Tutorial to the package StatDA

Peter Filzmoser

Institute of Statistics and Probability Theory
Vienna University of Technology, Austria

January 25, 2009

Contents

Chapter 1

Introduction

This document intends to show the main features of the package `StatDA` . The reader should get an idea about the main methods that are implemented, and about their use within R.

`StatDA` uses mainly examples from environmental studies. In fact, here mainly the Kola data set is used for demonstrating the methods. This data set resulted from a larger geochemical study carried out at the peninsula Kola in Northern Europe. A detailed description of the data set can be found in the book “Statistical Data Analysis Explained. Applied Environmental Statistics with R.” [1]. This book is also the basis for the package `StatDA` . It described all the methods in detail, shows various applications of the methods, and explains the results.

Of course, `StatDA` can also be used for other kind of data. The methods are mostly standard methods used for analyzing uni- and multivariate data. Many methods with focus on exploratory data analysis are included, because getting an appropriate view on the data can be very valuable in deepening the knowledge about the data structure.

The web page

http://www.statistik.tuwien.ac.at/StatDA/R-scripts

can also be helpful–especially for the non-experienced R user–to make first steps of analyzing data with R. There one can find all the figures included in the previously mentioned book, together with all R-code for generating the figures. With copy/paste of the code into an R-session it will be possible to reproduce the plots exactly. Including your own data instead of the data used in the book will allow for a perfect approach in analyzing your data.

[1] C. Reimann, P. Filzmoser, R.G. Garrett, R. Dutter; Wiley, Chichester, 2008

Chapter 2

Graphics to Display the Data Distibution

The Kola project includes the chemical analysis for ca. 50 chemical elements at about 600 sample sites. This large amount of information needs to be summarised. First of all it is important to know which distributions the data follow. Therefore some basic statistical plots can be made. The easiest plots are the histogram, density plot, scatterplot and a boxplot. The graphics can look very different for the different samples. For example, the distribution can be symmetrical or asymmetrical. The data can fall into two groups or single extreme outliers can appear.

It is not necessary to explain those standard tools of statistical analysis. It is more important to mention other graphics which compare the empirical distribution with an underlying distribution. The well known plots for such graphics are the QQ-plot, QP-plot and the PP-plot. In this chapter the main focus will be on the boxplot and its problems with outlier detection in the case of non normally distributed data sets. The displayed graphic includes the histogram, density trace, scatterplot, boxplot and the Empirical Cumulative Distribution Function (ECDF) of the original data (upper graphics) and the log-transformed data (lower graphics). The curve of the ECDF plot jumps witch each data point by $1/n$, where n is the number of data points. The x-axis shows the variable values and the y-axis the corresponding probabilities of the empirical cumulative distribution function (values between 0 and 1).

The package `StatDA` loads several other packages which are automatically installed when installing `StatDA` . Loading `StatDA` in an R session with `library(StatDA)` will thus give the message for some of the additional packages, that they are also loaded. The commands for generating some plots for graphically inspecting the data are given below.

```
-----------------------------------------------------------------
Analysis of geostatistical data
For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
geoR version 1.6-21 (built on 2008-04-23) is now loaded
-----------------------------------------------------------------


use of mclust requires a license agreement
see http://www.stat.washington.edu/mclust/license.txt
This is mgcv 1.4-1
```

An important plot is the log-boxplot. The boxplot is built around the median which divides any sorted data set into two equal halves. The length of the box is determined by the first (25%) and third (75%) quartile. The distance of these quartiles is the IQR (interquartile range), the from each side of the box 1.5 times IQR is considered. Values being further away are potential outliers. The so-called whiskers are drawn to the most extreme data points within this range (i.e. the

```
> library(StatDA)
> data(chorizon)
> Ba = chorizon[, "Ba"]
> n = length(Ba)
> par(mfcol = c(2, 2), mar = c(4, 4, 2, 2))
> edaplotlog(Ba, H.freq = F, box = T, H.breaks = 30, S.pch = 3,
+     S.cex = 0.5, D.lwd = 1.5, P.log = F, P.main = "", P.xlab = "Ba [mg/kg]",
+     P.ylab = "Density", B.pch = 3, B.cex = 0.5, B.log = TRUE)
> edaplot(log10(Ba), H.freq = F, box = T, S.pch = 3, S.cex = 0.5,
+     D.lwd = 1.5, P.ylab = "Density", P.log = T, P.logfine = c(5,
+         10), P.main = "", P.xlab = "Ba [mg/kg]", B.pch = 3, B.cex = 0.5)
> plot(sort(Ba), ((1:n) - 0.5)/n, pch = 3, cex = 0.8, main = "",
+     xlab = "Ba [mg/kg]", ylab = "Probability", cex.lab = 1, cex.lab = 1.4)
> abline(h = seq(0, 1, by = 0.1), lty = 3, col = gray(0.5))
> abline(v = seq(0, 1400, by = 200), lty = 3, col = gray(0.5))
> plot(sort(log10(Ba)), ((1:n) - 0.5)/n, pch = 3, cex = 0.8, main = "",
+     xlab = "Ba [mg/kg]", ylab = "Probability", cex.lab = 1, xaxt = "n",
+     cex.lab = 1.4)
> axis(1, at = log10(alog <- sort(c((10^(-50:50)) %*% t(c(5, 10))))),
+     labels = alog)
> abline(h = seq(0, 1, by = 0.1), lty = 3, col = gray(0.5))
> abline(v = log10(alog), lty = 3, col = gray(0.5))
```

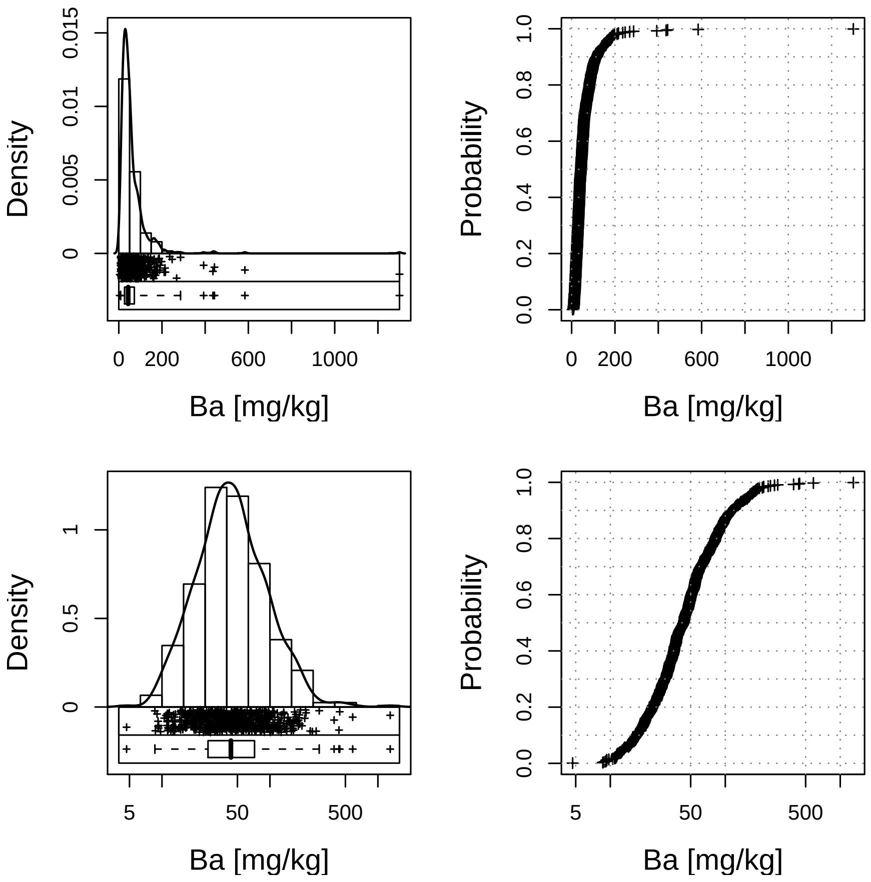


Figure 2.1: Histogram, density trace, scatterplot, and boxplot in one display combined with the ECDF-plot.Original data (upper diagrams) and log-transformed data (lower diagrams)

largest non-outliers). The problem is that the calculation of the whiskers of the boxplot assumes data symmetry. Thus also the recognition of extreme values is based on symmetry at the central data. In geochemistry we are often faced with right-skewed data distributions and therefore the boxplot will underestimate the number of lower extreme values and overestimate the number of upper extreme values. A solution for this problem is that the data can be log-transformed and then the boxplot can be computed. The result are the values (median, quantiles, whiskers,...) for the log-transformed data. It is possible to back-transform these values. Median and quartiles would not change for original or log-transformed data, but the whiskers for identifying outliers will change. This version of the boxplot is called the log-boxplot.

Chapter 3

Statistical Distribution Measures

The main point of view of this chapter are statistical measures of the distribution. The well known measures for the central value are the arithmetic mean, the geometric mean, the mode and the median. Robust measures like the median are useful for environmental data because the influence of outliers is reduced. Another possibility is to trim the most extreme data values. There are other methods for robust estimation such as a weighted estimator. If someone chooses this method, the data points far away from the centre of the distribution are down-weighted.

Other important values are the measures of spread. The well known values are the range, the Interquartile Range (IQR), the Standard Deviation (SD) and the Variance. One additional value is discussed in the book and this is the Median Absolute Deviation (MAD). The MAD is the robust equivalent to the SD. In contrast to the Standard Deviation the median is taken as the central value. Now the median of the absolute deviations around this median is computed. As it is easy to see, the MAD is robust against up to 50% of extreme values.

Other measures of spread are based on the quantiles, quartiles and percentiles. There is also a measure which is independent from the magnitude of the data because it is usually expressed in percent. This percentage exists in a robust and non robust version. The non robust version is called the coefficient of variation (CV) and the other one is called the robust coefficient of variation (CVR). The CV is defined as SD divided by the arithmetic mean, and the CVR is defined as MAD divided by the median (expressed in %).

In Table 3.1 the most important distribution measures are displayed for selected variables of the Kola Project moss data set.

