Tutorial showcasing the cerUB package for R

2019-09-24

Contents

cerUB tutorial

Tutorial showcasing the cerUB package for R (cerUB).

This document shows how to use *cerUB* protocols to explore petrographic and compositional data. As an example, we provide the 'amphorae' dataset concerning wine Roman amphorae from sites in Catalonia, NE Spain. It is a live version of the Appendix C of an article published in Journal of Archaeological Science (see reference below).

All rmarkdown (.Rmd) source files are available in the repository.

There, the publication appendices folder contains alternative versions of the appendices (supplementary materials) of the related article:

> Angourakis, A., Martínez Ferreras, V., Torrano A., Gurt Esparraguera, J.M. (2018) Presenting multivariate statistical protocols in R using Roman wine amphorae productions in Catalonia, Spain. *Journal of Archaeological Science*, 93:150-165. https://doi.org/10.1016/j.jas.2018.03.007

Additionaly, the repository also contains: - Poster presented in the International Symposium on Archaeometry (ISA) in Merida, Mexico, May 2018. - Poster presented in the Open Science and The Humanities Conference in Barcelona, Spain, June 2018.

Please, address any issues, suggestions, and comments to Andreas Angourakis (andros.spica@gmail.com, or through *GitHub*, user **Andros-Spica**)

Chapter 1

Installing cerUB

There are two options for installing the *cerUB* package:

a. Downloading the latest release version from Zenodo.org and installing it in RStudio (Tools > Install Packages… > Install from: Package Archive File).

   Angourakis, Andreas, & Martínez Ferreras, Verònica. (2017, September 23). cerUB - Protocols for exploring archaeometric data (R package). Zenodo. http://doi.org/10.5281/zenodo.975451

b. Installing the latest development version directly from GitHub (Andros-Spica/cerUB), using the *devtools* package:

```
# this will install devtools package, if not installed already
if (!require("devtools"))
    install.packages("devtools")

devtools::install_github("Andros-Spica/cerUB")
```

The second option is recommended, because it is a faster way to install and update packages that are not in CRAN.

The same options are available for installing the *biplot2d3d* package, which we use here to plot protocols results.

> Andreas Angourakis. (2017, September 20). biplot2d3d - an R package for generating highly-customizable biplots. Zenodo. http://doi.org/10.5281/zenodo.897603

```
devtools::install_github("Andros-Spica/biplot2d3d")
```

Any other package required by these two packages (*ade4*, *rgl*, etc.) should be automatically installed.

Chapter 2

Initial procedures

Load *cerUB* package.

```
library(cerUB)
```

2.1 Set directories for saving data

Set a list containing directories for easiness of reference:

```
directories <- list(
  # where you raw data is
  data = "data",
  # where to save transformed compositional data (CoDa)
  transCoDa = "transformed_CoDa",
  # where to save files concerning protocol workflow
  prot1 = "Protocol_1_geochemical_data",
  prot2 = "Protocol_2_petrographic_data",
  prot3 = "Protocol_3_geochemical_and_petrographic_data",
  prot4 = "Protocol_4_provenance_data",
  prot4_Shipwreck = "Protocols_4_provenance_data_with_shipwrecks"
)
```

Create the respective folders in the current R session working directory, if they do not exist:

```
lapply(directories, dir.create, showWarnings = FALSE)
```

2.2 Read data

Load the amphorae dataset:

```
data(amphorae)
```

Or, alternatively, your own dataset (e.g., CSV):

```
dt <- cbind(read.csv(paste(directories$data,
                           "petrographic_data.csv",sep="/"),
                     # assuming the first column contains row names
                     row.names=1),
            read.csv(paste(directories$data,
                           "geochemical_data.csv",sep="/"),
                     # assuming the first column contains row names
                     row.names=1))
```

Note that if you use your own dataset you must replace all references to "amphorae" with your data frame name (e.g. "dt").

2.3 Codify petrographical variables

Create a two-column data frame containing the original names of petrographic variables and their respective codes:

```
varCode <- code_variables(amphorae)
```

Petrographic variables must be named following *cerUB* naming system. Consult the documentation on the "amphorae" dataset by running:

```
?amphorae
```

The result of the **code_variables** function is used in the **apply_ordination** function for protocols "2a", "2b", "3", and "4" (i.e., whenever there are ordinal input variables). The data frame containing the codification ('varCode') is attached as "ordination_object$variable_tags" to the resulting ordination object.

Variable name	Variable code
Site_Name	Site_Name
LOCATION_SITE_INITIALS	LOCATION_SITE_INITIALS
CHARAC	CHARAC
FabricGroup	FabricGroup
ChemReferenceGroup	ChemReferenceGroup
INCLUS_DISTRIB	I1
INCLUS_ORIENT	I2
TEMP	F1
ATM	F2
POST_ATM	F3
VOID_OVERALL	V1
VOID_VESIC_MEGA	V2
...	...

2.4 Clean and format data

Cleaning and format procedures, including coercing variables as numeric or factor, excluding columns (constants, perturbed, unreliable) and rows (incomplete data, outliers).

```
cleanAmphorae <- clean_and_format(
  amphorae,
  completion_variable = c(
    # The variable with completion info
    "CHARAC",
    # the value indicating completion
    "complete"
  ),
  categorical_columns = 1:112,
  numerical_columns = 113:ncol(amphorae),
  # values converted to NA
  as_na = c("NULL", "indeterminate", "unfired"),
  # method for replacing NAs
  method = NULL,
  # don't use the following variables
  columns_to_exclude = c("VOID_VESIC_MEGA", "VOID_VUGH_MEGA",
                         "VOID_CHAN_MEGA", "VOID_PLAN_MEGA",
                         "COAR_R_DAC_AND", "COAR_R_EVAP",
                         "COAR_R_CONGBREC", "COAR_R_SERP",
                         "COAR_C_SPL", "COAR_C_OPX",
                         "COAR_C_OL", "COAR_C_SIL",
```

```
                        "COAR_C_ST", "COAR_C_ZRN",
                        "COAR_C_PY", "FINE_C_OPX",
                        "FINE_C_ZRN"),
  # don't use the following observations
  # (Italic amphorae from Port Vendres 4)
  rows_to_exclude = c("PV4033", # PV4-IND4
                      "PV4017", # PV4-CAMP
                      # PV4-ITT
                      "PV4021", "PV4023", "PV4024",
                      "PV4025", "PV4035", "PV4037",
                      # PV4-NAP
                      "PV4022", "PV4026", "PV4027",
                      "PV4028", "PV4029", "PV4030",
                      "PV4036")
)
```

	Variables	Observations
amphorae	138	238
cleanAmphorae	91	223

2.5 Subsetting criteria

Build vector indicating whether each observation is from a shipwreck:

```
isShipwreck <-
  cleanAmphorae$Site_Name=="Cap del Vol" |
  cleanAmphorae$Site_Name=="Ullastres I" |
  cleanAmphorae$Site_Name=="Port-Vendres 4"
```

	Workshops	Shipwrecks
	175	48

Build vectors indicating provenance group and whether observations are true outliers (IND, observations with no group assigned). Also, reformat "FabricGroup" and "ChemReferenceGroup", so true outliers are singled out separately and not as a extra group.

```
ProvenanceGroup <- c()
isTrueIND <- c()

# coerce the original group variables (factors) into character vectors
# so we can use stringr package to operate on them.
cleanAmphorae$FabricGroup <-
  as.character(cleanAmphorae$FabricGroup)
cleanAmphorae$ChemReferenceGroup <-
```

```
  as.character(cleanAmphorae$ChemReferenceGroup)

for (i in 1:nrow(cleanAmphorae)){
  groupChem <-
    stringr::str_split(cleanAmphorae$ChemReferenceGroup[i], "-")[[1]]
  groupFabric <-
    stringr::str_split(cleanAmphorae$FabricGroup[i], "-")[[1]]
  group <- ""
  isATrueInd <- FALSE

  if (groupChem[2] == "IND" || groupFabric[2] == "IND") {
    group <- cleanAmphorae$ChemReferenceGroup[i]
    if (!isShipwreck[i]) isATrueInd <- TRUE
    index <- 1
    for (j in 1:length(ProvenanceGroup)){
      if (ProvenanceGroup[j] == paste(group, index, sep = ""))
        index <- index + 1
    }
    group <- paste(group, index, sep = "")
    cleanAmphorae$ChemReferenceGroup[i] <- group
    cleanAmphorae$FabricGroup[i] <- group
  }
  else {
    if (groupChem[1] == "ULL" ||
        groupChem[1] == "PV4" ||
        groupChem[1] == "CDV") {
      group <- cleanAmphorae$ChemReferenceGroup[i]
    }
    else if (groupChem[1] == groupFabric[1]){
      group <- groupChem[1]
    }
  }
  ProvenanceGroup <- c(ProvenanceGroup, group[1])
  isTrueIND <- c(isTrueIND, isATrueInd)
}
```

	Assigned	Outliers
	205	18

2.6 Organizing groups

Build lists of named group factors for easiness of reference.

First, create a list aiming to define workshops productions, so no shipwrecks:

```
factor_list <-
  list(
    Site = factor(cleanAmphorae$Site_Name[!isShipwreck]),
    FabricGroup = factor(cleanAmphorae$FabricGroup[!isShipwreck]),
    ChemGroup = factor(cleanAmphorae$ChemReferenceGroup[!isShipwreck]),
    ProvGroup = factor(ProvenanceGroup[!isShipwreck])
  )
```

	Site	FabricGroup	ChemGroup	ProvGroup
ACM001	Ca L'Arnau-Can Pau Ferrer	ACM-2	ACM-C	ACM
ACM097	Ca L'Arnau-Can Pau Ferrer	ACM-1	ACM-A	ACM
BIF006	BDN-Pompeu Fabra	BIF-1	BIF-2	BIF
BIF048	BDN-Pompeu Fabra	BIF-1	BIF-3	BIF
CRC002	Cal Ros de les Cabres	CRC-1	CRC-1	CRC
ELV002	El Vilarenc	ELV-1	ELV-1	ELV
ELV051	El Vilarenc	ELV-1	ELV-2	ELV
FEU009	Can Feu	FEU-IND1	FEU-IND1	FEU-IND1
LLA010	Llafranc	LLA-1	LLA-A2	LLA
MOR018	El More	MOR-2	MOR-3	MOR
SAL025	La Salut	SAL-2	SAL-1	SAL
SBL045	Sant Boi Historic Centre	SBL-2	SBL-2	SBL

Then, create a second list, aiming to assign shipwreck observations to workshop productions, so with shipwreck samples but no true outliers:

```
factor_list_Shipwreck <-
  list(
    Site = factor(cleanAmphorae$Site_Name[!isTrueIND]),
    FabricGroup = factor(cleanAmphorae$FabricGroup[!isTrueIND]),
    ChemGroup = factor(cleanAmphorae$ChemReferenceGroup[!isTrueIND]),
    ProvGroup = factor(ProvenanceGroup[!isTrueIND])
  )
```

2.7 Helper objects for plotting

Build lists of named point types vectors for easiness of reference.

Create point type vectors for the whole dataset:

```
labels_code <- as.character(row.names(cleanAmphorae)) # using row names
labels_cross <- rep("+", nrow(cleanAmphorae)) # using +
labels_x <- rep(4, nrow(cleanAmphorae)) # using pch code
labels_point <- rep(20, nrow(cleanAmphorae)) # using pch code
```

Create a list aiming to define workshops productions:

```
labels_list <- list(
  Code = labels_code[!isShipwreck],
  Cross = labels_cross[!isShipwreck],
  X = labels_x[!isShipwreck],
  Point = labels_point[!isShipwreck]
)
```

Create a list aiming to assign shipwreck observations to workshop productions:

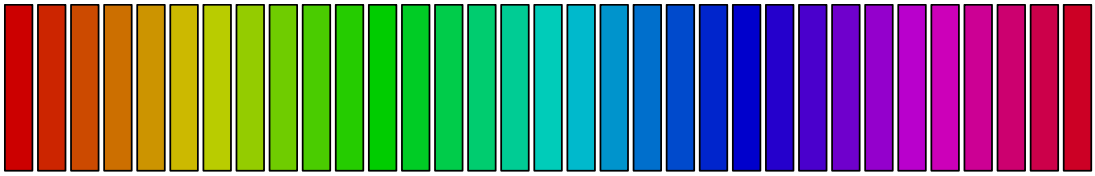
```
labels_list_Shipwreck <- list(
  Code = labels_code[!isTrueIND],
  Cross = labels_cross[!isTrueIND],
  X = labels_x[!isTrueIND],
  Point = labels_point[!isTrueIND]
)
```

Build two other lists containing named group color vectors, picking different colors within the **rainbow** palette:

```
color_list <- list()

for (i in 1:length(factor_list)){
  cv <- rainbow(nlevels(factor_list[[i]]), v=.8)
  color_list[[i]] <- cv
  names(color_list)[i] = names(factor_list)[i]
}

color_list_Shipwreck <- list()

for (i in 1:length(factor_list_Shipwreck)){
  cv <- rainbow(nlevels(factor_list_Shipwreck[[i]]), v=.8)
  color_list_Shipwreck[[i]] <- cv
  names(color_list_Shipwreck)[i] = names(factor_list_Shipwreck)[i]
}
```

To visualize the colors:

```
par(mar = c(0,0,0,0))
barplot(rep(100, nlevels(factor_list$ProvGroup)),
        col = color_list$ProvGroup,
        axes = FALSE)
```



2.8 Enunciate exception columns

Create a vector that enunciate which ordinal variables have "none" as a exceptional value when calculating the distance between values.

```
excep_cols <- c("INCLUS_DISTRIB","INCLUS_ORIENT","COAR_ROUNDNESS",
                "COAR_FORM","COAR_SPACING","COAR_SORTING","FINE_FORM")
```

In order to understand this "exceptional value" feature, compare the levels of regular and exceptional variables. However, to do that at this point you must re-assure the order of petrographic variables (i.e. format factors levels):

```
cleanAmphorae <- order_petro(cleanAmphorae)
```

This step is not necessary for applying the protocols because the apply_ordination function already does it internally, before calculating distances.

