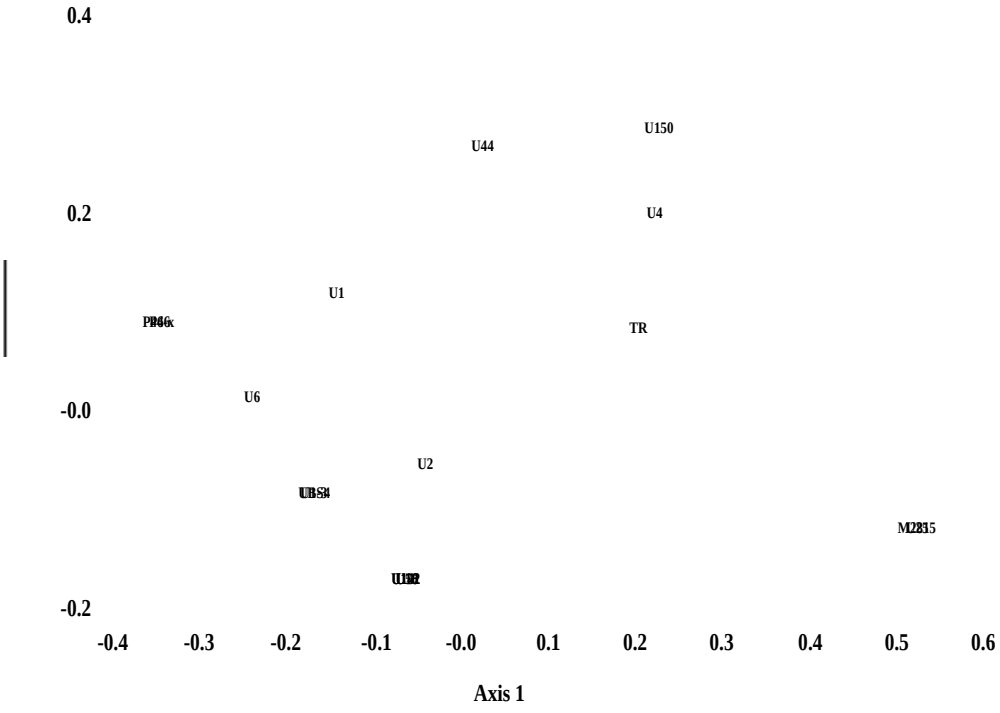




17 units; 32%, 23%, 19%

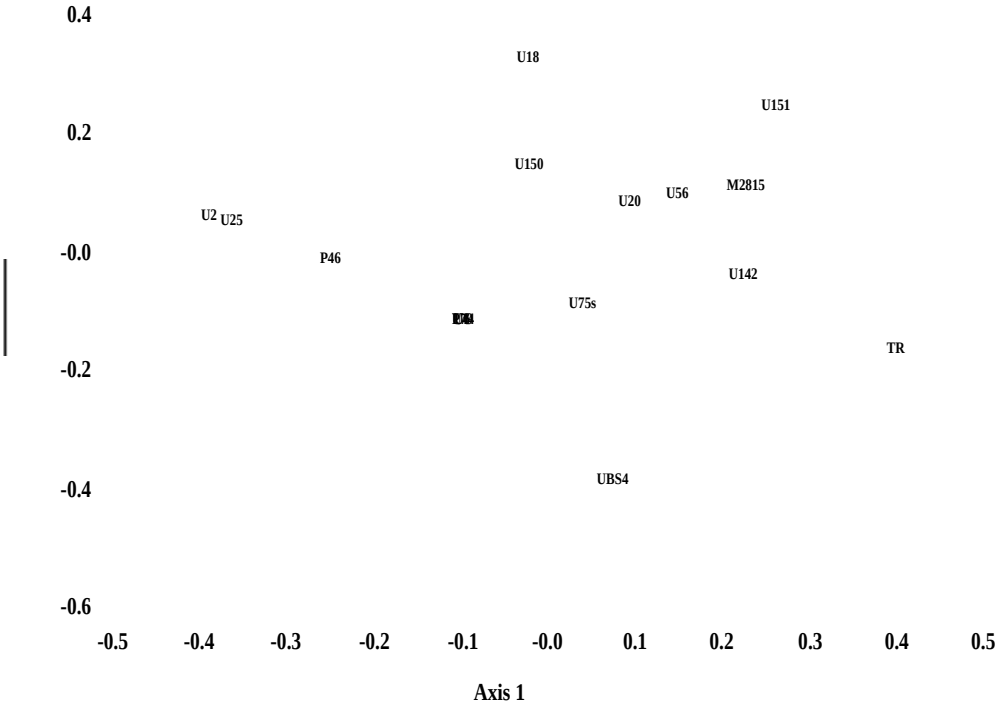
P46-x

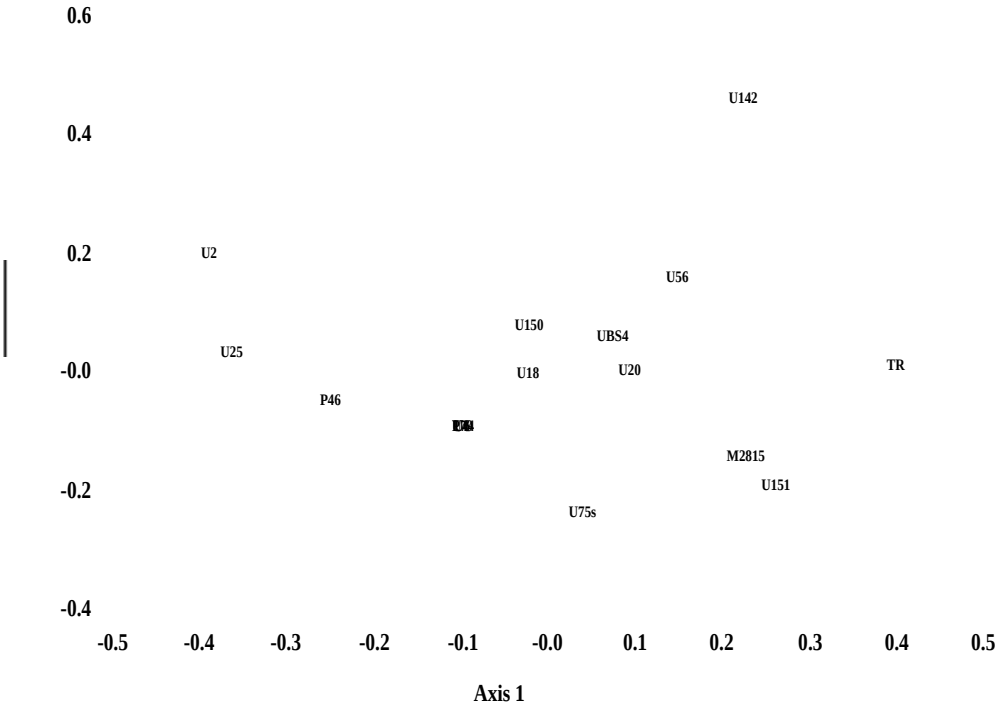




12 units; 43%, 22%, 15%

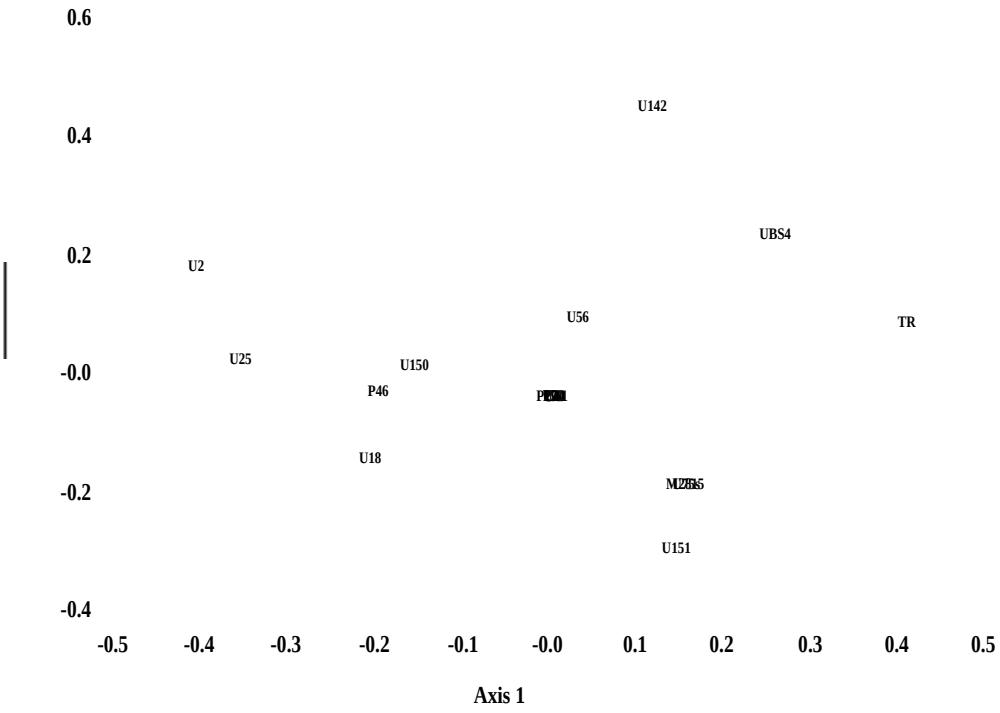
P79 (Fayyum, 650?)

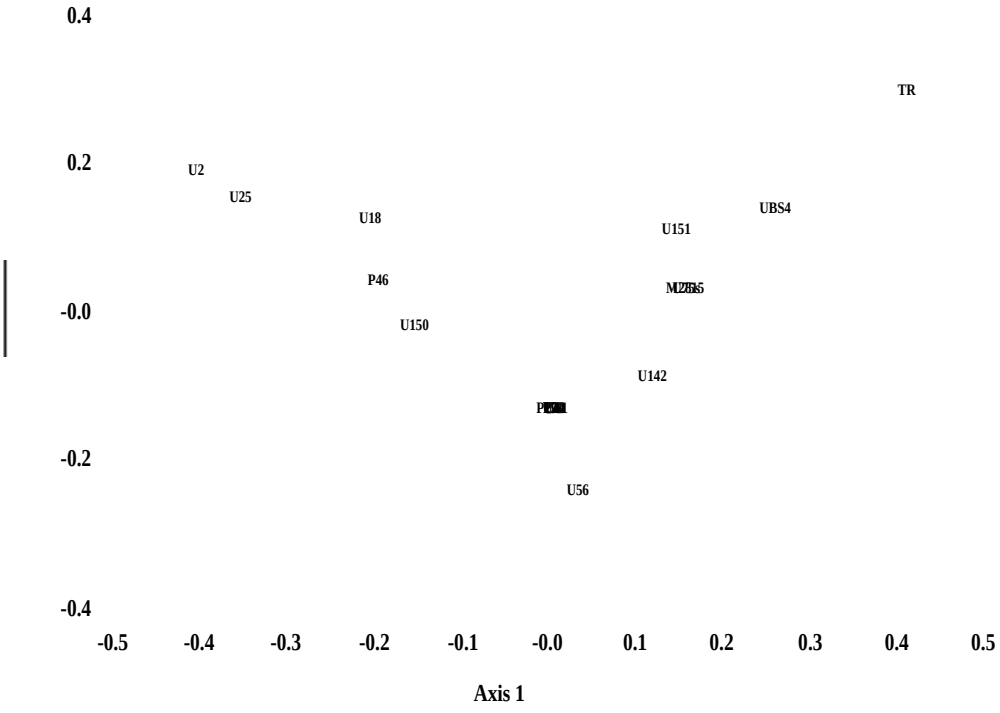




9 units; 30%, 19%, 18%

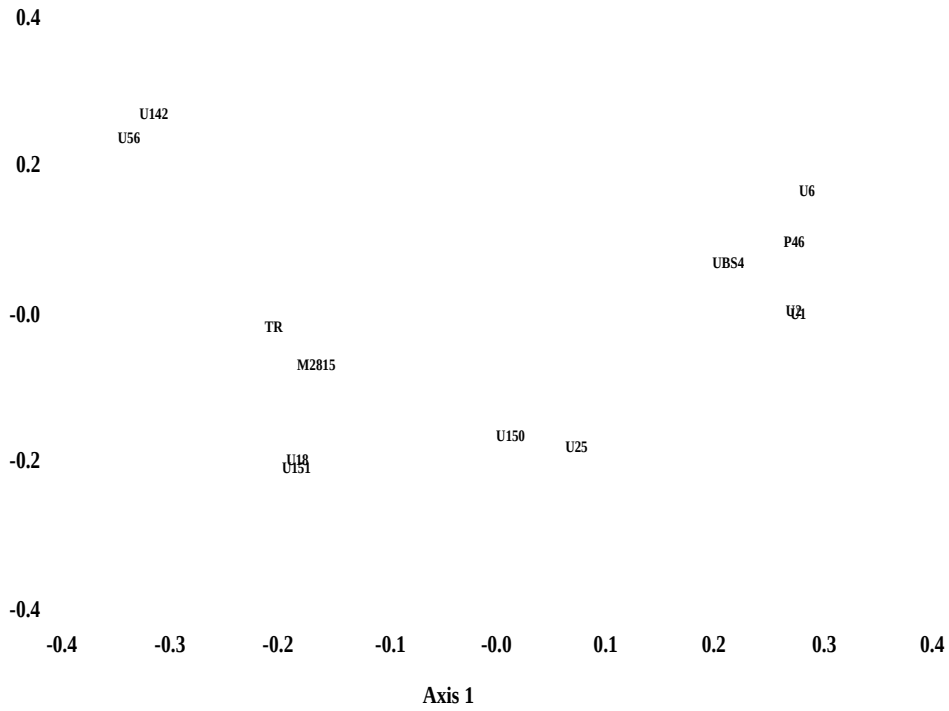
P79-1

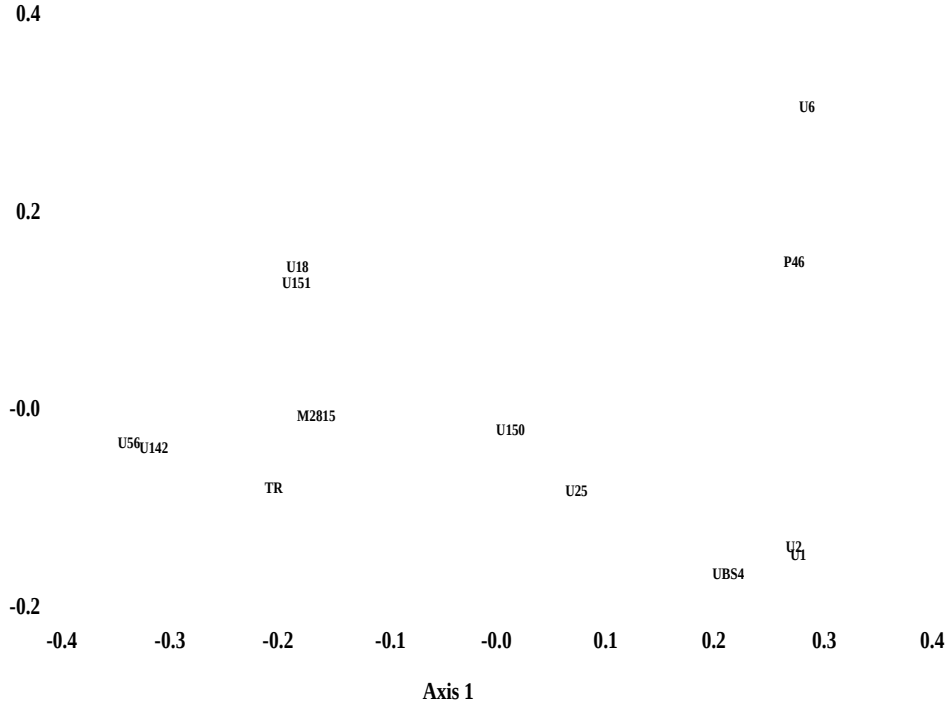




8 units; 31%, 22%, 12%

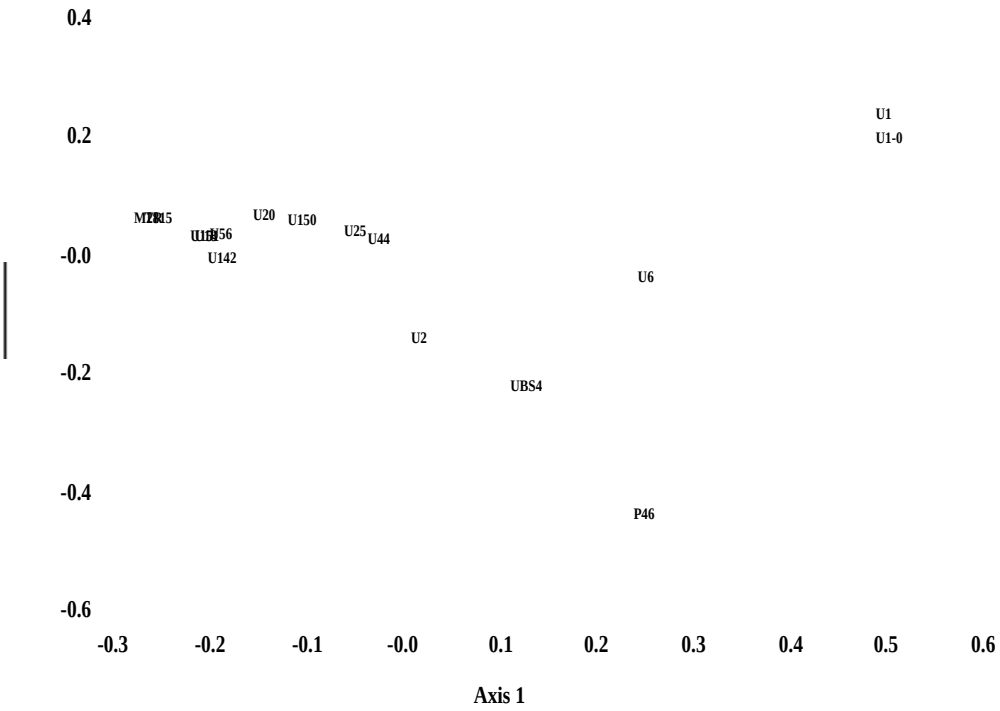
U1 (Caesarea? 350?)

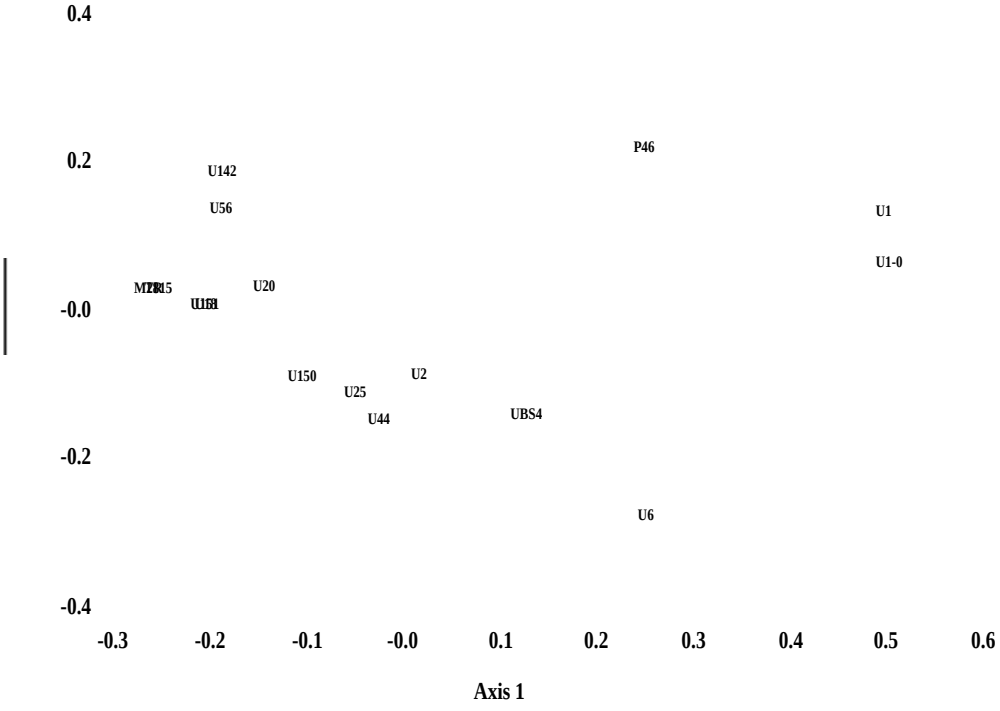




630 units; 31%, 14%, 10%

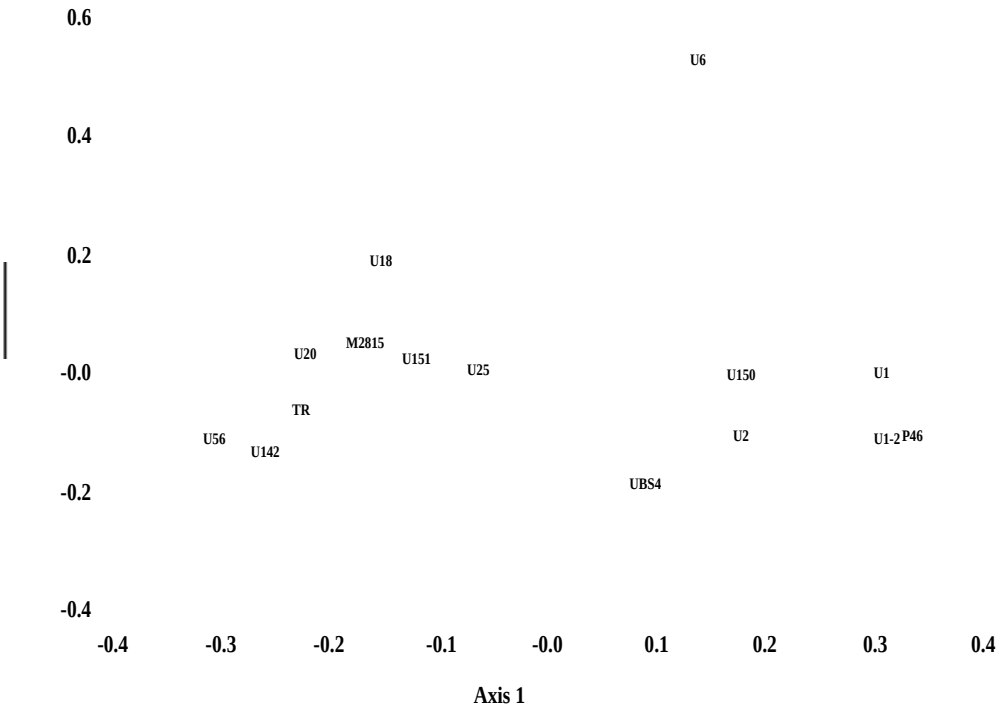
U1-0

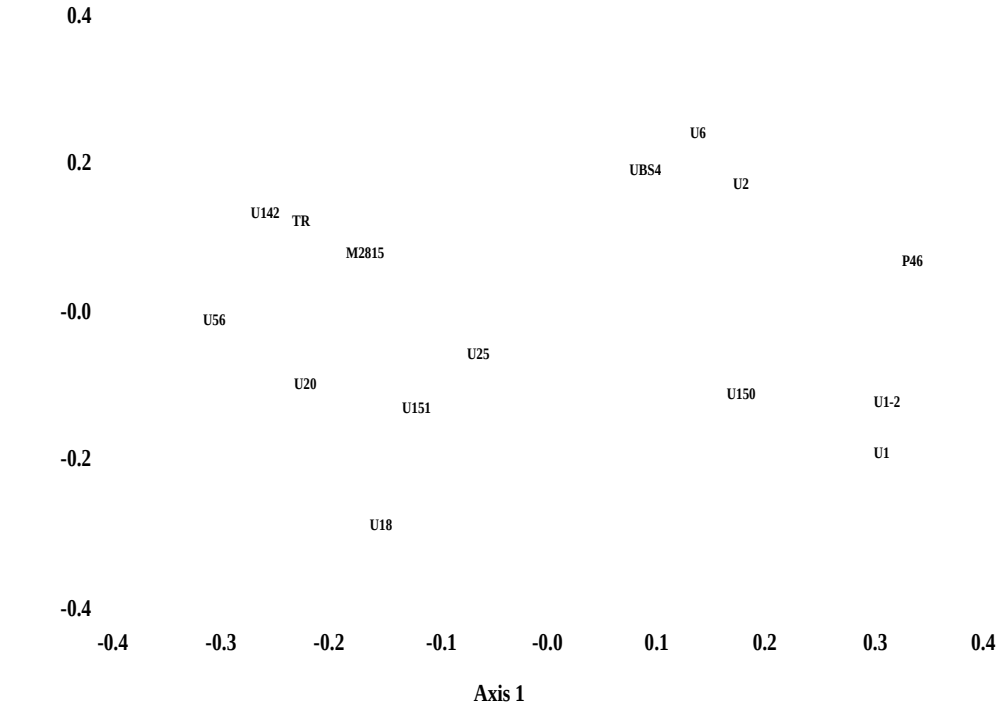




30 units; 40%, 15%, 11%

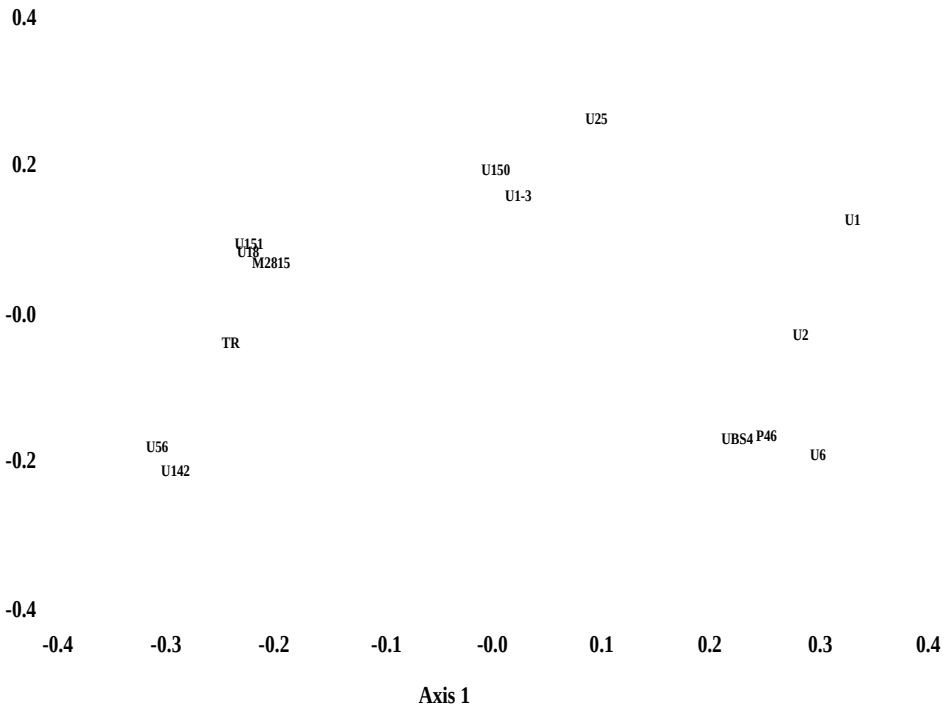
U1-2

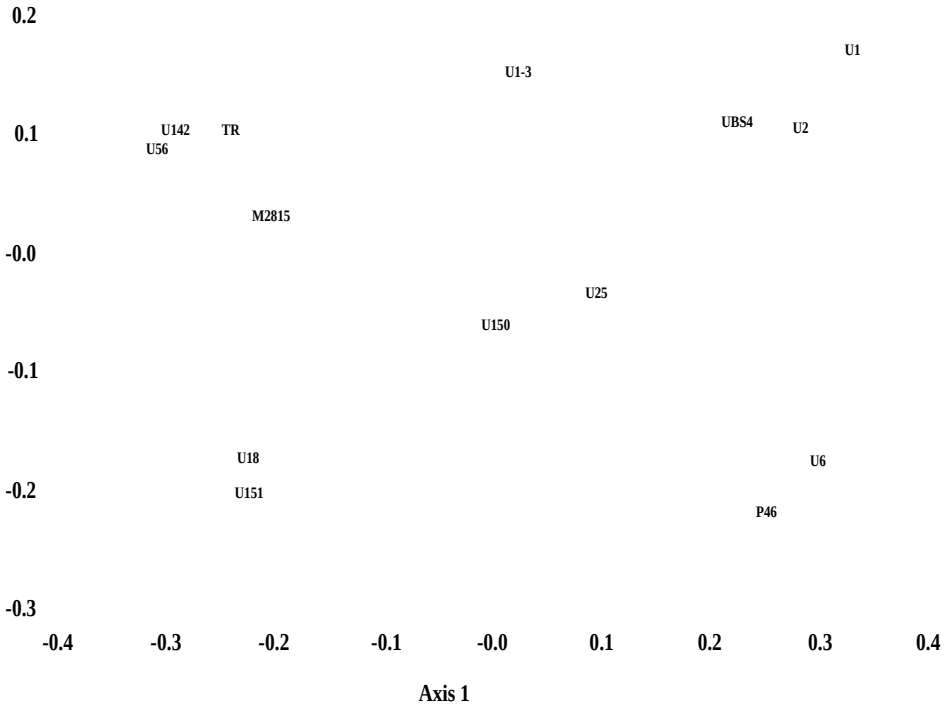




35 units; 29%, 17%, 13%

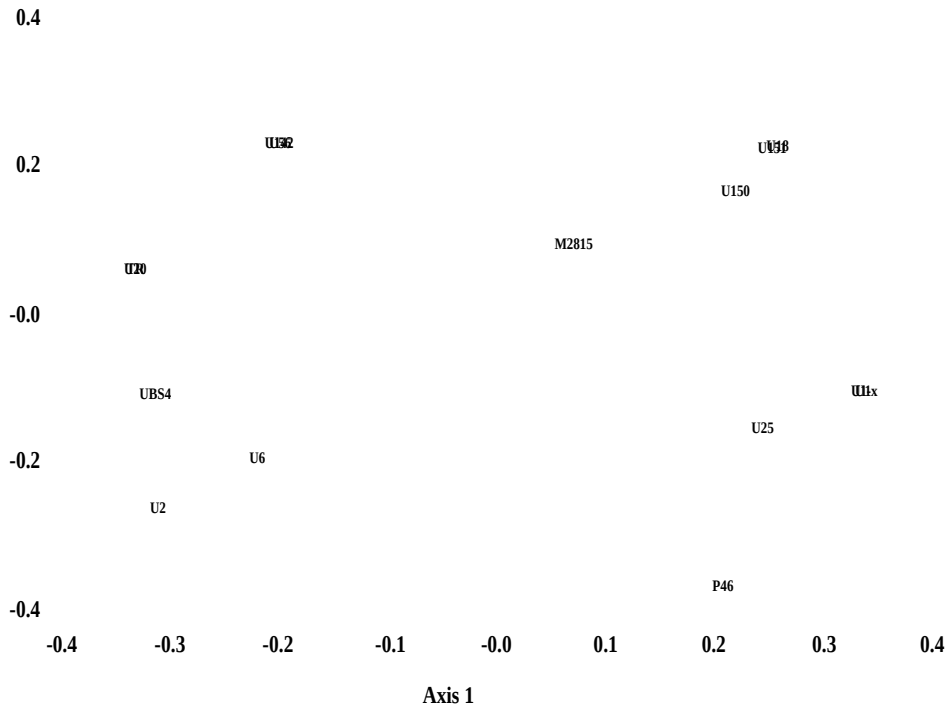
U1-3

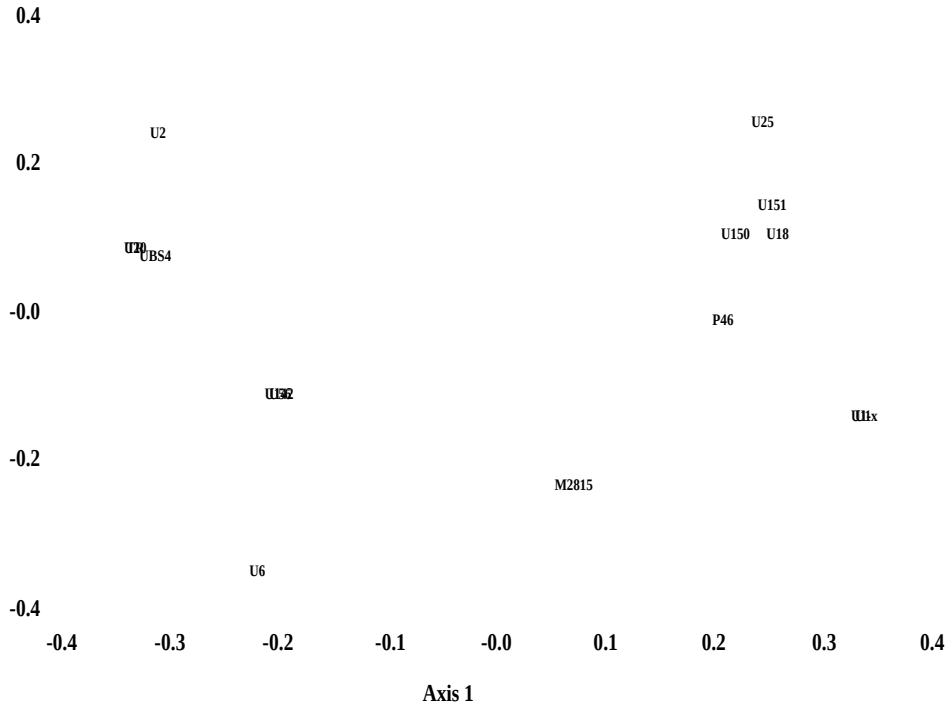




313 units; 31%, 13%, 10%

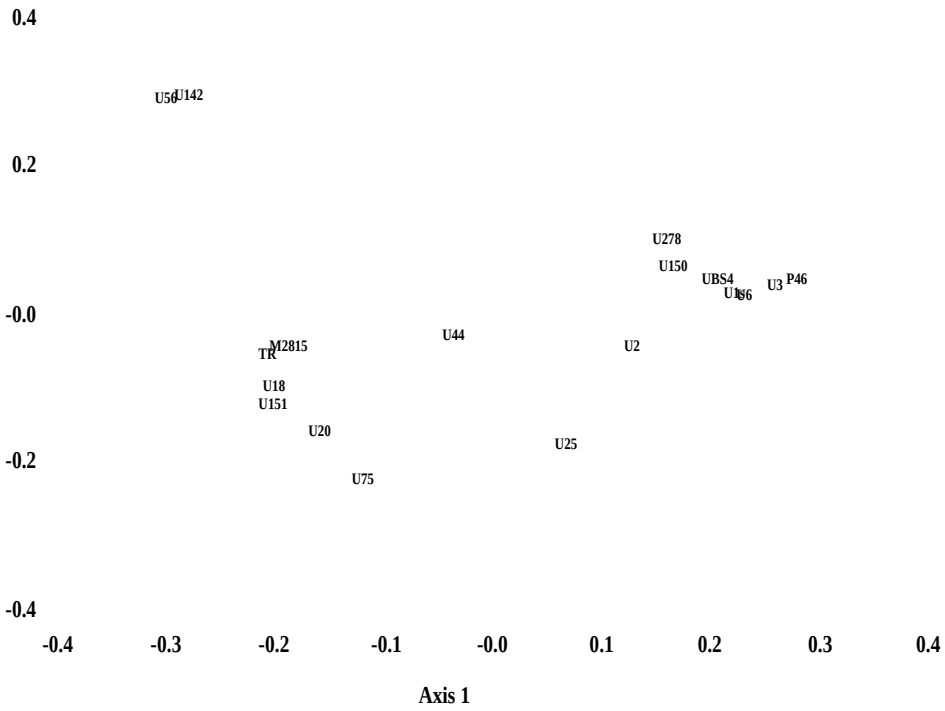
U1-x

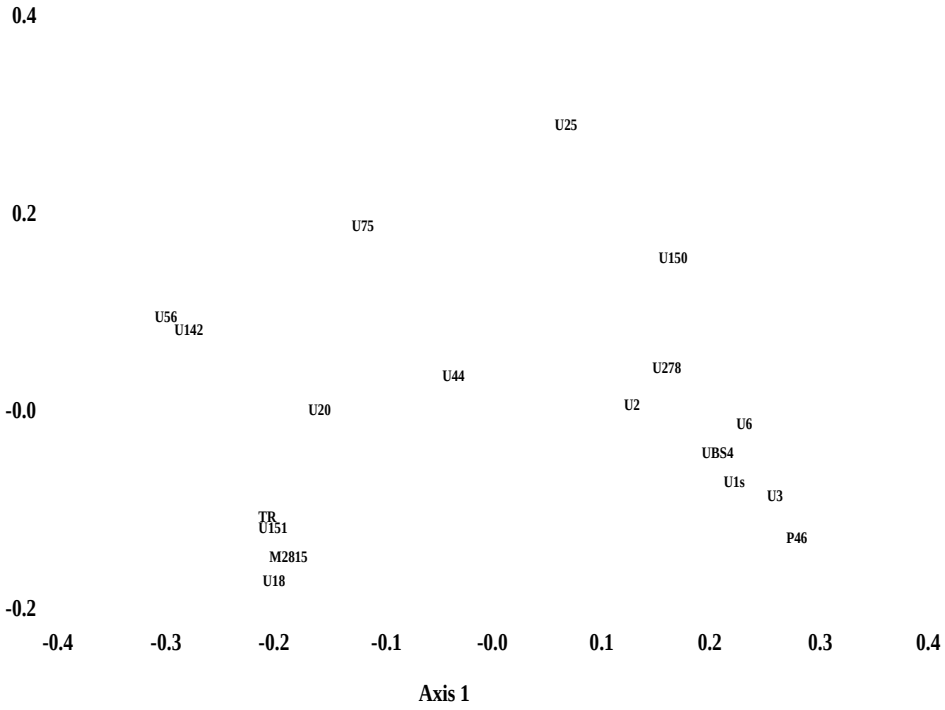




14 units; 38%, 20%, 15%

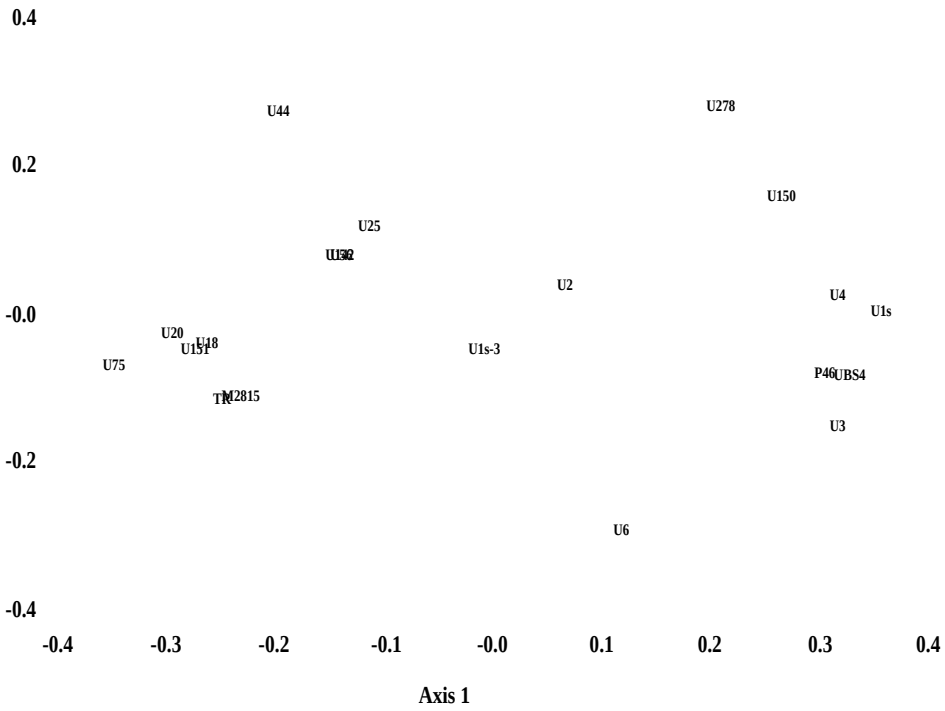
U1s (Caesarea? 350?)

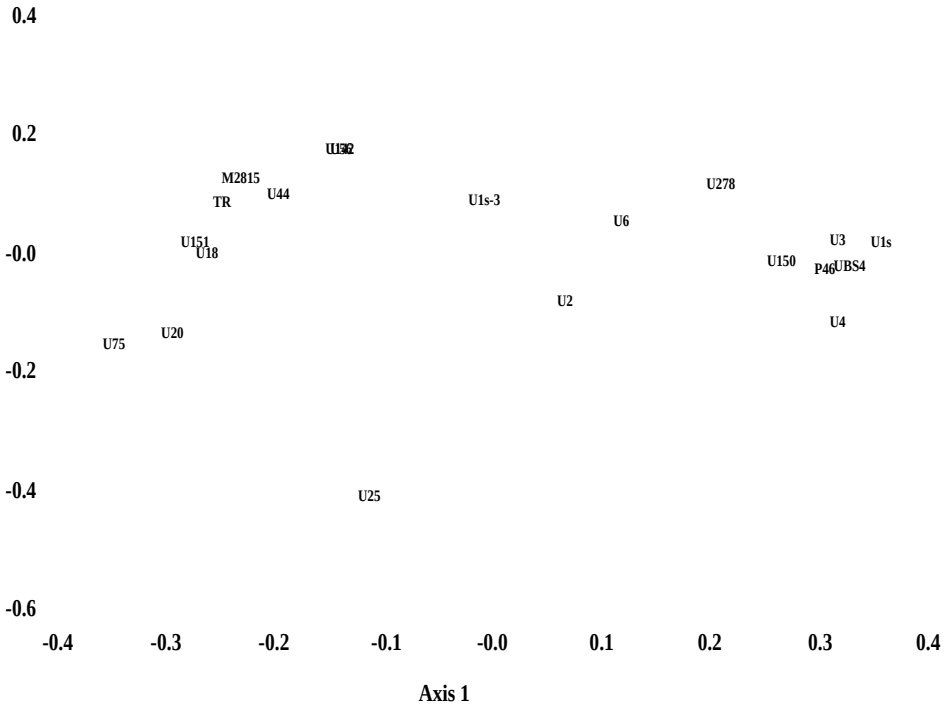




169 units; 26%, 12%, 9%

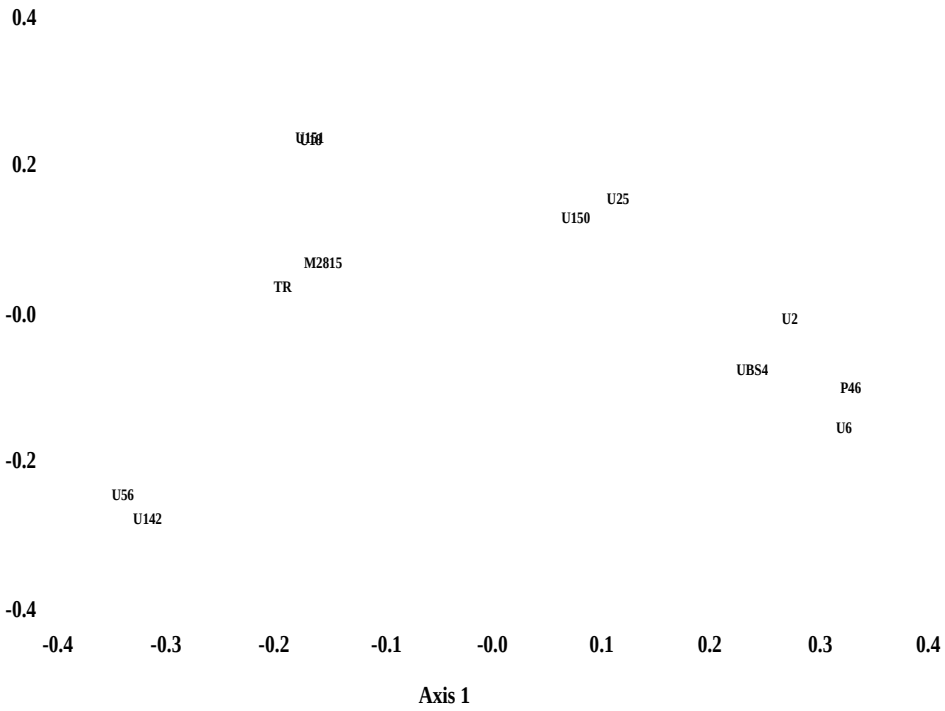
U1s-3

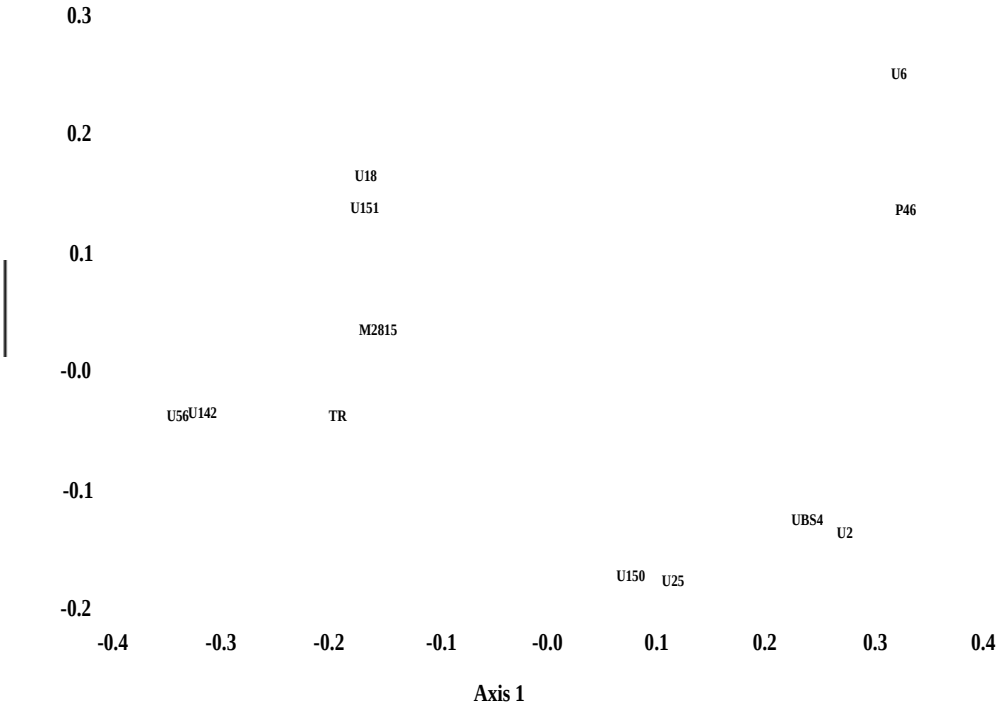




46 units; 37%, 11%, 11%

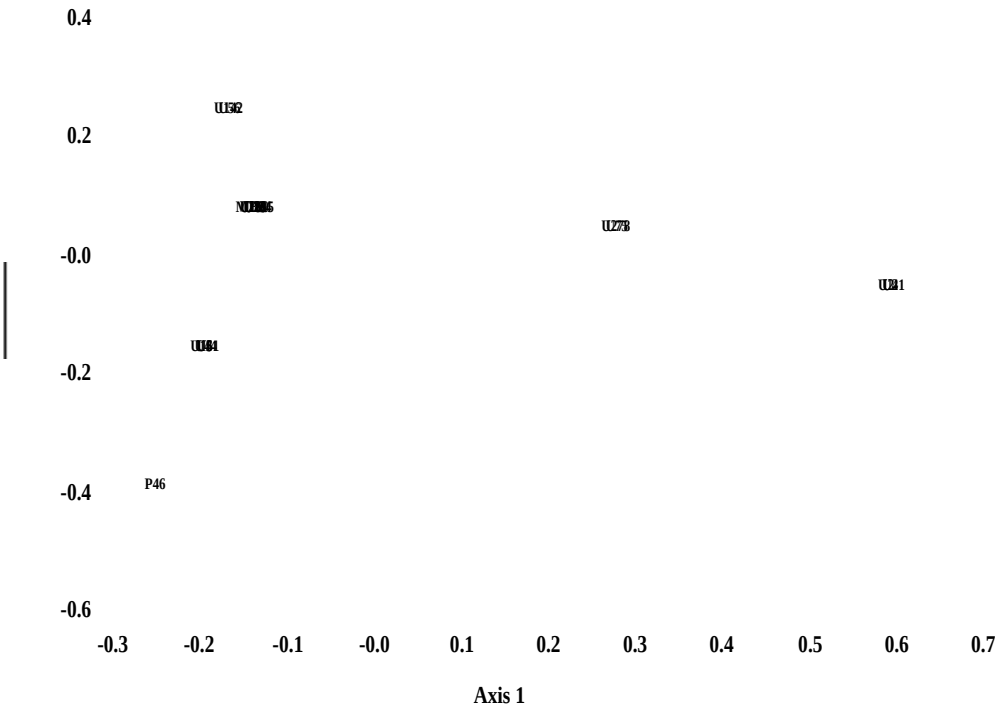
U2 (Alexandria? 450?)

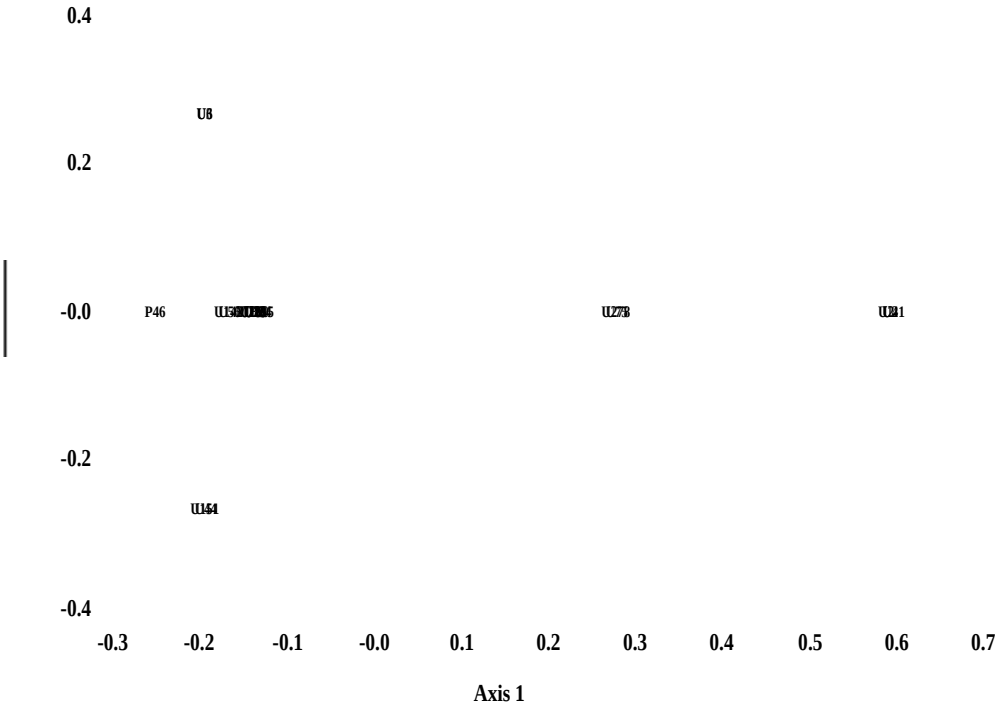




728 units; 32%, 15%, 10%

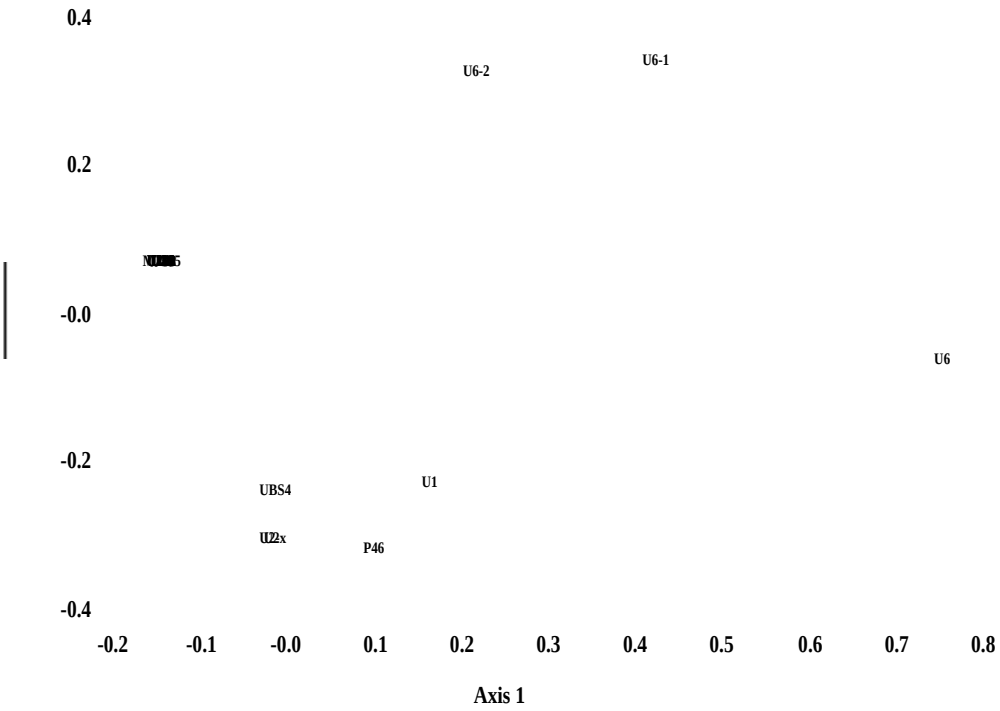
U2-1

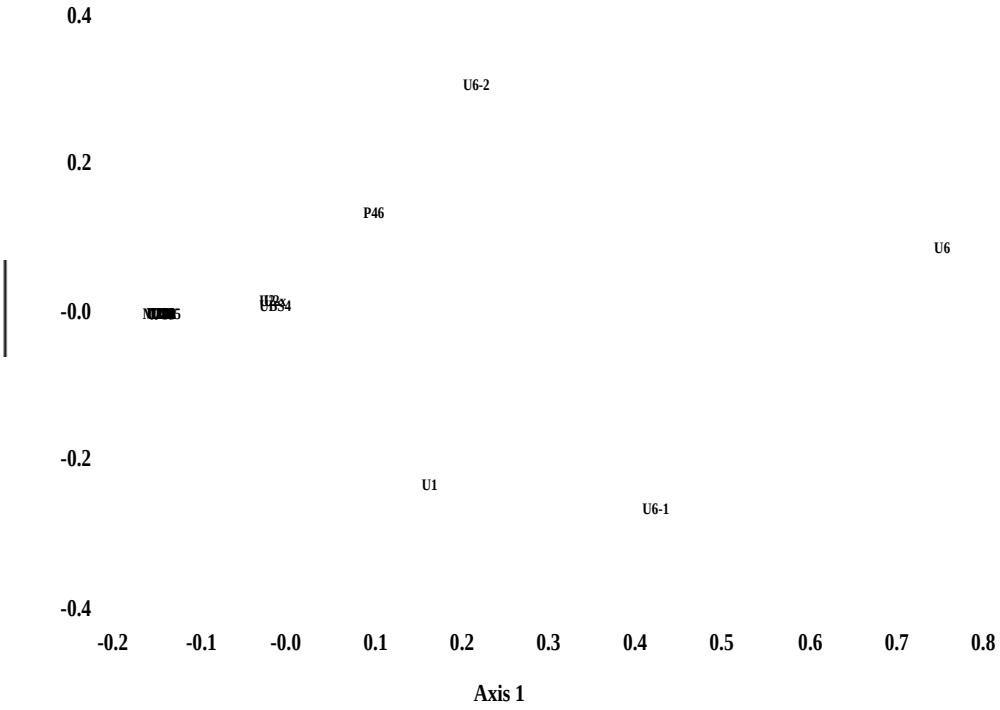




7 units; 58%, 15%, 10%

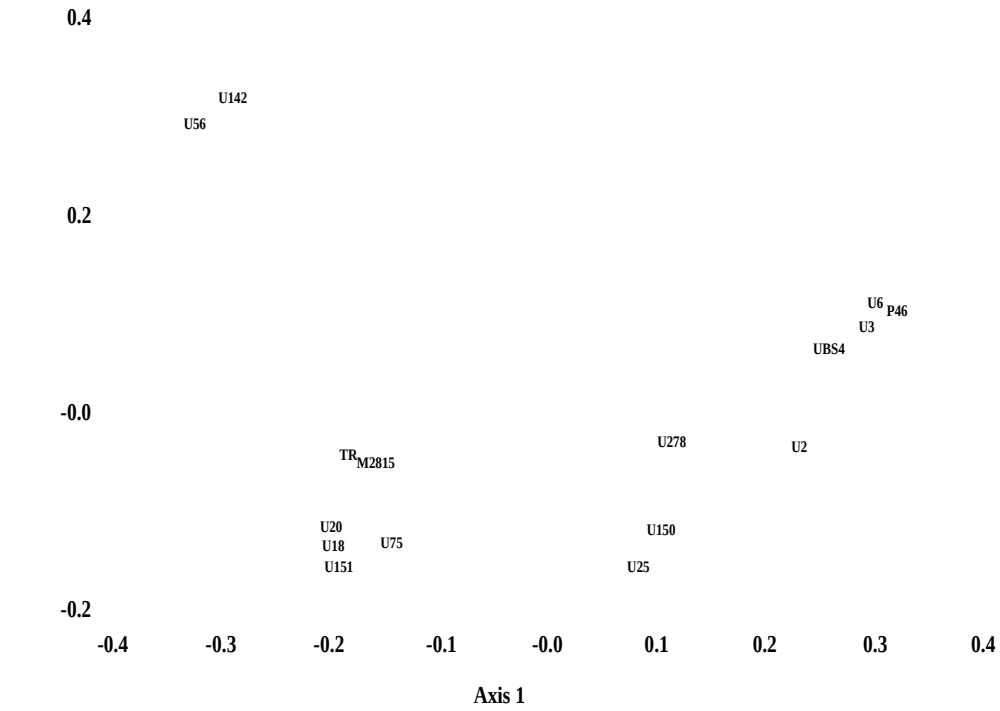
U2-x

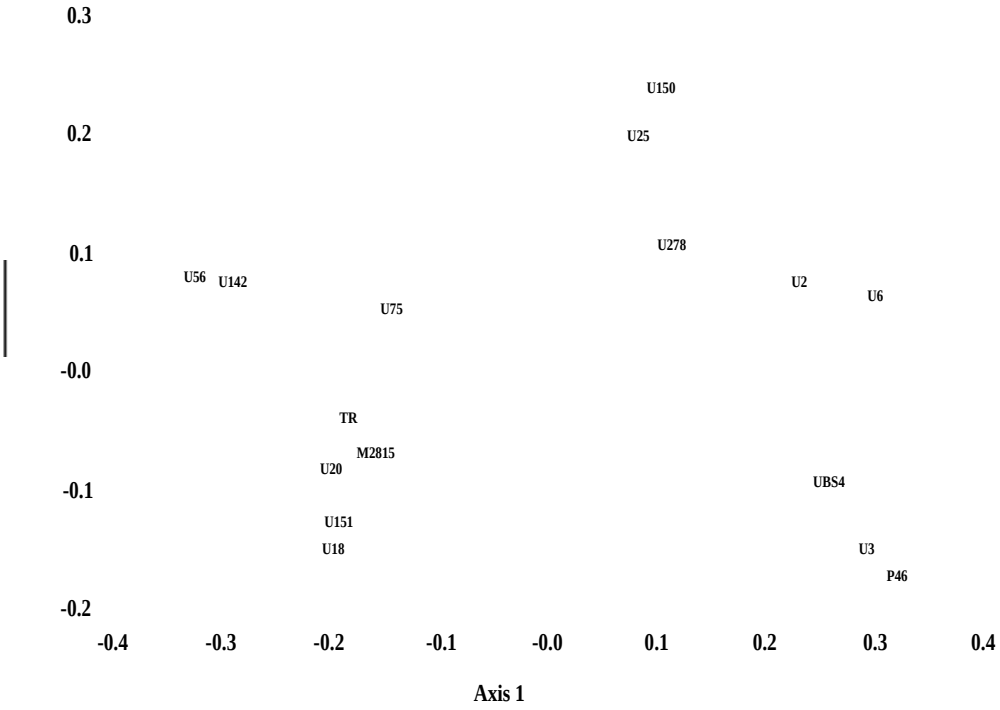




14 units; 46%, 29%, 11%

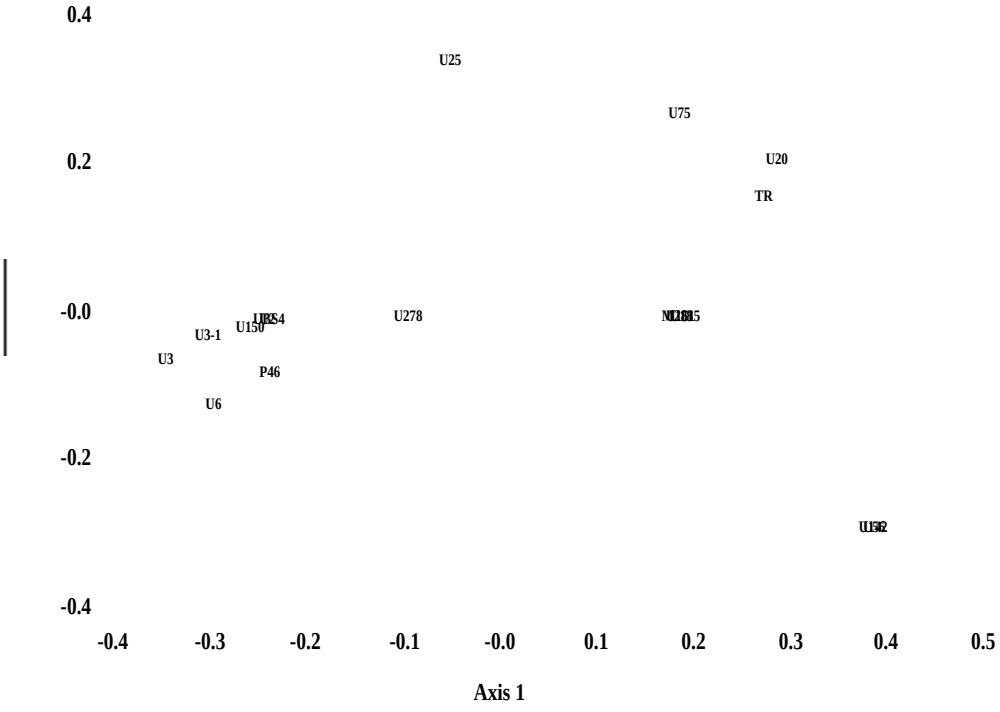
U3 (Alexandria? 325?)

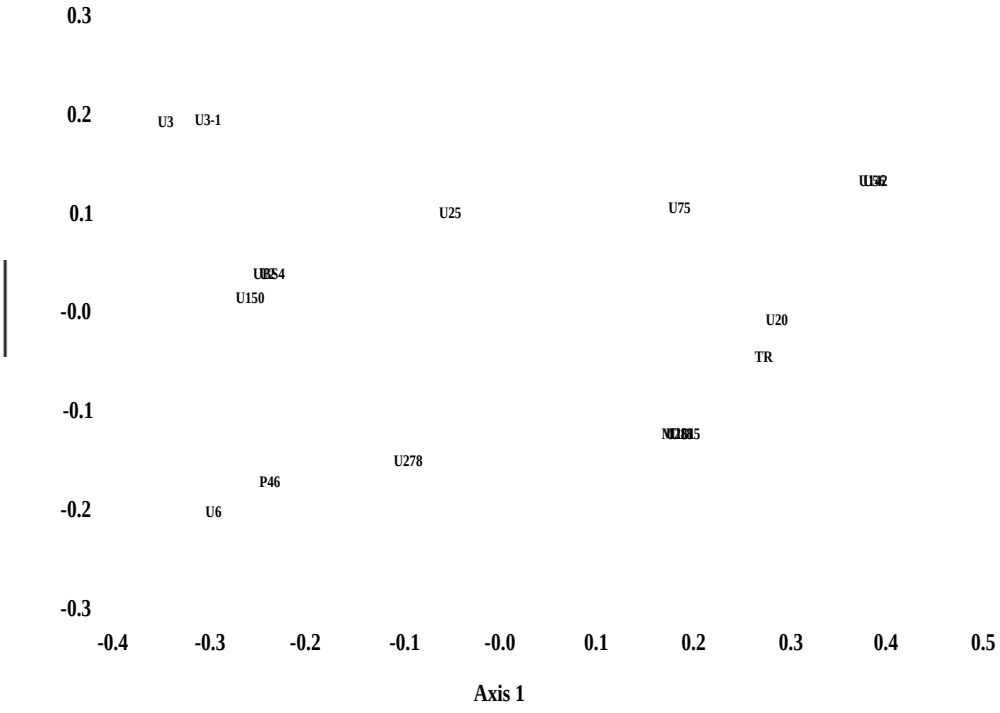




401 units; 30%, 13%, 9%

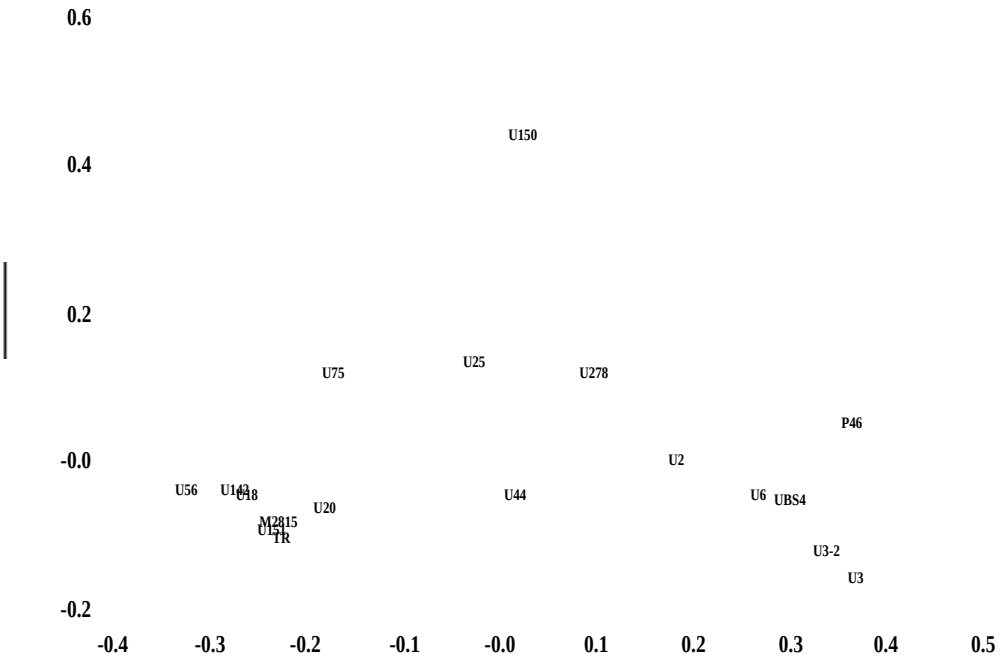
U3-1

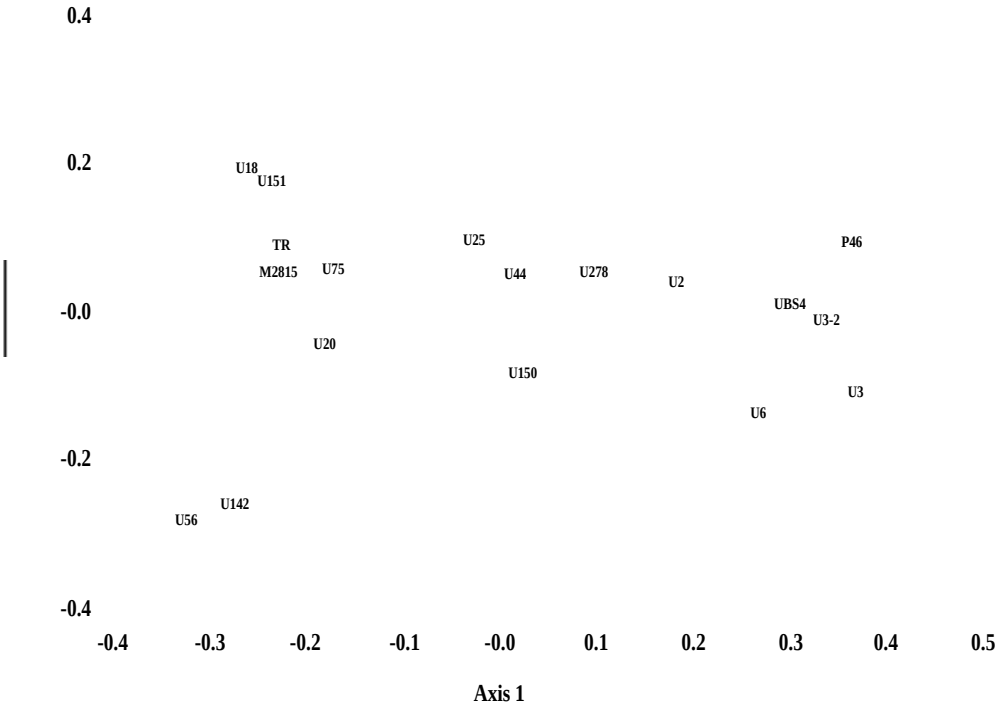




27 units; 44%, 18%, 11%

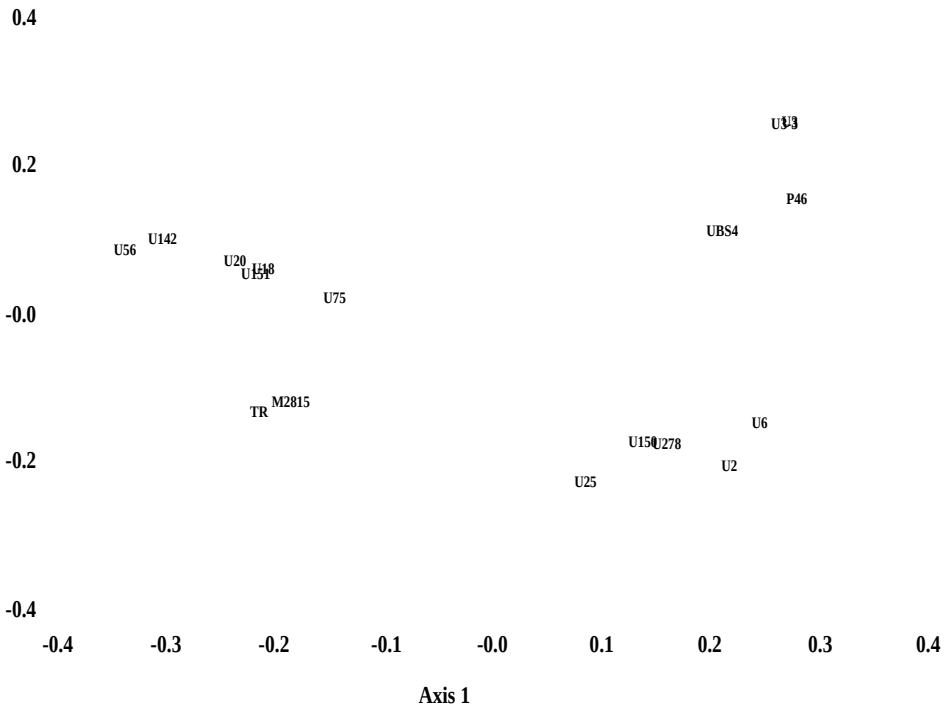
U3-2

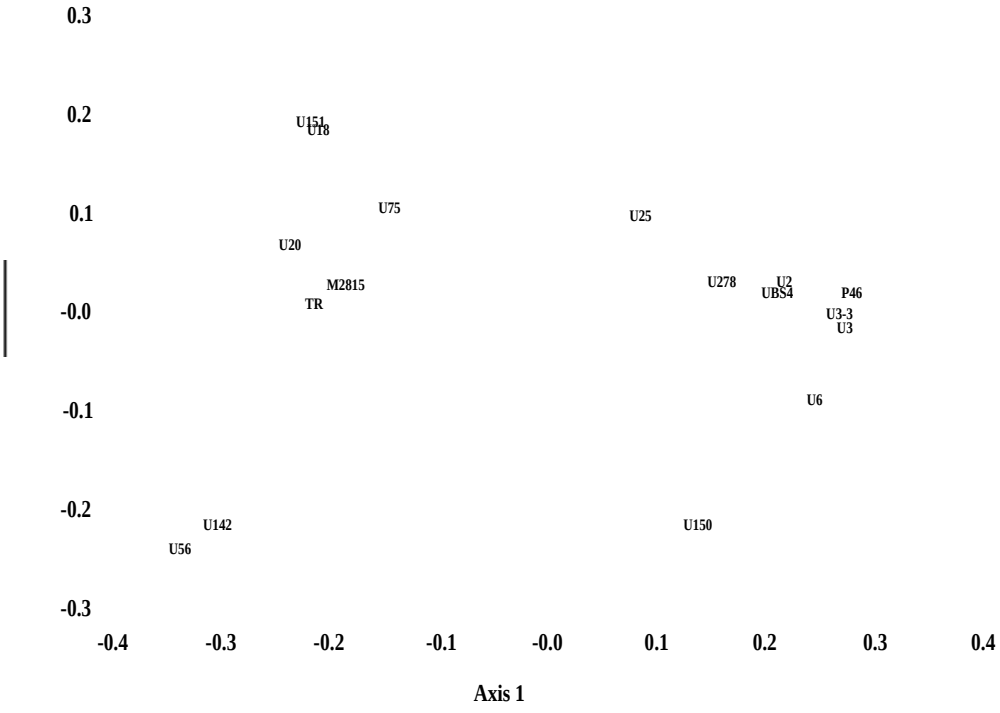




71 units; 35%, 11%, 10%

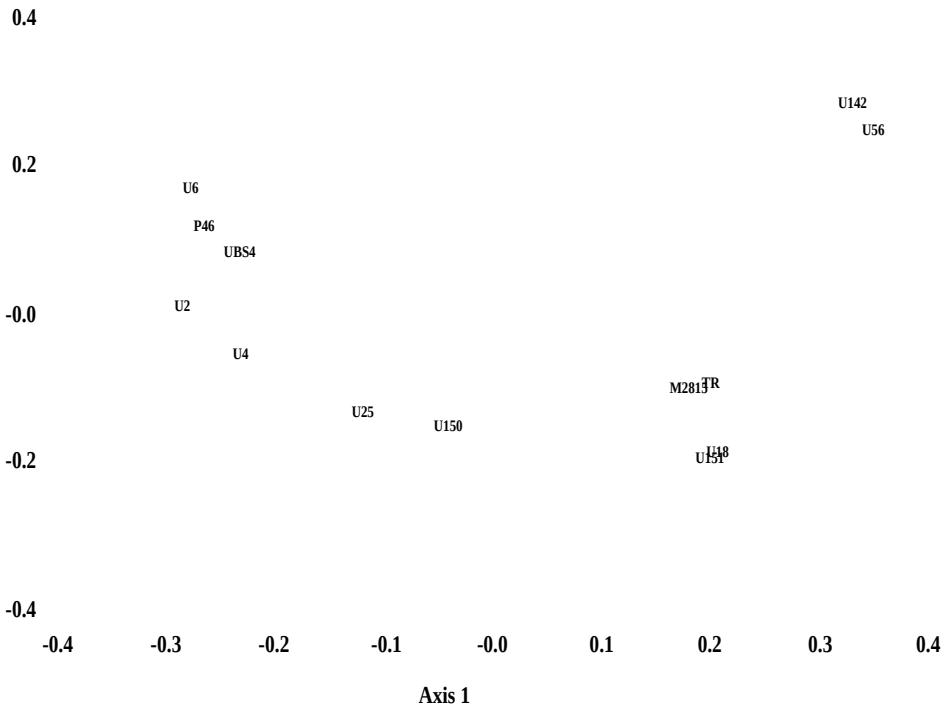
U3-3

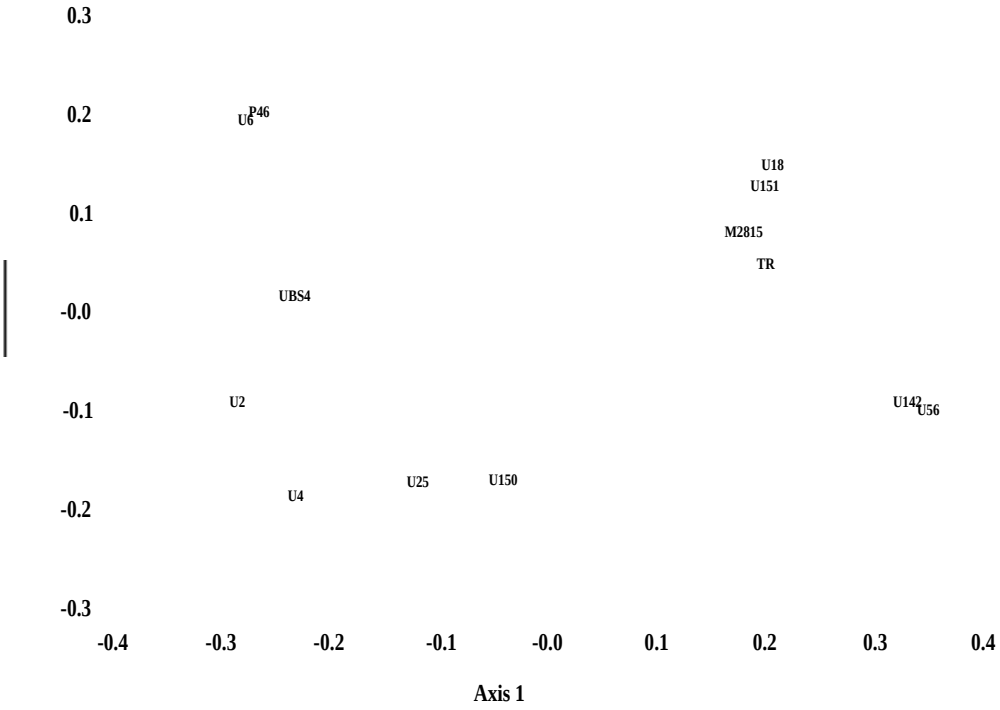




160 units; 31%, 14%, 9%

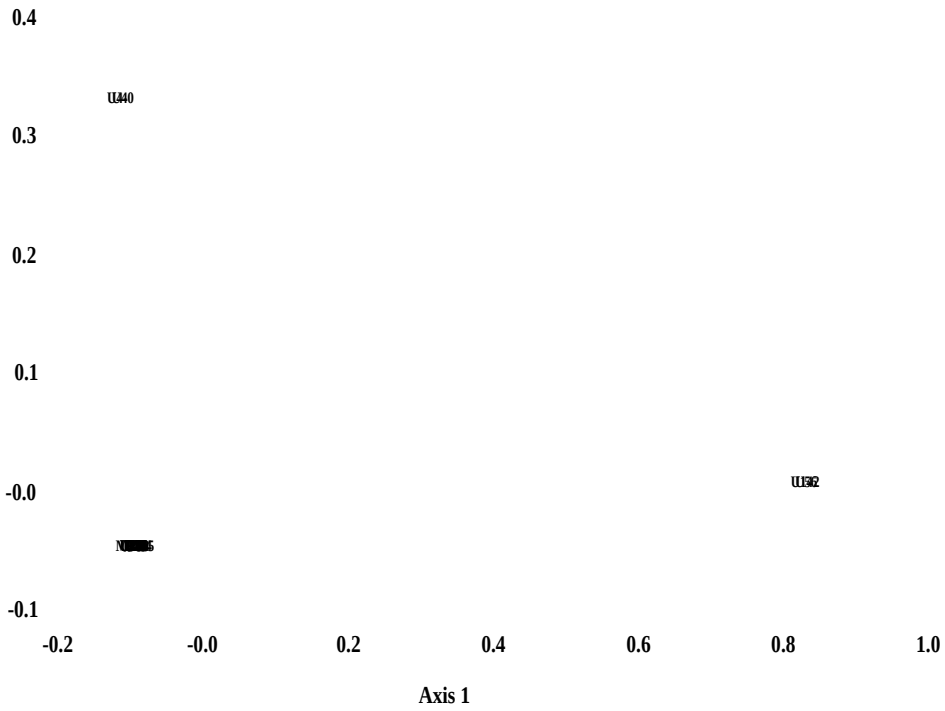
U4 (450?)





431 units; 32%, 14%, 10%

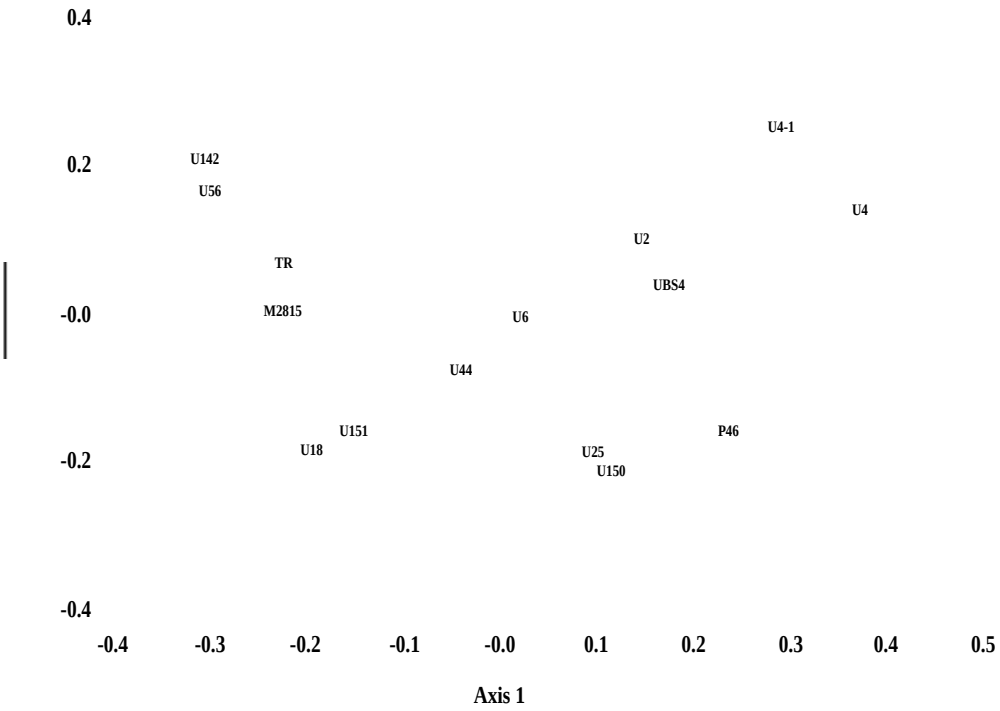
U4-0

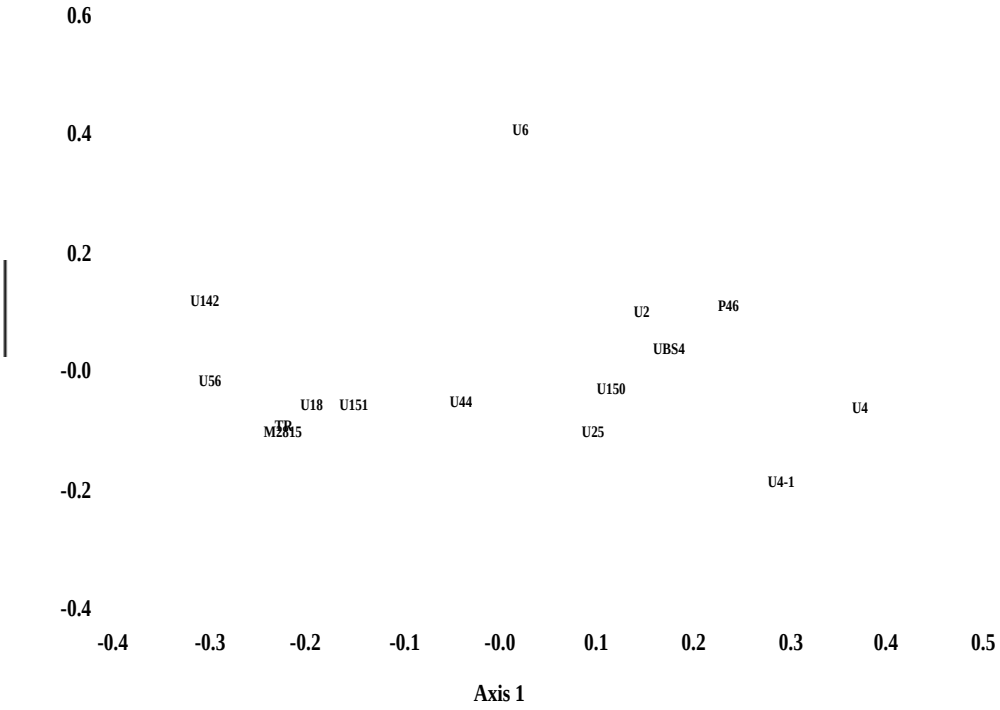


7 units; 86%, 14%

(All variation is entirely explained by the first two dimensions.)

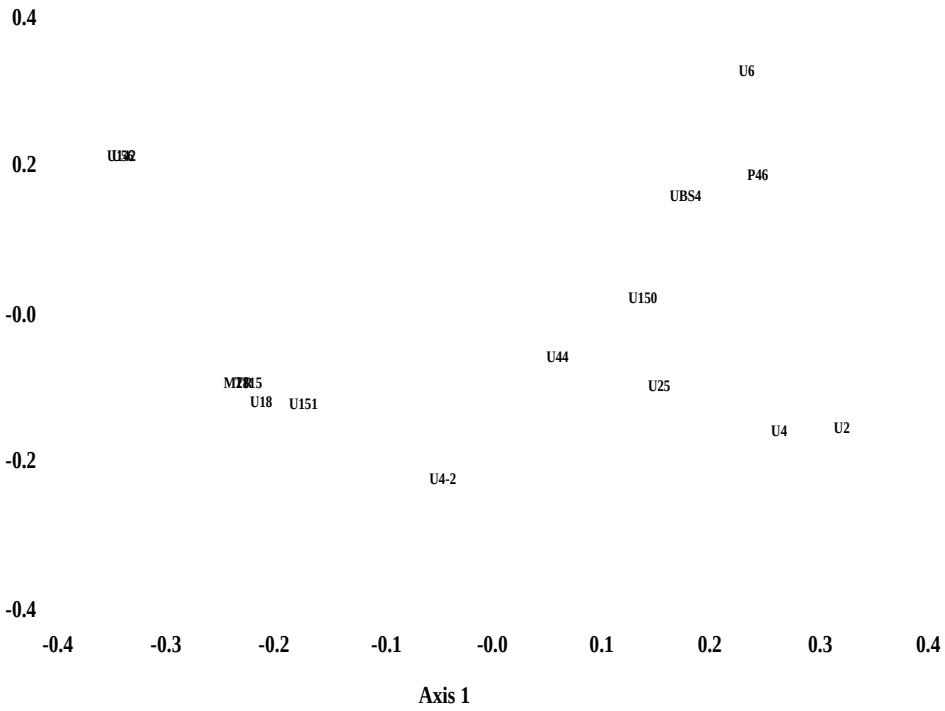
U4-1

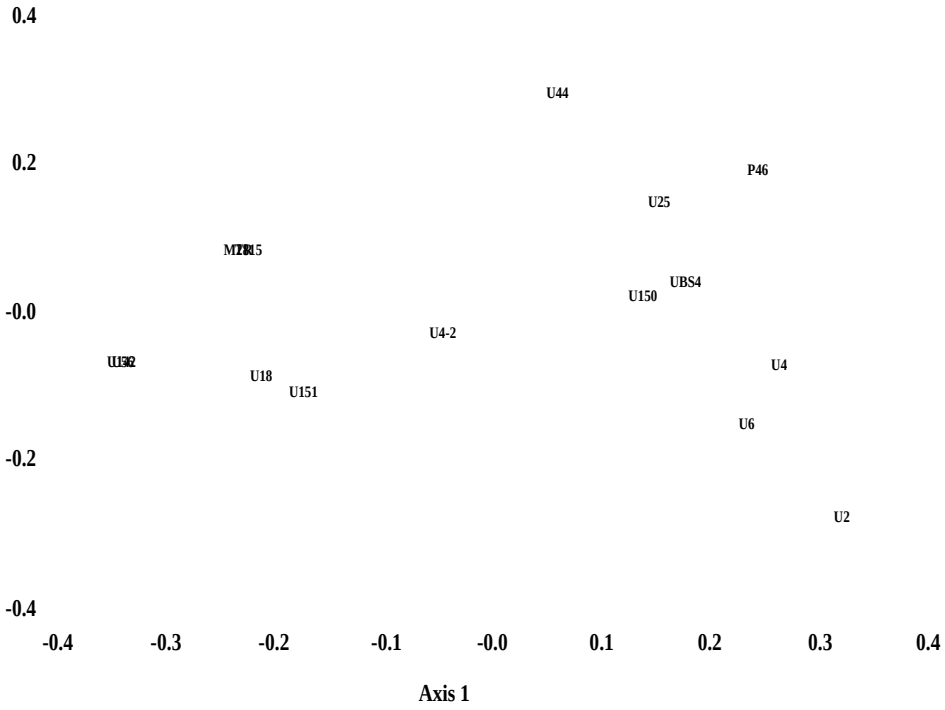




83 units; 29%, 14%, 12%

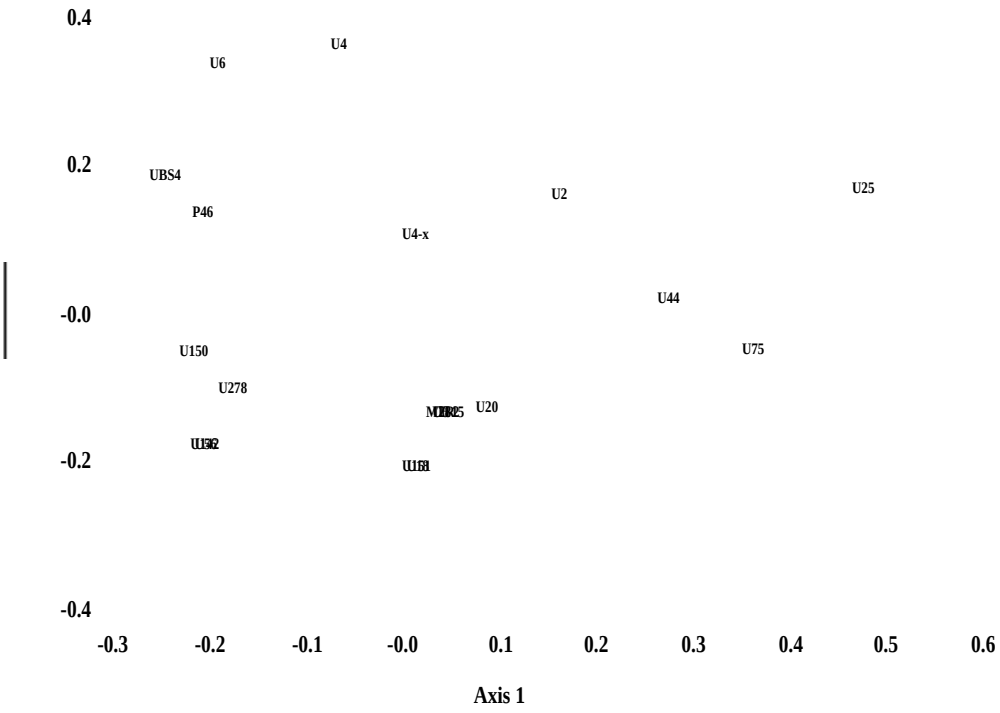
U4-2

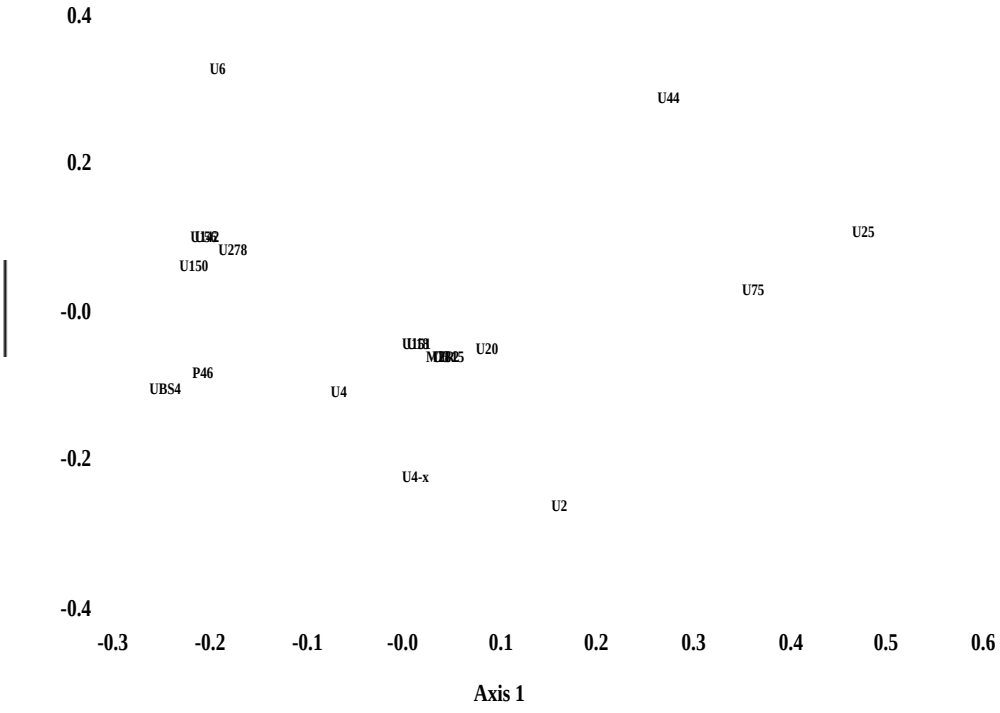




65 units; 30%, 16%, 11%

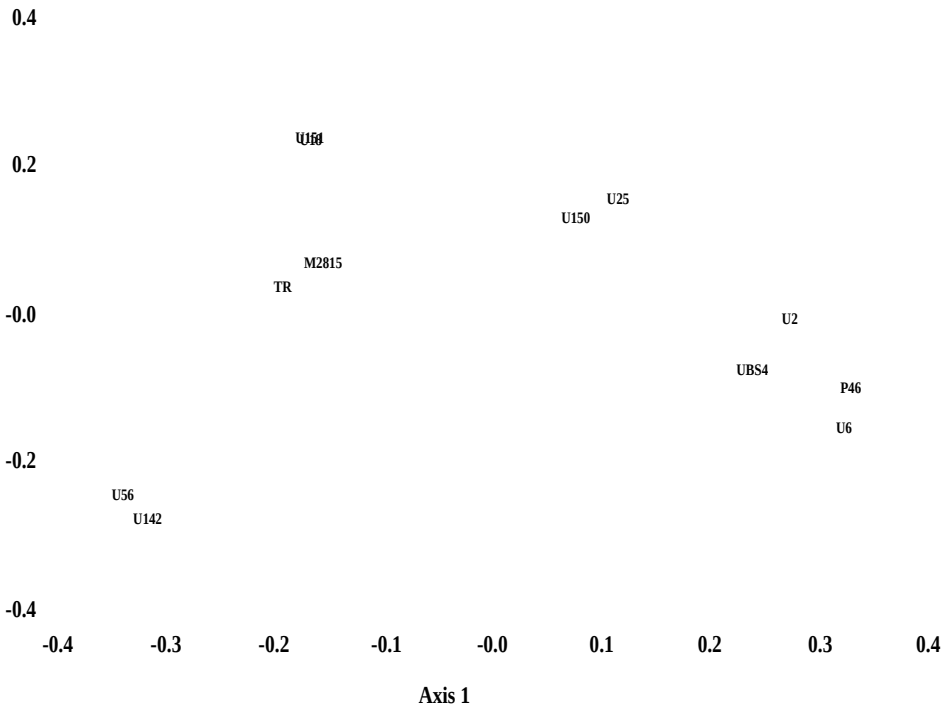
U4-x

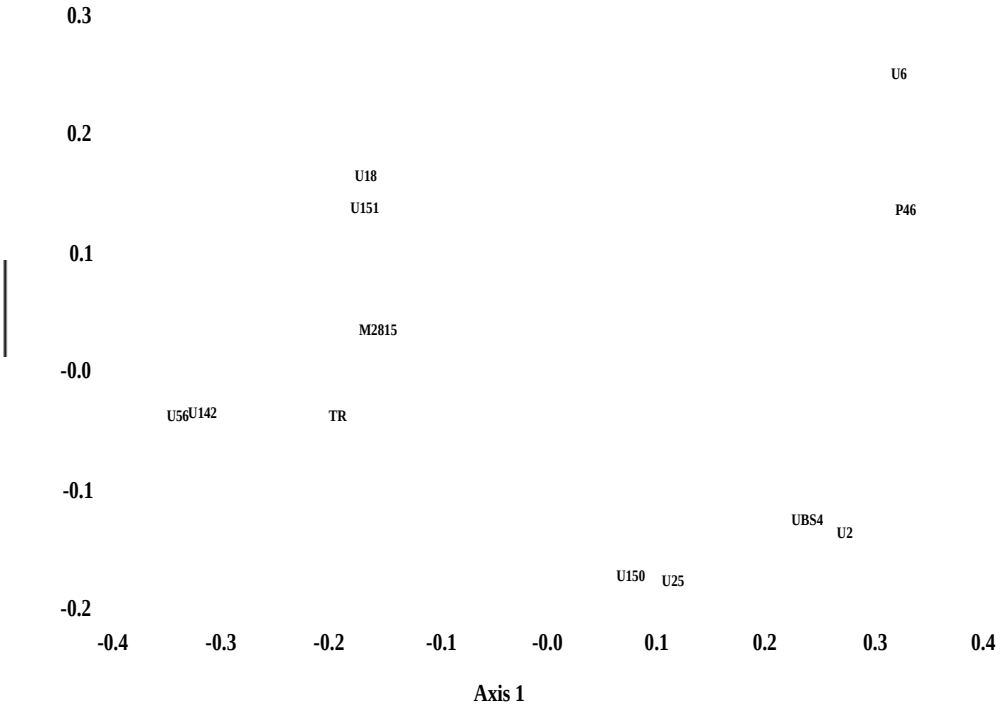




17 units; 29%, 22%, 15%

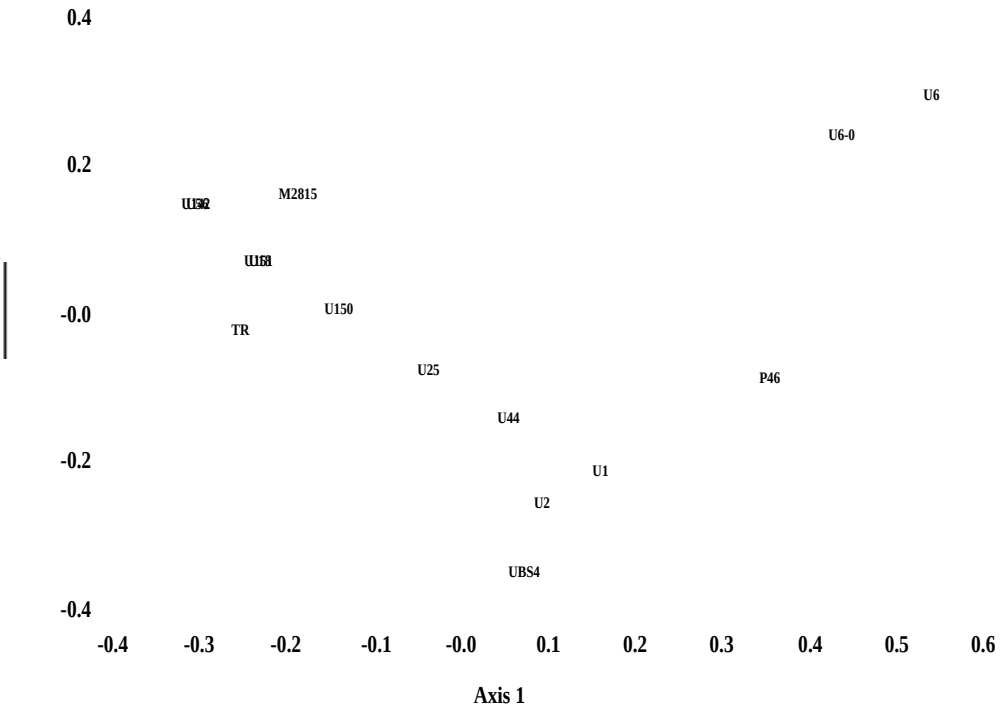
U6 (Sardinia? 500?)

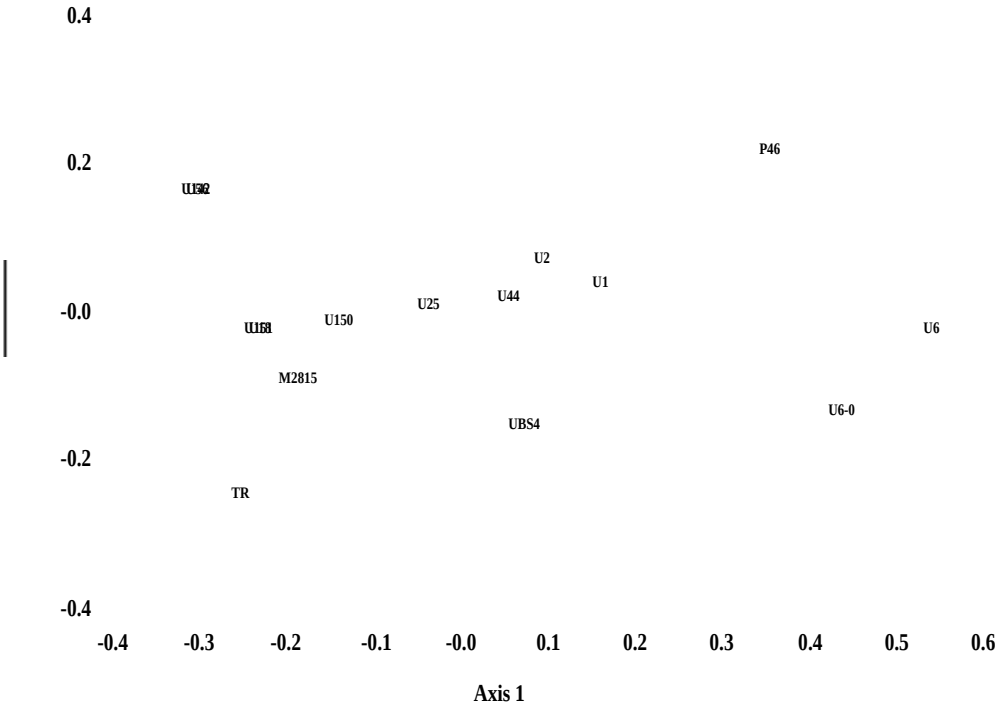




728 units; 32%, 15%, 10%

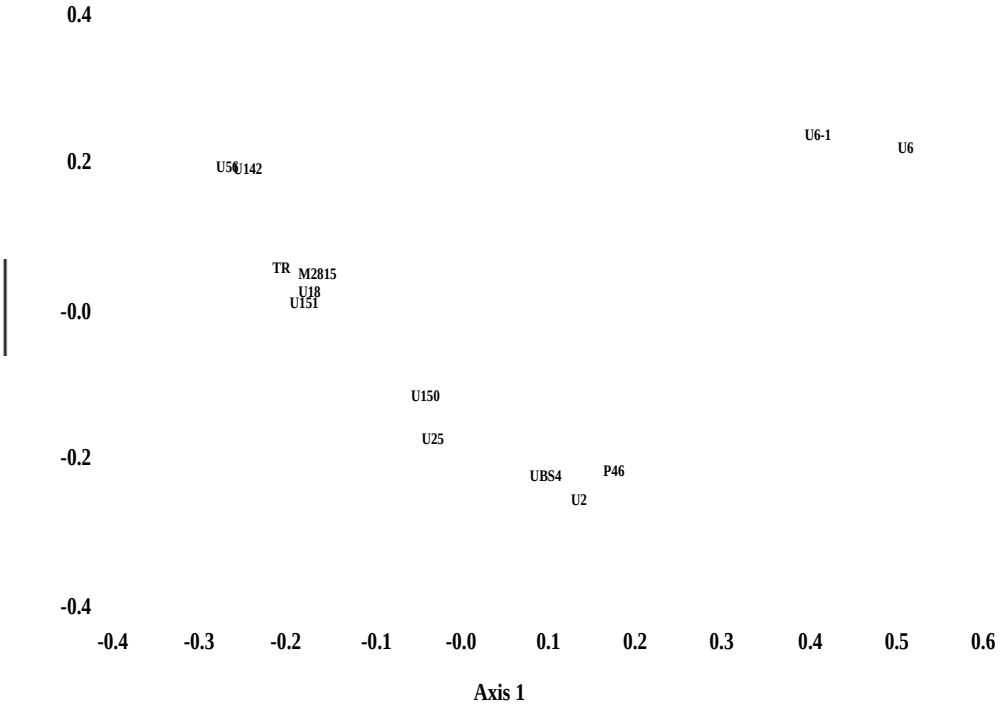
U6-0

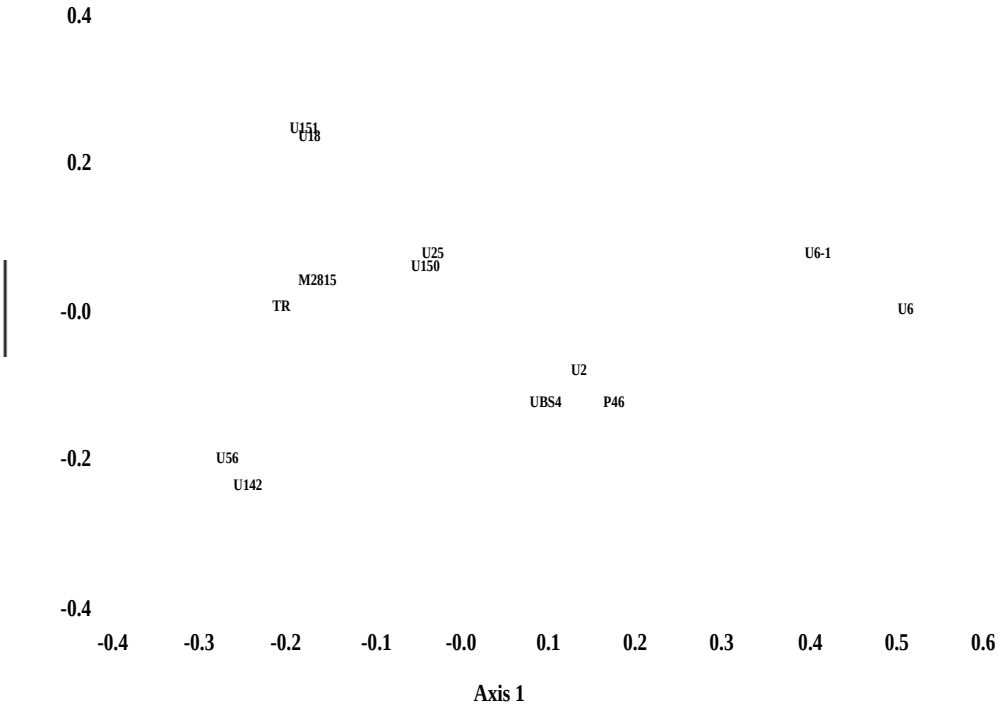




16 units; 45%, 21%, 9%

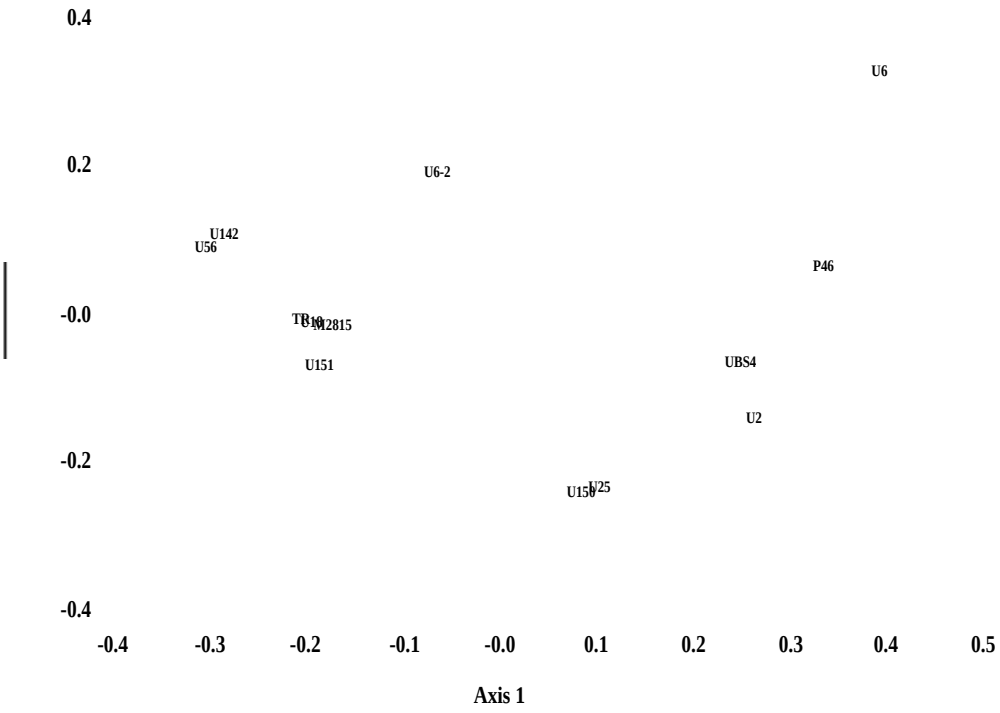
U6-1

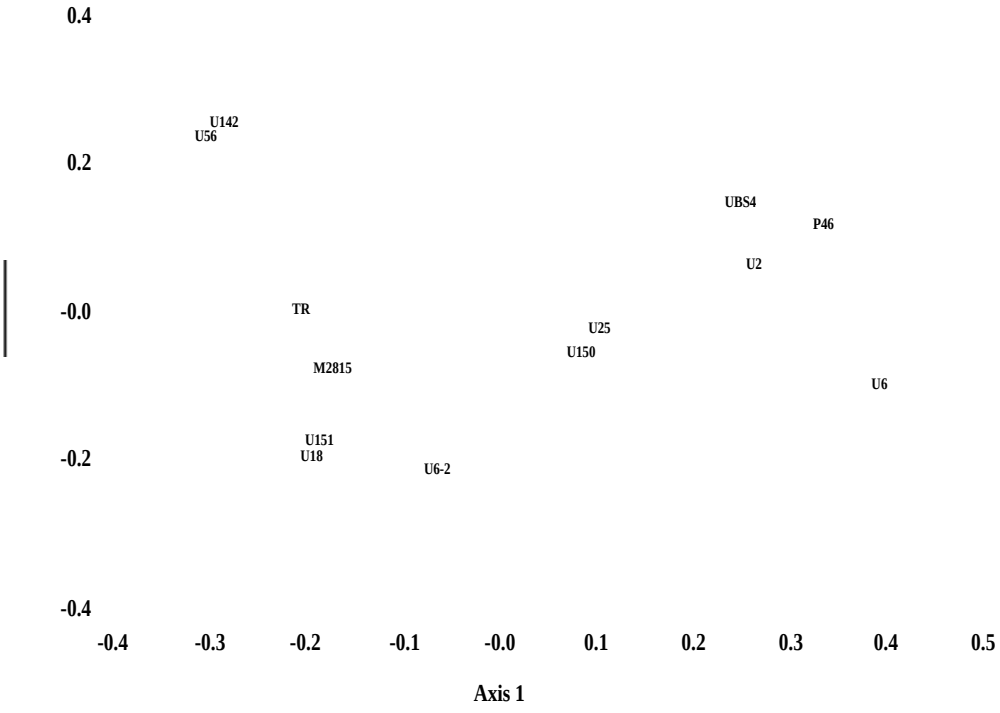




670 units; 33%, 17%, 12%

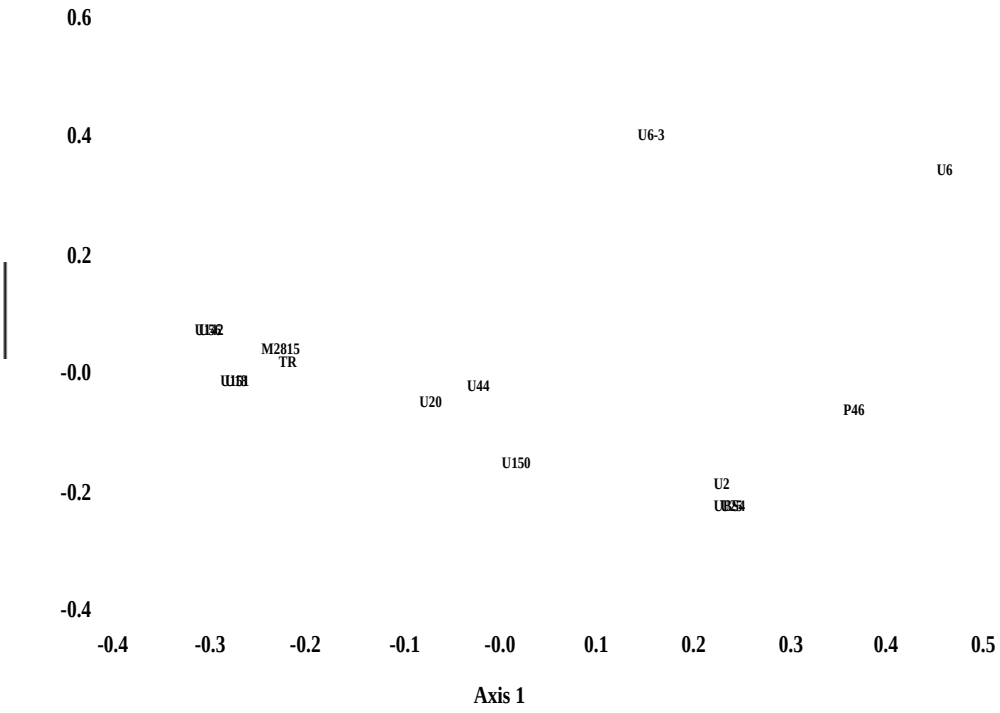
U6-2

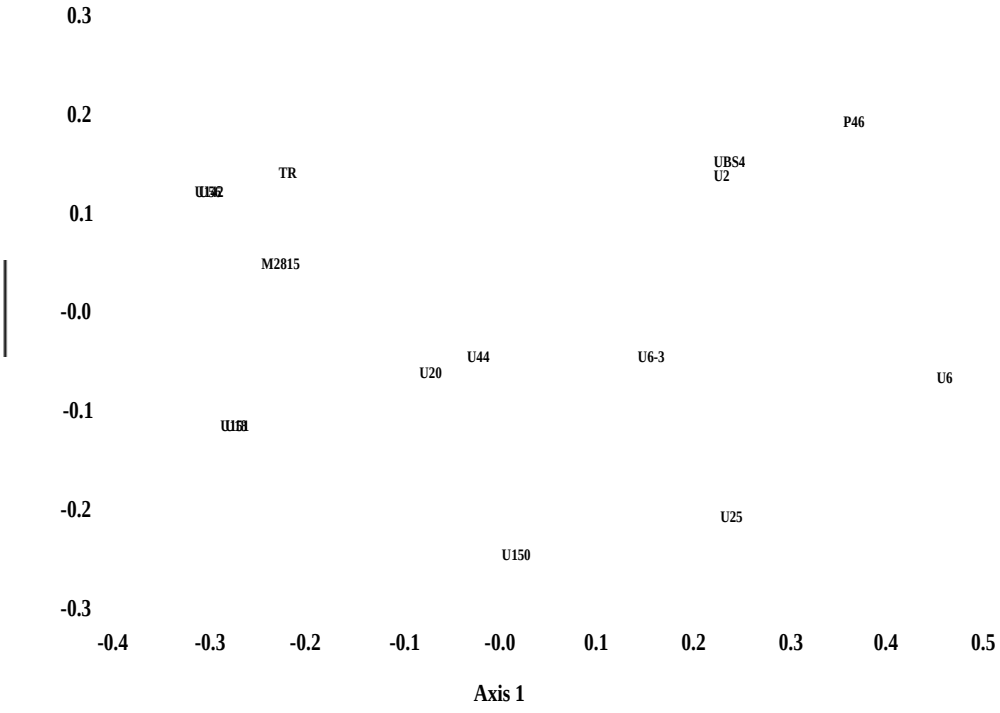




546 units; 32%, 13%, 13%

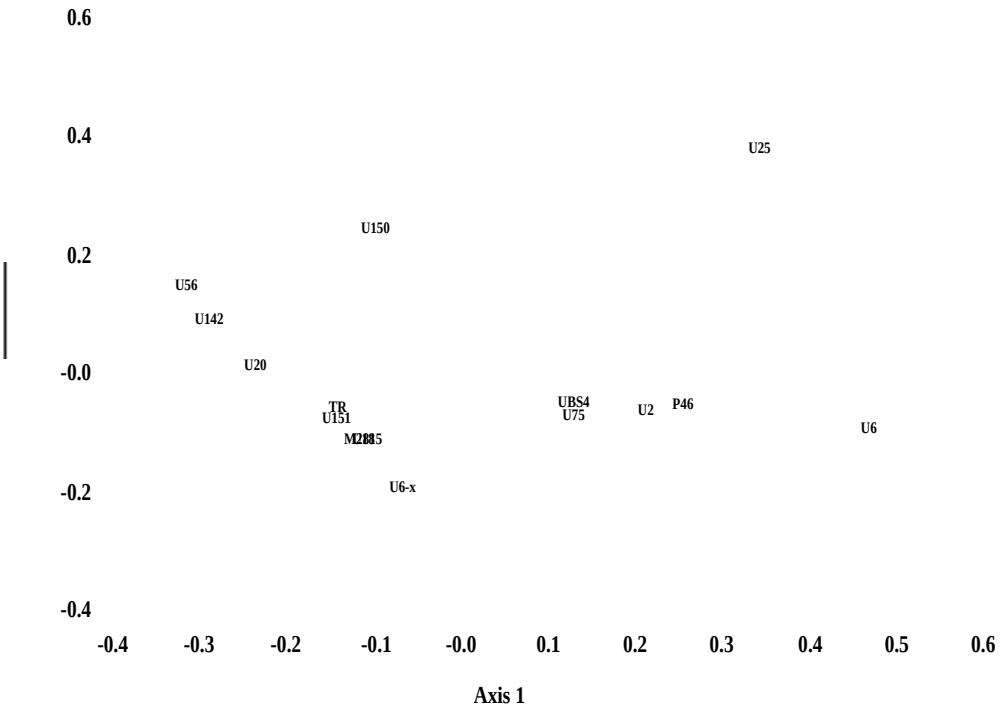
U6-3

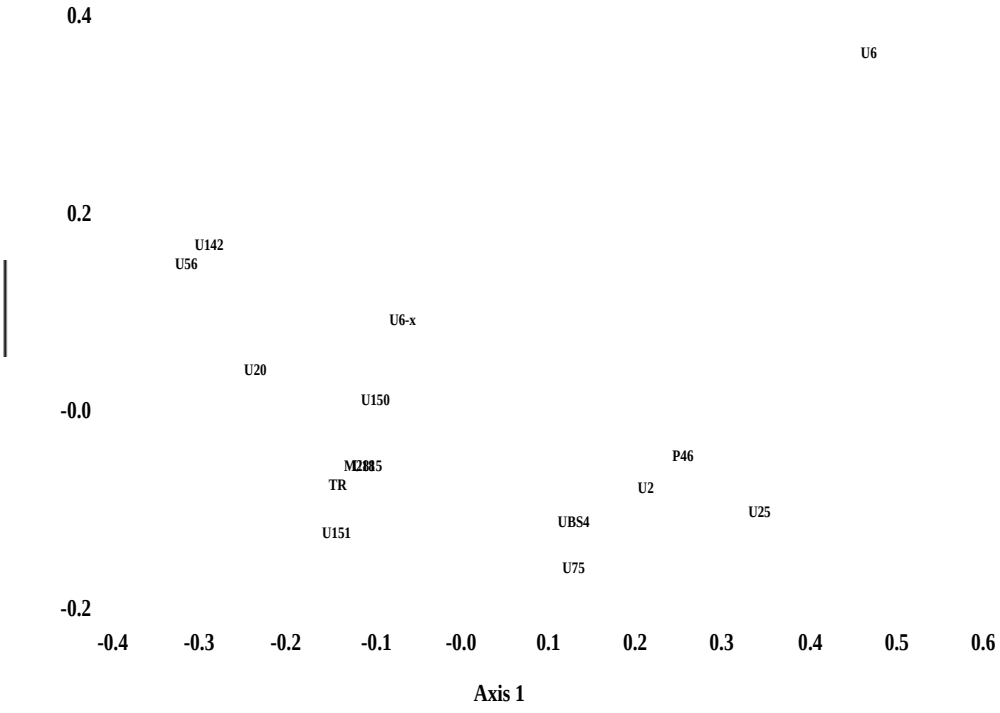




49 units; 39%, 19%, 11%

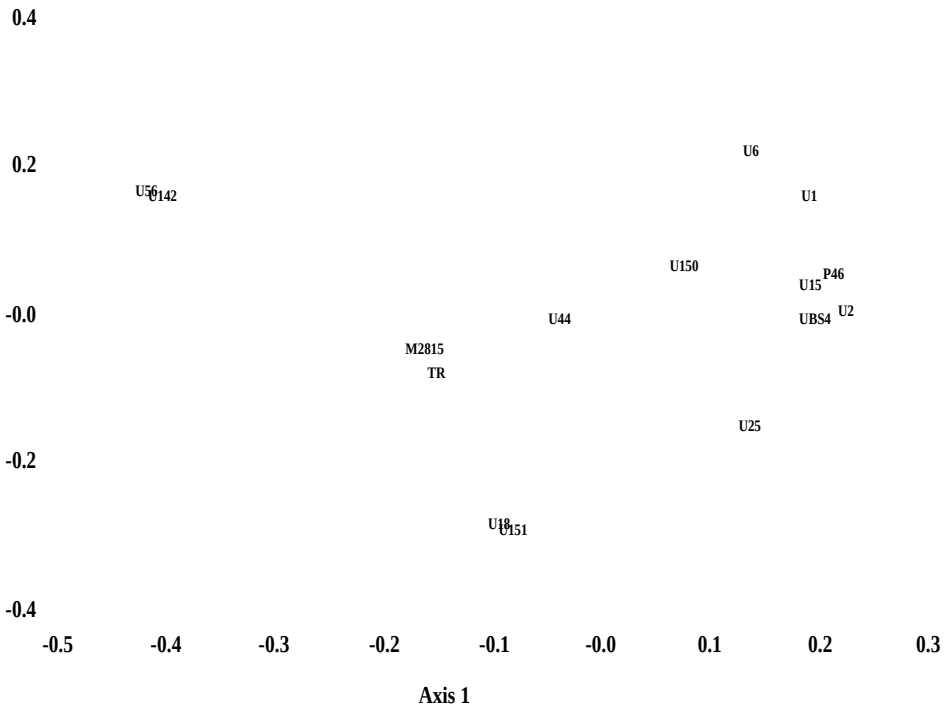
U6-x

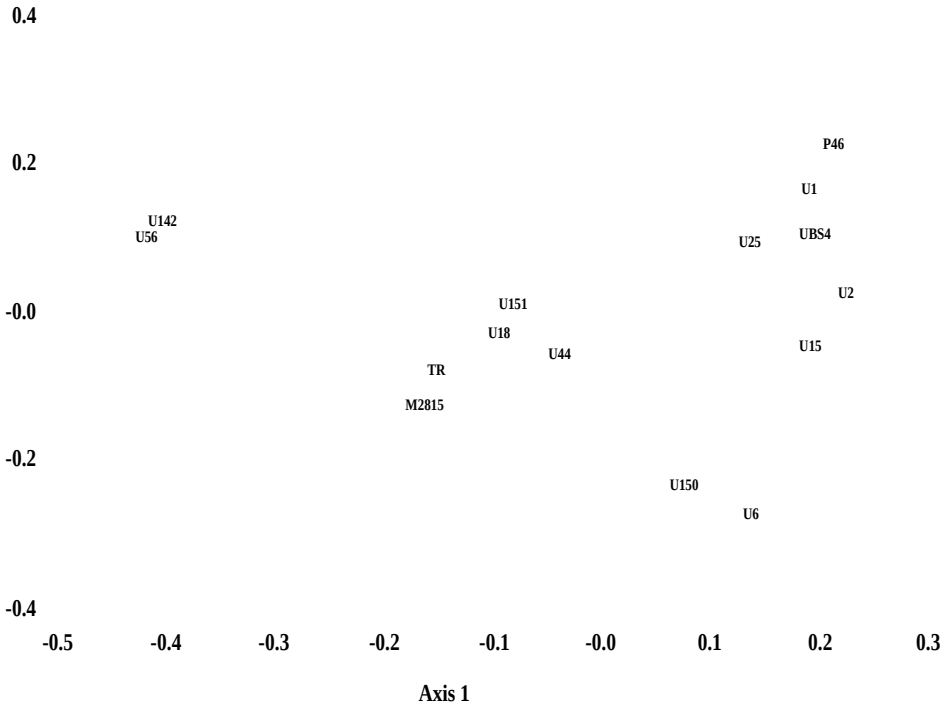




22 units; 36%, 15%, 12%

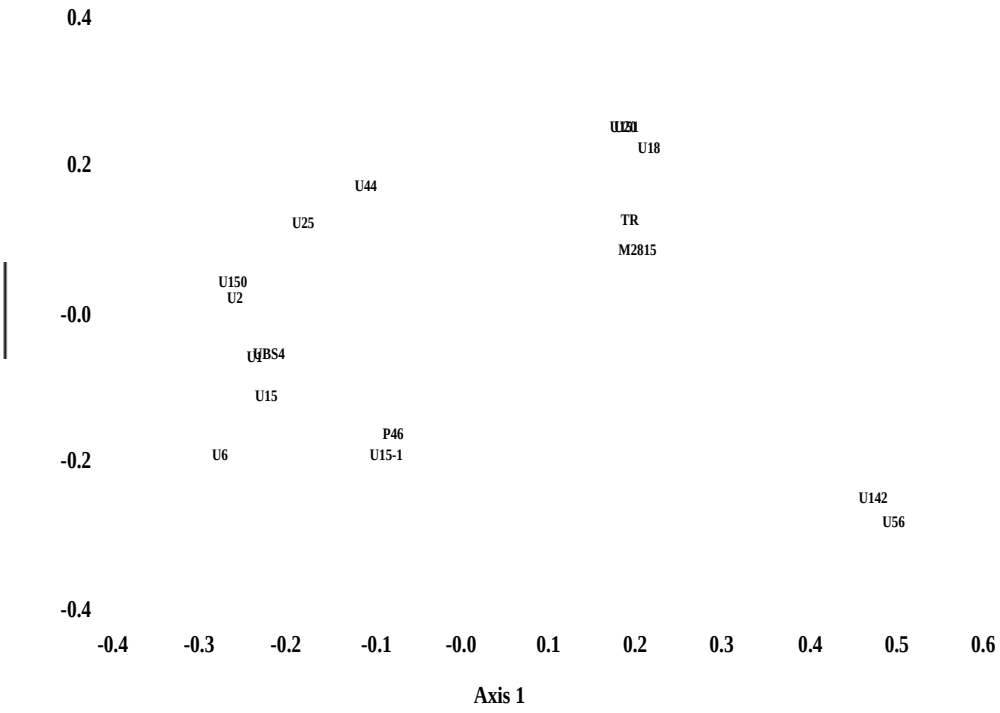
U15 (Caesarea? 500?)

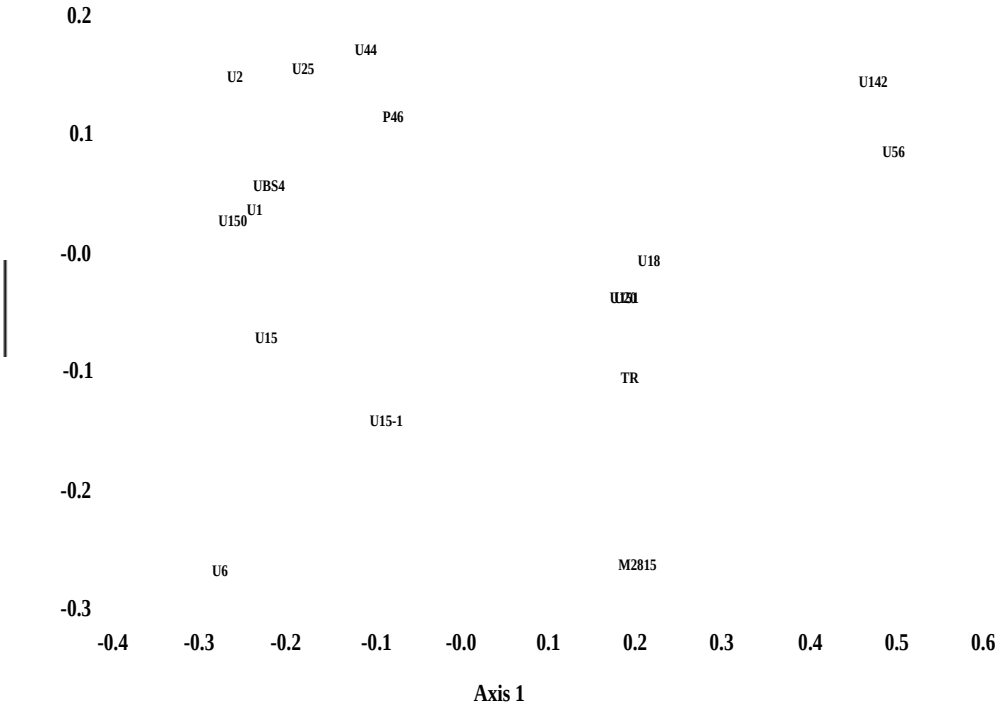




111 units; 27%, 14%, 12%

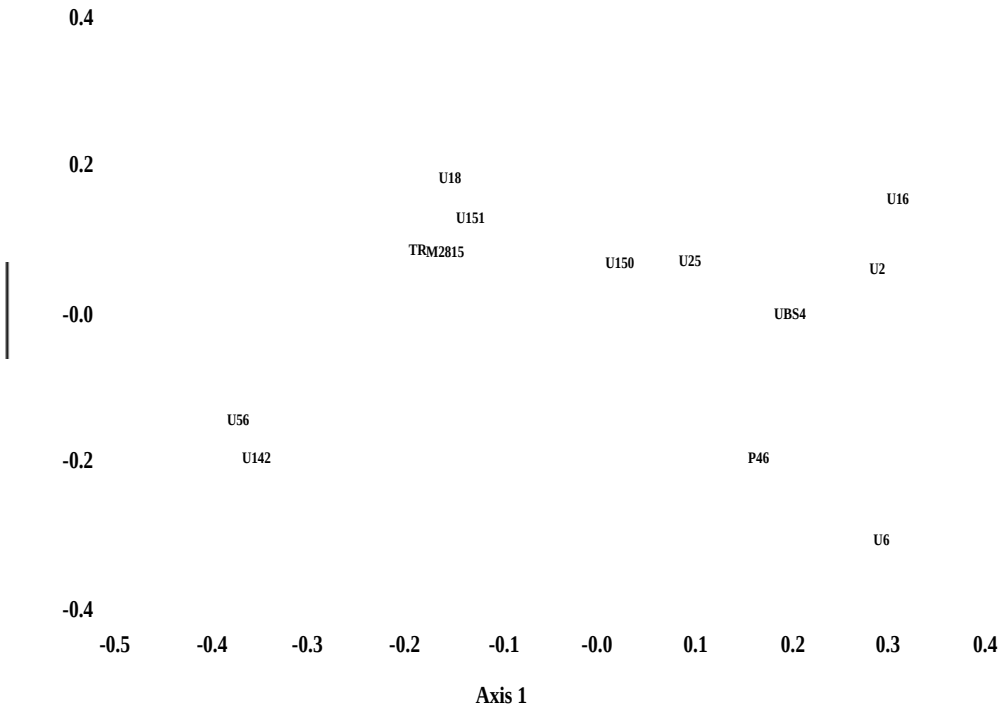
U15-1

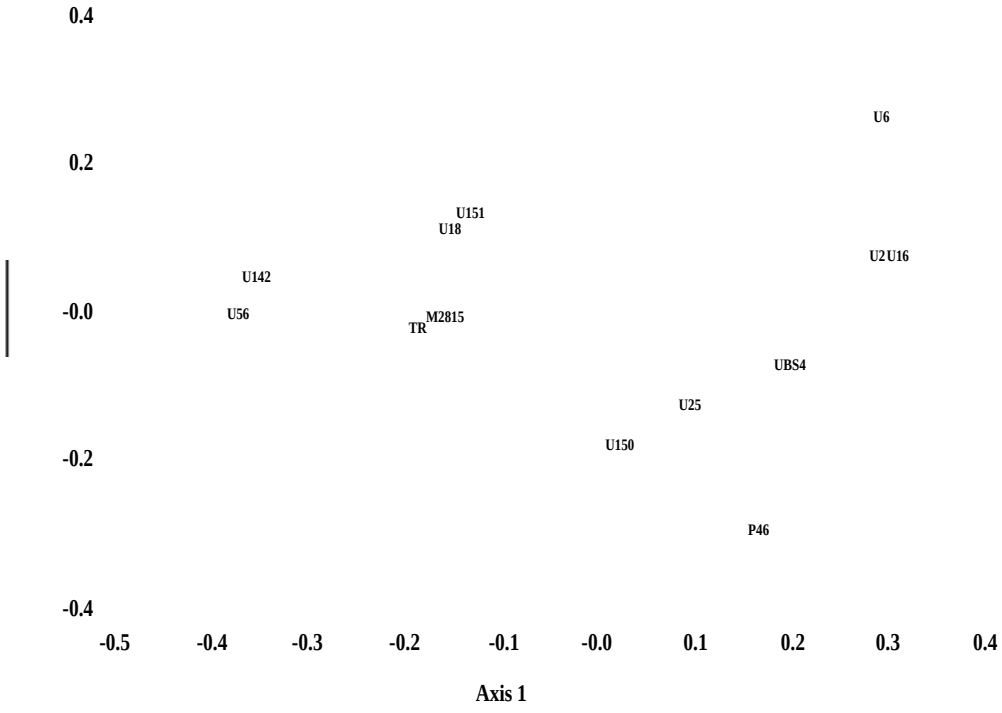




34 units; 39%, 18%, 11%

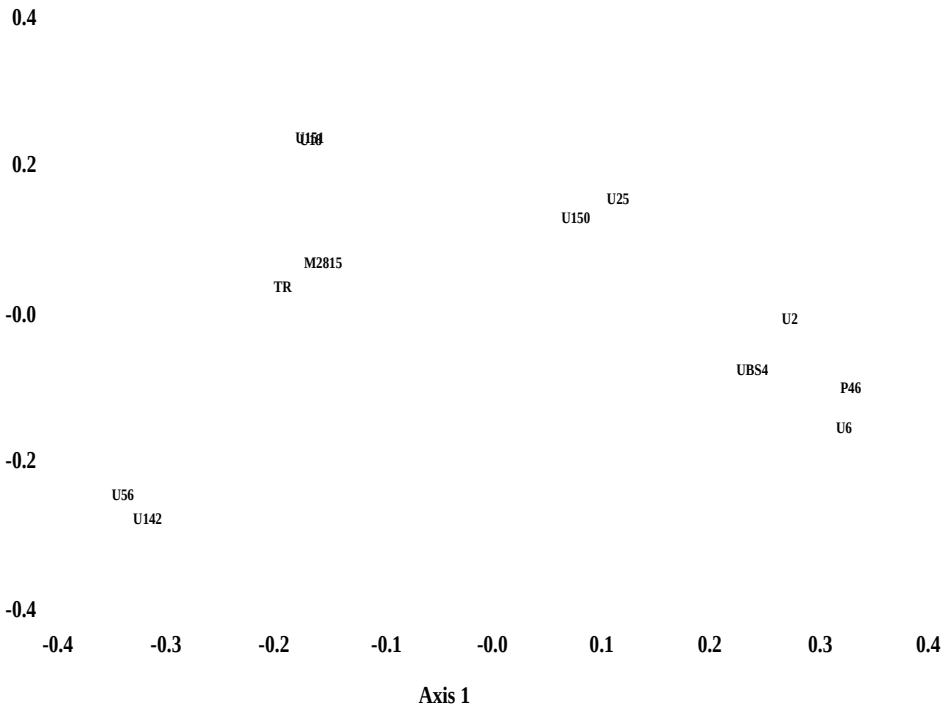
U16 (Northern Egypt? 450?)

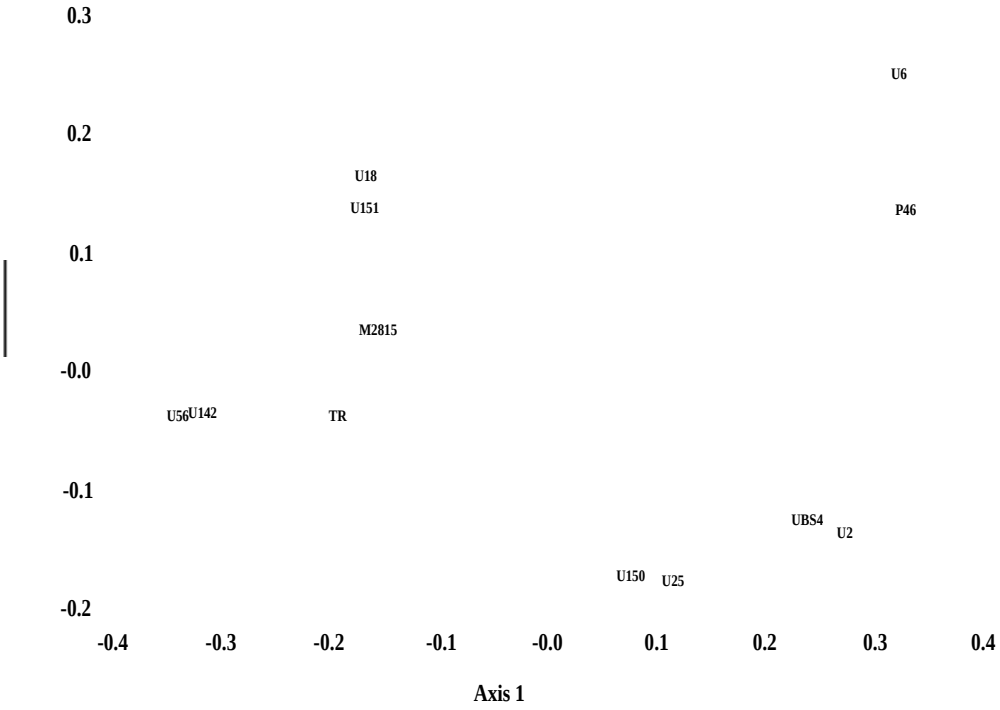




188 units; 31%, 13%, 11%

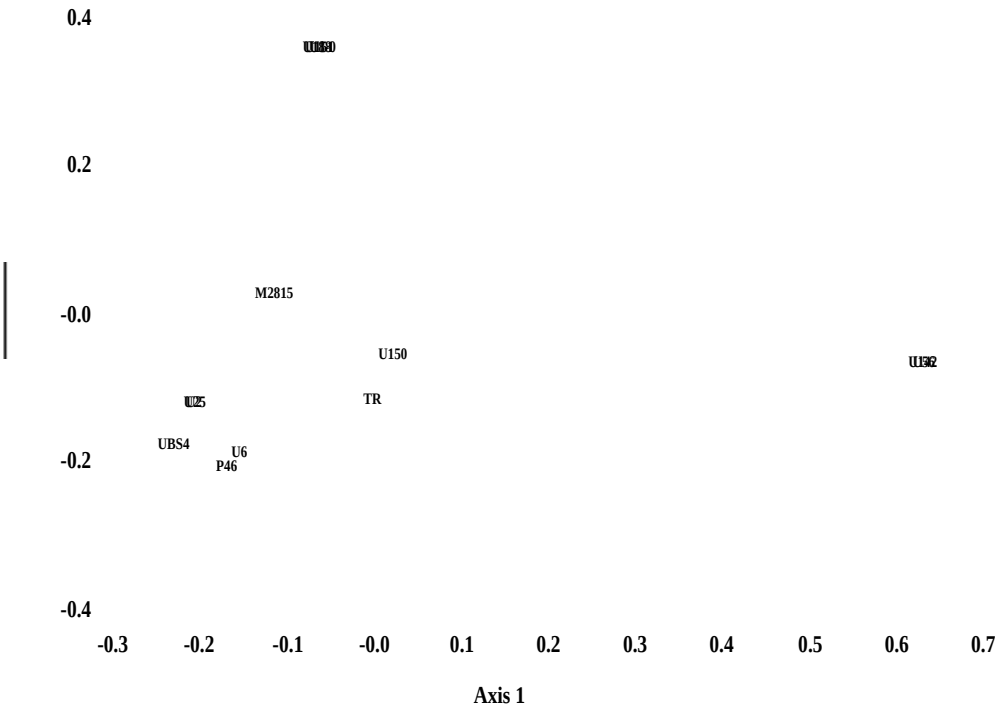
U18 (850?)