	MIN	Q1	MEDIAN	MEAN	Q3	MAX	SD	MAD	CV %	CVR %
Ag	0.01	0.02	0.03	0.05	0.05	0.82	0.06	0.02	114.42	57.02
Al	33.90	141.00	192.00	297.47	282.00	4850.00	454.33	89.70	152.73	46.72
As	0.04	0.13	0.17	0.26	0.27	3.42	0.30	0.08	115.65	48.85
Au	0.03	0.14	0.21	0.41	0.38	18.80	1.01	0.15	247.76	70.60
B	0.25	1.09	1.75	2.16	2.73	21.60	1.73	1.08	80.20	62.02
Ba	6.71	15.20	19.00	21.33	23.80	175.00	11.98	6.23	56.18	32.77
Be	0.01	0.01	0.01	0.03	0.01	1.51	0.08	0.00	289.28	0.00
Bi	0.00	0.01	0.02	0.03	0.03	0.54	0.03	0.01	130.12	65.89
Ca	1680.00	2360.00	2620.00	2733.40	2930.00	9320.00	668.86	415.13	24.47	15.84
Cd	0.02	0.07	0.09	0.11	0.12	1.23	0.09	0.04	80.40	39.98
Co	0.11	0.24	0.39	0.88	0.88	13.20	1.39	0.30	158.02	76.03
Cr	0.10	0.43	0.60	0.90	0.98	8.59	0.99	0.33	109.32	54.36
Cu	2.63	4.74	7.05	16.10	15.90	214.00	24.24	4.51	150.55	63.99
Fe	46.50	145.00	211.00	379.52	379.50	5140.00	531.24	126.02	139.98	59.73
Hg	0.02	0.04	0.05	0.05	0.07	0.15	0.02	0.02	35.86	32.62
K	2260.00	3750.00	4220.00	4349.18	4760.00	8590.00	882.59	756.13	20.29	17.92
La	0.35	0.35	0.35	0.65	0.35	35.20	1.87	0.00	286.59	0.00
Mg	518.00	918.25	1090.00	1131.79	1270.00	2380.00	281.98	260.20	24.91	23.87
Mn	28.50	296.00	434.00	445.41	579.25	1170.00	203.59	210.53	45.71	48.51
Mo	0.02	0.06	0.08	0.10	0.12	1.08	0.09	0.04	85.16	45.40
Na	5.00	42.32	71.20	106.93	135.75	918.00	98.62	53.15	92.23	74.65
Ni	0.96	2.26	5.36	18.75	17.58	396.00	39.44	5.37	210.38	100.22
P	511.00	1070.00	1260.00	1257.63	1420.00	3800.00	286.85	237.22	22.81	18.83
Pb	0.84	2.29	2.97	3.32	3.83	29.40	2.05	1.11	61.63	37.44
Pd	0.03	0.35	0.70	1.79	1.47	70.70	4.55	0.62	254.71	87.27
Pt	0.10	0.10	0.10	0.42	0.30	13.80	0.99	0.00	236.85	0.00
Rb	1.39	7.80	11.50	11.98	15.35	33.50	5.59	5.54	46.65	48.15
S	543.00	792.00	862.00	886.20	950.75	2090.00	153.00	117.13	17.26	13.59
Sb	0.01	0.03	0.04	0.05	0.06	0.62	0.05	0.02	87.58	45.76
Sc	0.05	0.05	0.05	0.06	0.05	1.10	0.06	0.00	95.37	0.00
Se	0.40	0.40	0.40	0.40	0.40	1.23	0.04	0.00	9.44	0.00
Si	24.90	142.00	196.50	211.60	260.75	983.00	106.40	85.25	50.28	43.38
Sr	2.47	6.47	9.37	15.55	15.00	435.00	29.48	5.26	189.54	56.09
Th	0.00	0.01	0.02	0.04	0.04	1.14	0.07	0.01	186.30	58.01
Tl	0.00	0.01	0.02	0.03	0.04	0.35	0.03	0.02	98.12	70.91
U	0.00	0.01	0.01	0.02	0.02	0.45	0.04	0.01	185.84	67.39
V	0.28	1.08	1.56	2.55	2.49	83.80	4.57	0.87	179.13	56.25
Y	0.05	0.05	0.05	0.14	0.05	5.90	0.38	0.00	272.96	0.00
Zn	11.70	26.10	32.20	33.78	38.90	81.90	10.80	9.34	31.96	29.01

Table 3.1: Summary statistics for selected elements of the Kola moss data.

Chapter 4

Mapping Spatial Data

Geochemical data, as well as many other data sets, are spatially dependent. This is a big challenge for analyzing the data, since the relation to the spatial coordinates can be taken into account. Thus we are not only interested in the statistical distribution of the data, but also in the geographical distribution, which can be represented in a map. Many different methods exist to show this structure. Some of them look informative and others are featureless. It is surprising that scientists pay so little attention to the preparation of geochemical maps. There is no common effective method to display environmental data on a map. Our main focus in this chapter is to present some mapping methods.

4.1 Mapping Geochemical Data with Proportional Dots

This method is very easy to understand and results in a map which is easy to read. It is also one of the most popular ways for black and white maps. The main idea is that the absolute value of the sample is important. All data values will be represented by dots in the map. If one takes linearly growing dots there can be two problems. The first one is that too many dots are similar in size and it is difficult to distinguish different values. On the other hand it is possible that the maximum is very far away from the rest of the data and then one huge dot will appear while the remaining dots are very small.

An alternative is exponentially growing dots. It is true that the problem with one huge dot can also appear. Therefore one has to modify the method by using the same point size for, say, the smallest 10 percent and the largest 1 percent of the data. In between the dots grow exponentiallay with the result that the dot size slows down near the detection limit and becomes steeper towards the larger values.

Figure 4.1 shows the map of the Kola project area, and the information of As in C-horizon is shown with exponentially growing dots with limited lower and upper point size.

4.2 Percentile Classes

Another approach for plotting geochemical data are classes. In this case the data set will be grouped and the resulting classes will be plotted. To distiguish the different groups the user can select different coloured symbols, or black and white. One possibility is to use percentiles for grouping the values. This is an easy way because there is no assumption about the underlying data distribution. However, it is necessary to make a choice of the percentiles for grouping the data. When someone wants to spread the symbols across the range of values it is possible to use the $20^{\text{th}}, 40^{\text{th}}, 60^{\text{th}}, 80^{\text{th}}$ percentiles. Geochemists are often interested in the tails of the distribution

```
> par(mfrow = c(1, 1))
> data(kola.background)
> el = chorizon[, "As"]
> X = chorizon[, "XCOO"]
> Y = chorizon[, "YCOO"]
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> bubbleFIN(X, Y, el, S = 9, s = 2, plottitle = "", legendtitle = "As [mg/kg]",
+     text.cex = 0.6, legtitle.cex = 0.7, ndigits = 2)
> text(min(X) + diff(range(X)) * 5/7, max(Y), "Exponentially",
+     cex = 0.7)
> text(min(X) + diff(range(X)) * 5/7, max(Y) - diff(range(Y))/25,
+     "growing dots", cex = 0.7)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 40000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```

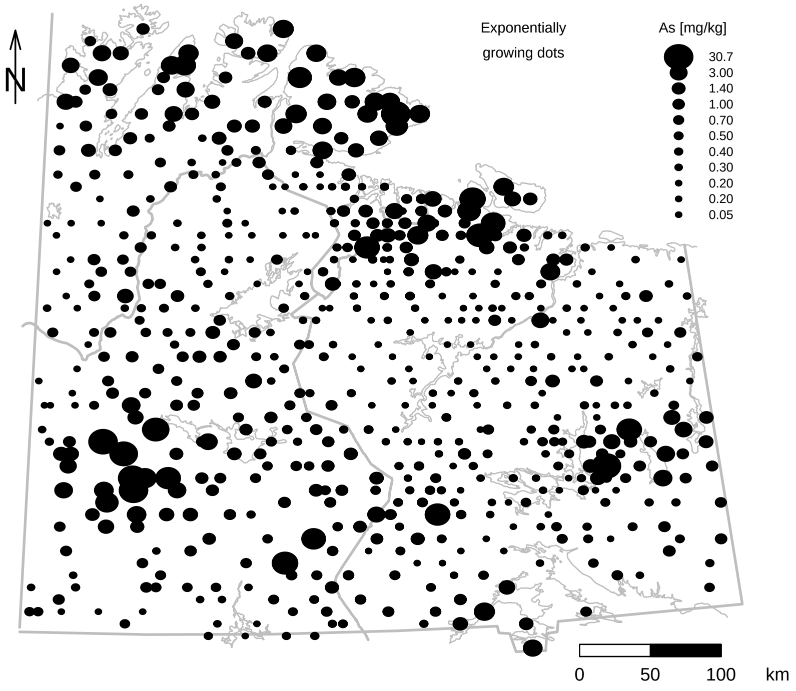


Figure 4.1: The distribution of As in C-horizon with exponentially growing dots.

and therefore they use the $5^{\text{th}}, 25^{\text{th}}, 75^{\text{th}}, 95^{\text{th}}$ percentiles. Further classes can also be added. Figure 4.2 shows the map of the As in C-horizon for the above mentioned percentiles.

Another possibility is to use the boxplot instead of percentiles for building the classes. The problem is the same as mentioned in the previous chapter. The boxplot identifies a reliable number of outliers if the data distribution is symmetric. Therefore it can be necessary to use another version of the boxplot such as the log-boxplot.

4.3 Surface Maps Constructed with Smoothing Techniques

Many users are interested in maps that look "pleasing", but for geochemist this representation has several disadvantages. Firstly, they loose information of the local variablility and secondly, the maps are dependend on the algorithm. It can also happen that the untrained user gets the impression of greater precision than the samples support. An easy way to produce surface maps is based on moving averages. To construct such a map the user has to define a window of fixed size. Now this window is moved across the map and all values inside the window will be represented by one central value (e.g. the median). With this method it is possible to estimate new values on the grid. This is a very simple method and there exist much more complex smoothing methods.

One possibility is to use a polynomial function for interpolation. This method results in a smoother display of the map but the number of classes is a critical step. This method has the same problems as a representation with proportional dots. This can be solved by using 100 percentile classes, each with its own grey value. If the user wants less classes he should choose the groups according to the data (e.g. $5^{\text{th}}, 25^{\text{th}}, 75^{\text{th}}, 95^{\text{th}}$ percentiles). Figure 4.3 shows a smoothing with a polynomial function with 100 percentiles.

4.4 Surface Maps Constructed With Kriging

Kriging is a statistically optimised estimator for interpolation. Many different methods exist for kriging, like block kriging (interpolation in a whole block) and point kriging (interpolation for any location in the space). The main idea is that points in the neighbourhood have more in common than points with a greater distance. The cruical point is to weight the different distances.

The basis for kriging is the (semi)variogram. This variogram displays the variance between data points at defined distances, and the distance is the Euclidean distance in the survey area. The maximum on the x-axis is 30 to 50 percent of the maximum distance in the survey area. In the case of the Kola project the maximum distance in the survey area is about 700 km. With the Euclidean distance any direction is possible but sometimes only one direction can be interesting. Therefore the distances can be calculated in only one direction, e.g. north-south or east-west. The omnidirectional variogram is a kind of average over all possible directions. In Figure 4.4 (left) the variogram for different directions is displayed, and the right graphic shows the so called spherical variogram that can be used as a model for kriging.

```
> el = chorizon[, "As"]
> X = chorizon[, "XCOO"]
> Y = chorizon[, "YCOO"]
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> SymbLegend(X, Y, el, type = "percentile", qutiles <- c(0, 0.05,
+     0.25, 0.75, 0.95, 1), symbtype = "EDA", symbmagn = 0.8, leg.position = "topright",
+     leg.title = "As [mg/kg]", leg.title.cex = 0.8, leg.round = 2,
+     leg.wid = 4, leg.just = "right")
> text(min(X) + diff(range(X)) * 4/7, max(Y), paste(qutiles * 100,
+     collapse = ","), cex = 0.8)
> text(min(X) + diff(range(X)) * 4/7, max(Y) - diff(range(Y))/25,
+     "Percentiles", cex = 0.8)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```

