Applied geostatistics
Exercise: Compositional variables

D G Rossiter
University of Twente, Faculty of Geo-Information Science & Earth Observation (ITC)

November 17, 2014

Contents

知不知上，不知知，病。是以圣人不病。以其病病，是以不病
"To know that one does not know is best; not to know but to believe one knows is a disease. Sages are free of this disease because they recognize it for what it is."
– 德道经 (DeDaoJing 71)

After completing this exercise you should be able to:

1. Transform a composition into additive-log-ratio form;
2. Analyze the spatial structure of a composition;
3. Interpolate the composition by Ordinary Kriging and Ordinary Cokriging.

1 Compositional variables

Certain (geo)statistical variables, when considered as a group, are not independent in feature space, because they are constrained to sum to some constant; the set of these is called a **composition**. Well-known examples include particle-size distribution of the fine earth (<2 mm diameter) mineral soil (e.g., sand, silt and clay)[1], and the basic cations (e.g. K^+, Na^+, Ca^{++}, Mg^{++}) in a total element analysis of a rock sample. If these are expressed as a percent, they must sum to 100%; if as a proportion, to 1.

> **Note:** Note that the constituents of a composition can often be expressed as absolute values, e.g., the actual weight of a particle-size separate in a sample (say, 10 g clay in a 20 g soil sample). The composition can be derived from these values by dividing the absolute value by the sum of the values, but the reverse is not possible unless the total is known, which is rarely the case. Indeed, the main interest of most compositions is usually in the relative amounts.

Because a composition is constrained, any random variate composition that appears to be of a given dimension in $\mathbb{R}_+^D$-space is in fact drawn from the **simplex** $\mathscr{S}^d$, where $d = D - 1$, embedded in this space[2]. A well-known example is the triangular **ternary diagram**, showing three particle-size separates (sand, silt, clay), which are apparently three variables and thus define a point in $\mathbb{R}_+^3$-space on a two-dimensional graph, i.e., a simplex $\mathscr{S}^2$. Knowing two of the separates, the third is determined – this is why we can visualize three variables in two dimensions in a ternary diagram, where any two axes determines the third.

> **Note:** The `soiltexture` R package, part of the "Soil Texture Wizard" project[3] from Julien Moeys, plots a wide variety of ternary diagrams representing soil particle-size composition. Figure 1 shows an example.

This invalidates the assumption of independence between the n members of a composition; in fact, there must be spurious negative correlations between

[1] these may be further subdivided, e.g., into very fine, fine, medium, coarse and very coarse sand
[2] The notation is from Aitchison [2]
[3] http://soiltexture.r-forge.r-project.org/

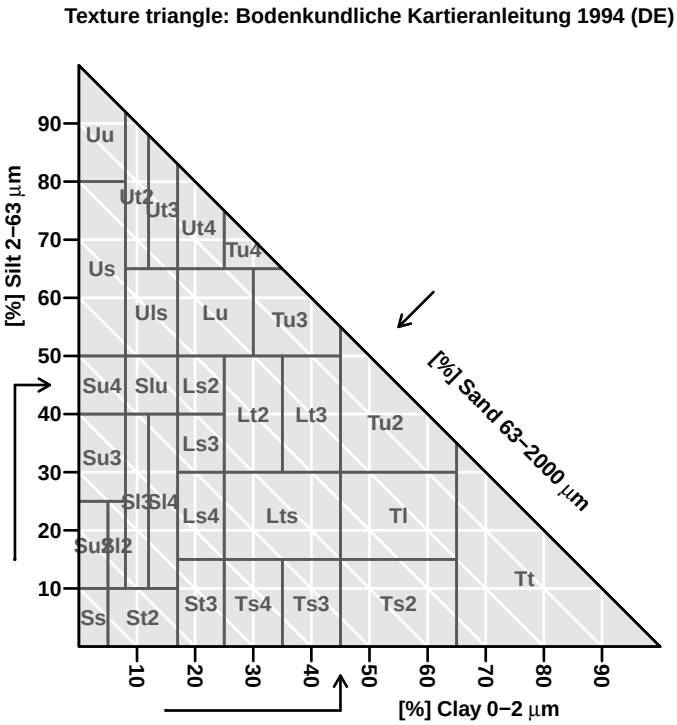


Figure 1: Example of a ternary diagram produced by the `soiltexture` package: German soil texture classification

the variables, as first pointed out by Pearson in 1897. Thus conventional multivariate analysis techniques (e.g., partial correlation matrices, multivariate regression) fail when applied to compositional variables.

These issues are comprehensively dealt with by Aitchison [2][4]; further, a R package `compositions` written by van den Boogaart and Tolosana-Delgado [9] is available to implement a proper analysis; these authors have also written a textbook in the UseR! series [10].

From a geostatistical perspective there is another issue: if the n members are modelled and interpolated separately (e.g., by kriging) in general they will not sum properly; see for example Odeh et al. [7]. A naive solution is to predict $n-1$ components and derive the remaining one by subtraction from the constant, but the result depends on which components are selected – a troubling inconsistency.

Further, it may be expected that the spatial structure of the elements of a composition may be similar, allowing a co-kriging approach to increase precision of the kriged estimate.

Pawlowsky-Glahn and Olea [8] devoted an entire book to the topic of geostatistical analysis of compositional data; this work has been refined by Lark and Bishop [5].

[4] an updated version of Aitchison [1]

2 An example of compositional data

We begin by illustrating some of the issues with compositional data.

Task 1 : If you do not already have the required package `compositions` on your system, install it. •

The `install.packages` function retrieves a named package and any dependencies. You must first have specified a **mirror**, i.e., the site from which you will download packages; do this with the `chooseCRANmirror` interactive function.

```
> chooseCRANmirror()
> install.packages("compositions")
```

Task 2 : Load the `compositions` package. •

The `require` function loads a package:

```
> require(compositions)
```

> **Note:** The help text for this package includes this interesting disclaimer: "The mere fact that the package computes something does not imply that this is reasonable." Words of wisdom indeed, that can applied to any computation!

We use a dataset from a transect near Sandford-on-Thames (Oxfordshire, England), originally reported by Webster and Cuanalo [12], and also used by Davis [3, Example 4.12]. The soil was sampled at three depth intervals (5–6 cm thick, centred on 8, 30, 65 cm depth) by augering every 10 m along a regular transect, resulting in 321 sites.

Task 3 : Load the example dataset and examine its structure. •

This dataset was provided by Richard Webster to Murray Lark[5], who formatted it for easier processing. He in turn kindly provided it to us as file `sandford.txt`.

Task 4 : Examine the text file structure in a text editor. •

```
> file.show("sandford.txt")
```

The first few lines are:

```
    Top        2        3
    Clay    Silt    Clay    Silt    Clay    Silt
1    70    15    85    10    84    14
2    65    20    75    15    85    10
3    65    20    75    10    70    20
```

[5] http://www.bgs.ac.uk/staff/profiles/40081.html

The meaning here is clear: each row (after the headers) is an observation number (in sequence along the transect), followed by the silt and clay in layers 1, 2 and 3 respectively. It's easiest to read this in without the header lines, and add our own column labels.

Task 5 : Read the text file into a data frame and assign field names. •

The `read.table` is the generic function to read text files; in this case the only non-default setting is the `skip` argument to skip the first two lines.

We name the data frame (unimaginatively) `ds`:

```
> ds <- read.table("sandford.txt", skip = 2)
```

To illustrate composite variables, we add variables for sand content, taken to be the complement of clay + silt.

> **Note:** In soils lab. practice, it is common to adjust the separates to sum to 100% before publishing the results. This can be by only analysing two separates and assuming the other to be the complement, or by adjusting all separates.

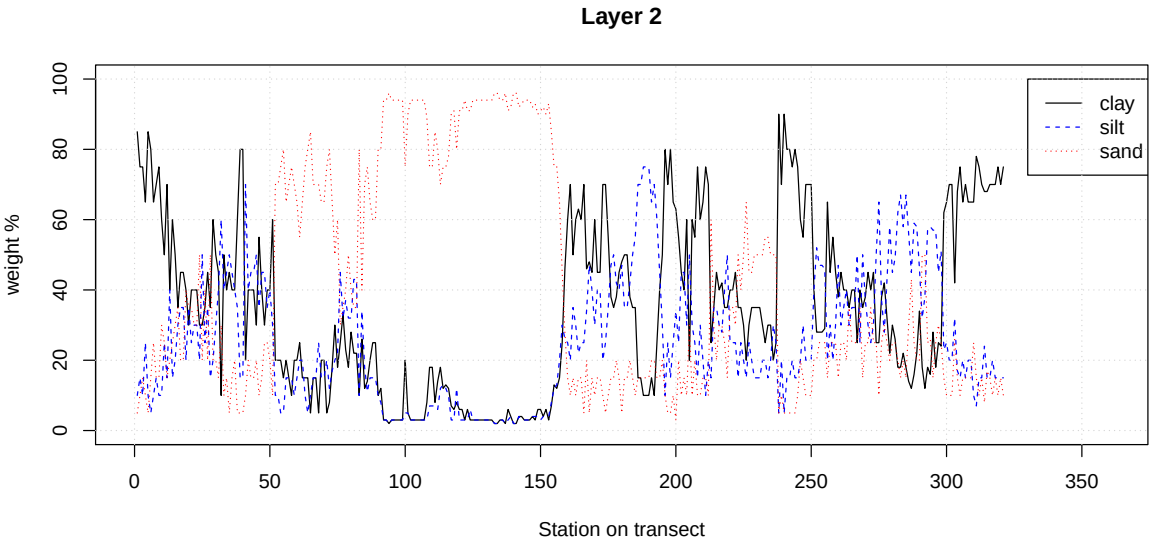
Task 6 : Add fields for the sand content of the three layers. •

```
> ds$sand1 <- 100 - (ds$clay1 + ds$silt1)
> ds$sand2 <- 100 - (ds$clay2 + ds$silt2)
> ds$sand3 <- 100 - (ds$clay3 + ds$silt3)
```

2.1 Visualizing the transect

Task 7 : Display the particle-size fractions along the transect for the second layer. •

```
> plot(ds$clay2, type = "l", ylim = c(0, 100), xlim = c(0,
+     360), main = "Layer 2", xlab = "Station on transect",
+     ylab = "weight %")
> lines(ds$silt2, lty = 2, col = "blue")
> lines(ds$sand2, lty = 3, col = "red")
> legend(330, 100, c("clay", "silt", "sand"), lty = 1:3,
+     col = c("black", "blue", "red"))
> grid()
```



This transect is quite variable and shows some clear boundaries, but also smooth transition zones.

Q1 : *Describe the pattern of spatial dependence in the sand-size fraction.* *Jump to A1* •

2.2 Splitting the dataset

Following Lark and Bishop [5], we remove every third datum (i.e., the observations at sites 3, 6, 9, . . .) into a separate evaluation dataset. This will be used to compare interpolation methods.

Task 8 : Remove every third observation from the modelling set and move it into a evaluation set. •

We first create a vector of the indices to remove with the `seq` function, then use these indices to select; note the use of the `-` subscript operator to exclude rows. Finally we check the dimensions of the resulting data frames.

```
> valid.ix <- seq(from = 3, to = length(ds$seq), by = 3)
> head(valid.ix)
```

```
[1]  3  6  9 12 15 18

> tail(valid.ix)

[1] 306 309 312 315 318 321

> ds.val <- ds[valid.ix, ]
> ds.cal <- ds[-valid.ix, ]
> dim(ds.val)

[1] 107  10

> dim(ds.cal)

[1] 214  10
```

2.3 Target composition

Following Lark and Bishop [5] we will model the particle size fractions for the depth interval centred on 30 cm, i.e., layer 2.

Task 9 : Create a list with the field names of the particle-size composition of layer 2 and show the summary of these fields in the calibration set. •

Since the `summary` method does not include the standard deviation, we use the `sapply` function to apply the `sd` function over the columns:

```
> names.2 <- c("sand2", "silt2", "clay2")
> summary(ds.cal[, names.2])

     sand2          silt2        clay2
 Min.   : 3.0   Min.   : 2   Min.   : 2.0
 1st Qu.:15.0   1st Qu.:10   1st Qu.:15.0
 Median :25.0   Median :20   Median :35.0
 Mean   :40.2   Mean   :25   Mean   :34.8
 3rd Qu.:70.0   3rd Qu.:35   3rd Qu.:53.8
 Max.   :96.0   Max.   :75   Max.   :90.0

> sapply(ds.cal[, names.2], sd)

 sand2  silt2  clay2
31.108 17.685 24.040
```

Q2 : *Describe the feature-space variability of this layer.* *Jump to A2* •

2.4 Choosing a composition geometry

There are four kinds of compositions, depending on the answers to two questions [9, Table 1]:

1. Is the total sum (also called the **size**) of the composition meaningful, or is it a side effect (artefact) of the measurement procedure?
2. Is the scale of distance between compositions absolute or relative?

These dictate which **geometry** is appropriate for the analysis.

For question (1), we can measure particle-size fractions as weights of sand, silt and clay in a given soil volume; these can be added to give the size of the sample in absolute terms. If we are interested in the absolute amount of some pollutant that could be absorbed onto clay particles, we would want to know absolute amount of that fraction. It is more common to measure these weights in the lab. based on a soil weight; then the absolute weights of the fractions are not meaningful, only their proportion of the weight[6].

The answer to this question determines whether the data are considered **open**, i.e., all positive but not constrained to sum to a constant) or **closed**, i.e., must sum to a constant.

For question (2), we need to decide if the measurement scale is intrinsically linear. Does an increase in clay content from 2% to 5% have the same significance as an increase from 32% to 35% and from from 92% to 95%? For most interpretations they are not the same; at low clay contents a few percent increase has much more effect on soil behaviour than at high clay contents, so we prefer a relative scale.

The answer to this question determines whether the "distance" between observations (in feature space) are computed by their absolute difference or as a quotient; this second case requires a logarithmic transformation, so data must all be strictly positive (no zeroes)[7].

The `compositions` package defines classes and functions for these four situations:

`rplus` : for meaningful size and absolute scale;

`rcomp` : for meaningful size and relative scale;

`aplus` : for proportions and absolute scale; and

`acomp` : for proportions and relative scale.

Q3 : *Which geometry is appropriate for particle-size fractions?* *Jump to A3* •

Task 10 : Convert the particle-size fractions of the second layer into an additive log-ratio composition. •

This is class `acomp`, also known the Aitchison log-ratio class.

Package `compositions` provides a function `acomp` for creating a composition of class `acomp` from a list of variables in a dataframe.

[6] To answer questions about absolute amounts in a given volume of soil, the fraction is converted to an absolute amount by the soil bulk density

[7] In practice, these can be avoided by replacing zeroes with some small number, conventionally half the minimum value

Note: The `acomp` function does not change any numbers, it just lets other operations know how to interpret them.

```
> comp.2 <- acomp(ds.cal[, names.2], total = 100)
> class(comp.2)

[1] "acomp"

> str(comp.2)

 acomp [1:214, 1:3] 5 10 10 5 25 15 30 25 20 25 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:214] "1" "2" "4" "5" ...
  ..$ : chr [1:3] "sand2" "silt2" "clay2"
```

The second step is to convert this composition of class `acomp`, which is just a wrapper for the original variables, into a compositional variable of class `rmult`, composed of ratios rather than original variables. Before doing this, we explain the mathematics,

The **additive log-ratio transformation** has the form:

$$\mathbf{x} = \left[\log \frac{z_1}{z_D}, \log \frac{z_2}{z_D}, \ldots, \log \frac{z_{D-1}}{z_D} \right]^T \tag{1}$$

where z_i is the i^{th} element of the vector $\mathbf{z}$ of the D variables for one observation:

$$\mathbf{z} = [z_1, z_2, \ldots, z_D]^T \tag{2}$$

Note that the transformed vector $\mathbf{x}$ has one less dimension than the original vector $\mathbf{x}$, and the first $(D-1)$ variables are standardized by the final one. Note also that the log-ratio transformation maps from the simplex (where the values must be on $0 \ldots 100$) to the entire real line – this is a major benefit, because it allows us to work in real-space geometry, e.g., with normal multivariate analysis.