Variable	Values
INCLUS_DISTRIB	poorly, poorly to moderately, moderately, moderately to well, well, none
TEMP	unfired, 700-800oC, 800-900oC, 900-1000oC, 1000-1100oC
COAR_FREQ	none, very few, few, common, abundant, very abundant
COAR_ROUNDNESS	angular, angular to subangular, subangular, subangular to subrounded, subrounded, subrounded to rounded, rounded, none
COAR_R_CALS	none, few, common, frequent, dominant, predominant
FINE_FORM	elongate, elongate to equidimensional, equidimensional, equidimensional to laminar, laminar, none
FINE_C_QTZ	none, few, frequent, predominant

2.9 Choose geochemical data

```
chemVars16 <- c("Fe2O3","Al2O3","TiO2","MgO","CaO","SiO2",
                "Th","Nb","Zr","Y","Ce","Ga","V","Zn","Ni","Cr")
```

2.10 Save transformed geochemical data to file (optional)

There is no need to save it in the environment, because **apply_ordination** will transform the data internally and save the results in "ordination_object$transformed_data", when applicable.

```
write(transform_coda(cleanAmphorae,
                     coda_variables = chemVars16,
                     method = c("CLR")),
      file = paste(directories$transCoDa,
                   "transAmphorae_clr.csv",
                   sep = "/"))
```

In the output table, columns will be ordered as:

1. variables not transformed,
2. Raw version of the selected variables,
3. Transformed version of the selected variables.

2.11 Other CoDa packages

Be aware that compositional data (CoDa) analysis can be much more complex than what cerUB currently allows for. For more possibilities, you may explore other R packages: *compositions*, *zCompositions*, and *robCompositions*.

However, before jumping into using more complex techniques, we do recommend a deeper introduction to CoDa:

> Pawlowsky-Glahn, V., Buccianti, A., 2011. Compositional data analysis: theory and applications. Wiley.

Chapter 3

Protocol 1 - Geochemical data

The following example applies protocol 1 to confirm the workshops' chemical reference groups.

Protocol 1 consist in:

1. Select ***geochemical*** compositional data (CoDa);
2. Apply ***transformation***;
3. Perform ***robust Principal Components Analysis*** (robPCA), implicitly using Euclidean distance;
4. Perform ***PERMANOVA & PERMDISP*** tests;

Last, search for outliers and re-do protocol excluding outliers.

NOTE: The initial procedures must be ran at least once before any protocol can be applied.

3.1 Ordination procedure

```
prot1 <- apply_ordination(cleanAmphorae[!isShipwreck,], # no shipwrecks
                          "1", # select protocol 1
                          coda_override = chemVars16,
                          coda_transformation_method = "ILR")
#> [1] "78.24% of variance explained in 2D"
#> [1] "86.54% of variance explained in 3D"
#> [1] "Protocol 1 ended."
```

The outcome is an *ordination object*. In this case, it is the output of **pcaCoDa** function in *robCompositions* package, in addition to several extra information, such as the transformed data ('transformed_data'), the distance matrix ('dist_matrix'), and the ready-to-plot texts indicating the fitness of the 2D/3D projections respect the

distance matrix ('sub2d', 'sub3d'). The later are printed in the console once the object is created.

```
class(prot1)
#> [1] "pcaCoDa"

names(prot1)
#>  [1] "scores"                "loadings"
#>  [3] "eigenvalues"           "method"
#>  [5] "princompOutputClr"     "mult_comp"
#>  [7] "seed"                  "init_seed"
#>  [9] "samples"               "sub2D"
#> [11] "sub3D"                 "transformation_method"
#> [13] "transformed_data"      "dist_matrix"
#> [15] "name"
```

3.2 Simplify CoDa names

We may want to simplify the names of the transformed variables before plotting them in a biplot. The **transform_coda** function, which is called inside **apply_ordination** for protocol 1, generates composite names with format "transformationMethod-component" for all transformed variables (e.g., "CLR-Fe2O3"). The **simplify_coda_names** function replaces these names back to the shorter version (e.g., "Fe2O3"). However, you must always remember that the variables projected in biplots are not the originals but the transformed versions. This is particularly important when dealing with log-ratio variables since they contain information that goes beyond the original variable (i.e., divider).

```
prot1 <- simplify_coda_names(prot1)
```

3.3 Test the given chemical reference groups

Perform four tests (**anosim**, **betadisper**, **permdisp2**, and **permanova**) that assess the separation and uniformity of the given group factor. For more details on these tests, we refer to:

> Anderson, M.J., Walsh, D.C.I., 2013. PERMANOVA, ANOSIM, and the Mantel test in the face of heterogeneous dispersions: What null hypothesis are you testing? Ecol. Monogr. 83, 557-574. doi:10.1890/12-2010.1

The whole test batch may take several minutes depending on the size of the data matrix and the number of groups.

```
prot1_tests <- test_groups(prot1$dist_matrix, factor_list$ChemGroup)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

The tests outputs can be accessed by their name:

```
names(prot1_tests)
#> [1] "permanova" "betadisp"  "permdisp2" "anosim"    "text"
```

The object also contains a "text" object, which is a function that generates a list of text lines for plotting the results of PERMANOVA and PERMDISP2 tests. It may feel confusing, but keep in mind that this "portable" function requires the same ordination object as an argument.

```
displayTestText <- function(test_text) {
  par(mar = c(0, 0, 0, 0), fig = c(0.05, 0.9, 0.05, 0.9))
  plot.new()
  for (i in 1:length(test_text)) {

    # this is for calculating the vertical
    # position of paragraphs and lines
    test_spacing_paragraph = 0.8
    test_spacing_line = 0.8

    first_line_pos_y <-
      1 - test_spacing_paragraph * ( (i - .9) / length(test_text) )

    pos_y <- first_line_pos_y

    if (length(test_text[[i]]) > 1) {

      next_paragraph_pos_y <-
        1 - test_spacing_paragraph * ( i / length(test_text) )

      for (j in 2:length((test_text[[i]])))
      {
        pos_y <-
          c(pos_y,
            first_line_pos_y - test_spacing_line *
```

```
              ((j - 1) / (length((test_text[[i]])))) *
              (first_line_pos_y - next_paragraph_pos_y)
          )
      }
    }
    # display a paragraph (a element of the list)
    text(0, pos_y, labels = test_text[[i]], cex = 0.8, pos = 4)
  }
}

displayTestText(prot1_tests$text(prot1_tests))
```

PERMANOVA:
$F = 30.845$ ($p \leq 0.001$)
$R^2 = 0.929$

PERMDISP2:
$F = 1.95$ ($p = 0.002$)

A rule-of-thumb for interpreting PERMANOVA and PERMDISP2 results is: if both p-values are low enough (e.g. < 0.05), the classification given is a good approximation of the data.

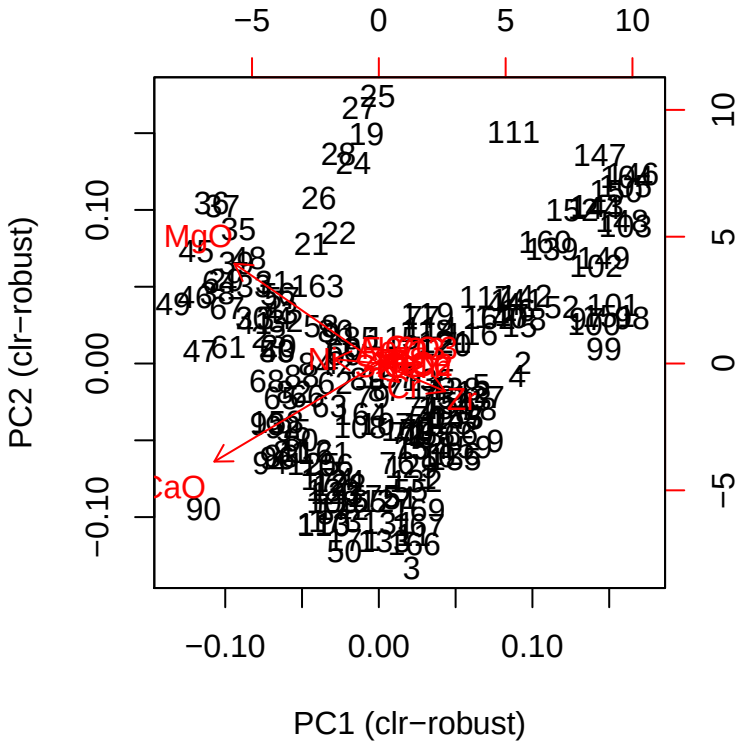


Figure 3.1: Default biplot in R

3.4 Biplots

Ordination objects are best represented in biplots, which simultaneously display the projections of observations (points) and variables (arrows) over the same space. There are several options for creating biplots in R, starting with the readily available **biplot** function:

```
biplot(prot1)
```

Although there are several options for customizing this default biplot function, we recommend the use of the *biplot2d3d* package. This package wraps a lot of possibilities in R.

```
library(biplot2d3d)
```

The *biplot2d3d* package use functions of other packages to allow the customization of virtually all aspects of a biplot. Another important feature of this package is the creation of three-dimensional interactive biplots using the *rgl* package.

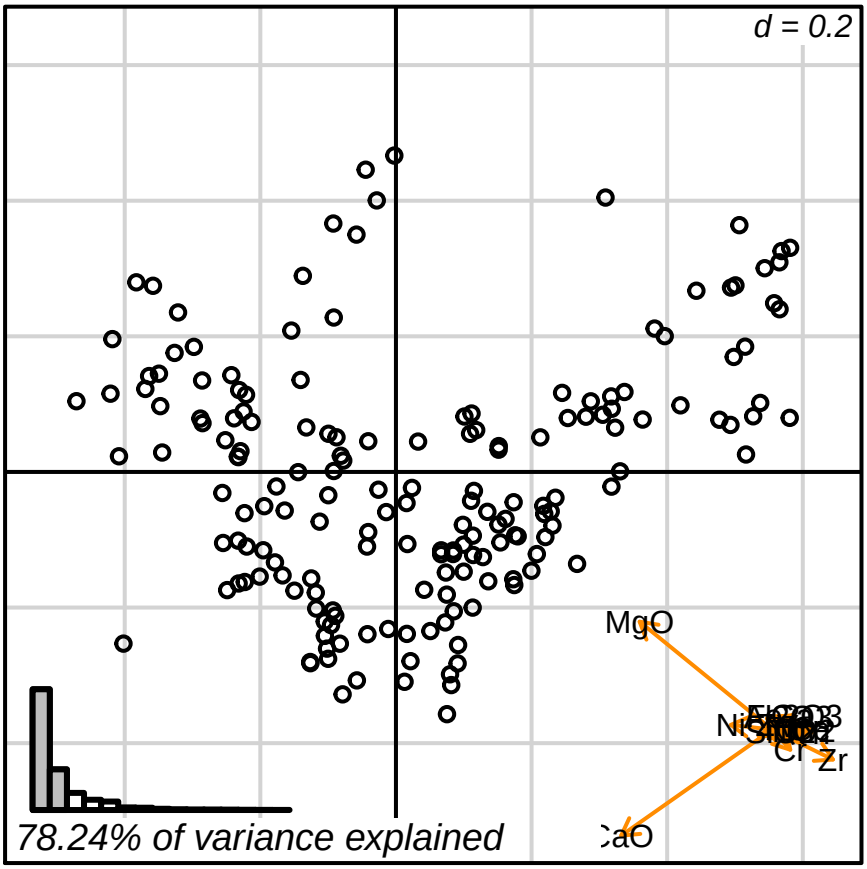


Figure 3.2: default 2D

3.4.1 Biplot 2D

You can consult all tuning options available in the **biplot_2d** function by calling up the help file:

```
?biplot_2d
```

The default configuration will probably give you a much clearer picture than the **biplot** function, specially if your dataset contains more than 100 observations.

