Tutorial and Manual for Geostatistical Analyses with the R package **georob**

Andreas Papritz

December 18, 2019

Contents

1 Summary

georob is a package for model-based Gaussian and robust analyses of geostatistical data. The software of the package performs two main tasks:

- It fits a linear model with spatially correlated errors to geostatistical data that are possibly contaminated by outliers. The coefficients of the linear model (so-called external-drift) and the parameters of the variogram model are estimated by robust or Gaussian (restricted) maximum likelihood ([RE]ML).
- It computes from a fitted model object customary and robust external drift point and block Kriging predictions.

Künsch *et al.* (2011) and Künsch *et al.* (in prep.) explain the theoretical foundations of the robust approach, and Diggle and Ribeiro (2007) is a good reference for model-based Gaussian geostatistical analyses.

This document provides a practical introduction to model-based Gaussian and robust analyses of geostatistical data. It contains a short summary of the modelling approach, illustrates the use of the software with two examples and explains in some depth selected aspects of (i) (robust) parameter estimation (ii) computing predictions by (robust) Kriging and (iii) model building.

2 Introduction

This section presents briefly

- the modelling assumptions and model parametrization,
- sketches how model parameters are estimated robustly and how robust Kriging predictions are computed, and
- summarizes the main functionality of the package.

Further information on selected aspects can be found in sections 5 and 6.

2.1 Model

We use the following model for the data $y_i = y(\boldsymbol{s}_i)$:

$$Y(\boldsymbol{s}_i) = Z(\boldsymbol{s}_i) + \varepsilon_i = \boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta} + B(\boldsymbol{s}_i) + \varepsilon_i, \tag{1}$$

where $\boldsymbol{s}_i$ denotes a data location, $Z(\boldsymbol{s}_i) = \boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta} + B(\boldsymbol{s}_i)$ is the so-called signal, $\boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta}$ is the external drift, $\{B(\boldsymbol{s})\}$ is an unobserved stationary or intrinsic Gaussian random field with zero mean, and ε_i is an *i.i.d* error from a possibly long-tailed distribution with scale parameter τ (τ^2 is usually called nugget effect). In vector form the model is written as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{B} + \boldsymbol{\varepsilon}, \tag{2}$$

where $\boldsymbol{X}$ is the model matrix with the rows $\boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}$.

The (generalized) covariance matrix of the vector of spatial Gaussian random effects $\boldsymbol{B}$ is denoted by

$$\mathrm{E}[\boldsymbol{B}\,\boldsymbol{B}^{\mathrm{T}}] = \boldsymbol{\Gamma}_\theta = \sigma_{\mathrm{n}}^2 \boldsymbol{I} + \sigma^2 \boldsymbol{V}_\alpha = \sigma_Z^2\, \boldsymbol{V}_{\alpha,\xi} = \sigma_Z^2\, ((1-\xi)\, \boldsymbol{I} + \xi\, \boldsymbol{V}_\alpha), \tag{3}$$

where σ_{n}^2 is the variance of seemingly uncorrelated micro-scale variation in $B(\boldsymbol{s})$ that cannot be resolved with the chosen sampling design, σ^2 is the variance of the captured auto-correlated variation in $B(\boldsymbol{s})$, $\sigma_Z^2 = \sigma_{\mathrm{n}}^2 + \sigma^2$ is the signal variance, and $\xi = \sigma^2/\sigma_Z^2$. To estimate both σ_{n}^2 and τ^2 (and not only their sum), one needs replicated measurements for some of the $\boldsymbol{s}_i$.
We define $\boldsymbol{V}_\alpha$ to be the matrix with elements

$$(\boldsymbol{V}_\alpha)_{ij} = \gamma_0 - \gamma(|\boldsymbol{A}\,(\boldsymbol{s}_i - \boldsymbol{s}_j)|), \tag{4}$$

where the constant γ_0 is chosen large enough so that $\boldsymbol{V}_\alpha$ is positive definite, $\gamma(\cdot)$ is a valid stationary or intrinsic variogram, and $\boldsymbol{A}$ is a matrix that is used to model geometrically anisotropic auto-correlation (see section 5.3).
Two remarks are in order:

1. Clearly, the (generalized) covariance matrix of the observations $\boldsymbol{Y}$ is given by

$$\mathrm{Cov}[\boldsymbol{Y}, \boldsymbol{Y}^{\mathrm{T}}] = \tau^2 \boldsymbol{I} + \boldsymbol{\Gamma}_\theta. \tag{5}$$

2. Depending on the context, the term "variogram parameters" denotes sometimes all parameters of a geometrically anisotropic variogram model, but in places only the parameters of an isotropic variogram model, i.e. $\sigma^2, \ldots, \alpha, \ldots$ and $f_1, \ldots, \zeta$ are denoted by the term "anisotropy parameters". In the sequel $\boldsymbol{\theta}$ is used to denote all variogram and anisotropy parameters except the nugget effect τ^2.

2.2 Estimation

The unobserved spatial random effects $\boldsymbol{B}$ at the data locations $\boldsymbol{s}_i$ and the model parameters $\boldsymbol{\beta}$, τ^2 and $\boldsymbol{\theta}^{\mathrm{T}} = (\sigma^2, \sigma_{\mathrm{n}}^2, \alpha, \ldots, f_1, f_2, \omega, \phi, \zeta)$ are unknown and are estimated in **georob** either by Gaussian or robust restricted maximum likelihood (REML) or Gaussian maximum likelihood (ML). Here ... denote further parameters of the variogram such as the smoothness parameter of the Whittle-Matérn model.
In brief, the robust REML method is based on the insight that for given $\boldsymbol{\theta}$ and τ^2 the Kriging predictions (= BLUP) of $\boldsymbol{B}$ and the generalized least squares (GLS = ML) estimates of $\boldsymbol{\beta}$ can be obtained simultaneously by maximizing

$$-\sum_i \left(\frac{y_i - \boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta} - B(\boldsymbol{s}_i)}{\tau} \right)^2 - \boldsymbol{B}^{\mathrm{T}} \boldsymbol{\Gamma}_\theta^{-1} \boldsymbol{B}$$

with respect to $\boldsymbol{B}$ and $\boldsymbol{\beta}$, e.g. Harville (1977).
Hence, the BLUP of $\boldsymbol{B}$, ML estimates of $\boldsymbol{\beta}$, $\boldsymbol{\theta}$ and τ^2 are obtained by maximizing

$$-\log(\det(\tau^2 \boldsymbol{I} + \boldsymbol{\Gamma}_\theta)) - \sum_i \left(\frac{y_i - \boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta} - B(\boldsymbol{s}_i)}{\tau} \right)^2 - \boldsymbol{B}^{\mathrm{T}} \boldsymbol{\Gamma}_\theta^{-1} \boldsymbol{B} \tag{6}$$

jointly with respect to $\boldsymbol{B}$, $\boldsymbol{\beta}$, $\boldsymbol{\theta}$ and τ^2 or by solving the respective estimating equations.
The estimating equations can then by robustified by

- replacing the standardized errors, say ε_i/τ, by a bounded or re-descending ψ-function, $\psi_c(\varepsilon_i/\tau)$, of them (e.g. Maronna *et al.*, 2006, chap. 2) and by
- introducing suitable bias correction terms for Fisher consistency at the Gaussian model,

see Künsch *et al.* (2011) for details. The robustified estimating equations are solved numerically by a combination of iterated re-weighted least squares (IRWLS) to estimate $\boldsymbol{B}$ and $\boldsymbol{\beta}$ for given $\boldsymbol{\theta}$ and τ^2 and non-linear root finding by the function `nleqslv()` of the R package **nleqslv** to get $\boldsymbol{\theta}$ and τ^2. The robustness of the procedure is controlled by the tuning parameter c of the ψ_c-function. For $c \geq 1000$ the algorithm computes Gaussian (RE)ML estimates and customary plug-in Kriging predictions. Instead of solving the Gaussian (RE)ML estimating equations, our software then maximizes the Gaussian (restricted) log-likelihood using `nlminb()` or `optim()`.

georob uses variogram models implemented in the R package **RandomFields** (see `RMmodel()`). For most variogram parameters, closed-form expressions of $\partial\gamma/\partial\theta_i$ are used in the computations. However, for the parameter ν of the models `"RMbessel"`, `"RMmatern"` and `"RMwhittle"` $\partial\gamma/\partial\nu$ is evaluated numerically by the function `numericDeriv()`, and this results in an increase in computing time when ν is estimated.

2.3 Prediction

Robust plug-in external drift point Kriging predictions can be computed for an unsampled location $\boldsymbol{s}_0$ from the covariates $\boldsymbol{x}(\boldsymbol{s}_0)$, the estimated parameters $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\theta}}$ and the predicted random effects $\widehat{\boldsymbol{B}}$ by

$$\widehat{Y}(\boldsymbol{s}_0) = \widehat{Z}(\boldsymbol{s}_0) = \boldsymbol{x}(\boldsymbol{s}_0)^{\mathrm{T}}\widehat{\boldsymbol{\beta}} + \boldsymbol{\gamma}_{\widehat{\theta}}^{\mathrm{T}}(\boldsymbol{s}_0)\boldsymbol{\Gamma}_{\widehat{\theta}}^{-1}\widehat{\boldsymbol{B}}, \tag{7}$$

where $\boldsymbol{\Gamma}_{\widehat{\theta}}$ is the estimated (generalized) covariance matrix of $\boldsymbol{B}$ and $\boldsymbol{\gamma}_{\widehat{\theta}}(\boldsymbol{s}_0)$ is the vector with the estimated (generalized) covariances between $\boldsymbol{B}$ and $B(\boldsymbol{s}_0)$. Kriging variances can be computed as well, based on approximated covariances of $\widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\beta}}$ (see Künsch *et al.*, 2011, and appendices of Nussbaum *et al.*, 2012, 2014, for details).
The package **georob** provides in addition software for computing robust external drift *block* Kriging predictions. The required integrals of the (generalized) covariance function are computed by functions of the R package **constrainedKriging** (Hofer and Papritz, 2011).

2.4 Functionality

For the time being, the functionality of **georob** is limited to robust geostatistical analyses of *single* response variables. No software is currently available for robust multivariate geostatistical analyses. **georob** offers functions for:

1. Robustly fitting a spatial linear model to data that are possibly contaminated by independent errors from a long-tailed distribution by robust REML (see functions `georob()` — which also fits such models efficiently by Gaussian (RE)ML — `profilelogLik()` and `control.georob()`).
2. Extracting estimated model components (see `residuals.georob()`, `rstandard.georob()`, `ranef.georob()`).

3. Robustly estimating sample variograms and for fitting variogram model functions to them (see `sample.variogram()` and `fit.variogram.model()`).

4. Model building by forward and backward selection of covariates for the external drift (see `waldtest.georob()`, `step.georob()`, `add1.georob()`, `drop1.georob()`, `extractAIC.georob()`, `logLik.georob()`, `deviance.georob()`). For a robust fit, the log-likelihood is not defined. The function then computes the (restricted) log-likelihood of an equivalent Gaussian model with heteroscedastic nugget (see `deviance.georob()` for details).

5. Assessing the goodness-of-fit and predictive power of the model by K-fold cross-validation (see `cv.georob()` and `validate.predictions()`).

6. Computing robust external drift point and block Kriging predictions (see `predict.georob()`, `control.predict.georob()`).

7. Unbiased back-transformation of both point and block Kriging predictions of log-transformed data to the original scale of the measurements (see `lgnpp()`).

3 Model-based Gaussian analysis of `zinc`, data set `meuse`

The package **sp** provides this data set. According to the help page, it "gives locations and topsoil heavy metal concentrations, along with a number of soil and landscape variable at the observation locations, collected in a flood plain of the river Meuse, near the village of Stein (NL)".

```
> data(meuse, package="sp")
> levels(meuse$ffreq) <- paste("ffreq", levels(meuse$ffreq), sep="")
> levels(meuse$soil) <- paste("soil", levels(meuse$soil), sep="")
> str(meuse)

'data.frame':        155 obs. of  14 variables:
 $ x       : num  181072 181025 181165 181298 181307 ...
 $ y       : num  333611 333558 333537 333484 333330 ...
 $ cadmium: num  11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
 $ copper : num  85 81 68 81 48 61 31 29 37 24 ...
 $ lead    : num  299 277 199 116 117 137 132 150 133 80 ...
 $ zinc    : num  1022 1141 640 257 269 ...
 $ elev    : num  7.91 6.98 7.8 7.66 7.48 ...
 $ dist    : num  0.00136 0.01222 0.10303 0.19009 0.27709 ...
 $ om      : num  13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
 $ ffreq   : Factor w/ 3 levels "ffreq1","ffreq2",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ soil    : Factor w/ 3 levels "soil1","soil2",..: 1 1 1 2 2 2 2 1 1 2 ...
 $ lime    : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
 $ landuse: Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15 ...
 $ dist.m : num  50 30 150 270 380 470 240 120 240 420 ...
```

Bivand *et al.* (2013) use the data to illustrate geostatistical analyses by the package **gstat** (Pebesma, 2004). We analyse here the data on `zinc` in the topsoil (Figure 1).

3.1 Exploratory analysis

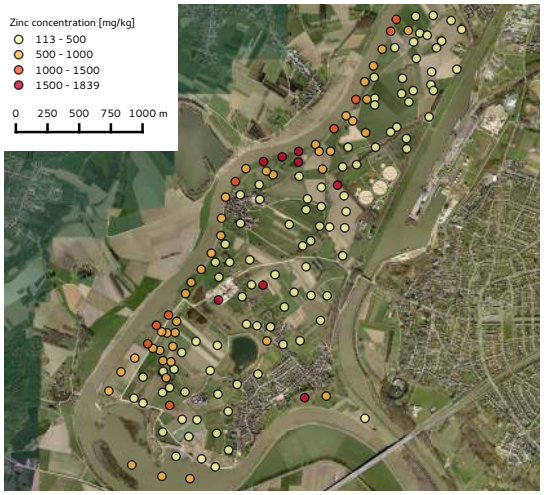

Figure 1: Meuse data set: zinc concentration at 155 locations in floodplain of river Meuse near Stein (NL) shown in Google Earth™.

Figure 1 suggests that `zinc` concentration depends on `dist`ance to the river. We check graphically whether the two factors `ffreq` (frequency of flooding) and `soil` (type) also influence `zinc`:

```
> library(lattice)
> palette(trellis.par.get("superpose.symbol")$col)
> plot(zinc~dist, meuse, pch=as.integer(ffreq), col=soil)
> legend("topright", col=c(rep(1, nlevels(meuse$ffreq)), 1:nlevels(meuse$soil)),
+   pch=c(1:nlevels(meuse$ffreq), rep(1, nlevels(meuse$soil))), bty="n",
+   legend=c(levels(meuse$ffreq), levels(meuse$soil)))
```

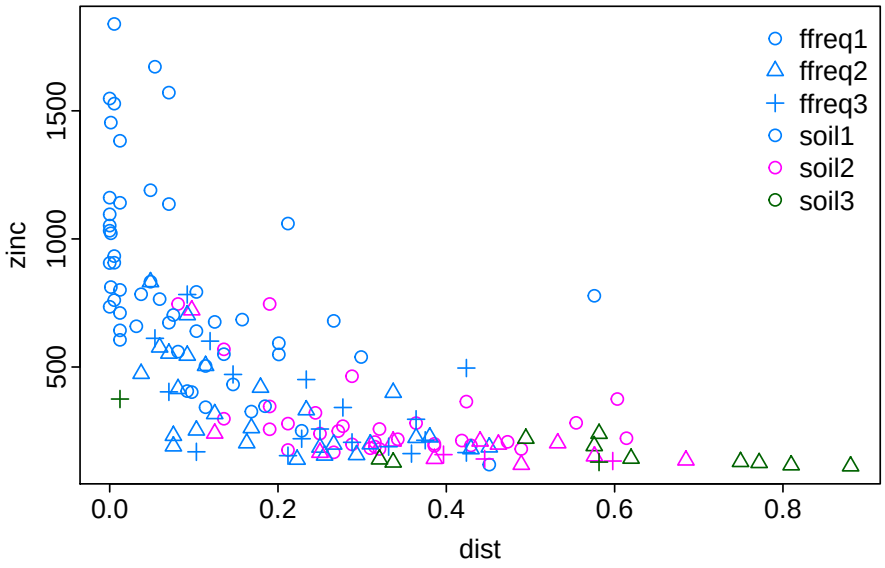

Figure 2: Dependence of `zinc` concentration on `dist`ance to river, frequency of flooding (`ffreq`) and `soil` type.

`zinc` depends non-linearly on `dist` and seems in addition to depend on `ffreq` (larger concentration at more often flooded sites). Furthermore, the scatter of `zinc` for given distance increases with decreasing distance (= increasing `zinc` concentration, heteroscedastic variation). We use `log(zinc)` to stabilize the variance:

```
> xyplot(log(zinc)~dist | ffreq, meuse, groups=soil, panel=function(x, y, ...){
+     panel.xyplot(x, y, ...)
+     panel.loess(x, y, ...)
+   }, auto.key=TRUE)
```

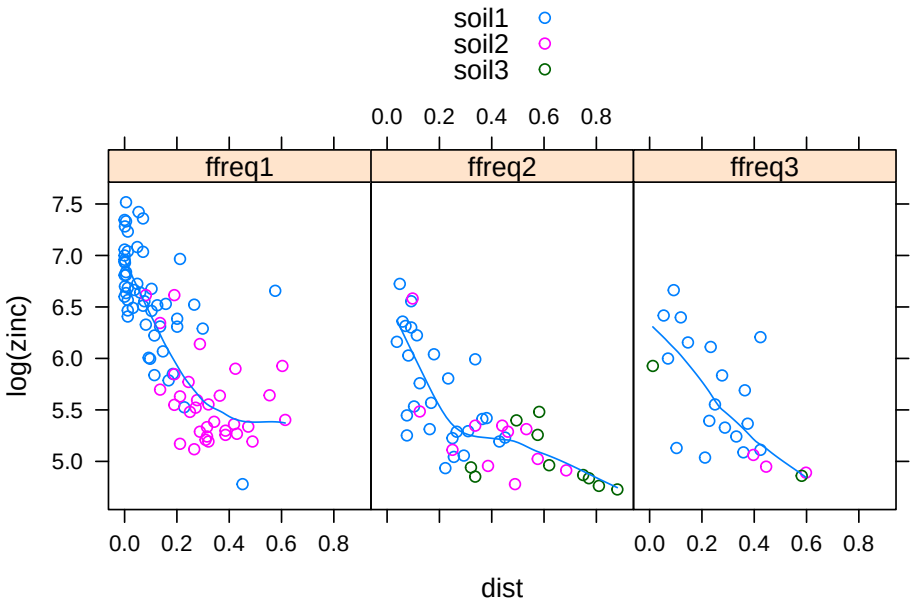

Figure 3: Dependence of `zinc` on `dist`ance to river, frequency of flooding (`ffreq`) and `soil` type.

The relation `log(zinc)~dist` is still non-linear, hence we transform `dist` by $\sqrt{\ }$:

```
> xyplot(log(zinc)~sqrt(dist) | ffreq, meuse, groups=soil, panel=function(x, y, ...){
+     panel.xyplot(x, y, ...)
+     panel.loess(x, y, ...)
+     panel.lmline(x, y, lty="dashed", ...)
+   }, auto.key=TRUE)
```

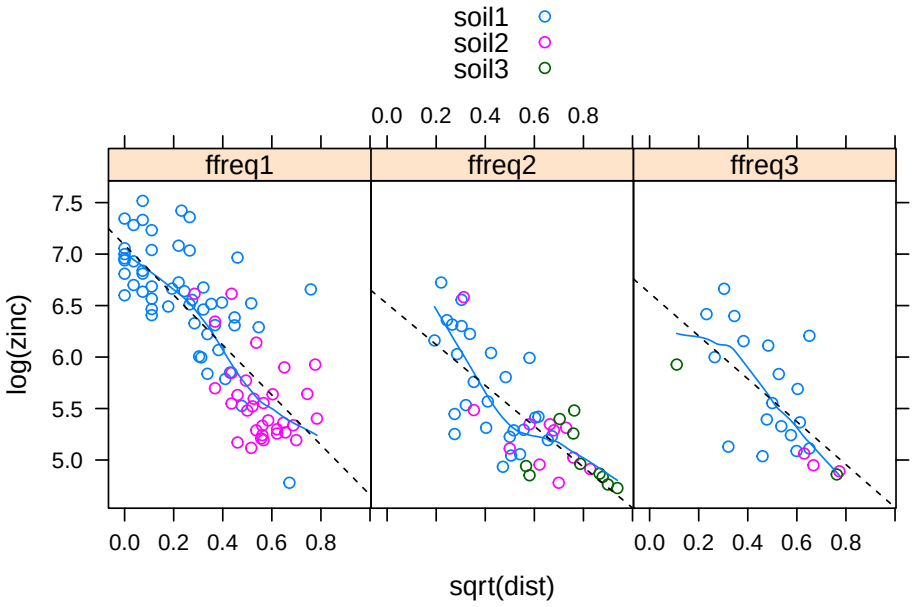

Figure 4: Dependence of `zinc` concentration on `dist`ance to river, frequency of flooding (`ffreq`) and `soil` type.

which approximately linearizes the relation.