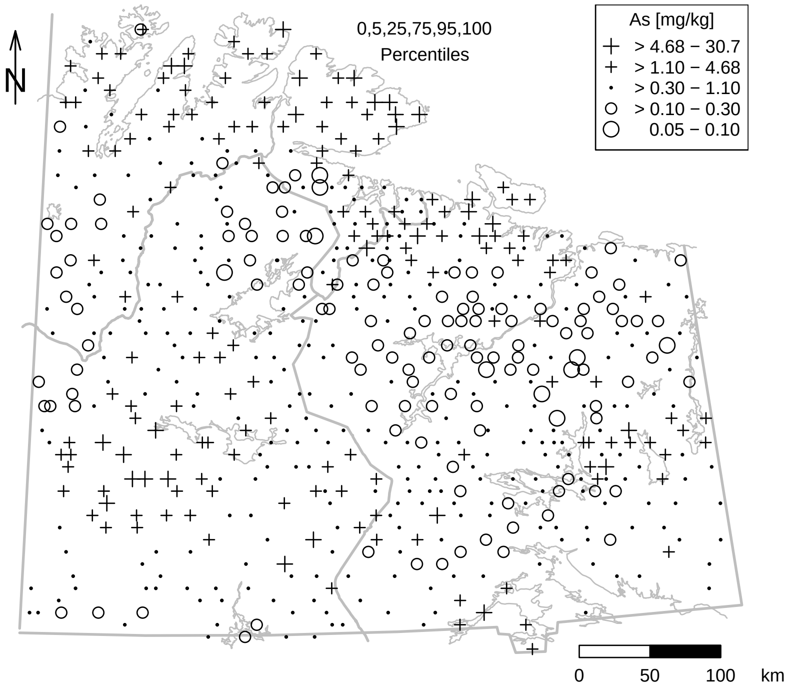


Figure 4.2: Geochemical map of the variable As

```
> X = chorizon[, "XCOO"]
> Y = chorizon[, "YCOO"]
> el = log10(chorizon[, "As"])
> data(kola.background)
> data(bordersKola)
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> SmoothLegend(X, Y, el, resol = 200, type = "contin", whichcol = "gray",
+     qutiles = c(0, 0.05, 0.25, 0.5, 0.75, 0.9, 0.95, 1), borders = "bordersKola",
+     leg.xpos.min = 780000, leg.xpos.max = 8e+05, leg.ypos.min = 7760000,
+     leg.ypos.max = 7870000, leg.title = "mg/kg", leg.title.cex = 0.7,
+     leg.numb.cex = 0.7, leg.round = 2, leg.wid = 4, leg.numb.xshift = 70000,
+     leg.perc.xshift = 40000, leg.perc.yshift = 20000, tit.xshift = 35000)
> plotbg(map.col = c("gray", "gray", "gray", "gray"), map.lwd = c(1,
+     1, 1, 1), add.plot = T)
> text(min(X) + diff(range(X)) * 4/7, max(Y), "As", cex = 1)
> text(min(X) + diff(range(X)) * 4/7, max(Y) - diff(range(Y))/28,
+     "in C-horizon", cex = 0.8)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```

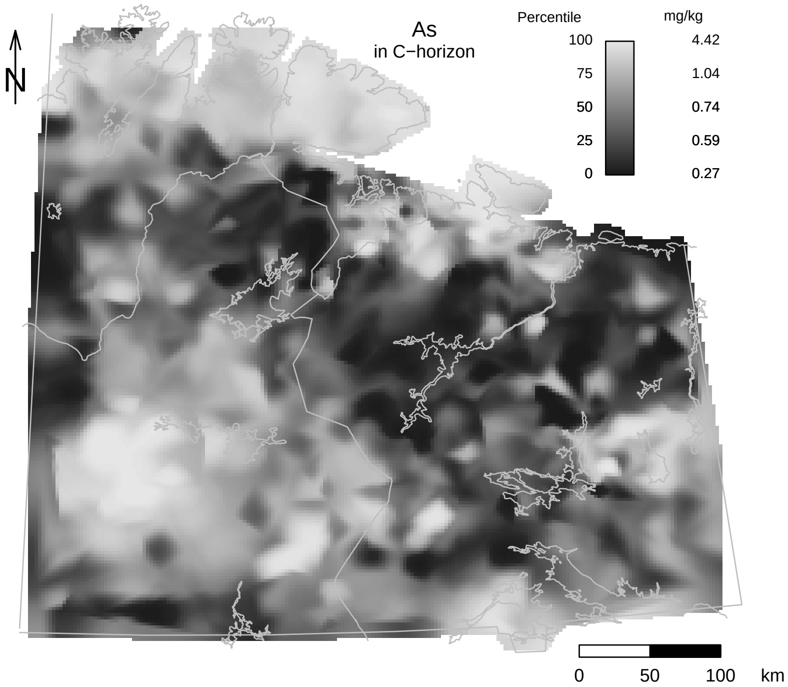


Figure 4.3: Smoothed surface map of As in Kola C-horizon using 100 percentiles.

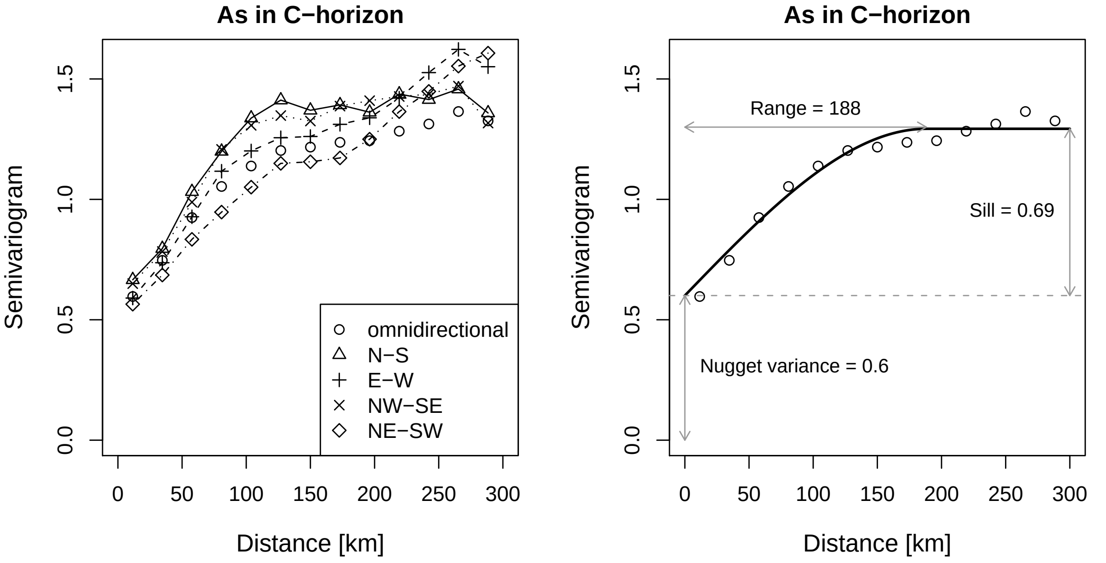


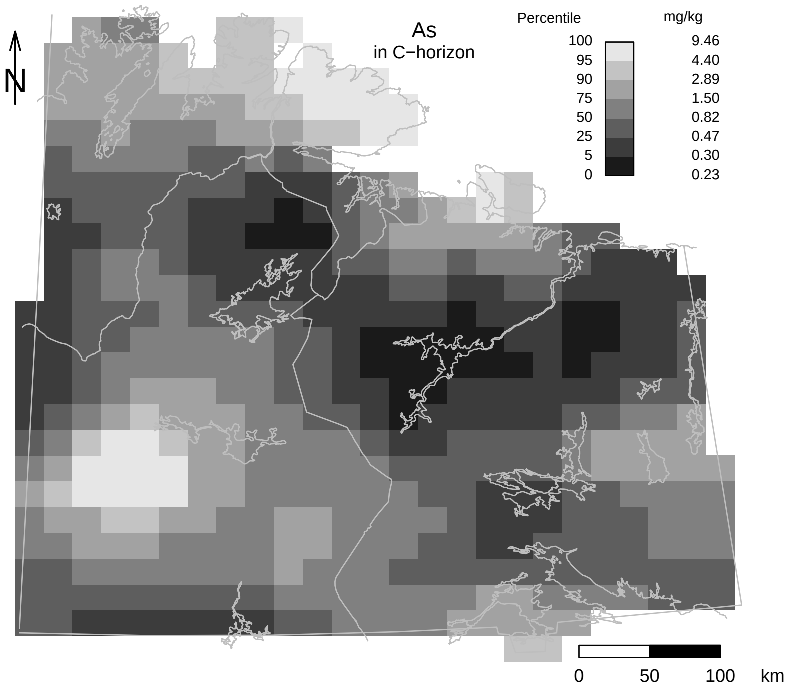
Figure 4.4: Different directions (left) for the As in C-horizon samples and spherical semivariogram for the omnidirectional semivariogram (right).

Only the important part of the code is shown below:

```
> X = chorizon[, "XCOO"]/1000
> Y = chorizon[, "YCOO"]/1000
> el = chorizon[, "As"]
> vario.b <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300)
> vario.c <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, op = "cloud")
> vario.bc <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, bin.cloud = TRUE)
> vario.s <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, op = "sm", band = 10)
> vario.0 <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, dir = 0, tol = pi/8)
> vario.90 <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, dir = pi/4, tol = pi/8)
> vario.45 <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, dir = pi/8, tol = pi/8)
> vario.120 <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 300, dir = 3 * pi/8, tol = pi/8)
> data(res.eyefit.As_C)
> v5 = variofit(vario.b, res.eyefit.As_C, cov.model = "spherical",
+     max.dist = 300)
```

For further information we refer to the book. Now we have done all of the preparition to create a kriging plot of the map for the Kola project. As mentioned above it is possible to make kriging for a block or a single point. In our case we plot the interpolation of the map with block kriging. Whether the blocks are visible or not depends on the size of the block. For a better resolution smaller block sizes need to be used.

```
> par(mfrow = c(1, 1))
> X = chorizon[, "XCOO"]
> Y = chorizon[, "YCOO"]
> xwid = diff(range(X))/120000
> ywid = diff(range(Y))/120000
> el = chorizon[, "As"]
> vario.b <- variog(coords = cbind(X, Y), data = el, lambda = 0,
+     max.dist = 3e+05)
> data(res.eyefit.As_C_m)
> data(bordersKola)
> v5 = variofit(vario.b, res.eyefit.As_C_m, cov.model = "spherical",
+     max.dist = 3e+05)
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> KrigeLegend(X, Y, el, resol = 25, vario = v5, type = "percentile",
+     whichcol = "gray", qutiles = c(0, 0.05, 0.25, 0.5, 0.75,
+         0.9, 0.95, 1), borders = "bordersKola", leg.xpos.min = 780000,
+     leg.xpos.max = 8e+05, leg.ypos.min = 7760000, leg.ypos.max = 7870000,
+     leg.title = "mg/kg", leg.title.cex = 0.7, leg.numb.cex = 0.7,
+     leg.round = 2, leg.numb.xshift = 70000, leg.perc.xshift = 40000,
+     leg.perc.yshift = 20000, tit.xshift = 35000)
> plotbg(map.col = c("gray", "gray", "gray", "gray"), map.lwd = c(1,
+     1, 1, 1), add.plot = T)
> text(min(X) + diff(range(X)) * 4/7, max(Y), "As", cex = 1)
> text(min(X) + diff(range(X)) * 4/7, max(Y) - diff(range(Y))/28,
+     "in C-horizon", cex = 0.8)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```



Chapter 5

Further Graphics for Exploratory Data Analysis

Another graphic to study the relationship between two different variables is the scatterplot. This plot is very easy to understand because the x-axis represents one variable and the y-axis represents the second variable. The result is a cloud corresponding to n data points, and the shape of the cloud depends on the examined variables. After one has plotted the cloud it is possible to add a regression line to the graphic. The first step is to find a linear relation instead of an non-linear one. This is the situation were the theory of linear regression enters the field. To calculate a regression line one has two possibilities: a robust and a non-robust estimation. Basic information on the topic linear regression will be given later on.

The next step is to focus on a trend in different directions of the geochemical samples. Therefore we have to specify a special point and to determine the concentration in east-west or any other direction. The concentration can be plotted in a diagram where the x-axis represents the distance to the chosen point and the y-axis shows the concentration of the element. In a map one can plot the points which are studied. In Figure 5.1 (right) we can see that the Cu concentration decreases with the distance to Monchegorsk (the chosen point).

Another possibility is to examine the concentration in one direction from the reference point. With this assumption it is possible to make an examination of the concentration in north-west to south-east direction. It is only necessary to use a subarea of the total data set.

In Figure 5.2 the reference point (Nikel/Zapojarnij) is in the north-west and the subarea goes to Monchegorsk in the south-east. In the plot on the right-hand side it is visible that near Nikel-Zapojarnij and Monchegorsk the concentration of Cu is higher than between the two places. The width of the peak says that the emissions from Monchegorsk have a larger spread.