```
biplot_2d(prot1)
```

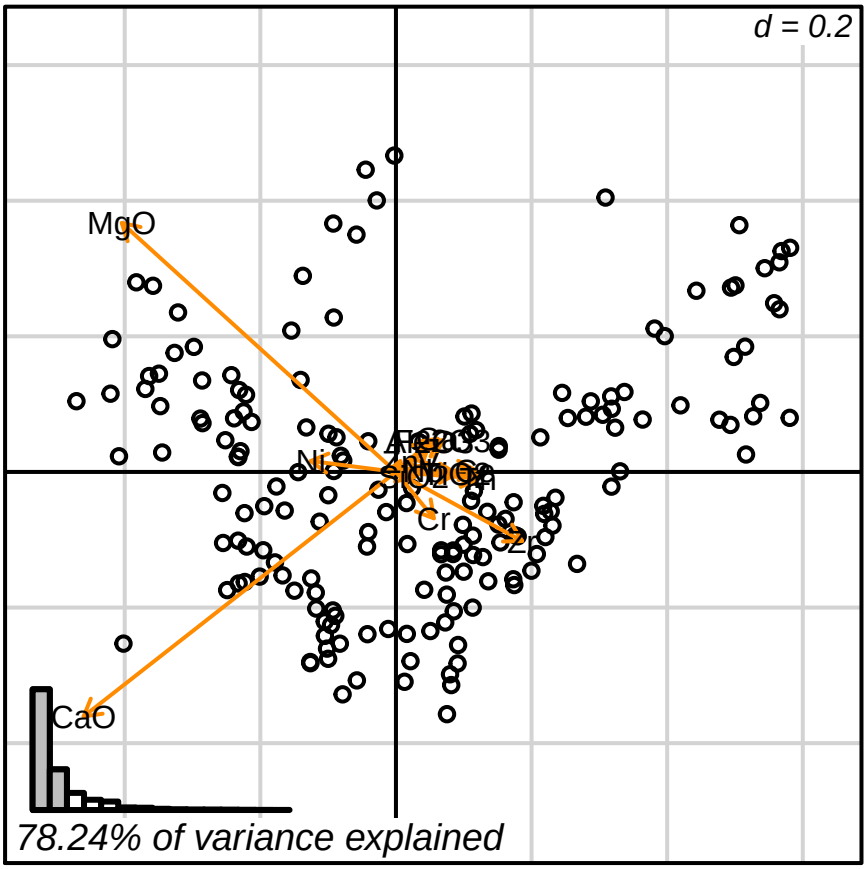


Figure 3.3: detach_arrows = FALSE

The default setting detaches the variables projections (arrows) and places them as a miniature in the bottom-right corner. In this format they may still be interpreted, much like the North when reading a map. Remember though: the more longer arrows you see, the less each one of them is reliable when comparing point values. Here, we were 'lucky' for getting two nearly orthogonal variables (CaO and MgO), which means, for instance, that those observations in the bottom-left corner are surely more calcareous than those in the top-right. See the Appendix section for more details.

If detaching the arrows is not of your preference, you can disable this:

```
biplot_2d(prot1,
          detach_arrows = FALSE,
          output_type = "preview")
```

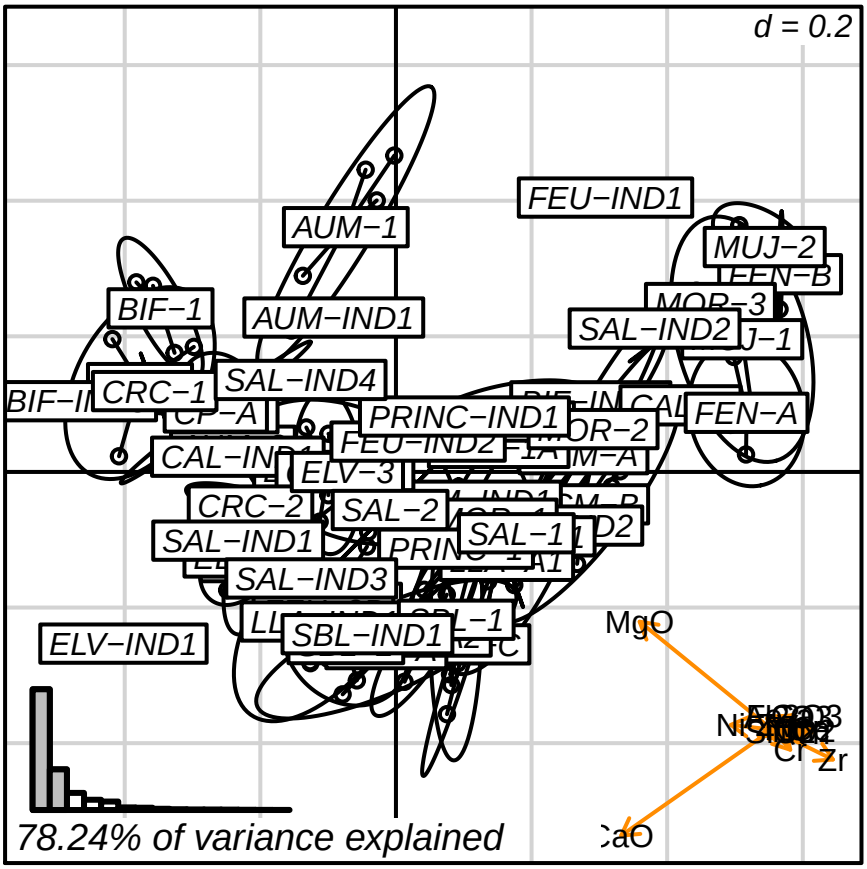


Figure 3.4: default with groups

You can also visualize how the projection of points respond to a given typology (in this case, the chemical reference groups defined in previous studies).

```
biplot_2d(prot1,
          groups = factor_list$ChemGroup,
          output_type = "preview")
```

To get a prettier plot or match your research needs, you can play with the options given by the **biplot_2d**.
Note that groups are by default marked using inertia ellipses. They can only be interpreted as confidence ellipses if each group can be assumed to be normally distributed in all variables considered (see more details on the scaling factor "group_ellipse_cex" in the help file). This is often not reasonable concerning groups that are either too small (< 10) or contain subgroups.
In the argument "test_text" you can introduce the "text" function of the "prot1_tests" object.

```
biplot_2d(prot1,
          groups = factor_list$ChemGroup,
          group_color = color_list$ChemGroup,
          group_label_cex = 0.6,
          invert_coordinates = c(TRUE, TRUE),
          arrow_label_cex = 0.7,
          test_text = prot1_tests$text(prot1_tests),
          test_cex = 0.8,
          test_fig = c(0, 0.5, 0.65, .99),
          output_type = "preview")
```

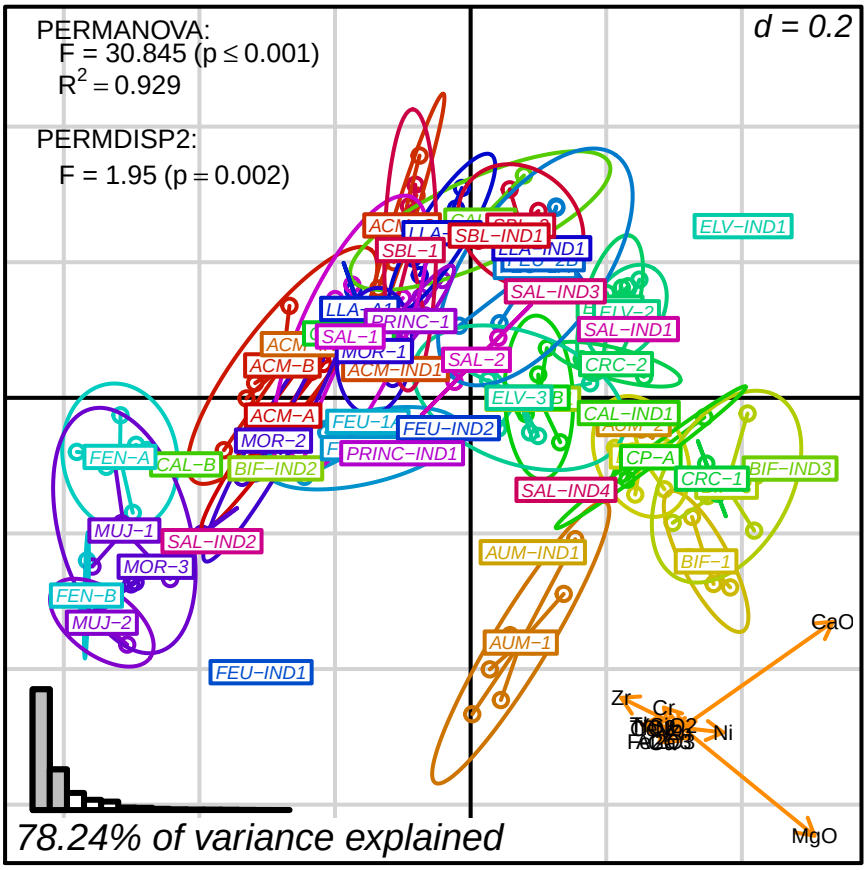


Figure 3.5: tunning appearance, groups with colors

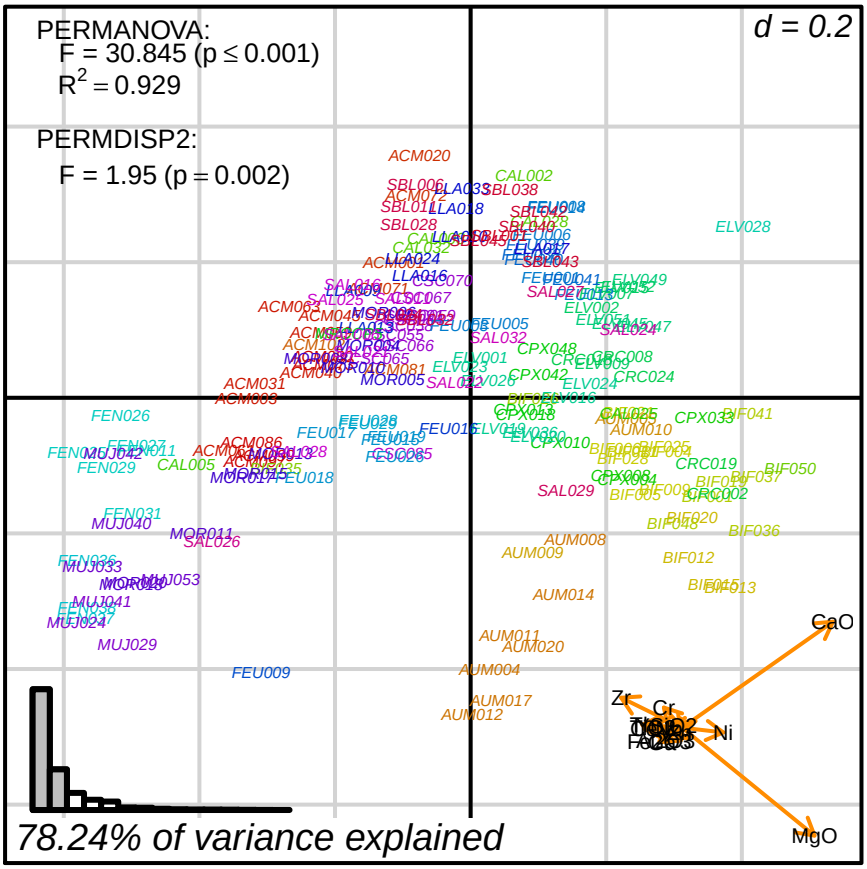


Figure 3.6: labeled points

```
biplot_2d(prot1,
          groups = factor_list$ChemGroup,
          group_color = color_list$ChemGroup,
          group_label_cex = 0,
          group_star_cex = 0,
          group_ellipse_cex = 0,
          point_type = "label",
          point_label = row.names(cleanAmphorae)[!isShipwreck],
          point_label_cex = 0.5,
          invert_coordinates = c(TRUE, TRUE),
          arrow_label_cex = 0.7,
          test_text = prot1_tests$text(prot1_tests),
          test_cex = 0.8,
          test_fig = c(0, 0.5, 0.65, .99),
          output_type = "preview")
```

It is also possible to save 2D biplots into various file formats (png, tiff, jpeg, eps):

```
# better PNG version
biplot_2d(prot1,
          ordination_method = "PCA",
          invert_coordinates = c (TRUE,TRUE),
          grid_cex = 2.5,
          ylim = c(-.1,.1),
          point_type = "point",
          groups = factor_list$ChemGroup,
          group_color = color_list$ChemGroup,
          group_label_cex = 1.5,
          arrow_label_cex = 2,
          arrow_cex = 0.2,
          arrow_lwd = 2.5,
          arrow_fig = c(.6,.95,0,.35),
          subtitle_cex = 2.5,
          test_text = prot1_tests$text(prot1_tests),
          test_fig =c(0, 0.5, 0.62, .99),
          test_cex = 2,
          fitAnalysis_fig = c(0,.7,.05,.5),
          # saving settings
          file_name = "Prot1_Biplot2D",
          directory = directories$prot1,
          width = 1000, height = 1000,
          output_type = "png")
```

3.4.2 Biplot 3D

Most 2D options are also available when generating 3D biplots. Consult the help file for details.

```
?biplot_3d
```

```
biplot_3d(prot1,
          ordination_method = "PCA",
          groups = factor_list$ChemGroup,
          group_color = color_list$ChemGroup,
          point_type = "point",
          group_representation = "stars",
          star_centroid_radius = 0,
          star_label_cex = .8,
          test_text = prot1_tests$text(prot1_tests),
          test_cex = 1.25,
          test_fig = c(0, 0.5, 0.65, .99))
```

You can save an animated GIF and a PNG snapshot using the **animation** function.