The slopes of the regression lines `log(zinc)~sqrt(dist)` are about the same for all levels of `ffreq`. But the intercept of `ffreq1` differs from the intercepts of the other levels. Hence, as an initial drift model we use

```
> r.lm <- lm(log(zinc)~sqrt(dist)+ffreq, meuse)
> summary(r.lm)


Call:
lm(formula = log(zinc) ~ sqrt(dist) + ffreq, data = meuse)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8559 -0.3084 -0.0304  0.2957  1.3465

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.0299     0.0714   98.44  < 2e-16 ***
sqrt(dist)   -2.2660     0.1559  -14.54  < 2e-16 ***
ffreqffreq2  -0.3605     0.0786   -4.58  9.5e-06 ***
ffreqffreq3  -0.3167     0.0982   -3.23   0.0015 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.407 on 151 degrees of freedom
Multiple R-squared:  0.689,        Adjusted R-squared:  0.683
F-statistic:  111 on 3 and 151 DF,  p-value: <2e-16
```

The residual diagnostic plots

```
> op <- par(mfrow=c(2, 2)); plot(r.lm); par(op)
```

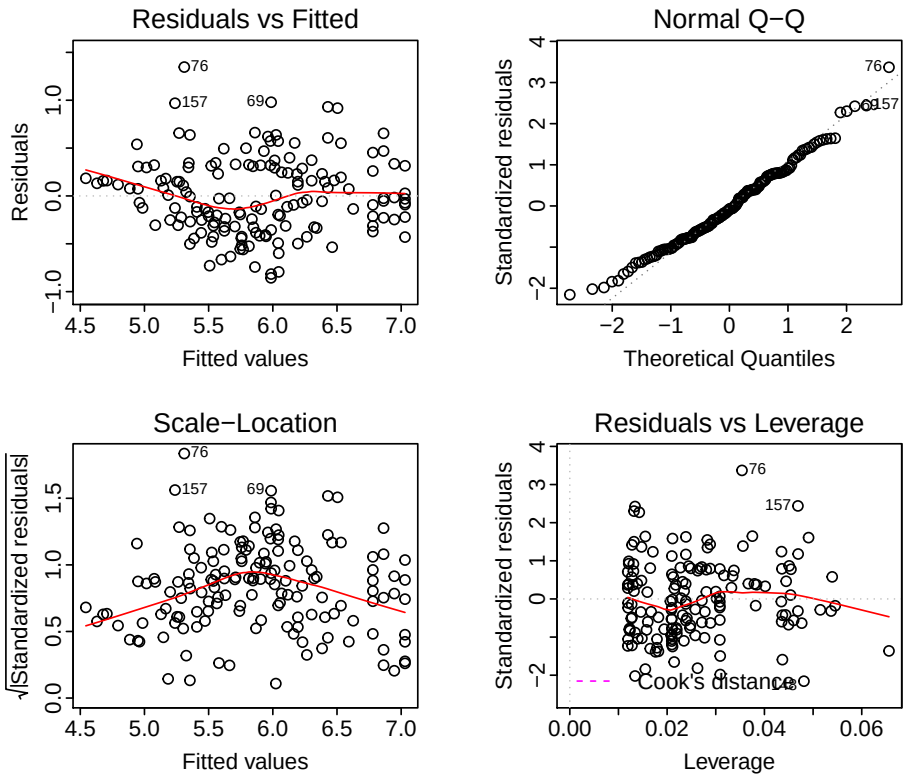

Figure 5: Residual diagnostic plots for linear drift model `log(zinc)~sqrt(dist)+ffreq`.

do not show violations of modelling assumptions.

Next, we compute the sample variogram of the residuals for the 4 directions N-S, NE-SW, E-W, SE-NW by the methods-of-moments estimator:

```
> library(georob)
> plot(sample.variogram(residuals(r.lm), locations=meuse[, c("x","y")],
+   lag.dist.def=100, max.lag=2000, xy.angle.def=c(0, 22.5, 67.5, 112.5, 157.5, 180),
+   estimator="matheron"), type="l",
+   main="sample variogram of residuals log(zinc)~sqrt(dist)+ffreq")
```

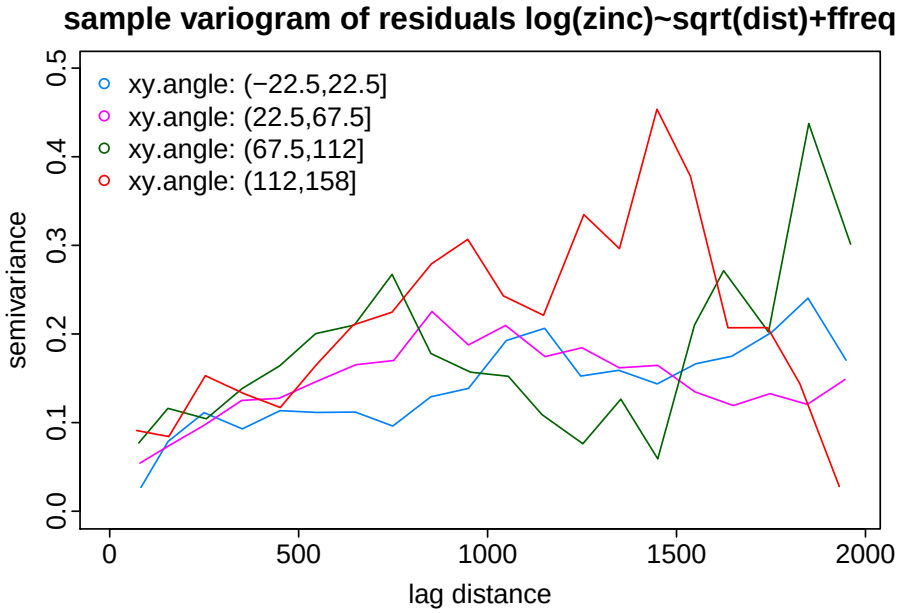

Figure 6: Direction-dependent sample variogram of regression residuals of `log(zinc)~sqrt(dist)+ffreq`.

The residuals appear to be spatially dependent. For the short lags there is no clear dependence on direction, hence, we assume that auto-correlation is isotropic.

To complete the exploratory modelling exercise, we compute the direction-indepdendent sample variogram and fit a spherical variogram model by weighted non-linear least squares (using Cressie's weights)

```
> library(georob)
> plot(r.sv <- sample.variogram(residuals(r.lm), locations=meuse[, c("x","y")],
+   lag.dist.def=100, max.lag=2000,
+   estimator="matheron"), type="l",
+   main="sample variogram of residuals log(zinc)~sqrt(dist)+ffreq")
> lines(r.sv.spher <- fit.variogram.model(r.sv, variogram.mode="RMspheric",
+   param=c(variance=0.1, nugget=0.05, scale=1000)))
```

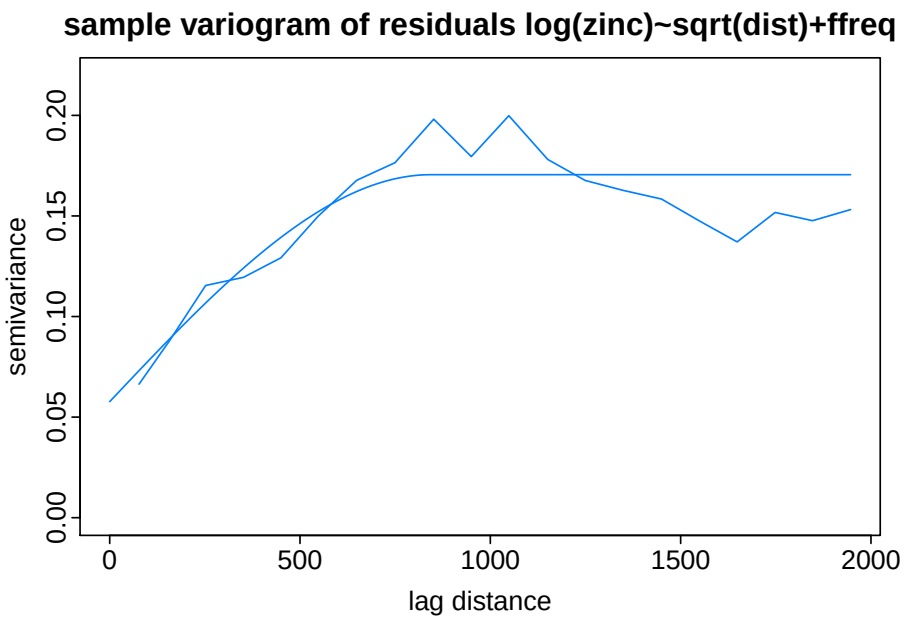

Figure 7: Sample variogram of regression residuals of `log(zinc)~sqrt(dist)+ffreq` along with fitted spherical variogram function.

and output the fitted variogram parameters

```
> summary(r.sv.spher)


Call:fit.variogram.model(sv = r.sv, variogram.model = "RMspheric",
    param = c(variance = 0.1, nugget = 0.05, scale = 1000))

Convergence in 18 function and 42 Jacobian/gradient evaluations

Residual Sum of Squares: 78.945

Residuals (epsilon):
     Min       1Q   Median       3Q      Max
-0.03342 -0.01342 -0.00374  0.00768  0.02932

Variogram:  RMspheric
               Estimate    Lower  Upper
variance         0.1128   0.1036   0.12
snugget(fixed)   0.0000       NA     NA
nugget           0.0577   0.0490   0.07
scale          844.2342 767.1050 929.12
```

3.2 Fitting a spatial linear model by Gaussian (RE)ML

We fit the model that we developed in the exploratory analysis now by Gaussian REML:

```
> r.georob.m0.spher.reml <- georob(log(zinc)~sqrt(dist)+ffreq, meuse, locations=~x+y,
+   variogram.model="RMspheric", param=c(variance=0.1, nugget=0.05, scale=1000),
+   tuning.psi=1000)


> summary(r.georob.m0.spher.reml)


Call:georob(formula = log(zinc) ~ sqrt(dist) + ffreq, data = meuse,
    locations = ~x + y, variogram.model = "RMspheric", param = c(variance = 0.1,
        nugget = 0.05, scale = 1000), tuning.psi = 1000)

Tuning constant:  1000

Convergence in 12 function and 9 Jacobian/gradient evaluations

Estimating equations (gradient)

                                  eta         scale
  Gradient           :   -6.5980e-03   -1.1617e-01

Maximized restricted log-likelihood: -54.584

Predicted latent variable (B):
    Min      1Q  Median      3Q     Max
-0.6422 -0.3020 -0.0158  0.1799  0.6099

Residuals (epsilon):
     Min       1Q   Median       3Q      Max
-0.62747 -0.11035 -0.00102  0.10224  0.59397

Standardized residuals:
     Min       1Q   Median       3Q      Max
-3.60760 -0.60304 -0.00571  0.58372  3.49971


Gaussian REML estimates

Variogram:  RMspheric
               Estimate    Lower   Upper
variance         0.1349   0.0677    0.27
snugget(fixed)   0.0000       NA      NA
nugget           0.0551   0.0327    0.09
scale          876.5812 746.9212 1028.75


Fixed effects coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.0889     0.1391   50.96  < 2e-16 ***
sqrt(dist)   -2.1319     0.2590   -8.23  8.1e-14 ***
ffreqffreq2  -0.5268     0.0689   -7.64  2.3e-12 ***
ffreqffreq3  -0.5383     0.1040   -5.17  7.2e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error (sqrt(nugget)): 0.235

Robustness weights:
 All 155 weights are ~= 1.
```

The diagnostics at the begin of the `summary` output suggest that maximization of the restricted log-likelihood by `nlminb()` was successful. Nevertheless, before we interprete the output, we compute the profile log-likelihood for the range to see whether the maximization has found the global maximum:

```
> r.prfl.m0.spher.reml.scale <- profilelogLik(r.georob.m0.spher.reml,
+   values=data.frame(scale=seq(500, 5000, by=50)))


> plot(loglik~scale, r.prfl.m0.spher.reml.scale, type="l")
> abline(v=coef(r.georob.m0.spher.reml, "variogram")["scale"], lty="dashed")
> abline(h=r.georob.m0.spher.reml$loglik - 0.5*qchisq(0.95, 1), lty="dotted")
```

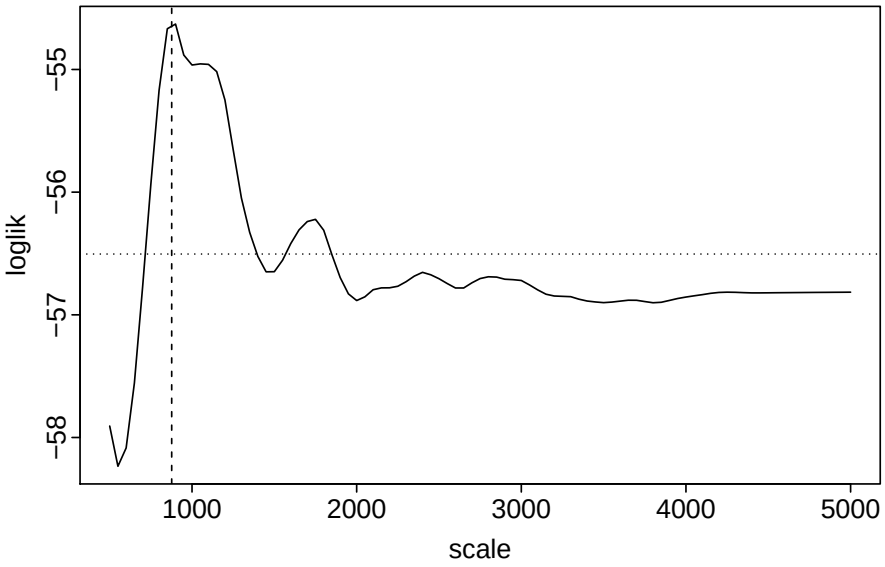

Figure 8: Restricted profile log-likelihood for range parameter (`scale`) of spherical variogram (vertical line: estimate of `scale` returned by `georob()`; intersection of horizontal line with profile defines a 95% confidence region for `scale` based on likelihood ratio test).

Although the restricted log-likelihood is multimodal — which is often observed for variogram models with compact support — we were lucky to find the global maximum because the initial values of the variogram parameters were close to the REML estimates. Estimates of `scale` (range of variogram) and `variance` (partial sill) are correlated, `nugget` and `scale` less so:

```
> op <- par(mfrow=c(1,2), cex=0.66)
> plot(variance~scale, r.prfl.m0.spher.reml.scale, ylim=c(0, max(variance)), type="l")
> plot(nugget~scale, r.prfl.m0.spher.reml.scale, ylim=c(0, max(nugget)), type="l")
> par(op)
```

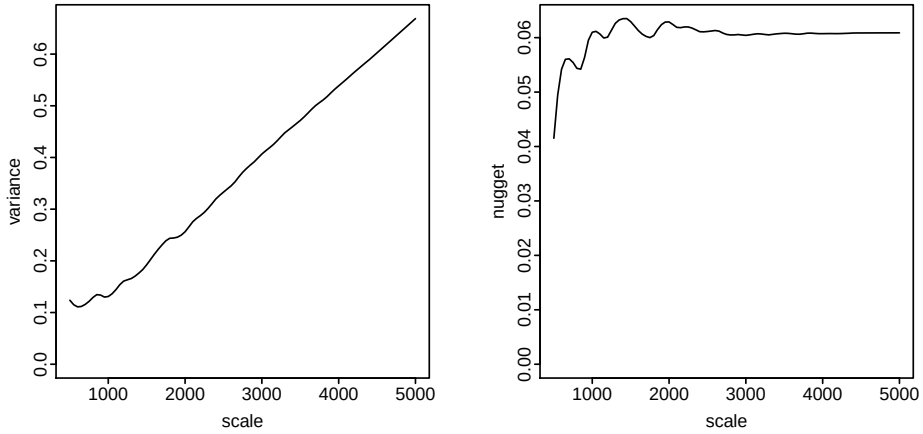

Figure 9: Estimates of `variance` (partial sill) and `nugget` as a function of the estimate of the range (`scale`) of the variogram.

We now study the `summary` output in detail: The estimated variogram parameters are reported along with 95% confidence intervals that are computed based on the asymptotic normal distribution of (RE)ML estimates from the observed Fisher information.
The dependence of `log(zinc)` on `sqrt(dist)` is highly significant, as is the dependence on `ffreq`:

```
> waldtest(r.georob.m0.spher.reml, .~.-ffreq)


Wald test

Model 1: log(zinc) ~ sqrt(dist) + ffreq
Model 2: log(zinc) ~ sqrt(dist)
  Res.Df Df    F  Pr(>F)
1    151
2    153 -2 31.2 4.6e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can test equality of all pairs of intercepts by functions of the package **multcomp**

```
> library(multcomp)
> summary(glht(r.georob.m0.spher.reml,
+   linfct = mcp(ffreq = c("ffreq1 - ffreq2 = 0", "ffreq1 - ffreq3 = 0",
+   "ffreq2 - ffreq3 = 0"))))


         Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: User-defined Contrasts


Fit: georob(formula = log(zinc) ~ sqrt(dist) + ffreq, data = meuse,
    locations = ~x + y, variogram.model = "RMspheric", param = c(variance = 0.1,
        nugget = 0.05, scale = 1000), tuning.psi = 1000)

Linear Hypotheses:
                     Estimate Std. Error z value Pr(>|z|)
```

```
ffreq1 - ffreq2 == 0   0.5268     0.0689    7.64   <1e-06 ***
ffreq1 - ffreq3 == 0   0.5383     0.1040    5.17   <1e-06 ***
ffreq2 - ffreq3 == 0   0.0115     0.0960    0.12     0.99
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

As suspected only the intercept of `ffreq1` differs from the others. Adding the interaction `sqrt(dist):ffreq` does not improve the model:

```
> waldtest(r.georob.m0.spher.reml, .~.+sqrt(dist):ffreq)


Wald test

Model 1: log(zinc) ~ sqrt(dist) + ffreq
Model 2: log(zinc) ~ sqrt(dist) + ffreq + sqrt(dist):ffreq
  Res.Df Df    F Pr(>F)
1    151
2    149  2 0.68   0.51
```

nor does adding `soil`:

```
> waldtest(r.georob.m0.spher.reml, .~.+soil)


Wald test

Model 1: log(zinc) ~ sqrt(dist) + ffreq
Model 2: log(zinc) ~ sqrt(dist) + ffreq + soil
  Res.Df Df    F Pr(>F)
1    151
2    149  2 2.27   0.11
```

Drift models may also be build by step-wise covariate selection based on AIC, either keeping the variogram parameters fixed (default)

```
> step(r.georob.m0.spher.reml, scope=log(zinc)~ffreq*sqrt(dist)+soil)


Start:  AIC=106.91
log(zinc) ~ sqrt(dist) + ffreq

                   Df AIC Converged
<none>                107
+ soil              2 107         1
+ ffreq:sqrt(dist)  2 110         1
- ffreq             2 166         1
- sqrt(dist)        1 178         1

Tuning constant:  1000

Fixed effects coefficients:
(Intercept)   sqrt(dist)  ffreqffreq2  ffreqffreq3
      7.094       -2.146       -0.526       -0.537

Variogram:  RMspheric
 variance(fixed)    snugget(fixed)     nugget(fixed)      scale(fixed)
           0.123             0.000             0.056           872.400
```

or re-estimating them afresh for each evaluated model

```
> step(r.georob.m0.spher.reml, scope=log(zinc)~ffreq*sqrt(dist)+soil,
+   fixed.add1.drop1=FALSE)


Start:  AIC=112.91
log(zinc) ~ sqrt(dist) + ffreq

                   Df AIC Converged
<none>                113         1
+ soil              2 113         1
+ ffreq:sqrt(dist)  2 116         1
- sqrt(dist)        1 144         1
- ffreq             2 158         1

Tuning constant:  1000

Fixed effects coefficients:
(Intercept)   sqrt(dist)  ffreqffreq2  ffreqffreq3
      7.094       -2.146       -0.526       -0.537

Variogram:  RMspheric
        variance    snugget(fixed)            nugget             scale
           0.123             0.000             0.056           872.400
```

which selects the same model. Note that step-wise covariate selection by `step.georob()`, `add1.georob()` and `drop1.georob()` requires the *non-restricted* log-likelihood. Before evaluating candidate models, the initial model is therefore re-fitted by ML, which can be done by

```
> r.georob.m0.spher.ml <- update(r.georob.m0.spher.reml,
+   control=control.georob(ml.method="ML"))


> extractAIC(r.georob.m0.spher.reml, REML=TRUE)


[1]   7.00 123.17


> extractAIC(r.georob.m0.spher.ml)


[1]   7.00 112.91


> r.georob.m0.spher.ml


Tuning constant:  1000

Fixed effects coefficients:
(Intercept)   sqrt(dist)  ffreqffreq2  ffreqffreq3
      7.094       -2.146       -0.526       -0.537

Variogram:  RMspheric
        variance    snugget(fixed)            nugget             scale
           0.123             0.000             0.056           872.400
```