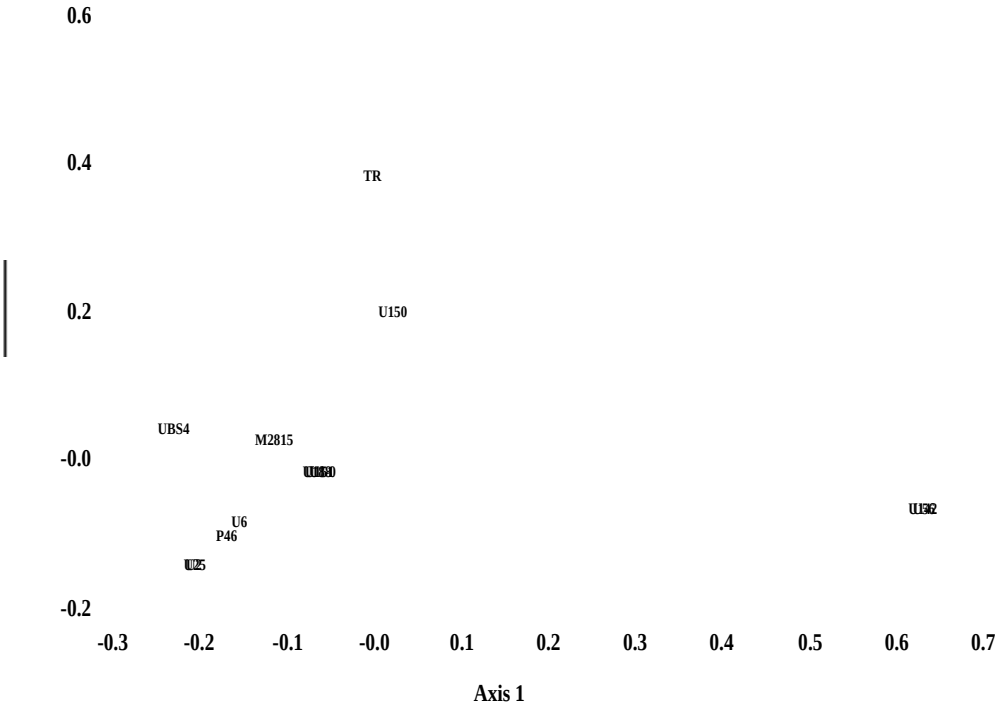




728 units; 32%, 15%, 10%

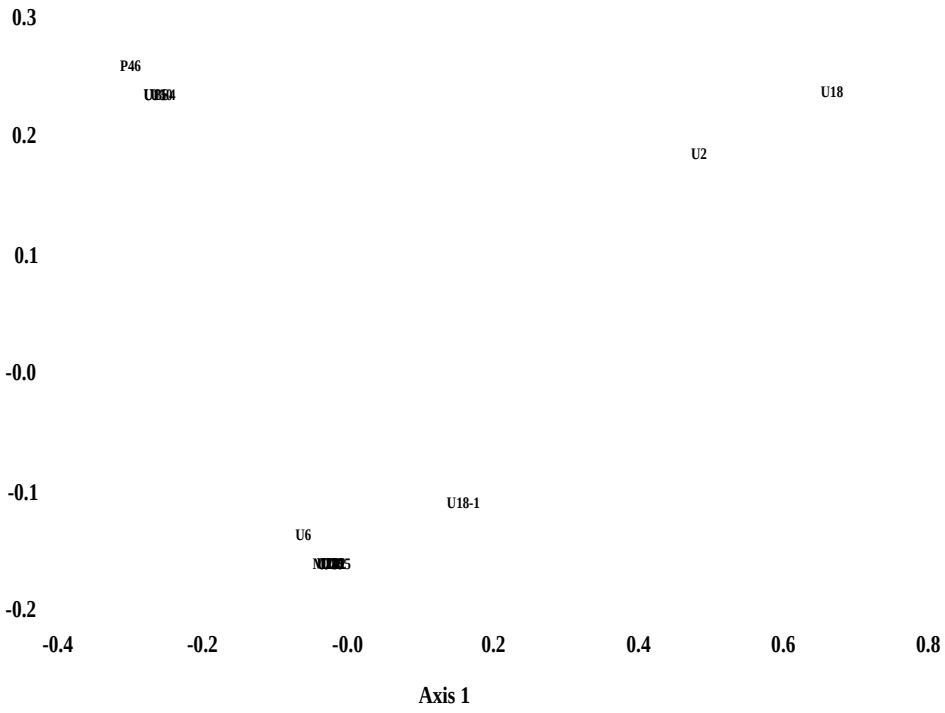
U18-0

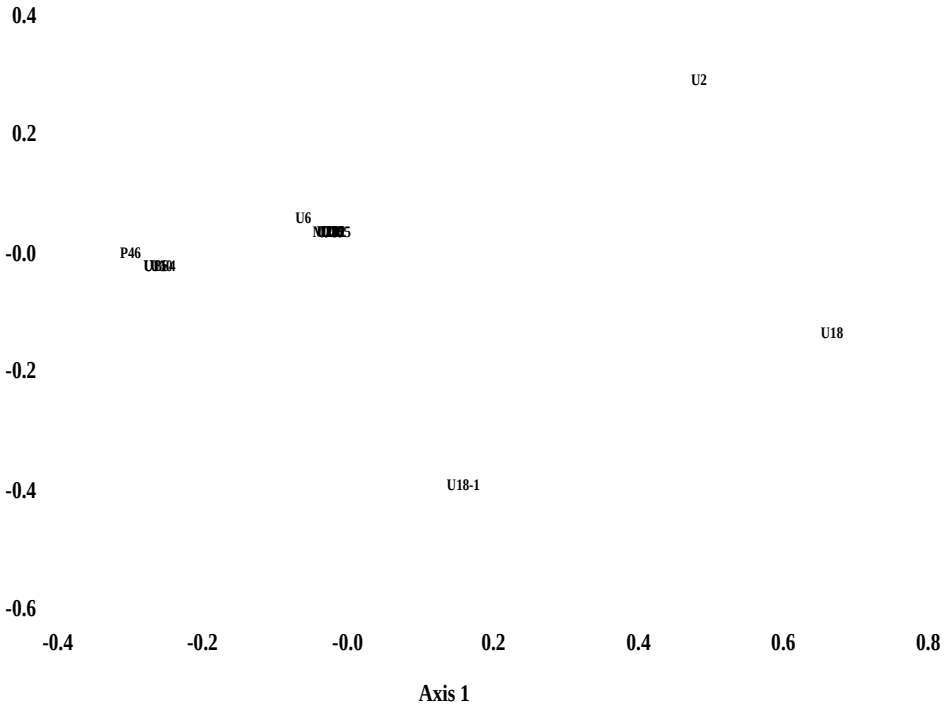




18 units; 48%, 26%, 12%

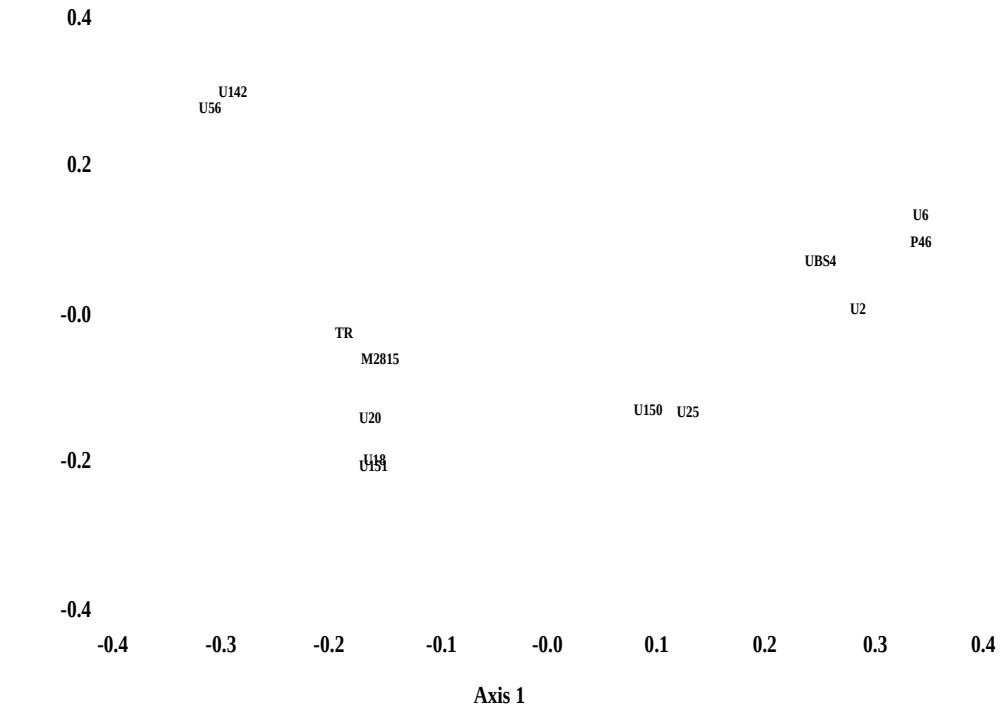
U18-1

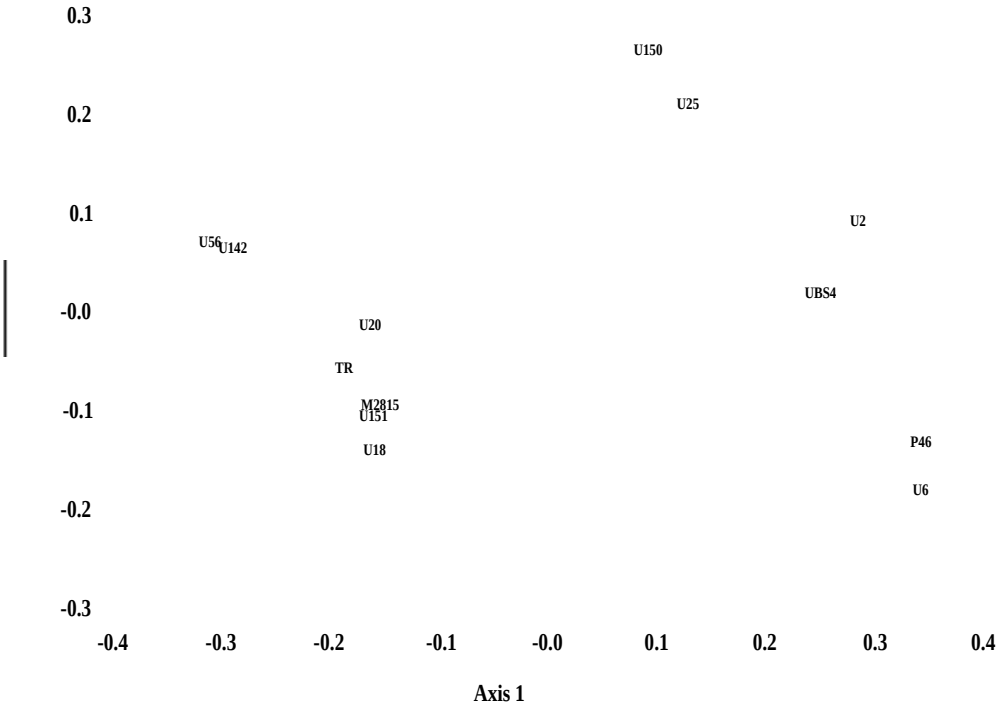




9 units; 50%, 27%, 14%

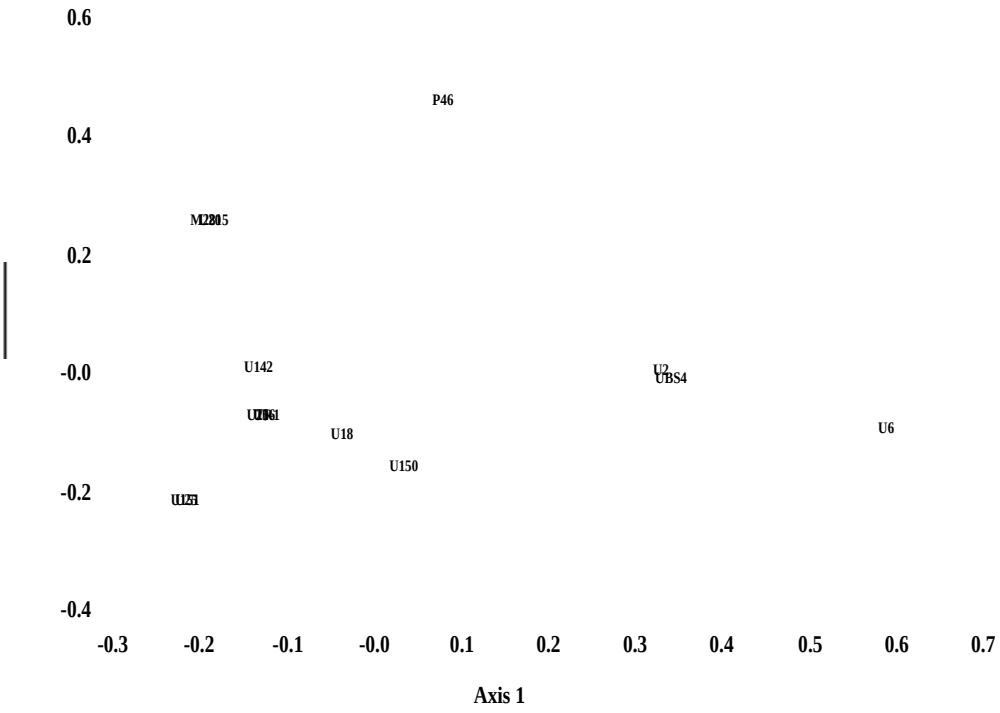
U20 (850?)

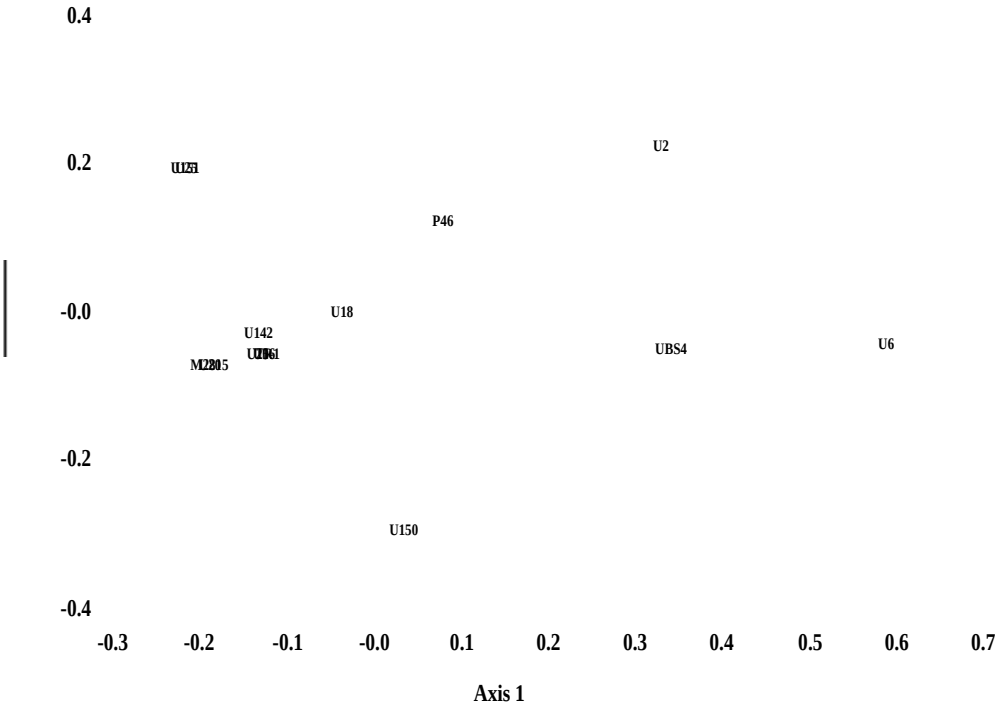




662 units; 31%, 15%, 10%

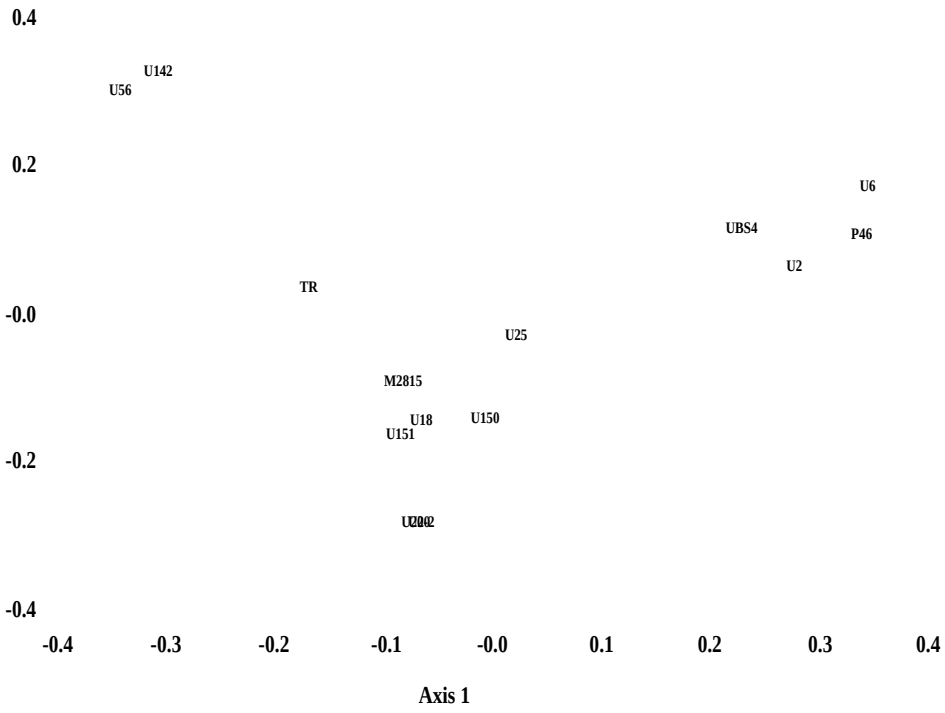
U20-1

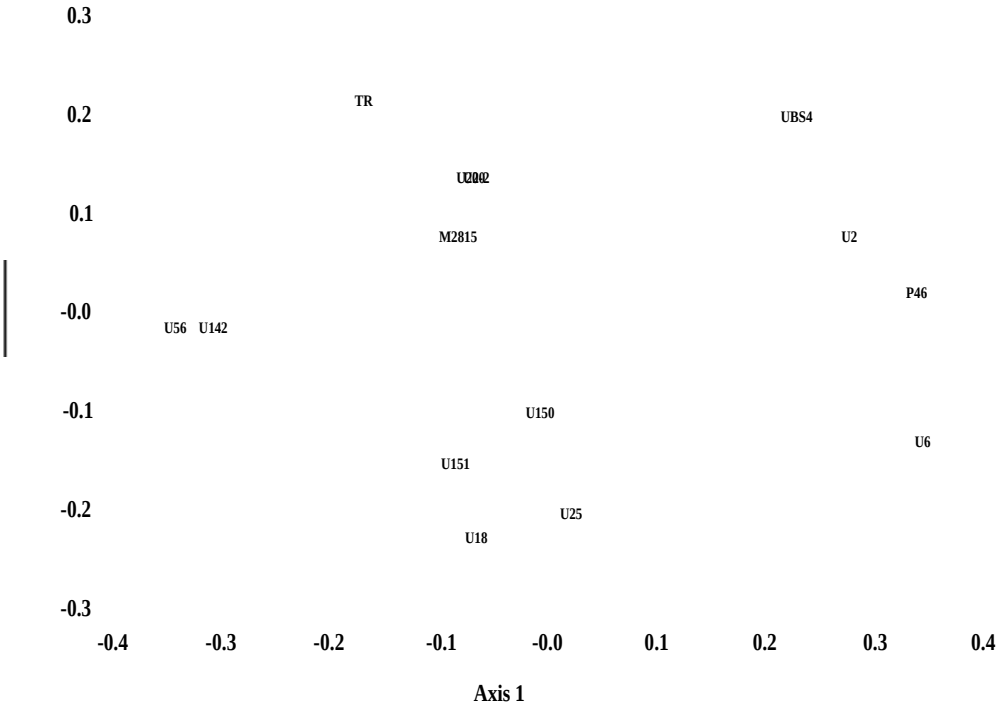




12 units; 38%, 23%, 12%

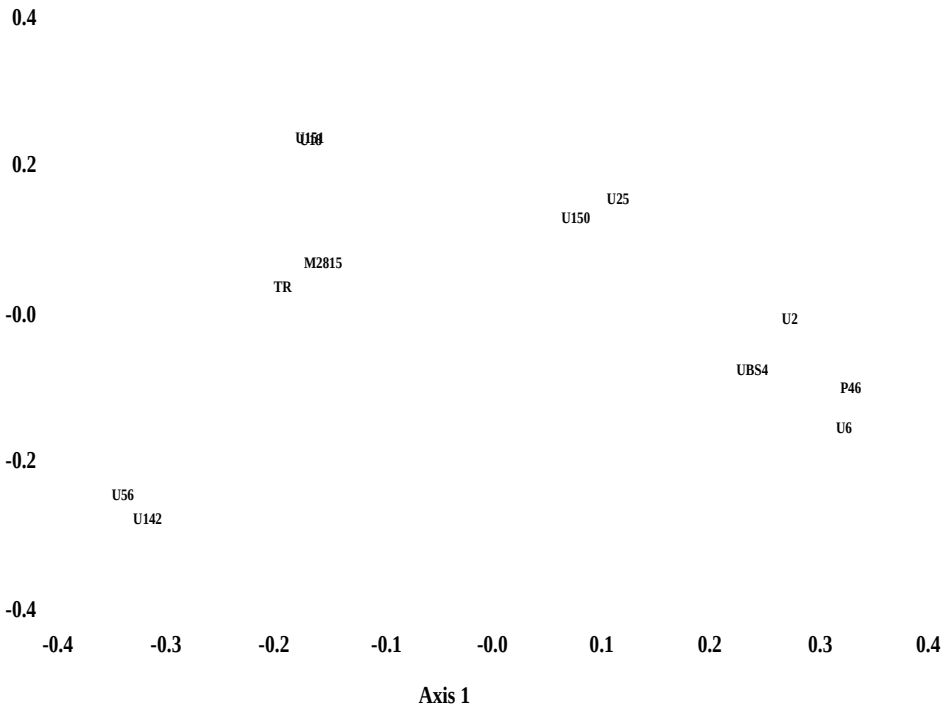
U20-2

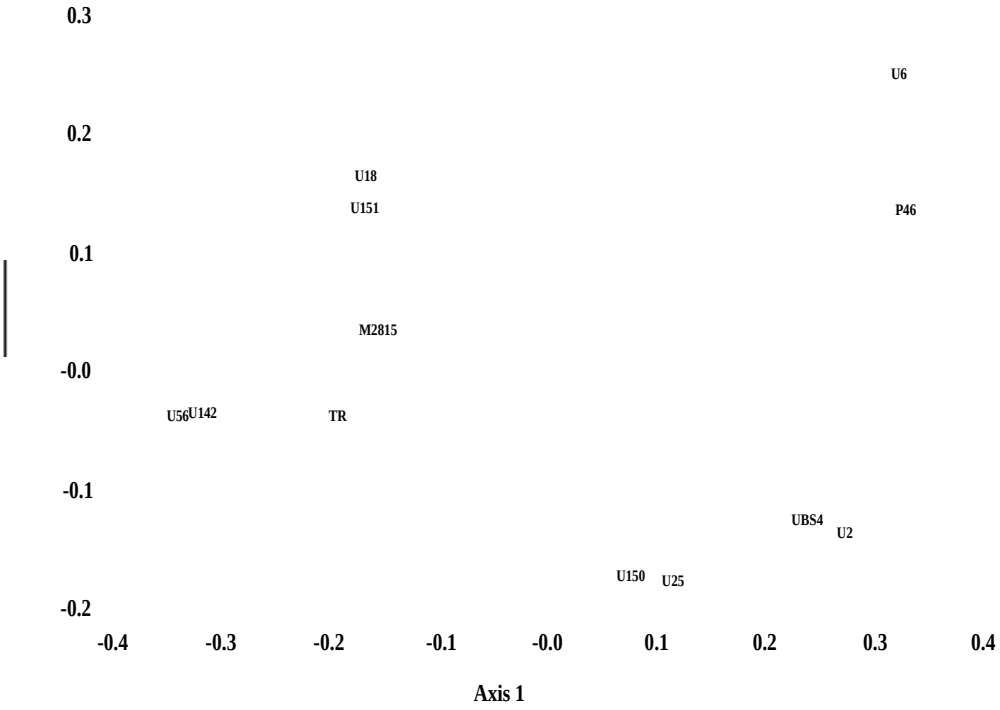




141 units; 26%, 20%, 12%

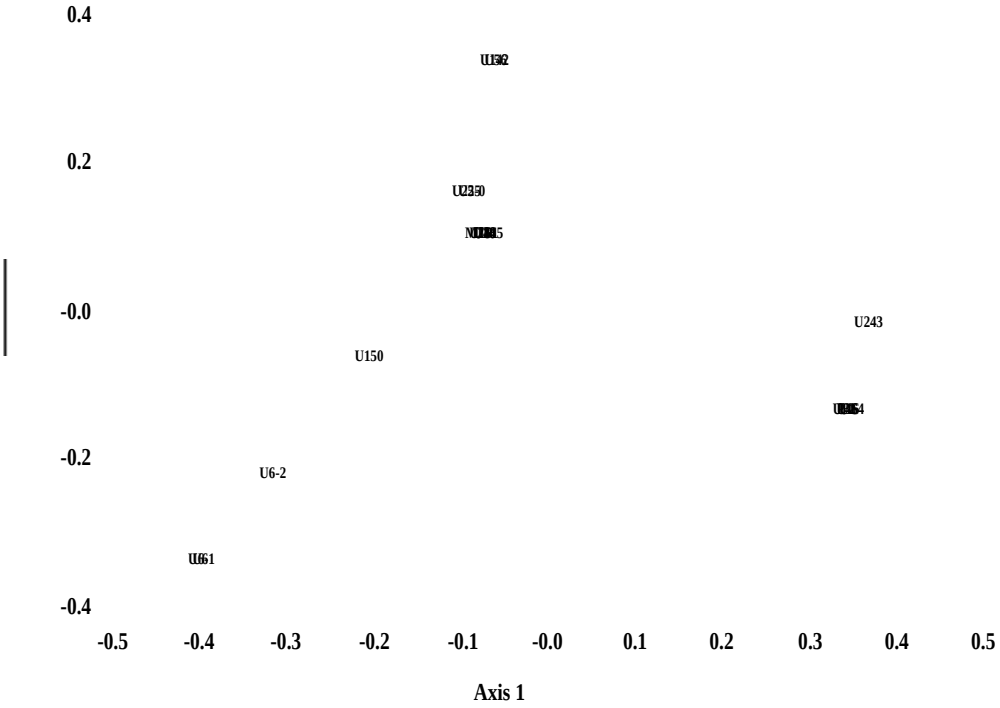
U25 (850?)

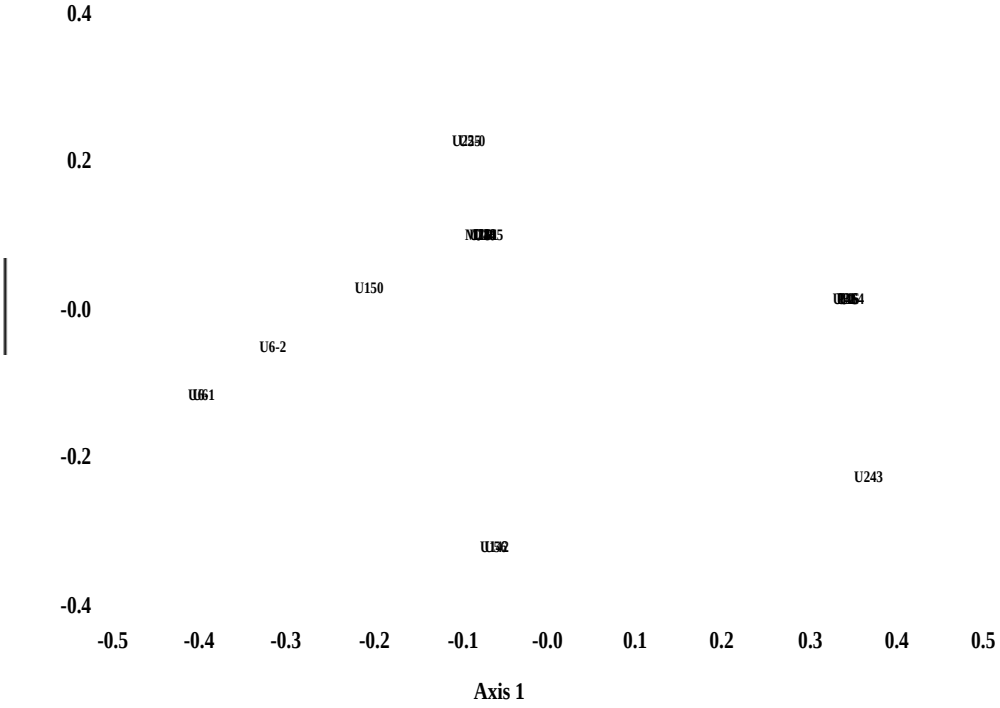




728 units; 32%, 15%, 10%

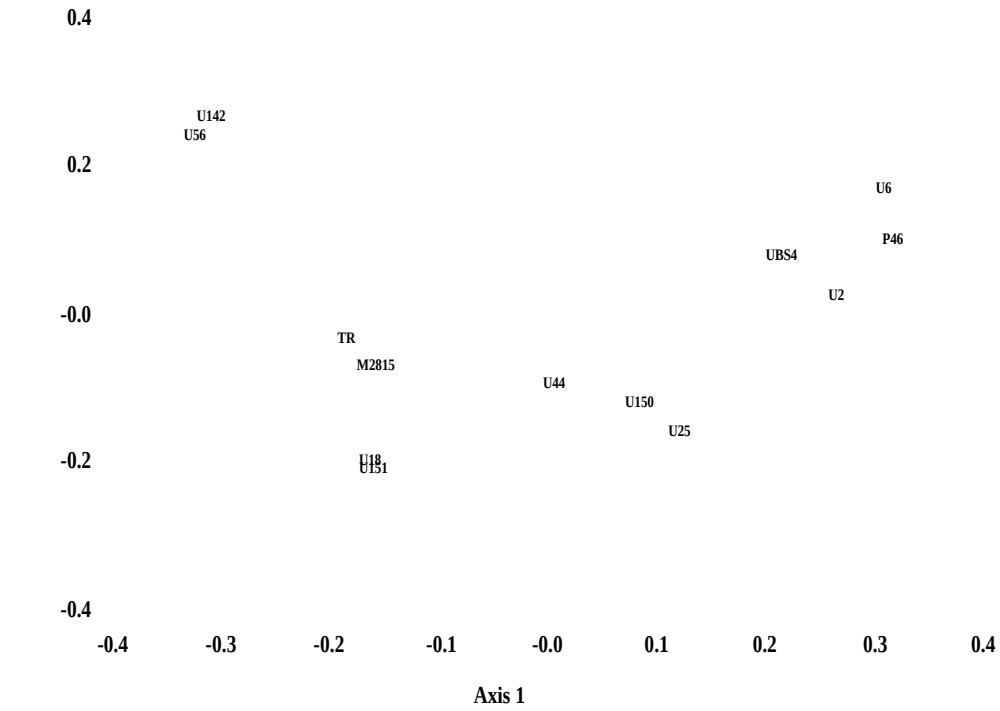
U25-0

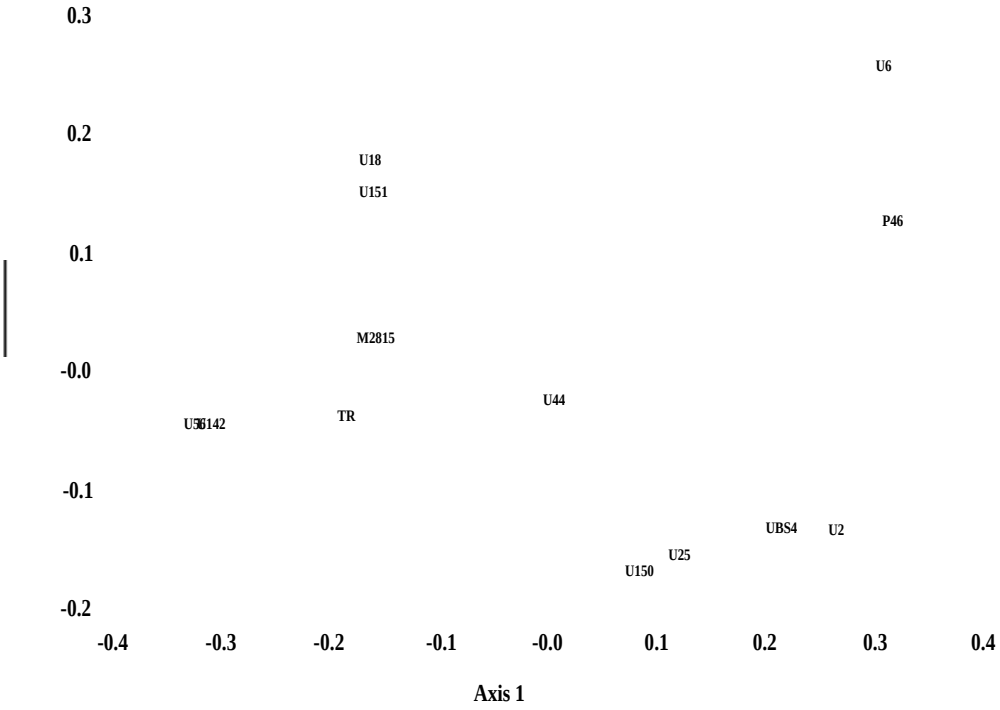




8 units; 45%, 26%, 16%

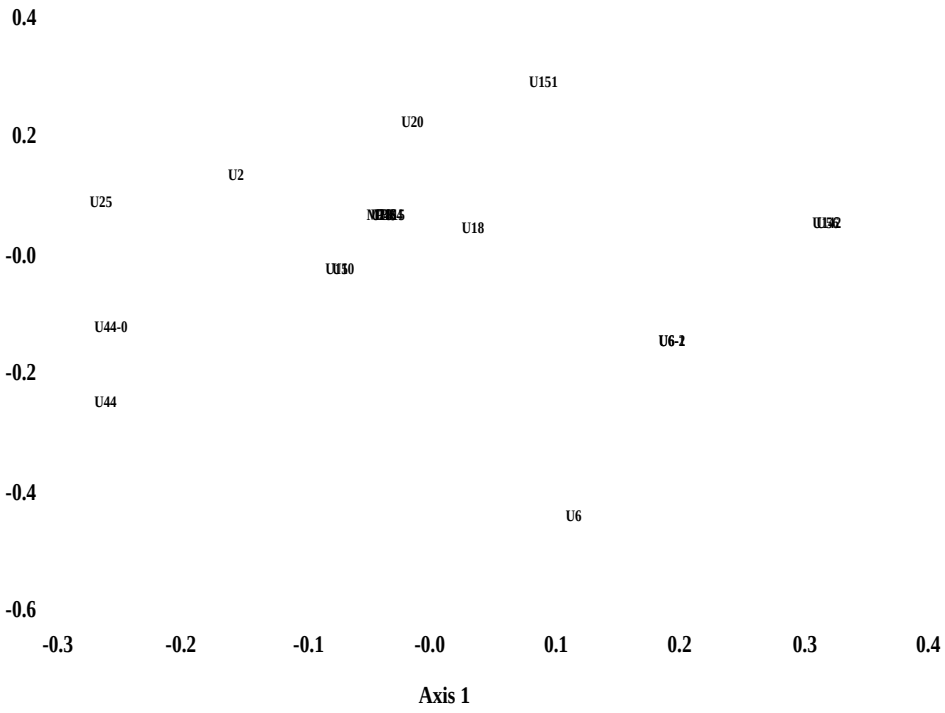
U44 (800?)

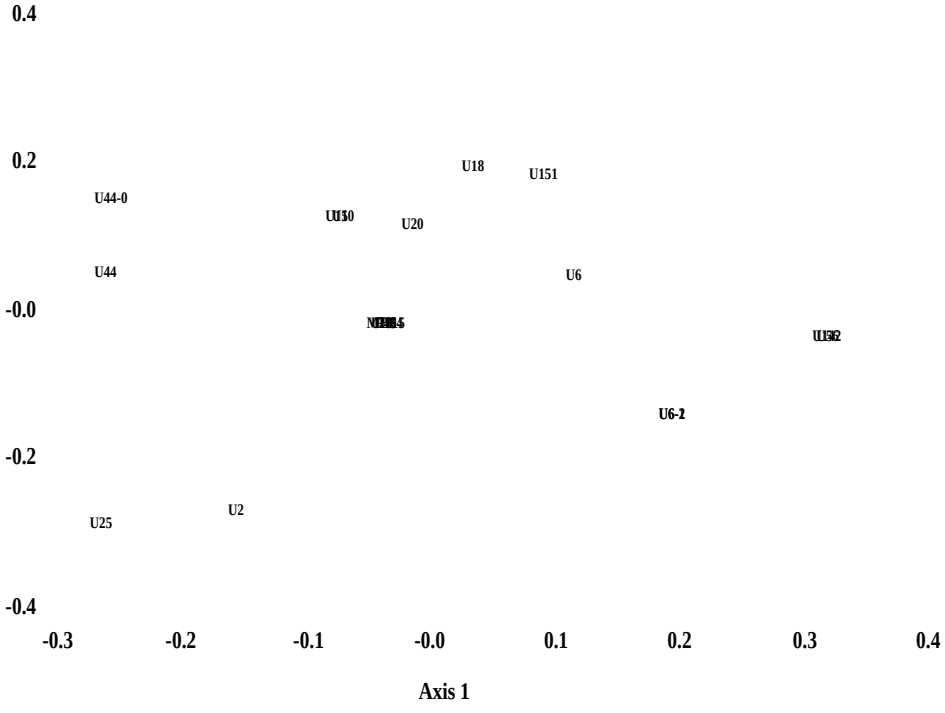




683 units; 28%, 14%, 10%

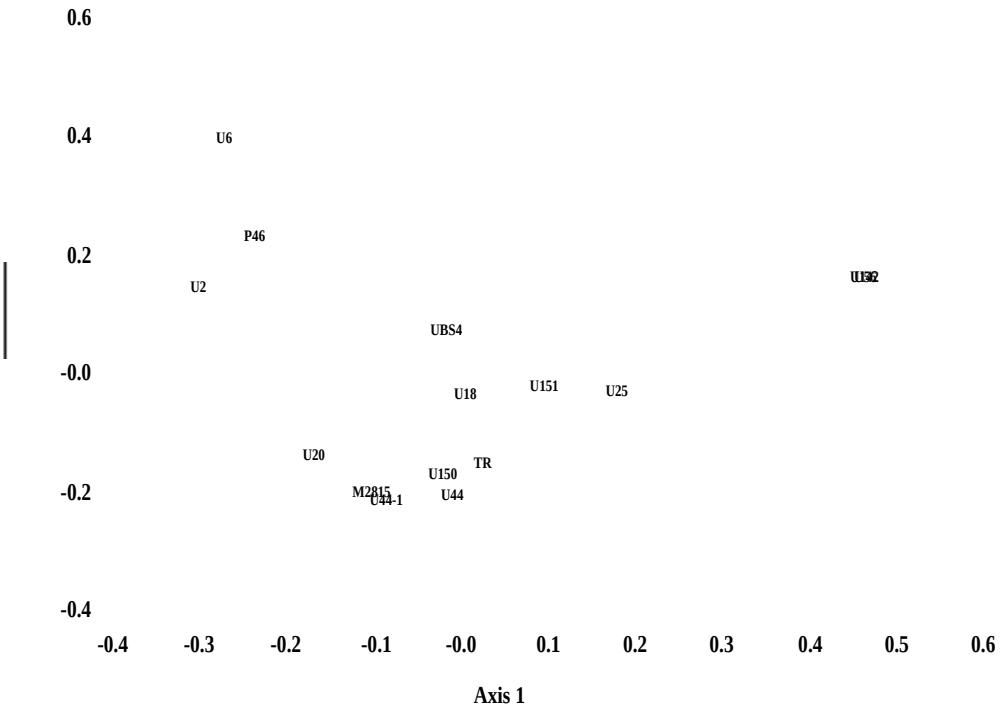
U44-0

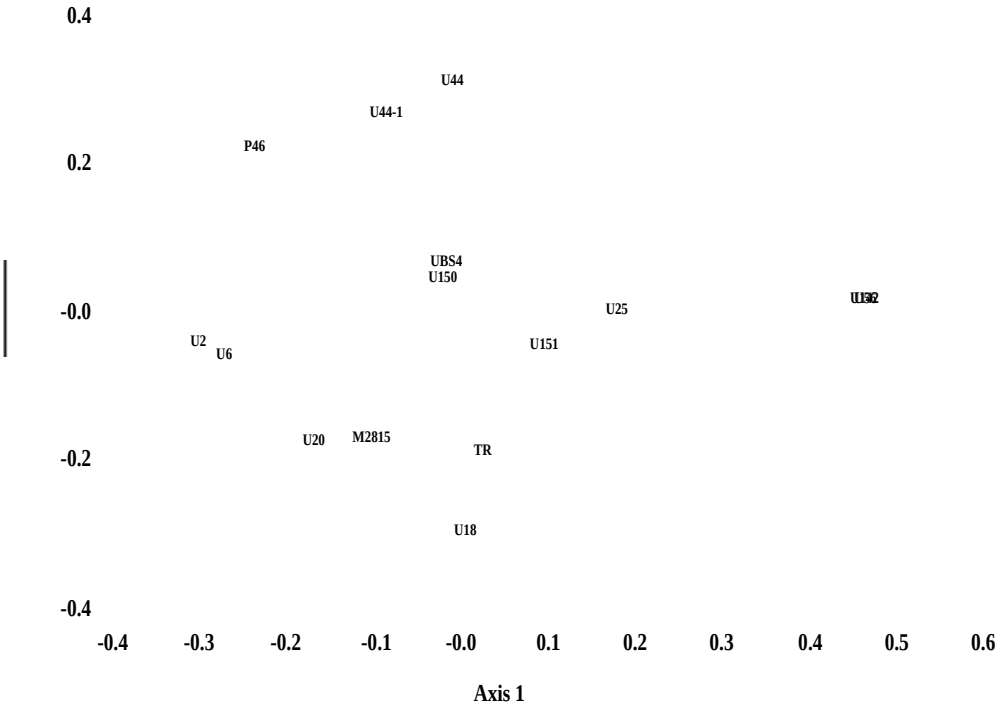




13 units; 24%, 22%, 15%

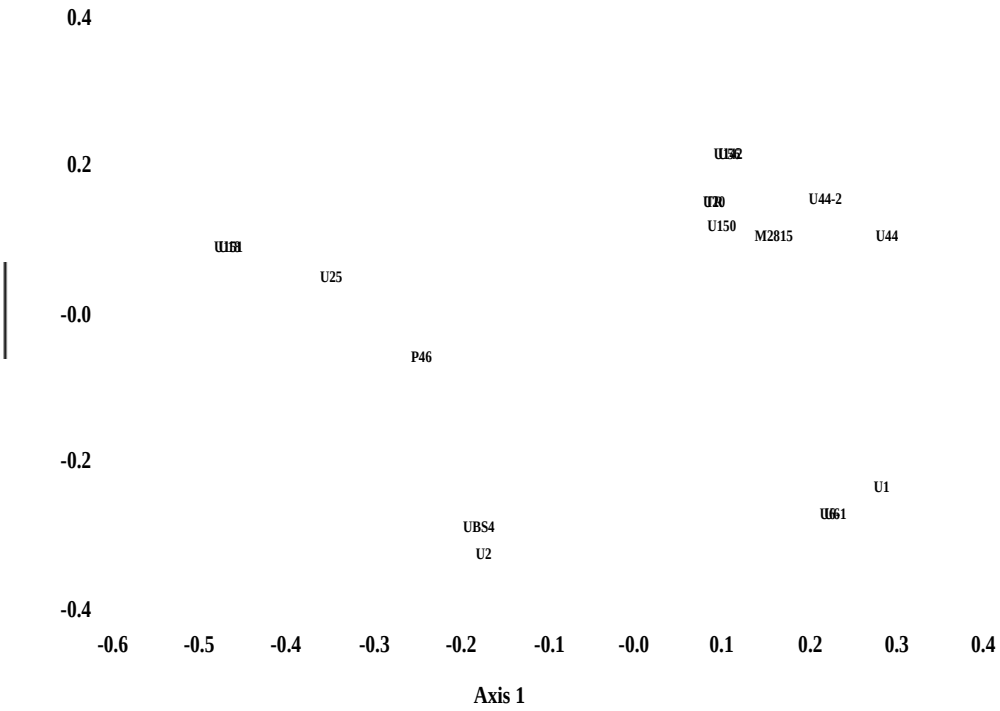
U44-1

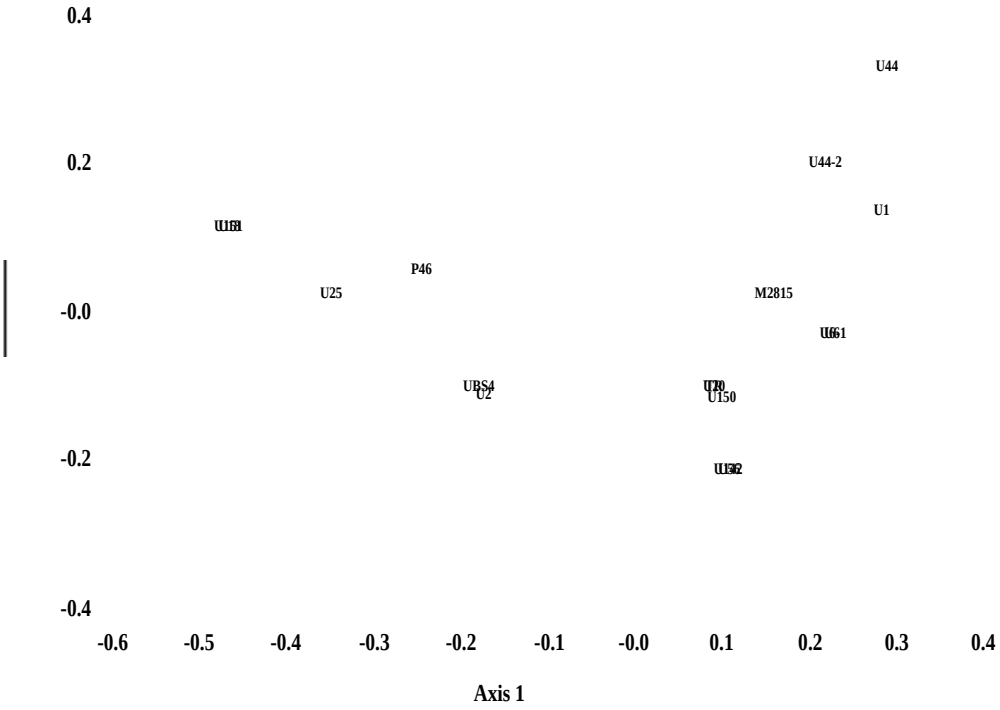




20 units; 29%, 19%, 17%

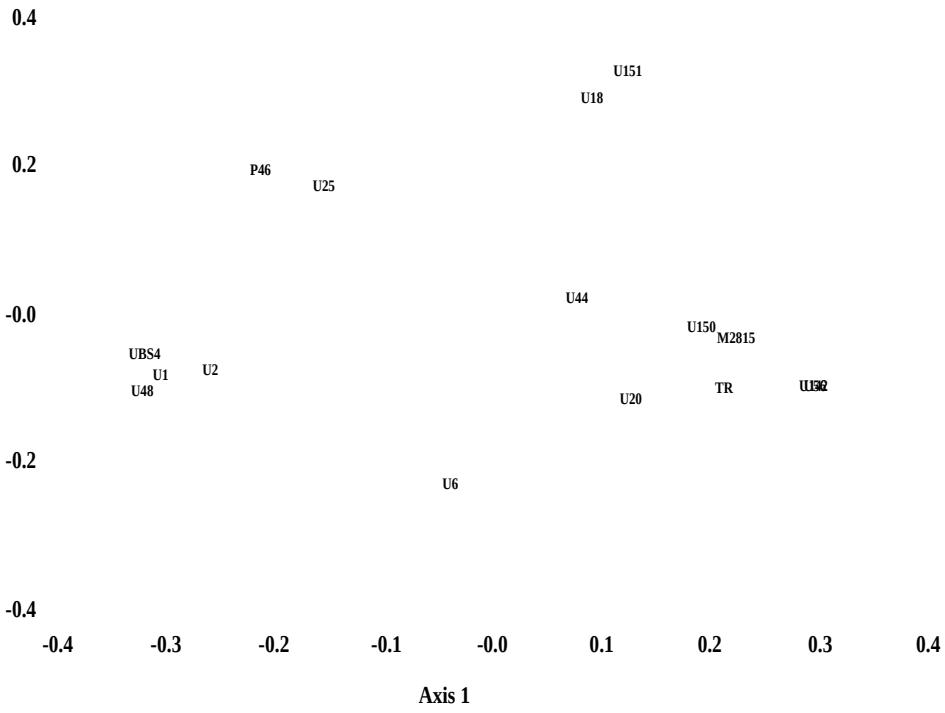
U44-2

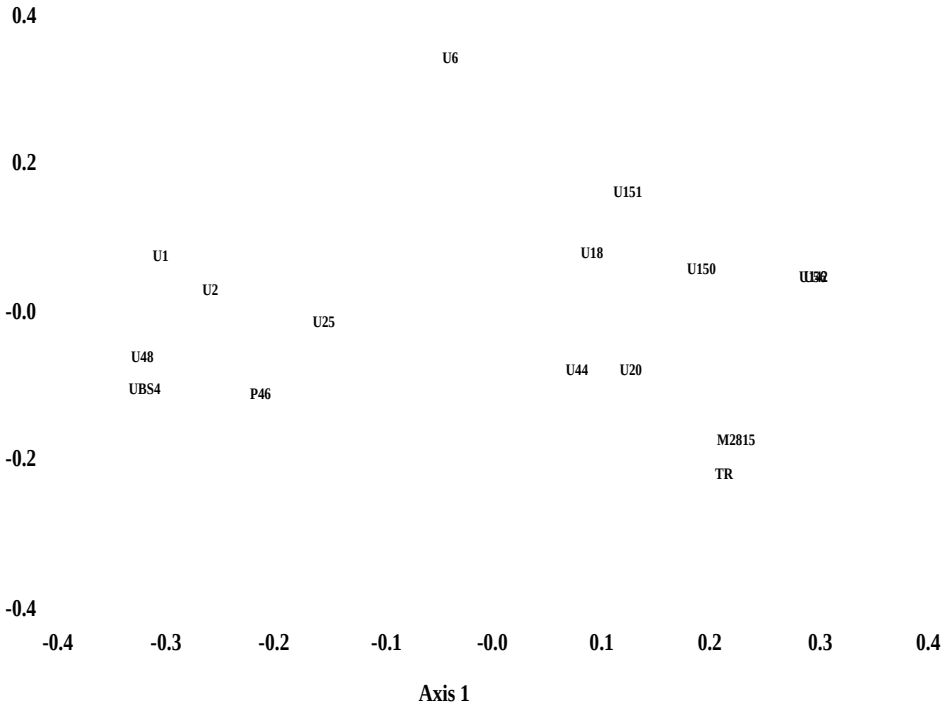




22 units; 38%, 22%, 13%

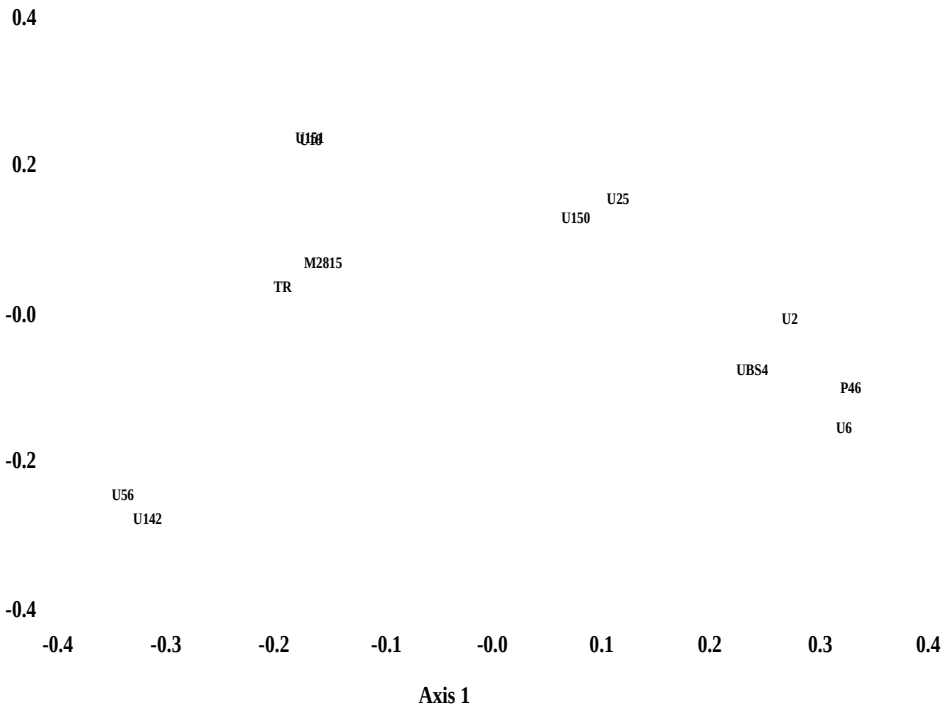
U48 (450?)

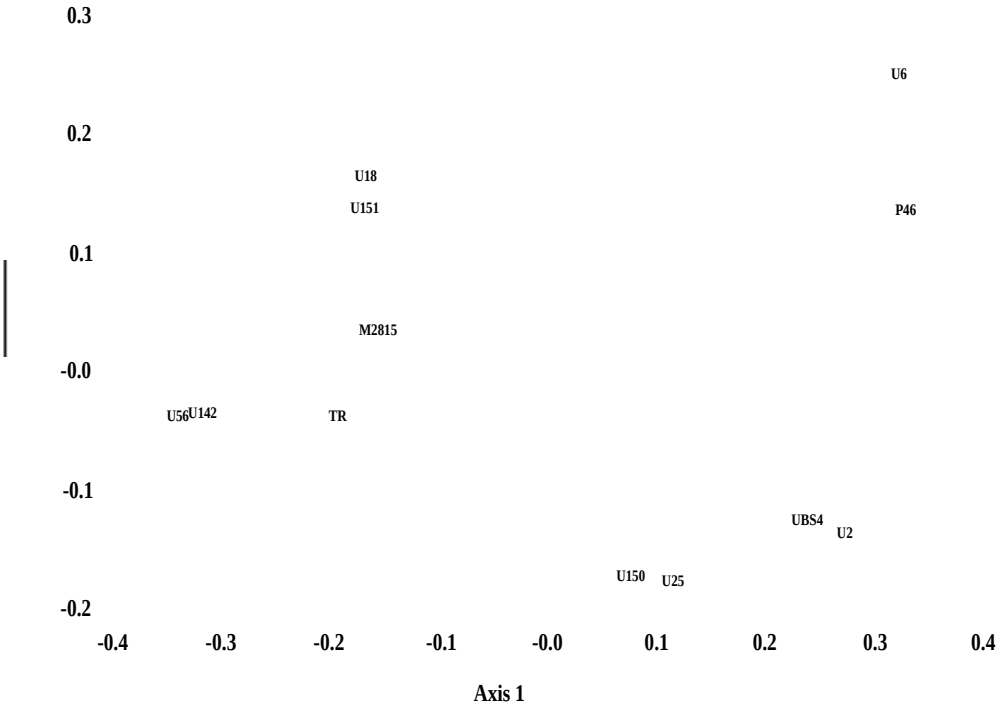




78 units; 30%, 15%, 11%

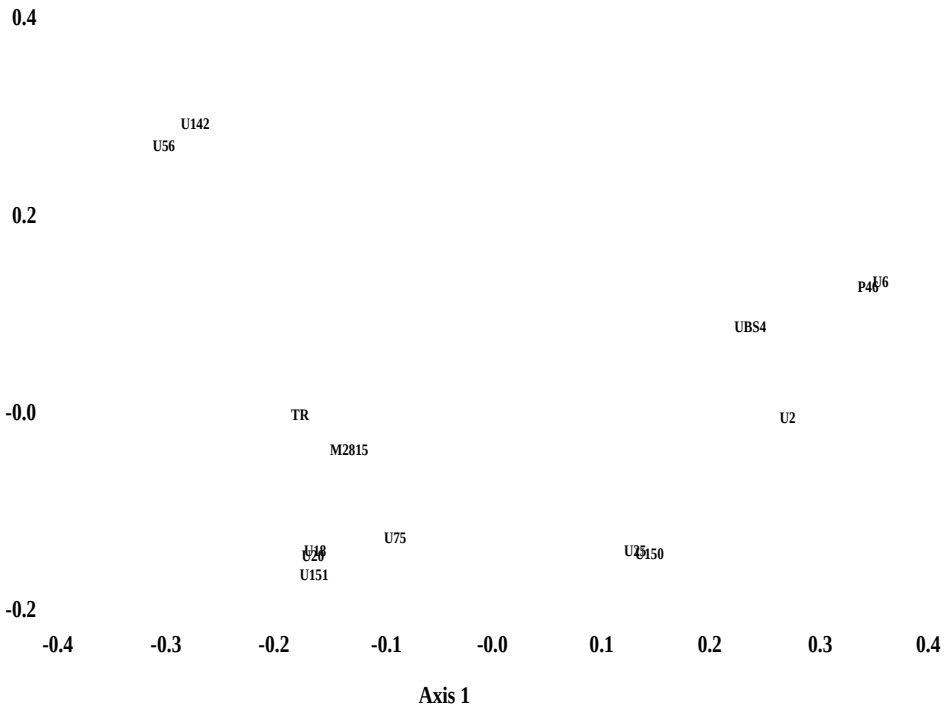
U56 (950?)

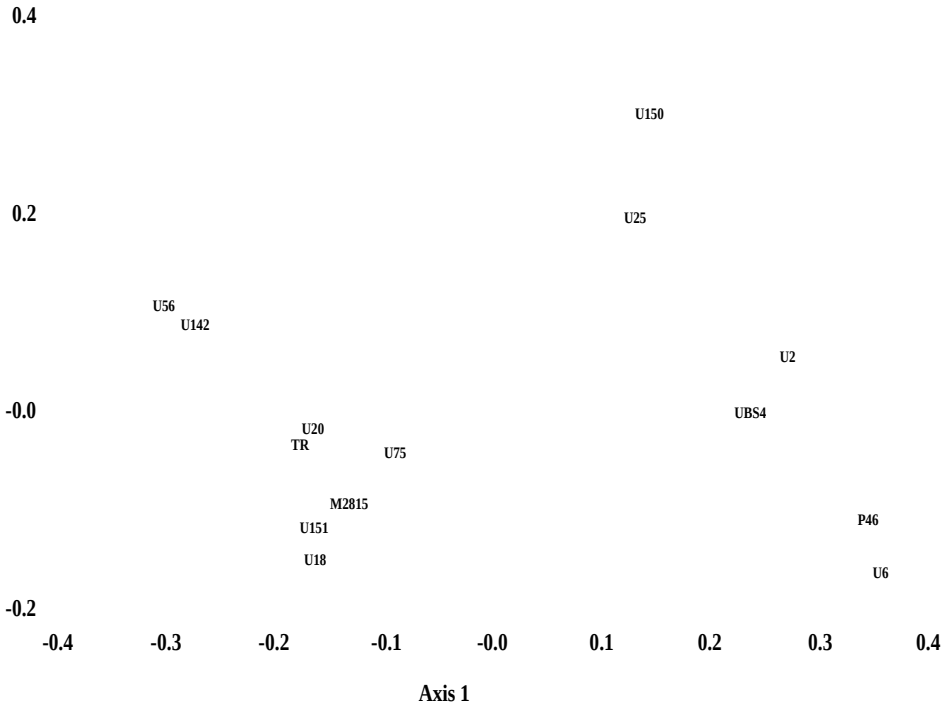




728 units; 32%, 15%, 10%

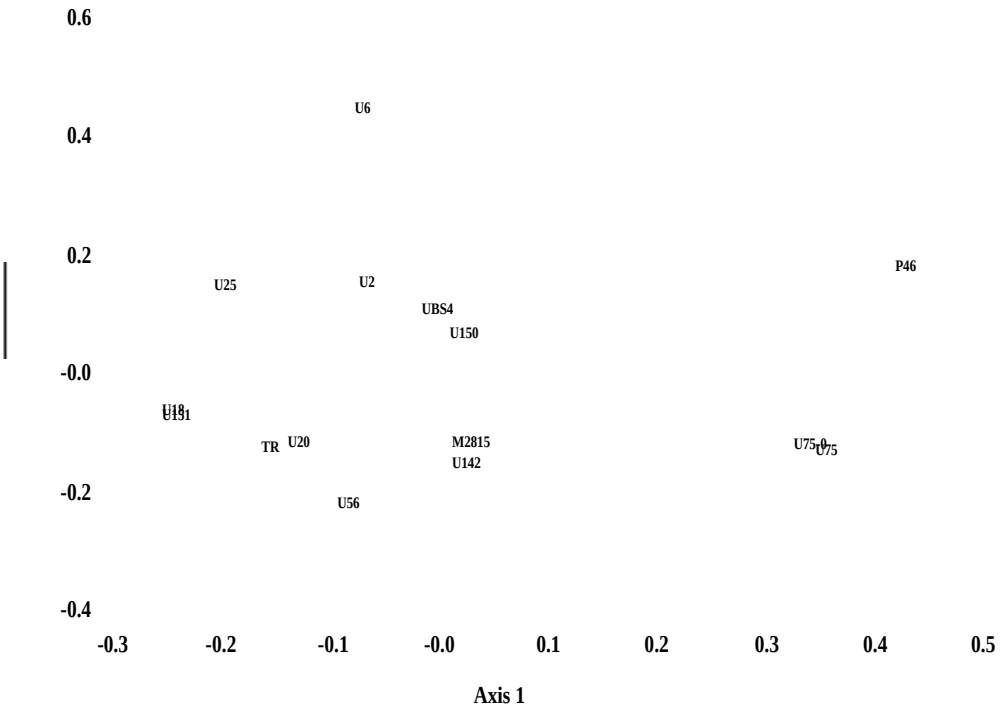
U75 (950?)

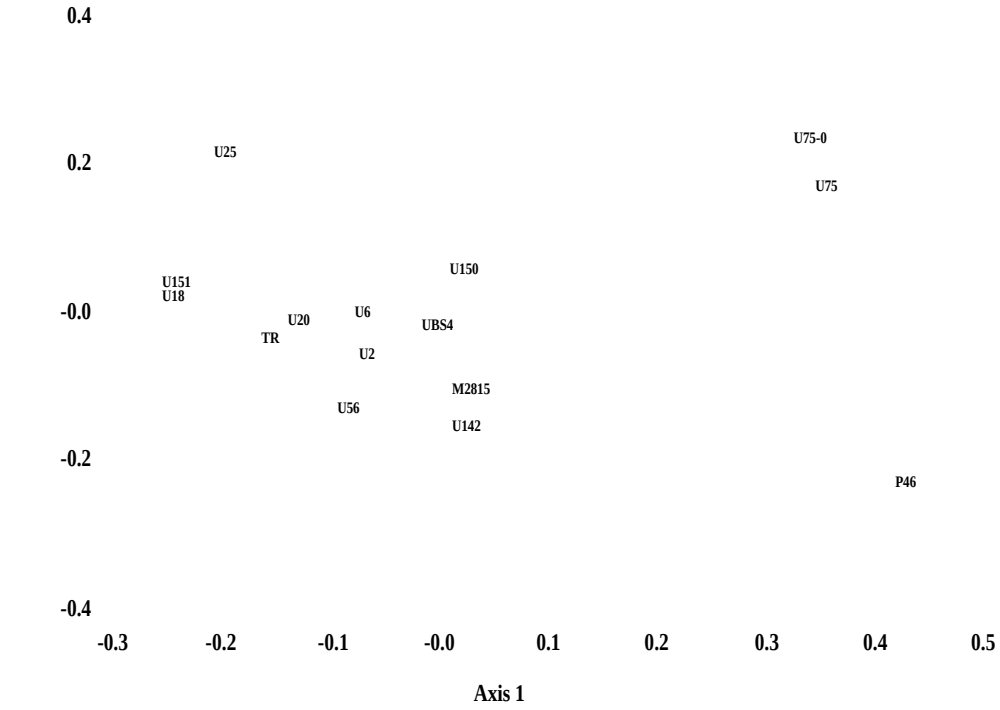




501 units; 30%, 14%, 10%

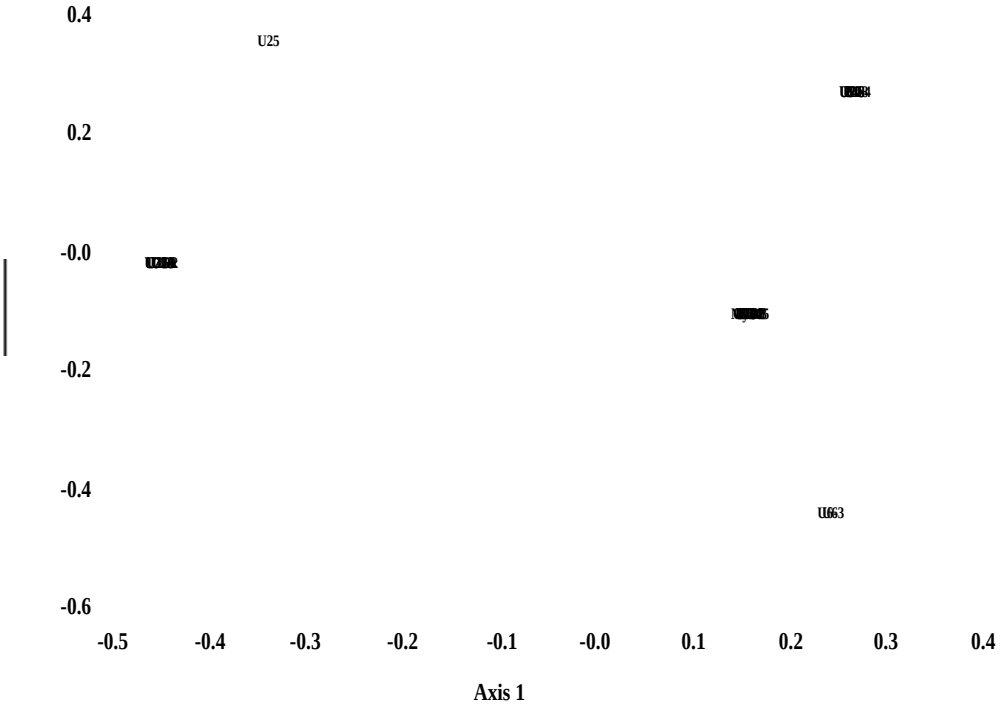
U75-0

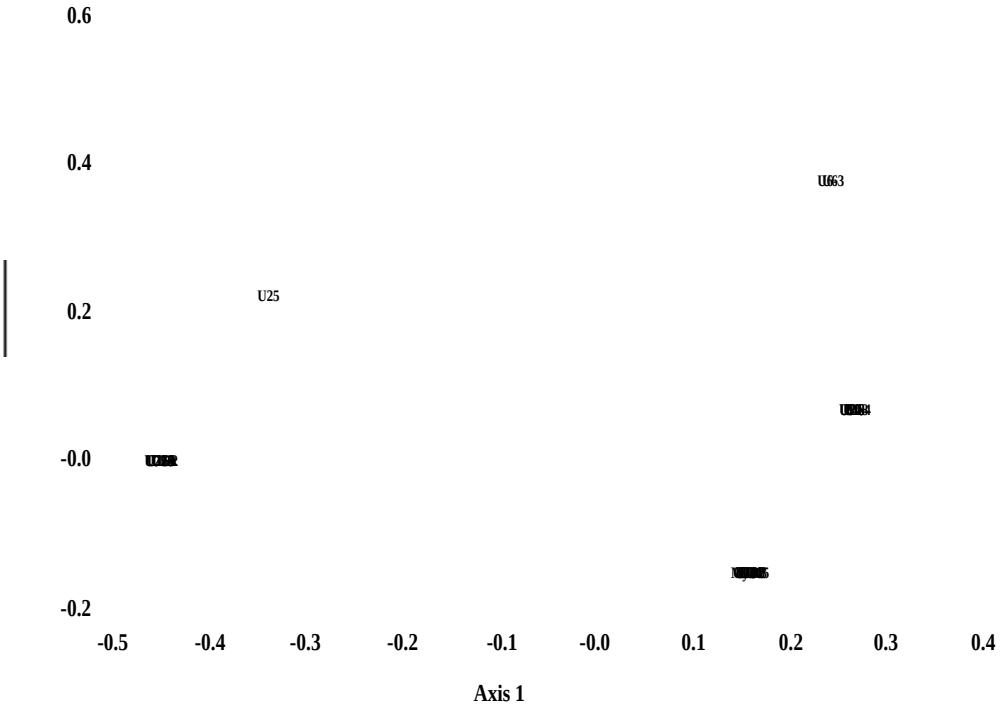




45 units; 29%, 20%, 11%

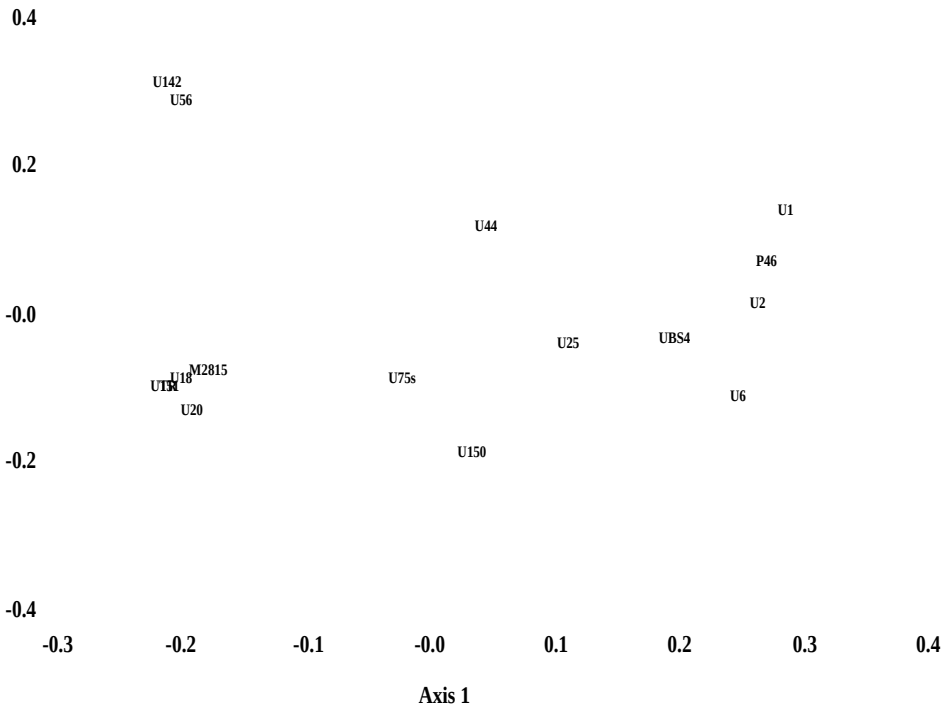
U75-x

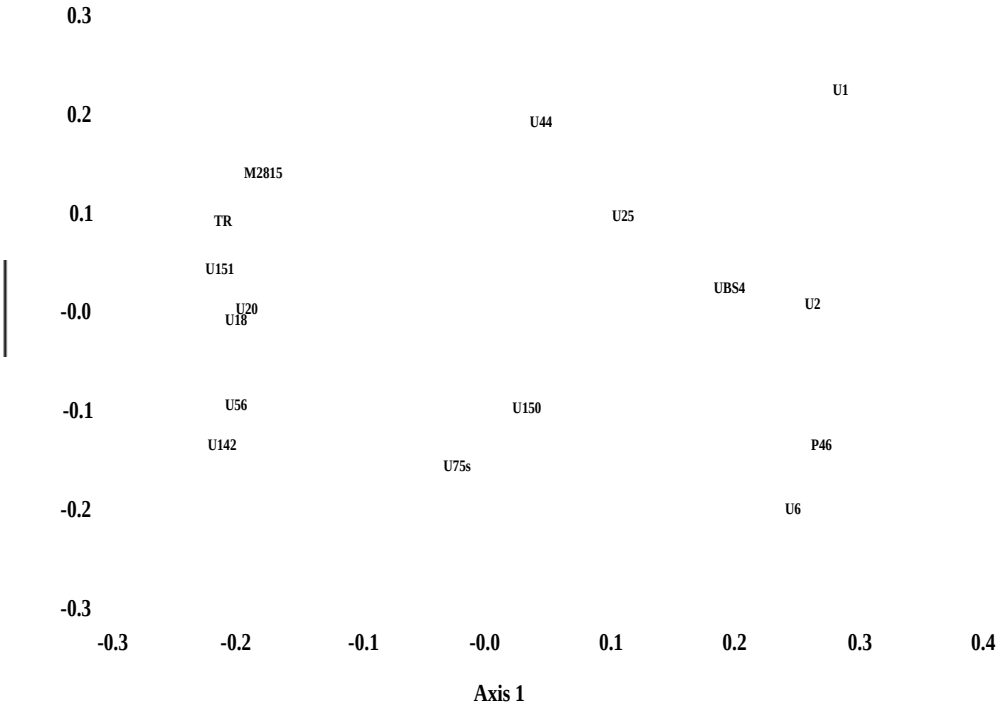




5 units; 59%, 27%, 15%

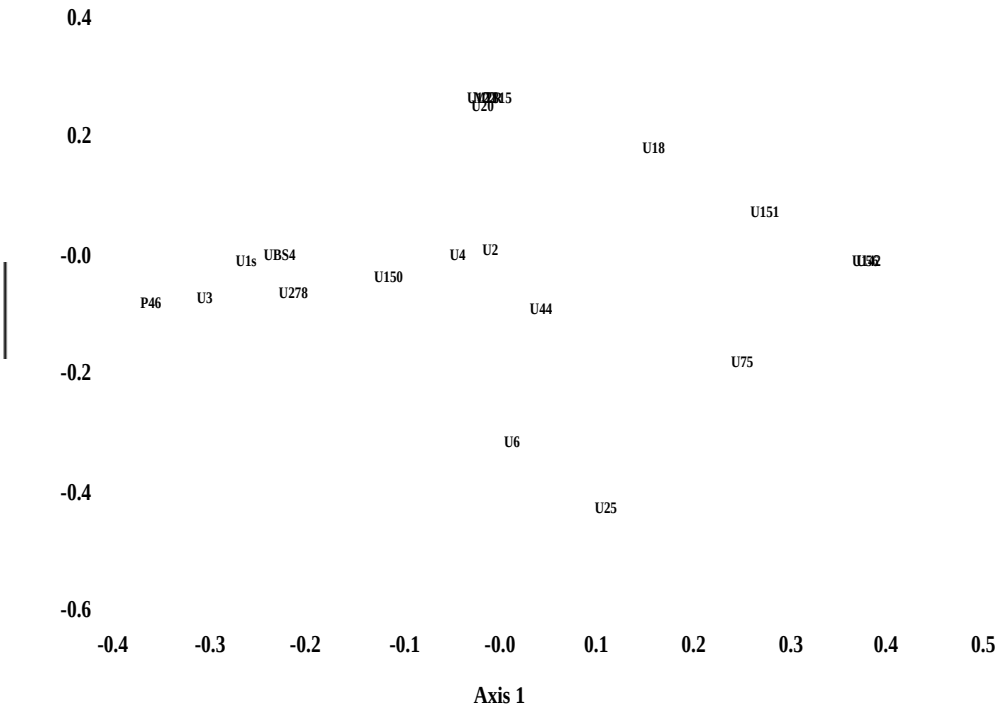
U75s (950?)

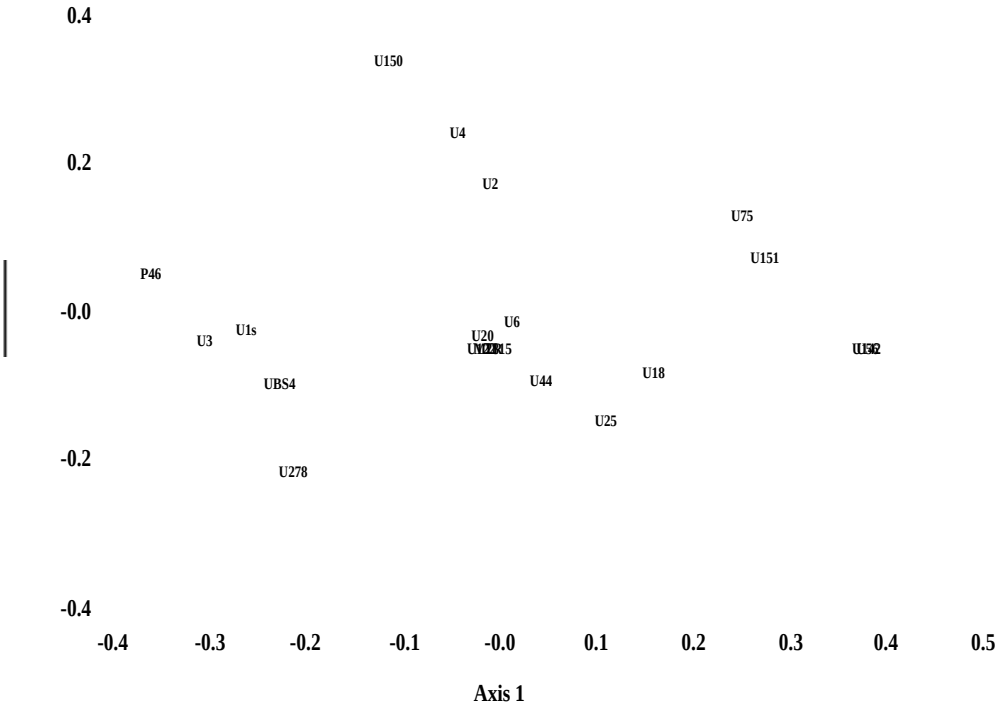




90 units; 25%, 13%, 10%

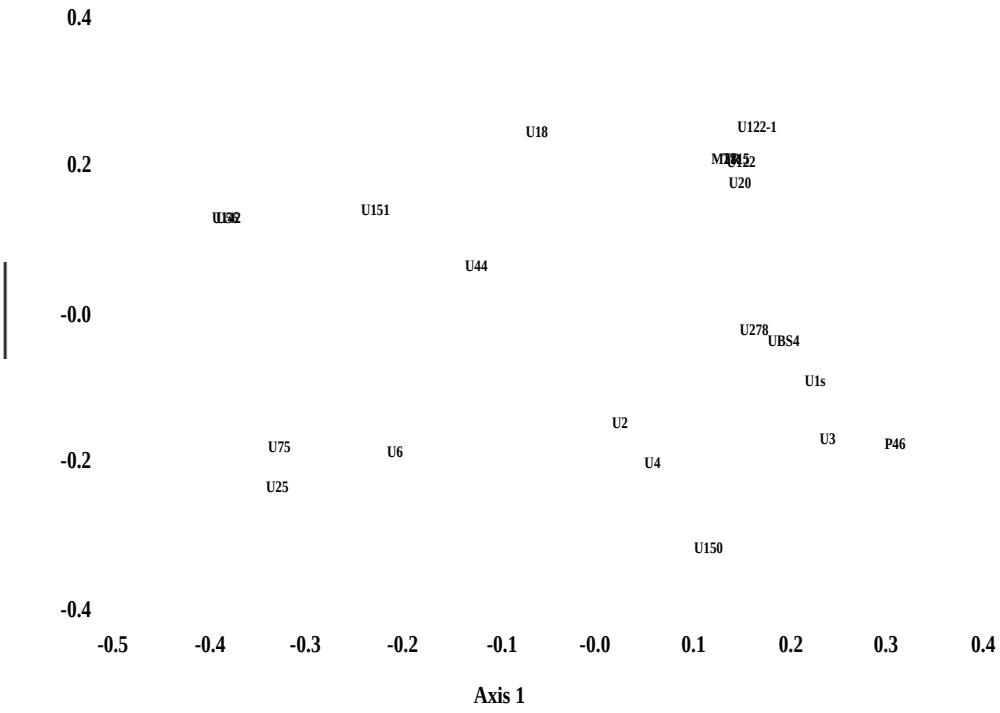
U122 (850?)

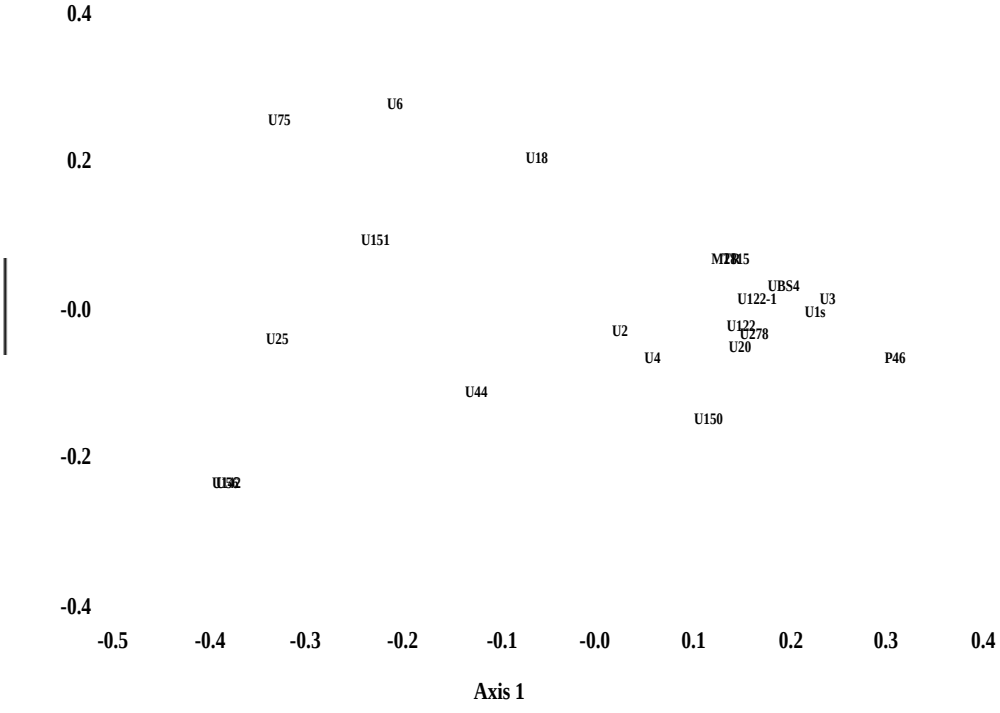




26 units; 27%, 21%, 11%

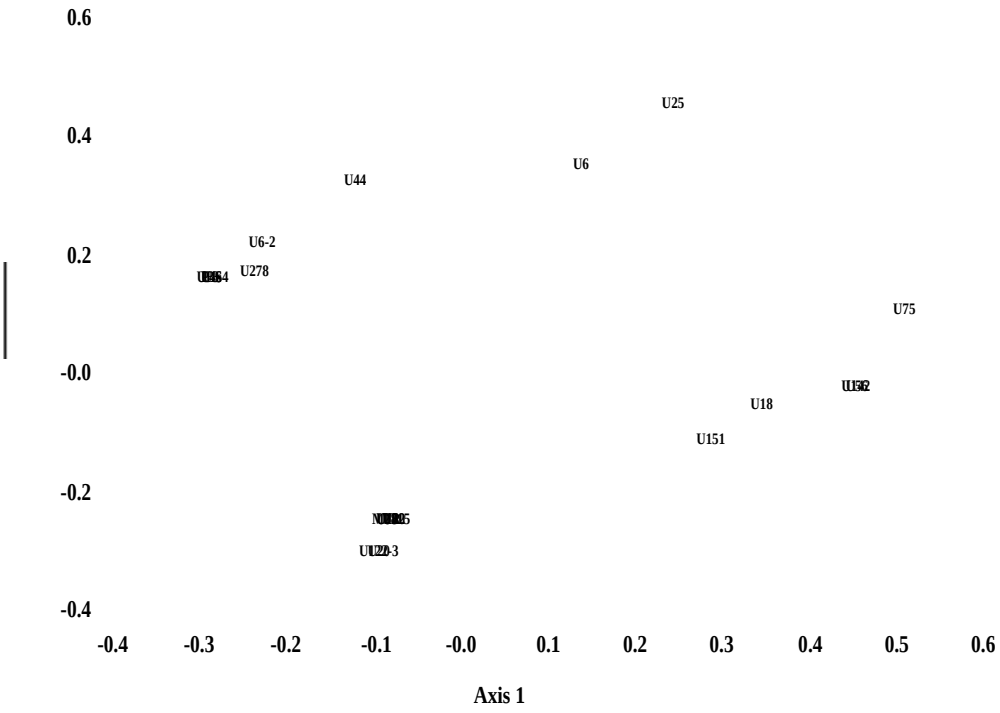
U122-1

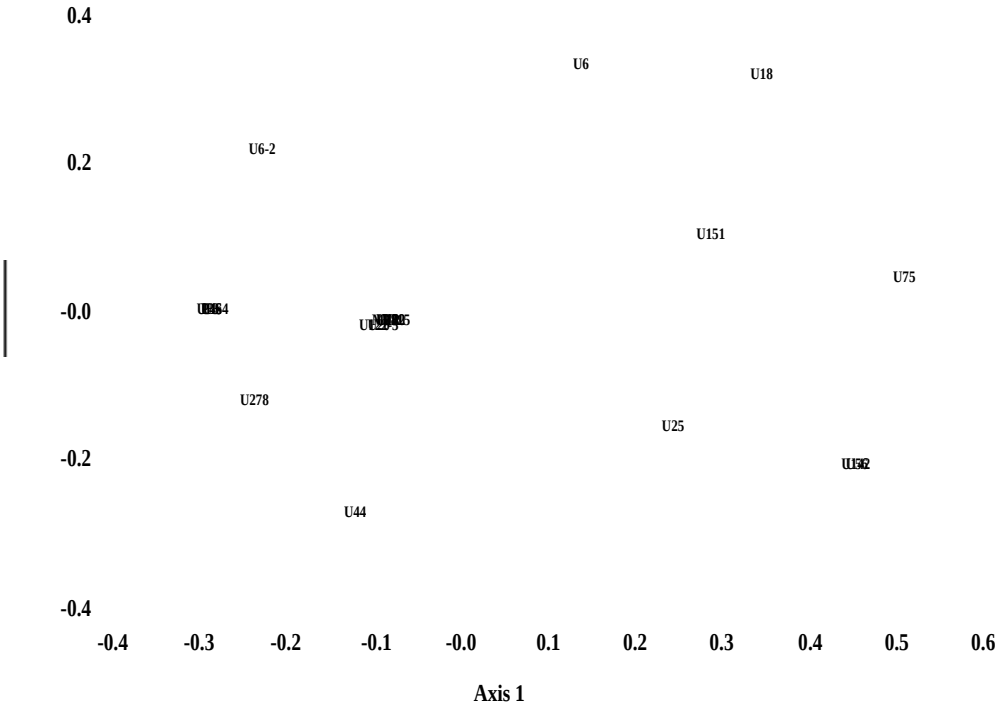




21 units; 31%, 21%, 11%

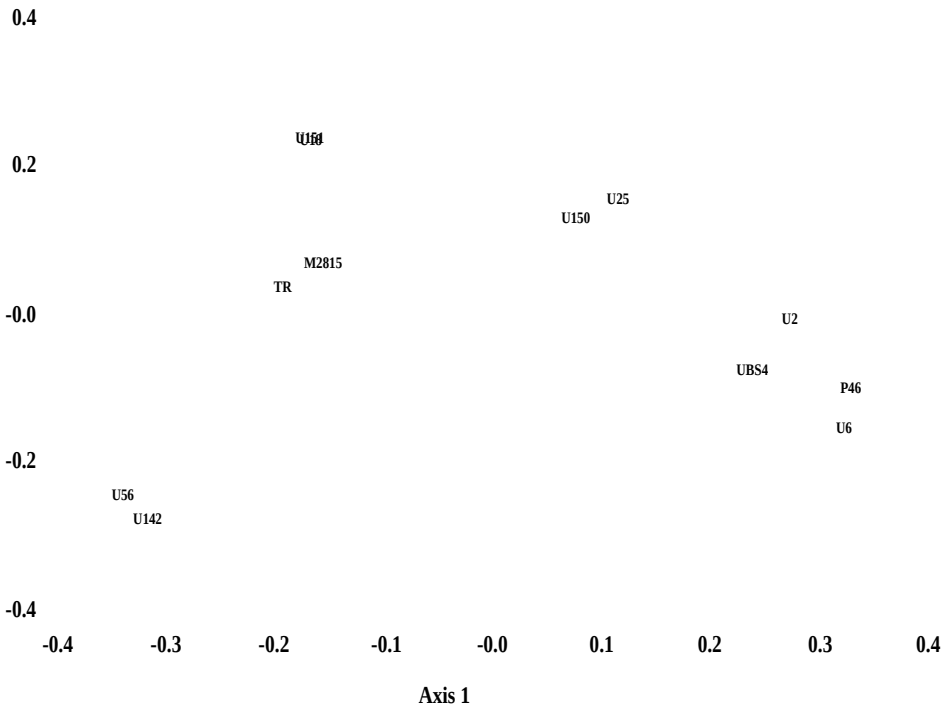
U122-3

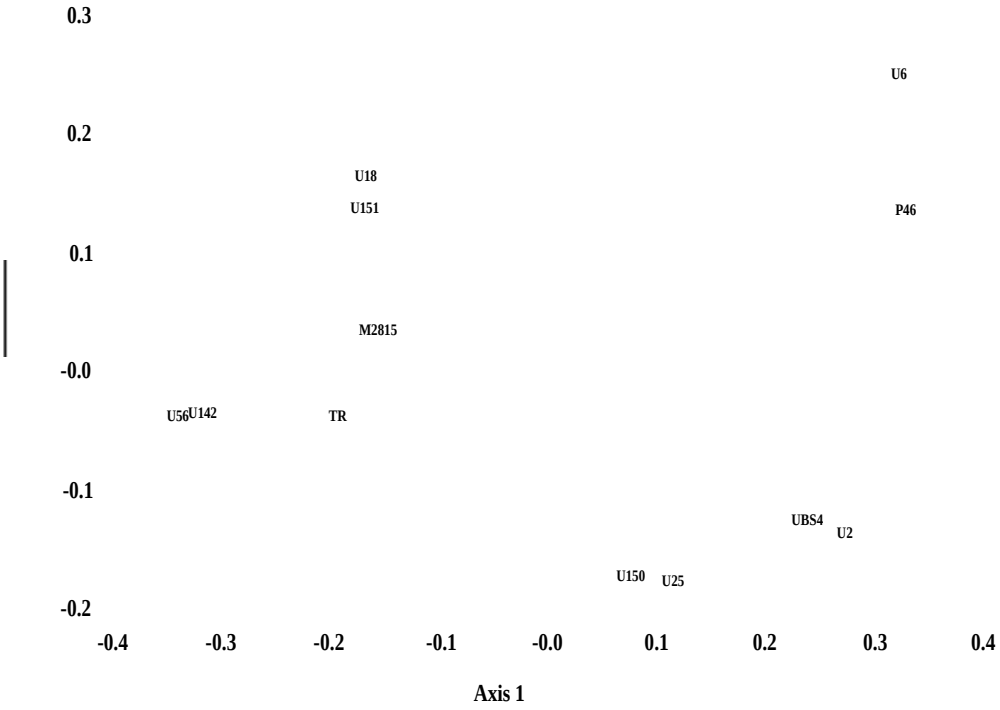




9 units; 38%, 31%, 12%

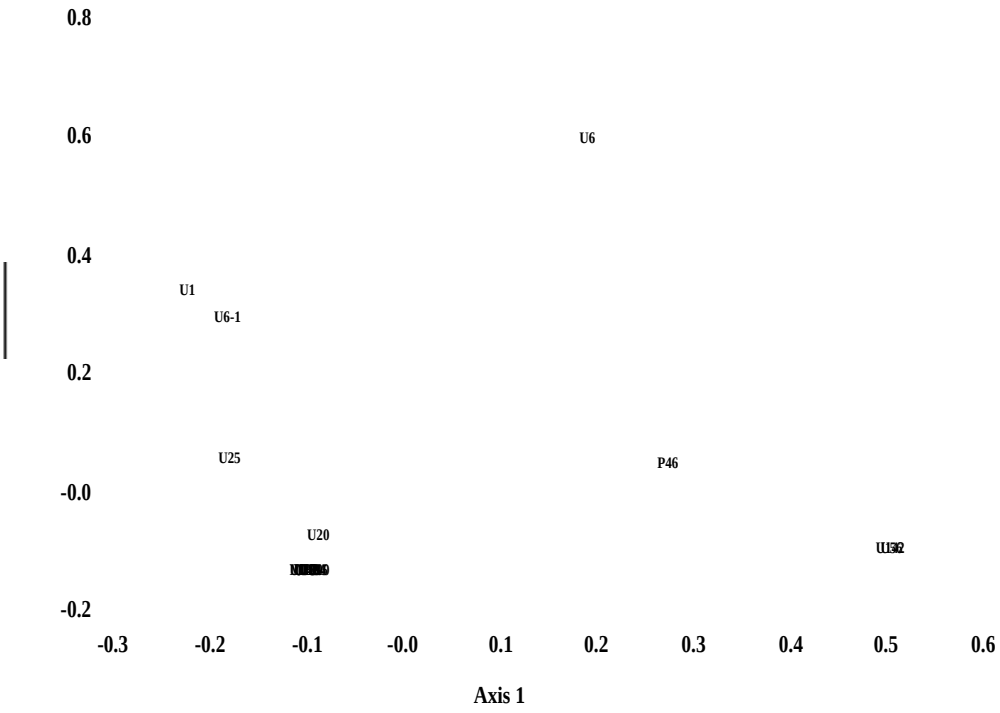
U142 (950?)

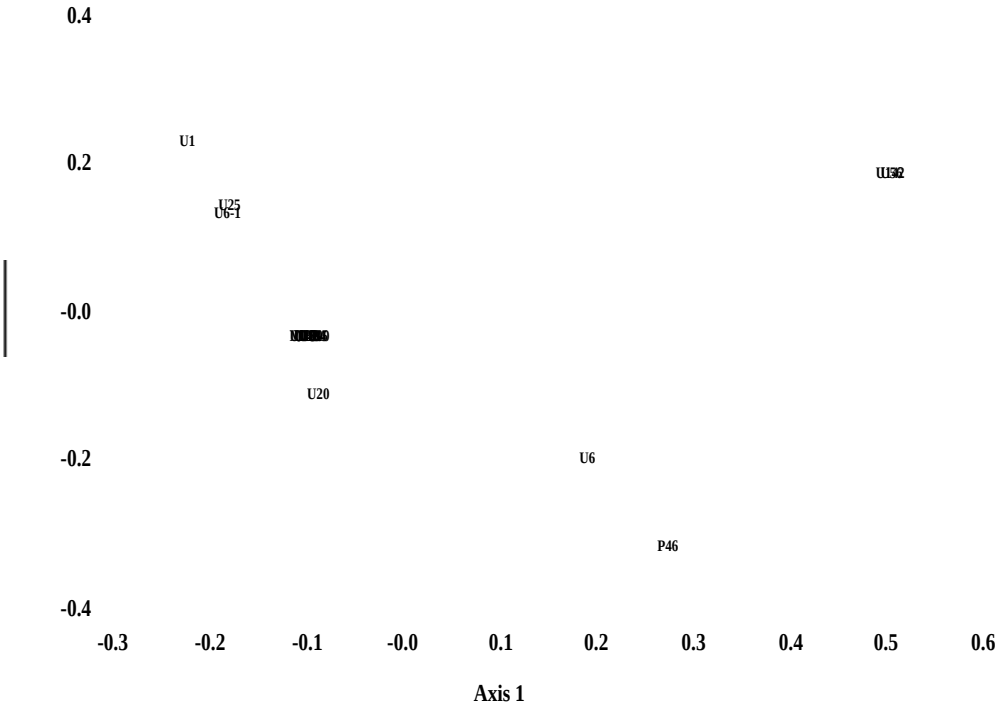




728 unit; 32%, 15%, 10%

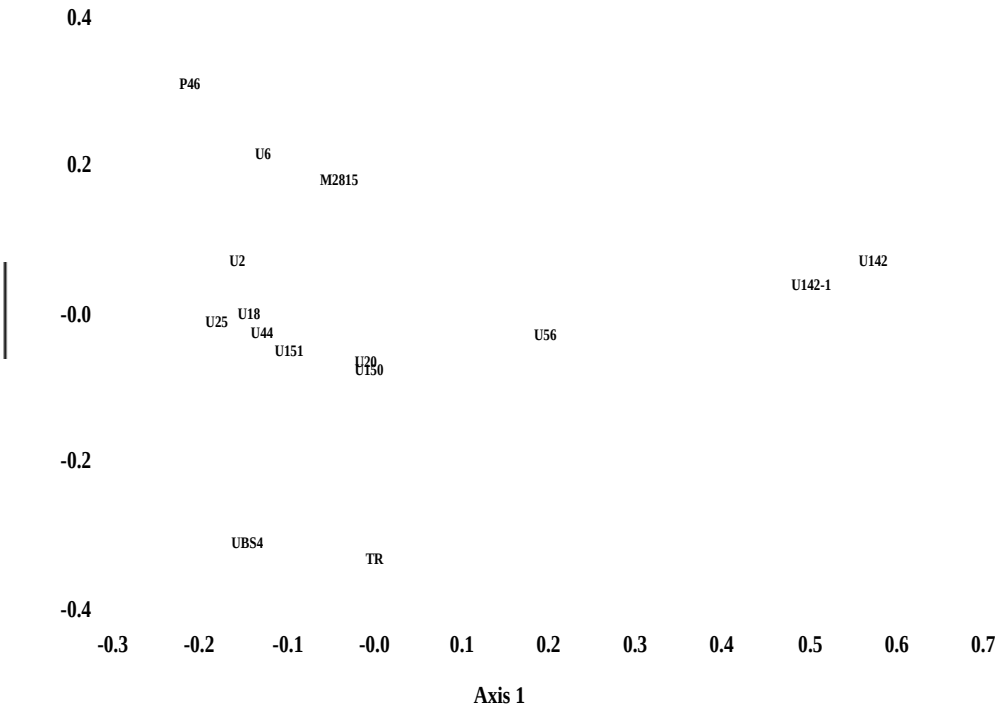
U142-0

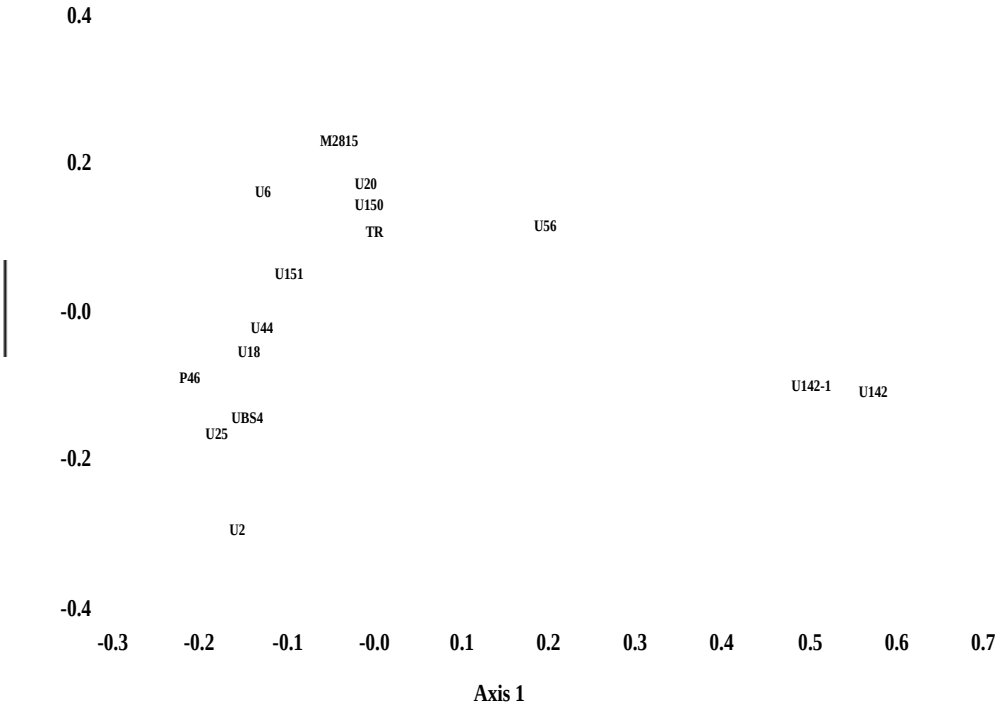




12 units; 38%, 33%, 15%

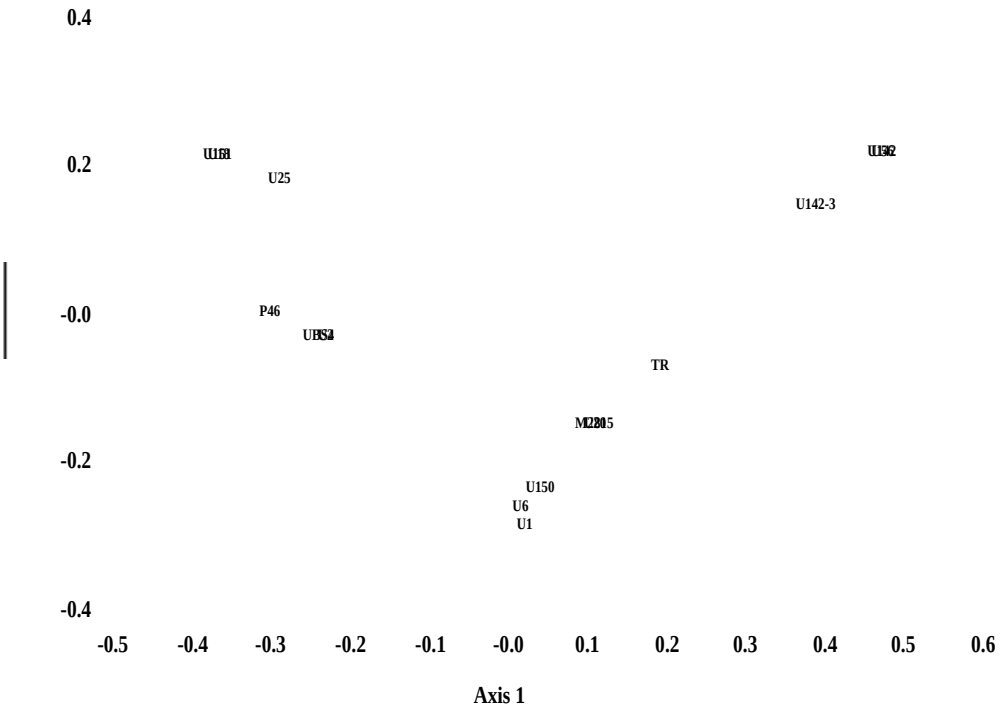
U142-1

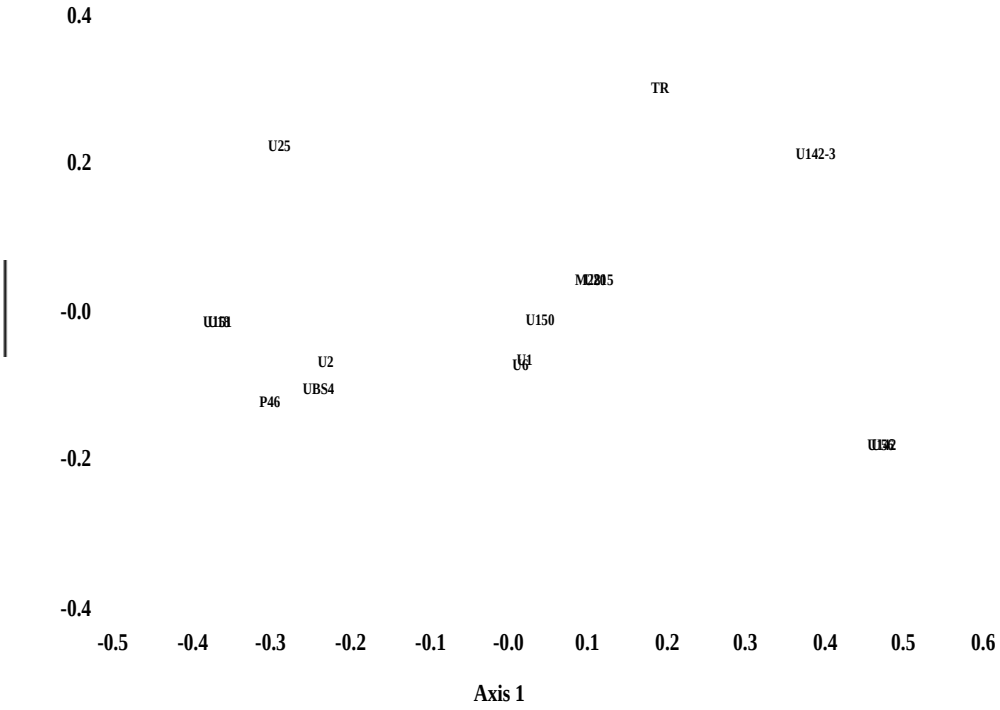




36 units; 36%, 18%, 14%

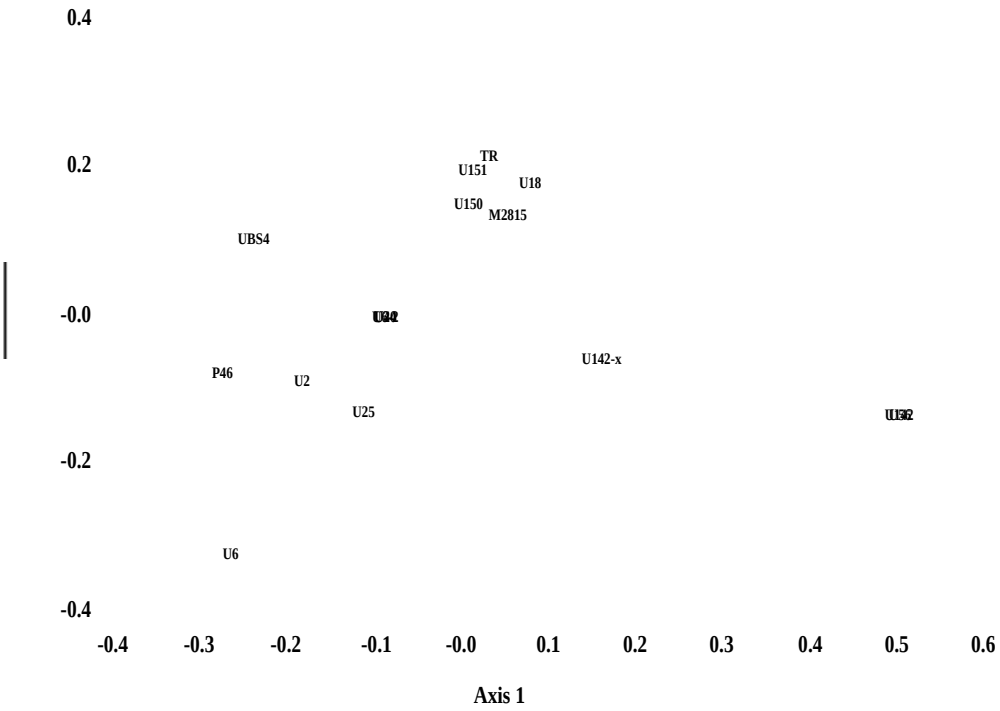
U142-3

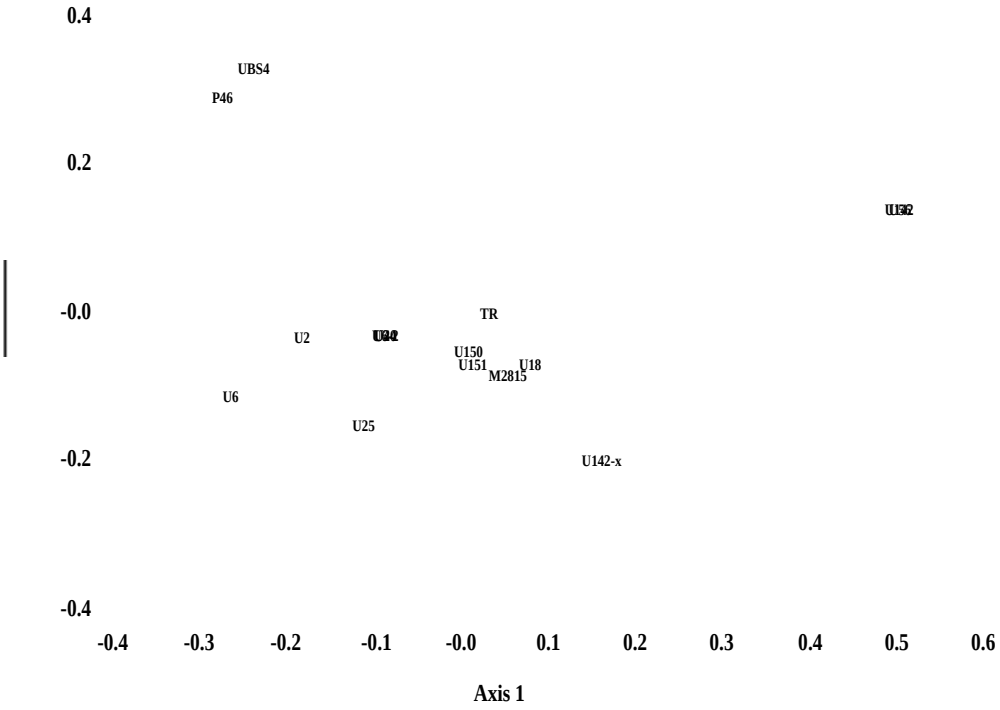




22 units; 47%, 19%, 11%

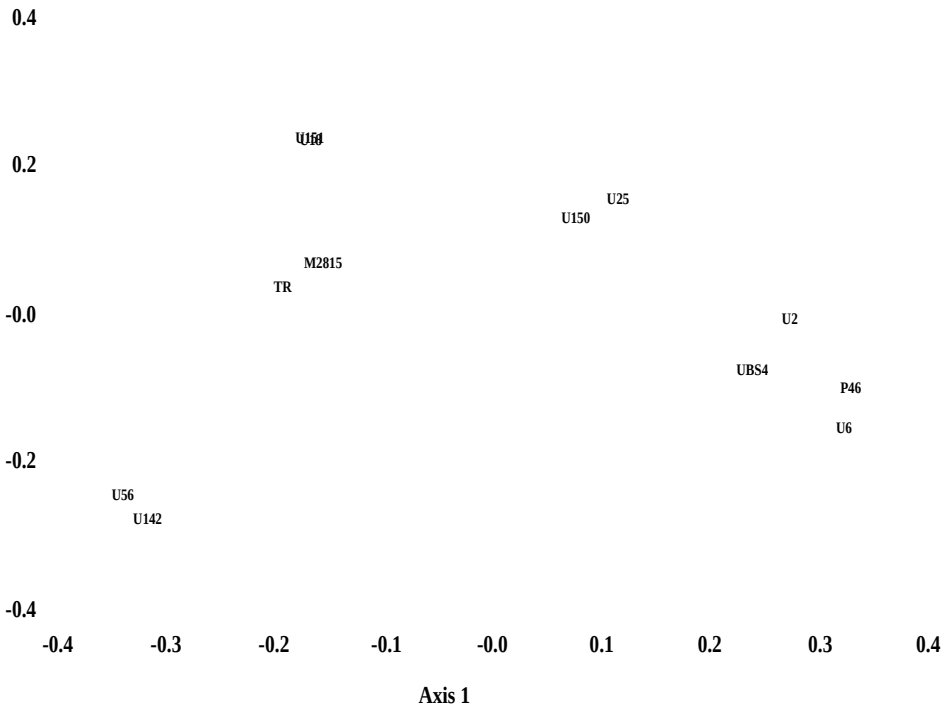
U142-x

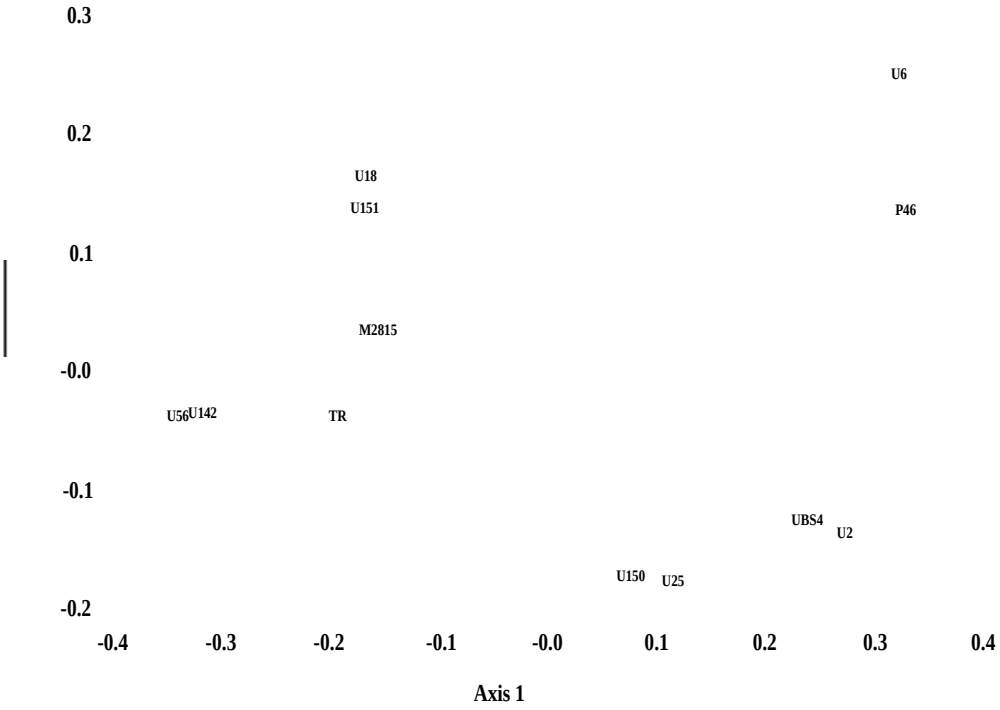




23 units; 37%, 16%, 15%

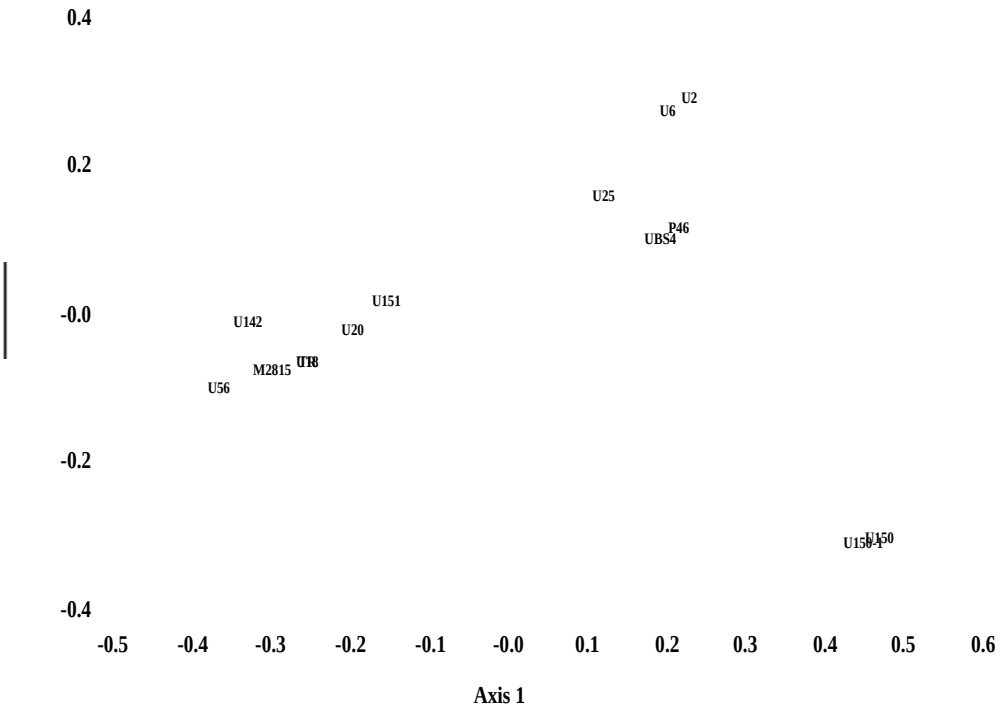
U150 (850?)

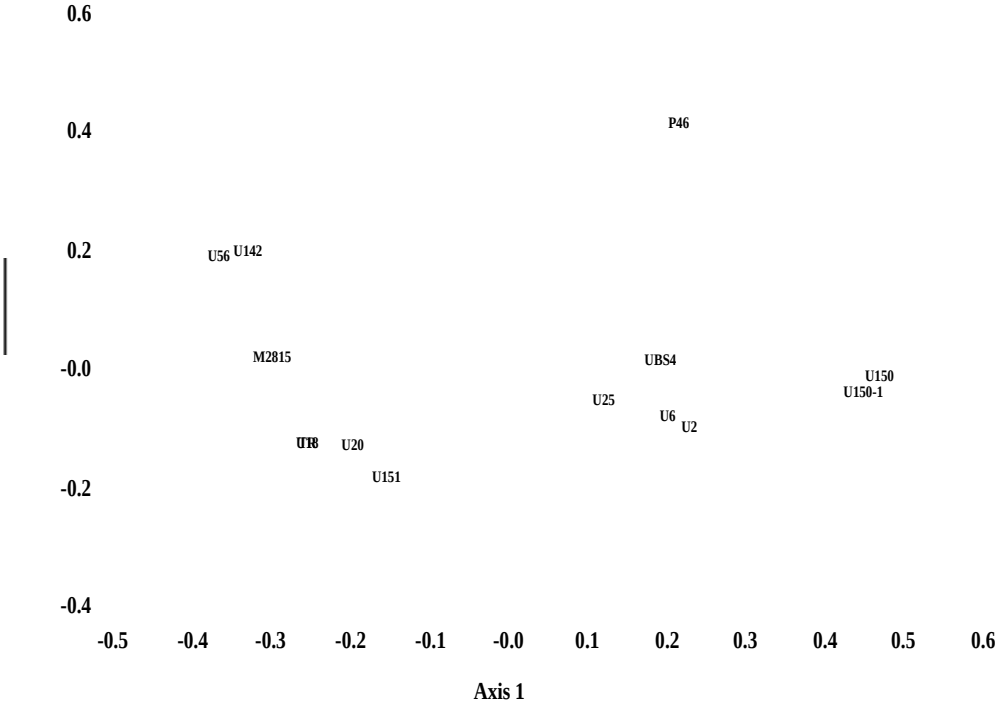




728 units; 32%, 15%, 10%

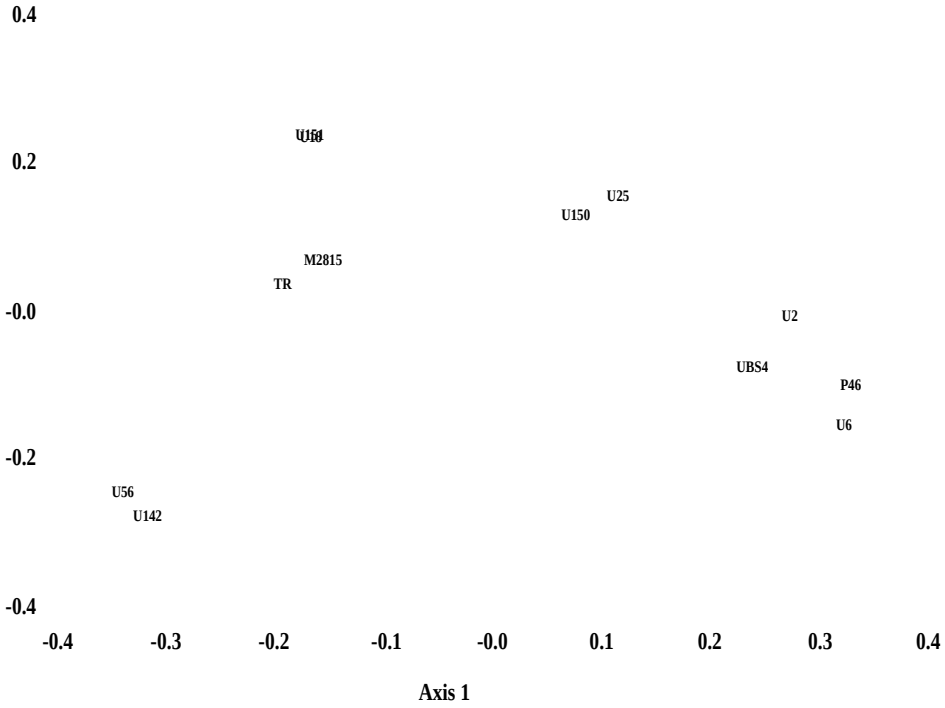
U150-1

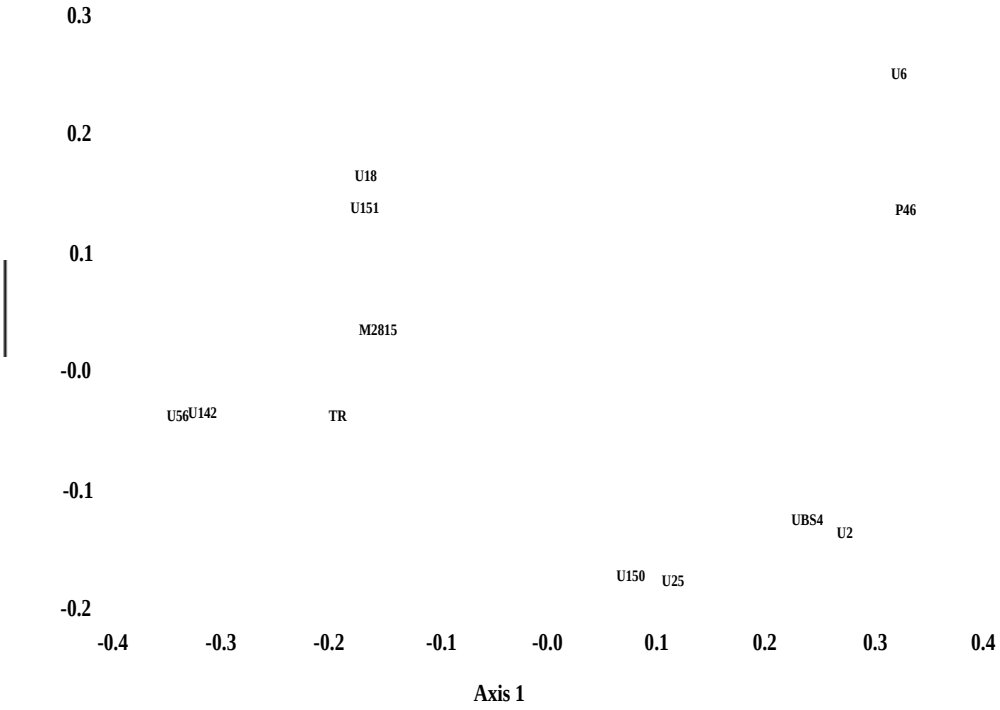




33 units; 46%, 17%, 14%

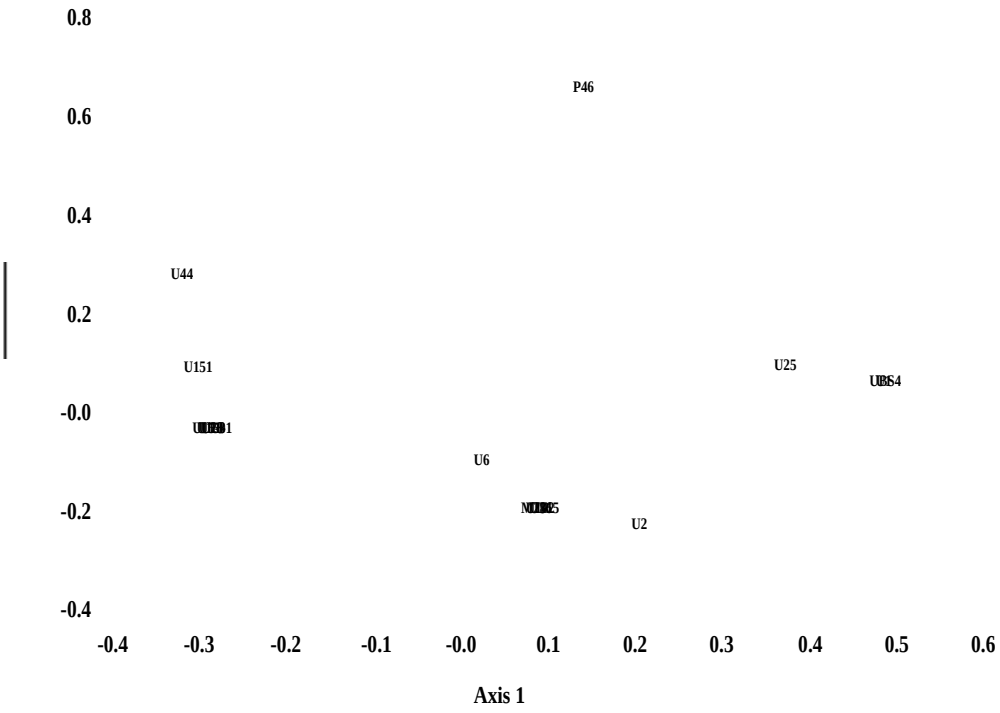
U151 (850?)

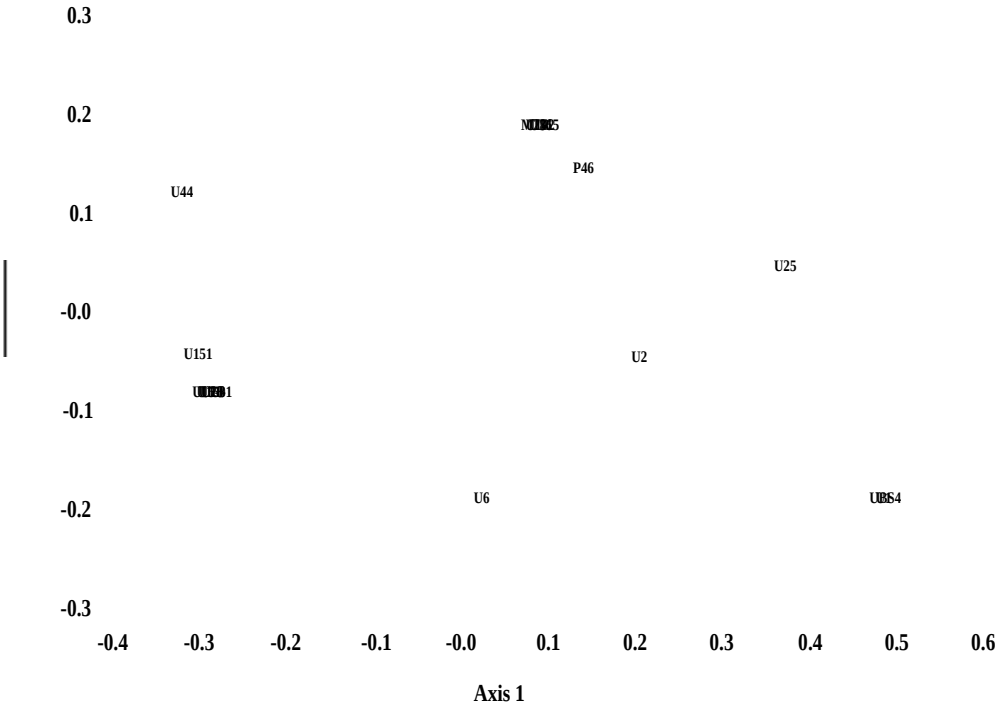




728 units; 32%, 15%, 10%

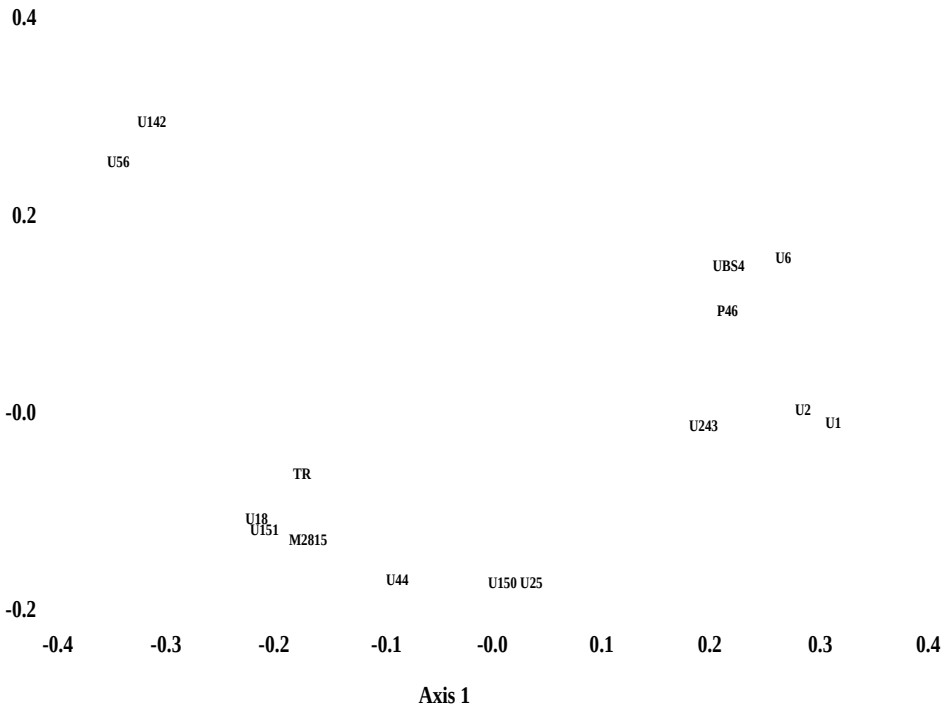
U151-1

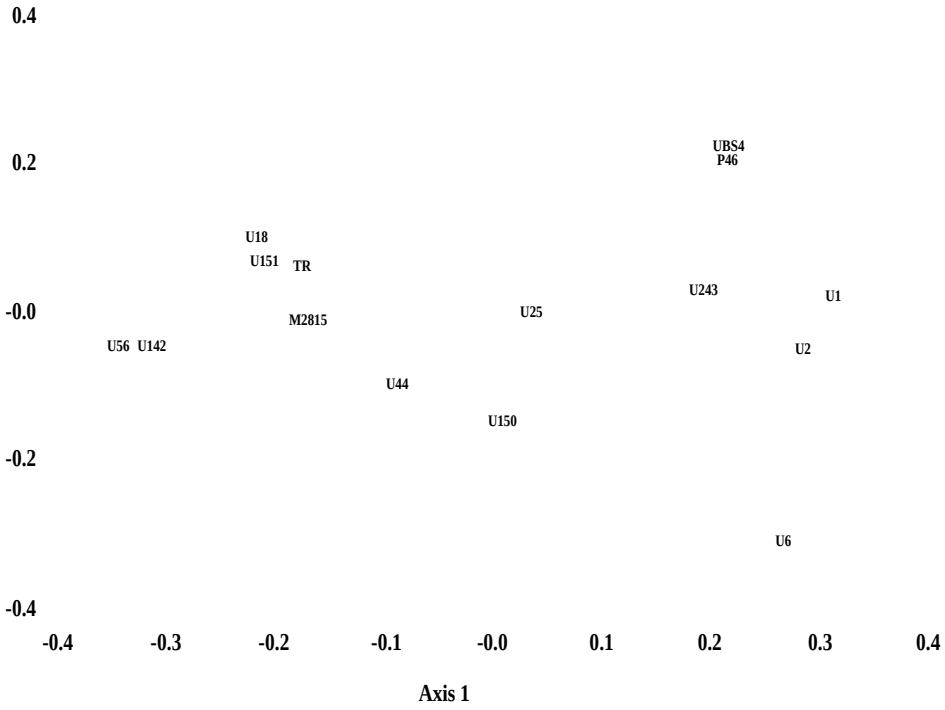




8 units; 46%, 27%, 11%

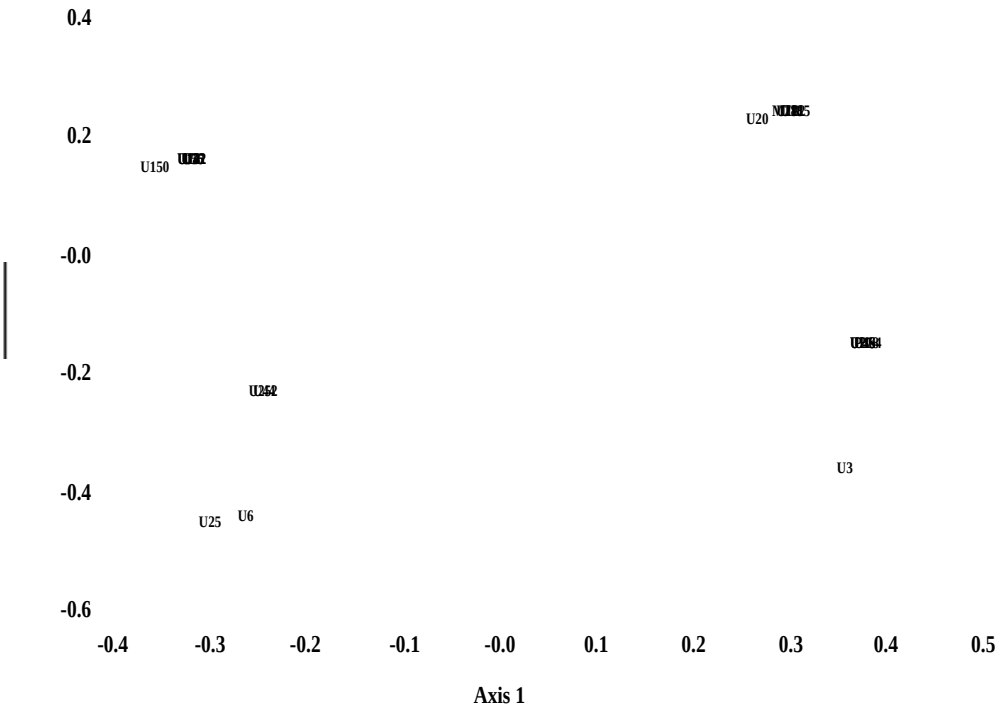
U243 (950?)

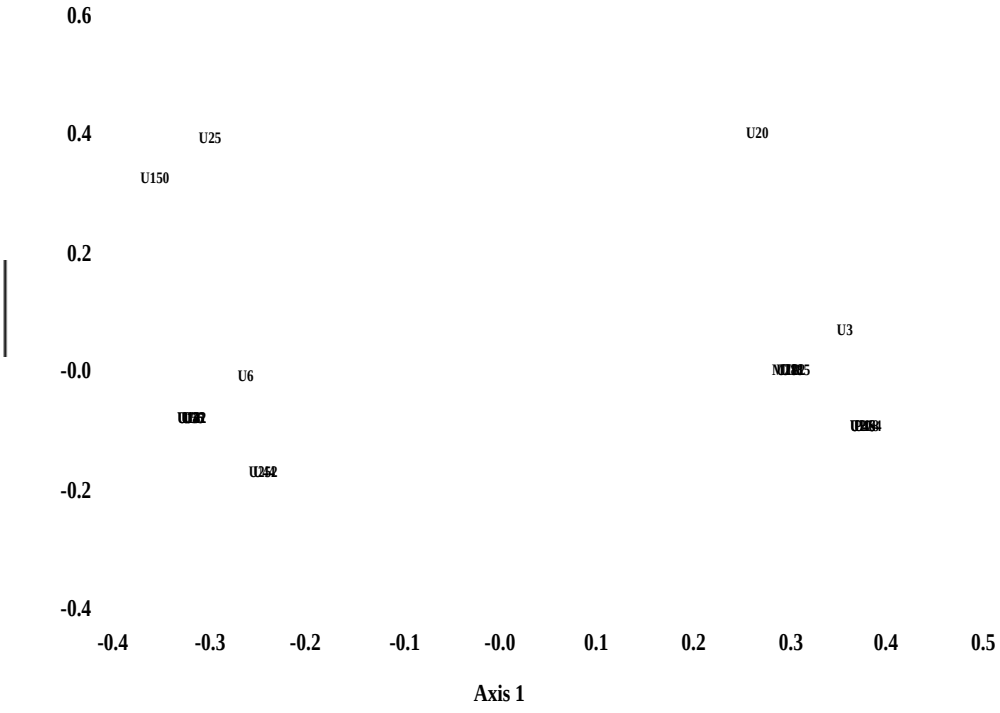




241 units; 29%, 13%, 9%

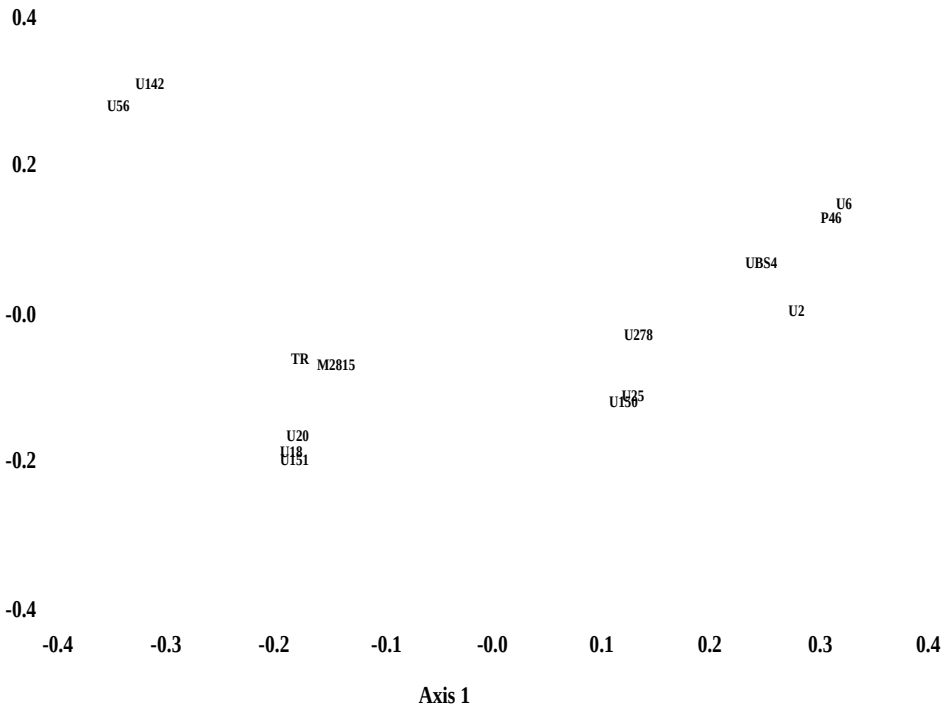
U252 (450?)

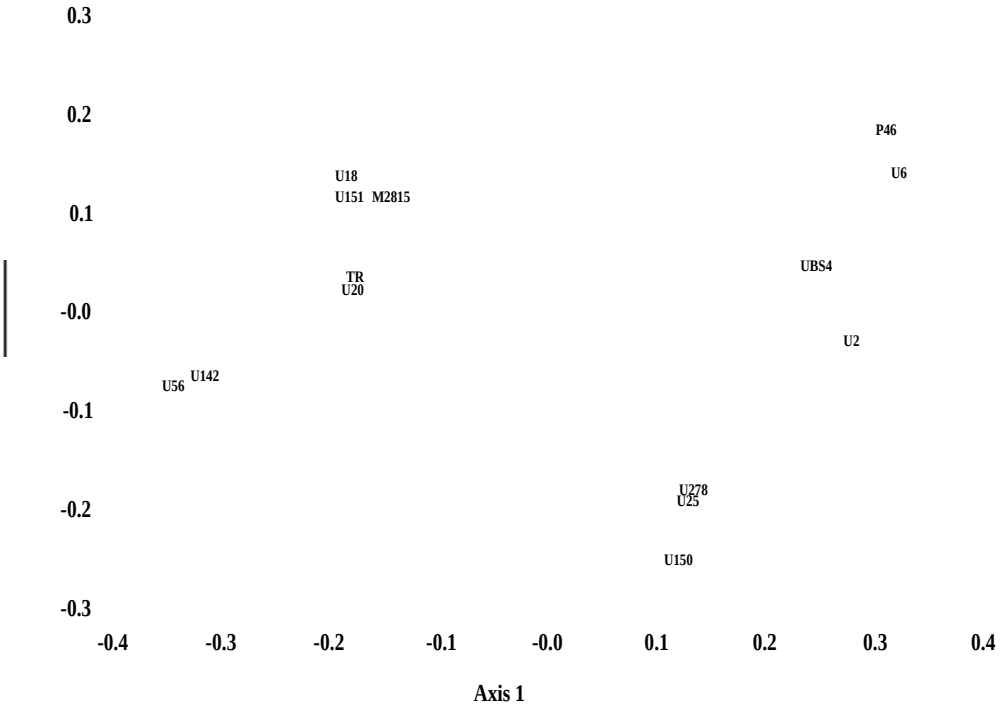




5 units; 52%, 29%, 13%

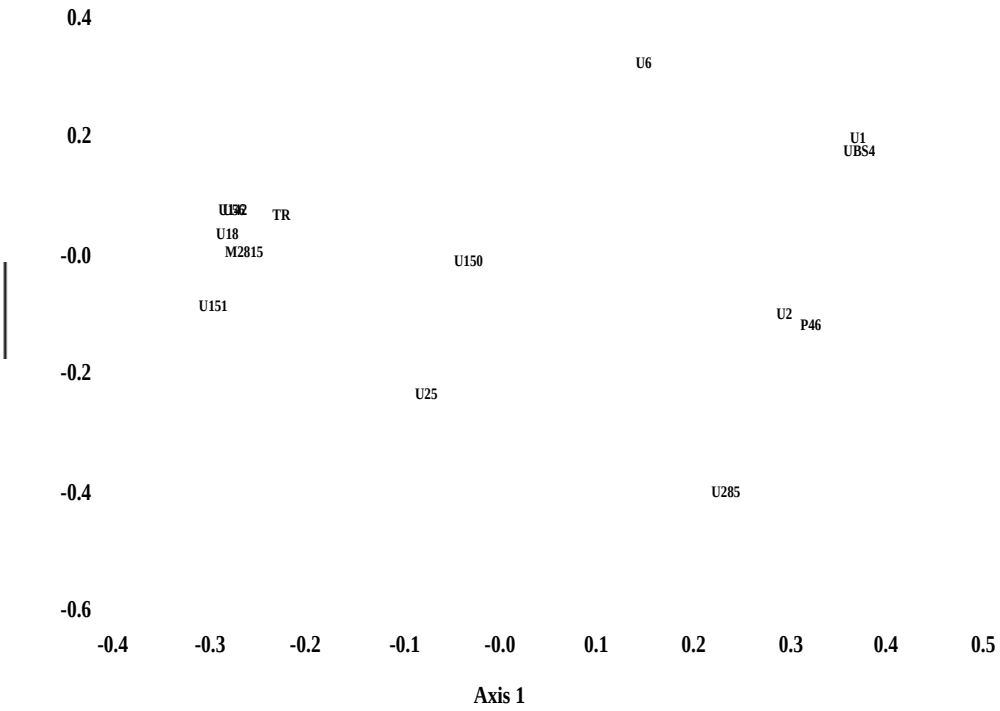
U278 (850?)

