Introduction to compositions

Serge-Étienne Parent

2013-05-29

Compositional data (CoDa) analysis

Compositional data (CoDa) are intrinsically multivariate data assembled in vectors of positive components showing the relative importance of a set of components on a whole, e.g. proportions, probabilities, concentrations, red-green-blue proportions, etc. They sum to a constant (unity, 100%, 1 000 000 ppm, etc.) and range between 0 and this constant sum.

The closure operation, C, computes the constant sum assignment, as follows.

$$S^D = C(c_1, c_2, ..., c_D) = [\frac{c_1 \kappa}{\sum_{i=1}^{D} c_i}, \frac{c_2 \kappa}{\sum_{i=1}^{D} c_i}, ..., \frac{c_D \kappa}{\sum_{i=1}^{D} c_i}]$$

CoDa have particular and important numerical properties and should be handled with care. Not convinced? Let's have a look at personal expenditures in the United States, in billions of $.

```
data(USPersonalExpenditure)
expends1 <- t(USPersonalExpenditure)
colnames(expends1) <- c("Food & Tob.", "House. Op.", "Med. &
Health", "Pers. Care",
    "Priv. Educ.")
rowSums(expends1)
```

```
##   1940   1945   1950   1955   1960
##  37.61  68.71 102.56 129.70 163.14
```

```
cor(expends1)
```

```
##                Food & Tob. House. Op. Med. & Health Pers. Care
Priv. Educ.
## Food & Tob.         1.0000     0.9798         0.9558     0.9466
0.9823
## House. Op.          0.9798     1.0000         0.9808     0.9555
0.9936
## Med. & Health       0.9558     0.9808         1.0000     0.9914
0.9934
## Pers. Care          0.9466     0.9555         0.9914     1.0000
0.9809
## Priv. Educ.         0.9823     0.9936         0.9934     0.9809
1.0000
```

Now, the correlation between expenditures as ratio on total measured expenditures.

```
expends2 <- expends1/rowSums(expends1)
cor(expends2)
```

```
##                Food & Tob. House. Op. Med. & Health Pers. Care
Priv. Educ.
## Food & Tob.         1.0000    -0.8726       -0.9156    -0.2909
-0.6456
## House. Op.         -0.8726     1.0000        0.6053    -0.1304
0.3167
## Med. & Health      -0.9156     0.6053        1.0000     0.6112
0.7538
## Pers. Care         -0.2909    -0.1304        0.6112     1.0000
0.2773
## Priv. Educ.        -0.6456     0.3167        0.7538     0.2773
1.0000
```

And the correlation between expenditures as ratio, excluding “Food and Tobacco”“ as well as "Household Operation”.

```
expends3 <- expends1[, 3:5]/rowSums(expends1[, 3:5])
cor(expends3)
```

```
##                Med. & Health Pers. Care Priv. Educ.
## Med. & Health         1.0000    -0.4287     -0.4092
## Pers. Care           -0.4287     1.0000     -0.6489
## Priv. Educ.          -0.4092    -0.6489      1.0000
```

And so on…

```
expends4 <- expends1[, c(3, 5)]/rowSums(expends1[, c(3, 5)])
cor(expends4)
```

```
##                Med. & Health Priv. Educ.
## Med. & Health              1          -1
## Priv. Educ.               -1           1
```

Now, the correlation is perfect. Indeed, if on increase by a certain fraction, the other one decreases with the same fraction.

The following table shows the correlation between “Private Education”“ and "Medical and Health”.

```
data.frame(D = c(5, 5, 4, 3, 2), Closure = c("Unclosed",
"Closed", "Closed",
    "Closed", "Closed"), Correlation = c(cor(expends1)[5, 3],
cor(expends2)[5,
    3], cor(expends1[, 2:5]/rowSums(expends1[, 2:5]))[4, 2],
cor(expends3)[1,
    3], cor(expends4)[2, 1]))
```

```
##   D  Closure Correlation
## 1 5 Unclosed      0.9934
## 2 5   Closed      0.7538
## 3 4   Closed      0.6056
## 4 3   Closed     -0.4092
## 5 2   Closed     -1.0000
```

The correlation depends on the basis we assume as analysts: total family expenses, home expenses, parental expenses, etc. Same data, different scales, retuning different correlations. This is what Pearson (1897) named **spurious correlations**. Ordinary log transformations would not correct the problem (try it!). Why? Because CoDa essentially convey relative information, and the constant sum forces at least one correlation coefficient to be negative. For example, in particle-size analysis, if sand increases, at least one of silt or clay **must** decrease, as can be observed in a ternary diagram.

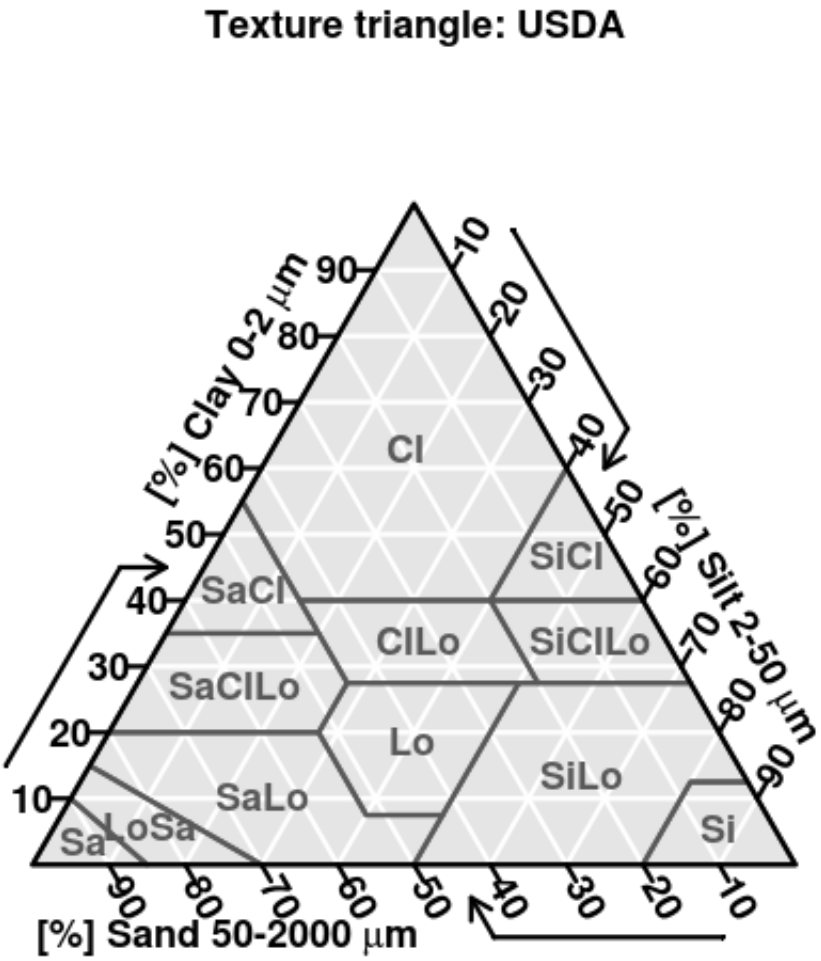
```
library(soiltexture)
```

```
## Loading required package: sp
```

```
## Loading required package: MASS
```

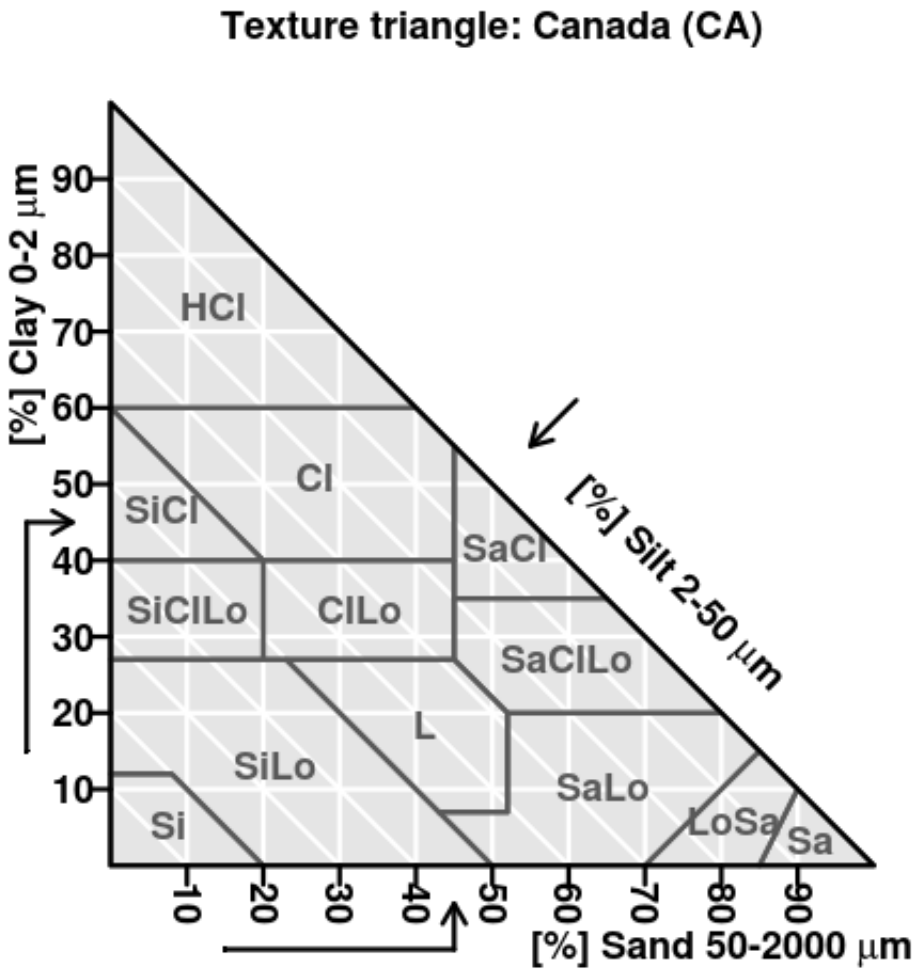
```
## 'soiltexture' loaded.
```

```
TT.plot(class.sys = "USDA.TT")
```



That is why the Canadian soil texture classification drops the silt part, which can be computed by subtracting sand and clay from 100%.

```
TT.plot(class.sys = "CA.EN.TT")
```



So, in addition to scale dependency, CoDa contain redundant information. In a simplex of *D* components, one component can be computed by subtracting the sum of the others from the constant sum. Accordingly, the simplex contains *D*-1 degrees of freedom.

Another source of bias is the distribution of CoDa. As the normal distribution scans the real space, the normality assumption is not valid. Normality is also not valid for ordinary log transformations, because the upper part is still unlimited.

Rock (1988) reported some problems that arise when using conventional statistics across CoDa, where he pointed out biases, little or no significance, illusory results, spurious effects and flawed tests. These issues are due to the properties of CoDa named previously:

1. are scale-dependant
2. contain redundant information
3. are not normally distributed

CoDa transformation techniques

Fortunately, since John Aitchison published a book dedicated to CoDa in 1986, these issues can be easily tackled by using data transformations techniques such as *additive log-ratios (alr)*, *centered log-ratios (clr)* and *isometric log-ratios (ilr)*.

Additive log-ratio

The *alr* is a log-ratio between a component and a reference component c^D.

$$alr_j = ln\frac{c_i}{c_D} \; for \; j = 1, 2, ..., D-1$$

The reference component can be selected arbitrary or in order to create an interpretable set of ratios. The choice of the selected component does not influence the results of linear statistics. However, it should be noted that *alr* variables are oblique to each other. Accordingly, distance-based analyses on *alrs* (PCoA, clustering, etc.) should be used with care.

Centered log-ratio

The *clr* is a log-ratio between a component and the geometric mean of all components.

$$clr_i = ln\frac{c_i}{g(c)} \; for \; i = 1, 2, ..., D$$

The sum of each *clr* row equals to zero, inducing several singularity issues. Moreover, *D* components generates *D clrs*, so *clrs* are not straight-forwardly interpretable variables (Aitchison and Greenacre, 2002).

Isometric log-ratio

Ilrs can be based on a special device of balances or linearly independent ratios among nutrients called sequential binary partition (SBP). The SBP describes the *D*-1 orthogonal (geometrically independent) balances between parts and groups of parts. The SBP is a (*D*-1) × *D* matrix, in which parts labeled “+1” (group numerator) are balanced with parts labeled “-1” (group denominator). A part labeled “0” is excluded from the balance between parts. The composition is partitioned sequentially into contrasts at every hierarchically ordered row until the (+1) and (-1) groups each contain a single part.