The choice of standardising variable is mathematically arbitrary, and is usually dictated by the application. In this case we will use the last-named (clay proportion) as the normalising variable.

Task 11 : Transform the composition by the additive log-ratio transformation. •

The `alr` function builds a composition by this transformation:

```
> str(alr2 <- alr(comp.2))

 rmult [1:214, 1:2] -2.833 -2.015 -1.872 -2.833 -0.956 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:214] "1" "2" "4" "5" ...
  ..$ : chr [1:2] "sand2" "silt2"

> summary(alr2)
```

```
          sand2  silt2
Min.    -3.040 -2.890
1st Qu. -1.180 -0.847
Median  -0.134 -0.168
Mean     0.181 -0.288
3rd Qu.  1.430  0.118
Max.     3.870  2.010
attr(,"class")
[1] "summary.rmult" "matrix"
```

The inverse transformation is:

$$z_i = \frac{e^{y_i}}{\sum_{i=1}^{D-1} e^{y_j}} \tag{3}$$

Task 12 : Check that the inverse transformation recovers the original data. •

The `alrInv` command inverts the transformation. It can not recover the name of the third variable unless the optional `orig` argument names an object which structure should be mimicked. Note also the back-transformation is to proportions, since there is no way to recover the original size from the log-ratios. We check the difference, allowing for small rounding errors.

```
> str(tmp <- alrInv(alr2, orig = comp.2))

 acomp [1:214, 1:3] 0.05 0.1 0.1 0.05 0.25 0.15 0.3 0.25 0.2 0.25 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:214] "1" "2" "4" "5" ...
  ..$ : chr [1:3] "sand2" "silt2" "clay2"

> sum((comp.2[, "sand2"] - tmp[, "sand2"] * 100) + (comp.2[,
+     "silt2"] - tmp[, "silt2"] * 100) + (comp.2[, "clay2"] -
+     tmp[, "clay2"] * 100))

[1] -4.8228e-13
```

Apart from small rounding errors the reverse transformation is exact.

2.5 Exploring the composition

We now look at the composition in feature space, both numerically and graphically.

2.5.1 Numerical

Task 13 : Summarize the composition in the transformed feature space. •

In R we generally use the `summary` method to summarize an object. In the case of compositions, the summaries must be presented for each component or pair-wise ratio of these in the selected geometry. So each aspect of the usual summary (e.g., minimum, maximum, mean, quantiles, variances) is

presented as an vector or matrix of log-ratios. We can see the various aspects with the `names` function and then extract the results we want with the `$` fields selection operator. There are also convenience methods such as the familiar `mean`, which specialise according to the geometry of the object.

```
> names(summary(comp.2))

 [1] "mean"         "mean.ratio"   "variation"    "expsd"
 [5] "invexpsd"     "min"          "q1"           "med"
 [9] "q3"           "max"          "missingness"
```

First, the means. For class `acomp` these are geometric means, rather than arithmetic means. This is because the geometry is relative. Thus there is a difference between the naïve means of the proportions as presented in the dataset (computed with `mean` on the original data, or the `meanCol` method on the composition object), and the mean of the compositional variable (computed with `meanCol`, for means of the columns of a dataframe, on the compositional variable):

```
> summary(comp.2)$mean

  sand2   silt2   clay2
0.40657 0.25423 0.33920
attr(,"class")
[1] acomp

> colMeans(ds.cal[, names.2])/100

  sand2   silt2   clay2
0.40164 0.25037 0.34799

> meanCol(comp.2)/100

  sand2   silt2   clay2
0.40164 0.25037 0.34799
```

For this dataset, since the means are close to the centre of the simplex, there is not much difference.

To summarize the variances, we use a so-called **variation matrix**. This is not the variance of any component, rather it is variances of all possible log-ratios among components. This is reported by the `variation` method applied to a composition; it is also the `var` field of the summary.

> **Note:** The standard variance-covariance matrix computed with `var` on the composition is not so meaningful.

```
> variation(comp.2)

       sand2   silt2   clay2
sand2 0.0000 2.70469 3.39560
silt2 2.7047 0.00000 0.72289
clay2 3.3956 0.72289 0.00000

> summary(comp.2)$var
```

```
        sand2   silt2   clay2
sand2 0.0000 2.70469 3.39560
silt2 2.7047 0.00000 0.72289
clay2 3.3956 0.72289 0.00000
```

Q4 : *Which two components have the least variation in their log ratio? What is the interpretation?* *Jump to A4* •

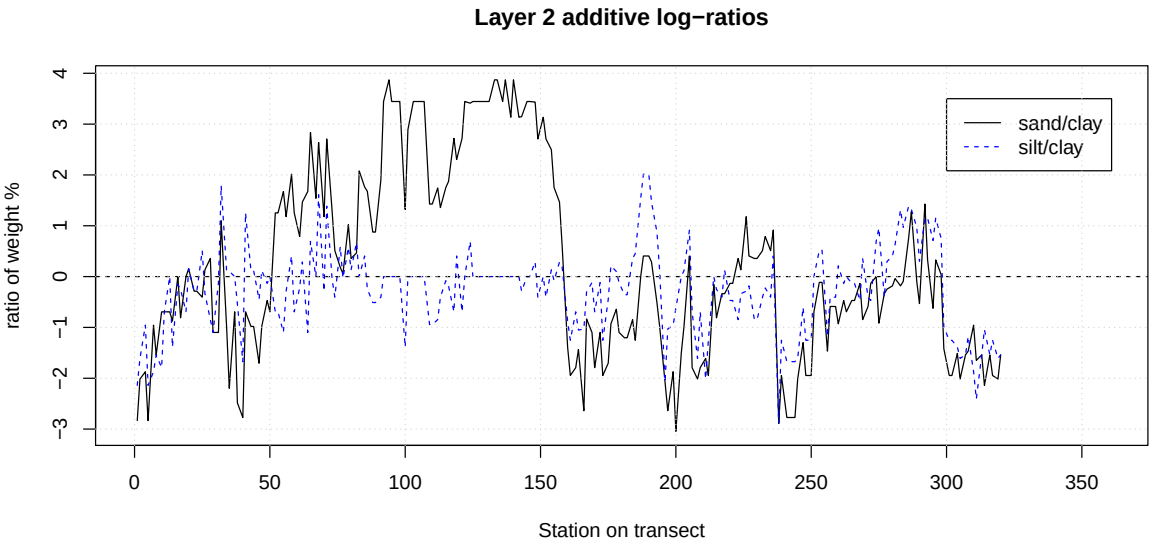
2.5.2 Graphical

In §2.1 we visualized the proportions along the transect; we can also visualize the additive log-ratios.

Task 14 : Visualize the log-ratios along the transect •

> **Note:** This code makes clever use of the `setdiff` function to set the x-axis points to $1, 2, 4, 5, 7, 8 \ldots$ by removing every third station number, since the composition is only computed for 2/3 of the points along the transect, yet we want to see the correct station.

```
> xpts <- setdiff(1:length(ds$seq), seq(3, length(ds$seq),
+     by = 3))
> plot(alr2[, "sand2"] ~ xpts, type = "l", xlim = c(0,
+     360), main = "Layer 2 additive log-ratios", xlab = "Station on transect",
+     ylab = "ratio of weight %")
> lines(alr2[, "silt2"] ~ xpts, lty = 2, col = "blue")
> abline(h = 0, lty = 2)
> legend(300, 3.5, c("sand/clay", "silt/clay"), lty = 1:2,
+     col = c("black", "blue"))
> grid()
```

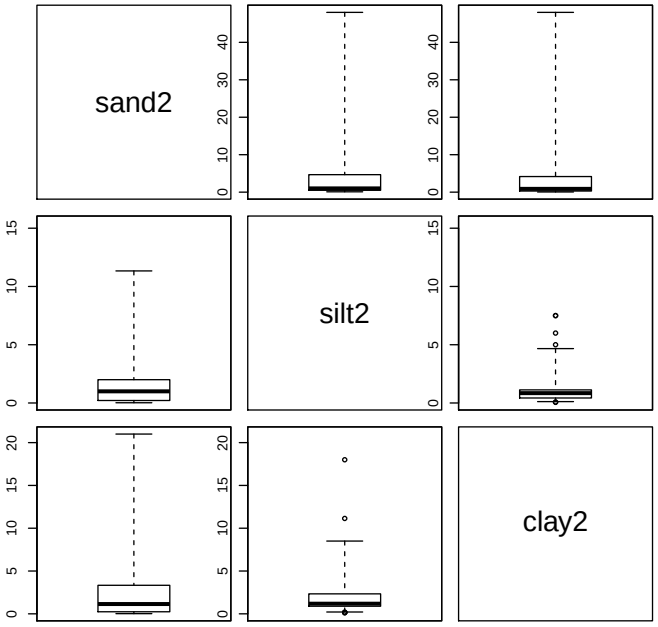


Q5 : *Describe the spatial variability of the ratios, compared to the untransformed particle-size fractions.* *Jump to A5*

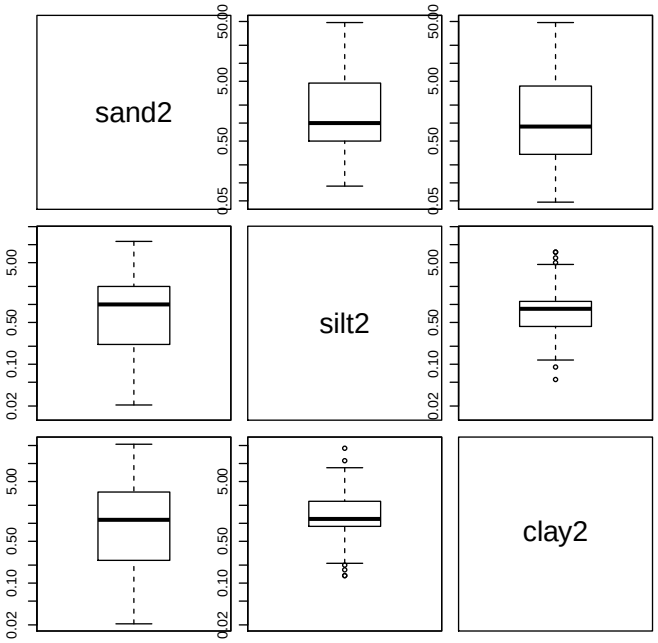
•

Another way to visualize the variability of a composition is by a boxplot of all the ratios; the generic `boxplot` method is adapted appropriately for compositions. This can be visualized untransformed or on a log-scale:

```
> boxplot(comp.2, log = F)
```




```
> boxplot(comp.2, log = T)
```



The boxplots will have different scales, depending on the value range, so you must look at the numbers to answer the following question.

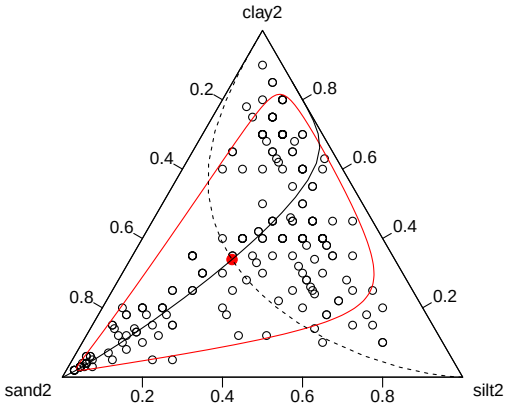
A **ternary diagram** is the familiar representation of a 3-component composition in a 2-D simplex.

Task 15 : Display a ternary diagram of the layer 2 particle-size distribution. •

The `plot` method specializes to `plot.acomp` when called to plot an object of class `acomp`. In the case of a composition that can be projected on a 2-D simplex, the plot is a ternary diagram.

We also plot the mean ("barycentre"); note this is at coordinates which we discovered with the `mean` method.

```
> plot(comp.2, axes = T)
> plot(mean(comp.2), add = T, col = "red", pch = 20, cex = 2)
> straight(rcomp(c(0, 0, 0)), rcomp(mean(comp.2)))
> ellipses(mean(comp.2), var(comp.2), col = "red", r = 2)
> straight(mean(comp.2), princomp(comp.2)$Loadings[1, ])
> straight(mean(comp.2), princomp(comp.2)$Loadings[2, ],
+     lty = 2)
```



The red curve on the ternary graph, drawn with the `ellipses` function, shows the variance of the dataset around the mean, represented by the large red dot.

The black curves on the ternary graph, drawn with the `straight` function, are the two principal component axes of the three variables (note that the system is rank deficient, since it is a composition); the solid line is component 1 and the dashed line component 2. These are straight lines in the simplex but display as curves on the ternary diagram.

Q6 : *Are the observations equally distributed in this simplex? If not, where*

are they concentrated? *Jump to A6* •

Q7 : *How does this diagram prove that the three variables are* not *independent in feature-space?* *Jump to A7*
•

3 Kriging compositional variables

In this section we will examine how to analyze the spatial structure of a composition, and use this for kriging interpolation. We will compare several approaches:

1. Naive analysis: each component of the composition is modelled and kriged independently (§3.2);
2. Co-kriging of two components, estimation of the third by subtraction (§3.3);
3. Kriging of compositional variables (§3.4);
4. Co-kriging of compositional variables (§3.5).

We will compare all these by validating against the one-third of the known observations that were held out for evaluation (§2.2); in §4 we will compare them against each other.

3.1 Spatial objects

To use the `gstat` package for spatial analysis, we must first create a spatial object acceptable to the `sp` package.

Task 16 : Convert the calibration dataset into a spatial object. •

We use the `sp` package for spatially-explicit data. This requires at least two dimensions, so we assume a constant value (0) for the second dimension, and use the sequence number as the first dimension.

The `coordinates` method assigns coordinates, in this case the sequence number along the transect and a constant; note that the actual length of the transect is ten times this.

```
> require(sp)
> require(gstat)
> ds.cal.sp <- cbind(ds.cal, y = 0)
> coordinates(ds.cal.sp) <- ~seq + y
> str(ds.cal.sp)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        214 obs. of  9 variables:
  .. ..$ clay1: int [1:214] 70 65 70 80 70 65 40 45 25 30 ...
  .. ..$ silt1: int [1:214] 15 20 10 10 10 15 40 25 40 50 ...
  .. ..$ clay2: int [1:214] 85 75 65 85 65 70 60 50 40 60 ...
```

```
.. ..$ silt2: int [1:214] 10 15 25 10 10 15 10 25 40 15 ...
.. ..$ clay3: num [1:214] 84 85 65 80 25 80 25 35 50 10 ...
.. ..$ silt3: num [1:214] 14 10 10 15 15 10 25 25 30 80 ...
.. ..$ sand1: num [1:214] 15 15 20 10 20 20 20 30 35 20 ...
.. ..$ sand2: num [1:214] 5 10 10 5 25 15 30 25 20 25 ...
.. ..$ sand3: num [1:214] 2 5 25 5 60 10 50 40 20 10 ...
..@ coords.nrs : int [1:2] 1 11
..@ coords     : num [1:214, 1:2] 1 2 4 5 7 8 10 11 13 14 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "seq" "y"
..@ bbox       : num [1:2, 1:2] 1 0 320 0
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "seq" "y"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr NA
```

3.2 Ordinary kriging of particle-size fractions

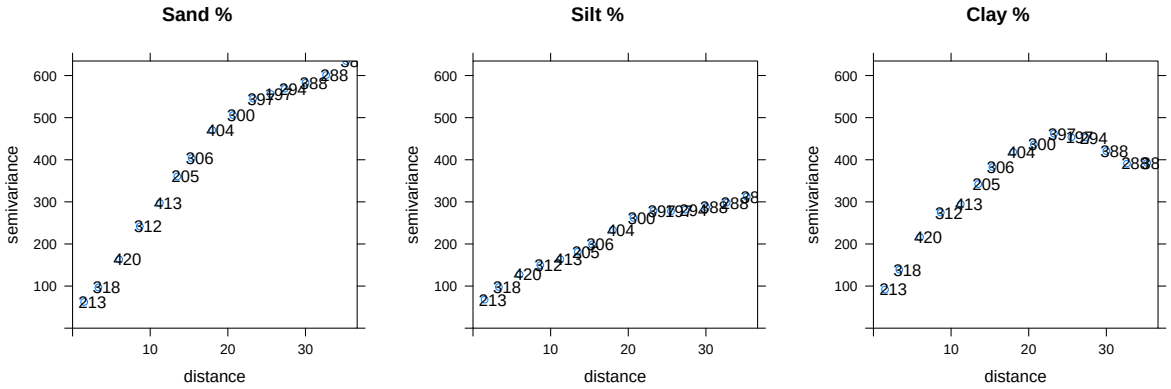
The naïve way to produce maps of the particle-size fractions is to model and interpolate each one separately. We will see the difficulties this causes.