Another interessting plot is the so called ternary diagnram. In this diagram it is possible to compare three variables in just one graphic. It is important to mention that the relative proportion is plotted and not the absolut value. It is clear that the three variables should sum up to 100 percent to be plotted in the diagram and it is important that the three elements have the same order of magnitude. Otherwise the points will be concentrated in one corner. If the difference between the variables is to high, they should be transformed so that they are in the same data range. Figure 5.3 shows a ternary plot of Al-Fe-Mn of the Kola project moss data. We can see that most of the points tend to the Mn corner of the diagram.

```
> data(moss)
> par(mfrow = c(1, 2), mar = c(4, 4, 2, 2))
> plotbg(map.col = c("gray", "gray", "gray", "gray"), xlab = "UTM east [m]",
+     ylab = "UTM north [m]", cex.lab = 1.2)
> X = moss[, "XCOO"]
> Y = moss[, "YCOO"]
> points(X[Y < 7600000 & Y > 7500000], Y[Y < 7600000 & Y > 7500000],
+     pch = 3, cex = 0.7)
> x = (X[Y < 7600000 & Y > 7500000] - 753970)/1000
> y = log10(moss[Y < 7600000 & Y > 7500000, "Cu"])
> plot(x, y, xlab = "Distance from Monchegorsk [km]", ylab = "Cu in Moss [mg/kg]",
+     yaxt = "n", pch = 3, cex = 0.7, cex.lab = 1.2)
> axis(2, at = log10(a <- sort(c((10^(-50:50)) %*% t(c(2, 5, 10))))),
+     labels = a)
> lines(smooth.spline(x, y), col = 1, lwd = 1.3)
```

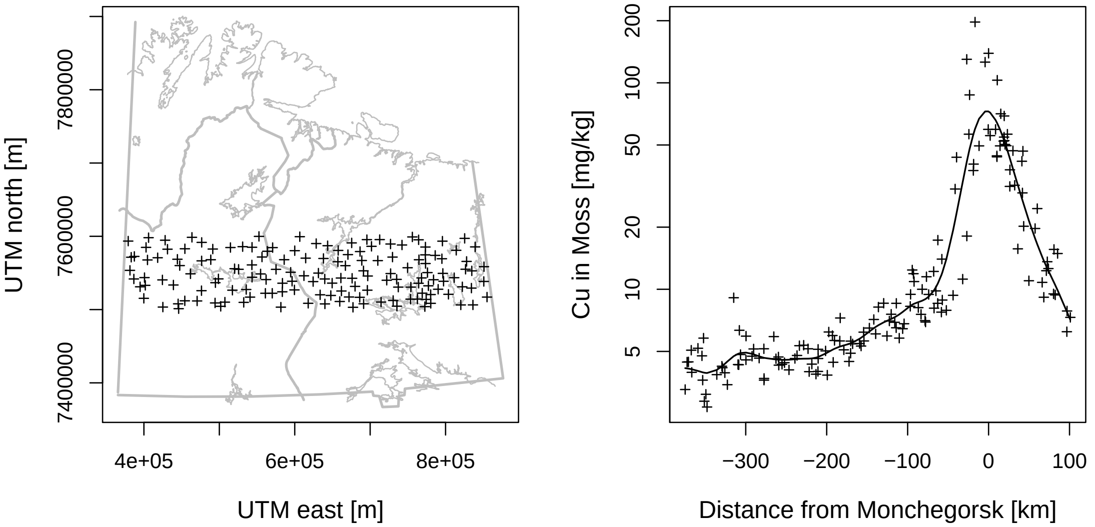


Figure 5.1: Points in the map used for the east-west transect (left). Plot which shows the decrease of the Cu concentration with larger distance to Monchegorsk.

```
> library(sgeostat)
> X = moss[, "XCOO"]
> Y = moss[, "YCOO"]
> Cu = moss[, "Cu"]
> loc = list(x = c(590565.1, 511872.8, 779702.8, 861156.3), y = c(7824652,
+     7769344, 7428054, 7510341))
> loc.in = in.polygon(X, Y, loc$x, loc$y)
> ref = c(542245.3, 7803068)
> par(mfrow = c(1, 2), mar = c(4, 4, 2, 2))
> plotbg(map.col = c("gray", "gray", "gray", "gray"), xlab = "UTM east [m]",
+     ylab = "UTM north [m]", cex.lab = 1.2)
> points(X[!loc.in], Y[!loc.in], pch = 16, cex = 0.3)
> points(X[loc.in], Y[loc.in], pch = 3, cex = 0.7)
> points(ref[1], ref[2], pch = 16, cex = 1)
> polygon(loc$x, loc$y, border = 1, lwd = 1.3)
> distanc = sqrt((X[loc.in] - ref[1])^2 + (Y[loc.in] - ref[2])^2)
> plot(distanc/1000, log10(Cu[loc.in]), xlab = "Distance from reference point [km]",
+     ylab = "Cu in Moss [mg/kg]", yaxt = "n", pch = 3, cex = 0.7,
+     cex.lab = 1.2)
> axis(2, at = log10(a <- sort(c((10^(-50:50)) %*% t(c(2, 5, 10))))),
+     labels = a)
> lines(smooth.spline(distanc/1000, log10(Cu[loc.in]), spar = 0.8),
+     col = 1, lwd = 1.3)
```

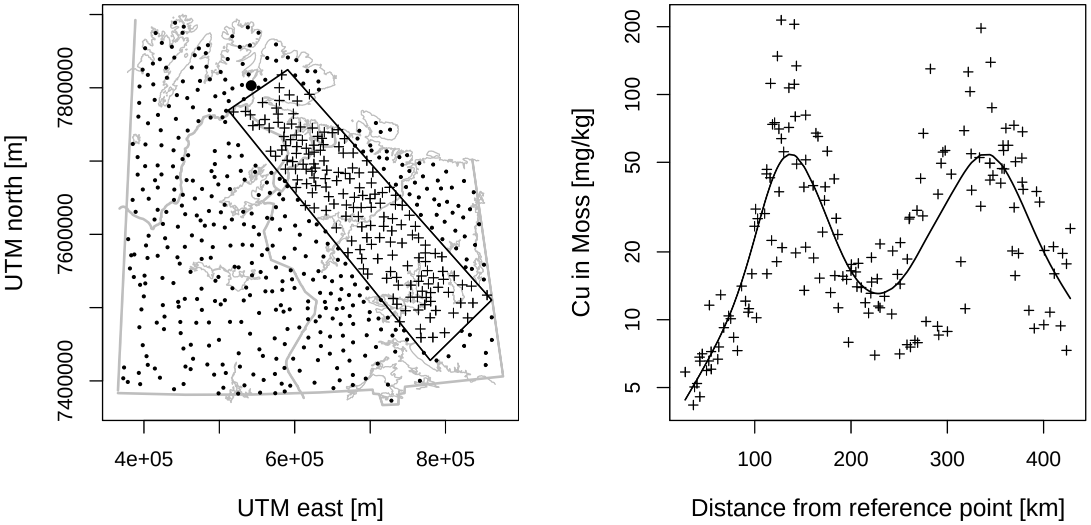


Figure 5.2: The subarea includes Nikel/Zapoljanij and Monchegorsk. The refernce point (dot) is in the north-west part of the survey area (left). Right: The spatial distance plot and the concentrations in the subarea.

```
> par(mfrow = c(1, 1))
> x = moss[, c("Al", "Fe", "Mn")]
> ternary(x, grid = TRUE, pch = 3, cex = 0.7, col = 1)
> text(0.21, 0.26, "563")
> text(0.1, 0.05, "726")
> text(0.21, 0.06, "325")
```

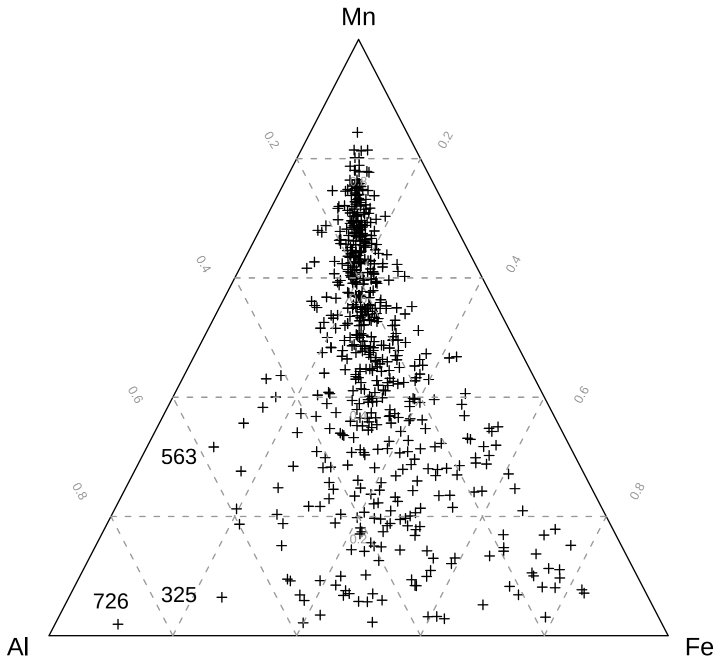


Figure 5.3: Ternary plot for selected elements of the moss data set.

Chapter 6

Comparing Data in Tables and Graphics

In this chapter we go back to our standard analysis methods. Sometimes it is not so important to examine a relation betweenn different elements but it can be important to analyse the change in the concentration of one element over the time. First of all the user has to ask if there are differences and what causes the differences. The next question is if the difference is significant in a statistical way. In the Kola project the change over the time is not so important but it is interessting to analyse the concentration in the different parts of the ecosystem. The data set of the project consists of four layers: moss, O-horizon, B-horizon and C-horizon. For detailed information about the different layers we recommend to read Chapter 1 of the book.

Now we are able to produce some interesting plots of the four layers of the ecosystem for one element (Al). Figure 6.1 displays the density traces of Al in all four sample materials. The data were log-transformed to make them more symmetric.

Another interesting graphic is a CP-plot (Cumulative Probability plot) of the data set. This can be made in the following way where every layer of the ecosystem is displayed with a different symbol. It can be a better solution to plot them with different colours.

The last method to display a statistical information of the Al element in the four layers is the boxplot. This graphic has the advantage that it provides a statistical summary and that it makes only minor use of a model (just for calculating the ends of the whiskers).

In Figure 6.3 it is visible that there are many upper outliers in moss. The number of outliers is lower in the other sample layers.

It is also possible to make such a comparison of the spatial data structure. Another type of comparison is a graphic (does not matter if boxplot, density trace or something else) of the same element in the same layer but in different countries. It is also feasible to make a scatterplot of two elements for the different countries. Instead of a graphic the user can make a table for the comparison. The problem is that a table contains not as much information because very soon it becomes incomprehensible. All plots presented in the previous chapters can be made to compare the different parts of the system or different countries. In this chapter only the three basic methods were presented.

```
> data(ohorizon)
> data(bhorizon)
> nam = "Al"
> M = density(log10(moss[, nam]))
> O = density(log10(ohorizon[, nam]))
> B = density(log10(bhorizon[, nam]))
> C = density(log10(chorizon[, nam]))
> plot(0, 0, xlim = c(min(M$x, O$x, B$x, C$x), max(M$x, O$x, B$x,
+     C$x)), ylim = c(min(M$y, O$y, B$y, C$y), max(M$y, O$y, B$y,
+     C$y)), main = "", xlab = paste(nam, " [mg/kg]", sep = ""),
+     ylab = "Density", xaxt = "n", cex.lab = 1.2, type = "n")
> axis(1, at = log10(a <- sort(c((10^(-50:50)) %*% t(c(2, 5, 10))))),
+     labels = a)
> lines(M, lty = 1)
> lines(O, lty = 2)
> lines(B, lty = 4)
> lines(C, lty = 5)
> text(1.5, 1.5, "Moss", pos = 4)
> text(2.5, 1.45, "O-horizon", pos = 4)
> text(4.6, 0.9, "B-horizon", pos = 4, srt = 90)
> text(3.7, 1.4, "C-horizon", pos = 4, srt = 90)
```