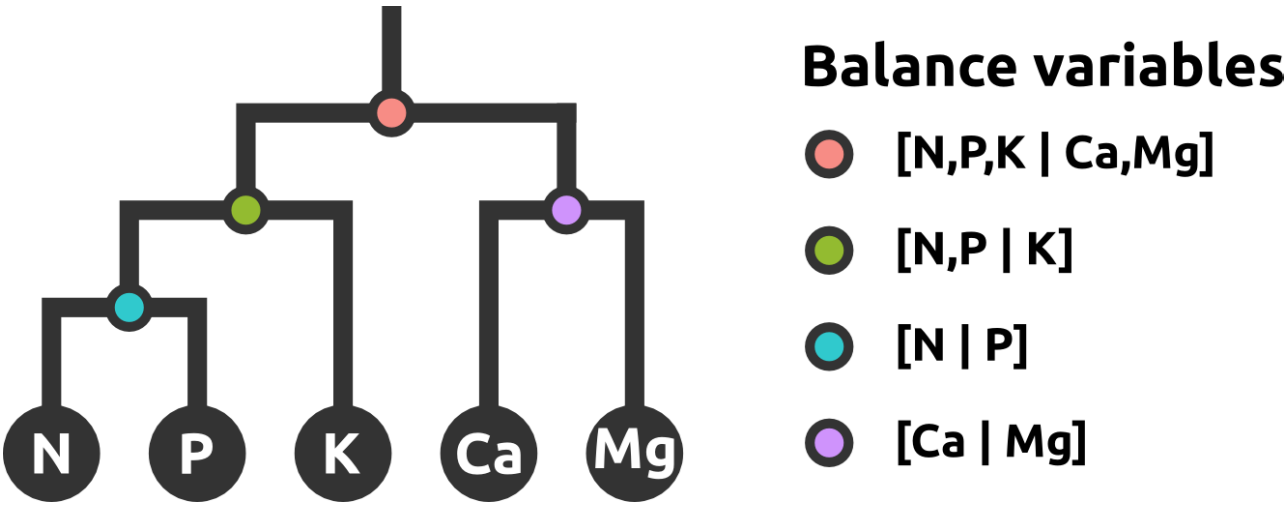
For example,

Sand	**Silt**	**Clay**
1	1	-1
1	-1	0

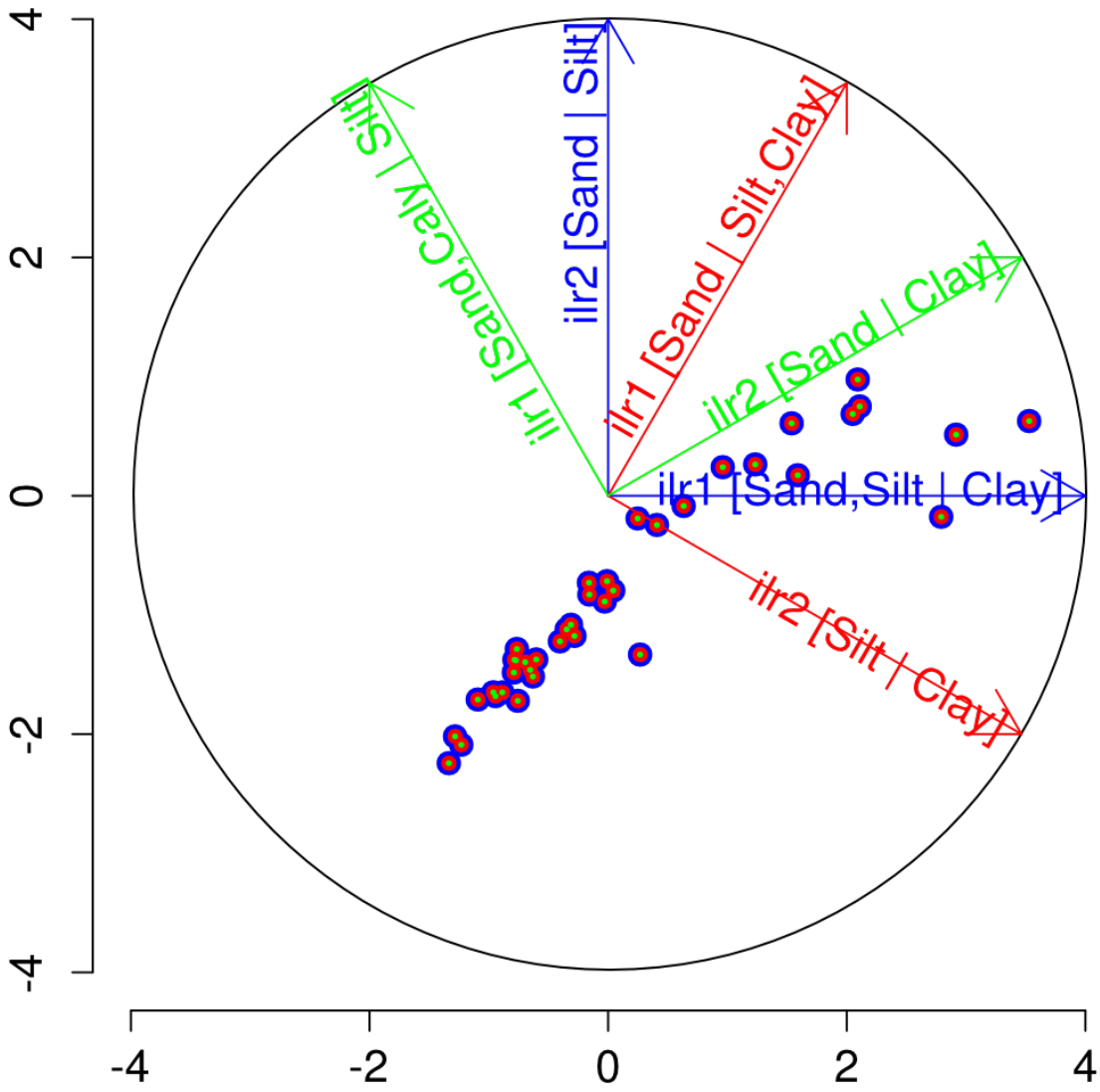
Or

N	P	K	Ca	Mg	Fv
1	1	1	-1	-1	0
1	1	-1	0	0	0
1	-1	0	0	0	0
0	0	0	1	-1	0
1	1	1	1	1	-1

SBPs can be represented by dendrograms. The last one follows the following scheme (the optional filling value, Fv, being removed).



The way the SBP is partitioned depends on how the analyst might create interpretable balances. The structure of the SBP does not influence results in linear statistics. Different SBPs are rotations of *ilr* coordinates from the origin.



Different SBPs will cause some discrepancies in geostatistics when variograms are not modelled consistently. We will cover this later.

To establish contrast orthogonality, it is necessary to imbed subcompositions into larger ones and assign orthogonal coefficients to each log-ratio contrast (Egozcue et al., 2003). The *ilr* is computed as follows (Egozcue and Pawlowsky-Glahn, 2005):

$$ilr_j = \sqrt{\frac{n_j^+ \cdot n_j^-}{n_j^+ + n_j^-}} ln \frac{g(c_j^+)}{g(c_j^-)} \; for \; j = 1, 2, ..., D-1$$

Where *ilr_j* is the j^{th} isometric log-ratio; the square root is the orthogonal coefficient of the j^{th} balance (or log contrast) designed in the SBP; r and s represent the number of parts in the (+1) and (−1) subcompositions on each side of the jth balance, respectively; g(c+) is the geometric mean of

components in group c+; and g(c-) is the geometric mean of components in group c-. The partition between two components or groups of components can be presented as [S- | S+].

The following table gives an example of the computations.

ilr	SBP contrasts N	P	K	Ca	Mg	F_v	Balance designation	n^+	n^-	ilr computation
ilr_1	+1	+1	+1	-1	-1	0	[Mg,Ca | K,P,N]	3	2	$\sqrt{\frac{3x2}{3+2}} ln \frac{g(c_N c_P c_K)}{g(c_{Ca} c_{Mg})}$
ilr_2	+1	+1	-1	0	0	0	[K | P,N]	2	1	$\sqrt{\frac{1 \times 2}{1+2}} ln \frac{g(c_N c_P)}{g(c_K)}$
ilr_3	+1	-1	0	0	0	0	[P | N]	1	1	$\sqrt{\frac{1 \times 1}{1+1}} ln \frac{g(c_N)}{g(c_P)}$
ilr_4	0	0	0	+1	-1	0	[Mg | Ca]	1	1	$\sqrt{\frac{1 \times 1}{1+1}} ln \frac{g(c_{Ca})}{g(c_{Mg})}$
Optional	1	1	1	1	1	-1	[F_v | Mg,Ca,K,P,N]	5	1	$\sqrt{\frac{5x1}{5+1}} ln \frac{g\left(c_N c_P c_K c_{Ca} c_{Mg}\right)}{g(c_{F_v})}$

All in all, the *ilr* transformation:

- The *ilr* values are orthogonal to each other, hence linearly independent, and the ilr matrix has rank D-1.
- The log ratio between sub-compositions A and B, i.e. log(A/B), scans the unconstrained real space while raw concentrations are constrained to the compositional space of the measurement unit; an unconstrained space is required (1) to conduct statistical analyses and determine data distribution and (2) to avoid confidence intervals reaching values below zero or above 100%, a physical impossibility;
- The *ilr* values are scale invariant; the difference between dry and fresh weight bases is the addition of a balance between water and components of the dry weight when scale is changed from dry to fresh weight basis; all other balances remain unchanged.
- The SBP device can be use to create *ilr* balances, structuring them as theory-driven variables of interest.

Which tranformation to use?

In order to avoid numerical biases, data must be transformed. Multivariate covariance functions and variograms of each raw compositions are singular, as spurious as the correlation matrix and do not insure that the interpolated components are positive and sum to 100% (Tolosana-Delgado et al., 2011). By ratioing logs, variables scans the real space. By removing a variable, *alrs* and *ilrs* contain no extra information (while *clrs* does, hence generating singularities). By hierarchically ratioing subcompositions, *ilrs* are scale independent. Moreover, the ilr transformation has the advantage

over the *alr* to be geometrically suited to conduct multivariate analysis (Filzmoser et al., 2009a). Indeed, Euclidean distance based analyses should be avoided with *alrs*, whose variables are not linearly independent (orthogonal). Our research group has been using *ilrs* for many multivariate analyses techniques (e.g. Parent, 2012). Nonetheless, choosing the *right* transformation is a matter of debate between scientists (Bacon-Shone, 2011).

CoDa operators

Operators + and * are meaningless in CoDa, because we can not just increase or decrease one component without reducing at least one of the others. Aitchison (1986) developed two basic CoDa operators: perturbation ⊕ and powering ⊗ (Pawlowsky-Glahn and Egozcue, 2006).

Perturbation

Composition x = [80, 15, 5] perturbs composition y = [50, 30, 20] by:

$x \oplus y = C[80 \bullet 50, 15 \bullet 30, 5 \bullet 20]$

Powering

Composition x = [80, 15, 5] powers composition y = [50, 30, 20] by:

$x \otimes y = C[80^{50}, 15^{30}, 5^{20}]$

Or, scalar t powers composition y = [50, 30, 20] by: $t \otimes y = C[50^{t}, 30^{t}, 20^{t}]$

CoDa straight line

A line in the simplex is defined with the linear model

$x(t) = b \oplus (t \otimes x)$

Where b is the compositional origin and v is the compositional slope and t is the dependant variable.

R “compositions” package

In this first example, I will review de basic commands of the “compositions” R package.

Here are some common functions

- acomp(x) : closing the simplex x.
- plot(acomp(x)) : plot the ternary diagram of the simplex x.
- alr(acomp(x)) : *Alr* transformation with last column as reference. To inverse, use alrInv(…).
- clr(acomp(x)) : *Clr* transformation. To inverse, use clrInv(…).
- ilr(acomp(x), V= gsi.buildilrBase(t(sbp))) : *Ilr* transformation, where the SBP is the matrix sbp. To inverse, use ilrInv(…, V= gsi.buildilrBase(t(sbp))).

Example

We aim to explore hydrogeochemical data by plotting a ternary diagram and compute *ilrs*. We suppose that the “compositions” package is already installed in R.

```
library("compositions")  # load compositions module
```

```
## Loading required package: rgl
```

```
## Loading required package: tensorA
```

```
## Attaching package: 'tensorA'
```

```
## The following object(s) are masked from 'package:base':
##
## norm
```

```
## Loading required package: robustbase
```

```
## Loading required package: energy
```

```
## Loading required package: boot
```

```
## Attaching package: 'boot'
```

```
## The following object(s) are masked from 'package:robustbase':
##
## salinity
```

```
## Warning: replacing previous import 'plot' when loading
'graphics'
```

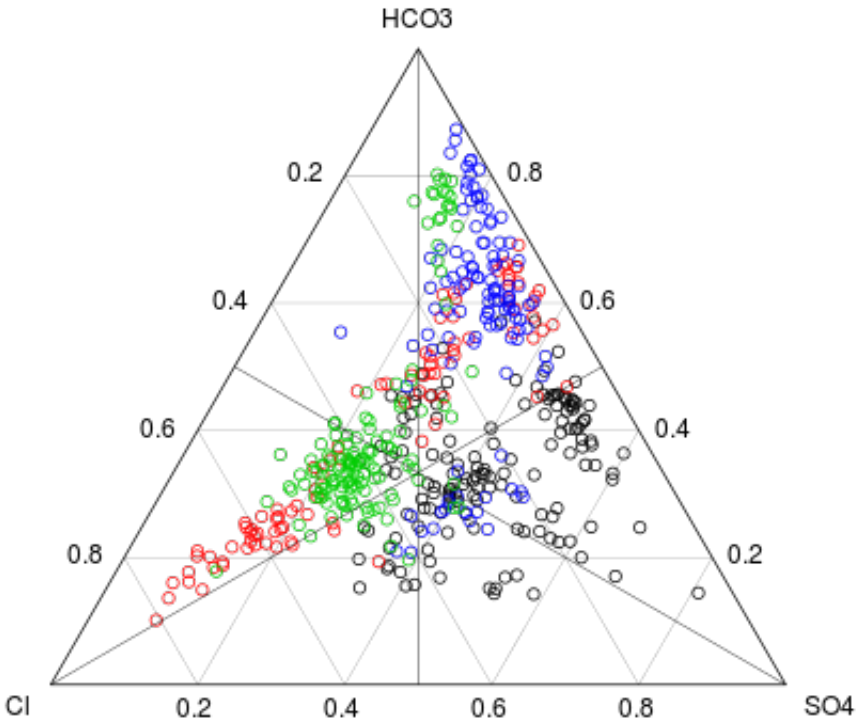
```
## Welcome to compositions, a package for compositional data
analysis. Find
## an intro with "? compositions"
```

```
## Attaching package: 'compositions'
```

```
## The following object(s) are masked from 'package:stats':
##
## cor, cov, dist, var
```

```
## The following object(s) are masked from 'package:base':
##
## %*%, scale, scale.default
```

```
data(Hydrochem)  # load R dataset
hydData <- Hydrochem[, c("Cl", "SO4", "HCO3")]  # select only 3
columns
Composition <- acomp(hydData)  # close the simplex and decalre
that hydData is of compositional data class
plot(Composition, col = Hydrochem$River)  # ternary diagram,
colors according to the river
isoPortionLines(by = 0.2, lwd = 0.2)  # grid
isoProportionLines(by = 0.5, lwd = 0.5, labs = FALSE)  #
isoproportion lines
```



We have declared that the matrix Composition is a compositional dataset.

```
class(Composition)
```

```
## [1] "acomp"
```

To compute statistics on CoDa, we can call generic R functions. R will detect the class and apply the corresponding function (e.g. if you call mean for an acomp class, R will apply mean.acomp). Be careful when computing operations (+ - * /) on acomp matrices. CoDa mathematics will be applied: these operators are not computed as they would with numeric data.

```
mean(Composition)  # mean
```

```
##     Cl    SO4   HCO3
## 0.2262 0.3267 0.4471
## attr(,"class")
## [1] acomp
```