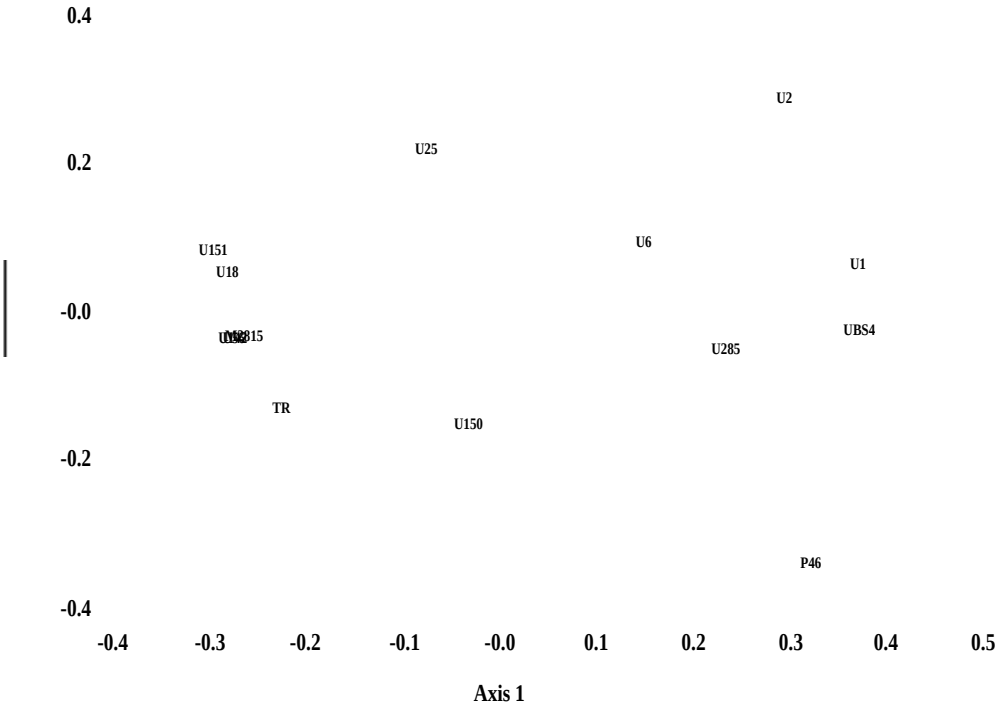




423 units; 31%, 14%, 10%

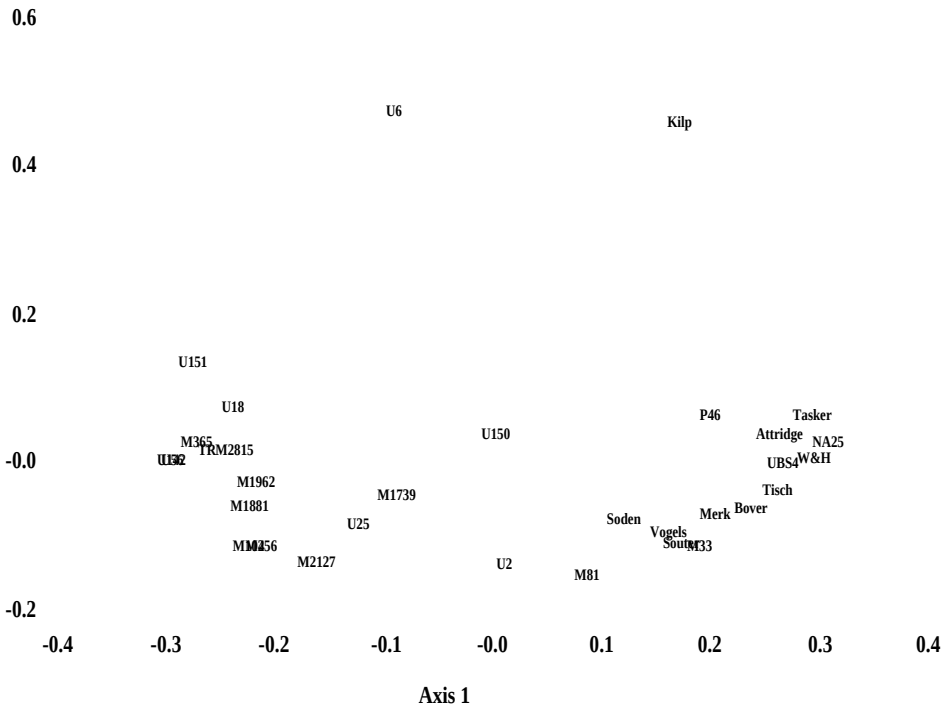
U285 (550?)

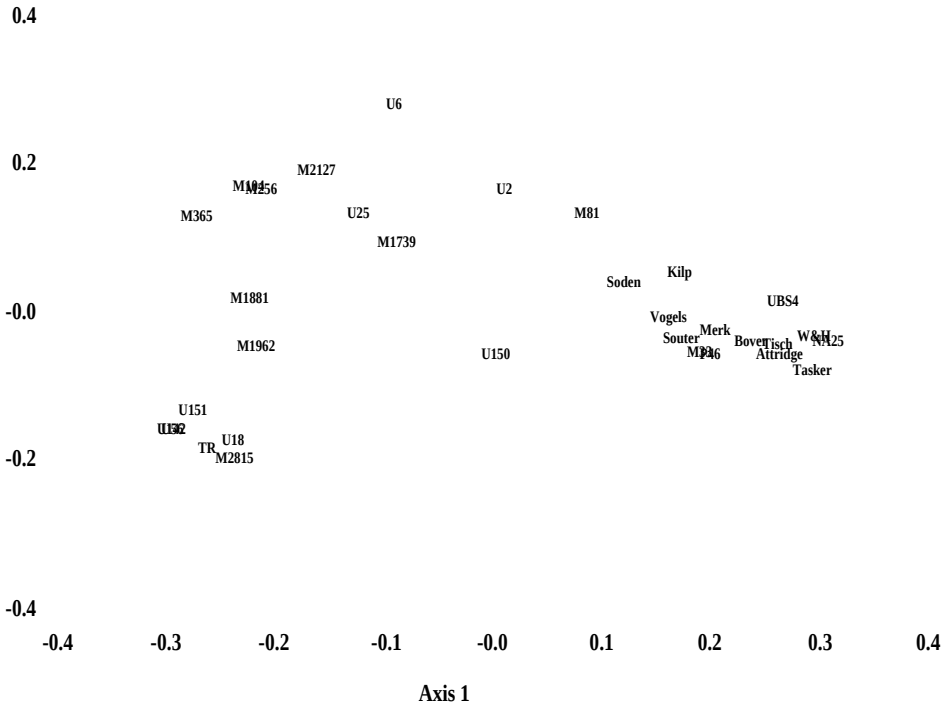




28 units; 37%, 16%, 12%

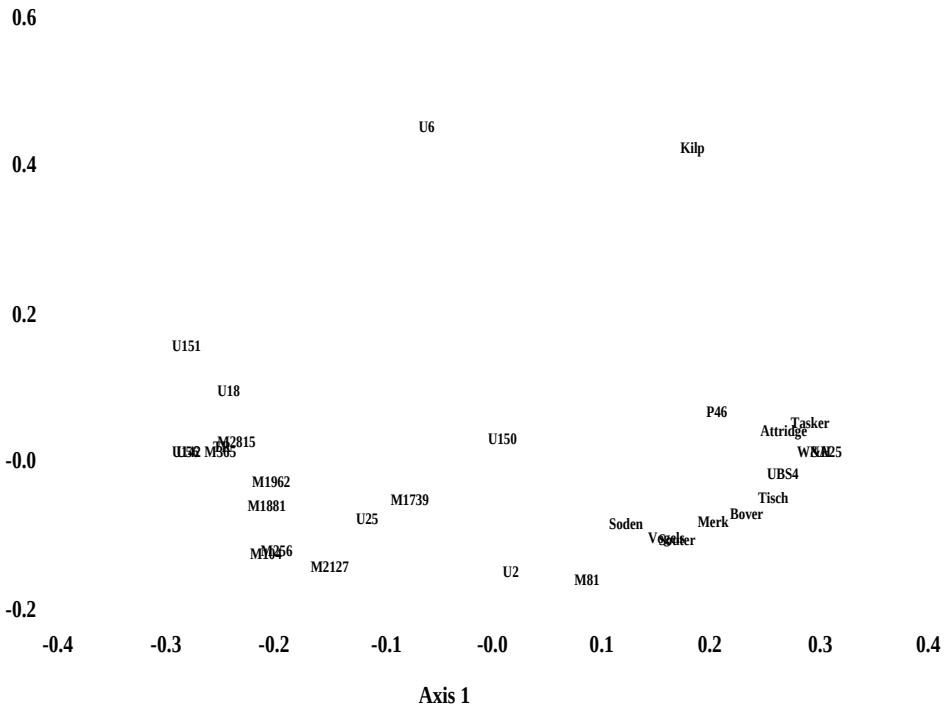
M33 (850?)

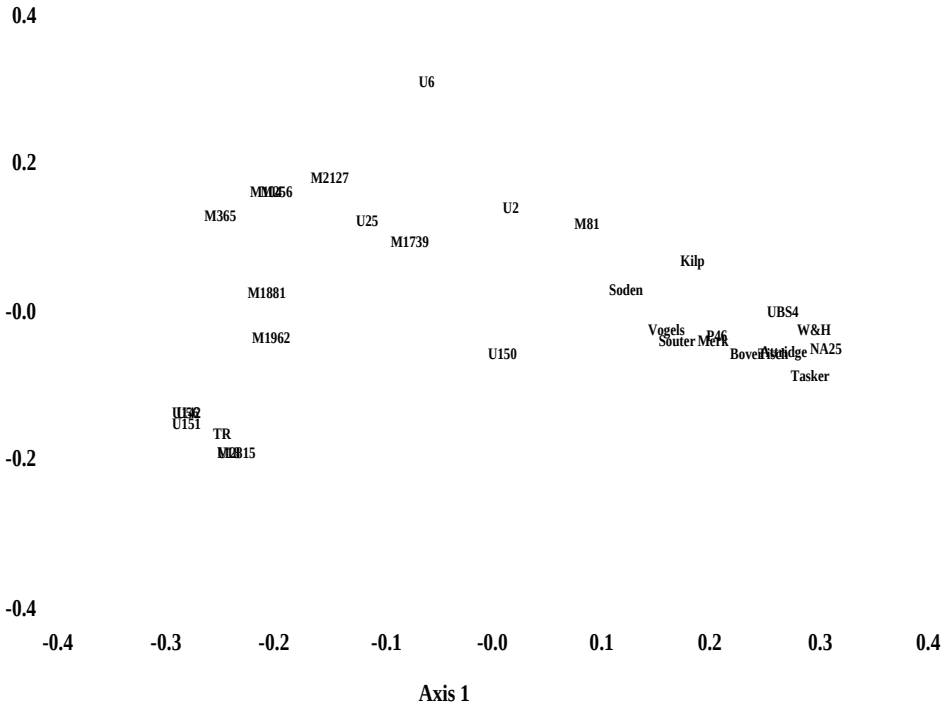




67 units; 31%, 13%, 10%

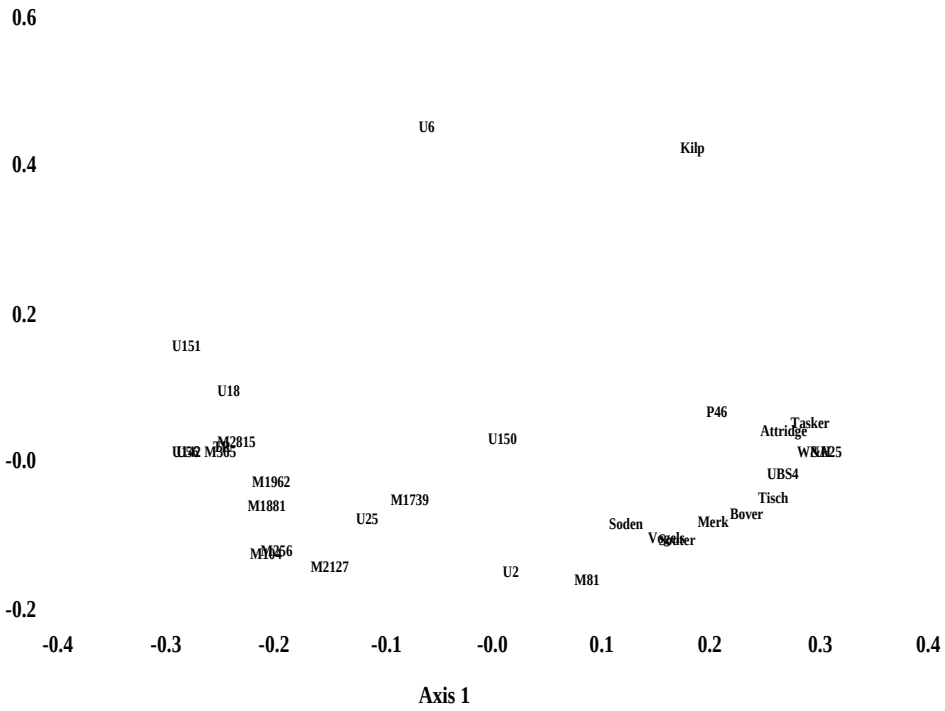
M81 (1044)

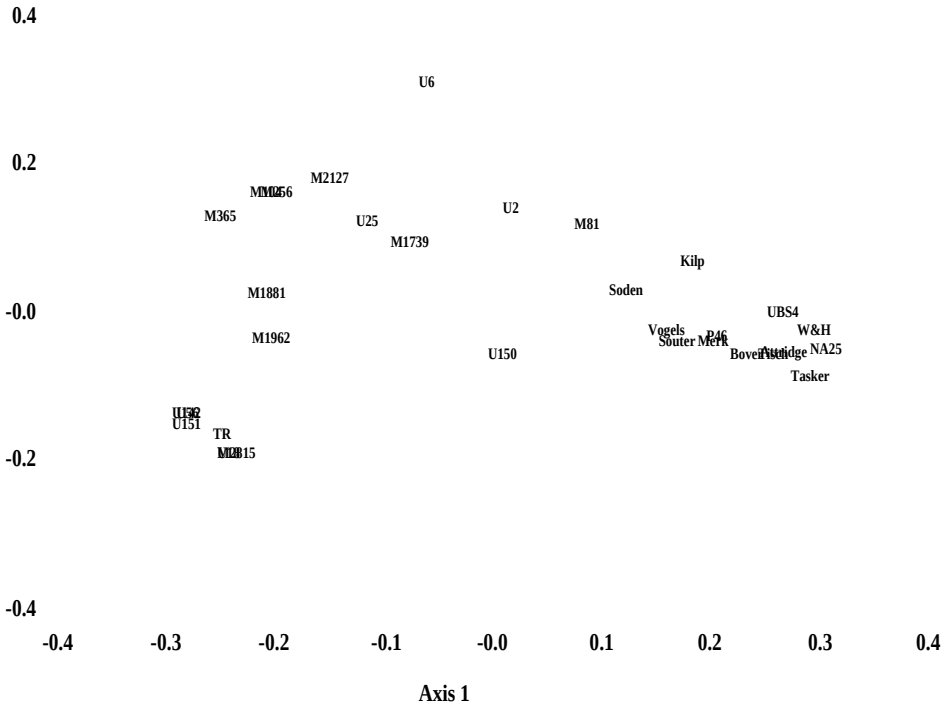




71 units; 30%, 12%, 10%

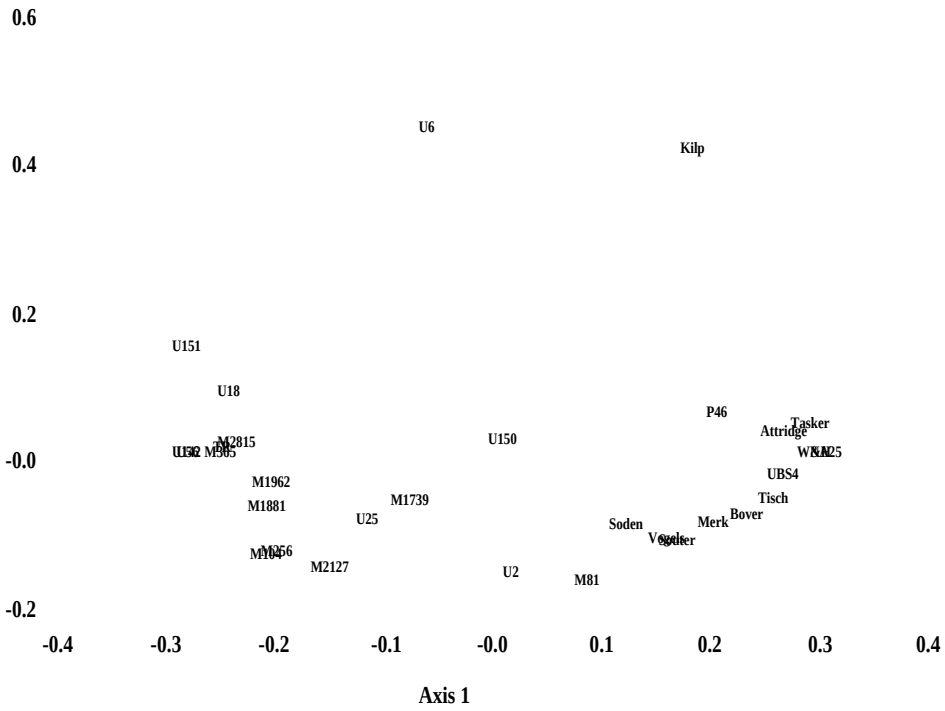
M104 (1087)

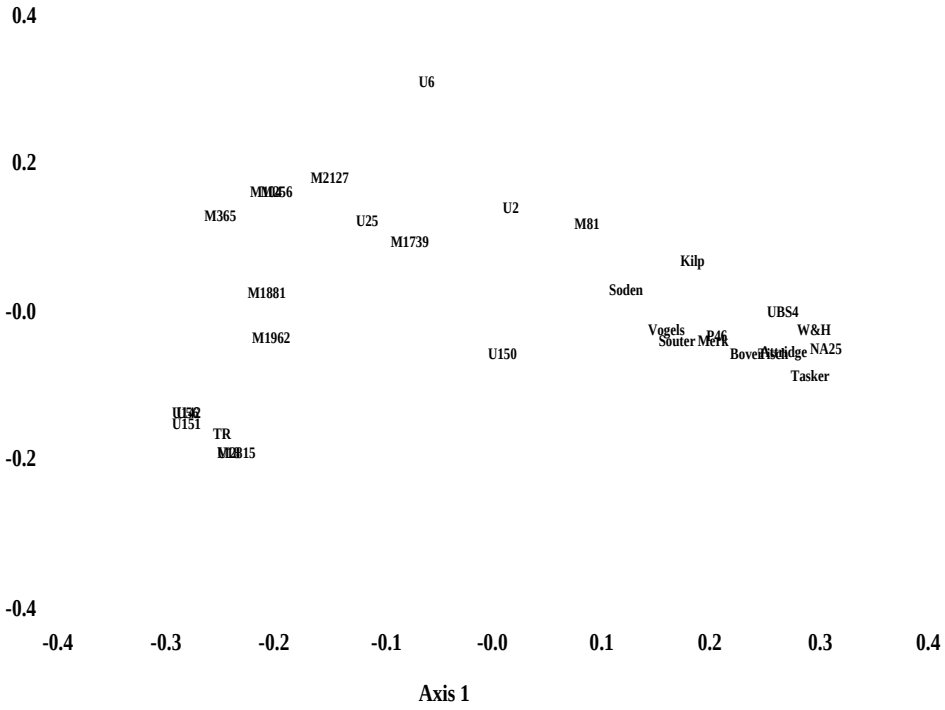




71 units; 30%, 12%, 10%

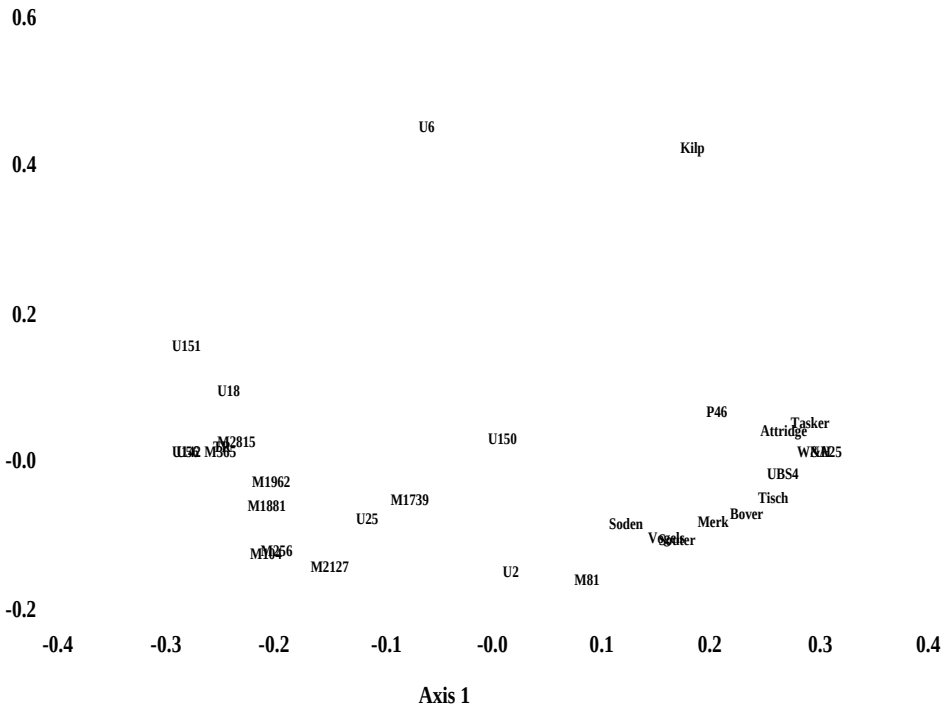
M256 (1100?)

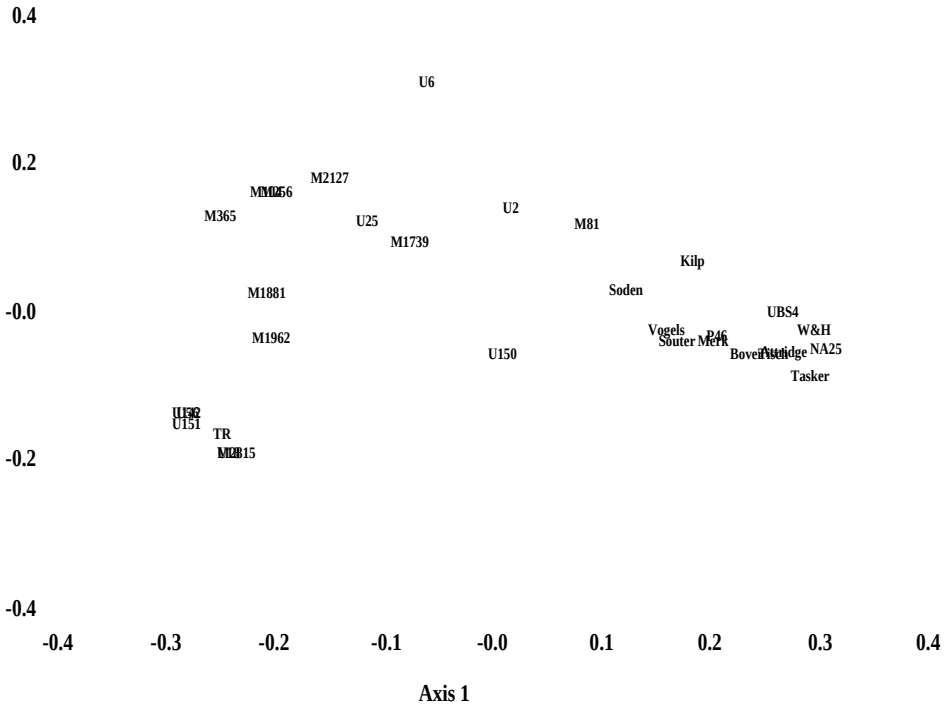




71 units; 30%, 12%, 10%

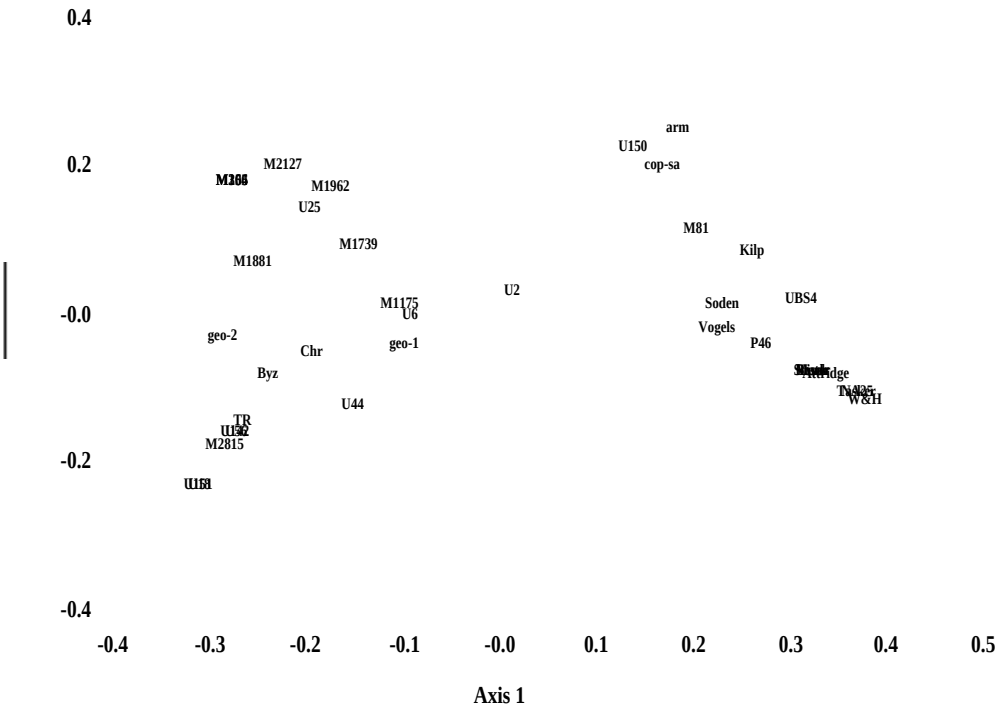
M365 (1250?)

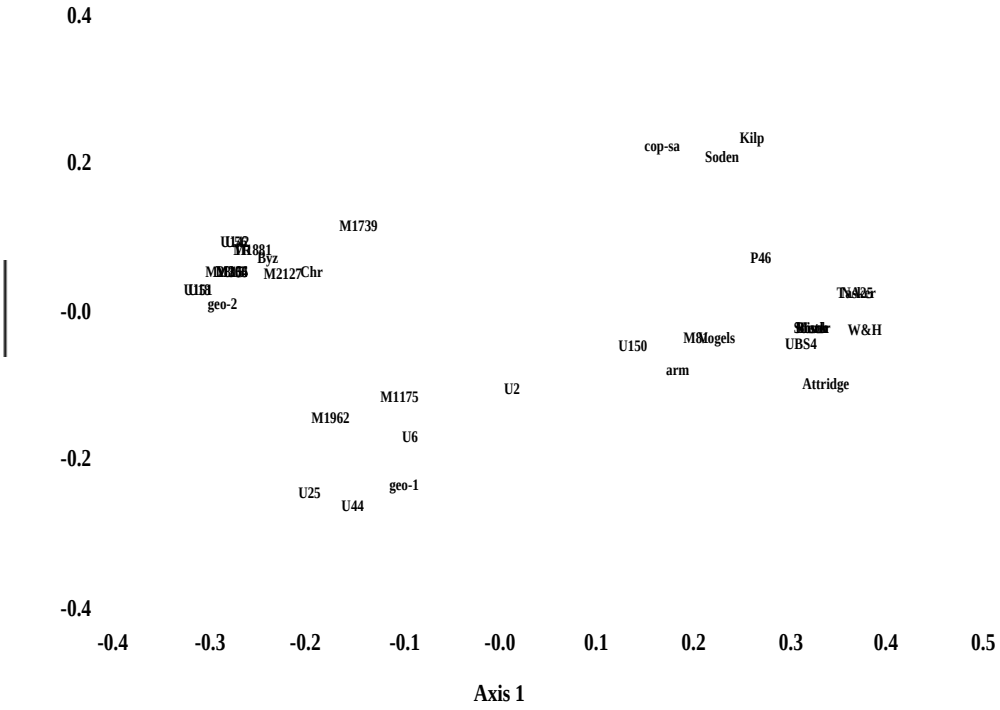




71 units; 30%, 12%, 10%

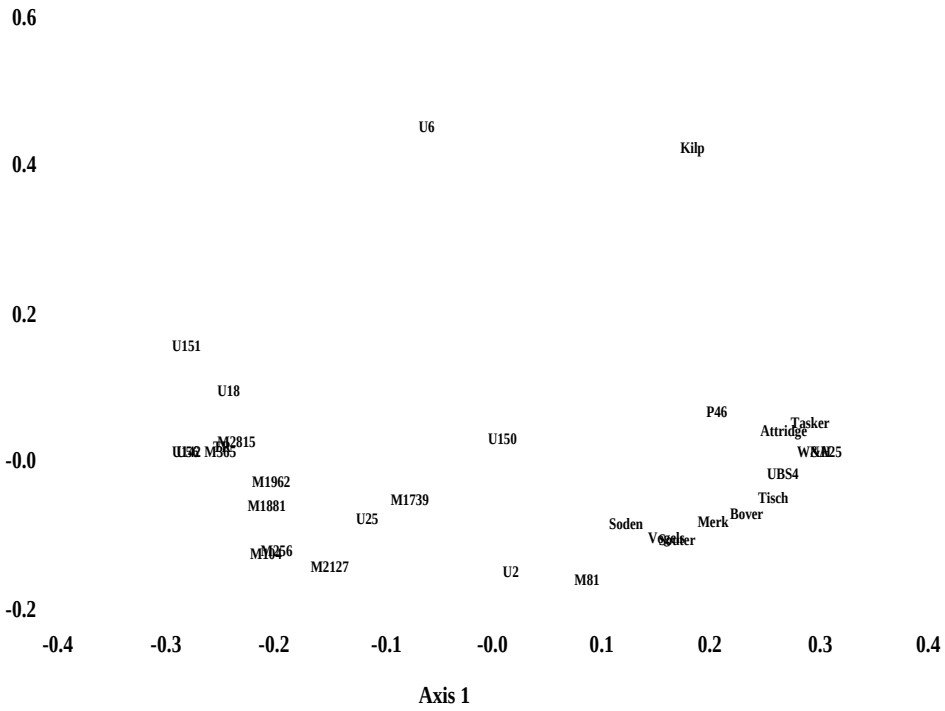
M1175 (1050?)

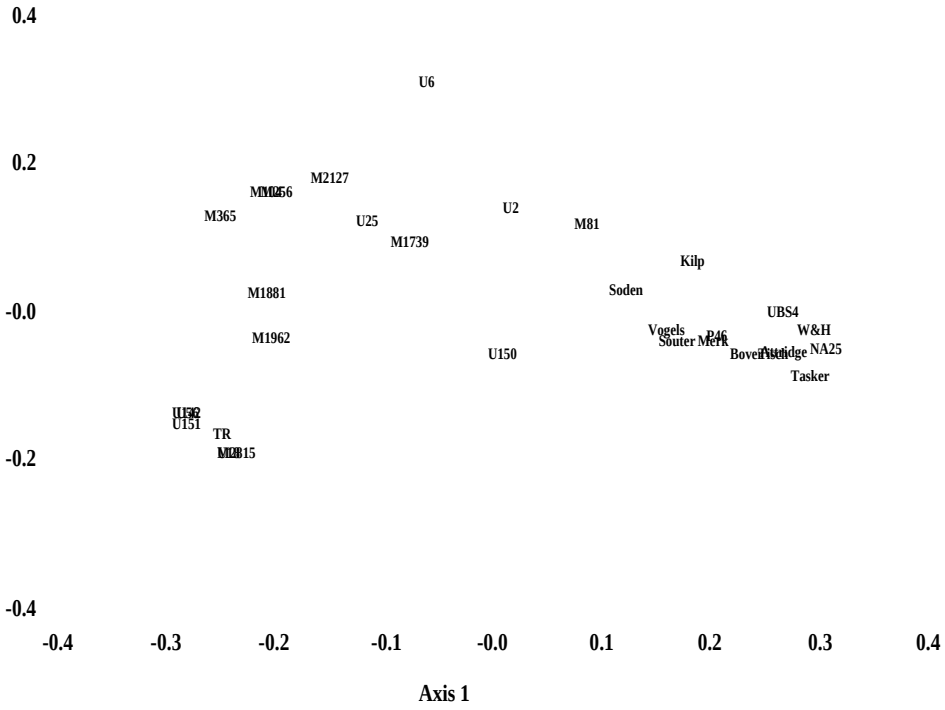




28 units; 42%, 11%, 8%

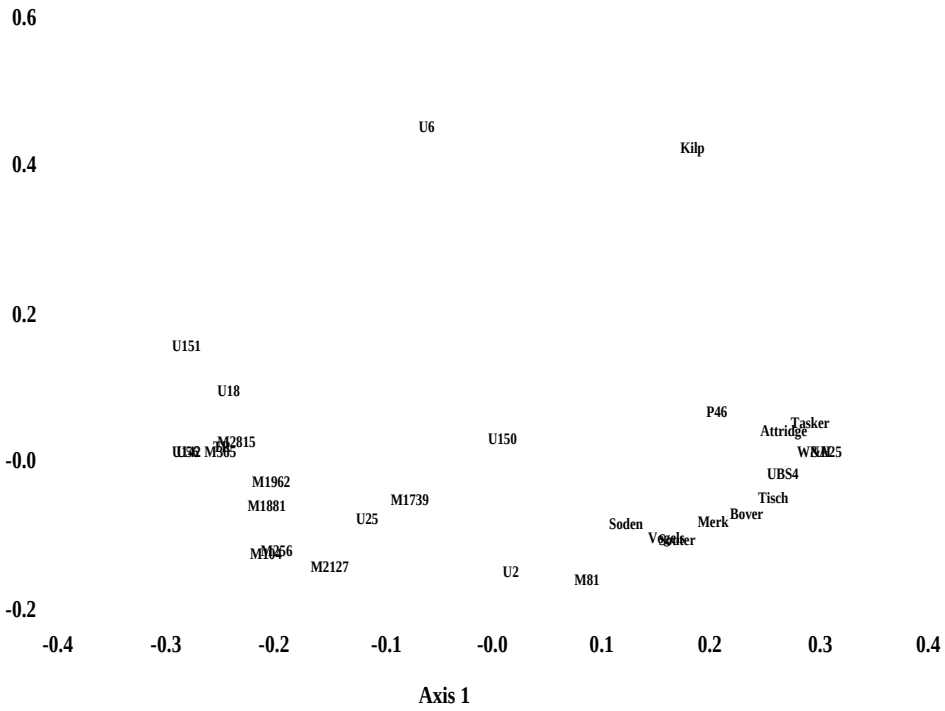
M1739 (950?)

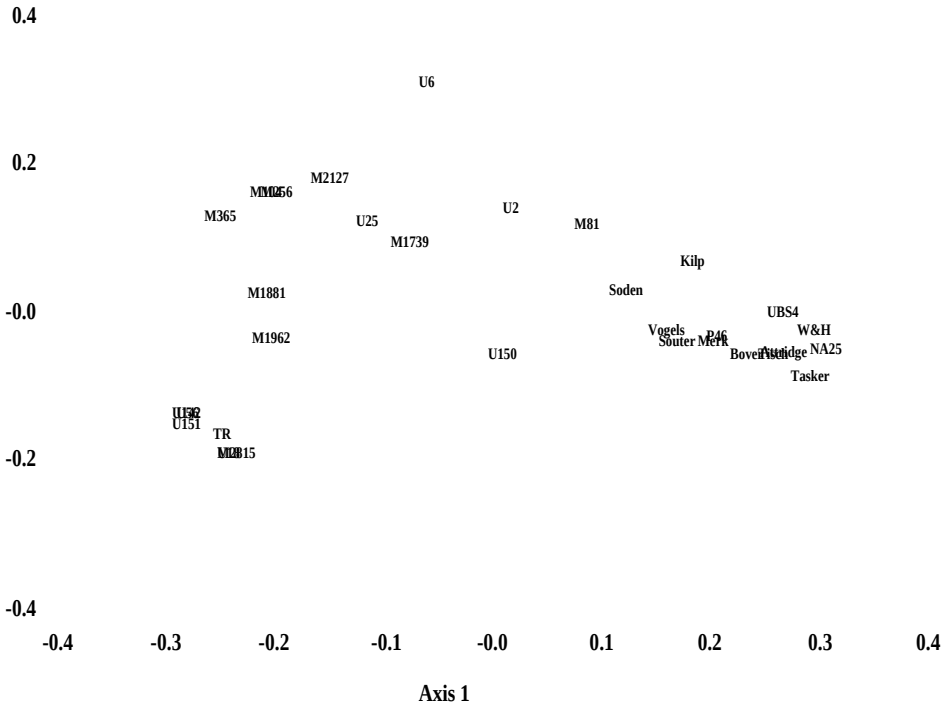




71 units; 30%, 12%, 10%

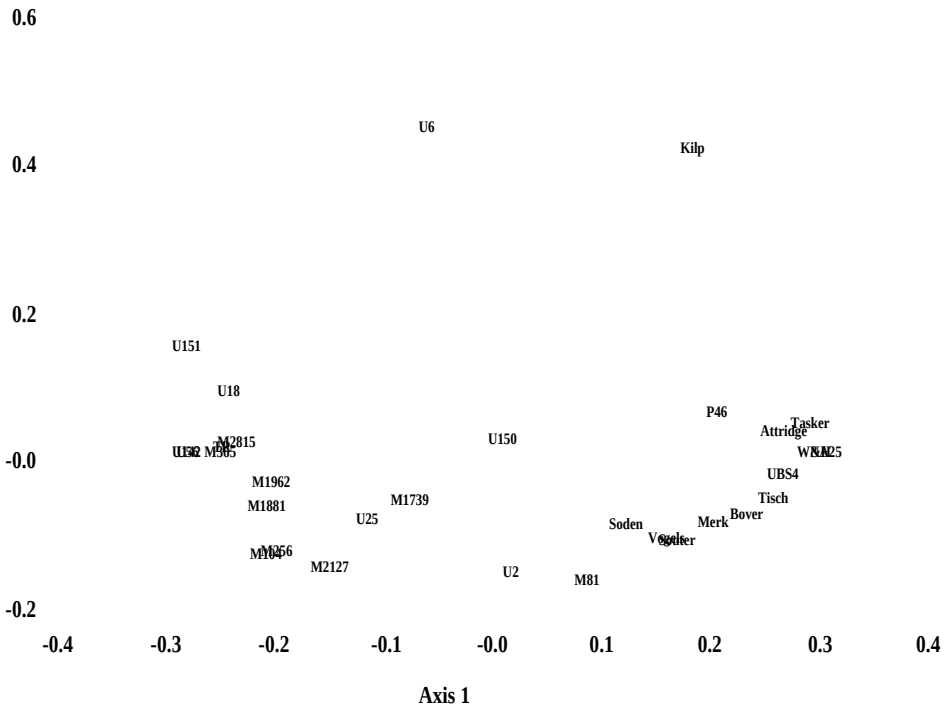
M1881 (1350?)

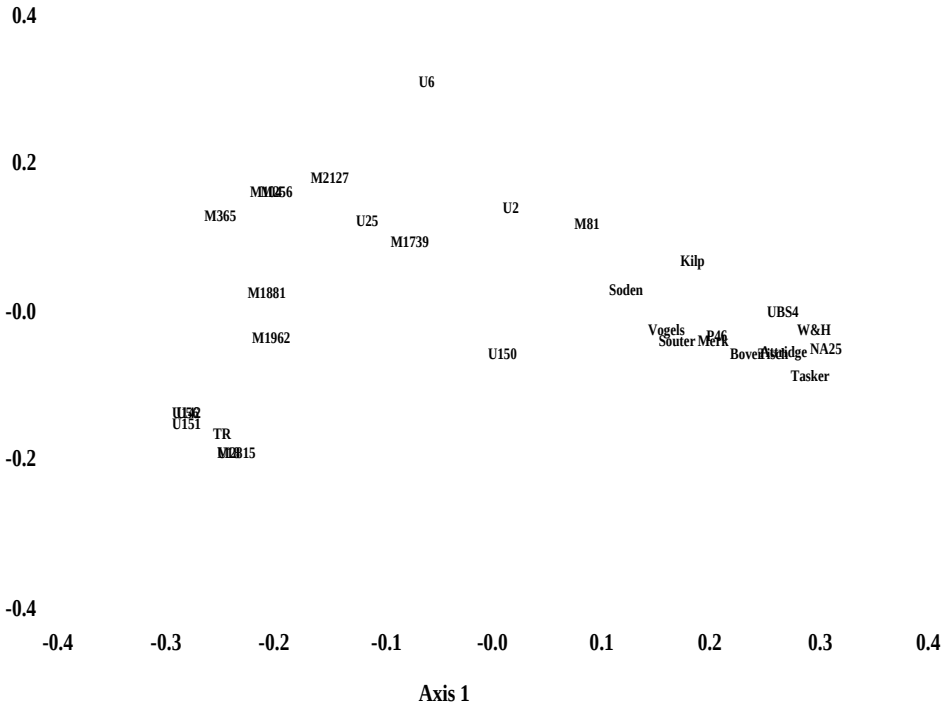




71 units; 30%, 12%, 10%

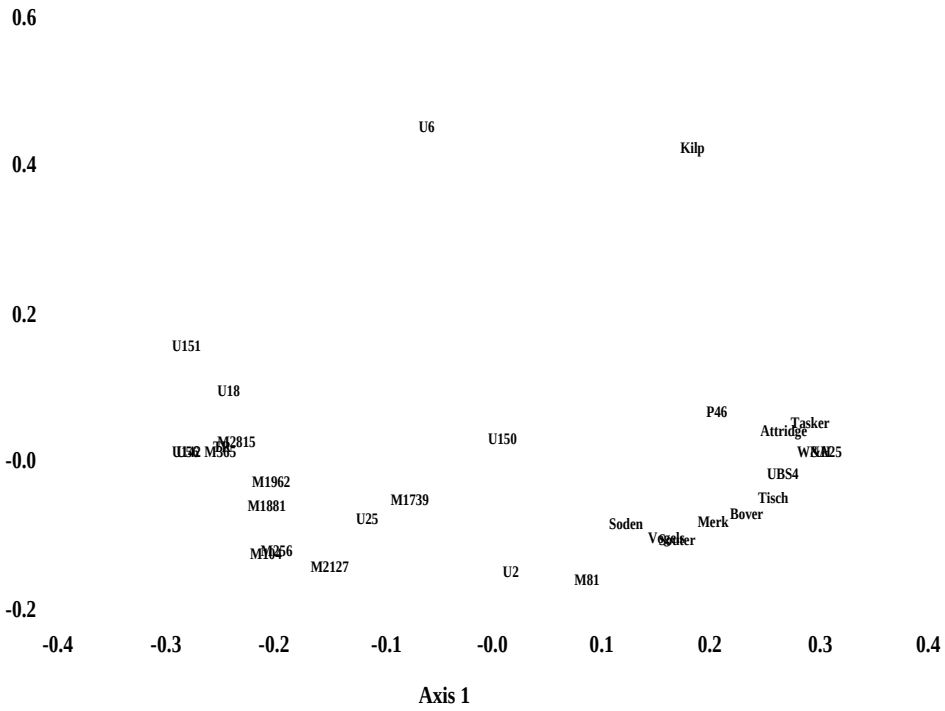
M1962 (1100?)

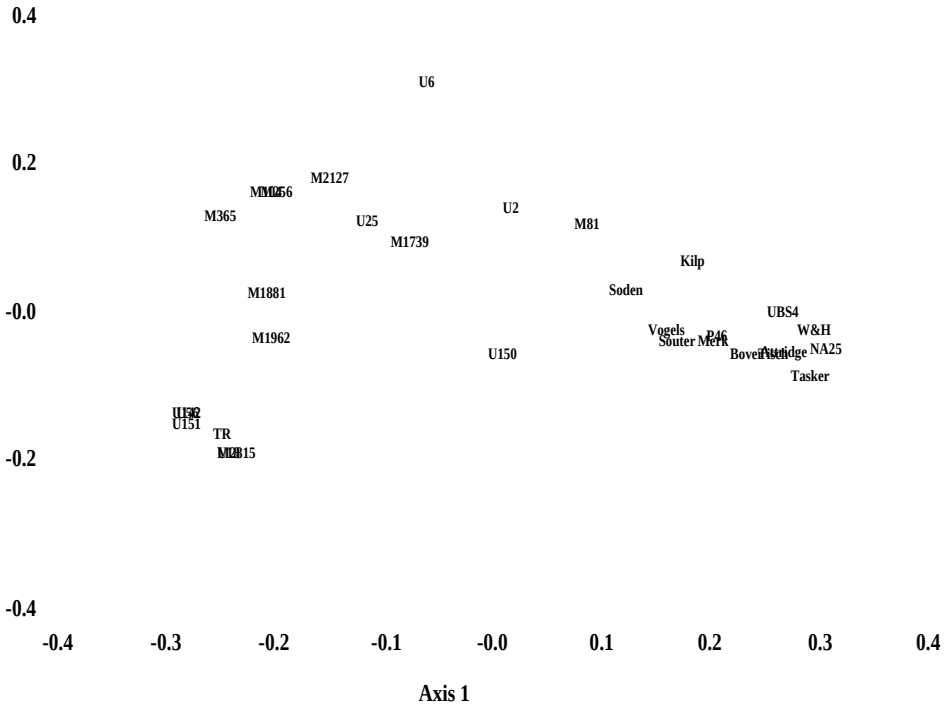




71 units; 30%, 12%, 10%

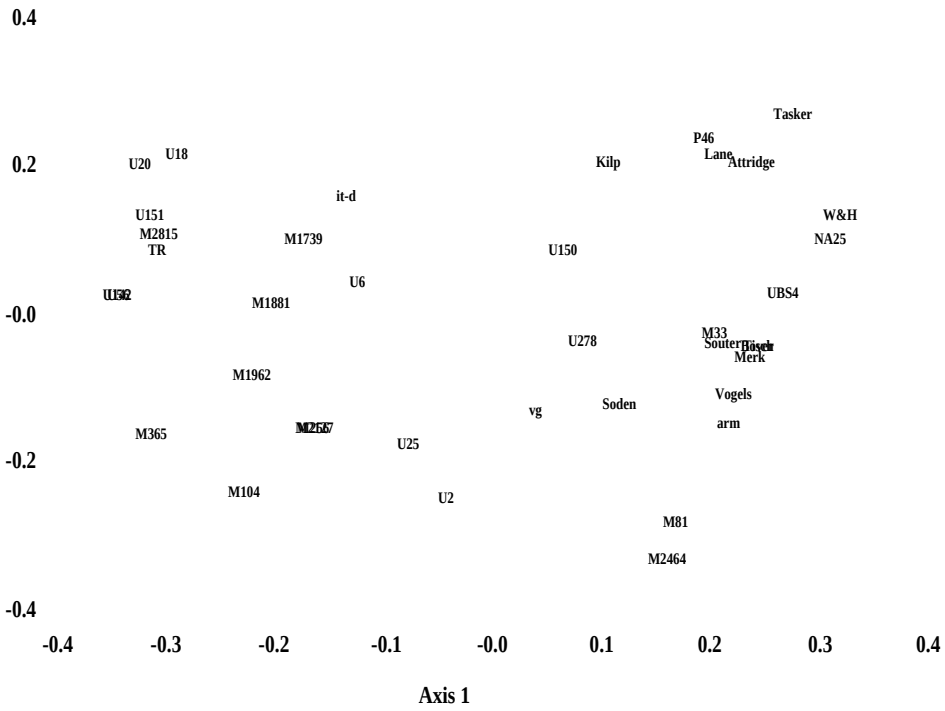
M2127 (1150?)

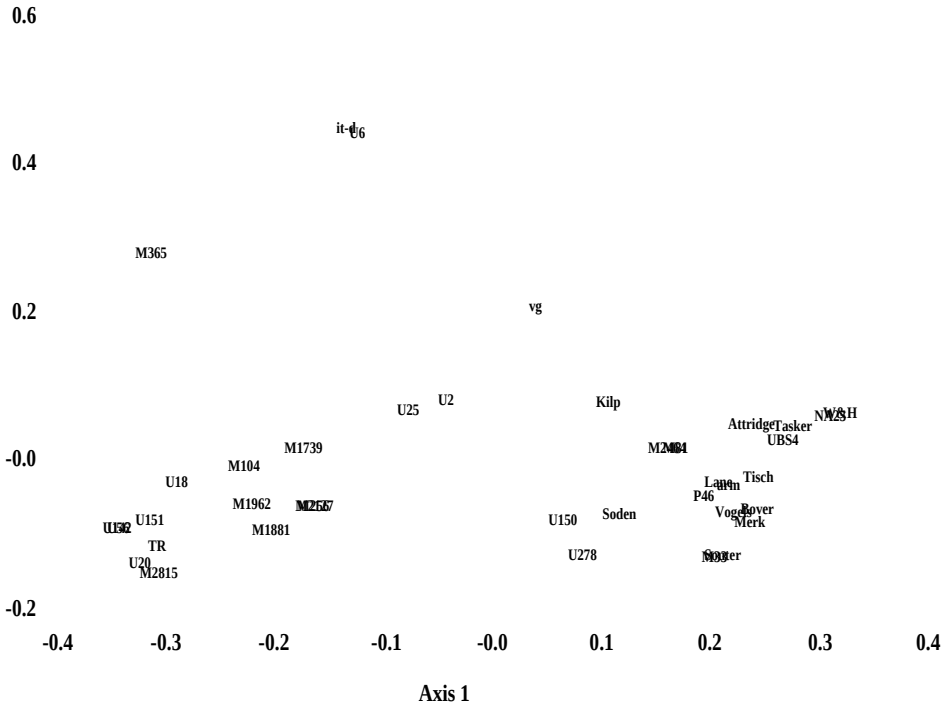




71 units; 30%, 12%, 10%

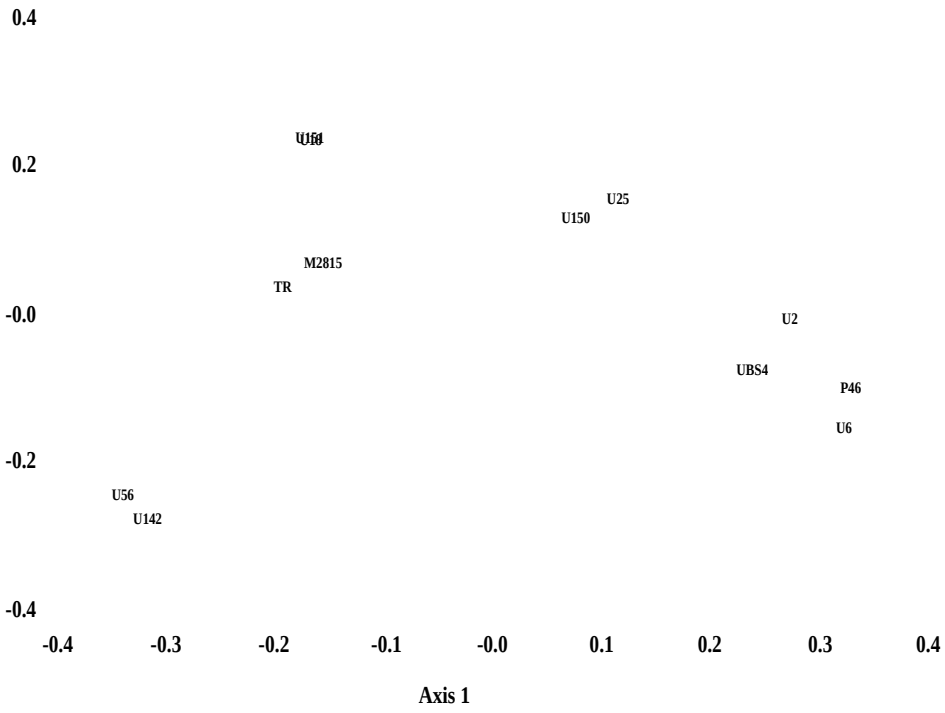
M2464 (850?)

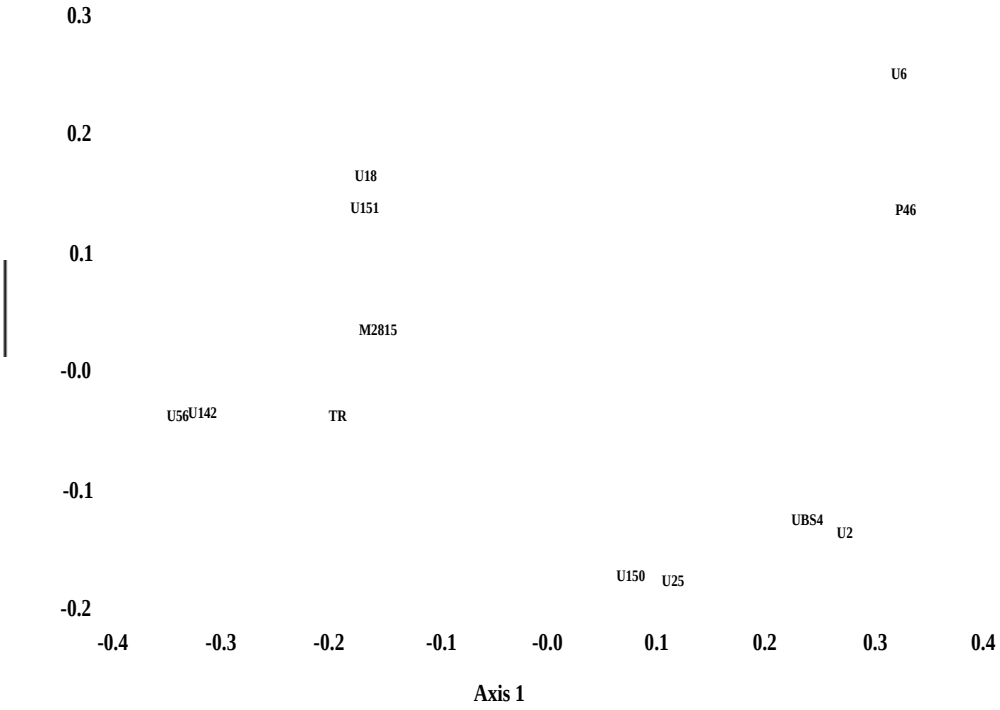




34 units; 29%, 14%, 11%

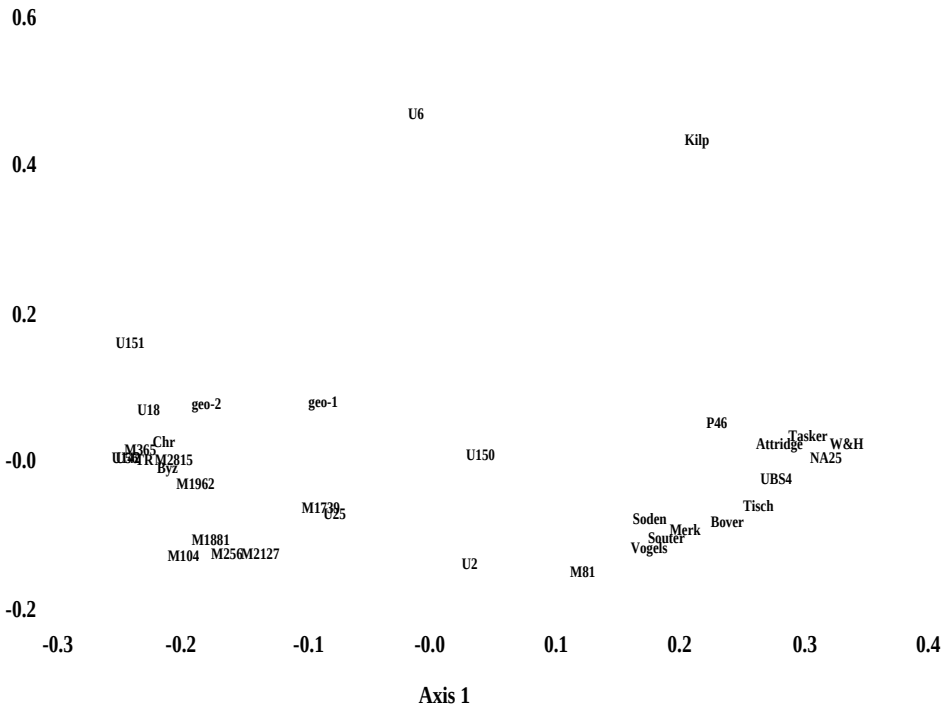
M2815 (1150?)

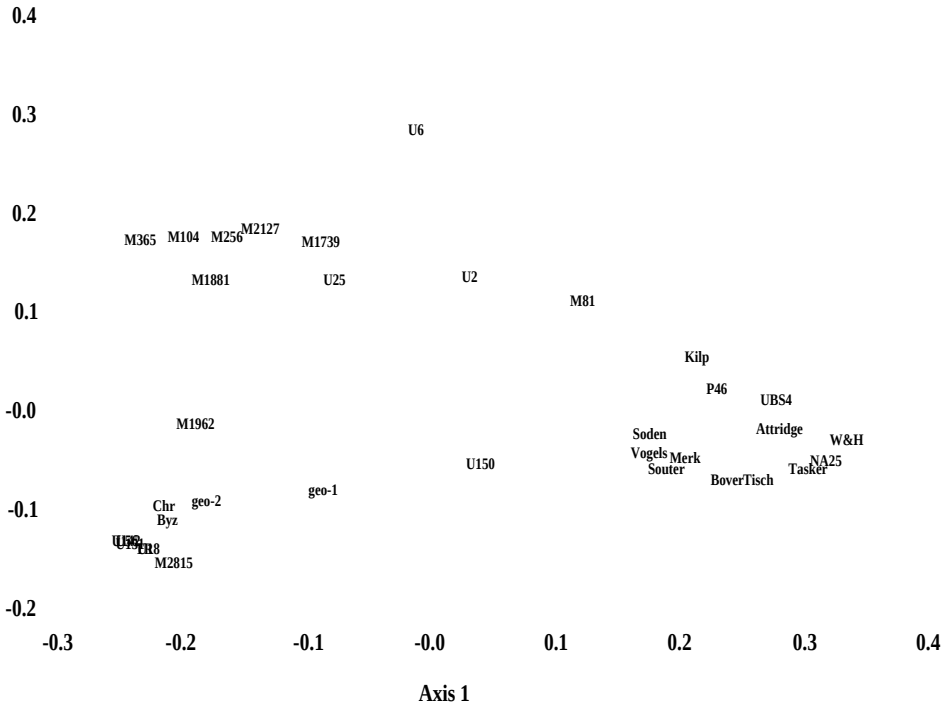




728 units; 32%, 15%, 10%

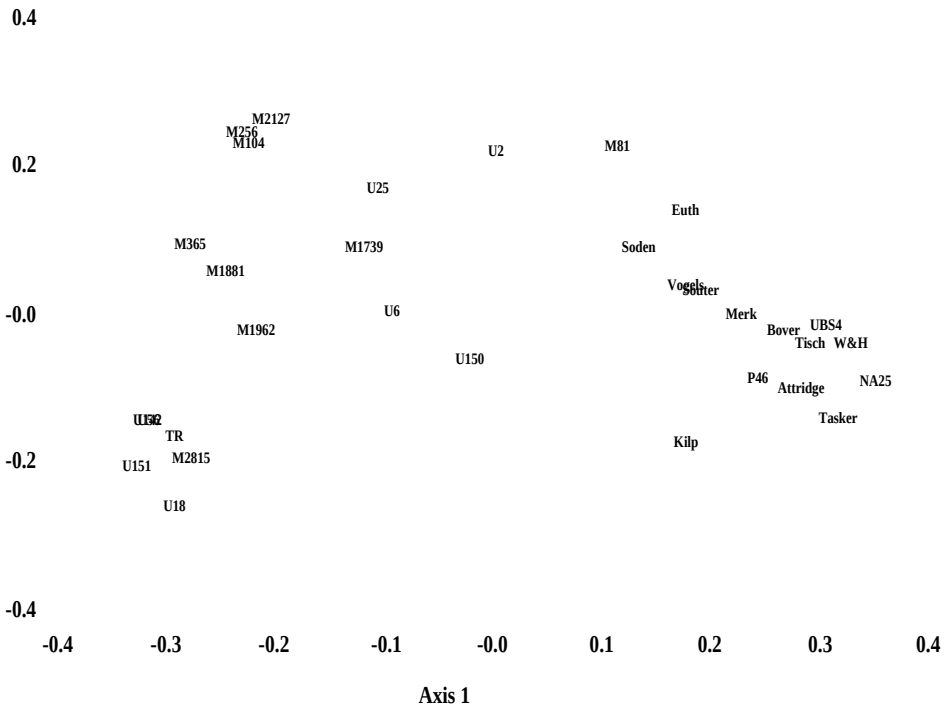
Byzantine (Lucian? Antioch, 300?)

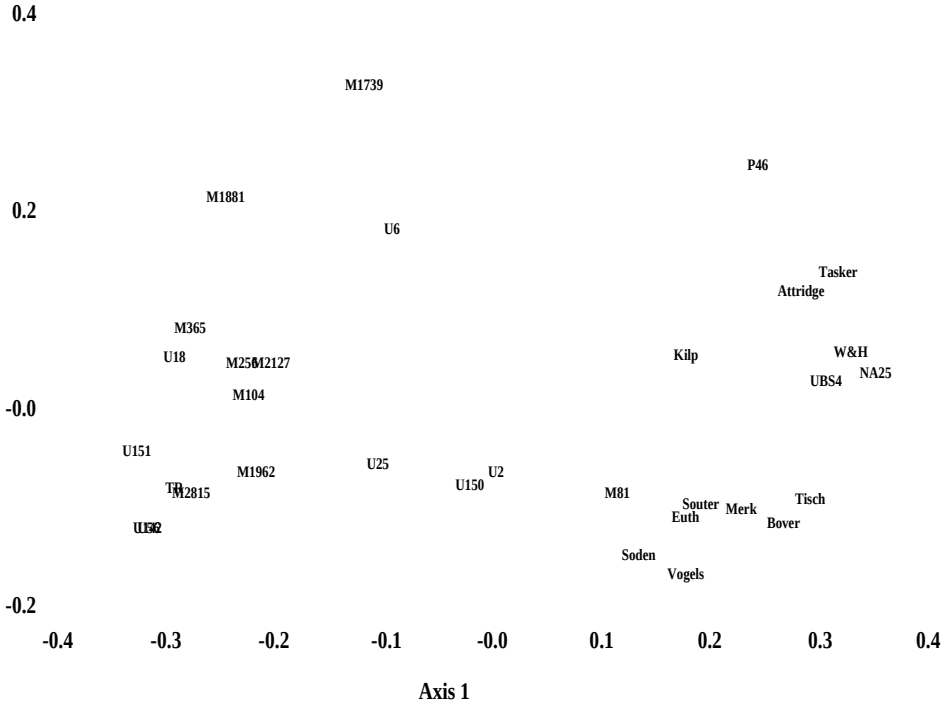




64 units; 31%, 13%, 10%

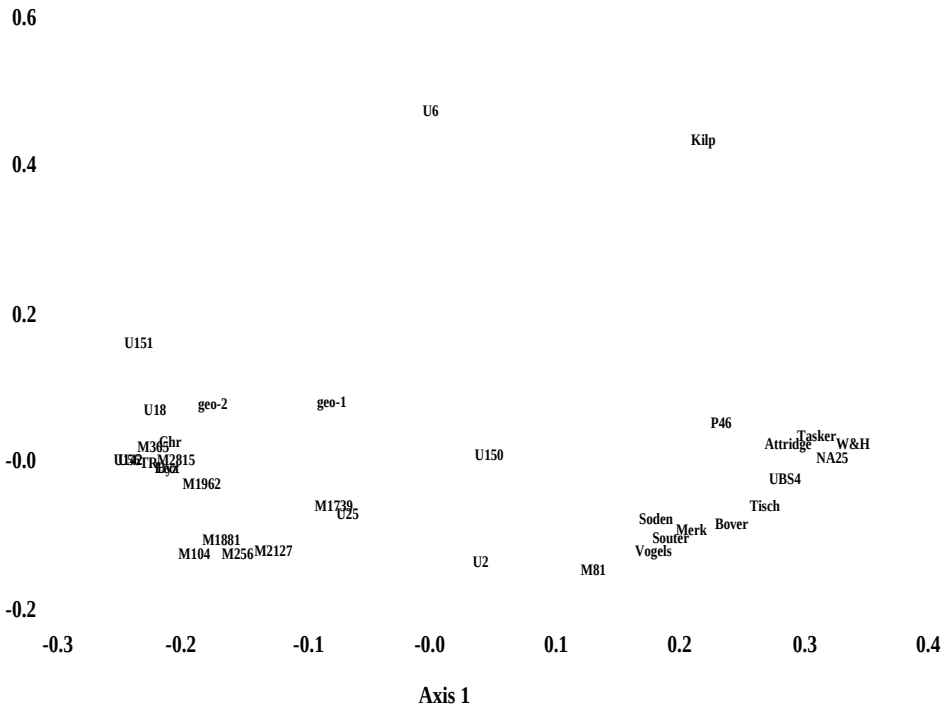
Euthalian manuscripts (Pamphilus? Caesarea, 305?)

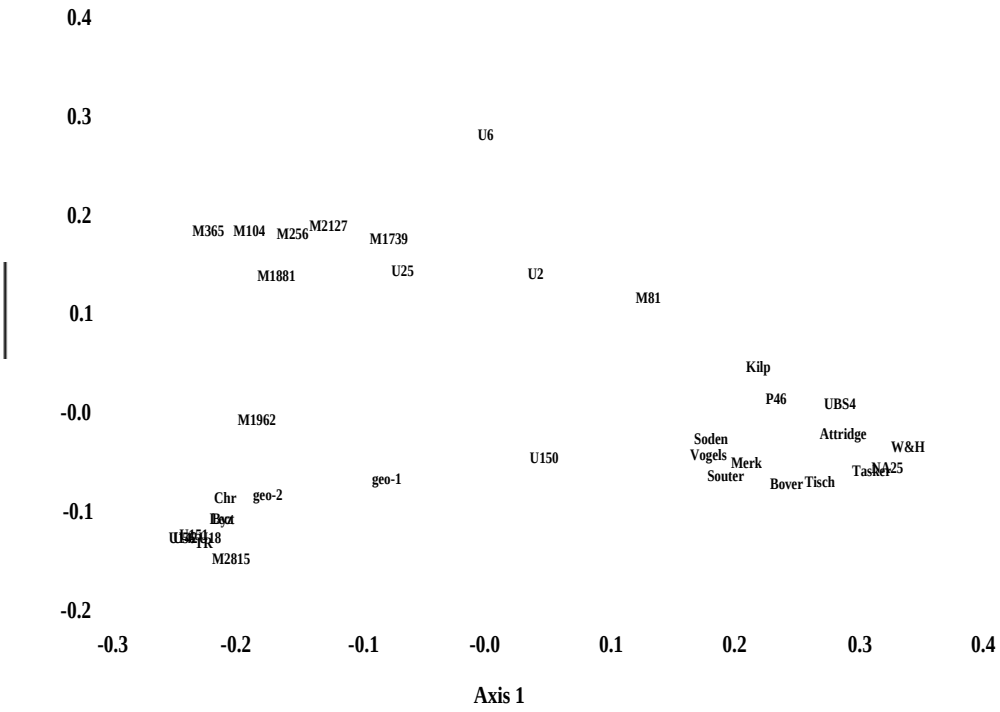




51 units; 33%, 12%, 8%

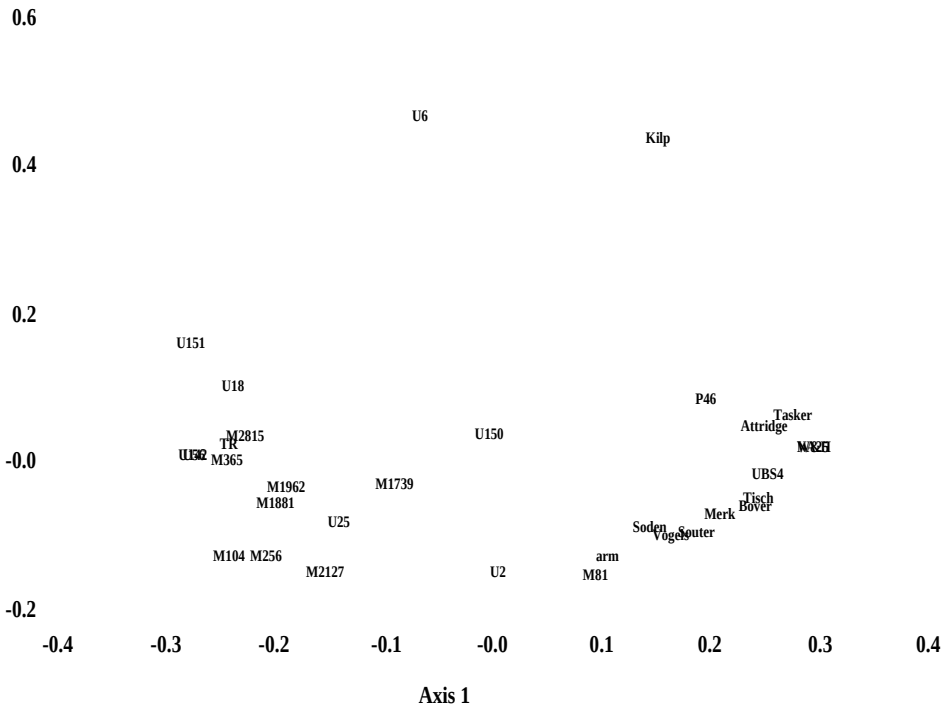
Lectionaries

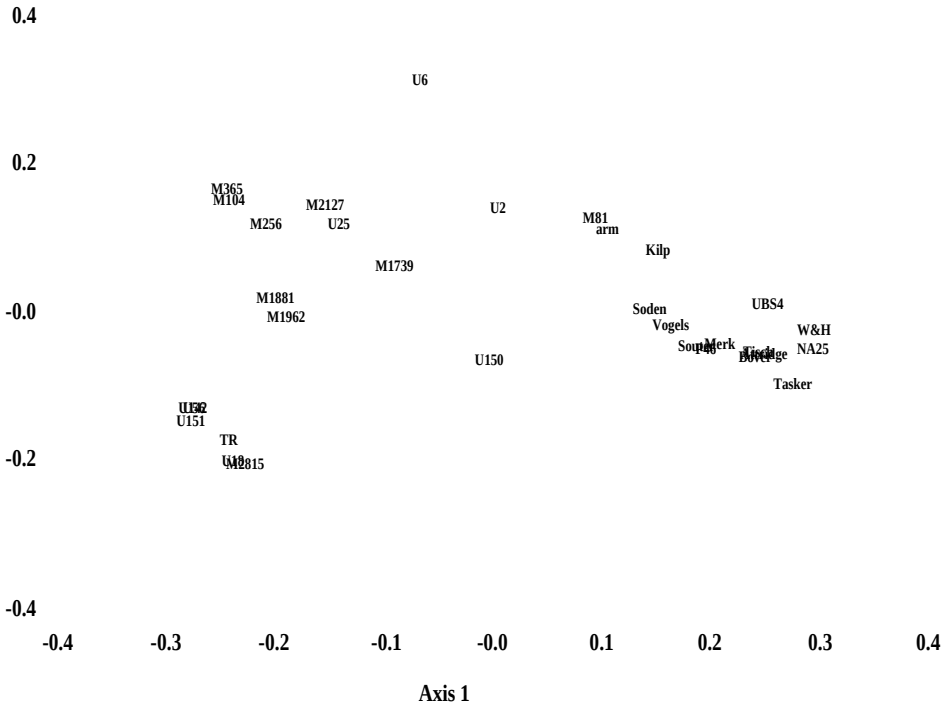




64 units; 31%, 12%, 10%

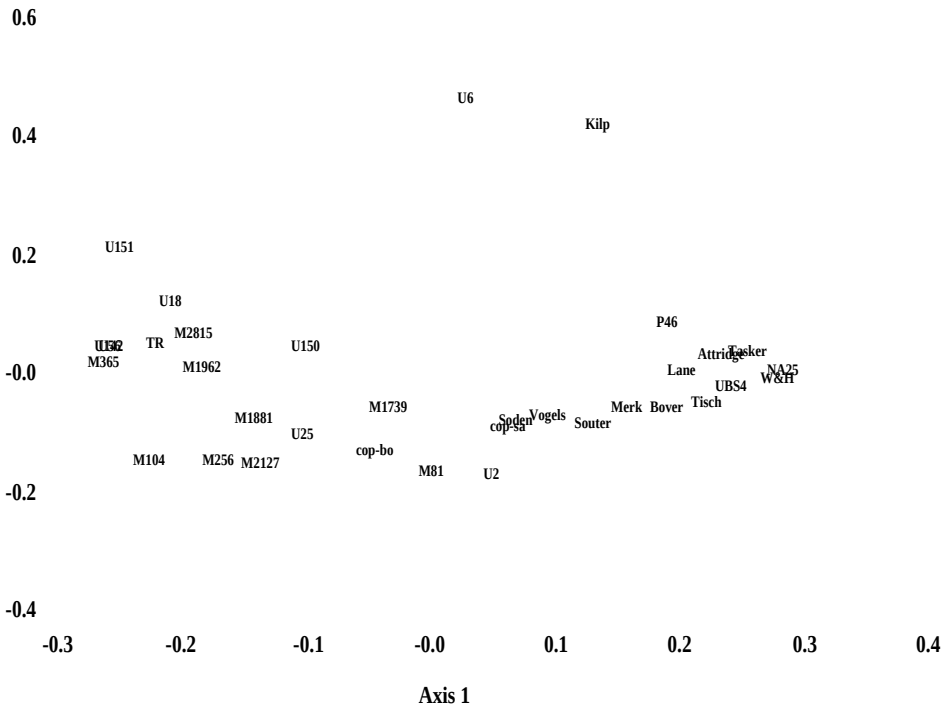
Armenian (Mesrop, Armenia, 410?)

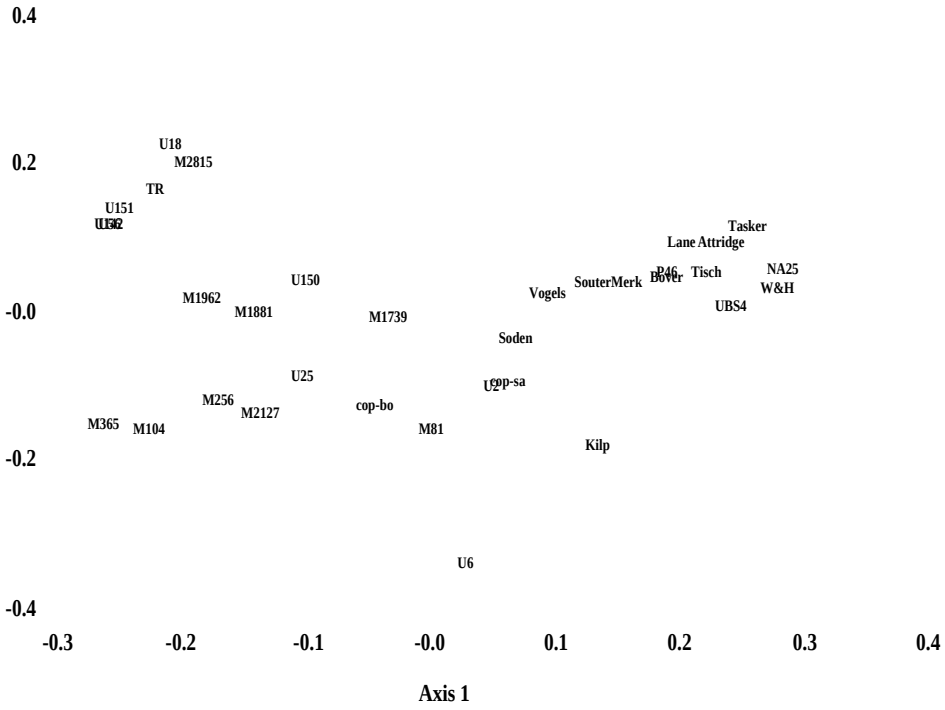




66 units; 30%, 13%, 10%

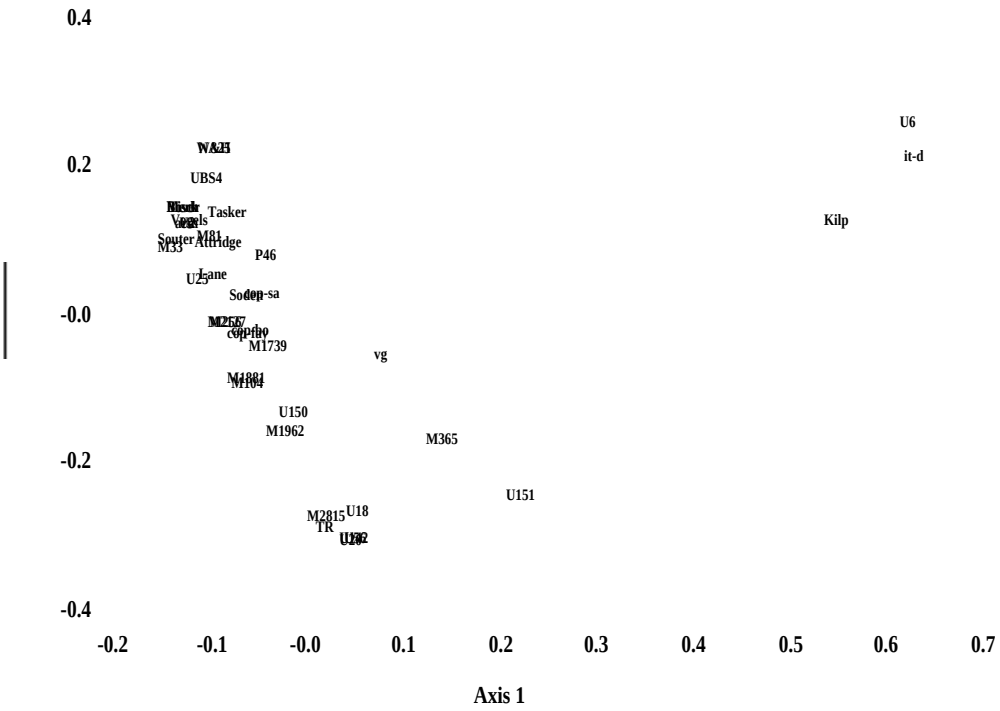
Coptic-Bohairic (Northern Egypt, 300?)

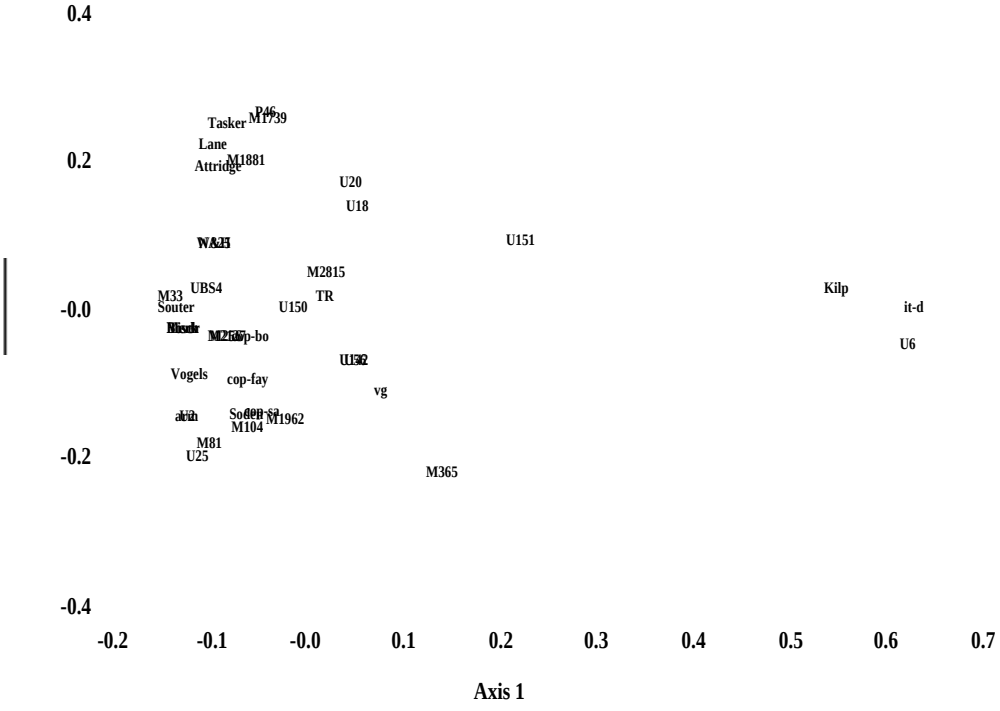




58 units; 25%, 14%, 11%

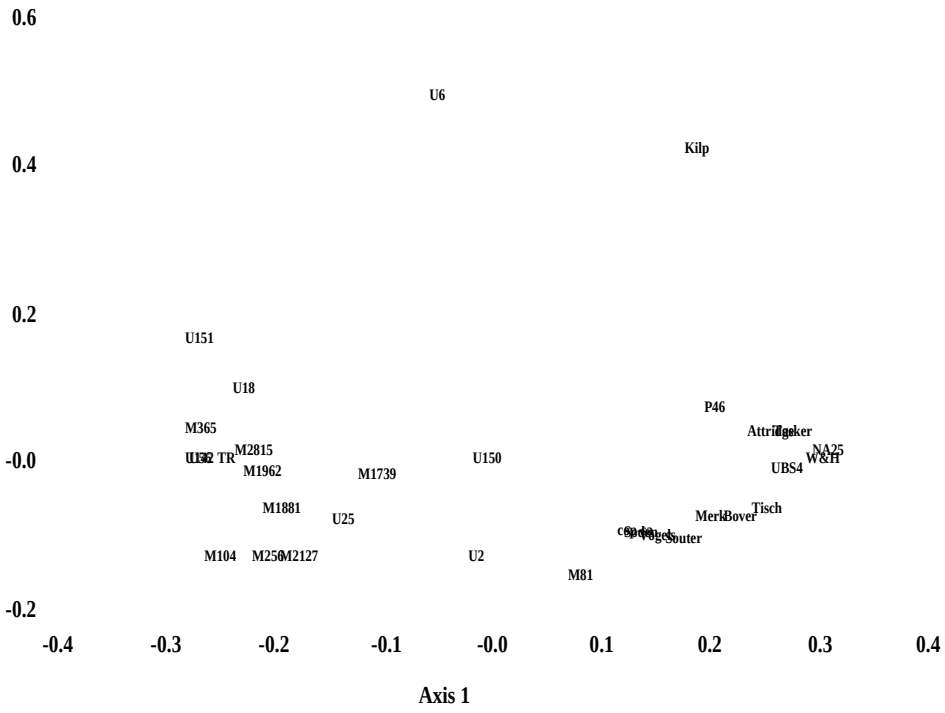
Coptic-Fayyumic (Fayyum, 325?)

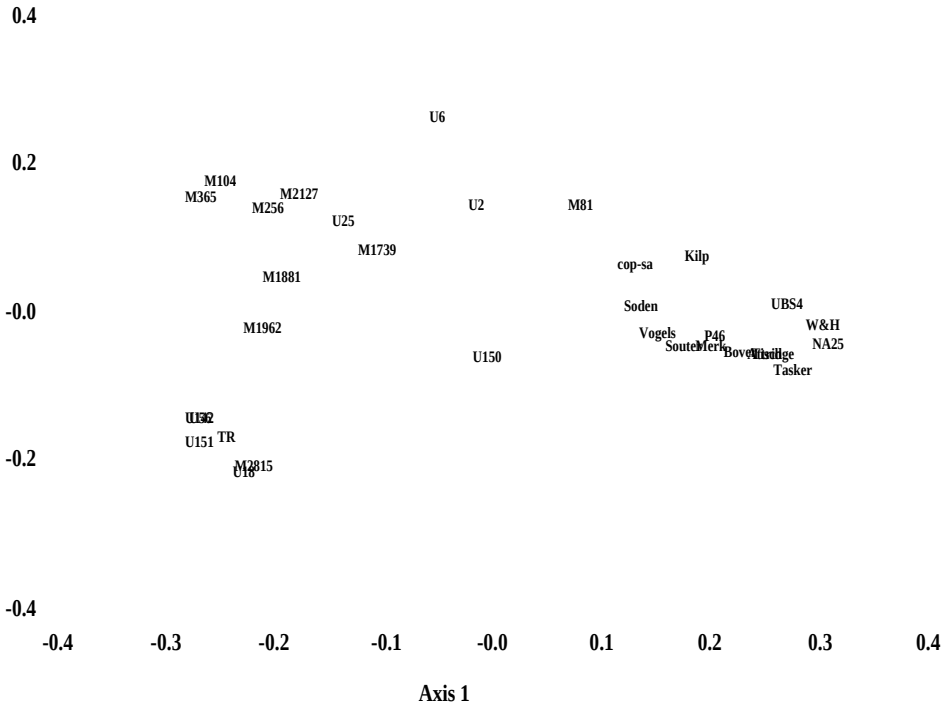




35 units; 26%, 21%, 13%

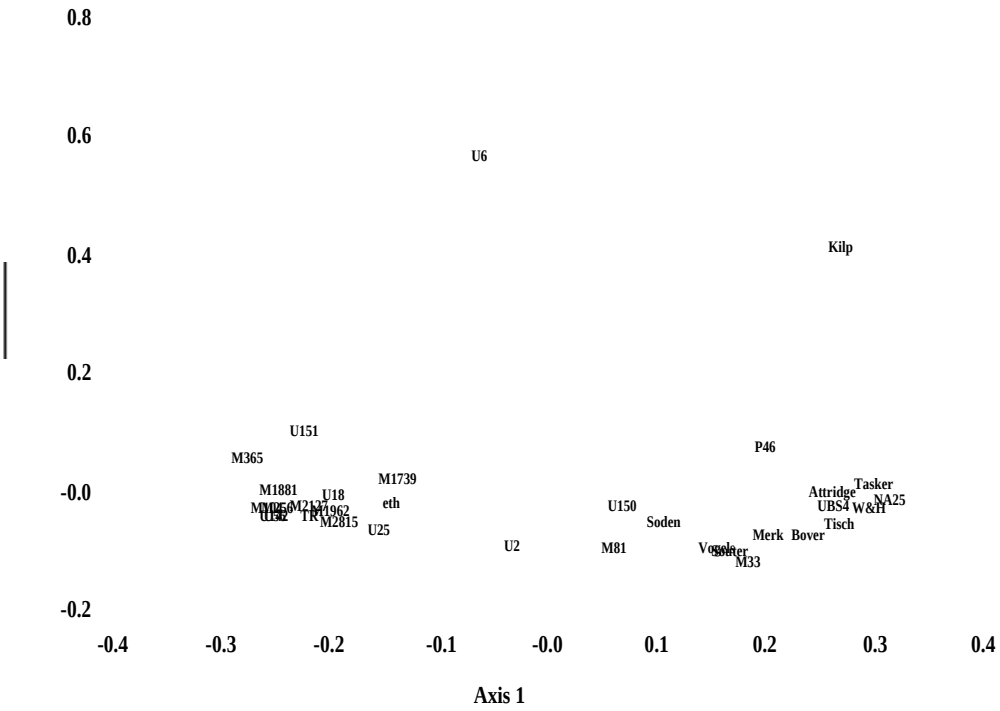
Coptic-Sahidic (Southern Egypt, 200?)

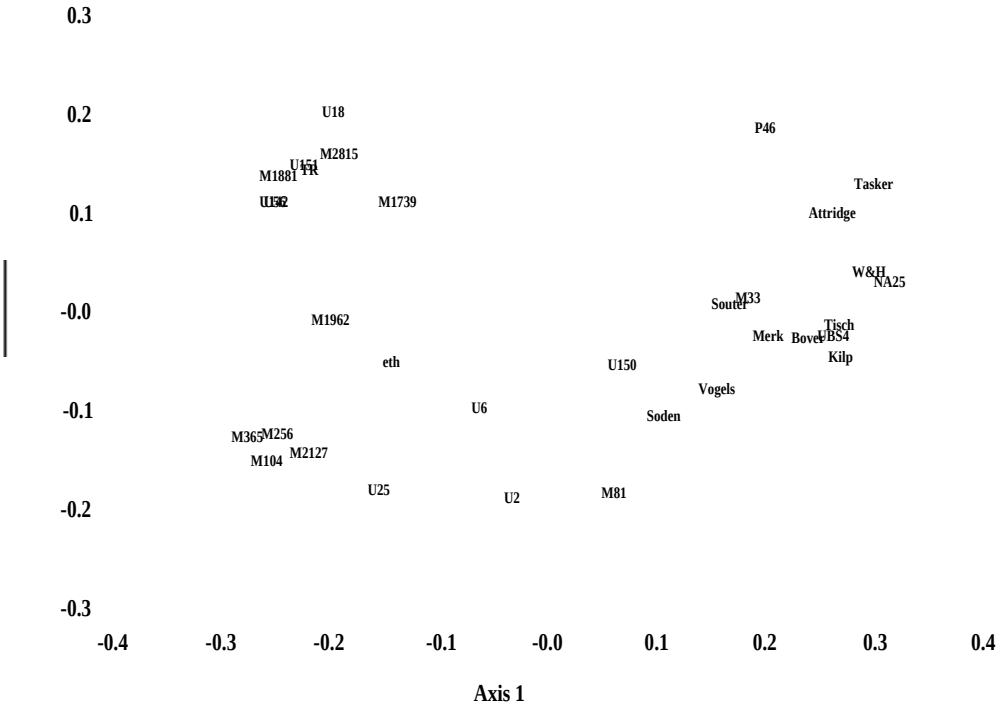




63 units; 30%, 13%, 10%

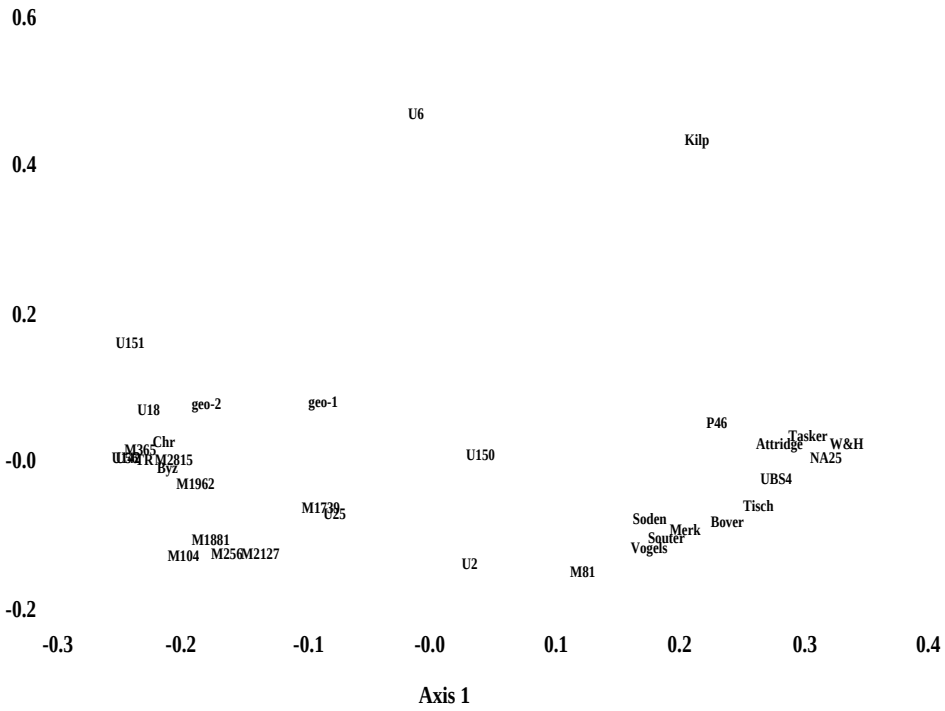
Ethiopic (Ethiopia, 500?)

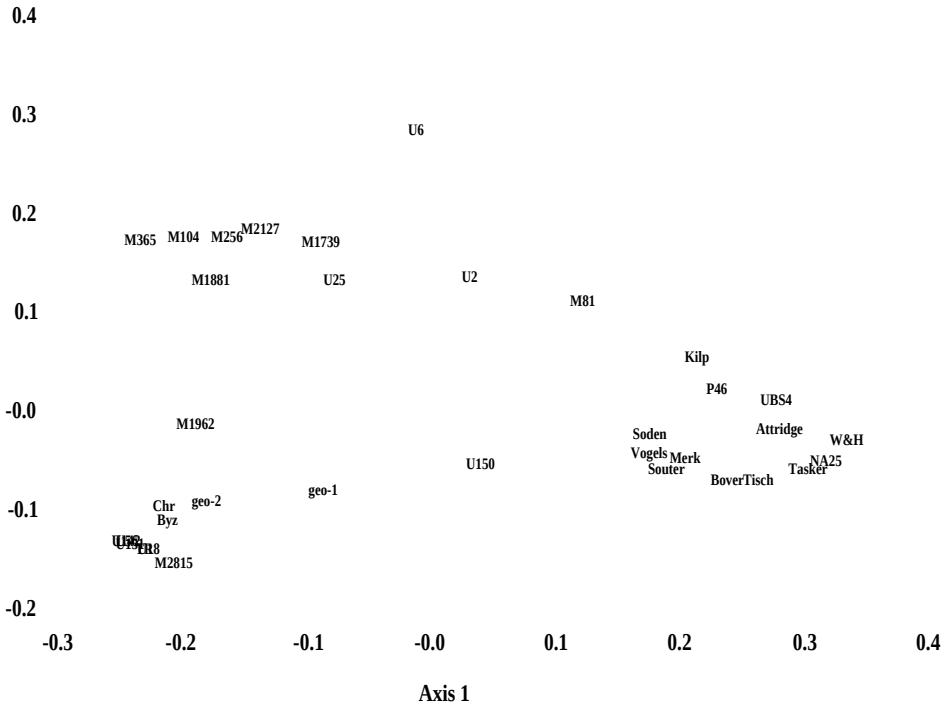




61 units; 31%, 12%, 9%

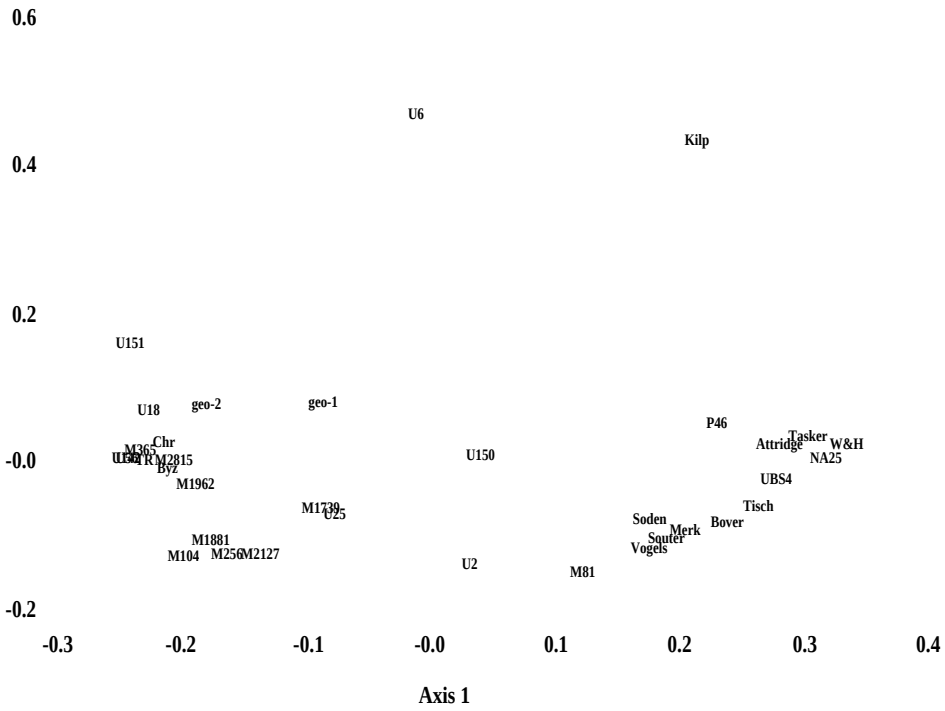
Georgian-1 (Georgia, 425?)

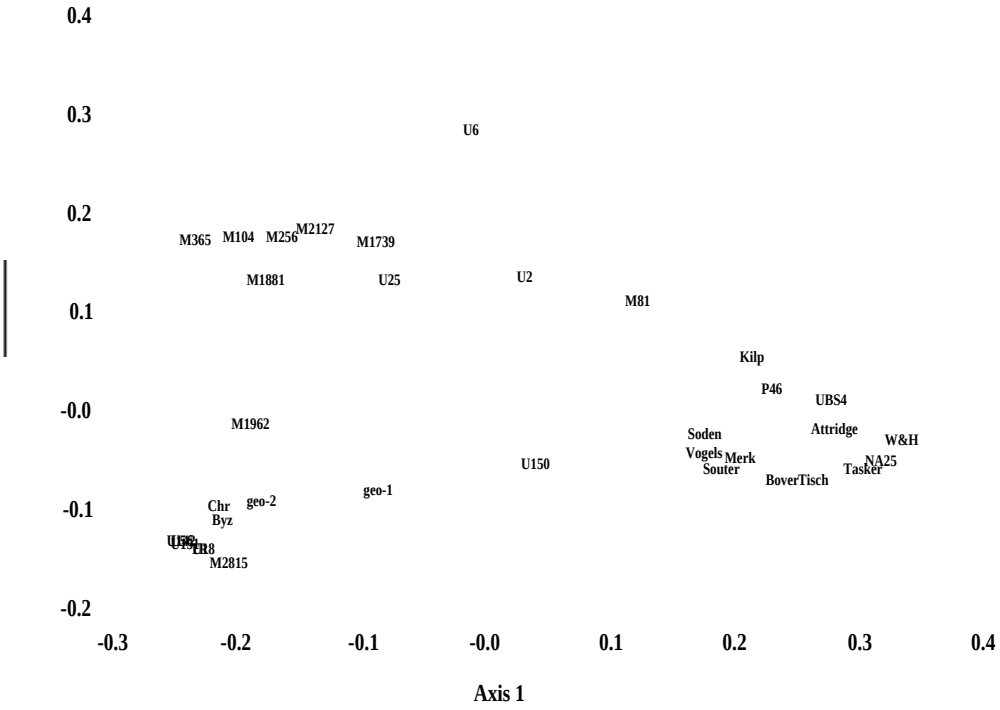




64 units; 31%, 13%, 10%

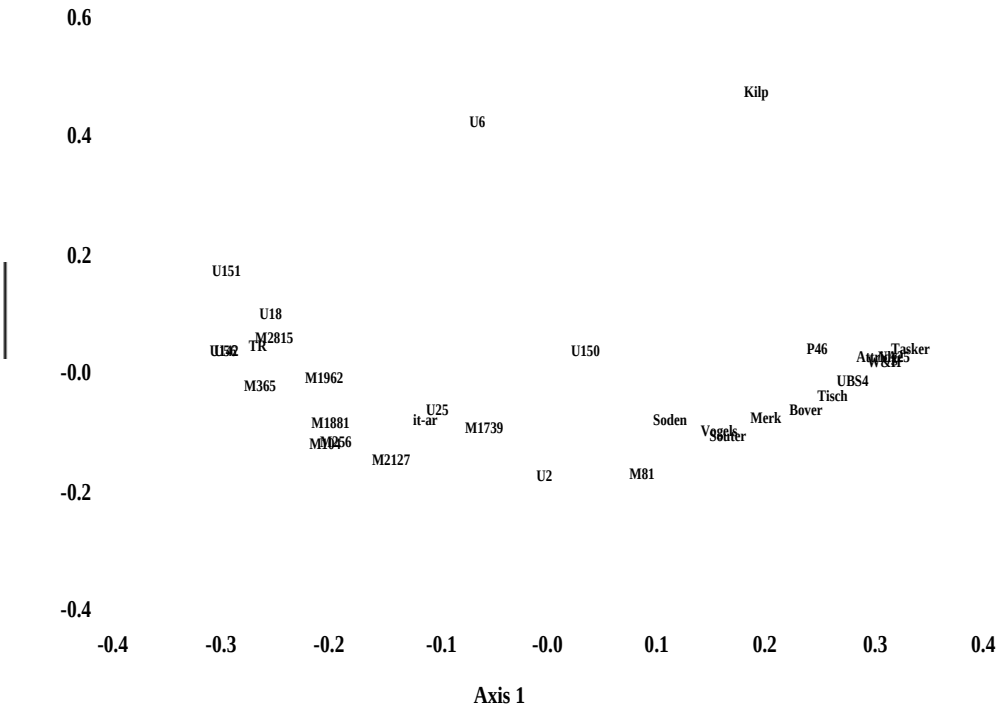
Georgian-2 (Georgia, 600?)

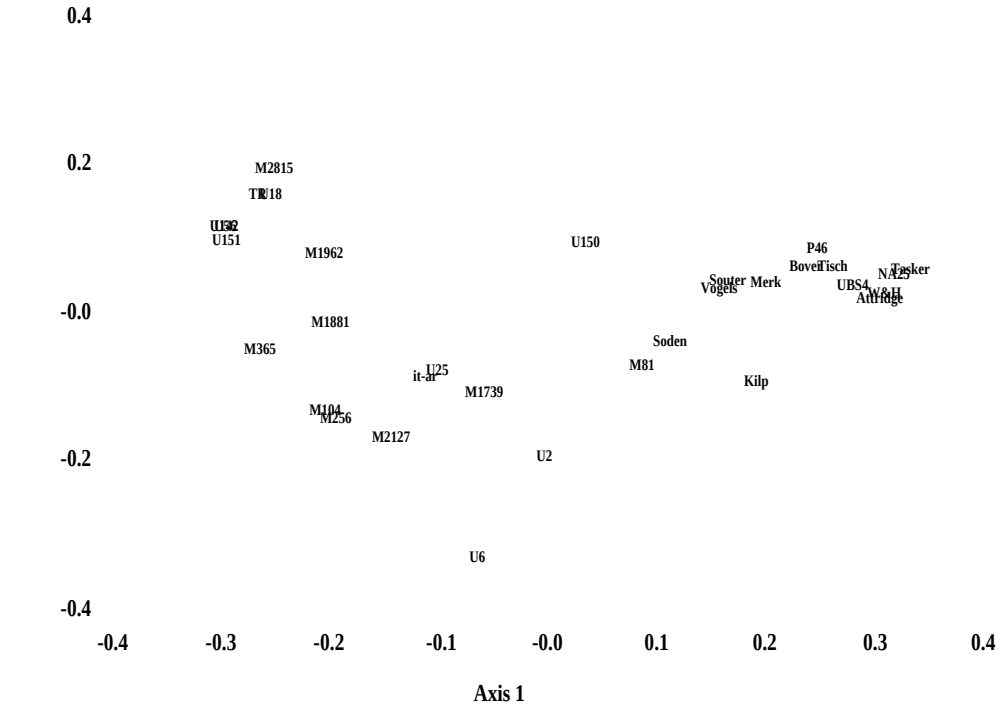




64 units; 31%, 13%, 10%

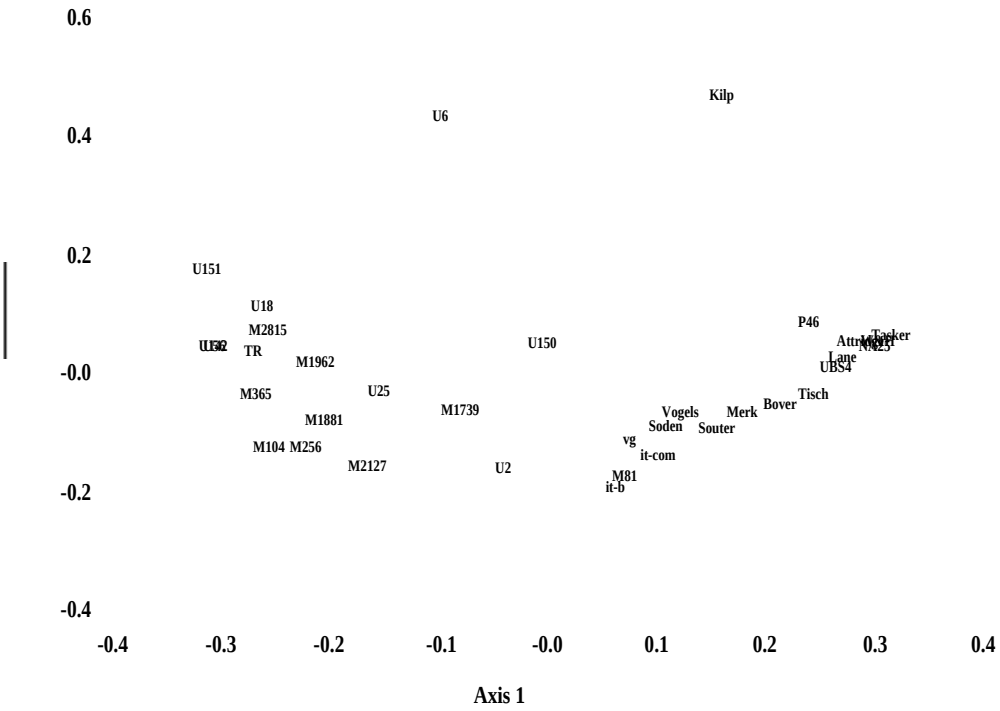
Latin 61: it-ar (850?)

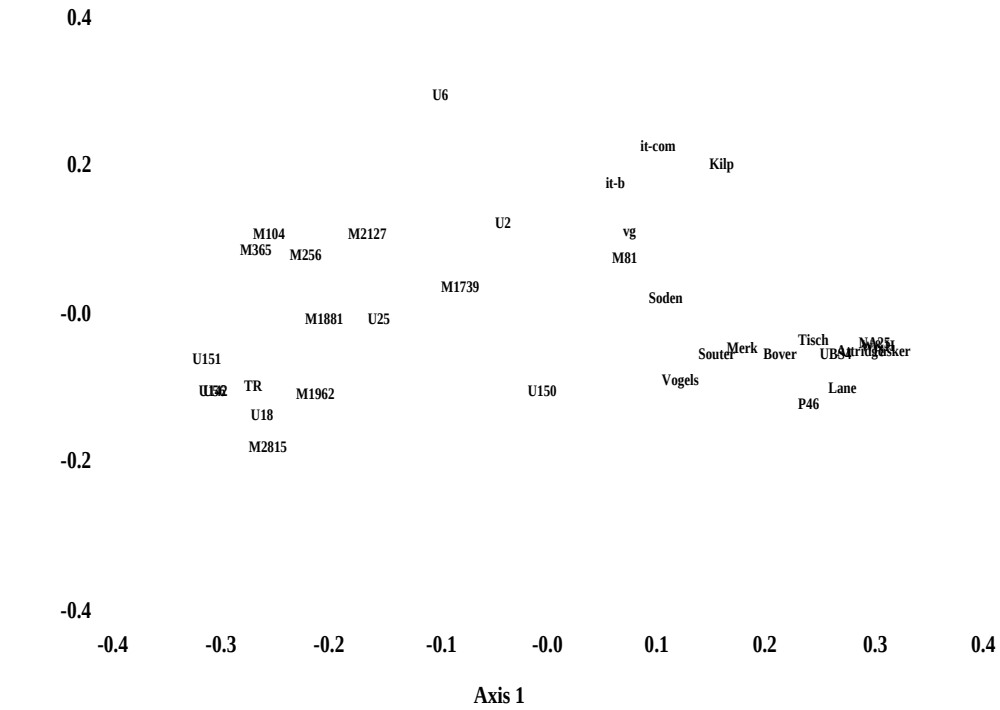




62 units; 32%, 13%, 9%

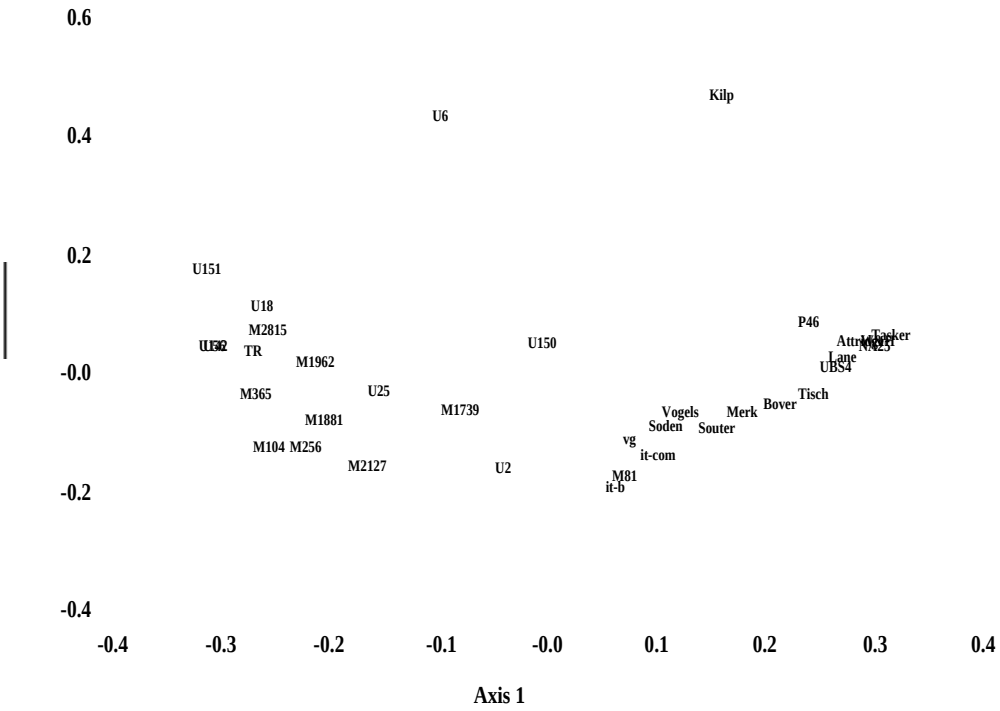
Latin 89: it-b (800?)

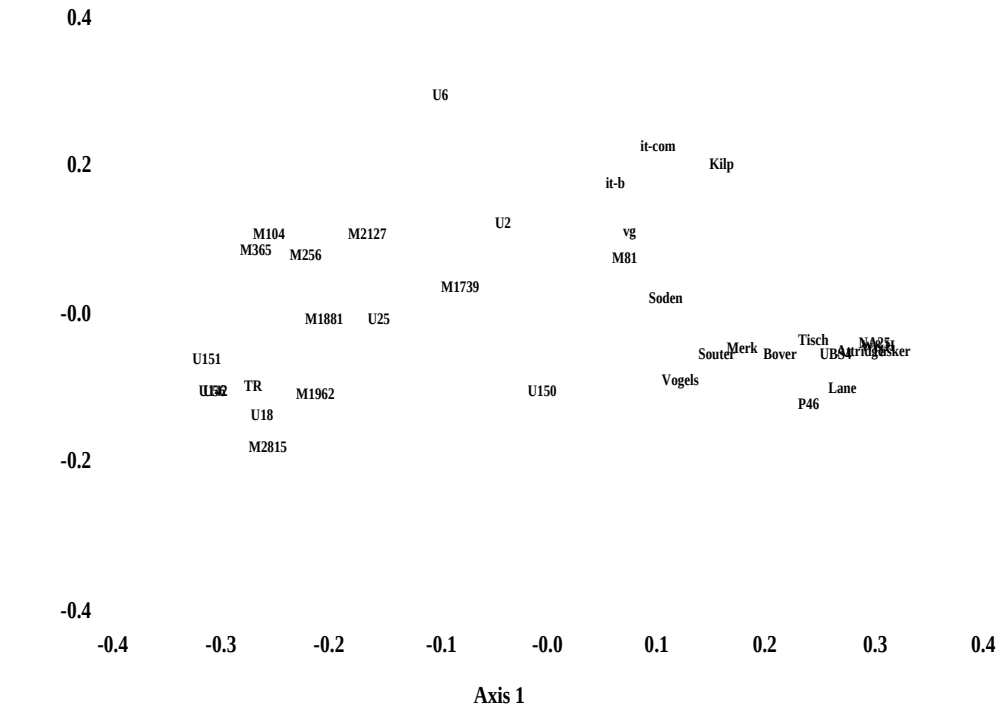




58 units; 31%, 14%, 9%

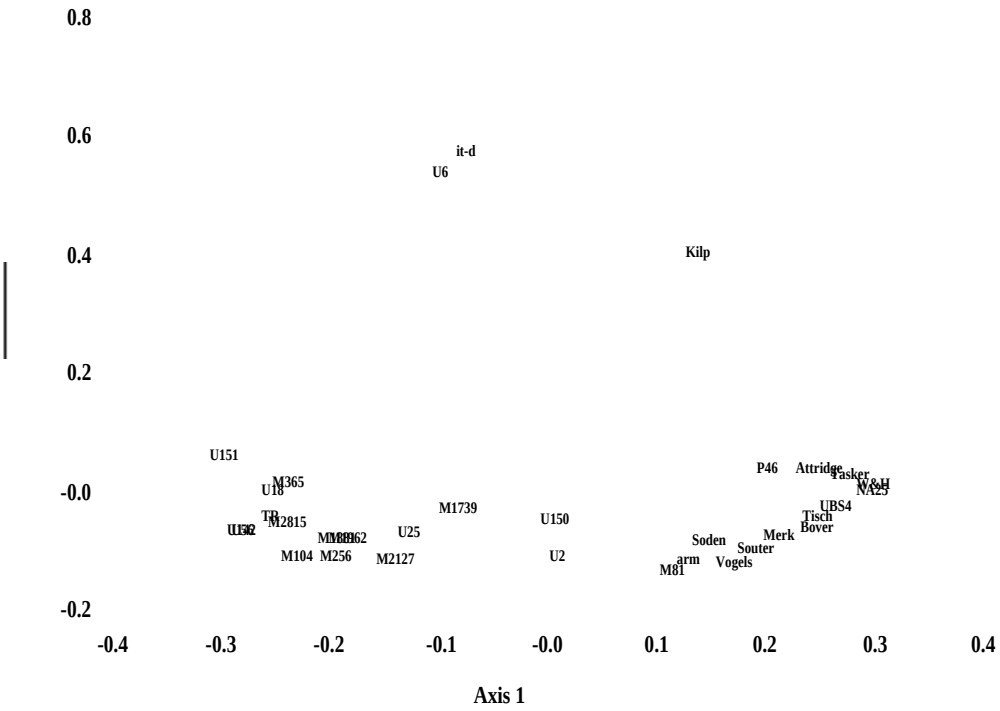
Latin 109: it-com (900?)

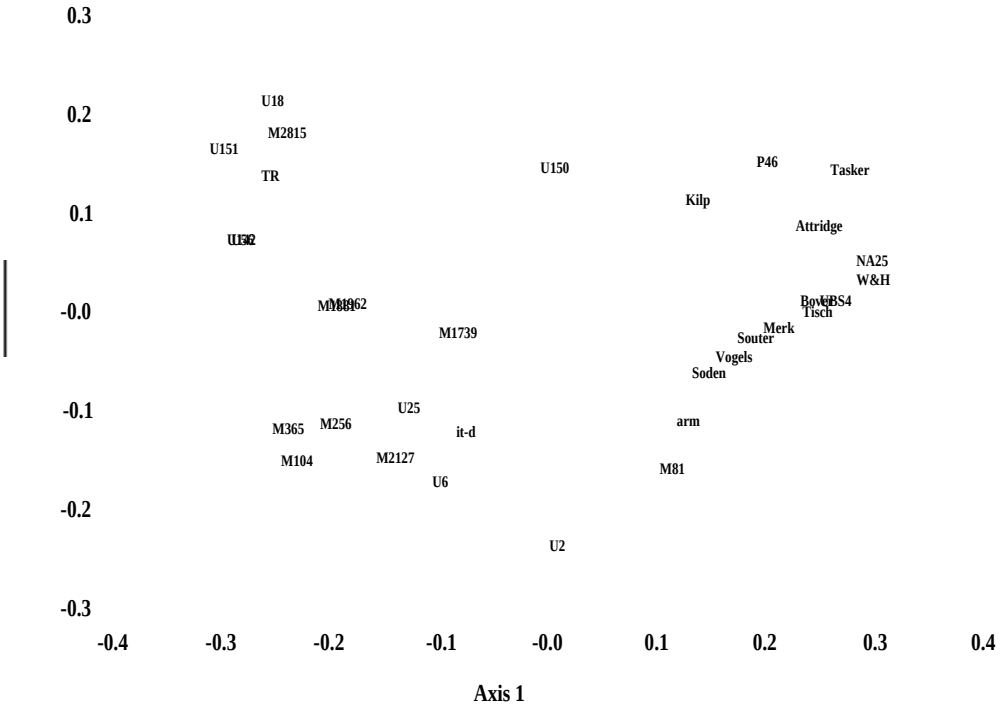




58 units; 31%, 14%, 9%

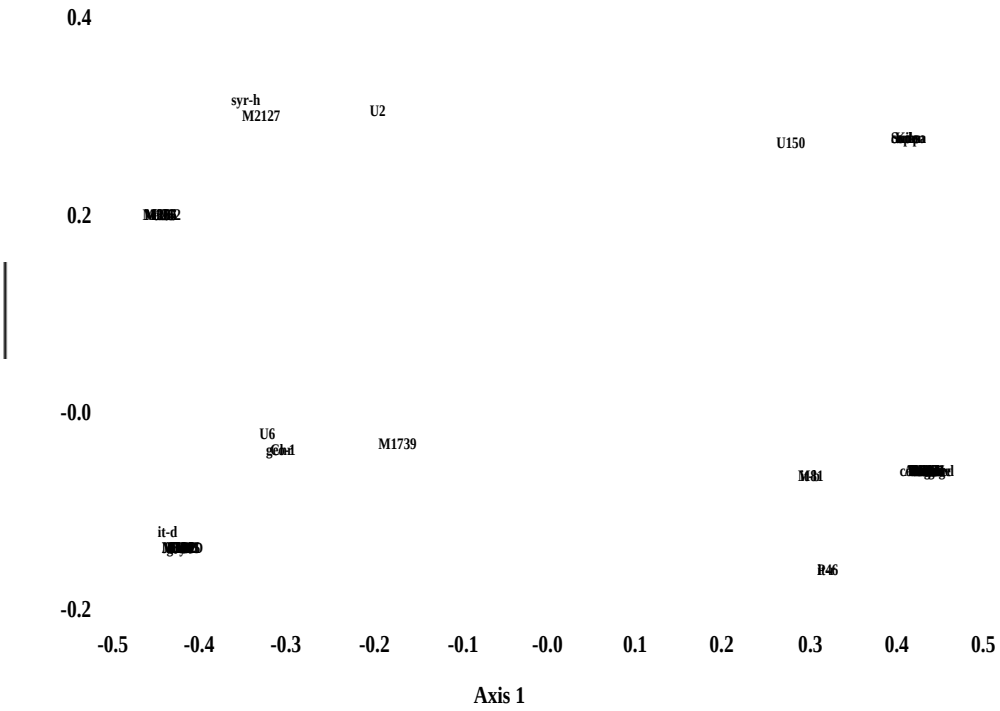
Latin 75: it-d (Sardinia? 500?)

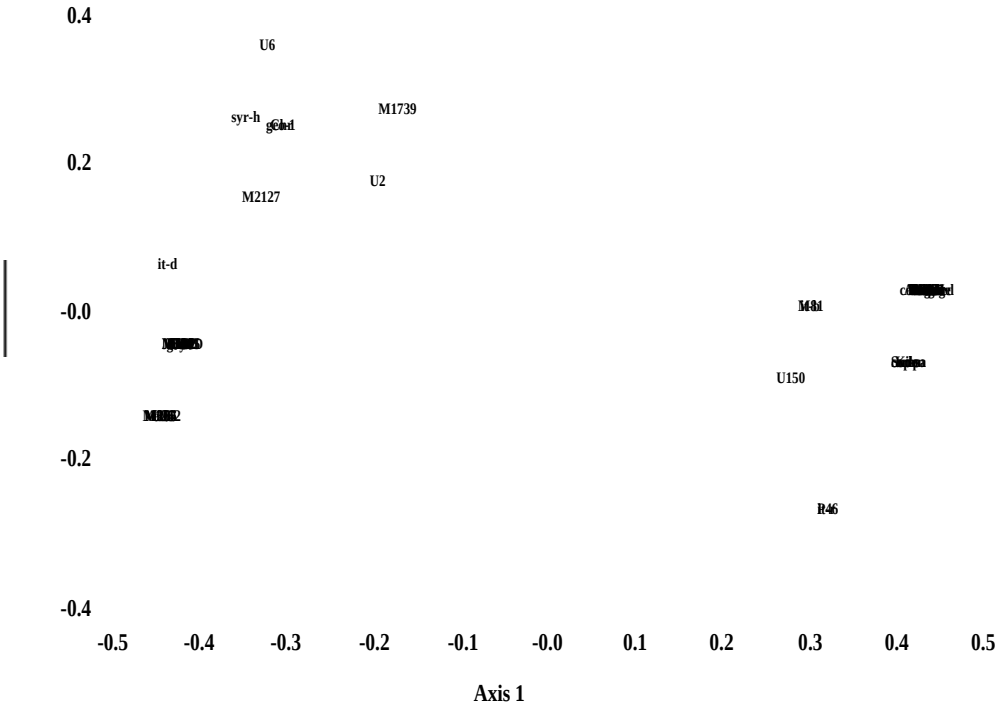




63 units; 29%, 19%, 9%

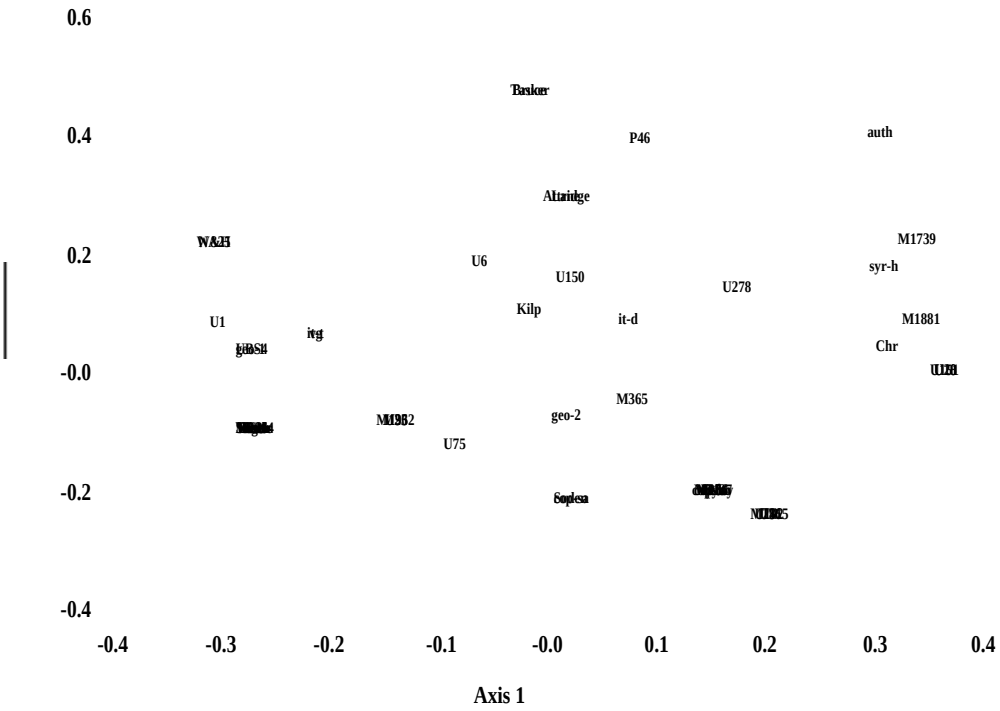
Latin 64: it-r (600?)

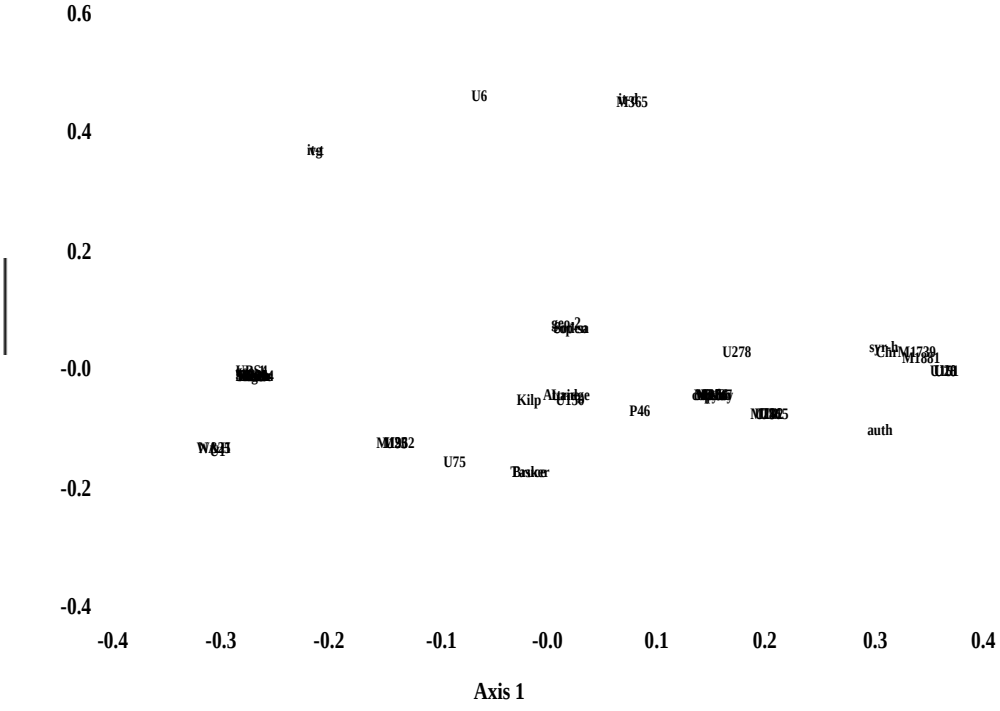




8 units; 75%, 11%, 7%

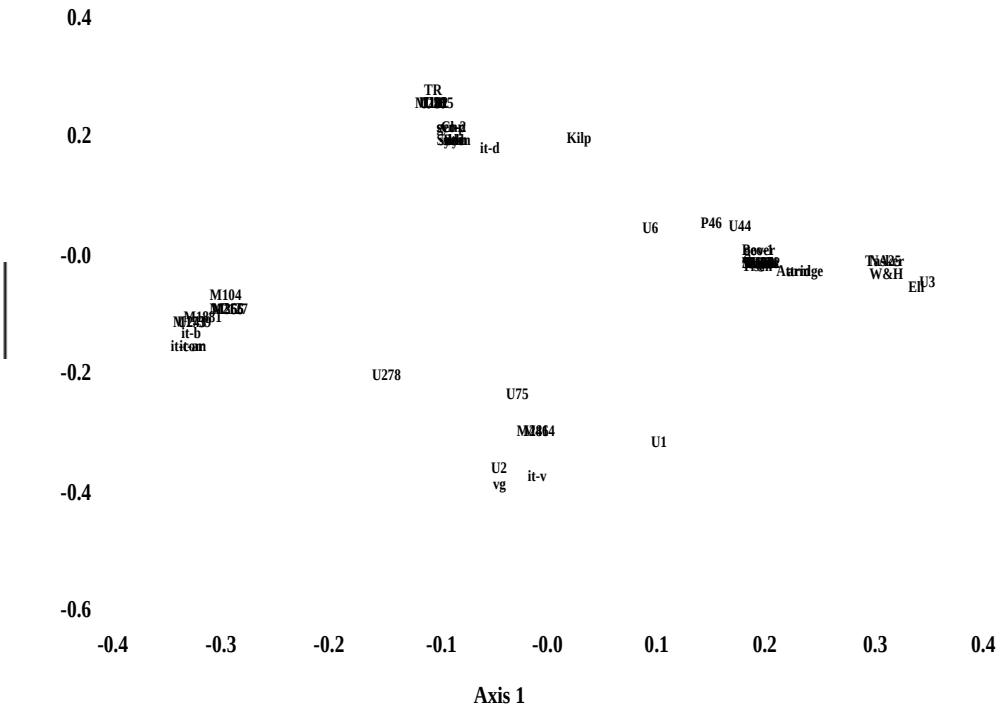
Latin 56: it-t (850?)

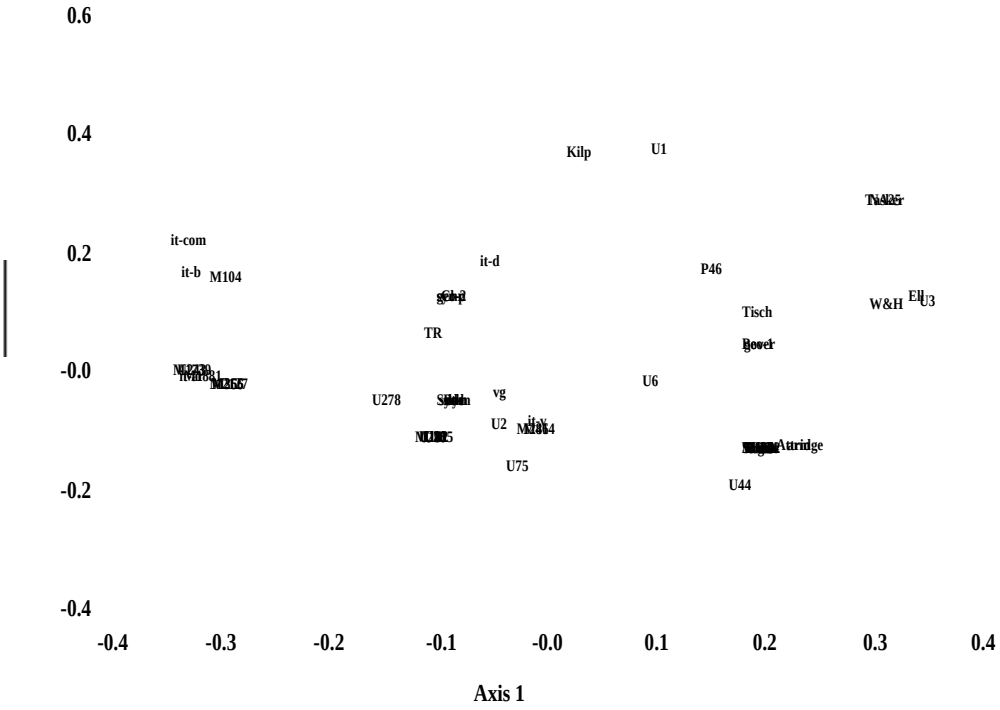




13 units; 31%, 24%, 14%

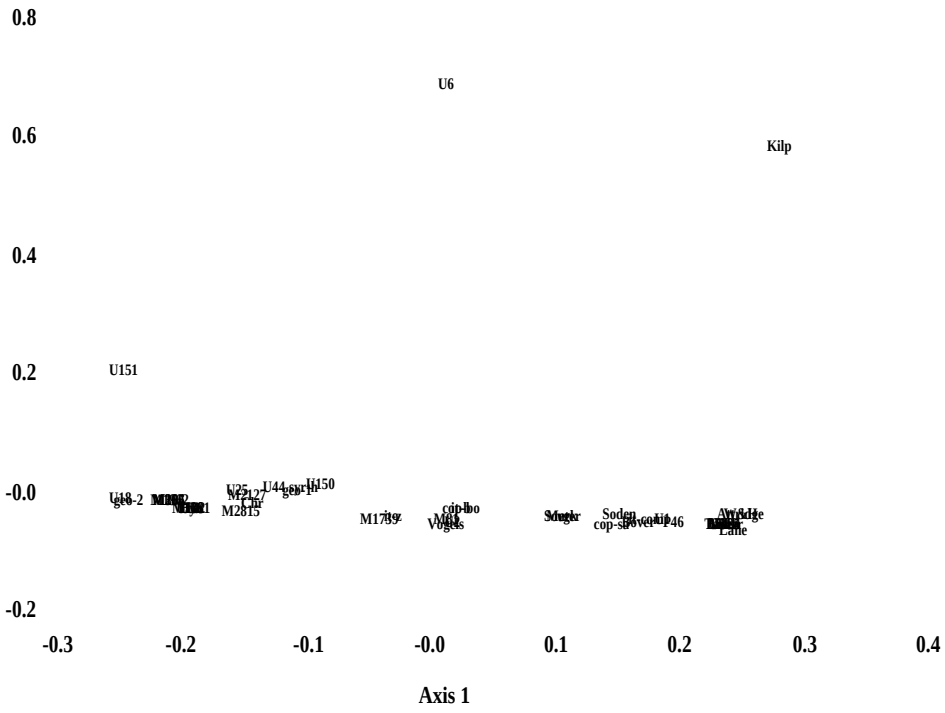
Latin 81: it-v (800?)

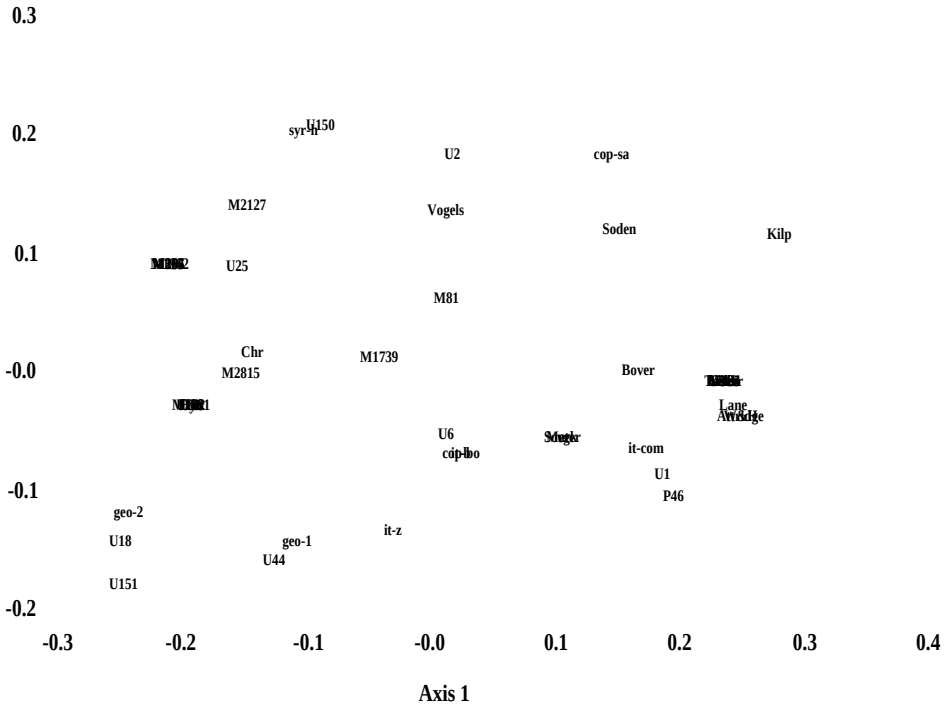




14 units; 29%, 23%, 14%

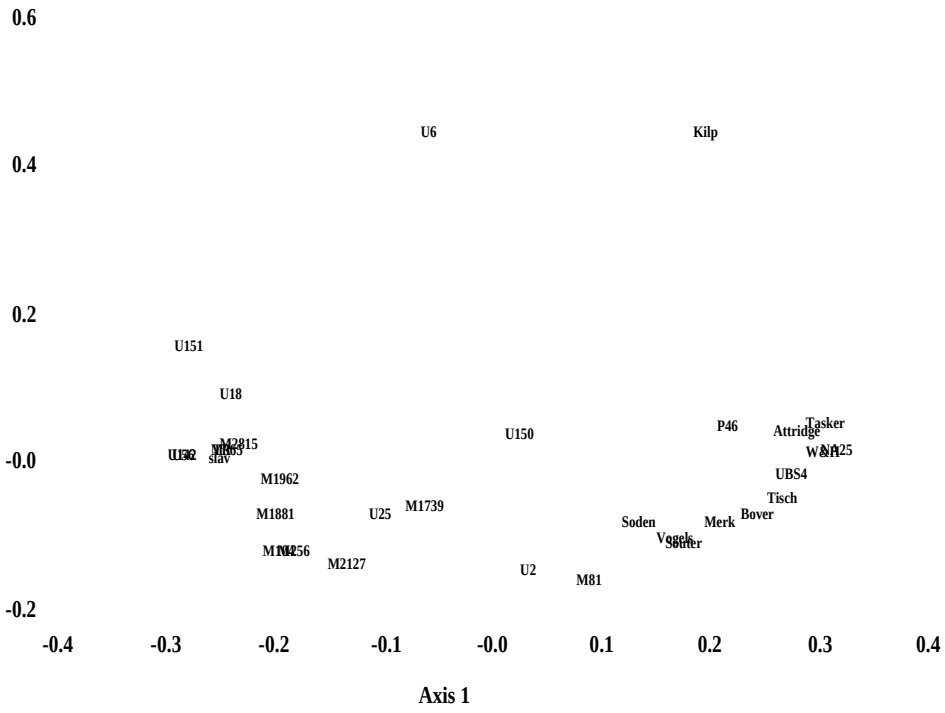
Latin 65: it-z (750?)