Task 17 : Compute variograms for the layer 2 particle-size fractions. Plot them on the same scale. •

We are only interested in the short-range structure, since kriging interpolation weights will be insignificant at longer separations. So we limit the range to 30 units (each representing 10 m).

```
> v.sand <- variogram(sand2 ~ 1, loc = ds.cal.sp, cutoff = 36)
> v.silt <- variogram(silt2 ~ 1, loc = ds.cal.sp, cutoff = 36)
> v.clay <- variogram(clay2 ~ 1, loc = ds.cal.sp, cutoff = 36)
> sv.max <- max(v.sand$gamma, v.silt$gamma, v.clay$gamma)
```

```
> p1 <- plot(v.sand, plot.numbers = T, main = "Sand %",
+     ylim = c(0, sv.max))
> p2 <- plot(v.silt, plot.numbers = T, main = "Silt %",
+     ylim = c(0, sv.max))
> p3 <- plot(v.clay, plot.numbers = T, main = "Clay %",
+     ylim = c(0, sv.max))
> print(p1, split = c(1, 1, 3, 1), more = T)
> print(p2, split = c(2, 1, 3, 1), more = T)
> print(p3, split = c(3, 1, 3, 1), more = F)
```



Q8 : *Describe the spatial structure (range, sill, nugget, form) of the three separates. Are these similar?* *Jump to A8* •

Task 18 : Model each variogram separately. •

We start with visual estimates specified with the `vgm` function, and then fit with the `fit.variogram` function, using the default weighted least squares (WLS) fit:

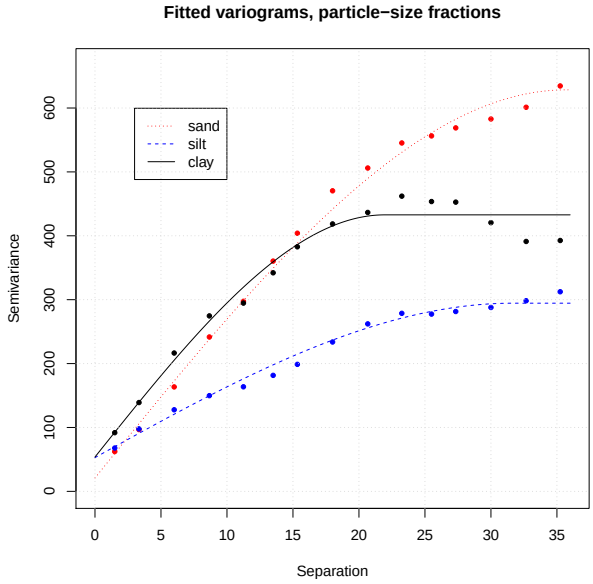
```
> (vm.sand <- fit.variogram(v.sand, vgm(1400, "Sph", 100,
+     100)))

  model  psill range
1   Nug  21.02  0.00
2   Sph 607.38 35.68

> (vm.silt <- fit.variogram(v.silt, vgm(300, "Sph", 50,
+     100)))

  model  psill  range
1   Nug  52.84  0.000
2   Sph 241.69 31.666

> (vm.clay <- fit.variogram(v.clay, vgm(425, "Sph", 40,
+     100)))

  model   psill  range
```

```
1   Nug  53.767  0.000
2   Sph 379.004 21.951
```

Task 19 : Plot the empirical variograms and the fitted models on one graph. •

We first compute a common scale from the maximum semivariances, then use the `plot` method to draw the first scatterplot, followed by `lines` and `points` functions to build up the final figure.

```
> sv.max <- max(v.sand$gamma, v.silt$gamma, v.clay$gamma) *
+     1.05
> plot(v.sand$gamma ~ v.sand$dist, xlim = c(0, 36), ylim = c(0,
+     sv.max), ylab = "Semivariance", xlab = "Separation",
+     main = "Fitted variograms, particle-size fractions",
+     col = "red", pch = 20)
> lines(variogramLine(vm.sand, maxdist = 36), lty = 3,
+     col = "red")
> points(v.silt$gamma ~ v.silt$dist, ylim = c(0, sv.max),
+     col = "blue", pch = 20)
> lines(variogramLine(vm.silt, maxdist = 36), lty = 2,
+     col = "blue")
> points(v.clay$gamma ~ v.clay$dist, ylim = c(0, sv.max),
+     col = "black", pch = 20)
> lines(variogramLine(vm.clay, maxdist = 36), lty = 1,
+     col = "black")
> legend(3, sv.max * 0.9, c("sand", "silt", "clay"), lty = 3:1,
+     col = c("red", "blue", "black"))
> grid()
```



Q9 : *How similar are the spatial structures? How do you explain the different ranges, structural sills, and nuggets, in terms of physical processes?*

Jump to A9 •

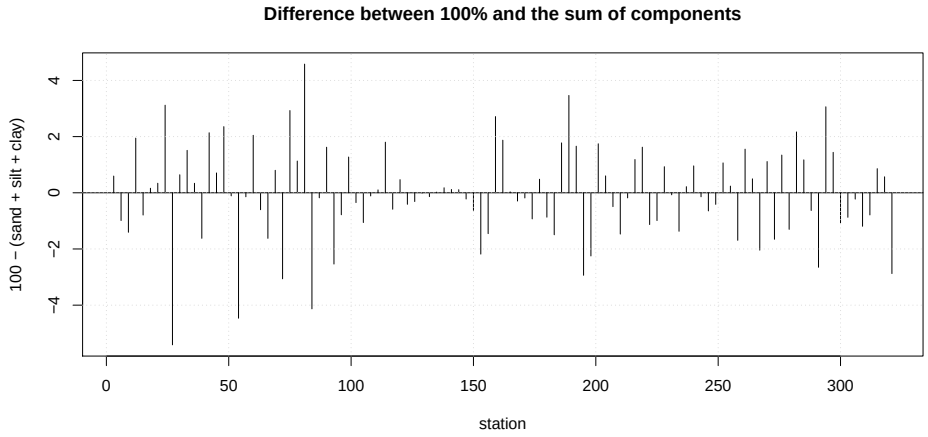
Task 20 : Predict at the evaluation locations. •

First we must make a spatial version of the evaluation set. Because there are many points, it is both computationally-efficient and justified by theory of local dependence to limit the maximum distance of points to use to compute weights to the range of each variable.

```
> ds.val.sp <- cbind(ds.val, y = 0)
> coordinates(ds.val.sp) <- ~seq + y
> k.sand <- krige(sand2 ~ 1, loc = ds.cal.sp, newdata = ds.val.sp,
+     model = vm.sand, maxdist = vm.sand[2, "range"])

[using ordinary kriging]

> k.silt <- krige(silt2 ~ 1, loc = ds.cal.sp, newdata = ds.val.sp,
+     model = vm.silt, maxdist = vm.silt[2, "range"])

[using ordinary kriging]

> k.clay <- krige(clay2 ~ 1, loc = ds.cal.sp, newdata = ds.val.sp,
+     model = vm.clay, maxdist = vm.clay[2, "range"])

[using ordinary kriging]
```

Task 21 : Sum the three predictions and compare to the required 100%. •

```
> summary(diff <- 100 -
+         (k.sand$var1.pred + k.silt$var1.pred + k.clay$var1.pred))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-5.4100 -0.9590 -0.1100 -0.0261  1.0900  4.5800

> plot(diff ~ coordinates(ds.val.sp)[,1],
+      ylab="100 - (sand + silt + clay)",
+      xlab="station", type="h",
+      main="Difference between 100% and the sum of components")
> abline(h=0)
> grid()
```



Q10 : *Do the separately-predicted values sum to 100%? Characterise the error. Is there any spatial pattern?* *Jump to A10* •

Clearly, this is not satisfactory.

3.2.1 Evaluation of the OK of particle-size fractions approach

---

Task 22 : Compute the bias and RMSE of the actual vs. predicted values of the three fractions. •

An easy way to compute the bias is to use the `summary` method; for a vector this reports the mean as part of the summary. We can also see the extremes, the IQR, and the median.

```
> summary(diff.k.sand <- (k.sand$var1.pred - ds.val.sp$sand2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-40.500  -4.360  -0.529  -0.738   2.750  35.500

> (rmse.k.sand <- sqrt(sum(diff.k.sand^2)/length(diff.k.sand)))

[1] 9.2043

> summary(diff.k.silt <- (k.silt$var1.pred - ds.val.sp$silt2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-20.800  -5.160   0.802   0.248   5.360  23.000

> (rmse.k.silt <- sqrt(sum(diff.k.silt^2)/length(diff.k.silt)))

[1] 7.6755

> summary(diff.k.clay <- (k.clay$var1.pred - ds.val.sp$clay2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-29.900  -3.640   1.030   0.516   4.620  29.500

> (rmse.k.clay <- sqrt(sum(diff.k.clay^2)/length(diff.k.clay)))
```

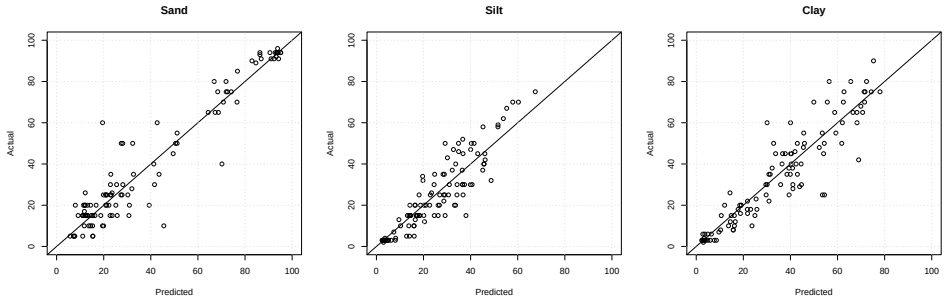
```
[1] 9.1903
```

Q11 : *Describe the biases and RMSE. In general, how is the quality of this prediction?* *Jump to A11* •

Task 23 : Display 1:1 plots of the actual vs. predicted for the three elements of the evaluation composition for the separate fractions OK approach. •

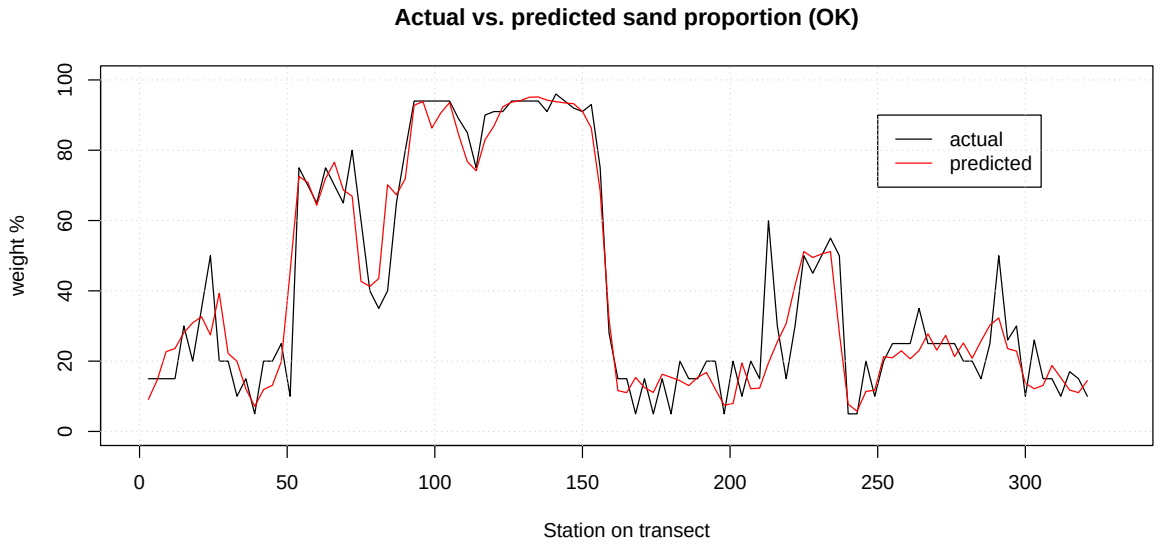
These are correctly displayed against a 1:1 line (i.e., predicted equals actual).

```
> par(mfrow=c(1,3))
> plot(ds.val[,"sand2"] ~ k.sand$var1.pred,
+      ylab="Actual", xlab="Predicted", main="Sand",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"silt2"] ~ k.silt$var1.pred,
+      ylab="Actual", xlab="Predicted", main="Silt",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"clay2"] ~ k.clay$var1.pred,
+      ylab="Actual", xlab="Predicted", main="Clay",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> par(mfrow=c(1,1))
```



Task 24 : Show the actual vs. predicted for sand along the transect for the evaluation points. •

```
> plot(ds.val$sand2 ~ ds.val$seq, type = "l", ylim = c(0,
+     100), xlim = c(0, 330), main = "Actual vs. predicted sand proportion (OK)",
+     xlab = "Station on transect", ylab = "weight %",
+     cex = 0.6, pch = 20)
> lines(k.sand$var1.pred ~ coordinates(ds.val.sp)[, 1],
+     col = "red", type = "l", cex = 0.6)
> legend(250, 90, c("actual", "predicted"), lty = 1, col = c("black",
+     "red"))
> grid()
```



Q12 : *Where are the largest errors? Describe the overall effect of kriging interpolation.* *Jump to A12* •

3.2.2 Recreating a composition

The evaluation of the previous section were of each particle-size fraction separately; each was predicted with its own variogram and so may be "optimal" if only that fraction is needed. But if we want to produce kriging predictions of all three fractions, the individual results can not be used, since they do not sum to 100%, i.e., the total of the composition.

Task 25 : Create a composition from the three independent OK predictions. •

Since we know the total, we can create a compositional variable with the `acomp` function; this automatically adjusts each prediction to its proportion of 100%. We illustrate this with the first few sand contents:

```
> tmp <- data.frame(sand2 = k.sand$var1.pred, silt2 = k.silt$var1.pred,
+     clay2 = k.clay$var1.pred)
```

```
> comp.2.ssc.ok <- acomp(tmp, total = 100)
> head(tmp[, "sand2"])

[1]  9.1355 14.6561 22.7077 23.5829 28.1034 30.8962

> head(comp.2.ssc.ok[, "sand2"])

[1]  9.1902 14.5127 22.3930 24.0509 27.8826 30.9464

> rm(tmp)
```

Note how some sand contents have increased and some decreased by the normalization; this is because of their relative proportions of the different totals.

The components of this composition can now be evaluated and compared to the next three compositional approaches, in which the total composition is correctly constrained.