To avoid losing the acomp class when an acomp matrix is subsetted, the acomp function should be called again.

```
mean(acomp(Composition[Hydrochem$River == "Cardener", ]))
```

```
##     Cl    SO4   HCO3
## 0.3027 0.2650 0.4324
## attr(,"class")
## [1] acomp
```

One might transform the CoDa in *alr*, compute the desired statistics, then backtransform statistics from *alrs* to compositions.

```
transf <- alr(Composition)  # alr transformation
moyenne <- mean(transf)  # mean across the alrs
alrInv(moyenne)  # backtransformation from mean of alrs to mean
of components
```

```
##     Cl    SO4
## 0.2262 0.3267 0.4471
## attr(,"class")
## [1] acomp
```

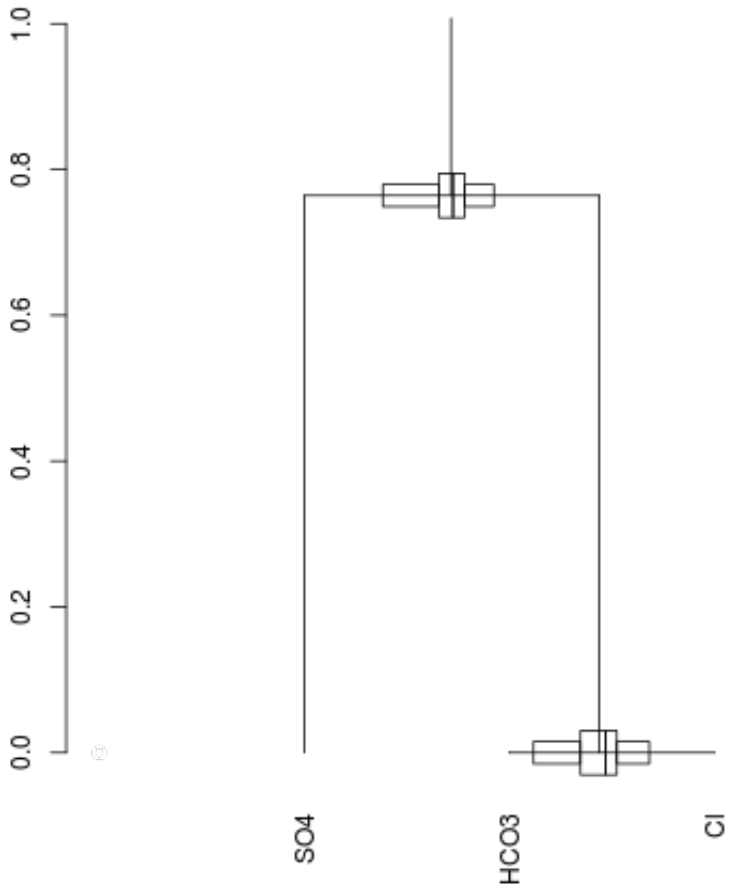
As said, *ilrs* needs an SBP. If no SBP is specified, R will generate one automatically. However, even though the SBP has no influence on the results of linear statistics, *ilr* variables themselves can be of great interest. Accordingly, an theory-driven SBP could help the analyst interpret *ilrs* as balances of components.

The SBP can be written in csv, then loaded to R.

```
sbp1 <- read.table("http://ubuntuone.com/7dT9tH7mtaZTrWYNRm4xh7",
header = TRUE,
    sep = ";", dec = ".")
```

You have to transform your SBP into an orthogonal basis (V), which is used as argument in the ilr function. The last line of the following block generates a compositional dendrogram.

```
V1 <- gsi.buildilrBase(t(sbp1))
balances1 <- ilr(Composition, V = V1)
CoDaDendrogram(Composition, V = V1)
```

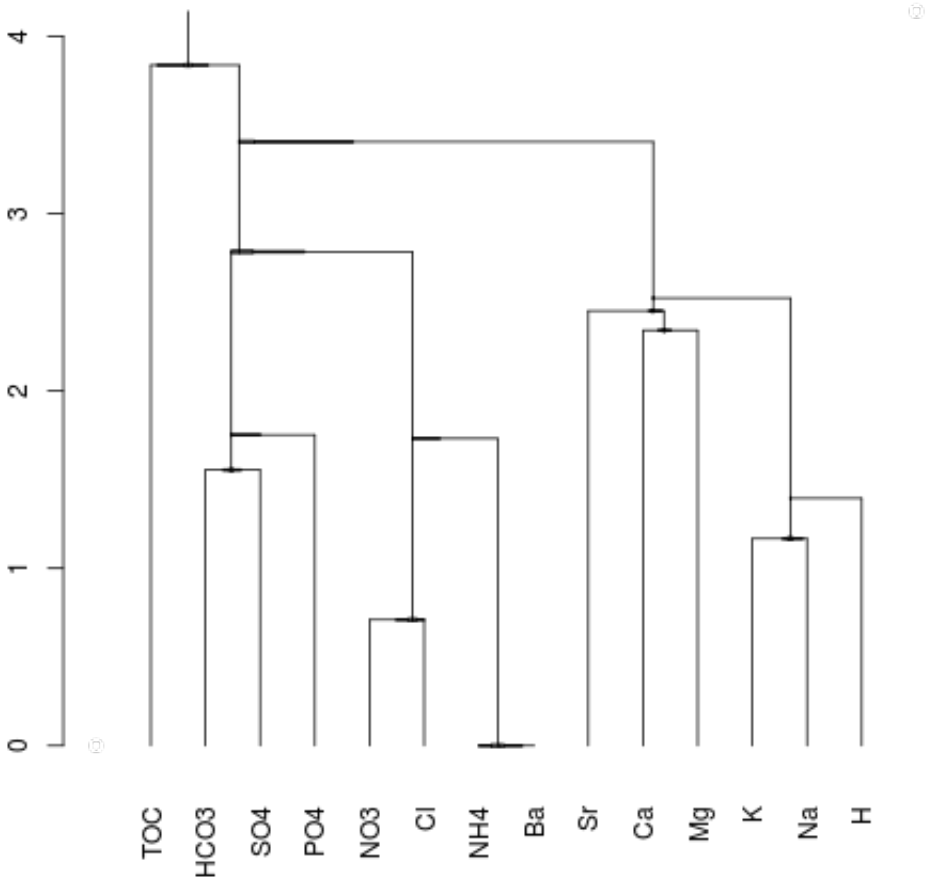


The compositional dendrogram is read as follows.

- Each intersection represents a balance. In this case, we had 2 balances (with 3 components)
- The junction of the vertical branch to the horizontal bar represents the mean of *ilr* values. If the junction lies left from the center, the *ilr* mean is negative. Accordingly, at the mean, components of the left are more important in the simplex than components of the right. Inversely if the juction is on the right. Rectangles at junction are the boxplots of *ilr* values.
- The length of the vertical link of each junction is the proportion of total variance of the corresponding balance.

Here is a more complex dendrogran.

```
sbp2 <- read.table("http://ubuntuone.com/7KFN8qJgfsOadpQuwfIZZE",
header = TRUE,
    sep = ";", dec = ".")
V2 <- gsi.buildilrBase(t(sbp2))
CoDaDendrogram(acomp(Hydrochem[, 6:19]), V = V2)
```



CoDa linear regressions

Linear regression is used to analyse the relation between a dependant variable and one or more explanatory varibables. Linear regression in the simplex can be used to model a vector of CoDa as a function of an external variable. For more information, the reader might refer to Tolosana-Delgado and van den Boogart (2011).

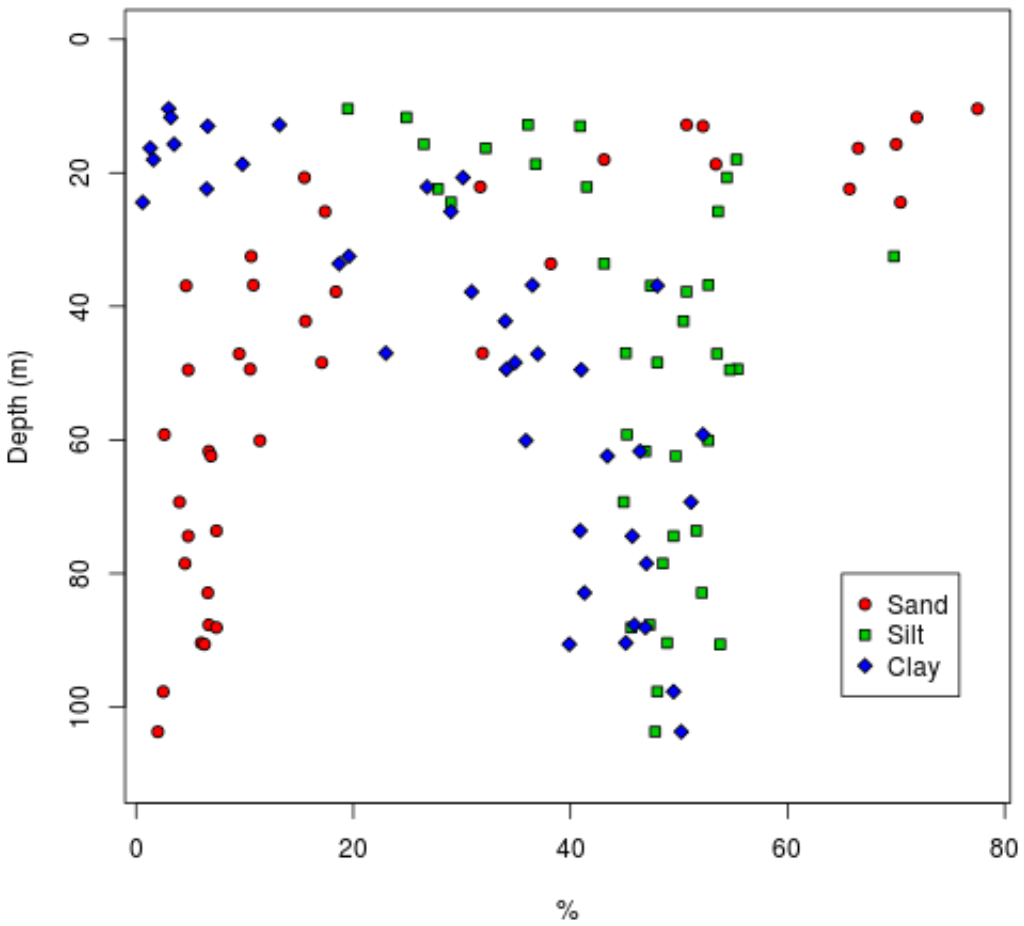
We will relate particle-size (sand, silt, clay) of sediments of an Arctic lake as a function of depth. Data are part of the “compositions” R package.

```
library(compositions)
data(ArcticLake)  # load data
ArcticLake[1:5, ]  # show lines 1 to 5
```

```
##   sand silt clay depth
## 1 77.5 19.5  3.0  10.4
## 2 71.9 24.9  3.2  11.7
## 3 50.7 36.1 13.2  12.8
## 4 52.2 40.9  6.6  13.0
## 5 70.0 26.5  3.5  15.7
```

We could explore the data by creating a plot of percentage versus depth:

```
plot(x = ArcticLake[, 1], y = ArcticLake[, 4], xlab = "%", ylab =
"Depth (m)",
    pch = 21, bg = 2, ylim = c(110, 0))
points(x = ArcticLake[, 2], y = ArcticLake[, 4], pch = 22, bg =
3)
points(x = ArcticLake[, 3], y = ArcticLake[, 4], pch = 23, bg =
4)
legend(x = 65, y = 80, legend = c("Sand", "Silt", "Clay"), pch =
c(21, 22, 23),
    pt.bg = c(2, 3, 4))
```

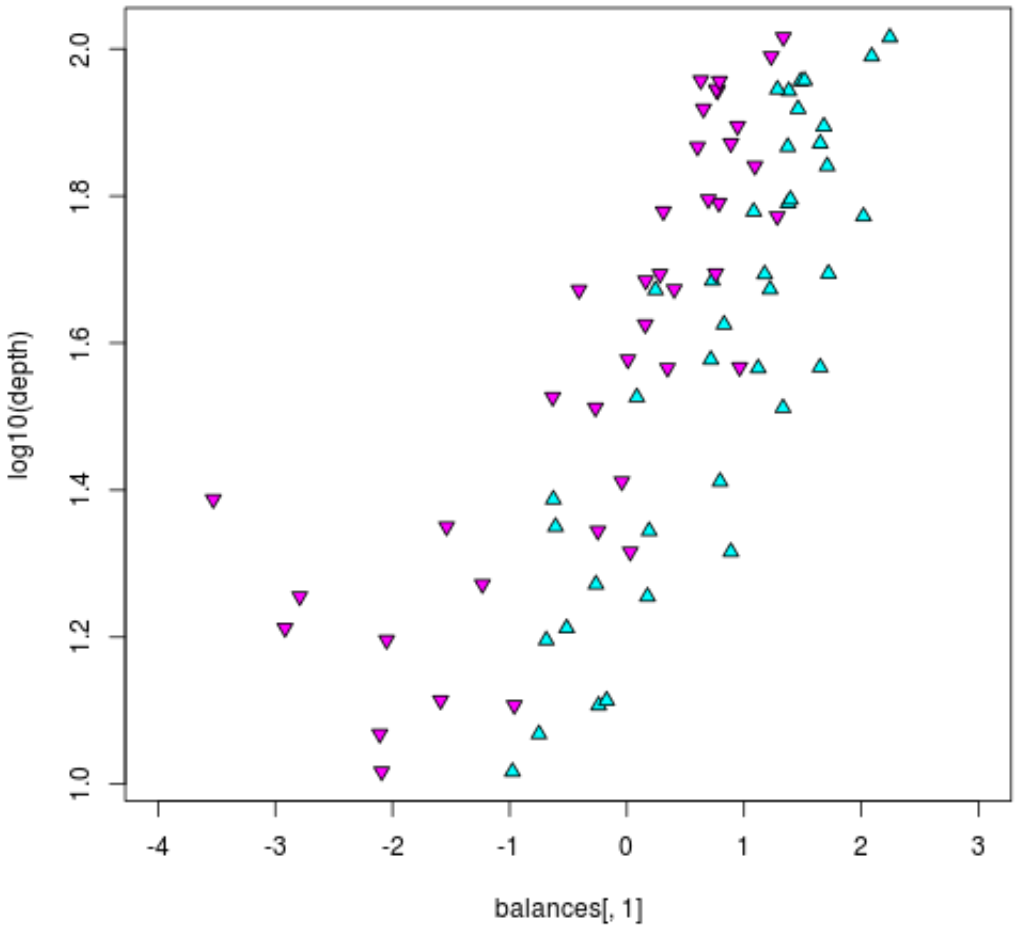


The relation is obviously not linear, at least not in the compositional representation. At first sight, one could apply on each of these particle-size class a Weibull model, a growth to saturation model or a third degree polynomial model. However, these regression would be applied on non-normal data, would allow total contents to take values more or less than 100%, and would not account for interrelations between particle-size classes. A solution is to perform the regression across the associated *ilr* coordinates of the composition:

```
comp <- acomp(ArcticLake[, 1:3])  # close the simplex across
column 1 to 3
balances <- ilr(comp)  # create balances with defaut SBP
depth <- ArcticLake[, 4]  # depth vector: column 4
```