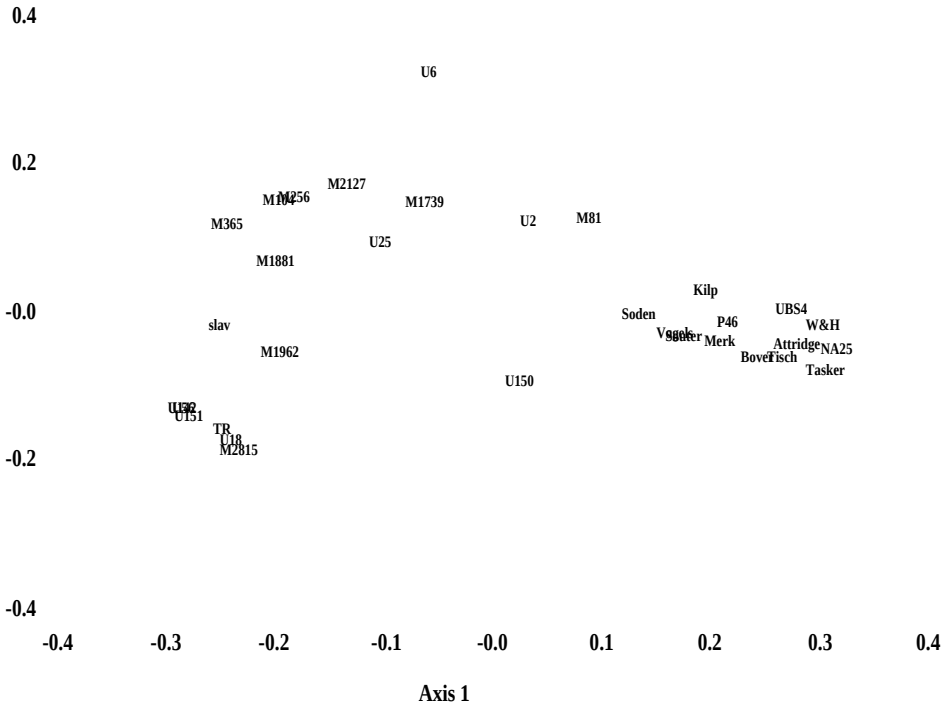




30 units; 33%, 20%, 10%

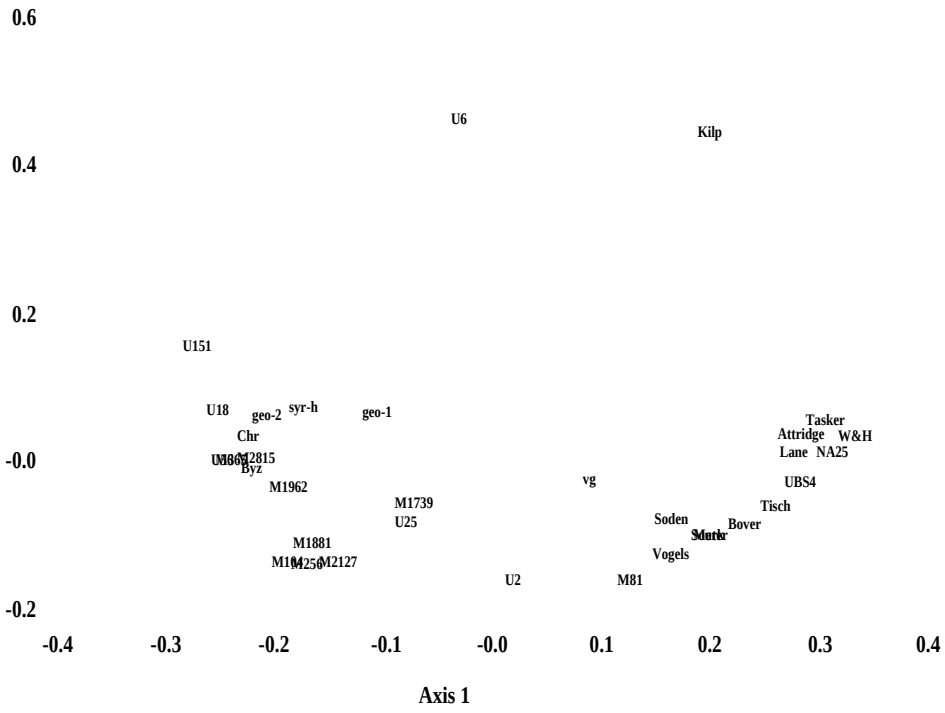
Slavonic (Methodius, Sirmium? 884)

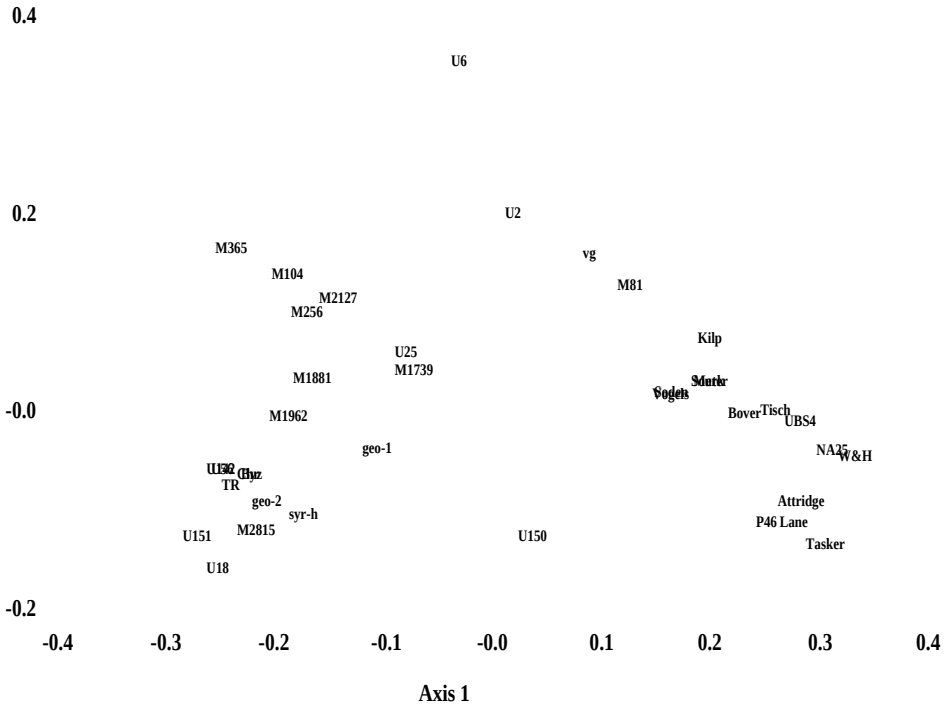




69 units; 31%, 12%, 9%

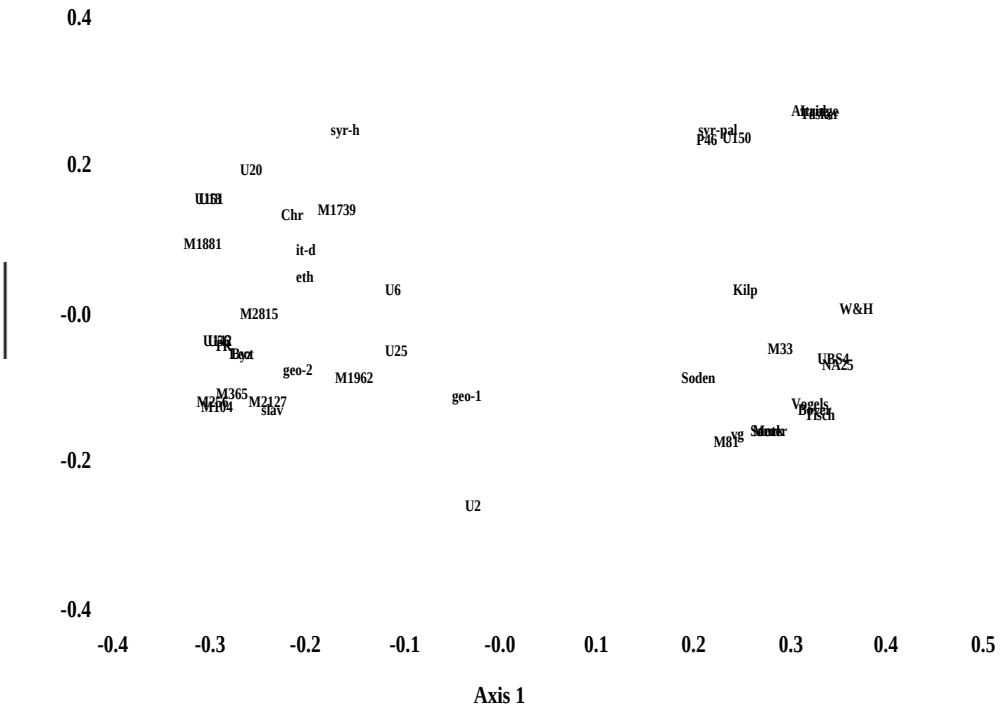
Syriac-Harklean (Thomas of Harkel, Enaton monastery, 616)

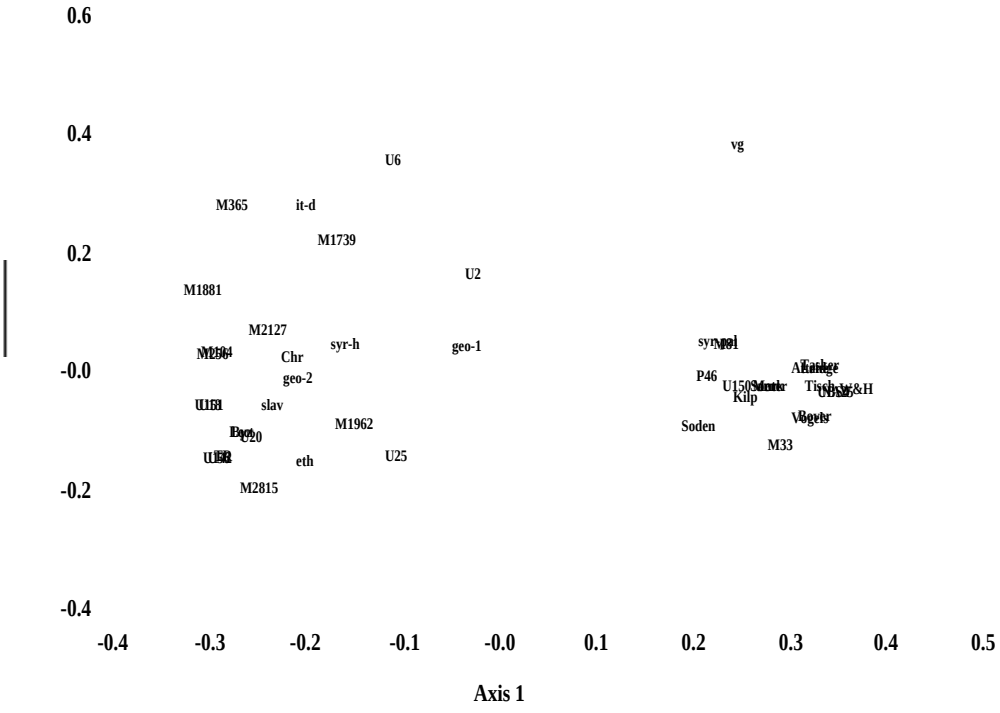




60 units; 32%, 12%, 9%

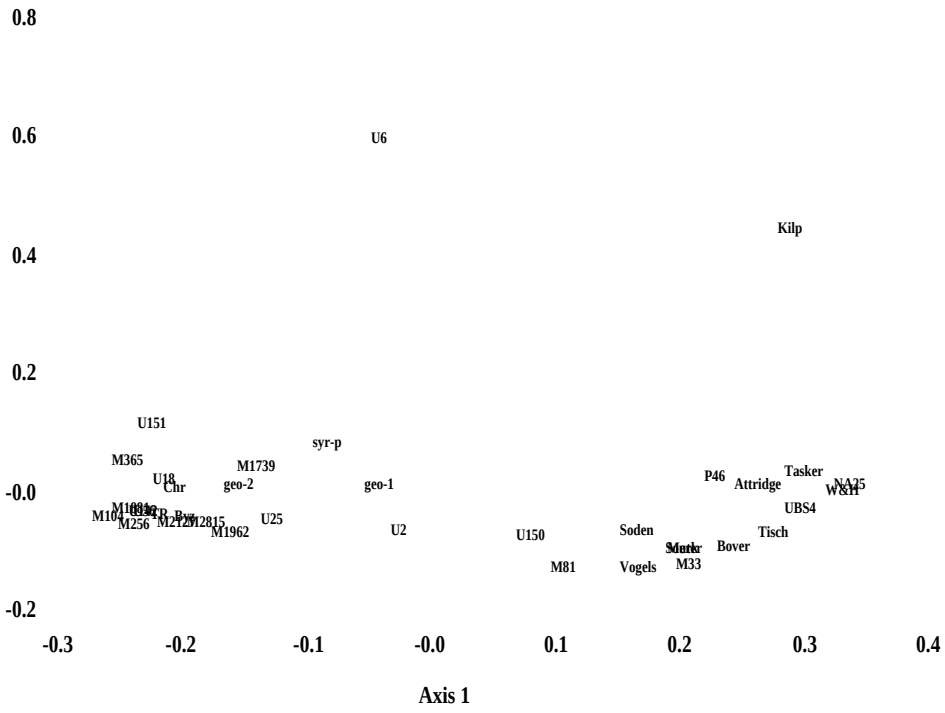
Syriac-Palestinian (Palestine, 550?)

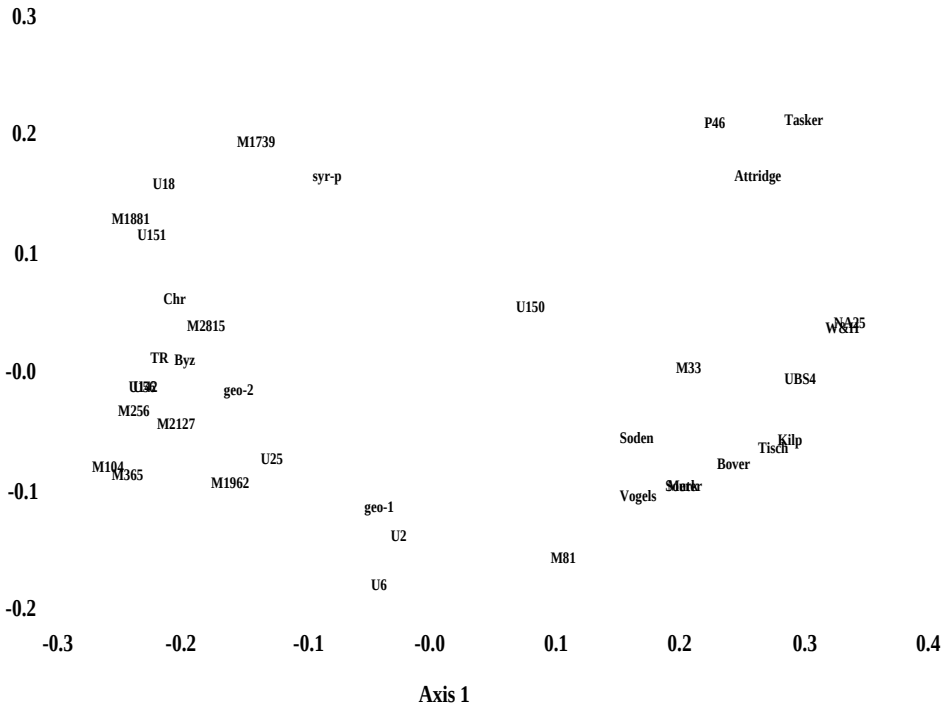




26 units; 41%, 13%, 10%

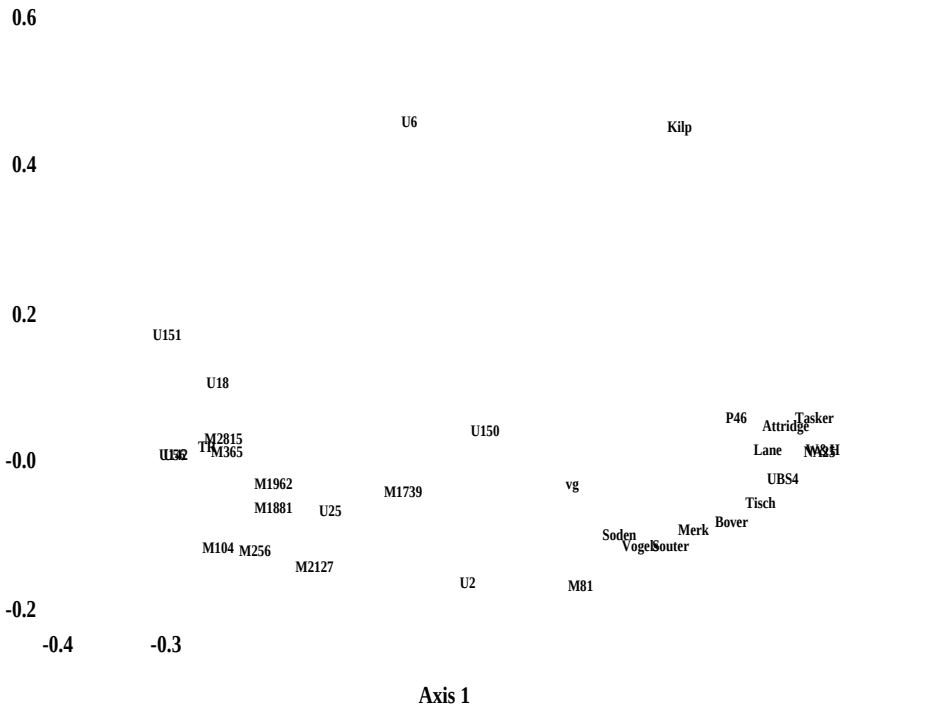
Syriac-Peshitta (Syria, 425?)

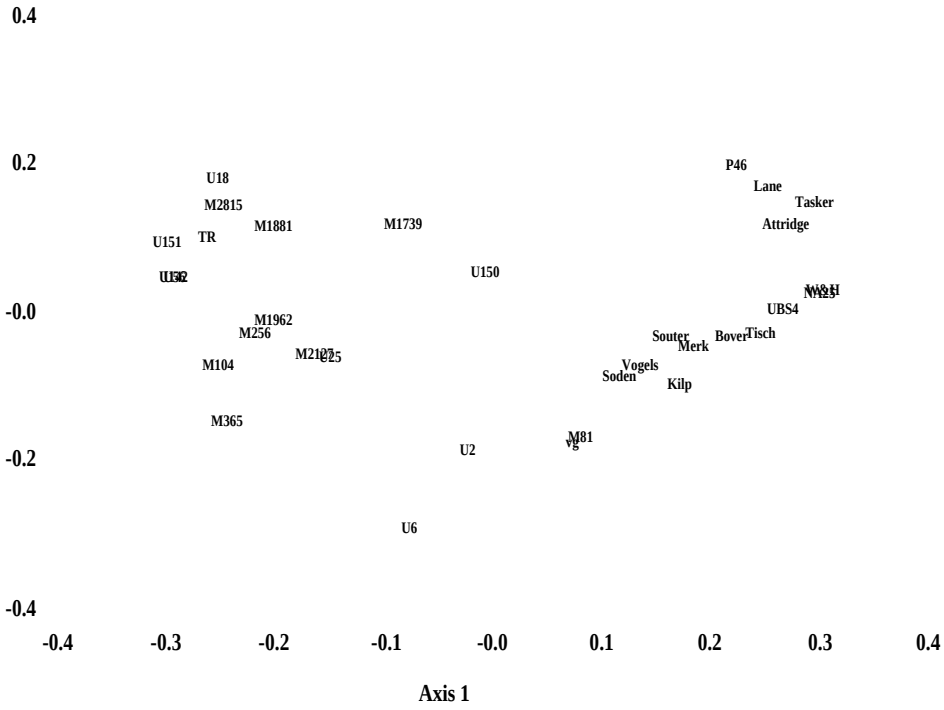




51 units; 34%, 14%, 8%

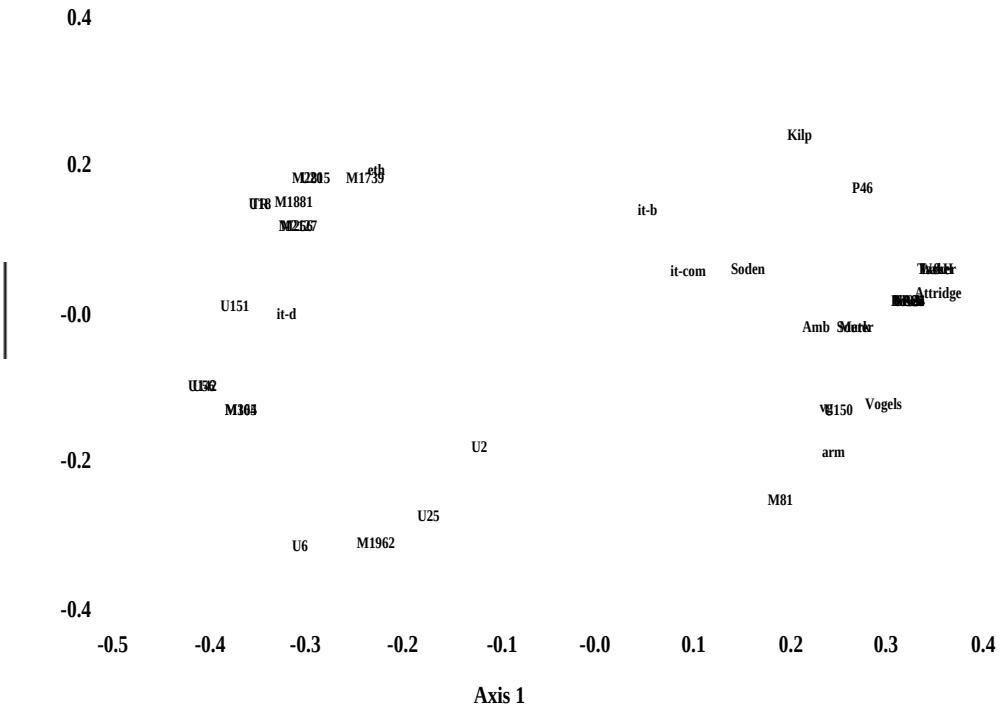
Vulgate (390?)

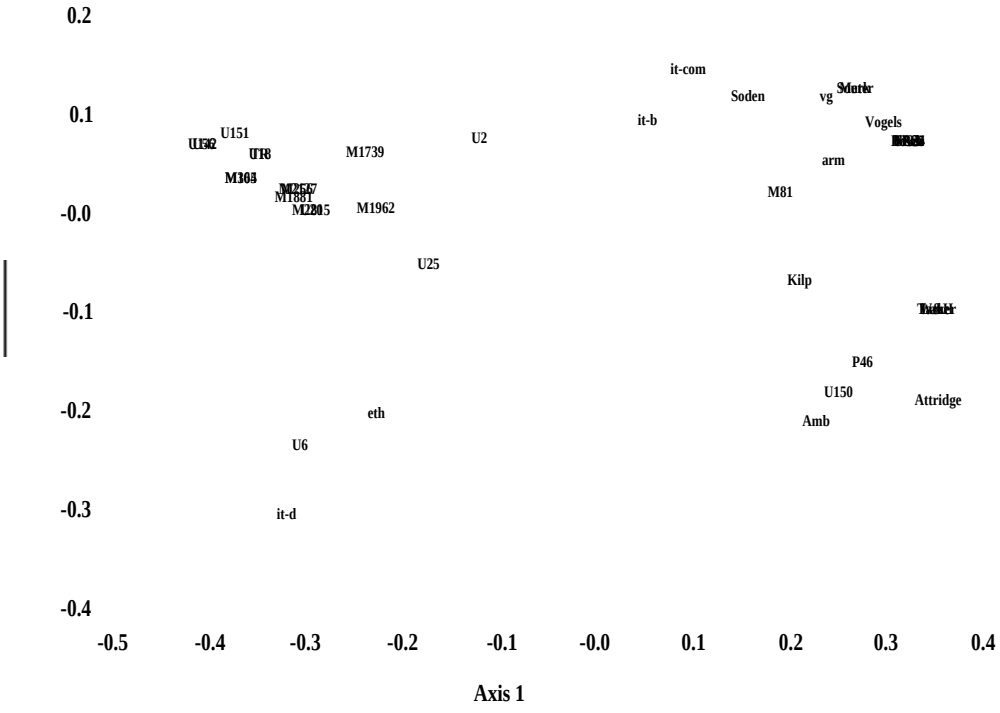




63 units; 31%, 13%, 9%

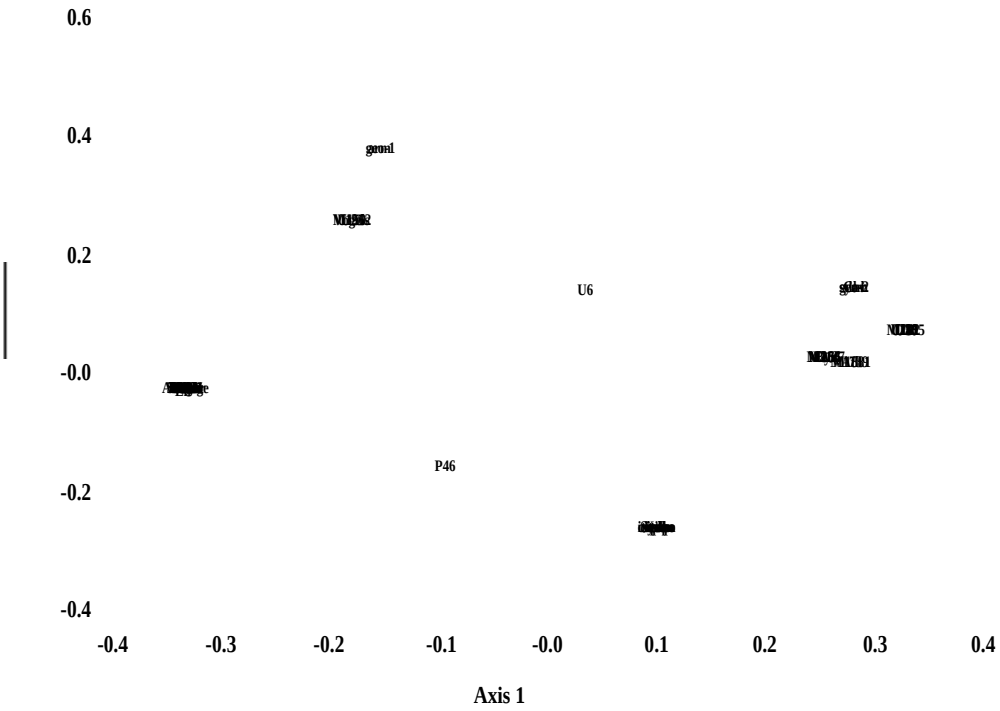
Ambrose (Milan, 380?)

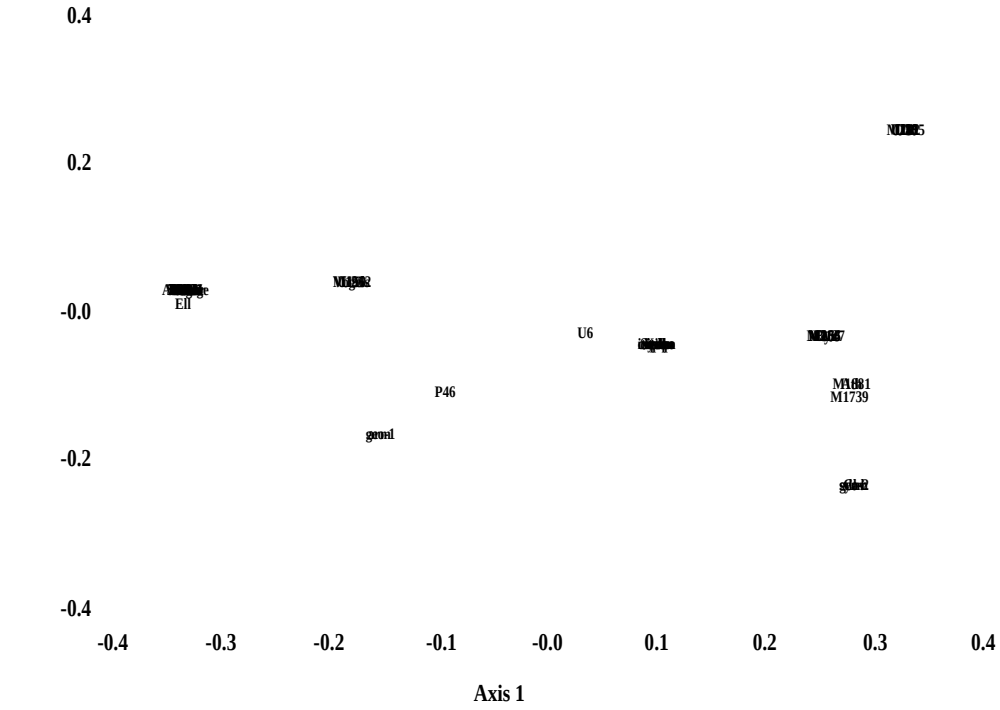




18 units; 50%, 12%, 7%

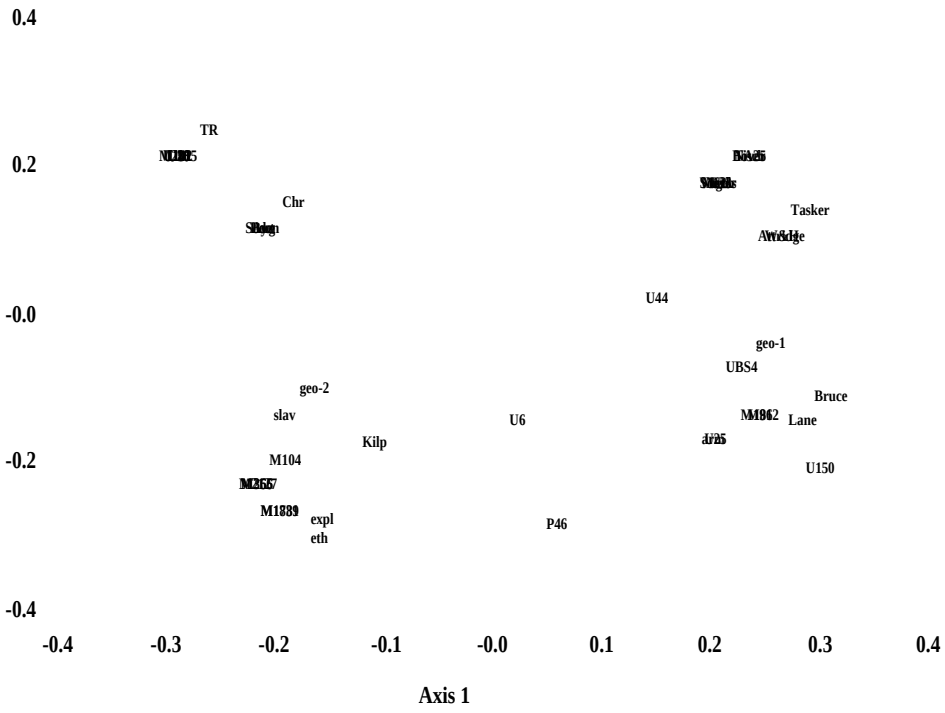
Athanasius (Alexandria, 350?)

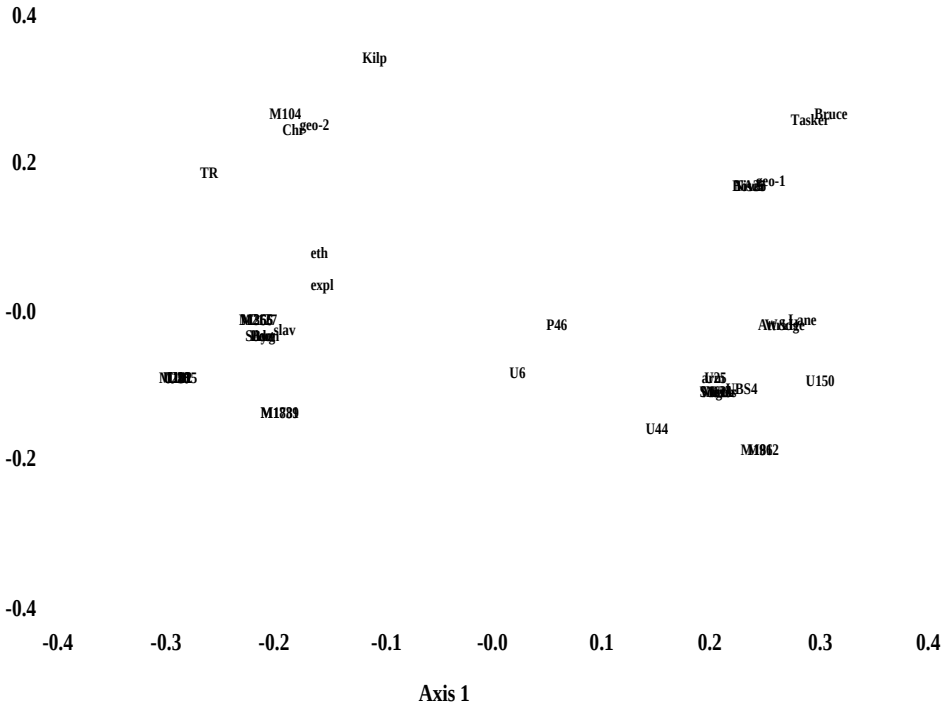




8 units; 52%, 19%, 10%

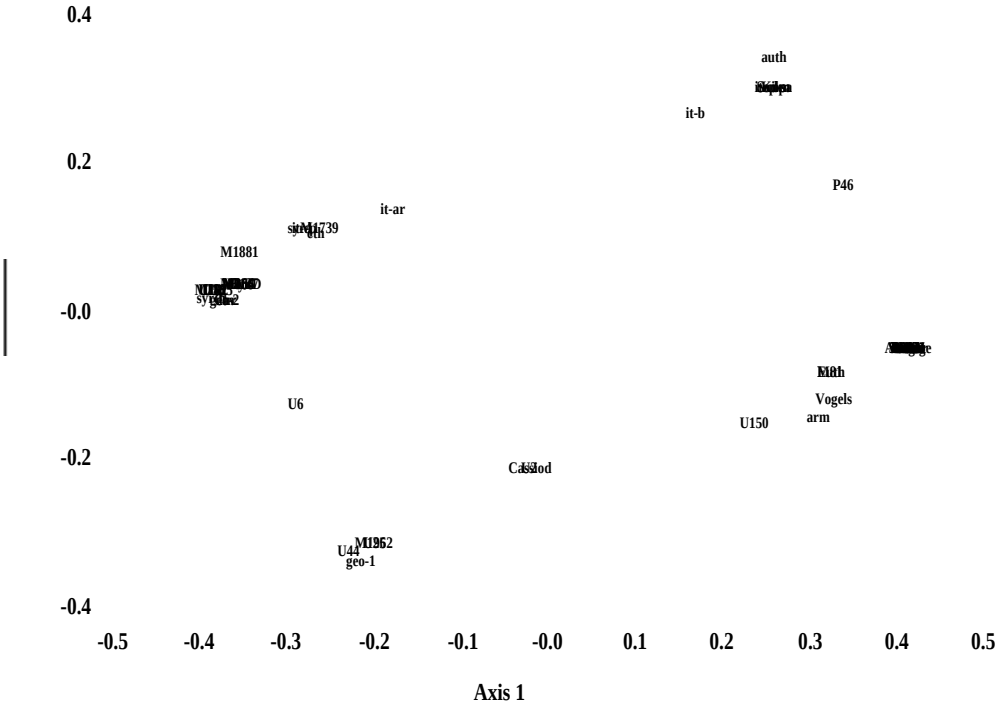
Augustine (Hippo, 410?)

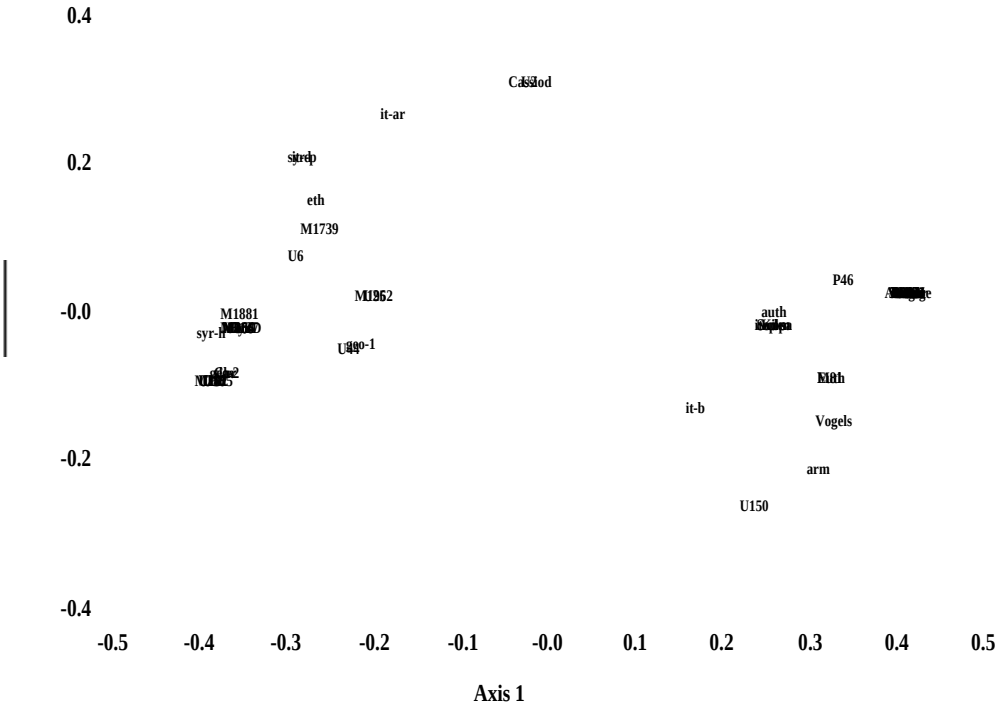




10 units; 33%, 22%, 12%

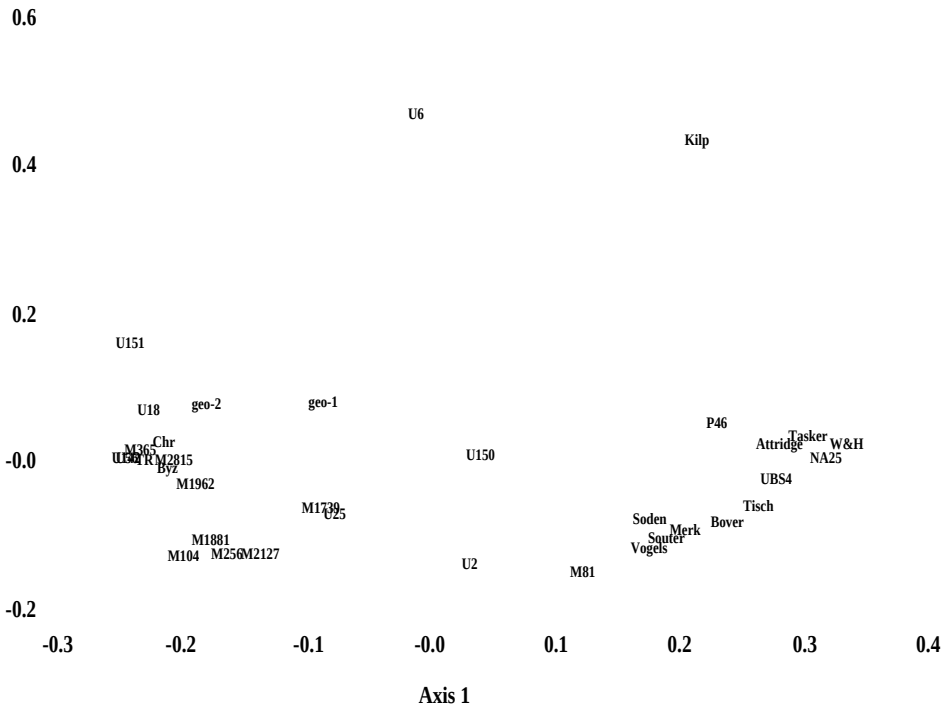
Cassiodorus (Vivarium, 550?)

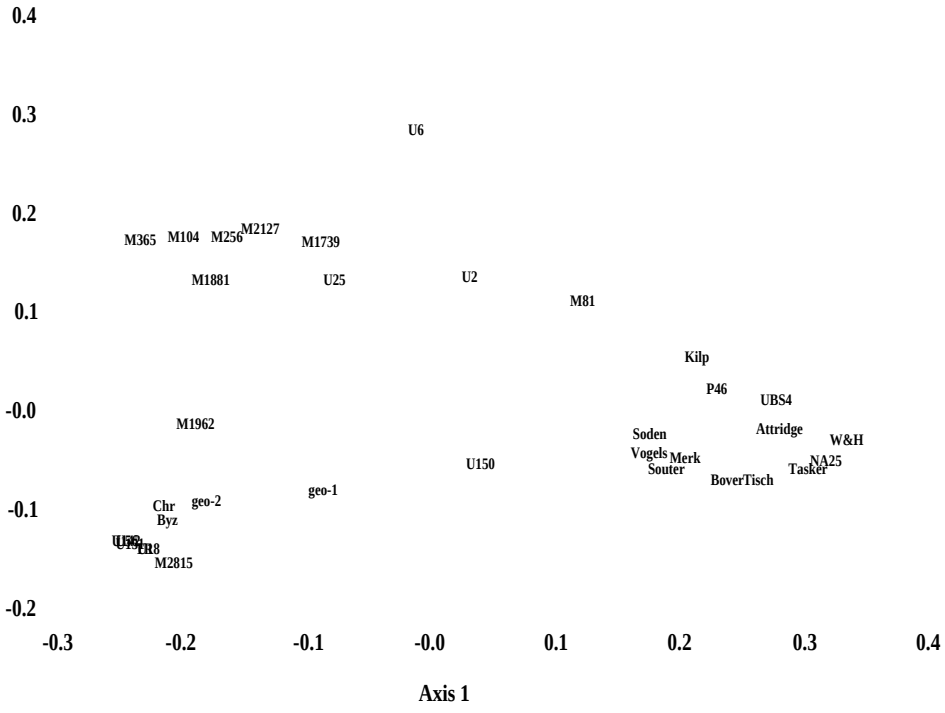




13 units; 63%, 13%, 6%

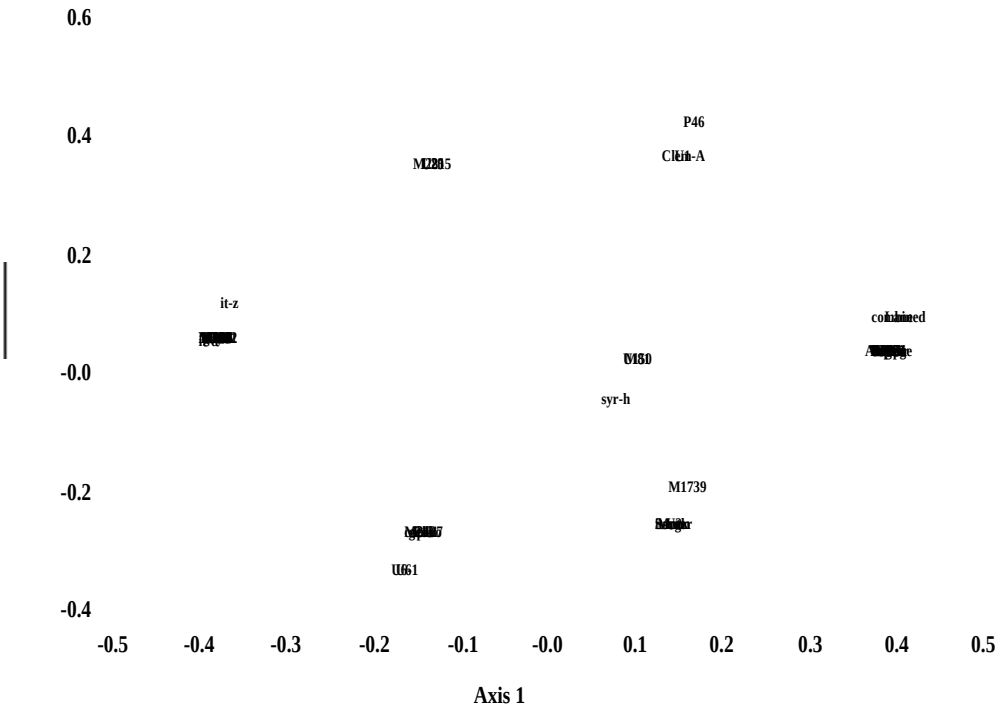
Chrysostom (Antioch, 390?)

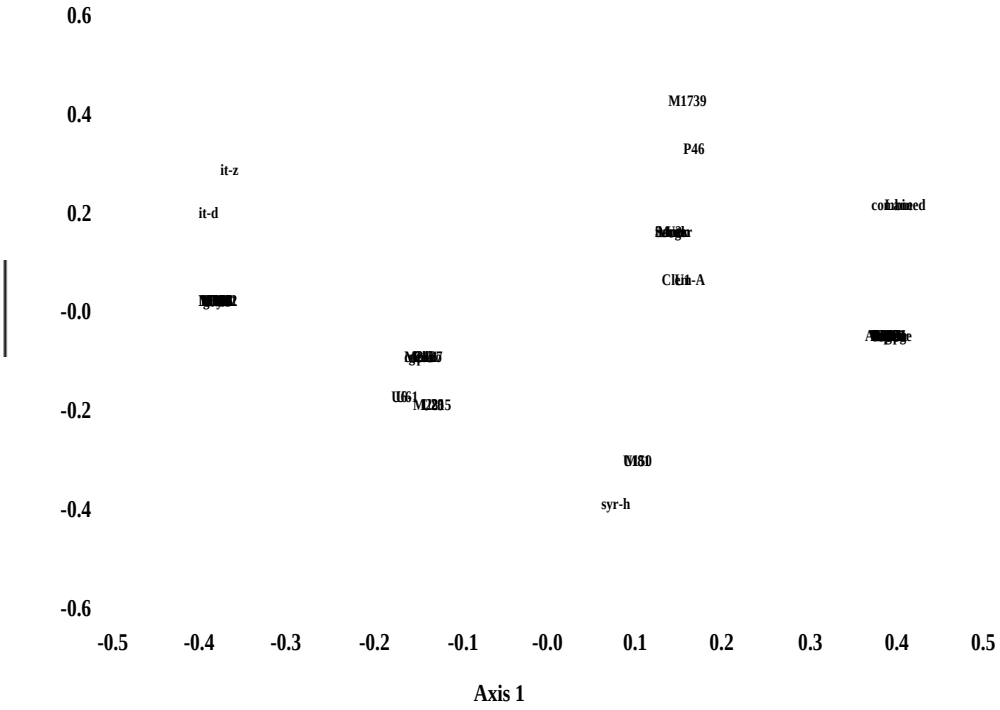




64 units; 31%, 13%, 10%

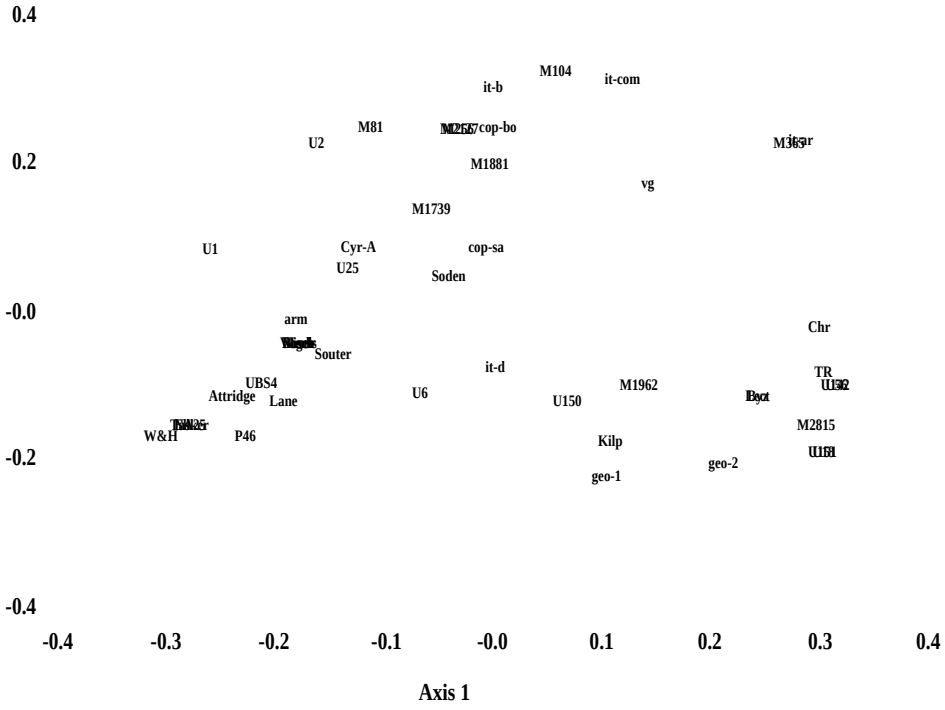
Clement (Alexandria, 200?)

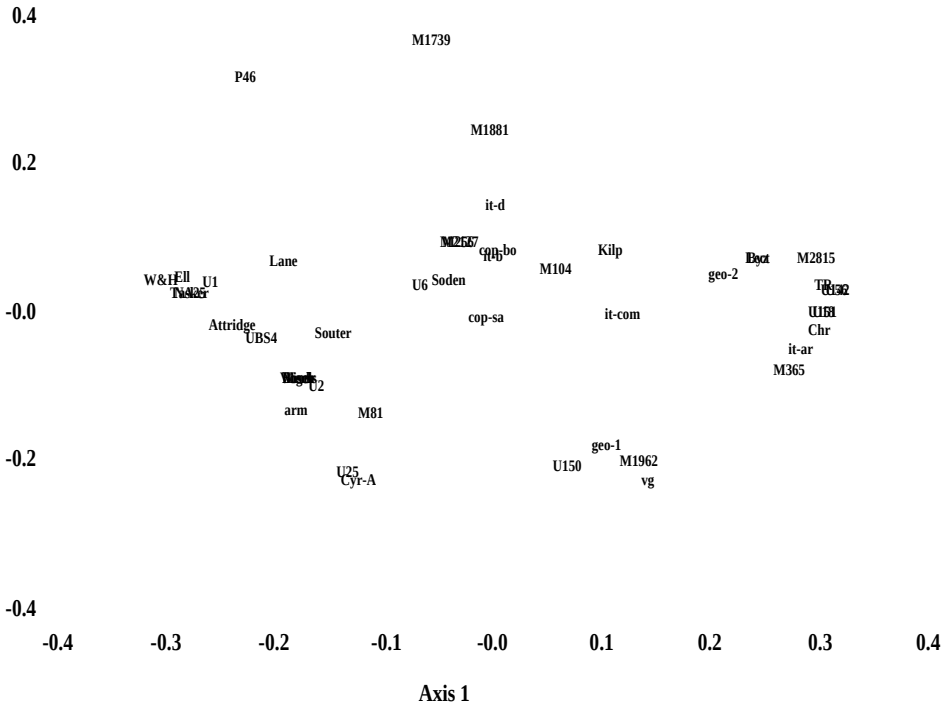




5 units; 53%, 20%, 12%

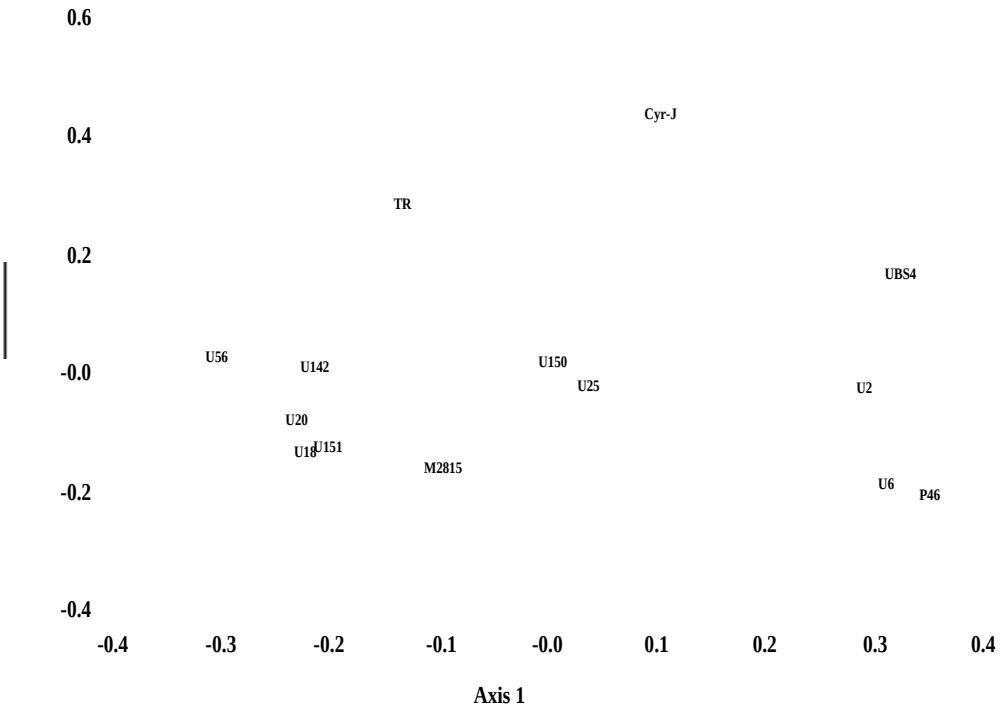
Cyril (Alexandria, 430?)

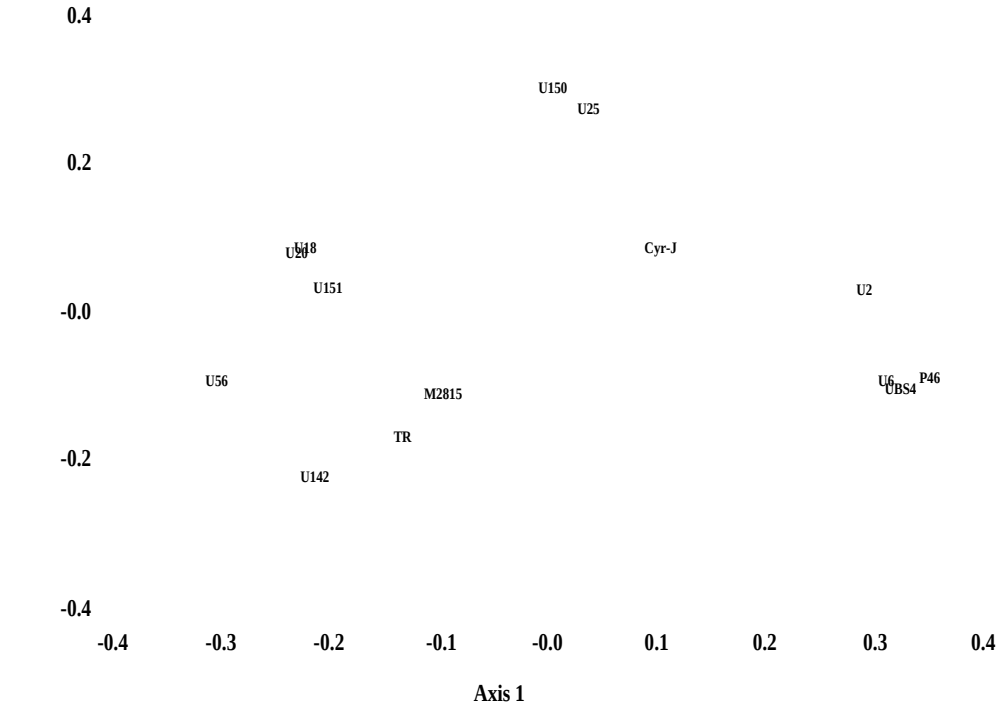




25 units; 27%, 18%, 11%

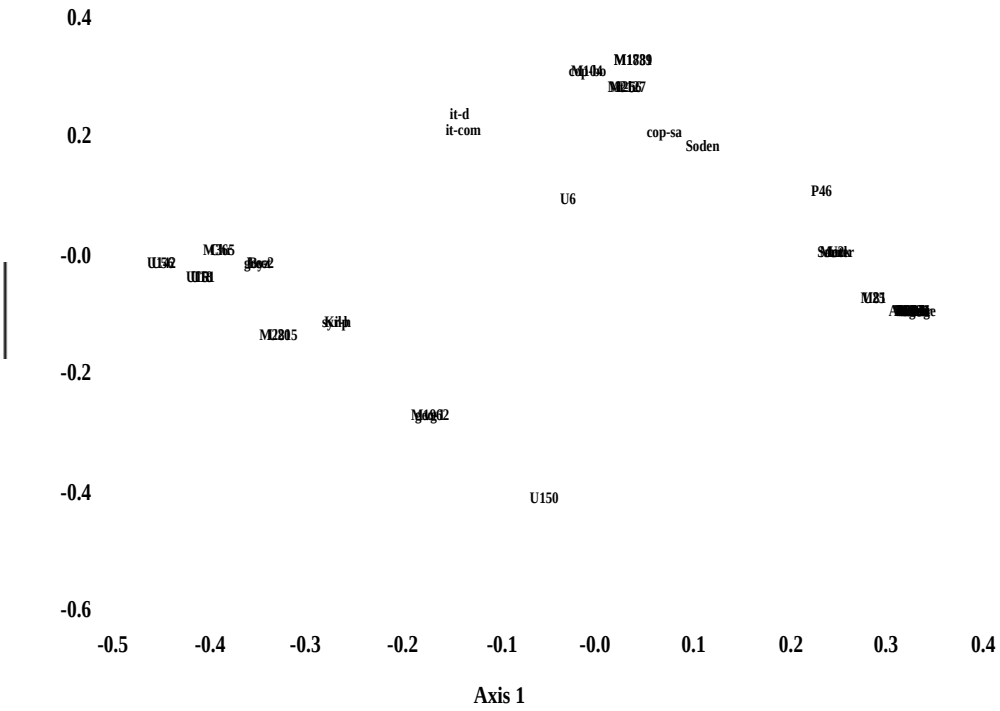
Cyril (Jerusalem, 350?)

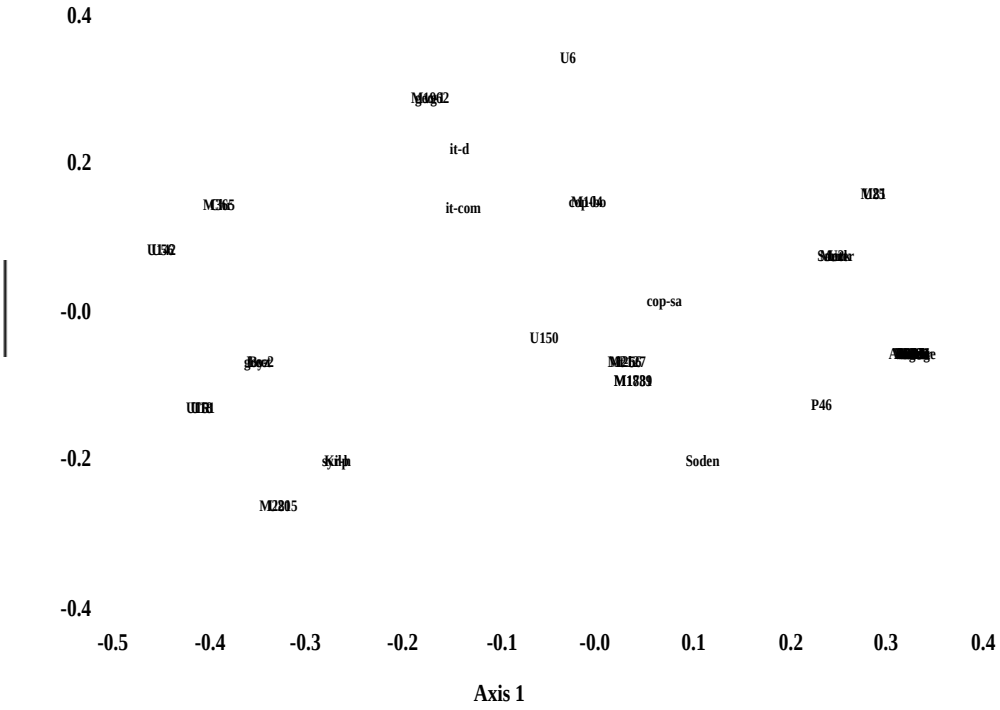




76 units; 30%, 18%, 13%

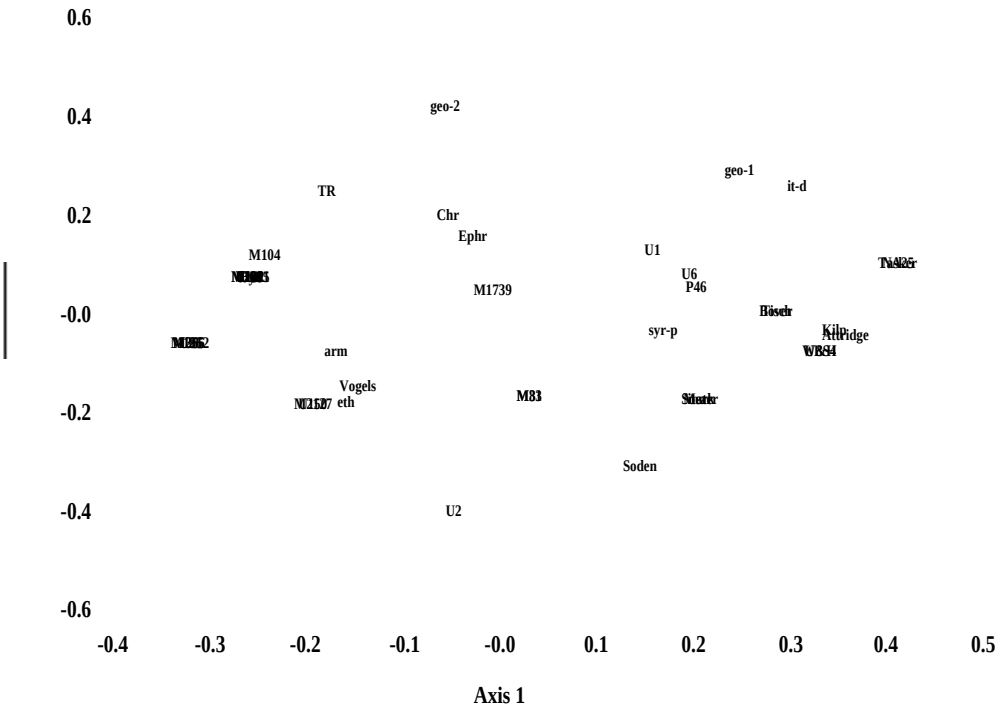
Didymus (Alexandria, 350?)

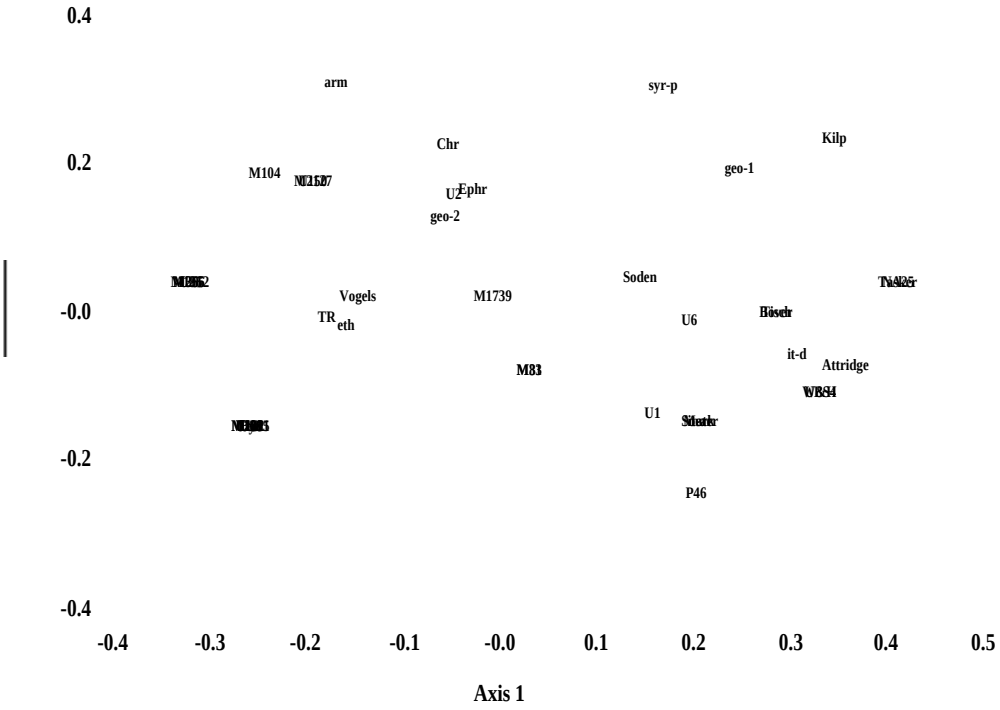




12 units; 48%, 18%, 13%

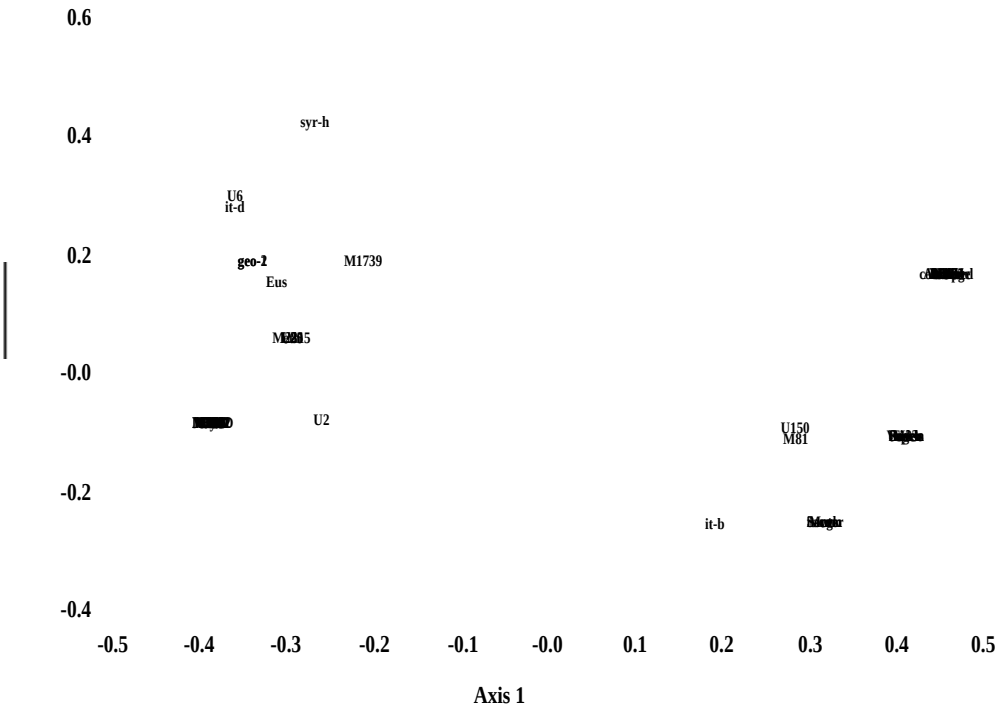
Ephraem (Edessa, 360?)

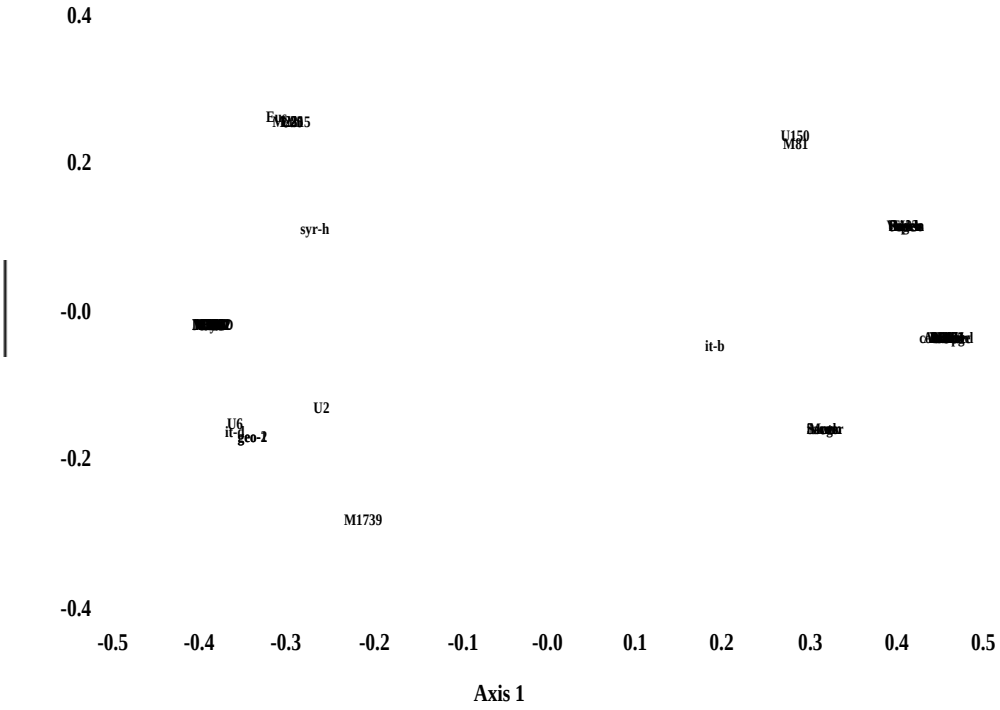




10 units; 36%, 16%, 12%

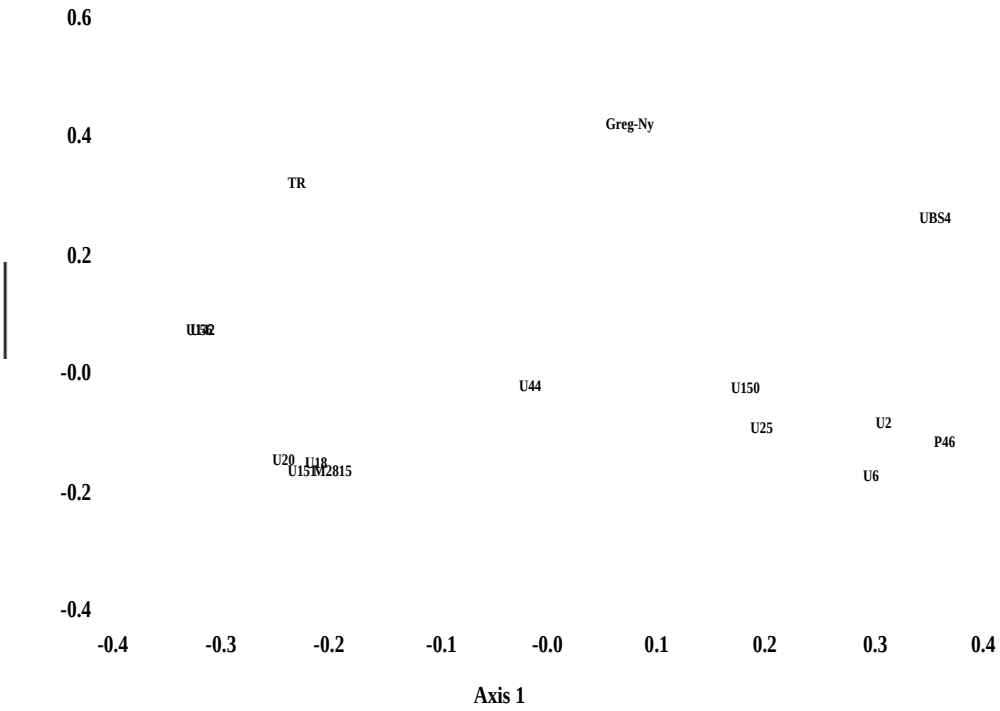
Eusebius (Caesarea, 320?)

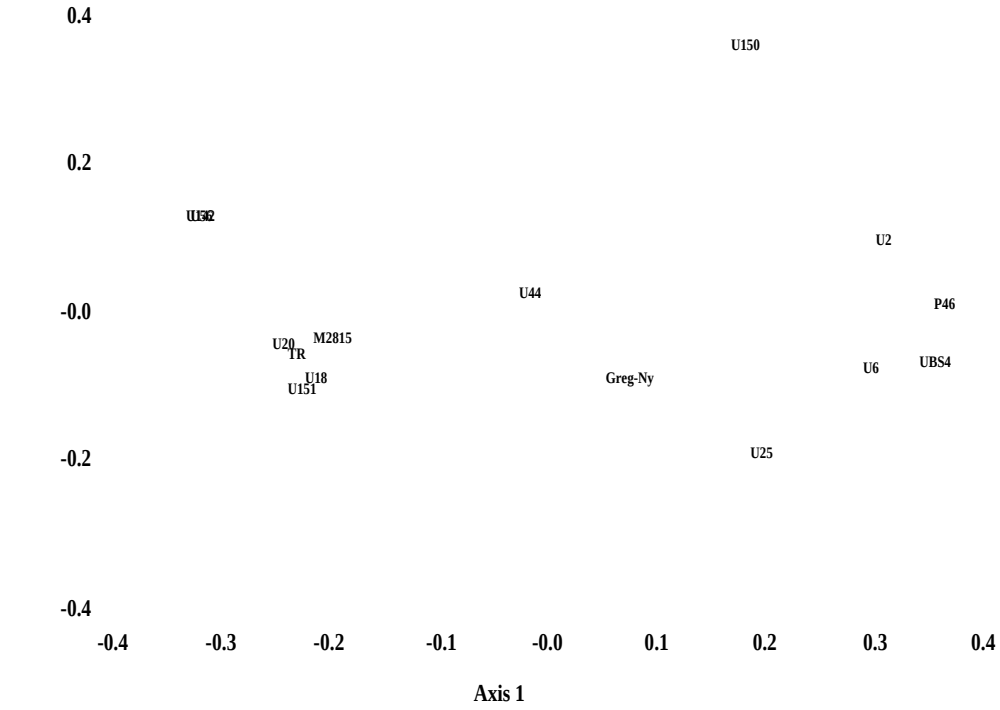




9 units; 71%, 13%, 8%

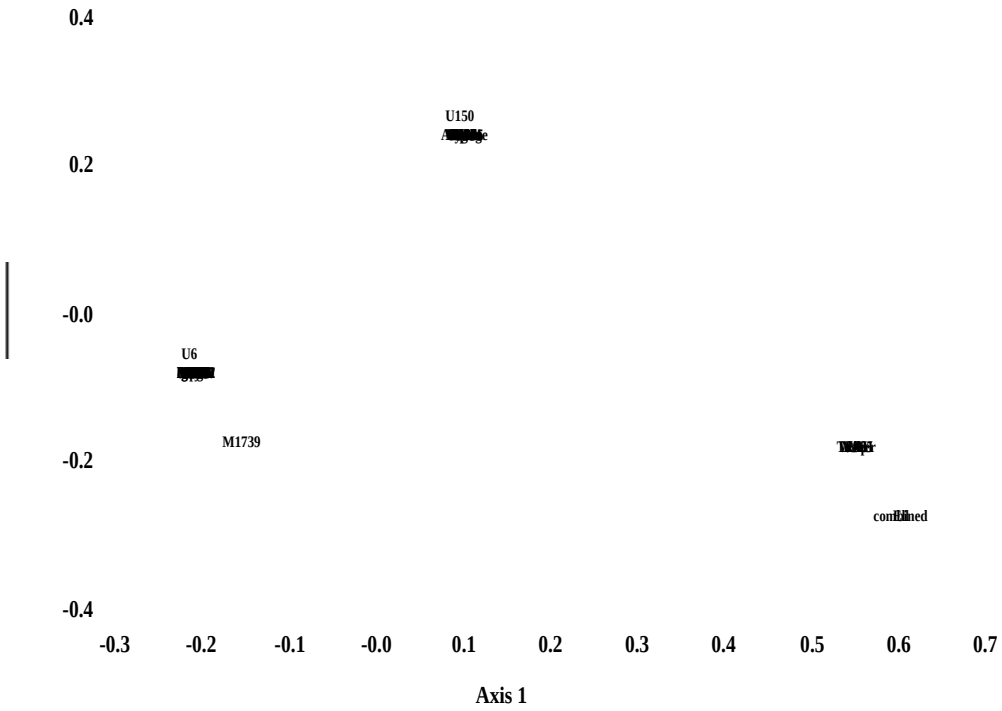
Gregory (Nyssa, 380?)

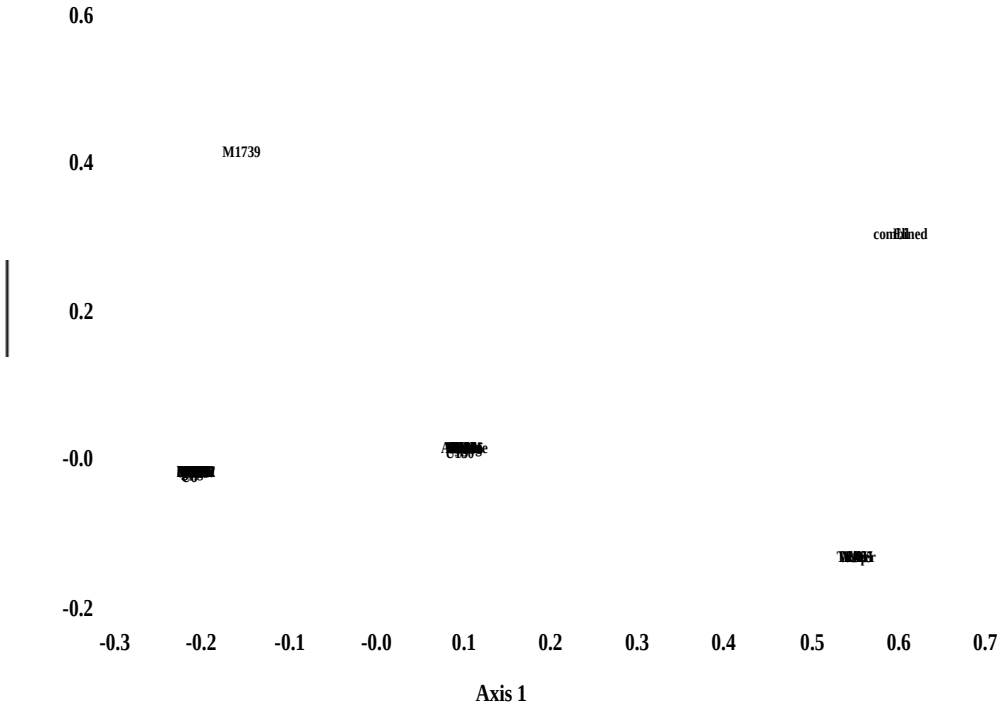




40 units; 36%, 19%, 9%

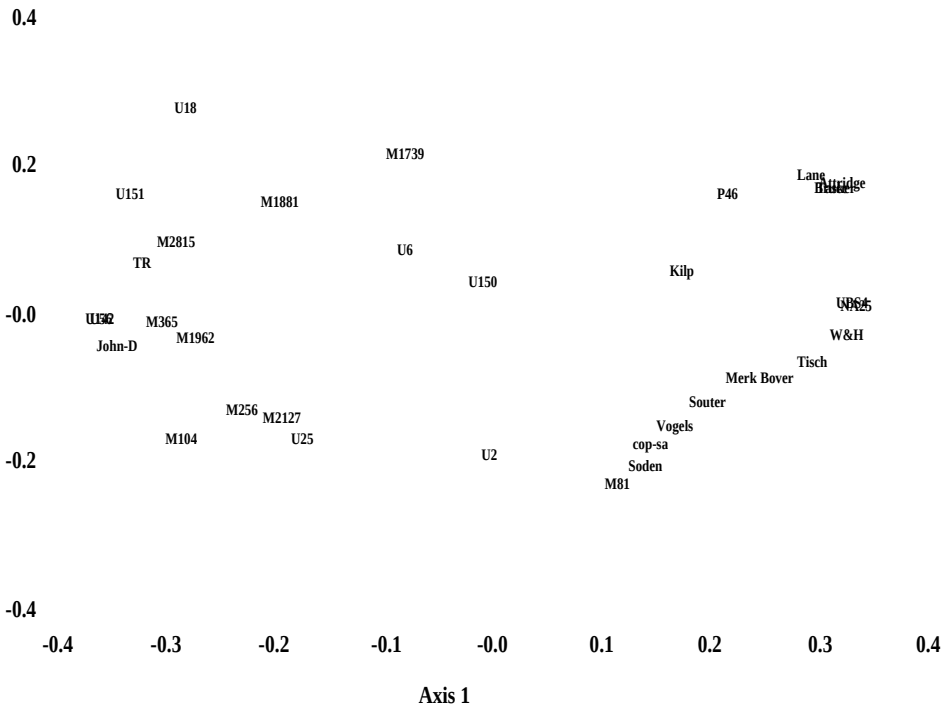
Jerome (Bethlehem, 390?)

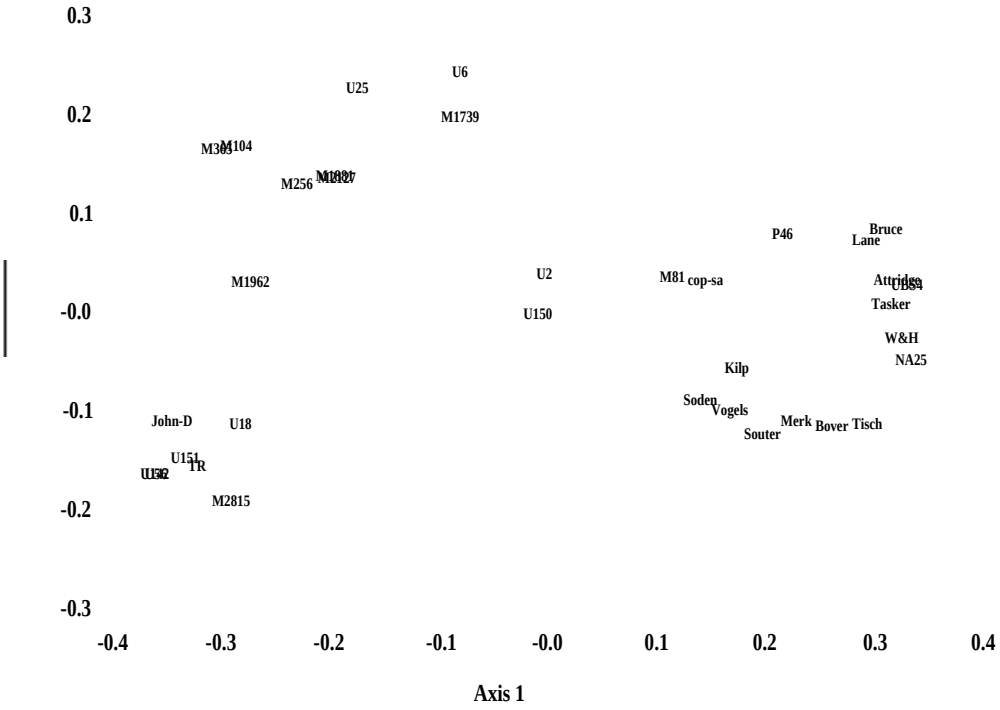




5 units; 63%, 24%, 7%

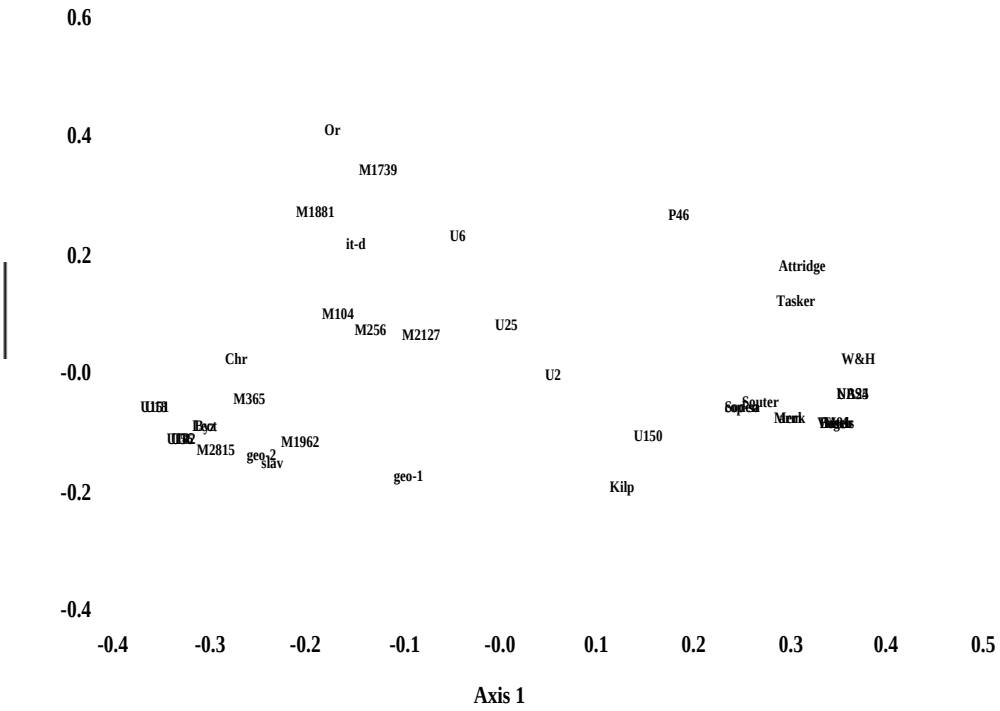
John (Damascus, 730?)

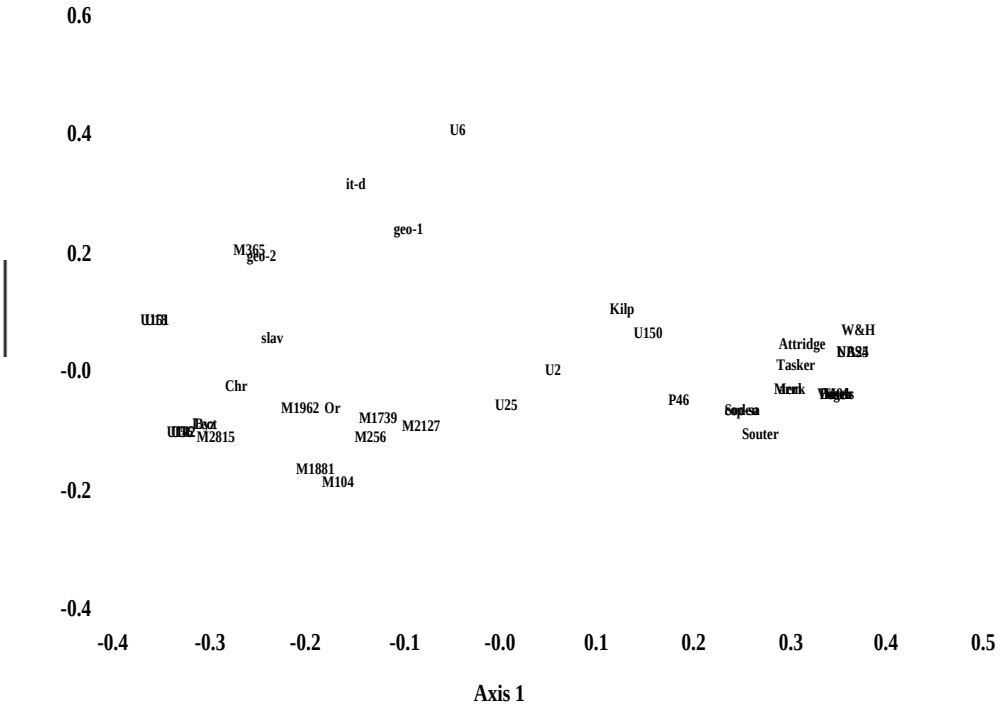




43 units; 37%, 11%, 9%

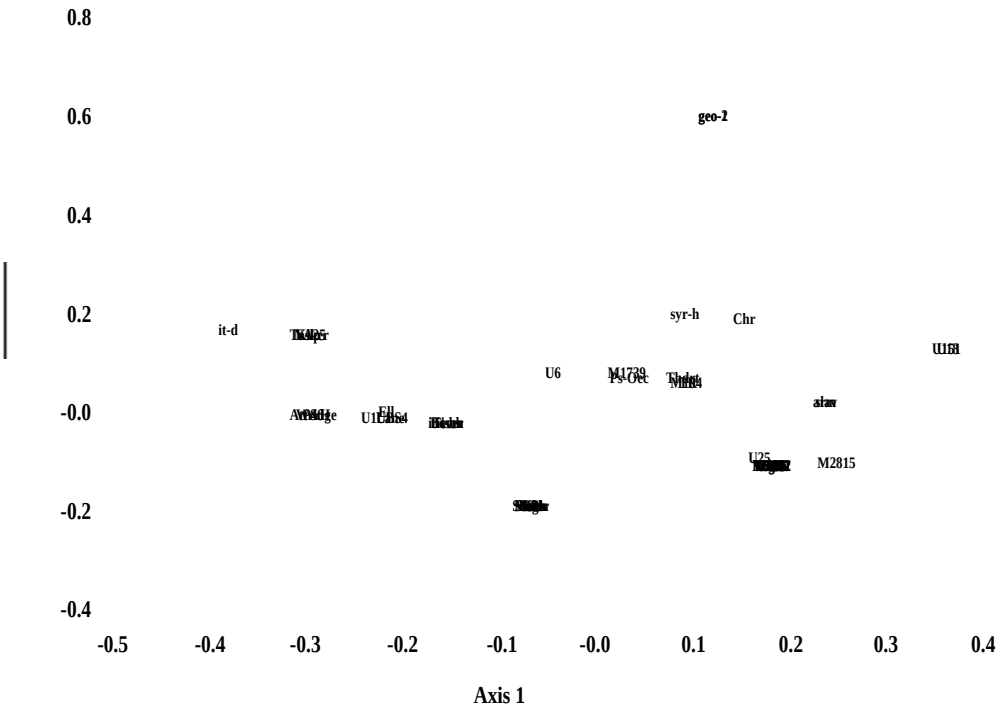
Origen (Caesarea? 230?)

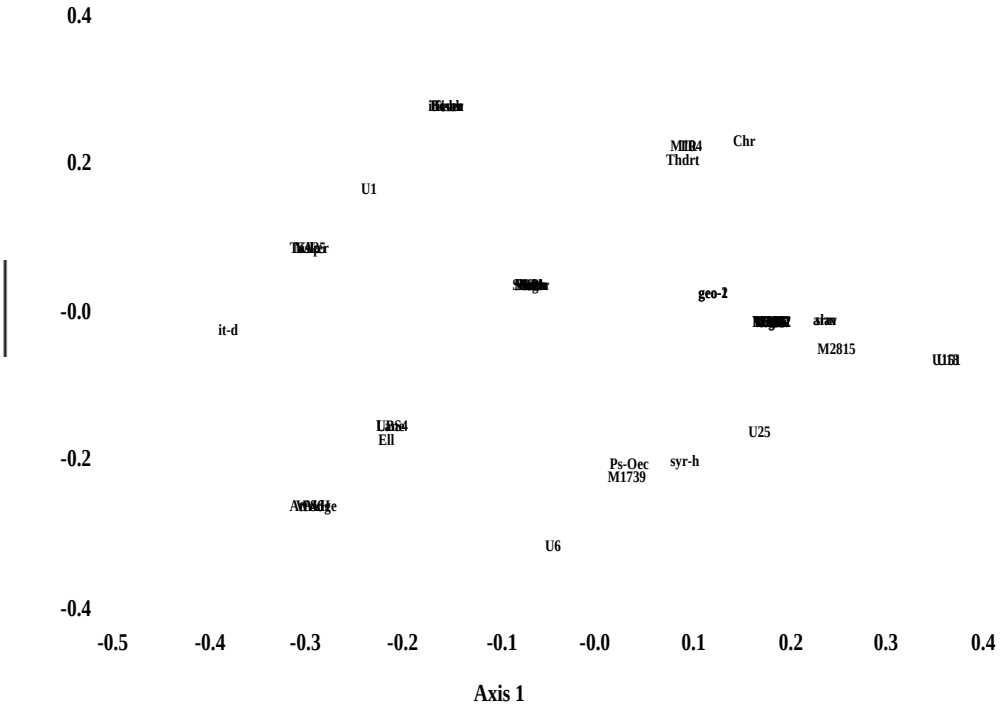




28 units; 42%, 13%, 9%

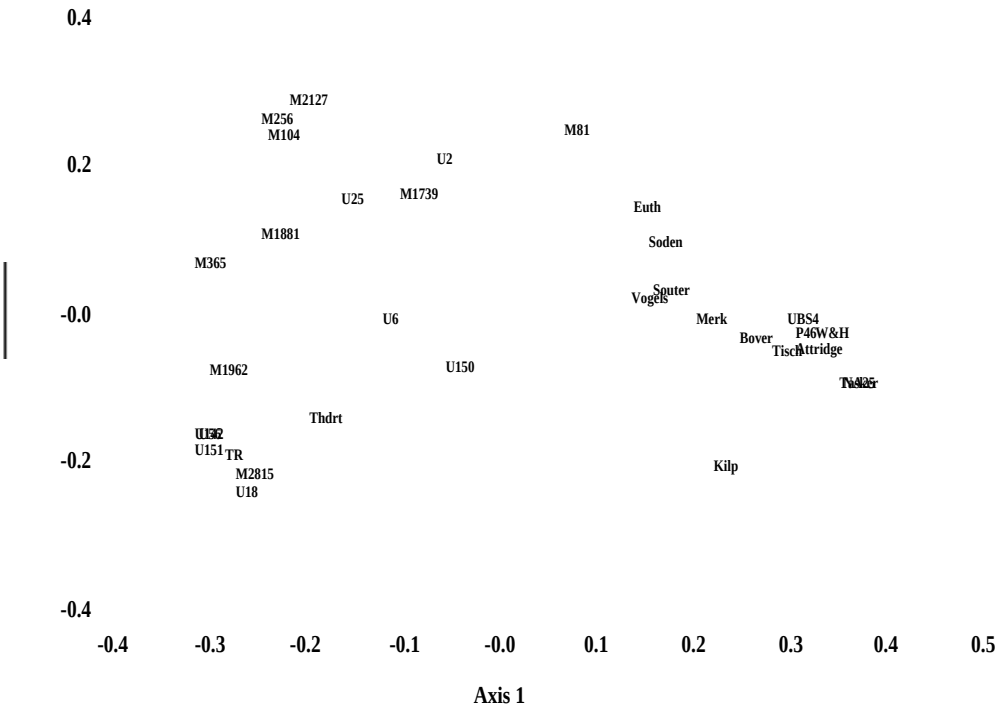
Pseudo-Oecumenius (550?)

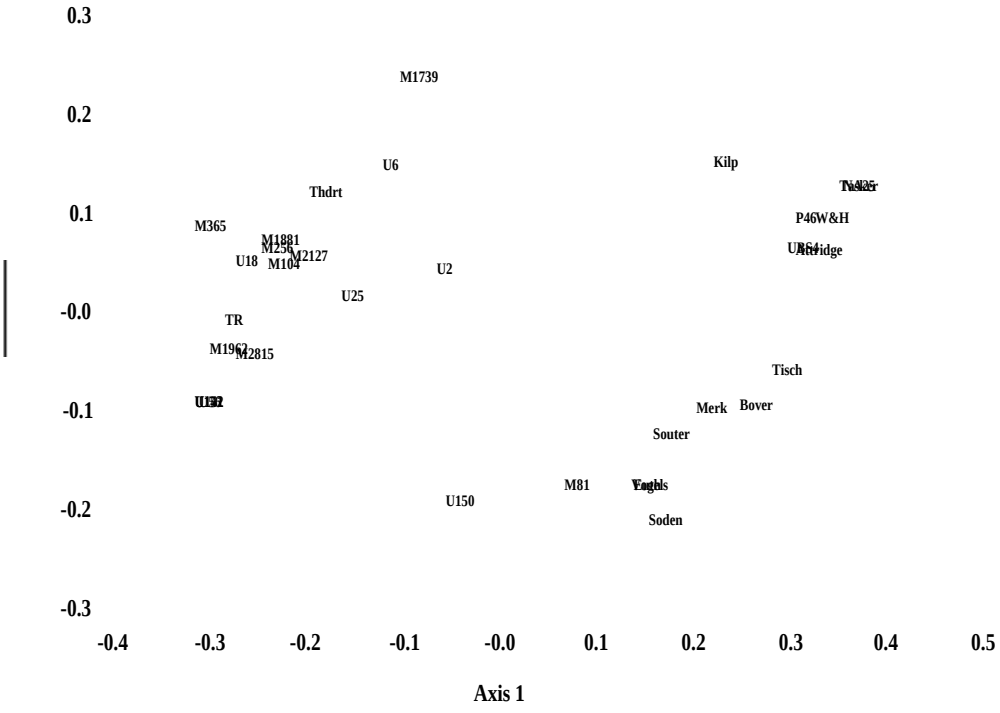




9 units; 29%, 21%, 17%

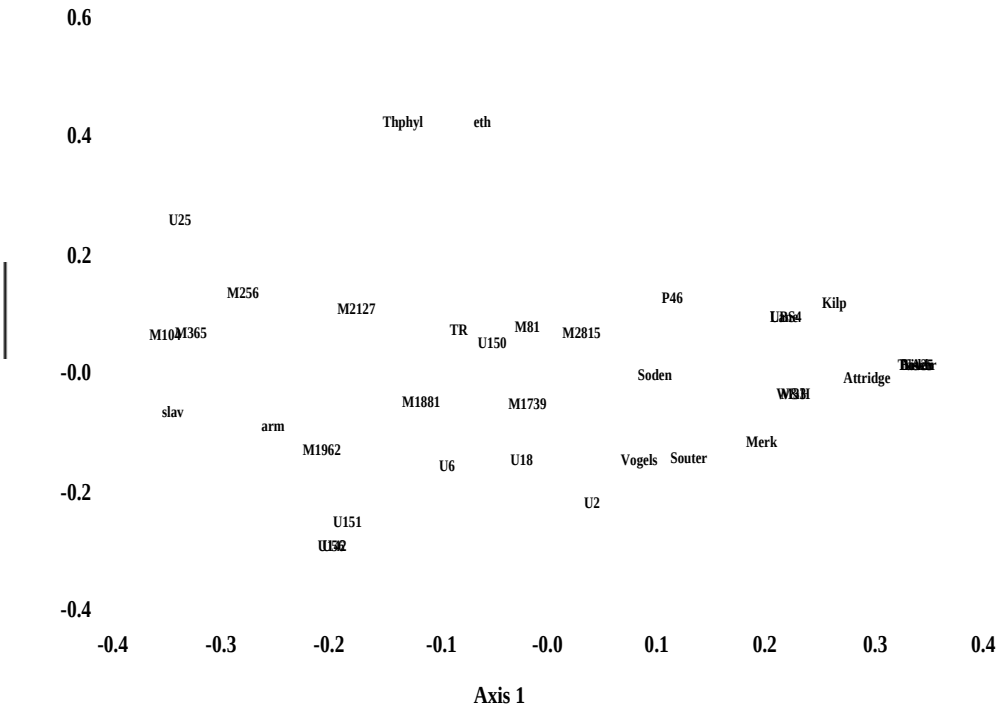
Theodoret (Cyrrhus, 430?)

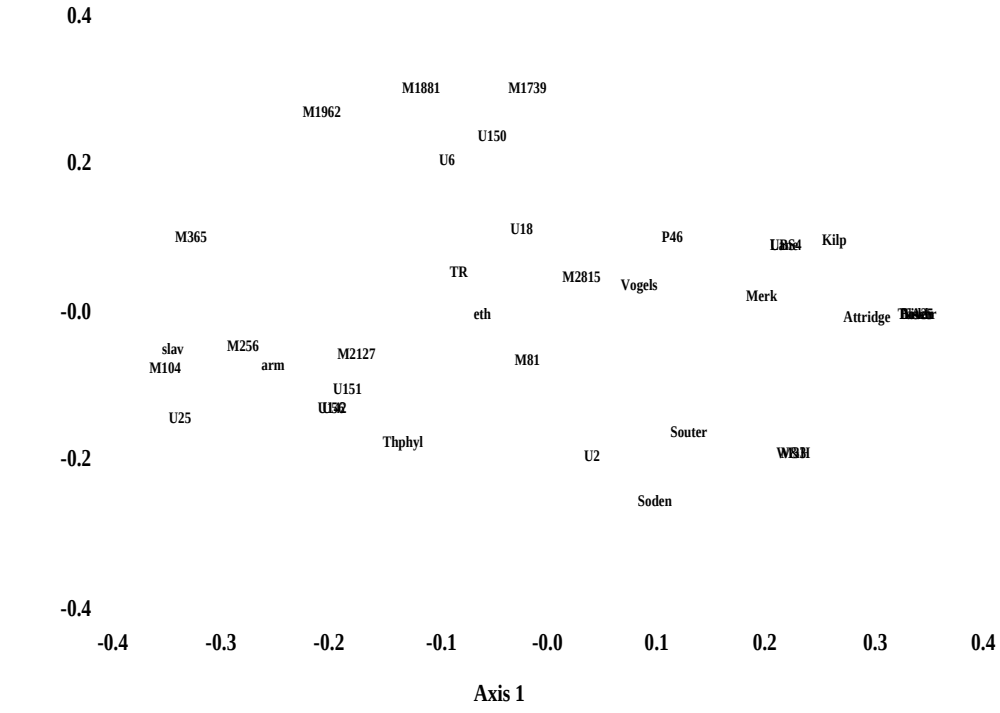




42 units; 34%, 13%, 8%

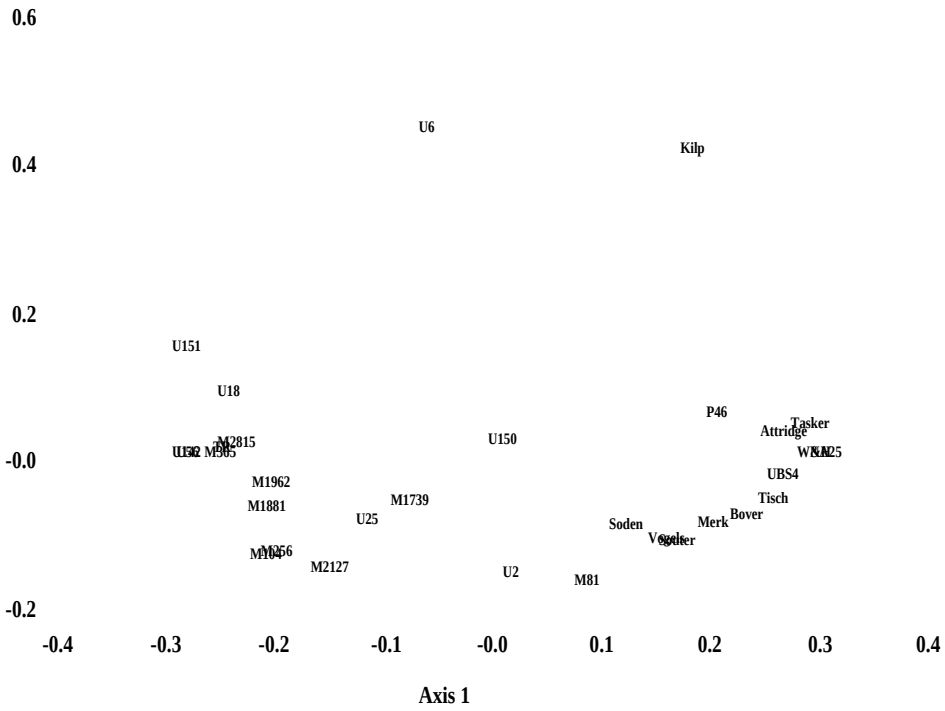
Theophylact (Ohrid, 1090?)

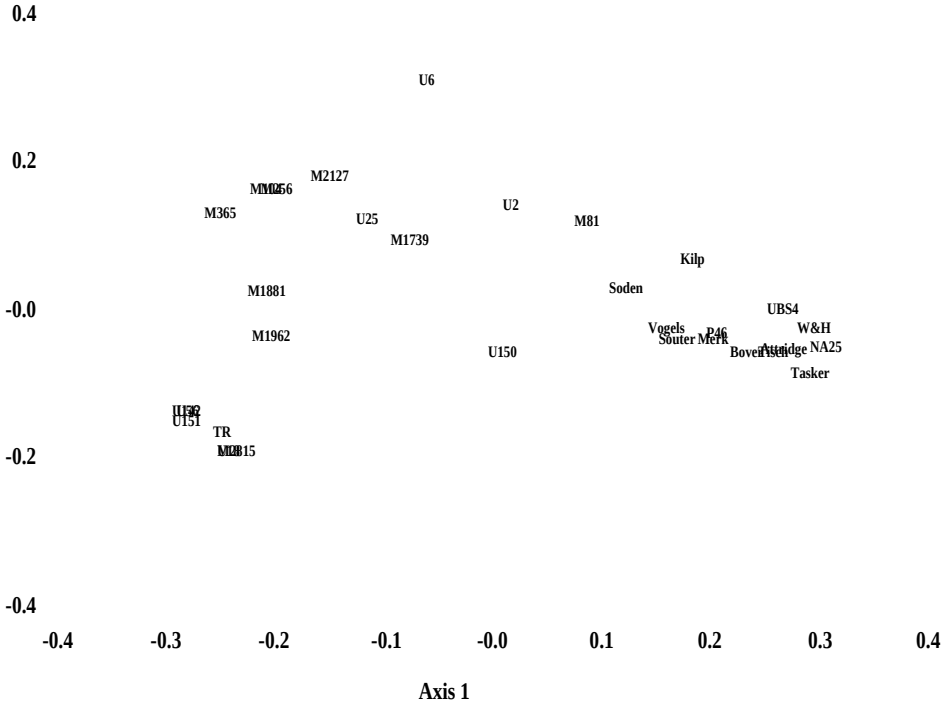




15 units; 29%, 16%, 12%

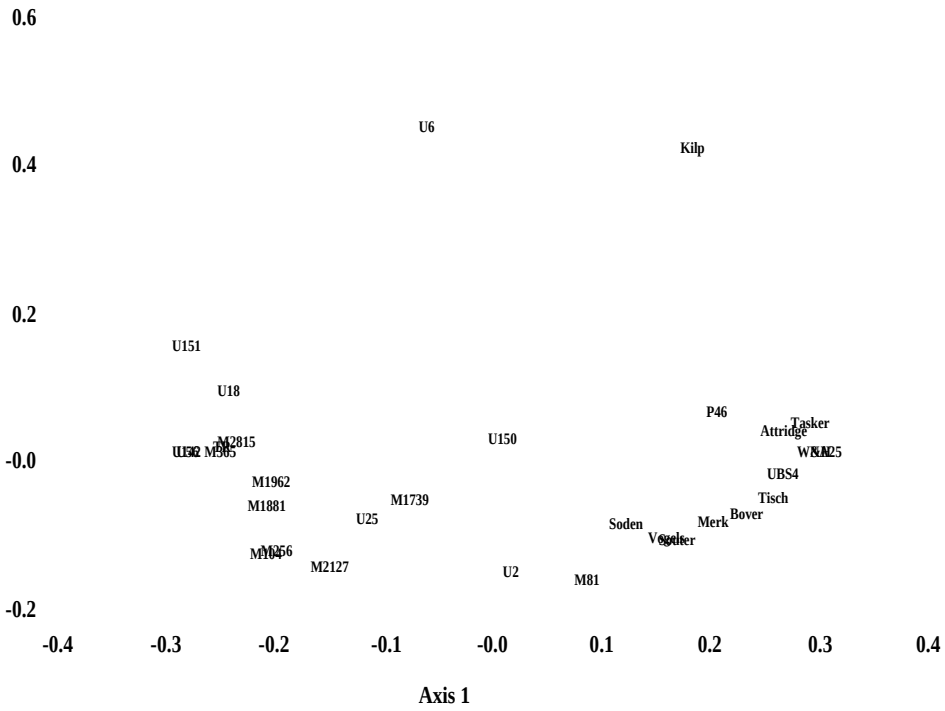
Attridge

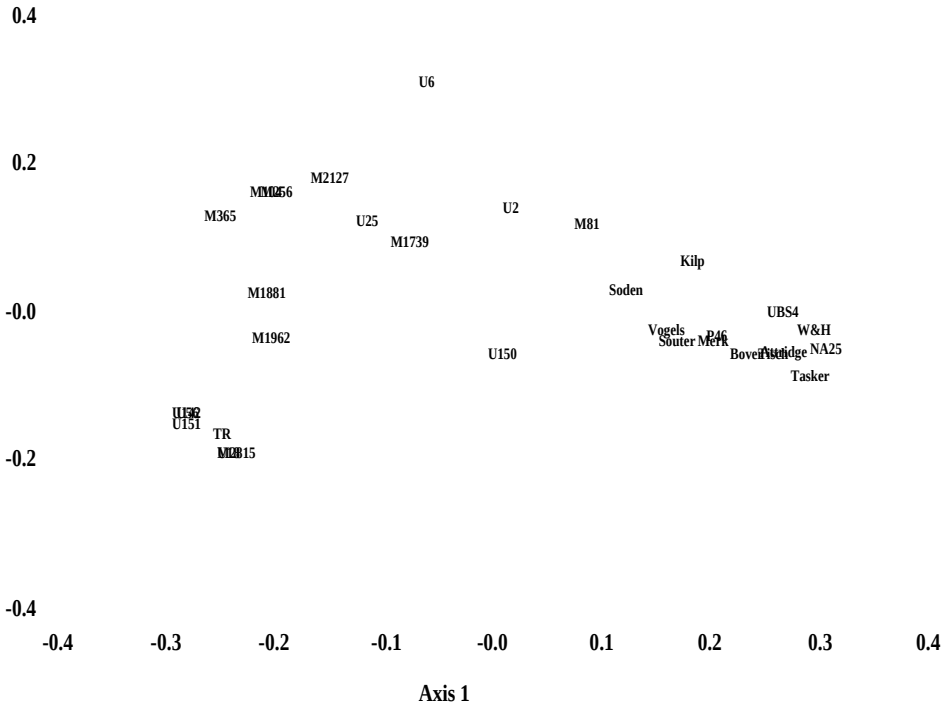




71 units; 30%, 12%, 10%

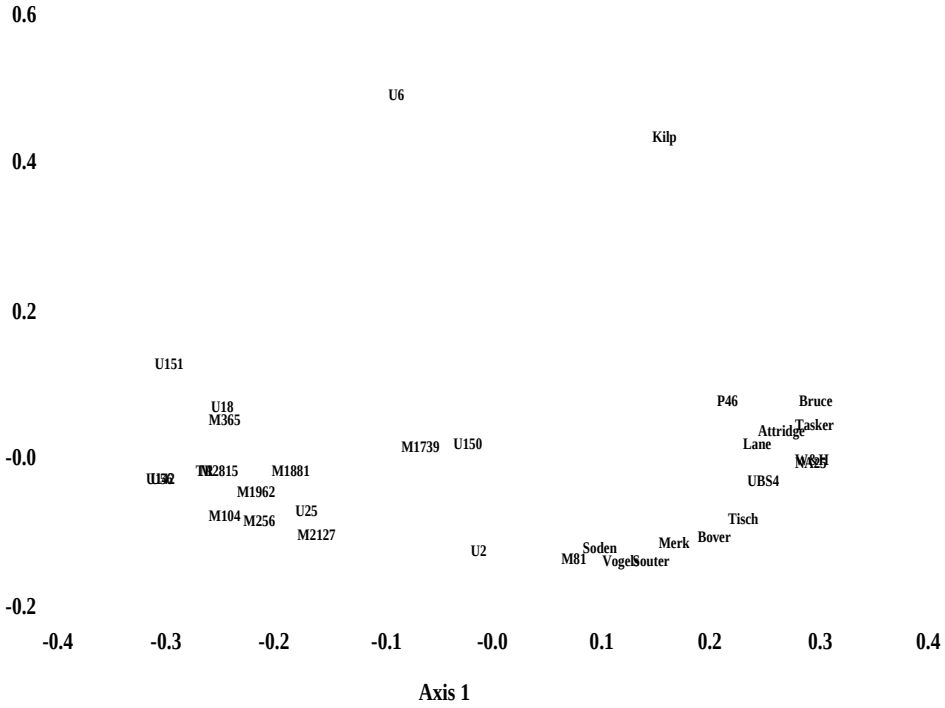
Bover

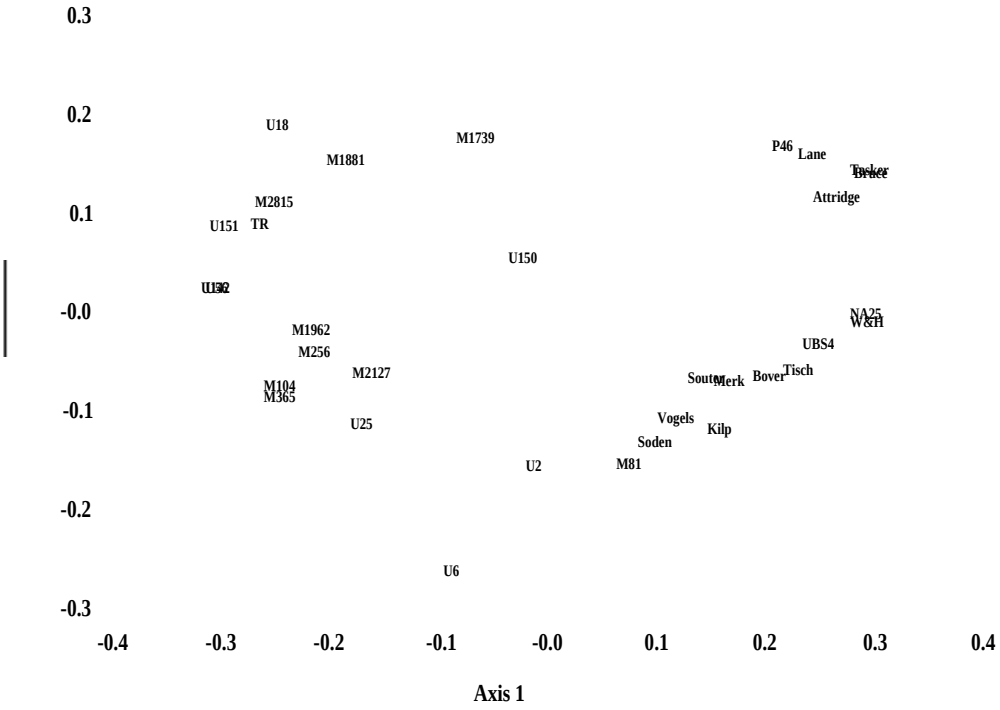




71 units; 30%, 12%, 10%

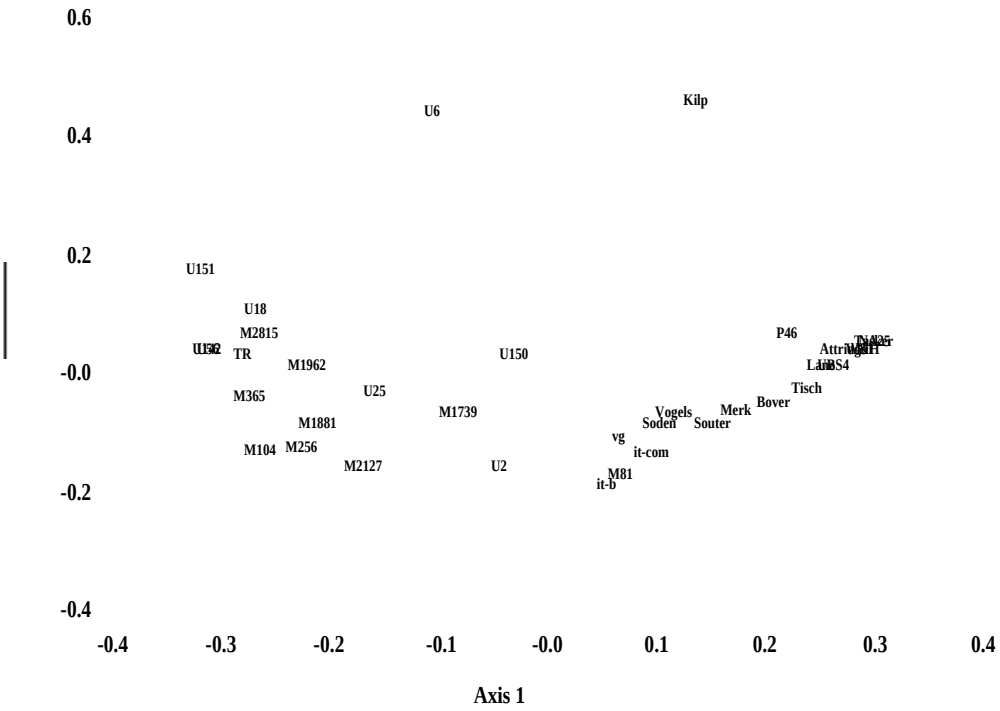
Bruce

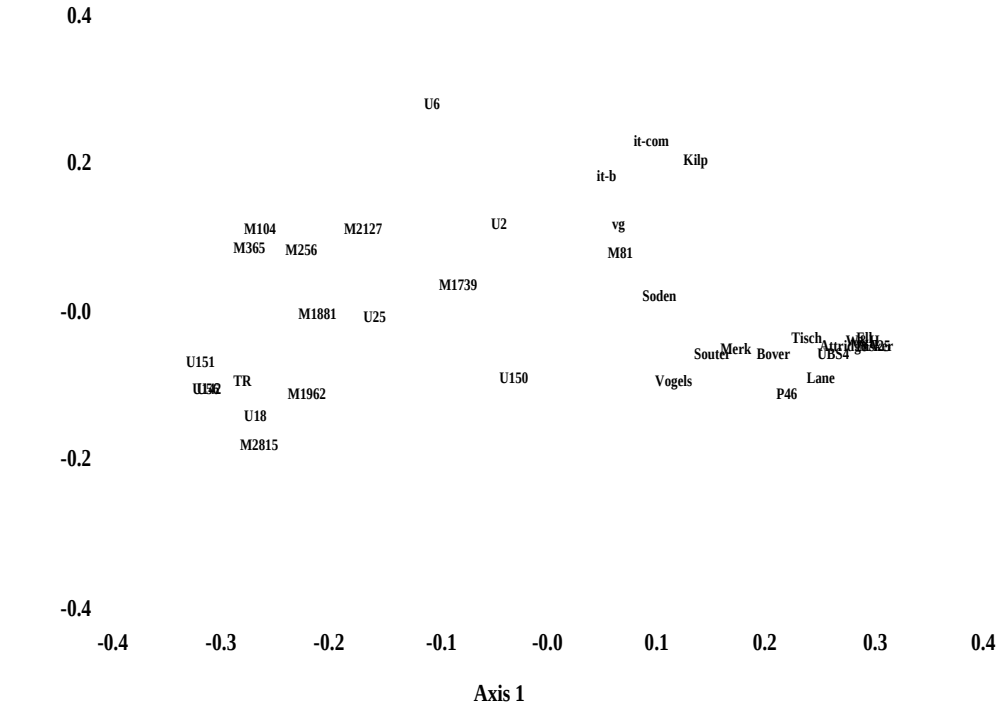




65 units; 32%, 13%, 9%

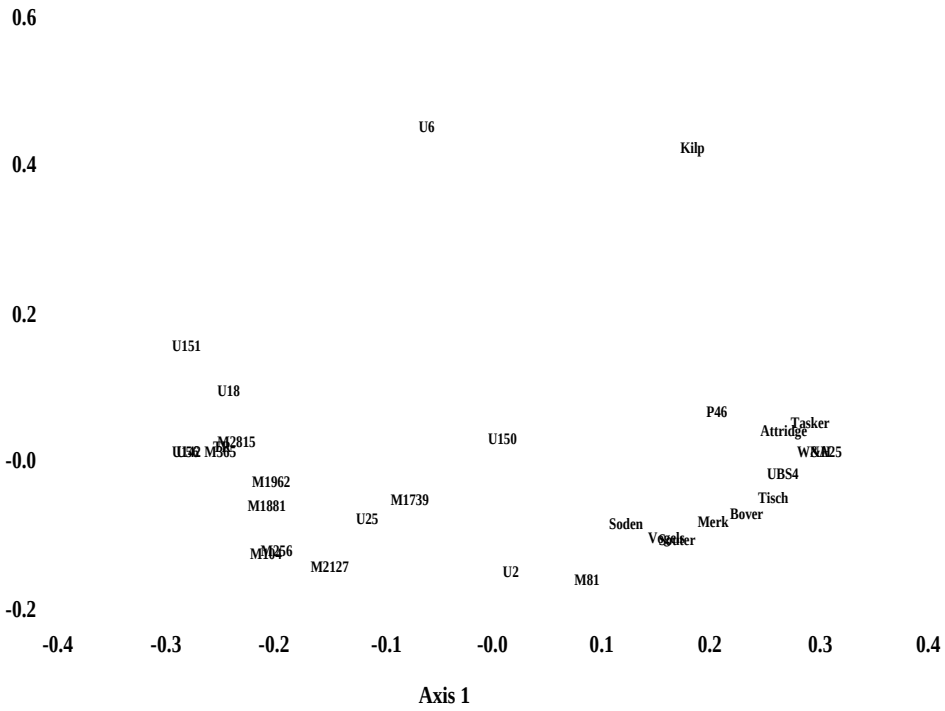
Ellingworth

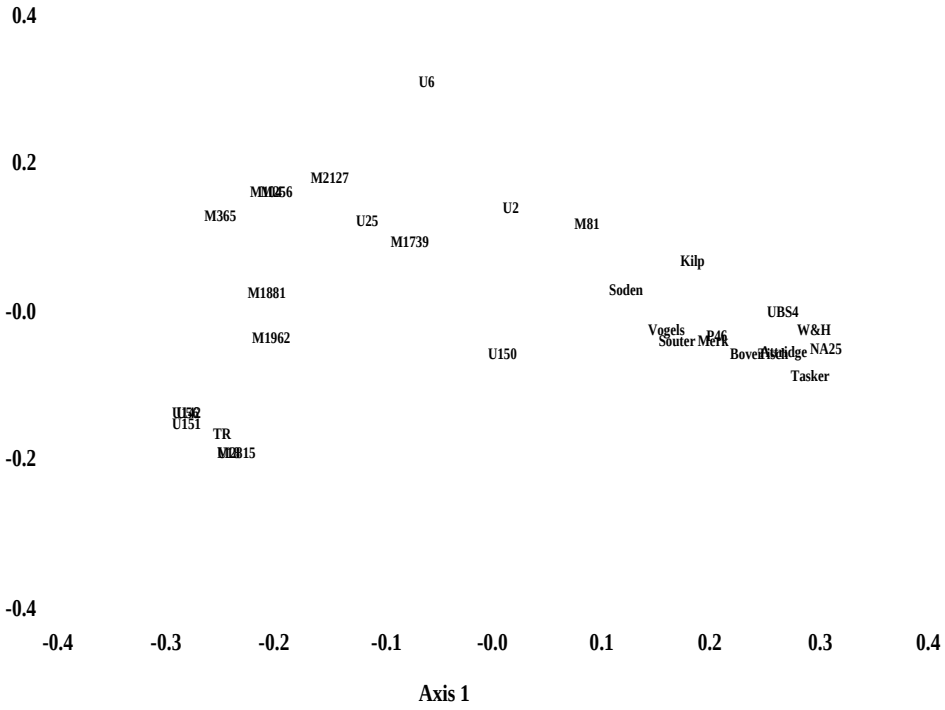




58 units; 32%, 13%, 8%

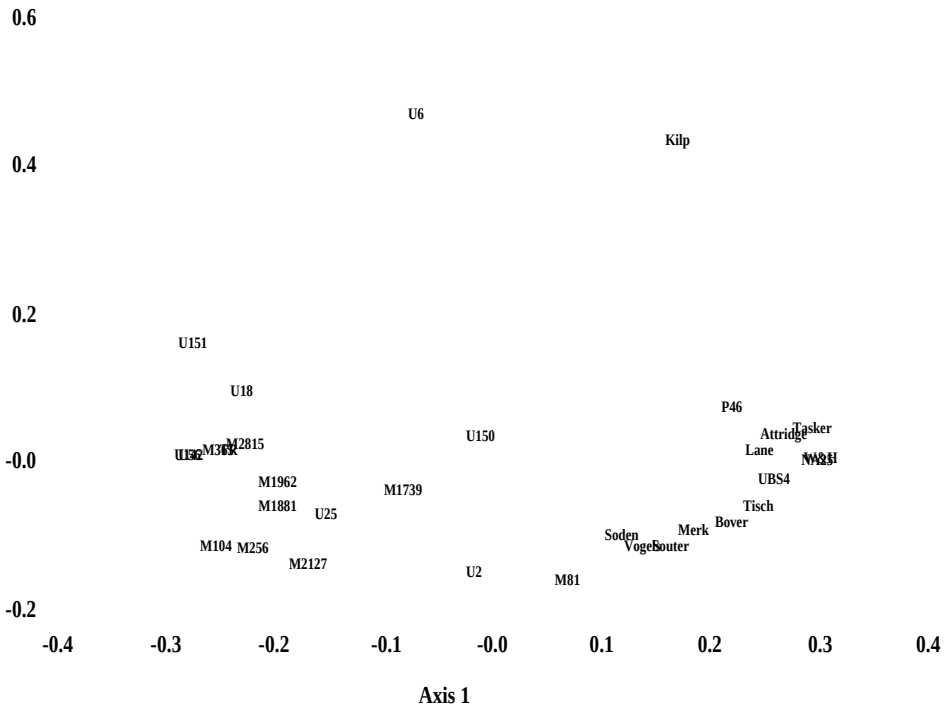
Kilpatrick

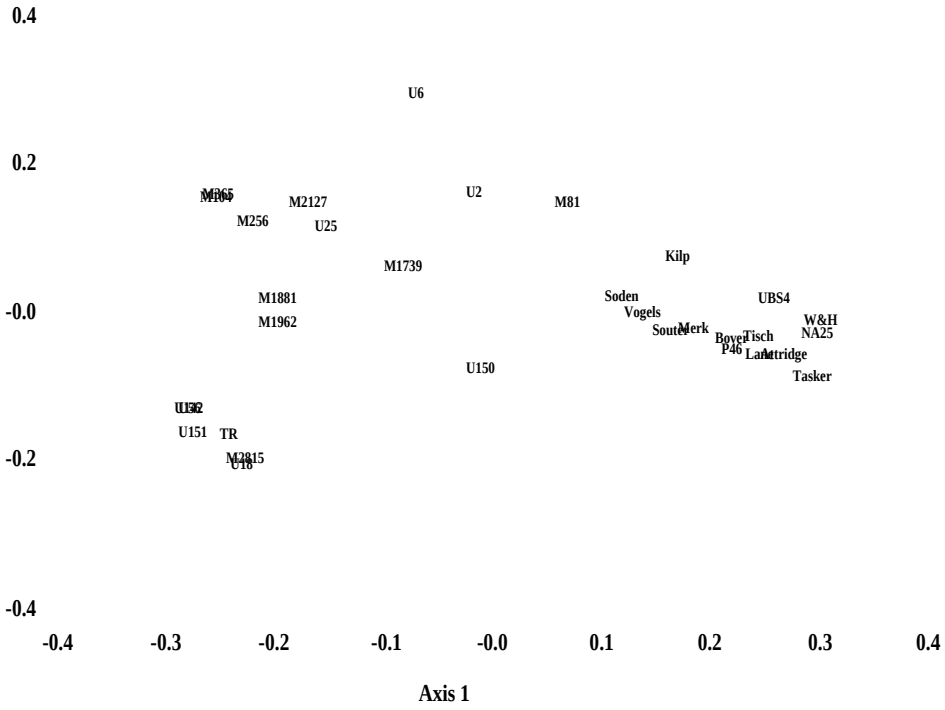




71 units; 30%, 12%, 10%

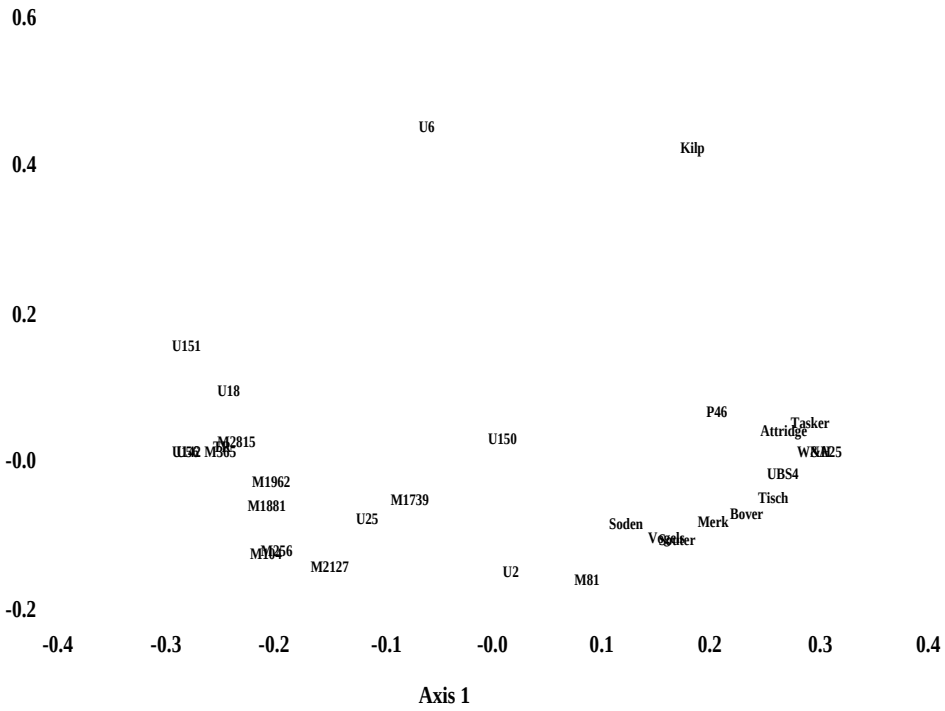
Lane

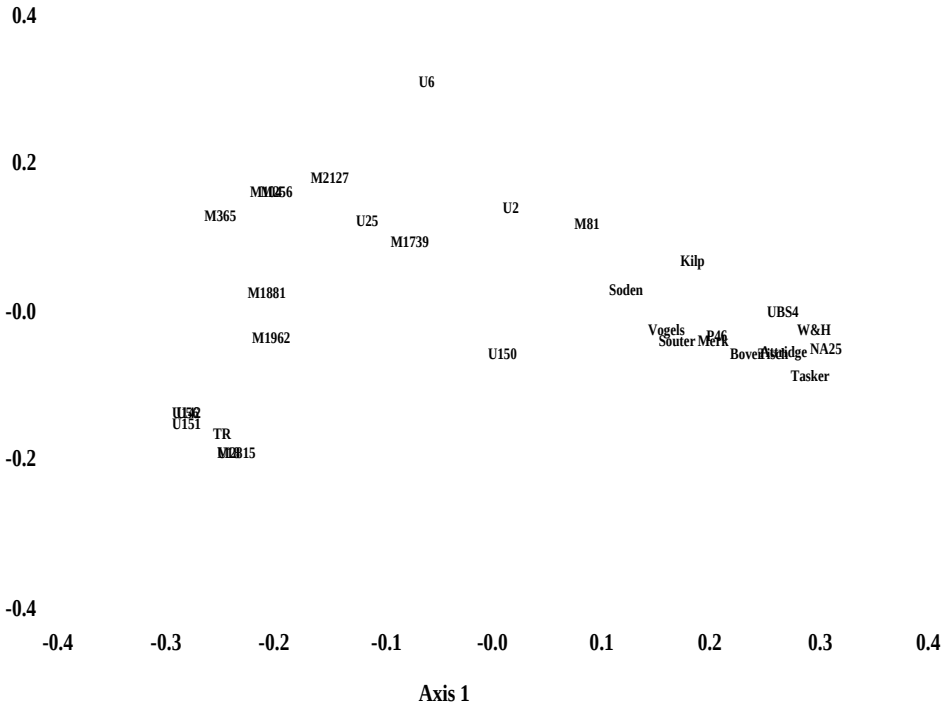




67 units; 31%, 13%, 9%

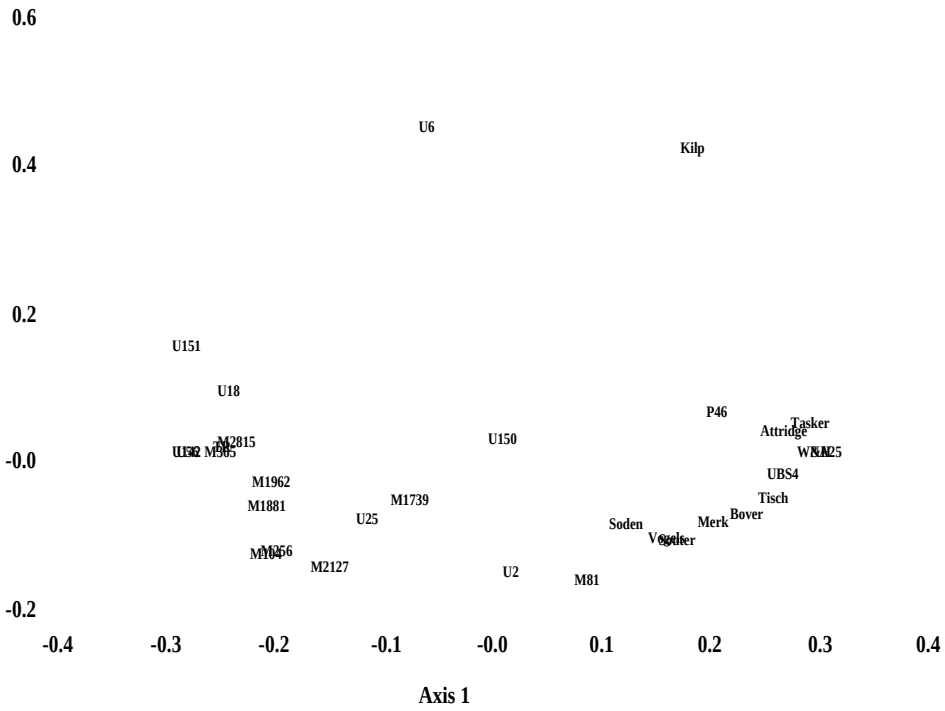
Merk

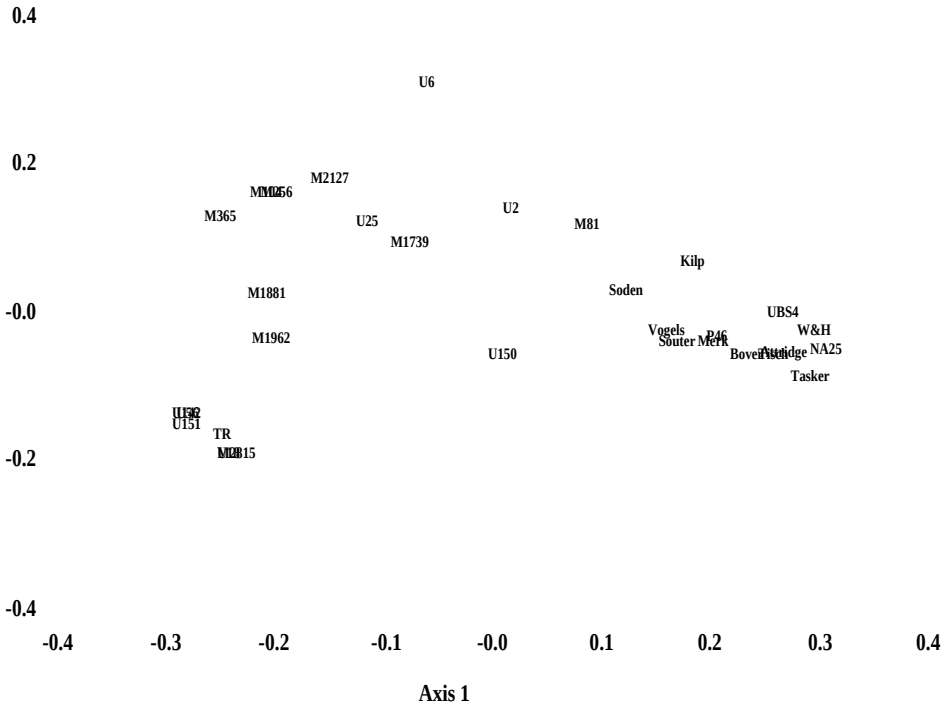




71 units; 30%, 12%, 10%

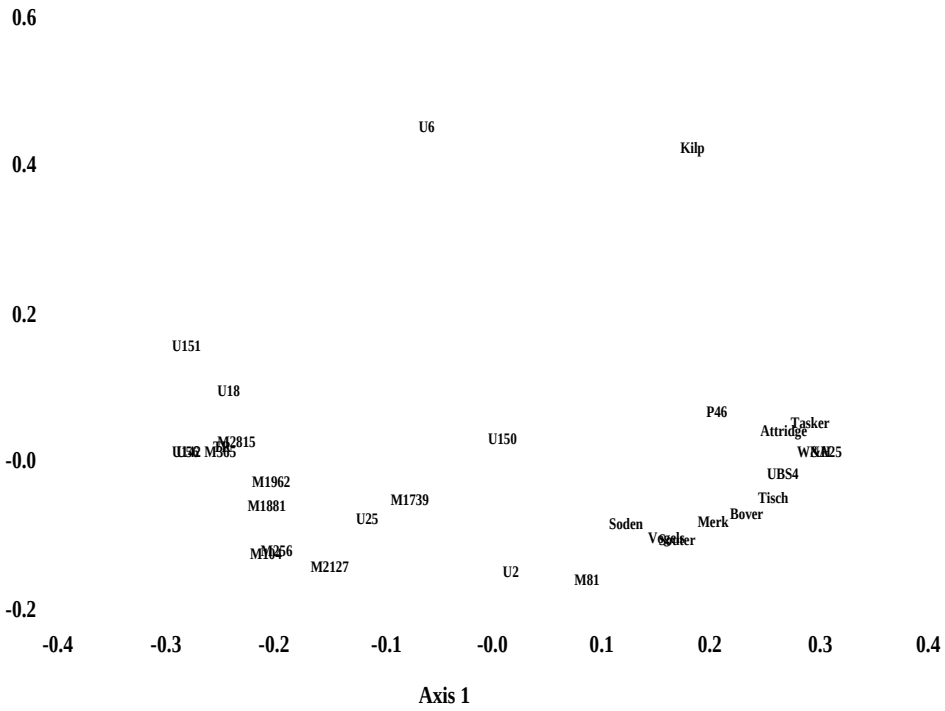
Nestle-Aland 25th ed.

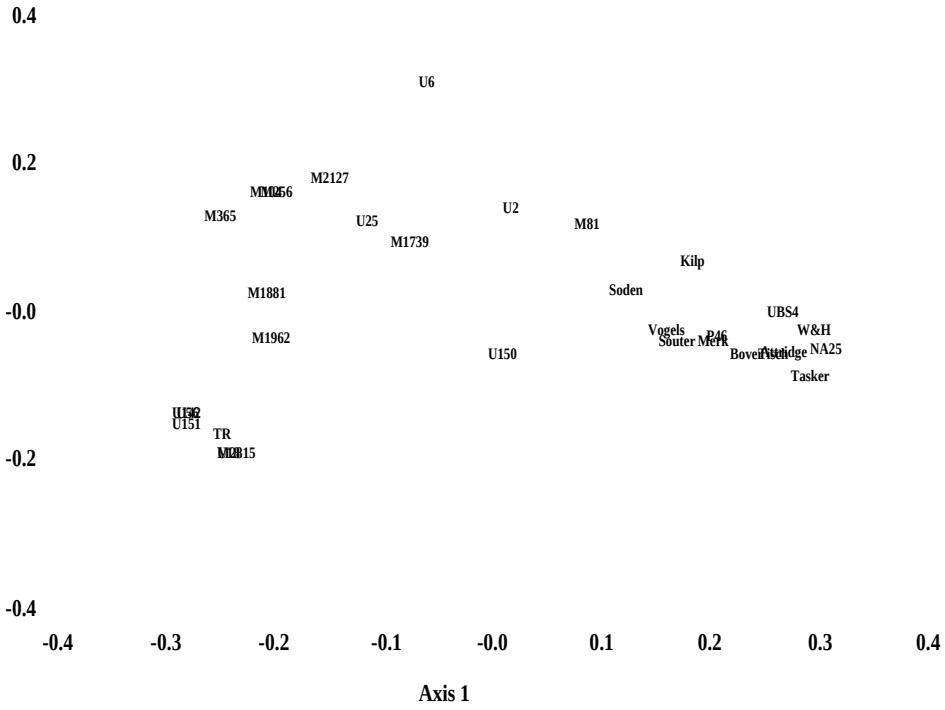




71 units; 30%, 12%, 10%

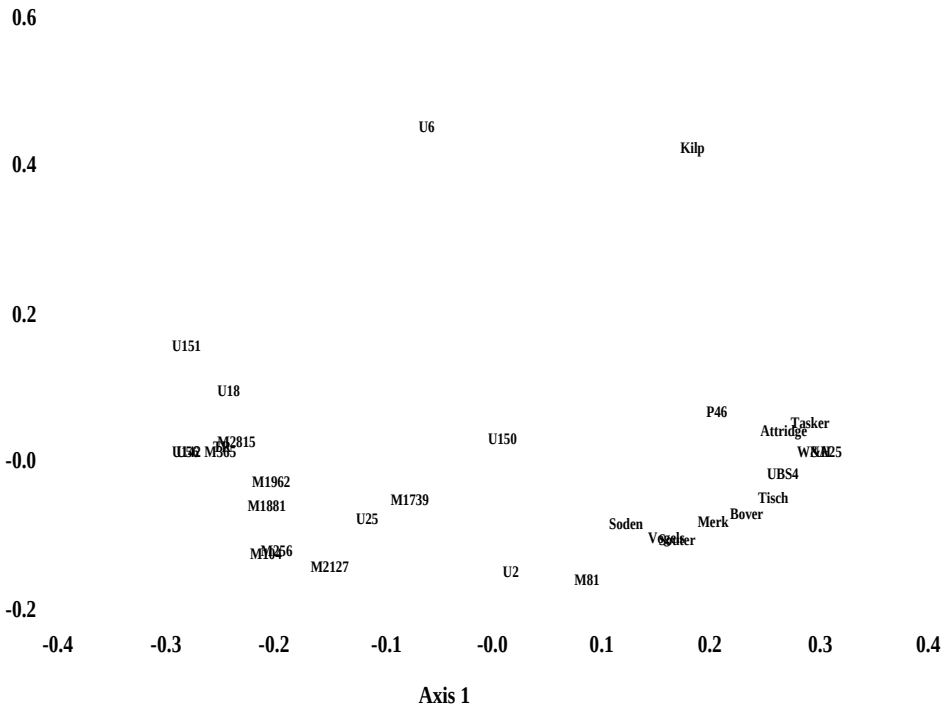
Souter

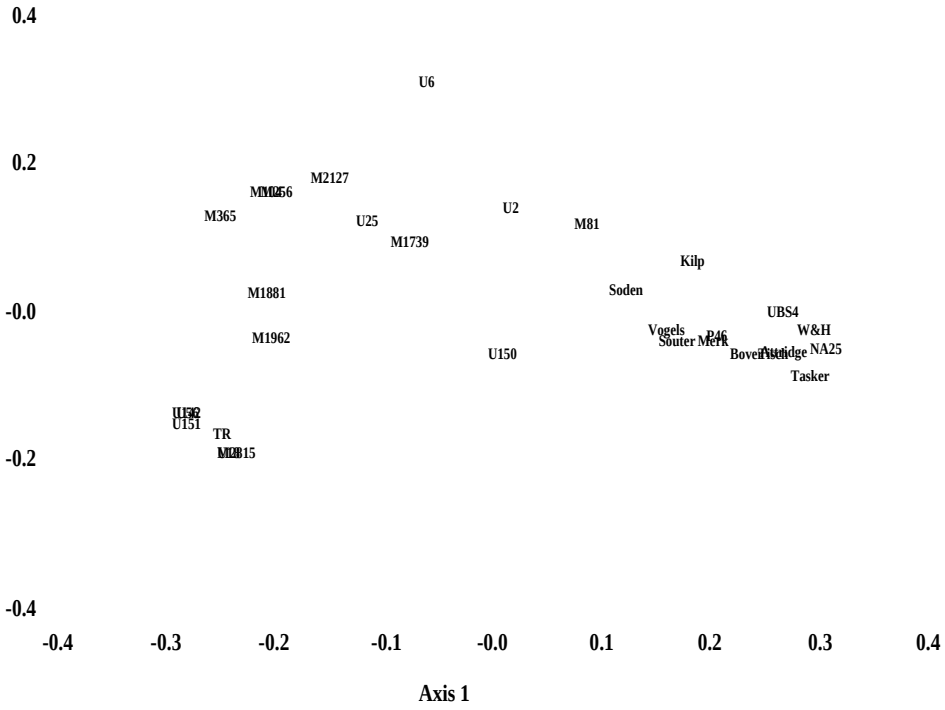




71 units; 30%, 12%, 10%

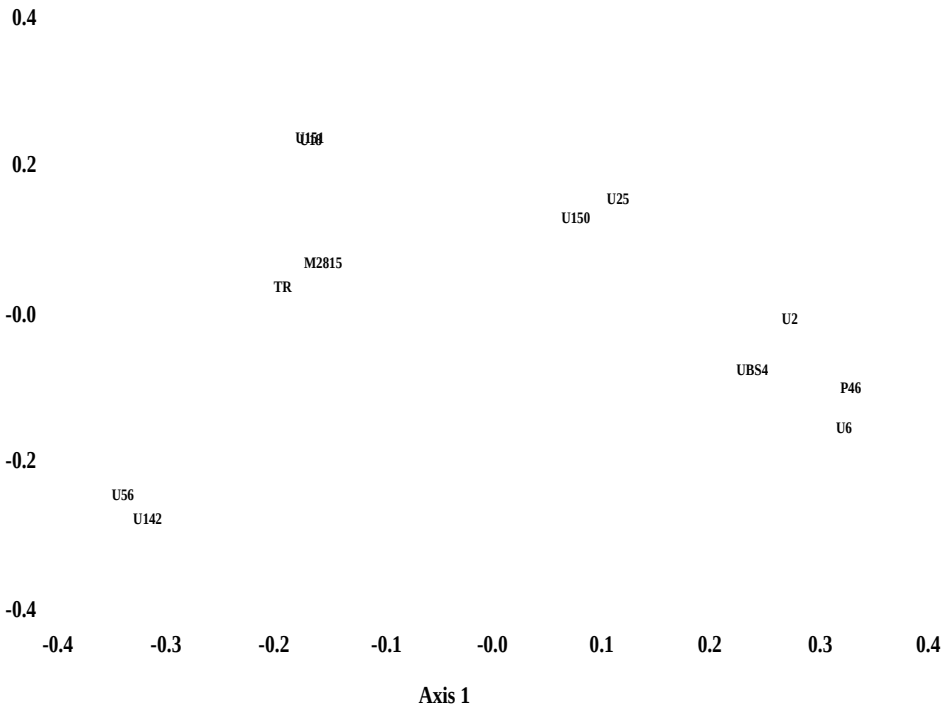
Tasker

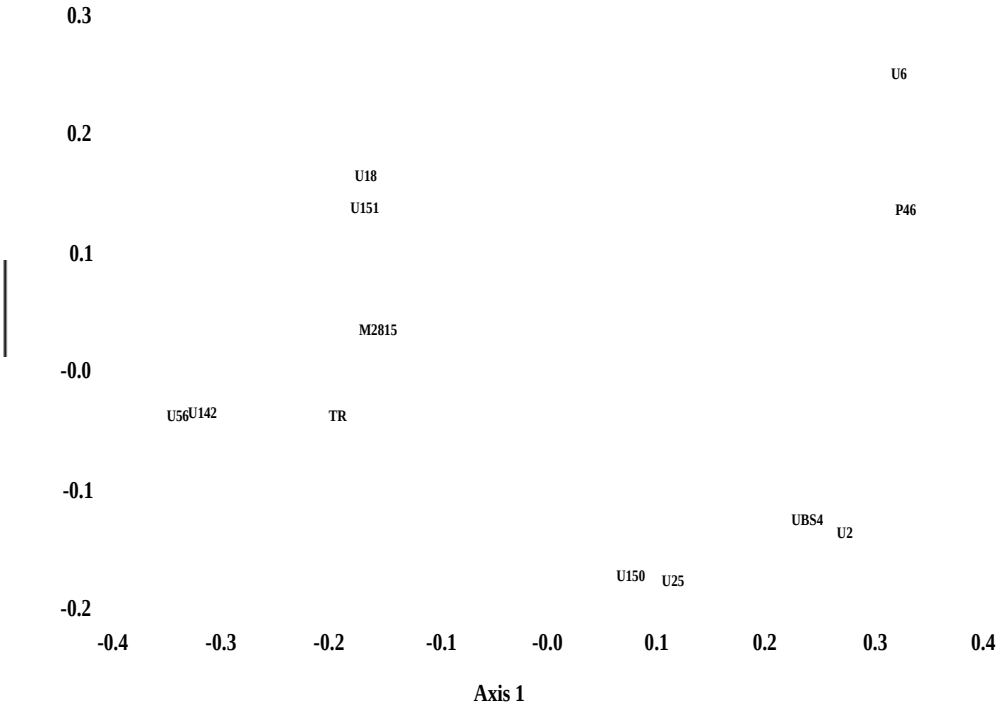




71 units; 30%, 12%, 10%

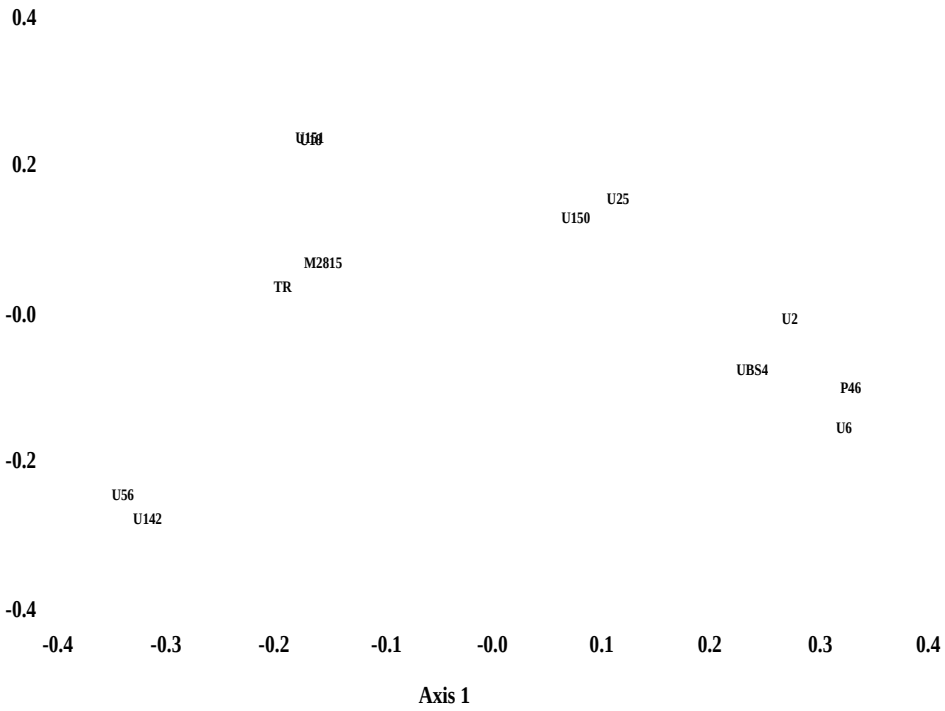
Textus Receptus

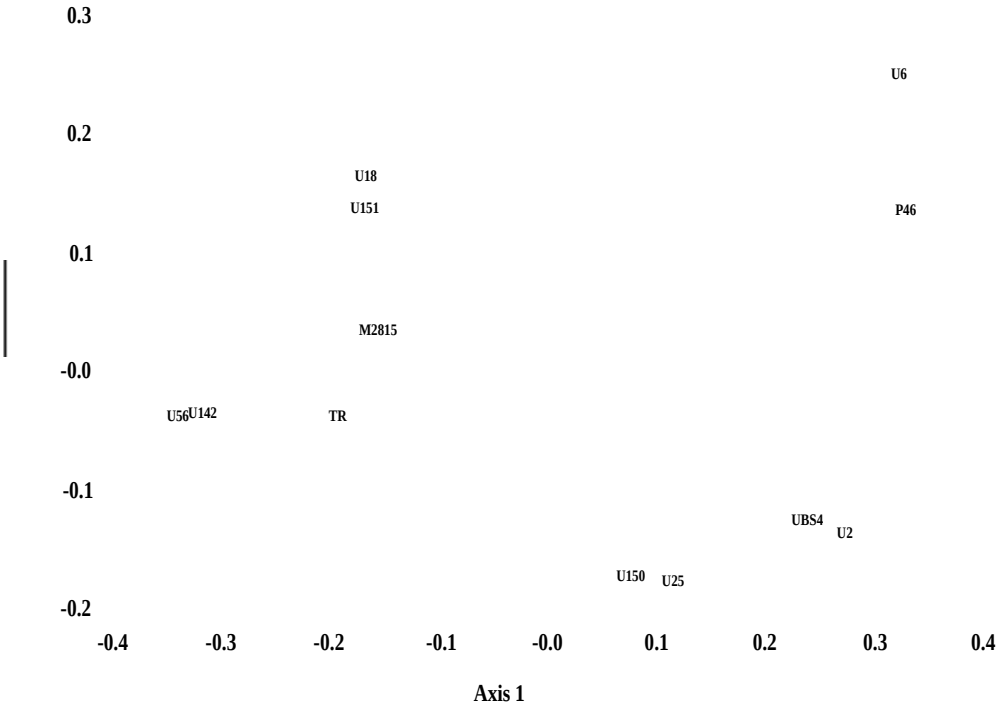




728 units; 32%, 15%, 10%

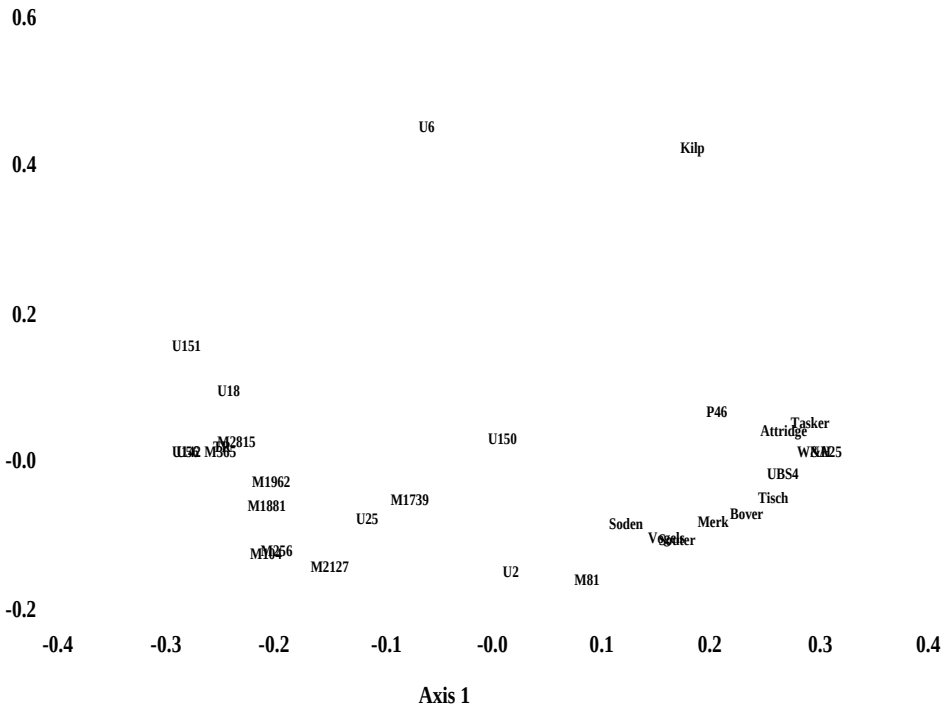
United Bible Societies 4th ed.