Task 26 : Re-compute the evaluation statistics. •

```
> summary(diff.k.sand <- (comp.2.ssc.ok[, "sand2"] - ds.val.sp$sand2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-40.500  -4.510  -0.602  -0.816   2.570  35.500

> (rmse.k.sand <- sqrt(sum(diff.k.sand^2)/length(diff.k.sand)))

[1] 9.1282

> summary(diff.k.silt <- (comp.2.ssc.ok[, "silt2"] - ds.val.sp$silt2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-20.300  -4.610   0.789   0.318   5.240  22.400

> (rmse.k.silt <- sqrt(sum(diff.k.silt^2)/length(diff.k.silt)))

[1] 7.5632

> summary(diff.k.clay <- (comp.2.ssc.ok[, "clay2"] - ds.val.sp$clay2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-29.900  -3.720   0.991   0.498   4.600  29.400

> (rmse.k.clay <- sqrt(sum(diff.k.clay^2)/length(diff.k.clay)))

[1] 9.2956
```

Q13 : *How do the evaluation statistics for the composition compare to those for each component separately?* *Jump to A13* •

3.3 Co-krige independent variables

Another approach is to co-krige the variables and predict them together. This should improve precision if (1) there is good correlation among the variables both at single points and spatially, (2) the spatial structures of the variables are similar.

It is not possible to model all three together, since they are linearly dependent. So we have to pick two, model and predict with these, and obtain the third separate by subtraction. This has the obvious problem that the the final result depends on which two are selected.

Since in the additive log-ratio we selected clay as the normalising variable (§2.4), we will omit clay here and select sand and silt as the two fractions to co-krige.

For cokriging there are no simple `gstat` functions; we have to use the general `gstat` method.

Task 27 : Build a `gstat` structure to represent the variables to model. •

The first time an object of class `gstat` is defined with the `gstat` method, we must give it a name as the left-hand side of the assignment and give the name `NULL` as the first argument to the `gstat` method. In subsequent calls to this method we give the same name but also announce that we're updating an existing object by naming the existing object as the first argument.

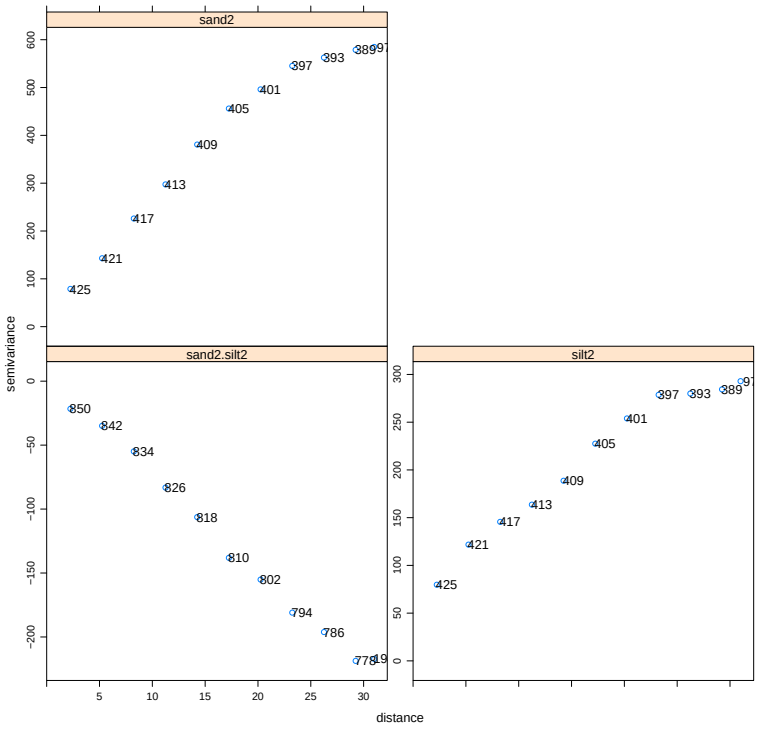
```
> (g <- gstat(NULL, id = "sand2", form = sand2 ~ 1, data = ds.cal.sp))

data:
sand2 : formula = sand2`~`1 ; data dim = 214 x 9

> (g <- gstat(g, id = "silt2", form = silt2 ~ 1, data = ds.cal.sp))

data:
sand2 : formula = sand2`~`1 ; data dim = 214 x 9
silt2 : formula = silt2`~`1 ; data dim = 214 x 9
```

Task 28 : Compute and display the direct and cross-variograms. •

As a cutoff we use the known shorter range of the direct variables, in this case, the silt proportion.

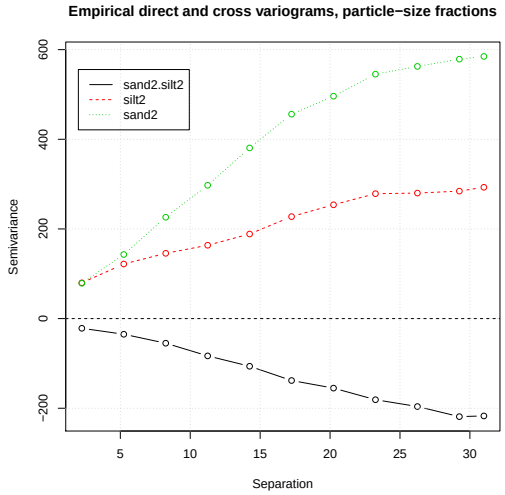
```
> vm.silt[2, "range"]

[1] 31.666

> v.cross <- variogram(g, cutoff = vm.silt[2, "range"],
+     width = 3)
> str(v.cross)

Classes 'gstatVariogram' and 'data.frame':        33 obs. of  6 variables:
 $ np     : num  850 842 834 826 818 810 802 794 786 778 ...
 $ dist   : num  2.25 5.25 8.25 11.25 14.25 ...
```

```
 $ gamma  : num  -21.6 -34.9 -54.9 -83.2 -106.3 ...
 $ dir.hor: num  0 0 0 0 0 0 0 0 0 0 ...
 $ dir.ver: num  0 0 0 0 0 0 0 0 0 0 ...
 $ id     : Factor w/ 3 levels "sand2.silt2",..: 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "direct")='data.frame':        3 obs. of  2 variables:
  ..$ id       : Factor w/ 3 levels "sand2","sand2.silt2",..: 2 3 1
  ..$ is.direct: logi  FALSE TRUE TRUE
 - attr(*, "boundaries")= num  0 3 6 9 12 15 18 21 24 27 ...
 - attr(*, "pseudo")= num 0
 - attr(*, "what")= chr "semivariance"

> print(plot(v.cross, pl = T))
```



Task 29 : Display the direct and cross-variograms on one graph. •

```
> sv.max <- max(v.cross$gamma)
> sv.min <- min(v.cross$gamma)
> plot(v.cross$gamma ~ v.cross$dist, ylim=c(sv.min,sv.max),
+   ylab="Semivariance", xlab="Separation",
+   main="Empirical direct and cross variograms, particle-size fractions",
+   type="n")
> for (i in 1:3) {
+   tmp <- subset(v.cross, as.numeric(v.cross$id) == i)
+   lines(tmp$gamma ~ tmp$dist, col=i, type="b", lty=i)
+   }
> grid()
> abline(h=0, lty=2)
> legend(2,sv.max*.95, levels(v.cross$id), lty=1:3, col=1:3)
```



Q14 : *Describe the structure of the direct and cross-variograms. How similar are they?* *Jump to A14* •

The simplest way to model a set of variograms together is the **linear model of co-regionalization** (LMC); this is limited in that the ranges must be the same, but it ensures positive-definiteness.

Task 30 : Fill the variogram models with an initial guess. •

To fit the linear model of co-regionalization, we must first establish a starting point for all the variogram models. The LMC requires a single range and structure. By filling all the frames with one model (using the `fill.all = T` argument), these conditions are automatically met.

We pick one of the direct variograms and model it by eye, adding the same model to all the others; the important point for them is the range and structure, since sills will be adjusted later.

The several variograms appear to have similar different ranges. Since we are most interested in the short-range structure, and there are plenty of points in the transect, we choose the shortest-range variogram (here, the variogram for silt) as the basis for the LMC, even though the longer-range structure of the other variograms will be ignored.

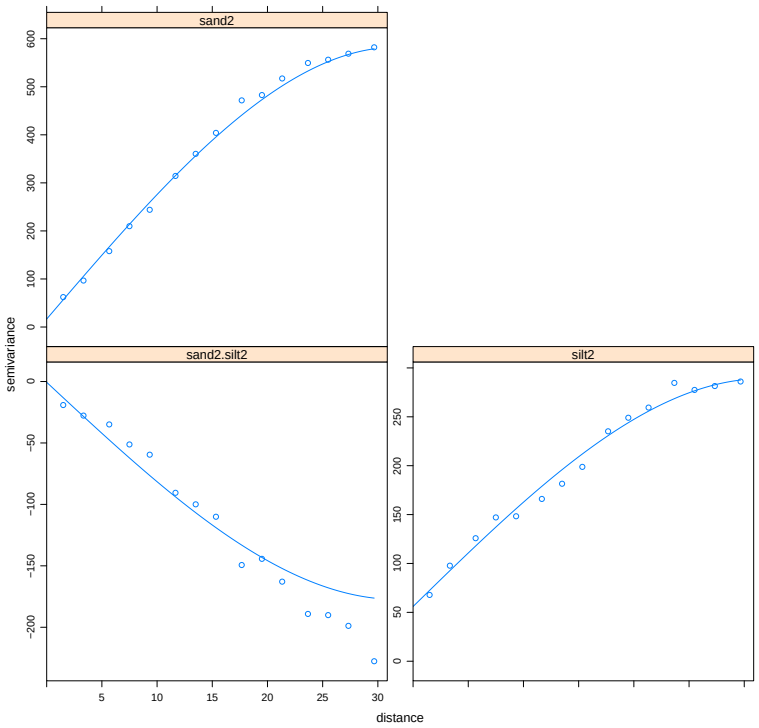
We could estimate the variogram parameters for silt by eye, but we've already fit that direct variogram, so we can use that as the starting point.

```
> vm.silt

  model  psill  range
1   Nug  52.84  0.000
2   Sph 241.69 31.666

> g <- gstat(g, id = "silt2", model = vm.silt, fill.all = T)
```

We fit all three variograms together, ensuring they lead to a positive definite co-kriging system. For this we use the `fit.lmc` method ("fit linear model of co-regionalization"). This takes the initial estimate, fits all the variograms, and then each of the partial sills is adjusted (by least squares) to the closest value that will result in a positive definite matrices.

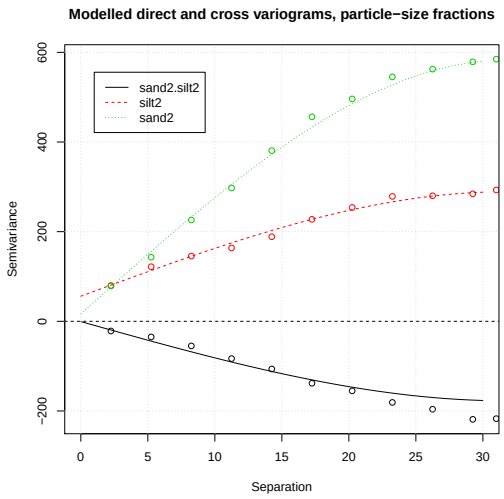
Task 31 : Fit the variograms. •

```
> (g <- fit.lmc(v.cross, g))

data:
sand2 : formula = sand2`~`1 ; data dim = 214 x 9
silt2 : formula = silt2`~`1 ; data dim = 214 x 9
variograms:
                model      psill  range
sand2[1]          Nug   16.57606  0.000
sand2[2]          Sph  565.76189 31.666
silt2[1]          Nug   55.97140  0.000
silt2[2]          Sph  232.99559 31.666
sand2.silt2[1]    Nug   -0.63118  0.000
sand2.silt2[2]    Sph -176.66398 31.666

> print(plot(variogram(g, cutoff = 30), model = g$model))
```




```
> plot(v.cross$gamma ~ v.cross$dist, xlim=c(0,30), ylim=c(sv.min,sv.max),
+      ylab="Semivariance", xlab="Separation",
+      main="Modelled direct and cross variograms, particle-size fractions",
+      type="p", col=as.numeric(v.cross$id))
> g$model
```

```
$sand2
  model   psill  range
1   Nug  16.576  0.000
2   Sph 565.762 31.666

$sand2.silt2
  model      psill  range
1   Nug   -0.63118  0.000
2   Sph -176.66398 31.666

$silt2
  model   psill  range
1   Nug  55.971  0.000
2   Sph 232.996 31.666

$silt2.NA
  model  psill  range
1   Nug  52.84  0.000
2   Sph 241.69 31.666

$silt2.silt2
  model  psill  range
1   Nug  52.84  0.000
2   Sph 241.69 31.666

> for (i in 1:3) {
+   tmp <- c("sand2.silt2","silt2","sand2")[i]
+   lines(variogramLine(g$model[[tmp]], maxdist=30), col=i, lty=i)
+   }
> grid()
> abline(h=0, lty=2)
> legend(1,sv.max*.95, levels(v.cross$id), lty=1:3, col=1:3)
```



Q15 : *How appropriate is the linear model of co-regionalization here? In other words, how valid is the assumption of same model form (here, spherical) and the same range?* *Jump to A15* •

Task 32 : Interpolate by co-kriging. •

The `predict.gstat` function is used to predict from an object of class `gstat`; if that object is properly set up with direct and cross-variograms, this will be by cokriging:

```
> k.c <- predict.gstat(g, ds.val.sp)

Linear Model of Coregionalization found. Good.
[using ordinary cokriging]

> summary(k.c)

Object of class SpatialPointsDataFrame
Coordinates:
    min max
seq   3 321
y     0   0
Is projected: NA
proj4string : [NA]
Number of points: 107
Data attributes:
   sand2.pred      sand2.var      silt2.pred      silt2.var
 Min.   : 5.79   Min.   :50.0   Min.   : 2.57   Min.   : 79.8
 1st Qu.:15.22   1st Qu.:50.0   1st Qu.:13.77   1st Qu.: 79.8
 Median :25.46   Median :50.0   Median :24.15   Median : 79.8
 Mean   :40.21   Mean   :50.3   Mean   :25.05   Mean   : 80.1
 3rd Qu.:69.49   3rd Qu.:50.0   3rd Qu.:36.01   3rd Qu.: 79.8
 Max.   :95.17   Max.   :81.0   Max.   :65.33   Max.   :100.5
 cov.sand2.silt2
 Min.   :-19.1
 1st Qu.:-10.1
 Median :-10.1
 Mean   :-10.2
 3rd Qu.:-10.1
 Max.   :-10.0
```

The two particle-size fractions, their prediction variances, the cross-variances, and the prediction covariances of these, are all predicted by the co-kriging system.

Q16 : *Look at the maxima and minima for the particle-size fractions. Are they within the required range (0 – 100%)? Is this guaranteed when kriging?* *Jump to A16* •

Task 33 : Predict clay as the complement of silt + sand. •

```
> k.c$clay2.pred <- 100 - (k.c$sand2.pred + k.c$silt2.pred)
> summary(k.c$sand2.pred)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.79   15.20   25.50   40.20   69.50   95.20
```

We hope this is also in the required range 0 . . . 100%.