```
biplot2d3d::animation(directory = directories$prot1,
                      file_name = "Prot1_Biplot3D")
```

You will need to install ImageMagick to be able to generate the GIF animation.

In these images, you get what you would see when running the biplot_3d function in a regular R session.

NOTE: Animated GIF will not be displayed in the pdf version of this document.

3.5 Comparing CoDa transformations

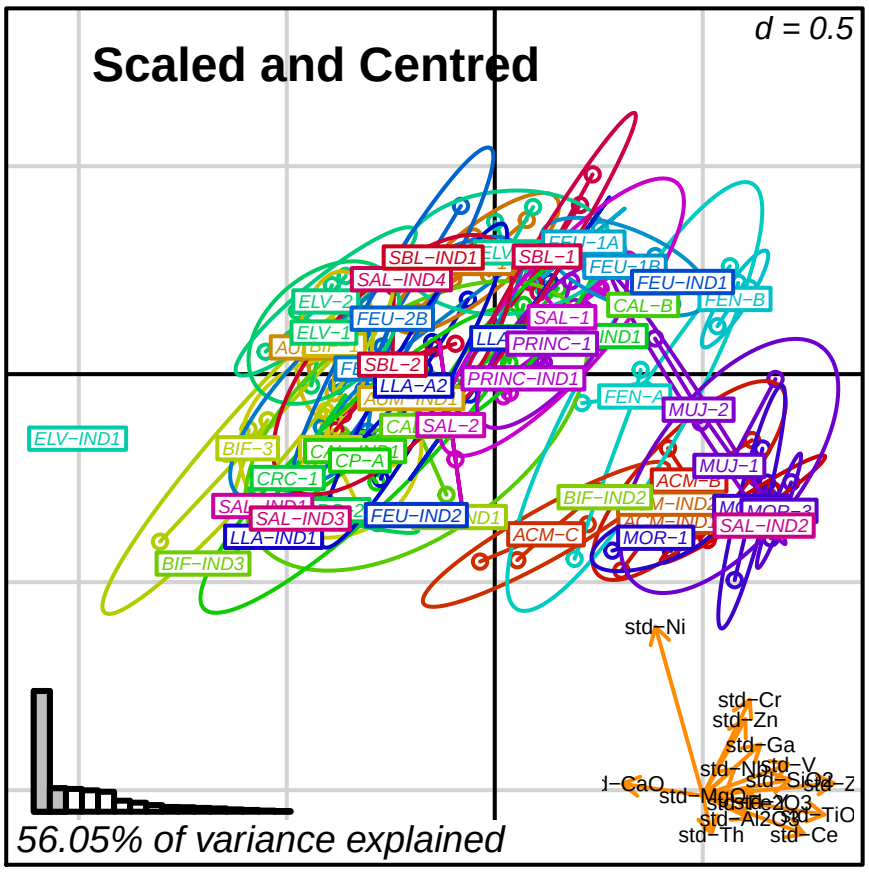
There is a lot of debate on which transformation is useful–or even *valid*–for analyzing geochemical compositions in Archaeometry. We show here how you can compare the results of applying different transformations to the same dataset.

First, create different ordination objects for each type of CoDa transformation that you wish to compare:

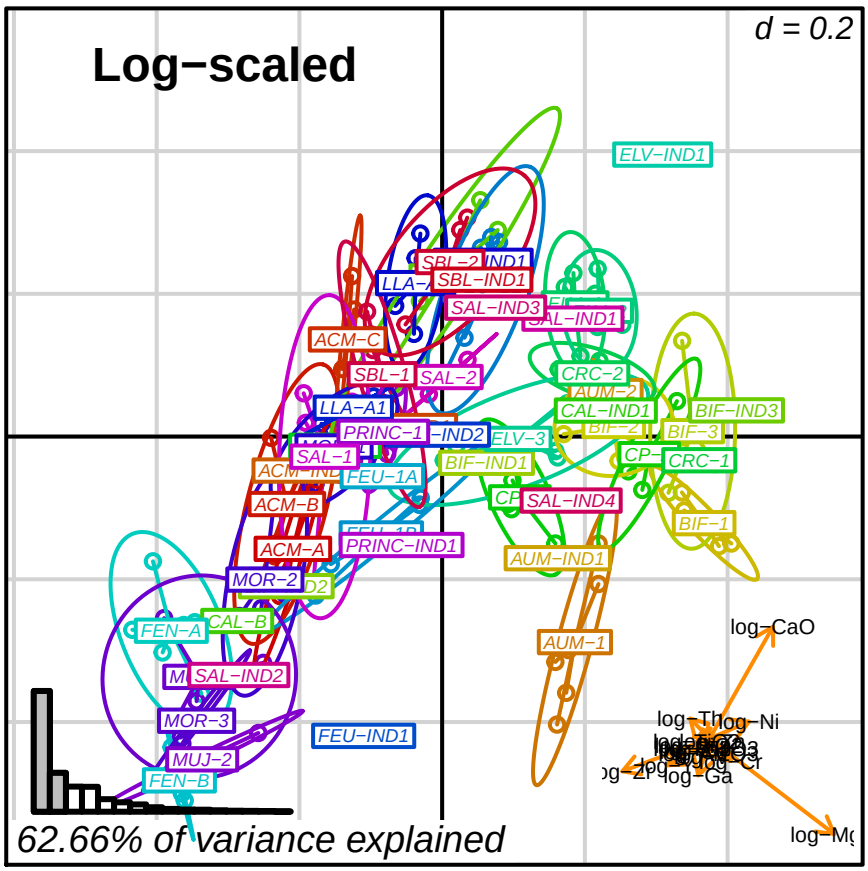
```
prot1_std <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "1", # select protocol 1
                              coda_override = chemVars16,
                              coda_transformation = "std")
#> [1] "56.05% of variance explained in 2D"
#> [1] "64.91% of variance explained in 3D"
#> [1] "Protocol 1 ended."

prot1_log <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "1", # select protocol 1
                              coda_override = chemVars16,
                              coda_transformation = "log")
#> [1] "62.66% of variance explained in 2D"
#> [1] "72.53% of variance explained in 3D"
#> [1] "Protocol 1 ended."

prot1_ALR <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "1", # select protocol 1
                              coda_override = chemVars16,
                              coda_transformation = "ALR",
                              # this is the divisor component
                              coda_alr_base = "Fe2O3")
#> [1] "70.3% of variance explained in 2D"
#> [1] "81.75% of variance explained in 3D"
#> [1] "Protocol 1 ended."

prot1_CLR <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "1", # select protocol 1
                              coda_override = chemVars16,
                              coda_transformation = "CLR")
#> [1] "64.16% of variance explained in 2D"
#> [1] "75.57% of variance explained in 3D"
#> [1] "Protocol 1 ended."
```

Then, create the respective biplots:

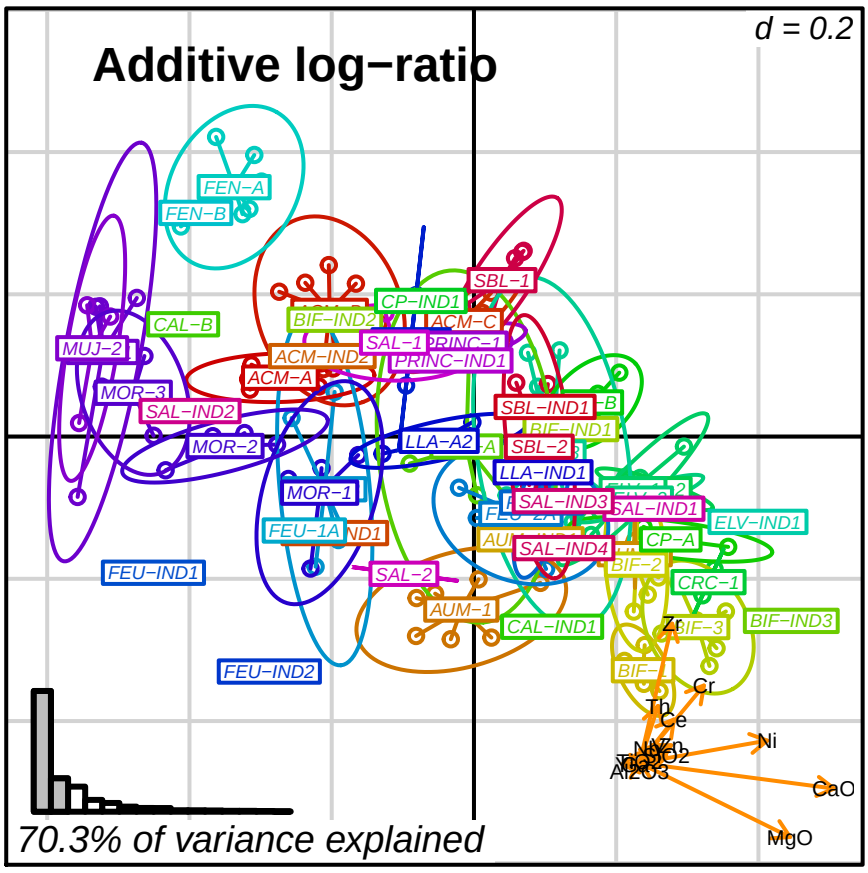
```
biplot2d3d::biplot_2d(prot1_std,
                      groups = factor_list$ChemGroup,
                      group_color = color_list$ChemGroup,
                      group_label_cex = 0.6,
                      arrow_label_cex = 0.7,
                      ylim = c(-0.9, 0.6),
                      x_title = "Scaled and Centred",
                      x_title_cex = 1.5,
                      x_title_fig = c(0.05, 0.9, 0.85, 1),
                      output_type = "preview")
```



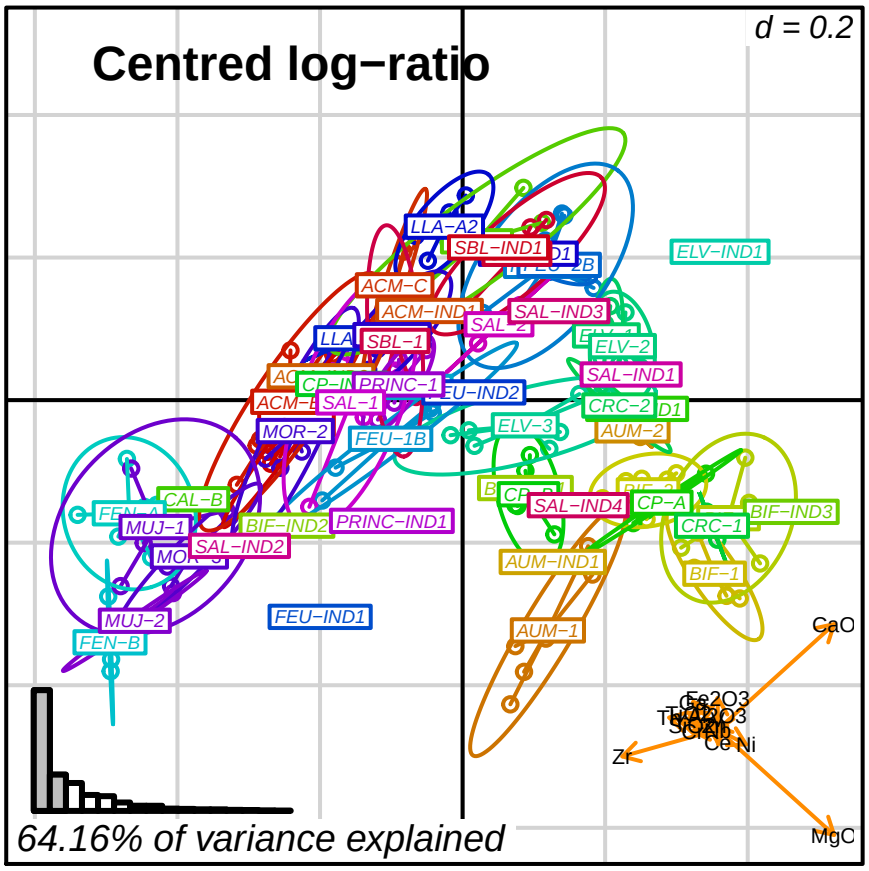
```
biplot2d3d::biplot_2d(prot1_log,
                      groups = factor_list$ChemGroup,
                      group_color = color_list$ChemGroup,
                      group_label_cex = 0.6,
                      arrow_label_cex = 0.7,
                      ylim = c(-0.6, 0.6),
                      x_title = "Log-scaled",
                      x_title_cex = 1.5,
                      x_title_fig = c(0.05, 0.9, 0.85, 1),
                      output_type = "preview")
```



```
biplot2d3d::biplot_2d(prot1_ALR,
                      groups = factor_list$ChemGroup,
                      group_color = color_list$ChemGroup,
                      group_label_cex = 0.6,
                      arrow_label_cex = 0.7,
                      ylim = c(-0.6, 0.6),
                      x_title = "Additive log-ratio",
                      x_title_cex = 1.5,
                      x_title_fig = c(0.05, 0.9, 0.85, 1),
                      output_type = "preview")
```



```
biplot2d3d::biplot_2d(prot1_CLR,
                      groups = factor_list$ChemGroup,
                      group_color = color_list$ChemGroup,
                      group_label_cex = 0.6,
                      arrow_label_cex = 0.7,
                      ylim = c(-0.5, 0.4),
                      x_title = "Centred log-ratio",
                      x_title_cex = 1.5,
                      x_title_fig = c(0.05, 0.9, 0.85, 1),
                      output_type = "preview")
```



Chapter 4

Protocol 2 - Petrographic data

The following example applies protocol 2 to confirm workshops' petrographic groups.

Protocol 2 consist in:

1. Select ordinal ***petrographic*** data;
2. Transform to ***ranks***;
3. ***Extended Gower distance***, using:

a. **Relative ranking difference** (RRD)
b. **Neighbor interchange** (NI)

4. Apply ordination procedure:

a. ***Principal Coordinates Analysis*** (PCoA)
b. ***Non-metric Dimensional Scaling*** (NMDS)

5. Perform ***PERMANOVA & PERMDISP*** tests;

Last, search for outliers and re-do protocol excluding outliers.

NOTE: The initial procedures must be ran at least once before any protocol can be applied.

The key references on the Extended Gower distance are:

> Pavoine, S., Vallet, J., Dufour, A.-B., Gachet, S., Daniel, H., 2009. On the challenge of treating various types of variables: application for improving the measurement of functional diversity. Oikos 118, 391-402. doi:10.1111/j.1600-0706.2008.16668.x
>
> Podani, J., 1999. Extending Gower's General Coefficient of Similarity to Ordinal Characters on JSTOR. Taxon 48, 331-340. doi:10.2307/1224438

Gower, J.C., 1971. A General Coefficient of Similarity and Some of Its Properties. Biometrics 27, 857-871. doi:10.2307/2528823

4.1 Ordination procedure

Depending on which type of distance calculation (RRD/NI), protocol 2 performs different ordination methods (PCoA/NMDS). Both PCoA and NMDS require specifying the number of dimensions in which to project the data. Therefore, you must generate specific 2D and 3D ordination objects:

```
prot2a_2d <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "2a", # select protocol 2a (RRD & PCoA)
                              exception_columns = excep_cols,
                              variable_tags = varCode)

prot2b_2d <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "2b", # select protocol 2a (NI & NMDS)
                              exception_columns = excep_cols,
                              variable_tags = varCode)

prot2a_3d <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "2a", # select protocol 2a (RRD & PCoA)
                              exception_columns = excep_cols,
                              variable_tags = varCode,
                              dimensions = 3)

prot2b_3d <- apply_ordination(cleanAmphorae[!isShipwreck,],
                              "2b", # select protocol 2a (NI & NMDS)
                              exception_columns = excep_cols,
                              variable_tags = varCode,
                              dimensions = 3)
```

The ordination objects generated with protocol 2 are different from those in protocol 1 since it uses different functions. However, the main components are still the same: the projection of observations or *scores* (**points**) and of variables or *loadings*.