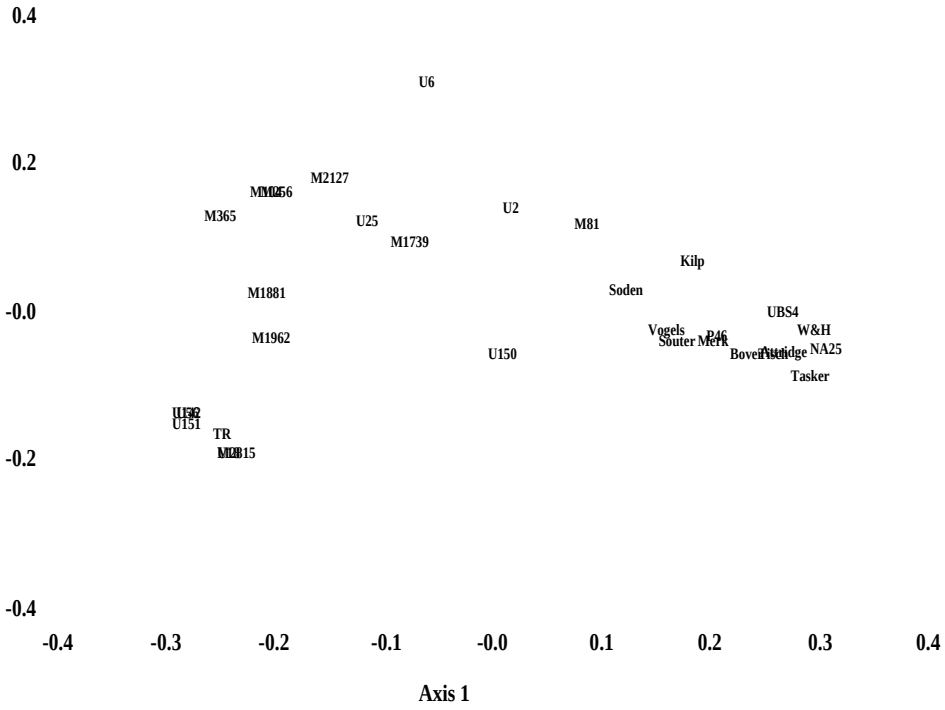




728 units; 32%, 15%, 10%

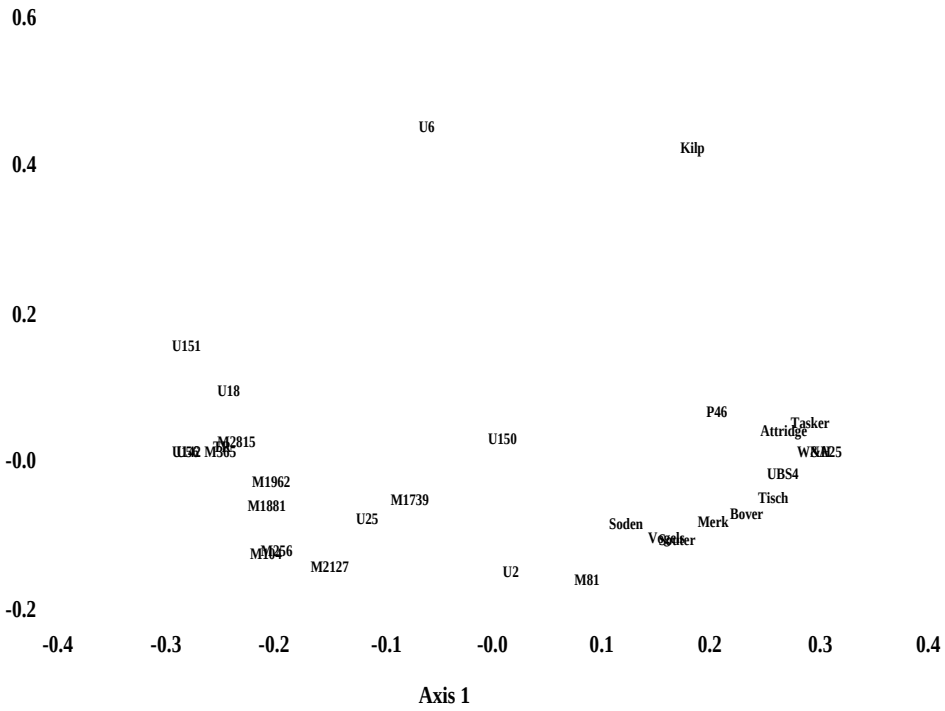
Vogels

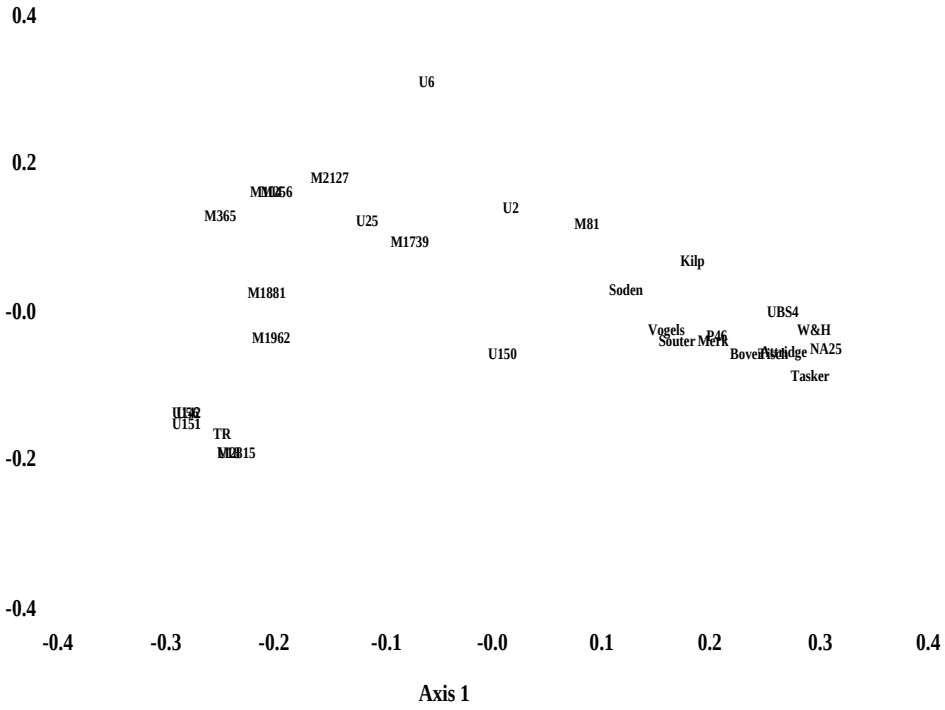




71 units; 30%, 12%, 10%

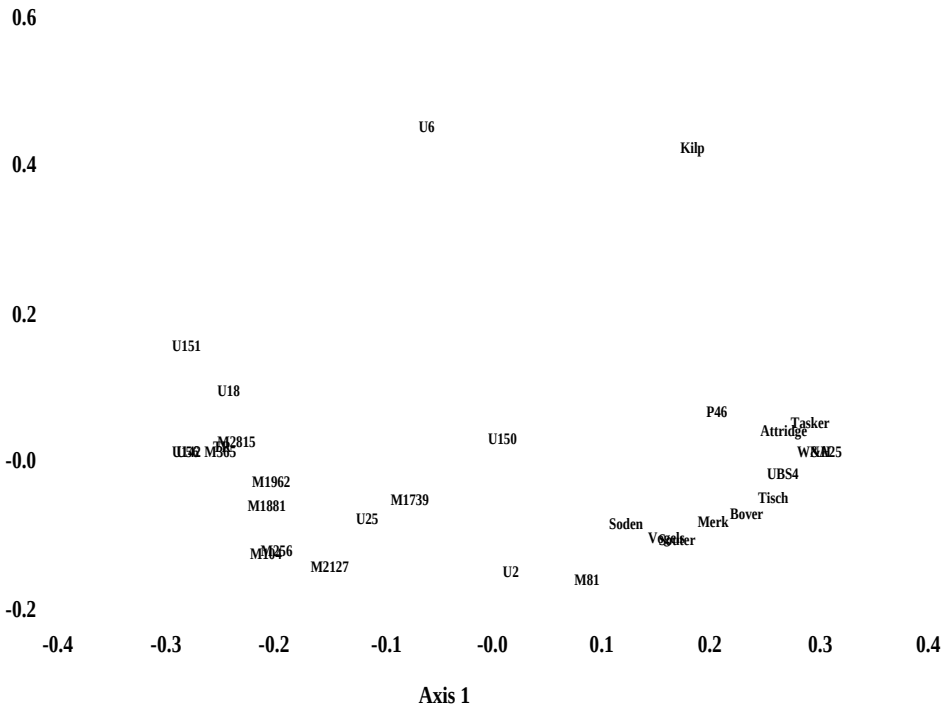
Von Soden

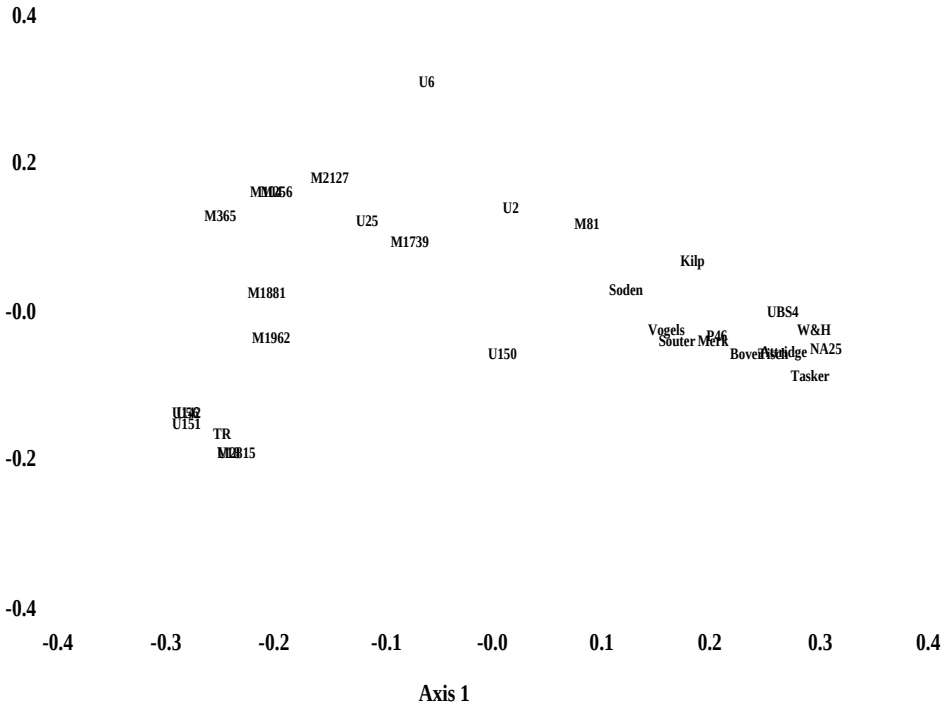




71 units; 30%, 12%, 10%

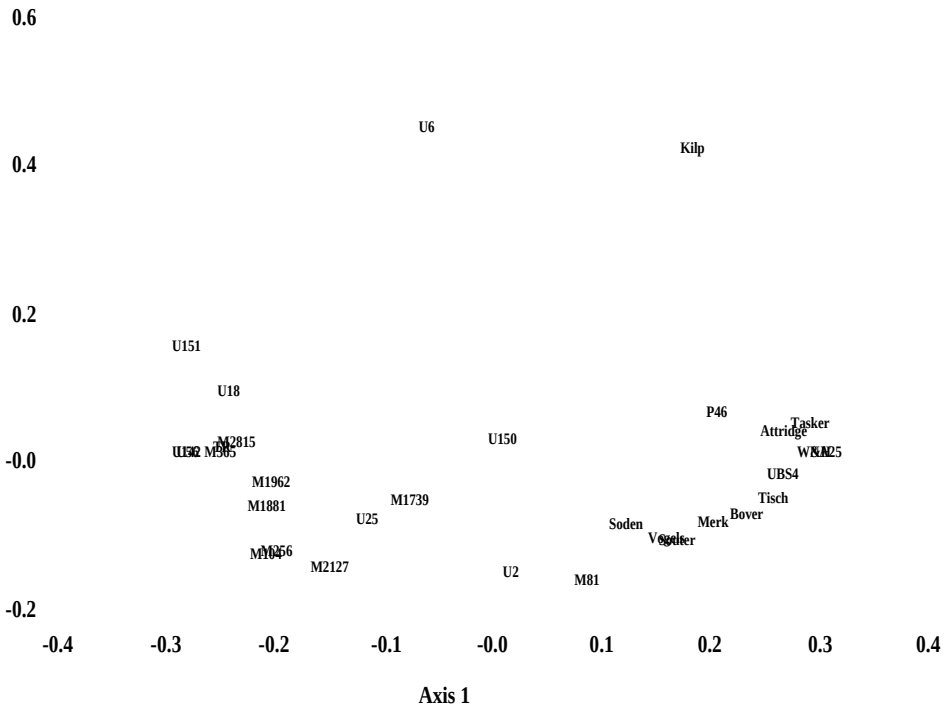
Von Tischendorf

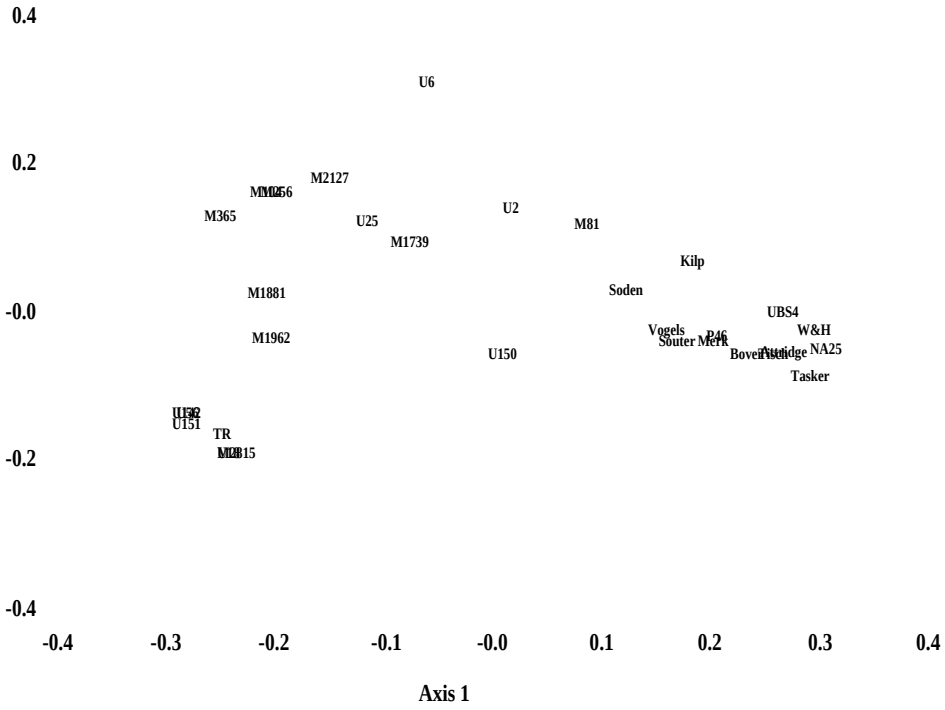




71 units; 30%, 12%, 10%

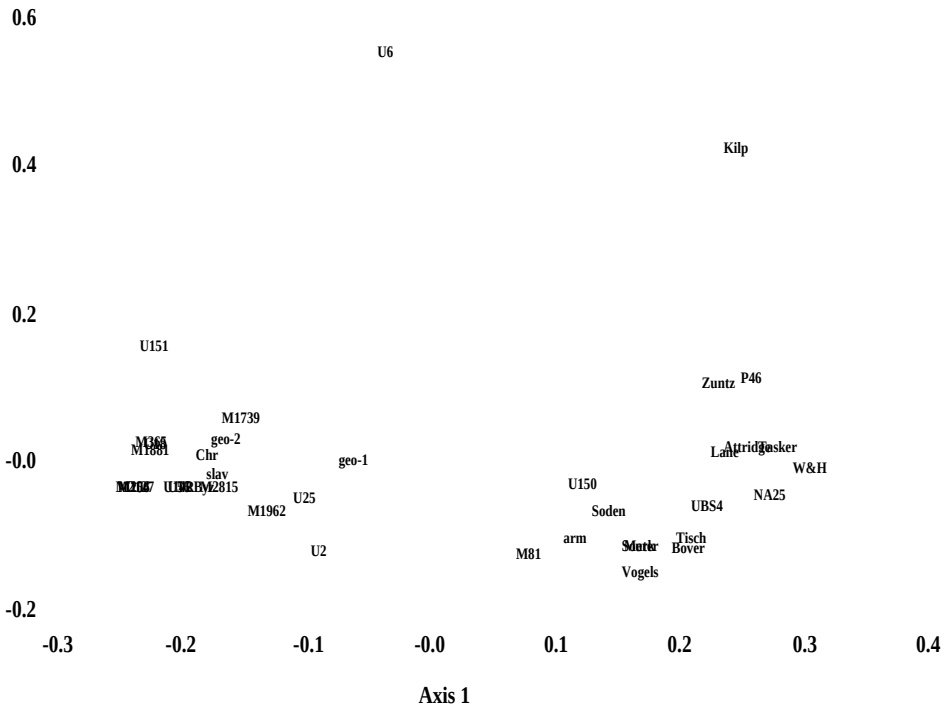
Westcott and Hort

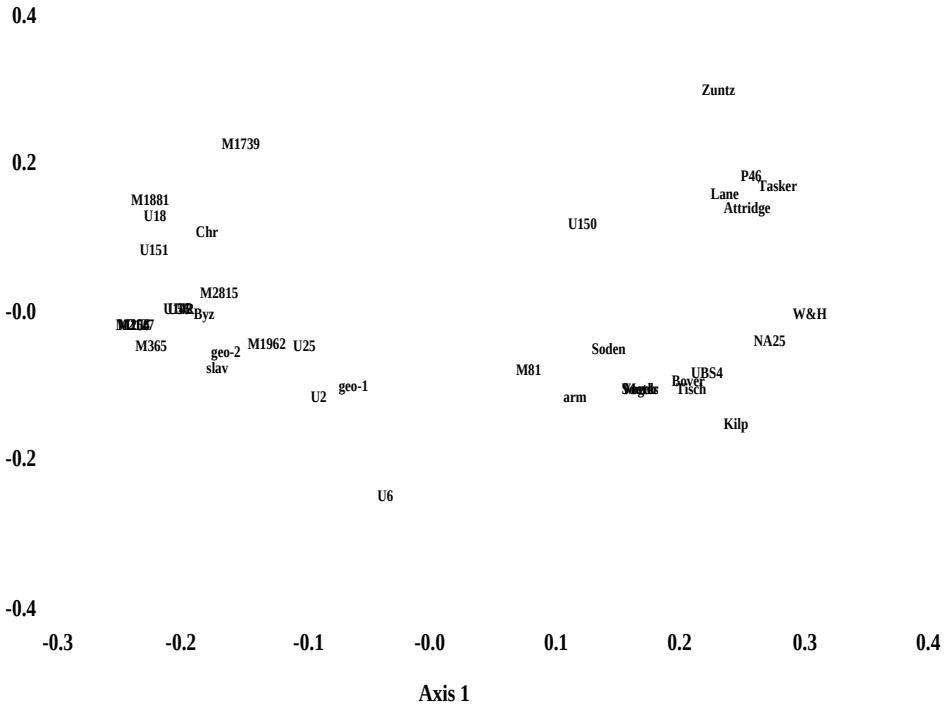




71 units; 30%, 12%, 10%

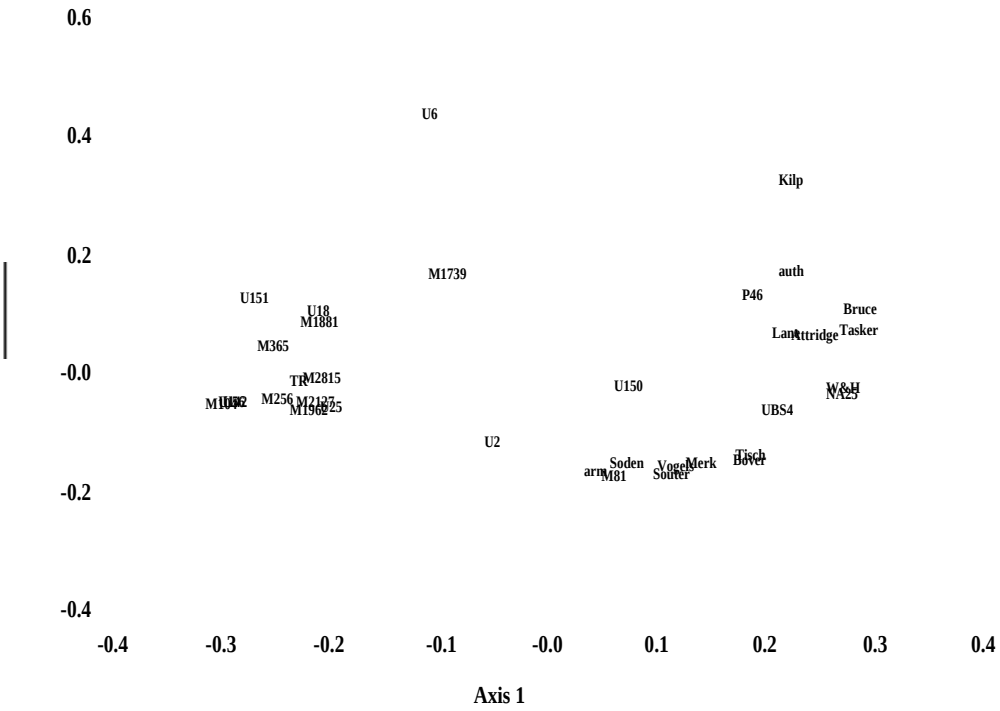
Zuntz

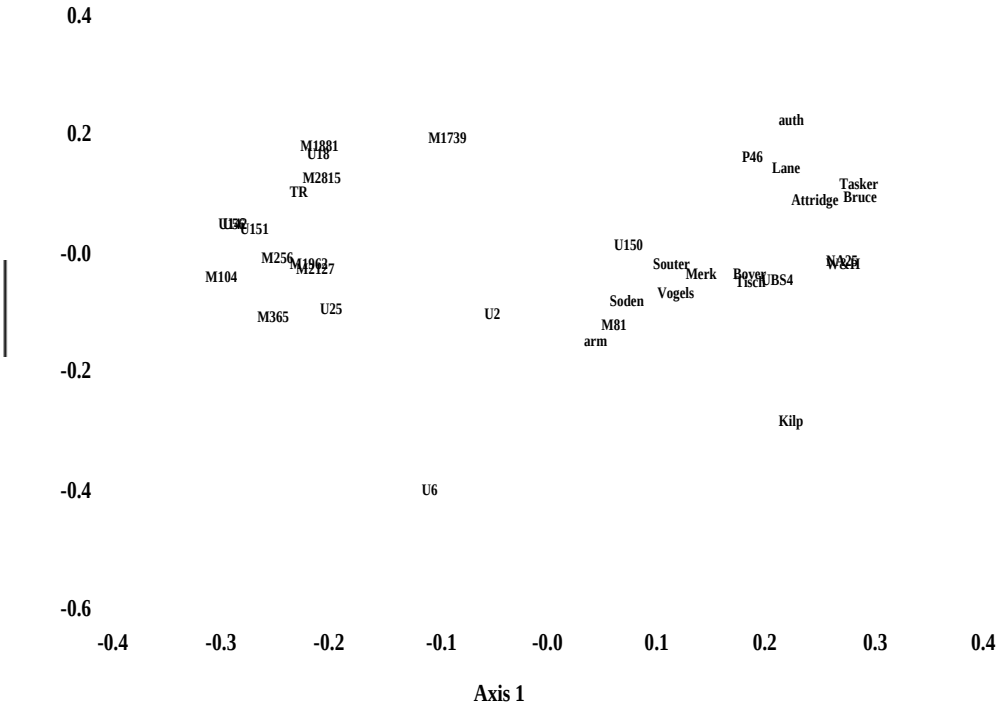




51 units; 32%, 14%, 11%

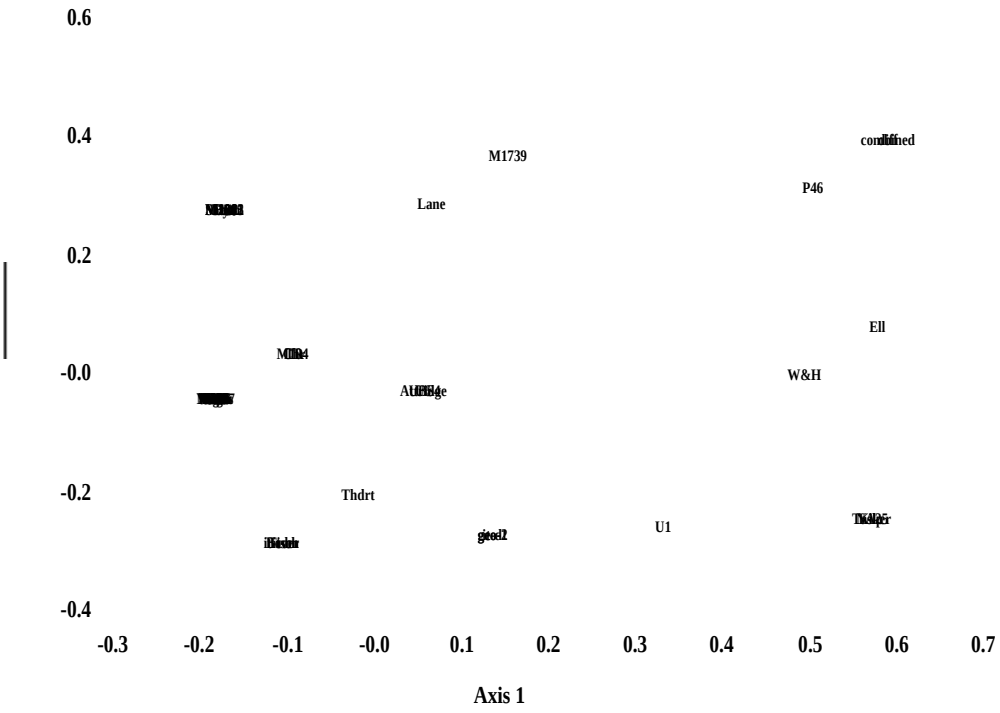
Authentic

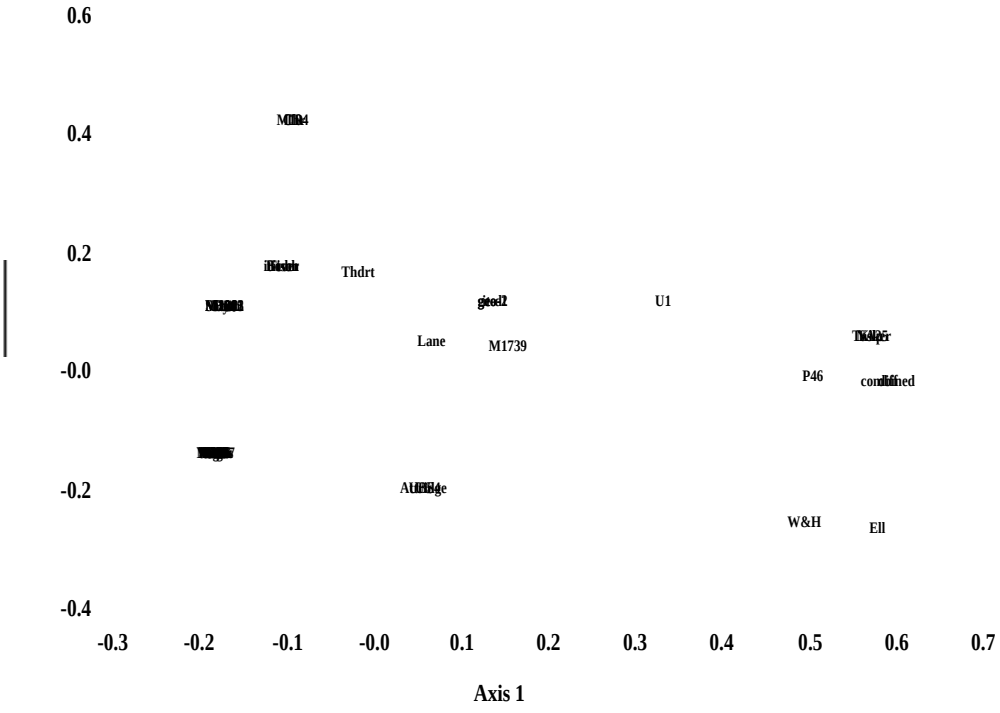




55 units; 30%, 13%, 12%

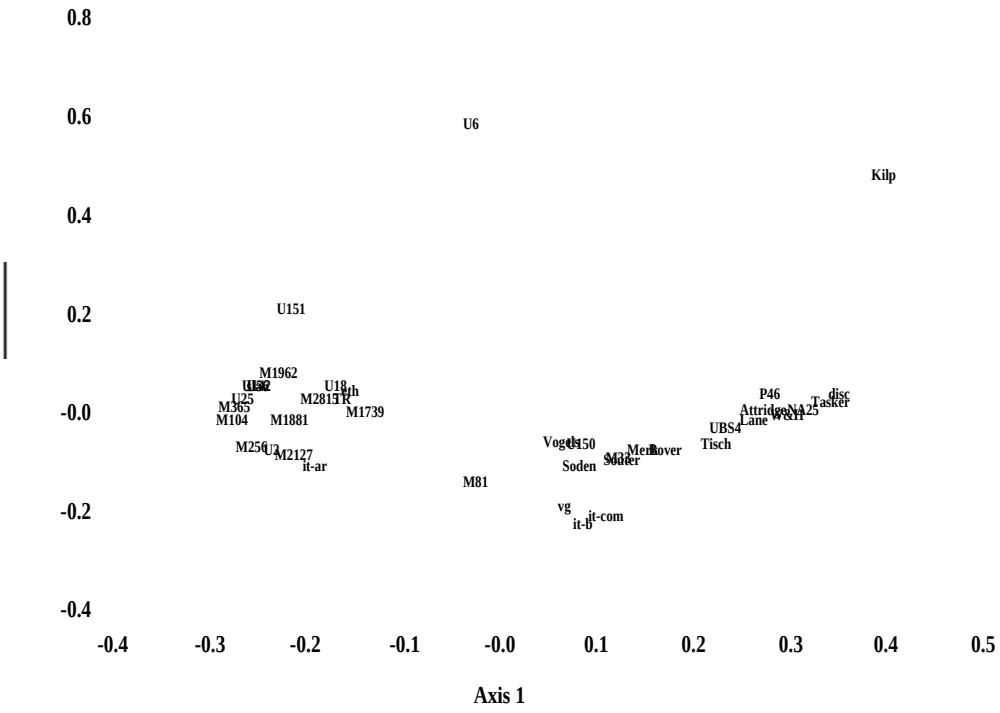
Difficult

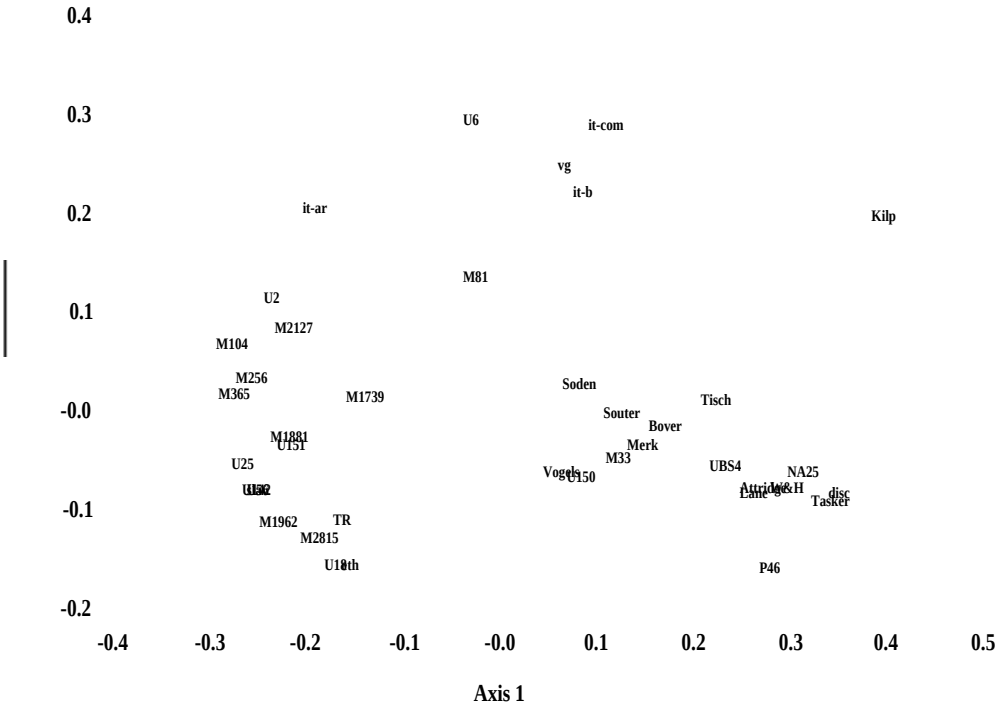




6 units; 42%, 24%, 17%

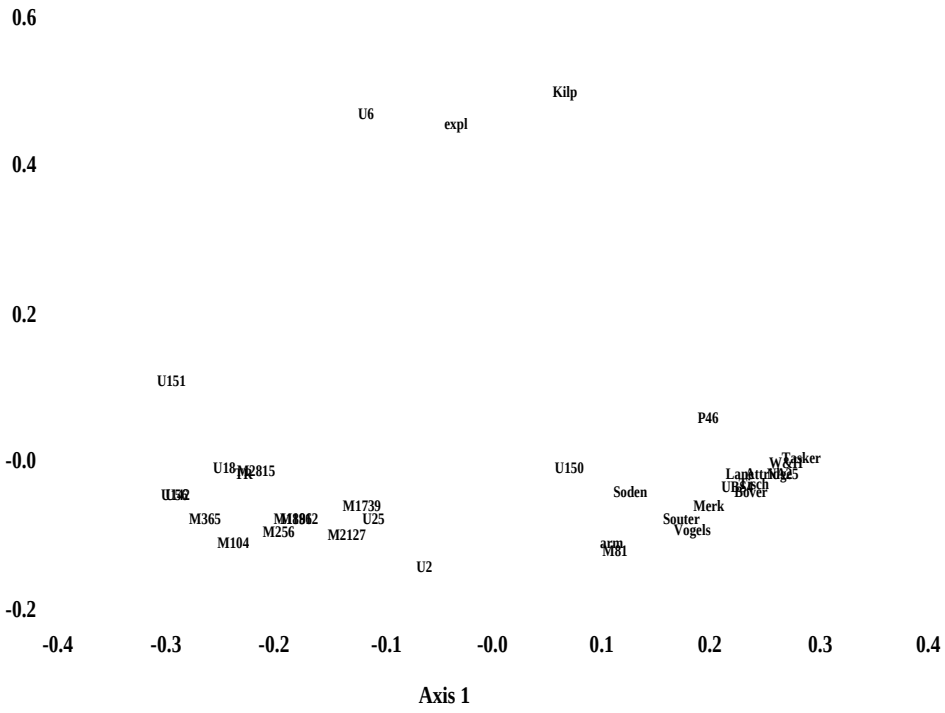
Discordant

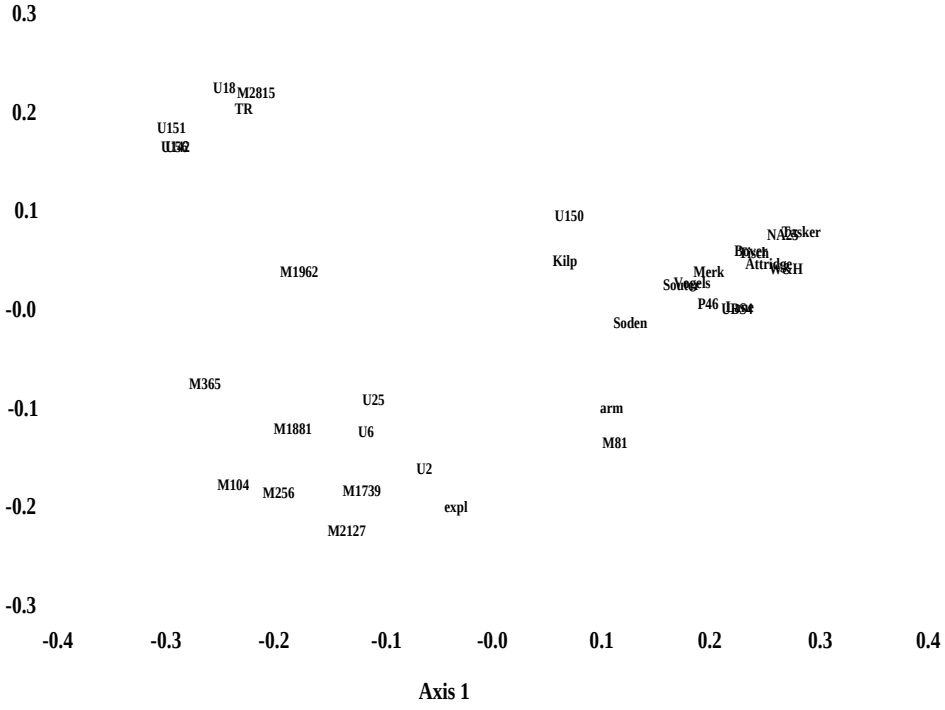




36 units; 36%, 16%, 11%

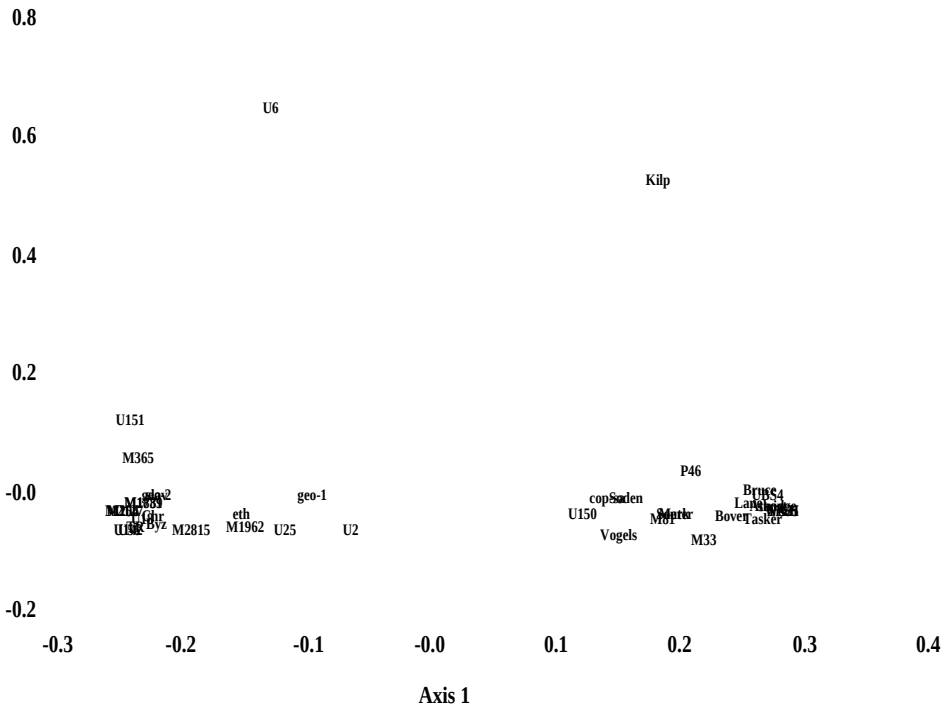
Explanatory

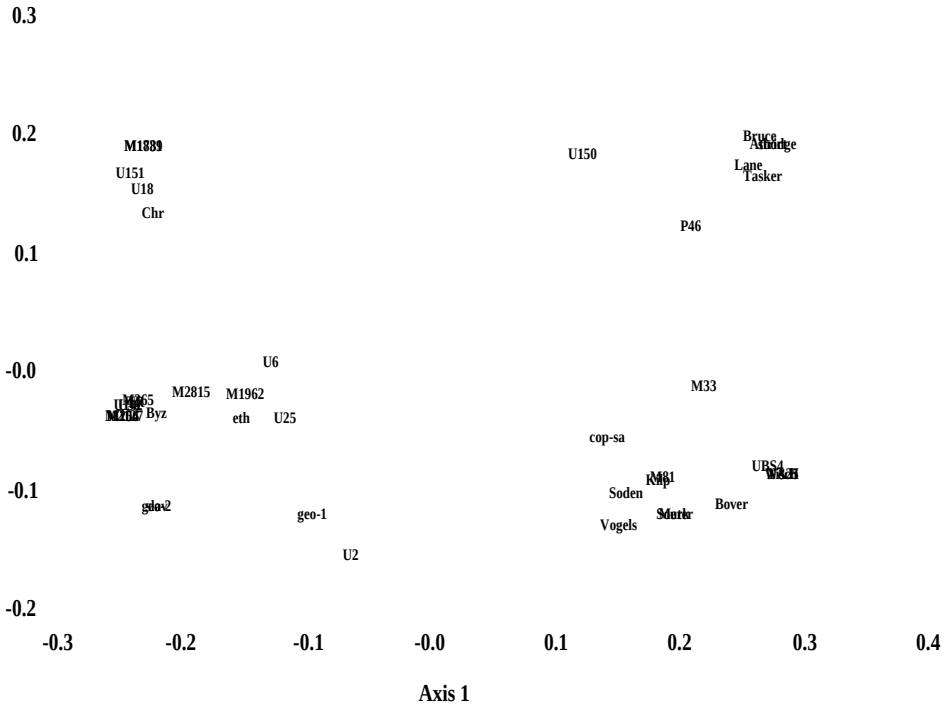




55 units; 29%, 17%, 11%

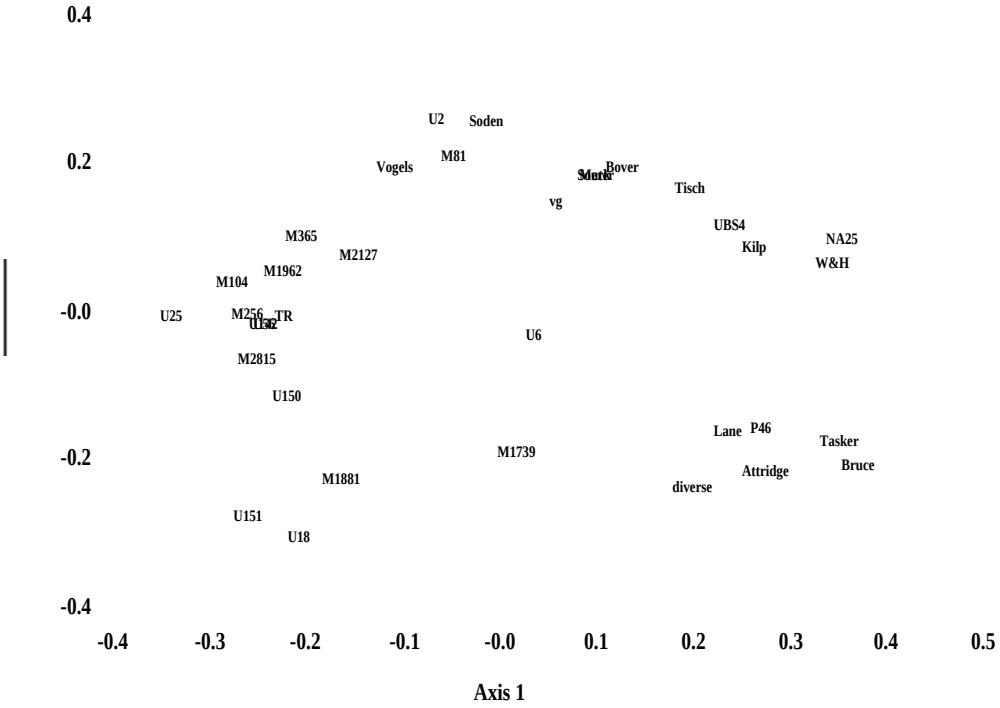
Short

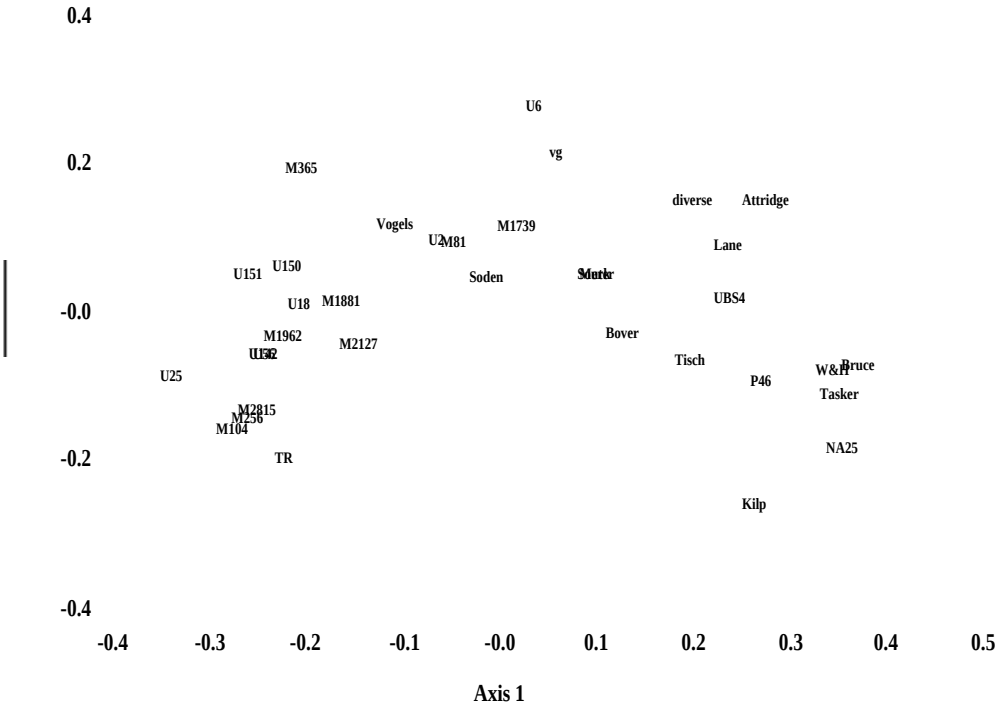




40 units; 39%, 15%, 11%

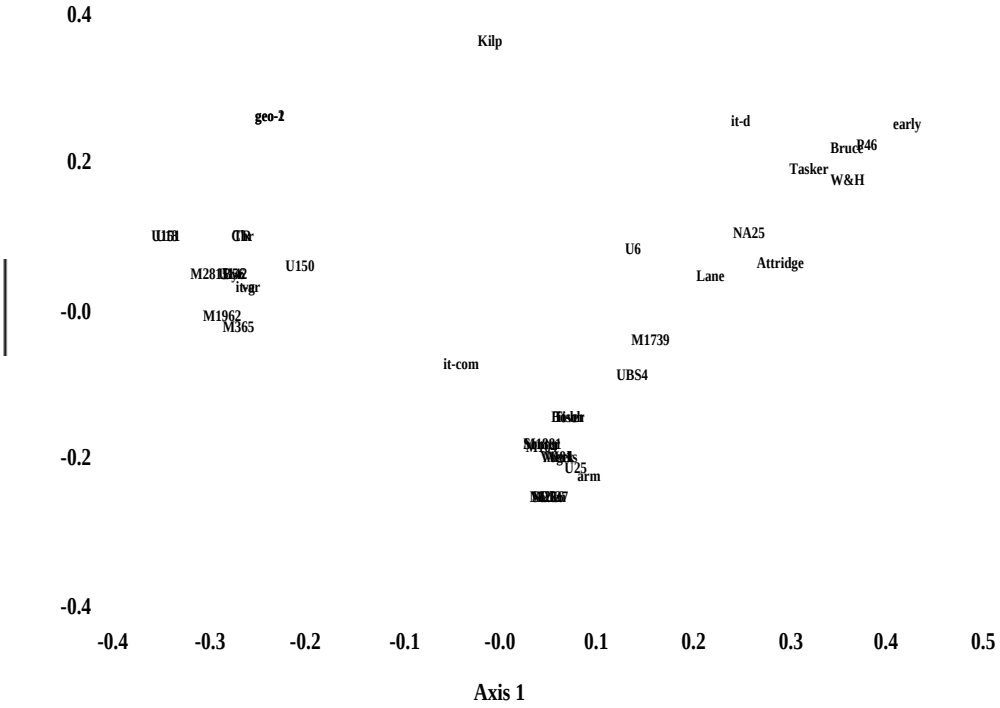
Diverse

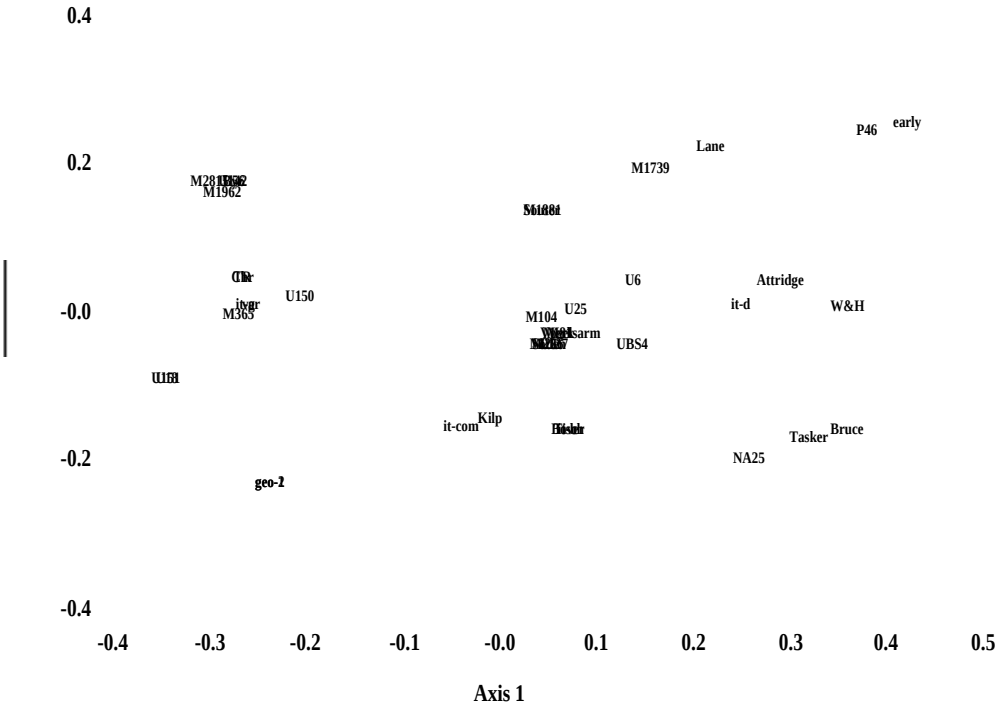




25 units; 31%, 17%, 9%

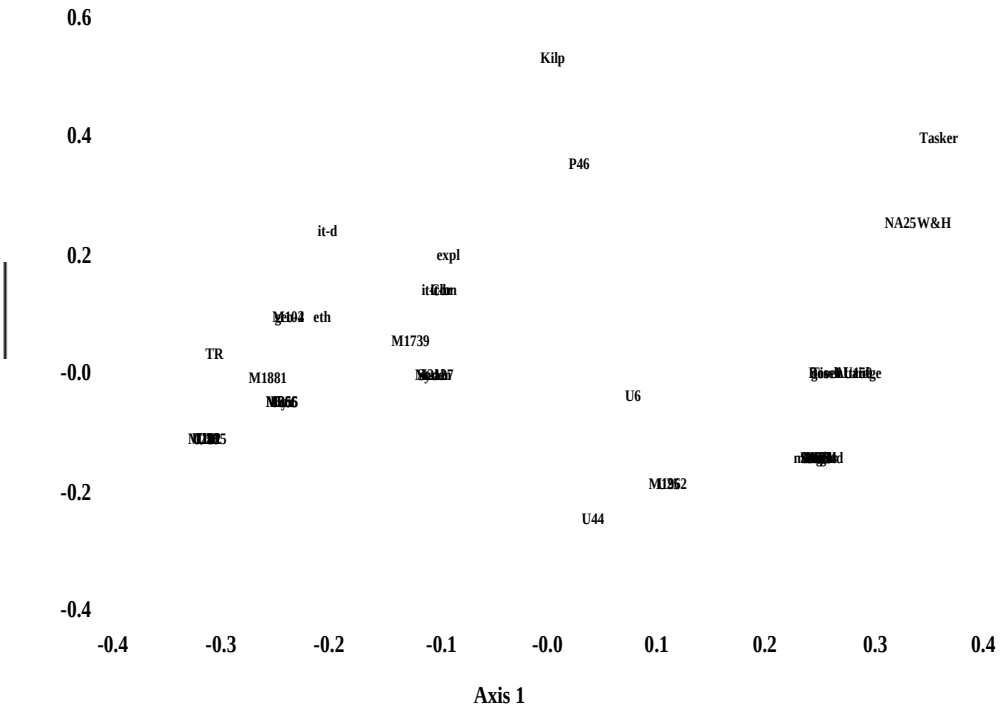
Early

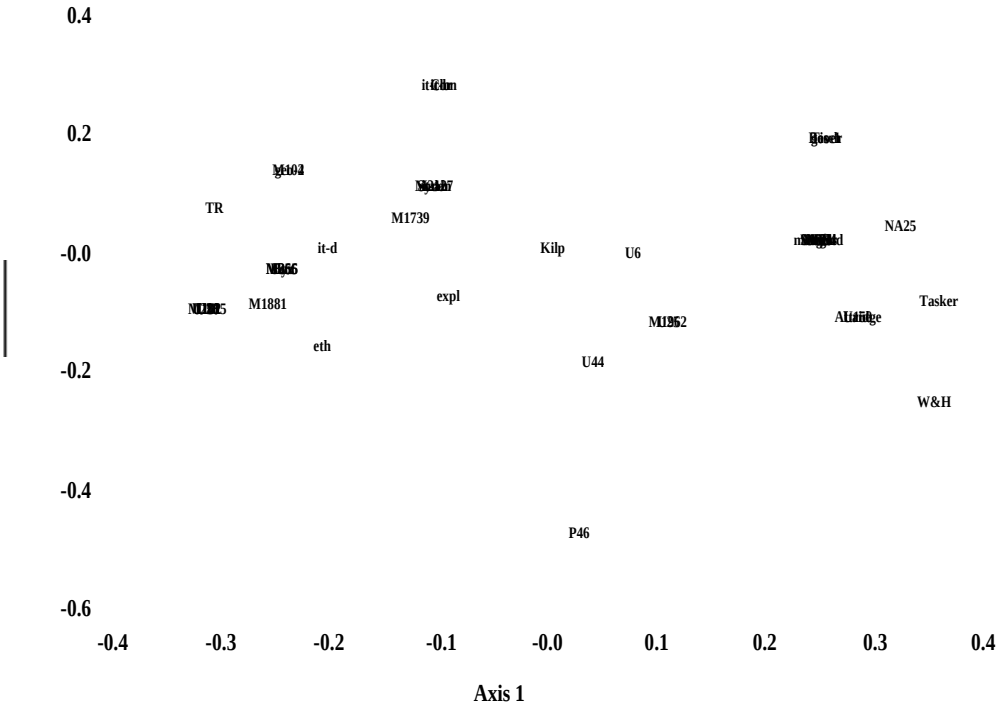




17 units; 32%, 19%, 11%

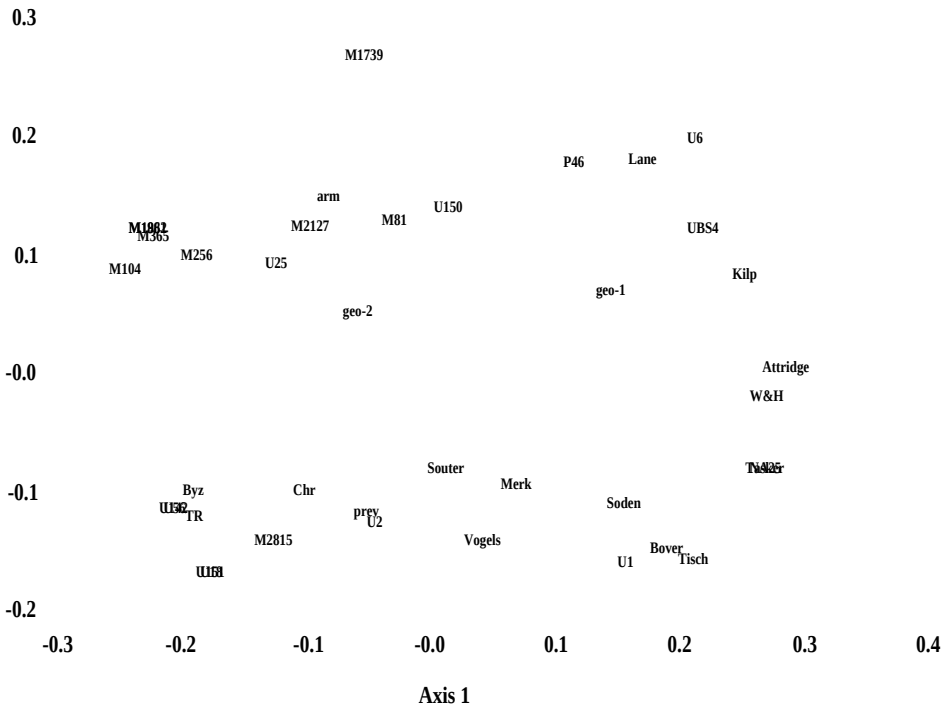
Manifold

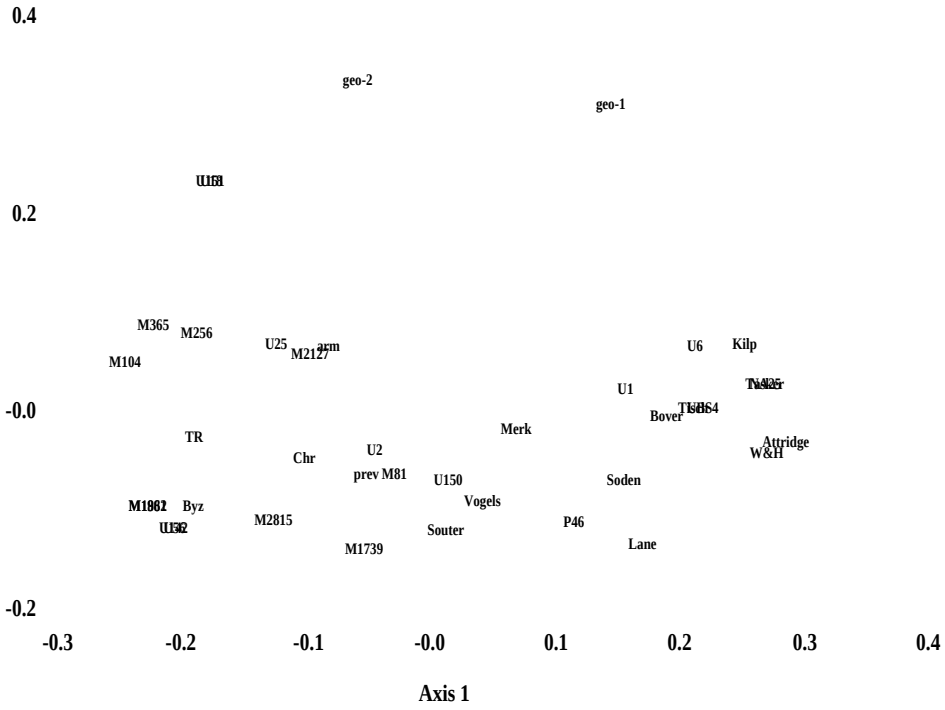




11 units; 39%, 18%, 13%

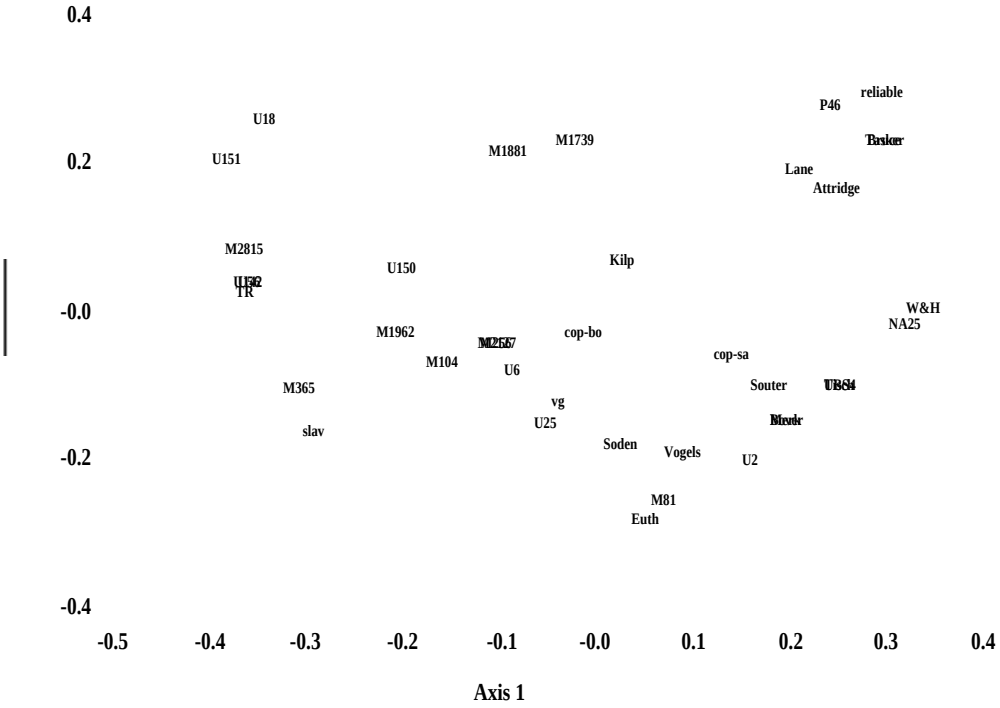
Prevalent

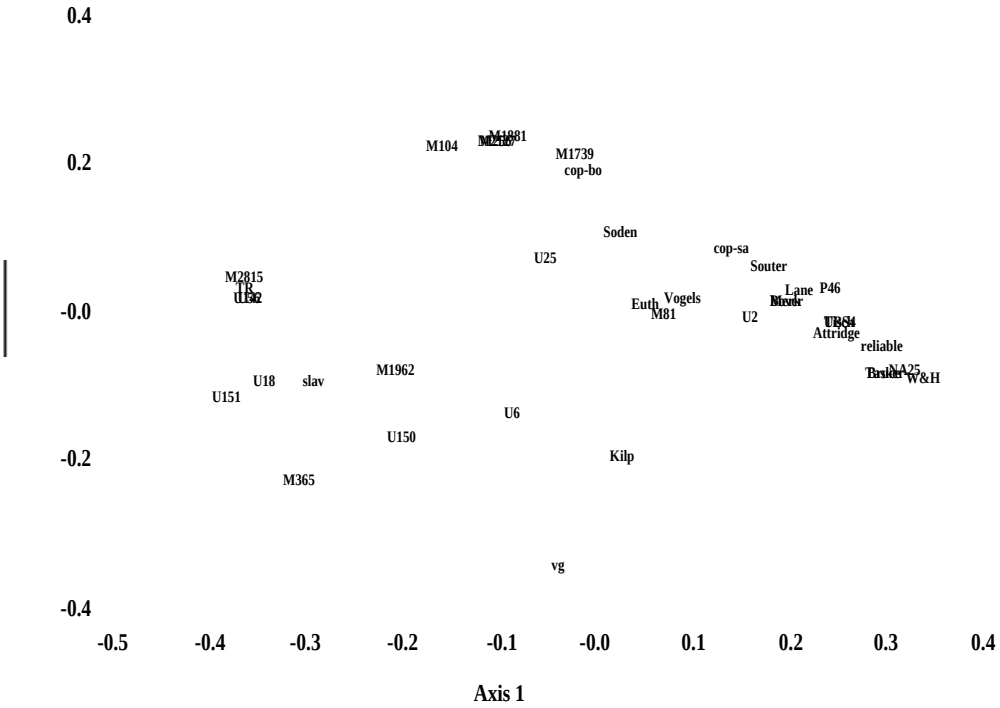




22 units; 24%, 13%, 10%

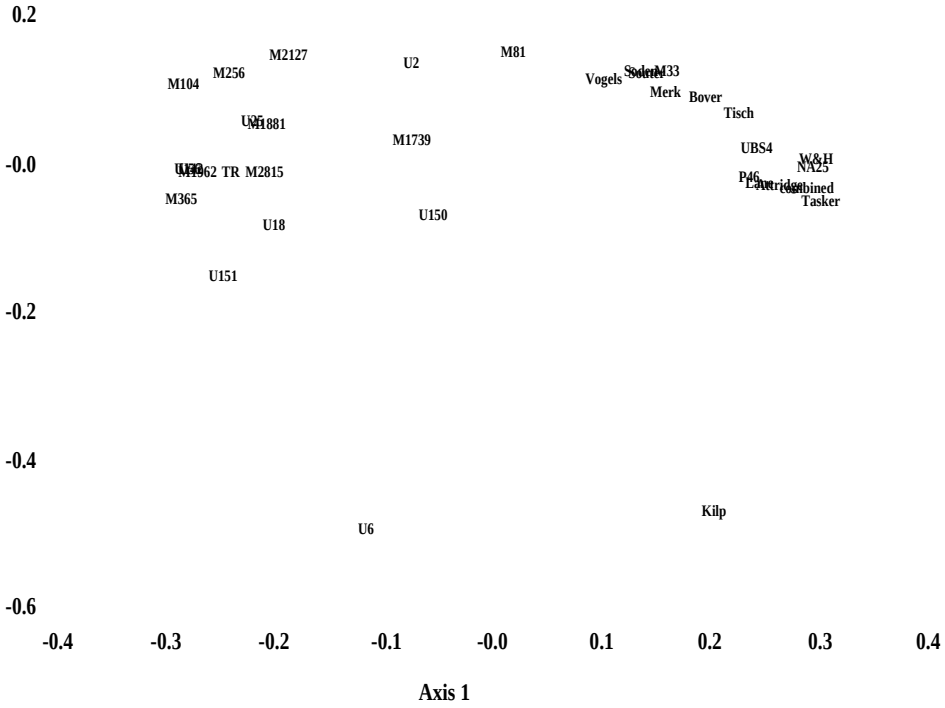
Reliable

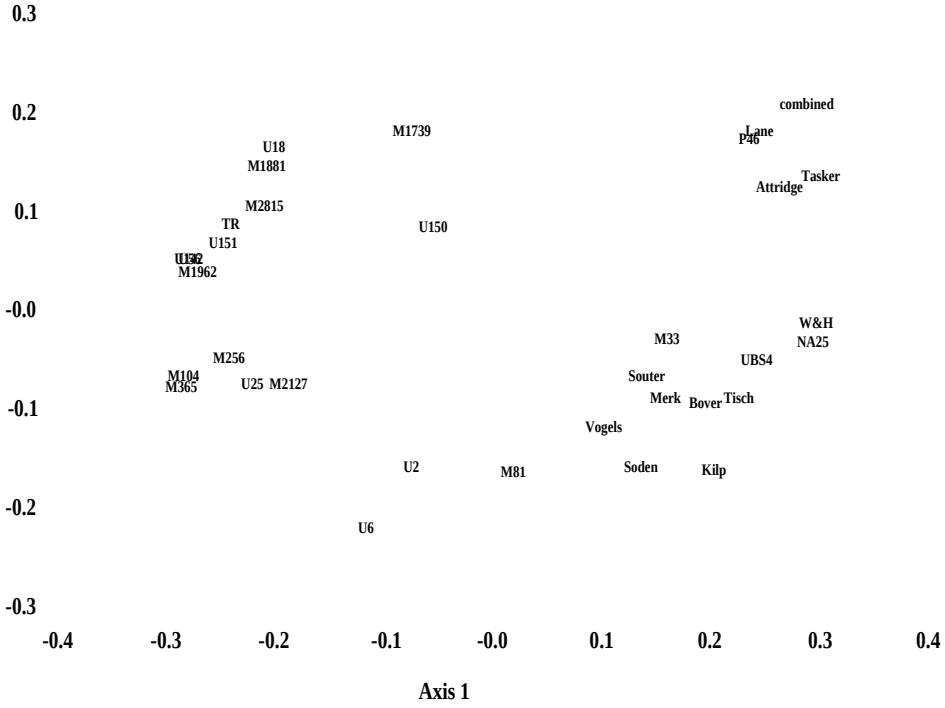




26 units; 32%, 15%, 10%

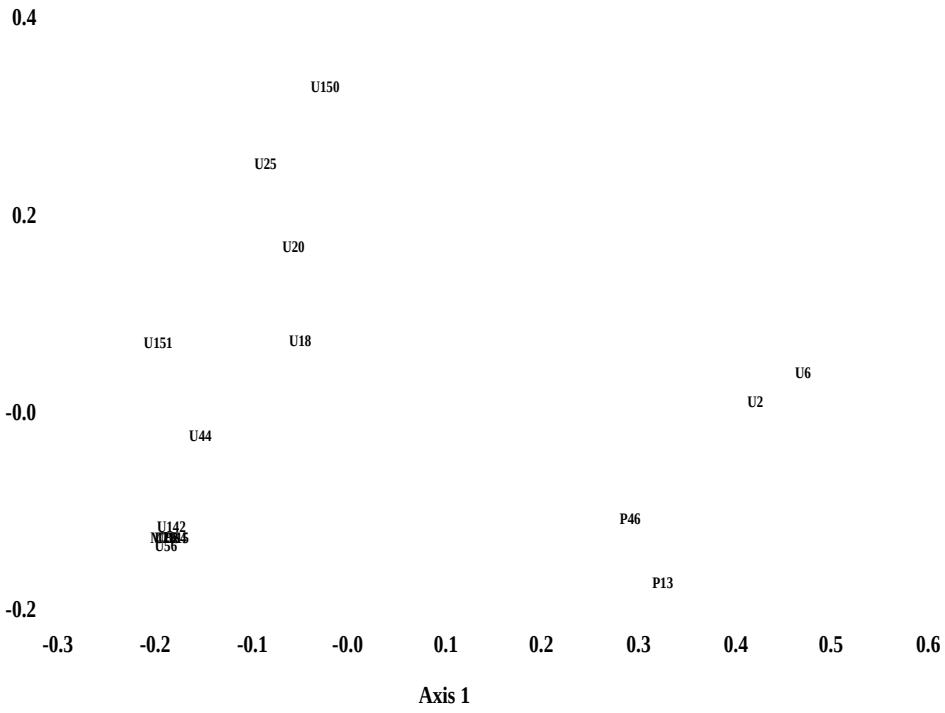
Combined

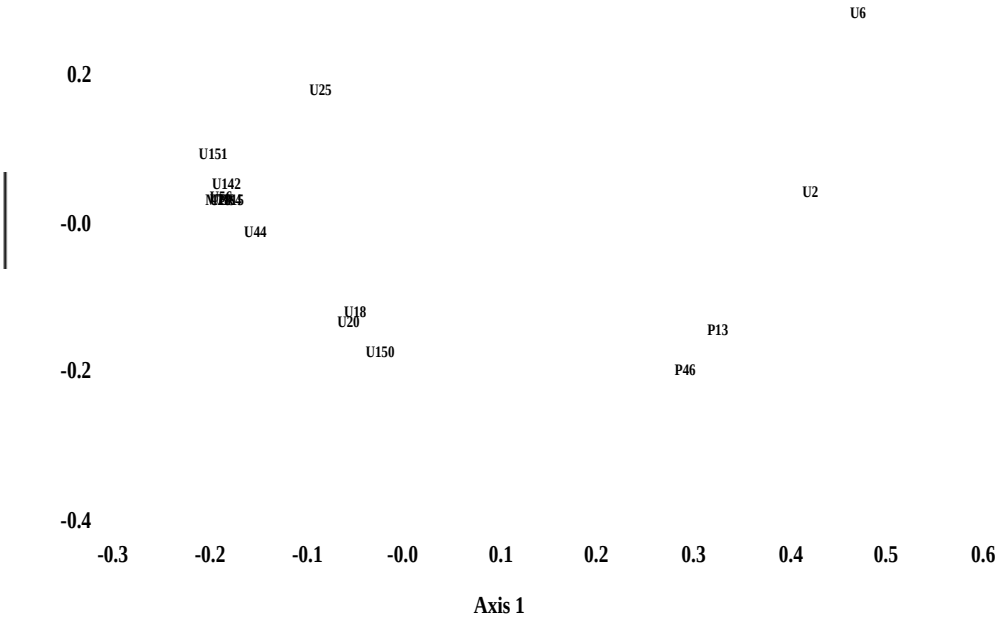




58 units; 32%, 13%, 9%

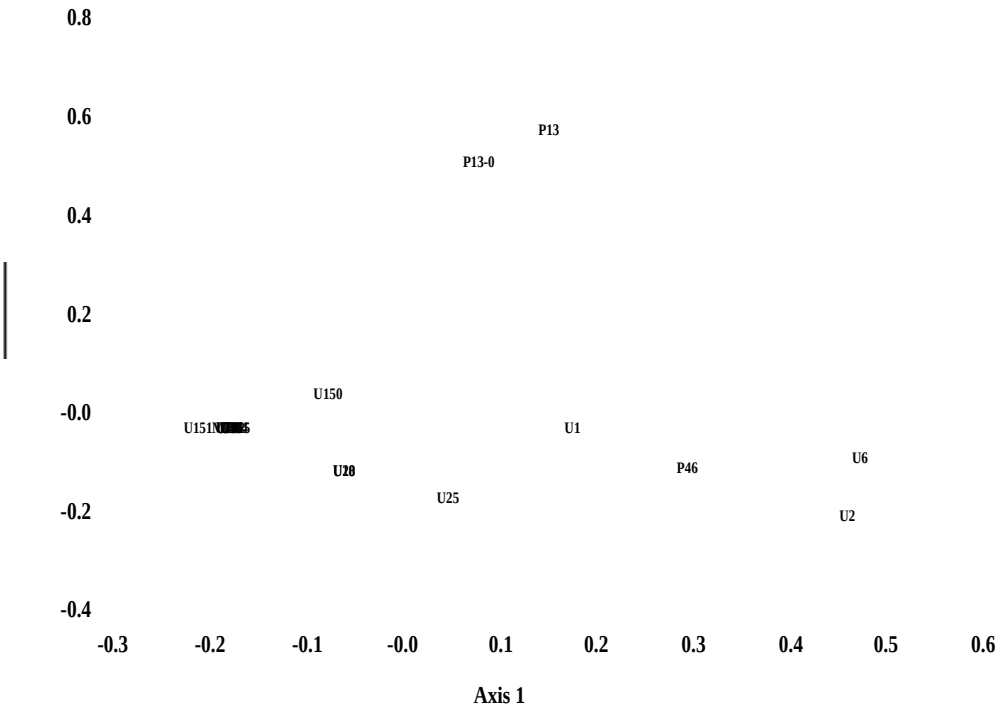
P13 (Oxyrhynchus, 300?)

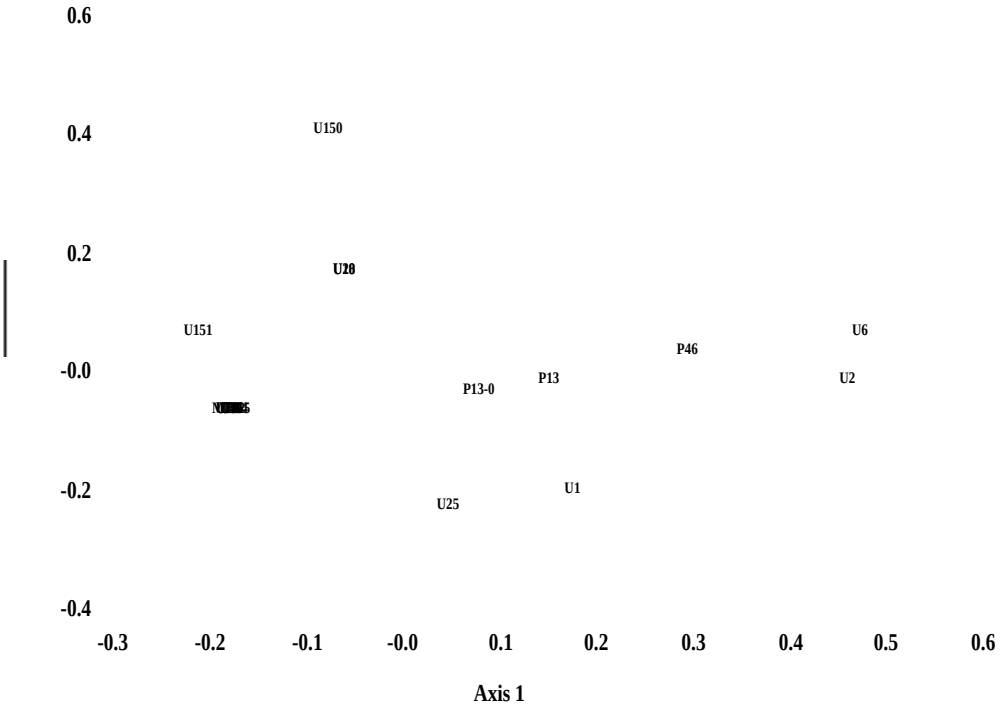




95 units; 37%, 15%, 11%

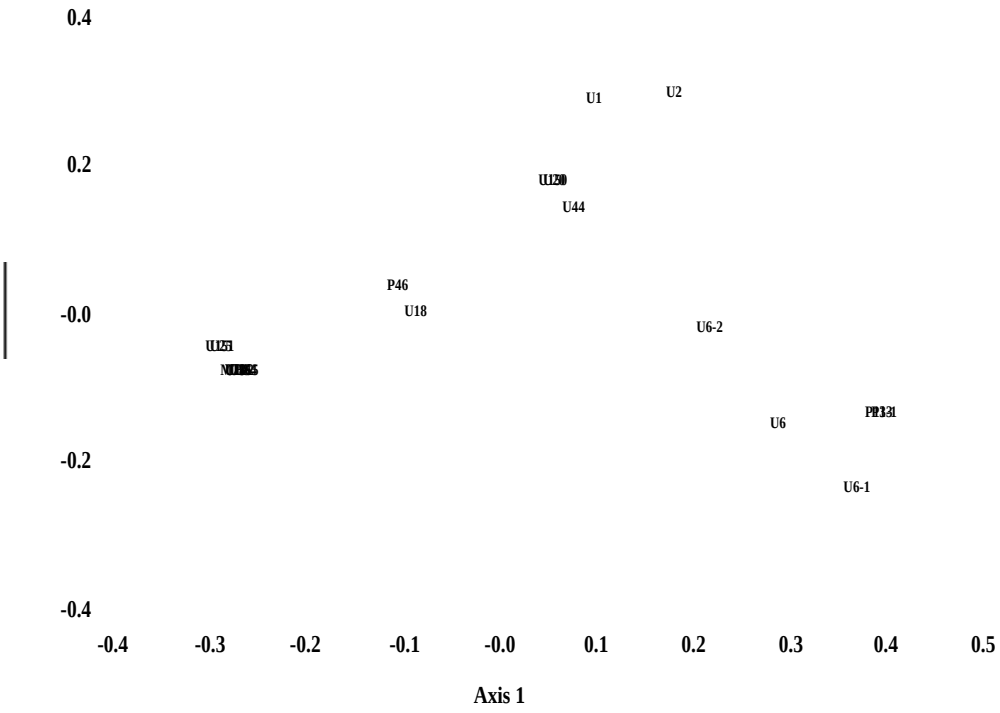
P13-0

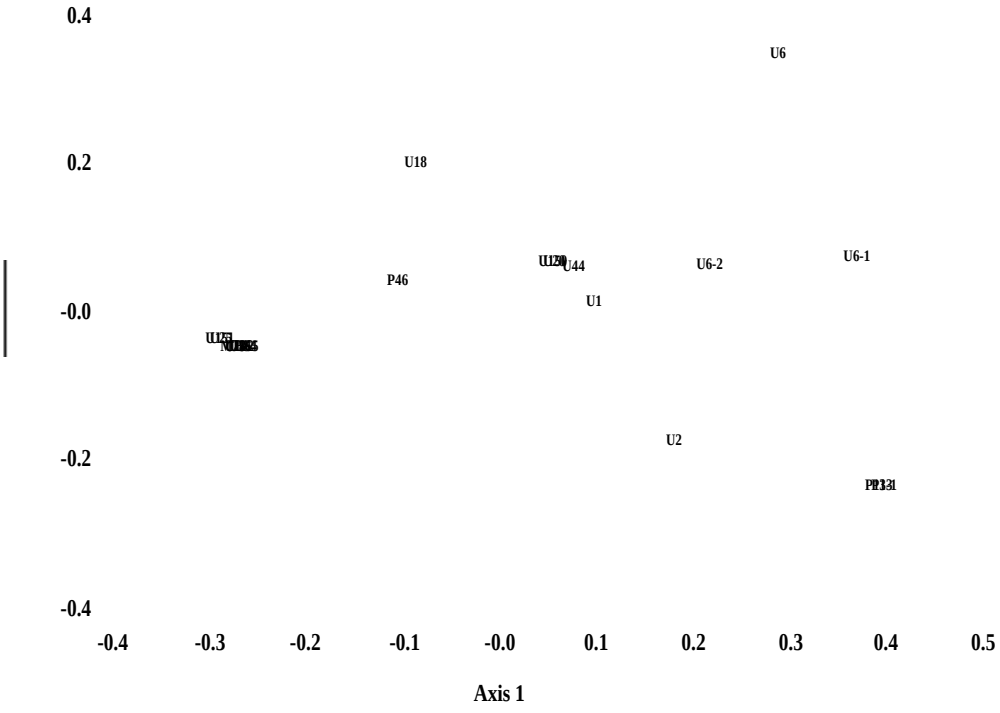




10 units; 35%, 29%, 14%

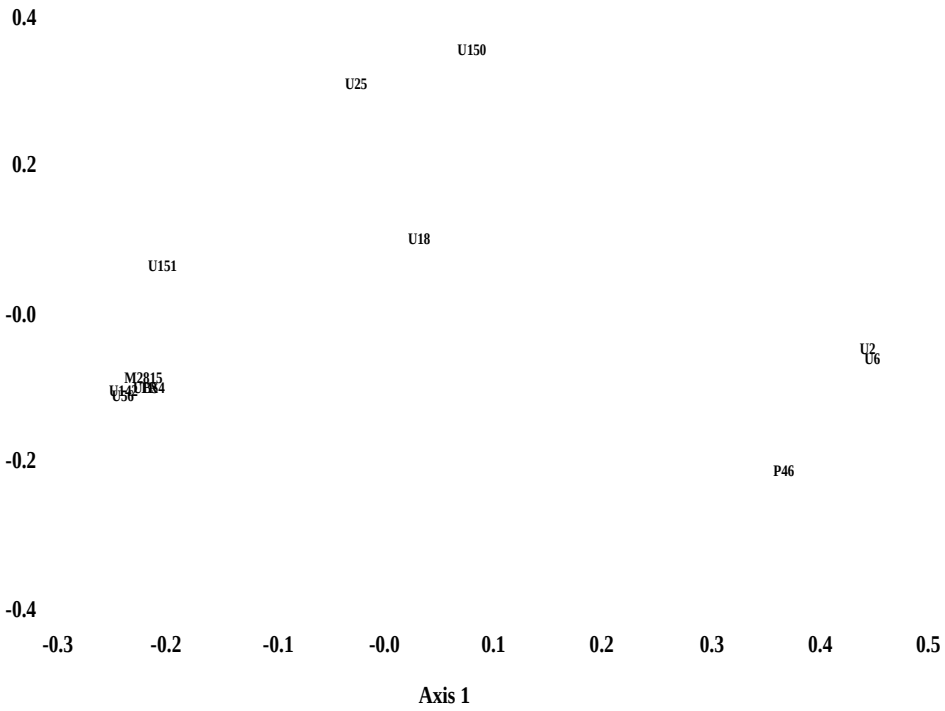
P13-1

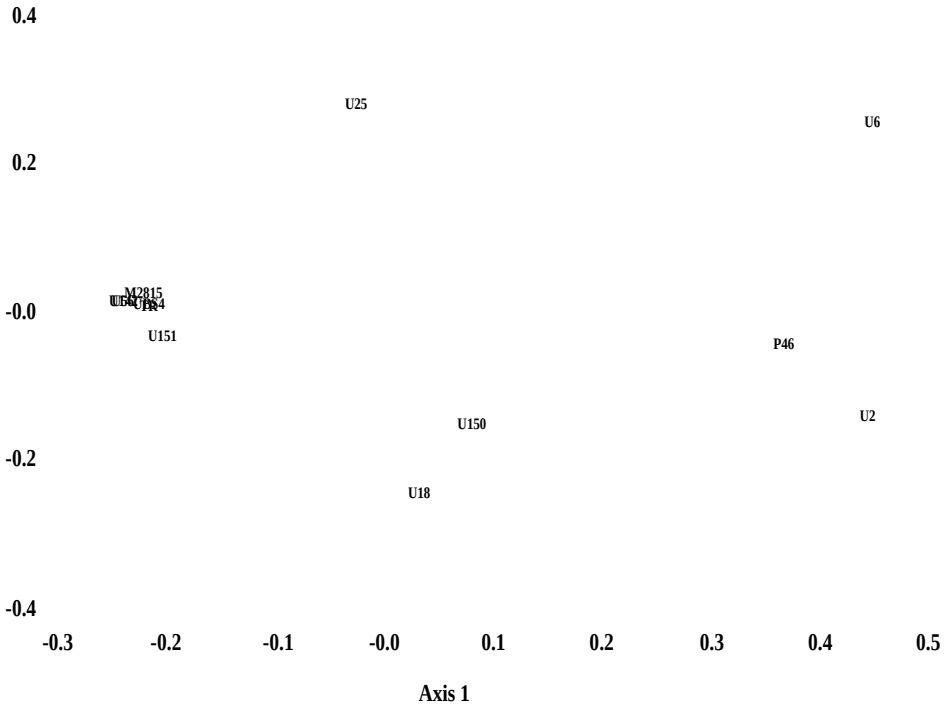




10 units; 44%, 15%, 13%

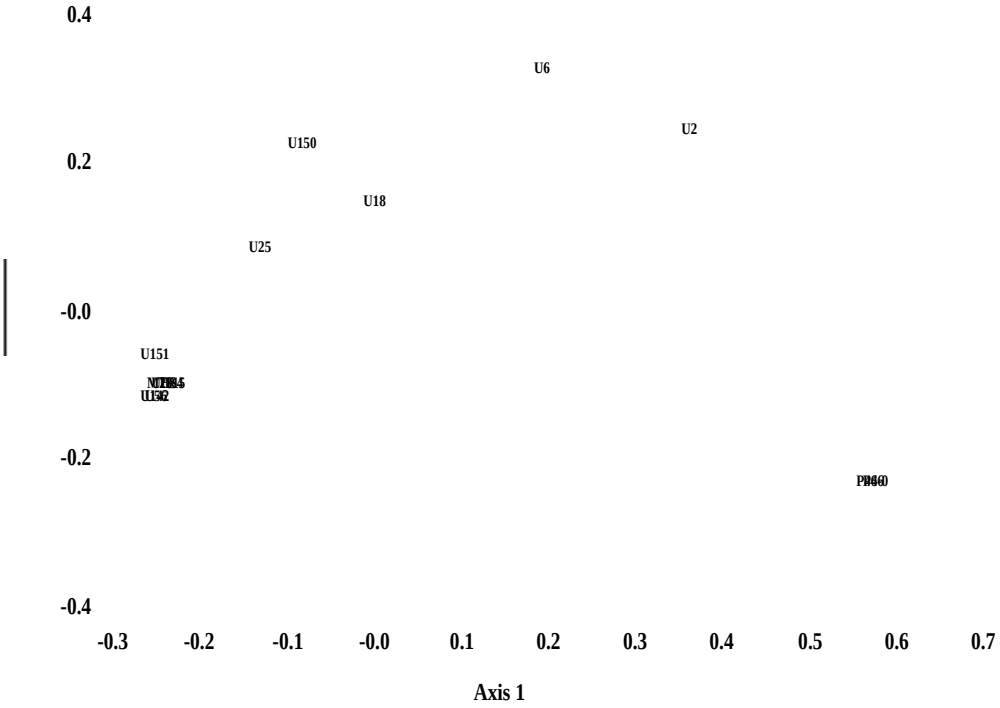
P46 (Fayyum, 200?)

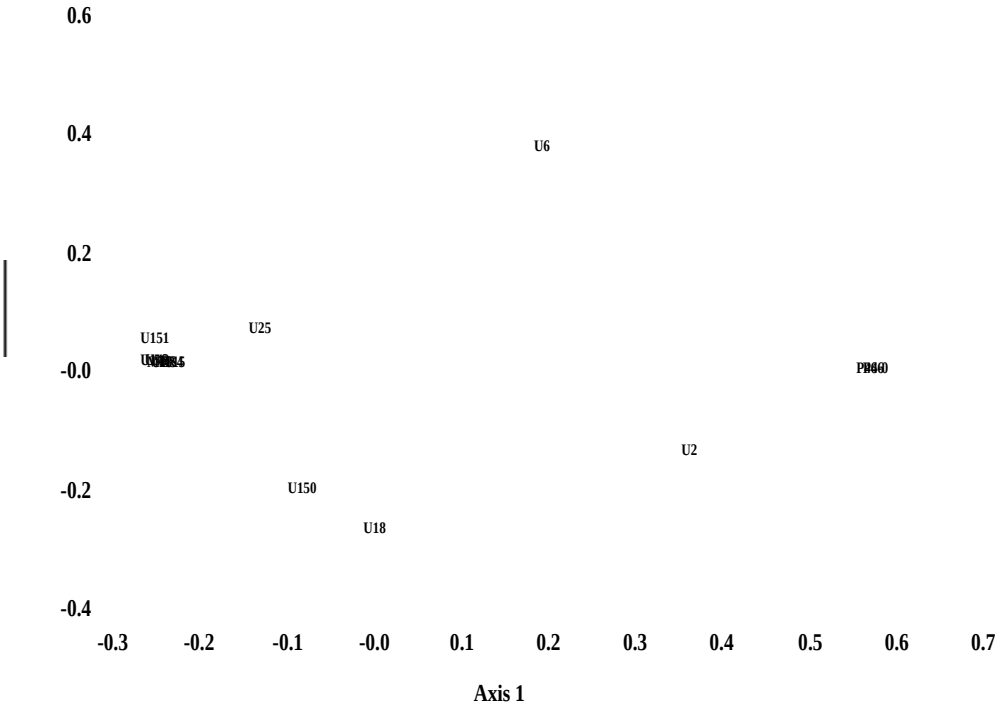




317 units; 40%, 16%, 12%

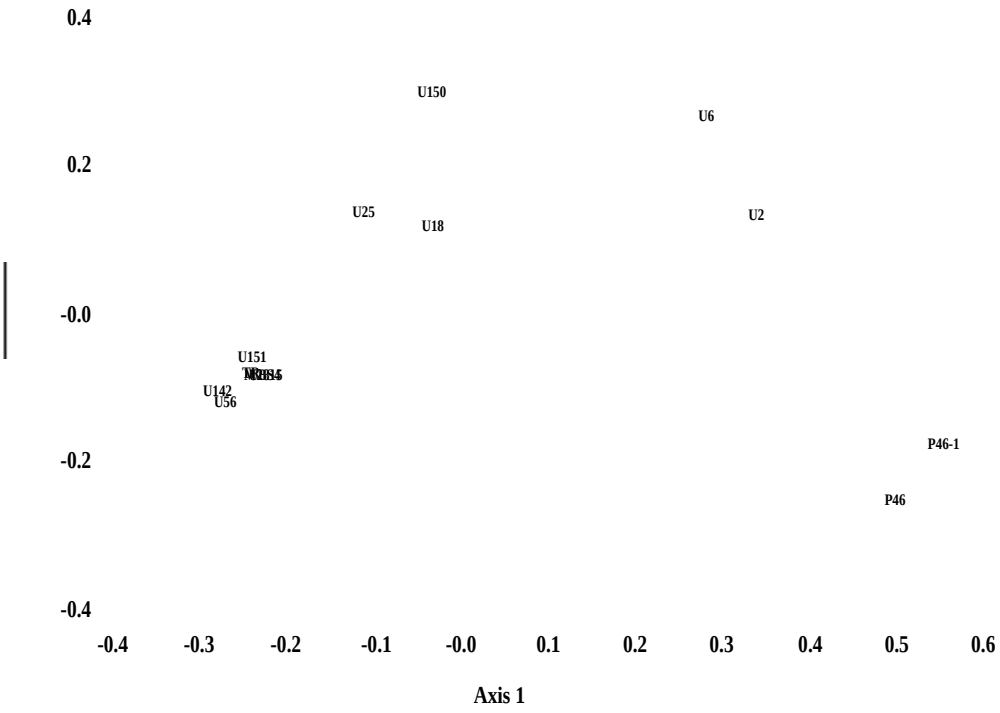
P46-0

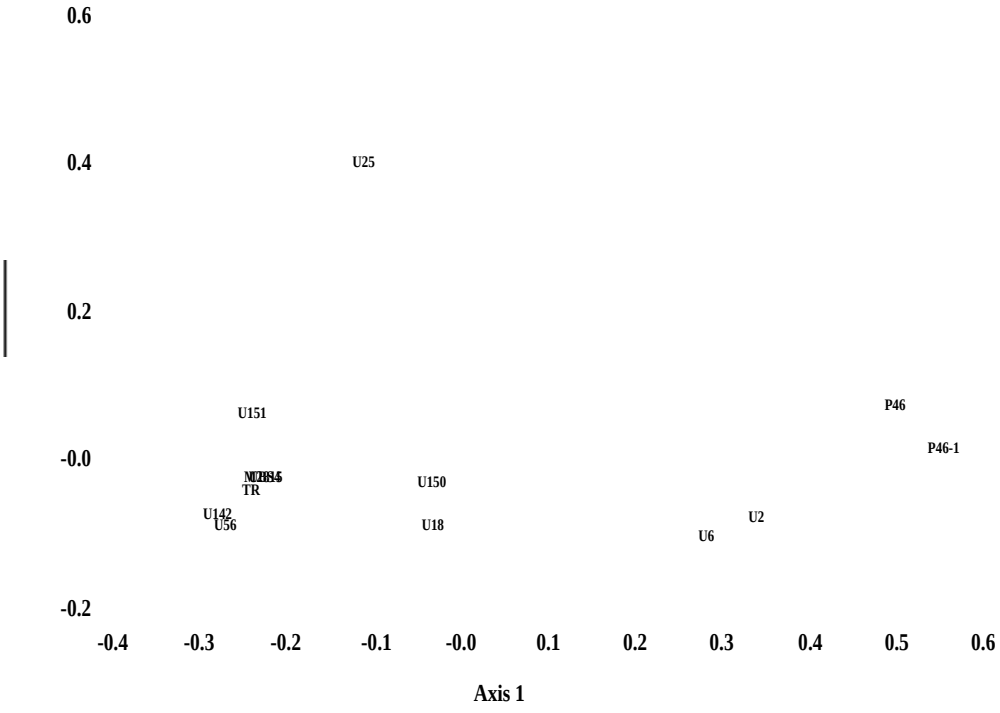




33 units; 53%, 18%, 12%

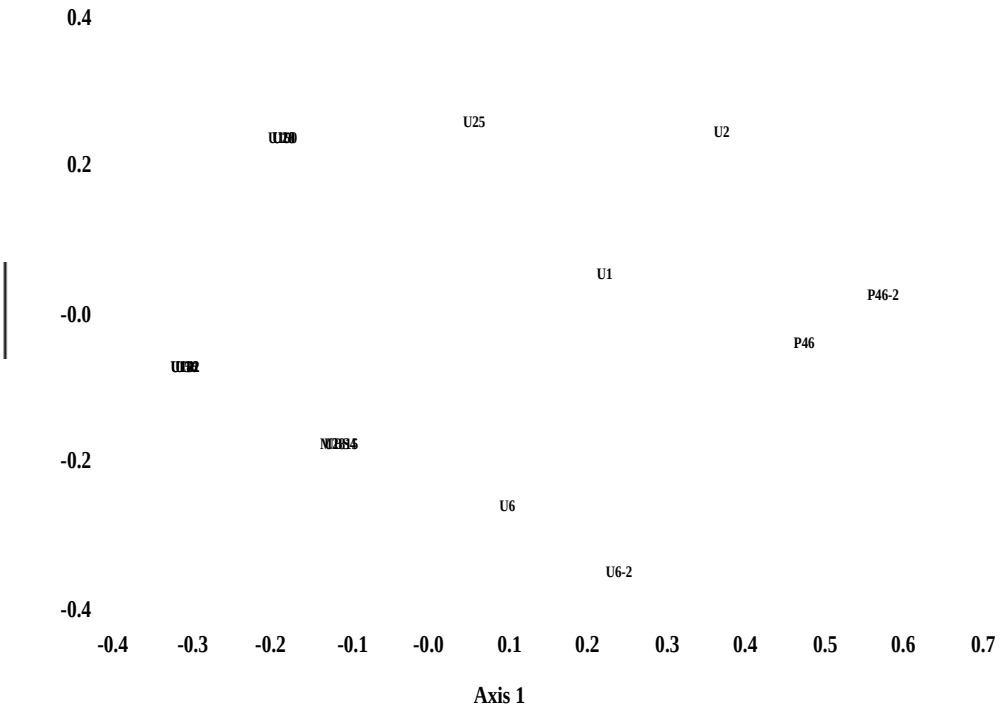
P46-1

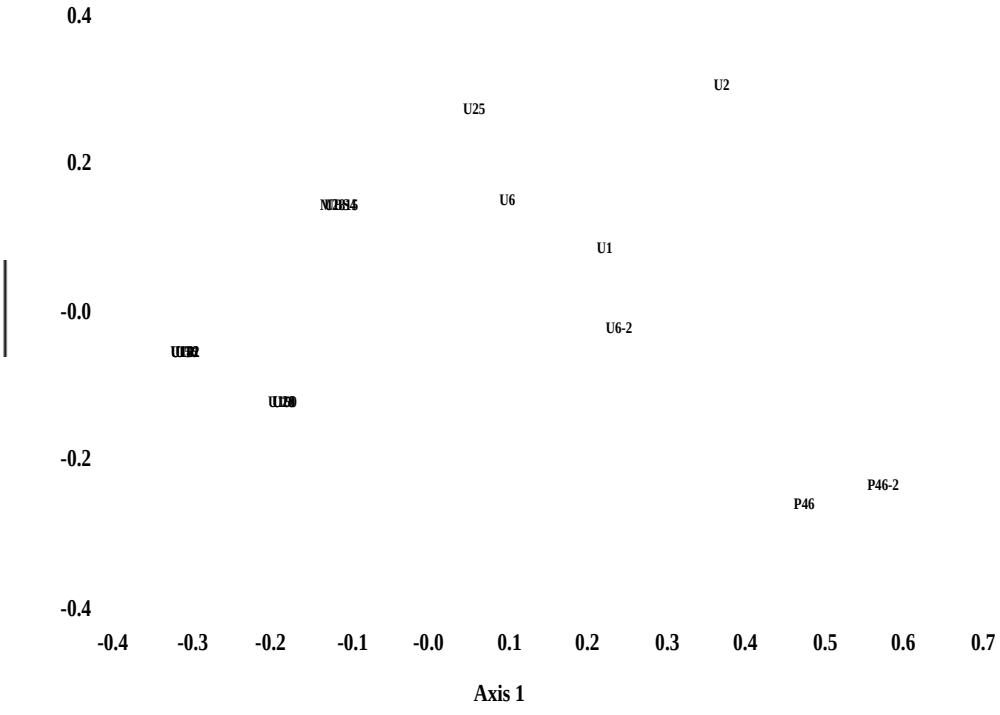




66 units; 50%, 16%, 9%

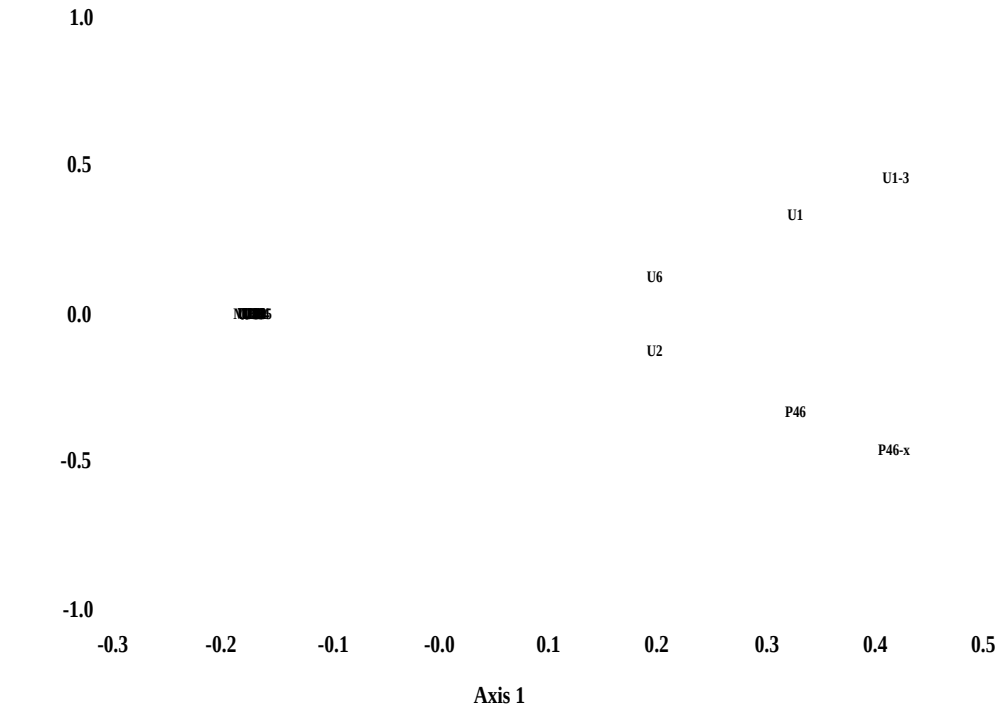
P46-2

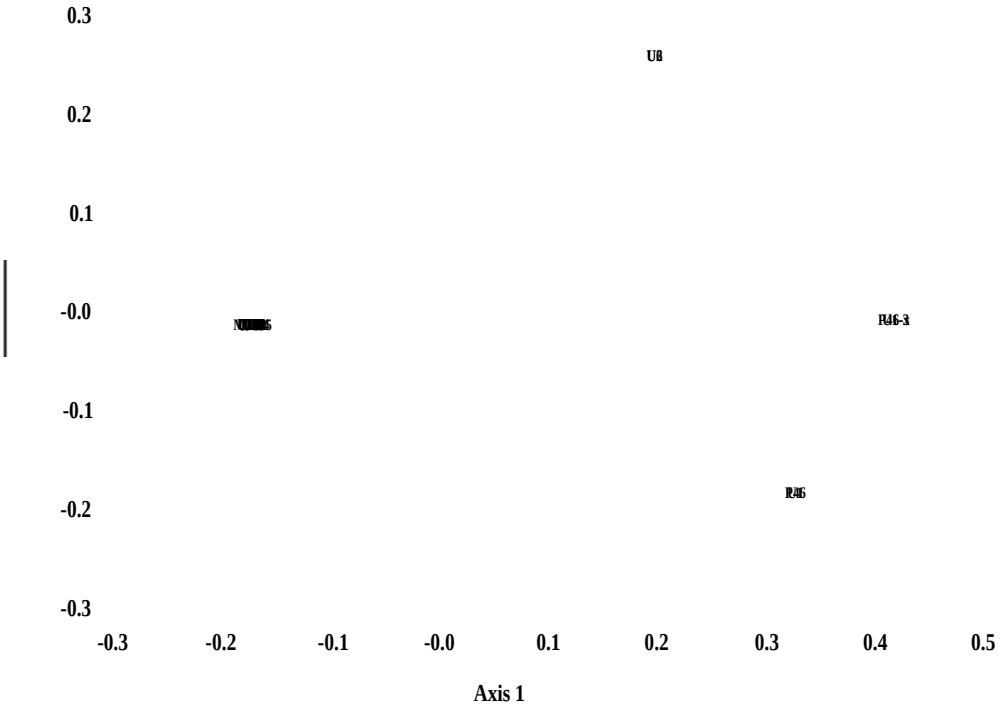




7 units; 48%, 21%, 15%

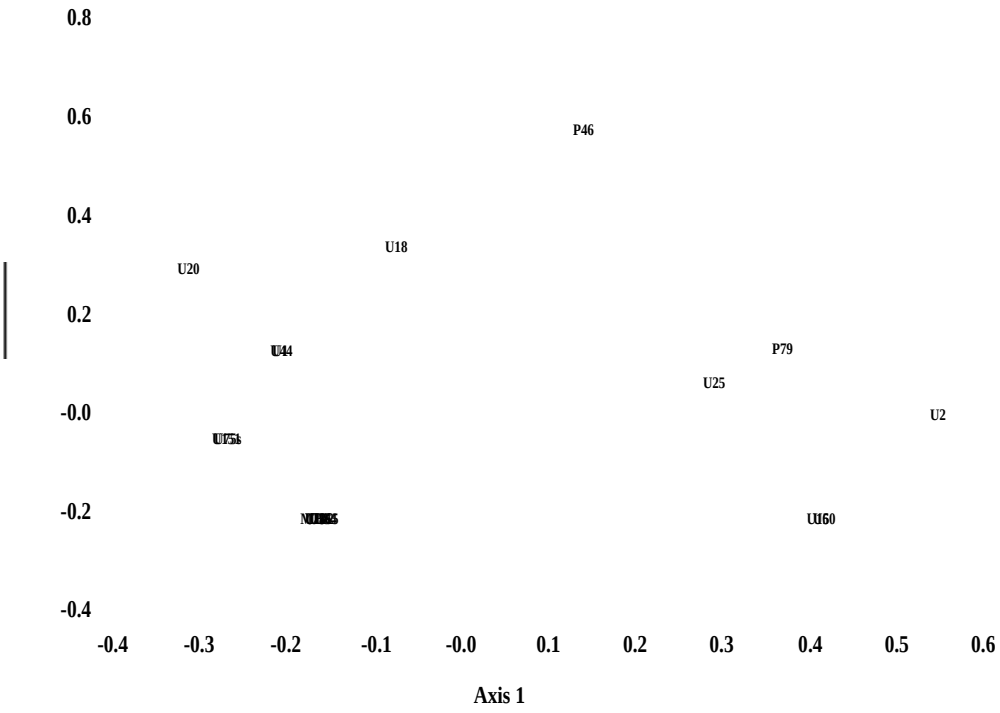
P46-x

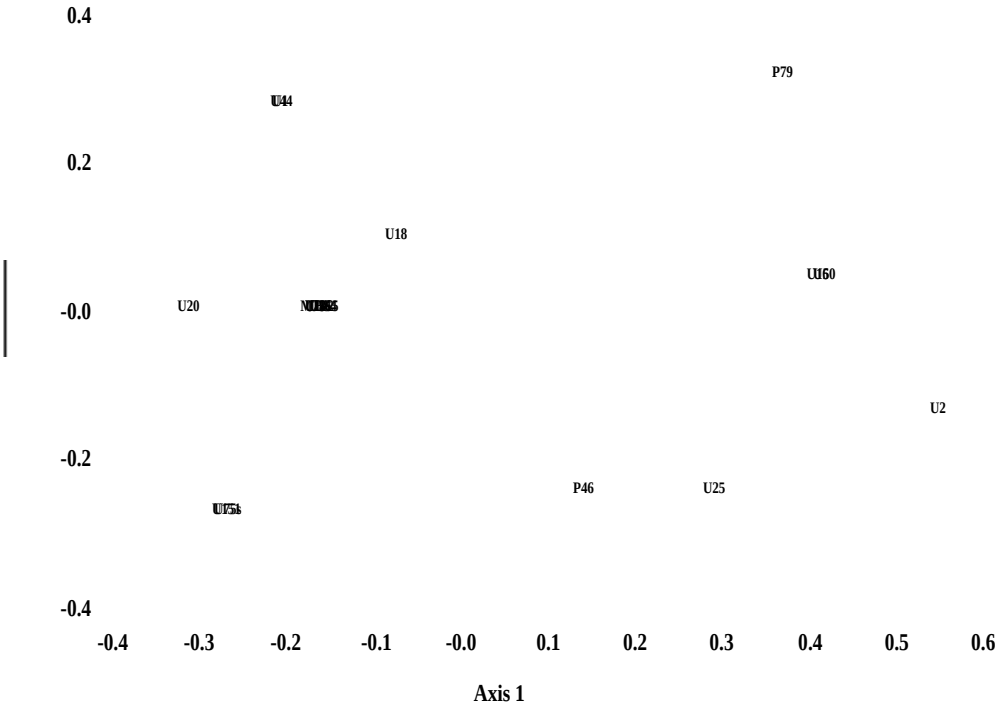




7 units; 47%, 33%, 10%

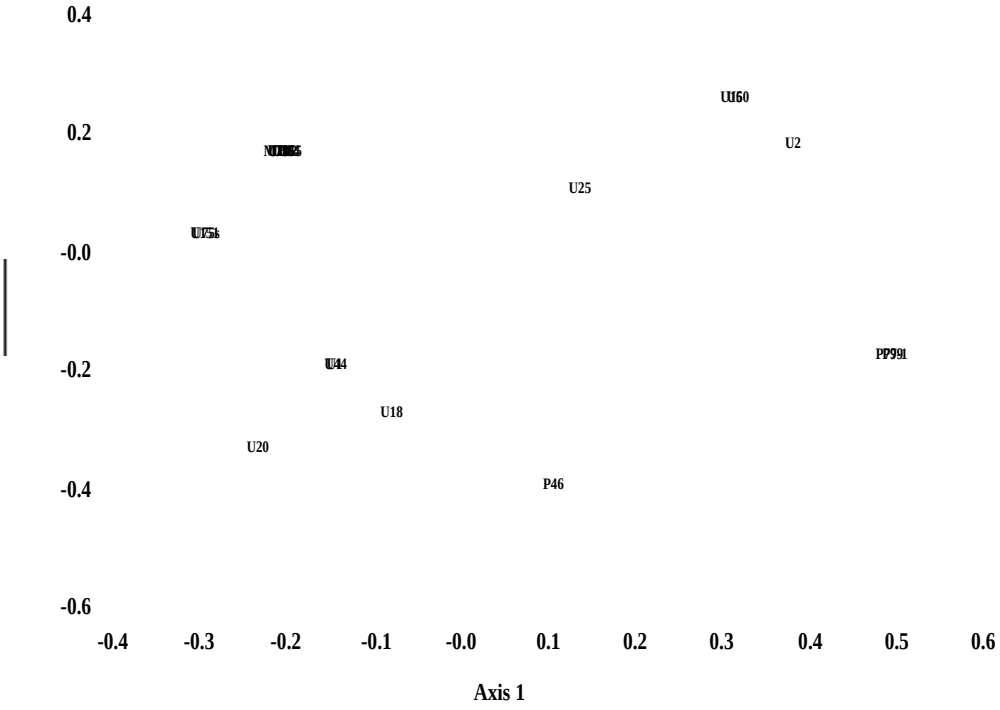
P79 (Fayyum, 650?)

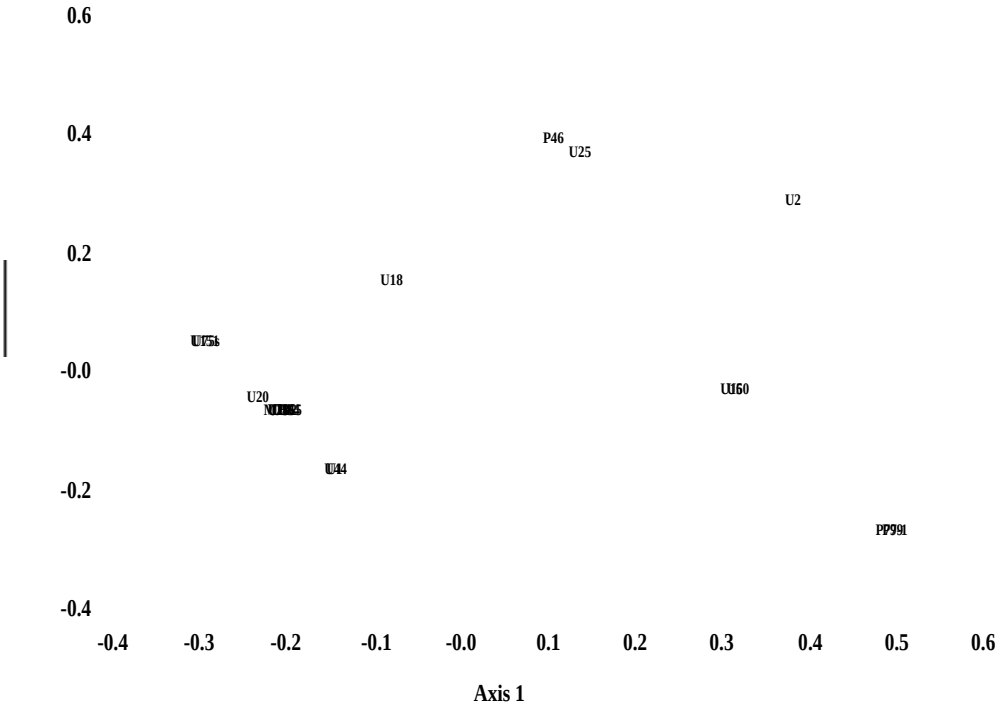




5 units; 40%, 27%, 16%

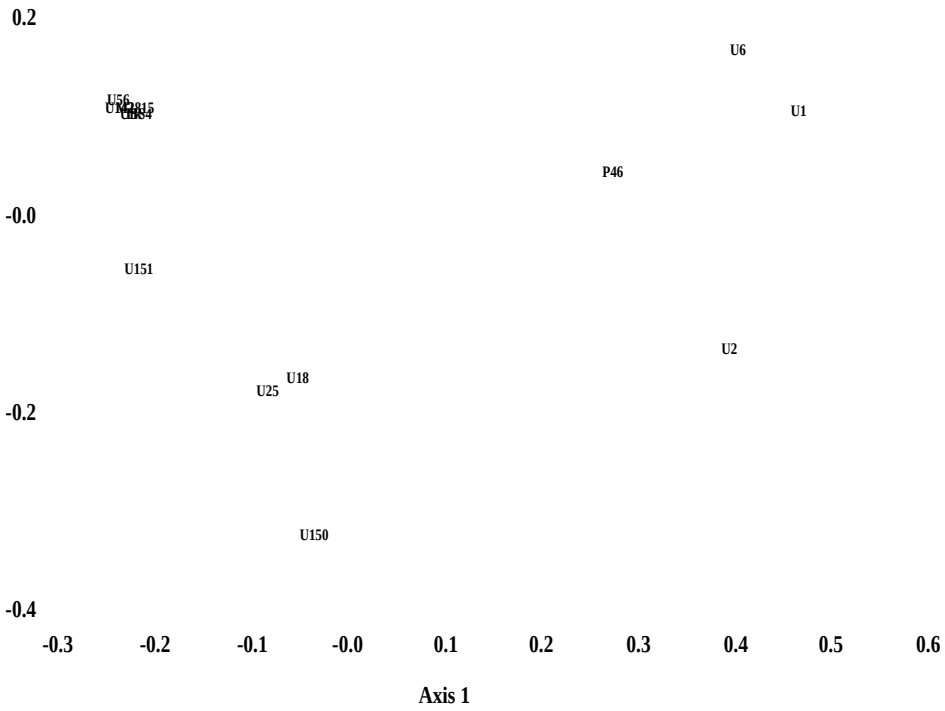
P79-1

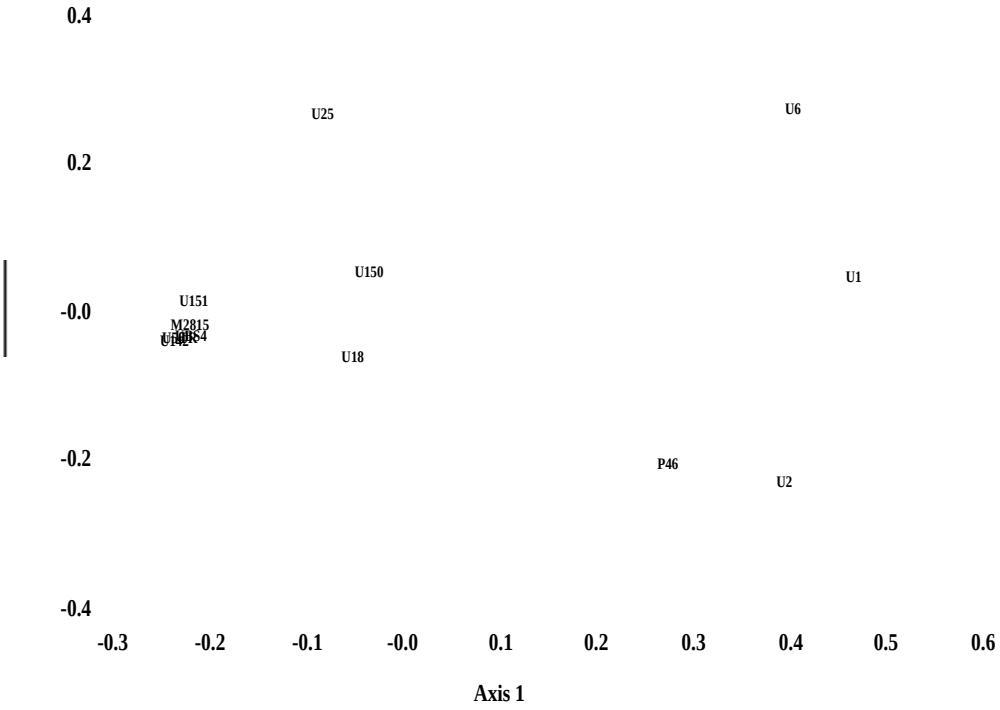




6 units; 40%, 24%, 19%

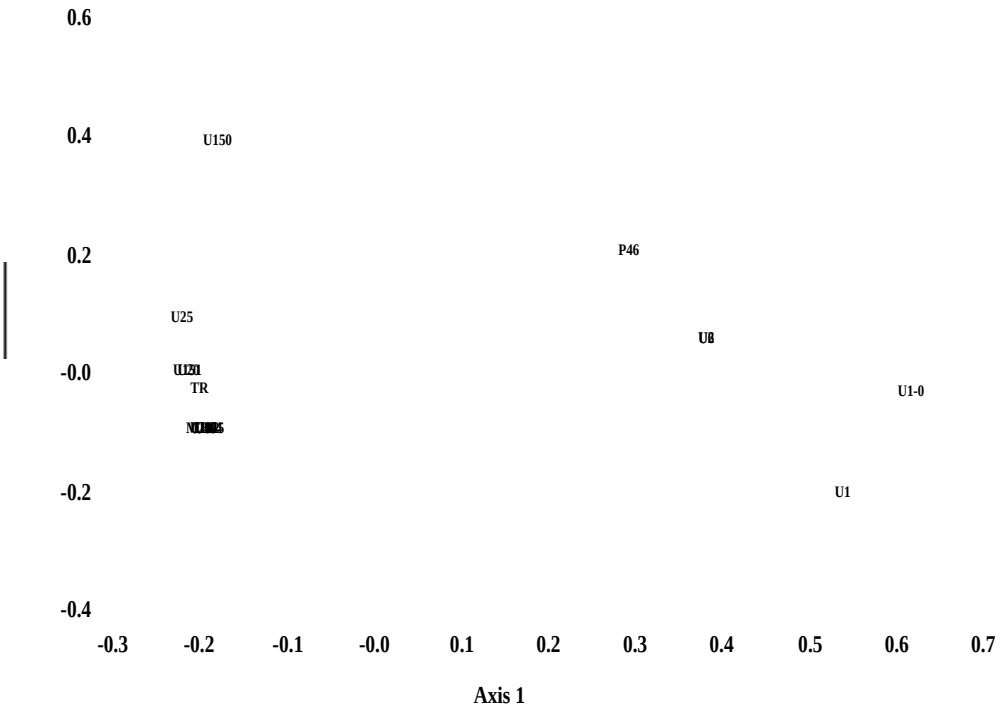
U1 (Caesarea? 350?)

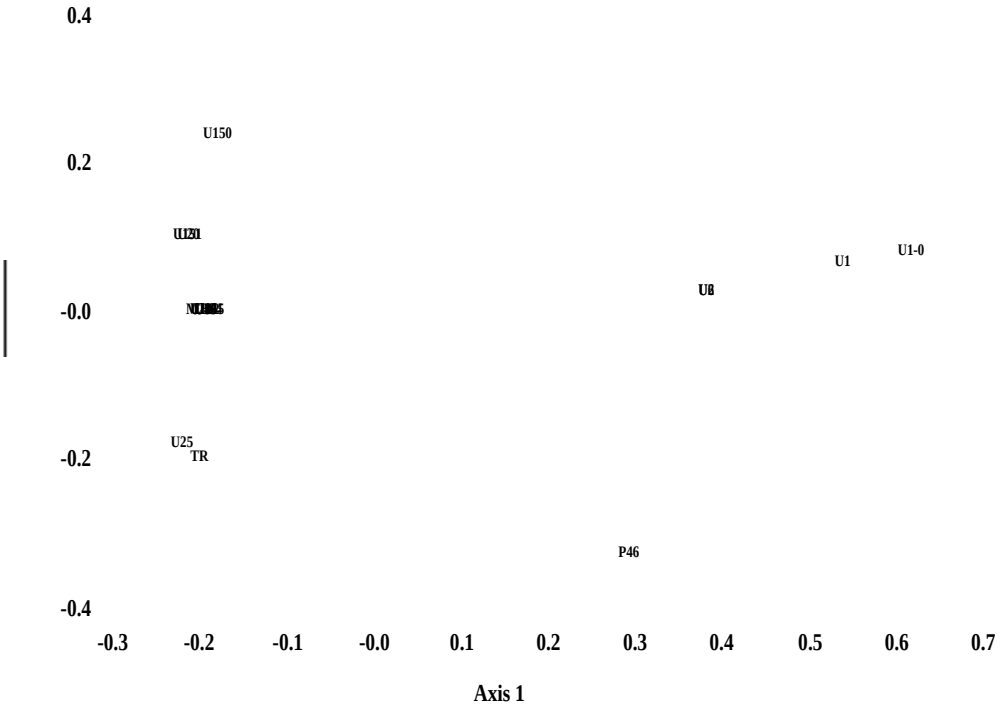




278 units; 42%, 13%, 11%

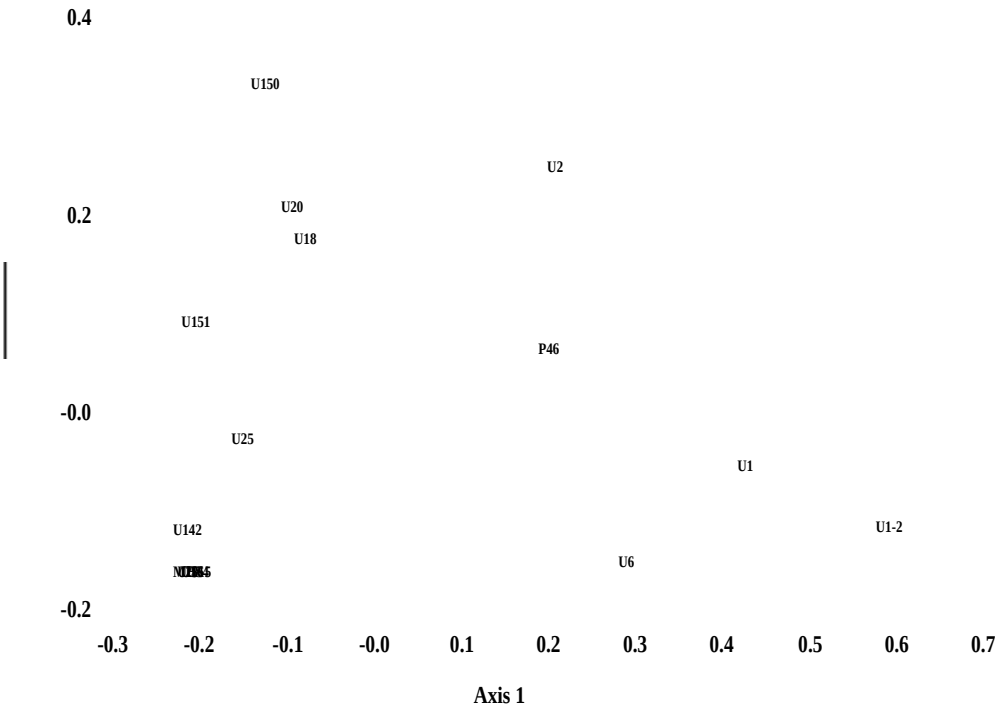
U1-0

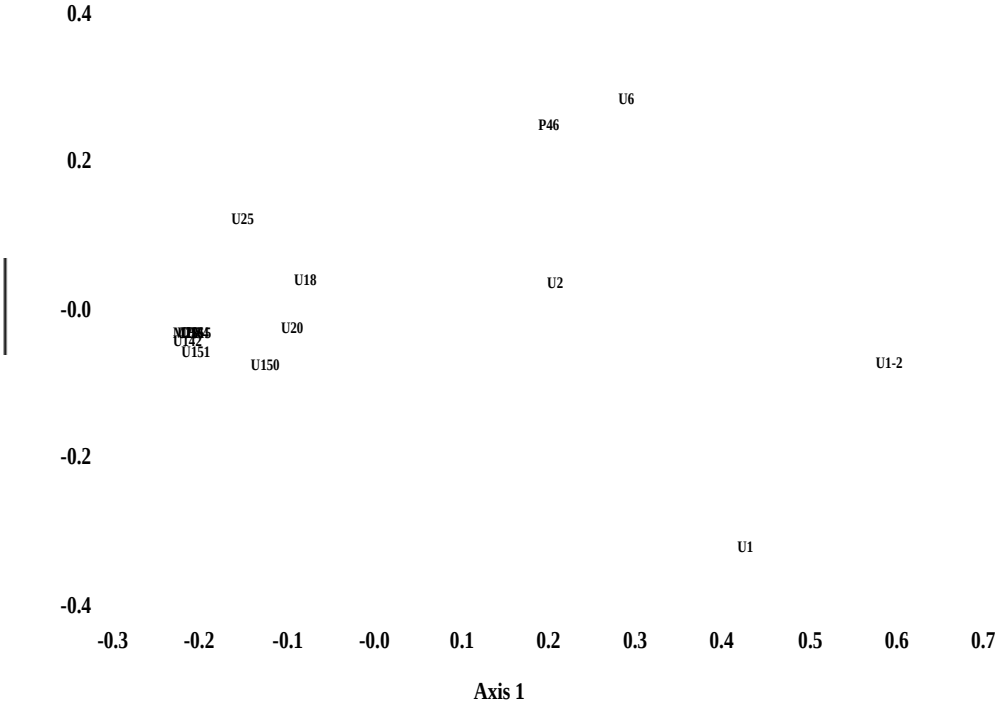




13 units; 58%, 12%, 10%

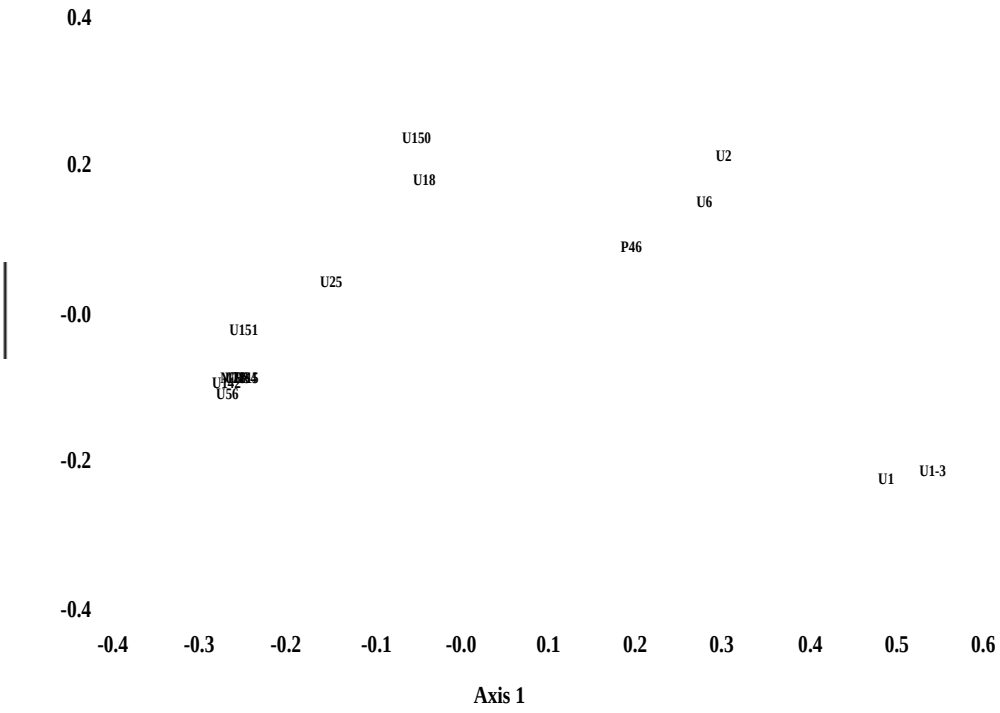
U1-2

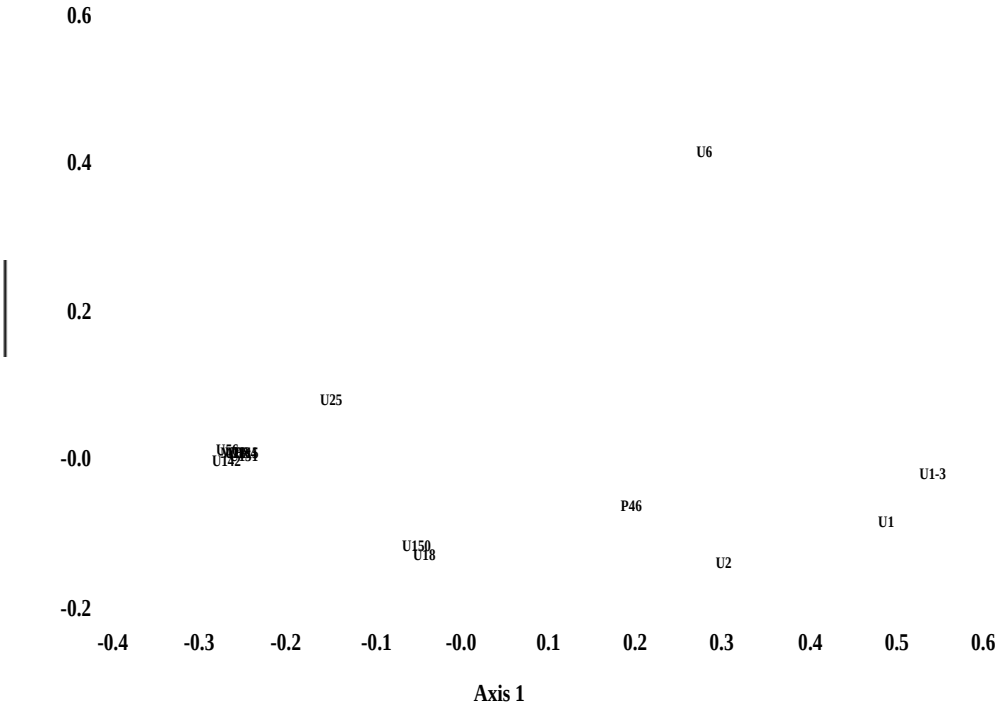




25 units; 41%, 17%, 11%

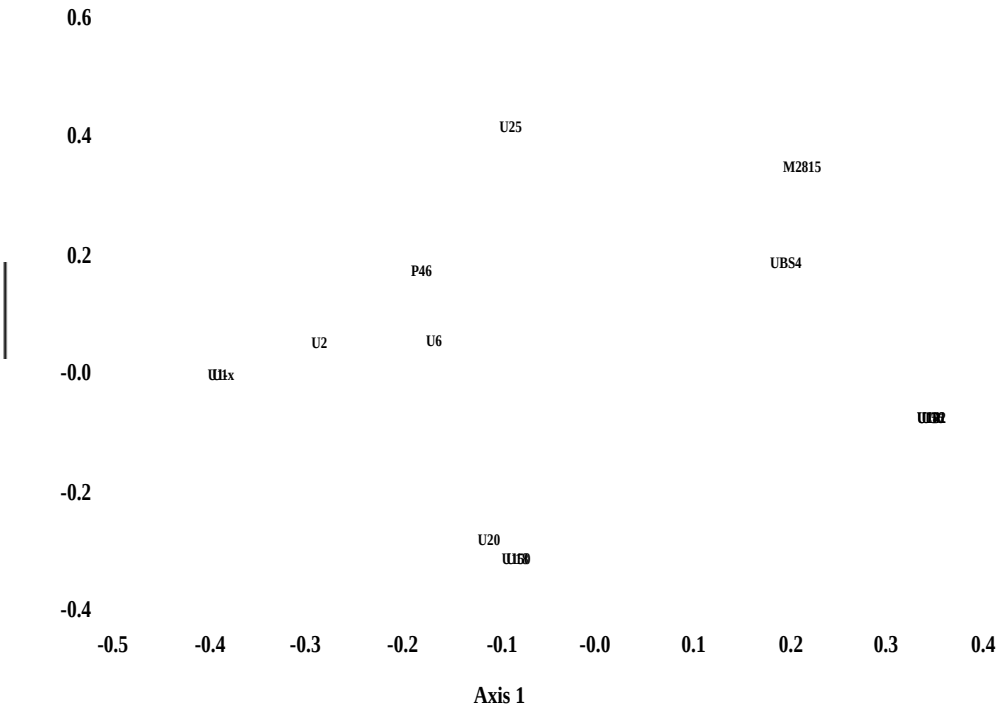
U1-3

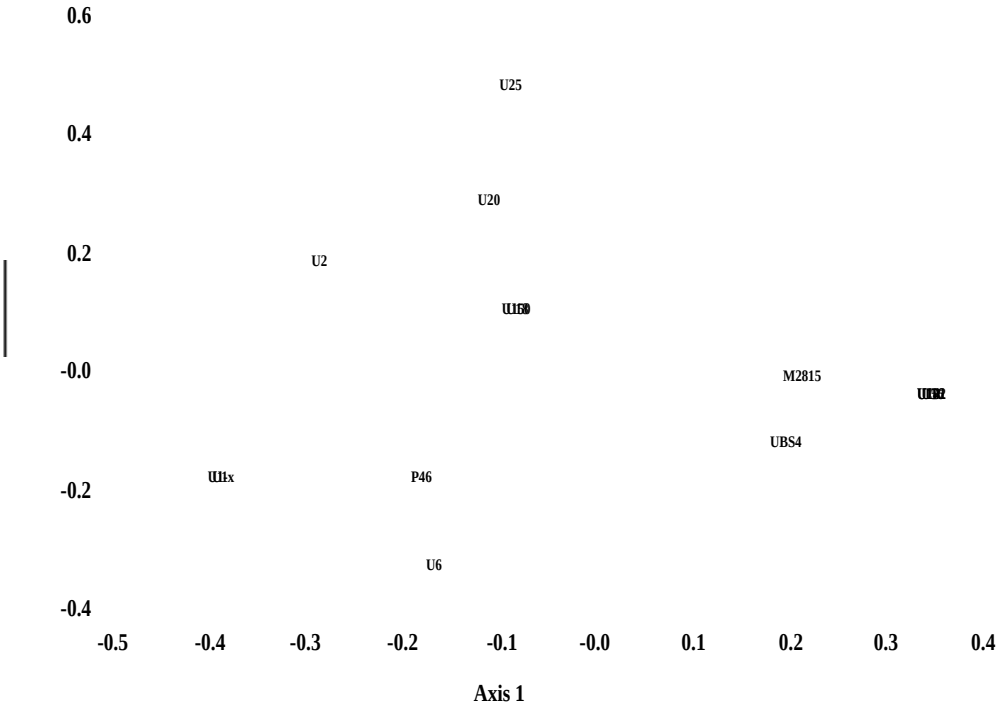




125 units; 48%, 13%, 10%

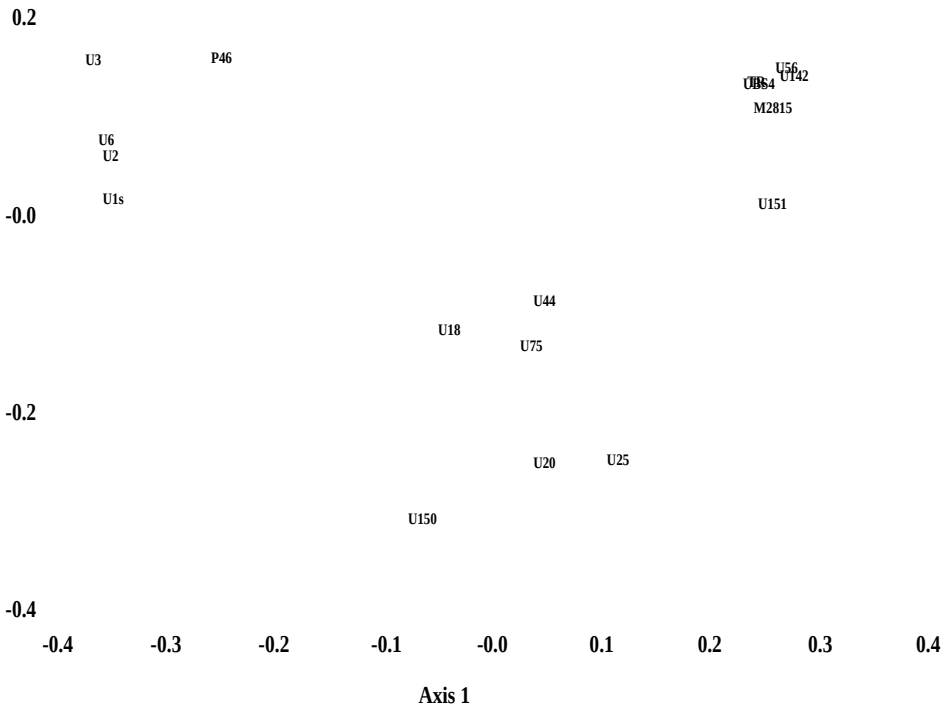
U1-x

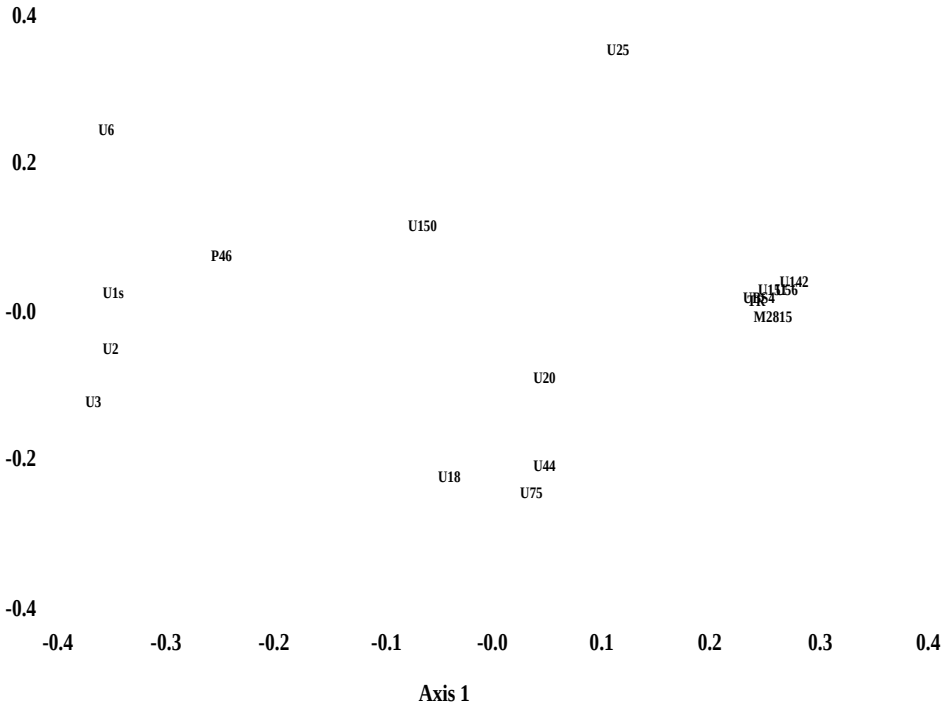




7 units; 38%, 24%, 22%

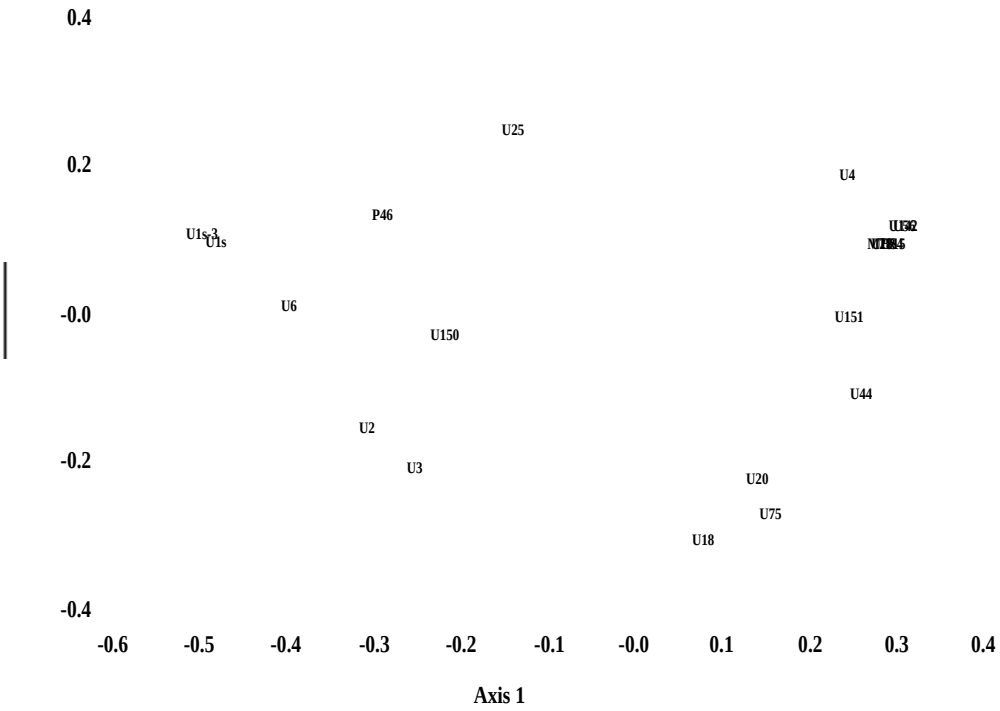
U1s (Caesarea? 350?)