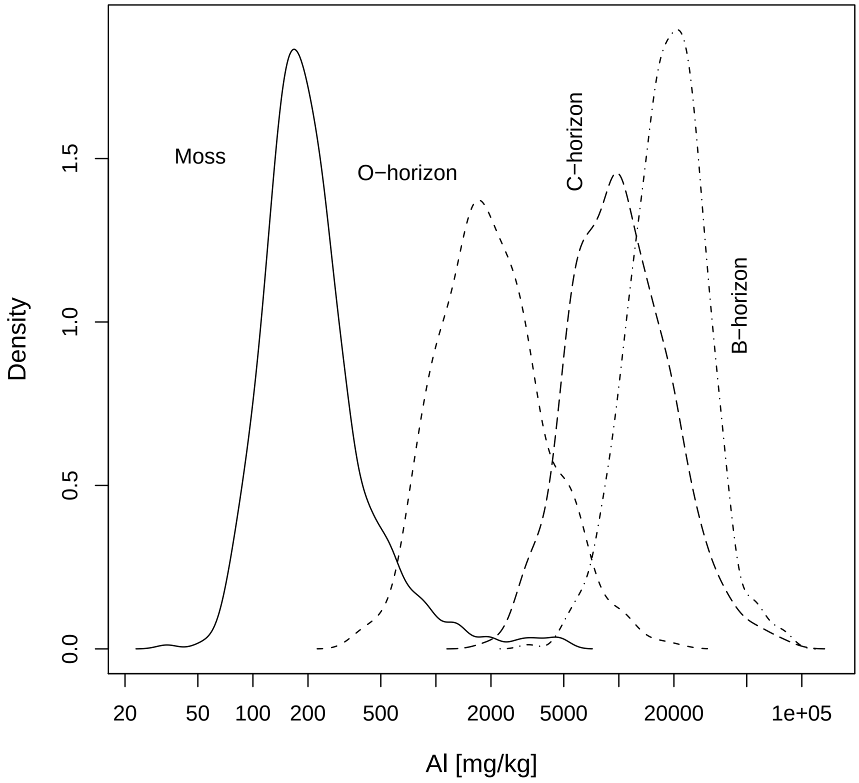


Figure 6.1: Density traces of Al in all four layers of the Kola data.

```
> nam = "Al"
> M = log10(moss[, nam])
> O = log10(ohorizon[, nam])
> B = log10(bhorizon[, nam])
> C = log10(chorizon[, nam])
> qpplot.das(M, qdist = qnorm, xlab = paste(nam, " [mg/kg]", sep = ""),
+     ylab = "Probability [%]", pch = 3, cex = 0.7, logx = TRUE,
+     logfinetick = c(2, 5, 10), logfinelab = c(2, 5, 10), line = F,
+     cex.lab = 1.2, xlim = c(min(M, O, B, C), max(M, O, B, C)))
> points(sort(O), qnorm(ppoints(length(O))), pch = 4, cex = 0.7)
> points(sort(B), qnorm(ppoints(length(B))), pch = 1, cex = 0.7)
> points(sort(C), qnorm(ppoints(length(C))), pch = 22, cex = 0.7)
> legend("topleft", legend = c("Moss", "O-horizon", "B-horizon",
+     "C-horizon"), pch = c(3, 4, 1, 22), bg = "white")
```

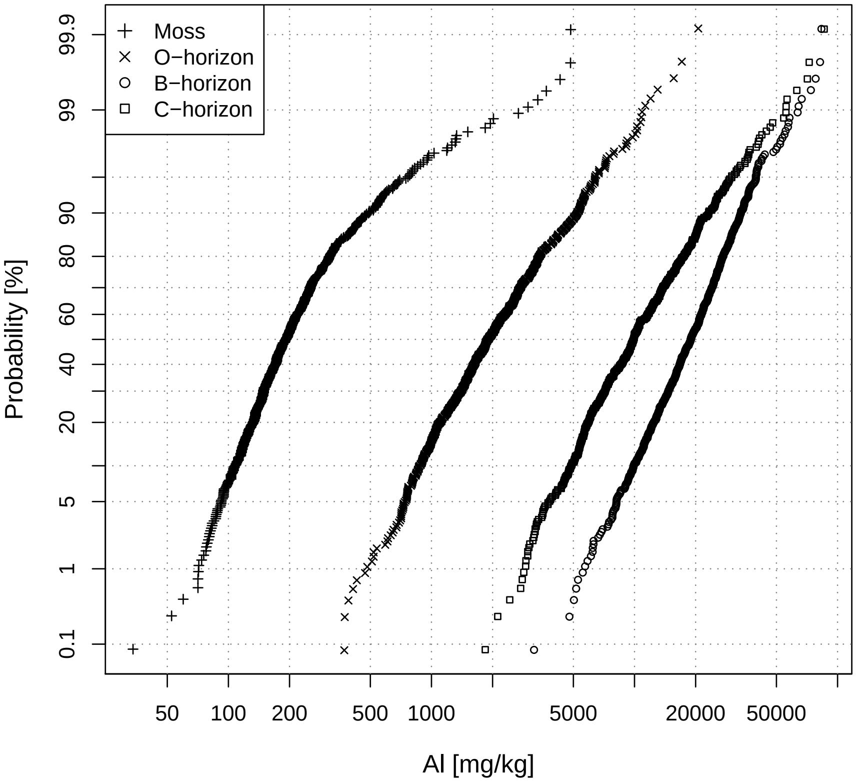


Figure 6.2: CP-plot of Al in the different sample layers.

```
> nam = "Al"
> M = moss[, nam]
> O = ohorizon[, nam]
> B = bhorizon[, nam]
> C = chorizon[, nam]
> boxplotlog(C, B, O, M, horizontal = T, xlab = paste(nam, " [mg/kg]",
+     sep = ""), cex = 0.7, cex.axis = 1.2, log = "x", cex.lab = 1.2,
+     pch = 3, names = c("C-hor", "B-hor", "O-hor", "Moss"), xaxt = "n")
> axis(1, at = (a <- sort(c((10^(-50:50)) %*% t(c(2, 5, 10))))),
+     labels = a)
```

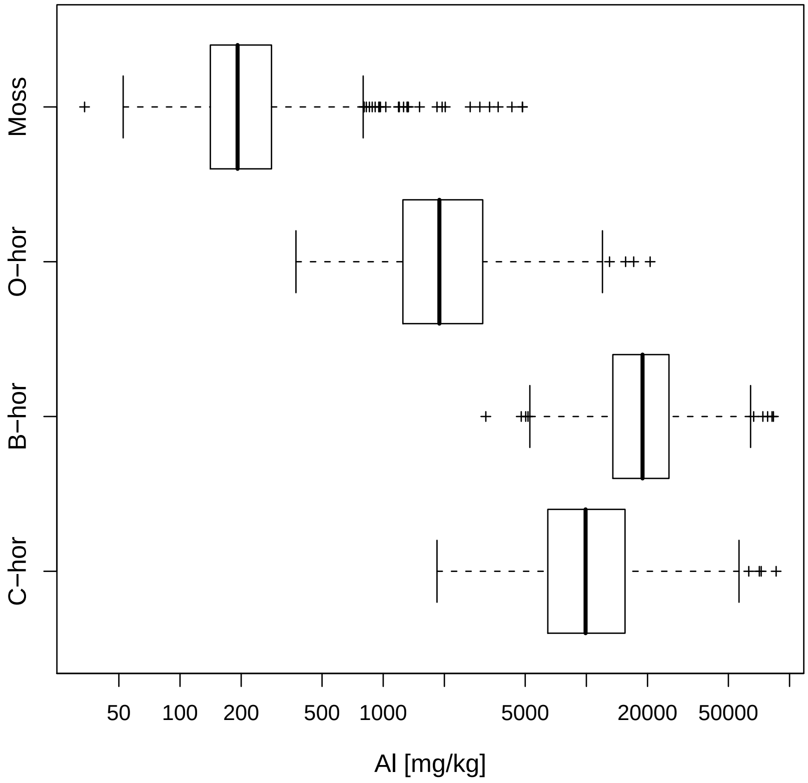


Figure 6.3: Log-boxplot (because of the problems mentioned in Chapter 2) of Al in the four different layers of the ecosystem.

Chapter 7

Correlation

Before we can talk about correlation we have to introduce the covariance and the difference between these two measures. The covariance can take any number and therefore only the sign is informative but the strength of the relation can not be interpreted. The reason is because the covariance depends on the variability of the considered variables. The correlation eliminates this dependency on the variability which means that with the correlation coefficient it is possible to say something about the strength of the relation between the variables.

The correlation results in a number between -1 and +1. ±1 shows a perfect relation and 0 means no systematic relation. There are different methods for computing the correlation. Well known are the Pearson, Spearman and Kendall methods. In this chapter we will only talk about the Pearson correlation because this is ideal when the data follow a bivariate normal distribution. The problem is that for geochemical data this is not very often reality. Therefore the data have to be log-transformed (as mentioned in Chapter 2) before calculating the Pearson correlation coefficients. The methods of Spearman and Kendall have the advantage that they do not depend on a distribution. Another advantage is that they do not look for linear relation, but if the relation is monotonically increasing or decreasing. The difference between those two methods is in the way they compute the correlation.

A more robust estimation of the correlation is the Minimum Covariance Determinant (MCD) estimator which searches for the most compact ellipsoid containing e.g. 50 percent of the data points. The user can choose any percentage between 50 and 100 percent. If the user goes near 100 percent, the robustness of the estimator is lost. Figure 7.1 demonstrates that the Pearson correlation and the MCD estimation deliver different results.

```
> x = chorizon[, c("Ca", "Cu", "Mg", "Na", "P", "Sr", "Zn")]
> par(mfrow = c(1, 1), mar = c(4, 4, 2, 0))
> R = covMcd(log10(x), cor = T)$cor
> P = cor(log10(x))
> CorCompare(R, P, labels1 = dimnames(x)[[2]], labels2 = dimnames(x)[[2]],
+     method1 = "Robust", method2 = "Pearson", ndigits = 2, cex.label = 1.2)
```

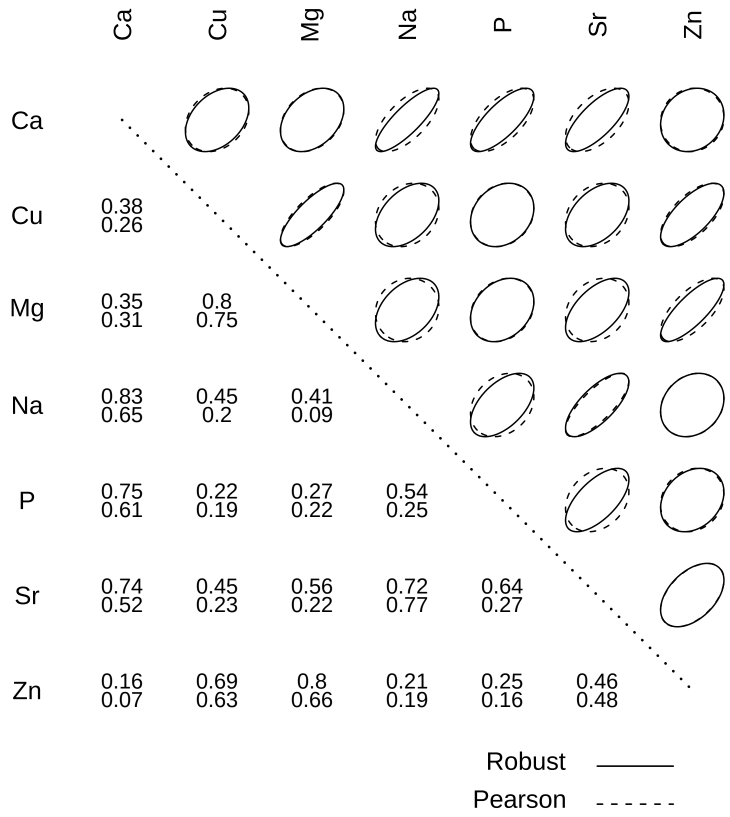


Figure 7.1: Correlation coefficients and correlation ellipses. MCD-based (upper number) and Pearson (lower number) correlation.