Models can be also compared by cross-validation

```
> r.cv.m0.spher.reml <- cv(r.georob.m0.spher.reml, seed=3245, lgn=TRUE)
> r.georob.m1.spher.reml <- update(r.georob.m0.spher.reml, .~.-ffreq)
> r.cv.m1.spher.reml <- cv(r.georob.m1.spher.reml, seed=3245, lgn=TRUE)


> summary(r.cv.m0.spher.reml)


Statistics of cross-validation prediction errors
     me     mede     rmse     made      qne     msse   medsse     crps
 0.0126  -0.0128   0.3501   0.3017   0.3161   0.8697   0.2561   0.1944

Statistics of back-transformed cross-validation prediction errors
     me     mede     rmse     made      qne     msse   medsse
 -7.172  -23.516  191.051  110.502  123.549    0.958    0.201


> summary(r.cv.m1.spher.reml)


Statistics of cross-validation prediction errors
    me    mede    rmse    made     qne    msse  medsse    crps
0.0435  0.0340  0.4272  0.3824  0.3941  0.9982  0.3778  0.2352

Statistics of back-transformed cross-validation prediction errors
    me    mede    rmse    made     qne    msse  medsse
 20.05  -25.02  221.19  129.42  139.00    1.52    0.27
```

Note that the argument `lng=TRUE` has the effect that the cross-validation predictions of a log-transformed response are transformed back to the original scale of the measurements.

```
> op <- par(mfrow=c(3,2))
> plot(r.cv.m1.spher.reml, "sc")
> plot(r.cv.m0.spher.reml, "sc", add=TRUE, col=2)
> abline(0, 1, lty="dotted")
> legend("topleft", pch=1, col=1:2, bty="n",
+   legend=c("log(zinc)~sqrt(dist)", "log(zinc)~sqrt(dist)+ffreq"))
> plot(r.cv.m1.spher.reml, "lgn.sc"); plot(r.cv.m0.spher.reml, "lgn.sc", add=TRUE, col=2)
> abline(0, 1, lty="dotted")
> plot(r.cv.m1.spher.reml, "hist.pit")
> plot(r.cv.m0.spher.reml, "hist.pit", col=2)
> plot(r.cv.m1.spher.reml, "ecdf.pit")
> plot(r.cv.m0.spher.reml, "ecdf.pit", add=TRUE, col=2)
> abline(0, 1, lty="dotted")
> plot(r.cv.m1.spher.reml, "bs")
> plot(r.cv.m0.spher.reml, add=TRUE, "bs", col=2)
> par(op)
```

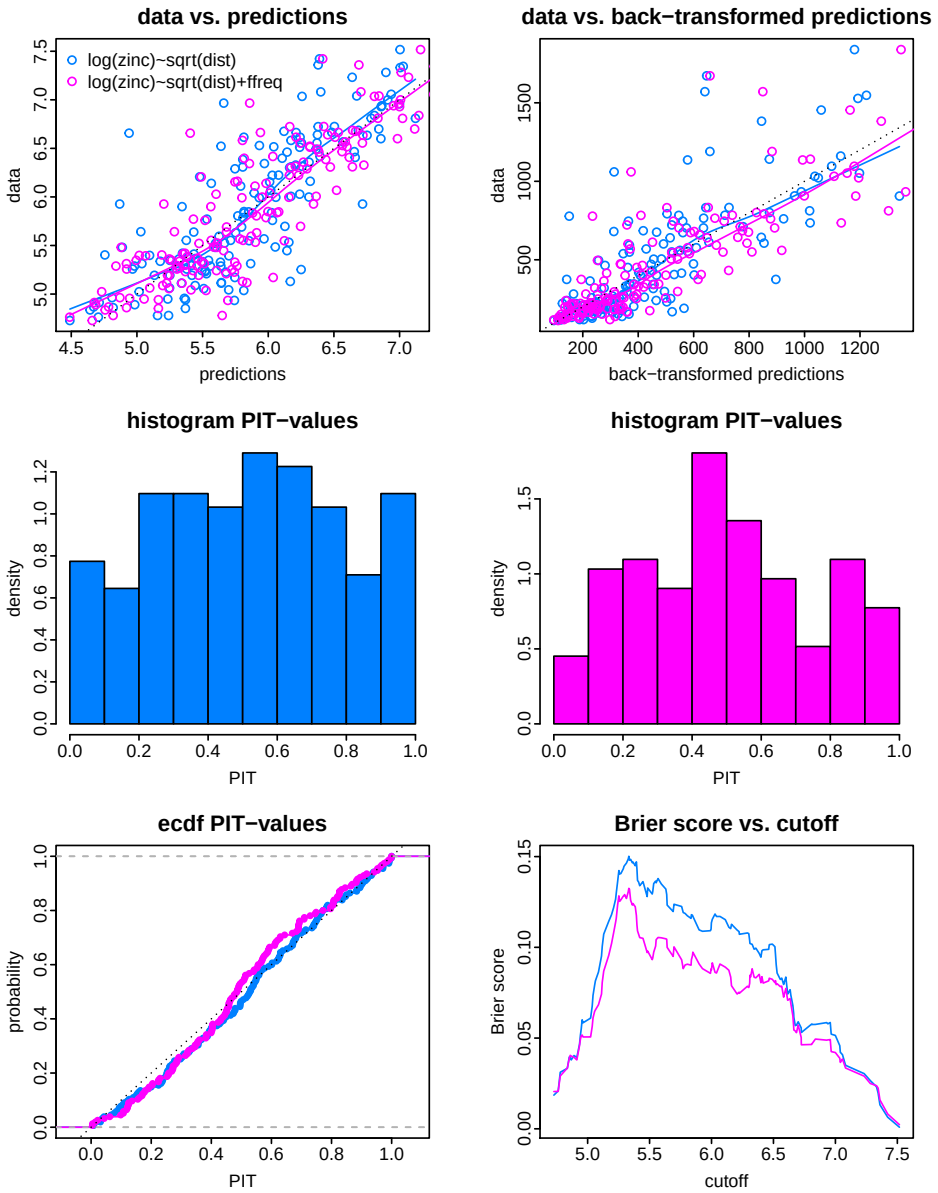

Figure 10: Diagnostic plots of cross-validation predictions of REML fits of models `log(zinc)~sqrt(dist)` (blue) and `log(zinc)~sqrt(dist)+ffreq` (magenta).

The simpler model gives less precise predictions (larger `rmse`, Brier score and therefore also larger continuous ranked probability score `crps`), but it models prediction uncertainty better (PIT closer to uniform distribution, see section 7.3 and Gneiting *et al.*, 2007).
We finish modelling by plotting residual diagnostics of the model `r.georob.m0.spher.reml` and comparing the estimated variogram with the ML estimate and the model fitted previously to the sample variogram of ordinary least squares (OLS) residuals:

```
> op <- par(mfrow=c(2,2), cex=0.66)
> plot(r.georob.m0.spher.reml, "ta"); abline(h=0, lty="dotted")
> plot(r.georob.m0.spher.reml, "qq.res"); abline(0, 1, lty="dotted")
> plot(r.georob.m0.spher.reml, "qq.ranef"); abline(0, 1, lty="dotted")
> plot(r.georob.m0.spher.reml, lag.dist.def=100, max.lag=2000)
> lines(r.georob.m0.spher.ml, col=2); lines(r.sv.spher, col=3)
> par(op)
```

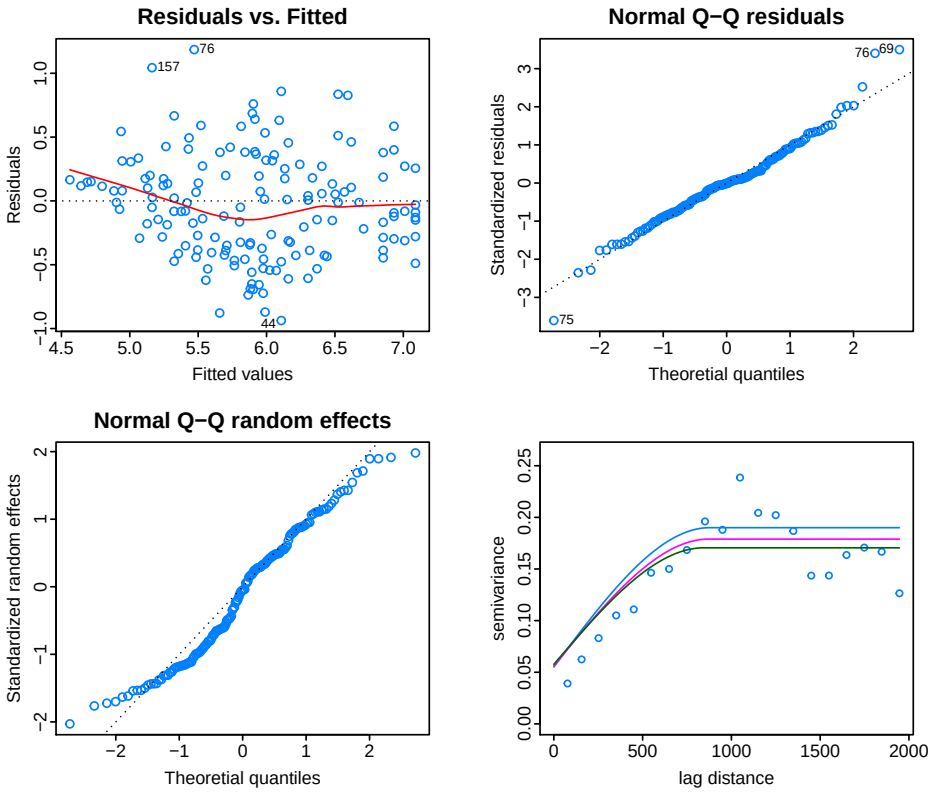

Figure 11: Residual diagnostic plots of model `log(zinc)~sqrt(dist)+ffreq` and spherical variogram estimated by REML (blue), ML (magenta) and fit to sample variogram (darkgreen).

As expected REML estimates larger range and partial sill parameters. The diagnostics plots do not reveal serious violations of modelling assumptions.

3.3 Computing Kriging predictions

3.3.1 Lognormal point Kriging

The data set `meuse.grid` (provided also by package **sp**) contains the covariates

```
> data(meuse.grid)
> levels(meuse.grid$ffreq) <- paste("ffreq", levels(meuse.grid$ffreq), sep="")
> levels(meuse.grid$soil) <- paste("soil", levels(meuse.grid$soil), sep="")
> coordinates(meuse.grid) <- ~x+y
> gridded(meuse.grid) <- TRUE
```

for computing Kriging predictions of `log(zinc)` and transforming them back to the original scale of the measurements by

```
> r.pk <- predict(r.georob.m0.spher.reml, newdata=meuse.grid,
+   control=control.predict.georob(extended.output=TRUE))
> r.pk <- lgnpp(r.pk)


> str(r.pk)
```

```
Formal class 'SpatialPixelsDataFrame' [package "sp"] with 7 slots
  ..@ data       :'data.frame':        3103 obs. of  12 variables:
  .. ..$ pred           : num [1:3103] 7.05 7.06 6.8 6.57 7.07 ...
  .. ..$ se             : num [1:3103] 0.277 0.248 0.252 0.259 0.212 ...
  .. ..$ lower          : num [1:3103] 6.51 6.57 6.3 6.06 6.65 ...
  .. ..$ upper          : num [1:3103] 7.59 7.54 7.29 7.07 7.48 ...
  .. ..$ trend          : num [1:3103] 7.09 7.09 6.85 6.64 7.09 ...
  .. ..$ var.pred       : num [1:3103] 0.0779 0.0899 0.0822 0.0757 0.1027 ...
  .. ..$ cov.pred.target: num [1:3103] 0.0681 0.0818 0.0768 0.0717 0.0963 ...
  .. ..$ var.target     : num [1:3103] 0.135 0.135 0.135 0.135 0.135 ...
  .. ..$ lgn.pred       : num [1:3103] 1189 1188 921 732 1191 ...
  .. ..$ lgn.se         : num [1:3103] 373 335 269 224 288 ...
  .. ..$ lgn.lower      : num [1:3103] 672 715 547 427 773 ...
  .. ..$ lgn.upper      : num [1:3103] 1987 1887 1469 1182 1777 ...
  .. ..- attr(*, "variogram.object")=List of 1
  .. .. ..$ :List of 9
  .. .. .. ..$ variogram.model: chr "RMspheric"
  .. .. .. ..$ param          : Named num [1:4] 0.1349 0 0.0551 876.5812
  .. .. .. .. ..- attr(*, "names")= chr [1:4] "variance" "snugget" "nugget" ""..
  .. .. .. ..$ fit.param      : Named logi [1:4] TRUE FALSE TRUE TRUE
  .. .. .. .. ..- attr(*, "names")= chr [1:4] "variance" "snugget" "nugget" ""..
  .. .. .. ..$ isotropic      : logi TRUE
  .. .. .. ..$ aniso          : Named num [1:5] 1 1 90 90 0
  .. .. .. .. ..- attr(*, "names")= chr [1:5] "f1" "f2" "omega" "phi" ...
  .. .. .. ..$ fit.aniso      : Named logi [1:5] FALSE FALSE FALSE FALSE FALSE
  .. .. .. .. ..- attr(*, "names")= chr [1:5] "f1" "f2" "omega" "phi" ...
  .. .. .. ..$ sincos         :List of 6
  .. .. .. .. ..$ co: num 6.12e-17
  .. .. .. .. ..$ so: num 1
  .. .. .. .. ..$ cp: num 6.12e-17
  .. .. .. .. ..$ sp: num 1
  .. .. .. .. ..$ cz: num 1
  .. .. .. .. ..$ sz: num 0
  .. .. .. ..$ rotmat         : num [1:2, 1:2] 1.00 -6.12e-17 6.12e-17 1.00
  .. .. .. ..$ sclmat         : Named num [1:2] 1 1
  .. .. .. .. ..- attr(*, "names")= chr [1:2] "" "f1"
  .. ..- attr(*, "type")= chr "signal"
  ..@ coords.nrs : num(0)
  ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
  .. .. ..@ cellcentre.offset: Named num [1:2] 178460 329620
  .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
  .. .. ..@ cellsize         : Named num [1:2] 40 40
  .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
  .. .. ..@ cells.dim        : Named int [1:2] 78 104
  .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
  ..@ grid.index : int [1:3103] 69 146 147 148 223 224 225 226 300 301 ...
  ..@ coords     : num [1:3103, 1:2] 181180 181140 181180 181220 181100 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:3103] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "x" "y"
  ..@ bbox       : num [1:2, 1:2] 178440 329600 181560 333760
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr NA
```

Note that

- `meuse.grid` was converted to a `SpatialPixelsDataFrame` object prior to Kriging.
- The argument `control=control.predict.georob(extended.output=TRUE)` of `predict.georob()` has the effect that all items required for the back-transformation are computed.
- The variables `lgn.`*`xxx`* in `r.pk` contain the back-transformed prediction items.

Finally, the function `spplot()` (package **sp**) allows to easily plot prediction results:

```
> brks <- c(25, 50, 75, 100, 150, 200, seq(500, 3500,by=500))
> pred <- spplot(r.pk, zcol="lgn.pred", at=brks, main="prediction")
> lwr <- spplot(r.pk, zcol="lgn.lower", at=brks, main="lower bound 95% PI")
> upr <- spplot(r.pk, zcol="lgn.upper", at=brks, main="upper bound 95% PI")
> plot(pred, position=c(0, 0, 1/3, 1), more=TRUE)
> plot(lwr, position=c(1/3, 0, 2/3, 1), more=TRUE)
> plot(upr, position=c(2/3, 0, 1, 1), more=FALSE)
```

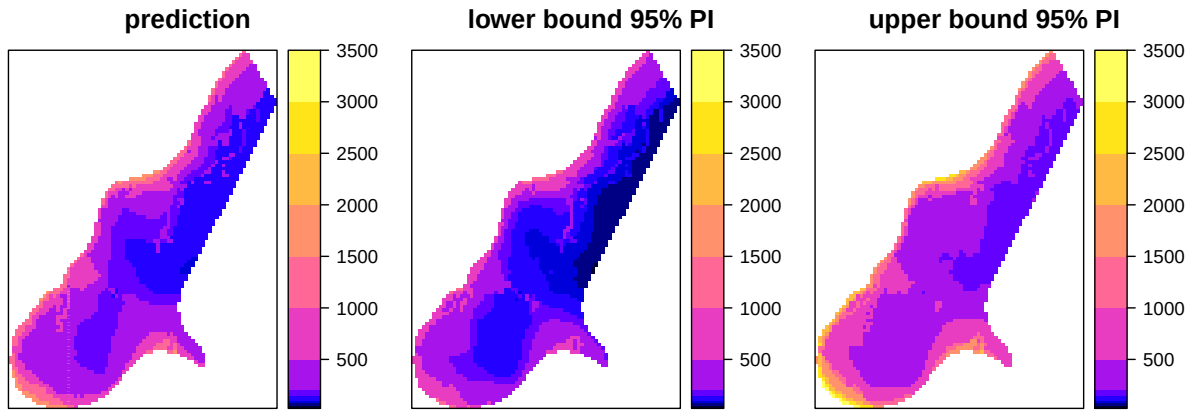

Figure 12: Lognormal point Kriging prediction of `zinc` (left: prediction; middle and right: lower and upper bounds of point-wise 95% prediction intervals).

3.3.2 Lognormal block Kriging

If `newdata` is a `SpatialPolygonsDataFrame` then `predict.georob()` computes block Kriging predictions. We illustrate this here with the `SpatialPolygonsDataFrame` `meuse.blocks` that is provided by the package **constrainedKriging**:

```
> data(meuse.blocks, package="constrainedKriging")
> str(meuse.blocks, max=2)
```

```
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        259 obs. of  2 variables:
  ..@ polygons   :List of 259
  ..@ plotOrder  : int [1:259] 177 179 180 178 188 182 181 92 51 201 ...
  ..@ bbox       : num [1:2, 1:2] 178438 329598 181562 333762
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

`meuse.blocks` contains `dist` as only covariate for the blocks, therefore, we use the model `log(zinc)~sqrt(dist)` for computing the block Kriging predictions of `log(zinc)`:

```
> r.bk <- predict(r.georob.m1.spher.reml, newdata=meuse.blocks,
+   control=control.predict.georob(extended.output=TRUE, pwidth=25, pheight=25, mmax=25))
```

and transform the predictions back by the approximately unbiased back-transformation proposed by Cressie (2006) (see section 16):

```
> r.bk <- lgnpp(r.bk, newdata=meuse.grid)
```

Note the following points:

- `pwidth` and `pheight` are the dimensions of the pixels used for efficiently computing "block-block-" and "point-block- averages" of the covariance function, see section 6.1.3).
- `mmax` controls parallelization when computing Kriging predictions, see section 6.1.4.
- `newdata` is used to pass point support covariates to `lgnpp()`, from which the spatial covariances of the covariates, needed for the back-transformation, are computed, (see equation 16 and Cressie, 2006, appendix C).

We display the prediction results again by `spplot()`:

```
> brks <- c(25, 50, 75, 100, 150, 200, seq(500, 3500,by=500))
> pred <- spplot(r.bk, zcol="lgn.pred", at=brks, main="prediction")
> lwr <- spplot(r.bk, zcol="lgn.lower", at=brks, main="lower bound 95% PI")
> upr <- spplot(r.bk, zcol="lgn.upper", at=brks, main="upper bound 95% PI")
> plot(pred, position=c(0, 0, 1/3, 1), more=TRUE)
> plot(lwr, position=c(1/3, 0, 2/3, 1), more=TRUE)
> plot(upr, position=c(2/3, 0, 1, 1), more=FALSE)
```

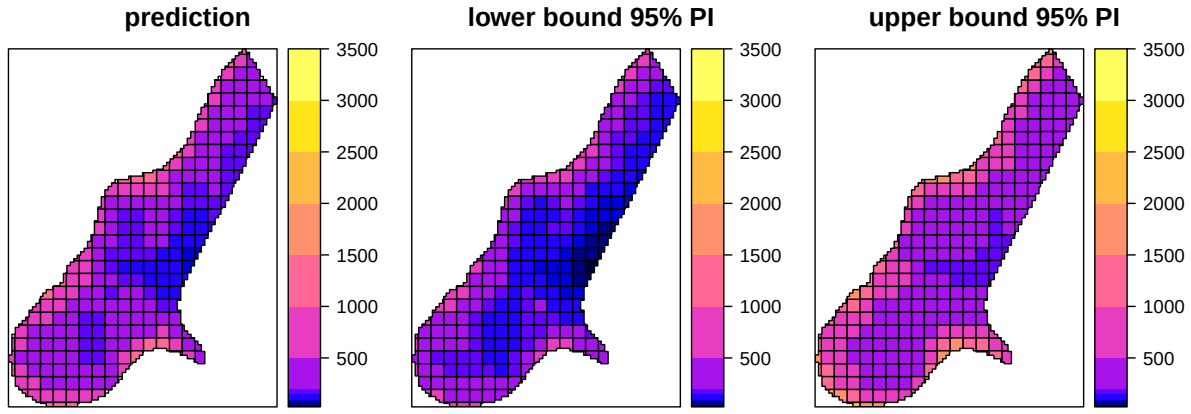

Figure 13: Lognormal block Kriging prediction of `zinc` computed under the assumption of permanence of log-normality (left: prediction; middle and right: lower and upper bounds of point-wise 95% prediction intervals).

The assumption that both point values and block means follow log-normal laws — which strictly cannot hold — does not much impair the efficiency of the back-transformation as long as the blocks are small (Cressie, 2006; Hofer *et al.*, 2013). However, for larger blocks, one should use the optimal predictor obtained by *averaging back-transformed point predictions*. `lgnpp()` allows to compute this as well. We illustrate this here by predicting the spatial mean over two larger square blocks:

```
> ## define blocks
> tmp <- data.frame(x=c(179100, 179900), y=c(330200, 331000))
> blks <- SpatialPolygons(sapply(1:nrow(tmp), function(i, x){
+   Polygons(list(Polygon(t(x[,i] + 400*t(cbind(c(-1, 1, 1, -1, -1), c(-1, -1, 1, 1, -1)))),
+   hole=FALSE)), ID=paste("block", i, sep=""))}, x=t(tmp)))
> ## compute spatial mean of sqrt(dist) for blocks
> ind <- over(as(meuse.grid, "SpatialPoints"), blks)
> tmp <- tapply(sqrt(meuse.grid$dist), ind, mean)
> names(tmp) <- paste("block", 1:length(tmp), sep="")
> ## create SpatialPolygonsDataFrame
> blks <- SpatialPolygonsDataFrame(blks, data=data.frame(dist=tmp^2))
> ## and plot
> plot(as(meuse.grid, "SpatialPoints"), axes=TRUE)
> plot(geometry(blks), add=TRUE, col=2)
```

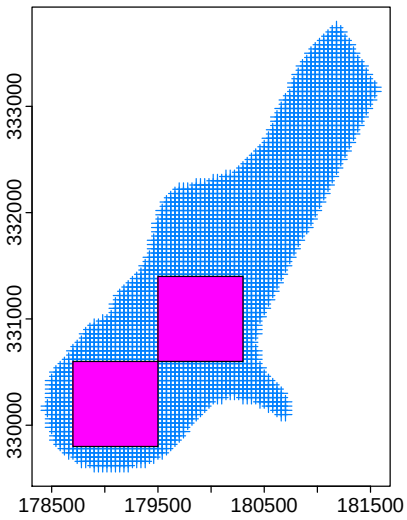

Figure 14: Point prediction grid and position of 2 target blocks for computing optimal log-normal block predictor.

We first compute block-Kriging predictions of `log(zinc)` and back-transform them as before under the permanence of log-normality assumption:

```
> r.blks <- predict(r.georob.m1.spher.reml, newdata=blks,
+   control=control.predict.georob(extended.output=TRUE, pwidth=800, pheight=800))
> r.blks <- lgnpp(r.blks, newdata=meuse.grid)
```

Note that we set `pwidth` and `pheight` equal to the dimension of the blocks. This is best for rectangular blocks because the blocks are then represented by a single pixel, which reduces computations.
Next we use `lgnpp()` for computing the optimal log-normal block predictions. As we need the full covariance matrix of the point prediction errors for this we must re-compute the points predictions of `log(zinc)` with the additional `control` argument `full.covmat=TRUE`:

```
> t.pk <- predict(r.georob.m0.spher.reml, newdata=as.data.frame(meuse.grid),
+   control=control.predict.georob(extended.output=TRUE, full.covmat=TRUE))
> str(t.pk)
```

```
List of 5
 $ pred           :'data.frame':        3103 obs. of  10 variables:
  ..$ x              : num [1:3103] 181180 181140 181180 181220 181100 ...
  ..$ y              : num [1:3103] 333740 333700 333700 333700 333660 ...
  ..$ pred           : num [1:3103] 7.05 7.06 6.8 6.57 7.07 ...
  ..$ se             : num [1:3103] 0.277 0.248 0.252 0.259 0.212 ...
  ..$ lower          : num [1:3103] 6.51 6.57 6.3 6.06 6.65 ...
  ..$ upper          : num [1:3103] 7.59 7.54 7.29 7.07 7.48 ...
  ..$ trend          : num [1:3103] 7.09 7.09 6.85 6.64 7.09 ...
  ..$ var.pred       : num [1:3103] 0.0779 0.0899 0.0822 0.0757 0.1027 ...
  ..$ cov.pred.target: num [1:3103] 0.0681 0.0818 0.0768 0.0717 0.0963 ...
  ..$ var.target     : num [1:3103] 0.135 0.135 0.135 0.135 0.135 ...
  ..- attr(*, "variogram.model")= chr "RMspheric"
  ..- attr(*, "param")= Named num [1:4] 0.1349 0 0.0551 876.5812
  .. ..- attr(*, "names")= chr [1:4] "variance" "snugget" "nugget" "scale"
  ..- attr(*, "psi.func")= chr "logistic"
  ..- attr(*, "tuning.psi")= num 1000
  ..- attr(*, "type")= chr "signal"
 $ mse.pred       : num [1:3103, 1:3103] 0.0766 0.0565 0.0606 0.0583 0.0373 ...
 $ var.pred       : num [1:3103, 1:3103] 0.0779 0.0833 0.0794 0.0748 0.0878 ...
 $ cov.pred.target: num [1:3103, 1:3103] 0.0681 0.0736 0.0714 0.0684 0.0781 ...
 $ var.target     : num [1:3103, 1:3103] 0.135 0.122 0.126 0.122 0.109 ...
```

The predictions are now stored in the list component `pred`, and list components `mse.pred`, `var.pred`, `var.target` and `cov.pred.target` store the covariance matrices of prediction errors, predictions, prediction targets and the covariances between predictions and targets. Then we back-transform the predictions and average them separately for the two blocks:

```
> ## index defining to which block the points predictions belong
> ind <- over(geometry(meuse.grid), geometry(blks))
> ind <- tapply(1:nrow(meuse.grid), factor(ind), function(x) x)
> ## select point predictions in block and predict block average
> tmp <- t(sapply(ind, function(i, x){
+   x$pred <- x$pred[i,]
+   x$mse.pred <- x$mse.pred[i,i]
+   x$var.pred <- x$var.pred[i,i]
+   x$cov.pred.target <- x$cov.pred.target[i,i]
+   x$var.target <- x$var.target[i,i]
+   res <- lgnpp(x, is.block=TRUE)
+   res
+   }, x=t.pk))
> colnames(tmp) <- c("opt.pred", "opt.se")
> r.blks <- cbind( r.blks, tmp)


> r.blks@data[, c("lgn.pred", "opt.pred", "lgn.se", "opt.se")]


       lgn.pred opt.pred lgn.se opt.se
block1   295.40   330.10 18.771 17.671
block2   191.67   190.69 12.082 12.480
```

Both the predictions and the prediction standard errors differ somewhat for `block1`. Based on the Cressie's simulation study, we prefer the optimal prediction results.

4 Robust analysis of `coalash` data

This data set reports measurements of ash content [% mass] in a coal seam in Pennsylvania (Gomez and Hazen, 1970). A subset of the data was analysed by Cressie (1993, section 2.2) to illustrate techniques for robust exploratory analysis of geostatistical data and for robust estimation of the sample variogram.

4.1 Exploratory analysis

The subset of the data analyzed by Cressie is available from the R package **gstat** (Pebesma, 2004) as data frame `coalash`:

```
> data(coalash, package="gstat")
> summary(coalash)
```

```
       x               y           coalash
 Min.   : 1.00   Min.   : 1.0   Min.   : 7.00
 1st Qu.: 5.00   1st Qu.: 8.0   1st Qu.: 8.96
 Median : 7.00   Median :13.0   Median : 9.79
 Mean   : 7.53   Mean   :12.9   Mean   : 9.78
 3rd Qu.:10.00   3rd Qu.:18.0   3rd Qu.:10.57
 Max.   :16.00   Max.   :23.0   Max.   :17.61
```

The coordinates are given as column and row numbers, the spacing between columns and rows is 2 500 ft. We display the spatial distribution of ash content by a "bubble plot" of the centred data:

```
> plot(y~x, coalash, cex=sqrt(abs(coalash - median(coalash))),
+   col=c("blue", NA, "red")[sign(coalash - median(coalash))+2], asp=1,
+   main="coalash - median(coalash)", ylab="northing", xlab="easting")
> points(y~x, coalash, subset=c(15, 50, 63, 73, 88, 111), pch=4); grid()
> legend("topleft", pch=1, col=c("blue", "red"), legend=c("< 0", "> 0"), bty="n")
```

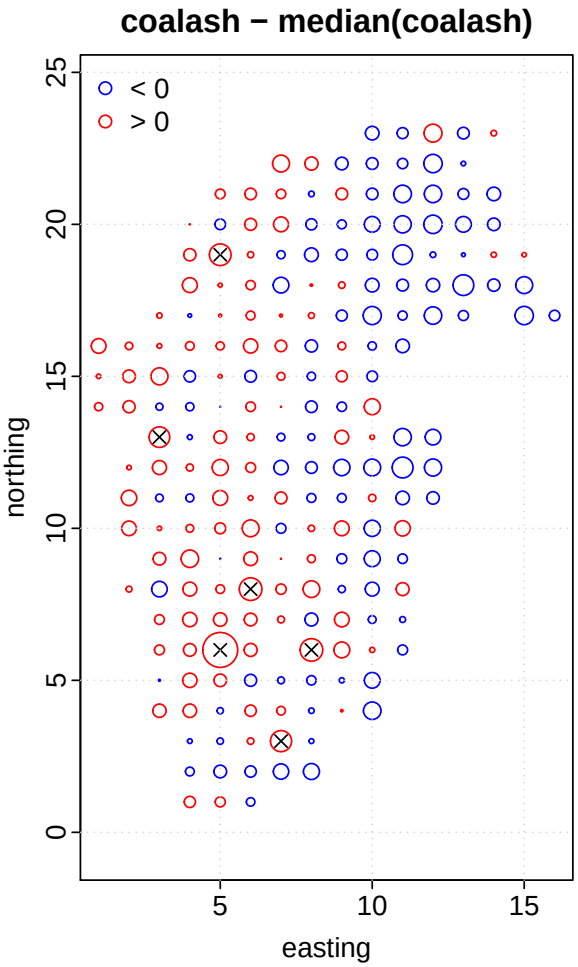

Figure 15: "Bubble plot" of coalash data centred by median (area of symbols $\propto$ moduli of centred data; $\times$: observation identified by Cressie as outlier).

Ash content tends to decrease from west to east and is about constant along the north-south direction:

```
> op <- par(mfrow=c(1,2))
> with(coalash, scatter.smooth(x, coalash, main="coalash ~ x"))
> with(coalash, scatter.smooth(y, coalash, main="coalash ~ y"))
> par(op)
```

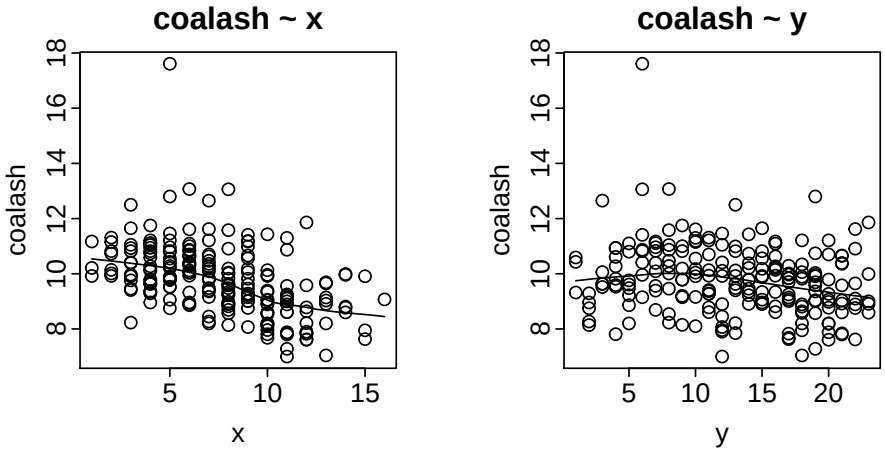

Figure 16: Coalash data plotted vs easting and northing.

There is a distinct outlier at position (5,6), other observations do not clearly stand out from the marginal distribution:

```
> qqnorm(coalash$coalash)
```

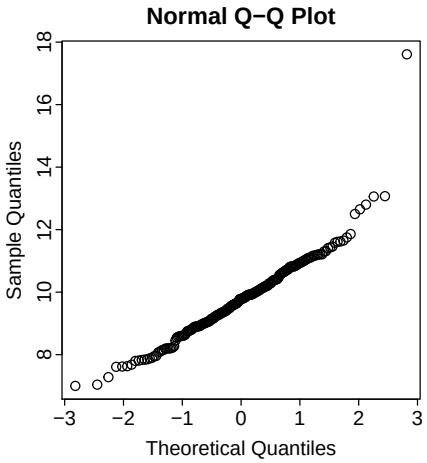

Figure 17: Normal QQ plot of coalash data.

Cressie identified the observations at locations (7,3), (8,6), (6,8), (3,13) and (5,19) as *local outliers* (marked by × in Figure 15) and the observations of row 2 as "pocket of non-stationarity". One could add to this list the observation at location (12,23), which is clearly larger than the values at adjacent locations.

To further explore the data we fit a linear function of the coordinates as drift to the data by a robust MM-estimator:

```
> library(robustbase)
> r.lmrob <- lmrob(coalash~x+y, coalash)
> summary(r.lmrob)
```

```
Call:
lmrob(formula = coalash ~ x + y, data = coalash)
 \--> method = "MM"
Residuals:
    Min      1Q  Median      3Q     Max
-2.2644 -0.6438  0.0181  0.6498  7.4702

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.036445   0.190651   57.89  < 2e-16 ***
x           -0.178235   0.021721   -8.21  2.5e-14 ***
y           -0.000917   0.012692   -0.07     0.94
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Robust residual standard error: 0.952
Multiple R-squared:  0.281,        Adjusted R-squared:  0.274
Convergence in 9 IRWLS iterations

Robustness weights:
 observation 50 is an outlier with |weight| = 0 ( < 0.00048);
 18 weights are ~= 1. The remaining 189 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.160   0.882   0.954   0.901   0.984   0.999
Algorithmic parameters:
       tuning.chi                bb        tuning.psi        refine.tol
         1.55e+00          5.00e-01          4.69e+00          1.00e-07
          rel.tol         scale.tol         solve.tol       eps.outlier
         1.00e-07          1.00e-10          1.00e-07          4.81e-04
            eps.x warn.limit.reject warn.limit.meanrw
         4.18e-11          5.00e-01          5.00e-01
     nResample         max.it       best.r.s       k.fast.s          k.max
           500             50              2              1            200
   maxit.scale      trace.lev            mts     compute.rd fast.s.large.n
           200              0           1000              0           2000
                  psi           subsampling                   cov
           "bisquare"         "nonsingular"         ".vcov.avar1"
compute.outlier.stats
                "SM"
seed : int(0)
```

and check the fitted model by residual diagnostic plots:

```
> op <- par(mfrow=c(2,2))
> plot(r.lmrob, which=c(1:2, 4:5))
> par(op)
```

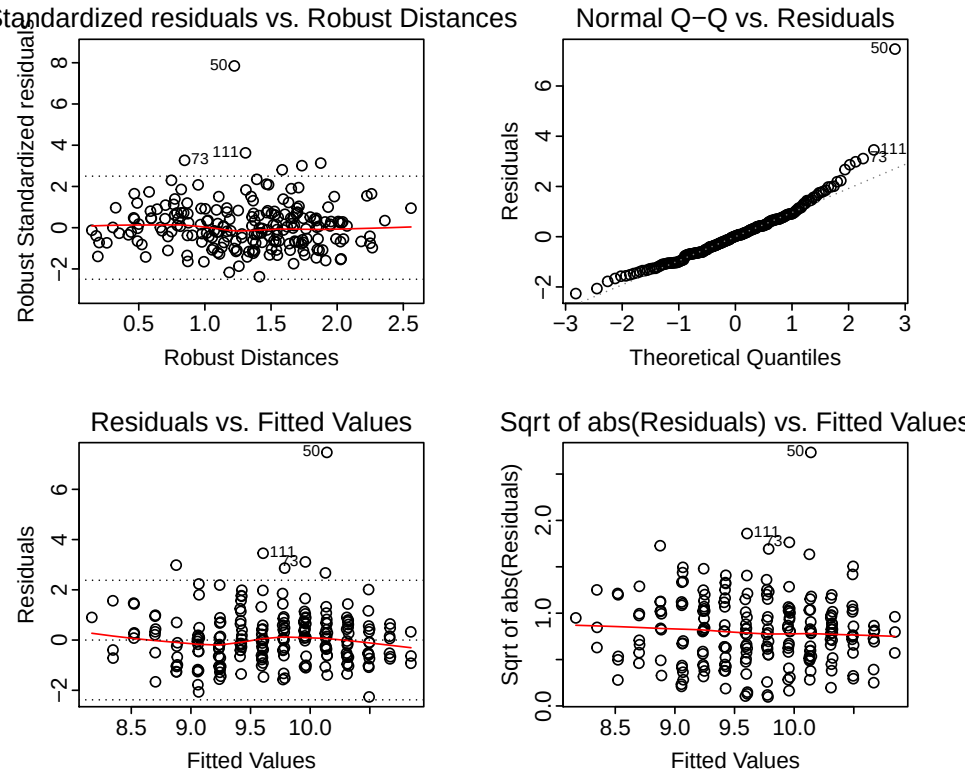

Figure 18: Residual diagnostic plots for robust fit of the linear drift model for coalash data.

Observations (5,6), (8,6) and (6,8) (Indices: 50, 111, 73) are labelled as outliers. Their robustness weights,

$$w_i = \frac{\psi_c(\widehat{r}_i/\mathrm{se}(\widehat{r}_i))}{\widehat{r}_i/\mathrm{se}(\widehat{r}_i)},$$

— $\widehat{r}_i$ are the regression residuals and $\mathrm{se}(\widehat{r}_i)$ denote their standard errors — are equal to ≈ 0, 0.16 and 0.26, respectively.

Next, we compute the isotropic sample variogram of the `lmrob` residuals by various (robust) estimators (e.g. Lark, 2000):

```
> library(georob)
> plot(sample.variogram(residuals(r.lmrob), locations=coalash[, c("x","y")],
+   lag.dist.def=1, max.lag=10, estimator="matheron"), pch=1, col="black",
+   main="sample variogram of residuals coalash~x+y")
> plot(sample.variogram(residuals(r.lmrob), locations=coalash[, c("x","y")],
+   lag.dist.def=1, estimator="qn"), pch=2, col="blue", add=TRUE)
> plot(sample.variogram(residuals(r.lmrob), locations=coalash[, c("x","y")],
+   lag.dist.def=1, estimator="ch"), pch=3, col="cyan", add=TRUE)
> plot(sample.variogram(residuals(r.lmrob), locations=coalash[, c("x","y")],
+   lag.dist.def=1, estimator="mad"), pch=4, col="orange", add=TRUE)
> legend("bottomright", pch=1:4, col=c("black", "blue", "cyan", "orange"),
+   legend=paste(c("method-of-moments", "Qn", "Cressie-Hawkins", "MAD"),
+   "estimator"), bty="n")
```

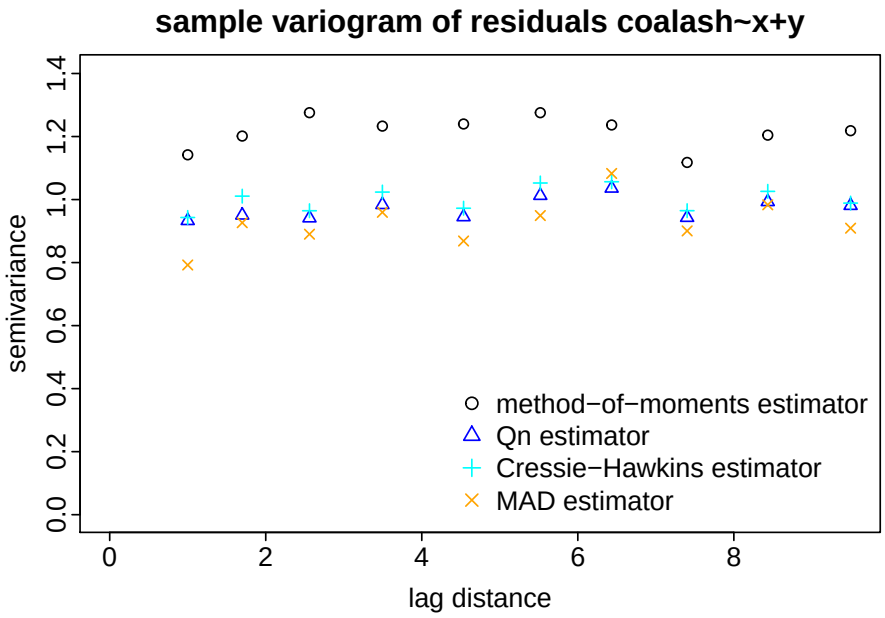

Figure 19: (Non-)robustly estimated isotropic sample variogram of robust regression residuals of coalash data.

Spatial dependence of the residuals is very weak, the outliers clearly distort the method-of-moment estimate, but the various robust estimates hardly differ.

To check whether drift removal accounted for directional effects we compute the sample variogram separately for the N-S and W-E directions (only Qn-estimator):