Our balances being created, we can explore them. In our case, the log_10 of depth is used as explanatory variable:

```
plot(y = log10(depth), x = balances[, 1], xlim = c(-4, 3), pch =
24, bg = 5)
points(y = log10(depth), x = balances[, 2], pch = 25, bg = 6)
```



We can use a linear model for balances against log10 of depth using the lm function:

```
ilr.lm <- lm(balances ~ log10(depth))
coefficients(ilr.lm)
```

```
##                [,1]   [,2]
## (Intercept)  -3.459 -5.921
## log10(depth)  2.681  3.609
```

```
summary(ilr.lm)
```

```
## Response Y1 :
##
## Call:
## lm(formula = Y1 ~ log10(depth))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8874 -0.2865  0.0355  0.2604  0.9074
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -3.459      0.394   -8.77  1.4e-10 ***
## log10(depth)    2.681      0.243   11.02  3.0e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.444 on 37 degrees of freedom
## Multiple R-squared: 0.766,   Adjusted R-squared: 0.76
## F-statistic:  121 on 1 and 37 DF,  p-value: 3.04e-13
##
##
## Response Y2 :
##
## Call:
## lm(formula = Y2 ~ log10(depth))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6149 -0.3257  0.0537  0.2995  1.2279
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -5.921      0.643   -9.21  4.1e-11 ***
## log10(depth)    3.609      0.396    9.10  5.6e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.723 on 37 degrees of freedom
## Multiple R-squared: 0.691,   Adjusted R-squared: 0.683
## F-statistic: 82.9 on 1 and 37 DF,  p-value: 5.59e-11
```

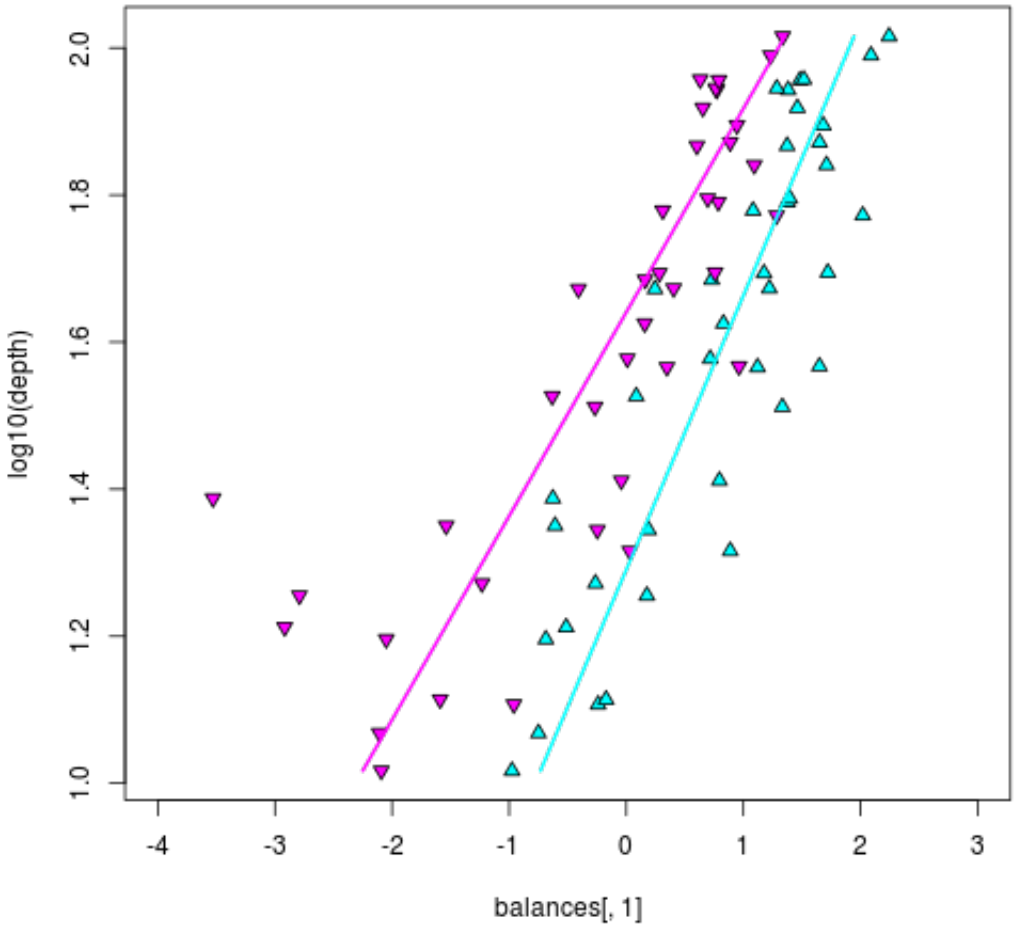
Using the summary command, you obain the R^2, as weel as the result of the hypothesis (Pr) on the parameters for:

- H0 : the parameter is null
- H1 : the parameter is non null

At a 0.05 level, we reject the null hypotheses for the origin and the slope for regressions on both balances. For more details on diagnoses, the reader might refer to chapter 6 of the book *Introductory statistics with R*, to Tolosana-Delgado and van den Boogart (2011) or to any other source about the lm function.

And the prediction:

```
plot(x = balances[, 1], y = log10(depth), xlim = c(-4, 3), pch =
24, bg = 5)
points(x = balances[, 2], y = log10(depth), pch = 25, bg = 6)
lines(x = predict(ilr.lm)[, 1], y = log10(depth), lwd = 2, col =
5)
lines(x = predict(ilr.lm)[, 2], y = log10(depth), lwd = 2, col =
6)
```



The coefficients are shown as *ilrs*. They can be ported to the simplex by inverting the ilr transformation.

```
coefficients <- ilrInv(coef(ilr.lm))
intercept <- acomp(coefficients[1, ])  #intercept (perturbation)
pente <- acomp(coefficients[2, ])  #log(depth) (powering)
```

As said previously, when a vector belongs to the acomp class, + and * operators are used for perturbation ⊕ and powering ⊗.

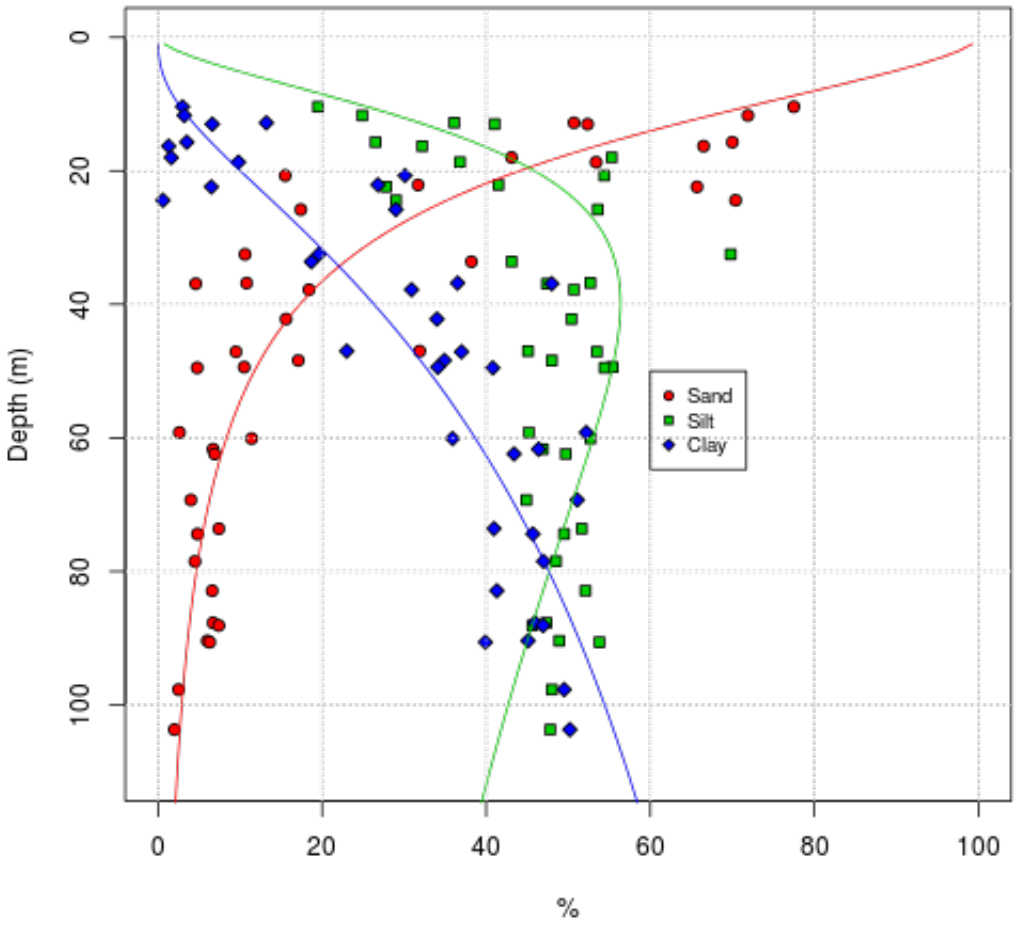
```
logDepthPred <- log10(1:500)  # generate a depth vector
compPred <- intercept + pente * logDepthPred  # compute pert. and
pow. for the linear relation
```

Note that to avoid the use of ⊕ and ⊗, the analyst could compute the ilr prediceted lines, then backtransform the predictions to compisitions:

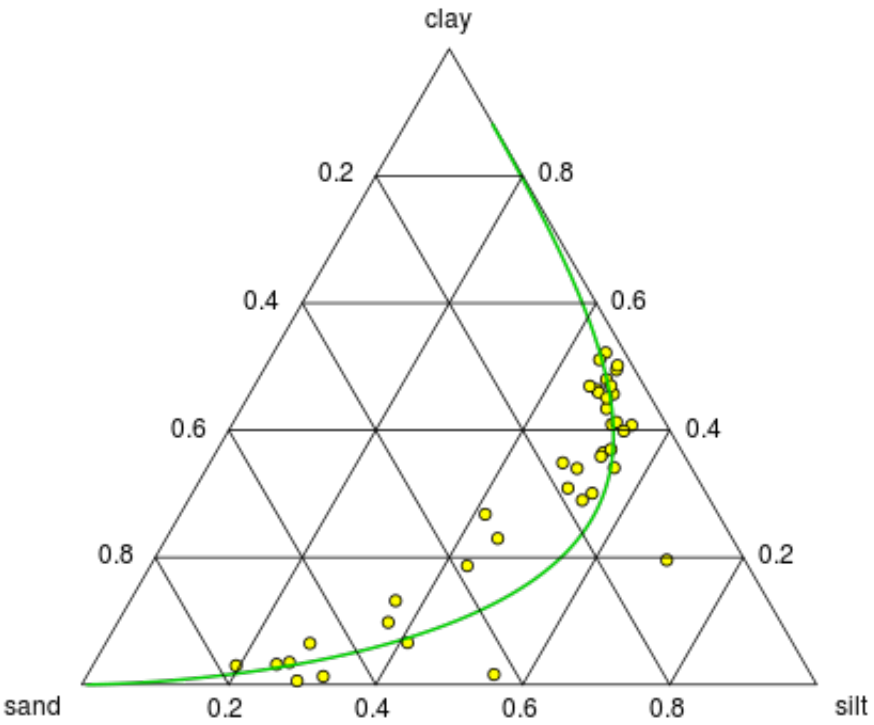
```
ilr1Pred <- coef(ilr.lm)[1, 1] + coef(ilr.lm)[2, 1] *
logDepthPred
ilr2Pred <- coef(ilr.lm)[1, 2] + coef(ilr.lm)[2, 2] *
logDepthPred
ilrPred <- data.frame(ilr1Pred, ilr2Pred)
compPred <- ilrInv(ilrPred)
```

The regression can be presented in percentages versus depth or in the ternary diagram:

```
# % versus depth
plot(y = depth, x = comp[, 1] * 100, ylab = "Depth (m)", xlab =
"%", pch = 21,
    bg = 2, ylim = c(110, 0), xlim = c(0, 100))
points(y = depth, x = comp[, 2] * 100, pch = 22, bg = 3)
points(y = depth, x = comp[, 3] * 100, pch = 23, bg = 4)
lines(y = 10^logDepthPred, x = compPred[, 1] * 100, col = 2)
lines(y = 10^logDepthPred, x = compPred[, 2] * 100, col = 3)
lines(y = 10^logDepthPred, x = compPred[, 3] * 100, col = 4)
grid()
legend(y = 50, x = 60, legend = c("Sand", "Silt", "Clay"), pch =
21:23, pt.bg = c(2,
    3, 4), cex = 0.8)
```




```
# ternary diagram
plot(comp, pch = 21, bg = 7)  # points
lines(compPred, col = 3, lwd = 2)  # linear regression
isoPortionLines()
```



Multivariate analysis

As said, *ilrs* are orthogonal variables suited for distance-based analyses. For sake of example, we could compute a cluster analysis.

```
library(compositions)  # CoDa analysis
library(vegan)  # Ecological analysis
```