Chapter 8

Multivariate Graphics

Most of the graphics presented in the last chapters displays up to two variables of the data set. In a map it is only possible to visualise one variable at the time. But sometimes it is necessary to study several variables at the same time. This is the reason why multivariate graphics have been developed. In this chapter a small selection of procedures is illustrated and at the end one graphic is displayed. One possibility to represent multivariate data is in a table where each observation is identified by its sample number. The problem of a multivariate graphic is that it needs more space than a single symbol and this reduces the number of observations that can be shown in a map. It is important that the data is transformed to reduce the influence of outliers and to get a more symmetrical distribution. Then it has to be range transformed to [0,1] so that each variable has equal influence.

After all these transformations the plot can be made. There are different methods to plot multivariate graphics such as:

- Profiles: The selected variable is shown on the x-axis and the observation (log- and range-transformed) is plotted on the y-axis. A line connects the values.
- Stars: Instead of plotting the variable along the x-axis it is possible to plot it radially with equal angle. The length of the line is proportional to the value. The graphic can be plotted in three different styles: as radii, as polygons drawn around the ends of the radii and as polygon alone.
- Segments: This method is closley related to stars. Each variable is represented by a segment and the area of each segment represents the value of the sample.
- Boxes: First the variables undergo a cluster analysis. Each cluster corresponds to one side of the box. This method is only applicable when the variables can be grouped in three clusters. The dimension of each axis is proportional to the sum of the values.
- Trees: The computation is similar to boxes. The advantage is that the variables are not forced to be grouped into three clusters. The result is the same as for a dendogram from hierarchical cluster analysis. The height of the battlements or length of the branches represent the values of the variables.

```
> X = ohorizon[, "XCOO"]
> Y = ohorizon[, "YCOO"]
> el = log10(ohorizon[, c("Co", "Cu", "Ni", "Rb", "Bi", "Na", "Sr")])
> data(kola.background)
> sel <- c(3, 8, 22, 29, 32, 35, 43, 69, 73, 93, 109, 129, 130,
+     134, 168, 181, 183, 205, 211, 218, 237, 242, 276, 292, 297,
+     298, 345, 346, 352, 372, 373, 386, 408, 419, 427, 441, 446,
+     490, 516, 535, 551, 556, 558, 564, 577, 584, 601, 612, 617)
> x = el[sel, ]
> dimnames(x)[[1]] = ohorizon[sel, 1]
> xwid = diff(range(X))/120000
> ywid = diff(range(Y))/120000
> par(mfrow = c(1, 2), mar = c(1.5, 1.5, 1.5, 1.5))
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n", xlim = c(360000, max(X)))
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> tree(x, locations = cbind(X[sel], Y[sel]), len = 700, key.loc = c(793000,
+     7760000), leglen = 1500, cex = 0.75, add = T, leglh = 6,
+     lh = 30, wmax = 120, wmin = 30, labels = NULL)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
> par(mar = c(0.1, 0.1, 0.1, 0.5))
> tree(x, key.loc = c(15, 0), len = 0.022, lh = 30, leglh = 4,
+     wmax = 120, wmin = 30, leglen = 0.05, ncol = 8, cex = 0.75)
```

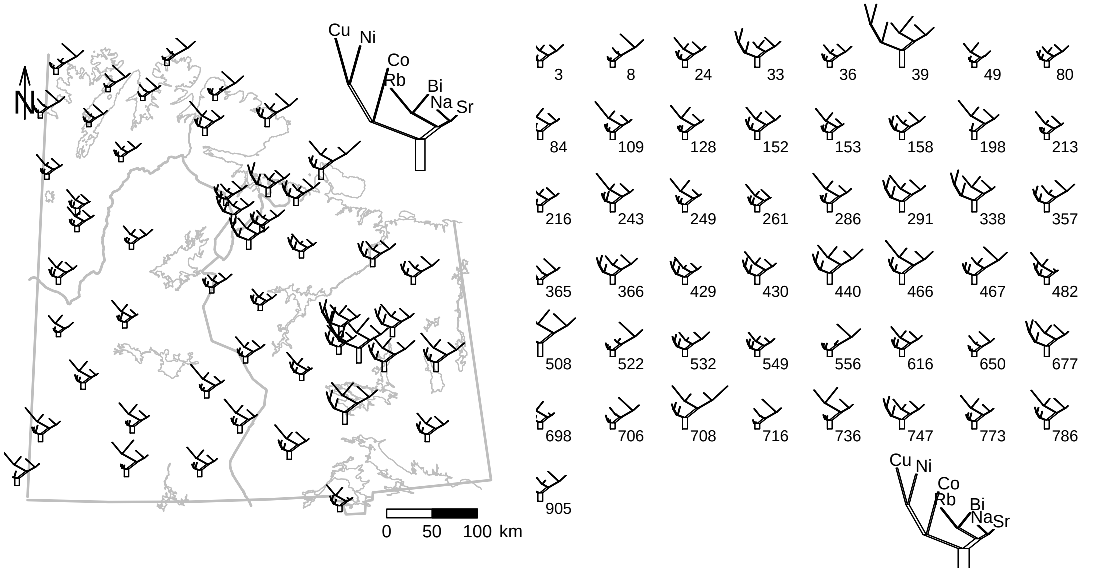


Figure 8.1: Trees constructed for 49 selected samples from Kola O-horizon soil data set.

Chapter 9

Multivariate Outlier Detection

The detection of data outliers and unusual data structures is very important in the statistical analysis of data. The basis for an outlier is the location and spread of the data. Outliers typically have a large distance to the center of the sample and therefore it is important to find a threshold, dividing the observations into background and outlier. This problem has received much attention in the literature, especialy in environmental sciences.

When we compare univariate and multivariate outlier detection we can see that there is an essential difference. In the univariate case a box (for example all observations except the lower and upper 2 percent) is defined. All values outside the box are characterized as outliers. The problem is that in this case the multivariate, often elliptical, structure of the data set is ignored. Therefore a better solution is to estimate ellipses (ellipsoids) to defined quantiles. All values outside the ellipse are outliers. Now there appears a new problem which is, how to estimate the central location and covariance. We have talked about the difference between robust and non-robust estimation in Chapter 3. Different methods result in different ellipses and therefore different characterizations.

Now it is important to find the multivariate outlier boundary. To demonstrate the problem we take seven elements of the Kola moss data and search for outliers. This results in Figure 9.1 (left) where all outliers are displayed. As one can see there are outliers near the Russian nickel industry and in northwest Norway and Finland which is one of the most pristine parts of the survey area. Therefore it is necessary to distinguish between those areas and this is the reason why it is important to take a closer look at the distribution of the robust Mahalanobis distances which are based on a robust estimation of the central value and covariance. We additionally provide a coding (grey scale) for the magnitude of the average element concentration. Now we get different types of outliers. The outliers near the Russian nickel industry are identified as "high" and the outliers in Norway and Finland are marked as low average element concentrations.

```
> X = moss[, "XCOO"]
> Y = moss[, "YCOO"]
> xwid = diff(range(X))/120000
> ywid = diff(range(Y))/120000
> par(mfrow = c(1, 2), mar = c(1.5, 1.5, 1.5, 1.5))
> el = c("Ag", "As", "Bi", "Cd", "Co", "Cu", "Ni")
> dat = log10(moss[, el])
> res <- plotmvoutlier(cbind(X, Y), dat, symb = F, map.col = c("grey",
+     "grey", "grey", "grey"), map.lwd = c(1, 1, 1, 1), xlab = "",
+     ylab = "", frame.plot = FALSE, xaxt = "n", yaxt = "n")
> legend("topright", pch = c(3, 21), pt.cex = c(0.7, 0.2), legend = c("outliers",
+     "non-outliers"), cex = 0.8)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
> res <- plotmvoutlier(cbind(X, Y), dat, symb = T, bw = T, map.col = c("grey",
+     "grey", "grey", "grey"), map.lwd = c(1, 1, 1, 1), lcex.fac = 0.6,
+     xlab = "", ylab = "", frame.plot = FALSE, xaxt = "n", yaxt = "n")
> perc.ypos = seq(from = 7740000, to = 7870000, length = 6)
> sym.ypos = perc.ypos[-6] + diff(perc.ypos)/2
> points(rep(820000, 5), sym.ypos, pch = c(1, 1, 16, 3, 3), cex = c(1.5,
+     1, 0.5, 1, 1.5) * 0.6)
> text(rep(8e+05, 6), perc.ypos[-5], round(100 * c(0, 0.25, 0.5,
+     0.75, 1), 2), pos = 2, cex = 0.7)
> text(8e+05, perc.ypos[5], "break", cex = 0.72, pos = 2)
> text(750000, 7900000, "Percentile", cex = 0.8)
> text(840000, 7900000, "Symbol", cex = 0.8)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```

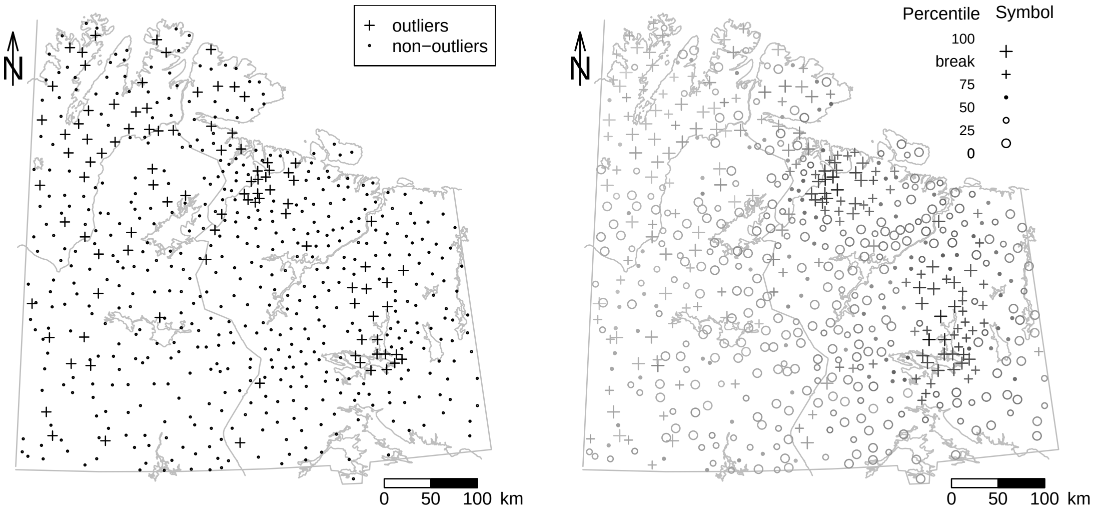


Figure 9.1: Identifying multivariate outliers in the Kola moss data. The left map shows the location of the outliers and the right one includes information on the relative distance to the multivariate data center.

Chapter 10

Principal Component Analysis and Factor Analysis

Here we will try to point out the differences between Principal Component Analysis (PCA) and Factor Analysis (FA). The main difference is that PCA focuses on maximum variance of the resulting components and FA accounts for maximum intercorrelation. The FA-model allows the common factor not to explain the total variability of the data and the existence of a unique factor which behaves completely different than the majority of the other factors. PCA always tries to accomodate the total structure of the data. Although FA is better suited to detect common factors, PCA is used more frequently in environmental sciences. Another difference is that in the case of FA the number of factors must be defined in advance. The aim of PCA and FA is dimension reduction. In applied geochemistry it is important to find e.g. two variables which describe much of the variability of the complete data set. These are the first two principal components. The problem is that the interpretation of the two components is very difficult because they are determined by a maximum variance criterion. To expand interpretability it can be useful to rotate the principal components. For more details we refer to Chapter 14 of the book.

The basis for PCA is the covariance matrix of the data set and now we have the same problem as many times before. How to estimate the covariance? If we take the classical approach the estimation can be influenced by outliers. Therefore we make a robust estimation of the covariance using the MCD estimator. Figure 10.1 displays the first two principal components based on a robust estimation of the covariance. In the graphic the influence of the outliers is reduced. When a plot of PCA with a non-robust estimation will be made it is visible that the components were influenced by the outlier.

Note that geochemical data are compositional data. The real information is in the ratios, and not in the absolute values. Especially for a multivariate analysis it is thus advisable to use an appropriate transformation. We used the isometric logratio transformation which accounts for the compositional nature of the data.