```
> r.sv <- sample.variogram(residuals(r.lmrob), locations=coalash[, c("x","y")],
+   lag.dist.def=1, max.lag=10, xy.angle.def=c(-0.1, 0.1, 89.9, 90.1),
+   estimator="qn")
> plot(gamma~lag.dist, r.sv, subset=lag.x < 1.e-6, xlim=c(0, 10), ylim=c(0, 1.4),
+   pch=1, col="blue",
+   main="directional sample variogram of residuals (Qn-estimator)")
> points(gamma~lag.dist, r.sv, subset=lag.y < 1.e-6, pch=3, col="orange")
> legend("bottomright", pch=c(1, 3), col=c("blue", "orange"),
+   legend=c("N-S direction", "W-E direction"), bty="n")
```

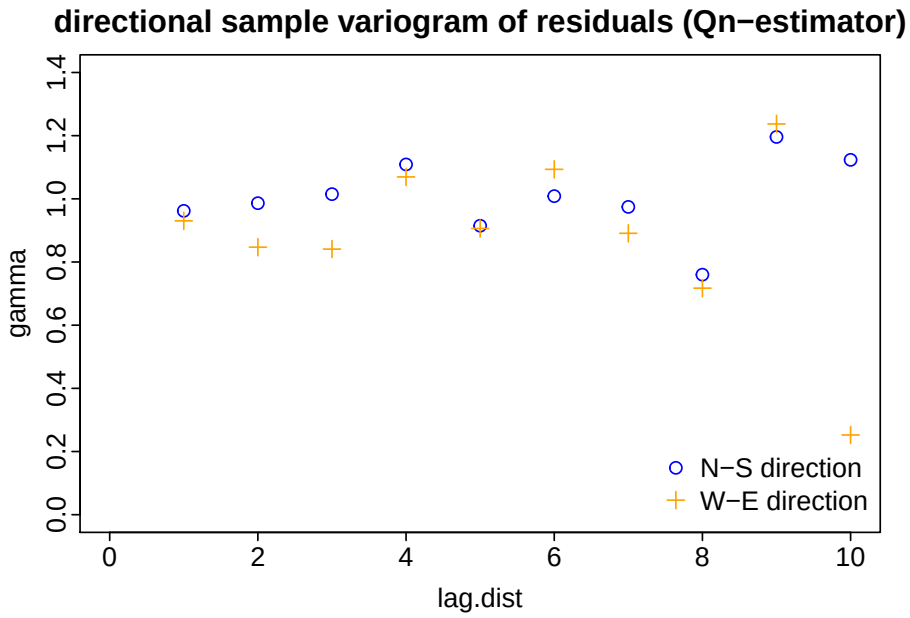

Figure 20: Direction-dependent sample variogram of robust regression residuals of coalash data (Q_n-estimate).

There is no indication that residual auto-correlation depends on direction.

4.2 Fitting a spatial linear model robust REML

Based on the results of the exploratory analysis, we fit a model with a linear drift in the coordinates and an isotropic exponential variogram to the data by robust REML. By default, `georob()` uses a scaled and shifted "logistic" ψ_c-function

$$\psi_c(x) = \tanh(x/c)$$

with a tuning constant $c = 2$ for robust REML, but the popular Huber function and a redescending ψ-function based on the t-distribution are also implemented (see section 5.9):

```
> r.georob.m0.exp.c2 <- georob(coalash~x+y, coalash, locations=~x+y,
+   variogram.model="RMexp", param=c(variance=0.1, nugget=0.9, scale=1))


> summary(r.georob.m0.exp.c2)


Call:georob(formula = coalash ~ x + y, data = coalash, locations = ~x +
    y, variogram.model = "RMexp", param = c(variance = 0.1, nugget = 0.9,
    scale = 1))

Tuning constant:  2

Convergence in 4 function and 1 Jacobian/gradient evaluations

Estimating equations (gradient)

                          variance        nugget         scale
  Gradient           :    7.7020e-06    9.7254e-07    1.3505e-05
```

```
Predicted latent variable (B):
    Min      1Q  Median      3Q     Max
-0.7757 -0.1906 -0.0242  0.2585  0.8967

Residuals (epsilon):
   Min     1Q Median     3Q    Max
-2.241 -0.598 -0.056  0.463  6.851

Standardized residuals:
   Min     1Q Median     3Q    Max
-2.591 -0.690 -0.065  0.543  7.867


Robust REML estimates

Variogram:  RMexp
               Estimate
variance           0.34
snugget(fixed)     0.00
nugget             0.83
scale              4.75


Fixed effects coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.5118     0.5784   18.17   <2e-16 ***
x            -0.1615     0.0499   -3.24   0.0014 **
y             0.0316     0.0345    0.92   0.3609
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error (sqrt(nugget)): 0.914

Robustness weights:
 17 weights are ~= 1. The remaining 191 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.266   0.913   0.964   0.933   0.991   0.999
```

The diagnostics at the begin of the `summary` output report that `nleqslv()` found the roots of the estimating equations of the variogram parameters (reported numbers are function values evaluated at the root). The criteria controlling convergence of the root-finding algorithm can be controlled by the arguments of `control.nleqslv()`, see section 5.9.

The drift coefficients confirm that there is no clear change of ash content along the N-S direction. A quadratic drift function does not fit the data any better:

```
> waldtest(update(r.georob.m0.exp.c2, .~.+I(x^2)+I(y^2)+I(x*y)), r.georob.m0.exp.c2)


Wald test

Model 1: coalash ~ x + y + I(x^2) + I(y^2) + I(x * y)
Model 2: coalash ~ x + y
  Res.Df Df    F Pr(>F)
1    202
2    205 -3 0.1   0.96
```

We simplify the drift model by dropping y:

```
> r.georob.m1.exp.c2 <- update(r.georob.m0.exp.c2, .~.-y)


> r.georob.m1.exp.c2


Tuning constant:  2

Fixed effects coefficients:
(Intercept)           x
     10.949      -0.163

Variogram:  RMexp
       variance    snugget(fixed)            nugget             scale
          0.241             0.000             0.802             1.706
```

and plot the robust REML estimate of the variogram along with a robust estimate of the sample variogram of the robust REML regression residuals:

```
> plot(r.georob.m1.exp.c2, lag.dist.def=1, max.lag=10, estimator="qn", col="blue")
```

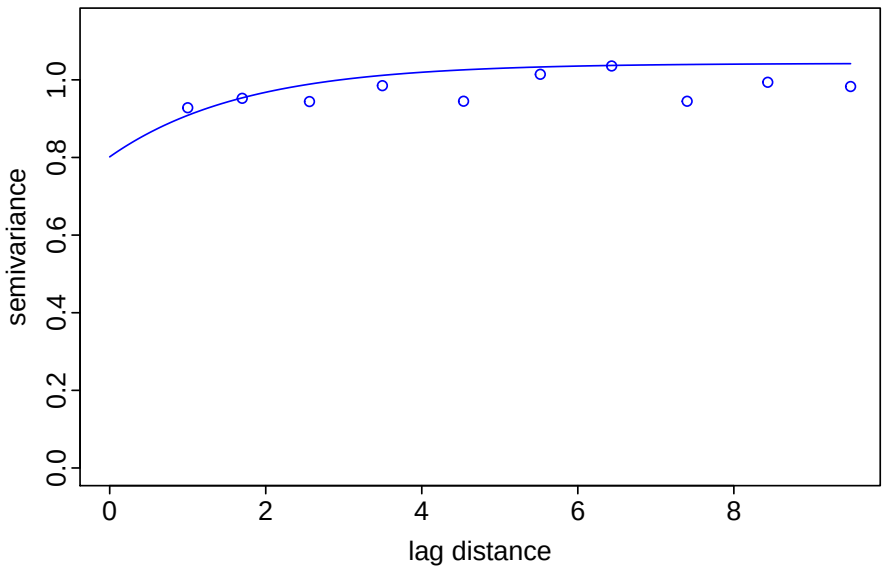

Figure 21: Robust REML estimate of exponential variogram).

As already seen before, residual auto-correlation is weak.

Next, we check the fit of the model by residual diagnostic plots:

```
> op <- par(mfrow=c(2,2))
> plot(r.georob.m1.exp.c2, what="ta")
> plot(r.georob.m1.exp.c2, what="sl")
> plot(r.georob.m1.exp.c2, what="qq.res"); abline(0, 1, lty="dotted")
> plot(r.georob.m1.exp.c2, what="qq.ranef"); abline(0, 1, lty="dotted")
> par(op)
```

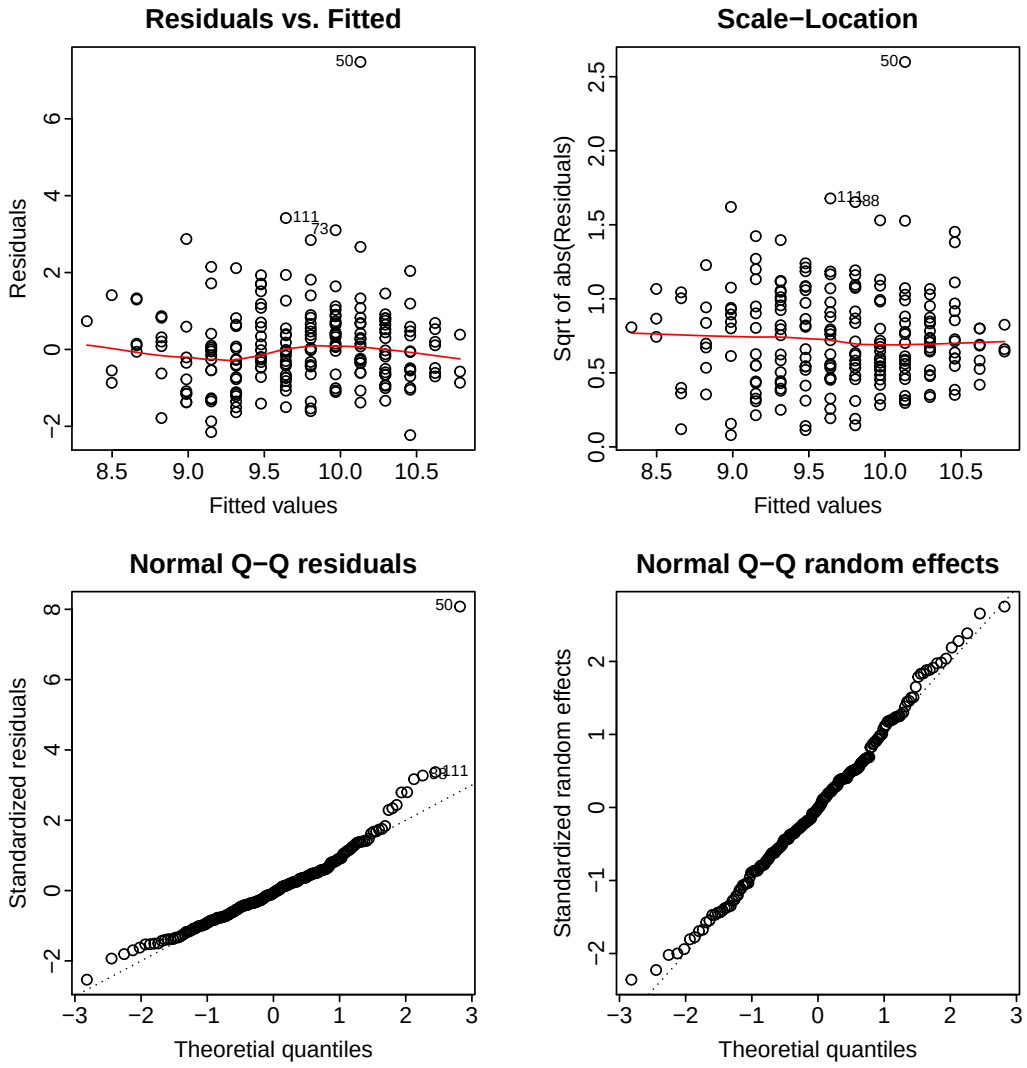

Figure 22: Residual diagnostic plots for robust REML fit of coalash data.

Note that the `plot` method for class `georob` displays for `what = "ta"` or `what = "sl"` the regression residuals $y_i - \boldsymbol{x}(\boldsymbol{s}_i)\widehat{\boldsymbol{\beta}}$, for `what = "qq.res"` the standardized errors $\widehat{\varepsilon}_i/\widehat{\text{se}}(\widehat{\varepsilon}_i) = (y_i - \boldsymbol{x}(\boldsymbol{s}_i)\widehat{\boldsymbol{\beta}} - \widehat{B}(\boldsymbol{s}_i))/\widehat{\text{se}}(\widehat{\varepsilon}_i)$ and for `what = "qq.ranef"` the standardized random effects $\widehat{B}(\boldsymbol{s}_i)/\widehat{\text{se}}(\widehat{B}(\boldsymbol{s}_i))$.

The robustness weights of the outliers identified so far are equal to

```
> round(cbind(coalash[, c("x", "y")],
+   rweights=r.georob.m1.exp.c2[["rweights"]])[c(15, 50, 63, 73, 88, 111, 192),],
+   2)
```

```
     x  y rweights
15   3 13     0.74
50   5  6     0.26
63   5 19     0.66
73   6  8     0.66
88   7  3     0.60
111  8  6     0.58
192 12 23     0.61
```

In addition, the observations

```
> sel <- r.georob.m1.exp.c2[["rweights"]] <= 0.8 &
+   !1:nrow(coalash) %in% c(15, 50, 63, 73, 88, 111, 192)
> round(cbind(coalash[, c("x", "y")],
+   rweights=r.georob.m1.exp.c2[["rweights"]])[sel,],
+   2)
```

```
     x  y rweights
20   3  8     0.70
157 10 14     0.73
171 11 10     0.72
173 11 12     0.80
```

have weights ≤ 0.8. All these outliers are marked in Figure 23 by $\times$:

```
> plot(y~x, coalash, cex=sqrt(abs(residuals(r.georob.m1.exp.c2))),
+   col=c("blue", NA, "red")[sign(residuals(r.georob.m1.exp.c2))+2], asp=1,
+   main="estimated errors robust REML", xlab="northing", ylab="easting")
> points(y~x, coalash, subset=r.georob.m1.exp.c2[["rweights"]]<=0.8, pch=4); grid()
> legend("topleft", pch=1, col=c("blue", "red"), legend=c("< 0", "> 0"), bty="n")
```

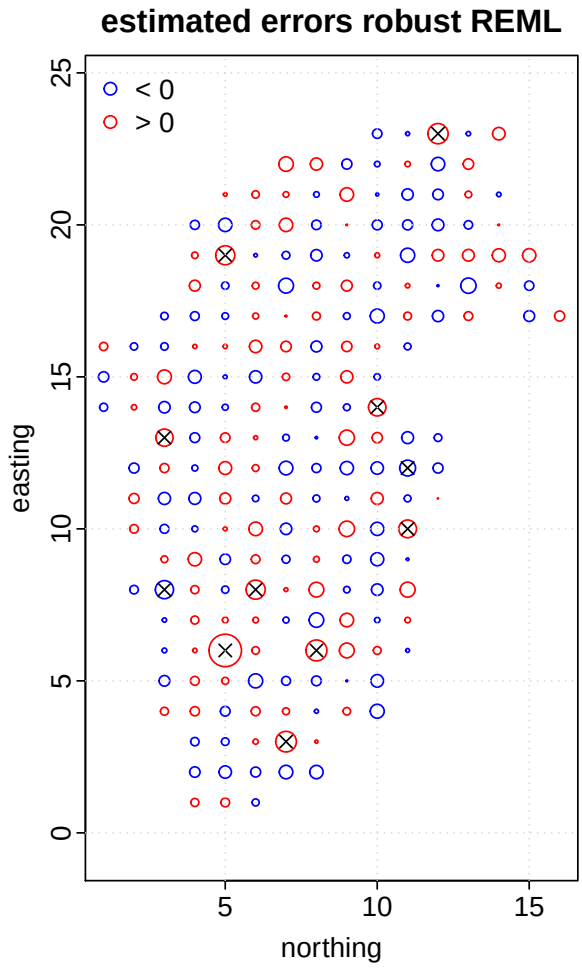

Figure 23: "Bubble plot" of independent errors $\hat{\varepsilon}$ estimated by robust REML (area of symbols $\propto$ moduli of residuals; $\times$: observation with robustness weights $w_i \leq 0.8$).

Comparison with Figure 15 reveals that the 5 additional mild outliers were visible in this plot as well, but were not identified by Cressie's exploratory analysis.

For comparison, we fit the same model by Gaussian REML by setting the tuning constant of the ψ_c-function $c \geq 1000$:

```
> r.georob.m1.exp.c1000 <- update(r.georob.m1.exp.c2, tuning.psi=1000)


> summary(r.georob.m1.exp.c1000)


Call:georob(formula = coalash ~ x, data = coalash, locations = ~x +
    y, variogram.model = "RMexp", param = c(variance = 0.1, nugget = 0.9,
    scale = 1), tuning.psi = 1000)

Tuning constant:  1000

Convergence in 5 function and 5 Jacobian/gradient evaluations

Estimating equations (gradient)

                                  eta         scale
  Gradient           :   -1.2256e-02    2.3135e-02

Maximized restricted log-likelihood: -319.51

Predicted latent variable (B):
    Min      1Q  Median      3Q     Max
-0.6332 -0.1703 -0.0198  0.1814  1.3713

Residuals (epsilon):
   Min     1Q Median     3Q    Max
-2.121 -0.612 -0.107  0.405  6.068

Standardized residuals:
   Min     1Q Median     3Q    Max
-2.284 -0.658 -0.115  0.434  6.500


Gaussian REML estimates

Variogram:  RMexp
               Estimate  Lower Upper
variance         0.2675 0.0745  0.96
snugget(fixed)   0.0000     NA    NA
nugget           1.0225 0.7151  1.46
scale            1.9067 0.3017 12.05


Fixed effects coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.9848     0.3218    34.1  < 2e-16 ***
x            -0.1629     0.0371    -4.4  1.8e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error (sqrt(nugget)): 1.01

Robustness weights:
 All 208 weights are ~= 1.
```

and compare the Gaussian and robust REML estimates of the variogram:

```
> plot(r.georob.m1.exp.c1000, lag.dist.def=1, max.lag=10, estimator="matheron")
> plot(r.georob.m1.exp.c2, lag.dist.def=1, max.lag=10, estimator="qn", add = TRUE,
+   col="blue")
> plot(update(r.georob.m1.exp.c2, subset=-50), lag.dist.def=1, max.lag=10, estimator="qn",
+   add = TRUE, col="orange")
> legend("bottomright", lt=1, col=c("black","blue", "orange"),
+   legend =c("Gaussian REML", "robust REML", "Gaussian REML without outlier (5,6)" ), bty="n")
```

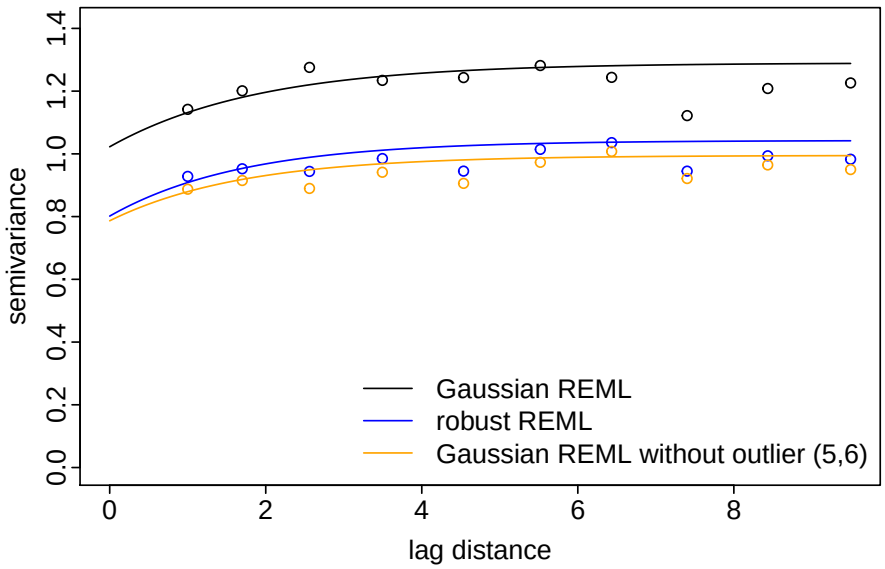

Figure 24: Gaussian and robust REML estimate of exponential variogram.

The outliers inflate mostly the nugget effect (by $\approx$ 20 %) and less so the signal variance and range parameter (by $\approx$ 10 %). When we eliminate the severest outlier at (5,6) then the Gaussian and robust REML estimates of the variogram hardly differ.

Gaussian REML masks the estimated independent errors ($\widehat{\varepsilon}_i$) somewhat at the cost of inflated estimates of random effects ($\widehat{B}(\boldsymbol{s}_i)$):

```
> op <- par(mfrow=c(1,2), cex=5/6)
> plot(residuals(r.georob.m1.exp.c2), residuals(r.georob.m1.exp.c1000),
+   asp = 1, main=expression(paste("Gaussian vs robust ", widehat(epsilon))),
+   xlab=expression(paste("robust ", widehat(epsilon))),
+   ylab=expression(paste("Gaussian ", widehat(epsilon))))
> abline(0, 1, lty="dotted")
> plot(ranef(r.georob.m1.exp.c2), ranef(r.georob.m1.exp.c1000),
+   asp = 1, main=expression(paste("Gaussian vs robust ", italic(widehat(B)))),
+   xlab=expression(paste("robust ", italic(widehat(B)))),
+   ylab=expression(paste("Gaussian ", italic(widehat(B)))))
> abline(0, 1, lty="dotted")
```

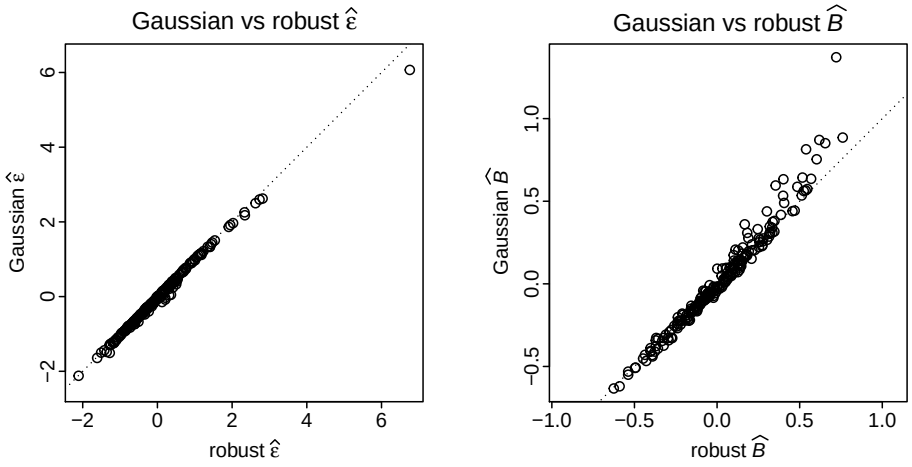

Figure 25: Comparison of estimated errors $\hat{\varepsilon}_i$ and random effects $\widehat{B}(\boldsymbol{s}_i)$ for Gaussian and robust REML fit of coalash data.

We compare the Gaussian and robust REML fit by 10-fold cross-validation:

```
> r.cv.georob.m1.exp.c2 <- cv(r.georob.m1.exp.c2, seed=1)
> r.cv.georob.m1.exp.c1000 <- cv(r.georob.m1.exp.c1000, seed=1)
```

```
Warning message:
In cv.georob(r.georob.m1.exp.c2, seed = 1, ) :
  lack of covergence for  1 cross-validation sets
```

The robustfied estimating equations could not be solved for one cross-valiation subset. This may happen if the initial guesses of the variogram parameters are too far away from the root (see section 5.7). Sometimes it helps then to suppress the computation of robust guesses of the variogram parameters and to use the robust parameter estimates computed from the whole data set as initial values (see argument `initial.param` of `control.georob()` and section 5.7):

```
> r.cv.georob.m1.exp.c2 <- cv(r.georob.m1.exp.c2, seed=1,
+   control=control.georob(initial.param=FALSE))
> r.cv.georob.m1.exp.c1000 <- cv(r.georob.m1.exp.c1000, seed=1)
```

By default, `cv.georob()` partitions the data set into 10 geographically compact subsets of adjacent locations (see argument `method` of `cv.georob()` and section 7.3):

```
> plot(y~x, r.cv.georob.m1.exp.c2$pred, asp=1, col=subset, pch=as.integer(subset))
```

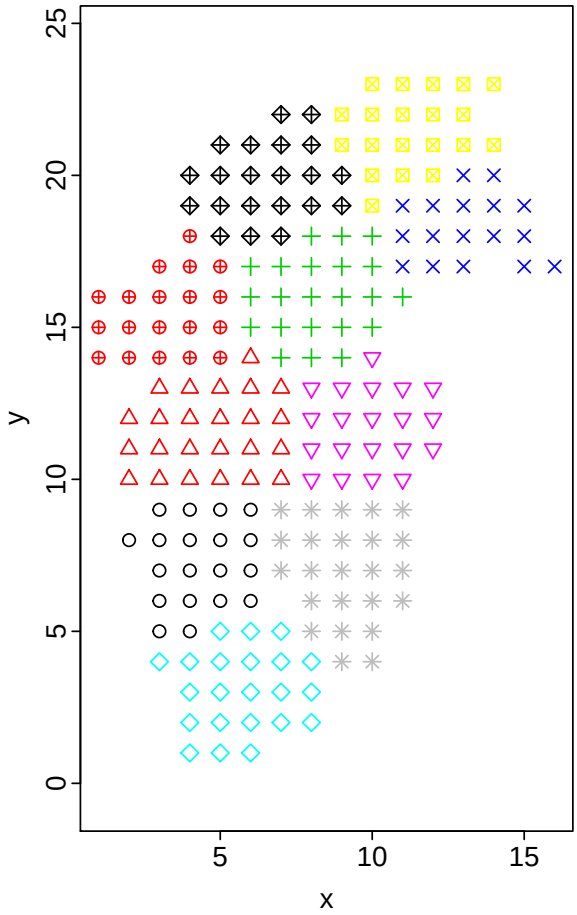

Figure 26: Default method for defining cross-validation subsets by `kmeans()` using argument `method="block"` of `cv.georob()`.

We compute now summary statistics of the (standardized) cross-validation prediction errors for the two model fits (see section 7.3):

```
> summary(r.cv.georob.m1.exp.c1000, se=TRUE)


Statistics of cross-validation prediction errors
    me        mede      rmse      made      qne       msse      medsse
    -0.00619  -0.09188   1.13358   1.05789   0.99910   1.08170   0.38084
se   0.12310   0.12965   0.11588   0.07763   0.05409   0.34814   0.06354
    crps
     0.60090
se   0.04566


> summary(r.cv.georob.m1.exp.c2, se=TRUE)


Statistics of cross-validation prediction errors
    me        mede      rmse      made      qne       msse      medsse    crps
     0.0441  -0.0659    1.1260    1.0427    0.9819    1.2392    0.4568    0.5920
se   0.1127   0.1235    0.1167    0.0693    0.0554    0.3317    0.0704    0.0471
```

The statistics of cross-validation errors are marginally better for the robust fit.

For robust REML, the standard errors of the cross-validation errors are likely too small. This is due to the fact that for the time being, the Kriging variance of the *response* Y is approximated by adding the estimated nugget $\widehat{\tau}^2$ to the Kriging variance of the signal Z. This approximation likely underestimates the mean squared prediction error of the response if the errors come from a long-tailed distribution. Hence, the summary statistics of the *standardized prediction errors* (`msse, medsse`) should be interpreted with caution. The same is true for the continuous-ranked probability score (`crps`), which is computed under the assumption that the predictive distribution of Y is Gaussian even if the errors come from a long-tailed distribution. This cannot strictly hold.

For illustration, we nevertheless show here some diagnostics plots of criteria of the cross-validation prediction errors that further depend on the predictive distributions of the response:

```
> op <- par(mfrow=c(3, 2))
> plot(r.cv.georob.m1.exp.c2, type="ta", col="blue")
> plot(r.cv.georob.m1.exp.c1000, type="ta", col="orange", add=TRUE)
> abline(h=0, lty="dotted")
> legend("topleft", pch=1, col=c("orange", "blue"), legend=c("Gaussian", "robust"), bty="n")
> plot(r.cv.georob.m1.exp.c2, type="qq", col="blue")
> plot(r.cv.georob.m1.exp.c1000, type="qq", col="orange", add=TRUE)
> abline(0, 1, lty="dotted")
> legend("topleft", lty=1, col=c("orange", "blue"), legend=c("Gaussian", "robust"), bty="n")
> plot(r.cv.georob.m1.exp.c2, type="ecdf.pit", col="blue", do.points=FALSE)
> plot(r.cv.georob.m1.exp.c1000, type="ecdf.pit", col="orange", add=TRUE, do.points=FALSE)
> abline(0, 1, lty="dotted")
> legend("topleft", lty=1, col=c("orange", "blue"), legend=c("Gaussian", "robust"), bty="n")
> plot(r.cv.georob.m1.exp.c2, type="bs", col="blue")
> plot(r.cv.georob.m1.exp.c1000, type="bs", col="orange", add=TRUE)
> legend("topright", lty=1, col=c("orange", "blue"), legend=c("Gaussian", "robust"), bty="n")
> plot(r.cv.georob.m1.exp.c1000, type="mc", main="Gaussian REML")
> plot(r.cv.georob.m1.exp.c2, type="mc", main="robust REML")
> par(op)
```

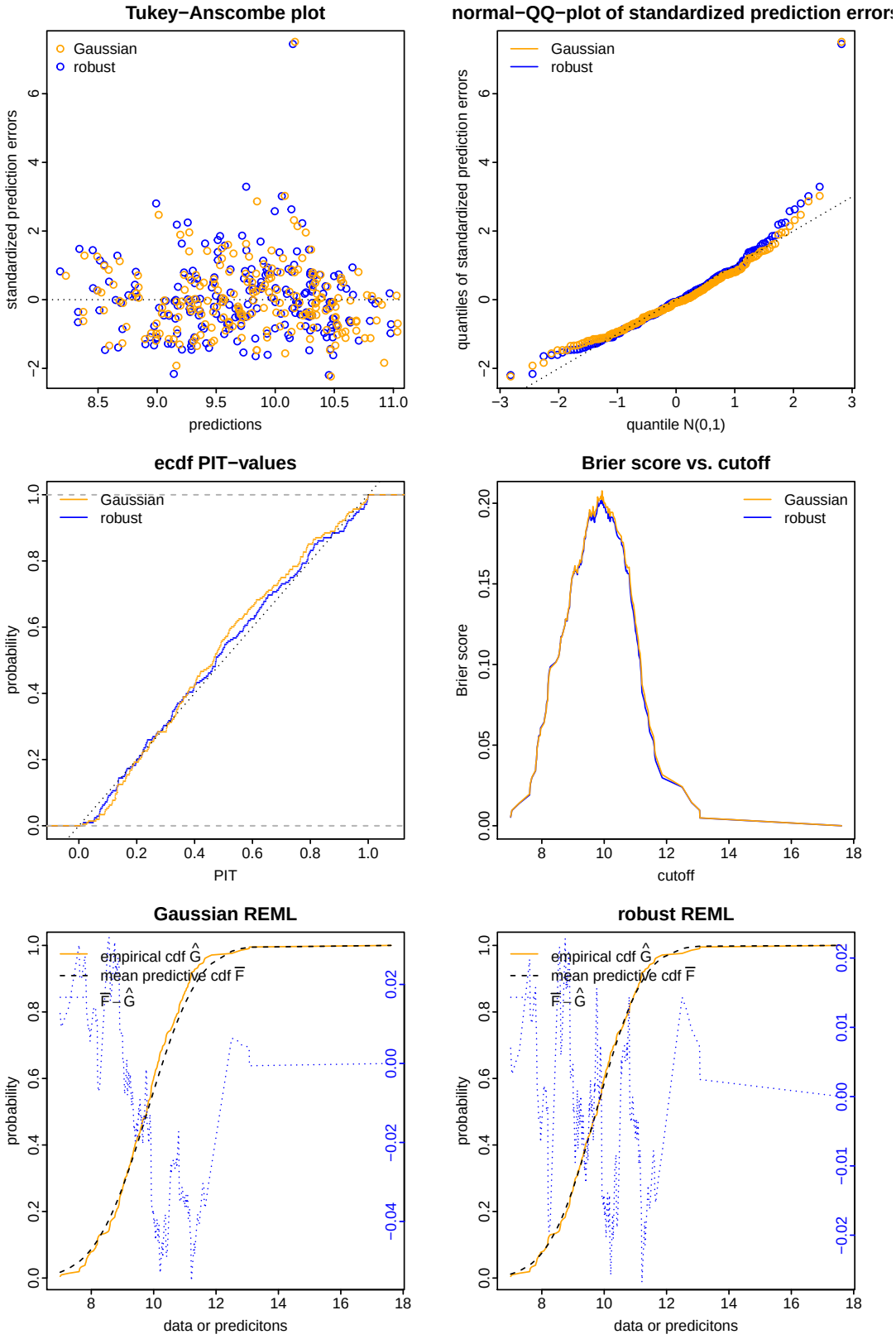

Figure 27: Diagnostic plots of the standardized cross-validation errors, the probability integral transform (PIT), the Brier score and of the predictive distributions for Gaussian and robust REML.

In general, the diagnostics are slightly better for robust than for Gaussian REML: The empirical distribution of the probability integral transform (PIT) is closer to a uniform

distribution, the Brier score (BS) is smaller and the average predictive distribution $\bar{F}$ is closer to the empirical distribution of the data $\widehat{G}$ (see section 7.3 and Gneiting *et al.*, 2007, for more details about the interpretation of these plots).

4.3 Computing robust Kriging predictions

4.3.1 Point Kriging

For point Kriging we must first generate a fine-meshed grid of predictions points (optionally with the covariates for the drift) that is passed as argument `newdata` to `predict.georob()`. `newdata` can be an customary `data.frame`, an object of class `SpatialPointsDataFrame`, `SpatialPixelsDataFrame` or `SpatialGridDataFrame` or `SpatialPoints`, `SpatialPixels` or `SpatialGrid`, all provided by the package **sp**. If `newdata` is a `SpatialPoints`, `SpatialPixels` or a `SpatialGrid` object then the drift model may only use the coordinates as covariates (universal Kriging), as we do it here.

```
> coalash.grid <- expand.grid(x=seq(-1, 17, by=0.2),
+   y=seq( -1, 24, by=0.2))
> coordinates( coalash.grid) <- ~x+y  # convert to SpatialPoints
> gridded( coalash.grid) <- TRUE      # convert to SpatialPixels
> fullgrid( coalash.grid) <- TRUE     # convert to SpatialGrid
> str(coalash.grid, max=2)


Formal class 'SpatialGrid' [package "sp"] with 3 slots
  ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
  ..@ bbox       : num [1:2, 1:2] -1.1 -1.1 17.1 24.1
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

Computing (robust) plug-in Kriging predictions is then straightforward:

```
> r.pk.m1.exp.c2 <- predict(r.georob.m1.exp.c2, newdata=coalash.grid)
> r.pk.m1.exp.c1000 <- predict(r.georob.m1.exp.c1000, newdata=coalash.grid)
```

and plotting the results by the function `spplot()` of the package **sp** as well:

```
> pred.rob <- spplot(r.pk.m1.exp.c2, "pred", at=seq(8, 12, by=0.25),
+   main="robust Kriging prediction", scales=list(draw=TRUE))
> pred.gauss <- spplot(r.pk.m1.exp.c1000, "pred", at=seq(8, 12, by=0.25),
+   main="Gaussian Kriging prediction", scales=list(draw=TRUE))
> se.rob <- spplot(r.pk.m1.exp.c2, "se", at=seq(0.35, 0.65, by=0.025),
+   main="standard error robust Kriging", scales=list(draw=TRUE))
> se.gauss <- spplot(r.pk.m1.exp.c1000, "se", at=seq(0.35, 0.65, by=0.025),
+   main="standard error Gaussian Kriging", scales=list(draw=TRUE))
> plot(pred.rob, pos=c(0, 0.5, 0.5, 1), more=TRUE)
> plot(pred.gauss, pos=c(0.5, 0.5, 1, 1), more=TRUE)
> plot(se.rob, pos=c(0, 0, 0.5, 0.5), more=TRUE)
> plot(se.gauss, pos=c(0.5, 0, 1, 0.5), more=FALSE)
```

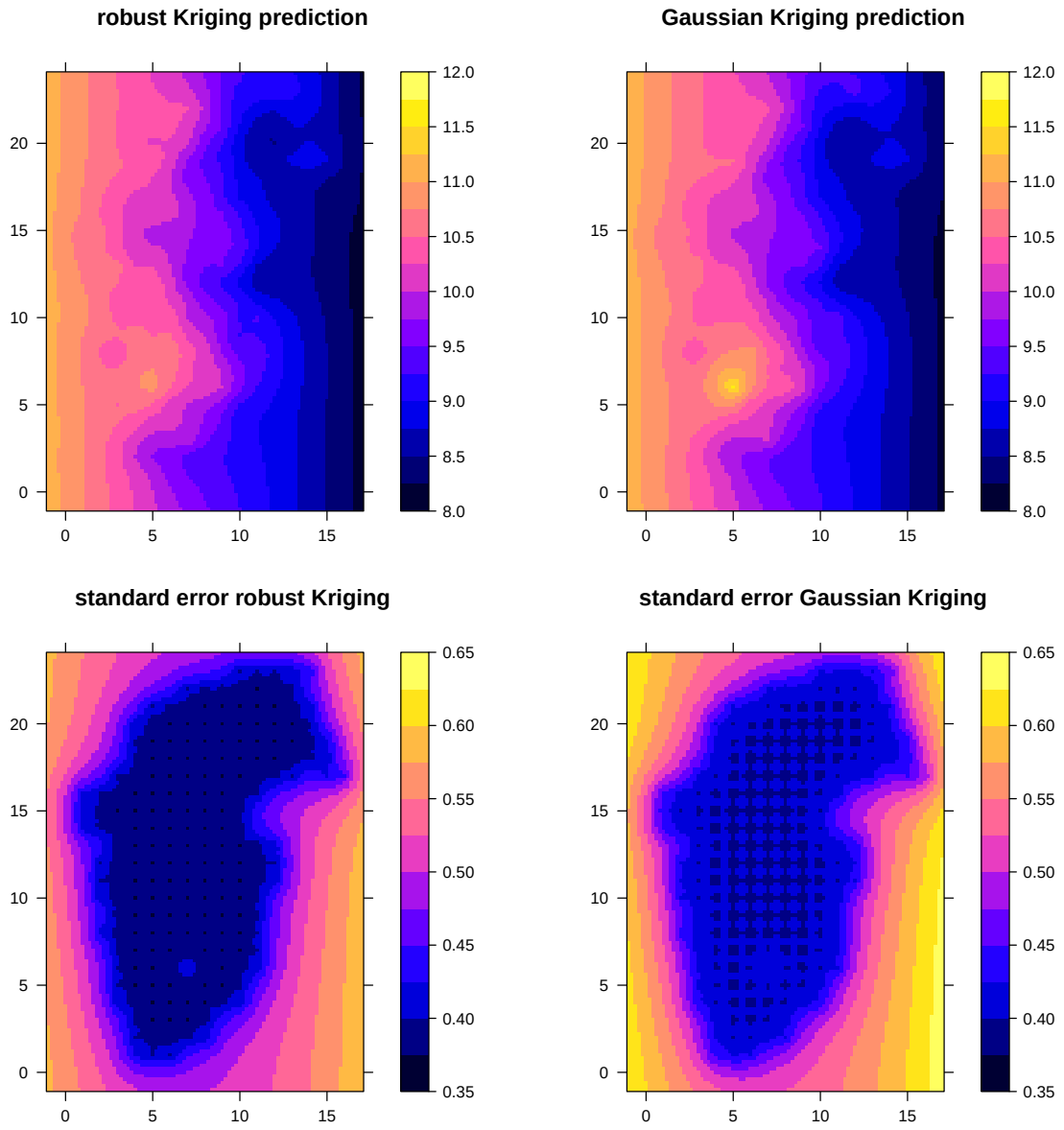

Figure 28: Robust (left) and Gaussian (right) point Kriging predictions (top) and Kriging standard errors (bottom).

By default, `predict.georob()` predicts the *signal* $Z(\boldsymbol{s})$, but predictions of the *response* $Y(\boldsymbol{s})$ or estimates of *model terms* and *trend* (drift) can be computed as well (see argument `type` of `predict.georob()` and section 6.1.1). Apart from Kriging predictions and standard errors, bounds (`lower`, `upper`) of point-wise 95%-prediction intervals (plug-in) are computed for assumed Gaussian predictive distributions.

The Kriging predictions visibly differ around the outlier at (5,6) and the Kriging standard errors are larger for Gaussian Kriging because the outliers inflated the non-robustly estimated sill of the variogram (Figure 24). To see the effects of outliers more clearly, we plot the relative difference of Gaussian and robust Kriging predictions and the ratio of the respective Kriging variances:

```
> library(lattice)
> # rel. difference of predictions
> r.pk.m1.exp.c2$reldiff.pred <- (r.pk.m1.exp.c1000$pred -
```

```
+   r.pk.m1.exp.c2$pred) / r.pk.m1.exp.c2$pred * 100
> reldiff.pred <- spplot(r.pk.m1.exp.c2, "reldiff.pred", at=-1:7,
+   main="Gaussian - robust Kriging predictions", scales=list(draw=TRUE))
> # ratio Kriging variances
> r.pk.m1.exp.c2$ratio.msep <- r.pk.m1.exp.c1000$se^2 /
+   r.pk.m1.exp.c2$se^2 * 100
> ratio.msep <- spplot(r.pk.m1.exp.c2, "ratio.msep",  at=105:115,
+   main="ratio of Gaussian to robust Kriging variances",scales=list(draw=TRUE))
> plot(reldiff.pred, pos=c(0, 0, 0.5, 1), more=TRUE)
> #  add bubble plot of centred data colored by "robustness" weights
> rw <- cut(r.georob.m1.exp.c2$rweights, seq(0.2, 1, by = 0.2))
> trellis.focus("panel", 1, 1)
> panel.points(coalash$x, coalash$y, lwd=2,
+   cex=sqrt(abs(coalash$coalash - median((coalash$coalash)))),
+   col=colorRampPalette(c("yellow", "orange", grey(0.4)))(4)[as.numeric(rw)])
> panel.text(rep(17, nlevels(rw)+1), 0:nlevels(rw), pos=2, cex=0.8,
+   labels=c(rev(levels(rw)), "rob. weights"),
+   col=c(rev(colorRampPalette(c("yellow", "orange", grey(0.4)))(4)), "white"))
> trellis.unfocus()
> plot(ratio.msep, pos=c(0.5, 0, 1, 1), more=FALSE)
```

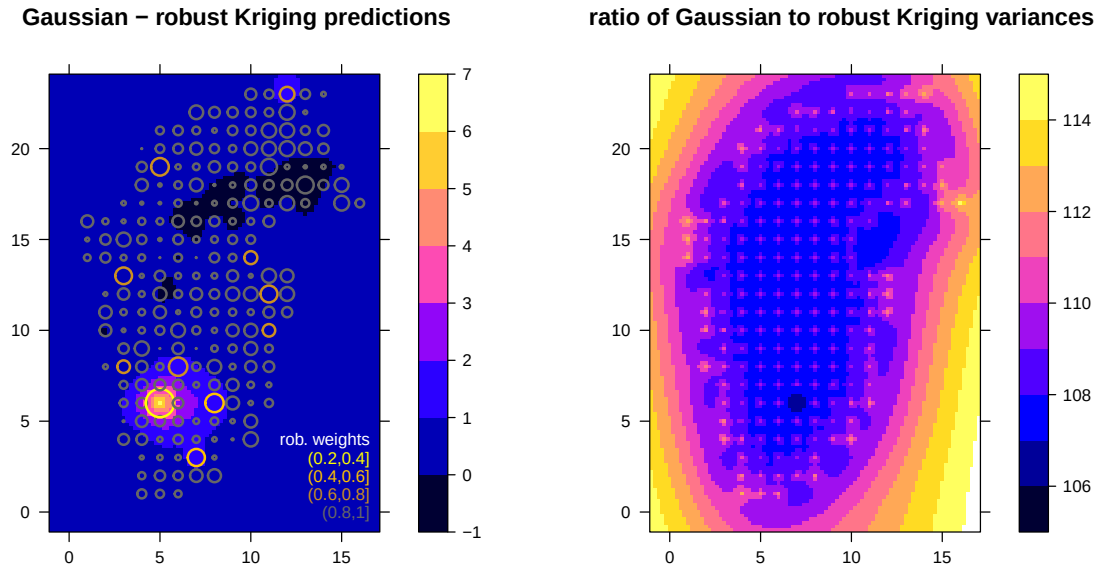

Figure 29: Relative differences of Gaussian and robust point Kriging predictions (%, left) and ratio of Gaussian to robust point Kriging variances (%, right). The area of the "bubbles" in the left panel is proportional to the moduli of centred ash content and their color codes the "robustness" weights of robust REML.