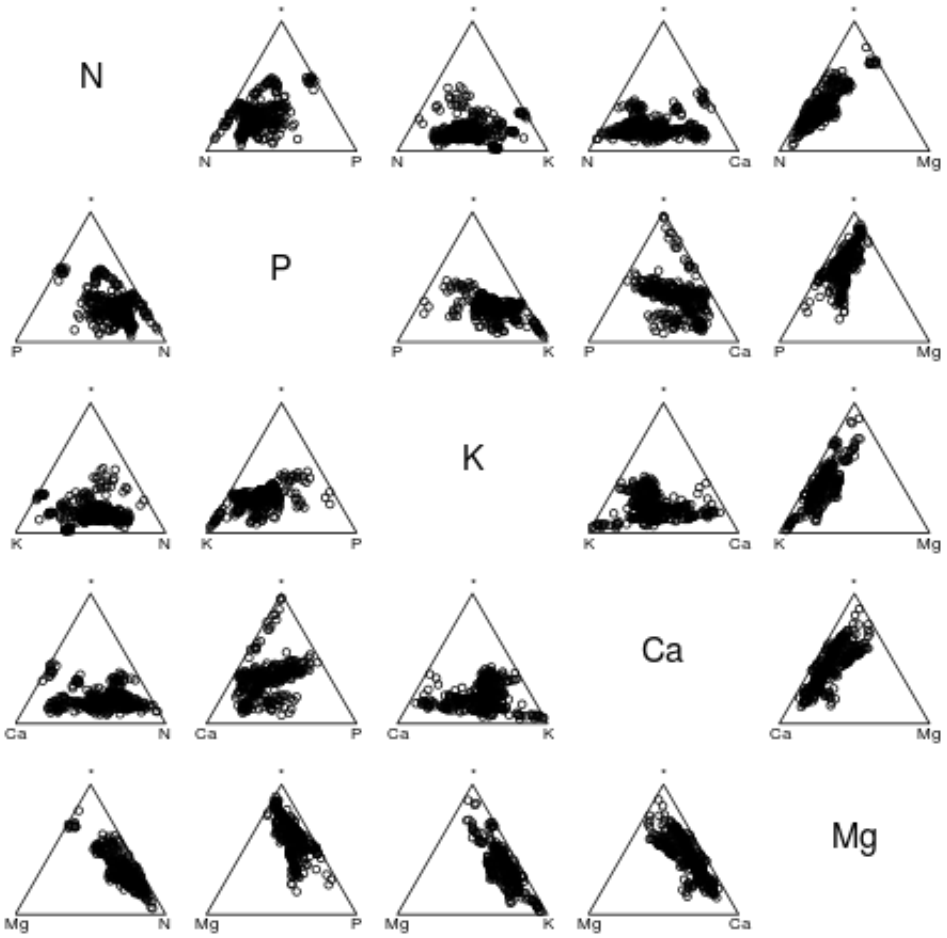
```
## Loading required package: permute
```

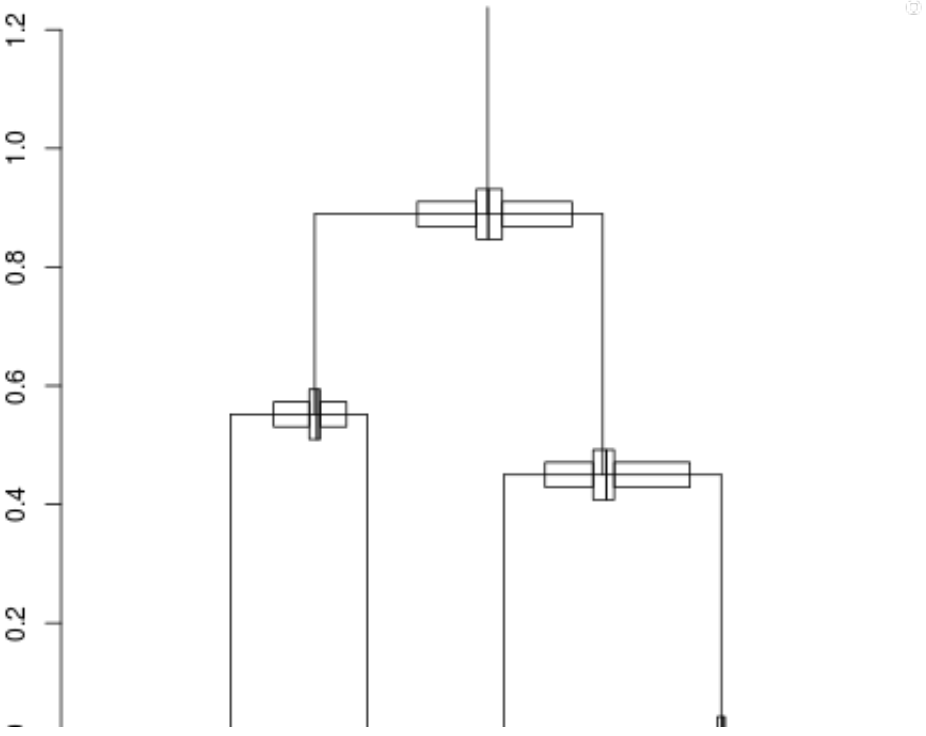
```
## This is vegan 2.0-7
```

```
source("http://ubuntuone.com/6Uh8qUI2W2szH5TBkCHqXg")  # custom
script for balance names

# Load data
fictveg <-
read.table("http://ubuntuone.com/5dylj9DpiN47MlvDYtaxGW", sep =
";",
    dec = ".", header = TRUE)
sbpveg <-
read.table("http://ubuntuone.com/5nOgHZoFGgg9o1RWkhg02u", sep =
";",
    dec = ".", header = TRUE)

# Format compositions
compveg <- acomp(fictveg[, -c(1, 2)])
plot(compveg)
balveg <- ilr(compveg, V = gsi.buildilrBase(t(sbpveg)))  #
compute the ilrs according to the orthonormal basis
CoDaDendrogram(compveg, V = gsi.buildilrBase(t(sbpveg)))  # plot
the dendrogram to visualise balances
```


N
P
K
Ca
Mg
N P
N K
N Ca
N Mg
P N
P K
P Ca
P Mg
K N
K P
K Ca
K Mg
Ca N
Ca P
Ca K
Ca Mg
Mg N
Mg P
Mg K
Mg Ca


1.2
1.0
0.8
0.6
0.4
0.2


```
ilrDef <- ilrDefinition(sbpveg, side = "-+")
colnames(balveg) <- ilrDef

# Clustering (Bocard et al., 2011)
distveg <- vegdist(x = balveg, method = "euc")  # Euclidean
distance

clustveg.single <- hclust(distveg, method = "single")
cor(distveg, cophenetic(clustveg.single))
```

```
## [1] 0.7646
```

```
clustveg.compl <- hclust(distveg, method = "complete")
cor(distveg, cophenetic(clustveg.compl))
```

```
## [1] 0.3832
```

```
clustveg.aver <- hclust(distveg, method = "average")
cor(distveg, cophenetic(clustveg.aver))
```

```
## [1] 0.8576
```

```
clustveg.centr <- hclust(distveg, method = "centroid")
cor(distveg, cophenetic(clustveg.centr))
```

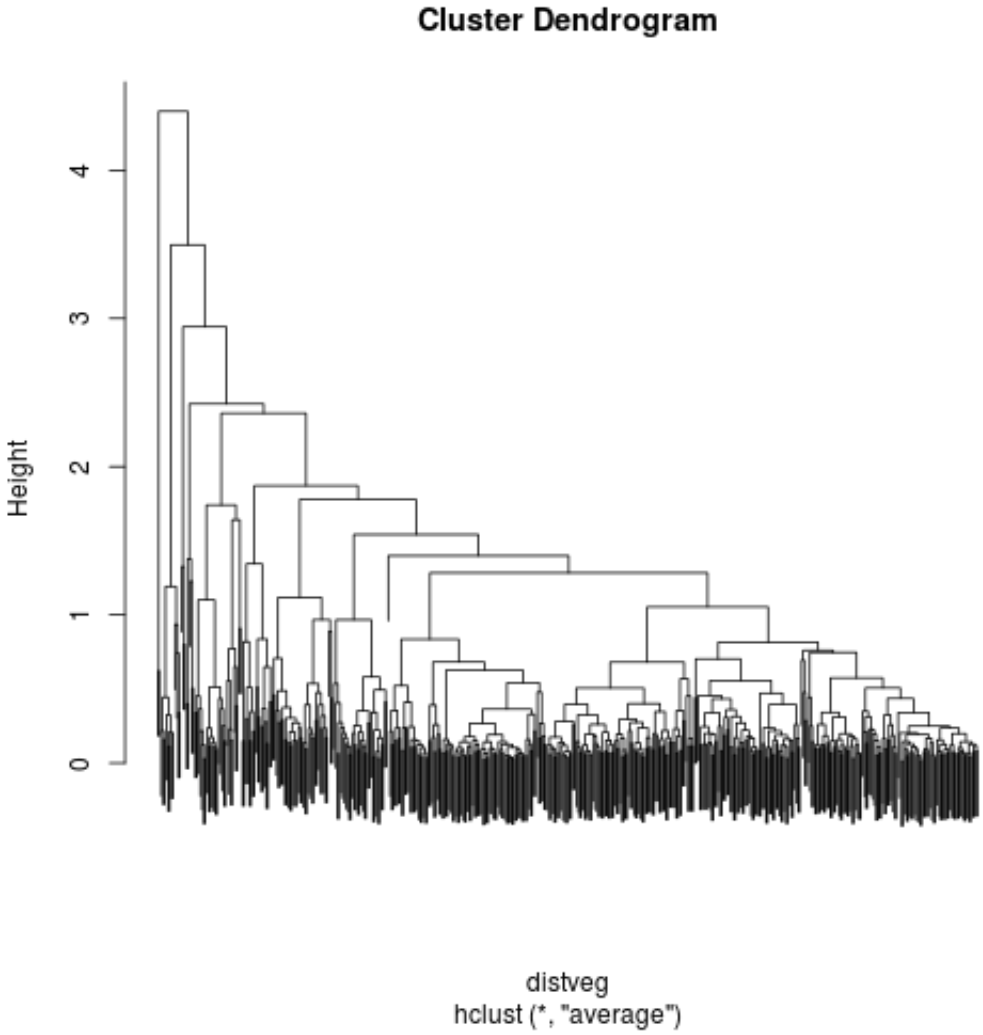
```
## [1] 0.8204
```

```
clustveg.ward <- hclust(distveg, method = "ward")
cor(distveg, cophenetic(clustveg.ward))
```

```
## [1] 0.3674
```

```
# The average method has the highest cophenetic correlation

plot(clustveg.aver, labels = FALSE)
```



For finding an optimal number of groups, the interested reader should refer to Bocard et al. (2011).

Linear discriminant analyses can be used to provide an axis-reduced overview of how predefined groups differentiate each other's. *Alrs* and *ilrs* will return identical scores, but *ilrs* can still be preferred when balances are variables of interest.

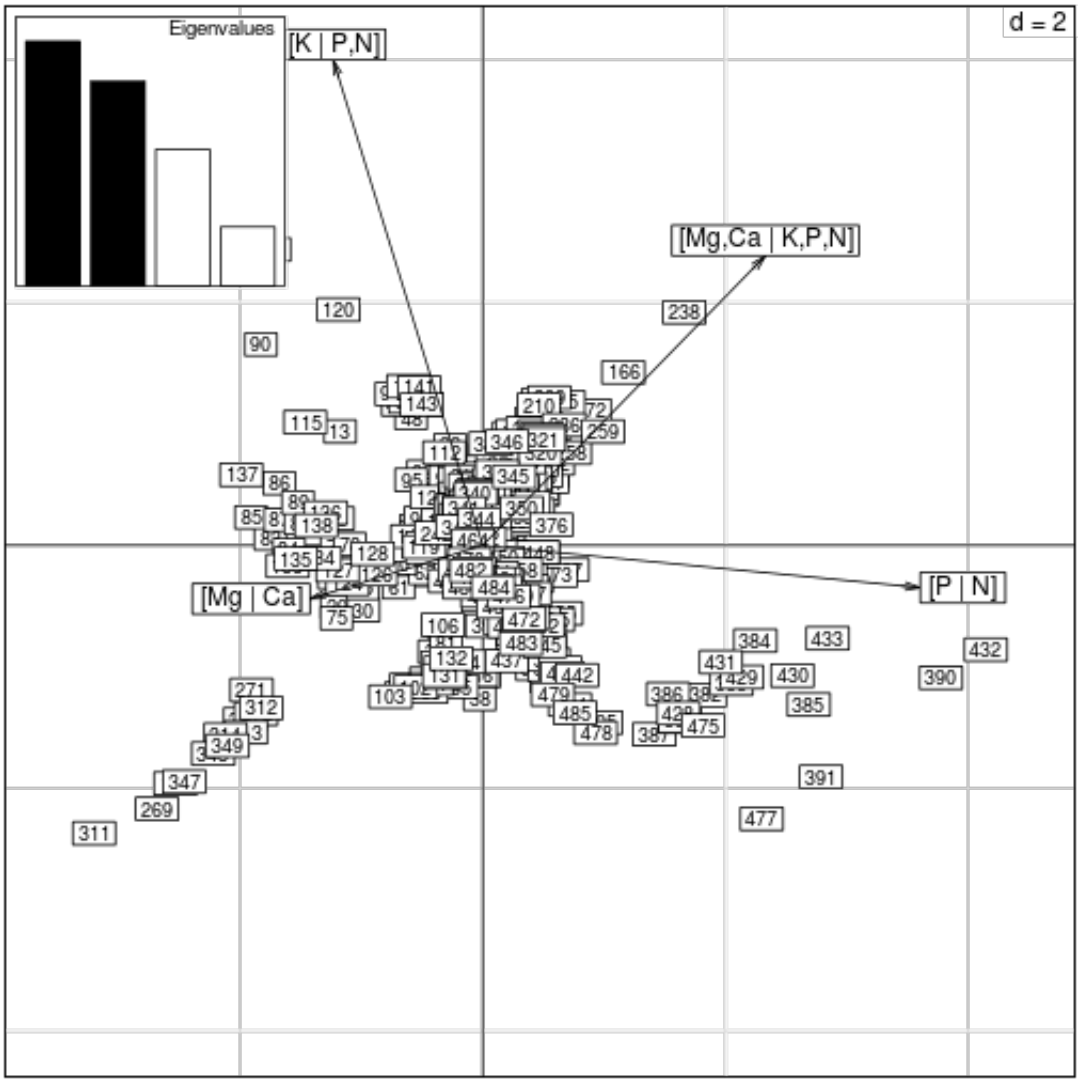
```
library(ade4)  # Ecological analysis
```

```
## Attaching package: 'ade4'
```