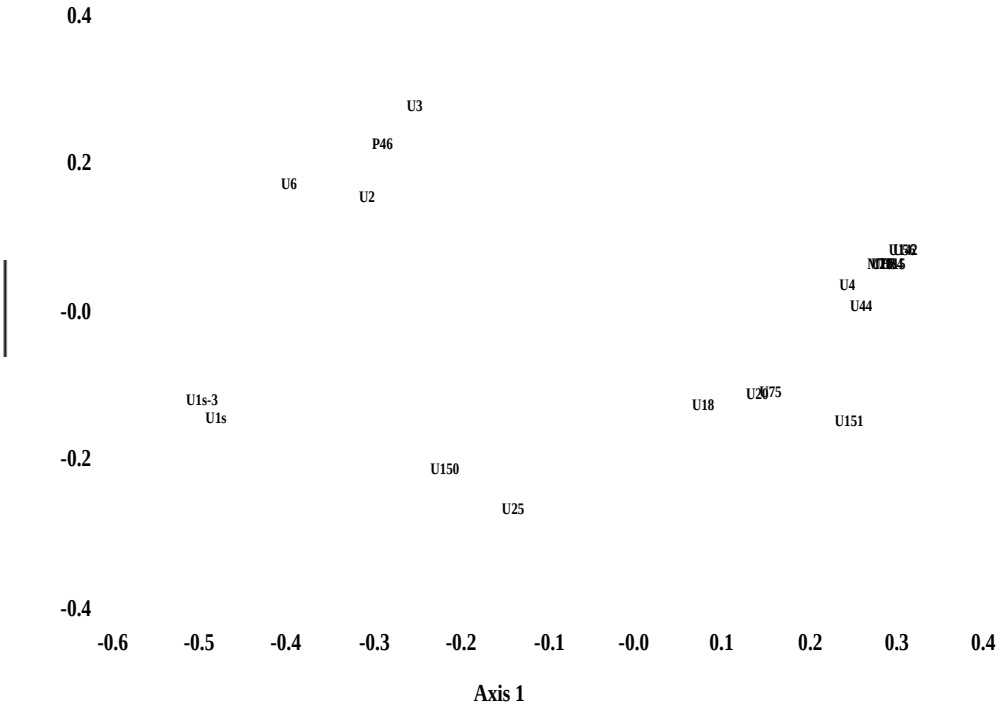




101 units; 35%, 14%, 13%

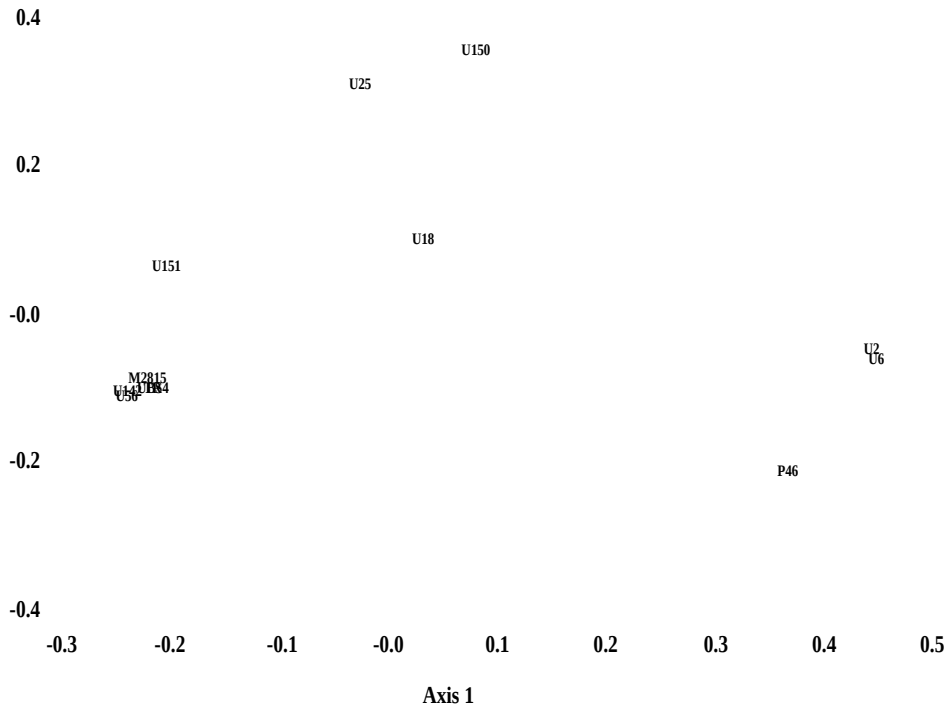
U1s-3

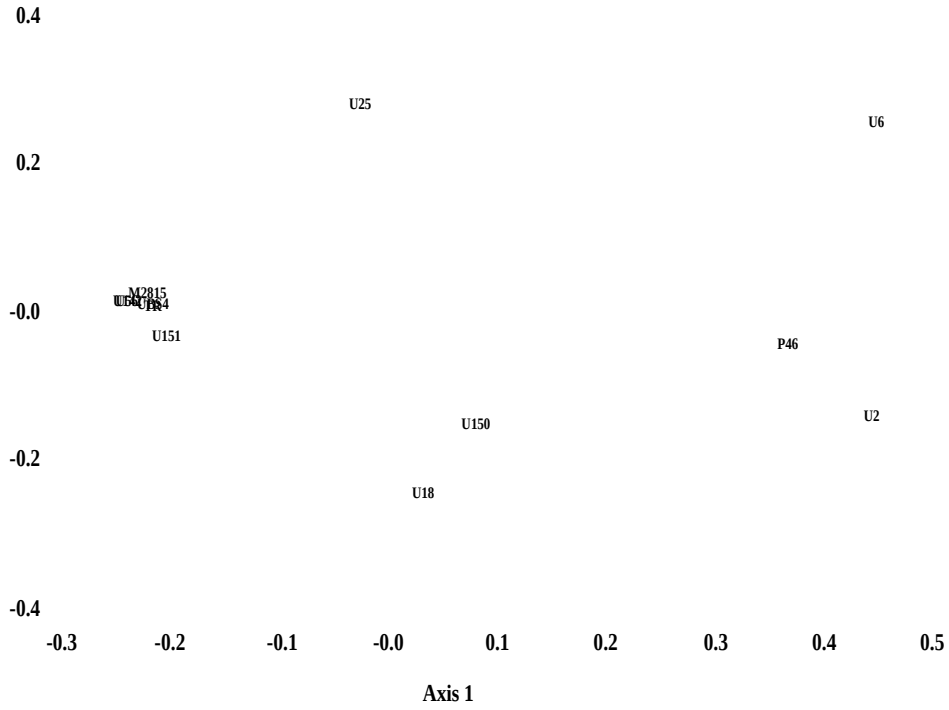




19 units; 46%, 14%, 12%

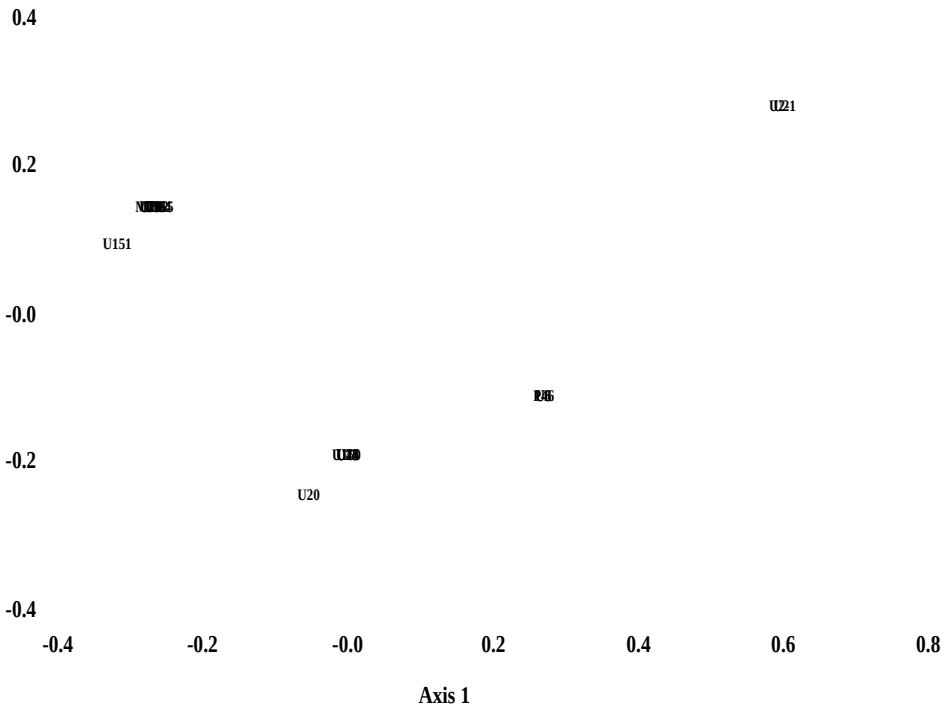
U2 (Alexandria? 450?)

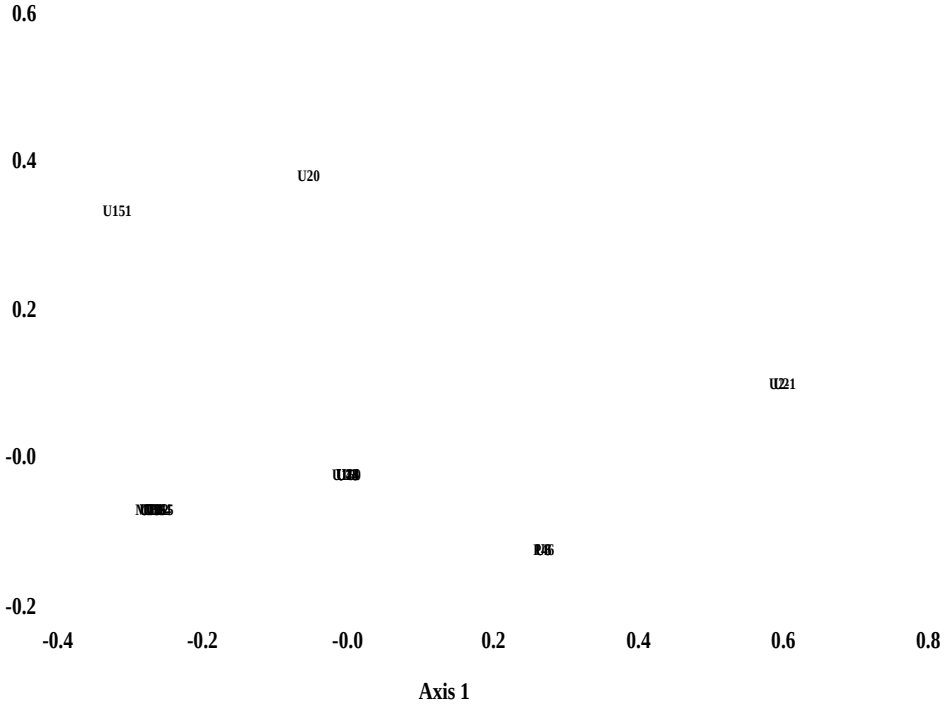




317 units; 40%, 16%, 12%

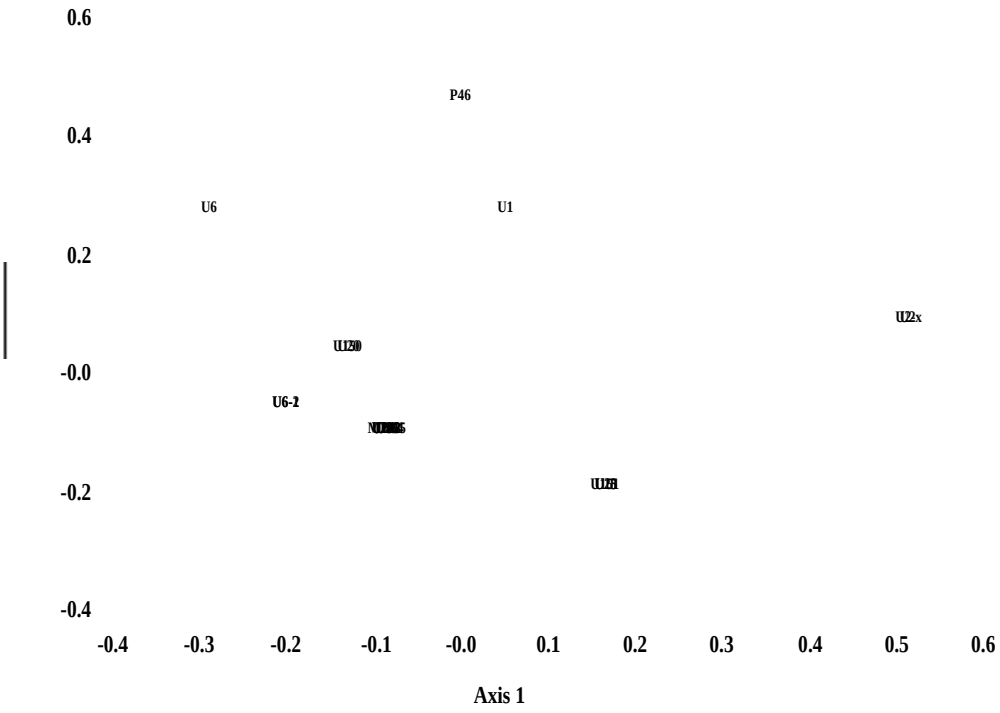
U2-1

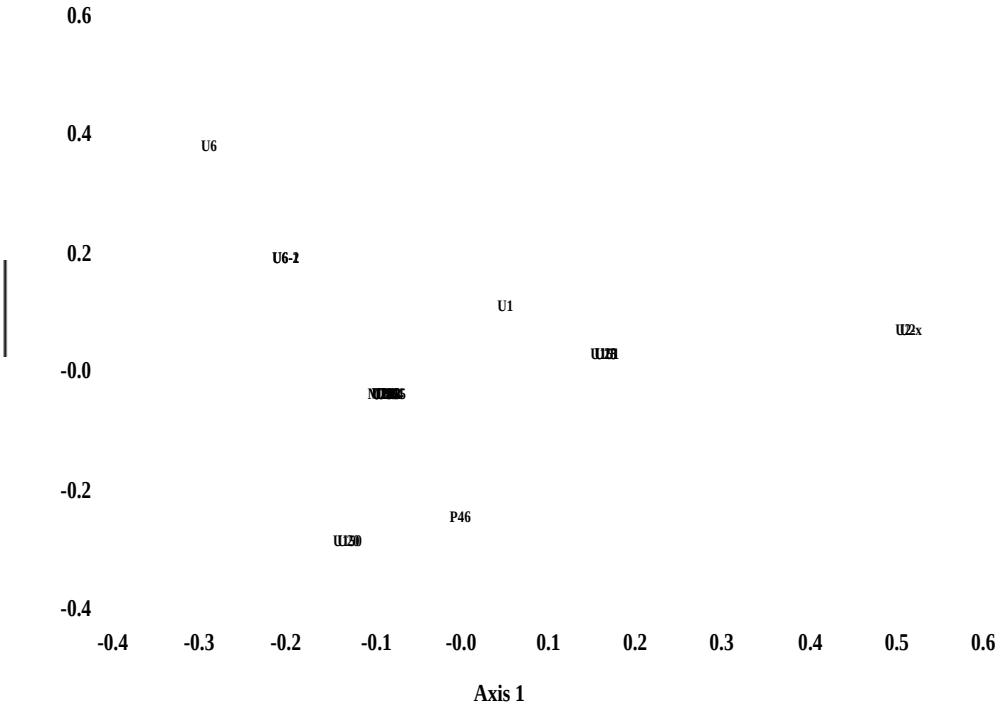




5 units; 55%, 22%, 13%

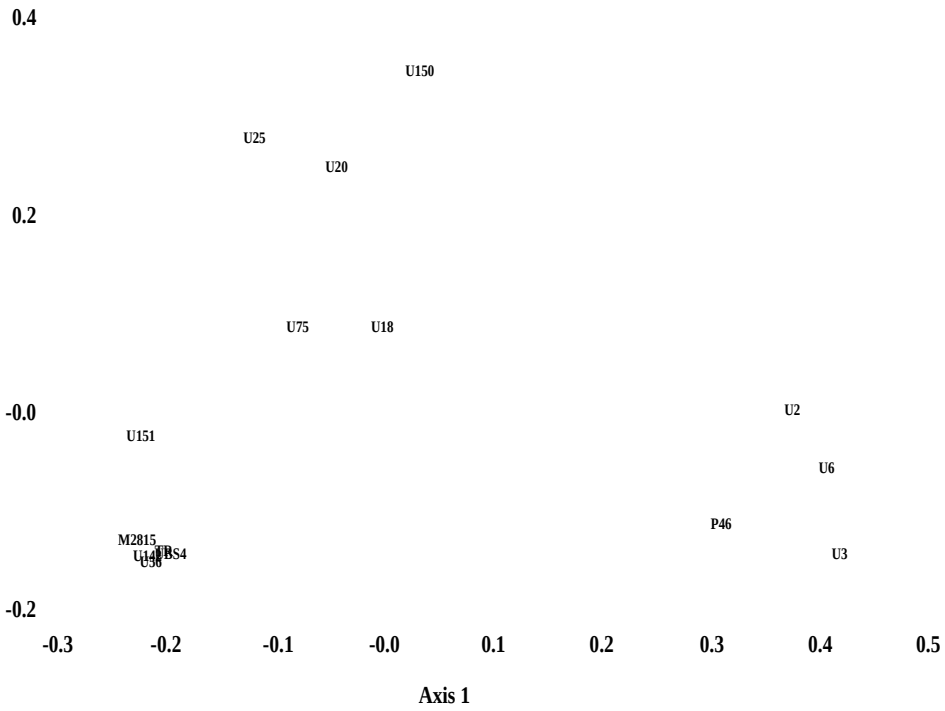
U2-x

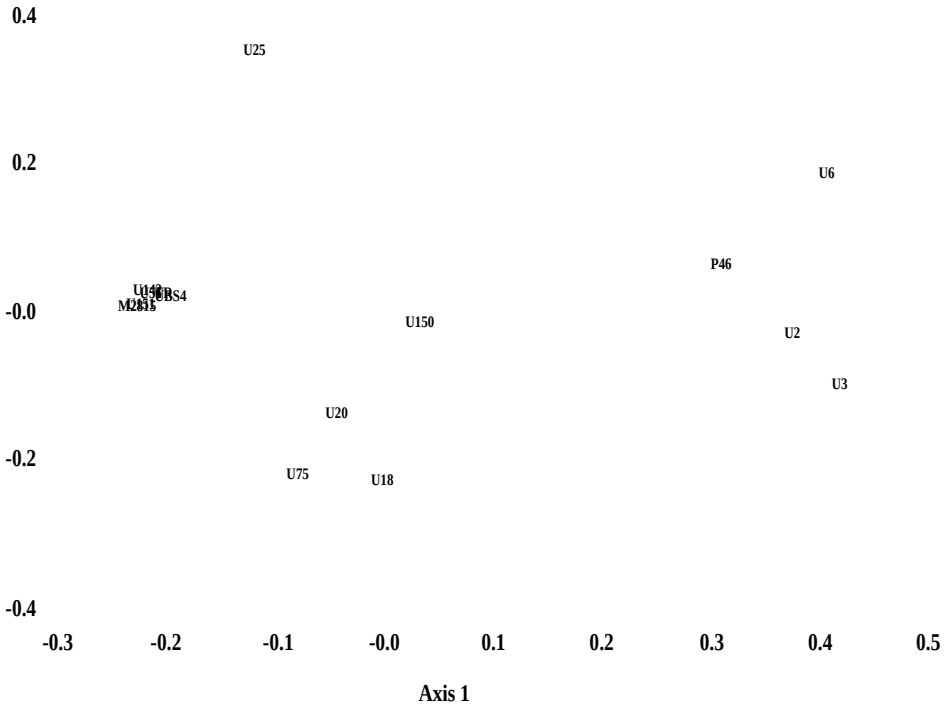




7 units; 35%, 23%, 19%

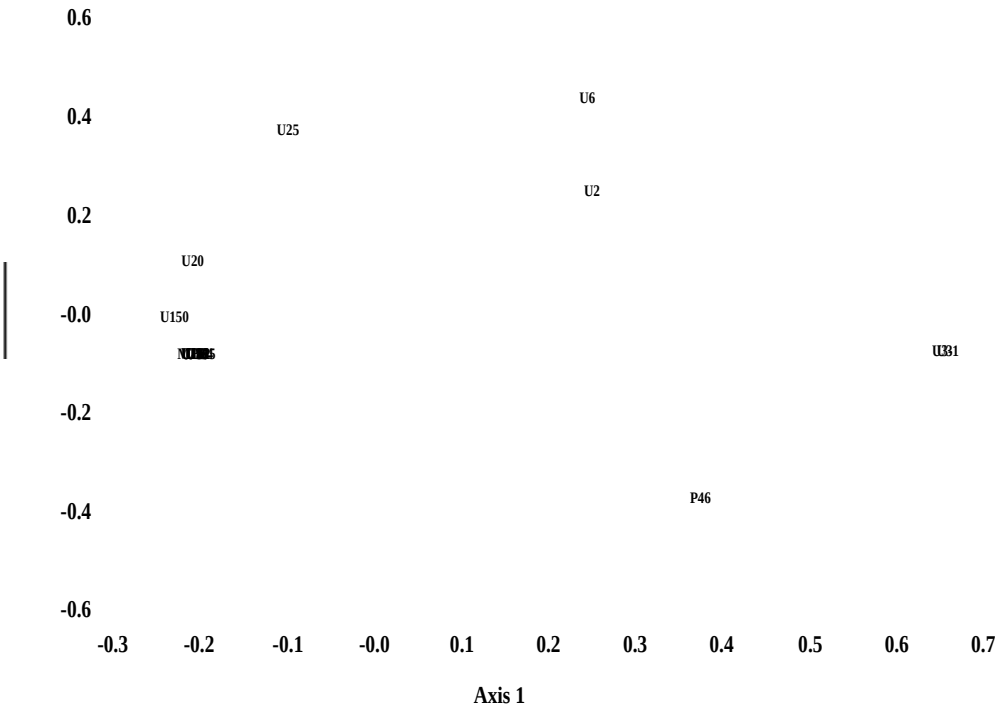
U3 (Alexandria? 325?)

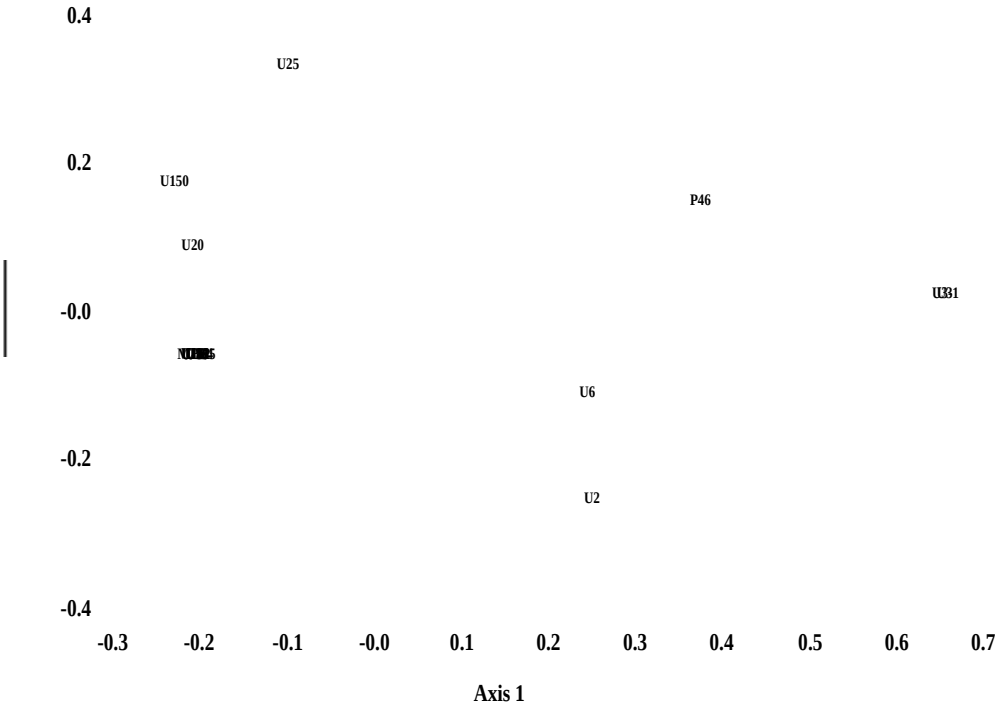




200 units; 35%, 17%, 12%

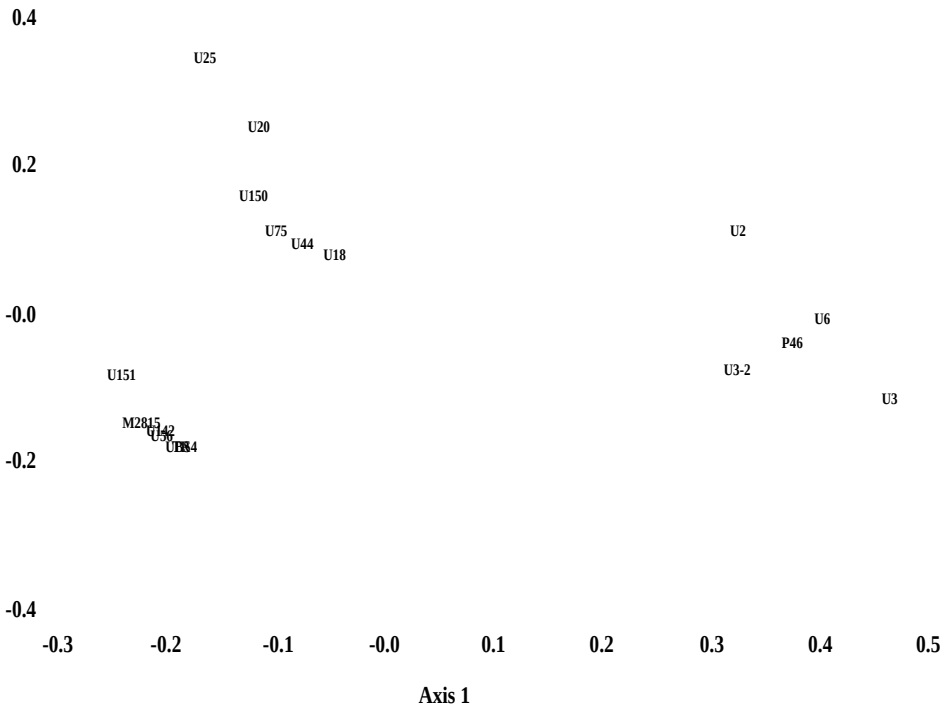
U3-1

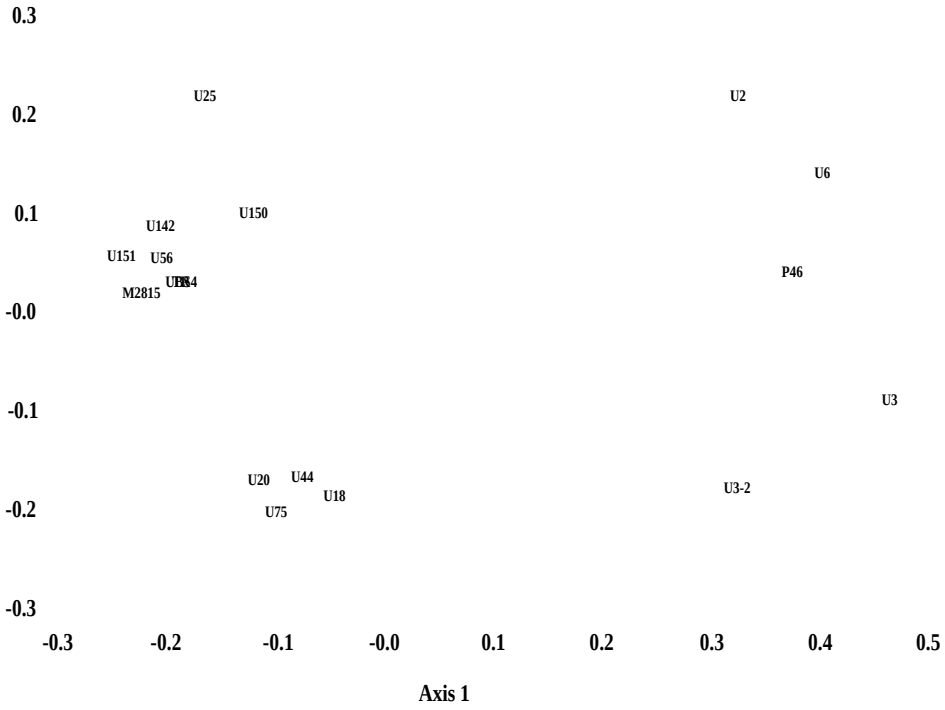




8 units; 58%, 23%, 10%

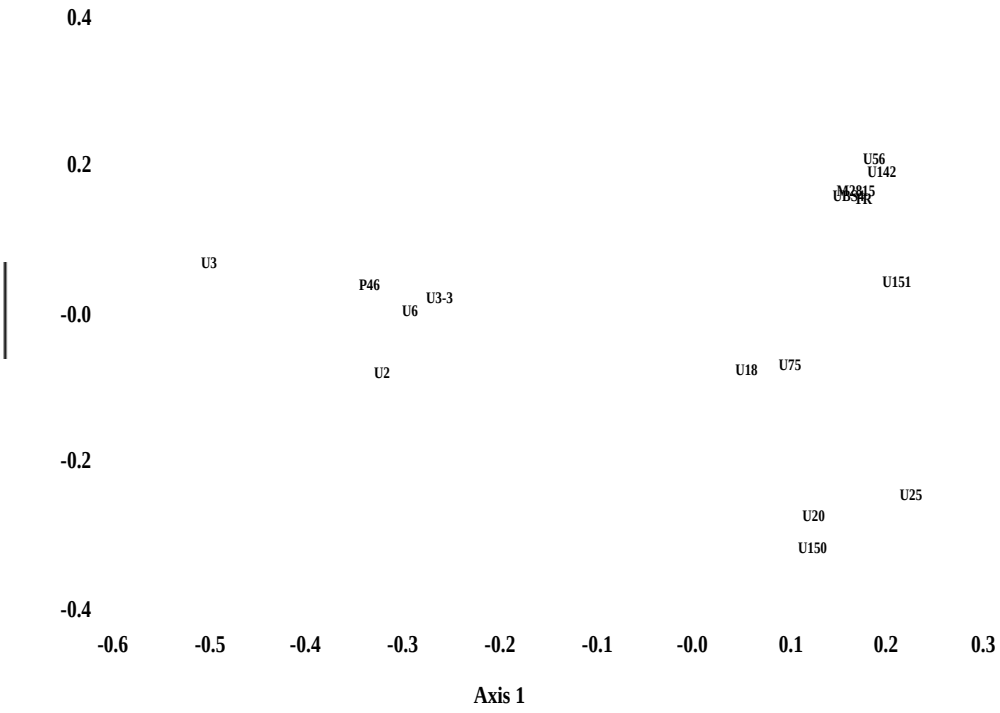
U3-2

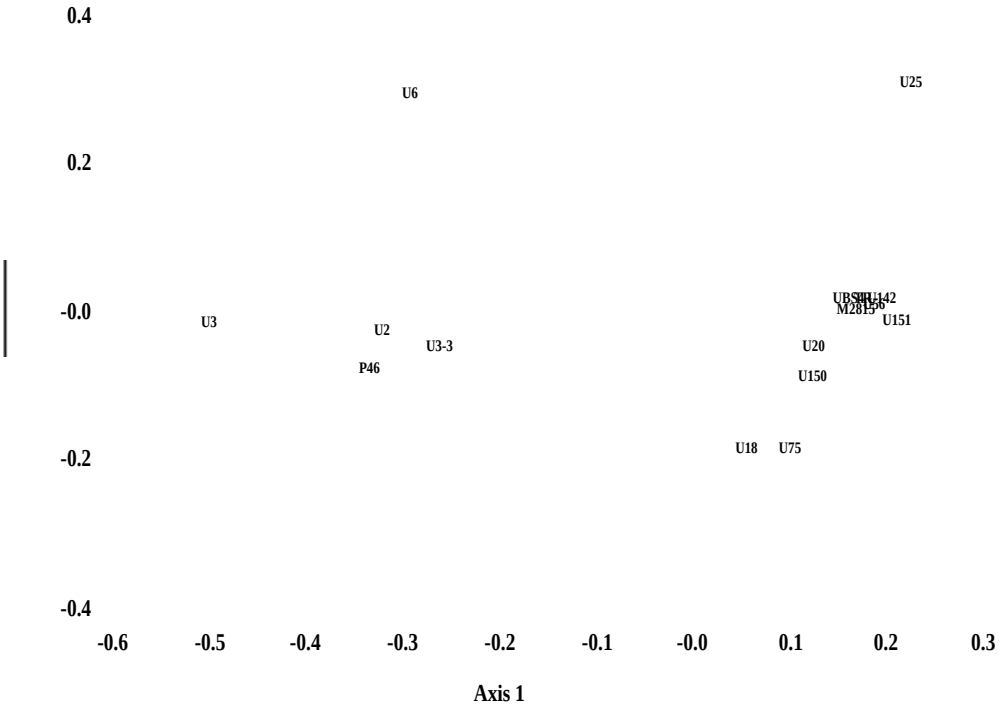




40 units; 38%, 15%, 11%

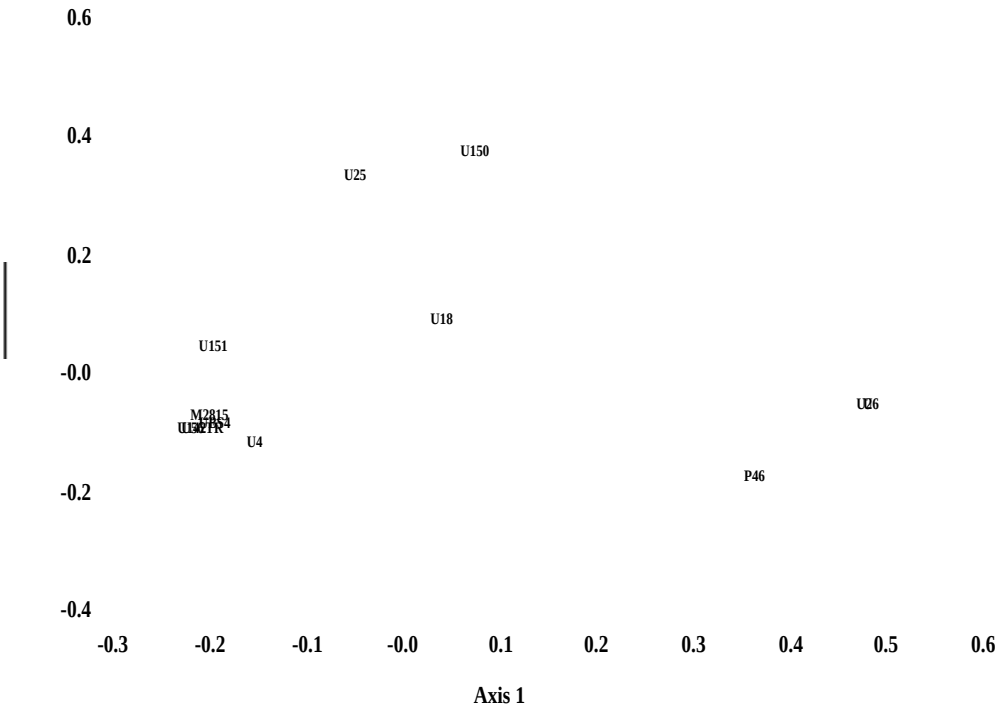
U3-3

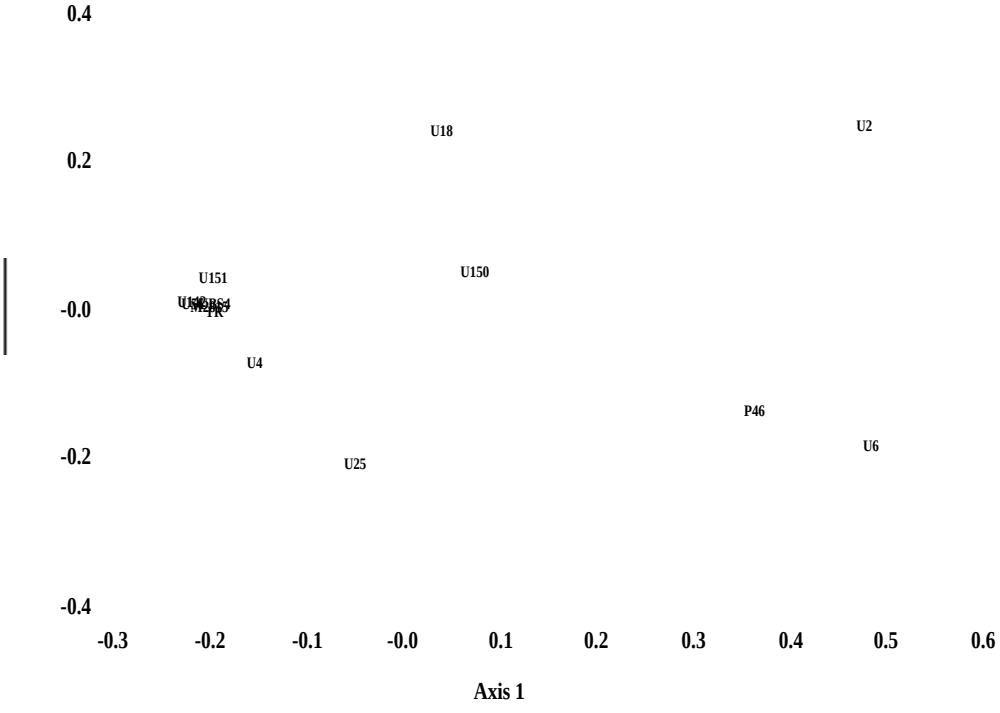




76 units; 36%, 16%, 11%

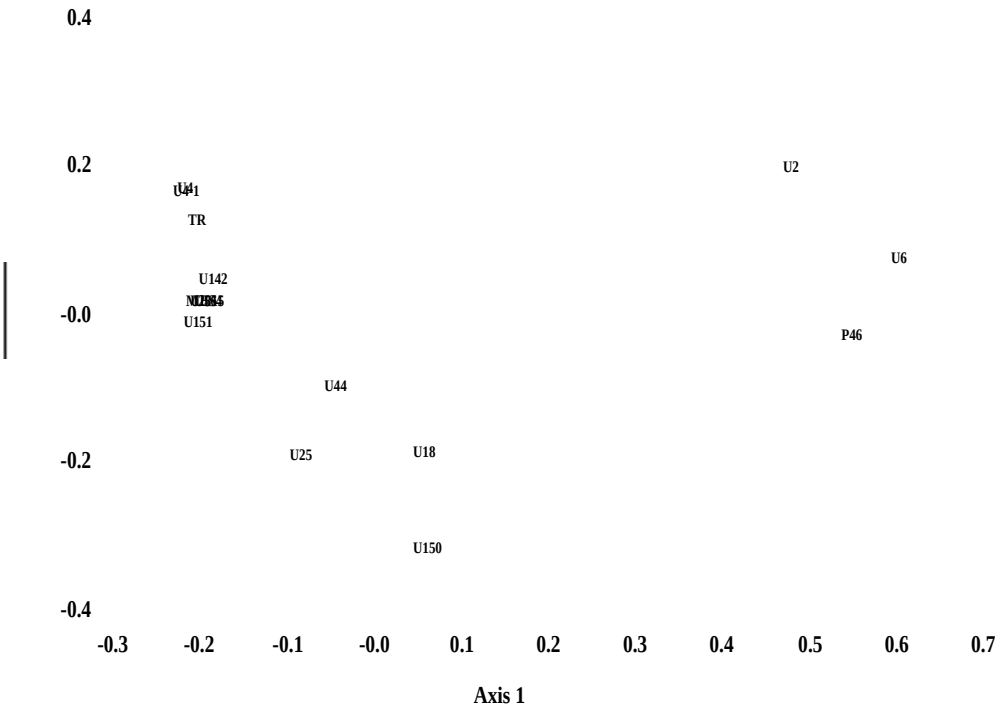
U4 (450?)

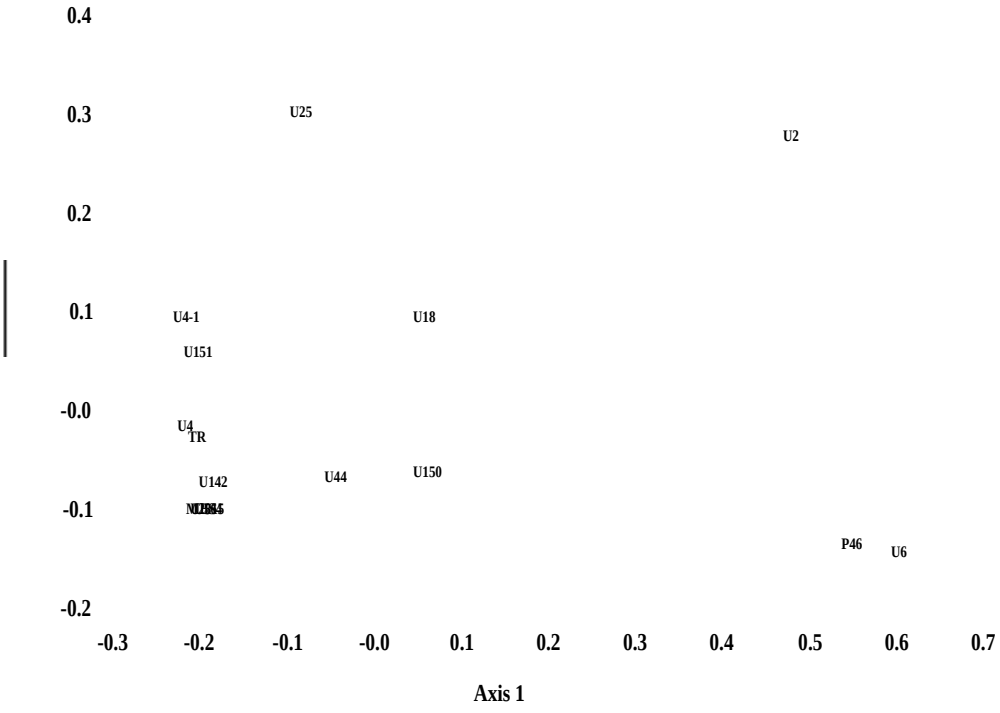




192 units; 40%, 16%, 10%

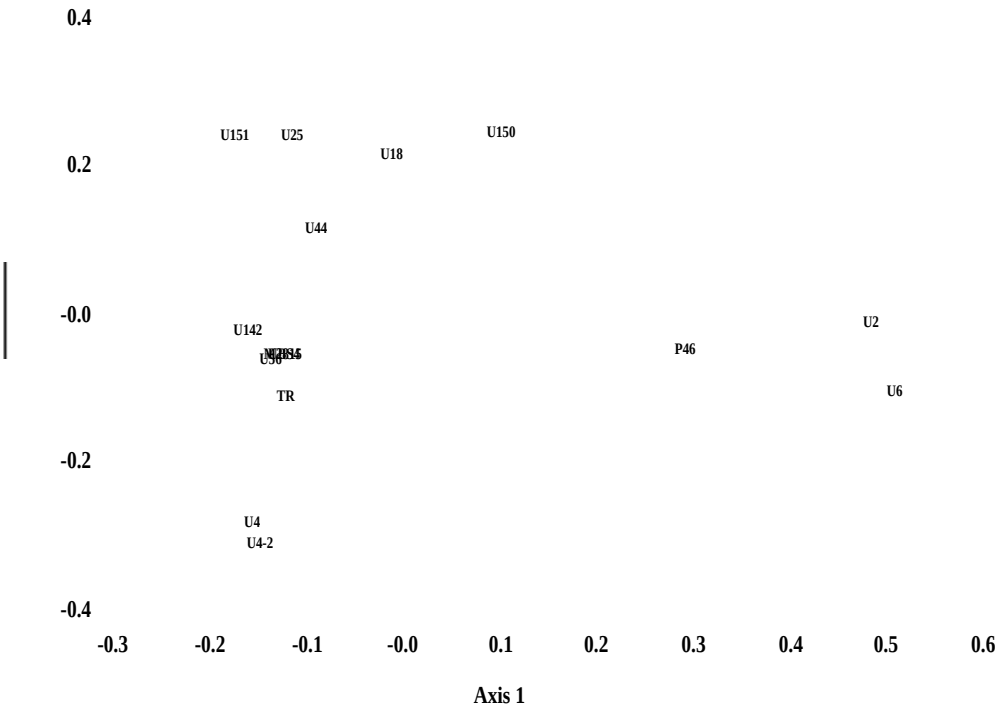
U4-1

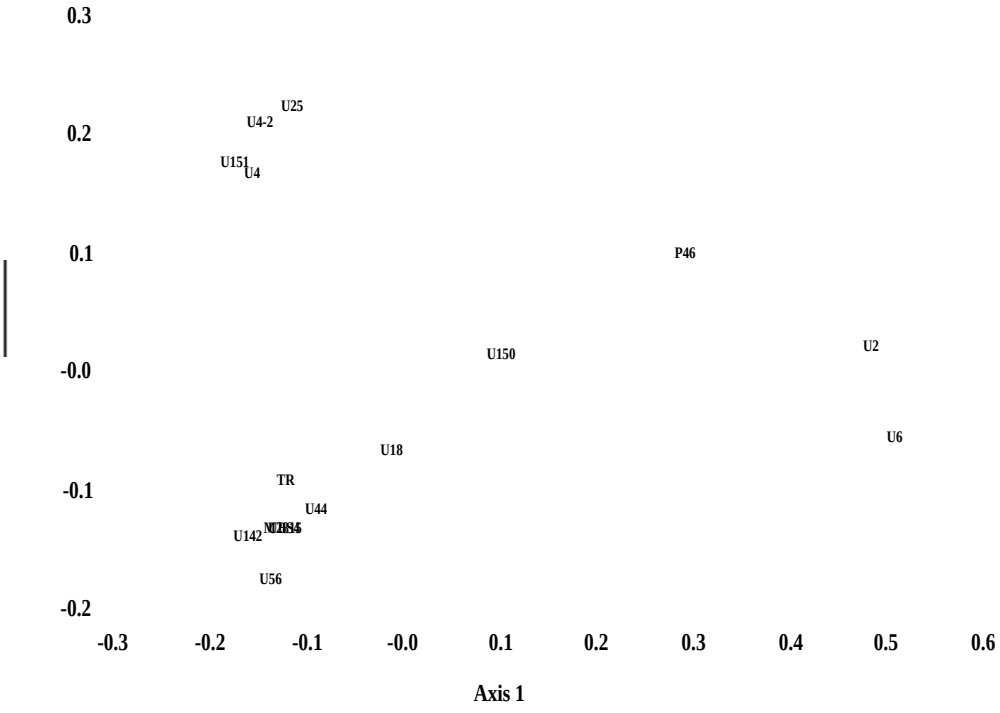




27 units; 50%, 12%, 11%

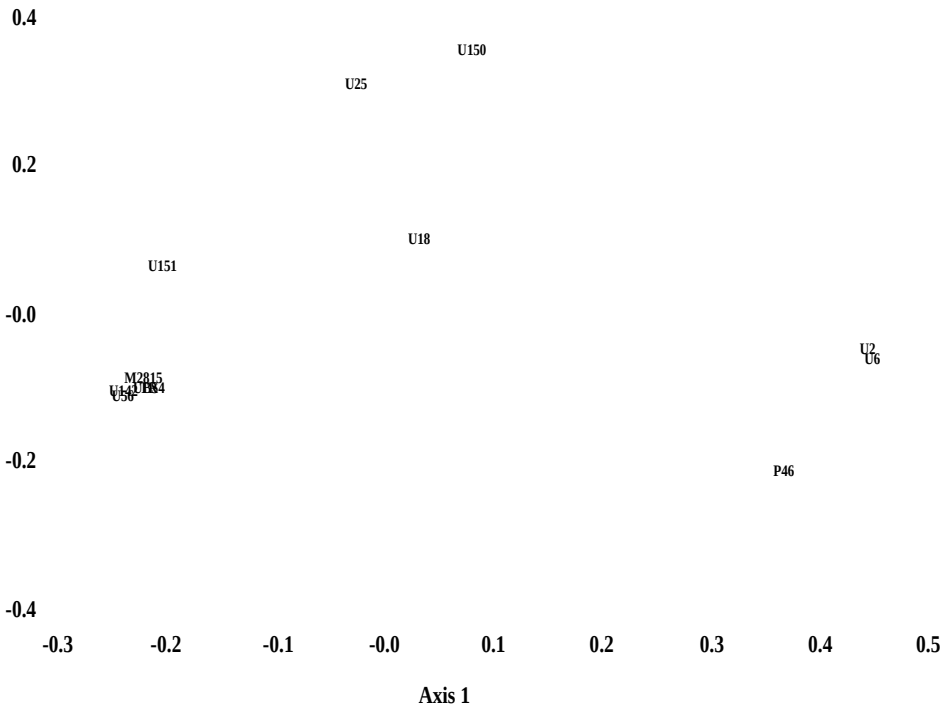
U4-2

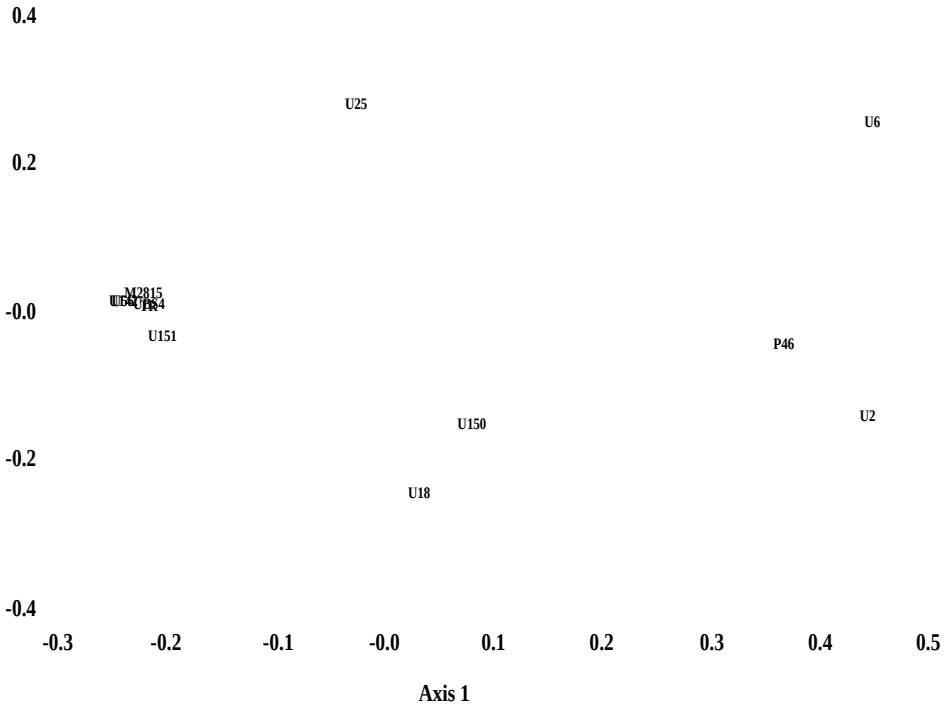




27 units; 34%, 20%, 12%

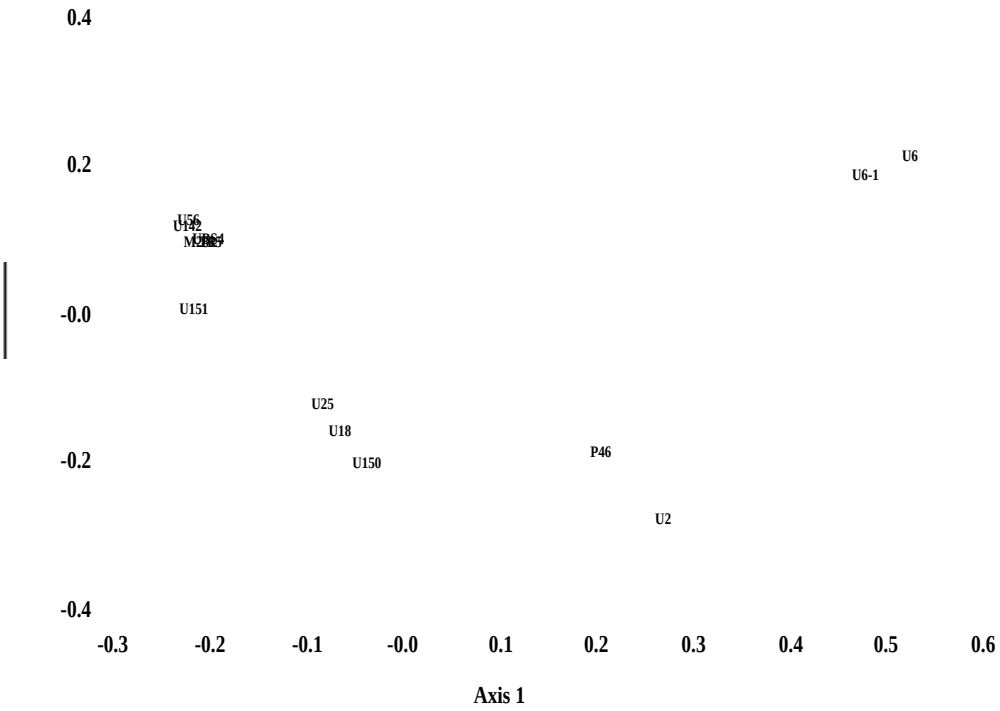
U6 (Sardinia? 500?)

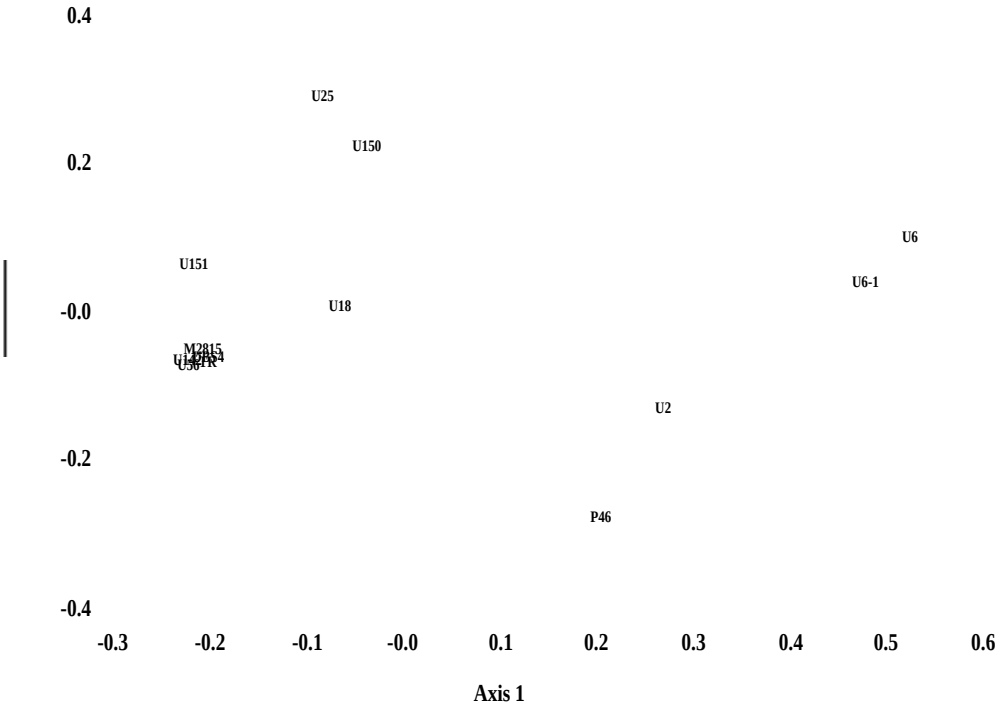




317 units; 40%, 16%, 12%

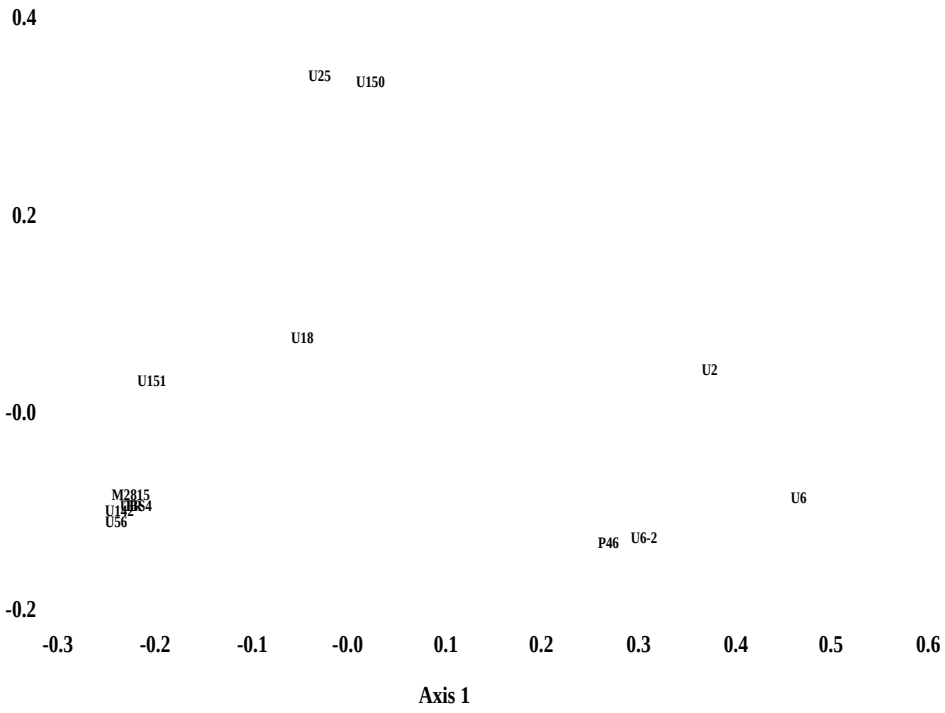
U6-1

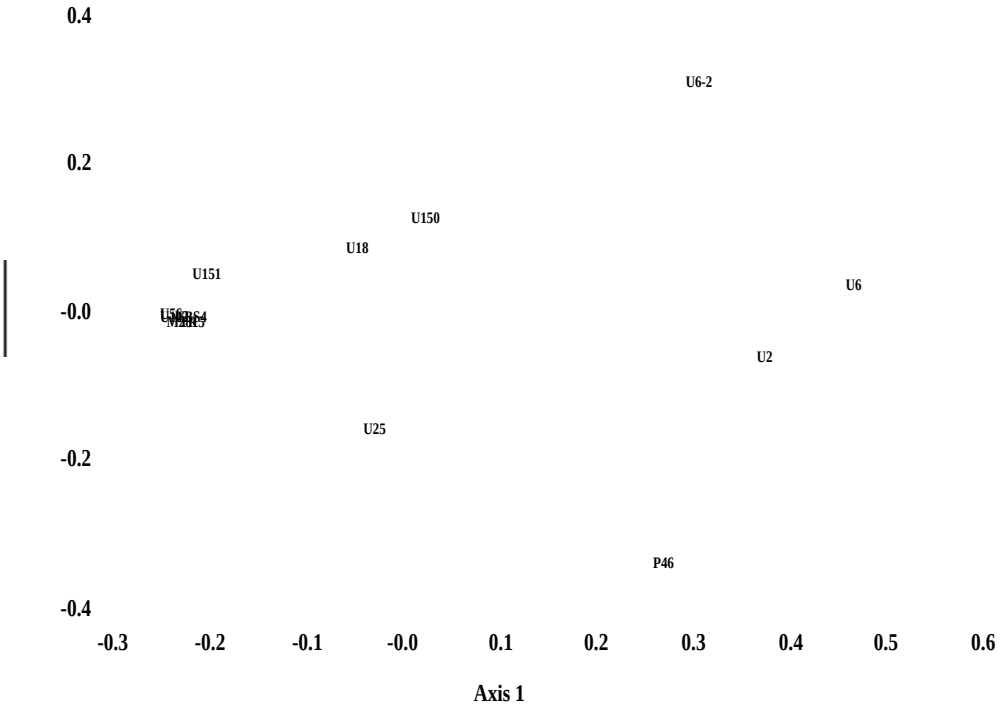




295 units; 42%, 15%, 12%

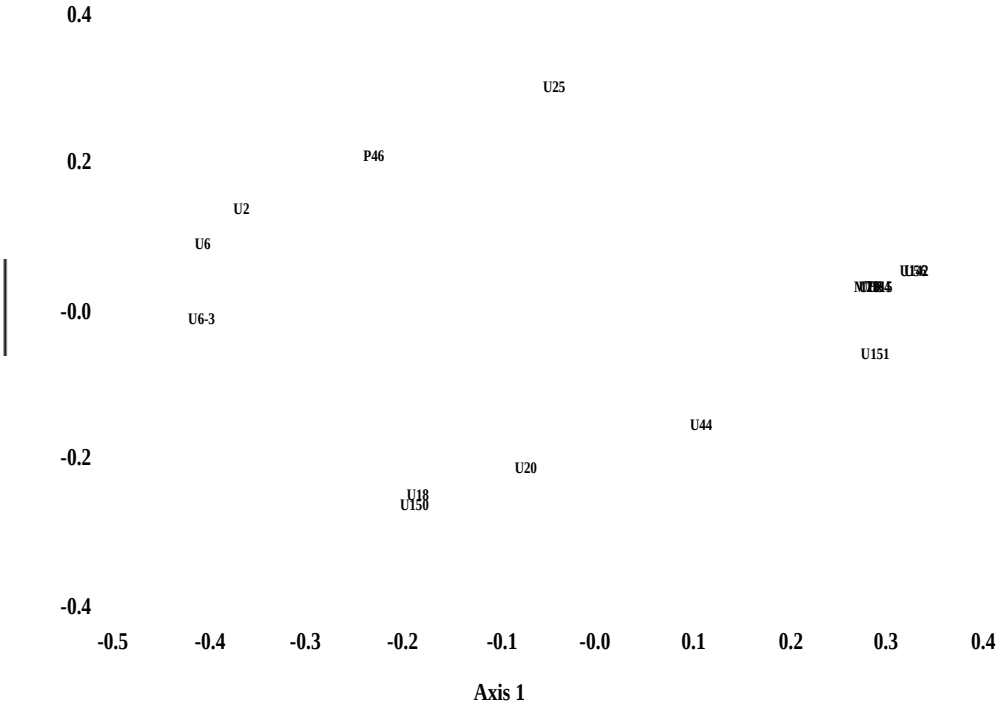
U6-2

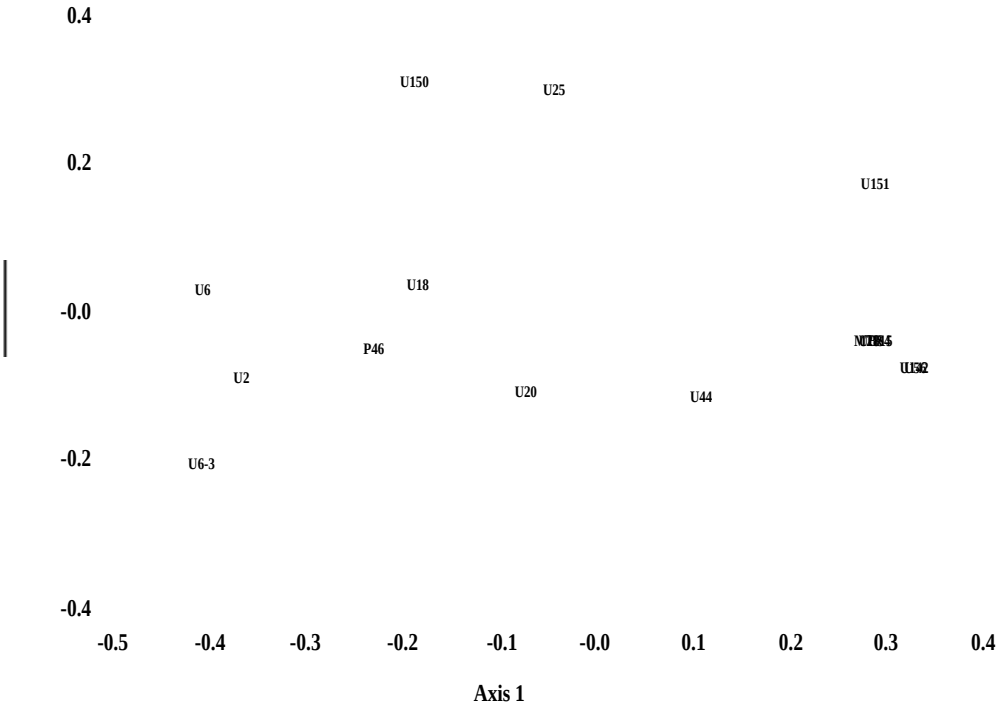




230 units; 38%, 15%, 12%

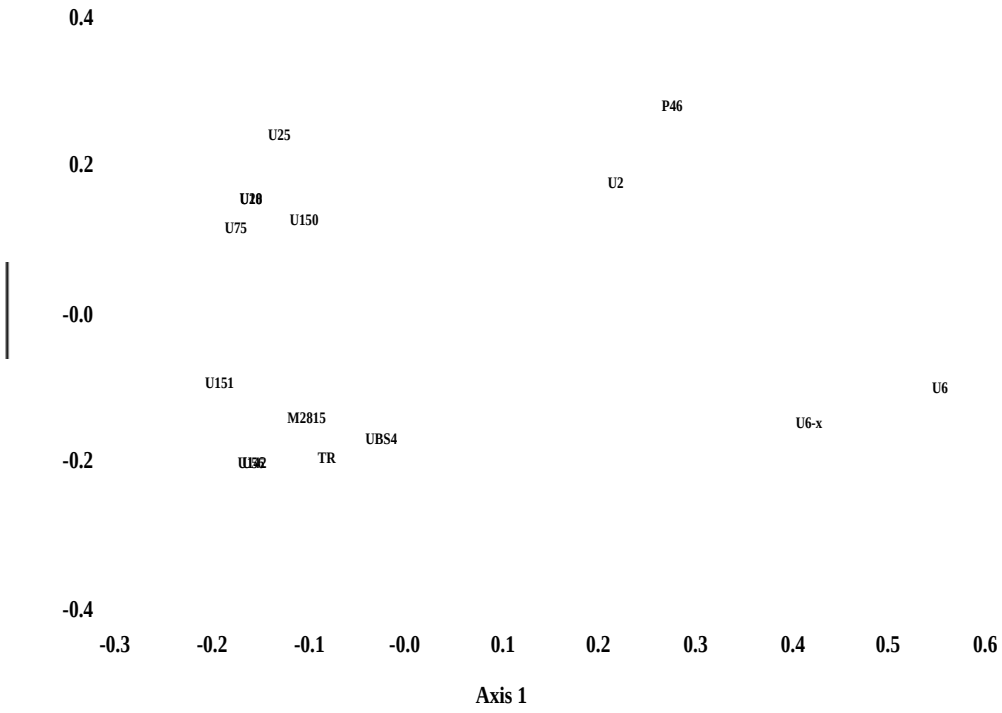
U6-3

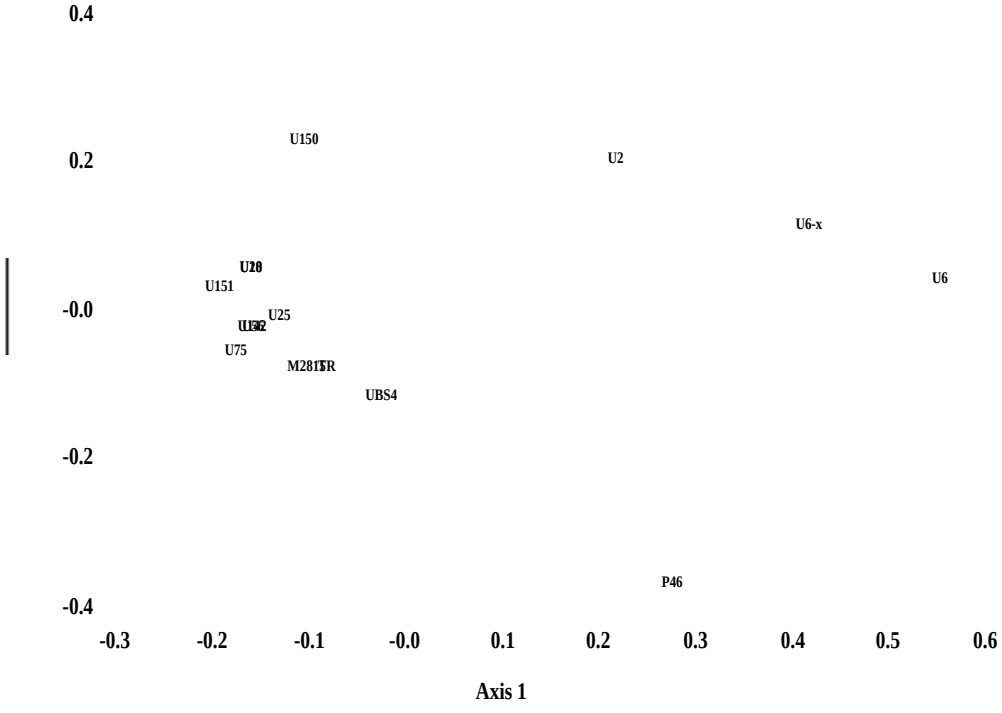




27 units; 45%, 14%, 12%

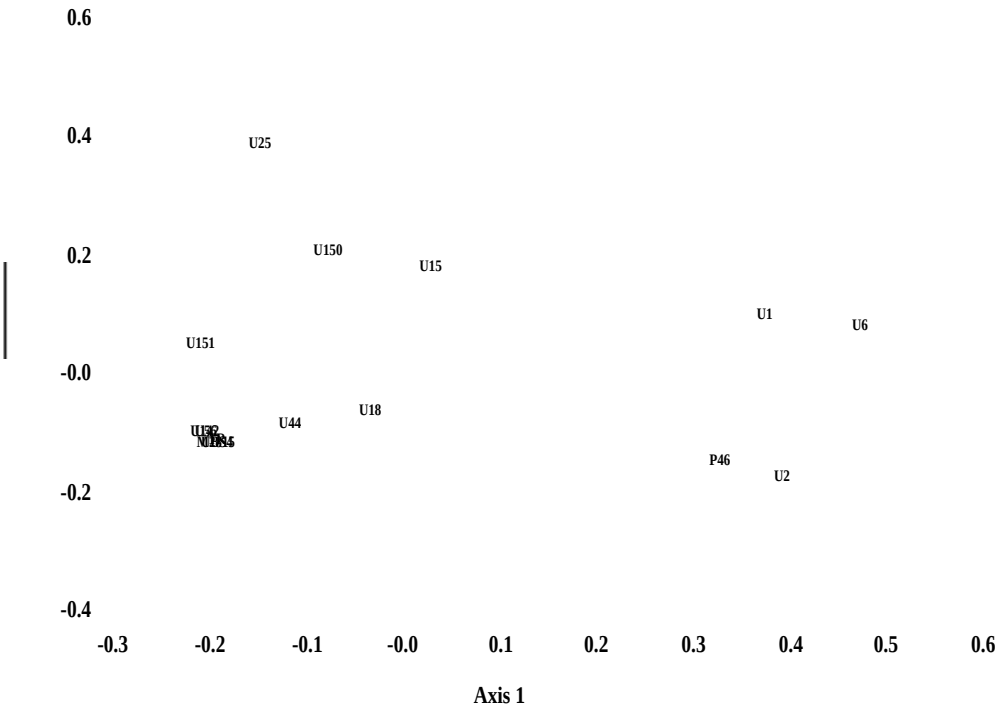
U6-x

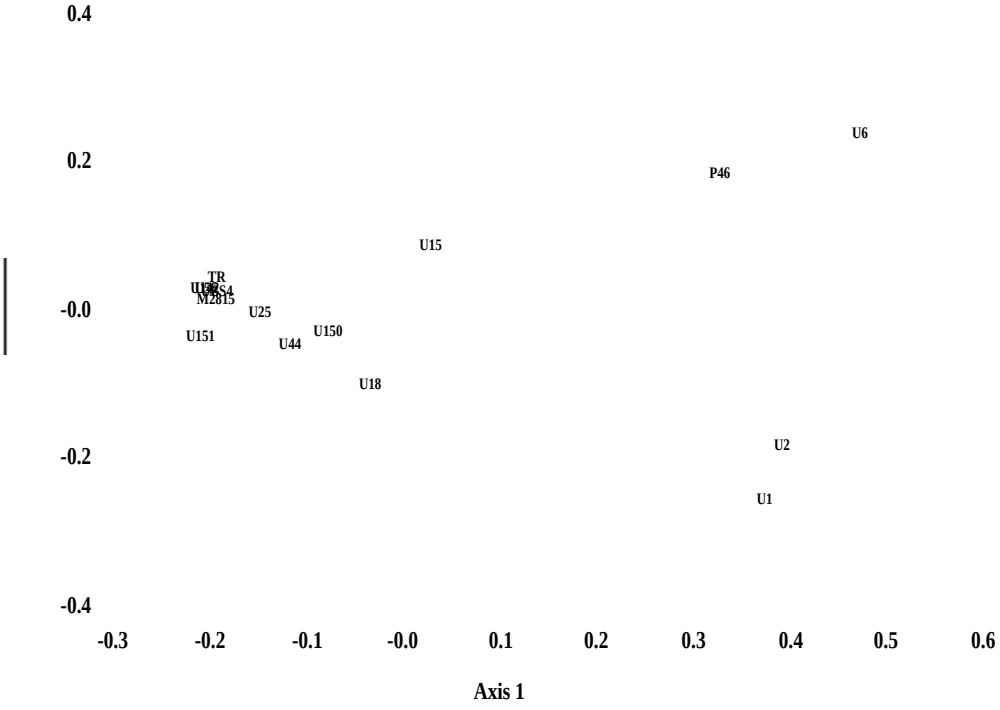




23 units; 35%, 19%, 12%

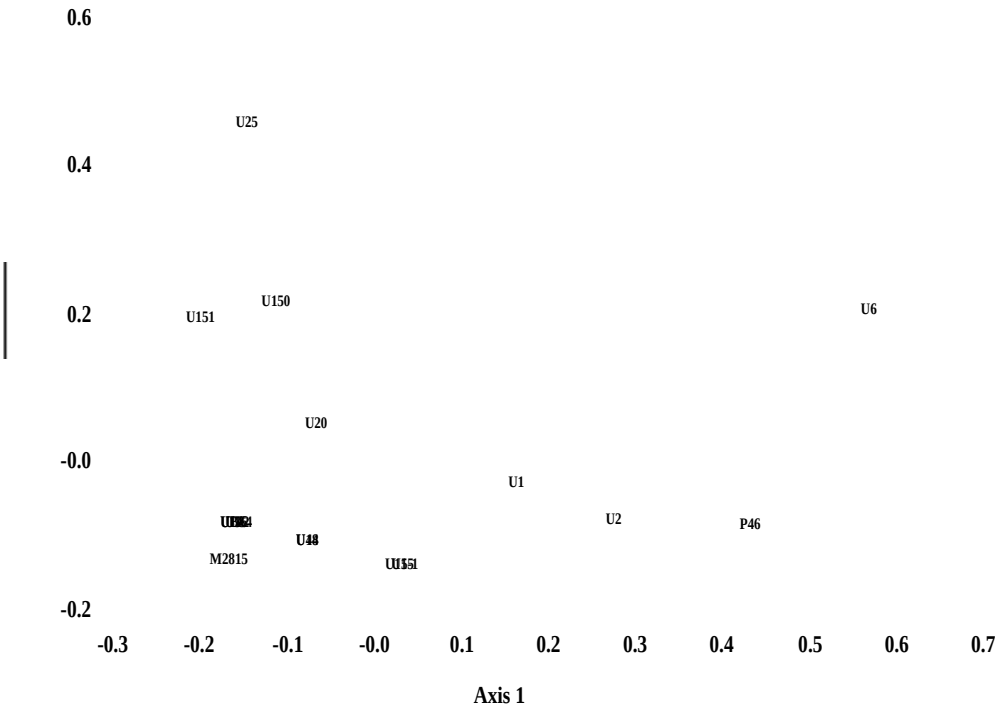
U15 (Caesarea? 500?)

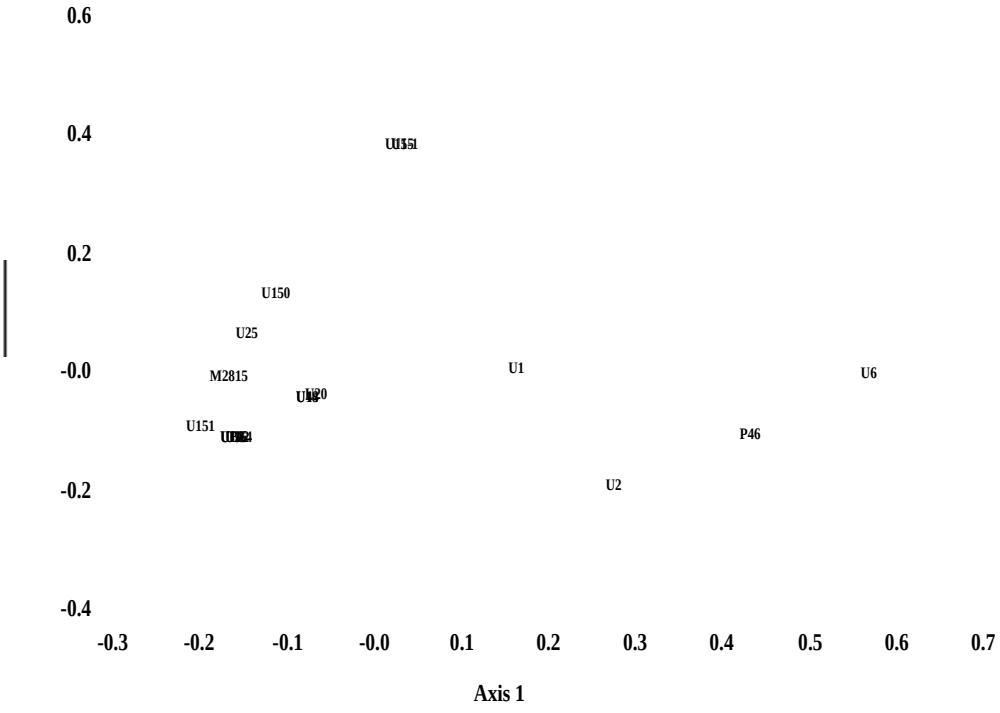




59 units; 38%, 15%, 9%

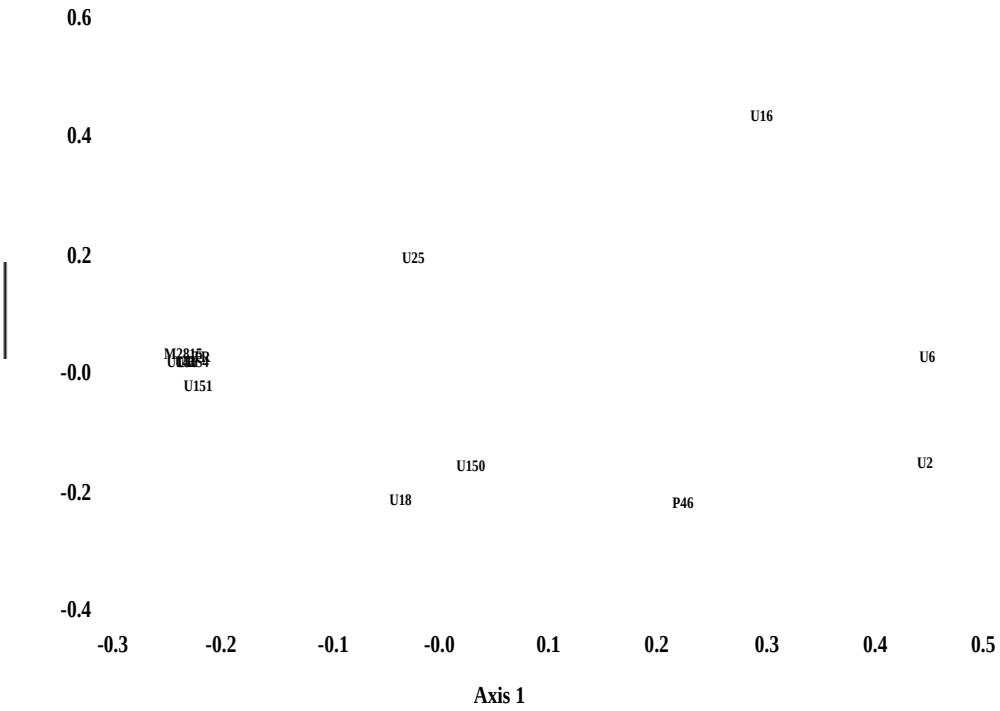
U15-1

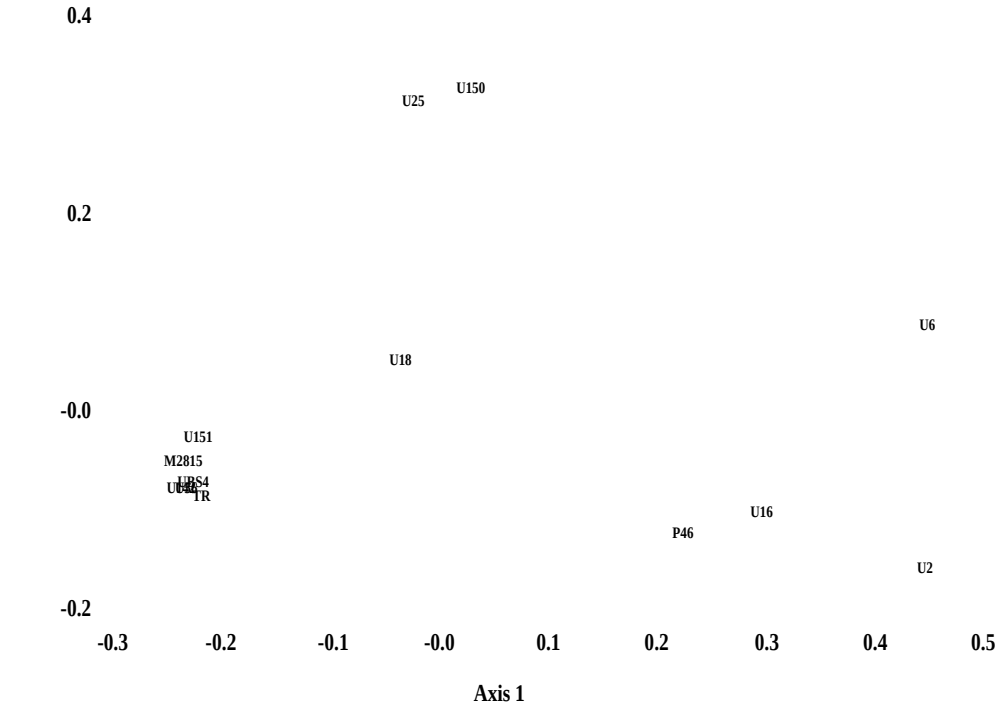




17 units; 34%, 19%, 18%

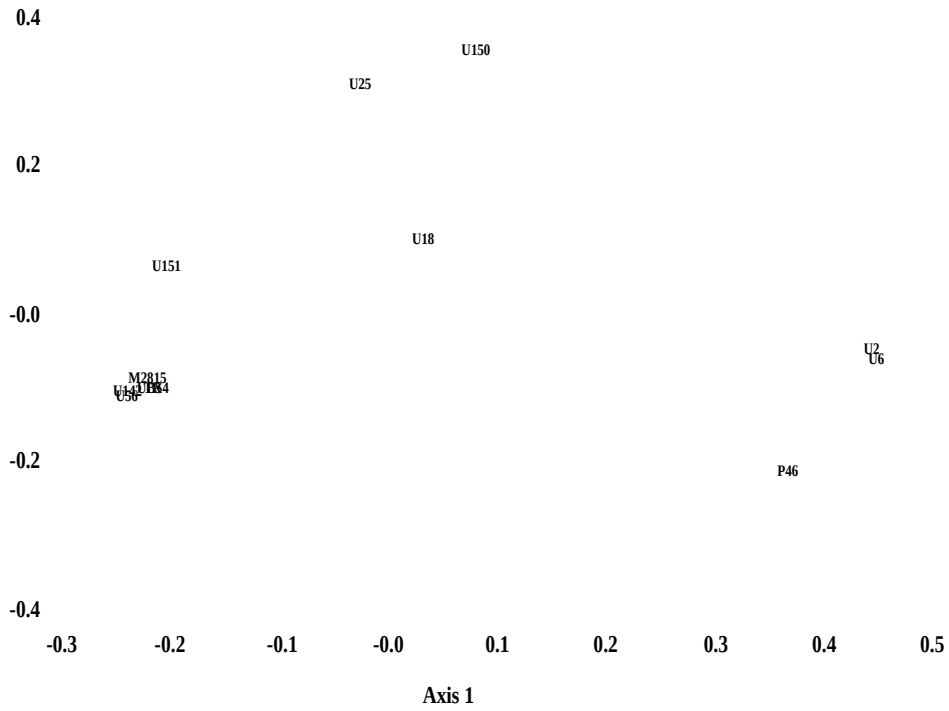
U16 (Northern Egypt? 450?)

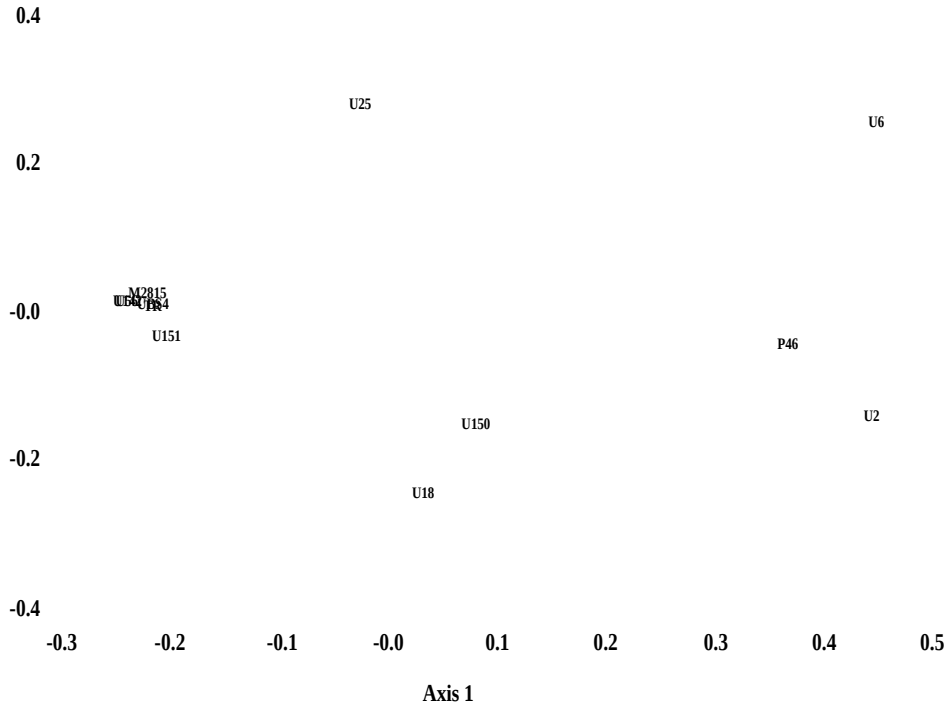




86 units; 38%, 17%, 13%

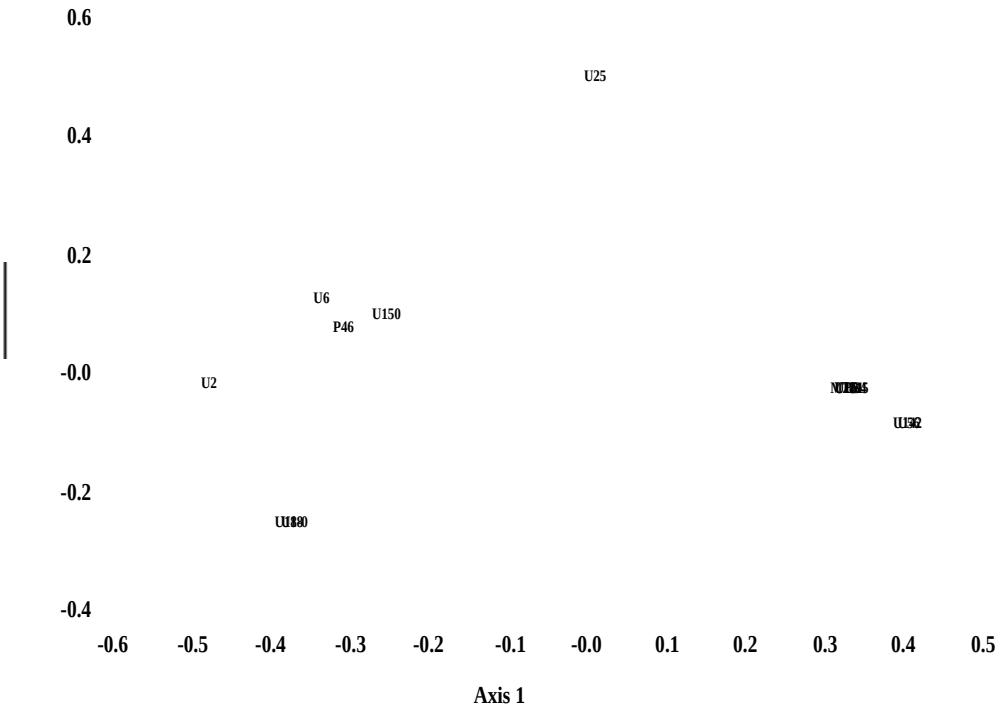
U18 (850?)

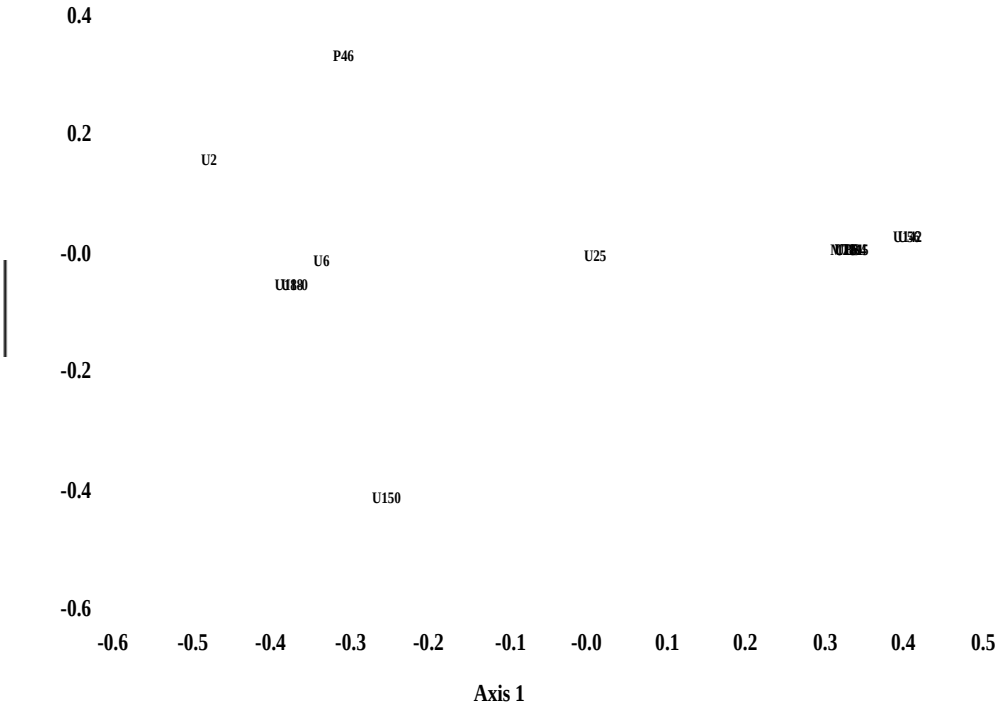




317 units; 40%, 16%, 12%

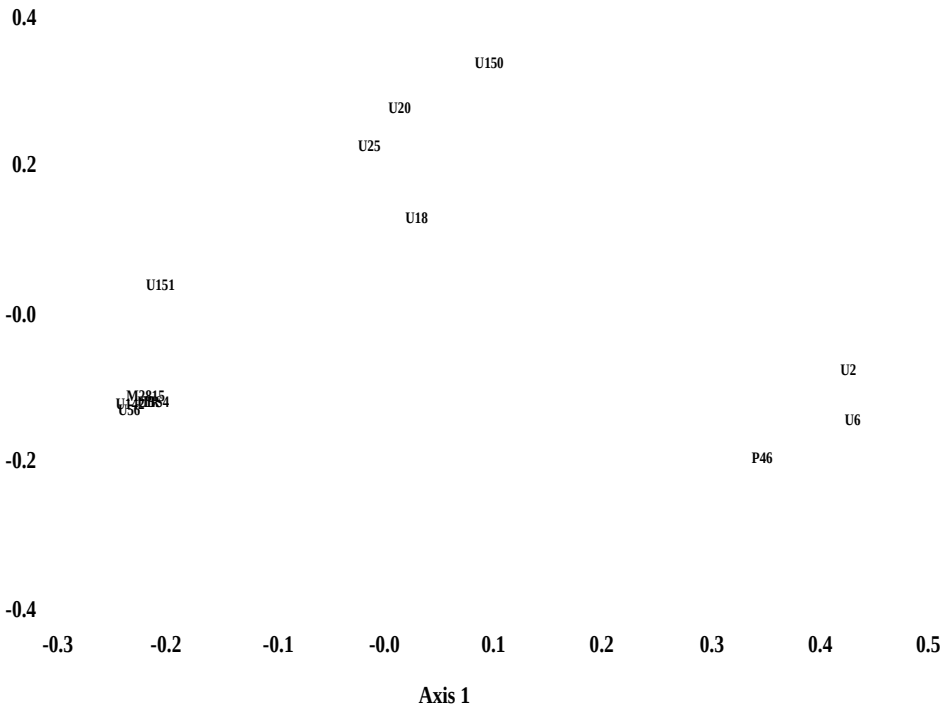
U18-0

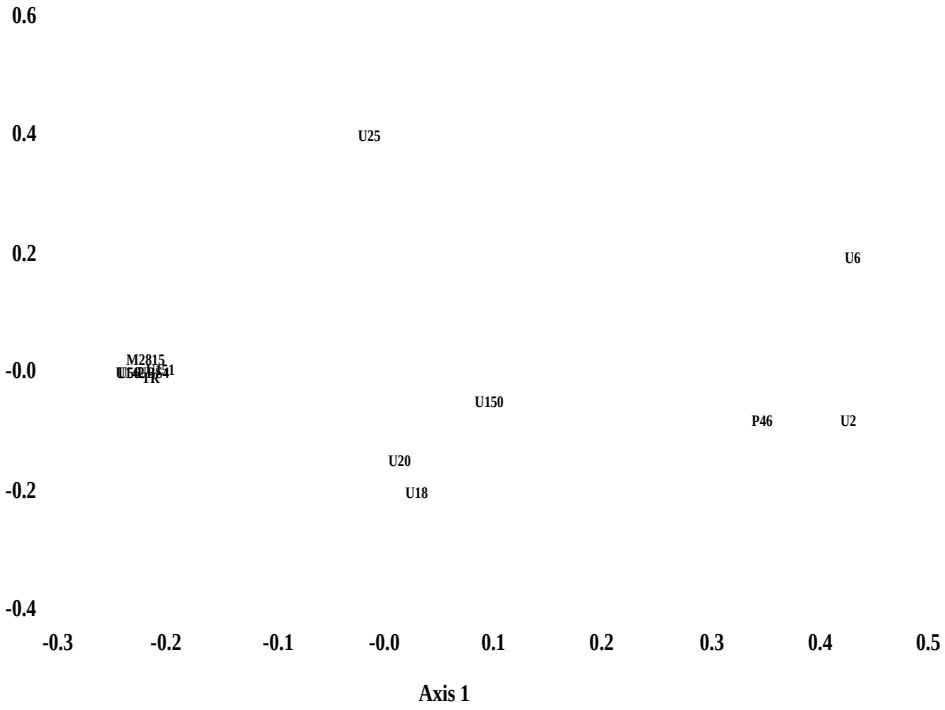




7 units; 59%, 16%, 12%

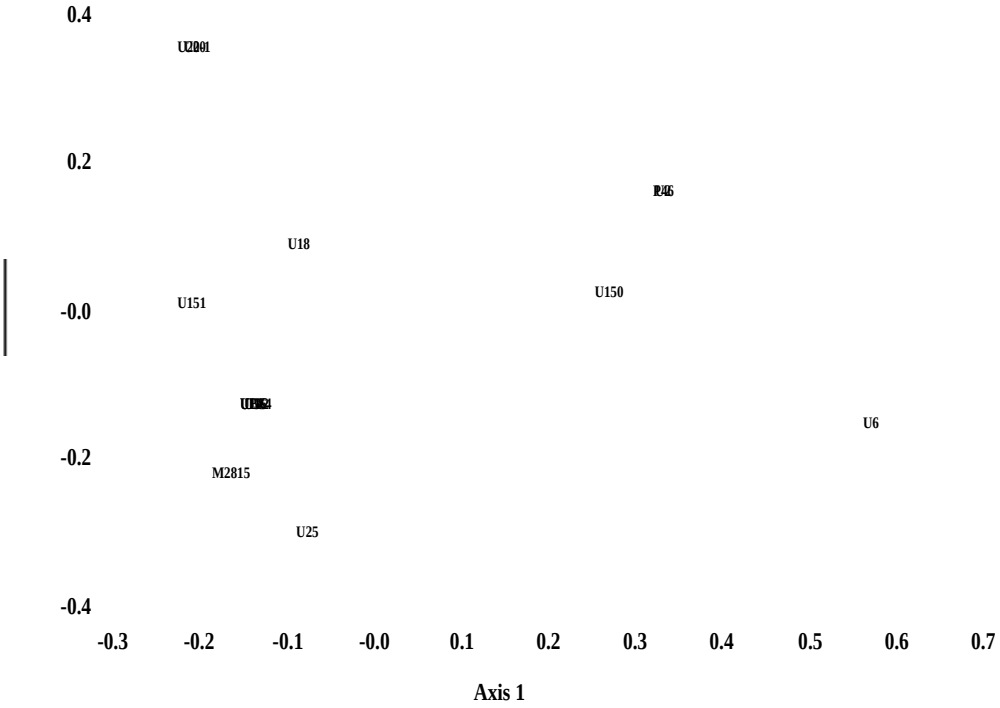
U20 (850?)

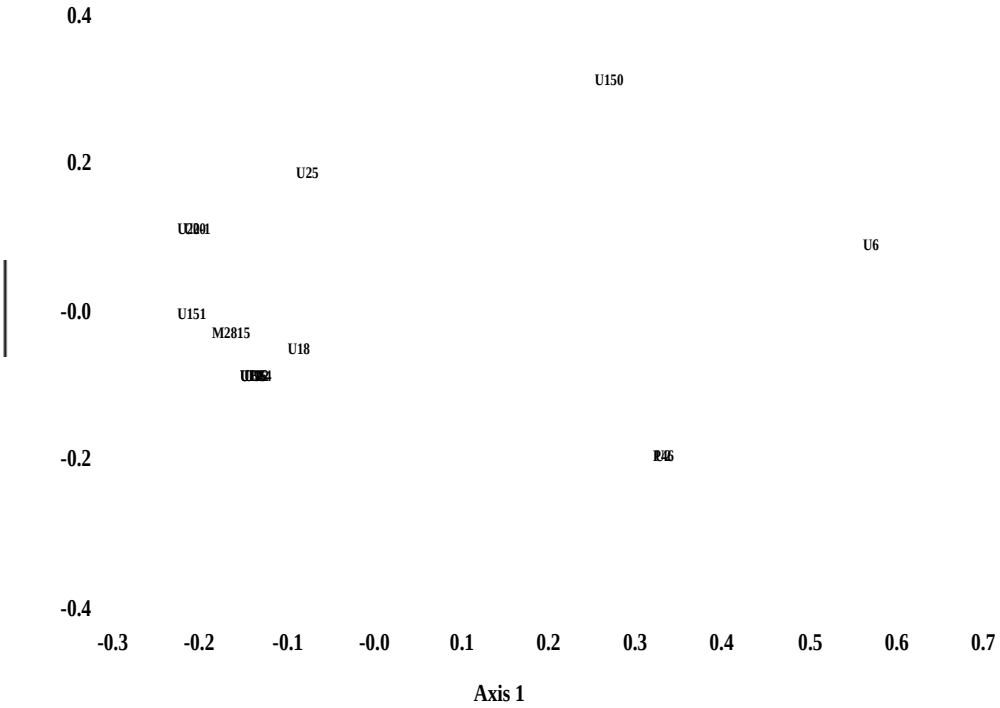




306 units; 36%, 18%, 13%

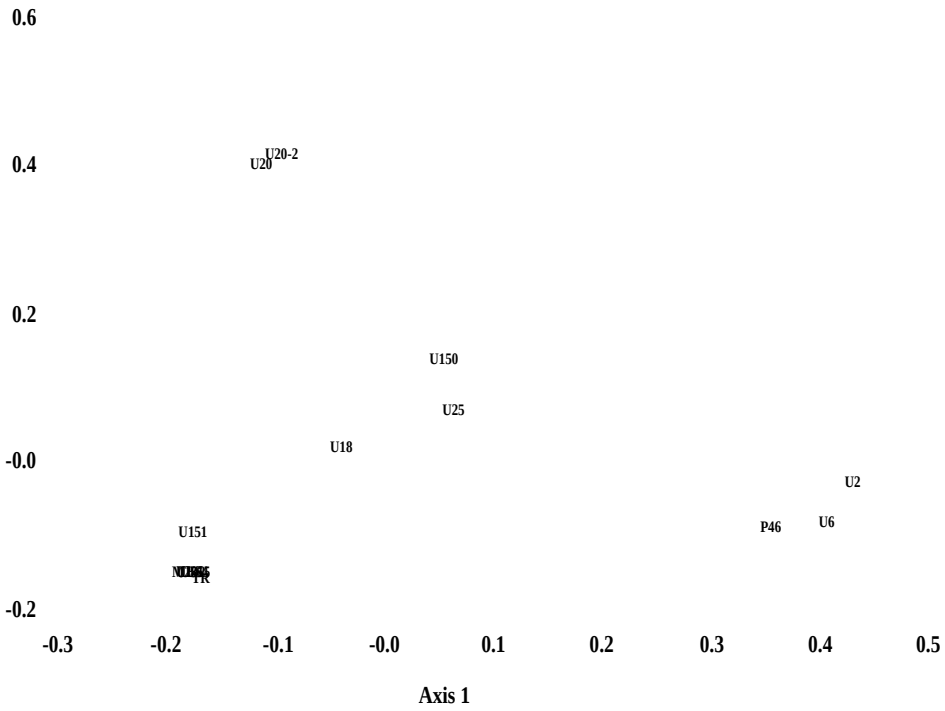
U20-1

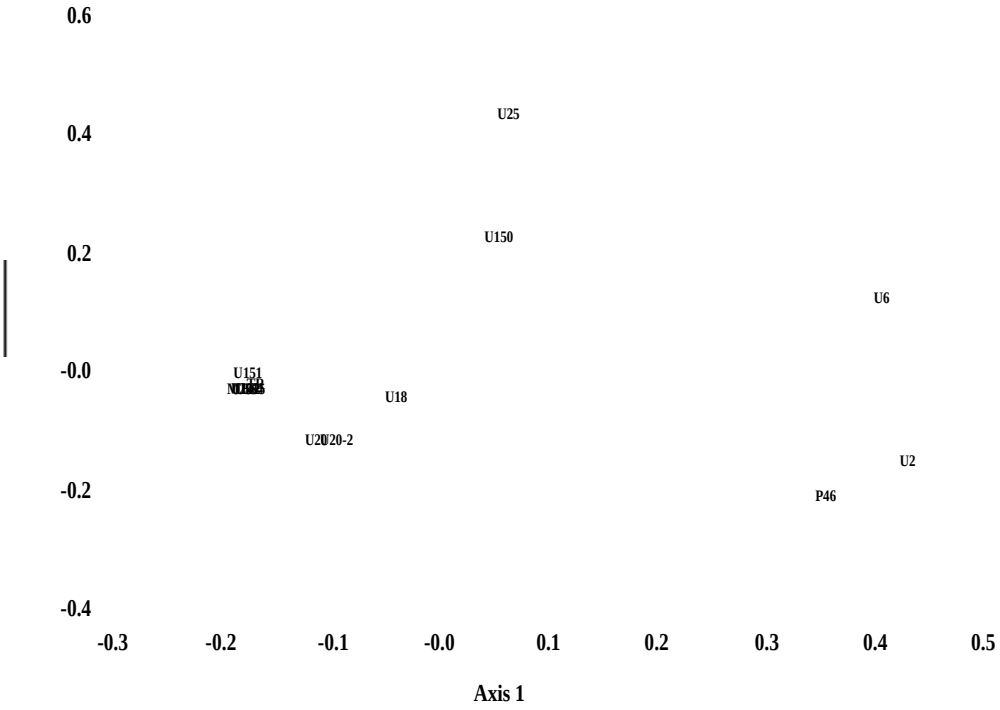




10 units; 40%, 25%, 13%

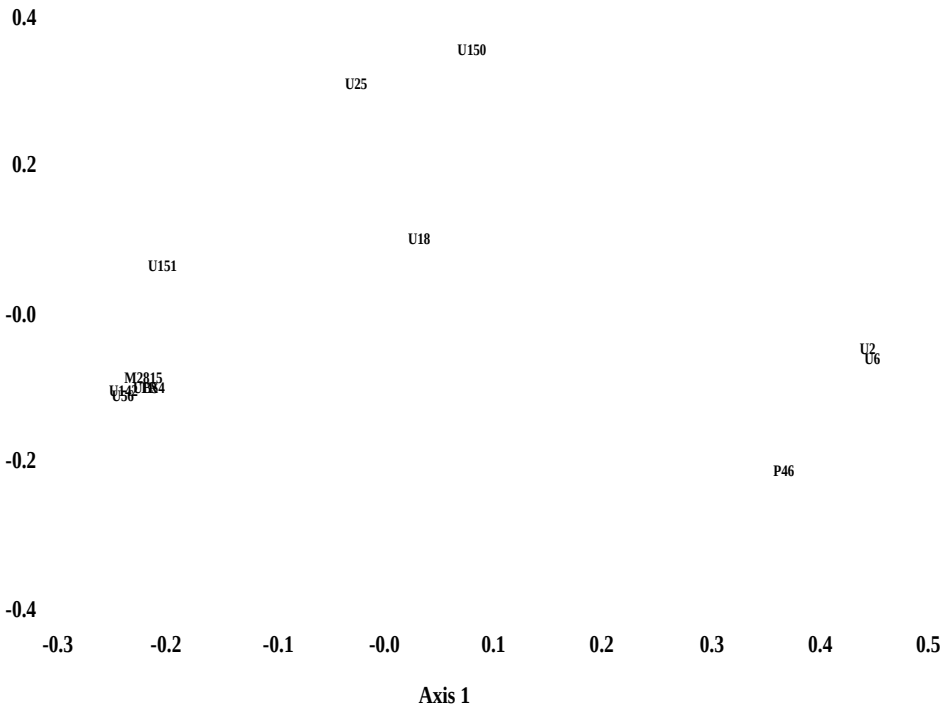
U20-2

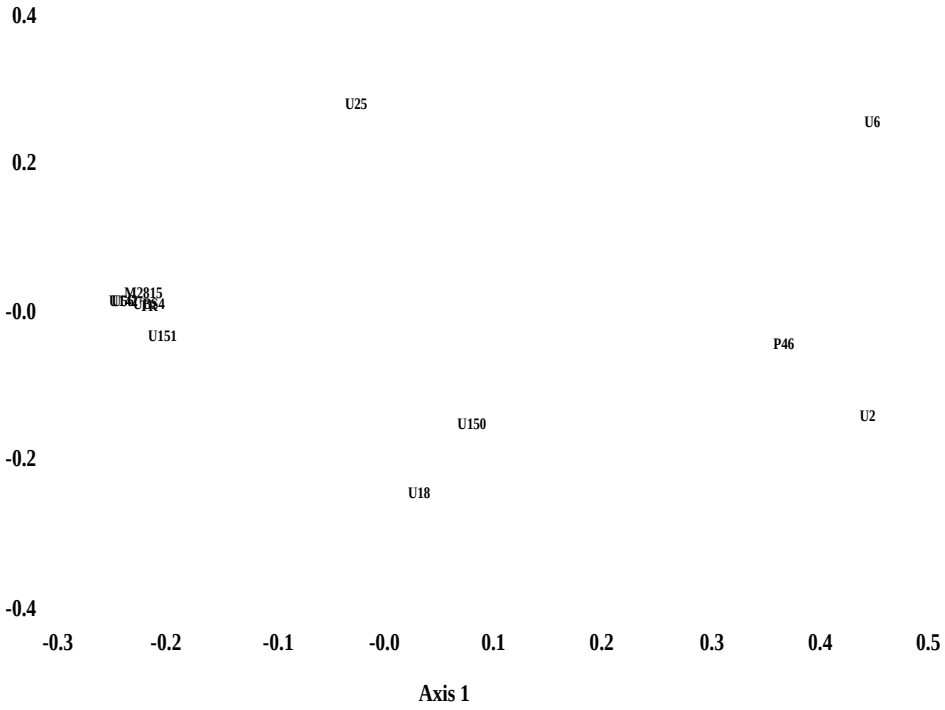




55 units; 31%, 23%, 16%

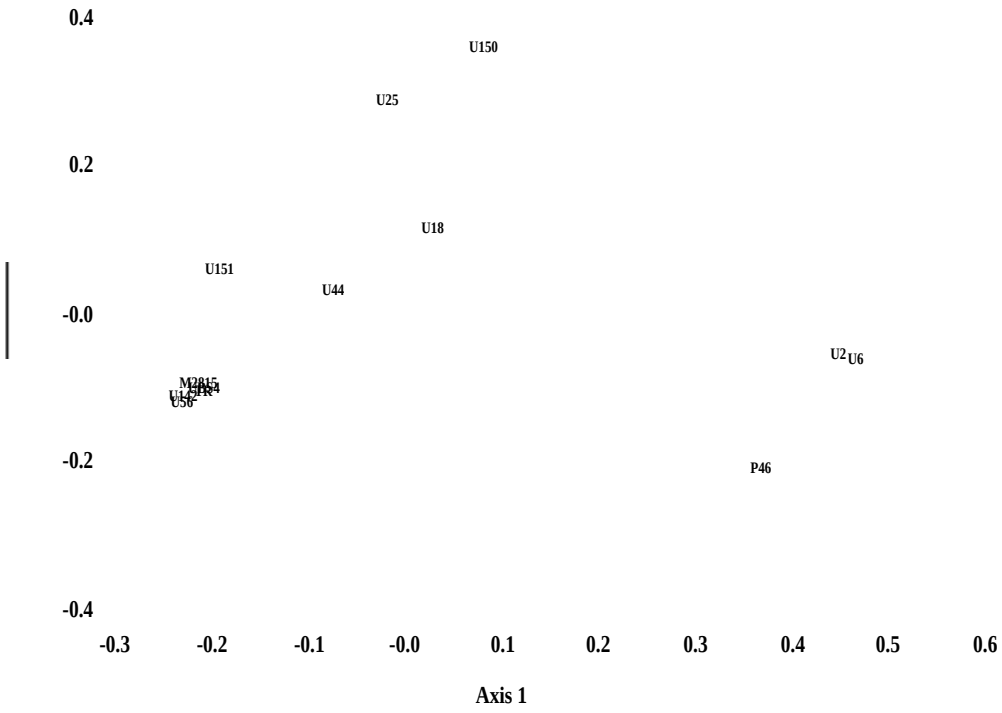
U25 (850?)

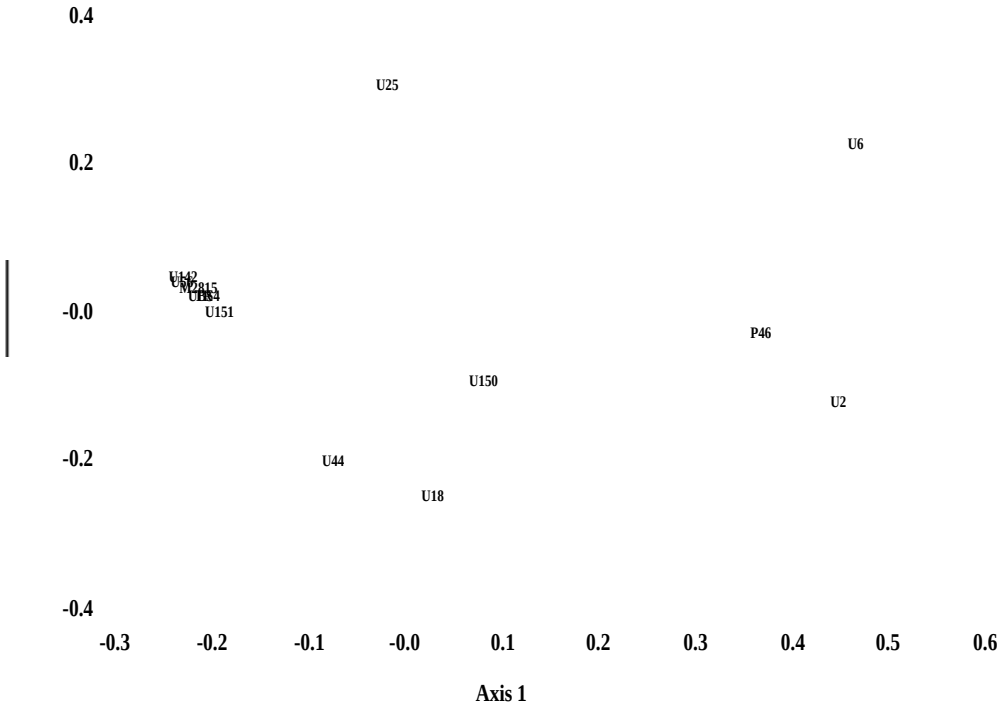




317 units; 40%, 16%, 12%

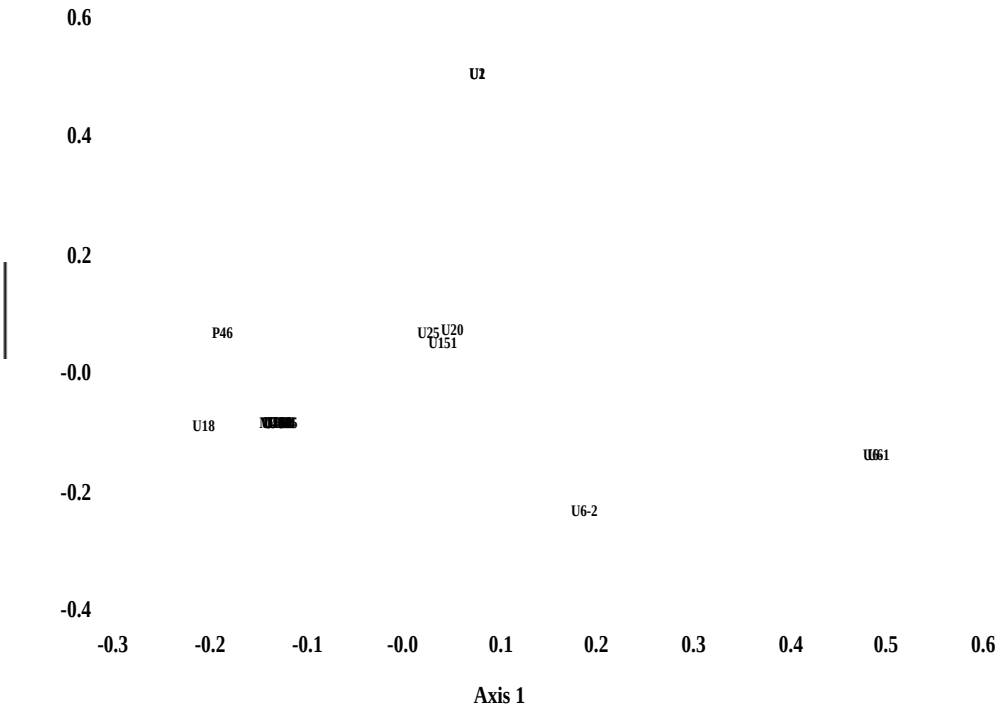
U44 (800?)

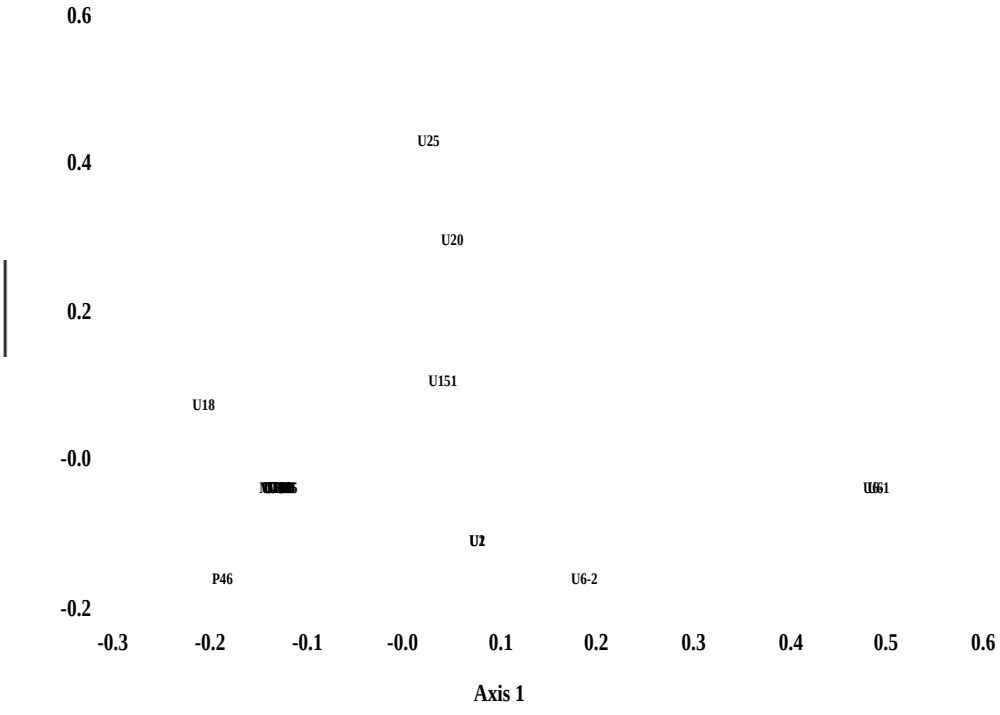




300 units; 39%, 15%, 13%

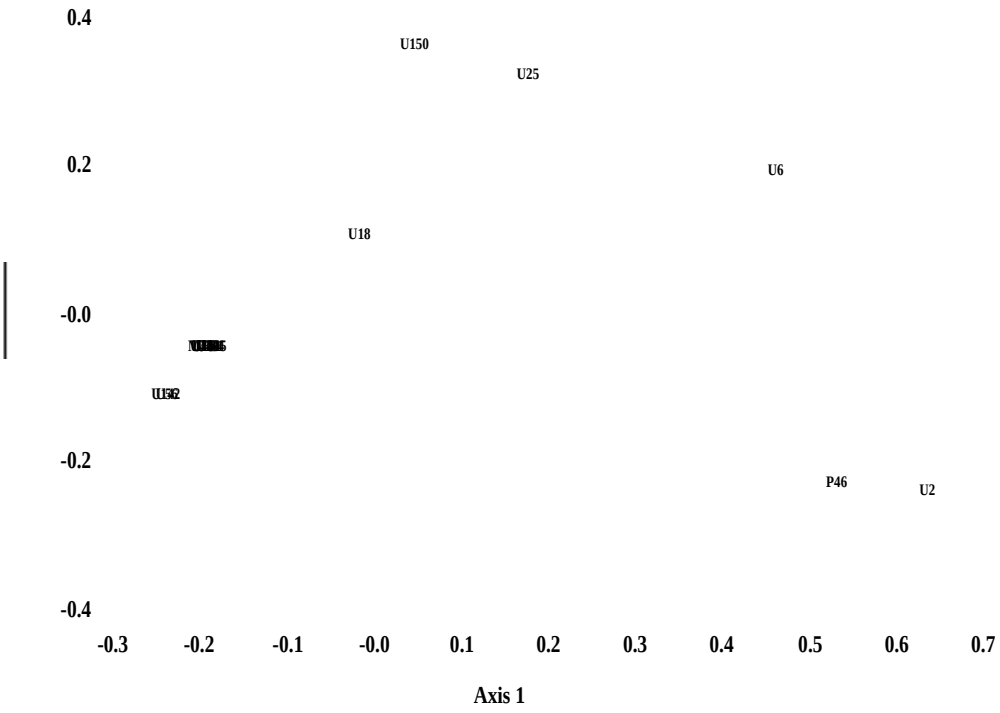
U44-0

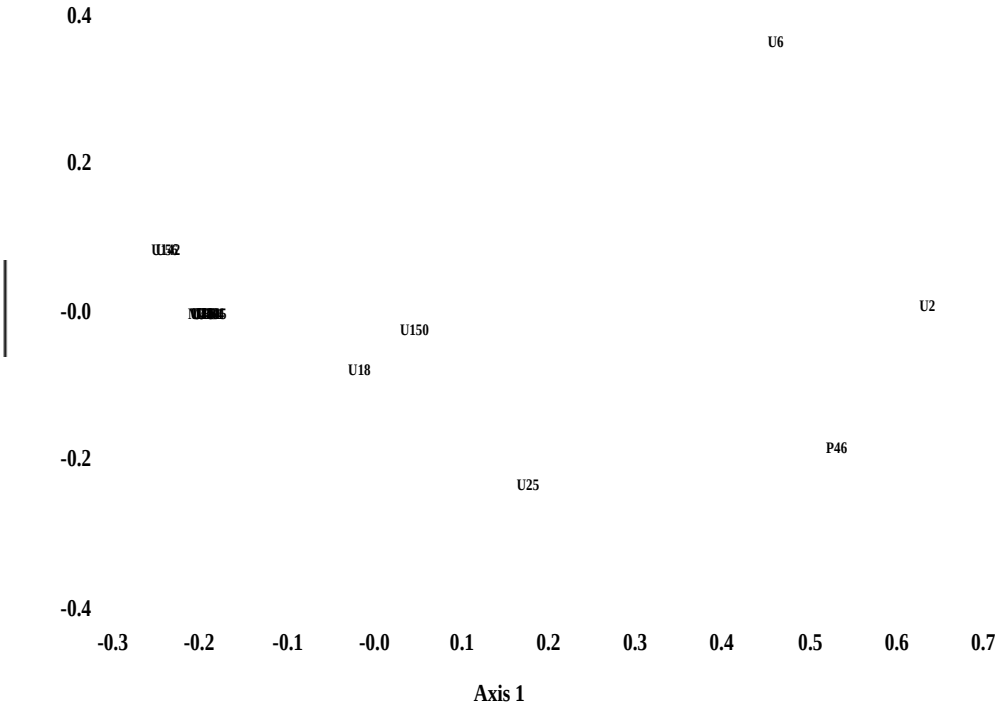




10 units; 33%, 30%, 17%

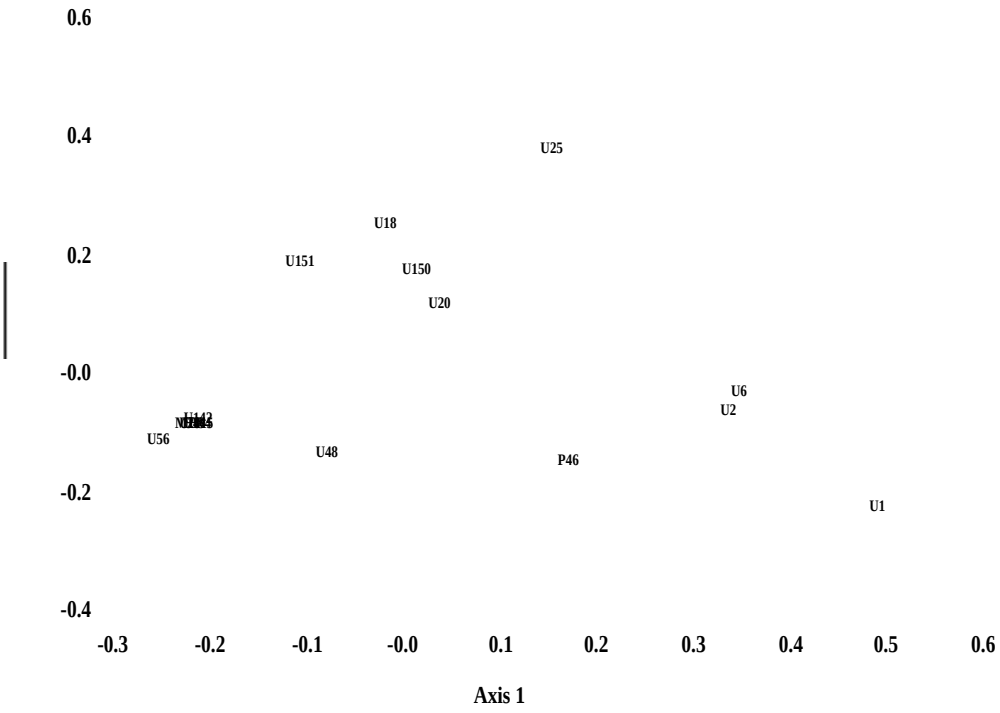
U44-1

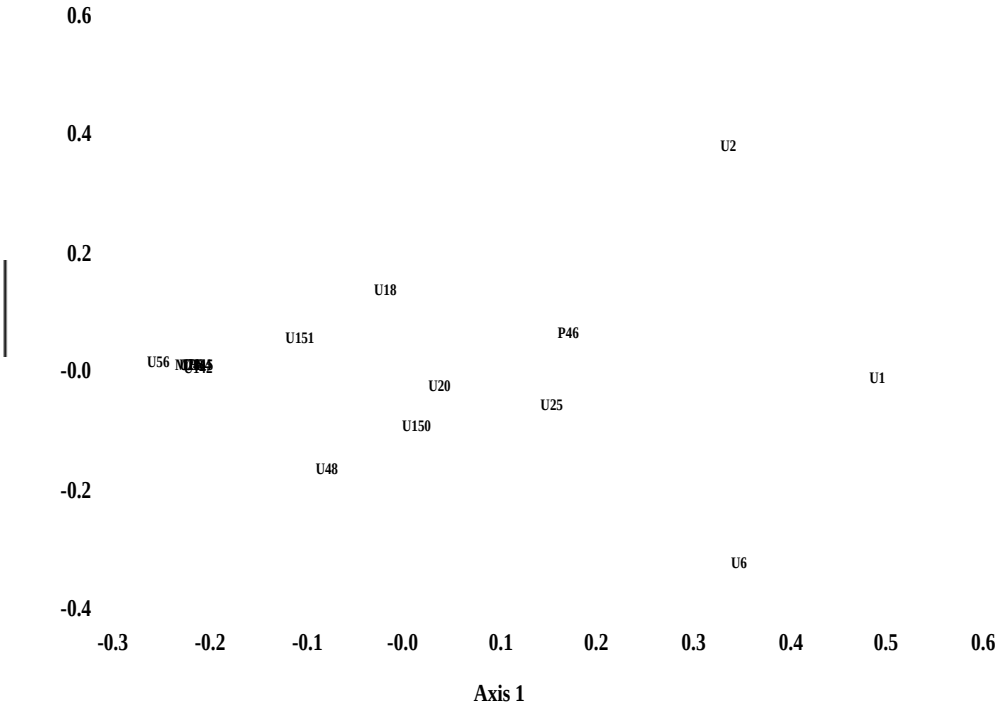




8 units; 57%, 19%, 11%

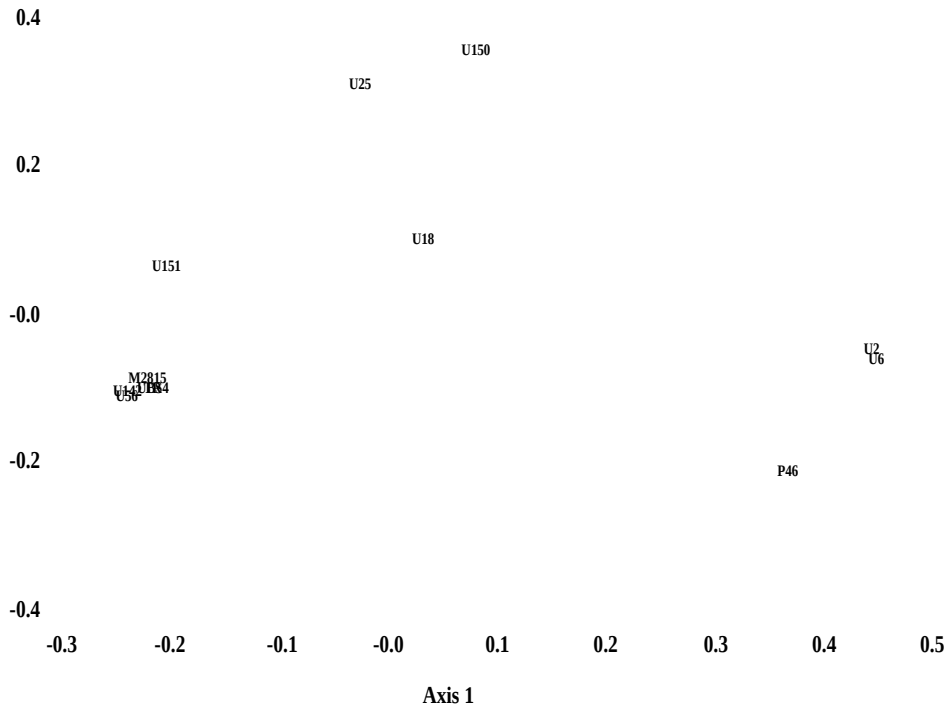
U48 (450?)

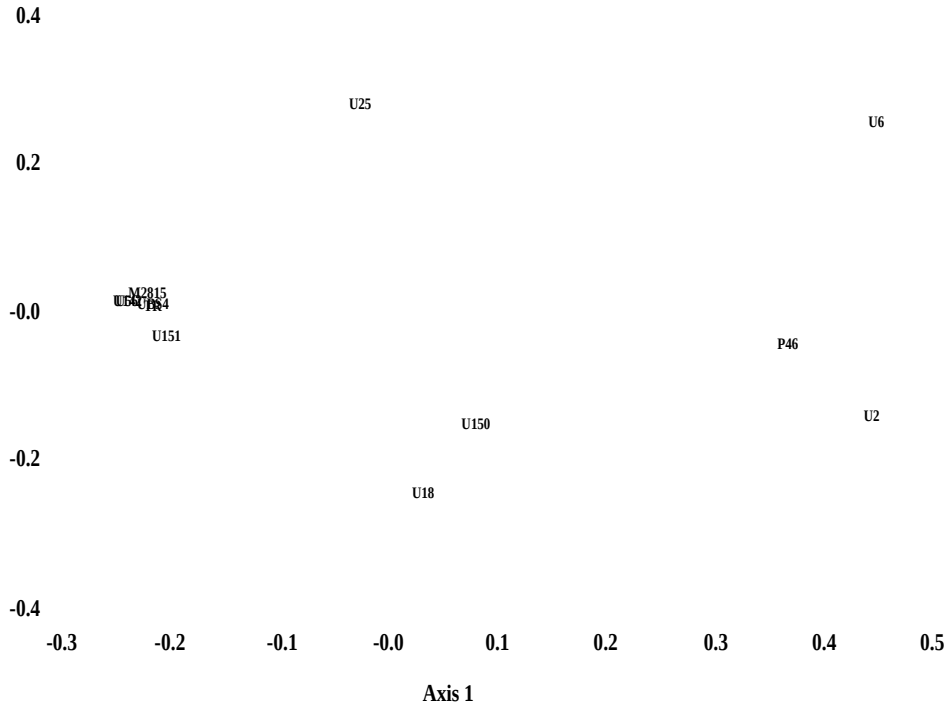




29 units; 33%, 17%, 12%

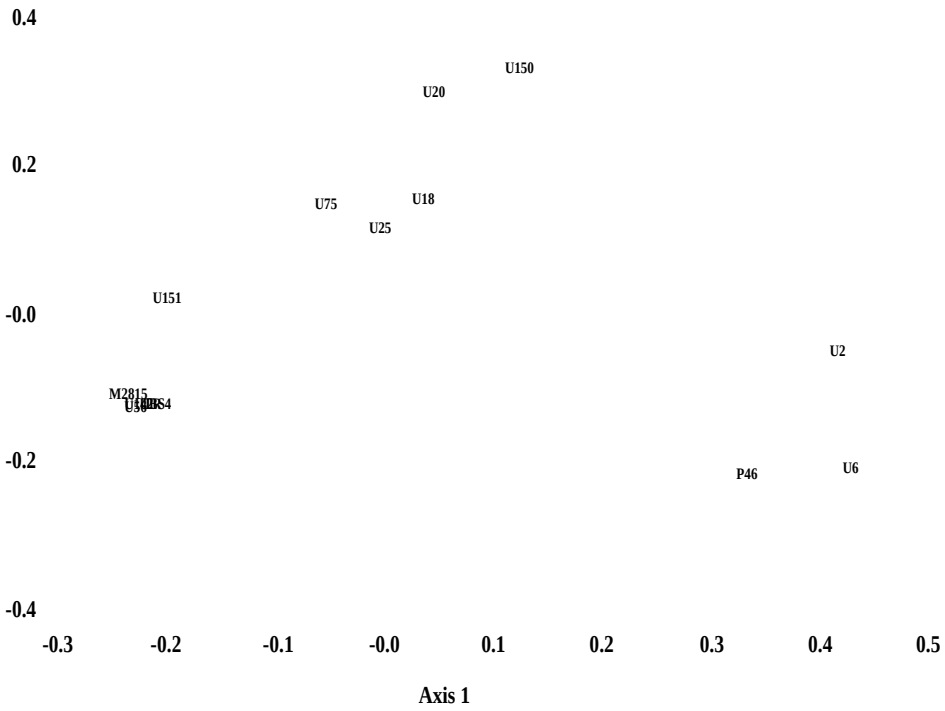
U56 (950?)