```
class(prot2a_2d)
#> [1] "list"
names(prot2a_2d)
#>  [1] "points"        "eig"           "x"             "ac"
#>  [5] "GOF"           "sub2D"         "GOF2_2D"       "loadings"
#>  [9] "variable_tags" "name"          "dist_matrix"
class(prot2b_2d)
#> [1] "metaMDS" "monoMDS"
names(prot2b_2d)
#>  [1] "nobj"          "nfix"          "ndim"          "ndis"
#>  [5] "ngrp"          "diss"          "iidx"          "jidx"
#>  [9] "xinit"         "istart"        "isform"        "ities"
#> [13] "iregn"         "iscal"         "maxits"        "sratmx"
#> [17] "strmin"        "sfgrmn"        "dist"          "dhat"
#> [21] "points"        "stress"        "grstress"      "iters"
#> [25] "icause"        "call"          "model"         "distmethod"
#> [29] "distcall"      "distance"      "converged"     "tries"
#> [33] "engine"        "species"       "data"          "init_seed"
#> [37] "trymax"        "sub_stress"    "sub2D"         "GOF2_2D"
#> [41] "loadings"      "variable_tags" "name"          "dist_matrix"
```

4.2 Test the given fabric groups

The fabric groups defined in previous studies can be tested against Protocol 2 distance matrices. We can use either "prot2a_2d$dist_matrix" or "prot2a_3d$dist_matrix", because they are the same. Remember that this test batch may take several minutes.

```
prot2a_tests <- test_groups(prot2a_2d$dist_matrix,
                            factor_list$FabricGroup)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
prot2b_tests <- test_groups(prot2b_2d$dist_matrix,
                            factor_list$FabricGroup)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> Warning in vegan::betadisper(distMatrix, groups): some squared distances
#> are negative and changed to zero
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

These tests were explained in protocol 1.

4.3 Biplots

The details on how to create biplots is already explained in protocol 1. Concerning protocol 2, we can compare the results of version *2a* (RRD, PCoA) and *2b* (NI, NMDS).

4.3.1 Biplot 2D

```
arrows_label_adj <- rbind(c(.5,.8),c(.5,1),c(.5,1),c(.5,0),c(.5,1),
                          c(.5,0),c(0,.5))
row.names(arrows_label_adj) <- c("L48","L24","L5","L36","S7",
                                 "S8","S11")

biplot2d3d::biplot_2d(prot2a_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,TRUE),
                      xlim = c(-.26,.35),
                      ylim = c(-.31,.35),
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_label_cex = 0.6,
                      arrow_mim_dist = 0.5,
                      arrow_label_cex = 0.6,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      subtitle = prot2a_2d$sub2D,
                      test_text = prot2a_tests$text(prot2a_tests),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.62, .99),
                      fitAnalysis_fig = c(0,.7,.05,.5),
                      output_type = "preview")
```

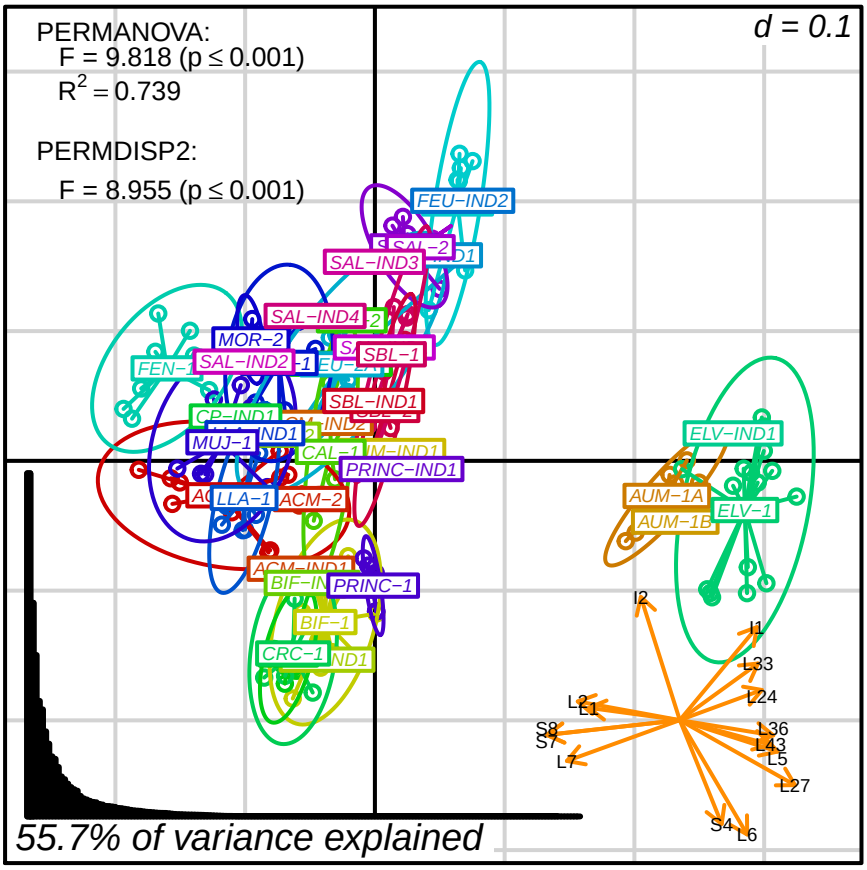


Figure 4.1: protocol 2a

```
arrows_label_adj <- rbind(c(.5,1),c(.5,0),c(.5,1),c(.5,1),c(.5,0),
                          c(0,.5),c(1,.5))
row.names(arrows_label_adj) <- c("S7","S8","CLAY","L24","L43",
                                 "L5","L36")

biplot2d3d::biplot_2d(prot2b_2d,
                      ordination_method = "NMDS",
                      xlim = c(-.42,.38),
                      ylim = c(-.45,.25),
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_label_cex = 0.6,
                      arrow_mim_dist = .5,
                      arrow_label_cex = 0.6,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      subtitle = prot2b_2d$sub2D,
                      test_text = prot2b_tests$text(prot2b_tests),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.62, .99),
                      fitAnalysis_stress_axis_cex = 0.8,
                      fitAnalysis_fig = c(.1,.6,.1,.4),
                      output_type = "preview")
```

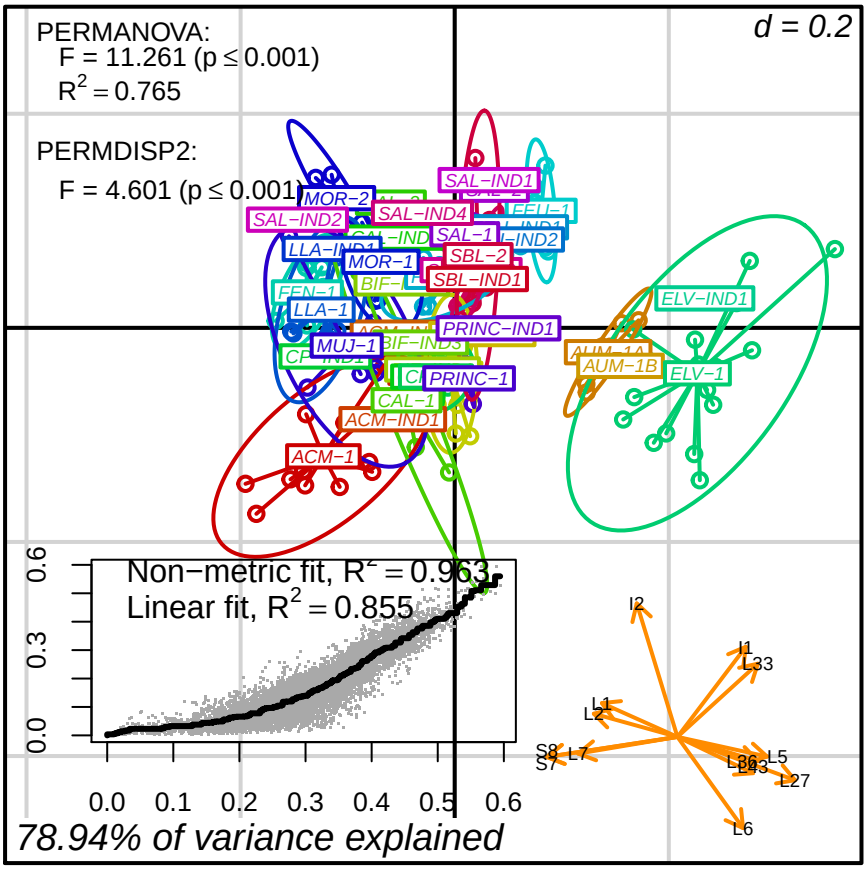


Figure 4.2: protocol 2b

```
                      test_cex = 1.25,
                      test_fig = c(0, 0.5, 0.65, .99),
                      view_zoom = 0.9)

biplot2d3d::animation(directory = directories$prot2,
                      file_name = "Prot2a_Biplot3D")
```

4.3.2 Biplot 3D

NOTE: Animated GIF will not be displayed in the pdf version of this document.

```
biplot2d3d::biplot_3d(prot2b_3d,
                      ordination_method = "NMDS",
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_representation = "stars",
                      star_centroid_radius = 0,
                      star_label_cex = .8,
                      arrow_min_dist = .5,
                      arrow_body_length = .025,
                      subtitle = prot2b_3d$sub3D,
                      test_text = prot2b_tests$text(prot2b_tests),
                      test_cex = 1.25,
                      test_fig = c(0, 0.5, 0.65, .99),
                      view_zoom = 0.9)

biplot2d3d::animation(directory = directories$prot2,
                      file_name = "Prot2b_Biplot3D")
```

NOTE: Animated GIF will not be displayed in the pdf version of this document.

Chapter 5

Protocol 3 - Geochemical and petrographic data

The following example applies protocol 3 to confirm workshops' provenance groups.

Protocol 3 consist in:

1. Select ***geochemical*** compositional data (CoDa) and ordinal ***petrographic*** data;
2. ***Centred log-ratio transformation*** (clr) and transform to ***ranks***;
3. ***Extended Gower distance***, using **Relative ranking difference** (RRD);
4. Apply ***Principal Coordinates Analysis*** (PCoA);
5. Perform ***PERMANOVA & PERMDISP*** tests;

Last, search for outliers and re-do protocol excluding outliers.

NOTE: The initial procedures must be ran at least once before any protocol can be applied.

See protocol 2, for consulting references on the extended Gower distance.

5.1 Ordination procedure

Protocol 3 performs PCoA on a distance matrix calculated with Extended Gower coefficient of dissimilarity, combining Euclidean distances on transformed compositional data (50%) and RRD on ranked petrographic data (50%). As in protocol 2, PCoA requires specifying the number of dimensions and so you must 2D and 3D ordination objects separately:

```
prot3_2d <- apply_ordination(cleanAmphorae[!isShipwreck,], # no shipwrecks
                             "3", # select protocol 3
                             exception_columns = excep_cols,
                             variable_tags = varCode,
```

```
                              coda_override = chemVars16,
                              coda_transformation_method = "CLR")

prot3_3d <- apply_ordination(cleanAmphorae[!isShipwreck,], # no shipwrecks
                              "3", # select protocol 3
                              exception_columns = excep_cols,
                              variable_tags = varCode,
                              coda_override = chemVars16,
                              coda_transformation = "CLR",
                              dimensions = 3)
```

5.2 Simplify CoDa names

We may want to simplify the names of the transformed variables before plotting them in a biplot.

```
prot3_2d <- simplify_coda_names(prot3_2d)
prot3_3d <- simplify_coda_names(prot3_3d)
```

5.3 Test the given provenance groups

Because protocol 3 uses both geochemical and petrographic information, we can test the provenance assigned to the amphorae samples.

```
prot3_tests <- test_groups(prot3_2d$dist_matrix,
                           factor_list$ProvGroup)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

These tests were explained in protocol 1.

5.4 Biplots

The details on how to create biplots is already explained in protocol 1. Unlike protocol 2, protocol 3 only generates one kind of projection (RRD, PCoA).