The Kriging predictions differ by more than $\pm$ 1 % around observations with robustness weights ≤ 0.6, and the efficiency of the Gaussian relative to robust Kriging varies between 106–114 %.

4.3.2 Block Kriging

First, we must define the blocks (and the covariates) for which predictions should be computed. The package **georob** computes block Kriging predictions for `newdata` objects of class `SpatialPolygonsDataFrame`s provided by **sp**:

```
> tmp <- expand.grid(x = seq(2.5, 16.5, by=4), y=seq(2, 22, by=4))
> rownames(tmp) <- paste("block", rownames(tmp), sep="")
> # create SpatialPolygonsDataFrame
> coalash.polygons <- sapply(1:nrow(tmp), function(i, x){
+   Polygons(list(Polygon(
+         t(x[,i] + t(cbind(c(-2, 2, 2, -2, -2), c(-2, -2, 2, 2, -2)))),
+         hole=FALSE)), ID=paste("block", i, sep=""))},
+   x=t(tmp))
> coalash.polygons <- SpatialPolygonsDataFrame(SpatialPolygons(coalash.polygons),
+   data = tmp)
> summary(coalash.polygons)


Object of class SpatialPolygonsDataFrame
Coordinates:
  min  max
x 0.5 16.5
y 0.0 24.0
Is projected: NA
proj4string : [NA]
Data attributes:
       x              y
 Min.   : 2.5   Min.   : 2
 1st Qu.: 5.5   1st Qu.: 6
 Median : 8.5   Median :12
 Mean   : 8.5   Mean   :12
 3rd Qu.:11.5   3rd Qu.:18
 Max.   :14.5   Max.   :22


> plot(coalash.polygons, col="grey", axes=TRUE); points(y~x, coalash)
```

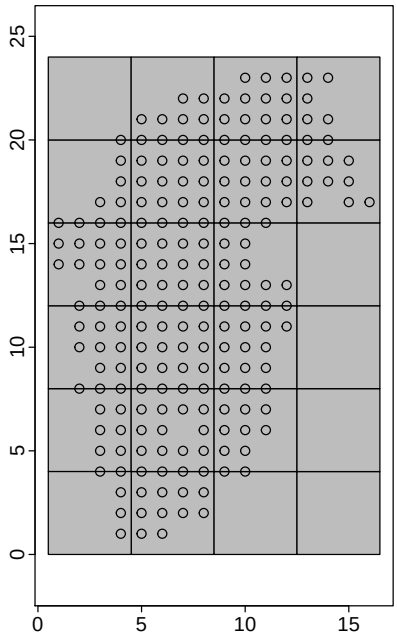

Figure 30: Geometry of blocks for which Kriging predictions are computed (dots are data locations).

Note that coordinates of the block centres are the covariates for the drift model of the block means. Block-block- and point-block-means of the semi-variance are computed by `predict.georob()` efficiently by functions of the R packages **constrainedKriging** and **spatialCovariance** (see section 6.1 and Hofer and Papritz, 2011; Clifford, 2005):

```
> r.bk.m1.exp.c2 <- predict(r.georob.m1.exp.c2, newdata=coalash.polygons,
+   control=control.predict.georob(pwidth=4, pheight=4, full.covmat=TRUE))
> r.bk.m1.exp.c1000 <- predict(r.georob.m1.exp.c1000, newdata=coalash.polygons,
+   control=control.predict.georob(pwidth=4, pheight=4, full.covmat=TRUE))
```

Since the blocks are squares of constant size, choosing a square "integration pixel" of the same dimension (arguments `pwidth`, `pheight` of `control.predict.georob()`) allows to compute the required mean semi-variances most efficiently (see Hofer and Papritz, 2011, for details). The third argument `full.covmat=TRUE` of `control.predict.georob()` has the effect that the full covariance matrix of the predictions errors of the block means is computed (and stored as list component `mse.pred` in `r.bk.m1.exp.c2`):

```
> str(r.bk.m1.exp.c2, max=2)

List of 2
 $ pred    :Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
 $ mse.pred: num [1:24, 1:24] 0.08216 0.01212 -0.00303 -0.01803 0.02881 ...
```

We can compute from these covariances the Kriging variances of the spatial means over groups of block, in particular of the spatial mean over the whole domain:

```
> c(pred=mean(r.bk.m1.exp.c2$pred$pred),
+   se=sqrt(sum(r.bk.m1.exp.c2$mse.pred))/24)

    pred       se
9.558919 0.087422
```

which is of course the same as predicting the mean over the whole domain by block Kriging:

```
> coalash.domain <- rbind(c(0.5,0), c(16.5,0), c(16.5,24), c(0.5,24), c(0.5,0))
> coalash.domain <- SpatialPolygonsDataFrame(
+   SpatialPolygons(list(Polygons(list(Polygon(coalash.domain)), ID= "domain"))),
+   data=data.frame(x=8.5,y=12,row.names="domain"))
> slot(predict(r.georob.m1.exp.c2, newdata=coalash.domain,
+   control=control.predict.georob(pwidth=16, pheight=24)), "data")

         pred       se  lower  upper
domain 9.5589 0.087422 9.3876 9.7303
```

We conclude this section by graphs of the robust and Gaussian block Kriging predictions:

```
> pred.rob <- spplot(r.bk.m1.exp.c2$pred, "pred", at=seq(8, 11, by=0.25),
+   main="robust Kriging prediction", scales=list(draw=TRUE))
> pred.gauss <- spplot(r.bk.m1.exp.c1000$pred, "pred", at=seq(8, 11, by=0.25),
+   main="Gaussian Kriging prediction", scales=list(draw=TRUE))
> se.rob <- spplot(r.bk.m1.exp.c2$pred, "se", at=seq(0.15, 0.45, by=0.025),
+   main="standard error robust Kriging", scales=list(draw=TRUE))
> se.gauss <- spplot(r.bk.m1.exp.c1000$pred, "se", at=seq(0.15, 0.45, by=0.025),
+   main="standard error Gaussian Kriging", scales=list(draw=TRUE))
> plot(pred.rob, pos=c(0, 0.5, 0.5, 1), more=TRUE)
> plot(pred.gauss, pos=c(0.5, 0.5, 1, 1), more=TRUE)
> plot(se.rob, pos=c(0, 0, 0.5, 0.5), more=TRUE)
> plot(se.gauss, pos=c(0.5, 0, 1, 0.5), more=FALSE)
```

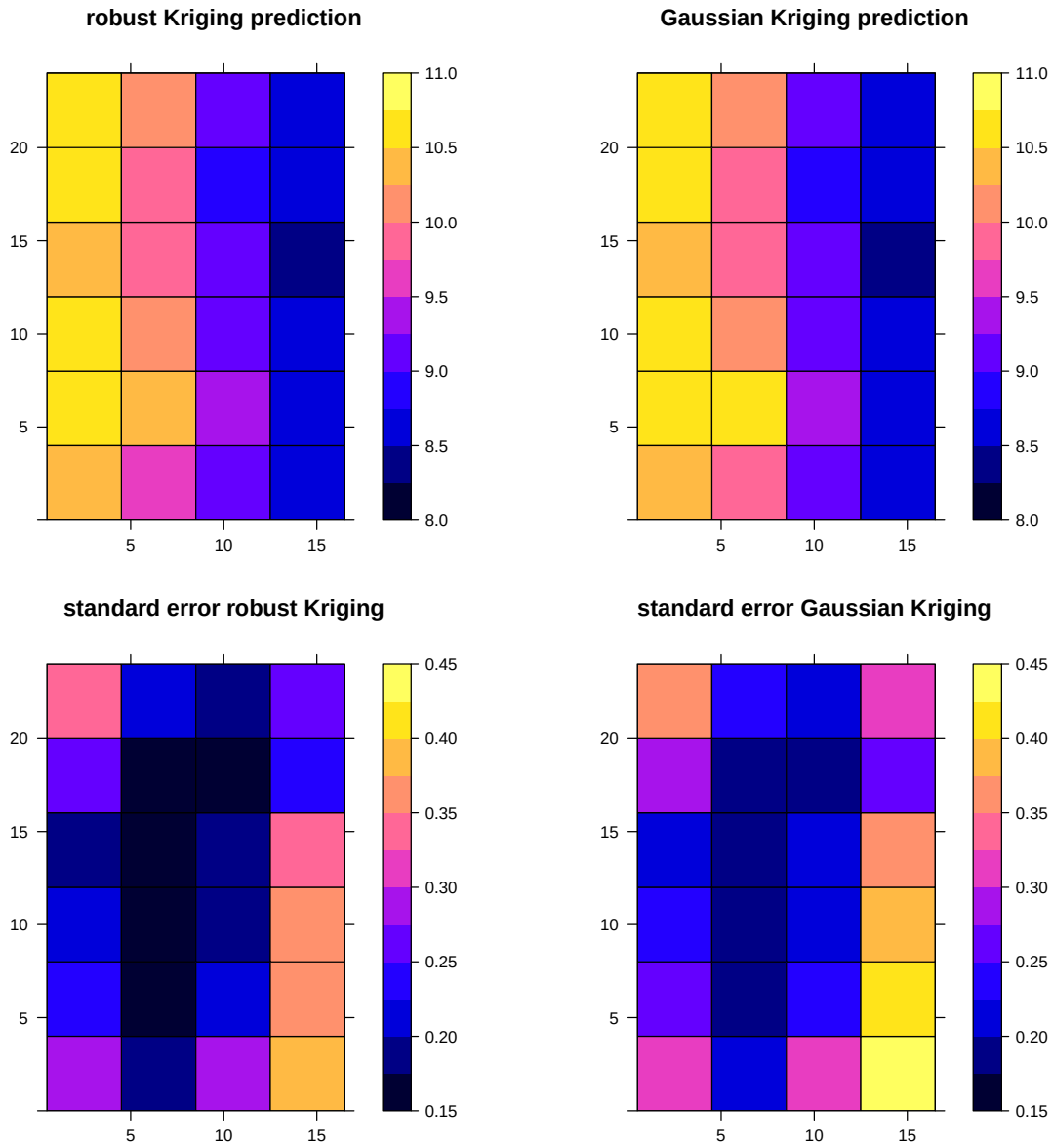

Figure 31: Robust (left) and Gaussian (right) block Kriging predictions (top) and Kriging standard errors (bottom).

The outlier at (5,6) also affects the the prediction of the block means and robust block Kriging is again more efficient than Gaussian block Kriging.

5 Details about parameter estimation

5.1 Implemented variogram models

Currently, estimation of the parameters of the following models is implemented (see argument `variogram.model` of `georob()`):
`"RMaskey"`, `"RMbessel"`, `"RMcauchy"`, `"RMcircular"`, `"RMcubic"`, `"RMdagum"`, `"RMdampedcos"`, `"RMdewijsian"`, `"RMexp"` (default), `"RMfbm"`, `"RMgauss"`, `"RMgencauchy"`, `"RMgenfbm"`, `"RMgengneiting"`, `"RMgneiting"`, `"RMlgd"`, `"RMmatern"`, `"RMpenta"`, `"RMqexp"`, `"RMspheric"`, `"RMstable"`, `"RMwave"`, `"RMwhittle"`.

Some of these models have in addition to `variance`, `snugget`, `nugget` and `scale` further parameters. Initial values of these parameters (`param`) and fitting flags (`fit.param`) must be passed to `georob()` by the same names as used by the functions `RM...()` of the package **RandomFields** (see `RMmodel()`). Use the function `param.names()` to list additional parameters of a given variogram.model.

The arguments `fit.param` and `fit.aniso` are used to control what variogram and anisotropy parameters are estimated and which are kept at the constant initial values. The functions `default.fit.param()` and `default.fit.aniso()` set reasonable default values for these arguments. Note, as an aside, that the function `default.aniso()` sets (default) values of the anisotropy parameters for an isotropic variogram.

5.2 Estimating parameters of power function variogram

The intrinsic variogram model `RMfbm` is over-parametrized when both the `variance` (plus possibly `snugget`) and the `scale` are estimated. Therefore, to estimate the parameters of this model, `scale` must be kept fixed at an arbitrary value by using `fit.param = default.fit.param(scale = FALSE)`.

5.3 Estimating parameters of geometrically anisotropic variograms

Subsection 2.1 describes how covariances are modelled in general. Here we describe in detail how geometrically anisotropic variogram are parametrized:

The matrix $\boldsymbol{A} = \boldsymbol{A}(\alpha, f_1, f_2; \omega, \phi, \zeta)$ (see equation 4) maps an arbitrary point on an ellipsoidal surface with constant (generalized) covariance in $\mathbb{R}^3$, centred on the origin, and having lengths of semi-principal axes, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$, equal to $|\boldsymbol{p}_1| = \alpha$, $|\boldsymbol{p}_2| = f_1\,\alpha$ and $|\boldsymbol{p}_3| = f_2\,\alpha$, $0 < f_2 \leq f_1 \leq 1$, respectively, onto the surface of the unit ball centred on the origin. The orientation of the ellipsoid is defined by the three angles ω, ϕ and ζ:

ω is the azimuth of $\boldsymbol{p}_1$ (= angle between north and the projection of $\boldsymbol{p}_1$ onto the x-y-plane, measured from north to south positive clockwise in degrees),

ϕ is 90 degrees minus the altitude of $\boldsymbol{p}_1$ (= angle between the zenith and $\boldsymbol{p}_1$, measured from zenith to nadir positive clockwise in degrees), and

ζ is the angle between $\boldsymbol{p}_2$ and the direction of the line, say y', defined by the intersection between the x-y-plane and the plane orthogonal to $\boldsymbol{p}_1$ running through the origin (ζ is measured from y' positive counter-clockwise in degrees).

The transformation matrix is given by

$$\boldsymbol{A} = \begin{pmatrix} 1/\alpha & 0 & 0 \\ 0 & 1/(f_1\,\alpha) & 0 \\ 0 & 0 & 1/(f_2\,\alpha) \end{pmatrix} (\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3) \tag{8}$$

where

$$\begin{aligned} \boldsymbol{C}_1^{\mathrm{T}} &= (\sin\omega\sin\phi, -\cos\omega\cos\zeta - \sin\omega\cos\phi\sin\zeta, \cos\omega\sin\zeta - \sin\omega\cos\phi\cos\zeta) \\ \boldsymbol{C}_2^{\mathrm{T}} &= (\cos\omega\sin\phi, \sin\omega\cos\zeta - \cos\omega\cos\phi\sin\zeta, -\sin\omega\sin\zeta - \cos\omega\cos\phi\cos\zeta) \\ \boldsymbol{C}_3^{\mathrm{T}} &= (\cos\phi, \sin\phi\sin\zeta, \sin\phi\cos\zeta) \end{aligned} \tag{9}$$

To model geometrically anisotropic variograms in $\mathbb{R}^2$ one has to set $\phi = 90$ and $f_2 = 1$, and for $f_1 = f_2 = 1$ one obtains the model for isotropic auto-correlation with range parameter α. Note that for isotropic auto-correlation the software processes data for which d may exceed 3.

Some additional remarks might be helpful:

- The first semi-principal axis points into the direction with the farthest reaching auto-correlation, which is described by the range parameter `scale` (α).
- The ranges in the direction of the second and third semi-principal axes are given by $f_1\alpha$ and $f_2\alpha$, with $0 < f_2 \leq f_1 \leq 1$.
- The default values for `aniso` ($f_1 = 1$, $f_2 = 1$) define an isotropic variogram model.
- Valid ranges for the angles characterizing the orientation of the semi-variance ellipsoid are (in degrees): ω [0, 180], ϕ [0, 180], ζ [-90, 90].

5.4 Estimating variance of micro-scale variation

Simultaneous estimation of the variance of the micro-scale variation (`snugget`, σ_{n}^2), which appears seemingly as spatially uncorrelated with a given sampling design, and of the variance (`nugget`, τ^2) of the independent errors requires that for some locations $\boldsymbol{s}_i$ replicated observations are available. Locations less or equal than `zero.dist` apart are thereby considered as being coincident (see `control.georob()`).

5.5 Estimating variance parameters by Gaussian (RE)ML

Unlike robust REML, where robustified estimating equations are solved for the variance parameters `nugget` (τ^2), `variance` (σ^2), and possibly `snugget` (σ_{n}^2), for Gaussian (RE)ML the variances can be re-parametrized to

- the signal variance $\sigma_Z^2 = \sigma^2 + \sigma_{\mathrm{n}}^2$,
- the inverse relative nugget $\eta = \sigma_Z^2/\tau^2$ and
- the relative auto-correlated signal variance $\xi = \sigma^2/\sigma_Z^2$.

`georob()` maximizes then a (restricted) *profile log-likelihood* that depends only on η, ξ, α, ..., and σ_Z^2 is estimated by an explicit expression that depends on these parameters (e.g. Diggle and Ribeiro, 2007, p. 113). This is usually more efficient than maximizing the (restricted) log-likelihood with respect to the original variance parameters τ^2, σ_{n}^2 and σ^2. `georob()` chooses the parametrization automatically, but the user can control it by the argument `reparam` of the function `control.georob()`.

5.6 Constraining estimates of variogram parameters

Parameters of variogram models can vary only within certain bounds (see `param.bounds()` and `RMmodel()` for allowed ranges). `georob()` uses three mechanisms to constrain parameter estimates to permissible ranges:

1. *Parameter transformations*: By default, all variance (`variance`, `snugget`, `nugget`), the range `scale` and the anisotropy parameters `f1` and `f2` are log-transformed before solving the estimating equations or maximizing the restricted log-likelihood and this warrants that the estimates are always positive (see `control.georob()` and section 5.9.4 for detailed explanations how to control parameter transformations).

2. *Checking permissible ranges*: The additional parameters of the variogram models such as the smoothness parameter ν of the Whittle-Matérn model are forced to stay in the permissible ranges by signalling an error to `nleqslv()`, `nlminb()` or `optim()` if the current trial values are invalid. These functions then graciously update the trial values of the parameters and carry their task on. However, it is clear that such a procedure likely gets stuck at a point on the boundary of the parameter space and is therefore just a workaround for avoiding runtime errors due to invalid parameter values.

3. *Exploiting the functionality of* `nlminb()` *and* `optim()`: If a spatial model is fitted non-robustly, then the arguments `lower`, `upper` (and `method` of `optim()`) can be used to constrain the parameters (see `control.optim()` how to pass them to `optim()`). For `optim()` one has to use the arguments `method = "L-BFGS-B"`, `lower = l`, `upper = u`, where l and u are numeric vectors with the lower and upper bounds of the *transformed* parameters in the order as they appear in `c(variance, snugget, nugget, scale, ...)[fit.param], aniso[fit.aniso])`, where `...` are additional parameters of isotropic variogram models (use `param.names(variogram.model)` to display the names and the order of the additional parameters for `variogram.model`).

5.7 Computing robust initial estimates of parameters for robust REML

To solve the robustified estimating equations for $\boldsymbol{B}$ and $\boldsymbol{\beta}$ the following initial estimates are used:

- $\widehat{\boldsymbol{B}} = \mathbf{0}$, if this turns out to be infeasible, initial values can be passed to `georob()` by the argument `bhat` of `control.georob()`.
- $\widehat{\boldsymbol{\beta}}$ is either estimated robustly by the function `lmrob()`, `rq()` or non-robustly by `lm()` (see argument `initial.fixef` of `control.georob()`).

Finding the roots of the robustified estimating equations of the variogram and anisotropy parameters is more sensitive to a good choice of initial values than maximizing the Gaussian (restricted) log-likelihood with respect to the same parameters. If the initial values for `param` and `aniso` are not sufficiently close to the roots of the system of nonlinear equations, then `nleqslv()` may fail to find them. Setting `initial.param = TRUE` (see `control.georob()`) allows one to find initial values that are often sufficiently close to the roots so that `nleqslv()` converges. This is achieved by:

1. Initial values of the regression parameters are computed by `lmrob()` irrespective of the choice for `initial.fixef` (see `control.georob()`).