3.3.1 Evaluation of the cokriging original variables approach

---

Task 34 : Compute the bias and RMSE of the actual vs. predicted values. •

```
> summary(diff.kc.sand <- (k.c$sand2.pred - ds.val.sp$sand2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-40.300  -4.590  -0.509  -0.735   3.030  35.800

> (rmse.kc.sand <- sqrt(sum(diff.kc.sand^2)/length(diff.kc.sand)))

[1] 9.2811

> summary(diff.kc.silt <- (k.c$silt2.pred - ds.val.sp$silt2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-20.800  -5.170   0.560   0.224   4.820  24.800

> (rmse.kc.silt <- sqrt(sum(diff.kc.silt^2)/length(diff.kc.silt)))

[1] 7.8753

> summary(diff.kc.clay <- (k.c$clay2.pred - ds.val.sp$clay2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-30.000  -3.980   0.912   0.511   5.090  28.900

> (rmse.kc.clay <- sqrt(sum(diff.kc.clay^2)/length(diff.kc.clay)))

[1] 9.6983
```

Q17 : *How do these compare with the bias and RMSE of the fractions predicted separately?* *Jump to A17* •

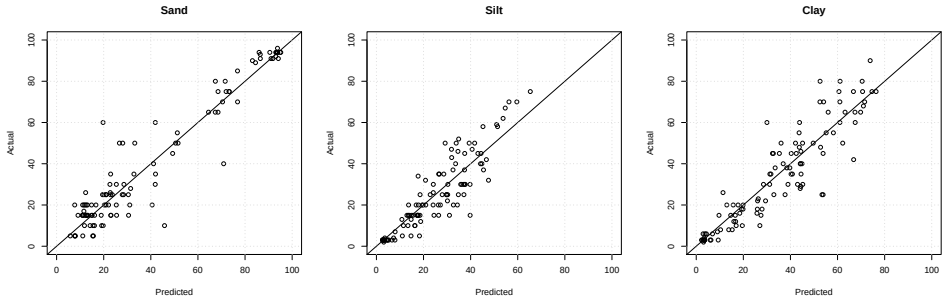
When comparing the RMSE, we expect co-kriging to be more accurate and precise, so we express the differences as the presumed improvement by co-kriging, i.e., the values for OK less those for CK, which (we hope) are positive:

```
> mean(diff.k.sand) - mean(diff.kc.sand)

[1] -0.081116

> mean(diff.k.silt) - mean(diff.kc.silt)

[1] 0.094019

> mean(diff.k.clay) - mean(diff.kc.clay)

[1] -0.012903

> rmse.k.sand - rmse.kc.sand
```

```
[1] -0.15294

> rmse.k.silt - rmse.kc.silt

[1] -0.31204

> rmse.k.clay - rmse.kc.clay

[1] -0.40274
```

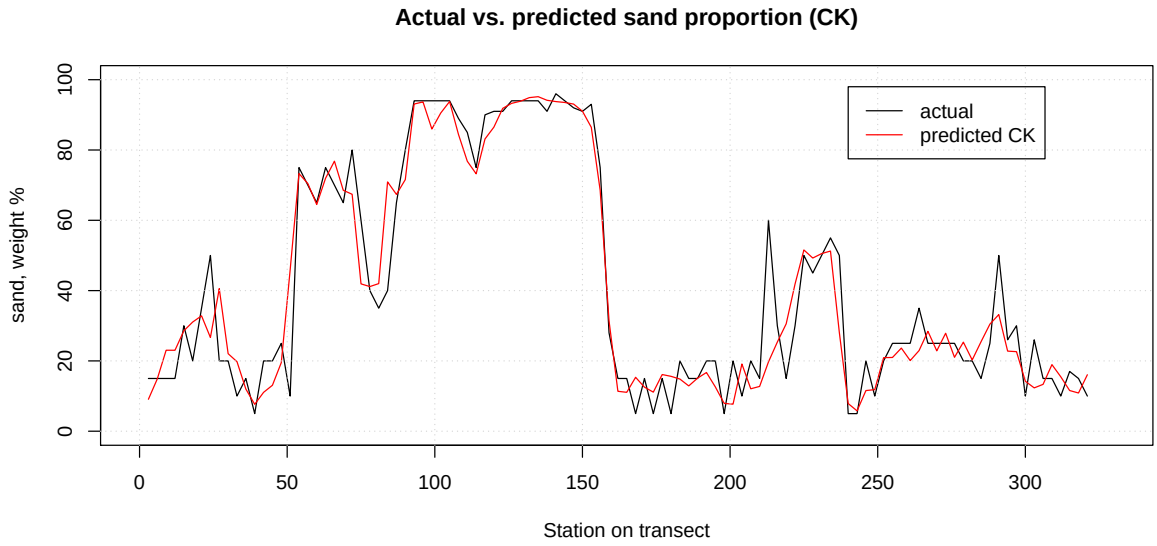
Task 35 : Display 1:1 plots of the actual vs. predicted for the three elements of the evaluation composition for the separate fractions CK approach. •

```
> par(mfrow=c(1,3))
> plot(ds.val[,"sand2"] ~ k.c$sand2.pred,
+      ylab="Actual", xlab="Predicted", main="Sand",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"silt2"] ~ k.c$silt2.pred,
+      ylab="Actual", xlab="Predicted", main="Silt",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"clay2"] ~ k.c$clay2.pred,
+      ylab="Actual", xlab="Predicted", main="Clay",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> par(mfrow=c(1,1))
```



Task 36 : Show the prediction of sand along the transect, along with the true values, for the CK approach. •

```
> plot(ds.val$sand2 ~ ds.val$seq, type = "l", ylim = c(0,
+     100), xlim = c(0, 330), main = "Actual vs. predicted sand proportion (CK)",
+     xlab = "Station on transect", ylab = "sand, weight %",
+     cex = 0.6)
> lines(k.c$sand2.pred ~ coordinates(ds.val.sp)[, 1], col = "red",
+     type = "l", cex = 0.6)
> grid()
> legend(240, 98, c("actual", "predicted CK"), lty = 1,
+     col = c("black", "red"))
```



Challenge: In this approach two components must be chosen to model and interpolate, and the third determined by subtraction. We chose to model sand and silt, and determined clay by subtraction. Repeat the analysis for the other two pairs: sand and clay, and silt and clay. Compare the results – in principle they should be identical. What does this imply for this method of geostatistical analysis of compositional variables?

3.4 Ordinary kriging ALR variables

Our first attempt with the ALR-transformed compositional variables is to consider each one separately, and back-transform the resulting composition to the three particle-size fractions. Below (§3.5) we will co-krige them.

Recall, we computed the ALR-transform of the calibration observations in §2.4:

```
> summary(alr2)

         sand2  silt2
Min.    -3.040 -2.890
1st Qu. -1.180 -0.847
Median  -0.134 -0.168
Mean     0.181 -0.288
```

```
3rd Qu.   1.430   0.118
Max.      3.870   2.010
attr(,"class")
[1] "summary.rmult" "matrix"
```

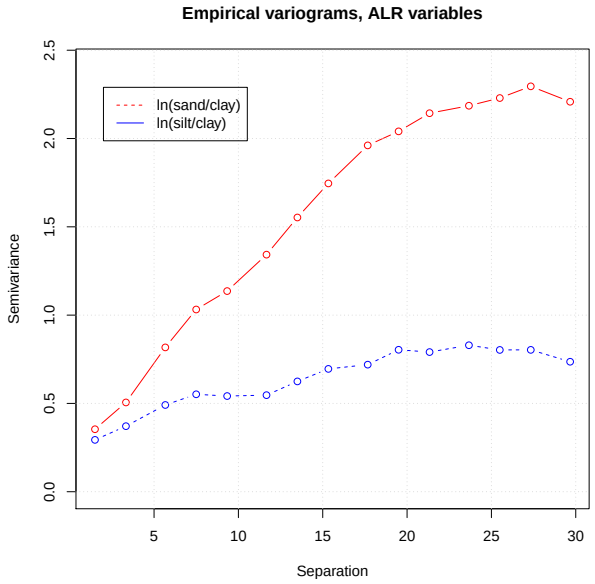
Task 37 : Add spatial reference to these variables. •

To work with these as spatial variables, we first add the transformed variables to the spatial object. Although they have no explicit reference, their sequence is the same as the spatial object of the calibration observations, so they can just be added as a field with the same sequence:

```
> ds.cal.sp$alr.1 <- alr2[, 1]
> ds.cal.sp$alr.2 <- alr2[, 2]
```

Task 38 : Compute and display the variograms for the two ALR-transformed variables. •

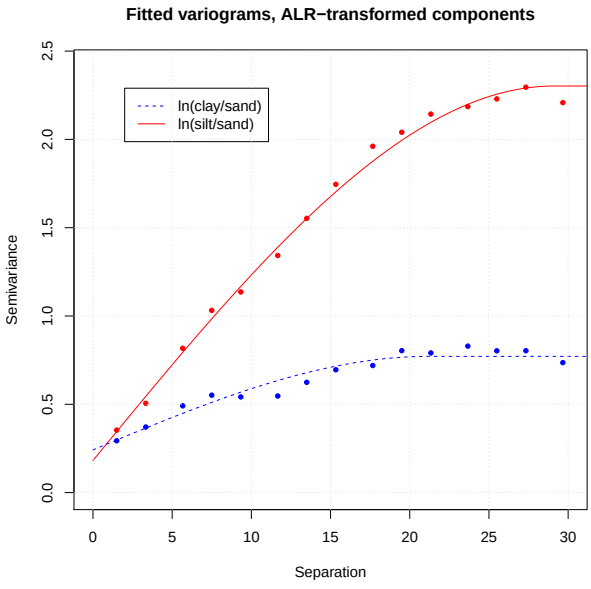
```
> v.alr.1 <- variogram(alr.1 ~ 1, loc = ds.cal.sp, cutoff = 30)
> v.alr.2 <- variogram(alr.2 ~ 1, loc = ds.cal.sp, cutoff = 30)

> sv.max <- max(v.alr.1$gamma, v.alr.2$gamma)*1.05
> plot(v.alr.1$gamma ~ v.alr.1$dist, ylim=c(0,sv.max),
+      ylab="Semivariance", xlab="Separation",
+      main="Empirical variograms, ALR variables",
+      type="b", col="red")
> lines(v.alr.2$gamma ~ v.alr.2$dist, type="b", lty=2, col="blue")
> grid()
> legend(2,sv.max*.95, c("ln(sand/clay)","ln(silt/clay)"),
+        lty=2:1, col=c("red","blue"))
```



Q18 : *Describe the spatial structure of the two components.* *Jump to A18* •

Task 39 : Model the variograms and display the fitted model on the empirical variograms. •

```
> (vm.alr.1 <- fit.variogram(v.alr.1, vgm(2, "Sph", 25,
+     0.4)))

  model  psill  range
1   Nug 0.1820  0.000
2   Sph 2.1202 29.083

> (vm.alr.2 <- fit.variogram(v.alr.2, vgm(1, "Sph", 25,
+     0.4)))

  model   psill  range
1   Nug 0.24175  0.000
2   Sph 0.52981 21.228

> plot(v.alr.1$gamma ~ v.alr.1$dist, ylim = c(0, sv.max),
+     ylab = "Semivariance", xlab = "Separation", xlim = c(0,
+         30), main = "Fitted variograms, ALR-transformed components",
+     col = "red", pch = 20)
> lines(variogramLine(vm.alr.1, maxdist = 100), lty = 1,
+     col = "red")
> points(v.alr.2$gamma ~ v.alr.2$dist, col = "blue", pch = 20)
> lines(variogramLine(vm.alr.2, maxdist = 100), lty = 2,
+     col = "blue")
> legend(2, sv.max * 0.95, c("ln(clay/sand)", "ln(silt/sand)"),
+     lty = 2:1, col = c("blue", "red"))
> grid()
```



Task 40 : Predict the composition components at the evaluation locations. •

```
> k.alr.1 <- krige(alr.1 ~ 1, loc = ds.cal.sp, newdata = ds.val.sp,
+     model = vm.alr.2)

[using ordinary kriging]

> k.alr.2 <- krige(alr.2 ~ 1, loc = ds.cal.sp, newdata = ds.val.sp,
+     model = vm.alr.2)

[using ordinary kriging]
```

3.4.1 Evaluation of the OK ALR variables approach

For ease of interpretation and to compare with non-compositional approaches, we want to evaluate in the space of the original variables, i.e., the back-transformed predictions, not in the space of the kriged estimates, i.e., the additive log-ratios. Unfortunately, the back-transform is biased; intuitively, this is because we've obtained the interpolated values by a weighted average of log-ratios, and addition of logarithms is equivalent to multiplication of original values. Further, these are ratios. The problem is explained and solved by Lark and Bishop [5], resulting in their equation (6). However, in practice the bias of the ALR back-transform, while unknown, is small, so we will ignore it here.

Task 41 : Back-transform the interpolated log-ratios to the original composition variables, and use these to evaluate the interpolation. •

To apply the inverse transformation, we must first prepare a dataframe with the two kriging predictions and then back-transform it with the `alrInv` function. After back-transforming, we convert the object to a composition with the known total (here, 100%) to recover percentages (the original size), using the `acomp` function:

```
> k.alr.comp <- data.frame(alr.1 = k.alr.1$var1.pred,
+                          alr.2 = k.alr.2$var1.pred)
> comp.2.ok <- acomp(as.data.frame(alrInv(k.alr.comp,
+                                  orig=comp.2)),total=100)
> str(comp.2.ok)

 acomp [1:107, 1:3] 10 13.4 21 24.9 28.4 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:3] "sand2" "silt2" "clay2"
```

By definition the compositions sum to the composition's size; we can check this with the `apply` function to apply the `sum` function to the rows (i.e., observations); the second argument to `apply` is the array margin to sum over, here 1 for rows.

```
> summary(apply(comp.2.ok, 1, sum))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    100     100     100     100     100     100
```

Task 42 : Compute the bias and RMSE of the actual vs. predicted values. •

```
> summary(diff.ka.sand <- (comp.2.ok[, "sand2"] - ds.val.sp$sand2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-40.800  -3.500  -0.584  -0.551   4.200  33.400

> (rmse.ka.sand <- sqrt(sum(diff.ka.sand^2)/length(diff.ka.sand)))

[1] 9.0655

> summary(diff.ka.silt <- (comp.2.ok[, "silt2"] - ds.val.sp$silt2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-21.700  -5.130   0.812   0.115   4.910  23.800

> (rmse.ka.silt <- sqrt(sum(diff.ka.silt^2)/length(diff.ka.silt)))

[1] 7.7842

> summary(diff.ka.clay <- (comp.2.ok[, "clay2"] - ds.val.sp$clay2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-28.200  -4.130   0.703   0.436   5.050  32.100

> (rmse.ka.clay <- sqrt(sum(diff.ka.clay^2)/length(diff.ka.clay)))

[1] 9.3731
```

Q19 : *How do these evaluation statistics compare to those from the direct approaches?* *Jump to A19* •

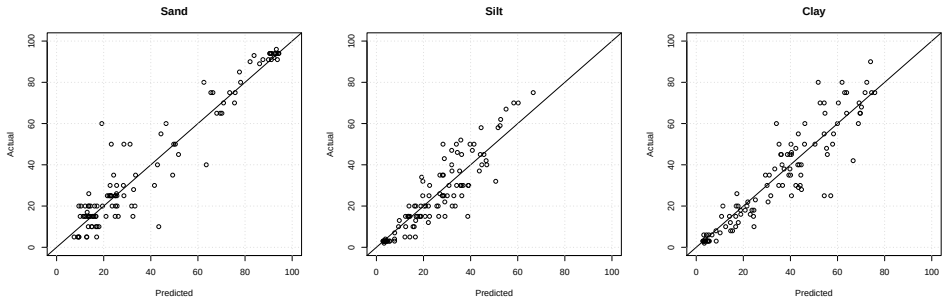
Since OK was superior to CK, we only need to compare this to the direct OK. We might expect OK of the ALR variables to be more accurate and precise, so express the differences as the presumed improvement by kriging the ALR-transformed variables instead of the untransformed variables:

```
> mean(diff.k.sand) - mean(diff.ka.sand)

[1] -0.26512

> mean(diff.k.silt) - mean(diff.ka.silt)

[1] 0.20288

> mean(diff.k.clay) - mean(diff.ka.clay)

[1] 0.062241

> rmse.k.sand - rmse.ka.sand

[1] 0.062652
```

```
> rmse.k.silt - rmse.ka.silt

[1] -0.22097

> rmse.k.clay - rmse.ka.clay

[1] -0.077539
```