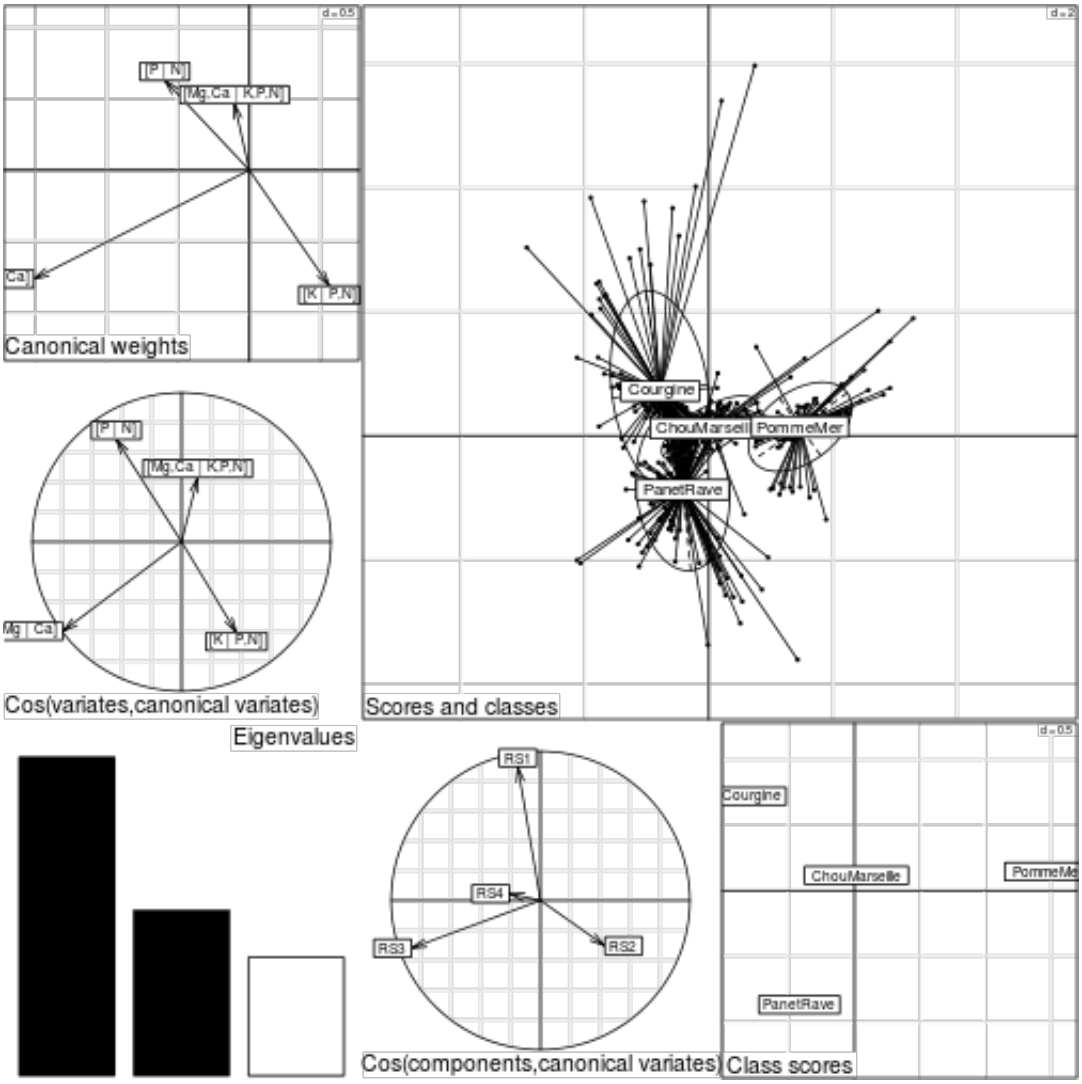
```
## The following object(s) are masked from 'package:vegan':
##
## cca
```

```
## The following object(s) are masked from 'package:base':
##
## within
```

```
balPCA <- dudi.pca(balveg, scannf = FALSE, scale = FALSE)
rownames(balPCA$c1) <- ilrDef  # to avoid messing up names
biplot(balPCA)
```

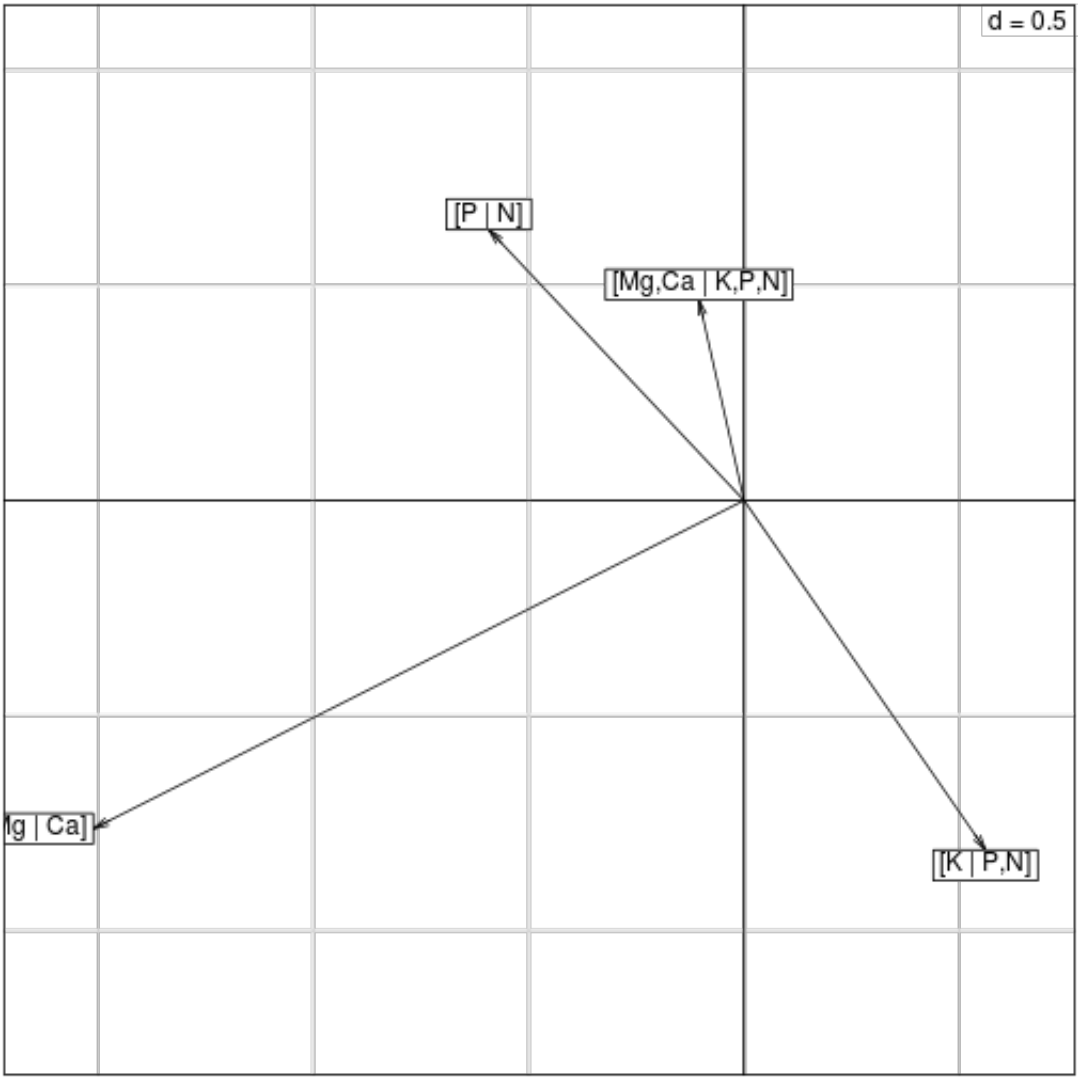



```
balDIS <- discrimin(balPCA, fictveg$Culture, scannf = F)  # AD
sur l'ACP
plot(balDIS)
```




```
p <- 0.95
mult <- sqrt(-2 * log(1 - p))  # approximative ellipse size for a
confidence region of p
s.class(balDIS$li, fictveg$Culture, col = 1:5, cellipse = mult,
clabel = 0.7)
```


```
axes <- balDIS$fa
rownames(axes) <- ilrDef
s.arrow(axes)
```



Geostatistics

Geostatistical analyses often imply compositional data: chemical concentrations (pollution, ore, hydrogeochemistry), class partitions (grain-size, living species, demographics, etc.). *ilrs* are used in the backend of the geostatistical computations of the "*compositions*" package. However, it is known that the structure of *ilrs* as defined in the SBP will impact the variogram fitting, thus the krigging.

In order to free compositions from arbitrary SBPs, Tolosana-Delgado et al. (2011) highlighted the need for *mutual consistency* between variogram models. This can be achieved by the use of variation-variograms defined across the variance of log-ratios of components, as follows:

$$\tau_{ij}(\vec{h}) = Var\left[ln\frac{Z_i(x+\vec{h})}{Z_j(x+\vec{h})} - ln\frac{Z_i(x)}{Z_j(x)}\right]$$

Variation-variograms create *D* X *D* variograms of log ratios.

For a more complete overview of geostatistics with the “compositions” package, the reader should refer to Tolosana-Delgado and van den Boogart (2012).

Example 1

Load libraries.

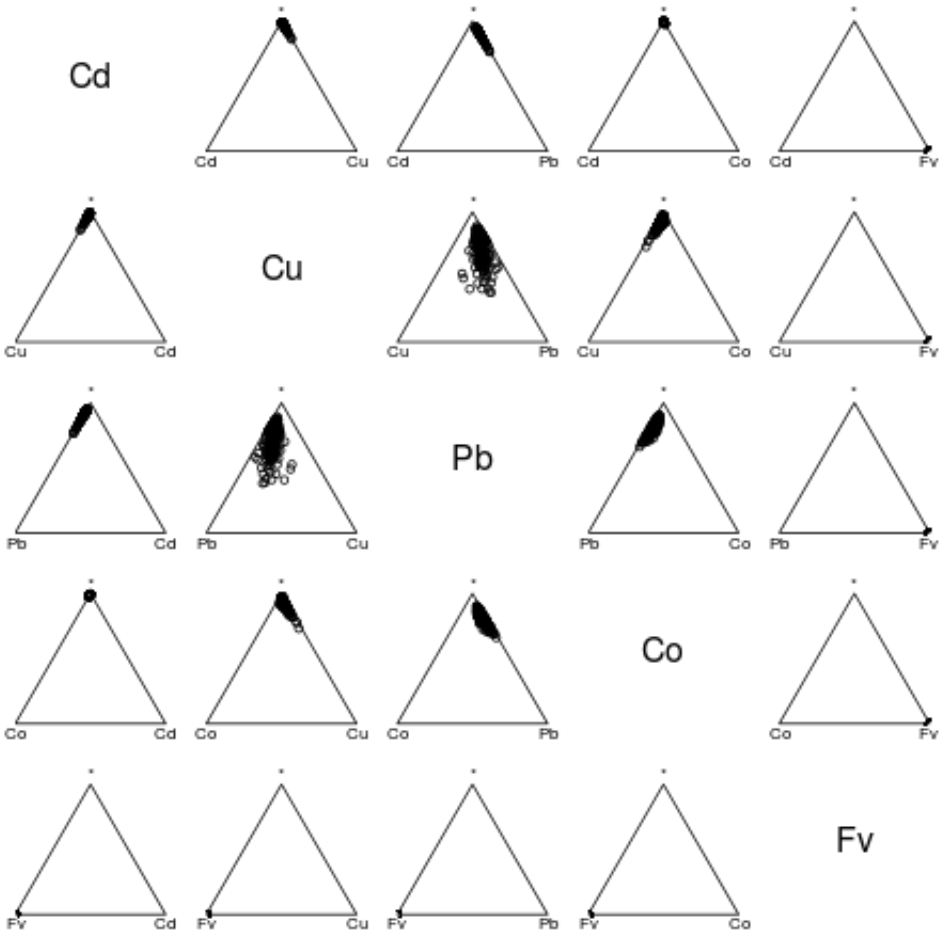
```
library(compositions)  # for CoDa analysis
library(ggplot2)  # Fancy plots
## Multiplot script: to plot multiple ggplots in a grid
## http://www.cookbook-
r.com/Graphs/Multiple_graphs_on_one_page_%28ggplot2%29/
source("http://ubuntuone.com/2kvV9Vrlkr1uZg7DNFMH13")
```

Load data

```
data(juraset)
coords <- with(juraset, cbind(X, Y))  # isolate spatial
coordinates
```