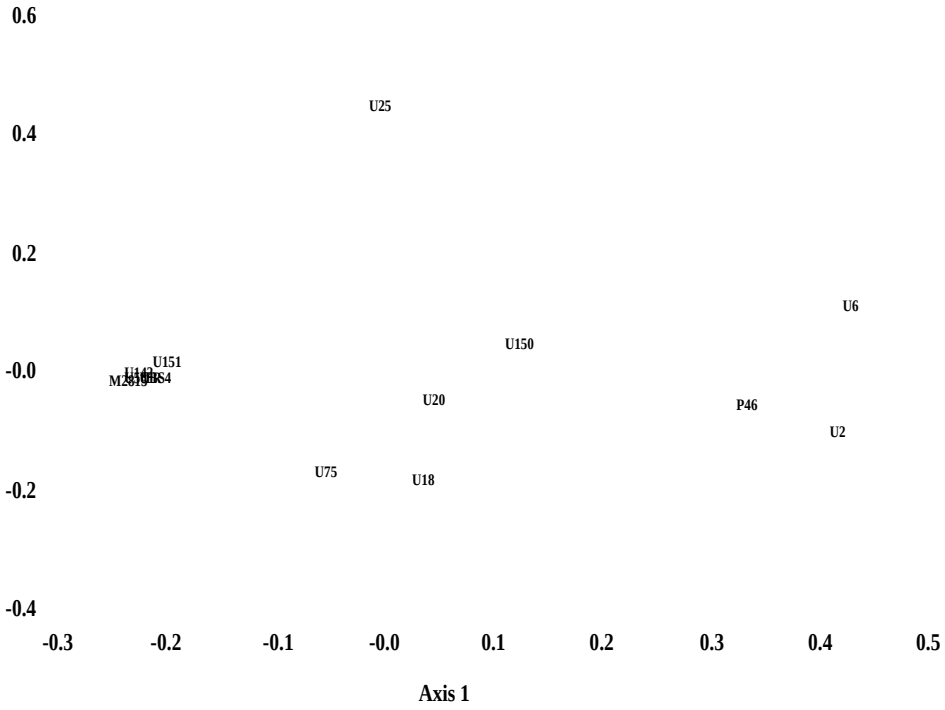




317 units; 40%, 16%, 12%

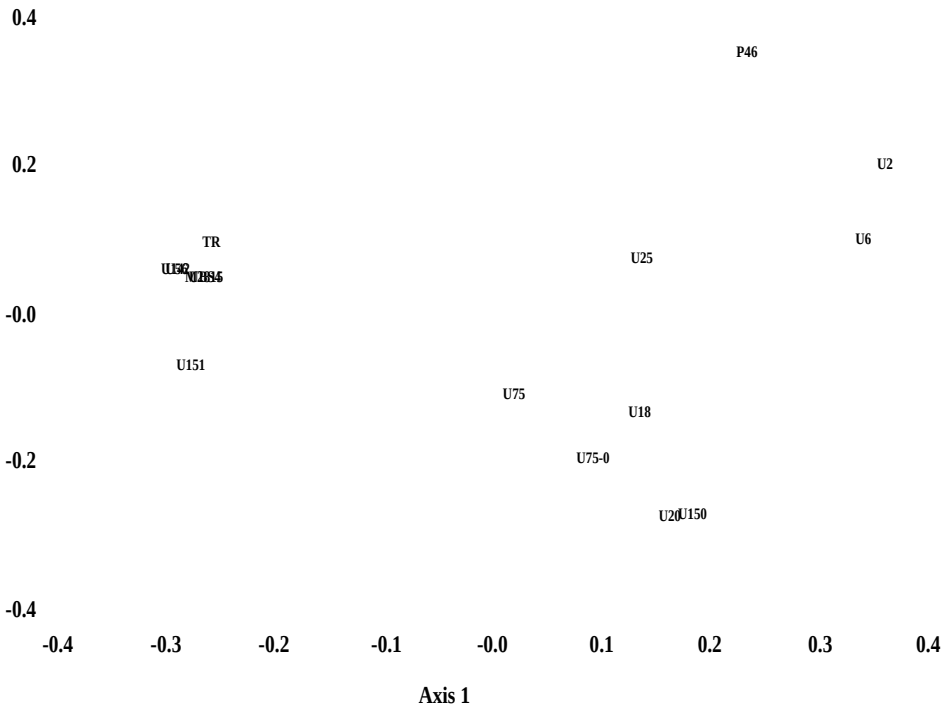
U75 (950?)

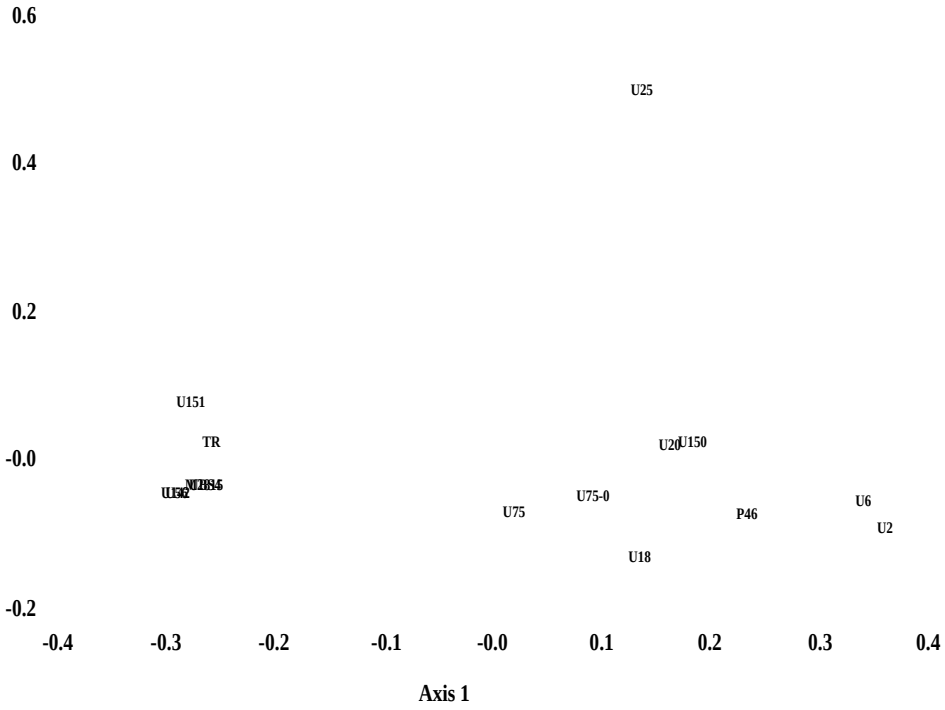




243 units; 33%, 18%, 13%

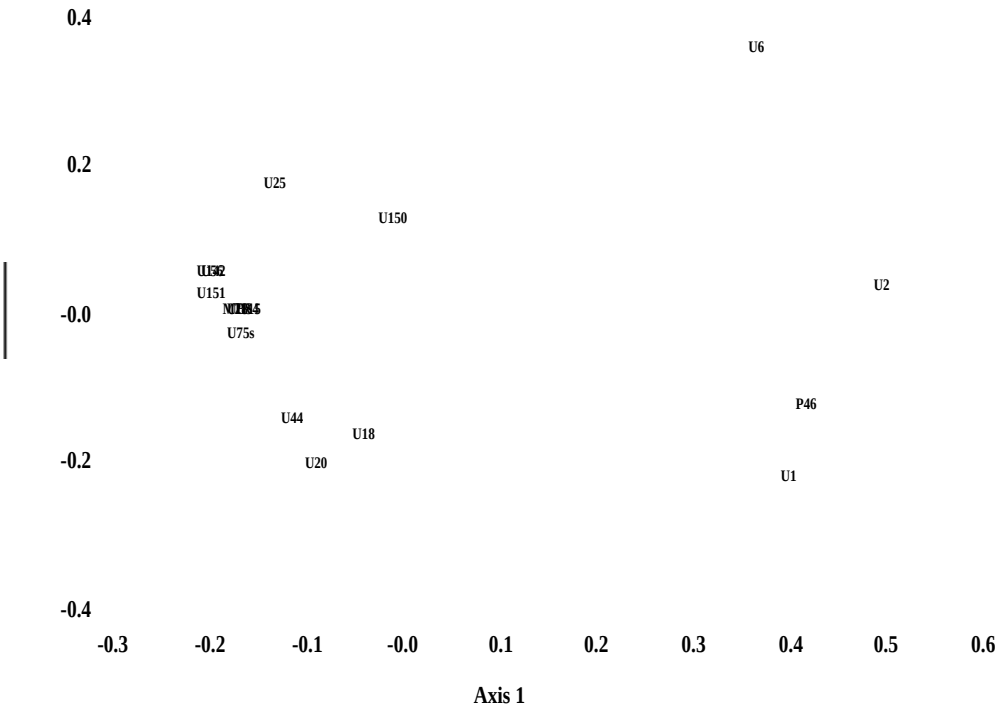
U75-0

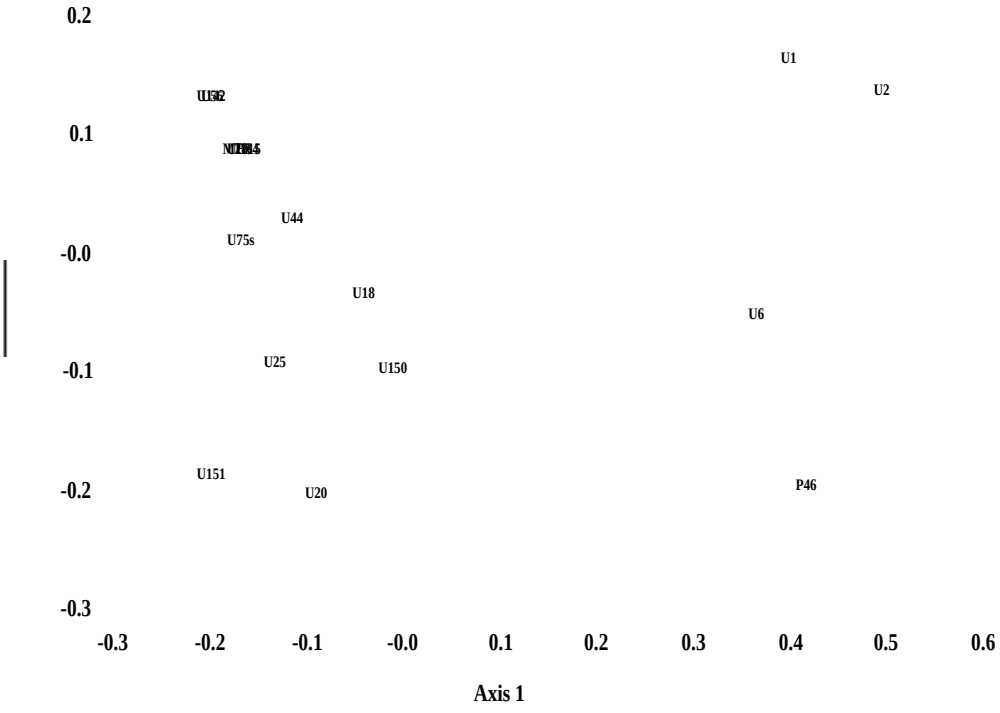




26 units; 36%, 18%, 13%

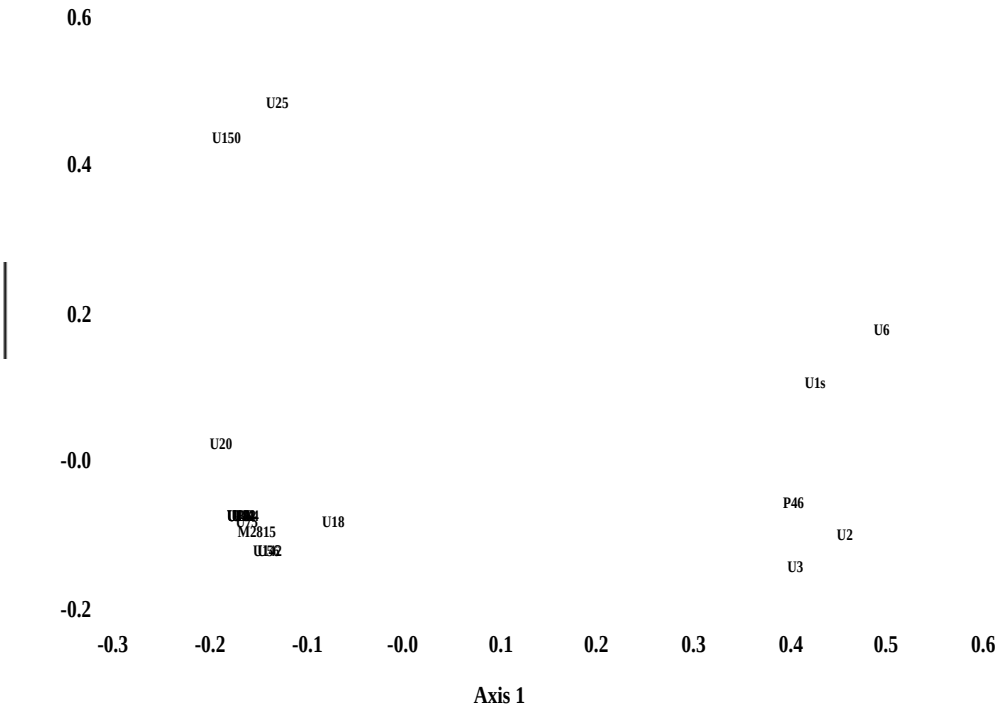
U75s (950?)

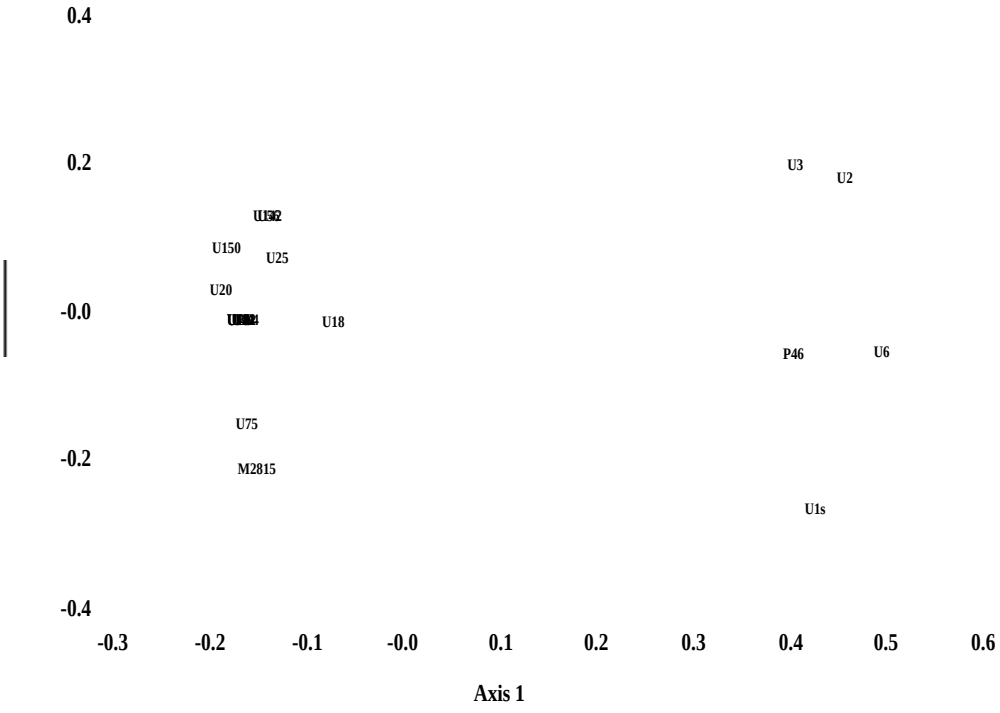




31 units; 40%, 14%, 10%

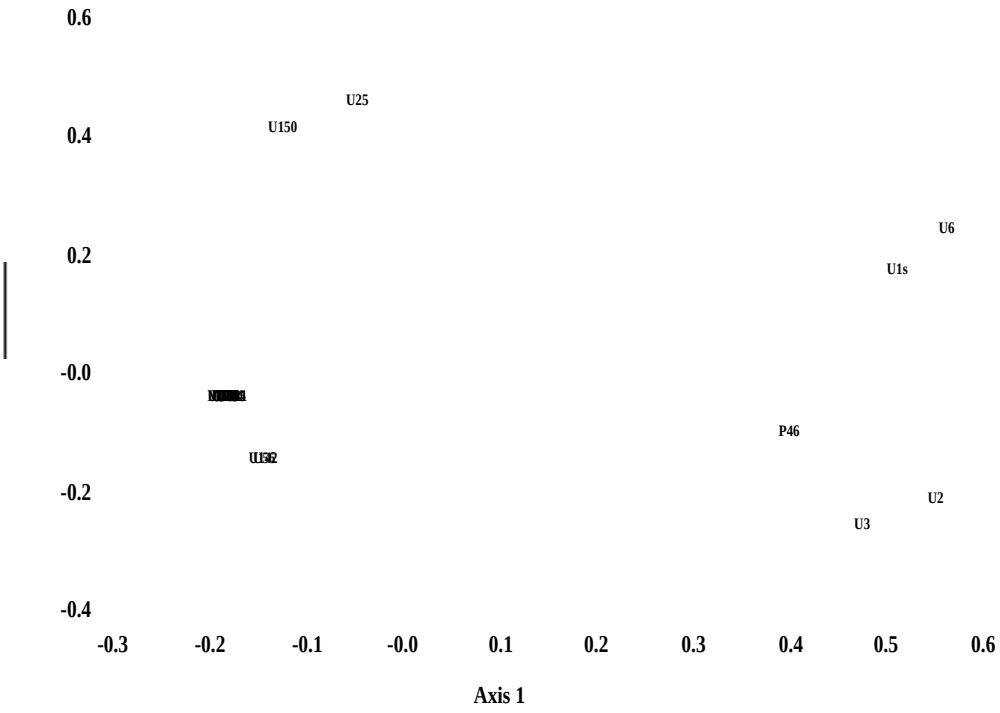
U122 (850?)

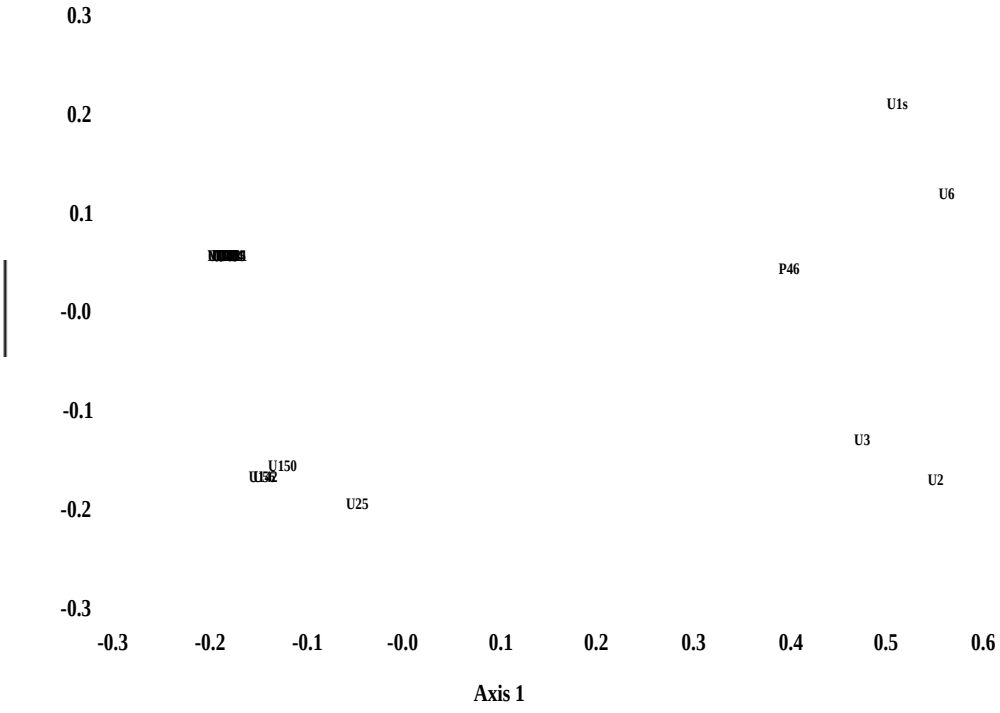




17 units; 47%, 21%, 9%

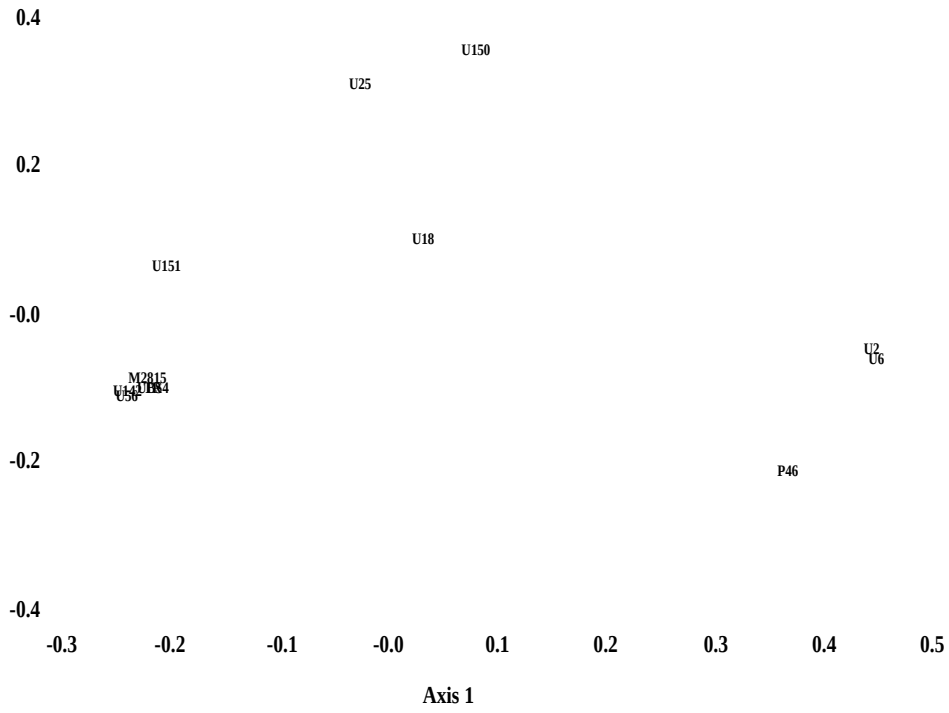
U122-1

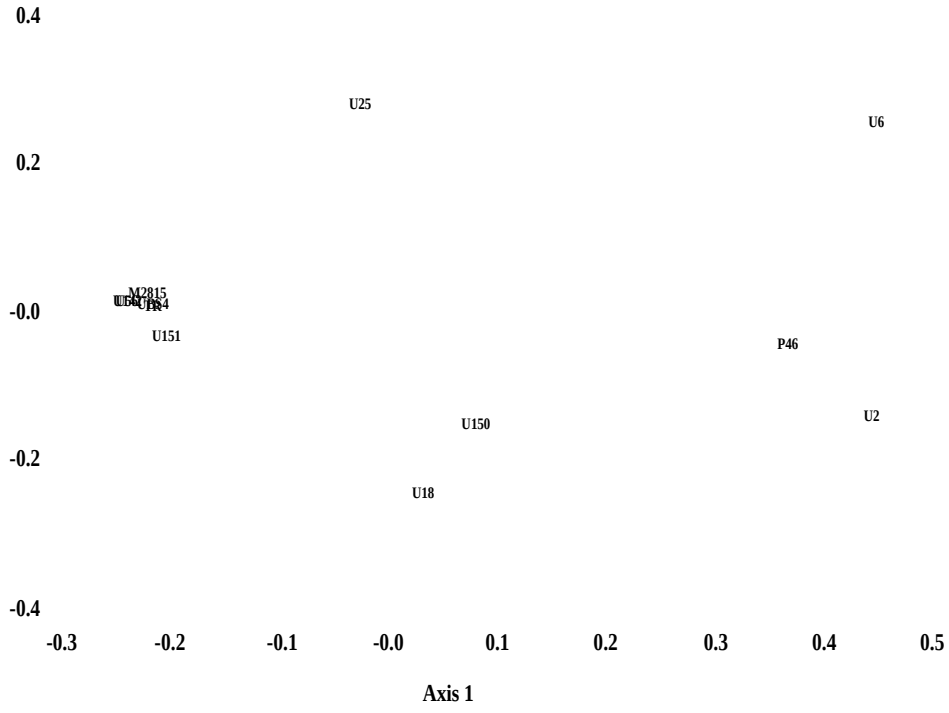




11 units; 56%, 22%, 9%

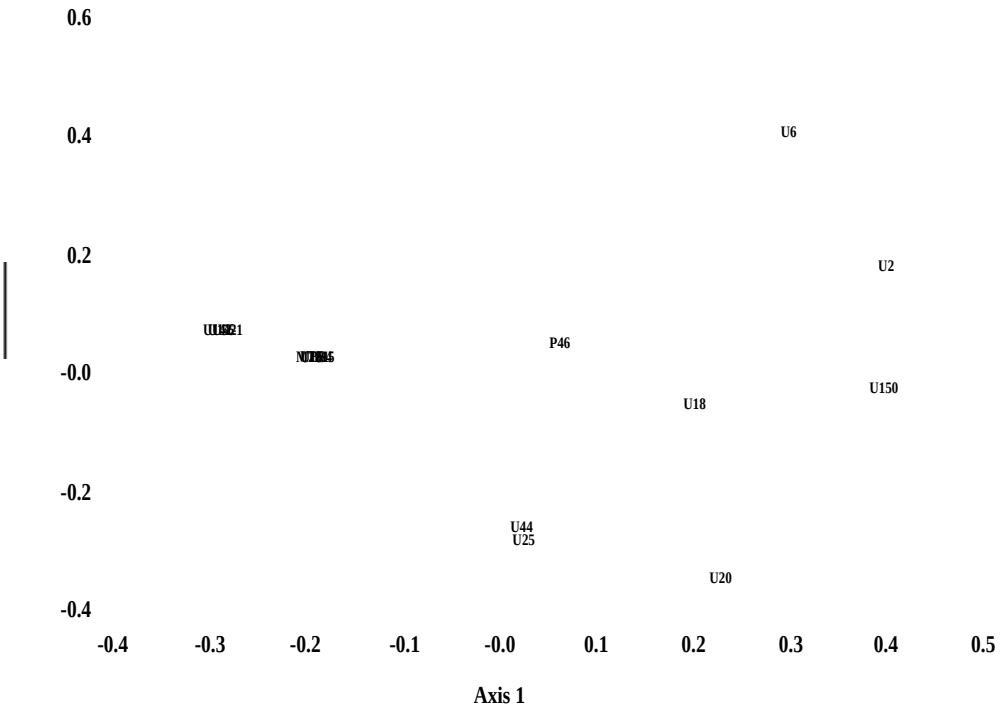
U142 (950?)

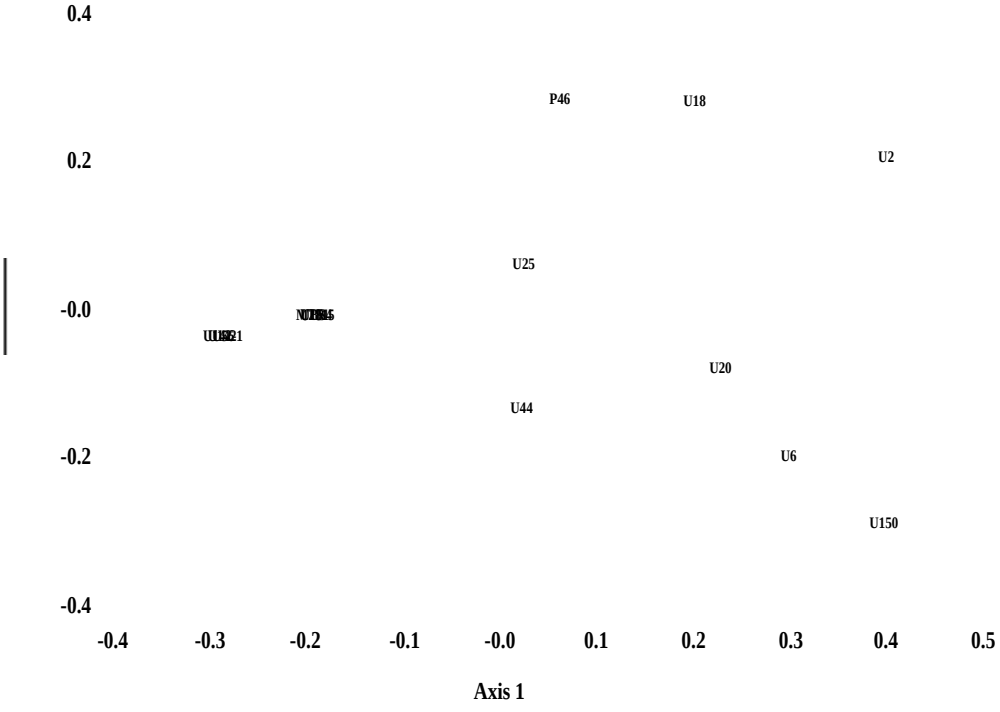




317 units; 40%, 16%, 12%

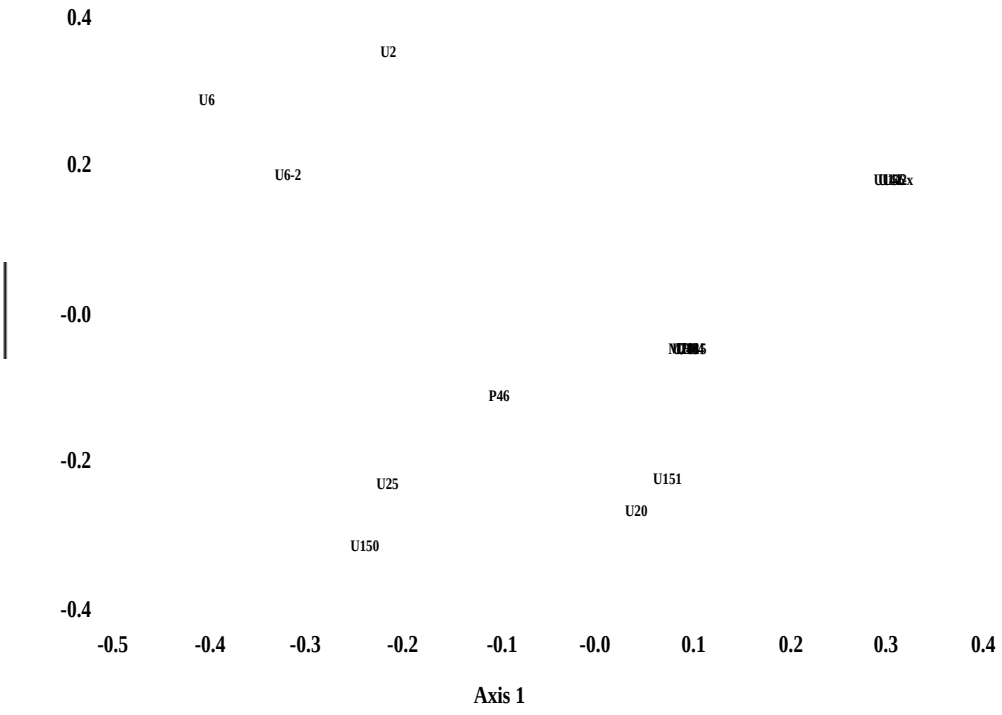
U142-1

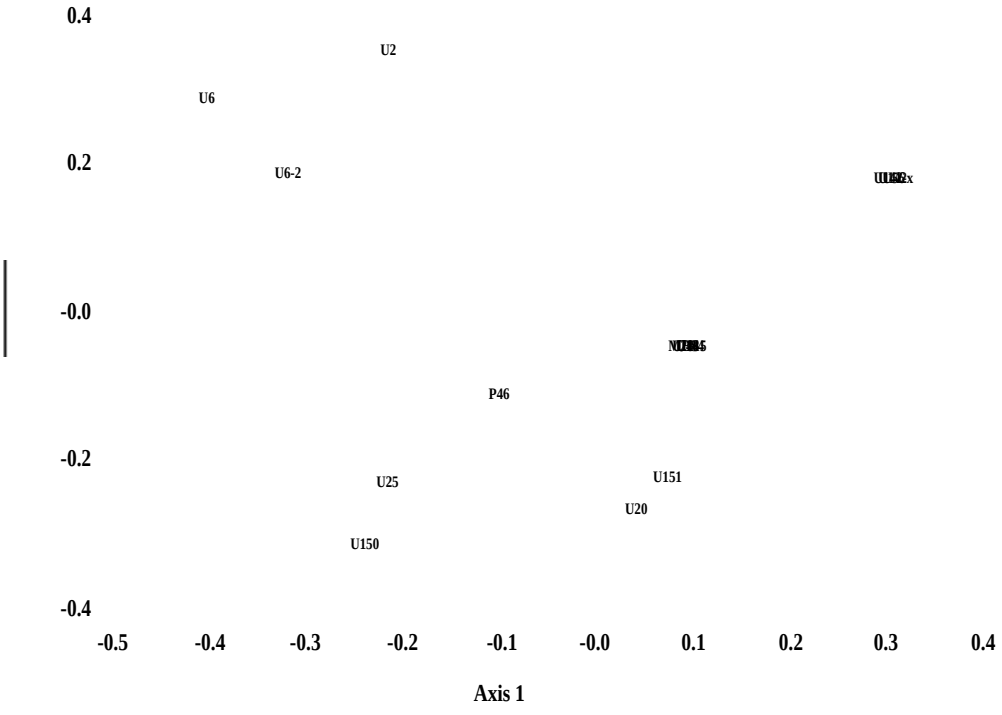




10 units; 37%, 20%, 15%

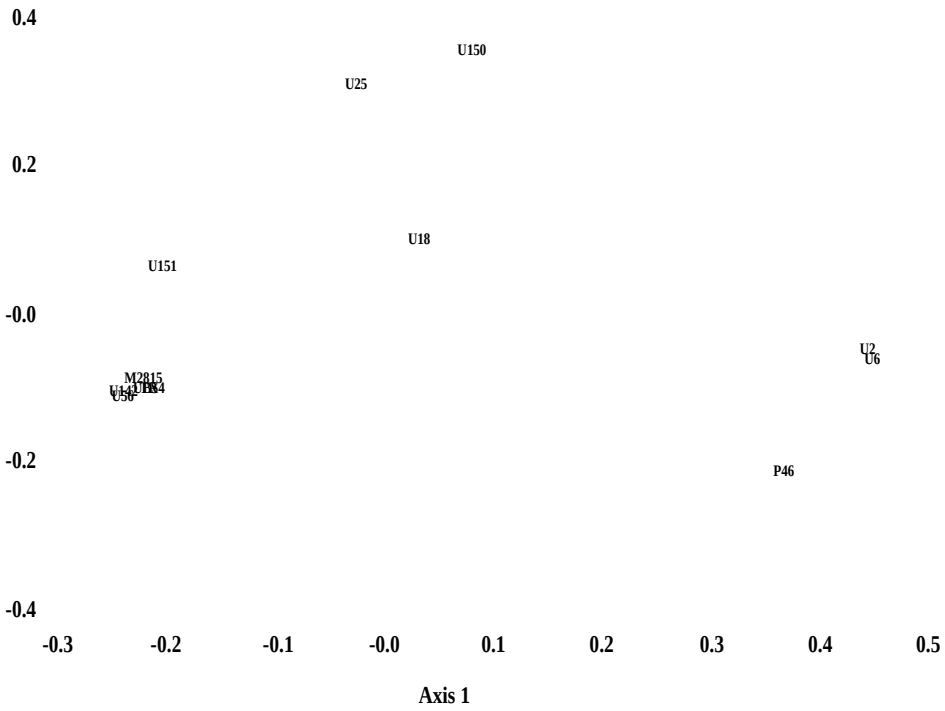
U142-x

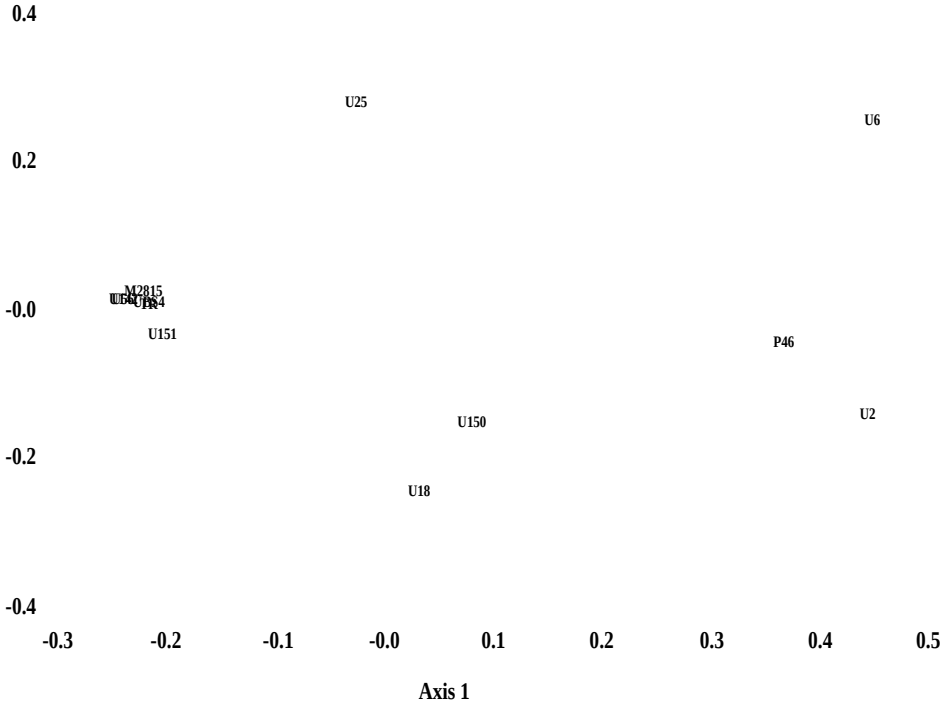




10 units; 32%, 27%, 16%

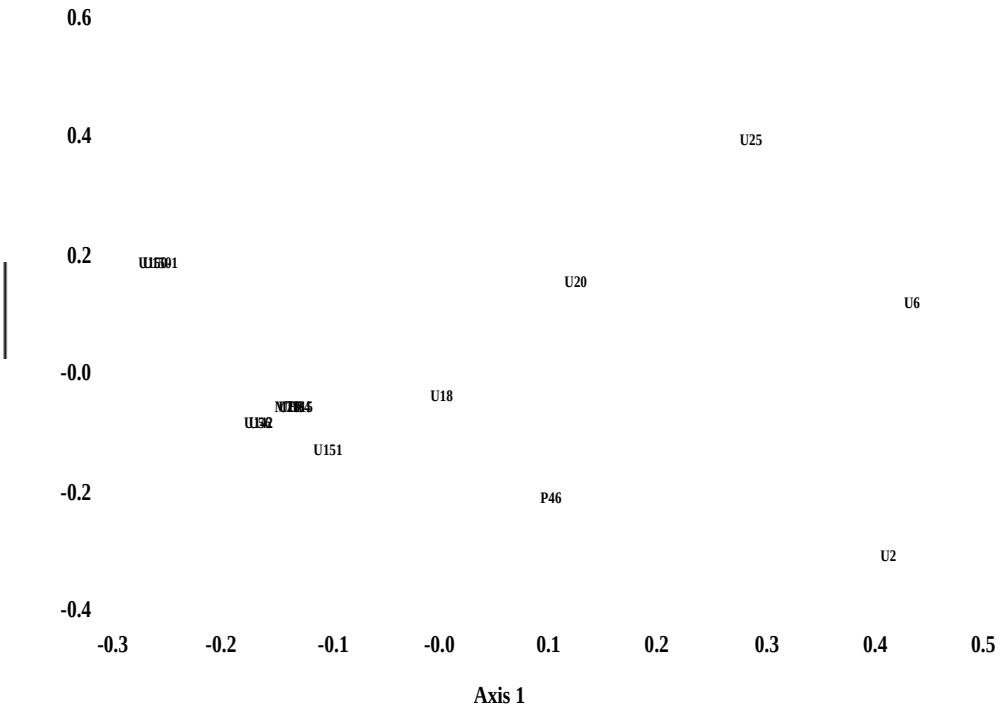
U150 (850?)

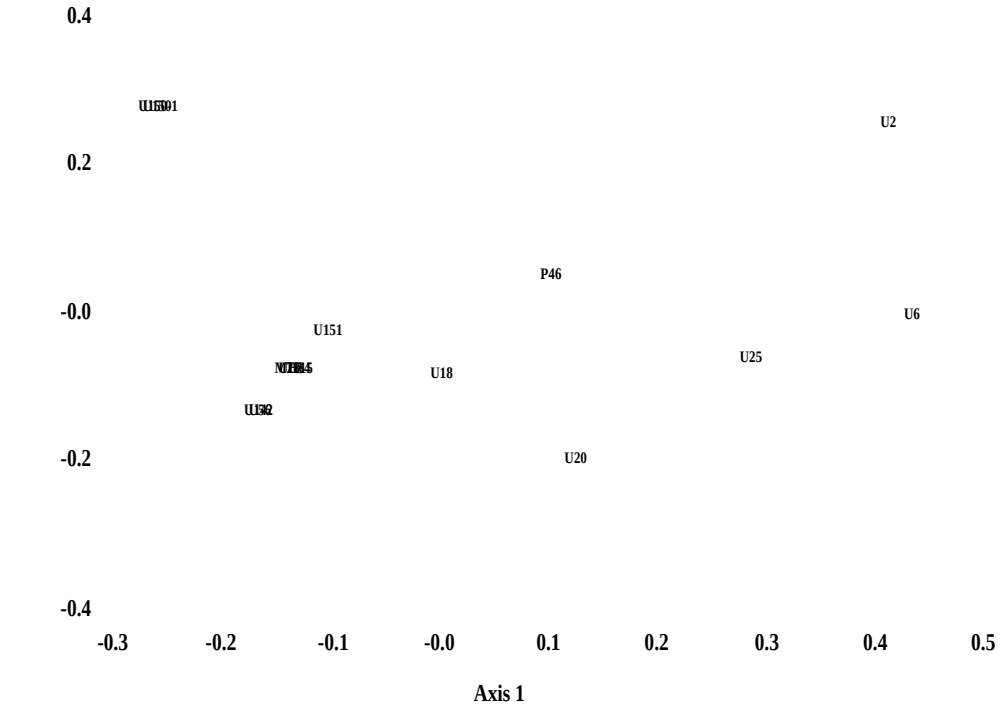




317 units; 40%, 16%, 12%

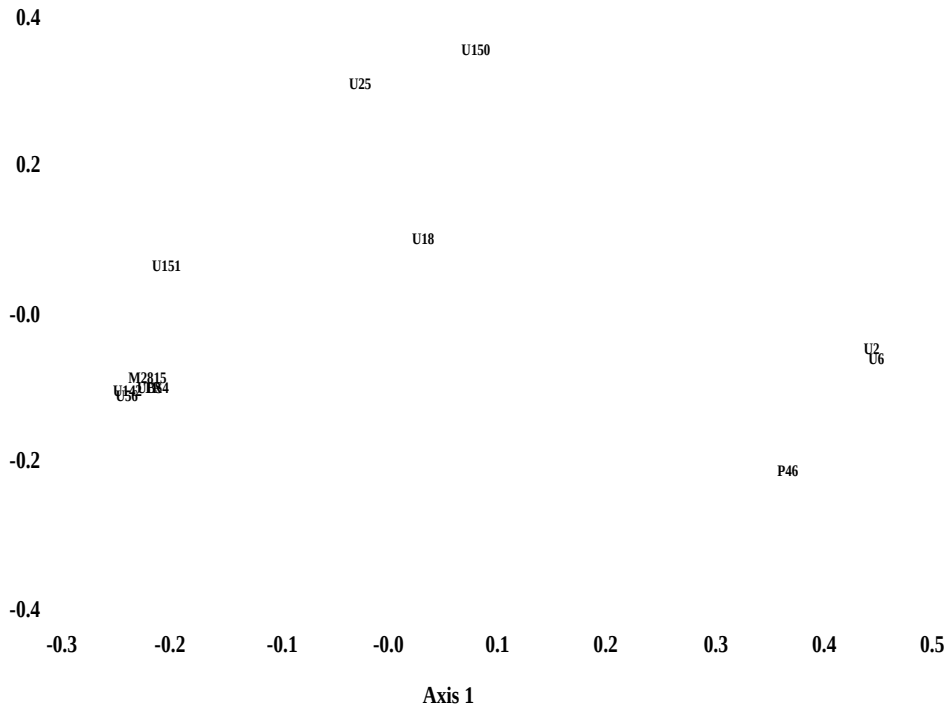
U150-1

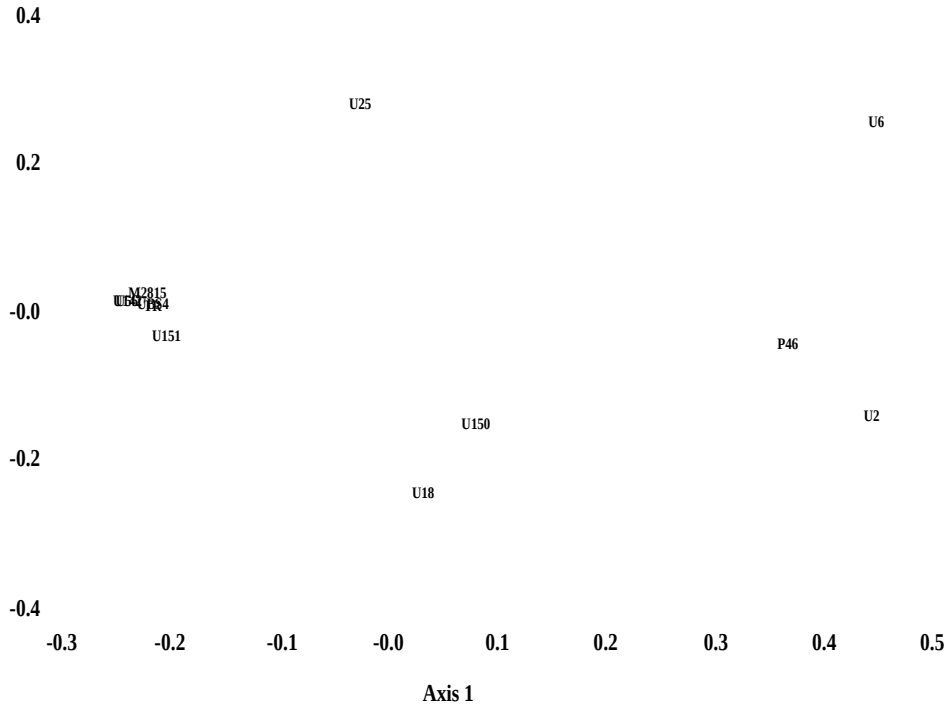




14 units; 37%, 22%, 16%

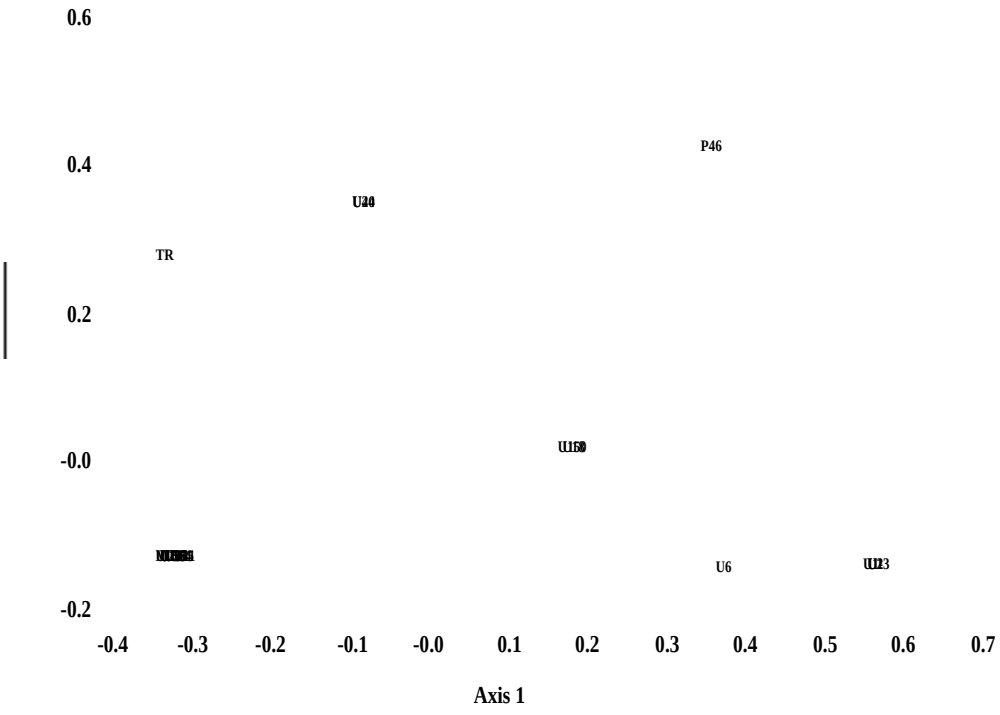
U151 (850?)

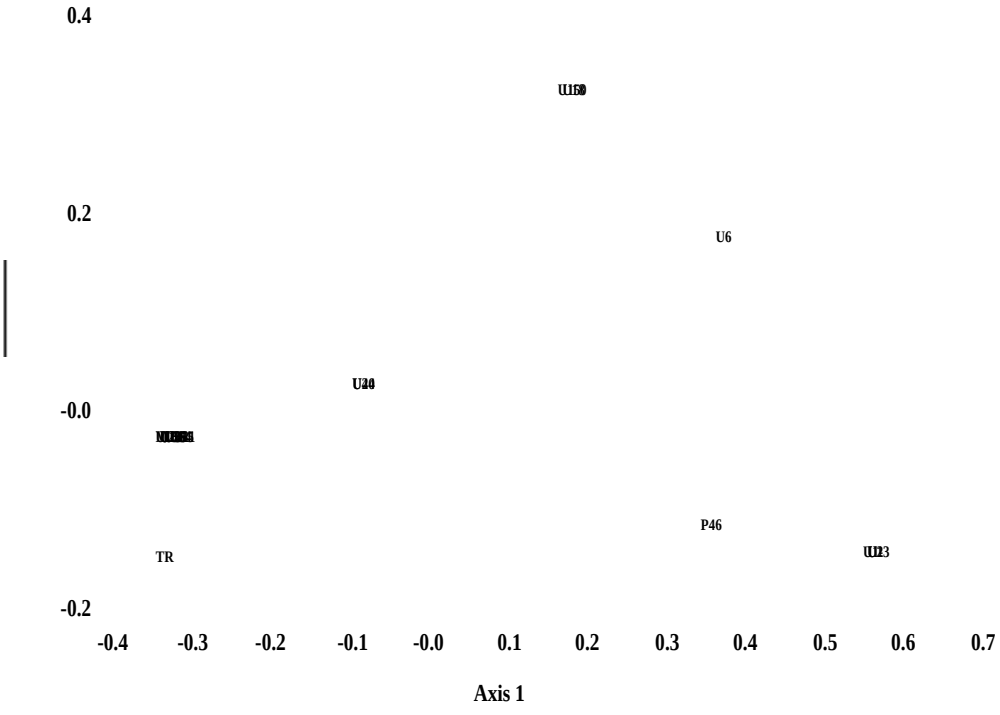




317 units; 40%, 16%, 12%

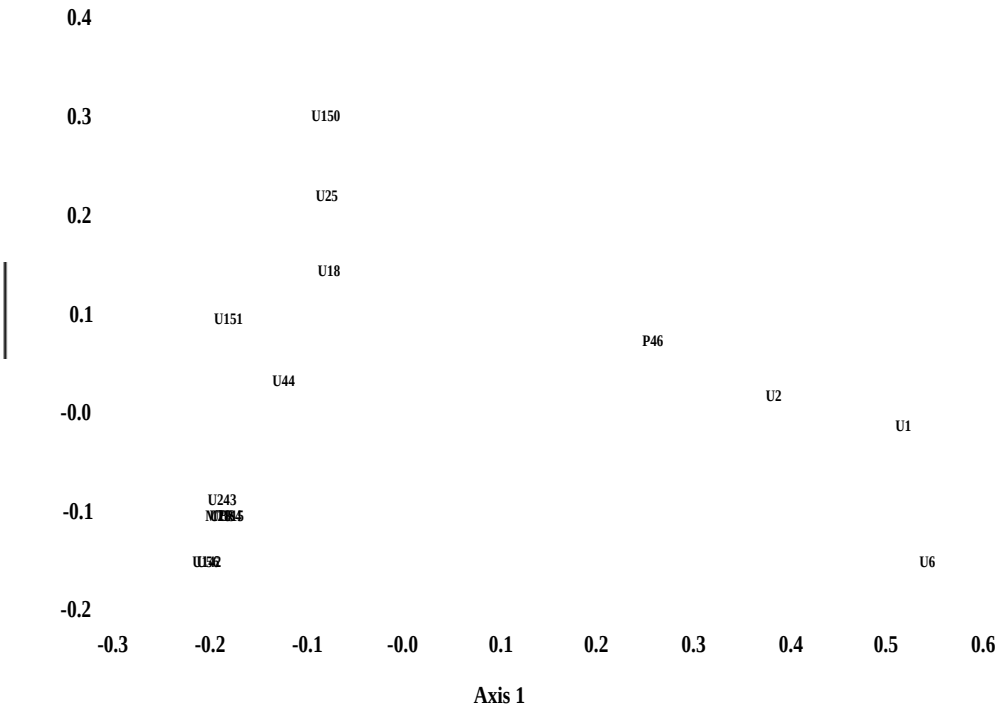
U151-1

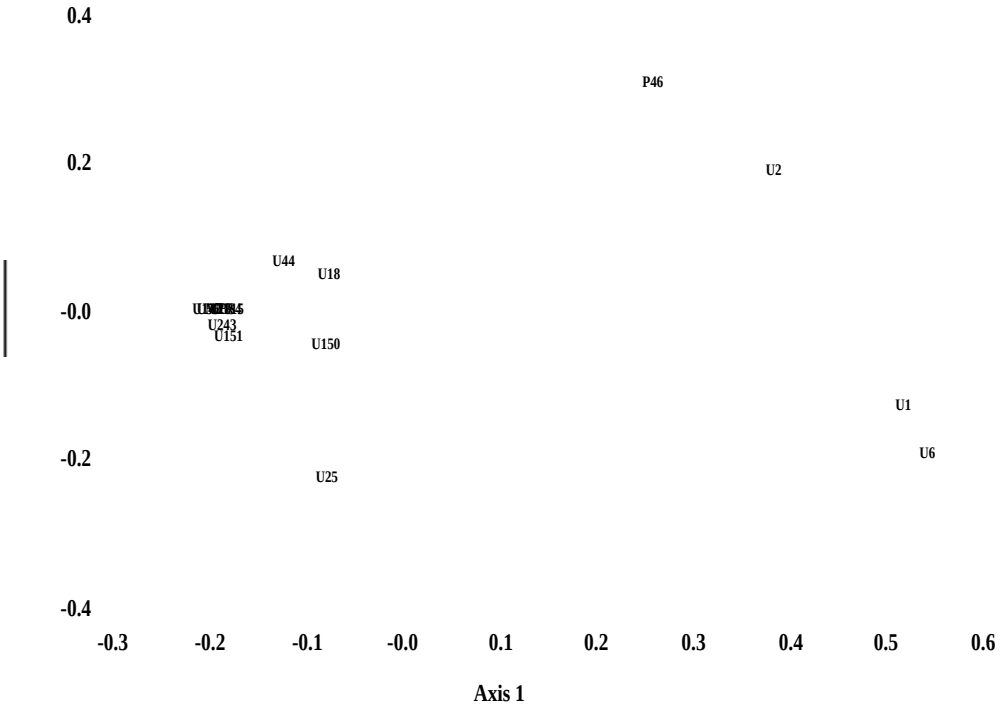




5 units; 61%, 20%, 10%

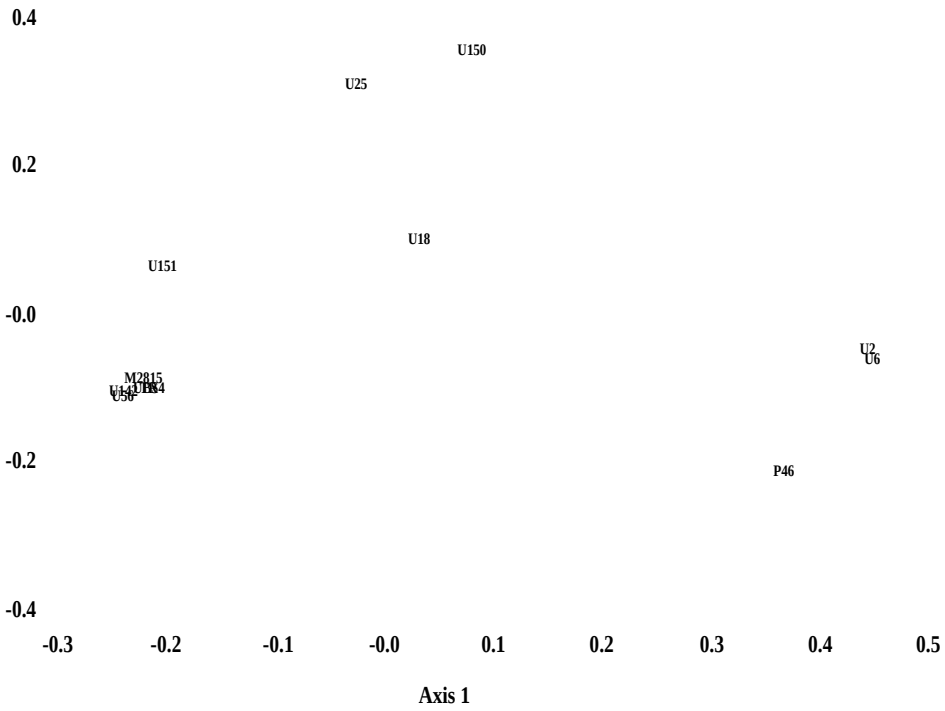
U243 (950?)

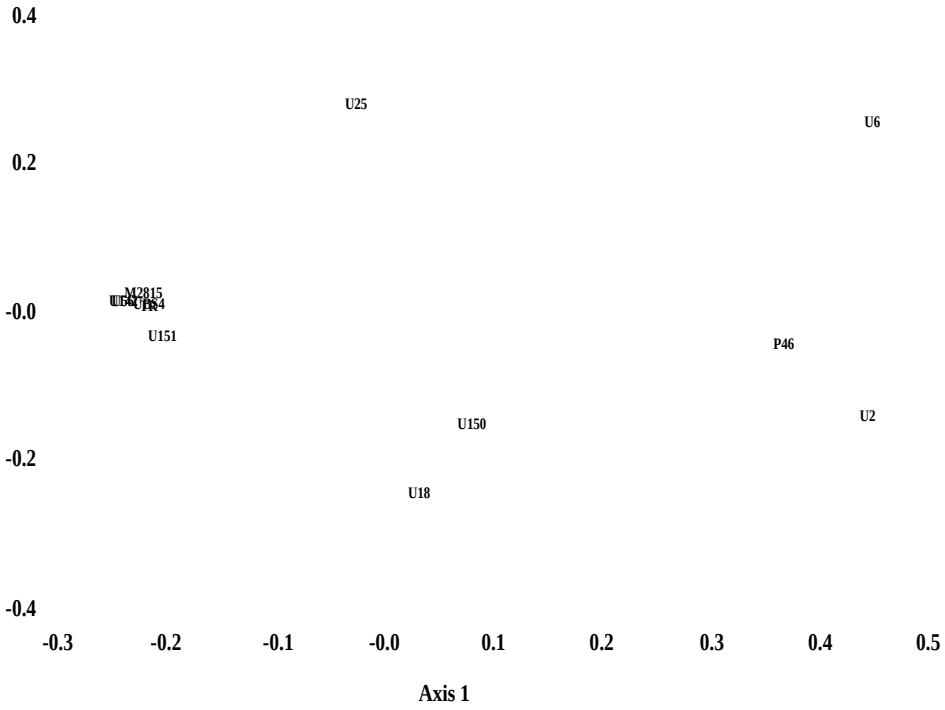




93 units; 45%, 12%, 10%

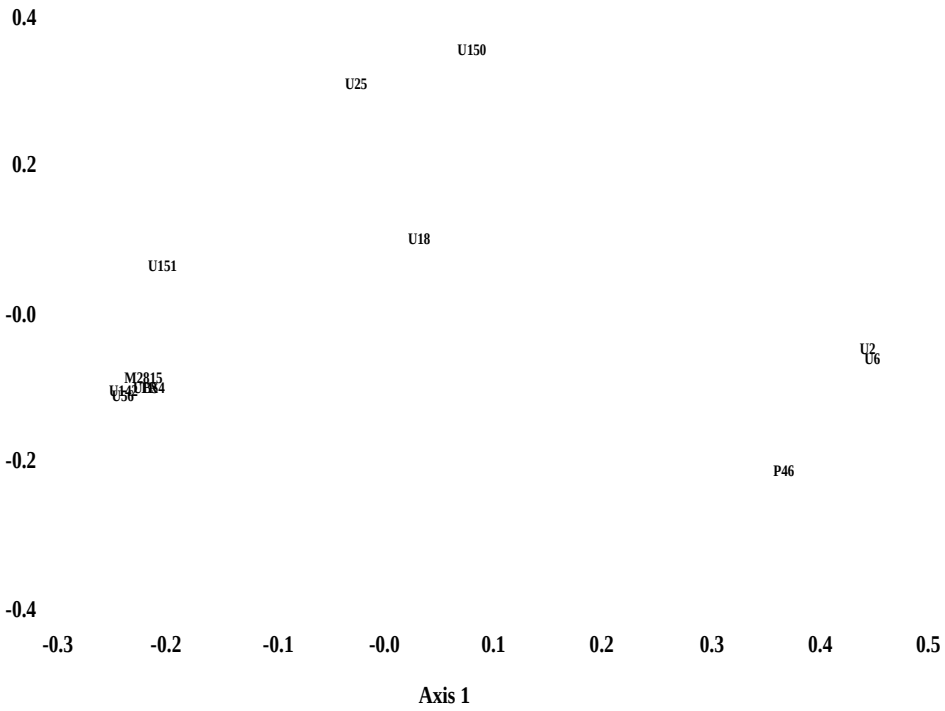
M2815 (1150?)

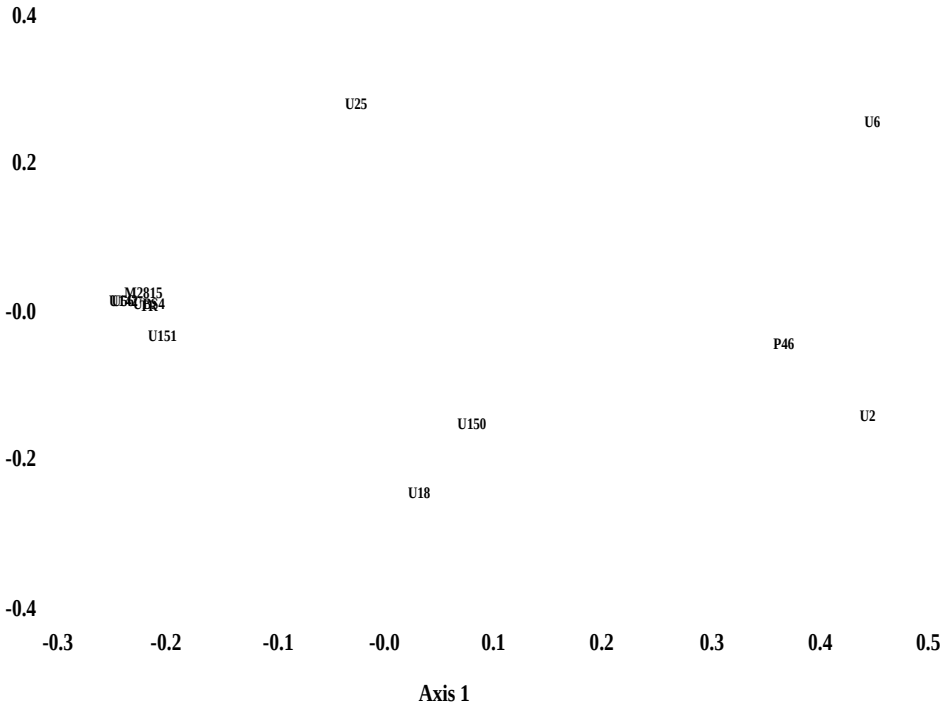




317 units; 40%, 16%, 12%

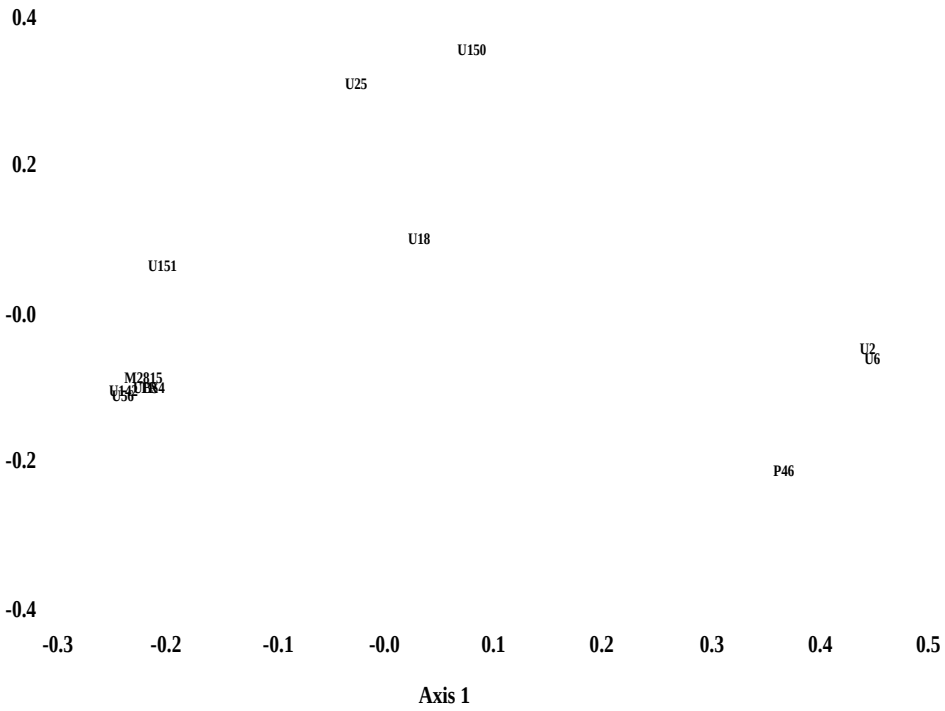
Textus Receptus

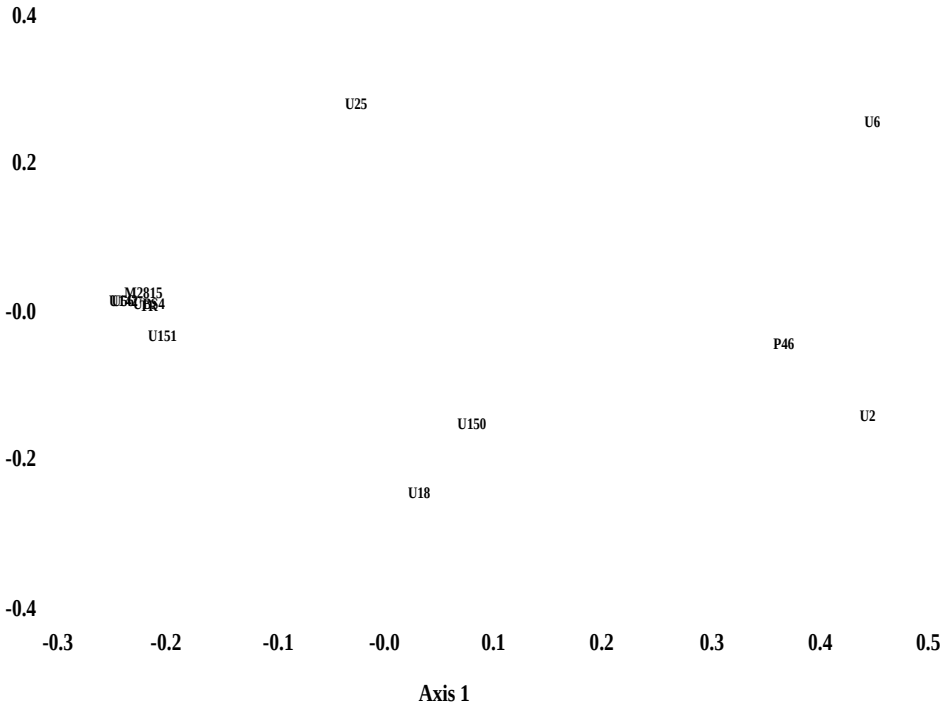




317 units; 40%, 16%, 12%

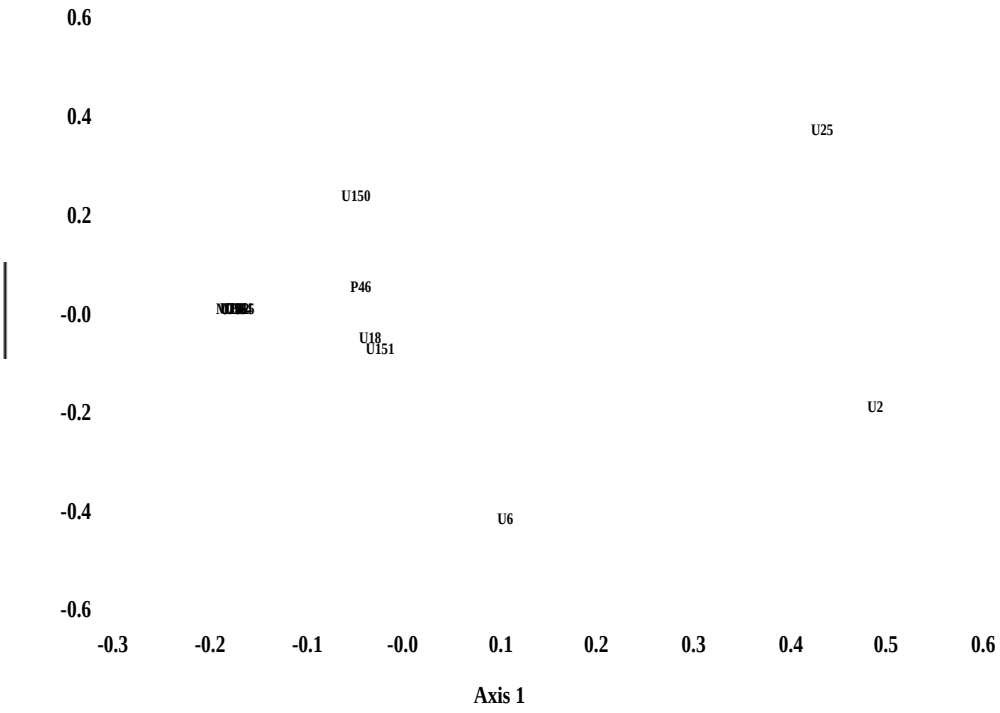
United Bible Societies 4th ed.

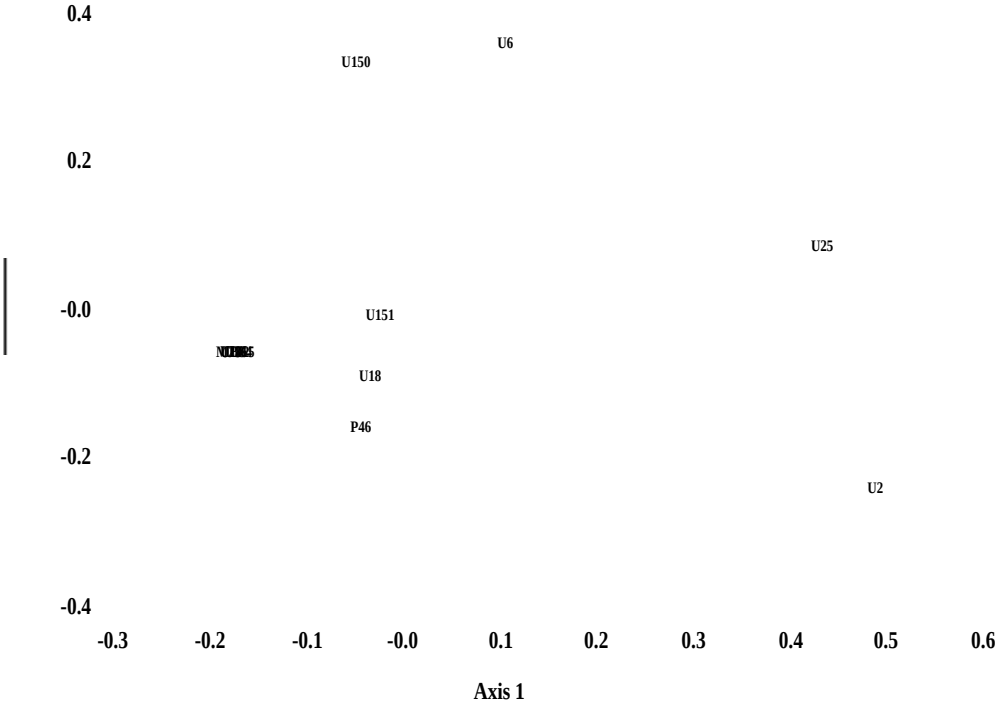




317 units; 40%, 16%, 12%

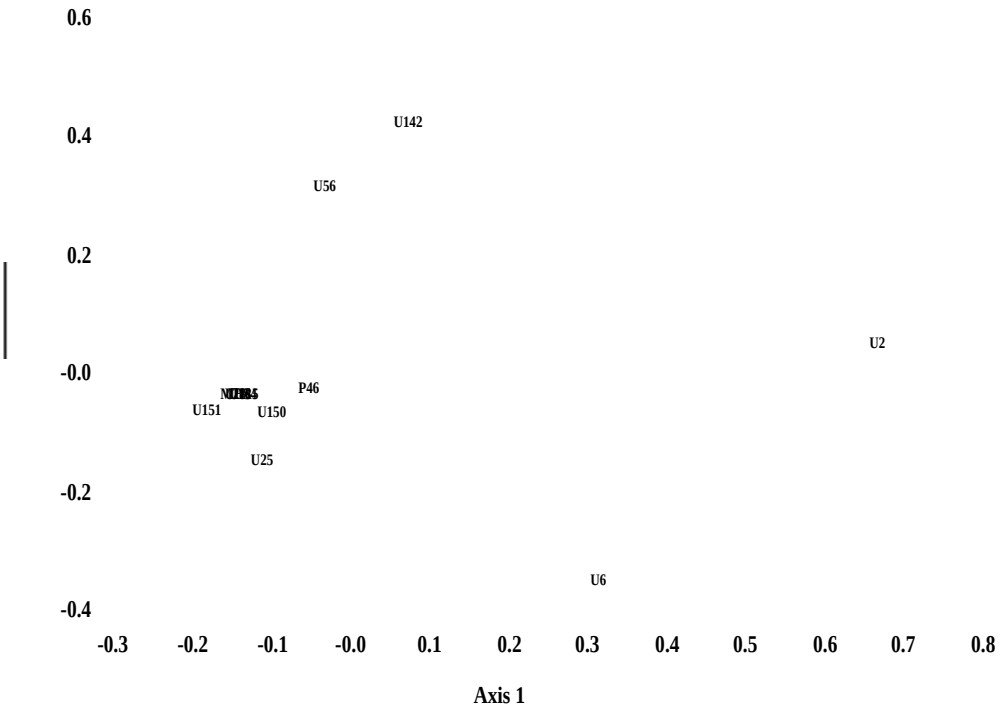
E -> AI

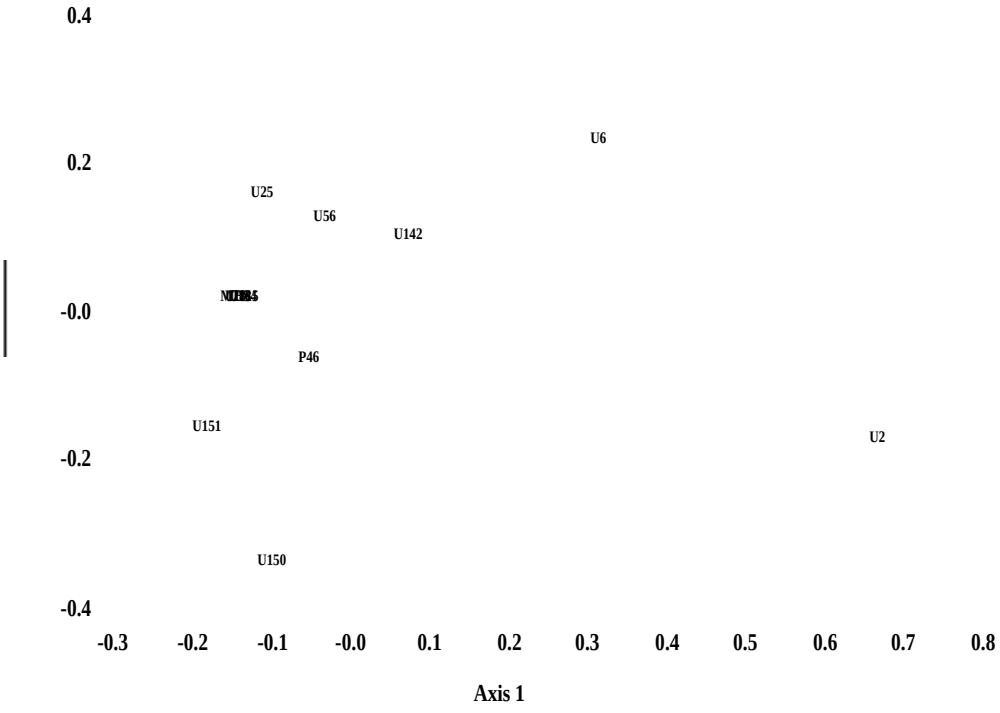




11 units; 33%, 23%, 20%

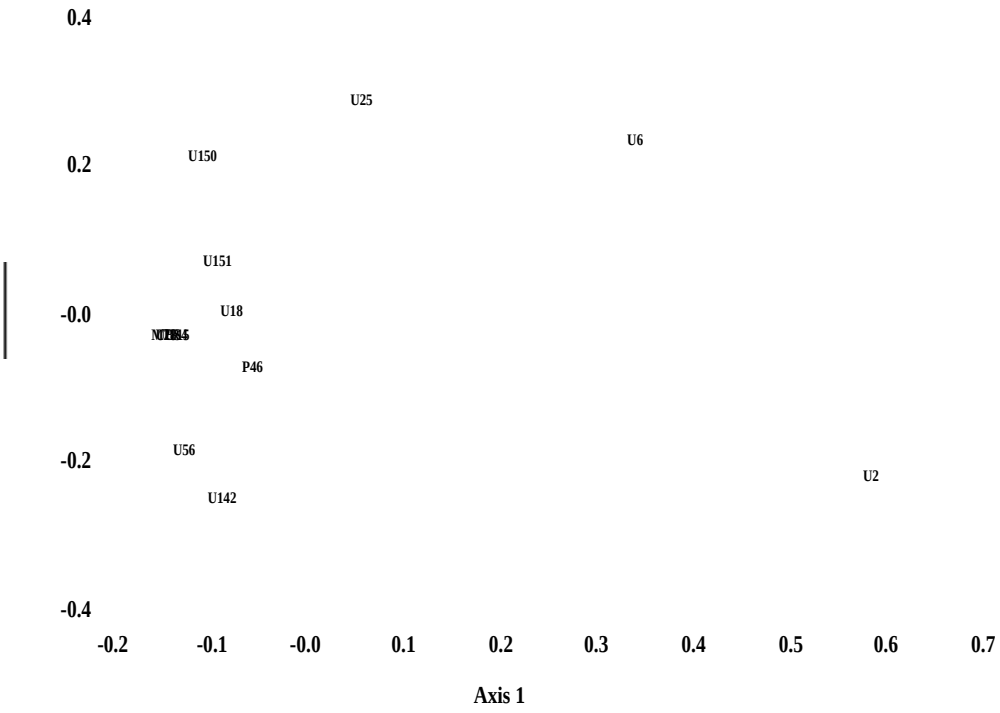
AI -> E

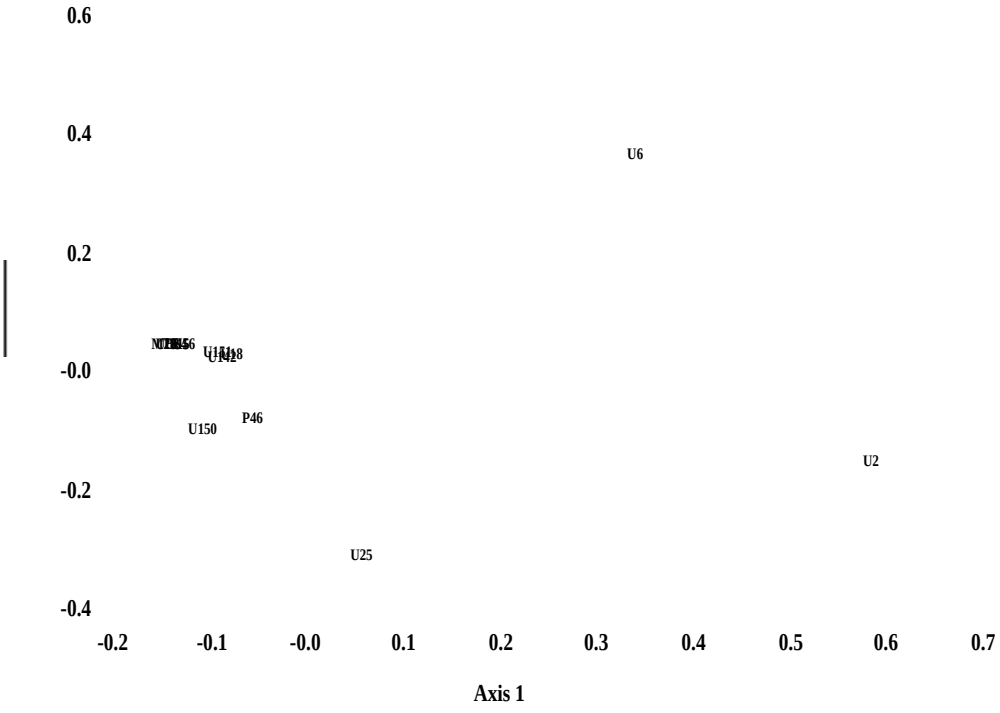




10 units; 40%, 25%, 16%

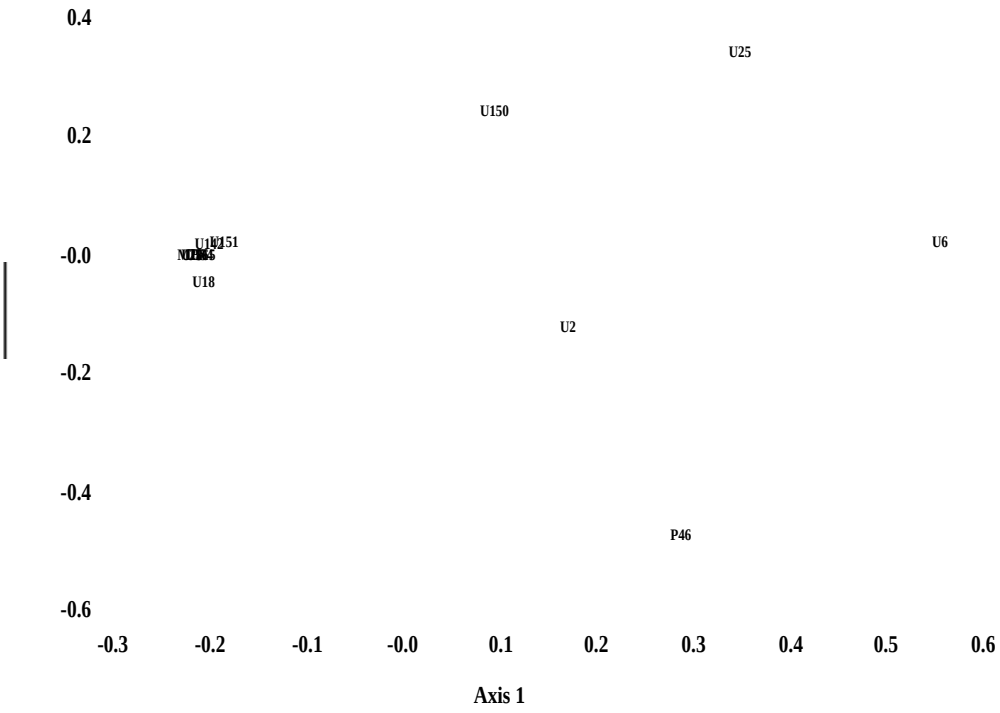
AI - E

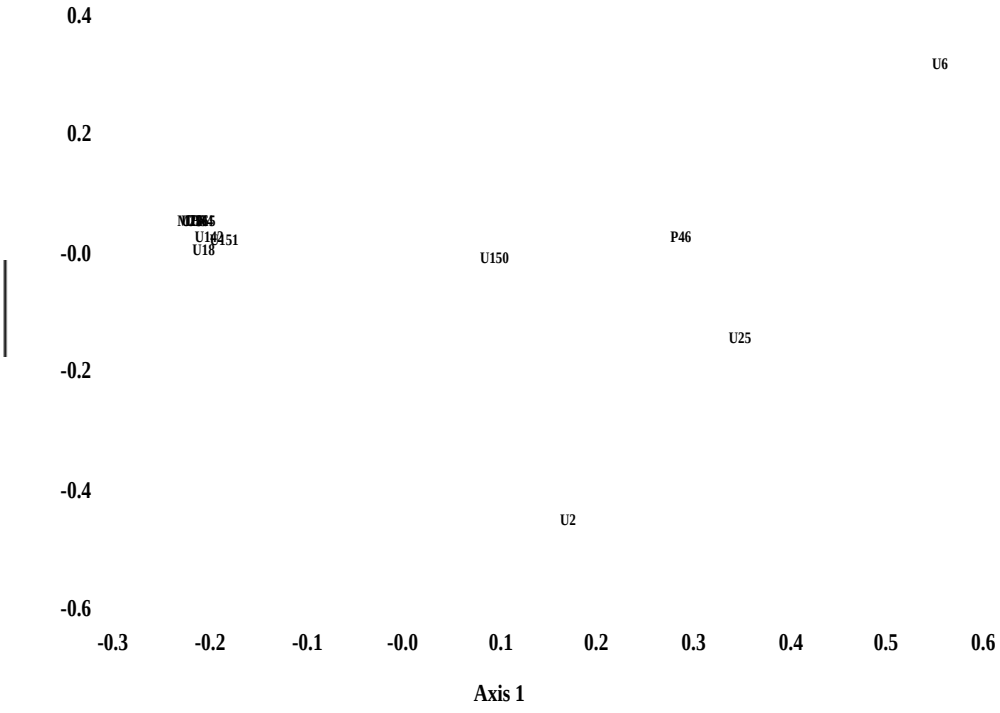




21 units; 32%, 19%, 16%

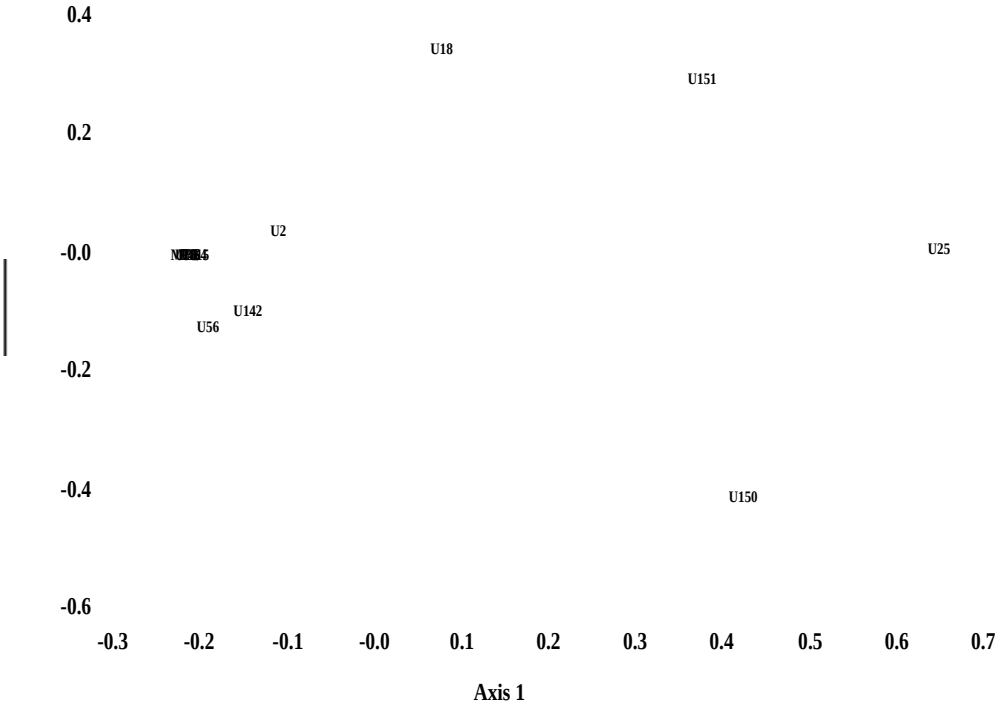
EI - I

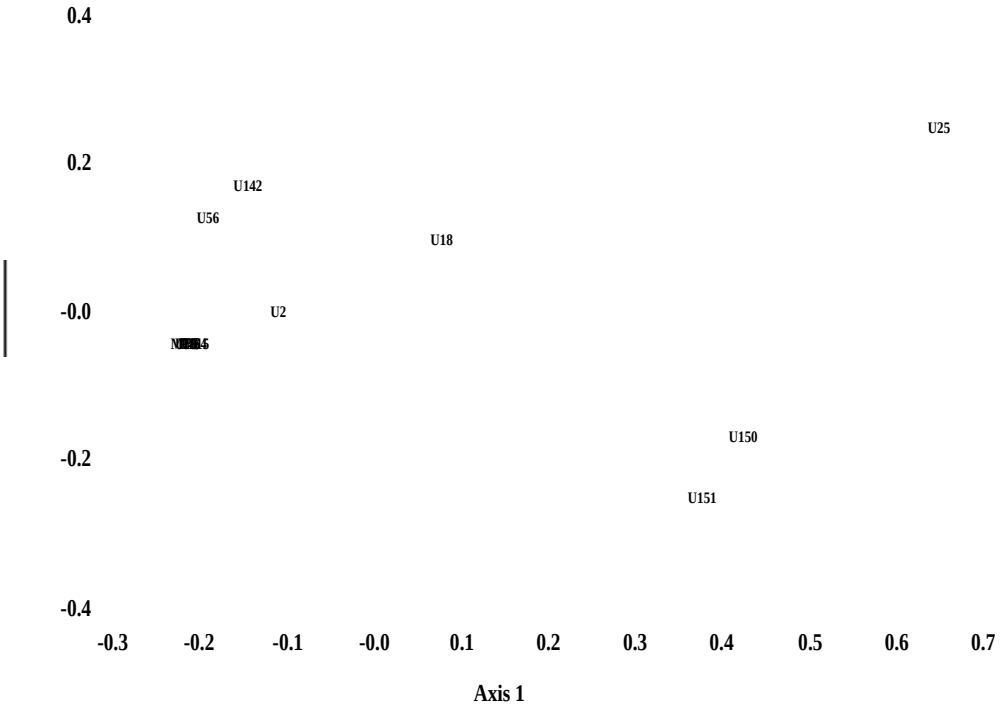




81 units; 43%, 21%, 17%

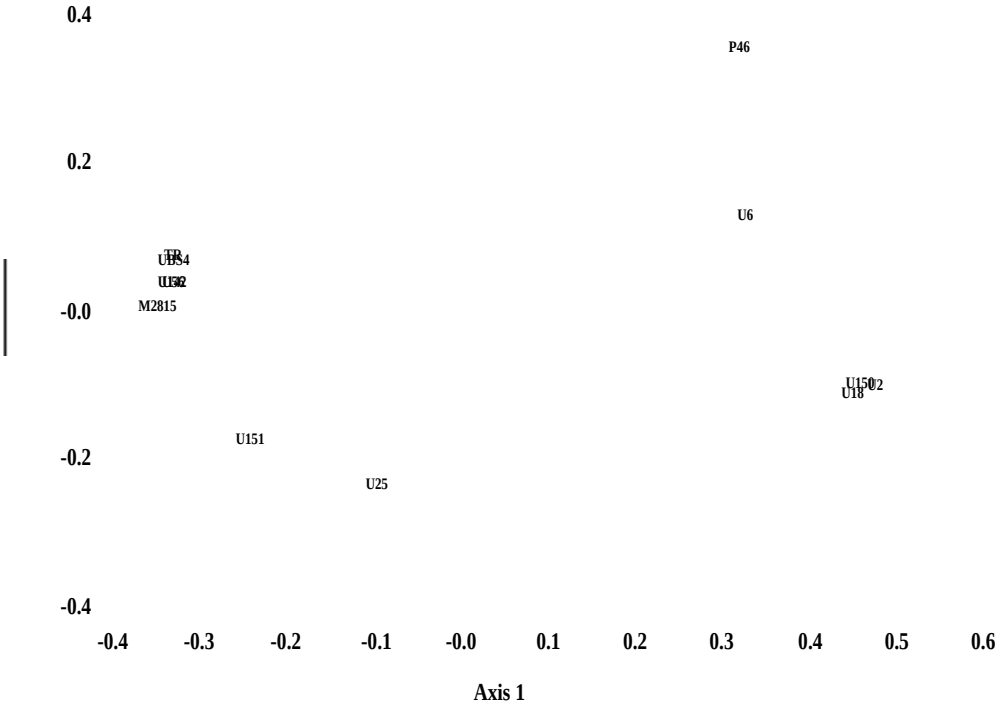
H - I

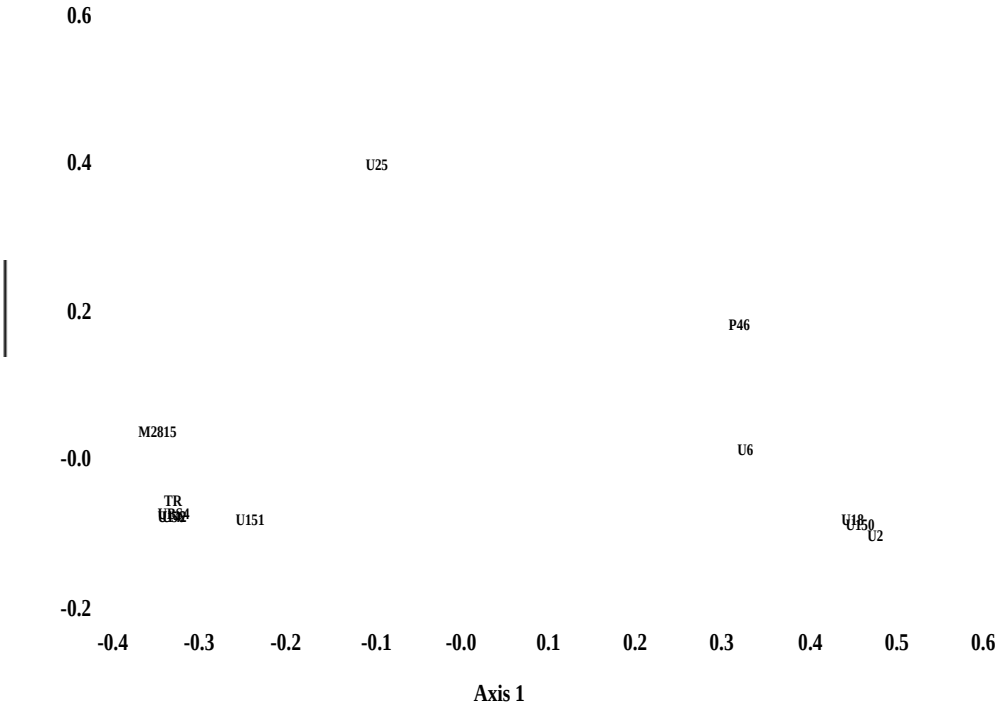




18 units; 52%, 20%, 11%

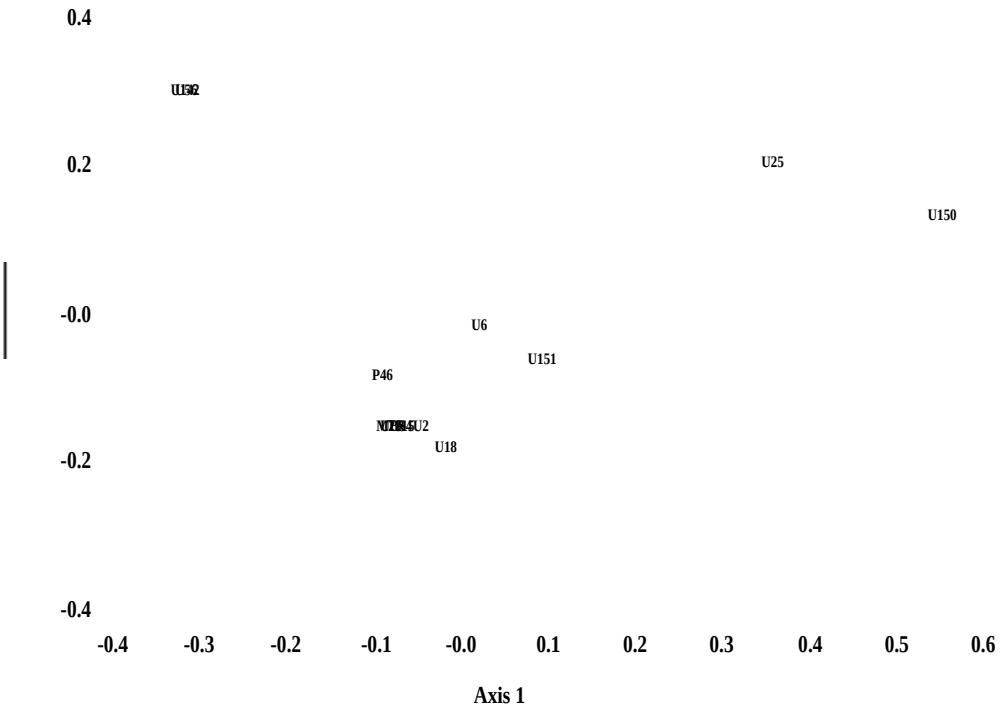
I - J

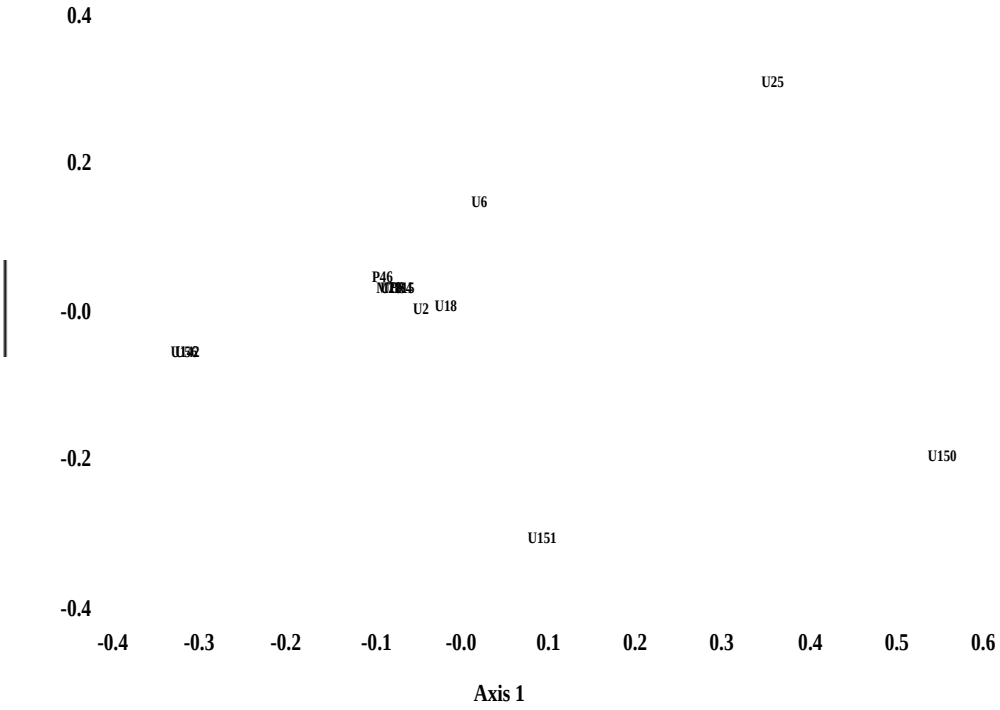




86 units; 59%, 11%, 10%

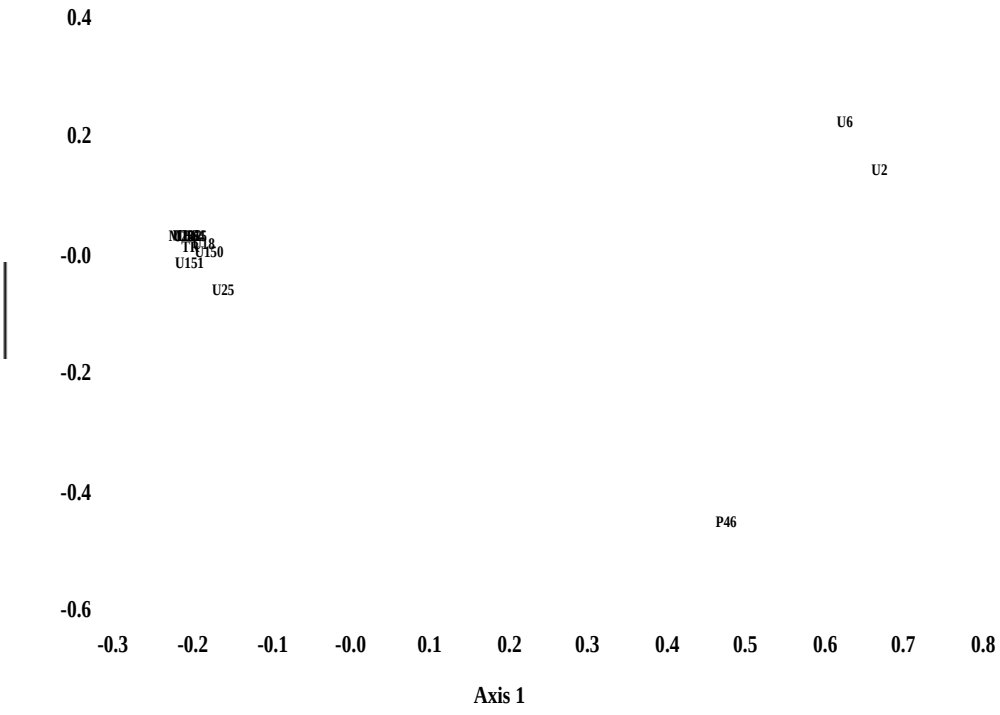
O - W

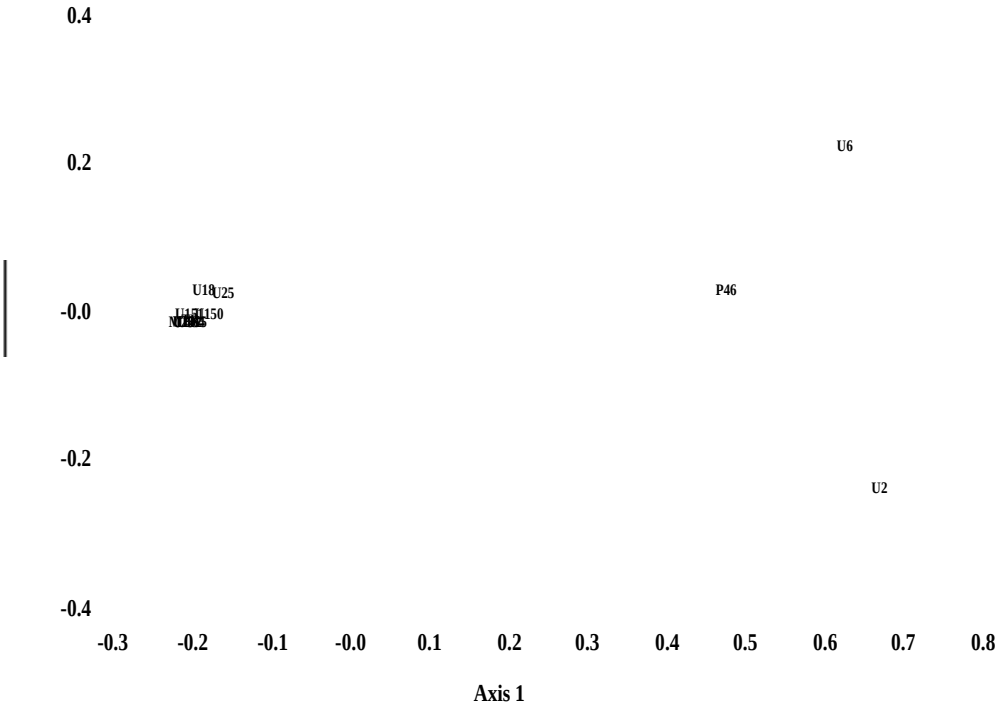




25 units; 39%, 22%, 15%

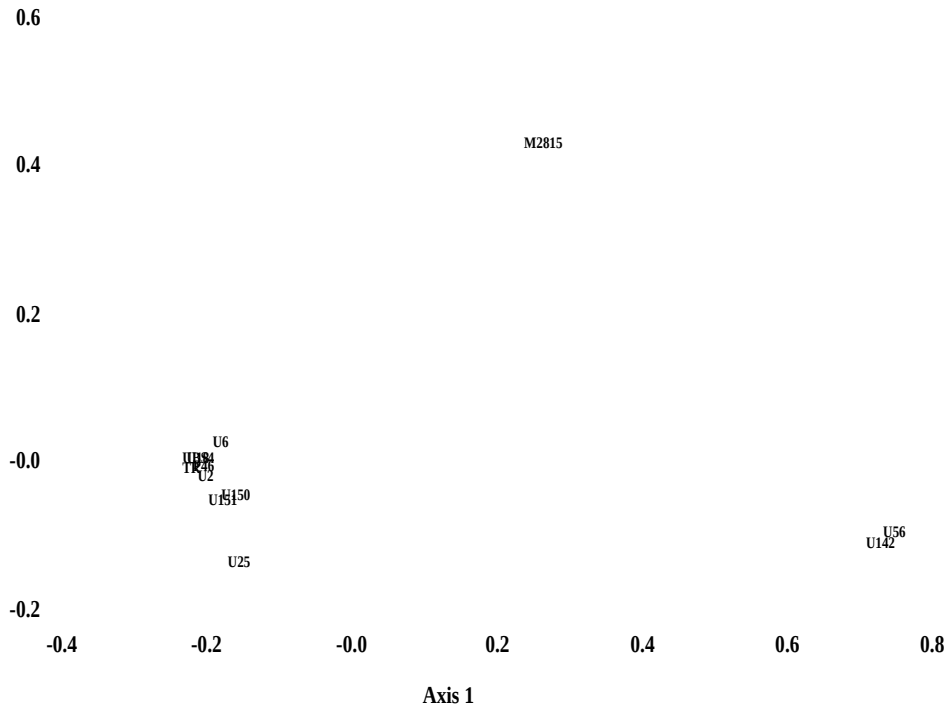
U - V

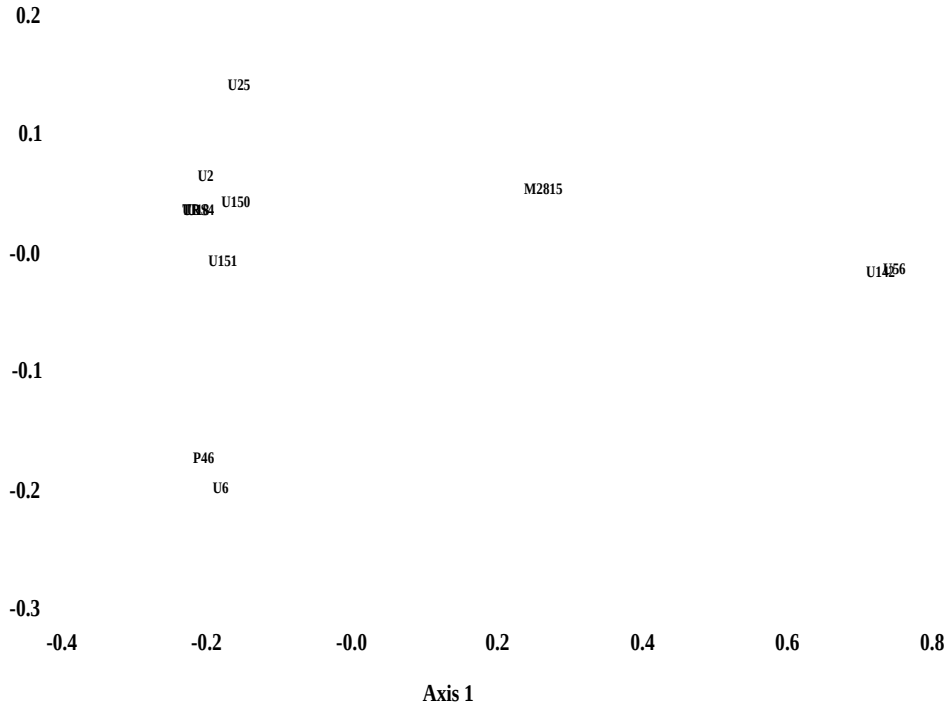




80 units; 72%, 14%, 6%

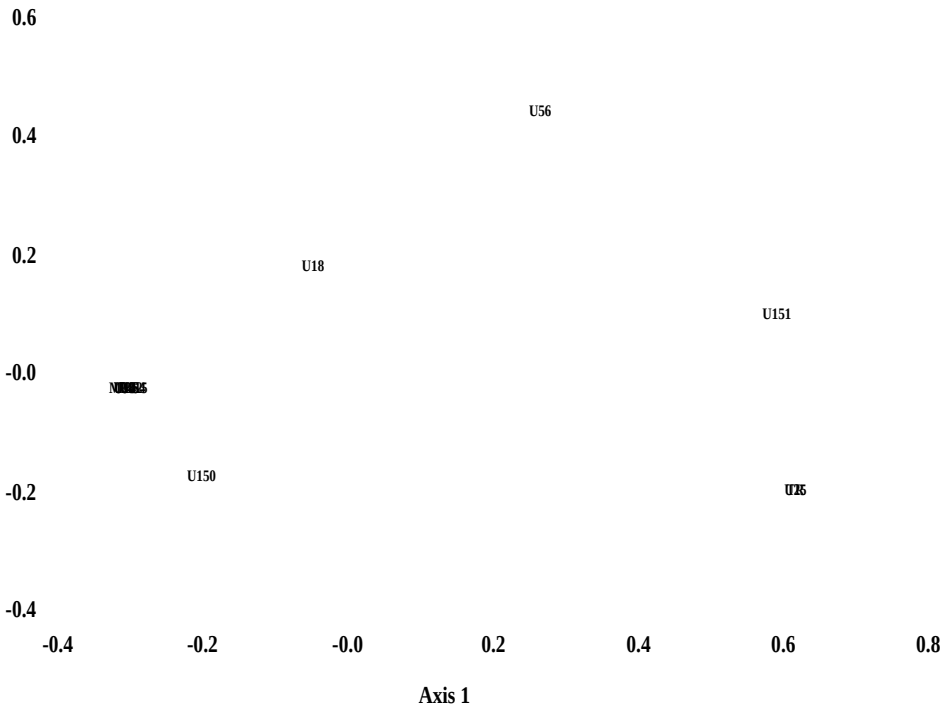
Movable nu

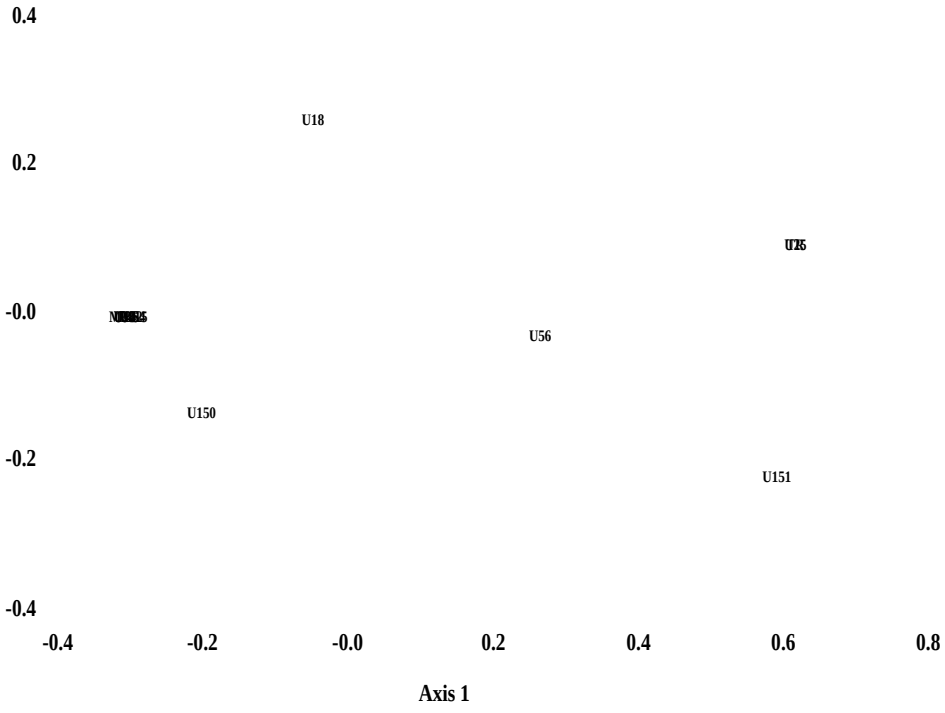




61 units; 73%, 11%, 5%

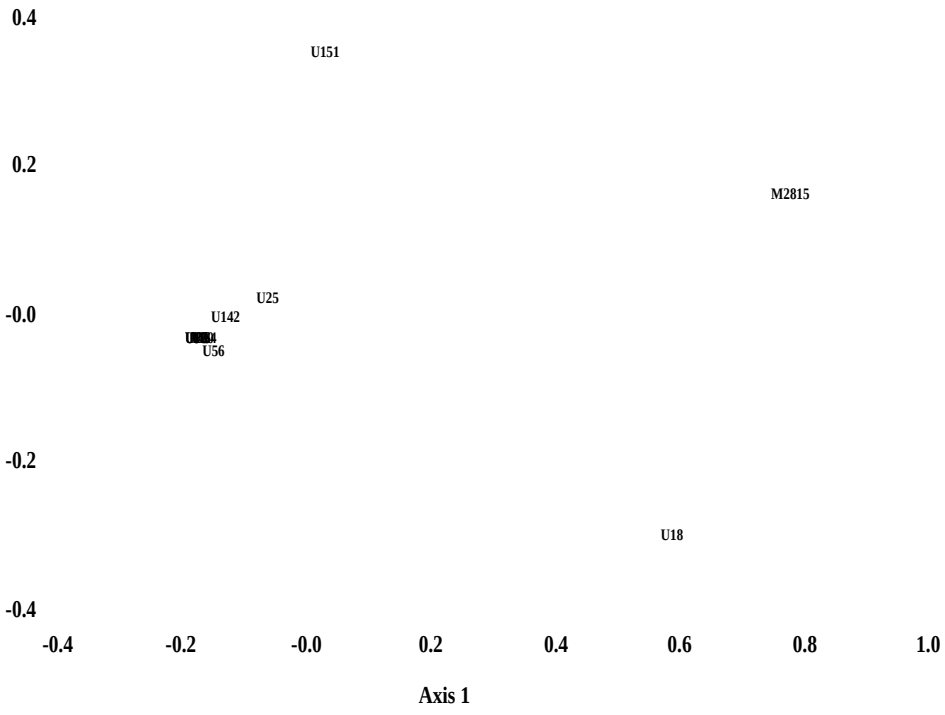
Word division

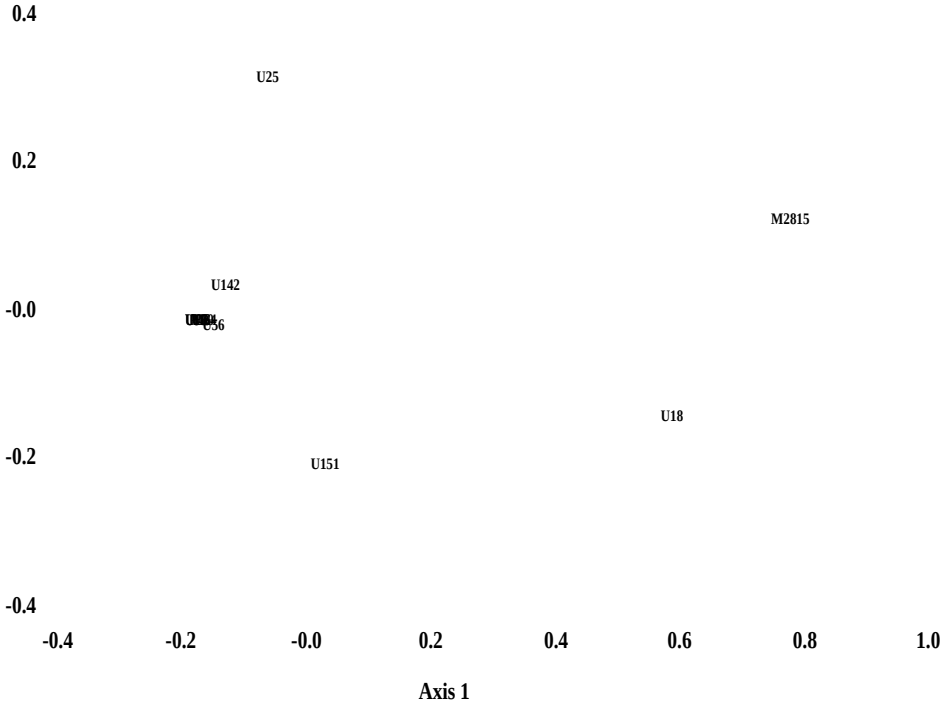




9 units; 74%, 15%, 6%

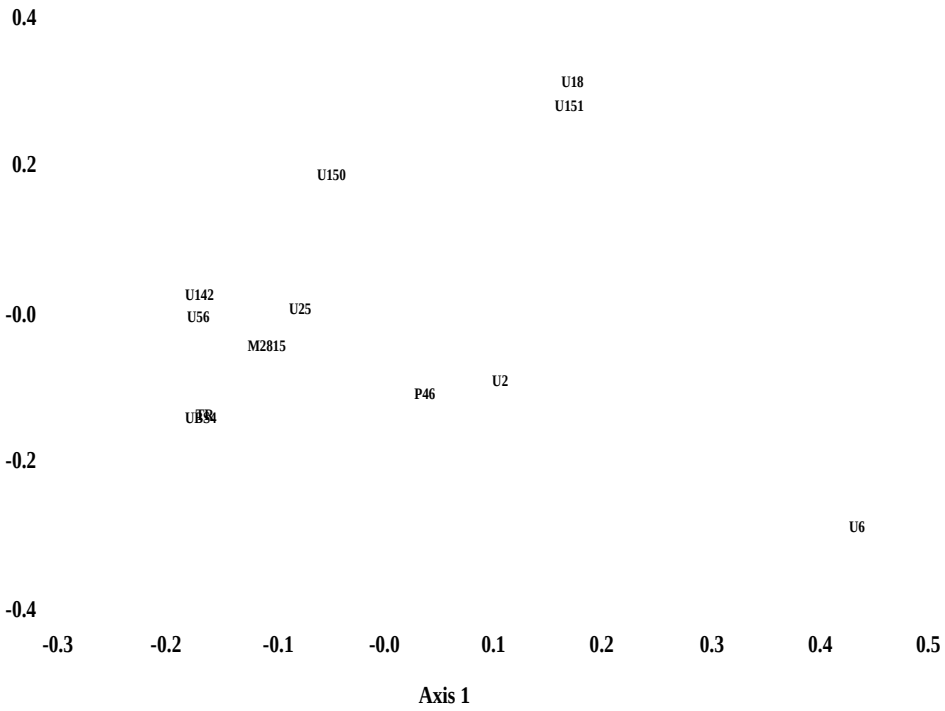
KAI compendium

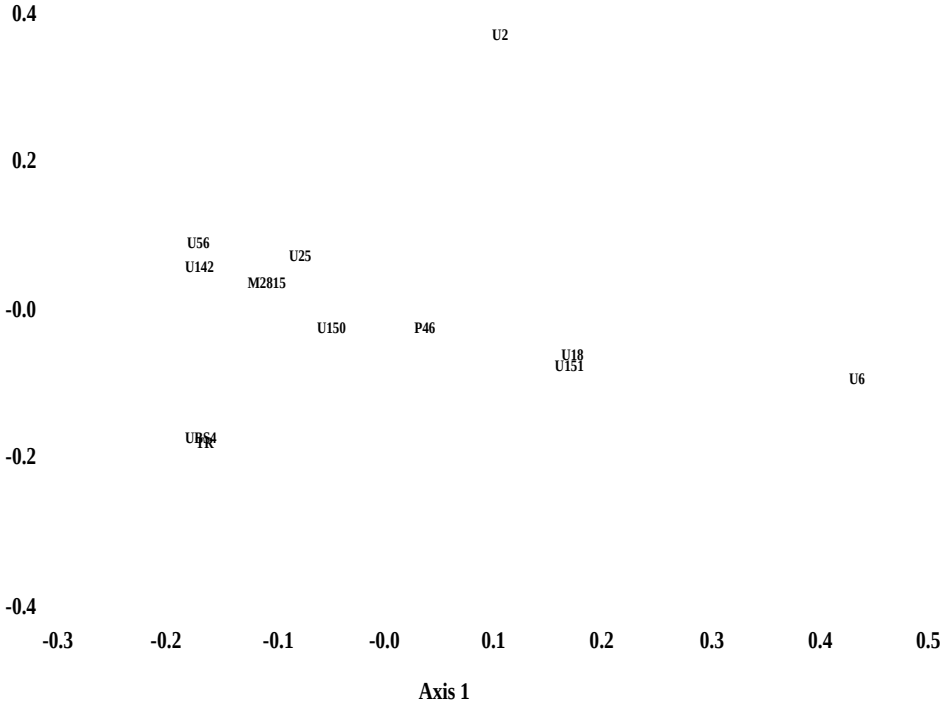




92 units; 67%, 14%, 10%

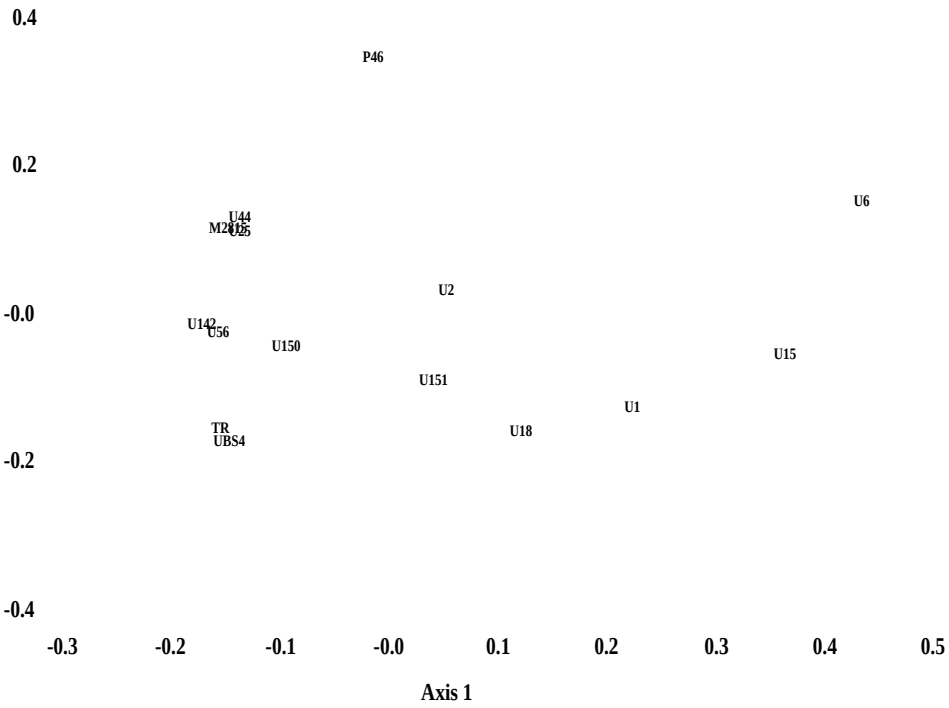
Line division

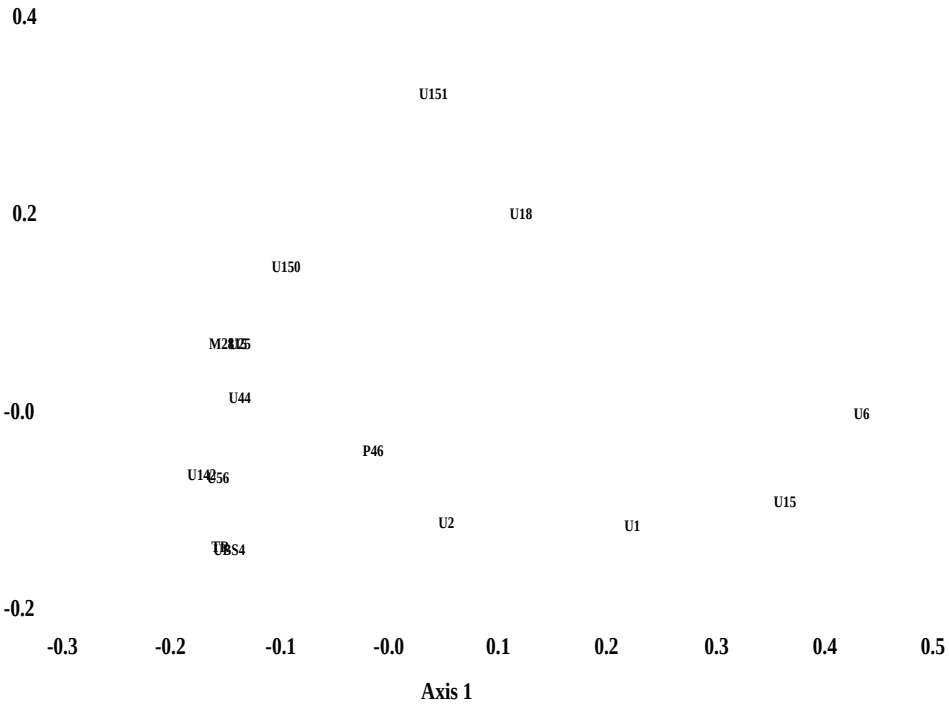




881 units; 18%, 16%, 11%

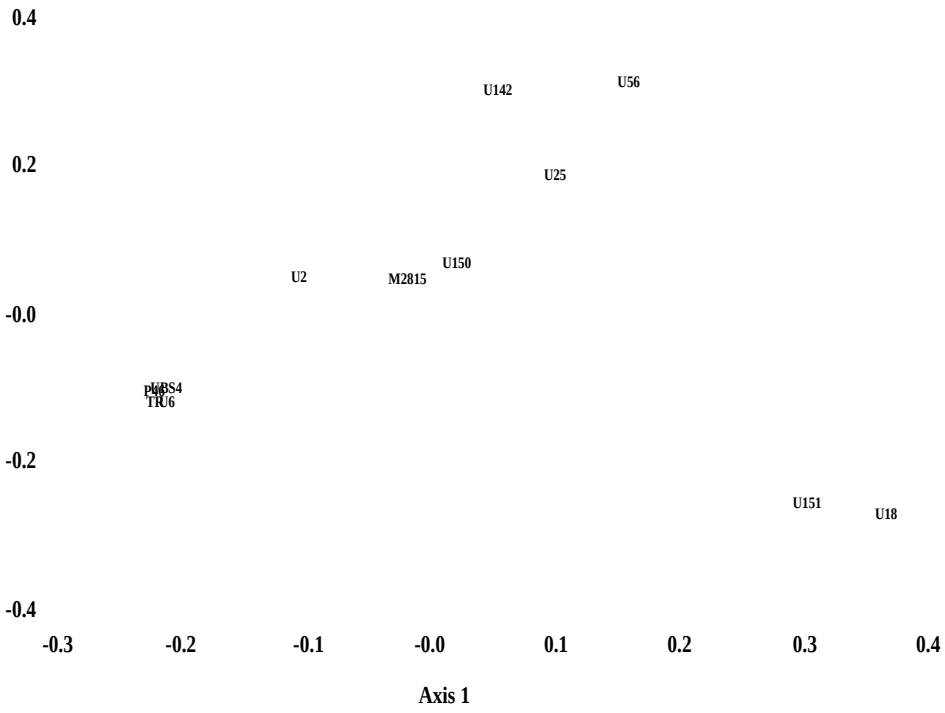
Line division (U15)

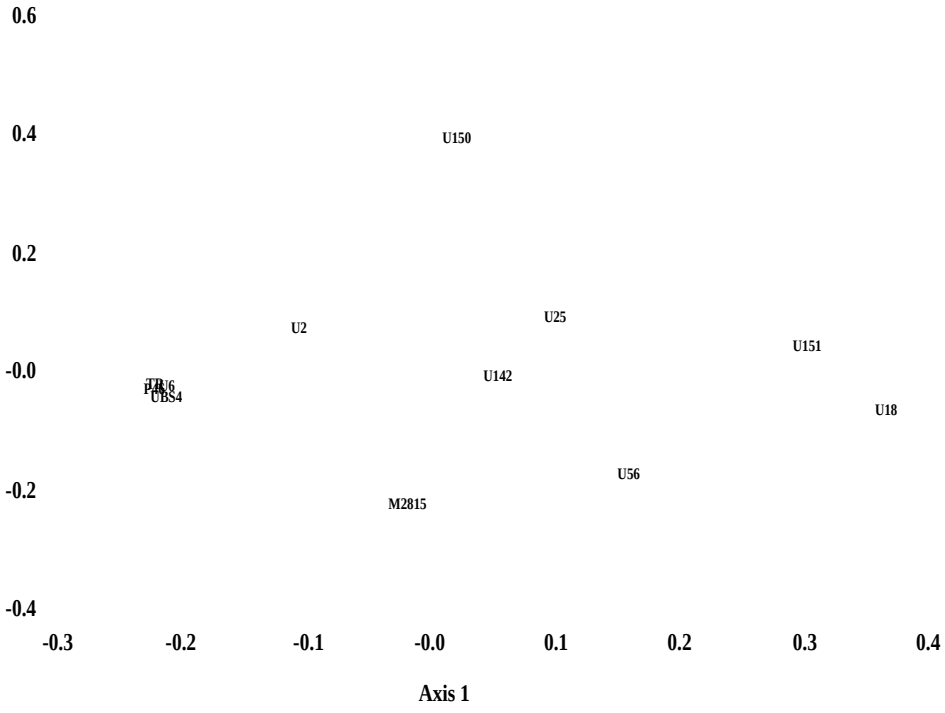




301 units; 20%, 10%, 9%

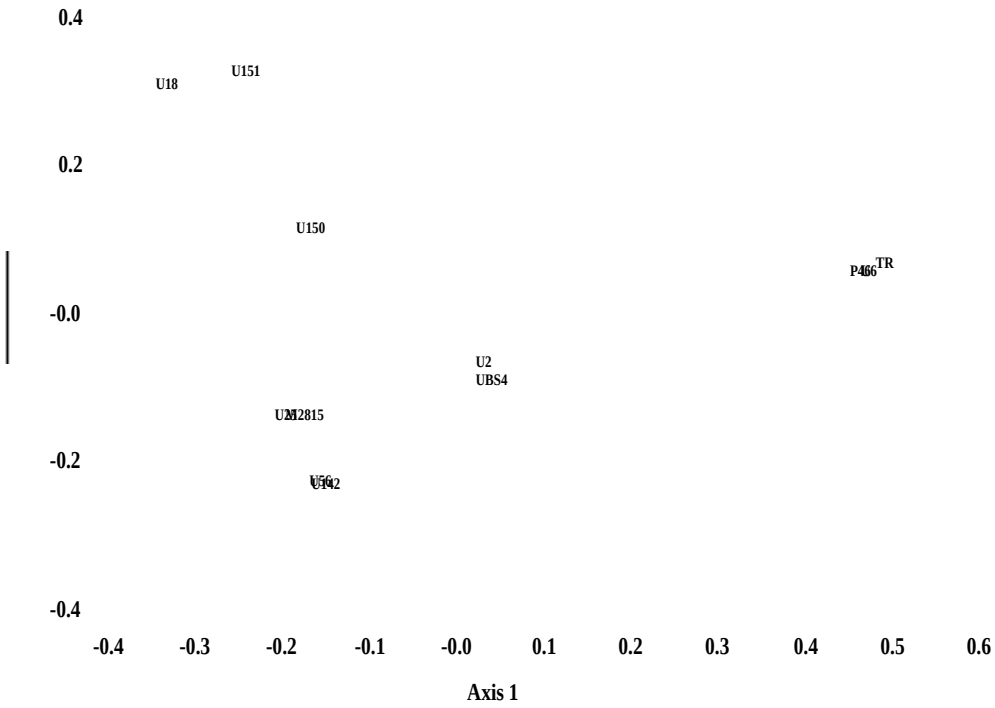
Punctuation

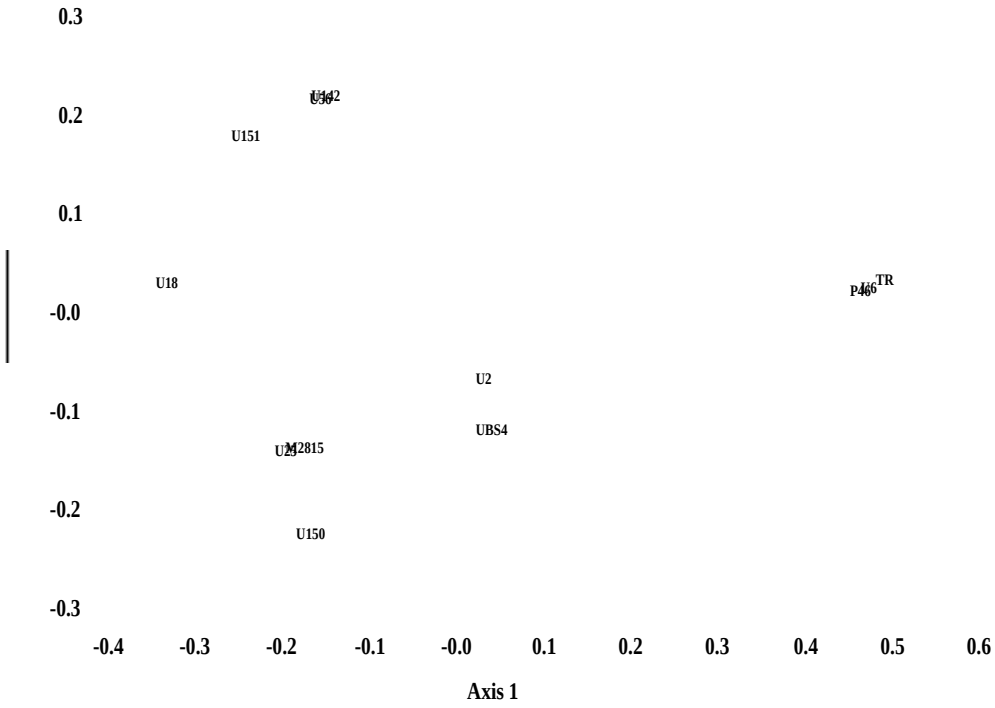




533 units; 22%, 20%, 12%

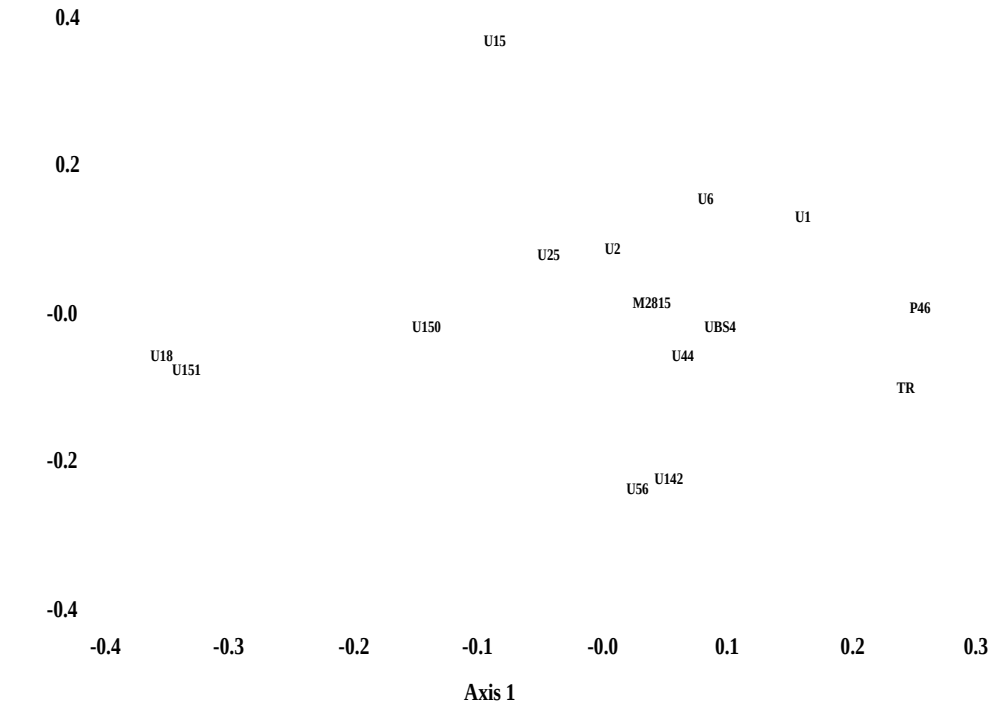
Punctuation (levelled)

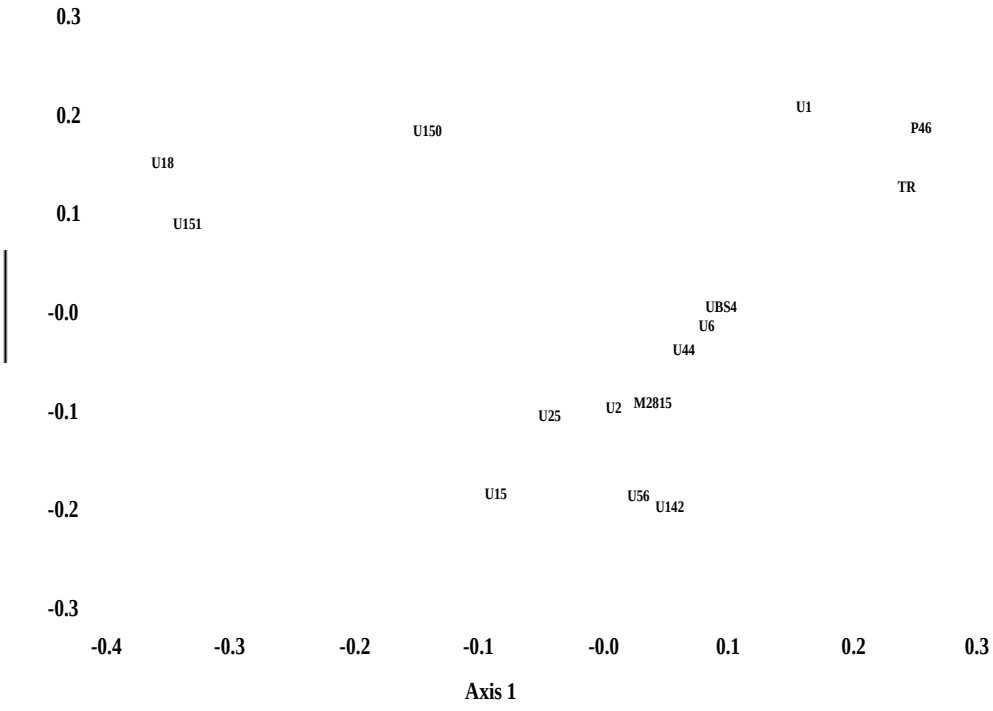




734 units; 43%, 17%, 10%

Line division and punctuation (U15, levelled)





507 units; 16%, 12%, 11%

PROGRAM LISTINGS

This appendix provides listings of the programs described in the chapter on collation.