```
arrows_label_adj <- rbind(c(.5,.8),c(.5,1),c(.5,1),c(.5,0),c(.5,1),
                          c(.5,0),c(0,.5))
row.names(arrows_label_adj) <- c("L48","L24","L5","L36","S7",
                                 "S8","S11")

biplot2d3d::biplot_2d(prot3_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,TRUE),
                      ylim = c(-.3,.29),
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_label_cex = 0.6,
                      arrow_mim_dist = 0.5,
                      arrow_label_cex = 0.6,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      subtitle = prot3_2d$sub2D,
                      test_text = prot3_tests$text(prot3_tests),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.65, .99),
                      fitAnalysis_fig = c(0,.7,.05,.5),
                      output_type = "preview")
```

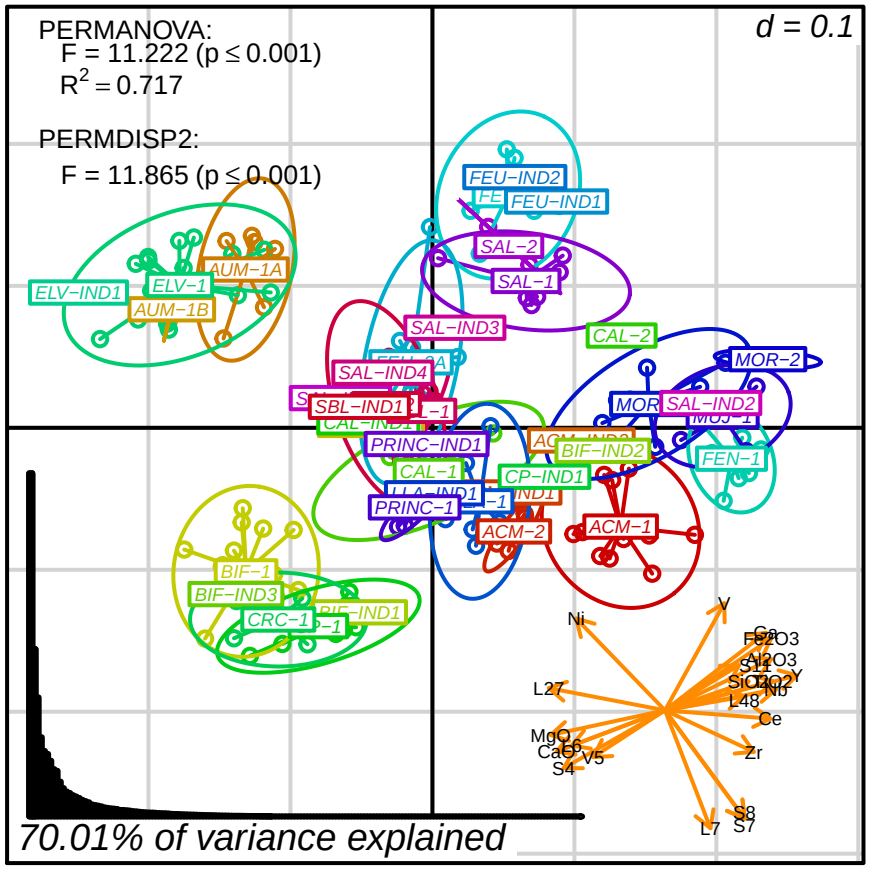


Figure 5.1: protocol 3

5.4.2 Biplot 3D

```
biplot2d3d::biplot_3d(prot3_3d,
                      ordination_method = "PCoA",
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_representation = "stars",
                      star_centroid_radius = 0,
                      star_label_cex = .8,
                      arrow_min_dist = .5,
                      arrow_body_length = .025,
                      subtitle = prot3_3d$sub3D,
                      test_text = prot3_tests$text(prot3_tests),
                      test_cex = 1.25,
                      test_fig = c(0, 0.5, 0.65, .99),
                      view_zoom = 0.9)

biplot2d3d::animation(directory = directories$prot3,
                      file_name = "Prot3_Biplot3D")
```

NOTE: Animated GIF will not be displayed in the pdf version of this document.

Chapter 6

Protocol 4 - Provenance data

The following example applies protocol 4 to confirm workshops' provenance groups.

Protocol 4 consist in:

1. Select provenance-specific variables in ***geochemical*** compositional data (CoDa) and ordinal ***petrographic*** data;
2. ***Centred log-ratio transformation*** (clr) and transform to ***ranks***;
3. ***Extended Gower coefficient of dissimilarity***, using **Relative ranking difference** (RRD);
4. Apply ***Principal Coordinates Analysis*** (PCoA);
5. Perform ***PERMANOVA & PERMDISP*** tests;

Last, search for outliers and re-do protocol excluding outliers.

NOTE: The initial procedures must be ran at least once before any protocol can be applied.

6.1 Ordination procedure

As protocol 3, protocol 4 performs PCoA on a distance matrix calculated with Extended Gower coefficient of dissimilarity, combining Euclidean distances on transformed compositional data (50%) and RRD on ranked petrographic data (50%).

```
prot4_2d <- apply_ordination(cleanAmphorae[!isShipwreck,],
                             "4", # select protocol 4
                             exception_columns = excep_cols,
                             variable_tags = varCode,
                             coda_override = chemVars16,
                             coda_transformation_method = "CLR")
```

```
prot4_3d <- apply_ordination(cleanAmphorae[!isShipwreck,],
                             "4", # select protocol 4
                             exception_columns = excep_cols,
                             variable_tags = varCode,
                             coda_override = chemVars16,
                             coda_transformation_method = "CLR",
                             dimensions = 3)
```

However, protocol 4 uses a finer selection of petrographic variables, which are considered indicative of provenance (raw materials) rather than technology. Compare the number of variables in protocol 3 and 4:

Protocol 3	Protocol 4
78	59

6.2 Simplify CoDa names

We may want to simplify the names of the transformed variables before plotting them in a biplot.

```
prot4_2d <- simplify_coda_names(prot4_2d)
prot4_3d <- simplify_coda_names(prot4_3d)
```

6.3 Test the given provenance groups

With protocol 4, we can test the provenance assigned to the amphorae samples based only on provenance-specific variables.

```
prot4_tests <- test_groups(prot4_2d$dist_matrix,
                           factor_list$ProvGroup)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

These tests were explained in protocol 1.

6.4 Biplots

The details on how to create biplots is already explained in protocol 1. As protocol 3, protocol 4 only generates one kind of projection (RRD, PCoA).

```
arrows_label_adj <- rbind(c(.5,1),c(0,0),c(1,.5),c(0,1),c(1,0),
                          c(0,.5),c(.5,1),c(1,.5),c(.5,1))
row.names(arrows_label_adj) <- c("CaO","S4","S7","S8","Ce",
                                 "Nb","Al2O3","S11","Fe2O3")

biplot2d3d::biplot_2d(prot4_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,FALSE),
                      ylim = c(-.35,.32),
                      point_type = "point",
                      groups = factor_list$ProvGroup,
                      group_color = color_list$ProvGroup,
                      group_label_cex = 0.6,
                      arrow_mim_dist = .5,
                      arrow_label_cex = 0.6,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      subtitle = prot4_2d$sub2D,
                      test_text = prot4_tests$text(prot4_tests),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.62, .99),
                      fitAnalysis_fig = c(0,.7,.05,.5),
                      output_type = "preview")
```

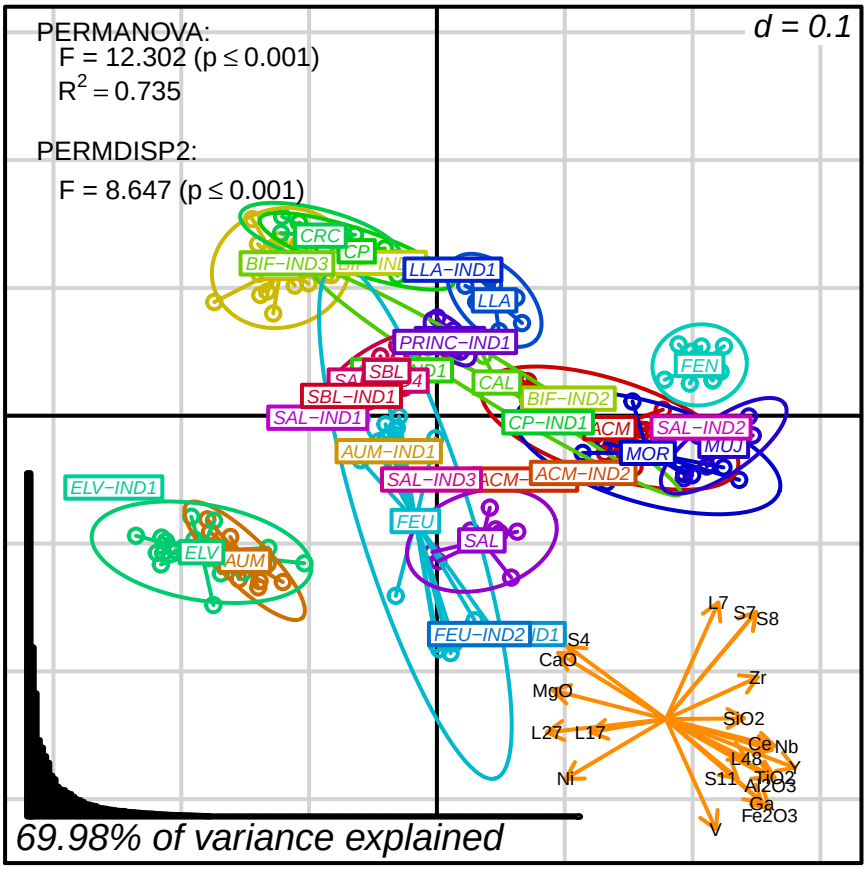


Figure 6.1: protocol 4

6.4.2 Biplot 3D

```
biplot2d3d::biplot_3d(prot4_3d,
                      ordination_method = "PCoA",
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_representation = "stars",
                      star_centroid_radius = 0,
                      star_label_cex = .8,
                      arrow_min_dist = .5,
                      arrow_body_length = .025,
                      subtitle = prot4_3d$sub3D,
                      test_text = prot4_tests$text(prot4_tests),
                      test_cex = 1.25,
                      test_fig = c(0, 0.5, 0.65, .99),
                      view_zoom = 0.9)

biplot2d3d::animation(directory = directories$prot4,
                       file_name = "Prot4_Biplot3D")
```

NOTE: Animated GIF will not be displayed in the pdf version of this document.

Chapter 7

Protocol 4 - Provenance data with shipwrecks

The following example applies protocol 4 to confirm shipwrecks samples attribution to workshops' provenance groups.

Protocol 4 consist in:

1. Select provenance-specific variables in ***geochemical*** compositional data (CoDa) and ordinal ***petrographic*** data;
2. ***Centred log-ratio transformation*** (clr) and transform to ***ranks***;
3. ***Extended Gower coefficient of dissimilarity***, using **Relative ranking difference** (RRD);
4. Apply ***Principal Coordinates Analysis*** (PCoA);
5. Perform ***PERMANOVA & PERMDISP*** tests;

Last, search for outliers and re-do protocol excluding outliers.

NOTE: The initial procedures must be ran at least once before any protocol can be applied.

7.1 Ordination procedure

As protocol 3, protocol 4 performs PCoA on a distance matrix calculated with Extended Gower coefficient of dissimilarity, combining Euclidean distances on transformed compositional data (50%) and RRD on ranked petrographic data (50%). In this case, we are not filtering out the shipwreck samples, but we do exclude the true outliers (IND, observations with no group assigned) so they don't pollute visualization.

```
prot4_Shipwreck_2d <- apply_ordination(# no true outliers
                                       cleanAmphorae[!isTrueIND,],
```

```
                                        "4", # select protocol 4
                                        exception_columns = excep_cols,
                                        variable_tags = varCode,
                                        coda_override = chemVars16,
                                        coda_transformation_method = "CLR")

prot4_Shipwreck_3d <- apply_ordination(# no true outliers
                                        cleanAmphorae[!isTrueIND,],
                                        "4", # select protocol 4
                                        exception_columns = excep_cols,
                                        variable_tags = varCode,
                                        coda_override = chemVars16,
                                        coda_transformation_method = "CLR",
                                        dimensions = 3)
```

7.2 Simplify CoDa names

We may want to simplify the names of the transformed variables before plotting them in a biplot.

```
prot4_Shipwreck_2d <- simplify_coda_names(prot4_Shipwreck_2d)
prot4_Shipwreck_3d <- simplify_coda_names(prot4_Shipwreck_3d)
```

7.3 Test the given provenance groups

We can test the provenance assigned to shipwrecks' amphorae samples together with those found and assigned to the workshops.

```
prot4_Shipwreck_tests <- test_groups(prot4_Shipwreck_2d$dist_matrix,
                                     factor_list_Shipwreck$ProvGroup)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

These tests were explained in protocol 1.

7.4 Biplots

The details on how to create biplots is already explained in protocol 1. Protocol 4 generates PCoA projections.