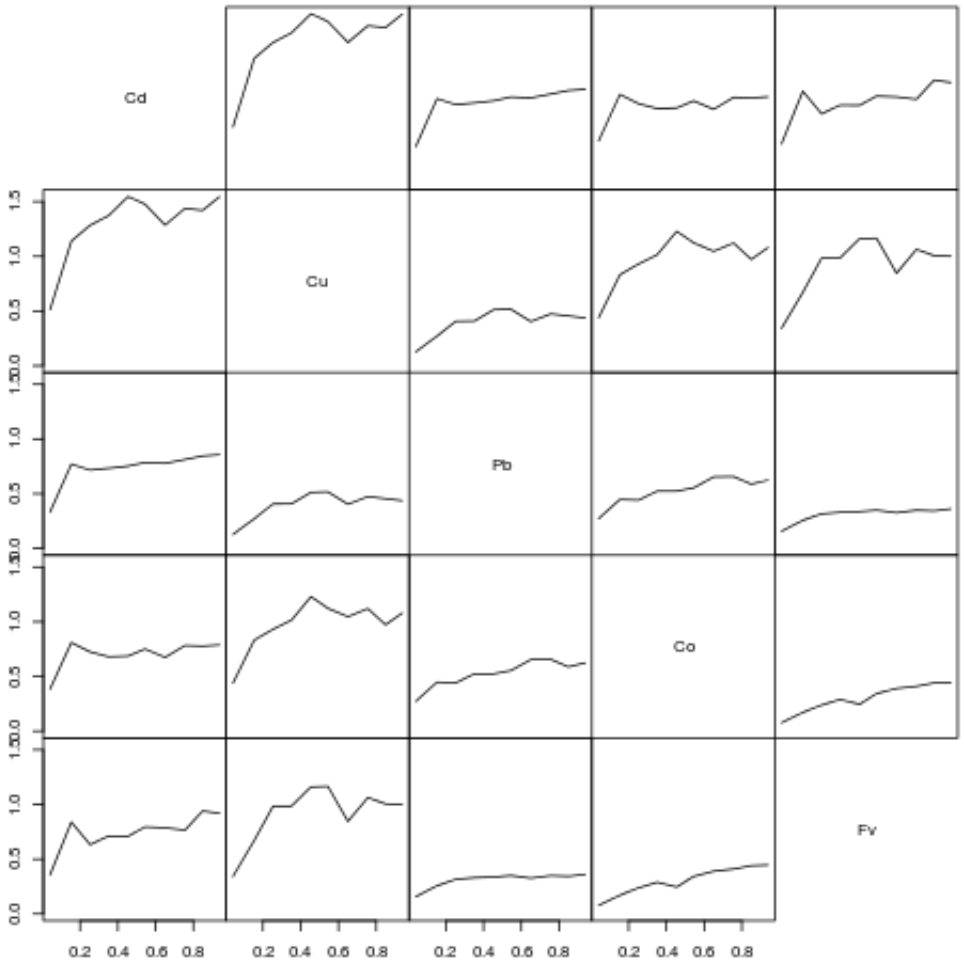
1. Compositional data formatting

```
Fv <- 1e+06 - rowSums(juraset[, 5:8])  # compute a filling value
in order to be able to recover original scale
comp <- acomp(cbind(juraset[, 5:8], Fv))  # close the simplex and
append an acomp class
plot(comp)  # plot ternary diagrams
```



2. Variogram estimation

```
lrv <- logratioVariogram(comp, coords, maxdist = 1, nbins = 10)
plot(lrv)
```



3. Definition of a formula that describes the model that you want to fit

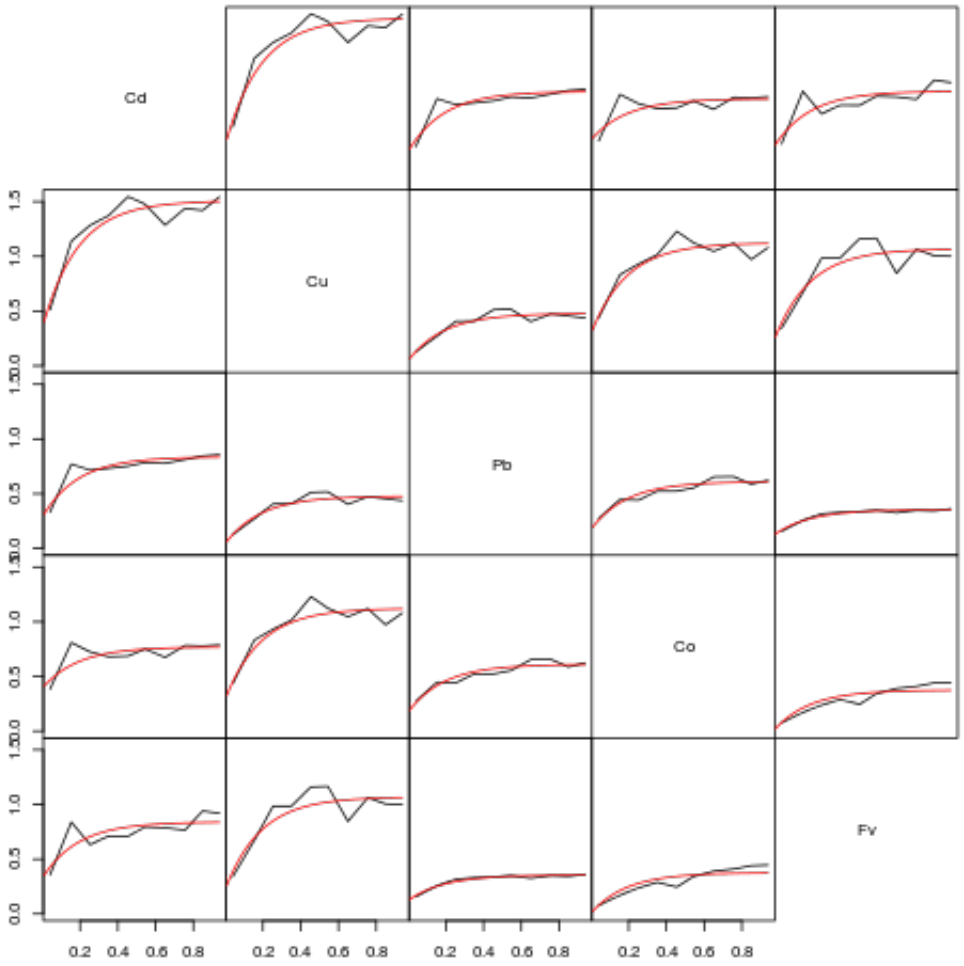
```
vgModel <- CompLinModCoReg(~nugget() + exp(0.5), comp)
```

4. Fitting the variogram parameters

```
fit <- vgmFit2lrv(lrv, vgModel, iterlim = 500)  # Increase the
max number of iterations with iterlim
```

```
## iteration = 0
## Step:
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Parameter:
##  [1] 1.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.5 1.0 0.0 1.0
0.0 0.0 1.0
## [18] 0.0 0.0 0.0 1.0
## Function Value
## [1] 432.5
## Gradient:
##  [1]  98.759  -5.462 137.571   4.514  -9.306 126.261   2.348
-12.763
##  [9]  -9.477 141.376 -94.667  82.620  -4.221 115.323   4.041
-7.758
## [17] 105.798   2.134 -10.863  -7.385 117.968
##
## iteration = 115
## Parameter:
##  [1] -4.148e-01 -1.690e-01 -1.548e-01 -1.704e-02 -2.279e-01
-3.239e-01
##  [7] -1.421e-02 -1.382e-01 -1.981e-01 -4.714e-07  4.322e-01
-7.630e-01
## [13] -4.401e-02 -3.646e-01  1.988e-01  2.951e-02 -4.396e-01
1.033e-01
## [19] -2.493e-01 -3.895e-02 -3.591e-01
## Function Value
## [1] 1.292
## Gradient:
##  [1] -8.287e-07  4.818e-07 -5.518e-07 -5.405e-07  1.616e-06
2.420e-06
##  [7] -4.317e-07 -1.445e-06 -1.902e-06  1.195e-07  6.721e-07
-1.021e-06
## [13]  1.328e-06 -1.165e-06 -2.508e-06 -3.247e-06  5.304e-06
3.362e-07
## [19]  2.499e-06 -3.953e-06  2.930e-06
##
## Itération successsives à l'intérieur du seuil de tolérance.
## L'itération courante est probablement la solution.
```

```
# vgmFit is deprecate, but still computes faster than vgmFit2lrv:
fit <-
# vgmFit(lrv,vgModel)
plot(lrv, lrvg = vgram2lrvgram(fit$vg))
```



5. Applying cokriging to the data

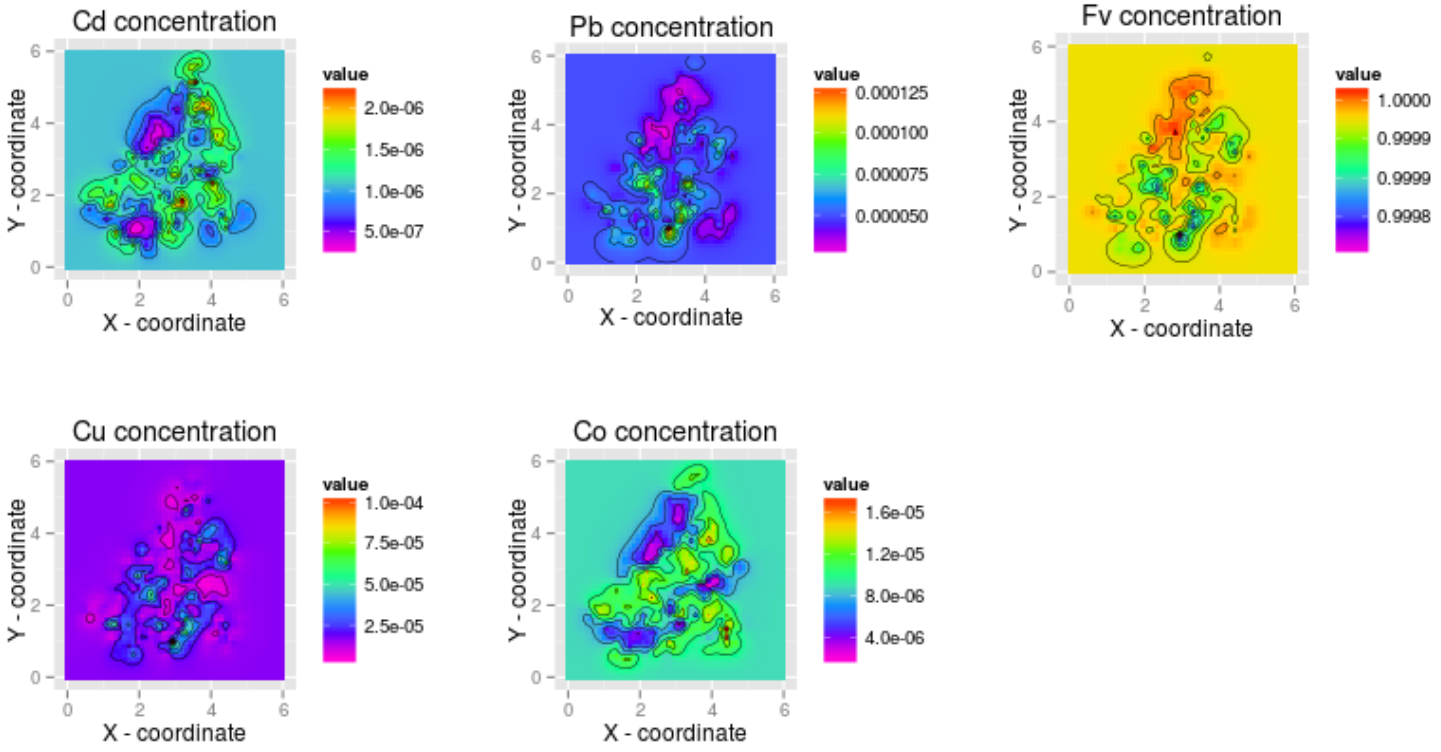
```
coordsNew <- expand.grid(x = seq(from = 0, to = 6, length = 50),
y = seq(from = 0,
    to = 6, length = 50))  ## define grid
krig <- compOKriging(comp, coords, coordsNew, fit$vg, err =
FALSE)  ## Kriging
```

6. Plot the contours

```
p <- list()
for (i in 1:ncol(krig$Z)) {
    ggdata <- data.frame(krig$X, value = krig$Z[, i])
    p[[i]] <- ggplot(ggdata, aes(x = x, y = y, z = value)) +
geom_tile(aes(fill = value)) +
        stat_contour(lwd = 0.2) + # geom_point(data=juraset,
aes(x=X, y=Y, z=0, colour=Rock), pch=4, size=1)
    # +
    ggtitle(paste(colnames(krig$Z)[i], "concentration")) +
xlab("X - coordinate") +
        ylab("Y - coordinate") + scale_fill_gradientn(colours =
rev(rainbow(7))) +
        coord_equal()
}

multiplot(p[[1]], p[[2]], p[[3]], p[[4]], p[[5]], cols = 3)
```

```
## Loading required package: grid
```



Let's have another go with the famous Meuse dataset.

Example 2

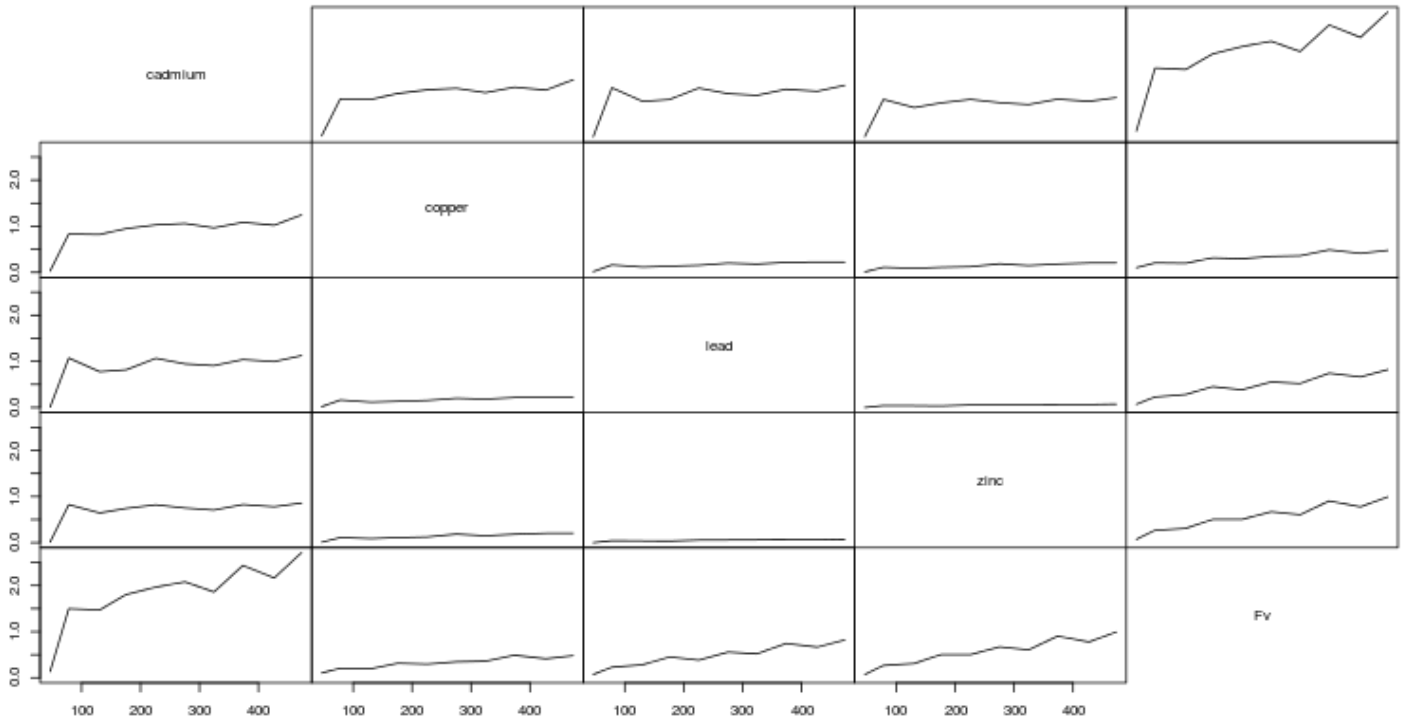
```
library(gstat)
data(meuse)
data(meuse.riv)

library(compositions)  # for CoDa analysis
library(ggplot2)  # Fancy plots
source("http://ubuntuone.com/2kvV9Vrlkr1uZg7DNFMH13")  #
multiplot

coords <- with(meuse, cbind(x, y))

Fv <- 1e+06 - rowSums(meuse[, 3:6])
comp <- acomp(cbind(meuse[, 3:6], Fv))

lrv <- logratioVariogram(comp, coords, maxdist = 500, nbins = 10)
plot(lrv)
```