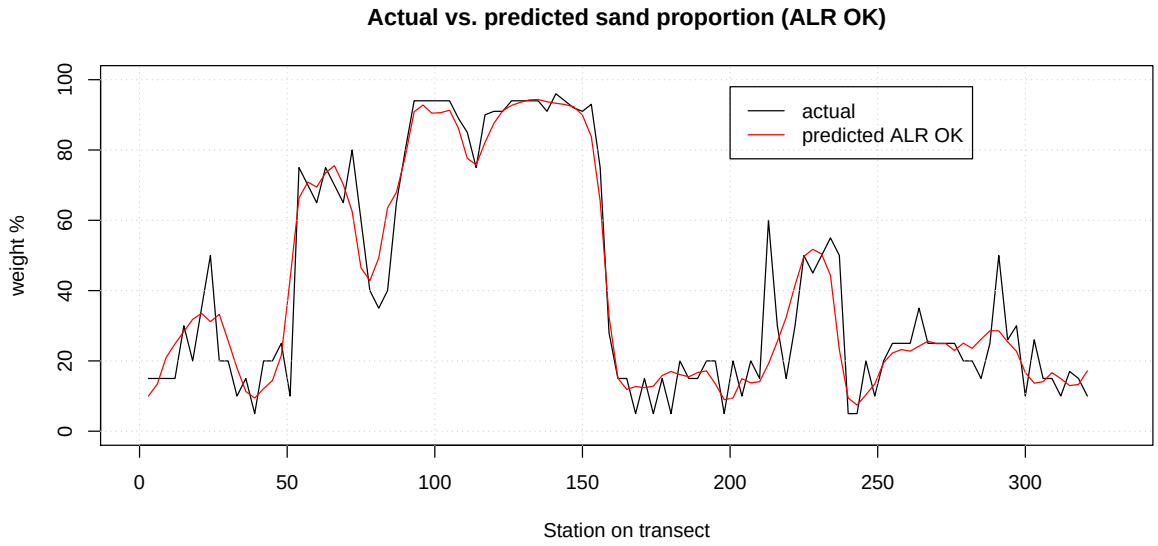
Task 43 : Display 1:1 plots of the actual vs. predicted for the three elements of the evaluation composition from the ALR OK approach. •

```
> par(mfrow=c(1,3))
> plot(ds.val[,"sand2"] ~ comp.2.ok[,"sand2"],
+      ylab="Actual", xlab="Predicted", main="Sand",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"silt2"] ~ comp.2.ok[,"silt2"],
+      ylab="Actual", xlab="Predicted", main="Silt",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"clay2"] ~ comp.2.ok[,"clay2"],
+      ylab="Actual", xlab="Predicted", main="Clay",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> par(mfrow=c(1,1))
```



Task 44 : Show the actual vs. predicted for sand along the transect from the ALR OK approach, for the evaluation points. •

```
> plot(ds.val$sand2 ~ ds.val$seq, type = "l", ylim = c(0,
+     100), xlim = c(0, 330), main = "Actual vs. predicted sand proportion (ALR OK)",
+     xlab = "Station on transect", ylab = "weight %",
+     cex = 0.6)
> lines(comp.2.ok[, "sand2"] ~ coordinates(ds.val.sp)[,
+     1], col = "red", type = "l", cex = 0.6)
> grid()
> legend(200, 98, c("actual", "predicted ALR OK"), lty = 1,
+     col = c("black", "red"))
```



Challenge: In this approach we had to choose one components to normalise the other two, i.e., as denominator of the log-ratio. We chose clay. Repeat the analysis with the other two possibilities, i.e., sand and silt as normalising components. Compare the results – in principle they should be identical. What does this imply for this method of geostatistical analysis of compositional variables?

3.5 Co-kriging ALR variables

The final approach is to co-krige the ALR-transformed variables. If these have a cross-correlated spatial structure which can be represented by a linear model of co-regionalization, we can hope for a more precise estimate. We repeat the procedures of §3.3, but with the two ALR-transformed variables.

Task 45 : Build a `gstat` structure to represent the variables to model. •

```
> (gc <- gstat(NULL, id = "sand.clay", form = alr.1 ~ 1,
+     data = ds.cal.sp))

data:
sand.clay : formula = alr.1`~`1 ; data dim = 214 x 11
```

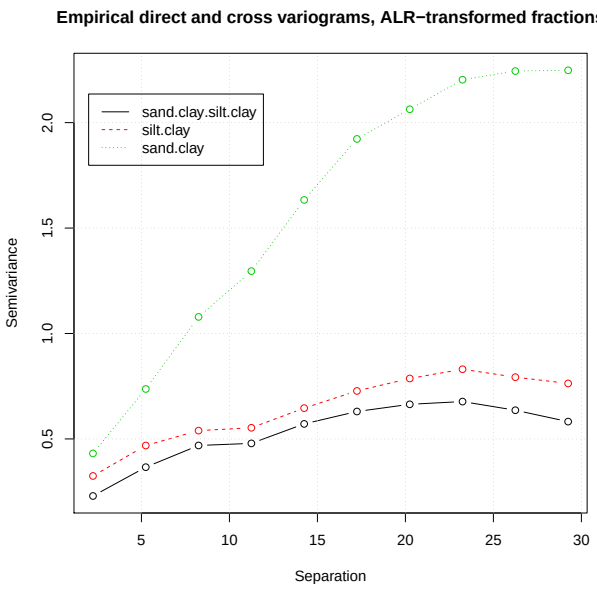
```
> (gc <- gstat(gc, id = "silt.clay", form = alr.2 ~ 1,
+     data = ds.cal.sp))

data:
sand.clay : formula = alr.1`~`1 ; data dim = 214 x 11
silt.clay : formula = alr.2`~`1 ; data dim = 214 x 11
```

Task 46 : Compute and display the direct and cross-variograms. •

```
> v.cross <- variogram(gc, cutoff = 30, width = 3)
```

Task 47 : Display the direct and cross-variograms on one graph. •

```
> sv.max <- max(v.cross$gamma)
> sv.min <- min(v.cross$gamma)
> plot(v.cross$gamma ~ v.cross$dist, ylim=c(sv.min,sv.max),
+   ylab="Semivariance", xlab="Separation",
+   main="Empirical direct and cross variograms, ALR-transformed fractions",
+   type="n")
> for (i in 1:3) {
+   tmp <- subset(v.cross, as.numeric(v.cross$id) == i)
+   lines(tmp$gamma ~ tmp$dist, col=i, type="b", lty=i)
+   }
> grid()
> abline(h=0, lty=2)
> legend(2,sv.max*.95, levels(v.cross$id), lty=1:3, col=1:3)
```



Q20 : *Why does the cross-variogram here have a positive sill, whereas the cross-variogram between sand and silt had a negative sill?* *Jump to A20* •

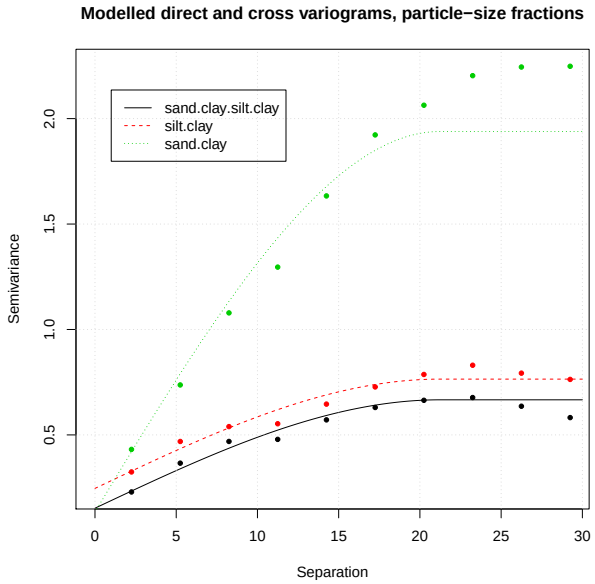
Task 48 : Fill the variogram models with an initial guess. •

Again we use the shorter-range of the fitted direct variograms from the previous section.

```
> gc <- gstat(gc, id = "silt.clay", model = vm.alr.2, fill.all = T)
```

Task 49 : Fit the variograms with the linear model of co-regionalization and display them on the empirical variogram. •

```
> (gc <- fit.lmc(v.cross, gc))

data:
sand.clay : formula = alr.1`~`1 ; data dim = 214 x 11
silt.clay : formula = alr.2`~`1 ; data dim = 214 x 11
variograms:
                       model   psill  range
sand.clay[1]             Nug 0.13475  0.000
sand.clay[2]             Sph 1.80399 21.228
silt.clay[1]             Nug 0.24639  0.000
silt.clay[2]             Sph 0.51823 21.228
sand.clay.silt.clay[1]   Nug 0.15247  0.000
sand.clay.silt.clay[2]   Sph 0.51399 21.228

> plot(v.cross$gamma ~ v.cross$dist, ylim=c(sv.min,sv.max),
+      xlim=c(0,max(v.cross$dist)),
+      ylab="Semivariance", xlab="Separation",
+      main="Modelled direct and cross variograms, particle-size fractions",
+      type="p",
+      col=as.numeric(v.cross$id), pch=20)
> for (i in 1:3) {
+   tmp <- c("sand.clay.silt.clay","silt.clay","sand.clay")[i]
+   lines(variogramLine(gc$model[[tmp]], maxdist=30), col=i, lty=i)
+   }
> grid()
> abline(h=0, lty=2)
> legend(1,sv.max*.95, levels(v.cross$id), lty=1:3, col=1:3)
```



Q21 : *How appropriate is the linear model of co-regionalization here? In other words, how valid is the assumption of same model form (here, spherical) and the same range?* *Jump to A21* •

Task 50 : Interpolate by co-kriging. •

The general `predict.gstat` must be used to predict in the cokriging system.

```
> k.c.c <- predict.gstat(gc, ds.val.sp)

Linear Model of Coregionalization found. Good.
[using ordinary cokriging]

> summary(k.c.c)

Object of class SpatialPointsDataFrame
Coordinates:
    min max
seq   3 321
y     0   0
Is projected: NA
proj4string : [NA]
Number of points: 107
Data attributes:
 sand.clay.pred   sand.clay.var   silt.clay.pred     silt.clay.var
 Min.   :-2.639   Min.   :0.302   Min.   :-1.65610   Min.   :0.338
 1st Qu.:-1.214   1st Qu.:0.302   1st Qu.:-0.52246   1st Qu.:0.338
 Median :-0.301   Median :0.302   Median :-0.21877   Median :0.338
 Mean   : 0.187   Mean   :0.303   Mean   :-0.28590   Mean   :0.338
 3rd Qu.: 1.531   3rd Qu.:0.302   3rd Qu.: 0.00156   3rd Qu.:0.338
 Max.   : 3.661   Max.   :0.448   Max.   : 1.35279   Max.   :0.407
 cov.sand.clay.silt.clay
 Min.   :0.224
```

```
 1st Qu.:0.224
 Median :0.224
 Mean   :0.224
 3rd Qu.:0.224
 Max.   :0.281
```

3.5.1 Evaluation of the CK ALR variables approach

Task 51 : Back-transform the interpolated log-ratios to the original composition variables, and use these to evaluate the interpolation. •

```
> kc.alr.comp <- data.frame(alr.1 = k.c.c$sand.clay.pred,
+                   alr.2 = k.c.c$silt.clay.pred)
> comp.2.ck <- acomp(as.data.frame(alrInv(kc.alr.comp,
+                                   orig=comp.2)), total=100)
> str(comp.2.ck)

 acomp [1:107, 1:3] 8.68 13.37 23.97 21.92 30.17 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:3] "sand2" "silt2" "clay2"
```

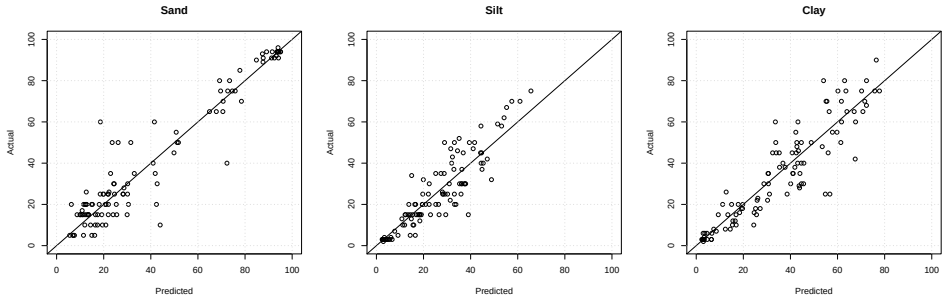
Again, by definition the compositions sum to the composition's size.

Task 52 : Compute the bias and RMSE of the actual vs. predicted values. •

```
> summary(diff.kca.sand <- (comp.2.ck[, "sand2"] - ds.val.sp$sand2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-41.400  -4.970  -0.091  -0.501   2.990  34.000

> (rmse.kca.sand <- sqrt(sum(diff.kca.sand^2)/length(diff.kca.sand)))

[1] 9.4599

> summary(diff.kca.silt <- (comp.2.ck[, "silt2"] - ds.val.sp$silt2))

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-21.1000  -4.4800   0.7630  -0.0225   4.7500  24.0000

> (rmse.kca.silt <- sqrt(sum(diff.kca.silt^2)/length(diff.kca.silt)))

[1] 7.7644

> summary(diff.kca.clay <- (comp.2.ck[, "clay2"] - ds.val.sp$clay2))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-26.400  -4.300   0.647   0.523   5.440  31.700

> (rmse.kca.clay <- sqrt(sum(diff.kca.clay^2)/length(diff.kca.clay)))

[1] 9.5597
```

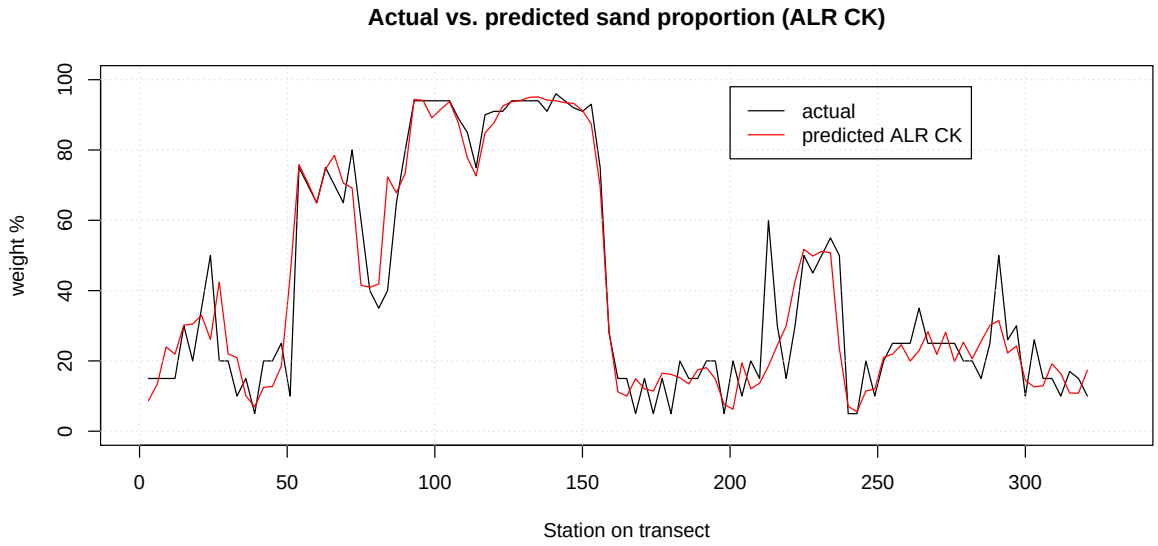
Task 53 : Display 1:1 plots of the actual vs. predicted for the three elements of the evaluation composition and the ALR CK approach. •

```
> par(mfrow=c(1,3))
> plot(ds.val[,"sand2"] ~ comp.2.ck[,"sand2"],
+      ylab="Actual", xlab="Predicted", main="Sand",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"silt2"] ~ comp.2.ck[,"silt2"],
+      ylab="Actual", xlab="Predicted", main="Silt",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> plot(ds.val[,"clay2"] ~ comp.2.ck[,"clay2"],
+      ylab="Actual", xlab="Predicted", main="Clay",
+      xlim=c(0,100), ylim=c(0,100))
> abline(0,1); grid()
> par(mfrow=c(1,1))
```



Task 54 : Show the actual vs. predicted for sand along the transect from the ALR CK approach, for the evaluation points. •

```
> plot(ds.val$sand2 ~ ds.val$seq, type = "l", ylim = c(0,
+     100), xlim = c(0, 330), main = "Actual vs. predicted sand proportion (ALR CK)",
+     xlab = "Station on transect", ylab = "weight %",
+     cex = 0.6)
> lines(comp.2.ck[, "sand2"] ~ coordinates(ds.val.sp)[,
+     1], col = "red", type = "l", cex = 0.6)
> grid()
> legend(200, 98, c("actual", "predicted ALR CK"), lty = 1,
+     col = c("black", "red"))
```



Challenge: The variogram fit was not particularly good. Since there are so many known points near any point to be predicted, maybe there would be a better linear model of co-regionalization if the variogram were computed and fit only out to the first "sill" at 12 stations. Try this.