```
arrows_label_adj <- rbind(c(.5,0),c(.5,1),c(.5,0),c(.5,1),c(.5,0),
                          c(.5,1),c(.8,0),c(1,.5),c(.5,0),c(1,.2),
                          c(.5,1),c(.2,.7))
row.names(arrows_label_adj) <- c("S7","S8","S4","CaO","MgO",
                                 "S11","L48","SiO2","Ce","Nb",
                                 "Th","TiO2")

biplot2d3d::biplot_2d(prot4_Shipwreck_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,TRUE),
                      ylim = c(-.3,.25),
                      point_type = "point",
                      groups = factor_list_Shipwreck$ProvGroup,
                      group_color = color_list_Shipwreck$ProvGroup,
                      group_label_cex = 0.6,
                      arrow_mim_dist = .5,
                      arrow_label_cex = 0.6,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      subtitle = prot4_Shipwreck_2d$sub2D,
                      test_text =
                        prot4_Shipwreck_tests$text(prot4_Shipwreck_tests),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.62, .99),
                      fitAnalysis_fig = c(0,.7,.05,.5),
                      output_type = "preview")
```

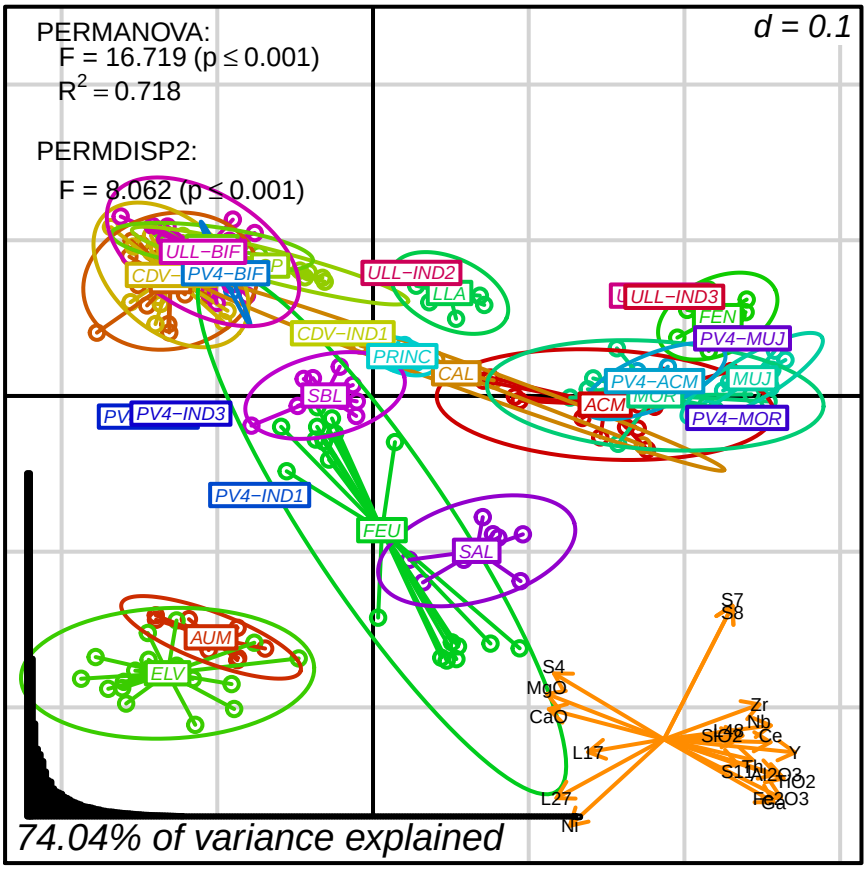


Figure 7.1: protocol 4 with shipwrecks

7.4.2 Biplot 3D

```
biplot2d3d::biplot_3d(prot4_Shipwreck_3d,
                      ordination_method = "PCoA",
                      point_type = "point",
                      groups = factor_list_Shipwreck$FabricGroup,
                      group_color = color_list_Shipwreck$FabricGroup,
                      group_representation = "stars",
                      star_centroid_radius = 0,
                      star_label_cex = .8,
                      arrow_min_dist = .5,
                      arrow_body_length = .025,
                      subtitle = prot4_Shipwreck_3d$sub3D,
                      test_text =
                        prot4_Shipwreck_tests$text(prot4_Shipwreck_tests),
                      test_cex = 1.25,
                      test_fig = c(0, 0.5, 0.65, .99),
                      view_zoom = 0.9)

biplot2d3d::animation(directory = directories$prot4_Shipwreck,
                       file_name = "Prot4_Shipwreck_Biplot3D")
```

NOTE: Animated GIF will not be displayed in the pdf version of this document.

Chapter 8

Interpreting biplots

This section is a reminder of the possible caveats of interpreting multivariate projections (biplots) as bivariate plots (e.g., scatter plots).

The first big difference between biplots and scatter plots lies in their names. Contrary to common intuition, 'bi' in 'biplot' does not stand for two **axes** or **dimensions** but the two ***plots*** that share the same axes or dimensions. Graphically, those plots consist of points, which is analogous to a scatterplot, and arrows, which represent the covariance between variables and the dimensions of the plot. As these dimensions are given by an ordination method (e.g., PCA), they express the fact that the dataset itself has two dimensions (a matrix with rows and columns). Consequently, three-dimensional biplots are still biplots, not 'triplots'.

There is another, more subtle, difference between biplots and scatter plots. The latter will unequivocally place points according to their values in each of the variables considered. Biplots, in turn, are projections of distributions or 'point clouds' that are multidimensional (i.e., multivariate data). Even in the best scenarios, biplots cannot represent such clouds in their full form. Imagine trying to draw a dice on a sheet of paper.

As an example, consider the outcome of protocol 1. In this case, robust PCA generated a good 2D projection (around 78% of variance) where CaO and MgO are the major contributors.

```
# Recover protocol 1 override for variable label positions
arrows_label_adj <- rbind( c(.5, -.5), c(1, .5), c(1.2, 1.2),
                           c(1.2, .4), c(.8, .5), c(0, 0),
                           c(-.2, 1), c(.5, 1.2), c(-.5, .5),
                           c(-.2, .5), c(0, .5), c(0, 0))
row.names(arrows_label_adj) <- c("Fe2O3", "Al2O3", "SiO2",
                                 "TiO2", "MgO", "Th",
```

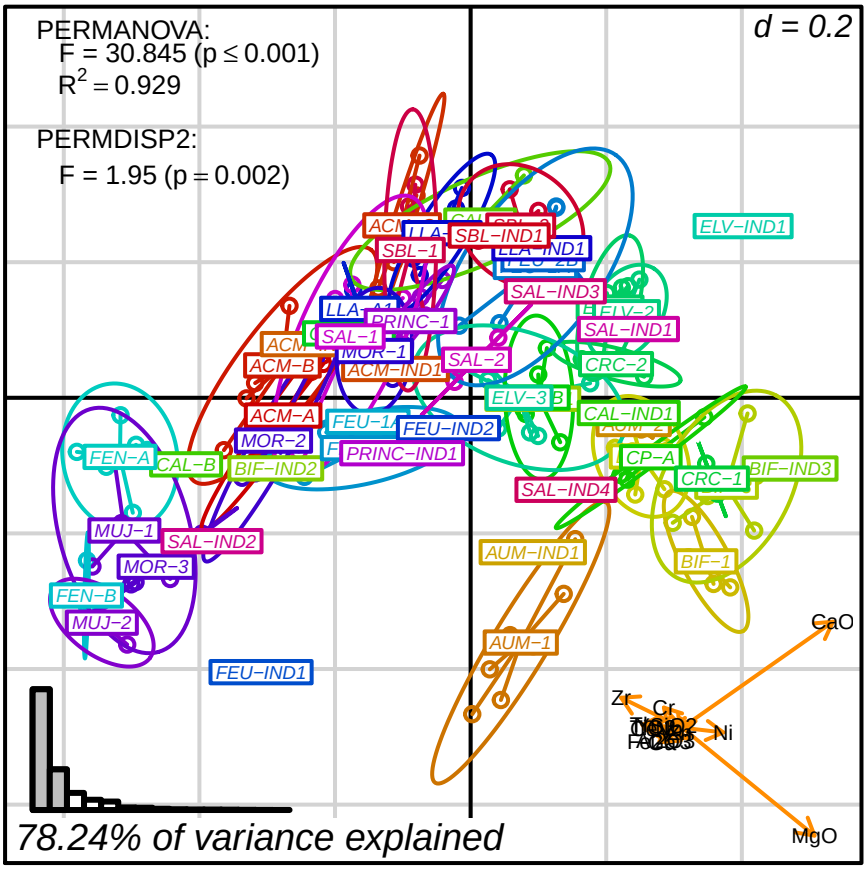


Figure 8.1: Protocol 1, representing and testing chemical reference groups

```
                                "Nb", "Cr", "Ce",
                                "Ga", "Zn", "Y")
```

```
biplot_2d(prot1,
          groups = factor_list$ChemGroup,
          group_color = color_list$ChemGroup,
          group_label_cex = 0.6,
          invert_coordinates = c(TRUE, TRUE),
          arrow_label_cex = 0.7,
          test_text = prot1_tests$text(prot1_tests),
          test_cex = 0.8,
          test_fig = c(0, 0.5, 0.65, .99),
          output_type = "preview")
```

Therefore, we can safely interpret positions in terms of having more or less CaO and MgO. For instance, we can classify observations by levels of CaO content and test it against protocol 1 distance matrix and 2D projection:

```
# Create factor variable containing the classification (5 categories)
CaO_level <- cut(cleanAmphorae$CaO[!isShipwreck], 5)

# Select 5 colours from the 'topo.colors' palette
CaO_level_colors <- topo.colors(nlevels(CaO_level))

# Test the classification
prot1_tests_CaO <- test_groups(prot1$dist_matrix, CaO_level)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

```r
# This is for highlighting CaO arrow
arrow_colors <- rep("darkorange", nrow(prot1$loadings))
arrow_colors[row.names(prot1$loadings) == "CaO"] <- "red"

biplot_2d(prot1,
          groups = CaO_level,
          group_color = CaO_level_colors,
          group_star_cex = 0,
          group_label_cex = 0,
          show_group_legend = TRUE,
          group_legend_title = "CaO",
          group_legend_title_pos = c(0.5,0.9),
          group_legend_text_cex = 0.8,
          group_legend_fig = c(0.7,0.99,0.68,0.95),
          invert_coordinates = c(TRUE, TRUE),
          arrow_label_cex = 0.8,
          arrow_fig = c(.6,.95,0,.35),
          arrow_label_adj_override = arrows_label_adj,
          arrow_color = arrow_colors,
          test_text = prot1_tests_CaO$text(prot1_tests_CaO),
          test_cex = 0.8,
          test_fig = c(0, 0.5, 0.65, .99),
          output_type = "preview")
```

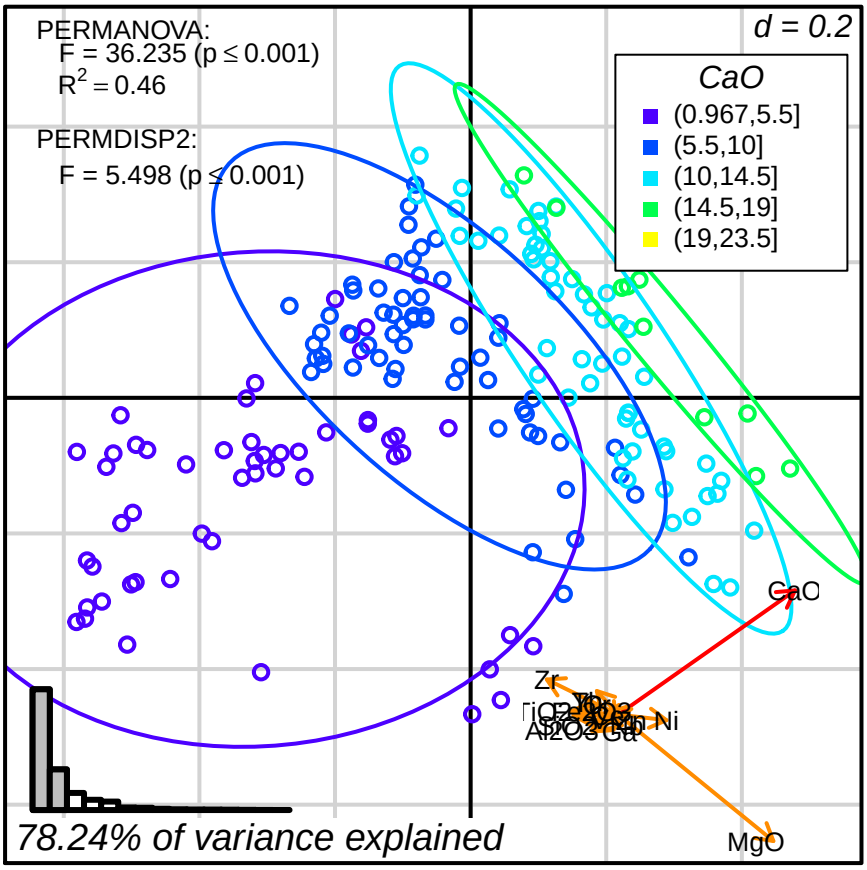


Figure 8.2: Protocol 1, grouping by level of CaO content

However, interpretation is less straightforward when more than two variables contribute significantly to the total variation. Regarding biplots, such situation implies that a smaller portion of variation is represented, and that several variables are stretch on many directions over the two principal coordinates.