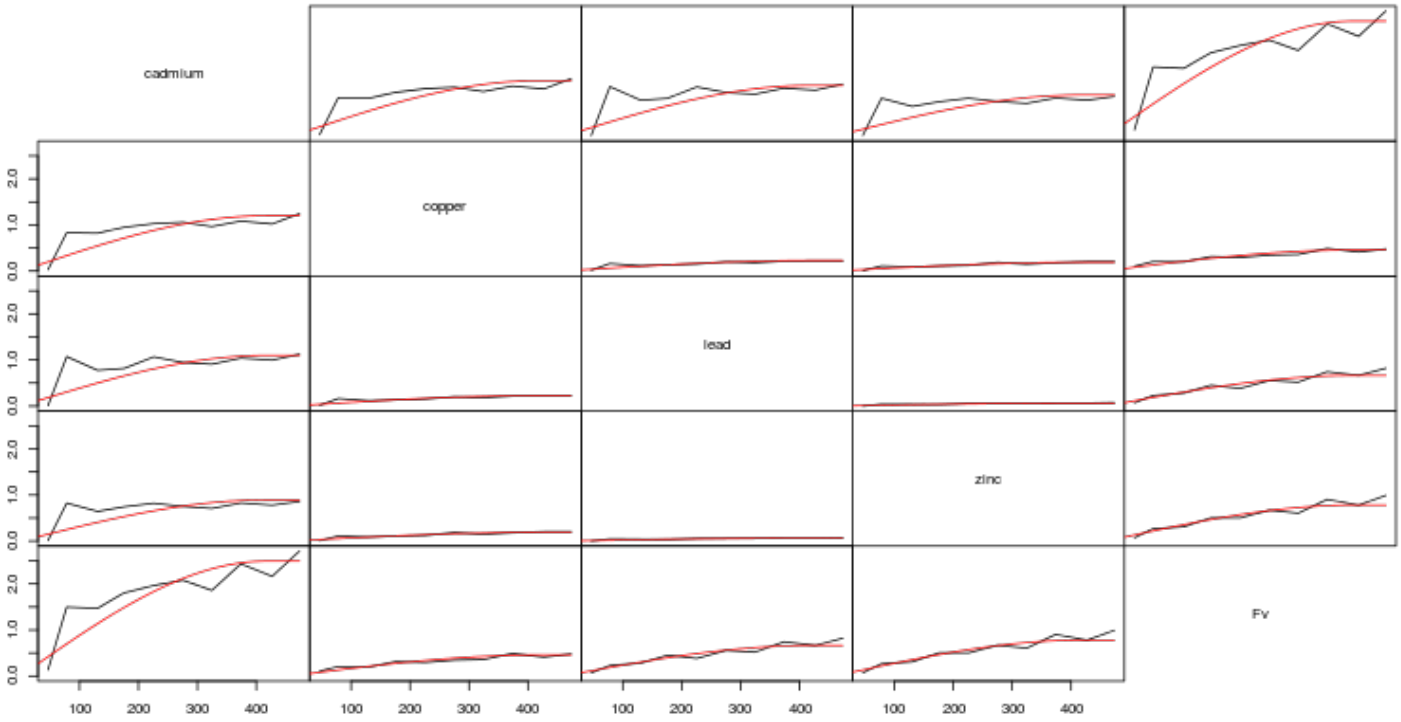



```
vgModel <- CompLinModCoReg(~nugget() + sph(100), comp)

fit <- vgmFit2lrv(lrv, vgModel, iterlim = 500)
```

```
## iteration = 0
## Step:
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Parameter:
##  [1]   1   0   1   0   0   1   0   0   0   1 100   1   0   1
0   0   1
## [18]   0   0   0   1
## Function Value
## [1] 20662
## Gradient:
##  [1] 2990.724 -698.729 4006.455 -491.125 -571.931 4553.456
-406.192
##  [8] -134.821  -47.577 2822.842   -6.529 2841.481 -696.400
3831.605
## [15] -486.606 -554.596 4363.374 -403.043 -131.120  -45.103
2704.497
##
## iteration = 263
## Parameter:
##  [1] -1.207e-02  8.141e-03  1.366e-07  5.173e-03  1.103e-07
1.672e-08
##  [7]  5.514e-02  5.185e-07  3.906e-07 -1.261e-07  4.156e+02
7.749e-01
## [13]  3.250e-01 -3.693e-01  1.635e-01 -1.407e-01 -1.723e-01
6.708e-01
## [19]  3.820e-02  1.691e-01  4.607e-01
## Function Value
## [1] 534.5
## Gradient:
##  [1]  1.596e-04 -1.334e-04 -1.313e-05  1.359e-05  2.867e-04
1.056e-04
##  [7]  8.500e-05  2.619e-06  4.169e-05 -1.464e-05 -3.499e-07
1.886e-04
## [13]  3.972e-05 -2.259e-04 -1.617e-04 -1.218e-04 -3.712e-04
-3.203e-05
## [19] -8.070e-05 -9.806e-05 -6.559e-05
##
## Gradient relatif proche de zéro.
## L'itération courante est probablement la solution.
```

```
plot(lrv, lrvg = vgram2lrvgram(fit$vg))
```


cadmium
copper
lead
zinc
Fv
0.0
1.0
2.0
100
200
300
400

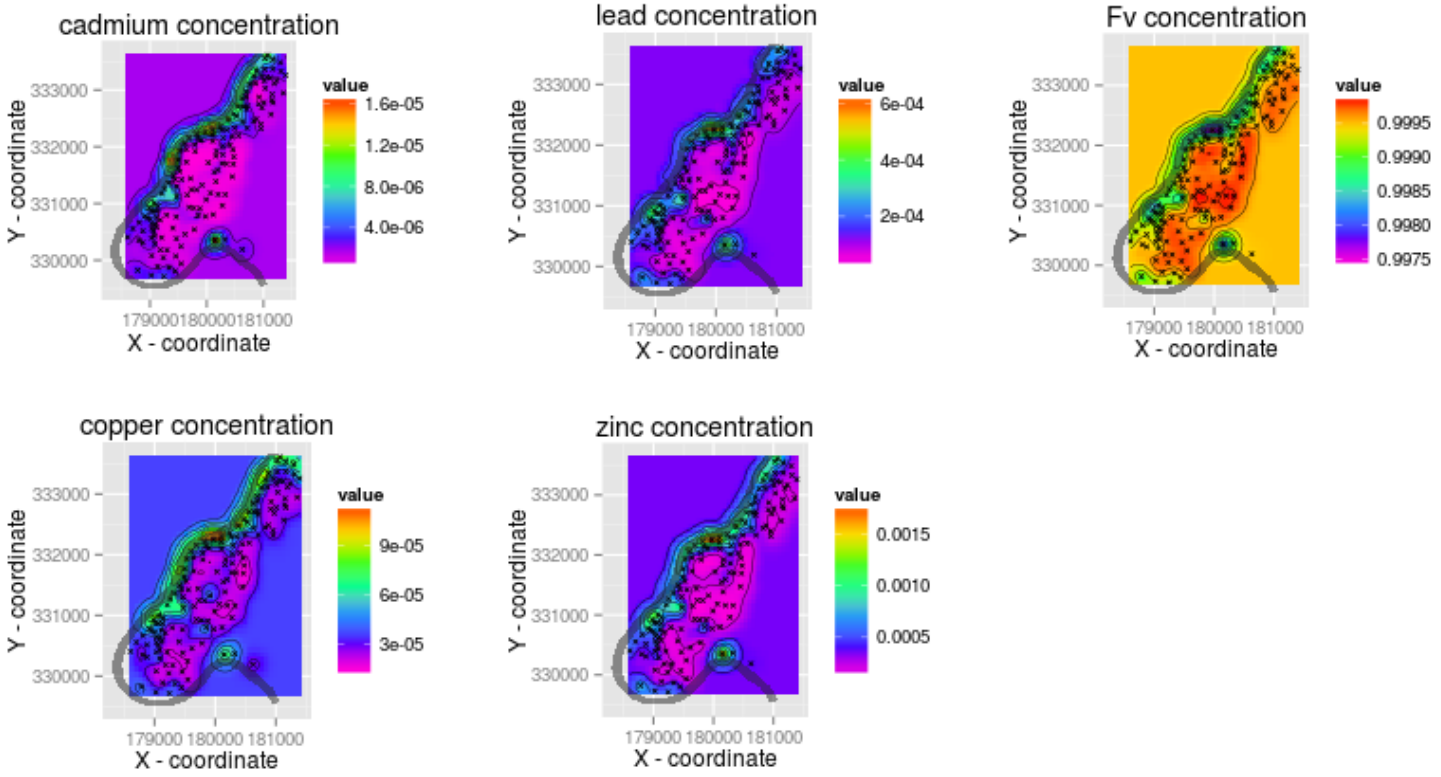
```
coordsNew <- expand.grid(x = seq(from = min(coords[, 1]), to =
max(coords[,
    1]), length = 50), y = seq(from = min(coords[, 2]), to =
max(coords[, 2]),
    length = 50))
krig <- compOKriging(comp, coords, coordsNew, fit$vg, err =
FALSE)  ## Kriging

meuse.riv2 <- as.data.frame(meuse.riv)
colnames(meuse.riv2) <- c("x", "y")

meuse.riv3 <- meuse.riv2[meuse.riv2[, 2] > min(coords[, 2]) - 200
& meuse.riv2[,
    2] < max(coords[, 2]) + 200, ]

p <- list()
for (i in 1:ncol(krig$Z)) {
    ggdata <- data.frame(krig$X, value = krig$Z[, i])
    p[[i]] <- ggplot(ggdata, aes(x = x, y = y, z = value)) +
geom_tile(aes(fill = value)) +
        stat_contour(lwd = 0.2) + geom_point(data = meuse, aes(x
= x, y = y,
        z = 0), pch = 4, size = 1) + geom_polygon(data =
meuse.riv3, aes(x = x,
        y = y, z = 0), fill = "grey15", alpha = 0.5) +
ggtitle(paste(colnames(krig$Z)[i],
        "concentration")) + xlab("X - coordinate") + ylab("Y -
coordinate") +
        coord_equal() + scale_fill_gradientn(colours =
rev(rainbow(7)))
}

multiplot(p[[1]], p[[2]], p[[3]], p[[4]], p[[5]], cols = 3)
```



References

Aitchison J. (1986). The Statistical Analysis of Compositional Data. London: Chapman and Hall.

Aitchison, J. and Greenacre, M., (2002). Biplots of compositional data. J. Royal Stat. Soc. Series C (Appl. Stat.) 51(4):375-392.

Bacon-Shone, J. (2011). “A short history of compositional data analysis,” in Compositional data analysis: Theory and Applications, ed. Pawlowsky-Glahn V., and Buccianti A., [NY: John Wiley and Sons], 3-11.

Bocard, D., Gillet, F. and Legendre, P. (2011). Numerical ecology with R. NY: Springer, 319 p.

Pawlowsky-Glahn, V. and Egozcue, J.J. (2006) Compositional data and their analysis: an introduction. In Compositional Data Analysis in the Geosciences: From Theory to Practice. Geological Society (London, England), 145-159.

Filzmoser, P., Hron, K., and Reimann, C. (2009). Univariate statistical analysis of environmental (compositional) data: Problems and possibilities. Sci. Total Environ. 407(23):6100-8.

Parent, S.- É., Parent, L. E. Rozane, D. E. Hernandes, A., Natale, W. 2012. Nutrient balance as paradigm of plant and soil chemometrics. Chapter 4 (32 pp.). In Issaka, R.N. (ed.). Soil Fertility, InTech Publ., http://www.intechopen.com/books/soil-fertility.

Pearson, K. (1897). Mathematical contributions to the theory of evolution. On a form of spurious correlation which may arise when indices are used in the measurement of organs. Proc. Roy. Soc, London LX, 489-502.

Rock, N.M.S. (1988). Numerical Geology. Lecture Notes in Earth Sciences, 18. Springer-Verlag, Berlin.

Tolosana-Delgado et van den Boogart (2011). “Linear models with compositions in R”, in Compositional data analysis: Theory and Applications, ed. Pawlowsky-Glahn V., and Buccianti A., [NY: John Wiley and Sons], 356-371.

Tolosana-Delgado, R., van den Boogart, K.G. and Pawlowsky-Glahn, V. (2011). “Geostatistics for compositions”, in Compositional data analysis: Theory and Applications, ed. Pawlowsky-Glahn V., and Buccianti A., [NY: John Wiley and Sons], 73-86.

Tolosana-Delgado, R. and van den Boogart, K.G. (2012). A guide to the geostatistical analysis of compositional data, Z. geol. Wiss., Berlin 40 4/5: 281 – 294, 5 Abb., 1 Tab. http://www.zgw-online.de/pages/_ausgabe2012.php