```
> par(mfrow = c(1, 1))
> X = moss[, "XCOO"]
> Y = moss[, "YCOO"]
> var = c("Ag", "Al", "As", "B", "Ba", "Bi", "Ca", "Cd", "Co",
+     "Cr", "Cu", "Fe", "Hg", "K", "Mg", "Mn", "Mo", "Na", "Ni",
+     "P", "Pb", "Rb", "S", "Sb", "Si", "Sr", "Th", "Tl", "U",
+     "V", "Zn")
> x = moss[, var]
> ilr <- function(x) {
+     x.ilr = matrix(NA, nrow = nrow(x), ncol = ncol(x) - 1)
+     for (i in 1:ncol(x.ilr)) {
+         x.ilr[, i] = sqrt((i)/(i + 1)) * log(((apply(as.matrix(x[,
+             1:i]), 1, prod))^(1/i))/(x[, i + 1]))
+     }
+     return(x.ilr)
+ }
> invilr <- function(x.ilr, x.ori) {
+     y = matrix(0, nrow = nrow(x.ilr), ncol = ncol(x.ilr) + 1)
+     for (i in 1:(ncol(y) - 1)) {
+         for (j in i:ncol(x.ilr)) {
+             y[, i] = y[, i] + x.ilr[, j]/sqrt(j * (j + 1))
+         }
+     }
+     for (i in 2:ncol(y)) {
+         y[, i] = y[, i] - sqrt((i - 1)/i) * x.ilr[, i - 1]
+     }
+     yexp = exp(y)
+     x.back = yexp/apply(yexp, 1, sum)
+     return(x.back)
+ }
> x2 = ilr(x)
> res2 = princomp(x2, cor = TRUE)
> x2.mcd = covMcd(x2, cor = T)
> res2rob = princomp(x2, covmat = x2.mcd, cor = T)
> V = matrix(0, nrow = 31, ncol = 30)
> for (i in 1:ncol(V)) {
+     V[1:i, i] <- 1/i
+     V[i + 1, i] <- (-1)
+     V[, i] <- V[, i] * sqrt(i/(i + 1))
+ }
> xgeom = 10^apply(log10(x), 1, mean)
> xlc1 = x/xgeom
> xlc = log10(xlc1)
> reslc = princomp(xlc, cor = TRUE)
> res2robback = res2rob
> res2robback$loadings <- V %*% res2rob$loadings
> res2robback$scores <- res2rob$scores %*% t(V)
> dimnames(res2robback$loadings)[[1]] <- names(x)
> res2robback$sdev <- apply(res2robback$scores, 2, mad)
> biplot(res2robback, xlab = "PC1 (robust)", ylab = "PC2 (robust)",
+     col = c(gray(0.6), 1), xlabs = rep("+", nrow(x)), cex = 0.8)
```

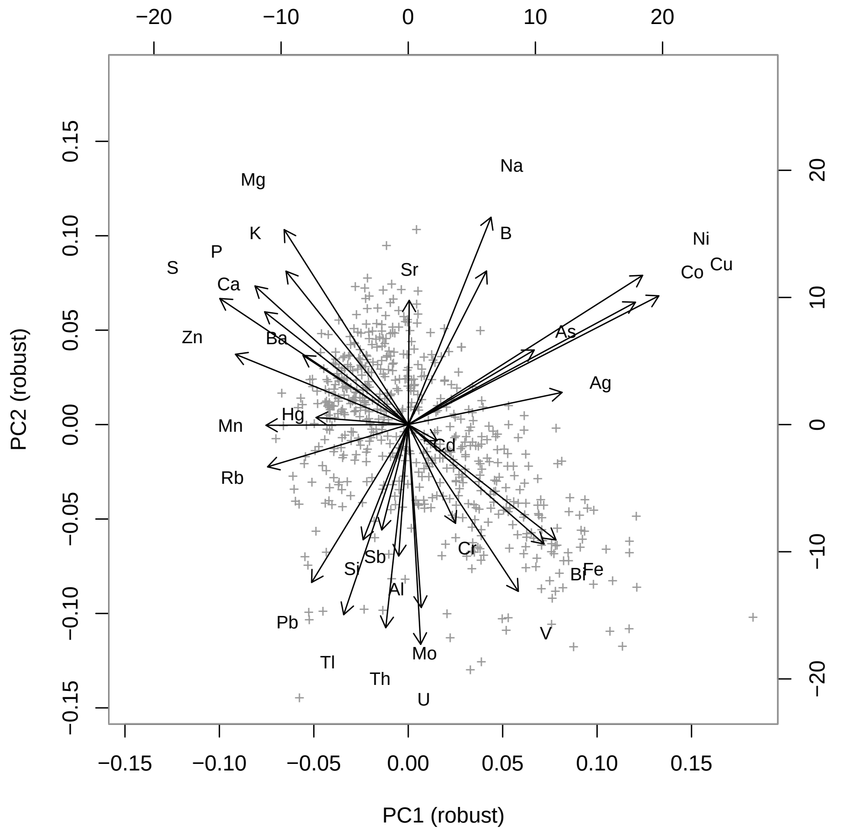


Figure 10.1: Biplot of robust PCA.

Chapter 11

Cluster Analysis

The aim of cluster analysis is to divide the data set into groups. Within one cluster the samples should be very similar and observations in different clusters should be very dissimilar. As one can imagine some problems appear. One question is the number of groups the data should be split and another one is how to define similarity. It is also a problem that different methods give different results and if one variable is added or deleted to/from the data set the result can change drastically. Many techniques are based on a distance measure and this has the great advantage that there are a priori no statistical assumptions. Modern software implementation of clustering can calculate with different distance measures but the "favourites" are the Eulidean and the Manhattan distances.

It is important to mention that clustering is no proof of a relationship between the variables or samples. In theory it can be useful to start with cluster anlaysis and then to make further statistical anaylses with the homogeneous data sets.

Important clustering techniques are:

- hierarchical methods
- partitioning methods
- model based methods
- fuzzy methods

The hierarchical method starts with a distance matrix and defines every sample as one cluster. Then the clusters will be combined stepwise. Another possibility is to go the other way round which means to start with one cluster (includes all observations) and then split this cluster stepwise. We will concentrate on the method which starts with many clusters (every sample is one cluster). In this version the number of clusters is reduced one by one because in every step two clusters are combined. At the end one single cluster is left which includes all samples. The best known methods to link two clusters are average linkage (average of all pairs of distances between the samples in the group), complete linkage (looks for the maximum distance between two clusters) and single linkage (the minimum distance between two clusters). When one of these linkage methods is computed for all cluster combinations the combination with the minimum distance is chosen. The visualization of this technique is called dendrogram.

For hierarchical methods it is not necessary to define the number of clusters in advance. When using partitioning methods the number of clusters must be pre-determind. To find out in how many groups the data set should be splitted it can be useful to start with a hierarchical cluster analysis. A very popular algorithm for partitioning is the k-means algorithm. k-means starts with k given initial cluster centroids and then the observations are assigned to the nearest centroid.

Now the centroids are recomputed and again the observations are assigned to the nearest centroid. This algorithm is iterated until convergence.

The former two methods are based on the distance measure in contrast to model based methods. The difference is that model based methods describe the shape of the possible clusters. The algorithm for this technique is Mclust which selects the cluster models (e.g. spherical or elliptical). Then the algorithm determines the cluster membership for all samples over a range of different numbers of clusters. Finally the combination of model and number of groups with the highest BIC (Bayesian Information Criterion) can be chosen.

The last techniques are the fuzzy methods. In this case the observations are not allocated to one group. Instead, every observation has a probability to be assigned to each cluster and this probability can be represented in different grey-scales. A popular fuzzy clustering algorithm is the fuzzy c-means (FCM) algorithm. For this algorithm the number of clusters has to be determined by the user. Then the cluster centroids and the membership coefficients are calculated. Figure 11.1 shows the results of fuzzy clustering for Cu, Mg, Ni, and Sr of the moss data in 4 different cluster.

```
> el = c("Cu", "Ni", "Mg", "Sr")
> x = scale(log10(moss[, el]))
> set.seed(100)
> res = cmeans(x, 4)
> X = moss[, "XCOO"]
> Y = moss[, "YCOO"]
> xwid = diff(range(X))/120000
> ywid = diff(range(Y))/120000
> par(mfrow = c(2, 2), mar = c(1.5, 1.5, 1.5, 1.5))
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> points(moss[, "XCOO"], moss[, "YCOO"], col = gray(1 - res$mem[,
+     1]), pch = 15, cex = 1)
> text(752000, 7880000, "Cluster 1", cex = 1.1)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> points(moss[, "XCOO"], moss[, "YCOO"], col = gray(1 - res$mem[,
+     2]), pch = 15, cex = 1)
> text(752000, 7880000, "Cluster 2", cex = 1.1)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> points(moss[, "XCOO"], moss[, "YCOO"], col = gray(1 - res$mem[,
+     3]), pch = 15, cex = 1)
```

```
> text(752000, 7880000, "Cluster 3", cex = 1.1)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> points(moss[, "XCOO"], moss[, "YCOO"], col = gray(1 - res$mem[,
+     4]), pch = 15, cex = 1)
> text(752000, 7880000, "Cluster 4", cex = 1.1)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```

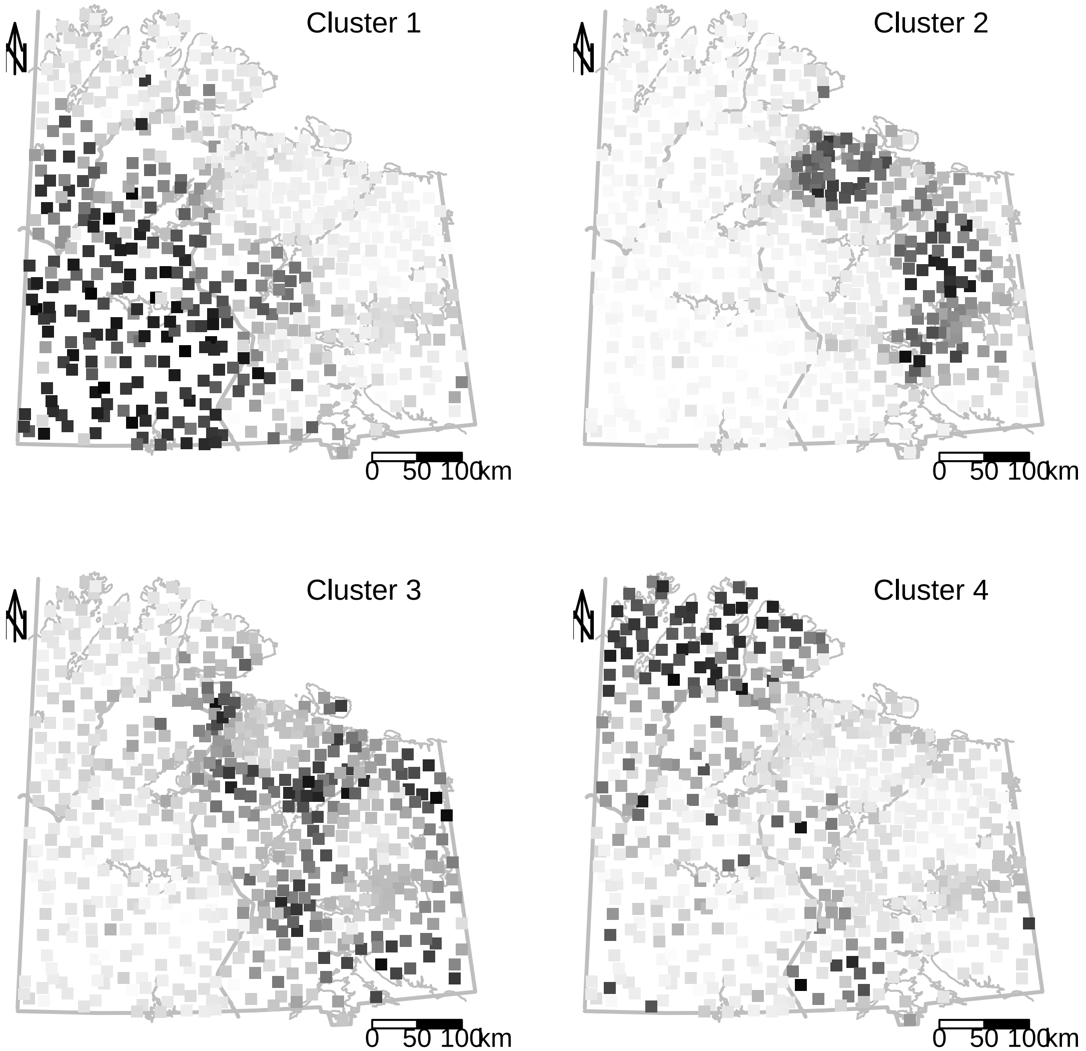


Figure 11.1: Result for FCM into four cluster for Cu, Mg, Ni and Sr in the moss data (log-transformed and standardised).