Preprocessing program to prepare my transcriptions for *Collate 2*:
prep

My collation programs:
Filter
Separate
Format
Normalise
Synthesise
Matrix
Report

My utility programs:
Check
Punctuation
Tabulate
LSI
CT
Halve
LI

Preprocessing program

prep

```
/* Transcription pre-processing program */

/* Where possible, general parameters such as tag control characters have
been */
/* treated as globals and placed in the header 'prep.h'. The transcription
file */
/* to be processed must be in the same directory as the executable program.
*/

prep.h

#define MAXBLOCK    1000         /* max. block size */
#define MAXNAME     32           /* max. file name size */

#define BLOCK_OPEN  '<'          /* block open character */
#define BLOCK_CLOSE '>'          /* block close character */

#define MAXTAG            10           /* max. tag size */

#define TAG_OPEN    '['          /* tag open character */
#define TAG_CLOSE   ']'          /* tag close character */
#define TAG_END     '/'          /* tag end character */

#define COR_TAG     "c"          /* corrector tag initial string (max. of 5
characters) */
#define DEL_TAG     "d"          /* deletor tag initial string (max. of 5
characters) */

/* max. no. of scribes incl. original and unidentified scribes */
/* if numbers 0 to 9 are used as scribe identifier characters, max. no. = 10
*/
/* if letters a to z are used as scribe identifier characters, max. no. = 25
*/
/* (one letter, usually x, is reserved for the unidentified scribe) */
/* other series of ascii characters may be used, in which case max. no. may
be higher */

#define MAXSCR            10

#define ORIG_ID           '0'          /* original scribe identifier
character for transcripts */

#define UNID_ID           'x'          /* unidentified scribe identifier
character for transcripts */

#define COR1_ID           'a'          /* first corrector identifier
character for output file names */

prep.c

#include <stdio.h>
#include <time.h>
#include "prep.h"
#include "block.h"
#include "tag.h"
#include "sep.h"


main()
{
       char buffer[MAXBLOCK];
       char name[MAXNAME];
```

```
	FILE *fp1, *fp2, *fp3;
	clock_t start, secs;

	printf("Source file: ");
	scanf("%s", name);

	start = clock();

	fp1 = fopen(name, "r");		/* Open source file */
	fp2 = tmpfile();		/* Create temporary detag file */
	fp3 = tmpfile();		/* Create temporary error file */

	if (fp1 == NULL)
	{
		fprintf(stderr, "File not found: %s\n", name);
		exit(1);
	}

	while (fgetb(buffer, MAXBLOCK, fp1, BLOCK_OPEN, BLOCK_CLOSE) != NULL)
	{
		if (tagchk(buffer, TAG_OPEN, TAG_CLOSE, TAG_END) == NULL)  /*
Find tag errors */
			fprintf(fp3, "Unmatched tag in block: %s\n", buffer);

		detag(buffer);			/* Detag source */

		fputs(buffer, fp2); /* Write to temporary file */
	}

	fclose(fp1);

	fsep(name, fp2, fp3);		/* Separate correctors */

	fclose(fp2);

	rewind(fp3);

	if (fgets(buffer, MAXBLOCK, fp3))	/* If there are any errors */
	{
		printf("Errors detected: see error file\n");

		strcpy(buffer, name);
		strcat(buffer, ".err");

		fp2 = fopen(buffer, "w");	/* Open errors file */

		rewind(fp3);

		while (fgets(buffer, MAXBLOCK, fp3))
		{
			fputs(buffer, fp2);
		}

		fclose(fp2);
	}

	fclose(fp3);

	secs = (clock() - start) / CLOCKS_PER_SEC;

	printf("Run time: %lu seconds.\n", secs);
}
```

block.h

```
char *fgetb(char *, int, FILE *, int, int);
```

block.c

```
#include <stdio.h>
```

```
/* Read a block from stream */
/* c1 is left block marker, c2 is right block marker*/

char *fgetb(char *s, int n, FILE *stream, int c1, int c2)
{
	int i;

	for (i = 0; i < n; i++)
	{
		*(s + i) = getc(stream);

		if (i > 0 && *(s + i) == c1 && *(s + i - 1) != c2) break;  /*
end of block reached */

		if (*(s + i) == EOF) break;	/* end of file reached */
	}

	ungetc(*(s + i), stream); /* push back last character read */
	*(s + i) = '\0';    /* terminate with null character */

	if (i == 0)  /* no characters read before EOF */
	{
		return NULL; /* unsuccessful return value */
	}

	return s;
}
```

tag.h

```
char *tagchk(char *, int, int, int);
char *tagfnd(char *, int, int, int);
char *strrpl(char *, const char *, const char *);
char *strlwr(char *, const char *, const char *);
char *strfil(char *, const char *, const char *, int);
char *strdis(char *, const char *, const char *);
char *detag(char *);
```

tag.c

```
#include <string.h>
#include <ctype.h>


/* check s for unpaired tags */
/* c1 opening control, c2 closing control, c3 end indicator*/

char *tagchk(char *s, int c1, int c2, int c3)
{
	int i;
	int n1, n2, n3;

	for (i = 0, n1 = 0, n2 = 0, n3 = 0; i < strlen(s); i++)
	{
		if (s[i] == c1) n1++;
		if (s[i] == c2) n2++;
		if (s[i] == c3) n3++;
	}

	if (n1 != n2 || n1 + n2 - 4 * n3 != 0) return NULL;
	else return s;
}


/* check s for tags */
/* c1 opening control, c2 closing control, c3 end indicator */

char *tagfnd(char *s, int c1, int c2, int c3)
{
```

```
	int i;
	int n1, n2, n3;

	for (i = 0, n1 = 0, n2 = 0, n3 = 0; i < strlen(s); i++)
	{
		if (s[i] == c1) n1++;
		if (s[i] == c2) n2++;
		if (s[i] == c3) n3++;
	}

	if (n1 || n2 || n3) return NULL;
	else return s;
}


/* replace s1 with s2 in s */

char *strrpl(char *s, const char *s1, const char *s2)
{
	char *ptr;

	while (ptr = strstr(s, s1))	/* if s1 is in s */
	{
		memmove(ptr + strlen(s2) , ptr + strlen(s1), strlen(ptr));
		memcpy(ptr, &s2[0], strlen(s2));
	}
	return s;
}


/* lower case of s between s1 and s2, delete s1 and s2 */

char *strlwr(char *s, const char *s1, const char *s2)
{
	int i;
	size_t len1, len2;
	char *ptr1, *ptr2;

	while ((ptr1 = strstr(s, s1)) && (ptr2 = strstr(s, s2)))   /* if s1
and s2 are in s */
	{
		len1 = strlen(s) - strlen(ptr1); /* length up to s1 */
		len2 = strlen(s) - strlen(ptr2); /* length up to s2 */

		for (i = len1 + strlen(s1); i < len2; i++)
		{
			s[i] = tolower(s[i]);
		}
		memcpy(&s[len1], &s[len1 + strlen(s1)], strlen(ptr1));
		len2 = len2 - strlen(s1);
		memcpy(&s[len2], &s[len2 + strlen(s2)], strlen(ptr2));
	}
	return s;
}


/* fill s between s1 and s2 with c, delete s1 and s2 */

char *strfil(char *s, const char *s1, const char *s2, int c)
{
	int i;
	size_t len1, len2;
	char *ptr1, *ptr2;

	while ((ptr1 = strstr(s, s1)) && (ptr2 = strstr(s, s2)))   /* if s1
and s2 are in s */
	{
		len1 = strlen(s) - strlen(ptr1); /* length up to s1 */
		len2 = strlen(s) - strlen(ptr2); /* length up to s2 */
```

```
		for (i = len1 + strlen(s1); i < len2; i++)
		{
			if (isgraph(s[i])) s[i] = c;
		}
		memcpy(&s[len1], &s[len1 + strlen(s1)], strlen(ptr1));
		len2 = len2 - strlen(s1);
		memcpy(&s[len2], &s[len2 + strlen(s2)], strlen(ptr2));
	}
	return s;
}


/* discard s between s1 and s2, delete s1 and s2 */

char *strdis(char *s, const char *s1, const char *s2)
{
	char *ptr1, *ptr2;

	while ((ptr1 = strstr(s, s1)) && (ptr2 = strstr(s, s2)))   /* if s1
and s2 are in s */
	{
		memcpy(ptr1, ptr2 + strlen(s2), strlen(ptr2));
	}
	return s;
}


/* detag block */

char *detag(char *s)
{
	strrpl(s, "[st]", "");
	strrpl(s, "[/st]", "");
	strrpl(s, "[it]", "");
	strrpl(s, "[/it]", "");

	strrpl(s, "[fn]", "");
	strrpl(s, "[/fn]", "N");
	strrpl(s, "[di]", "");
	strrpl(s, "[/di]", "");
	strrpl(s, "[sb]", "");
	strrpl(s, "[/sb]", "");
	strrpl(s, "[rb]", "");
	strrpl(s, "[/rb]", "");
	strrpl(s, "[sc]", "");
	strrpl(s, "[/sc]", "");

	strrpl(s, "[kc]K[/kc]", "KAI");

	strrpl(s, "[ns]", "");
	strrpl(s, "[/ns]", "");

	strlwr(s, "[ut]", "[/ut]");

	strfil(s, "[rt]", "[/rt]", '_');

	return s;
}
```

sep.h

```
void fsep(char *, FILE *, FILE *);
```

sep.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "prep.h"
#include "block.h"
```

```
#include "tag.h"

/* Separate scribes from stream */

void fsep(char *s, FILE *stream, FILE *errors)
{
       int i, n, m;
       char buffer[MAXBLOCK];
       char output[MAXNAME];
       char cot[MAXSCR][MAXTAG]; /* Corrector open tags */
       char cct[MAXSCR][MAXTAG]; /* Corrector close tags */
       char dot[MAXSCR][MAXTAG]; /* Deletor open tags */
       char dct[MAXSCR][MAXTAG]; /* Deletor close tags */
       FILE *fp;

       /* Initialise arrays of correction and deletion tags */

       /* Step 1: Initialise tags for unidentified scribe */

       /* 1st token: character */

       cot[MAXSCR - 1][0] = cct[MAXSCR - 1][0] = dot[MAXSCR - 1][0] = dct
[MAXSCR - 1][0] = TAG_OPEN;

       /* 2nd token: character */

       cot[MAXSCR - 1][1] = dot[MAXSCR - 1][1] = '\0';
       cct[MAXSCR - 1][1] = dct[MAXSCR - 1][1] = TAG_END;

       /* 3rd token: string or character */

       strcat(cot[MAXSCR - 1], COR_TAG);
       strcat(dot[MAXSCR - 1], DEL_TAG);
       cct[MAXSCR - 1][2] = dct[MAXSCR - 1][2] = '\0';

       /* 4th token: character or string */

       cot[MAXSCR - 1][strlen(COR_TAG) + 1] = dot[MAXSCR - 1][strlen(DEL_TAG)
+ 1] = UNID_ID;
       strcat(cct[MAXSCR - 1], COR_TAG);
       strcat(dct[MAXSCR - 1], DEL_TAG);

       /* 5th token: character */

       cot[MAXSCR - 1][strlen(COR_TAG) + 2] = dot[MAXSCR - 1][strlen(DEL_TAG)
+ 2] = TAG_CLOSE;
       cct[MAXSCR - 1][strlen(COR_TAG) + 2] = dct[MAXSCR - 1][strlen(DEL_TAG)
+ 2] = UNID_ID;

       /* 6th token: null or character */

       cot[MAXSCR - 1][strlen(COR_TAG) + 3] = dot[MAXSCR - 1][strlen(DEL_TAG)
+ 3] = '\0';
       cct[MAXSCR - 1][strlen(COR_TAG) + 3] = dct[MAXSCR - 1][strlen(DEL_TAG)
+ 3] = TAG_CLOSE;

       /* 7th token: null */

       cct[MAXSCR - 1][strlen(COR_TAG) + 4] = dct[MAXSCR - 1][strlen(DEL_TAG)
+ 4] = '\0';

       /* Step 2: Initialise numbered correction and deletion tags */

       for (i = 0; i < MAXSCR - 1; i++)
       {
              strcpy(cot[i], cot[MAXSCR - 1]);
              strcpy(cct[i], cct[MAXSCR - 1]);
              strcpy(dot[i], dot[MAXSCR - 1]);
              strcpy(dct[i], dct[MAXSCR - 1]);
```

```
            /* 4th token: character or string */

            cot[i][strlen(COR_TAG) + 1] = dot[i][strlen(DEL_TAG) + 1] =
ORIG_ID + i;

            /* 5th token: character */

            cct[i][strlen(COR_TAG) + 2] = dct[i][strlen(DEL_TAG) + 2] =
ORIG_ID + i;
      }

      /* Test for corrections to original */

      m = 0; /* test flag */

      rewind(stream);

      while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE) !=
NULL)
      {
            if ((strstr(buffer, dot[0]) && strstr(buffer, dct[0]))
                  || (strstr(buffer, cot[0]) && strstr(buffer, cct[0])))
            {
                  m = 1;
            }
      }

      /* Separate uncorrected original from s if required */

      if (m == 1)
      {
            printf("Corrections by original scribe found\n");

            strcpy(output, s);
            strcat(output, "*");
            fp = fopen(output, "w");

            rewind(stream);

            while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE)
!= NULL)
            {
                  if ((strstr(buffer, dot[0]) && strstr(buffer, dct[0]))
                        || (strstr(buffer, cot[0]) && strstr(buffer, cct
[0])))
                  {
                        for (i = 0; i < MAXSCR; i++)
                        {
                              strdis(buffer, cot[i], cct[i]);        /*
discard corrections */
                              strrpl(buffer, dot[i], "");
      /* delete d.o. tags*/
                              strrpl(buffer, dct[i], "");
      /* delete d.c. tags*/
                        }
                        fputs(buffer, fp);

                        if (tagfnd(buffer, TAG_OPEN, TAG_CLOSE, TAG_END)
== NULL)     /* Find persistant tags */
                              fprintf(errors, "Persistant tag: %s\n",
buffer);
                  }
            }
            fclose(fp);
      }

      /* Separate corrected original from s */

      strcpy(output, s);
      strcat(output, " ");
```

```
	fp = fopen(output, "w");

	rewind(stream);

	while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE) !=
NULL)
	{
		for (i = 0; i < 1; i++)
		{
			strdis(buffer, dot[i], dct[i]);		/* discard
deletions */
			strrpl(buffer, cot[i], "");			/* delete
c.o. tags*/
			strrpl(buffer, cct[i], "");			/* delete
c.c. tags*/
		}

		for (i = 1; i < MAXSCR; i++)
		{
			strdis(buffer, cot[i], cct[i]);		/* discard
corrections */
			strrpl(buffer, dot[i], "");			/* delete
d.o. tags*/
			strrpl(buffer, dct[i], "");			/* delete
d.c. tags*/
		}
		fputs(buffer, fp);

		if (tagfnd(buffer, TAG_OPEN, TAG_CLOSE, TAG_END) == NULL)  /*
Find persistant tags */
			fprintf(errors, "Persistant tag: %s\n", buffer);
	}
	fclose(fp);

	/* Find number of identified correctors */

	m = 2; /* scribe counter: start with ID no. of corrected original */

	for (i = 0; i < MAXSCR; i++)
	{
		rewind(stream);

		while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE)
!= NULL)
		{
			if ((strstr(buffer, dot[m - 1]) && strstr(buffer, dct[m -
1]))
				|| (strstr(buffer, cot[m - 1]) && strstr(buffer,
cct[m - 1])))
			{
				m++;
				break;
			}
		}
	}
	printf("Number of identified correctors: %d\n", m - 2);

	/* Separate identified correctors from s */

	for (n = 2; n < m; n++)
	{
		strcpy(output, s);
		buffer[0] = COR1_ID + n - 2;
		buffer[1] = '\0';
		strcat(output, buffer);
		fp = fopen(output, "w");

		rewind(stream);

		while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE)
```

```
!= NULL)
            {
                  if ((strstr(buffer, dot[n - 1]) && strstr(buffer, dct[n -
1]))
                        || (strstr(buffer, cot[n - 1]) && strstr(buffer,
cct[n - 1])))
                  {
                        for (i = 0; i < n; i++)
                        {
                              strdis(buffer, dot[i], dct[i]);        /*
discard deletions */
                              strrpl(buffer, cot[i], "");
      /* delete c.o. tags*/
                              strrpl(buffer, cct[i], "");
      /* delete c.c. tags*/
                        }
                        for (i = n; i < MAXSCR; i++)
                        {
                              strdis(buffer, cot[i], cct[i]);        /*
discard corrections */
                              strrpl(buffer, dot[i], "");
      /* delete d.o. tags*/
                              strrpl(buffer, dct[i], "");
      /* delete d.c. tags*/
                        }
                        fputs(buffer, fp);

                        if (tagfnd(buffer, TAG_OPEN, TAG_CLOSE, TAG_END)
== NULL)     /* Find persistant tags */
                              fprintf(errors, "Persistant tag: %s\n",
buffer);
                  }
            }
      }
      fclose(fp);


      /* Test for unidentified scribe */

      m = 0; /* test flag */

      rewind(stream);

      while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE) !=
NULL)
      {
            if ((strstr(buffer, dot[MAXSCR - 1]) && strstr(buffer, dct
[MAXSCR - 1]))
                  || (strstr(buffer, cot[MAXSCR - 1]) && strstr(buffer, cct
[MAXSCR - 1])))
            {
                  m = 1;
            }
      }

      /* Separate unidentified scribe from s if required */

      /* This scribe is treated as the last corrector. */

      if (m == 1)
      {
            printf("Corrections by unidentified scribe found\n");

            strcpy(output, s);
            buffer[0] = UNID_ID;
            buffer[1] = '\0';
            strcat(output, buffer);
            fp = fopen(output, "w");

            rewind(stream);
```

```
            while (fgetb(buffer, MAXBLOCK, stream, BLOCK_OPEN, BLOCK_CLOSE)
!= NULL)
            {
                  if ((strstr(buffer, dot[MAXSCR - 1]) && strstr(buffer,
dct[MAXSCR - 1]))
                        || (strstr(buffer, cot[MAXSCR - 1]) && strstr
(buffer, cct[MAXSCR - 1])))
                  {
                        for (i = 0; i < MAXSCR; i++)
                        {
                              strdis(buffer, dot[i], dct[i]);        /*
discard deletions */
                              strrpl(buffer, cot[i], "");
      /* delete c.o. tags*/
                              strrpl(buffer, cct[i], "");
      /* delete c.c. tags*/
                        }
                        fputs(buffer, fp);

                        if (tagfnd(buffer, TAG_OPEN, TAG_CLOSE, TAG_END)
== NULL)     /* Find persistant tags */
                              fprintf(errors, "Persistant tag: %s\n",
buffer);
                  }
            }
            fclose(fp);
      }
}
```

Collation programs

Filter

```
#!/usr/local/bin/perl -w

# Filter.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, output)

$InputDirectory = 'Transcribed';
$OutputDirectory = 'Filtered';

# Delimiters bracketing text to be discarded

@Delimiters = (
      '{', '}',
      '|', '|'
      );

# Substitutions

@Substitutions = (
      '[/fn]', 'N',
      '[di]I[/di]', 'J',
      '[di]U[/di]', 'V',
      '[di]UI[/di]', 'VJ',
      );
```

```
# Items to be discarded

@Items = (
       '[st]', '[/st]', '[it]', '[/it]', '[kc]', '[/kc]',
       '[sc]', '[/sc]', '[ns]', '[/ns]', '[fn]', '[/fn]',
       '[rb]', '[/rb]', '[sb]', '[/sb]', '[di]', '[/di]'
       );

# END SPECIFICATIONS

# Print introduction

print "Filter: Filter unwanted items from transcriptions.\n\n";

print "Note:\n\n";

print "(1) Newline conversion: ASCII 13 is changed to ASCII 10\n";
print "or vice versa, depending on the operating system.\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input directory: $InputDirectory.\n";
print "Output directory: $OutputDirectory.\n\n";

for($i = 0; $i < $#Delimiters; $i += 2)
{ print $Delimiters[$i], "This will be discarded",
       $Delimiters[$i + 1], "\n" }
print "\n";

for($i = 0; $i < $#Substitutions; $i += 2)
{ print $Substitutions[$i], " will be replaced with ",
       $Substitutions[$i + 1], "\n" }
print "\n";

print "These items will then be discarded:\n";
$Length = 0;
$i = 0;
while($i <= $#Items)
{
       print $Items[$i];
       print " ";
       $Length += length($Items[$i]) + 2;
       if($Length > 60)
       {
              print "\n";
              $Length = 0;
       }
       $i += 1;
}
print "\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
       $InputPath = '.'.$InputPath;
```

```
	$OutputPath = '.'.$OutputPath;
}

# Make output directory

mkdir($OutputDirectory, 0770);

# Open input directory

opendir(INDIR, $InputDirectory) or die
	"Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# FILTER TRANSCRIPTIONS

print "\nFilter transcriptions:\n";

foreach $File (@Contents)
{
	print "$InputPath$File -> $OutputPath$File\n";

	# Read input file

	open(INPUT, '<'.$InputPath.$File) or die
	"Can't read $InputPath$File";

	@List = <INPUT>;

	# Close file

	close(INPUT);

	# Make output string

	$String = join('', @List);

	# Convert newlines

	if($System eq 'UNIX') { $Find = chr(13) }
	elsif($System eq 'MAC') { $Find = chr(10) }

	$String =~ s/$Find/\n/g;

	# Discard delimited text

	for($i = 0; $i <= $#Delimiters; $i += 2)
	{
		$Open = quotemeta($Delimiters[$i]);
		$Close = quotemeta($Delimiters[$i + 1]);
		$String =~ s/$Open[\S\W]*?$Close//g;
	}

	# Perform substitutions

	for($i = 0; $i <= $#Substitutions; $i += 2)
	{
		$Find = quotemeta($Substitutions[$i]);
		$Replace = $Substitutions[$i + 1];
		$String =~ s/$Find/$Replace/g;
	}

	# Discard unwanted items
```

```
	foreach $Item (@Items)
	{
		$Find = quotemeta($Item);
		$String =~ s/$Find//g;
	}

	# Remove multiple spaces

	$String =~ tr/ / /s;

	# Write output file

	open(OUTPUT, '>'.$OutputPath.$File) or die
	"Can't write $OutputPath$File";

	# Write output to file

	print OUTPUT $String;

	# Close file

	close(OUTPUT);
}

# Print end message

$Duration = time - $^T;

print "\nFilter finished in $Duration seconds.\n";
```

Separate

```
#!/usr/local/bin/perl -w

# Separate.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, output)

$InputDirectory = 'Filtered';
$OutputDirectory = 'Separated';

# Division marker delimiters

@Division = (
	'<',	# Open delimiter
	'>'		# Close delimiter
	);

# Tag delimiters

@Tag = (
	'[',	# Start-tag open delimiter
	'[/',	# End-tag open delimiter
	'd',	# Deletion code
	'c',	# Insertion code
	']'		# Tag close delimiter
	);

# END SPECIFICATIONS

# Print introduction
```

```
print "Separate: Separates scribal stages.\n\n";

print "Notes:\n\n";

print "Alteration tag generic identifiers must contain:\n";
print "(a) an alphanumeric insertion or deletion code, then\n";
print "(b) a scribal identification number (0,1,2,...,x).\n";
print " x denotes alteration by an unidentified scribe.\n";
print "The original scribe stage is separated for all divisions.\n";
print "Subsequent stages are separated for altered divisions alone.\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input directory: $InputDirectory\n";
print "Output directory: $OutputDirectory\n\n";

# Make division marker components

$DMO = $Division[0];      # Division Marker Open
$DMC = $Division[1];      # Division Marker Close

print "Division markers should look like this:\n",
      $DMO, "DIVISION MARKER", $DMC, "\n\n";

# Make tag components

$DTSO = $Tag[0].$Tag[2];  # Deletion Tag Start Open
$DTEO = $Tag[1].$Tag[2];  # Deletion Tag End Open
$ITSO = $Tag[0].$Tag[3];  # Insertion Tag Start Open
$ITEO = $Tag[1].$Tag[3];  # Insertion Tag End Open
$TC = $Tag[4];                          # Tag Close

print "Alteration tags should look like this (n = 0,1,2,...,x):\n",
      $DTSO, "n", $TC, "DELETION", $DTEO, "n", $TC, " ",
      $ITSO, "n", $TC, "INSERTION", $ITEO, "n", $TC, "\n\n";

# Make patterns for regular expressions

$DMO = quotemeta($DMO);
$DMC = quotemeta($DMC);
$DTSO = quotemeta($DTSO);
$DTEO = quotemeta($DTEO);
$ITSO = quotemeta($ITSO);
$ITEO = quotemeta($ITEO);
$TC = quotemeta($TC);

# Make output directory

mkdir($OutputDirectory, 0770);

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
      $InputPath = '.'.$InputPath;
```

```
	$OutputPath = '.'.$OutputPath;
}

# Open input directory

opendir(INDIR, $InputDirectory) or die
	"Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# SEPARATE SCRIBAL STAGES

print "Separate scribal stages:\n";

foreach $InputName (@Contents)
{
	# Open input file for reading

	open(INPUT, '<'.$InputPath.$InputName) or die
		"Can't open input file $InputPath$InputName\n";

	print $InputPath, $InputName, "\n";

	# Make transcription into one string

	@InputList = <INPUT>;
	$Input = join('', @InputList);

	# Close file

	close(INPUT);

	# Make list of divisions

	@Divisions = &ListDivisions($Input);

	# Make list of scribes

	@Scribes = &ListScribes($Input);

	# SEPARATE ORIGINAL STAGE FOR ALL DIVISIONS

	# Make string of relevant divisions

	$Original = $Input;

	# Make lists of earlier and later scribes

	@Earlier = ();
	@Later = @Scribes;

	# Exclude later alterations

	$Original = &ExcludeEarlier($Original, @Later);

	# Make output file name

#	if(@Scribes > 0) { $OutputName = $InputName.'-*' }
#	else { $OutputName = $InputName }

	$OutputName = $InputName;

	# Open output file for writing
```

```
        open(OUTPUT, '>'.$OutputPath.$OutputName) or die
                     "Can't write to file $OutputName in $OutputDirectory";
        print "-> ", $OutputPath.$OutputName, "\n";

        # Write output to file

        print OUTPUT $Original;

        # Close file

        close(OUTPUT);

        # SEPARATE SUBSEQUENT STAGES FOR ALTERED DIVISIONS ALONE

        foreach $Scribe (@Scribes)
        {
               # Make string of relevant divisions

               @Relevant = grep(/[$DTSO$ITSO]$Scribe$TC/, @Divisions);
               $Subsequent = join('', @Relevant);

               # Update lists of earlier and later scribes

               push(@Earlier, shift(@Later));

               # Exclude later alterations

               $Subsequent = &ExcludeEarlier($Subsequent, @Later);

               # Include earlier alterations

               $Subsequent = &IncludeEarlier($Subsequent, @Earlier);

               # Make output file name (1st corr. = 1, 2nd corr. = 2, ...)

               $OutputName = $InputName.'-'.$Scribe;

               # Open output file for writing

               open(OUTPUT, '>'.$OutputPath.$OutputName) or die
                      "Can't write to file $OutputName in $OutputDirectory";
               print "-> ", $OutputPath.$OutputName, "\n";

               # Print out result

               print OUTPUT $Subsequent;

               # Close file

               close(OUTPUT);
        }
}

# Print end message

$Duration = time - $^T;

print "\nSeparate finished in $Duration seconds.\n";

sub ListDivisions   # Make list of divisions
{
        my($Input) = $_[0];
        my(@Parts) = ();
        my(@List) = ();
        my($Item);
        my($Division) = "";

        @Parts = split(/($DMO[\S\W]*?$DMC)/, $Input);

        foreach $Item (@Parts)
```

```
	{
		if($Item =~ /$DMO[\S\W]*?$DMC/)
		{ $Division = $Division.$Item }
		elsif($Item =~ /[\S\W]+/)
		{
			$Division = $Division.$Item;
			push(@List, $Division);
			$Division = "";
		}
	}

	return @List;
}

sub ListScribes # Make compact list of scribes
{
	# Make list of scribal identifiers

	my(@Identifiers) = ();
	push(@Identifiers, $_[0] =~ /$DTSO.*?(\d+|x)$TC/g);
	push(@Identifiers, $_[0] =~ /$ITSO.*?(\d+|x)$TC/g);

	# Sort and compact list

	@Identifiers = sort @Identifiers;

	my($Identifier) = '';
	my($Previous) = '';
	my(@List) = ();

	foreach $Identifier (@Identifiers)
	{
		if($Identifier ne $Previous)
		{ push(@List, $Identifier) }

		$Previous = $Identifier;
	}

	return @List;
}

sub ExcludeEarlier  # Exclude alterations by later scribes
{
	my($Output) = $_[0];
	my($i);

	for($i = 1; $i <= $#_; $i += 1)
	{
		# Delete later deletion tags

		$Output =~ s/$DTSO$_[$i]$TC([\S\W]*?)$DTEO$_[$i]$TC/$1/g;

		# Delete later insertions and associated tags

		$Output =~ s/$ITSO$_[$i]$TC[\S\W]*?$ITEO$_[$i]$TC//g;
	}

	return $Output;
}

sub IncludeEarlier  # Include alterations by earlier scribes
{
	my($Output) = $_[0];
	my($i);

	for($i = 1; $i <= $#_; $i += 1)
	{
		# Delete earlier deletions and associated tags

		$Output =~ s/$DTSO$_[$i]$TC[\S\W]*?$DTEO$_[$i]$TC//g;
```

```
              # Insert earlier insertions and delete associated tags

              $Output =~ s/$ITSO$_[$i]$TC([\S\W]*?)$ITEO$_[$i]$TC/$1/g;
       }

       return $Output;
}
```

Format

```
#!/usr/local/bin/perl -w

# Format.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, output)

$InputDirectory = 'Separated';
$OutputDirectory = 'Formatted';

# Division marker delimiters

@Division = (
       '<',   # Open delimiter
       '>'           # Close delimiter
       );

# Tag delimiters

@Tag = (
       '[',   # Start-tag open delimiter
       '[/',  # End-tag open delimiter
       ']'           # Tag close delimiter
       );

# Certainty tag generic identifiers

@Certainties = (
       'ut',  # Uncertain text
       'rt'   # Reconstructed text
       );

# Corresponding certainties

$Uncertain = 0.5;
$Reconstructed = 0.05;

# Concatenation marker

$Concatenation = '=';

# Newline character and punctuation. These must not be
# alphanumeric, the underscore (_), or empty ('').
# Punctuation must not include the newline character.

$NewLine = '|';

@Punctuation = (
       ".", "≥", "˜", ",", "≤", ";", ":", "…", "'", "-",
       "+", "¶"
       );
```

```
# Include newline and punctuation characters?
# 0 = no, 1 = yes.

$IncNewLines = 0;
$IncPunctuation = 1;

# END SPECIFICATIONS

# Print introduction

print "Format:\n\n";

print "This program rearranges each witness file into the\n";
print "format required for subsequent steps:\n";
print "(1) eradicate potential problems\n";
print "- discard text preceding first div. marker\n";
print "- replace tabs with spaces\n";
print "(2) process line divisions\n";
print "- discard concatenation markers and associated newlines\n";
print "- discard multiple consecutive newlines\n";
print "- replace remaining newlines with specified character\n";
print "- isolate newline characters\n";
print "(3) process punctuation\n";
print "- isolate specified punctuation\n";
print "(4) process tags\n";
print "- discard all tags except certainty level tags\n";
print "- assign certainty to each character according to tags\n";
print "(5) generate output.\n\n";

print "Notes:\n\n";

print "(1) The newline character and punctuation must not be\n";
print "    alphanumeric characters or the underscore (_).\n";
print "(2) Ensure that specified punctuation is comprehensive.\n";
print "    (The Punctuation utility lists possible punctuation.)\n";
print "    Punctuation must not include the newline character.\n";
print "(3) Each output character is assigned a certainty\n";
print "    (i.e. probability of being what is reported)\n";
print "    according to its certainty level tags:\n";
print "    (a) no tags = certainty of 1.00\n";
print "    (b) uncertain text tags  = as specified\n";
print "    (c) reconstructed text tags  = as specified.\n";
print "    The certainty of each output item is calculated from\n";
print "    the certainties of its characters.\n";
print "(4) Output is formatted as follows:\n";
print "    Item TAB Certainty NEWLINE\n";
print "(5) Newline and punctuation characters must be specified\n";
print "    but may be excluded from output if desired.\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input will be read from the directory named ",
      $InputDirectory, ".\n";
print "Output will be placed in the directory named ",
      $OutputDirectory, ".\n\n";

print "Division markers should look like this:\n",
      $Division[0], "DIVISION MARKER", $Division[1], "\n\n";

print "Certainty tags should look like this:\n";
print $Tag[0], $Certainties[0], $Tag[2], "UNCERTAIN TEXT",
      $Tag[1], $Certainties[0], $Tag[2], "\n";
print $Tag[0], $Certainties[1], $Tag[2], "RECONSTRUCTED TEXT",
      $Tag[1], $Certainties[1], $Tag[2], "\n\n";
```

```
print "Certainties are assigned as follows:\n";
print "Uncertain text probability = $Uncertain\n";
print "Reconstructed text probability = $Reconstructed\n\n";

print "Concatenation marker: $Concatenation\n\n";
if($Concatenation eq '')
{ die "Error: The concatenation character has not been specified.\n" }

print "Newline character: $NewLine\n\n";
if($NewLine eq '')
{ die "Error: The newline character has not been specified.\n" }
if($NewLine =~ /\w/) { die "Error: $NewLine is alphanumeric.\n" }

print "Punctuation (ASCII value and character equivalent):\n";
$i = 1;
foreach $Item (sort(@Punctuation))
{
       printf("%2d: %3d = %s\n", $i, ord($Item), $Item);
       if($Item =~ /\w/) { die "Error: $Item is alphanumeric.\n" }
       if($Item eq '')
              { die "Error: This punctuation specification is empty.\n" }
       if($Item eq $NewLine)
              { die "Error: Punctuation is newline character.\n" }
       $i += 1;
}

unless($IncNewLines and $IncPunctuation)
{ print "\nFeatures excluded from output:\n" }
unless($IncNewLines) { print "newline characters\n" }
unless($IncPunctuation) { print "punctuation\n" }

# Make output directory

mkdir($OutputDirectory, 0770);

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
       $InputPath = '.'.$InputPath;
       $OutputPath = '.'.$OutputPath;
}

# Open input directory

opendir(INDIR, $InputDirectory) or die
       "Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Make patterns for regular expressions

$UTS = quotemeta($Tag[0].$Certainties[0].$Tag[2]);  # Unc. start
$UTE = quotemeta($Tag[1].$Certainties[0].$Tag[2]);  # Unc. end
```

```
$RTS = quotemeta($Tag[0].$Certainties[1].$Tag[2]);   # Rec. start
$RTE = quotemeta($Tag[1].$Certainties[1].$Tag[2]);   # Rec. end
$DMO = quotemeta($Division[0]);  # Division Marker Open
$DMC = quotemeta($Division[1]);  # Division Marker Close

# FORMAT TRANSCRIPTIONS

print "\nFormat transcriptions:\n\n";

foreach $InputName (@Contents)
{
	# Open input file for reading

	open(INPUT, '<'.$InputPath.$InputName) or die
		"Can't open input file $InputPath$InputName";
	print "$InputPath$InputName\n";

	# Make transcription into one string

	$Input = join('', <INPUT>);

	# Close file

	close(INPUT);

	# Eradicate potential problems:

	# (1) Discard text preceding first div. marker

	unless($Input =~ /^$DMO.*?$DMC/)
	{
		$Input =~ s/(^[\S\W]+?)($DMO.*?$DMC)/$2/;
		print "Text preceding first div. marker discarded:\n";
		print "$1\n";
	}

	# (2) Replace tabs with spaces

	if($Input =~ /\t/)
	{
		$Input =~ s/\t/ /g;
		print "Tabs replaced with spaces.\n";
	}

	# (3) Add newline at end

	$Input = $Input."\n";

	# Discard concatenation markers and associated newlines

	$Pattern = quotemeta($Concatenation);
	$Input =~ s/$Pattern\n//g;

	# Discard multiple consecutive newlines

	$Input =~ tr/\n/\n/s;

	# Replace remaining newlines with specified character and
	# isolate if required. Otherwise replace with space.

	if($IncNewLines) { $Input =~ s/\n/ $NewLine /g }
	else { $Input =~ s/\n/ /g }

	# Isolate or exclude punctuation

	if($IncPunctuation)
	{
		foreach $Item (@Punctuation)
		{
			$Pattern = quotemeta($Item);
```

```
                $Input =~ s/($Pattern)/ $1 /g;
        }
}
else
{
        foreach $Item (@Punctuation)
        {
                $Pattern = quotemeta($Item);
                $Input =~ s/$Pattern//g;
        }
}

# Discard all tags except certainty level tags

@Identifiers = &ListIdentifiers($Tag[0], $Tag[2], $Input);

foreach $GI (@Identifiers)
{
        unless(($GI eq $Certainties[0]) or ($GI eq $Certainties[1]))
        {
                $Start = $Tag[0].$GI.$Tag[2];
                $End = $Tag[1].$GI.$Tag[2];
                print "Discard ", $Start, " and ", $End, " tags.\n";
                $Discard = quotemeta($Start);
                $Input =~ s/$Discard//g;
                $Discard = quotemeta($End);
                $Input =~ s/$Discard//g;
        }
}

# Separate text according to certainty levels

$Input =~ s/($UTS)/\n$1/g;
$Input =~ s/($UTE)/$1\n/g;
$Input =~ s/($RTS)/\n$1/g;
$Input =~ s/($RTE)/$1\n/g;

# Separate division marker components

$Input =~ s/($DMO.*?$DMC)/\n$1\n/g;

# Split separated lines into list

@List = split(/\n/, $Input);

# Specify certainty of each text character

$Output = '';

foreach $Line (@List)
{
        # Skip blank lines

        if($Line eq '') { next }

        # Enter division markers

        elsif($Line =~ /$DMO.*?$DMC/)
        { $Output = $Output.$Line."\n" }

        # Handle uncertain text

        elsif($Line =~ /$UTS(.*)$UTE/)
        {
                $Line = $1;
                $Line =~ s/(.)/$1\t$Uncertain\n/g;
                $Output = $Output.$Line;
        }

        # Handle reconstructed text
```

```
	elsif($Line =~ /$RTS(.*)$RTE/)
	{
		$Line = $1;
		$Line =~ s/(.)/$1\t$Reconstructed\n/g;
		$Output = $Output.$Line;
	}

	# Handle certain text

	else
	{
		$Line =~ s/(.)/$1\t1.00\n/g;
		$Output = $Output.$Line;
	}
}

# Join div. marker components

$Output =~ s/($DMC)\n($DMO)/$1$2/g;

# Split output into list

@List = split(/\n/, $Output);

# Build output

$Output = '';
$Characters = '';
$Certainty = 0;

foreach $Item (@List)
{
	$Size = length($Characters);

	# Add character to character list and add certainty
	# to sum of certainties if character encountered

	if($Item =~ /([^ ])\t(.+)/)
	{
		$Characters = $Characters.$1;
		$Certainty += $2;
	}

	# Add characters and certainty to output if space
	# encountered and characters accumulated.

	elsif(($Item =~ /^ /) and ($Size > 0))
	{
		$Certainty = $Certainty/$Size;
		$Item = sprintf "%s\t%.3f\n", $Characters, $Certainty;
		$Output = $Output.$Item;

		# Reset characters and certainty

		$Characters = '';
		$Certainty = 0;
	}

	# Add div. marker to output if encountered.

	elsif($Item =~ /$DMO.*$DMC/)
	{ $Output = $Output.$Item."\n" }

	# Skip if space encountered and no characters accumulated.

	elsif(($Item =~ /^ /) and ($Size == 0)) { next }

	else { die "Error: Incorrect input format: $Item\n" }
}
```

```
	# Check for remaining cert. tags

	foreach $Cert (@Certainties)
	{
		$Find = quotemeta($Tag[0].$Cert.$Tag[2]);

		if($Output =~ /($Find.*)/)
		{ die "Error: Unprocessed certainty tag: $1" }

		$Find = quotemeta($Tag[1].$Cert.$Tag[2]);

		if($Output =~ /(.*$Find)/)
		{ die "Error: Unprocessed certainty tag: $1" }
	}


	# Make output file name

	$OutputName = $InputName;

	# Open output file for writing

	open(OUTPUT, '>'.$OutputPath.$OutputName) or die
			"Can't write to file $OutputName in $OutputDirectory";

	print "->  $OutputPath$OutputName\n";

	# Write output to file

	print OUTPUT $Output;

	# Close file

	close(OUTPUT);
}

# Print end message

$Duration = time - $^T;

print "\nFormat finished in $Duration seconds.\n";

sub ListIdentifiers
{
	# Make list of generic identifiers from start tags

	my($Open) = quotemeta($_[0]);
	my($Close) = quotemeta($_[1]);
	my($Input) = $_[2];
	my(@Identifiers) = ();

	push(@Identifiers, $Input =~ /$Open(\w*?)$Close/g);

	# Sort and compact list

	@Identifiers = sort @Identifiers;

	my($Identifier);
	my($Previous) = '';
	my(@List) = ();

	foreach $Identifier (@Identifiers)
	{
		if($Identifier ne $Previous)
		{ push(@List, $Identifier) }

		$Previous = $Identifier;
	}
```

```
	return @List;
}
```

Normalise

```
#!/usr/local/bin/perl -w

# Normalise.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, table, output)

$InputDirectory = 'Formatted';
$TableDirectory = 'Equivalents';
$OutputDirectory = 'Normalised';

# File (table)

$TableName = 'Hebrews';

# Division marker delimiters

@Division = (
	'<',	# Open delimiter
	'>'		# Close delimiter
	);

# Newline character

$NewLine = '|';

# Special characters to be treated as alphabetical

@Alphabetical = ( "?", "°" );

# END SPECIFICATIONS

# Print introduction

print "Normalise:\n\n";

print "This program normalises witness files according to a\n";
print "specified equivalents table.\n\n";

print "Notes:\n\n";

print "(1) Witness files must be properly formatted.\n";
print " (The Format utility does this.)\n";
print "(2) Each record (i.e. row) of the equivalents table must\n";
print " have the following format:\n";
print " Pattern TAB Replacement ... NEWLINE\n";
print " The table may contain other tab-separated fields.\n";
print "(3) Patterns and replacements may be space-separated\n";
print " phrases.\n";
print "(4) Non-alphabetical items in the pattern or replacement\n";
print " will be isolated from preceding text.\n";
print "(5) Ensure that special characters to be treated as\n";
print " alphabetical are specified.\n";
print "(6) The certainty of each replacement phrase item is the\n";
print " least of the pattern phrase items\' certainties.\n";
print "(7) Normalised witness file records are output in the\n";
print " same format as input witness file records:\n";
```

```
print " Item TAB Certainty NEWLINE\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input witness files directory: $InputDirectory.\n";
print "Equivalents table directory: $TableDirectory.\n";
print "Equivalents table name: $TableName.\n";
print "Output witness files directory: $OutputDirectory.\n\n";

print "Division markers should look like this:\n",
      "$Division[0]DIVISION MARKER$Division[1]\n\n";

print "Newline character:\n$NewLine\n\n";
if($NewLine eq '')
{ die "Error: The newline character has not been specified" }

print "Special characters to be treated as alphabetical:\n";
foreach $Item (@Alphabetical) { print "$Item\n" }

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$TablePath = $Dir.$TableDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
      $InputPath = '.'.$InputPath;
      $TablePath = '.'.$TablePath;
      $OutputPath = '.'.$OutputPath;
}

# Open input directory

opendir(INDIR, $InputDirectory) or die
      "Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Make output directory

mkdir($OutputDirectory, 0770);

# Make patterns for regular expressions

$Special = quotemeta(join("", @Alphabetical));
$DMO = quotemeta($Division[0]);
$DMC = quotemeta($Division[1]);

# Open table file for reading

print "\nRead equivalents table\n\n";
```

```
open(TABLE, '<'.$TablePath.$TableName) or die
"Error: Can't open table file $TablePath$TableName";

@Table = <TABLE>;

# Close table file

close(TABLE);

# Remove newline from each record

chomp(@Table);

# Convert table file to arrays

@Simple = ();       # For simple substitutions
@Complex = ();      # For complex substitutions

foreach $Record (@Table)
{
       @Fields = split(/\t/, $Record);

       if(@Fields < 2)
       { die "Error: Pattern or replacement missing: $Record" }

       $Pattern = $Fields[0];
       $Replacement = $Fields[1];

       # Space out non-alphabetical characters

       $Pattern =~
              s/([a-zA-Z$Special])([^a-zA-Z$Special])/$1 $2 /g;
       $Replacement =~
              s/([a-zA-Z$Special])([^a-zA-Z$Special])/$1 $2 /g;

       # Discard multiple spaces

       $Pattern =~ tr/ / /s;
       $Replacement =~ tr/ / /s;

       # Add to appropriate array

       if(($Pattern !~ / /) and ($Replacement !~ / /))
       { push(@Simple, $Pattern, $Replacement) }
       else
       { push(@Complex, $Pattern, $Replacement) }
}

# NORMALISE TRANSCRIPTIONS

print "Normalise transcriptions:\n\n";

foreach $InputName (@Contents)
{
       # Read input file

       open(INPUT, '<'.$InputPath.$InputName) or die
              "Can't open input file $InputPath$InputName";

       print "$InputPath$InputName\n";

       @InputList = <INPUT>;

       # Close file

       close(INPUT);

       # Check format

       unless($InputList[0] =~ /^$DMO.+$DMC\n$/)
```

```
{ die "Error: First item not a div. marker: $InputList[0]" }

unless($InputList[1] =~ /^.+\t(.+)\n$/)
{ die "Error: Incorrect input format: $InputList[1]" }
$CertSize = length($1);

foreach $Item (@InputList)
{
	if($Item =~ /^.+\t(.+)\n$/)
	{
		unless(length($1) == $CertSize)
		{ die "Error: Differing certainty format: $1" }
	}
	elsif($Item =~ /^$DMO.+$DMC\n$/) { next }
	else { die "Error: Incorrect input format: $Item" }
}

# Convert to string

$Output = join("", @InputList);
@InputList = ();

# Add initial newline

$Output = "\n".$Output;

# Make complex substitutions

for($i = 0; $i < @Complex; $i += 2)
{
	# Initialise pattern and replacement arrays

	@Pattern = split(" ", $Complex[$i]);
	@Replacement = split(" ", $Complex[$i+1]);

	# Turn on printing switch

	$Switch = 1;

	# Locate start of pattern

	$Index = index($Output, "\n$Pattern[0]\t", 0);

	while($Index != -1)
	{
		# print "Index: $Index\n";

		$Index += 1; # Move past initial newline
		$Start = $Index;    # Set start
		$Match = 1;
		@Certainties = ();
		@NewLines = ();

		# Check for match of each item in pattern

		foreach $Item (@Pattern)
		{
			$Size = length($Item);
			$Segment = substr($Output, $Index, $Size);
			$Next = substr($Output, $Index + $Size, 1);
			# print "Item: $Item\n";
			# print "Segment: $Segment\n";
			# print "Next character: $Next\n";

			# Check for match

			if(($Segment eq $Item) and ($Next eq "\t"))
			{
				# Get certainty and add to list
```

```
                                $Cert =
                                substr($Output, $Index + $Size + 1,
$CertSize);
                                # print "Certainty: $Cert\n";
                                push(@Certainties, $Cert);

                                # Move index to first character of next
segment

                                $Index += $Size + $CertSize + 2;

                                # Handle newline character.

                                if(substr($Output, $Index, 1) eq $NewLine)
                                {
                                       push(@NewLines, $NewLine);       #
Add to list
                                       $Index += $CertSize + 3;  # Move
index
                                       if(substr($Output, $Index, 1) eq
$NewLine)
                                       { die "Error: Multiple newlines
detected" }
                                }
                                else { push(@NewLines, '') }

                                # print "Next: ",substr($Output,$Index,1),
"\n";
                        }
                        else
                        {
                                $Match = 0;
                                last;
                        }
                }

                # Perform substitution if match detected

                if($Match == 1)
                {
                        # Print substitution once

                        if($Switch)
                        {
                                # print "$Complex[$i] -> $Complex[$i+1]\n";
                                $Switch = 0; # Turn off printing switch
                        }

                        # Get least certainty

                        $Cert = $Certainties[0];
                        foreach $Item (@Certainties)
                        { if($Item < $Cert) { $Cert = $Item } }

                        # Make parts for output string

                        $First = substr($Output, 0, $Start);

                        $Middle = "";

                        foreach $Item (@Replacement)
                        {
                                # Add replacement

                                $Segment = $Item."\t".$Cert."\n";
                                $Middle = $Middle.$Segment;

                                # Add newline character if required

                                if(@NewLines)
```

```
                                {
                                        if(shift(@NewLines) eq $NewLine)
                                        {
                                                $Segment =
$NewLine."\t".$Cert."\n";
                                                $Middle = $Middle.$Segment;
                                        }
                                }
                        }

                        # Add final newline character if required

                        if(@NewLines)
                        {
                                if(pop(@NewLines) eq $NewLine)
                                {
                                        $Segment = $NewLine."\t".$Cert."\n";
                                        $Middle = $Middle.$Segment;
                                }
                        }

                        $Remainder = length($Output) - $Index;
                        $Last = substr($Output, $Index, $Remainder);

                        # Make substitution

                        $Output = $First.$Middle.$Last;

                        # Recalculate index

                        $Index = $Start + length($Middle);
                }

                # Locate next start of pattern

                $Index = index($Output, "\n$Pattern[0]\t", $Index);
        }
}

# Discard consecutive newline characters

$Pattern = quotemeta($NewLine);
$Output =~ s/$Pattern\t.+?\n$Pattern/$NewLine/g;

# Make simple substitutions

for($i = 0; $i < @Simple; $i += 2)
{
        $Pattern = quotemeta($Simple[$i]);
        $Replacement = $Simple[$i+1];
        if($Output =~ /\n$Pattern\t/)
        {
                # print "$Simple[$i] -> $Simple[$i+1]\n";
                $Output =~ s/\n$Pattern\t/\n$Replacement\t/g;
        }
}

# Remove initial newline

$Output =~ s/^\n//;

# Make output file name

$OutputName = $InputName;

# Open output file for writing

open(OUTPUT, '>'.$OutputPath.$OutputName) or die
        "Can't write to file $OutputPath$OutputDirectory";
```

```
        print "-> $OutputPath$OutputName\n\n";

        # Write output to file

        print OUTPUT $Output;

        # Close file

        close(OUTPUT);
}

# Print end message

$Duration = time - $^T;

print "Normalise finished in $Duration seconds.\n";
```

Synthesise

```
#!/usr/local/bin/perl -w

# Synthesise.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, synthesis)

$InputDirectory = 'Normalised';
$SynthDirectory = 'Synthesised';

# File (synthesis)

$SynthName = 'Synthesis';

# Division marker delimiters

@Division = (
        '<',   # Open delimiter
        '>'          # Close delimiter
        );

# END SPECIFICATIONS

# Print introduction

print "Synthesise:\n\n";

print "Synthesise a sequence from a set of witnesses so\n";
print "that each witness can be reproduced by selecting\n";
print "sequential items from the synthetic sequence.\n\n";

print "Notes:\n\n";

print "(1) Witness files must be properly formatted.\n";
print "    (The Format or Normalise utility does this.)\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }
```

```
# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$OutputPath = $Dir.$SynthDirectory.$Dir;
unless($System eq 'MAC')
{
	$InputPath = '.'.$InputPath;
	$OutputPath = '.'.$OutputPath;
}

# Open input directory

opendir(INDIR, $InputDirectory) or die
	"Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Witnesses directory: $InputDirectory.\n";
print "Synthetic sequence file: $OutputPath$SynthName.\n\n";

print "Division markers should look like this:\n",
	"$Division[0]DIVISION MARKER$Division[1]\n\n";

# SYNTHESISE PRIMARY SEQUENCE FROM WITNESSES

%Synthesis = ();
$DMO = quotemeta($Division[0]);
$DMC = quotemeta($Division[1]);

print "Synthesise:\n\n";

foreach $InputName (@Contents)
{
	# Read input file

	open(INPUT, "<$InputPath$InputName") or die
		"Can't open input file $InputPath$InputName";

	print "<- $InputPath$InputName\n";

	@InputList = <INPUT>;

	# Close file

	close(INPUT);

	# Remove newlines from input

	chomp(@InputList);

	# Convert input to list of div. markers and associated texts
	# (Input reversed to facilitate separation of divisions)

	@List = ();
	$Text = '';

	foreach $Item (reverse(@InputList))
```

```
	{
		# Update text if text encountered

		if($Item =~ /^(.+)\t.+$/) { $Text = $1."\n".$Text }

		# Update list if div. marker encountered

		elsif($Item =~ /^$DMO.+$DMC$/)
		{
			$Text =~ s/\n$//;   # Remove last newline from text
			push(@List, $Text); # Add text to list
			push(@List, $Item); # Add div. marker to  list
			$Text = '';               # Reset text
		}

		else { die "Error: Incorrect input format: $Item" }
	}

	# Reverse list again to restore original order

	@List = reverse(@List);

	# Build synthetic sequence for each witness division

	for($i = 0; $i < @List; $i += 2)
	{
		$Div = $List[$i];
		$Text = $List[$i+1];

#		print "$Div\n";

		if(exists($Synthesis{$Div}))
		{ $Synthesis{$Div} = &Synthesise($Synthesis{$Div}, $Text) }
		else { $Synthesis{$Div} = $Text }
	}
}

# Make output directory

mkdir($SynthDirectory, 0770);

# Open output file for writing

open(OUTPUT, '>'.$OutputPath.$SynthName) or die
	"Can't write to file $OutputPath$SynthDirectory";

print "\n-> $OutputPath$SynthName\n\n";

# Write output to file

foreach $Div (sort DivSort keys(%Synthesis))
{ print OUTPUT "$Div\n$Synthesis{$Div}\n"  }

# Close file

close(OUTPUT);

# Print end message

$Duration = time - $^T;

print "Synthesise finished in $Duration seconds.\n";

sub DivSort
{
	unless($a =~ /.+?(\d+).+?(\d+)/)
	{ die "Bad division format: $a" }
	$a1 = $1;
	$a2 = $2;
```

```
	unless($b =~ /.+?(\d+).+?(\d+)/)
	{ die "Bad division format: $b" }
	$b1 = $1;
	$b2 = $2;
	# print "b1: $b1 b2: $b2\n";

	if($a1 > $b1) { $i = 1 }
	if($a1 == $b1)
	{
		if($a2 > $b2) {$i = 1 }
		if($a2 == $b2) {$i = 0 }
		if($a2 < $b2) {$i = -1 }
	}
	if($a1 < $b1) { $i = -1 }

	return $i;
}

sub Synthesise
{
	# Synthesise a new sequence (NS) containing both a primary
	# sequence (PS) and secondary sequence (SS).
	# Input PS, SS strings must be formatted as follows:
	# Item NEWLINE Item NEWLINE ...
	# The NS string is returned in the same format.

	my(@PS) = split(/\n/, $_[0]);
	my(@SS) = split(/\n/, $_[1]);

	# Check whether SS contained in PS

	my($Test, $s, $p);

	$Test = 1;

	while(@SS)
	{
		# Get next SS and PS items

		$s = shift(@SS);

		unless(defined($p = shift(@PS)))
		{
			$Test = 0;
			last;
		}

#		print "s: $s\n";
#		print "p: $p\n";

		# Scroll through PS until SS item matched

		until($p eq $s)
		{
			unless(defined($p = shift(@PS)))
			{
				$Test = 0;
				last;
			}

#			print "p: $p\n";
		}

		# Exit loop if SS not in PS

		unless($Test) { last }
	}

	# Return PS if SS in PS
```

```
	if($Test) { return $_[0] }

	# Synthesise NS if SS not in PS

#	print "Synthesise\n";

	@PS = split(/\n/, $_[0]);
	@SS = split(/\n/, $_[1]);
	my(@IPS, @ISS);
	my(@NS) = ();
	my($Pattern);

	while(@SS)
	{
		# Split off initial PS (IPS) and initial SS (ISS)
		# Division is made at common item that is not first in SS.

#		print "PS: ", join(' ', @PS), "\n";
#		print "SS: ", join(' ', @SS), "\n";

		@IPS = ();
		@ISS = (shift(@SS));	# Shift first SS item to ISS

		$Test = 0;
		while(@SS)
		{
			$Find = quotemeta($SS[0]);
			if(grep(/^$Find$/, @PS))
			{
#				print "Match: $SS[0]\n";

				$Test = 1;
				until($PS[0] eq $SS[0]) { push(@IPS, shift(@PS)) }
			}
			else
			{
				$Test = 0;
				push(@ISS, shift(@SS));
			}

			if($Test) { last }
		}

		# Make IPS equal to PS if no match

		unless($Test) { @IPS = @PS }

		# Synthesise NS from ISS and IPS

#		print "IPS: ", join(' ', @IPS), "\n";
#		print "ISS: ", join(' ', @ISS), "\n";

		while(@ISS)
		{
			# If first ISS item matched in IPS, output IPS to NS
			# until match reached, then discard match.

			$Find = quotemeta($ISS[0]);
			if(grep(/^$Find$/, @IPS))
			{
				until($IPS[0] eq $ISS[0])
				{
#					print "PS -> $IPS[0]\n";
					push(@NS, shift(@IPS));
				}

				# Discard match

				shift(@IPS);
			}
```

```
                    # Shift first ISS item to NS

#                   print "SS -> $ISS[0]\n";
                    push(@NS, shift(@ISS));
              }

              # Join remaining IPS to PS

              unshift(@PS, @IPS);

              # Exit loop if no match

              unless($Test) { last }
       }

       # Output remaining IPS to NS

       while(@IPS)
       {
#             print "IPS -> $IPS[0]\n";
              push(@NS, shift(@IPS));
       }

       # Return new sequence

       return join("\n", @NS);
}
```

Matrix

```
#!/usr/local/bin/perl -w

# Matrix.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, synthesis, unwanted, output)

$InputDirectory = 'Normalised';
$SynthDirectory = 'Synthesised';
$UnwantedDirectory = 'Unwanted';
$OutputDirectory = 'Matrices';

# Files (synthesis, unwanted)

$SynthName = 'Synthesis';
$UnwantedName = 'Textual';

# Division marker delimiters

@Division = (
       '<',   # Open delimiter
       '>'           # Close delimiter
       );

# Threshold. Any item with a certainty below the
# threshold will be classified as a missing datum.

$Threshold = 0.5;

# Features to include. 0 = no, 1 = yes.
```

```
$IncLabels = 0;           # Row and column labels
$IncDivs = 0;             # Division markers
$IncSingles = 0;    # Singleton variations
$IncNoVars = 0;           # Rows with no variation
$IncMissing = 0;    # Rows with missing data

# END SPECIFICATIONS

# Print introduction

print "Matrix:\n\n";

print "Build a data matrix for each input file.\n\n";

print "Notes:\n\n";

print "(1) Each data matrix is based on a synthetic sequence\n";
print " containing all sequences found in the witness files.\n";
print " (The Synthesise utility builds this.)\n";
print " The same witnesses as used to build the synthetic\n";
print " sequence must be used here.\n";
print "(2) The synthetic sequence must be properly formatted.\n";
print " (The Synthesise utility does this.)\n";
print " Witness files must also be properly formatted.\n";
print " (Either the Format or Normalise utility does this.)\n";
print "(3) Each data matrix is restricted to the divisions for\n";
print " which its witness file (the principal witness) exists.\n";
print " Other witnesses are only included if they exist for\n";
print " all of the principal witness file divisions.\n";
print "(4) Data matrix entries have the following values:\n";
print " 1 = present\n";
print " 0 = absent\n";
print " -1 = missing data\n";
print " An item is classified as missing data if its\n";
print " certainty is below a specified threshold.\n";
print "(5) Output data matrices consist of one tab separated\n";
print " row vector per principal witness item, with one column\n";
print " per witness.\n";
print "(6) The following data matrix features may be included\n";
print " or excluded as desired:\n";
print " - row and column labels\n";
print " - division markers\n";
print " - rows with a singleton variation\n";
print " - rows with no variation\n";
print " - rows with missing data.\n";
print " Rows corresponding to items listed in a specified\n";
print " unwanted items file will also be excluded.\n";
print "(7) A separate file containing the following items is\n";
print " generated if labels and div. markers are excluded:\n";
print " - column labels with a fixed length of 8 characters\n";
print " - row labels\n";
print " - dimensions of the data matrix.\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$SynthPath = $Dir.$SynthDirectory.$Dir;
$UnwantedPath = $Dir.$UnwantedDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
       $InputPath = '.'.$InputPath;
```

```
	$SynthPath = '.'.$SynthPath;
	$UnwantedPath = '.'.$UnwantedPath;
	$OutputPath = '.'.$OutputPath;
}

# Open input directory

opendir(INDIR, $InputDirectory) or die
	"Can't open input directory $InputDirectory\n";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Make output directory

mkdir($OutputDirectory, 0770);

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input witness files directory: $InputDirectory.\n";
print "Synthetic sequence file: $SynthPath$SynthName.\n";
print "Unwanted items file: $UnwantedPath$UnwantedName.\n";
print "Output matrix files directory: $OutputDirectory.\n\n";

print "Division markers should look like this:\n";
print "$Division[0]DIVISION MARKER$Division[1]\n\n";

print "Any witness item with a certainty below $Threshold\n";
print "will be classified as a missing datum.\n\n";

unless($IncLabels and $IncDivs and $IncSingles and $IncNoVars
	and $IncMissing)
{ print "Excluded data matrix features:\n" }
unless($IncLabels) { print "column and row labels\n" }
unless($IncDivs) { print "division markers\n" }
unless($IncSingles) { print "rows with a singleton variation\n" }
unless($IncNoVars) { print "rows with no variation\n" }
unless($IncMissing) { print "rows with missing data\n" }

if(($IncLabels == 0) and ($IncDivs == 0))
{ print "\nSeparate labels files will be written out.\n" }

# Make patterns for regular expressions

$DMO = quotemeta($Division[0]);
$DMC = quotemeta($Division[1]);

# Read unwanted items file if it exists.

if(open(UNWANTED, '<'.$UnwantedPath.$UnwantedName))
{
	print "\nRows corresponding to items specified in\n",
	"$UnwantedPath$UnwantedName will also be excluded.\n";

	@Unwanted = <UNWANTED>;

	# Close file

	close(UNWANTED);

	# Remove newlines
```

```
	chomp(@Unwanted);

	# Check format

	foreach $Item (@Unwanted)
	{
		if($Item eq '')
		{ die "Error: Blank line detected" }
		elsif($Item =~ /\t/)
		{ die "Error: Tab detected: $Item" }
		elsif($Item =~ /$DMO.*$DMC/)
		{ die "Error: Division marker detected: $Item" }
	}
}
else
{
	@Unwanted = ();
	print "\nCan't find specified unwanted items file\n",
	"$UnwantedPath$UnwantedName.\n",
	"Unwanted items will not be discarded.\n";
}

# Open synthesis file for reading

print "\nRead synthesis file $SynthPath$SynthName\n";

open(SYNTH, '<'.$SynthPath.$SynthName) or die
	"Error: Can't open $SynthPath$SynthName";

@InputList = <SYNTH>;

# Close synthesis file

close(SYNTH);

# Remove newlines

chomp(@InputList);

# Check format

foreach $Item (@Unwanted)
{
	if($Item eq '') { die "Error: Blank line detected" }
	elsif($Item =~ /\t/) { die "Error: Tab detected: $Item" }
}

# Convert to string

$Input = join("\n", @InputList);

# Convert synthesis to associative array with divisions as keys

%Synthesis = ();

@InputList = &DivArray($DMO, $DMC, $Input);

for($i = 0; $i < @InputList; $i += 2)
{ $Synthesis{$InputList[$i]} = $InputList[$i+1] }

# BUILD MATRICES

print "\nBuild matrices:\n\n";

foreach $PrincipalName (@Contents)
{
	# Read principal file

	open(PRINCIPAL, '<'.$InputPath.$PrincipalName) or die
```

```
	"Can't open principal file $InputPath$PrincipalName";

print "Principal witness: $InputPath$PrincipalName\n";

@InputList = <PRINCIPAL>;

# Close file

close(PRINCIPAL);

# Remove newlines

chomp(@InputList);

# Check format

foreach $Item (@InputList)
{
	if($Item =~ /^.+\t.+$/) { next }
	elsif($Item =~ /^$DMO.+$DMC$/) { next }
	else { die "Error: Incorrect input format: $Item" }
}

# Convert to string

$Input = join("\n", @InputList);

# Make list of principal divisions

@Principal = ();

@InputList = &DivArray($DMO, $DMC, $Input);

for($i = 0; $i < @InputList; $i += 2)
{ push(@Principal, $InputList[$i]) }

# Initialise columns list

@Columns = ('R/C');

# Add row labels

foreach $Div (@Principal)
{
	# Add div. marker

	{ push(@Columns, $Div) }

	# Add div. items

	@SynthesisList = split(/\n/, $Synthesis{$Div});
	push(@Columns, @SynthesisList);
}

# Count rows

$RowCount = @Columns;

# Add column for each witness which exists for all principal
# witness divisions.

foreach $InputName (@Contents)
{
	# Read input file

	open(INPUT, '<'.$InputPath.$InputName) or die
		"Can't open input file $InputPath$InputName";

	@InputList = <INPUT>;
```

```
# Close file

close(INPUT);

# Convert to associative array with divisions as keys

%Witness = ();

$Input = join('', @InputList);

@InputList = &DivArray($DMO, $DMC, $Input);

# Enter div. marker as key and text as value

for($i = 0; $i < @InputList; $i += 2)
{ $Witness{$InputList[$i]} = $InputList[$i+1] }

# Test whether principal and witness files have
# all divisions in common

$Test = 1;   # 0 = not all common, 1 = all common

foreach $Div (@Principal)
{
       unless(exists($Witness{$Div}))
       {
              $Test = 0;
              last;
       }
}

# Make witness column if all divisions common

if($Test)
{
       print "<- $InputPath$InputName\n";

       # Add column label

       push(@Columns, $InputName);

       # Add entries for each item of each division

       foreach $Div (@Principal)
       {
              # Add blank for div. marker

              push(@Columns, '');

              # Split master and witness div. texts into lists

              @SynthesisList = split(/\n/, $Synthesis{$Div});
              @WitnessList = split(/\n/, $Witness{$Div});

              # Initialise previous certainty

              $PrevCert = 0;

              # Make column entries for witness items

              foreach $W (@WitnessList)
              {
                     # Get witness item and certainty

                     $W =~ /(.+)\t(.+)/;
                     $W = $1;
                     $Cert = $2;

                     # Set matrix entry
```

```
                                        if($PrevCert < $Threshold) { $Entry = '-1' }
                                        else { $Entry = '0' }

                                        until($W eq ($M = shift(@SynthesisList)))
                                        {
                                               # Enter 0 if master item not in
witness
                                               # or -1 if previous cert. below
threshold

                                               push(@Columns, $Entry);

                                               # Die if witness item not in master

                                               unless(defined($M))
                                               { die "Error: $W not in master\n" }
                                        }

                                        # Set matrix entry

                                        if($Cert < $Threshold) { $Entry = '-1' }
                                        else { $Entry = '1' }

                                        # Enter 1 if master item in witness
                                        # or -1 if cert. below threshold

                                        push(@Columns, $Entry);

                                        # Update previous cert.

                                        $PrevCert = $Cert;
                                 }

                                 # Add column entries for remaining master items

                                 # Set matrix entry

                                 if($PrevCert < $Threshold) { $Entry = '-1' }
                                 else { $Entry = '0' }

                                 while(@SynthesisList)
                                 {
                                        push(@Columns, $Entry);
                                        shift(@SynthesisList);
                                 }
                          }
                   }
            }

            # Make output file name

            $OutputName = $PrincipalName;

            # Open output file for writing

            open(OUTPUT, '>'.$OutputPath.$OutputName) or die
                   "Can't write to file $OutputPath$OutputDirectory";

            print "-> $OutputPath$OutputName\n";

            # Make column labels list

            @ColumnLabels = ();

            $i = 0;
            for($j = 0; $j < @Columns; $j += $RowCount)
            { push(@ColumnLabels, $Columns[$i + $j]) }

            # Write out column labels if required
```

```
if($IncLabels)
{ print OUTPUT join("\t", @ColumnLabels), "\n" }

# Shift first entry

shift(@ColumnLabels);

# Write out matrix rows as required

@RowLabels = ();

for($i = 1; $i < $RowCount; $i += 1)
{
	# Get row

	@Row = ();

	for($j = 0; $j < @Columns; $j += $RowCount)
	{ push(@Row, $Columns[$i + $j]) }

	# Skip div. marker if excluded

	unless($IncDivs)
	{ if($Row[0] =~ /^$DMO.+$DMC$/) { next } }

	# Skip unwanted item if excluded

	if(@Unwanted)
	{
		$Pattern = quotemeta($Row[0]);
		if(grep(/^$Pattern$/, @Unwanted))
		{ next }
	}

	# Shift row label out of row

	$RowLabel = shift(@Row);

	# Count witnesses with item absent (coded by 0)

	$Absent = grep(/0/, @Row);

	# Count witnesses with item present (coded by 1)

	$Present = grep(/^1/, @Row);

	# Count witnesses with missing datum (coded by -1)

	$Missing = grep(/^-1/, @Row);

	# Count witnesses which could have item

	$Possible = $Present + $Missing;

	# Skip singleton if excluded

	unless($IncSingles)
	{ if(($Absent == 1) or ($Possible == 1)) { next } }

	# Skip rows with no variation if excluded

	unless($IncNoVars)
	{ if(($Absent == 0) or ($Possible == 0)) { next } }

	# Skip rows with missing data if excluded

	unless($IncMissing)
	{ if($Missing) { next } }

	# Add row label to row labels list
```

```
                push(@RowLabels, $RowLabel);

                # Write out row label if required

                if($IncLabels) { print OUTPUT $RowLabel, "\t" }

                # Write out rest of row

                print OUTPUT join("\t", @Row), "\n";
        }

        # Close file

        close(OUTPUT);

        # Write out labels file if required

        if(($IncLabels == 0) and ($IncDivs == 0))
        {
                # Make labels file name

                $OutputName = $PrincipalName."_labels";

                # Open labels file for writing

                open(LABELS, '>'.$OutputPath.$OutputName) or die
                "Can't write to file $OutputPath$OutputName";

                print "-> $OutputPath$OutputName\n";

                # Format and write out column labels.
                # Each label has 8 characters.
                # There is a maximum of 10 labels per line.

                $i = 0;
                foreach $Item (@ColumnLabels)
                {
                        $Remainder = $i % 10;
                        if(($i) and ($Remainder == 0)) { print LABELS "\n" }
                        printf LABELS "%-8s", $Item;
                        $i += 1;
                }

                # Write out row labels

                print LABELS "\n\nRow Labels:\n", join("\t", @RowLabels);

                # Add matrix dimensions

                $C = @ColumnLabels;
                $R = @RowLabels;

                print LABELS
                "\n\nMatrix dimensions: $R rows x $C columns.\n";

                # Close file

                close(LABELS);
        }

        print "\n";
}

# Print end message

$Duration = time - $^T;

print "Matrix finished in $Duration seconds.\n";
```

```
sub DivArray
{
	# Convert input to list of div. markers and associated texts.
	# First two items of input list are div. markers with
	# metacharacters quoted.
	# Input assumed to be list of items separated by newlines.
	# Each div. marker must be first item on its line.
	# Extraneous characters following div. marker are returned.

	my($DMO) = $_[0];
	my($DMC) = $_[1];
	my(@InputList) = split(/\n/, $_[2]);
	my($Item);
	my(@List) = ();
	my($Text) = '';

	# (Input reversed to facilitate separation of divisions)

	foreach $Item (reverse(@InputList))
	{
		# Update list if div. marker encountered

		if($Item =~ /^$DMO.+$DMC$/)
		{
			$Text =~ s/\n$//;   # Remove last newline from text
			push(@List, $Text); # Add text to list
			push(@List, $Item); # Add div. marker to list
			$Text = '';               # Reset text
		}

		# Otherwise update text

		else { $Text = $Item."\n".$Text }
	}

	# Reverse list again to restore original order and return

	return reverse(@List);
}
```

Report

```
#!/usr/local/bin/perl -w

# Report.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, synthesis, output)

$InputDirectory = 'Normalised';
$SynthDirectory = 'Synthesised';
$OutputDirectory = 'Report';

# Files (synthesis, output)

$SynthName = 'Synthesis';
$OutputName = 'Report';

# Division marker delimiters

@Division = (
	'<',   # Open delimiter
```

```
	'>'		# Close delimiter
	);

# Threshold. Any item with a certainty below the
# threshold will be classified as a missing datum.

$Threshold = 0.5;

# END SPECIFICATIONS

# Print introduction

print "Report:\n\n";

print "Produce a concise report of input texts.\n\n";

print "Notes:\n\n";

print "(1) The report is based on a synthetic sequence which\n";
print "    contains every sequence of items found in the\n";
print "    input texts.\n";
print "(2) Entries have the following values:\n";
print "    * = present\n";
print "    - = absent\n";
print "    ? = missing data\n";
print "    An item is classified as missing data if its\n";
print "    certainty is below a specified threshold.\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$SynthPath = $Dir.$SynthDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
	$InputPath = '.'.$InputPath;
	$SynthPath = '.'.$SynthPath;
	$OutputPath = '.'.$OutputPath;
}

# Open input directory

opendir(INDIR, $InputDirectory) or die
	"Can't open input directory $InputDirectory\n";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Make output directory

mkdir($OutputDirectory, 0770);

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";
```

```
print "Input directory: $InputDirectory.\n";
print "Synthetic sequence file: $SynthPath$SynthName.\n";
print "Output file: $OutputPath$OutputName.\n\n";

print "Division markers should look like this:\n";
print "$Division[0]DIVISION MARKER$Division[1]\n\n";

print "Any items with a certainty below $Threshold\n";
print "will be classified as missing data.\n\n";

# Make patterns for regular expressions

$DMO = quotemeta($Division[0]);
$DMC = quotemeta($Division[1]);

# Read synthetic sequence file

print "Read synthetic sequence file $SynthPath$SynthName\n\n";

open(SYNTH, '<'.$SynthPath.$SynthName) or die
       "Error: Can't open $SynthPath$SynthName";

@Input = <SYNTH>;

# Close file

close(SYNTH);

# Remove newlines

chomp(@Input);

# Check format

foreach $Line (@Input)
{
       if($Line eq '') { die "Error: Blank line detected" }
       elsif($Line =~ /\t/) { die "Error: Tab detected: $Line" }
}

# Convert to string

$String = join("\n", @Input);

# Convert to synthetic sequence associative array.
# Div. is key, div. + text is value.

%Synth = ();

@Input = &DivArray($DMO, $DMC, $String);

for($i = 0; $i < @Input; $i += 2)
{ $Synth{$Input[$i]} = $Input[$i+1] }

# Initialise report

%Report = ();

# MAKE REPORT

print "Make report:\n\n";

foreach $File (@Contents)
{
       # Read input file

       open(IN, "<$InputPath$File") or die
       "Can't read $InputPath$File";
```

```
	print "<- $InputPath$File\n";

	@Input = <IN>;

	# Close file

	close(IN);

	# Remove newlines

	chomp(@Input);

	# Check format

	foreach $Line (@Input)
	{
		if($Line =~ /^.+\t.+$/) { next }
		elsif($Line =~ /^$DMO.+$DMC$/) { next }
		else { die "Error: Incorrect input format: $Line" }
	}

	# Convert to string

	$String = join("\n", @Input);

	# Make list of divisions and texts

	@List = &DivArray($DMO, $DMC, $String);

	# Report text for each div.

	while(@List)
	{
		$Div = shift(@List);
#		print "$Div\n";

		# Get synthetic sequence for div.

		$MS = $Synth{$Div};

		# Get witness sequence for div.

		$WS = shift(@List);

		# Make report record

		$Record = &MakeRecord($MS, $WS, $Threshold, $Div);

		# Add record to report

		if(exists($Report{$Div}))
		{ $Report{$Div} = $Report{$Div}.$File."\n".$Record."\n" }
		else
		{ $Report{$Div} = $File."\n".$Record."\n" }
	}
}

# Open output file

open(OUTPUT, ">$OutputPath$OutputName") or die
"Can't write $OutputPath$OutputName";

# Write output

print "\n-> $OutputPath$OutputName\n";

foreach $Div (sort DivSort keys(%Report))
{
	# Get list of elements and matrix dimensions.
```

```
	@List = ($Div);                                                 #
Add div. marker
	push(@List, split(/\n/, $Synth{$Div}));             # Add items
	$Rows = @List;                                                  #
Number of rows
	push(@List, split(/\n/, $Report{$Div}));      # Add elements
	$Total = @List;

	# Write out rows

	for($i = 0; $i < $Rows; $i += 1)
	{
		@Row = ();

		for($j = 0; $j < $Total; $j += $Rows)
		{ push(@Row, $List[$i+$j]) }

		# Write out row unless all entries absent

		$Absent = grep(/^-/, @Row);

		unless($Absent == @Row - 1)
		{ print OUTPUT join("\t", @Row), "\n" }
	}
}

# Print end message

$Duration = time - $^T;

print "\nReport finished in $Duration seconds.\n";

sub DivArray
{
	# Convert input to list of div. markers and associated texts.
	# First two items of input list must be div. markers with
	# metacharacters quoted.
	# Input assumed to be list of items separated by newlines.
	# Each div. marker must be only item on its line.

	my($DMO) = $_[0];
	my($DMC) = $_[1];
	my(@Input) = split(/\n/, $_[2]);
	my($Item);
	my(@List) = ();
	my($Text) = '';

	# (Input reversed to facilitate separation of divisions)

	foreach $Item (reverse(@Input))
	{
		# Update list if div. marker encountered

		if($Item =~ /^$DMO.+$DMC$/)
		{
			$Text =~ s/\n$//;   # Remove last newline from text
			push(@List, $Text); # Add text to list
			push(@List, $Item); # Add div. marker to  list
			$Text = '';               # Reset text
		}

		# Otherwise update text

		else { $Text = $Item."\n".$Text }
	}

	# Reverse list again to restore original order and return

	return reverse(@List);
}
```

```
sub MakeRecord
{
	# Make a record from a primary and secondary sequence.
	# The primary sequence string must have the following format:
	# Item NEWLINE Item NEWLINE ...
	# The secondary sequence string must have the following format:
	# Item TAB Certainty NEWLINE Item TAB Certainty NEWLINE ...
	# The primary sequence forms the basis of the record.
	# Entries have the following values:
	# * = present
	# - = absent
	# ? = missing data (i.e. certainty below threshold).

	my(@PS) = split(/\n/, $_[0]);
	my(@SS) = split(/\n/, $_[1]);
	my($Threshold) = $_[2];
	my($Div) = $_[3];
	my($p, $Line, $s, $Cert);
	my($PrevCert) = 0;
	my(@List) = ();

#	$Cert = @PS;
#	print $Cert, "\t", join("\t", @PS), "\n";

	while (@SS)
	{
		# Initialise primary item

		unless(defined($p = shift(@PS)))
		{ die "Error: Primary sequence insufficient at $Div" }

		# Get secondary item and cert.

		$Line = shift(@SS);
		unless($Line =~ /^(.+)\t(.+)$/)
		{ die "Error: Incorrect input format: $Line" }
		$s = $1;
		$Cert = $2;

		# Set entry

		if($PrevCert < $Threshold) { $Entry = '?' }
		else {  $Entry = '-' }

		until($p eq $s)
		{
			# Enter - if absent
			# or ? if prev. cert. below threshold

			push(@List, $Entry);

			unless(defined($p = shift(@PS)))
			{ die "Error: Primary sequence insufficient at $Div" }
		}

		# Set entry

		if($Cert < $Threshold) { $Entry = '?' }
		else {  $Entry = '*' }

		# Enter * if present
		# or ? if cert. below threshold

		push(@List, $Entry);

		# Update prev. cert.

		$PrevCert = $Cert;
	}
```

```
        # Complete record

        # Set entry

        if($PrevCert < $Threshold) { $Entry = '?' }
        else {  $Entry = '-' }

        while(@PS)
        {
                push(@List, $Entry);
                shift(@PS);
        }

        # Return record

#       $Cert = @List;
#       print $Cert, "\t", join("\t", @List), "\n";

        return join("\n", @List);
}

sub DivSort
{
        # Sort div. markers

        my($a1, $a2, $b1, $b2, $i);

        unless($a =~ /.+?(\d+).+?(\d+)/)
        { die "Bad division format: $a" }
        $a1 = $1;
        $a2 = $2;

        unless($b =~ /.+?(\d+).+?(\d+)/)
        { die "Bad division format: $b" }
        $b1 = $1;
        $b2 = $2;

        if($a1 > $b1) { $i = 1 }
        if($a1 == $b1)
        {
                if($a2 > $b2) {$i = 1 }
                if($a2 == $b2) {$i = 0 }
                if($a2 < $b2) {$i = -1 }
        }
        if($a1 < $b1) { $i = -1 }

        return $i;
}
```

Utility programs

Check

```
#!/usr/local/bin/perl -w

# Check.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directory (input)
```

```
$InputDirectory = 'Constructed';

# Division marker delimiters

@Division = (
       '<',   # Open delimiter
       '>'          # Close delimiter
       );

# Tag delimiters

@Tag = (
       '[',   # Start-tag open delimiter
       '[/',  # End-tag open delimiter
       ']'          # Tag close delimiter
       );

# END SPECIFICATIONS

# Print intoduction

print "Check: Detect tag errors.\n\n";

print "Notes:\n\n";

print "Generic identifiers must be alphanumeric.\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input directory: $InputDirectory.\n\n";

# Make division marker components

$DMO = $Division[0];      # Division Marker Open
$DMC = $Division[1];      # Division Marker Close

print "Division markers should look like this:\n",
       $DMO, "DIVISION MARKER", $DMC, "\n\n";

# Make tag components

$TSO = $Tag[0];     # Tag Start Open
$TEO = $Tag[1];     # Tag End Open
$TC = $Tag[2];      # Tag Close

print "Tags should look like this (gi = generic identifier):\n",
       $TSO, "gi", $TC, "TAGGED TEXT", $TEO, "gi", $TC, "\n\n";

# Make patterns for regular expressions

$DMO = quotemeta($DMO);
$DMC = quotemeta($DMC);
$TSO = quotemeta($TSO);
$TEO = quotemeta($TEO);
$TC = quotemeta($TC);

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make path to directory
```

```
$InputPath = $Dir.$InputDirectory.$Dir;
unless($System eq 'MAC') { $InputPath = '.'.$InputPath }

# Open input directory

opendir(INDIR, $InputDirectory) or die
       "Can't open input directory $InputDirectory";

# Make directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# CHECK TRANSCRIPTIONS

print "Check transcriptions:\n";

foreach $InputName (@Contents)
{
       # Open input file for reading

       open(INPUT, '<'.$InputPath.$InputName) or die
              "Can't open input file $InputPath$InputName\n";

       print "$InputPath$InputName\n";

       # Make input into one string

       @InputList = <INPUT>;
       $Input = join('', @InputList);

       # Close file

       close(INPUT);

       # Make list of generic identifiers

       @Identifiers = &ListIdentifiers($Input);

       # Check input

       foreach $GI (@Identifiers)
       {
              # Check for block markers within matching tags

              while($Input =~ /$TSO$GI$TC([\S\W]*?)$TEO$GI$TC/g)
              {
                     $Check = $1;
                     if($Check =~ /$DMO.*?$DMC/)
                     { die "Block marker within $GI tags: $Check\n" }
              }

              # Remove matching start and end tags

              $Input =~ s/$TSO$GI$TC([\S\W]*?)$TEO$GI$TC/$1/g;
       }

       # Check for remaining tags

       if($Input =~ /([\S\W]{0,20}[$TSO$TEO$TC][\S\W]{0,20})/g)
       { die "Tag error: $1\n" }
}

# Print end message

$Duration = time - $^T;
```

```
print "\nCheck finished in $Duration seconds.\n";

sub ListIdentifiers # Make compact list of generic identifiers
{
	# Make list of generic identifiers from start tags

	my(@Identifiers) = ();

	push(@Identifiers, $_[0] =~ /$TSO(\w*?)$TC/g);

	# Sort and compact list

	@Identifiers = sort @Identifiers;

	my($Identifier) = '';
	my($Previous) = '';
	my(@List) = ();

	foreach $Identifier (@Identifiers)
	{
		if($Identifier ne $Previous)
		{ push(@List, $Identifier) }

		$Previous = $Identifier;
	}

	return @List;
}
```

Punctuation

```
#!/usr/local/bin/perl -w

# Punctuation.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directory (input)

$InputDirectory = 'Transcribed';

# Division marker delimiters

@Division = (
	'<',	# Open delimiter
	'>'		# Close delimiter
	);

# Tag delimiters

@Tag = (
	'[',	# Start-tag open delimiter
	'[/',	# End-tag open delimiter
	']'		# Tag close delimiter
	);

# Concatenation marker

$Concatenation = '=';

# END SPECIFICATIONS
```

```
# Print introduction

print "Punctuation: List possible punctuation marks.\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input directory: $InputDirectory.\n\n";

# Make division marker components

$DMO = $Division[0];      # Division Marker Open
$DMC = $Division[1];      # Division Marker Close

print "Division markers should look like this:\n",
      $DMO, "DIVISION MARKER", $DMC, "\n\n";

# Make tag components

$TSO = $Tag[0];     # Tag Start Open
$TEO = $Tag[1];     # Tag End Open
$TC = $Tag[2];      # Tag Close

print "Tags should look like this (gi = generic identifier):\n",
      $TSO, "gi", $TC, "TAGGED TEXT", $TEO, "gi", $TC, "\n\n";

print "Concatenation marker:\n",
      $Concatenation, "\n\n";

# Make patterns for regular expressions

$DMO = quotemeta($DMO);
$DMC = quotemeta($DMC);
$TSO = quotemeta($TSO);
$TEO = quotemeta($TEO);
$TC = quotemeta($TC);
$CON = quotemeta($Concatenation);

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make path to directory

$InputPath = $Dir.$InputDirectory.$Dir;
unless($System eq 'MAC') { $InputPath = '.'.$InputPath }

# Open input directory

opendir(INDIR, $InputDirectory) or die
      "Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# LIST POSSIBLE PUNCTUATION

print "Search for possible punctuation:\n\n";
```

```
# Initialise punctuation array

@Punctuation = ();

foreach $InputName (@Contents)
{
	# Open input file for reading

	open(INPUT, '<'.$InputPath.$InputName) or die
		"Can't open input file $InputPath$InputName\n";

	print $InputPath, $InputName, "\n";

	# Make transcription into one string

	@InputList = <INPUT>;
	$Input = join('', @InputList);

	# Close file

	close(INPUT);

	# Make output string

	$Output = $Input;

	# Make punctuation string

	while($Output =~ /(\W)\s/g)
	{
		$Match = quotemeta($1);

		if($Match !~ /[$DMC$TC$CON\s]/)
		{
			if(not grep(/$Match/, @Punctuation))
			{
				push(@Punctuation, $1);
				printf("%s : %d\n", $1, ord($1));
			}
		}
	}
}

# List possible punctuation

@Punctuation = sort @Punctuation;

print "\nPossible punctuation marks and ASCII values:\n";

foreach $Item (@Punctuation)
{ printf("%s : %d\n", $Item, ord($Item)) }

# Print end message

$Duration = time - $^T;

print "\nPunctuation finished in $Duration seconds.\n";
```

Tabulate

```
#!/usr/local/bin/perl -w

# Tabulate.pl

# SPECIFICATIONS
```

```
# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, output)

$InputDirectory = 'Formatted';
$OutputDirectory = 'Equivalents';

# Files (standard text, output)

$Standard = 'UBS4';
$OutputName = 'Rudimentary';

# Division marker delimiters

@Division = (
       '<',   # Open delimiter
       '>'          # Close delimiter
       );

# END SPECIFICATIONS

# Print introduction

print "Tabulate:\n\n";

print "This program builds a rudimentary equivalents table:\n";
print "- list items found in witness files\n";
print "- delete items found in specified standard text\n";
print "- add adjacent list of duplicates\n";
print "- output item, duplicate, blank, and location.\n\n";

print "Notes:\n\n";

print "(1) Item refers to a division marker, a word, a newline\n";
print "    character, or a punctuation mark.\n";
print "(2) Witness files must be properly formatted.\n";
print "    (The Format utility does this.)\n";
print "(3) Location is that of the first occurrence encountered.\n";
print "(4) Output is formatted as follows:\n";
print "    Item TAB Duplicate TAB Blank TAB Location NEWLINE\n";
print "    The blank field is for descriptive notes.\n\n";

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input directory: $InputDirectory\n";
print "Standard text: $Standard\n";
print "Output directory: $OutputDirectory\n";
print "Table name: $OutputName\n\n";

print "Division markers should look like this:\n",
       "$Division[0]DIVISION MARKER$Division[1]\n\n";

# Make output directory

mkdir($OutputDirectory, 0770);

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }
```

```
# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
	$InputPath = '.'.$InputPath;
	$OutputPath = '.'.$OutputPath;
}

# BUILD EQUIVALENTS TABLE

print "Build equivalents table:\n";

# Open input directory

opendir(INDIR, $InputDirectory) or die
	"Error: Can't open input directory $InputDirectory.\n";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from UNIX directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Initialise

$DMO = quotemeta($Division[0]);
$DMC = quotemeta($Division[1]);
%Array = ();

# ADD ITEMS

foreach $InputName (@Contents)
{
	# Skip if standard

	if($InputName eq $Standard) { next }

	# Open input file for reading

	open(INPUT, '<'.$InputPath.$InputName) or die
		"Error: Can't open input file $InputPath$InputName\n";

	print "<- $InputPath$InputName\n";

	while(<INPUT>)
	{
		# Set location

		if($_ =~ /^($DMO.*$DMC)\n$/)
		{ $Location = $InputName." ".$1 }

		# Add item to array if not previously encountered

		elsif($_ =~ /^(.+)\t.+\n$/)
		{ unless(exists($Array{$1})) { $Array{$1} = $Location } }

		else { die "Error: Incorrect input format: $_.\n" }
	}

	# Close input file

	close(INPUT);
}

# DELETE STANDARD ITEMS
```

```
# Open standard file for reading

$InputName = $Standard;

open(INPUT, '<'.$InputPath.$InputName) or die
       "Error: Can't open standard file $InputName.\n";

print "Delete standard items found in $InputName\n";

while(<INPUT>)
{
       if(($_ =~ /($DMO.+$DMC)\n/) or ($_ =~ /(.+)\t.+\n/))
       { delete($Array{$1}) }
       else
       { die "Error: Incorrect input format: $_.\n" }
}

# Close file

close(INPUT);

# Open output file for writing if safe to do so

if(-e $OutputPath.$OutputName)
{ die "Error: Output file $OutputName already exists.\n" }

open(OUTPUT, '>'.$OutputPath.$OutputName) or die
       "Error: Can't write to output file $OutputPath$OutputName.\n";
print "->  $OutputPath$OutputName\n";

# Write output file

foreach $Item (sort(keys(%Array)))
{ print OUTPUT "$Item\t$Item\t\t$Array{$Item}\n" }

# Close file

close(OUTPUT);

# Print end message

$Duration = time - $^T;

print "\nTabulate finished in $Duration seconds.\n";
```

LSI

```
#!/usr/local/bin/perl -w

# LSI.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (table, list)

$TableDirectory = 'Equivalents';
$ListDirectory = 'Unwanted';

# Files (table, list)

$TableName = 'Hebrews';
$ListName = 'PuncLev';
```

```
# Data field number

$Data = 1;

# Descriptive field number

$Desc = 1;

# Key words

@Descriptions = (".");

# Mode (1 = positive; -1 = negative)

$Mode = -1;

# END SPECIFICATIONS

# Print introduction

print "List Specified Items:\n\n";

print "This program lists contents of a specified data field for\n";
print "each record of a table in which key words are matched in a\n";
print "specified descriptive field of the same record.\n\n";

print "Notes:\n\n";

print "(1) This program has a positive and negative mode:\n";
print "    (+) List data field if described by key words\n";
print "    (-) List data field unless described by key words.\n";
print "(2) Key word matching is case-sensitive.\n";
print "(3) Table records must have the following format:\n";
print "    Field 1 TAB Field 2 TAB Field 3 TAB ... NEWLINE\n";
print "    The output list contains one item per line.\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known" }

# Make paths to directories

$TablePath = $Dir.$TableDirectory.$Dir;
$ListPath = $Dir.$ListDirectory.$Dir;
unless($System eq 'MAC')
{
       $TablePath = '.'.$TablePath;
       $ListPath = '.'.$ListPath;
}

# Make list directory

mkdir($ListDirectory, 0770);

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input table: $TablePath$TableName\n";
print "Output list: $ListPath$ListName\n\n";

print "Data field number: $Data.\n";
print "Descriptive field number: $Desc.\n\n";
```

```
print "Key words:\n";
foreach $Item (@Descriptions) { print "$Item\n" }

print "\nMode:\n";
if($Mode == 1)
{ print "(+) List data field if described by key words.\n" }
elsif($Mode == -1)
{ print "(-) List data field unless described by key words.\n" }
else { die "Error: Incorrect mode specification" }

# Read table file

open(INPUT, '<'.$TablePath.$TableName) or die
"Error: Can't open table file $TablePath$TableName";

@Table = <INPUT>;

# Close table file

close(INPUT);

# Remove newlines

chomp(@Table);

# Check for enough fields

foreach $Record (@Table)
{
	@Fields = split(/\t/, $Record);
	if((@Fields < $Data) or (@Fields < $Desc))
	{ die "Error: Not enough table fields: $Record" }
}

# BUILD LIST

print "\nBuild list:\n";

@List = ();
@New = @Table;
@Table = ();
$Data -= 1;
$Desc -= 1;

foreach $Description (@Descriptions)
{
	print "$Description\n";
	$Pattern = quotemeta($Description);

	@Old = @New;
	@New = ();

	while(@Old)
	{
		# Get record

		$Record = shift(@Old);

		# Split into item, equivalent, description, ...

		@Fields = split(/\t/, $Record);

		# Add item to list and discard record if match found.
		# Otherwise, keep record for next search.

		if($Fields[$Desc] =~ /$Pattern/)
		{ push(@List, $Fields[$Data]) }
		else { push(@New, $Record) }
	}
}
```

```
# Open list file for writing if safe to do so

if(-e $ListPath.$ListName)
{ die "Error: List file $ListName already exists" }

open(OUTPUT, '>'.$ListPath.$ListName) or die
       "Error: Can't write to list file $ListPath$ListName";

print "->  $ListPath$ListName\n";

# Obtain output list

if($Mode == 1)      # Positive mode
{ @List = sort(@List) }
else                # Negative mode
{
       @List = ();

       foreach $Record (@New)
       {
              @Fields = split(/\t/, $Record);
              push(@List, $Fields[$Data]);
       }

       @List = sort(@List);
}

# Write output to file

print OUTPUT join("\n", @List), "\n";

# Close file

close(OUTPUT);

# Print end message

$Duration = time - $^T;

print "\nList Specified Items finished in $Duration seconds.\n";
```

CT

```
#!/usr/local/bin/perl -w

# CT.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (data and output)

$Data = 'Data';
$Output = 'Constructed';

# Data files (standard text, variation units key, and data matrix)

$ST = 'Standard';
$VK = 'Key';
$DM = 'Matrix';

# Division markers (open and close)
```

```
$DMO = '<';
$DMC = '>';

# Reconstructed text markers (open and close)

$RTO = '[rt]';
$RTC = '[/rt]';

# Minimum number of variation units required before text is output

$Minimum = 5;

# END OF SPECIFICATIONS

# Print introduction

print "CT:\n\n";

print "Construct texts from a standard text using a variation\n";
print "units key and a multistate data matrix.\n\n";

print "Notes:\n\n";

print "(1) Texts are constructed by replacing pattern phrases\n";
print "    in the standard text with phrases from the key\n";
print "    according to the data matrix entries.\n";
print "(2) The key consists of a list of variation units.\n";
print "    Each variation unit must be formatted as follows:\n";
print "    Division label\n";
print "    Pattern\n";
print "    First variant\n";
print "    ...\n";
print "    Last variant\n";
print "(3) Variation unit division labels must be formatted as\n";
print "    follows: a.bc\n";
print "    where a is the chapter number\n";
print "          b is the verse number\n";
print "    and   c is an optional letter.\n";
print "(4) The data matrix must be formatted as follows:\n";
print "    R/C W1  W2  ... Wp\n";
print "    V1  D11 D12 ... D1p\n";
print "    V2  D21 D22 ... D2p\n";
print "    ... ... ... Dxy ...\n";
print "    Vn  Dn1 Dn2 ... Dnp\n";
print "    where Vs are variation unit division labels\n";
print "          Ws are witnesses\n";
print "    and   Ds are multistate data entries.\n";
print "    Each D is either a numeral representing a variant or\n";
print "    a negative one (-1) representing a missing datum.\n";
print "(5) Replacements are made as follows:\n";
print "    Dxy  Replacement\n";
print "    0    Pattern\n";
print "    1    First variant\n";
print "    2    Second variant\n";
print "    ...  ...\n";
print "(6) Output texts are only constructed for divisions\n";
print "    in which they are defined by the data matrix.\n";
print "    They are tagged as reconstructed text everywhere\n";
print "    except where their texts are defined.\n";
print "(7) Output texts are only constructed for witnesses\n";
print "    that are defined in at least the specified number\n";
print "    of variation units.\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }
```

```
# Make paths to directories

$DPath = $Dir.$Data.$Dir;
$OPath = $Dir.$Output.$Dir;
unless($System eq 'MAC')
{
       $DPath = ".$DPath";
       $OPath = ".$OPath";
}

# Print specifications

print "Specifications:\n\n";

print "System: $System\n\n";

print "Standard text:       $DPath$ST\n";
print "Variation units key: $DPath$VK\n";
print "Data matrix:         $DPath$DM\n\n";

print "Standard text division markers must look like this:\n";
print $DMO, "ch a", $DMC, $DMO, "v b", $DMC, "\n\n";

print "Reconstructed text will look like this:\n";
print $RTO, "RECONSTRUCTED", $RTC, "\n\n";

print "A text must be defined in at least $Minimum variation\n";
print "units before it is written to the output.\n\n";

# Make hash array of standard

print "Read standard text\n";

open(STAN, "$DPath$ST") or die "Can't open $DPath$ST";

@List = <STAN>;

close <STAN>;

$Standard = join('', @List);

$Standard =~ s/^[\S\W]*?$DMO/$DMO/;
$Standard =~ s/$DMO/\n$DMO/g;
$Standard =~ s/^\n+//;
$Standard =~ s/$DMC/$DMC\n/g;
$Standard =~ s/\n+$//;
$Standard =~ tr/\n/\n/s;

@List = split(/\n/, $Standard);

%Standard = ();

for($i = 0; $i < @List; $i += 3)
{
       $Div = $List[$i].$List[$i+1];
       unless($Div =~ /^$DMO.+?\d+$DMC$DMO.+?\d+$DMC$/)
       { die "Bad division format: $Div" }
       $Text = $List[$i+2];
       $Text =~ s/^ //;
       $Text =~ s/ $//;
       $Standard{$Div} = $RTO.$Text.$RTC;
}

# @Divs = sort DivSort (keys(%Standard));
# foreach $Div (@Divs) { print "$Div $Standard{$Div}\n" }

# Make hash array of variants

print "Read variation units key\n";
```

```
open(VAR, "$DPath$VK") or die "Can't open $DPath$VK";

@List = <VAR>;

close <VAR>;

chomp(@List);
@List = reverse(@List);

%VA = ();
@Variants = ();

foreach $Item (@List)
{
	if($Item =~ /^\d+?\.\d+?[a-z]*$/)
	{
		$VA{$Item} = [ reverse(@Variants) ];
		@Variants = ();
	}
	else
	{ push(@Variants, $Item) }
}

# @Labels = sort(keys(%VA));
# foreach $Label (@Labels)
# { print "$Label\n", join("\n", @{$VA{$Label}}), "\n" }

# Read data matrix

print "Read data matrix\n";

open(MAT, "$DPath$DM") or die "Can't open $DPath$DM";

@Matrix = <MAT>;

close <MAT>;

chomp(@Matrix);

@Names = split(/\s+/, shift(@Matrix));  # Text names
shift(@Names);      # Remove blank

@DM = ();
@Labels = ();
$Rows = 0;
$Columns = @Names;

foreach $Record (@Matrix)
{
	@Fields = split(/\s+/, $Record);
	$Label = shift(@Fields);
	push(@Labels, $Label);
	unless(@Fields == $Columns)
	{ die "Error: Wrong number of data matrix columns at $Label" }
	push(@DM, [@Fields]);
	$Rows += 1;
}

print "Dimensions: $Rows r x $Columns c\n";

# Make output directory

mkdir("$Output", 0770);

# Construct texts

print "\nConstruct texts:\n";

for($j = 0; $j < $Columns; $j += 1)
```

```
{
	print "$Names[$j]\n";

	%Text = ();
	$Counter = 0;

	for($i = 0; $i < $Rows; $i += 1)
	{
		$State = $DM[$i][$j];
		if($State eq '-1') { next }
		unless($State =~ /^\d$/)
		{ die "Error: Unrecognisable matrix entry: $State" }
		$Counter += 1;

		$Label = $Labels[$i];
		unless(defined($VA{$Label}[$State]))
		{ die "Error: State $State undefined: $Names[$j], $Label" }
		$Pattern = quotemeta($VA{$Label}[0]);
		$Replacement = "$RTC $VA{$Label}[$State] $RTO";
#		print "P: $Pattern\n";
#		print "R: $Replacement\n";

		$Label =~ /(\d+)\.(\d+)/;
		$Div = $DMO.'ch '.$1.$DMC.$DMO.'v '.$2.$DMC;

		unless(exists($Text{$Div}))
		{ $Text{$Div} = $Standard{$Div} }

		$String = $Text{$Div};
		$String =~ s/$Pattern/$Replacement/;
		$Pattern = quotemeta("$RTO$RTC");
		$String =~ s/$Pattern//;
		$Text{$Div} = $String;
	}

	if($Counter < $Minimum)
	{
		print "Less than $Minimum v.u.s\n";
		next;
	}

	print "$Counter v.u.s -> $OPath$Names[$j]\n";

	open(OUT, ">$OPath$Names[$j]") or die
	"Can't open $OPath$Names[$j]";

	foreach $Div (sort DivSort (keys(%Text)))
	{ print OUT "$Div $Text{$Div}\n" }

	close(OUT);
}

# Print end message

$Duration = time - $^T;

print "\nCT finished in $Duration seconds.\n";

sub DivSort
{
	# print "a: $a\n";
	unless($a =~ /^$DMO.+?(\d+)$DMC$DMO.+?(\d+)$DMC$/)
	{ die "Bad division format: $a" }
	$a1 = $1;
	$a2 = $2;
	# print "a1: $a1 a2: $a2\n";

	# print "b: $b\n";
	unless($b =~ /^$DMO.+?(\d+)$DMC$DMO.+?(\d+)$DMC$/)
	{ die "Bad division format: $b" }
```

```
	$b1 = $1;
	$b2 = $2;
	# print "b1: $b1 b2: $b2\n";

	if($a1 > $b1) { $i = 1 }
	if($a1 == $b1)
	{
		if($a2 > $b2) {$i = 1 }
		if($a2 == $b2) {$i = 0 }
		if($a2 < $b2) {$i = -1 }
	}
	if($a1 < $b1) { $i = -1 }

	# print "$i\n";
	return $i;
}
```

Halve

```
#!/usr/local/bin/perl -w

# Halve.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# File (input)

$Input = 'U';

# Directory (output)

$OutputDirectory = 'Halved';

# END SPECIFICATIONS

# Print introduction

print "Halve: Recursively halve the records in a file.\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make path to directory

$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC') { $OutputPath = '.'.$OutputPath }

# Make output directory

mkdir($OutputDirectory, 0770);

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input file: $Input.\n";
```

```
print "Output directory: $OutputDirectory.\n\n";

# Halve

$Old = "$Input";
print "<- $Old\n";
$New = $Old;
$Counter = 2;

while($Counter > 1)
{
	open(IN, "$New") or die "Can't read $New";
	@List = <IN>;
	close(IN);

	$Counter = 0;
	$Output = '';

	for($i = 0; $i < @List; $i += 2)
	{
		$Output = $Output.$List[$i];
		$Counter += 1;
	}

	$New = "$OutputPath$Old ($Counter)";
	print "-> $New\n";
	open(OUT, ">$New") or die "Can't write $New";
	print OUT $Output;
	close(OUT);
}

# Print end message

$Duration = time - $^T;

print "\nHalve finished in $Duration seconds.\n";
```

II

```
#!/usr/local/bin/perl -w

# LI.pl

# SPECIFICATIONS

# System (Mac, UNIX, or PC)

$System = 'Mac';

# Directories (input, output)

$InputDirectory = 'UBS4';
$OutputDirectory = 'Unwanted';

# Division marker delimiters

@Division = (
	'<',	# Open delimiter
	'>'		# Close delimiter
	);

# Certainty threshold

$Threshold = 0.051;

# END SPECIFICATIONS
```

```
# Print introduction

print "List Items:\n\n";

print "For each input file:\n";
print "- output a list of distinct items\n";
print "- count certain and uncertain items.\n\n";

print "Notes:\n\n";

print "(1) Input files must be properly formatted.\n";
print "    (The Format or Normalise utility does this.)\n";
print "(2) An item is counted as uncertain if its certainty is\n";
print "    below a specified threshold.\n\n";

# Select directory character

$System = uc($System);    # Make all upper-case
if($System eq 'MAC') { $Dir = ':' }
elsif($System eq 'UNIX') { $Dir = '/' }
elsif($System eq 'PC') { $Dir = '\\' }
else { die "Error: Directory character for $System not known\n" }

# Make paths to directories

$InputPath = $Dir.$InputDirectory.$Dir;
$OutputPath = $Dir.$OutputDirectory.$Dir;
unless($System eq 'MAC')
{
       $InputPath = '.'.$InputPath;
       $OutputPath = '.'.$OutputPath;
}

# Show specifications

print "Specifications:\n\n";

print "$System version.\n\n";

print "Input directory: $InputDirectory.\n";
print "Output directory: $OutputDirectory.\n\n";

print "Division markers should look like this:\n",
       "$Division[0]DIVISION MARKER$Division[1]\n\n";

print "Any item with a certainty below $Threshold\n";
print "will be counted as uncertain.\n\n";

# Make patterns for regular expressions

$DMO = quotemeta($Division[0]);
$DMC = quotemeta($Division[1]);

# Open input directory

opendir(INDIR, $InputDirectory) or die
       "Can't open input directory $InputDirectory";

# Make input directory contents list

@Contents = readdir(INDIR);

# Remove . and .. from directory list

unless($System eq 'MAC') { splice(@Contents, 0, 2) }

# Make output directory

mkdir($OutputDirectory, 0770);
```

```
# MAKE LISTS FOR EACH INPUT FILE

print "Make lists:\n\n";

foreach $InputName (@Contents)
{
	# Initialise

	%Array = ();
	$Certs = 0;
	$Uncerts = 0;

	# Open output file for writing

	open(OUTPUT, ">$OutputPath$InputName") or die
		"Can't write to $OutputPath$InputName";

	# Read input file

	open(INPUT, "<$InputPath$InputName") or die
		"Can't open input file $InputPath$InputName";

	print "$InputPath$InputName -> $OutputPath$InputName\n";

	@List = <INPUT>;

	# Close file

	close(INPUT);

	# Remove newlines from input

	chomp(@List);

	# List items

	foreach $Line (@List)
	{
		# Skip div. markers

		if($Line =~ /^$DMO.*$DMC$/) { next }

		# Check format

		unless($Line =~ /(.+)\t(.+)/)
		{ die "Error: Incorrect input format: $Line" }

		# Get item and certainty

		$Item = $1;
		$Cert = $2;

		# Increment counters

		if($Cert < $Threshold) { $Uncerts += 1 }
		else { $Certs += 1 }

		# Add item to array if not already there

		unless(exists($Array{$Item}))
		{ $Array{$Item} = $Item }
	}

	# Print number of items

	print "Certain: $Certs  Uncertain: $Uncerts\n";

	# Write output file

	print OUTPUT join("\n", sort(keys(%Array)));
```

```
        # Close file
        close(OUTPUT);
}
# Print end message
$Duration = time - $^T;
print "\nList Items finished in $Duration seconds.\n";
```

DATA

The four data disks contain programs and data files that have been created in the course of my research. The disks are formatted for Macintosh computers. The files and program-listings are plain text, which means that they can be transferred to other platforms. Executable files such as *prep* and *MacPerl* require a Macintosh to run. If a Macintosh is not available, *prep* will have to be recompiled from the program listing using a C compiler. PERL is available for Mac, PC, and Unix platforms. (It can be obtained from CPAN sites on the world-wide web.)

Details of my programs are given in the chapter on collation. The 'specifications' section of each program should be changed to suit the user's requirements. The programs are presently configured to use files and directories contained on the disks.

The PERL programs definitely work on Unix machines. They should work on PC's as well, but I have not confirmed this. (The platform needs to be specified in the program listing.) The following instructions apply to Macintosh computers.

In order to run *prep*, double-click on the icon and enter the transcription file name. (My transcription of P46 is included for demonstration purposes.) The PERL programs will run provided that *MacPerl* is installed. I have included a copy of *MacPerl*, along with the required licensing information, on one of the disks. To run a PERL program, double-click on the relevant icon, then select "Run X" from the Script menu, where X is the program name. The collation programs should be run sequentially, from left to right, beginning with *Filter*.

The following list describes the contents of the disks. Brief descriptions of directory contents are given in brackets.

Disk 1

Transcriptions (primary witnesses)
Supplementary (supplementary witnesses, sample data matrix)

Disk 2

Equivalents (table of equivalents)
Unwanted (lists to eliminate unwanted items from matrices)

Disk 3

Preprocess (*prep*)
Collation (my collation programs)
Utilities (my utility programs).

Disk 4

MacPerl and licence.

Some of the programs take a long time to process a full complement of transcriptions.

BIBLIOGRAPHY

Aland, Barbara and Klaus Wachtel. 1995. 'The Greek minuscule manuscripts of the New Testament'. In *The text of the New Testament in contemporary research: essays on the status quaestionis*. Festschrift in honour of Bruce M. Metzger. Studies and documents, 46. Ed. Bart D. Ehrman and Michael W. Holmes. Grand Rapids: Eerdmans, 43-60.

Aland, Barbara, Kurt Aland, Johannes Karavidopoulos, Carlo M. Martini, and Bruce M. Metzger (eds.). 1993. *Novum Testamentum Graece*. Nestle-Aland, 27th rev. ed. Stuttgart: Deutsche Bibelgesellschaft.

Aland, Barbara, Kurt Aland, Johannes Karavidopoulos, Carlo M. Martini, and Bruce M. Metzger (eds.). 1993. *The Greek New Testament*. United Bible Societies, 4th rev. ed. Stuttgart: Deutsche Bibelgesellschaft.

Aland, Kurt and Barbara. 1989. *The text of the New Testament: an introduction to the critical editions and to the theory and practice of modern textual criticism*. 2nd rev. and enlarged ed. Trans. Erroll F. Rhodes. Grand Rapids: Eerdmans.

Aland, Kurt, Matthew Black, Carlo M. Martini, Bruce M. Metzger, and Allen Wikgren (eds.). 1979. *Novum Testamentum Graece*. Nestle-Aland, 26th ed. Stuttgart. Deutsche Bibelgesellschaft.

Aland, Kurt, Matthew Black, Carlo M. Martini, Bruce M. Metzger, and Allen Wikgren (eds.). 1983. *The Greek New Testament*. United Bible Societies, 3rd corr. ed. New York: United Bible Societies.

Aland, Kurt. 1994. *Kurzgefasste Liste der Griechischen Handschriften des Neuen Testaments*. 2nd ed. Berlin: Walter de Gruyter.

The Analytical Greek lexicon. n.d. London: Samuel Bagster and Sons.

Atlas of the Roman world. 1982. Ed. Timothy Cornell and John

Matthews. Oxford: Phaidon Press.

Atlas zur Kirchengeschichte: die christlichen Kirchen in Geschichte und Gegenwart. 1987. Ed. Hubert Jedin, Kenneth Scott Latourette, and Jochen Martin. Freiburg: Herder.

Attridge, Harold, W. 1989. *The epistle to the Hebrews*. Hermeneia. Philadelphia: Fortress Press.

Barbrook, Adrian C., Christopher J. Howe, Norman Blake, and Peter M. W. Robinson. 1998. 'The phylogeny of The Canterbury Tales'. *Nature* 394 (27 August), 839.

Bartoletti V. and M. Norsa. 1951. *Pubblicazioni della Società Italiana (papyri Greci e Latini)*. Vol. 12, 209-210.

Bauer, Walter. 1979. *A Greek-English lexicon of the New Testament and other early Christian literature*. Trans. William F. Arndt and F. Wilbur Gingrich. 2nd rev. and augmented ed. Ed. F. Wilbur Gingrich and Frederick W. Danker. Chicago: University of Chicago Press.

Bearman, Gregory H. and Sheila I. Spiro. 1996. 'Archaeological applications of advanced imaging techniques'. *Biblical archaeologist* 59/1, 56-66.

Benduhn-Mertz, Annette, Gerd Mink, and Horst Bachmann. 1991. *Text und Textwert der griechischen Handschriften des Neuen Testaments*. Part 2. *Die Paulinischen Briefe*. Vol. 4. *Kolosserbrief bis Hebräerbrief*. Arbeiten zur neutestamentlichen Textforschung, 19. Ed. Kurt Aland. Berlin: Walter de Gruyter.

Bevington, Philip R. 1969. *Data reduction and error analysis for the physical sciences*. New York: McGraw-Hill.

Black, David Alan. 1988. *Linguistics for students of New Testament Greek: a survey of basic concepts and applications*. Grand Rapids: Baker Book House.

Blass, Friedrich and Albert Debrunner. 1961. *A Greek grammar of the New Testament and other early Christian literature: a translation and revision of the ninth-tenth edition incorporating supplementary notes of A. Debrunner†*. Trans. and ed. Robert W. Funk. Chicago: University of Chicago Press.

Borenstein, Philip, and Jeff Matson. 1991. *Think C: user manual*. Cupertino: Symantec.

Brooks, James A. 1991. *The New Testament text of Gregory of Nyssa*. Society of Biblical Literature: the New Testament in the Greek Fathers, 2. Ed. Gordon D. Fee. Atlanta: Scholars Press.

Bruce, F. F. 1990. *The epistle to the Hebrews*. The new international commentary on the New Testament. Rev. ed. Grand Rapids: Eerdmans.

Bruce, F. F. 1992. 'Textual problems in the epistle to the Hebrews'. In *Scribes and scripture: New Testament essays in honor of J. Harold Greenlee*. Ed. David Alan Black. Winona Lake: Eisenbrauns.

Chatfield, Christopher, and Alexander J. Collins. 1980. *Introduction to multivariate analysis*. London: Chapman and Hall.

Colwell, Ernest Cadman 1965. 'Scribal habits in early papyri: a study in the corruption of the text'. In *The Bible in modern scholarship: papers read at the 100th meeting of the Society of Biblical Literature*. Ed. J. Philip Hyatt. Nashville: Abingdon Press, 370-389.

Contemporary English Version. 1995. Canberra: The Bible Society in Australia.

Dagpunar, John. 1988. *Principles of random variate generation*. Oxford: Clarendon Press.

Dobson, John H. 1989. *Learn New Testament Greek*. Grand Rapids: Baker Book House.

Duff, Jeremy. 1998. '$\mathfrak{P}$46 and the pastorals: a misleading consensus?' *New Testament studies* 44/4, 578-590.

Duplacy, Jean. 1966. 'Histoire des manuscrits et histoire du texte du N.T.' *New Testament studies* 12/2, 124-139.

Ehrman, Bart D. 1986. *Didymus the Blind and the text of the Gospels*. Society of Biblical Literature: the New Testament in the Greek Fathers, 1. Ed. Gordon D. Fee. Atlanta: Scholars Press.

Ehrman, Bart D. 1987. 'Methodological developments in the analysis and classification of New Testament documentary evidence'. *Novum Testamentum* 29/1, 22-45.

Ehrman, Bart D. 1993. *The Orthodox corruption of scripture: the effect of early Christological controversies on the text of the New Testament*. New York: Oxford University Press.

Ehrman, Bart D. 1995. 'The text as window: New Testament manuscripts and the social history of early Christianity'. In *The text of the New Testament in contemporary research: essays on the status quaestionis*. Festschrift in honour of Bruce M. Metzger. Studies and documents, 46. Ed. Bart D. Ehrman and Michael W. Holmes. Grand Rapids: Eerdmans, 361-379.

Ellingworth, Paul. 1993. *The epistle to the Hebrews: a commentary on the Greek text*. The new international Greek Testament commentary. Grand Rapids: Eerdmans.

Elliott, J. Keith. 1972. 'When Jesus was apart from God: an examination of Hebrews 2^{9}'. *The expository times* 83/11, 339-341.

Elliott, J. Keith. 1989. *A bibliography of Greek New Testament manuscripts*. Society for New Testament Studies monograph series, 62. Cambridge: Cambridge University Press.

Emmel, Thomas C. 1976. *Population biology*. New York: Harper &

Row.

The Englishman's Greek New Testament giving the Greek text of Stephens 1550, with the various readings of the editions of Elzevir 1624, Griesbach, Lachmann, Tischendorf, Tregelles, Alford, and Wordsworth: together with an interlinear literal translation and the Authorised Version of 1611. 1896. Third ed. London: Samuel Bagster and Sons.

Epp, Eldon Jay. 1989. 'The New Testament papyrus manuscripts in historical perspective'. In *To touch the text: biblical and related studies in honor of Joseph A. Fitzmyer, S.J.* Ed. Maurya P. Horgan and Paul J. Kobelski. New York: Crossroad.

Epp, Eldon Jay. 1993a. 'Decision points in past, present, and future New Testament textual criticism'. In *Studies in the theory and method of New Testament textual criticism*. Ed. Eldon Jay Epp and Gordon D. Fee. Studies and documents, 45. Grand Rapids: Eerdmans, 141-173.

Epp, Eldon Jay. 1993b. 'The significance of the papyri for determining the nature of the New Testament text in the second century: a dynamic view of textual transmission'. In *Studies in the theory and method of New Testament textual criticism*. Ed. Eldon Jay Epp and Gordon D. Fee. Studies and documents, 45. Grand Rapids: Eerdmans, 274-297.

Epp, Eldon Jay. 1995. 'The papyrus manuscripts of the New Testament'. In *The text of the New Testament in contemporary research: essays on the status quaestionis*. Festschrift in honour of Bruce M. Metzger. Studies and documents, 46. Ed. Bart D. Ehrman and Michael W. Holmes. Grand Rapids: Eerdmans, 3-21.

Epp, Eldon Jay. 1997. 'The codex and literacy in early Christianity and at Oxyrhynchus: issues raised by Harry Y. Gamble's Books and readers in the early Church'. *Critical review of books in religion* 10, 15-37.

Farthing, Geoffrey P. 1994. 'Detailed textual stemmata by means of

probability theory'. *AIBI 4 — Bible et informatique*, 214-222.

Fee, Gordon D. 1968. 'Codex Sinaiticus in the Gospel of John: a contribution to methodology in establishing textual relationships'. *New Testament studies* 15/1, 23-44.

Fee, Gordon D. 1993. 'P75, P66, and Origen: the myth of early textual recension in Alexandria'. In *Studies in the theory and method of New Testament textual criticism*. Ed. Eldon Jay Epp and Gordon D. Fee. Studies and documents, 45. Grand Rapids: Eerdmans, 247-273.

Finney, T. J. 1991. 'Establishing the text of PROS JEBRAIOUS: a collation of 3 manuscripts against the text of Nestle-Aland^{26}'. Dissertation (Honours). Murdoch University.

Finney, T. J. 1994. 'A proposed reconstruction of Hebrews 7.28a in P46'. *New Testament studies* 40, 472-473.

Finney, T. J. 1997. 'Mapping the textual history of Hebrews'. *Revue informatique et statistique dans les sciences humaines* 33, 125-147.

Fischer, Bonifatius. 1970. 'The use of computers in New Testament studies, with special reference to textual criticism'. *Journal of theological studies* 21/2, 297-308.

Friberg, Barbara and Timothy (eds.). 1981. *Analytical Greek New Testament*. Grand Rapids: Baker Book House.

Gamble, Harry Y. 1995. *Books and readers in the early church: a history of early Christian texts*. New Haven: Yale University Press.

Gardthausen, V. 1913. *Griechische Palaeographie*. Vol. 2. *Die Schrift, Unterschriften und Chronologie im Alterum und im Byzantinischen Mittelalter*. 2nd ed. Leipzig: Veit. Repr. 1978. Berlin: Nationales Druckhaus.

Garnet, Paul. 1985. 'Hebrews 2: 9: CARITI or CWRIS?'. *Studia Patristica* 18/1, 321-325.

Gignac, Francis Thomas. [1975]. *A grammar of the Greek papyri of the Roman and Byzantine periods*. Vol. 1. *Phonology*. Testi e documenti per lo studio dell'antichità, 55-1. Milan: Istituto Editoriale Cisalpino - La Goliardica. (The date is that given in the preface.)

Gignac, Francis Thomas. [1977]. *A grammar of the Greek papyri of the Roman and Byzantine periods*. Vol. 2. *Morphology*. Testi e documenti per lo studio dell'antichità, 55-2. Milan: Istituto Editoriale Cisalpino - La Goliardica. (The date is that given in the preface.)

Gottfried, Byron S. 1990. *Schaum's outline of theory and problems of programming with C*. Schaum's outline series. New York: McGraw-Hill.

Grenfell, Bernard P. and Arthur S. Hunt (eds.). 1900. *The Amherst papyri: being an account of the Greek papyri in the collection of the Right Hon. Lord Amherst of Hackney, F.S.A. at Didlington Hall, Norfolk*. Part 1. *The ascension of Isaiah, and other theological fragments*. London: Oxford University Press. Repr. 1975. Milan: Istituto Editoriale Cisalpino - La Goliardica.

Grenfell, Bernard P. and Arthur S. Hunt (eds.). 1901. *The Amherst papyri: being an account of the Greek papyri in the collection of the Right Hon. Lord Amherst of Hackney, F.S.A. at Didlington Hall, Norfolk*. Part 2. *Classical fragments and documents of the Ptolemaic Roman and Byzantine periods*. London: Oxford University Press. Repr. 1975. Milan: Istituto Editoriale Cisalpino - La Goliardica.

Grenfell, Bernard P. and Arthur S. Hunt (eds.). 1904. '657. Epistle to the Hebrews'. *The Oxyrhynchus papyri*. Part 4. London: Egypt Exploration Fund, 36-48.

Griffith, John G. 1969. 'Numerical taxonomy and some primary manuscripts of the Gospels'. *Journal of theological studies* 20/2,

389-406.

Griffith, John G. 1979. 'Non-stemmatic classification of manuscripts by computer methods'. *Colloques internationaux du CNRS 579 — La pratique des ordinateurs dans la critique des textes*, 74-86.

Guillemette, Pierre. 1986. *The Greek New Testament analysed*. Kitchener: Herald Press.

Harnack, Adolf. 1908. *The mission and expansion of Christianity in the first three centuries*. Vol. 2. *The spread of the Christian religion*. Theological translation library, 20. Trans. and ed. James Moffatt. 2nd enlarged and rev. ed. London: Williams and Norgate.

Hartley, B. M. 1988. 'A computer method for simulating the decay of radon daughters'. *Radiation protection in Australia* 6/4, 126-130.

Head, Peter M. 1990. 'Observations on early papyri of the Synoptic Gospels, especially on the "scribal habits"'. *Biblica* 71/2, 240-247.

Head, Peter M. and M. Warren. 1997. 'Re-inking the pen: evidence from P. Oxy. 657 (P^{13}) concerning unintentional scribal errors'. *New Testament studies* 43, 466-473.

Heath, Dale Eldon. 1965. 'A transcription and description of Manuscript Vatican Greek 2061 (Gregory 048)'. Dissertation (PhD). Michigan State University.

Hewett, James Allen. 1986. *New Testament Greek: a beginning and intermediate grammar*. Peabody: Hendrickson.

Hoffman, Paul E. 1997. *Perl 5 for dummies*. Foster City: IDG Books Worldwide.

Horsley, G. H. R. 1982. *New documents illustrating early Christianity*. Vol. 2. Sydney: Macquarie University, 139.

Hunt, Arthur S. (ed.). 1911. '1078. Epistle to the Hebrews ix.' *The*

Oxyrhynchus papyri. Part 8. London: Egypt Exploration Fund, 11-13.

Jeff Matson. 1991. *Think C: standard libraries reference*. Cupertino: Symantec.

H KAINH DIAQHKH. 1873. Fascicle 7. Oxford: Oxford University Press. Repr. n.d. Chicago: University of Chicago Press.

Kaye, Brian Howard. 1993. *Chaos and complexity: discovering the surprising patterns of science and technology*. Weinham: VCH.

Kenyon, Frederic G. (ed.). 1933. *The Chester Beatty biblical papyri: descriptions and texts of twelve manuscripts on papyrus of the Greek Bible*. Fasciculus 1. *General introduction*. London: Emery Walker.

Kenyon, Frederic G. (ed.). 1936. *The Chester Beatty biblical papyri: descriptions and texts of twelve manuscripts on papyrus of the Greek Bible*. Fasciculus 3 supplement. *Pauline epistles: text*. London: Emery Walker.

Kenyon, Frederic G. (ed.). 1937. *The Chester Beatty biblical papyri: descriptions and texts of twelve manuscripts on papyrus of the Greek Bible*. Fasciculus 3 supplement. *Pauline epistles: plates*. London: Emery Walker.

Kenyon, Frederic G. 1950. *The text of the Greek Bible: a student's handbook*. Studies in theology, 38. Rev. ed. London: Gerald Duckworth.

Kenyon. Frederic G. (ed.). 1909. *The Codex Alexandrinus (Royal ms. 1 D v-viii) in reduced photographic facsimile*. London: British Museum.

Kernighan, Brian W. and Dennis M. Ritchie. 1988. *The C programming language*. 2nd ed. Englewood Cliffs: Prentice Hall.

Kilpatrick, George D. 1941. 'The Chester Beatty papyrus Π^{46} and Hebrews xi. 4'. *Journal of theological studies* 42, 68-69.

Kilpatrick, George D. 1980. 'The text of the epistles: the contribution of Western witnesses'. *In Text – Wort – Glaube: Studien zur Überlieferung, Interpretation und Autorisierung biblischer Texte*. Festschrift in honour of Kurt Aland. Ed. Martin Brecht. Arbeiten zur Kirchengeschichte, 50. Berlin: Walter de Gruyter, 47-68.

Kilpatrick, George D. n.d. 'Unpublished handwritten amendments to *H KAINH DIAQHKH* (British and Foreign Bible Society, 2nd ed., 1958)'. Courtesy of J. Keith Elliott, University of Leeds.

Kim, Young Kyu. 1988. 'Palaeographical dating of P^{46} to the later first century'. *Biblica* 69, 248-57.

Kleinlogel, Alexander. 1979. 'Fundamentals of a formal theory of manuscript classification and genealogy'. *Colloques internationaux du CNRS 579 — La pratique des ordinateurs dans la critique des textes*, 74-86.

Kraft, Robert A., Emanuel Tov, et. al. 1986. *Computer assisted tools for Septuagint studies (CATSS)*. Vol. 1. *Ruth*. Society of Biblical Literature: Septuagint and cognate studies series, 20. Atlanta: Scholars Press.

Kubo, Sakae. 1975. *A reader's Greek-English lexicon of the New Testament and a beginner's guide for the translation of New Testament Greek*. Andrews University monographs, 4. Grand Rapids: Zondervan.

Kvalheim, O. M., D. Apollon, and R. H. Pierce. 1988. 'A data-analytical examination of the Claremont Profile Method for classifying and evaluating manuscript evidence'. *Symbolae Osloenses* 63, 133-144.

Lake, Helen and Kirsopp. 1911. *Codex Sinaiticvs Petropolitanvs: the New Testament: the epistle of Barnabas and the Shepherd of*

Hermas. Photographic facsimile with an introduction by Kirsopp Lake. Oxford: Clarendon Press.

Lake, Kirsopp. 1906. *The text of the New Testament*. Oxford church text books. 3rd ed. London: Rivingtons.

Lampe, G. W. H. (ed.). 1961. *A patristic Greek lexicon*. Oxford: Clarendon Press.

Lane, William L. 1991a. *Hebrews 1-8*. Word biblical commentary, 47A. Dallas: Word Books.

Lane, William L. 1991b. *Hebrews 9-13*. Word biblical commentary, 47B. Dallas: Word Books.

Liddell, Henry George and Robert Scott. 1968. *A Greek-English lexicon*. Rev. and augmented ed. with a supplement. Oxford: Clarendon Press.

Lundberg, Marilyn J. 1992. *A descriptive list of holdings*. 2nd ed. Claremont: Ancient Biblical Manuscript Center.

Lyon, Robert W. 1959. 'A re-examination of Codex Ephraemi Rescriptus'. *New Testament studies* 5/4, 260-72.

Lyon, Robert W. [1958]. 'A re-examination of Codex Ephraemi Rescriptus'. Dissertation (PhD). University of St Andrews. (Date deduced from the preface.)

Manson, William. 1951. *The epistle to the Hebrews: an historical and theological reconsideration*. Baird Lecture of 1949. London: Hodder and Stoughton.

Martini, Carlo M. (ed.). 1968. *Novum Testamentum e Codice Vaticano Graeco 1209 (Codex B)*. Trans. David Barry. Vatican City: Bibliotheca Apostolica Vaticana.

Matloff, Norman S. 1988. *Probability modeling and computer*

simulation: an integrated introduction with applications to engineering and computer science. Boston: PWS-Kent.

McIntosh, Lawrence D. 1995. *A style manual for the presentation of papers and theses in religion and theology*. Corrected ed. Wagga Wagga: Centre for Information Studies.

Mealand, D. L. 1995. 'The extent of the Pauline corpus: a multivariate approach'. *Journal for the study of the New Testament* 55, 61-92.

Merk, Augustine (ed.). 1964. *Novum Testamentum Graece et Latine*. 9th ed. Rome: Pontifical Biblical Institute.

Metzger, Bruce M. 1963. 'Explicit references in the works of Origen to variant readings in New Testament manuscripts'. In *Biblical and patristic studies: in memory of R. P. Casey*. Ed. J. N. Birdsall and R. W. Thomson. Freiburg: Herder, 78-95.

Metzger, Bruce M. 1977. *The early versions of the New Testament: their origin, transmission, and limitations*. Oxford: Clarendon Press.

Metzger, Bruce M. 1981. *Manuscripts of the Greek Bible: an introduction to Greek palaeography*. New York: Oxford University Press.

Metzger, Bruce M. 1992. *The text of the New Testament: its transmission, corruption, and restoration*. 3rd enlarged ed. New York: Oxford University Press.

Metzger, Bruce M. 1994. *A textual commentary on the Greek New Testament*. 2nd ed. Stuttgart: Deutsche Bibelgesellschaft.

Milne, H. J. M. and T. C. Skeat. 1938. *Scribes and correctors of the Codex Sinaiticus*. Oxford: Oxford University Press.

Mink, Gerd. 1993. 'Eine umfassende Genealogie der neutestamentlichen Überlieferung'. *New Testament studies* 39/4, 481-499.

Moore, David S. and George P. McCabe. 1993. *Introduction to the practice of statistics*. 2nd ed. New York: W. H. Freeman.

Moore, Richard K. 1992. *New Testament Greek*. 5th corrected ed. Perth: Murdoch University.

Moulton, James Hope and Wilbert Francis Howard. 1928. *A grammar of New Testament Greek*. Vol. 2. *Accidence and word-formation with an appendix on Semitisms in the New Testament*. Edinburgh: T. & T. Clark.

Mullen, Roderic L. 1997. *The New Testament text of Cyril of Jerusalem*. Society of Biblical Literature: the New Testament in the Greek Fathers, 7. Ed. Bart D. Ehrman. Atlanta: Scholars Press.

Murphy, Harold S. 1959. 'On the text of codices H and 93'. *Journal of biblical literature* 78, 228-237.

Nestle, Eberhard and Kurt Aland (eds.). 1963. *Novum Testamentum Graece*. Nestle-Aland, 25th ed. Stuttgart: Württembergische Bibelanstalt.

Omont, M. H. 1890. 'Notice sur un très ancien manuscrit grec en onciales des Épîtres de Saint Paul'. In *Notices et extraits des manuscrits de la Bibliothèque Nationale*. Vol. 33. Paris: Bibliothèque Nationale, 141-192.

Osburn, Carroll D. 1995. 'The Greek lectionaries of the New Testament'. In *The text of the New Testament in contemporary research: essays on the status quaestionis*. Festschrift in honour of Bruce M. Metzger. Studies and documents, 46. Ed. Bart D. Ehrman and Michael W. Holmes. Grand Rapids: Eerdmans, 61-74.

Panten, Kenneth E. 1995. 'A history of research on Codex Bezae, with special reference to the Acts of the Apostles: evaluation and future directions'. Dissertation (PhD). Murdoch University.

Payne, Philip B. 1995. 'Fuldensis, Vaticanus, and 1 Cor 14.34-5'. *New*

Testament studies 41/2, 240-262.

Petersen, William L. 1994. *Tatian's Diatessaron: its creation, dissemination, significance, and history in scholarship*. Supplements to Vigiliae Christianae, 25. Leiden: E. J. Brill.

Pickering, S. R. 1991. *Recently published New Testament papyri: P89 - P95*. Papyrology and historical perspectives, 2. Sydney: Macquarie University, 6-10.

Pintaudi, Rosario. 1981. 'N.T. Ad Hebraeos, VI, 7-9; 15-17 (PL III/92)'. *Zeitschrift für Papyrologie und Epigraphik* 42, 42-44.

Plotkin, Wendy and C. M. Sperberg-McQueen. 1996. *Text encoding initiative*. WWW document (http://www.uic.edu/orgs/tei).

Podani, János. 1994. *Multivariate data analysis in ecology and systematics: a methodological guide to the SYN-TAX 5.0 package*. Ecological computations series, 6. The Hague: SPB Publishing.

Podani, János. 1995. *SYN-TAX 5.02. Mac: computer programs for multivariate data analysis on the Macintosh system*. Computer program and user's guide. Budapest: Scientia Publishing.

Price, James D. 1987. 'A computer aid for textual criticism'. *Grace theological journal* 8/1, 115-129.

Rahlfs, Alfred (ed.). 1935. *Septuaginta: Id est Vetus Testamentum Graece iuxta LXX interpretes*. 2 vols. 9th ed. Stuttgart: Deutsche Bibelstiftung.

Revised Standard Version. 1946. New York: Thomas Nelson & Sons.

Ralston, T. R. 1992. 'The *"Majority Text"* and Byzantine origins'. *New Testament studies* 38, 122-137.

Robertson, A. T. 1923. *A grammar of the Greek New Testament in the light of historical research*. 4th ed. Nashville: Broadman Press.

Robinson, J. Armitage. 1895. *Euthaliana: studies of Euthalius, Codex H of the Pauline epistles, and the Armenian version*. Texts and studies, vol. 3, no. 3. Cambridge: Cambridge University Press. Repr. 1967. Nendeln: Kraus Reprint.

Robinson, Peter M. W. 1994a. *The transcription of primary textual sources using SGML*. Office for Humanities Communication publications, 6. Oxford: Office for Humanities Communication.

Robinson, Peter M. W. 1994b. *Collate: interactive collation of large textual traditions, version 2*. Computer program. Oxford: Oxford University Centre for Humanities Computing.

Robinson, Peter M. W. 1994c. *Collate 2: a user guide*. Oxford: The Computers and Variant Texts Project.

Roca-Puig, R. 1965. 'Papyrus Barcinonensis, inv. no. 6 (Hebr. 6, 2-4, 6-7)'. *Helmantica* 16, 145-9.

Ropes, James Hardy. 1926. *The beginnings of Christianity*. Part 1. *The Acts of the Apostles*. Vol. 3. *The text of Acts*. Ed. F. J. Foakes Jackson and Kirsopp Lake. London: Macmillan. Repr. 1979. Grand Rapids: Baker Book House.

Ross, J. M. 1992. 'Floating words: their significance for textual criticism'. *New Testament studies* 38, 153-156.

Royse, James Ronald. 1981. 'Scribal habits in early Greek New Testament papyri'. Dissertation (ThD). Graduate Theological Union.

Sanders, Henry A. (ed.). 1918. *The New Testament manuscripts in the Freer Collection*. Part 2. *The Washington manuscript of the epistles of Paul*. University of Michigan studies humanistic series, 9. New York: Macmillan.

Sanders, Henry A. (ed.). 1935. *A third-century papyrus codex of the epistles of Paul*. Ann Arbor. University of Michigan Press.

Schick, Robert. 1998. 'Palestine in the early Islamic period: luxuriant legacy'. *Near Eastern archaeology* 61/2, 74-108.

Schofield, E. M. 1936. 'The papyrus fragments of the Greek New Testament'. Dissertation (PhD). Southern Baptist Theological Seminary.

Scrivener, Frederick H. 1867. *A full collation of the Codex Sinaiticus with the received text of the New Testament*. 2nd rev. ed. Cambridge: Deighton, Bell, and Co.

Skeat, T. C. 1969. 'Early Christian book production: papyri and manuscripts'. In *The Cambridge history of the Bible*. Vol. 2. *The West from the Fathers to the Reformation*. Ed. G. W. H. Lampe. Cambridge: Cambridge University Press.

Skeat, T. C. 1984. 'The Codex Vaticanus in the fifteenth century'. *Journal of theological studies* 35/2, 454-465.

Skeat, T. C. 1997. 'The oldest manuscript of the four Gospels?' *New Testament studies* 43/1, 1-34.

Souter, Alexander (ed.). 1947. *Novvm Testamentvm Graece*. 2nd ed. Oxford: Clarendon Press.

Souter, Alexander. 1954. *The text and canon of the New Testament*. Studies in theology, 25. 2nd ed., rev. C. S. C. Williams. London: Gerald Duckworth.

Spiegel, Murray R. 1982. *Theory and problems of probability and statistics*. Schaum's outline series. Asian student ed. Singapore: McGraw-Hill.

Spuler, David. 1992. *Comprehensive C*. New York: Prentice Hall.

Stanton, Graham N. 1997. 'The fourfold Gospel'. *New Testament studies* 43/3, 317-346.

Stark, Rodney. 1997. *The rise of Christianity: how the obscure, marginal Jesus movement became the dominant religious force in the western world in a few centuries*. San Francisco: Harper Collins.

Stewart-Sykes, Alistair. 1996. 'Ancient editors and copyists and modern partition theories: the case of the Corinthian correspondence'. *Journal for the study of the New Testament* 61, 53-64.

Streeter, Burnett Hillman. 1925. *The four Gospels: a study of origins treating of the manuscript tradition, sources, authorship, & dates*. New York: Macmillan.

TEI classical Greek supplemental to ISOgrk1 and ISOgrk2. 1992. FTP archive (ftp://ftp-tei.uic.edu/pub/tei/dtd/teigrk.ent).

Tasker, R. V. G. (ed.). 1964. *The Greek New Testament: being the text translated in The New English Bible 1961*. Oxford: Oxford University Press.

The Oxford dictionary of the Christian Church. 1997. 3rd ed. Ed. F. L. Cross and E. A. Livingstone. Oxford: Oxford University Press.

Thompson, E. Maunde. 1883. *Facsimile of the Codex Alexandrinus*. Vol. 4. *New Testament and Clementine epistles*. London: British Museum.

'The Times' atlas of the world. 1990. 8th comprehensive ed. London: Times Books.

Tischendorf, Konstantin von (ed.). 1845. *Codex Ephraemi Syri Rescriptus sive fragmenta utriusque Testamenti e codice Graeco Parisiensi celeberrimo quinti ut videtur post Christum seculi*. Leipzig: Bernh. Tauchnitz Jun.

Tischendorf, Konstantin von (ed.). 1852. *Codex Claromontanus sive epistulae Pauli omnes Graece et Latine ex codice Parisiensi celeberrimo nomine Claromontani plerumque dicto sexti ut videtur*

post Christum saeculi. Leipzig: F. A. Brockhaus.

Tischendorf, Konstantin von (ed.). 1862. *Bibliorum Codex Sinaiticus Petropolitanus*. Vol. 4. *Novum Testamentum cum Barnaba et Pastore*. Repr. 1969. Hildesheim: Georg Olms.

Tischendorf, Konstantin von (ed.). 1867. *Novum Testamentum Vaticanum*. Leipzig: Giesecke and Devrient.

Tischendorf, Konstantin von (ed.). 1872. *Novum Testamentum Graece*. Vol. 2. 8th major critical ed. Leipzig: Giesecke and Devrient. Repr. 1965. Graz: Akademischen Druck.

Treu, Kurt. 1966. 'Neue neutestamentliche Fragmente'. *Archiv für Papyrusforschung* 18, 37-38.

Turner, E. G. 1987. *Greek manuscripts of the ancient world*. Institute of Classical Studies bulletin supplement, 46. 2nd rev. and enlarged ed. Ed. P. J. Parsons. London: Institute of Classical Studies.

Wachtel, Klaus and Klaus Witte. 1994. *Das Neue Testament auf Papyrus*. Vol. 2. *Die Paulinischen Briefe*. Part 2. *Gal, Eph, Phil, Kol, 1 u. 2 Thess, 1 u. 2 Tim, Tit, Phlm, Hebr*. Arbeiten zur neutestamentlichen Textforschung, 22. Berlin: Walter de Gruyter.

Wachtel, Klaus. 1995. *Der byzantinische Text der katholischen Briefe: eine untersuchung zur Entstehung der Koine des Neuen Testaments*. Arbeiten zur neutestamentlichen Textforschung. Band 24. Berlin: Walter de Gruyter.

Wenham, J. W. 1965. *The elements of New Testament Greek*. Cambridge: Cambridge University Press.

Westcott, Brooke Foss and Fenton John Anthony Hort (eds.). 1881a. *The New Testament in the original Greek*. Vol. 1. *Text*. Cambridge: Macmillan. Repr. 1974. Graz: Akademische Druck.

Westcott, Brooke Foss and Fenton John Anthony Hort. 1881b. *The New*

Testament in the original Greek. Vol. 2. *Introduction* [and] *Appendix*. Cambridge: Macmillan. Repr. 1974. Graz: Akademische Druck.

Wolmarans, Johannes L. P. 1984. 'The text and translation of Hebrews 8 8'. *Zeitschrift für die neutestamentliche Wissenschaft und die Kunde der älteren Kirche* 75, 139-144.

World of learning. 1995. 46th ed. London: Europa Publications.

Youtie, Herbert C. 1966. 'Text and context in transcribing papyri'. *Greek, Roman, and Byzantine studies* 7, 251-8.

Youtie, Herbert C. 1974. *The textual criticism of documentary papyri: prolegomena*. 2nd rev. and enlarged ed. Institute of Classical Studies bulletin supplement, 33. London: University of London.

Zipf, George Kingsley. 1949. *Human behavior and the principle of least effort: an introduction to human ecology*. Repr. 1972, New York: Hafner.

Zuntz, Günther. 1945. *The ancestry of the Harklean New Testament*. British Academy supplemental papers, 7. London: Oxford University Press.

Zuntz, Günther. 1953. *The text of the epistles: a disquisition upon the corpus Paulinum*. Schweich Lectures of 1946. London: British Academy.