For example, protocol 2 gave us a much worse 2D projection (55.7%) where fifteen variables are well represented.

```
# Recover protocol 2a override for variable label positions
arrows_label_adj <- rbind(c(.5,.8),c(.5,1),c(.5,1),c(.5,0),c(.5,1),
                          c(.5,0),c(0,.5))
row.names(arrows_label_adj) <- c("L48","L24","L5","L36","S7",
                                 "S8","S11")

# This will help us select different arrow colours
isDisplayed <-
  row.names(prot2a_2d$loadings) %in% row.names(
    filter_arrows(prot2a_2d$loadings, min_dist = 0.5))
```

```
biplot2d3d::biplot_2d(prot2a_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,TRUE),
                      xlim = c(-.26,.35),
                      ylim = c(-.31,.35),
                      point_type = "point",
                      groups = factor_list$FabricGroup,
                      group_color = color_list$FabricGroup,
                      group_label_cex = 0.6,
                      arrow_mim_dist = 0.5,
                      arrow_label_cex = 0.6,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      subtitle = prot2a_2d$sub2D,
                      test_text = prot2a_tests$text(prot2a_tests),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.65, .99),
                      fitAnalysis_fig = c(0,.7,.05,.5),
                      output_type = "preview")
```

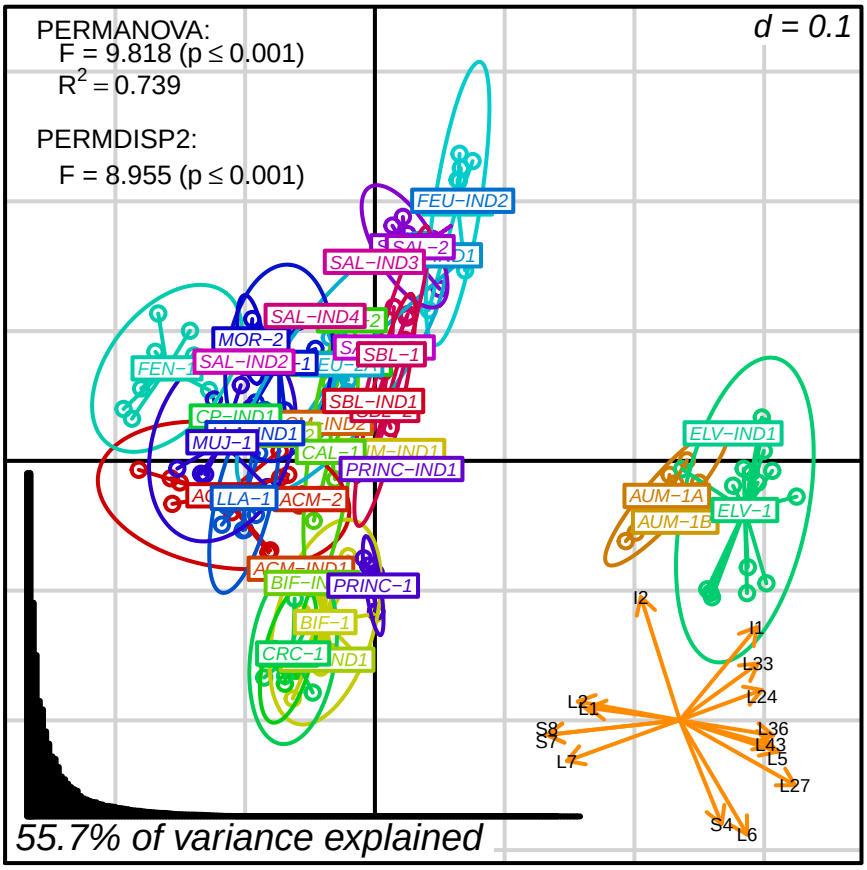


Figure 8.3: protocol 2a, representing and testing fabric groups

Like in protocol 1, we may want to interpret this projection in terms of a single variable. A obvious candidate is I2 since it is displayed long and quite isolated from other variables.

In this case, I2 (or INCLUS_ORIENT) is already a factor variable (classification) with 3 categories (plus "none" as a missing value). Note that the selected dataset will have cases in only two of those categories.

```
# You may want to assure that the true
# categories are corectly represented:
cleanAmphorae <- order_petro(cleanAmphorae)

levels(cleanAmphorae$INCLUS_ORIENT[!isShipwreck])
#> [1] "unparallel"        "slightly parallel" "parallel"
#> [4] "none"

# Declare this factor separately as an object for clearness
I2 <- cleanAmphorae$INCLUS_ORIENT[!isShipwreck]

# Select colours from the 'topo.colors' palette
I2_colors <- topo.colors(nlevels(I2))

# Test the classification
prot1_tests_I2 <- test_groups(prot2a_2d$dist_matrix, I2)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

```r
# This is for highlighting I2 arrow
arrow_colors <- rep("darkorange", nrow(prot2a_2d$loadings))
arrow_colors[row.names(prot2a_2d$loadings) == "I2"] <- "red"
# filter arrows colours, since not all variables are displayed
arrow_colors <- arrow_colors[isDisplayed]

biplot2d3d::biplot_2d(prot2a_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,TRUE),
                      xlim = c(-.26,.35),
                      ylim = c(-.31,.35),
                      groups = I2,
                      group_color = I2_colors,
                      group_star_cex = 0,
                      group_label_cex = 0,
                      show_group_legend = TRUE,
                      group_legend_title = "INCLUS_ORIENT",
                      group_legend_title_pos = c(0.5,0.9),
                      group_legend_text_cex = 0.8,
                      group_legend_fig = c(0.6,0.99,0.68,0.95),
                      arrow_mim_dist = .5,
                      arrow_label_cex = 0.8,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      arrow_color = arrow_colors,
                      subtitle = prot2a_2d$sub2D,
                      test_text = prot1_tests_I2$text(prot1_tests_I2),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.65, .99),
                      output_type = "preview")
```

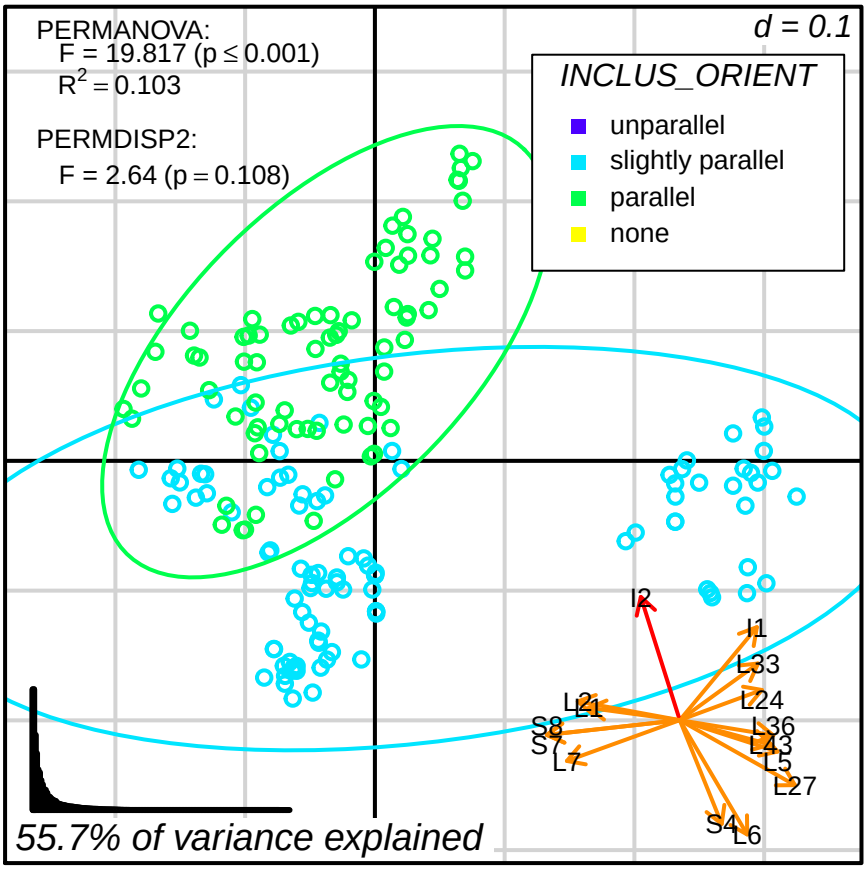


Figure 8.4: Protocol 2a, grouping by INCLUS_ORIENT

This kind of reading becomes increasingly difficult when focusing in variables that are not so well aligned, such as L33 (or COAR_R_CHERT)

```
# Declare this factor separately as an object for clearness
L33 <- cleanAmphorae$COAR_R_CHERT[!isShipwreck]

# Select colours from the 'topo.colors' palette
L33_colors <- topo.colors(nlevels(L33))

# Test the classification
prot1_tests_L33 <- test_groups(prot2a_2d$dist_matrix, L33)
#> [1] "initiating test batch..."
#> [1] "vegan::anosim done."
#> [1] "vegan::betadisper done."
#> [1] "vegan::permutest done."
#> [1] "vegan::adonis done."
#> [1] "Test batch completed."
```

```r
# This is for highlighting L33 arrow
arrow_colors <- rep("darkorange", nrow(prot2a_2d$loadings))
arrow_colors[row.names(prot2a_2d$loadings) == "L33"] <- "red"
# filter arrows colours, since not all variables are displayed
arrow_colors <- arrow_colors[isDisplayed]

biplot2d3d::biplot_2d(prot2a_2d,
                      ordination_method = "PCoA",
                      invert_coordinates = c (TRUE,TRUE),
                      xlim = c(-.26,.35),
                      ylim = c(-.31,.35),
                      groups = L33,
                      group_color = L33_colors,
                      group_star_cex = 0,
                      group_label_cex = 0,
                      show_group_legend = TRUE,
                      group_legend_title = "COAR_R_CHERT",
                      group_legend_title_pos = c(0.5,0.9),
                      group_legend_text_cex = 0.8,
                      group_legend_fig = c(0.6,0.99,0.68,0.95),
                      arrow_mim_dist = .5,
                      arrow_label_cex = 0.8,
                      arrow_fig = c(.6,.95,0,.35),
                      arrow_label_adj_override = arrows_label_adj,
                      arrow_color = arrow_colors,
                      subtitle = prot2a_2d$sub2D,
                      test_text = prot1_tests_L33$text(prot1_tests_L33),
                      test_cex = 0.8,
                      test_fig = c(0, 0.5, 0.65, .99),
                      output_type = "preview")
```

Biplots and ordinal methods (PCA, PCoA, CA, etc.) are exploratory tools that play a game of compromise in order to define the best projections given the whole variance in a dataset. Do not expect them to display patterns that are clear when looking into specific variables. For that kind of analysis, you should use bivariate statistics and graphical displays, such as scatter plots or box plots.

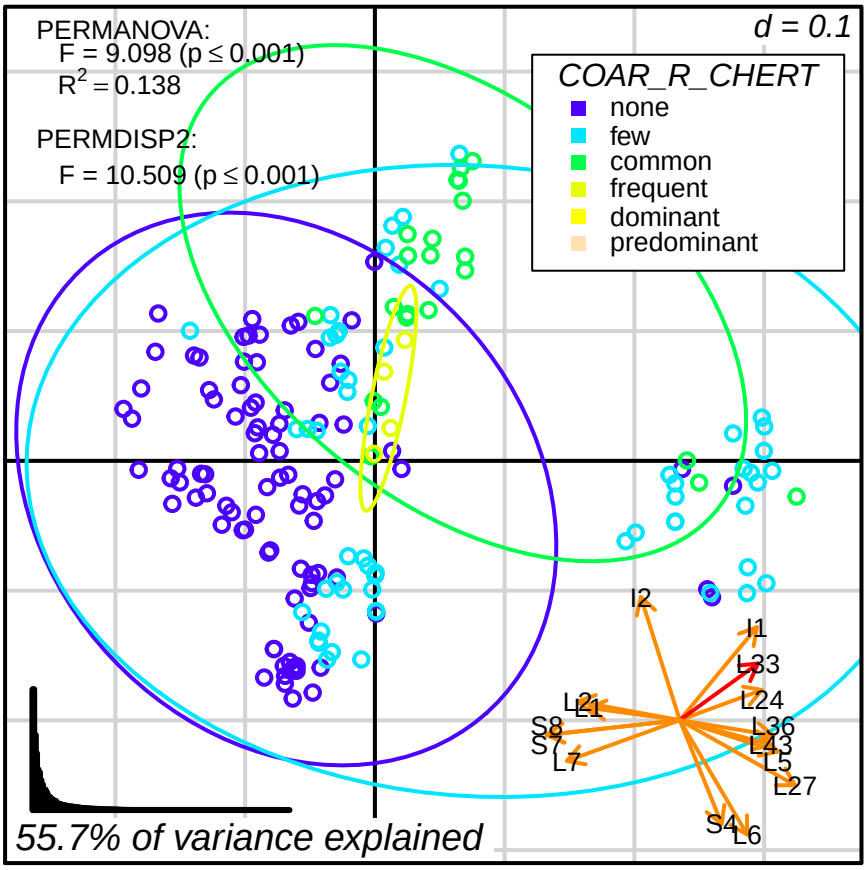


Figure 8.5: Protocol 2a, grouping by COAR_R_CHERT