Chapter 12

Regression Analysis

For regression analysis dependent and independent variables are used. The dependent variable y should be predicted using the information of the independent variables. Regression analysis is very important when it is expensive to analyse one element, or if the quality of the element is very poor. Now it could be useful to predict this element from all the other variables. In practise it is necessary to know some values of the x and y variable to fit any model. Often only a linear model is fitted and the chosen criterion is the least squares (LS) criterion. Least squares means that the sum of the squared errors (=residuals) should be minimised. The resulting parameters define the orientation of the line or curve (when a quadratic model is computed). A robust counterpart to the LS criterion is least trimmed sum of squares (LTS) which downweights the largest residuals. Regression analysis has some requirements to the data input.

- Homogeneity: The variance of the dependent variables across the range of the data should be similar.
- Normality: The residuals should be normally distributed.
- Data outliers, extreme values: The presence of data outliers can influence the result because they can "lever" the regression line.
- Correlation: The base for regression analysis is that the dependent and independent variables are correlated. For environmental studies it can happen that when the data is spatially correlated a relationship is over-represented.
- Dimensionality: It is important that the number of observations is larger than the number of independent variables. If this is not the case, other methods have to be employed.

We mentioned the problem of outliers in a regression model because they can influence the regression line or hyperplane. Therefore it is interesting to find a robust method for estimating the regression coefficients. As mentioned above, a possibility is to use least trimmed sum of squares (LTS) which downweights the larger residuals. After fitting an LTS model we can make some checks if the assumptions are fulfilled. With a plot, where the x-axis represents the robust Mahalanobis distances and the y-axis the values of the residuals, it is possible to identify outliers.

```
> attach(chorizon)
> X = chorizon[, "XCOO"]
> Y = chorizon[, "YCOO"]
> xdat = log10(chorizon[, 101:109]/chorizon[, 110])
> set.seed(101)
> xdat.mcd = covMcd(xdat, cor = T)
> md = sqrt(xdat.mcd$mah)
> crit = sqrt(qchisq(0.975, ncol(xdat)))
> set.seed(104)
> res = ltsReg(log10(Be) ~ Al_XRF + Ca_XRF + Fe_XRF + K_XRF + Mg_XRF +
+     Mn_XRF + Na_XRF + P_XRF + Si_XRF, data = xdat)
> resid = res$residuals/res$scale
> xwid = diff(range(X))/120000
> ywid = diff(range(Y))/120000
> par(mfrow = c(2, 2), mar = c(4, 4, 2, 2))
> psymb = res$lts.wt
> psymb[res$lts.wt == 1] <- 1
> psymb[res$lts.wt == 0] <- 3
> pcex = res$lts.wt
> pcex[res$lts.wt == 1] <- 1.3
> pcex[res$lts.wt == 0] <- 0.8
> qqnorm(resid, xlab = "Quantiles of standard normal distribution",
+     ylab = "Standardised LTS residuals", pch = psymb, cex = pcex,
+     main = "", cex.lab = 1.2)
> qqline(resid)
> plot(res$fitted, resid, cex.lab = 1.2, xlab = "Fitted values",
+     ylab = "Standardised LTS residuals", type = "n")
> points(res$fitted[res$lts.wt == 0], resid[res$lts.wt == 0], cex = 0.8,
+     pch = 3)
> points(res$fitted[res$lts.wt == 1], resid[res$lts.wt == 1], cex = 0.8,
+     pch = 1)
> abline(h = 0, col = "grey", lty = 2)
> abline(h = c(-2.5, 2.5), lty = 3, cex = 1.1)
> symb.nor = 16
> symb.resl = 1
> symb.resh = 22
> symb.goodl = 3
> symb.badll = 1
> symb.badlh = 15
> plot(md, resid, cex = 0.5, pch = 3, type = "n", xlab = "Robust Mahalanobis distances",
+     ylab = "Standardised LTS residuals", cex.lab = 1.2)
> abline(h = c(2.5, -2.5))
> abline(v = crit)
> md.resid = as.data.frame(cbind(md, resid))
> points(md.resid[md < crit & abs(resid) < 2.5, ], cex = 0.3, pch = symb.nor)
> points(md.resid[md < crit & resid >= 2.5, ], cex = 0.9, pch = symb.resh)
> points(md.resid[md < crit & resid <= (-2.5), ], cex = 0.9, pch = symb.resl)
> points(md.resid[md >= crit & abs(resid) < 2.5, ], cex = 0.6,
+     pch = symb.goodl)
> points(md.resid[md >= crit & resid >= 2.5, ], cex = 0.9, pch = symb.badlh)
> points(md.resid[md >= crit & resid <= (-2.5), ], cex = 1.1, pch = symb.badll)
> par(mar = c(1.5, 1.5, 1.5, 1.5))
> XY = cbind(X, Y)
> plot(X, Y, frame.plot = FALSE, xaxt = "n", yaxt = "n", xlab = "",
+     ylab = "", type = "n")
> plotbg(map.col = c("gray", "gray", "gray", "gray"), add.plot = T)
> points(XY[md < crit & abs(resid) < 2.5, ], cex = 0.3, pch = symb.nor)
> points(XY[md < crit & resid >= 2.5, ], cex = 0.9, pch = symb.resh)
> points(XY[md < crit & resid <= (-2.5), ], cex = 0.9, pch = symb.resl)
> points(XY[md >= crit & abs(resid) < 2.5, ], cex = 0.6, pch = symb.goodl)
> points(XY[md >= crit & resid >= 2.5, ], cex = 0.9, pch = symb.badlh)
> points(XY[md >= crit & resid <= (-2.5), ], cex = 1.1, pch = symb.badll)
> legend("topright", pch = c(symb.nor, symb.resh, symb.resl, symb.goodl,
+     symb.badlh, symb.badll), pt.cex = c(0.3, 0.9, 0.9, 0.6, 0.9,
+     1.1), legend = c("Normal observations", "High vertical outliers",
+     "Low vertical outliers", "Good leverage points", "High bad leverage points",
+     "Low bad leverage points"), cex = 0.7)
> scalebar(761309, 7373050, 861309, 7363050, shifttext = -0.5,
+     shiftkm = 37000, sizetext = 0.8)
> Northarrow(362602, 7818750, 362602, 7878750, 362602, 7838750,
+     Alength = 0.15, Aangle = 15, Alwd = 1.3, Tcex = 1.6)
```

In Figure 12.1 (lower left) we can see a plot of standardised LTS residuals versus Mahalanobis distances. The samples can be divided into six groups were all observation with low absolute residuals and Mahalanobis distances have low leverage. Only a little leverage on a classical regression model comes from the samples with high absolut residuals and low Mahalanobis distances. Interessting are those variables which have a high Mahalanobis distance because they exert leverage on the model. Those with low absolut residual are good leverage points because they stabilize the regression line or hyperplane.

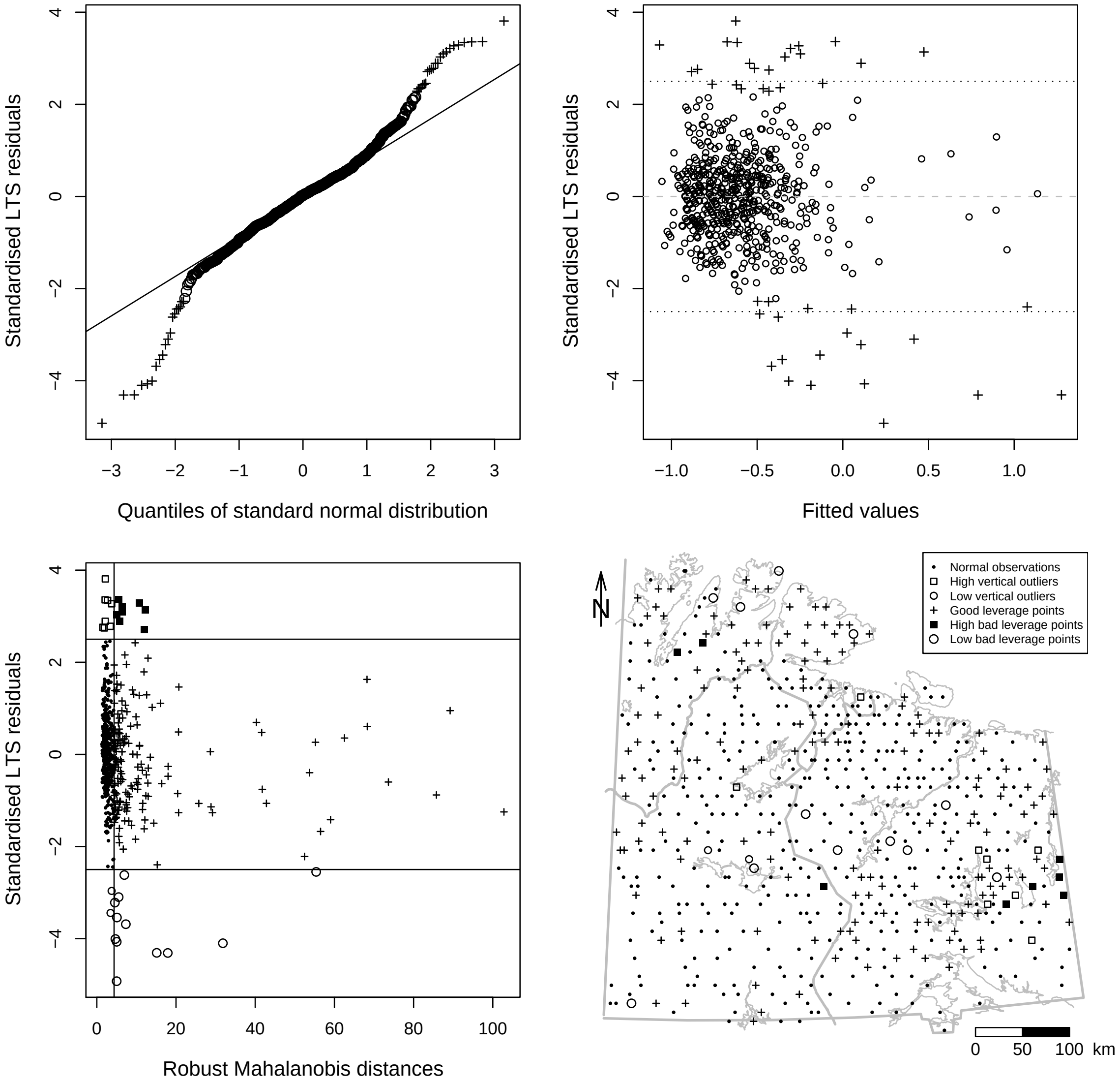


Figure 12.1: Robust regression diagnostics for the Be variable. QQ-plot (upper left), LTS residuals versus fitted (upper right), standardised LTS residuals versus Mahalanobis distances (lower left), map indicating the location and characterization of the outliers (lower right)