Challenge: As in ALR-OK, here we had to choose one components to normalise the other two, i.e., as denominator of the log-ratio. We chose clay. Repeat the analysis with the other two possibilities, i.e., sand and silt as normalising components. Compare the results – in principle they should be identical. What does this imply for this method of geostatistical analysis of compositional variables?

4 Comparing kriging approaches

We compare the success of the four kriging approaches two ways: (1) how well they predict each component of the composition (§4.1); (2) composite measures of success over the whole composition (§4.2).

4.1 Reproducing components of the evaluation composition

Task 55 : Build a table to compare the evaluation statistics of the four approaches (§3.2.1, §3.3, §3.5, §3.5: their biases (mean errors), precision (RMSE), and the RMSE averaged over the composition. •

Note that the evaluation statistics for the "OK of separate variables" approach uses the composition created from the separates in §3.2.2 since we are comparing compositions.

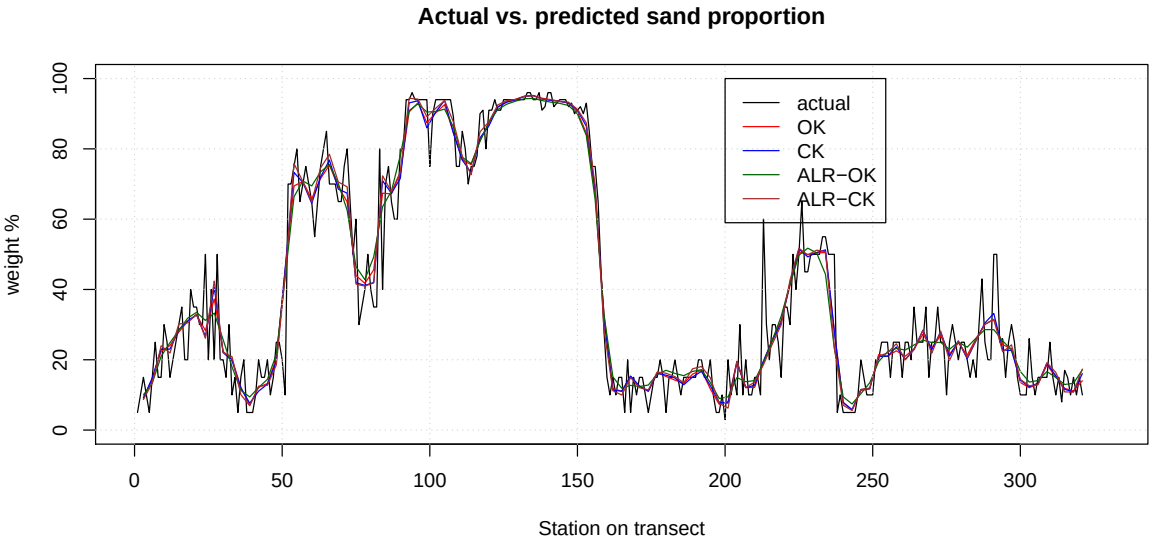
```
> compare <- data.frame(me.sand = 0, me.silt = 0, me.clay = 0,
+     rmse.sand = 0, rmse.silt = 0, rmse.clay = 0)
> compare[1, ] <- c(mean(diff.k.sand), mean(diff.k.silt),
+     mean(diff.k.clay), rmse.k.sand, rmse.k.silt, rmse.k.clay)
> compare[2, ] <- c(mean(diff.kc.sand), mean(diff.kc.silt),
+     mean(diff.kc.clay), rmse.kc.sand, rmse.kc.silt, rmse.kc.clay)
> compare[3, ] <- c(mean(diff.ka.sand), mean(diff.ka.silt),
+     mean(diff.ka.clay), rmse.ka.sand, rmse.ka.silt, rmse.ka.clay)
> compare[4, ] <- c(mean(diff.kca.sand), mean(diff.kca.silt),
+     mean(diff.kca.clay), rmse.kca.sand, rmse.kca.silt,
+     rmse.kca.clay)
> compare <- cbind(compare, mean.me = round(as.vector(apply(compare[,
+     1:3], 1, sum)/3), 5))
> compare <- cbind(compare, mean.rmse = as.vector(apply(compare[,
+     4:6], 1, sum)/3))
> rownames(compare) <- c("OK", "CK", "ALR-OK", "ALR-CK")
> print(compare)

         me.sand   me.silt me.clay rmse.sand rmse.silt rmse.clay
OK     -0.81613  0.318231 0.49789    9.1282    7.5632    9.2956
CK     -0.73501  0.224212 0.51080    9.2811    7.8753    9.6983
ALR-OK -0.55100  0.115349 0.43565    9.0655    7.7842    9.3731
ALR-CK -0.50080 -0.022544 0.52334    9.4599    7.7644    9.5597
       mean.me mean.rmse
OK           0    8.6623
CK           0    8.9516
ALR-OK       0    8.7410
ALR-CK       0    8.9280
```

Q22 : *Which approach gave the best results in this case?* *Jump to A22* •

Task 56 : Plot the actual values of sand content, and the four predictions, over the whole evaluation transect. •

We show all points for the actual transect, but only the evaluation points for the four interpolations.

```
> plot(ds$sand2 ~ ds$seq, type="l", ylim=c(0,100),
+      xlim=c(0,330),
+      main="Actual vs. predicted sand proportion",
+      xlab="Station on transect",
+      ylab="weight %", cex=0.6)
> lines(comp.2.ssc.ok[,"sand2"]   ~
+       coordinates(ds.val.sp)[,1], col="red",
+       type="l", cex=0.6)
> lines(k.c$sand2.pred  ~
+       coordinates(ds.val.sp)[,1], col="blue",
+       type="l", cex=0.6)
> lines(comp.2.ok[,"sand2"]  ~
+       coordinates(ds.val.sp)[,1], col="darkgreen",
+       type="l", cex=0.6)
> lines(comp.2.ck[,"sand2"]  ~
+       coordinates(ds.val.sp)[,1], col="brown",
+       type="l", cex=0.6)
> grid()
> legend(200,100,
+        c("actual","OK","CK","ALR-OK","ALR-CK"),
+        lty=1,
+        col=c("black","red","blue","darkgreen","brown"))
```

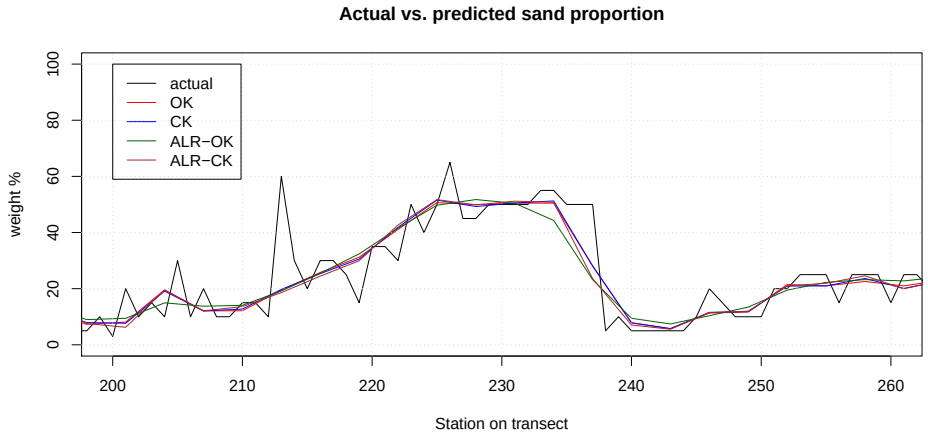


It's difficult to evaluate the whole transect together.

Task 57 : Plot the actual values of sand content, and the four predictions, at one window in the evaluation transect. •

```
> plot(ds$sand2 ~ ds$seq, type = "l", ylim = c(0, 100),
+      xlim = c(200, 260), main = "Actual vs. predicted sand proportion",
+      xlab = "Station on transect", ylab = "weight %",
+      cex = 0.6)
```

```
> lines(comp.2.ssc.ok[, "sand2"] ~ coordinates(ds.val.sp)[,
+     1], col = "red", type = "l", cex = 0.6)
> lines(k.c$sand2.pred ~ coordinates(ds.val.sp)[, 1], col = "blue",
+     type = "l", cex = 0.6)
> lines(comp.2.ok[, "sand2"] ~ coordinates(ds.val.sp)[,
+     1], col = "darkgreen", type = "l", cex = 0.6)
> lines(comp.2.ck[, "sand2"] ~ coordinates(ds.val.sp)[,
+     1], col = "brown", type = "l", cex = 0.6)
> grid()
> legend(200, 100, c("actual", "OK", "CK", "ALR-OK", "ALR-CK"),
+     lty = 1, col = c("black", "red", "blue", "darkgreen",
+         "brown"))
```



Q23 : *In this window, what might explain the similarities and differences between the different interpolation methods?* *Jump to A23* •

Challenge: Repeat the transect evaluation, for silt and clay.

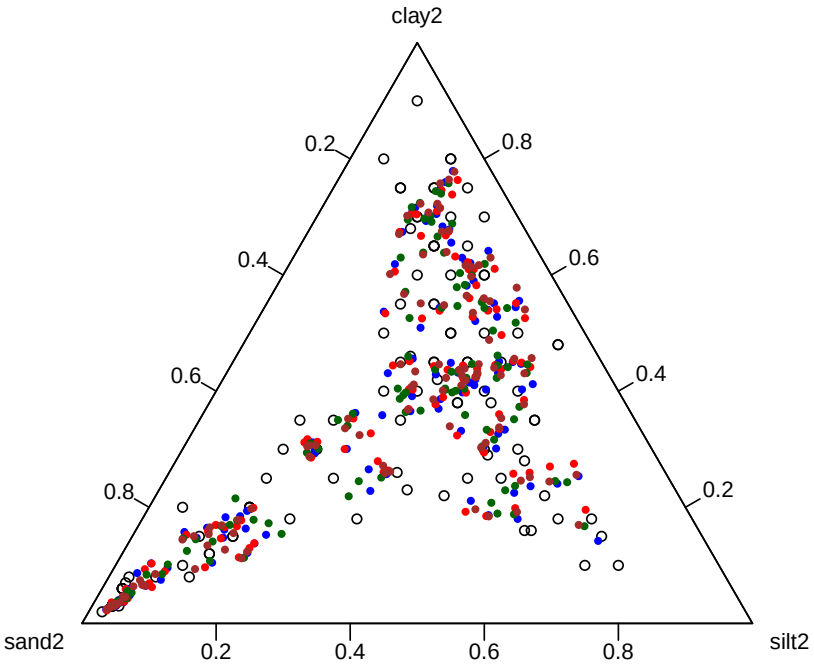
4.2 Overall evaluation of a kriged composition

In the previous sections we evaluated the kriging prediction of each component. One overall measure of evaluation is the distance in composition space (e.g., in the ternary diagram) between the actual and kriged values at the evaluation points.

Task 58 : Display a ternary diagram of the evaluation composition, with the four predictions superimposed. •

Recall, the `plot` method, when when called to plot an object of class `acomp`, uses the `plot.acomp` to plot a ternary diagram. Several point sets can be plotted together, using the `add=T` argument to when called to plot an object of class `acomp`. So first we have to convert the evaluation set and all the predictions to objects of class `acomp`, using the `acomp` function. Note that the OK and CK approaches with ALR variables, and the separate OK approach, are already in this form.

```
> class(comp.2.ok)

[1] "acomp"

> class(comp.2.ck)

[1] "acomp"

> class(comp.2.ssc.ok)

[1] "acomp"

> comp.2.val <- acomp(ds.val[, names.2], total = 100)
> comp.2.ssc.ck <- acomp(data.frame(sand = k.c$sand2.pred,
+     silt = k.c$silt2.pred, clay = k.c$clay2.pred), total = 100)

> opar <- par(no.readonly = T)
> par(pch = 20, cex = 0.8)
> plot(comp.2.val, axes = T, col = "black", pch = 1)
> plot(acomp(comp.2.ssc.ok, total = 100), col = "blue",
+     add = T)
> plot(acomp(comp.2.ssc.ck, total = 100), col = "red",
+     add = T)
> plot(acomp(comp.2.ok, total = 100), col = "darkgreen",
+     add = T)
> plot(acomp(comp.2.ck, total = 100), col = "brown", add = T)
> par(opar)
```

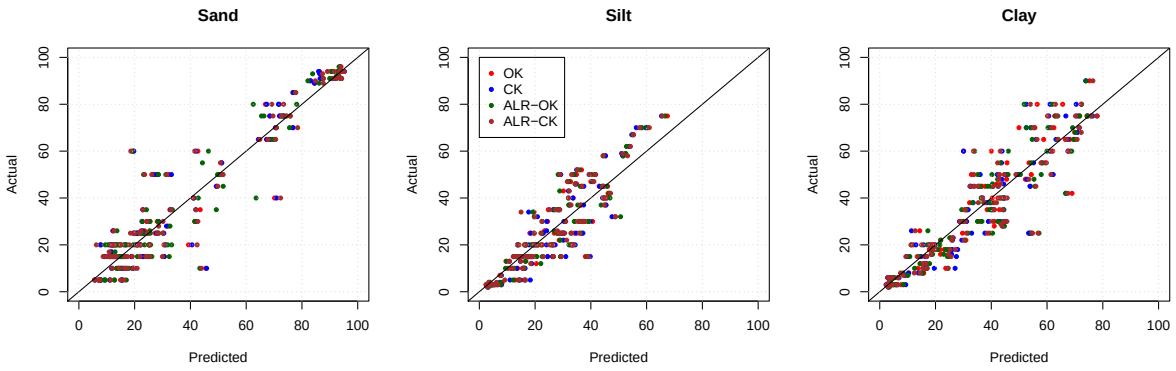


This distance is not directly interpretable. Another way to compare compositions is to see the 1:1 evaluation plots for each component separately.

Task 59 : Display 1:1 plots of the actual vs. predicted for the three elements of the evaluation composition for all approaches. •

These are correctly displayed against a 1:1 line (i.e., predicted equals actual); we use the `plot` method to display the first result, and then the `points` method to add the others in contrasting colours.

```
> opar <- par(no.readonly = T)
> par(mfrow=c(1,3))
> par(pch=20, cex=.8)
> plot(ds.val[,"sand2"] ~ k.sand$var1.pred, col="red",
+      ylab="Actual", xlab="Predicted", main="Sand",
+      xlim=c(0,100), ylim=c(0,100))
> points(ds.val[,"sand2"] ~ k.c$sand2.pred, col="blue")
> points(ds.val[,"sand2"] ~ comp.2.ok[,"sand2"], col="darkgreen")
> points(ds.val[,"sand2"] ~ comp.2.ck[,"sand2"], col="brown")
> #
> abline(0,1); grid()
> plot(ds.val[,"silt2"] ~ k.silt$var1.pred, col="red",
+      ylab="Actual", xlab="Predicted", main="Silt",
+      xlim=c(0,100), ylim=c(0,100))
> points(ds.val[,"silt2"] ~ k.c$silt2.pred, col="blue")
> points(ds.val[,"silt2"] ~ comp.2.ok[,"silt2"], col="darkgreen")
> points(ds.val[,"silt2"] ~ comp.2.ck[,"silt2"], col="brown")
> abline(0,1); grid()
> legend(0,100, c("OK","CK","ALR-OK","ALR-CK"), pch=20,
+        col=c("red","blue","darkgreen","brown"))
> plot(ds.val[,"clay2"] ~ k.clay$var1.pred, col="red",
+      ylab="Actual", xlab="Predicted", main="Clay",
+      xlim=c(0,100), ylim=c(0,100))
> points(ds.val[,"clay2"] ~ k.c$clay2.pred, col="blue")
> points(ds.val[,"clay2"] ~ comp.2.ok[,"clay2"], col="darkgreen")
> points(ds.val[,"clay2"] ~ comp.2.ck[,"clay2"], col="brown")
> abline(0,1); grid()
> par(opar)
```



Q24 : *Which approach, if any, appears to predict better overall?* *Jump to A24* •

Another feature of a composition is its **overall variability**.

Pawlowsky-Glahn and Olea [8] use a multivariate measure known as STRESS (**s**tandarized **r**esidual **s**um of **s**quares) to measure the overall similarity of the predictions and the evaluation data; this is explained by Lark and Bishop [5, Eqn. (7)], following Martin-Fernandez et al. [6, Eqn. (11)]:

$$\text{STRESS} = \left\{ \frac{\sum_{i<j} (\delta_{i,j} - \delta^*_{i,j})^2}{\sum_{i<j} (\delta_{i,j})^2} \right\}^{1/2} \tag{4}$$

where $\delta_{i,j}$ is a distance measure between two observed compositions $\mathbf{y}_i$ and $\mathbf{y}_j$, and $\delta^*_{i,j}$ is the same, but for the compositions projected into a sub-space. In geostatistical applications, Lark and Bishop [5] consider the smoothed space produced by kriging to be the sub-space, because it in general occupies a smaller hypervolume of the D-dimensional space formed by the compositional variables. In this case STRESS measures how well the variations of the kriged estimates match the variations of the observations. The distances are summed over all point-pairs in each set[8] and then normalized by the sum in the original (higher-dimension) set. If the lower-dimensional observations (in this case the kriged estimates) match the true values, STRESS is 0; at the other extreme the kriged observations would all be the same no matter what the true values, and STRESS is 1. Clearly, the lower the STRESS, the more successfully the kriging reproduces the variability of the evaluation data set.

There are various ways to measure distance between two vectors, in this case two compositions. Following [5, Eqn. (8)], we use the Aitchison distance, which is the Euclidean distance between the **centred log-ratio transform** of the compositions:

$$\text{clr}(\mathbf{x}) = \left[\ln \frac{x_1}{\check{\mathbf{x}}}, \ldots, \ln \frac{x_D}{\check{\mathbf{x}}} \right] \tag{5}$$

where $\check{\mathbf{z}}$ is the geometric mean of the the D-dimensional composition:

$$\check{\mathbf{z}} = \left(\prod_{i=1}^{D} z_i \right)^{1/D} \tag{6}$$

This transformation transforms all the elements of the composition[9] so is permutation-invariant. The reason for not just using the Euclidean distance in the original space is to account for the constant-sum constraint.

We illustrate this approach with the ALR-OK approach of §3.4.

Task 60 : Compute the centred log-ratio transform of the actual and ALR-OK kriged evaluation compositions and summarize their difference. •

This uses the `clr` function:

```
> c.k <- clr(comp.2.ok)
> c.o <- clr(comp.2.val)
> summary(c.k) - summary(c.o)
```

[8] this is the meaning of the $\sum_{i<j}$ in Equation 4

[9] unlike the additive log-ratio transform, which uses one as a denominator

```
          sand2  silt2   clay2
Min.      0.450  0.280  0.1200
1st Qu. -0.100  0.089  0.0870
Median  -0.055  0.064  0.0763
Mean    -0.038  0.032  0.0069
3rd Qu. -0.039 -0.006 -0.0310
Max.    -0.240 -0.289 -0.6500
attr(,"class")
[1] "summary.rmult" "matrix"
```

Task 61 : Compute the distances between all observations in the evaluation set; do the same for all kriged estimates of the evaluation set. •

The `dist` function returns a distance matrix of size $[n \cdot (n-1)]/2$, by default using the Euclidean distance, between the rows of a data matrix, i.e., the observations. Since we apply it to the CLR-transformed matrices, these are Aitchison distances as required by Equation 4.

```
> dk <- dist(c.k)
> do <- dist(c.o)
```

Task 62 : Compute the STRESS of the evaluation for ALR-OK. •

```
> sqrt(sum((do - dk)^2)/sum(do^2))

[1] 0.27131
```

Q25 : *Evaluate this STRESS value: how well does the kriging prediction reproduce the variability of the original evaluation composition? Recall: STRESS=0 means a perfect reproduction, STRESS=1 is when all kriging predictions are the same (smoothing to the spatial mean).* *Jump to A25* •

The above steps can be combined in a function, which can then be used to evaluate any prediction.

Task 63 : Write a function to compute the STRESS from two compositions. •

The `function` function is used to defined a function in the workspace:

```
> stress <- function(v1, v2)
+   {
+     d1 <- dist(clr(v1)); d2 <- dist(clr(v2))
+     return(sqrt(sum((d2 - d1)^2)/sum(d1^2)))
+   }
```

Note that the reference composition (denominator in STRESS) is listed first.

Test that it gives the same result for OK:

```
> print(stress(ds.val[, names.2], comp.2.ok))

[1] 0.27131
```

Task 64 : Compute the STRESS for the other three approaches. •

```
> stress(comp.2.val, comp.2.ssc.ok)

[1] 0.26708

> stress(comp.2.val, comp.2.ssc.ck)

[1] 0.274

> stress(comp.2.val, comp.2.ok)

[1] 0.27131

> stress(comp.2.val, comp.2.ck)

[1] 0.26921
```

Q26 : *Which approach best reproduces the variability of the evaluation composition?* *Jump to A26* •

5 Compositional kriging

Walvoort and de Gruijter [11] developed another approach to spatial interpolation of compositional data, called "compositional kriging"; it has been applied to soil particle-size distribution by these authors and Odeh et al. [7], and to fuzzy memberships (which must sum to 1) by de Gruijter et al. [4]. In this approach the compositional variables, transformed by the ALR transform, are modelled independently, without considering cross-correlations. This avoids the restriction to the linear model of co-regionalization.

The kriging system is further restricted such that (1) the prediction of each ratio of the composition is unbiased (as in OK); (2) all predictions of the ratios are non-negative; (3) the predictions sum to a constant 1 at each prediction location.

A future version of these notes may include this approach.

6 Conclusions

In this case study we went to a lot of work to explain and use compositions, rather than naïvely work with the original particle-size fractions. The results were disappointing, in the sense that the prediction accuracy and precision were not improved; as it turned out, the compositional variable created from the three separate predictions by OK performed best. This composition was a simple solution to the fact that OK of the separates will not in general result in a sum to the original composition size (here, 100%). Surprisingly,

ordinary co-kriging of the composition (not transformed) did not improve predictions, perhaps because the linear model of co-regionalization imposed unrealistic restrictions on the prediction.

Lark and Bishop [5] obtained a similar result, and explain it by the good distribution of the Sandford observations within the simplex. They conclude:

> "We conjecture that the practical advantages of the [ALR] method over ordinary cokriging of the raw compositions would be much larger than seen here in a study area where the size fractions are centred near a vertex of the simplex."

This would be the case if, for example, the soils were all coarse-textured (sands, sandy loams, loamy sands).

As a final note, our results are somewhat poorer (higher RMSE and STRESS) than those of Lark and Bishop [5] on the same dataset and with the same evaluation points; compare their Table 2 with our evaluation results in §4.1. The only possible source of the discrepancy is the different fitting of the variograms. These authors used a simulated annealing program, while we used the `gstat` package's automatic fit. This shows clearly the importance of fitting a model of spatial covariance (e.g., a variogram) from the presumed spatial stochastic process that gave rise to the data, from a single realization (i.e., nature) and a limited sample.

7 Answers

A1 : *There are clear regions of high and low sand (e.g., high between stations 50 and 150), but there a high local variability at some groups of stations.* *Return to Q1* •

A2 : *The feature-space variability is quite high: all fractions range from almost none to very high (almost 100% for clay); the inter-quartile ranges are also wide.* *Return to Q2* •

A3 : *The* `acomp` *geometry, because particle-size fraction percentages are proportions of a total, and the scale is relative: there is no absolute meaning to a percentage.* *Return to Q3* •

A4 : *The silt/clay log ratio is the smallest, thus it varies the least along the transect.* *Return to Q4* •

A5 : *The sand/clay ratio has as much or more variability than any of the untransformed fractions, but the silt/clay ratio is much less variable. Notice especially the portion from stations 50 – 100.* *Return to Q5* •

A6 : *No, there are very few observations at the extreme silt corner (no pure silts; recall the maximum silt was only about 75%), and few at sand/clay near 0.5 and clay/silt near 0.5.* *Return to Q6* •

A7 : *A ternary diagram shows that knowing any two variables the other is automatically determined.* *Return to Q7* •

A8 : *The sills correspond to the total variability, much less for silt than for sand; clay is intermediate. All three nuggets are low and about the same. The range for clay is about 22, but for silt and sand it seems the sill is not reached by the variogram limit of 35 stations separation. All seem to show a double structure: at shorter ranges a steeply-increasing variogram, then much less steeply increasing, for clay even decreasing.* *Return to Q8* •

A9 : *The structures are not similar. Sand is most variable (higher sill) and has a long range; silt is least variable (lowest sill) but with a long range; clay has an intermediate sill and the shortest range. The different nuggets might be explained by the higher laboratory precision for sand (if determined by sieving) than silt and clay (if determined by sedimentation). The shorter range for clay might be because of clay neoformation, but this is difficult to interpret.* *Return to Q9* •

A10 : *No, there is a discrepancy of up to* $\approx \pm 4\%$. *But, there appears to be little*

spatial pattern. Only around stations 120–150 does there appear to be consistently low discrepancies. High and low discrepancies show no pattern. Return to Q10 •

A11 : *There is a slight negative overall bias (over-prediction) of the sand fraction, but less than 1% sand; this is compensated by slight overall under-predictions of silt and clay. The RMSE seem high, about 7.5–9.2%.* Return to Q11 •

A12 : *The largest errors are where the actual values make sharp jumps, e.g., near station 220. Kriging smooths away local fluctuations (e.g., near stations 160–200) and misses local extremes.* Return to Q12 •

A13 : *The biases are slightly more extreme, and the RMSE are slightly higher (worse) than for the separately-interpolated fractions.* Return to Q13 •

A14 : *All three variograms have good structure; the sills are different (sand much higher than silt); the cross-variogram is negative as expected.* Return to Q14 •

A15 : *The LMC seems quite appropriate here, all three variograms are reasonably well fit.* Return to Q15 •

A16 : *All predictions are within the required range* $[0 \ldots 100]$*; this is not guaranteed when co-kriging (it is not a convex predictor).* Return to Q16
•

A17 : *The biases are slightly more extreme, and the RMSE are slightly higher (worse) than for the separately-interpolated fractions. The statistics are quite close to those from ordinary kriging of the composition.* Return to Q17 •

A18 : *The sand/clay log ratio is much more variable than the silt/clay log ratio, and also seems to have a longer range. The nuggets are the same.* Return to Q18
•

A19 : *The biases are slightly less but the RMSE quite similar to the other approaches.* Return to Q19
•

A20 : *Both variables are log ratios with the same denominator (clay). So, when clay decreases both of these increase, and vice-versa.* Return to Q20 •

A21 : *The LMC is not so appropriate here because it can not reproduce the longer range of the sand/clay log-ratio. However at short ranges the model fits fairly well.*

Return to Q21 •

A22 : *There is no clear winner. ALR-CK has the lowest biases for sand and silt, but the highest for clay. The mean RMSE is best for OK but all are very close, and disappointingly high (near 9%).* *Return to Q22* •

A23 : *There is not much difference, but the OK of the single target fraction, as well as the CK of this fraction with the others, comes closer to the sharp transitions. The ALR-OK is smoothest.* *Return to Q23* •

A24 : *It is difficult to interpret such "busy" plots. There is very little difference for silt, somewhat for sand, but large differences for clay. The two ALR approaches seem to be somewhat better than OK and CK.* *Return to Q24* •

A25 : *The STRESS is much closer to 0 than to 1, so the predictions reproduce much of the variability.* *Return to Q25* •

A26 : *All four are quite close; the "winner" is OK.* *Return to Q26* •

References

[1] J. Aitchison. *The statistical analysis of compositional data*. Chapman and Hall, London, 1986. 2

[2] J. Aitchison. *The statistical analysis of compositional data*. Blackburn Press, 2003. 1, 2

[3] J. C. Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 3

[4] J. J. de Gruijter, D. J. J. Walvoort, and P. F. M. van Gaans. Continuous soil maps – a fuzzy set approach to bridge the gap between aggregation levels of process and distribution models. *Geoderma*, 77:169–195, 1997. 53

[5] R. M. Lark and T. F. A. Bishop. Cokriging particle size fractions of the soil. *European Journal of Soil Science*, 58(3):763–774, 2007. 2, 5, 6, 35, 50, 51, 54

[6] J. A. Martin-Fernandez, R. A. Olea-Meneses, and V. Pawlowsky-Glahn. Criteria to compare estimation methods of regionalized compositions. *Mathematical Geology*, 33(8):889–909, Nov 2001. doi: 10.1023/A:1012293922142. 50

[7] I. O. A. Odeh, A. J. Todd, and J. Triantafilis. Spatial prediction of soil particle-size fractions as compositional data. *Soil Science*, 168(7):501–515, 2003. 2, 53

[8] V. Pawlowsky-Glahn and R. A. Olea. *Geostatistical analysis of compositional data*. Oxford University Press, New York, 2004. 2, 50

[9] K. Gerald van den Boogaart and R. Tolosana-Delgado. "compositions": A unified R package to analyze compositional data. *Computers & Geosciences*, 34(4):320–338, 2008. 2, 6

[10] K. Gerald van den Boogaart and Raimon Tolosana-Delgado. *Analyzing Compositional Data with R*. Springer, 2013. ISBN 978-3-642-36809-7. 2

[11] D. J. J. Walvoort and J. J de Gruijter. Compositional kriging: A spatial interpolation method for compositional data. *Mathematical Geology*, 33(8):951–966, 2001. 53

[12] R.C Webster and H. E. de la C. Cuanalo. Soil transect correlograms of north oxfordshire and their interpolation. *Journal of Soil Science*, 26:176–194, 1975. 3

Index of R Concepts