2. Observations with "robustness weights" of the `lmrob` fit, satisfying $\psi_c(\widehat{\varepsilon}_i/\widehat{\tau})/(\widehat{\varepsilon}_i/\widehat{\tau}) \leq$ `min.rweight`, are discarded (see `control.georob()`).

3. The model is fit to the pruned data set by Gaussian REML using `optim()` or `nlminb()`.

4. The resulting estimates of the variogram parameters (`param`, `aniso`) are used as initial estimates for the subsequent robust fit of the model by `nleqslv()`.

Note that for step 3 above, initial values of `param` (and possibly `aniso`) must be provided to `georob()`.

5.8 Estimating parameters of "nested" variogram models

As a further option, `georob()` allows to estimate parameters of so-called "nested" variogram models. For this one assumes that the signal process $Z(\boldsymbol{s})$ is the sum of several independent auto-correlated Gaussian processes $B_k(\boldsymbol{s})$

$$B(\boldsymbol{s}) = \sum_{k=1}^{K} B_k(\boldsymbol{s}), \tag{10}$$

each characterized by a parametric variogram function with parameters ($\sigma^2_{Z,k}$, $\boldsymbol{\theta}_k$), see equation 3. Initial values for nested variogram models are passed to `georob()` by the argument `variogram.object`, which must be a list of length K. The kth component of `variogram.object` is itself a list with the mandatory components `variogram.model` and `param` and the optional components `fit.param`, `aniso` and `fit.aniso`. Note that sensible defaults are used if the optional components are missing. Note further that `nugget` and `snugget` may be specified for all k model strctures but these variances are summed-up and assigned to the first model structure ($k = 1$) and all `nugget` and `snugget` are set to zero for $k > 1$.

5.9 Controlling `georob()` by the function `control.georob()`

All control and tuning parameters except the robustness tuning constant c of the ψ_c-function (argument `tuning.psi` of `georob()`) are set by the arguments of the function `control.georob()`, which generates a list that is passed by the `control` argument to `georob()`. This section describes in some detail how to control `georob()` by the various arguments of `control.georob()`.

5.9.1 Gaussian (RE)ML estimation

Gaussian (RE)ML estimates are computed provided `tuning.psi` ≥ 1000. Use the argument `ml.method` to select either `"ML"` or `"REML"` (default) and the argument `reparam` to control whether the re-parametrized (default `TRUE`) or the original variogram parameters are estimated (see section 5.5). The function used to maximize the (restricted) log-likelihood is chosen by the argument `maximizer` (default `"nlminb"`). Use the argument `nlminb` along with the function `control.nlminb()` (or the argument `optim` along

with `control.optim()` to pass arguments to `nlminb()` (or `optim()`), in particular the argument `rel.tol`, which controls convergence. The argument `hessian` controls whether confidence intervals of variogram parameters are computed from the observed Fisher information based on the asymptotic normal distribution of (RE)ML estimates (default `TRUE`, see `summary.georob()`).

5.9.2 Robust REML estimation

The argument `psi.func` selects the ψ_c-function for robust REML. Apart from a shifted and scaled logistic CDF

$$\psi_c(x) = \tanh(x/c)$$

(`"logistic"`, default), the Huber (`"huber"`) or a re-descending ψ_c-function based on the t-distribution (`"t.dist"`)

$$\psi_c(x) = \frac{c^2\, x}{c^2 + x^2}$$

can be used.
The argument `initial.fixef` chooses the function for computing (robust) initial estimates of $\boldsymbol{\beta}$. Possible choices are `"lmrob"` (default) `"rq"` and `"lm"`. Use the argument `lmrob` along with the function `lmrob.control()` (or the argument `rq` along with the function `control.rq()` to pass arguments to `lmrob()` (or `rq()`). The argument `initial.param` controls whether robust initial estimates of the variogram parameters are computed (default `TRUE`, see section 5.7).
For given variogram parameters, the estimating equations for $\boldsymbol{\beta}$ and $\boldsymbol{B}$ are solved by iterated re-weighted least squares (IRWLS). The argument `irwls.maxiter` sets the maximum number of iterations (default 50), and the argument `irwls.ftol` controls convergence: Convergence is assumed if the largest absolute function value at the current root is less than `irwls.ftol` (default 10^{-5}). The current estimates $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{B}}$ are then plugged into the estimating equations for $\boldsymbol{\theta}$ and τ^2 that are solved in turn by `nleqslv()`. Use the argument `nleqslv` along with the function `control.nleqslv()` to pass arguments to `nleqslv()`, in particular `ftol`, which controls convergence for root finding.

5.9.3 Approximation of covariances of fixed and random effects and residuals

The robustified estimating equations of robust REML depend on the covariances of $\widehat{\boldsymbol{B}}$. These covariances (and the covariances of $\boldsymbol{B} - \widehat{\boldsymbol{B}}$, $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\varepsilon}}$, $\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}}$) are approximated by expressions that in turn depend on the variances of ε, $\psi_c(\varepsilon/\tau)$ and the expectation of $\psi_c'(\varepsilon/\tau)$ $(= \partial/\partial\varepsilon\, \psi_c(\varepsilon/\tau))$. The arguments `error.family.estimation`, `error.family.cov.effects` and `error.family.cov.residuals` of `control.georob()` control what parametric distribution for ε is used to compute the variances of ε, $\psi_c(\varepsilon/\tau)$ and the expectation of $\psi_c'(\varepsilon/\tau)$ when

- solving the estimating equations (`error.family.estimation`),
- computing the covariances of $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{B}}$ and $\boldsymbol{B} - \widehat{\boldsymbol{B}}$ (`error.family.cov.effects`) and
- computing the covariances of $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ (`error.family.cov.residuals`).

Possible options are: `"gaussian"` or `"long.tailed"`. In the latter case, the PDF of ε is assumed to be proportional to $1/\tau \exp(-\rho_c(\varepsilon/\tau))$, where $\psi_c(x) = \rho_c'(x)$.

The logical arguments `cov.`*`xxx`* and `full.cov.`*`xxx`* of `control.georob()` control what (co-)variances should be computed by `georob()`. `full.cov.`*`xxx`* controls whether the full covariance matrix (`TRUE`) or only the variances (`FALSE`) are computed.

xxx	(co-)variances of	default `cov.`*`xxx`*	default `full.cov.`*`xxx`*
`bhat`	$\widehat{\boldsymbol{B}}$	`TRUE`	`FALSE`
`betahat`	$\widehat{\boldsymbol{\beta}}$	`TRUE`	–
`delta.bhat`	$\boldsymbol{B} - \widehat{\boldsymbol{B}}$	`TRUE`	`TRUE`
`delta.bhat.betahat`	$\boldsymbol{B} - \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\beta}}$	`TRUE`	–
`ehat`	$\widehat{\boldsymbol{\varepsilon}}$	`TRUE`	`FALSE`
`ehat.p.bhat`	$\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}}$	`FALSE`	`FALSE`

5.9.4 Transformations of variogram parameters for (RE)ML estimation

The arguments `param.tf`, `fwd.tf`, `deriv.fwd.tf`, `bwd.tf` of `control.georob()` define the transformations of the variogram parameters for RE(ML) estimation. Implemented are currently `"log"`, `"logit1"`, `"logit2"`, `"logit3"` (various variants of logit-transformation, see code of function `fwd.transf`) and `"identity"` (= no) transformations. These are the possible values that the many arguments of the function `param.transf()` accept (as quoted character strings) and these are the names of the list components returned by `fwd.transf()`, `dfwd.transf()` and `bwd.transf()`. Additional transformations can be implemented by:

1. Extending the function definitions by arguments like

   ```
   fwd.tf = fwd.transf(my.fun = function(x) your transformation),
   deriv.fwd.tf = dfwd.transf(my.fun = function(x) your derivative),
   bwd.tf = bwd.transf(my.fun = function(x) your back-transformation),
   ```

2. Assigning to a given argument of `param.transf` the name of the new function, e.g. `variance = "my.fun"`.

Note that the values given for the arguments of `param.transf()` must match a name of the functions returned by `fwd.transf()`, `dfwd.transf()` and `bwd.transf()`.

5.9.5 Miscellaneous arguments of `control.georob()`

`control.georob()` has the following additional arguments:

`bhat` initial values for the spatial random effects $\widehat{\boldsymbol{B}}$, with $\widehat{\boldsymbol{B}} = \boldsymbol{0}$ if `bhat` is equal to `NULL` (default).

`force.gradient`
logical controlling whether the estimating equations or the gradient of the Gaussian restricted log-likelihood are evaluated even if all variogram parameters are fixed (default `FALSE`).

`min.condnum` positive numeric. Minimum acceptable ratio of smallest to largest singular value of the model matrix $\boldsymbol{X}$ (default `1.e-12`).

`zero.dist` positive numeric equal to the maximum distance, separating two sampling locations that are still considered as being coincident.

Note that `georob()` can fit models with rank-deficient model matrices. It uses then the Moore-Penrose inverse of the matrix $\boldsymbol{X}\boldsymbol{V}_{\alpha,\xi}^{-1}\boldsymbol{X}$ to compute the projection matrices $\boldsymbol{A}_\alpha$ and $\boldsymbol{P}_\alpha$ (see Künsch *et al.*, in prep.).

5.10 Parallelized computations

Parallelized computations shorten computing time for large data sets ($n > 1000$). `georob()` and other functions of the package therefore use on non-windows OS the function `mclapply()` (forking, package **parallel**) and on windows OS the functions `parLapply` (socket cluster, package **parallel**) as well as `sfLapply()` (socket cluster, package **snowfall**) for parallelized computations. The following tasks may be executed in parallel:

1. Simultaneously fitting multiple models by functions `cv.georob()`, `profilelogLik()` (section 7.2), `add1.georob()`, `drop1.georob()` and `step.georob()` (section 7.1);
2. computing Kriging predictions by `predict.georob()` (section 6.1);
3. matrix multiplication by function `pmm()`;
4. computing the (generalized) covariance matrix $\boldsymbol{\Gamma}_\theta$ of the data by the (non-exported) function `f.aux.gcr()`.

For tasks 1 and 2 the functions have an argument `ncores` to control how many cores should be used for parallel computations. For tasks 3 and 4 one can use the argument `pcmp` of `control.georob()` along with the function `pcmp.control()` to control parallelized computations. The function `pcmp.control()` has the following arguments:

`pmm.ncores` number (integer, default 1) of cores used for parallelized matrix multiplication.

`gcr.ncores` number (integer, default 1) of cores used for parallelized computation of (generalized) covariance matrix.

`max.ncores` allowed maximum number of cores (integer, default all cores of a machine) used for parallelized computations.

`f` number (integer, default 2) of tasks assigned to each core in parallelized computations.

`sfstop` logical controlling whether the SNOW socket cluster is stopped after each parallelized computations on windows OS (default `FALSE`).

`allow.recursive`
logical controlling whether nested parallelized computation should be allowed (default `TRUE`).

Generating child processes requires itself resources and increasing the number of cores for tasks 3 and 4 does not always result in reduced computing time. A sensible default for the number of cores for tasks 3 and 4 is likely 2. `max.ncores` controls how many child processes are generated in total. This can be used to prevent that child processes generate themselves children which may result in too many child processes.

Note, however, that very substantial reductions in computing time results when one uses the **OpenBLAS** library instead of the reference BLAS library that ships with R, see `http://www.openblas.net/` and R FAQ for your OS. With OpenBLAS no gains are obtained by using more than one core for matrix multiplication, and one should therefore use the default argument `pmm.ncores = 1` for `control.pcmp()`.

6 Details about Kriging

6.1 Functionality of `predict.georob()`

The `predict` method of class `georob` computes customary or robust external drift point or block plug-in Kriging predictions (see equation 7). Data about the prediction targets are passed as argument `newdata` to `predict.georob()`, where `newdata` can be either an ordinary **data frame**, a **SpatialPoints-**, **SpatialPixels-**, **SpatialGrid** or a **SpatialPolygons-DataFrame**, a **SpatialPoints**, **SpatialPixels** or a **SpatialGrid** object, all the latter provided by the package **sp**. If `newdata` is a `SpatialPoints`, `SpatialPixels` or a `SpatialGrid` object then the drift model may only use the coordinates as covariates (universal Kriging). If `newdata` is a `SpatialPolygonsDataFrame` then block Kriging predictions are computed, otherwise point Kriging predictions.

6.1.1 Prediction targets

The argument `type` controls what quantities `predict.georob()` computes (given here for a target point $\boldsymbol{s}$, but the same quantities are also computed for a block):

- `"signal"`: the "signal" $Z(\boldsymbol{s}) = \boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\beta} + B(\boldsymbol{s})$ of the process (default),
- `"response"`: the observations $Y(\boldsymbol{s}) = Z(\boldsymbol{s}) + \varepsilon(\boldsymbol{s})$,
- `"trend"`: the external drift $\boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\beta}$,
- `"terms"`: the model terms. The argument `terms` can then be used to select the terms (default all terms) and `se.fit` controls whether standard errors of the terms are computed (default `TRUE`).

6.1.2 Further control

Use the argument `control` along with the function `control.predict.georob()` to further control what `predict.georob()` actually does:

Covariance matrices The argument `full.covmat` of `control.predict.georob()` controls whether the full covariance matrices of prediction errors, fitted trend, etc. are computed (`TRUE`) or only the vector with their variances (`FALSE`, default).

Computing auxiliary items for log-normal Kriging Use the argument `extended.output = TRUE` of `control.predict.georob()` to compute all quantities required for the (approximately) unbiased back-transformation of Kriging predictions of log-transformed data to the original scale of the measurements by `lgnpp()` (see section 6.2). In more detail, the following items are computed:

- `trend`: the fitted values, $\boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\widehat{\boldsymbol{\beta}}$,
- `var.pred`: the variances of the Kriging predictions, $\mathrm{Var}_{\hat{\theta}}[\widehat{Y}(\boldsymbol{s})]$ or $\mathrm{Var}_{\hat{\theta}}[\widehat{Z}(\boldsymbol{s})]$,
- `cov.pred.target`: the covariances between the predictions and the prediction targets, $\mathrm{Cov}_{\hat{\theta}}[\widehat{Y}(\boldsymbol{s}), Y(\boldsymbol{s})]$ or $\mathrm{Cov}_{\hat{\theta}}[\widehat{Z}(\boldsymbol{s}), Z(\boldsymbol{s})]$,
- `var.target`: the variances of the prediction targets $\mathrm{Var}_{\hat{\theta}}[Y(\boldsymbol{s})]$ or $\mathrm{Var}_{\hat{\theta}}[Z(\boldsymbol{s})]$.

Note that the component `var.pred` is also present if `type` is equal to `"trend"`, irrespective of the choice for `extended.output`. This component contains then the variances of the fitted values.

Contents and structure of returned object Depending on the values passed to `type`, `full.covmat` and `extended.output`, `predict.georob()` returns the following object:

1. If `type` is equal to `"terms"` then a vector, a matrix, or a list with prediction results along with bounds and standard errors, see `predict.lm()`.

2. If `type` is not equal to `"terms"` and `full.covmat` is `FALSE` then the result is an object of the same class as `newdata` (data frame, `SpatialPointsDataFrame`, `SpatialPixelsDataFrame` `SpatialGridDataFrame`, `SpatialPolygonsDataFrame`). The data frame or the `data` slot of the `Spatial`***xxx***`DataFrame` objects have the following components:

   - the coordinates of the prediction points (only present if `newdata` is a data frame).
   - `pred`: the Kriging predictions (or fitted values).
   - `se`: the root mean squared prediction errors (Kriging standard errors).
   - `lower`, `upper`: the limits of tolerance/confidence intervals (the confidence level is set by the argument `signif`) of `predict.georob()`),
   - `trend`, `var.pred`, `cov.pred.target`, `var.target`: only present if `extended.output` is `TRUE`, see above.

3. If `type` is not equal to `"terms"` and `full.covmat` is `TRUE` then `predict.georob()` returns a list with the following components:

   - `pred`: a data frame or a `Spatial`***xxx***`DataFrame` object as described above for `full.covmat = FALSE`.
   - `mse.pred`: the full covariance matrix of the prediction errors, $Y(\boldsymbol{s}) - \widehat{Y}(\boldsymbol{s})$ or $Z(\boldsymbol{s}) - \widehat{Z}(\boldsymbol{s})$ see above.
   - `var.pred`: the full covariance matrix of the Kriging predictions, see above.
   - `cov.pred.target`: the full covariance matrix of the predictions and the prediction targets, see above.
   - `var.target`: the full covariance matrix of the prediction targets, see above.

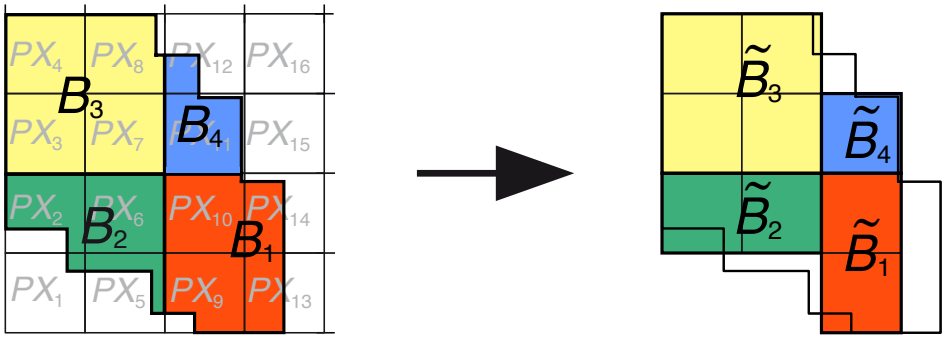

Figure 32: Approximation of four blocks $B_1, \ldots, B_4$ by a group of 16 pixels $PX_1 \ldots, PX_{16}$. The original geometry of blocks with the grid of the pixels is shown on the left and the approximation on the right.

6.1.3 Block Kriging

`georob.predict()` uses functions of the package **constrainedKriging** (Hofer and Papritz, 2011) for efficiently computing the required integrals of the covariance function. The target blocks are approximated for this by sets of *rectangular pixels* arranged on a regular lattice (Figure 32). The integrals can then be computed very efficiently because the PDF of the distance between two points, that are uniformly distributed in two rectangles on a regular lattice, can be computed by closed-form expressions (Clifford, 2005). Also the PDF of the distance between a fixed point and a point uniformly distributed in a rectangle is available in closed form (Hofer and Papritz, 2011). Hence, for a two-dimensional study region, only 1-dimensional integrals of the covariance function must be evaluated numerically instead of 2- and 4-dimensional integrals (see Hofer and Papritz, 2011, for details).

The arguments `pwidth` and `pheight` of `control.predict.georob()` define the dimension of the pixel used for the approximation of the blocks, and `napp` defines how many repetitions of the approximation by randomly placed grid of pixels should be averaged (see help page of the function `preCKrige` of package **constrainedKriging** for more details).

For the time being **constrainedKriging** does not integrate the following (generalized) covariances : `RMaskey`, `RMdagum`, `RMdewijsian`, `RMfbm` and `RMgenfbm`. Hence, these models cannot be used for block Kriging.

6.1.4 Parallelized computations

`predict.georob()` computes its results in parallel. The parallelization is controlled by the arguments `mmax` and `ncores` of `control.predict.georob()` (and `pcmp` along with the function `control.pcmp()`, see section 5.10): If there are m items to compute, the task is split into `ceiling(m/mmax)` sub-tasks that are then distributed to `ncores` CPUs. Evidently, `ncores = 1` suppresses parallel execution. By default, the function uses all available CPUs as returned by `detectCores()`. Note that if `full.covmat` is `TRUE` `mmax` must exceed m (and parallel execution is not possible).

6.2 Lognormal Kriging

The function `lgnpp()` back-transforms point or block Kriging predictions of a log-transformed response variable computed by `predict.georob()`. Alternatively, the

function averages log-normal point Kriging predictions for a block and approximates the mean squared prediction error of the block mean. The items required for the back-transformation are computed by `predict.georob()` if the argument `control = control.georob.predict(extended.output = TRUE)` is used, see section 6.1.

`lgnpp()` then performs the following three tasks:

6.2.1 Back-transformation of point Kriging predictions of a log-transformed response

The usual, marginally unbiased back-transformation for log-normal point Kriging is used:

$$\widehat{U}(\boldsymbol{s}) = \exp(\widehat{Z}(\boldsymbol{s}) + 1/2(\mathrm{Var}_{\hat\theta}[Z(\boldsymbol{s})] - \mathrm{Var}_{\hat\theta}[\widehat{Z}(\boldsymbol{s})])), \tag{11}$$

$$\begin{aligned}\mathrm{Cov}_{\hat\theta}[U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i), U(\boldsymbol{s}_j) - \widehat{U}(\boldsymbol{s}_j)] = \mu_{\hat\theta}(\boldsymbol{s}_i)\mu_{\hat\theta}(\boldsymbol{s}_j)\{ &\exp(\mathrm{Cov}_{\hat\theta}[Z(\boldsymbol{s}_i), Z(\boldsymbol{s}_j)]) \\ -2\exp(\mathrm{Cov}_{\hat\theta}[\widehat{Z}(\boldsymbol{s}_i), Z(\boldsymbol{s}_j)]) &+ \exp(\mathrm{Cov}_{\hat\theta}[\widehat{Z}(\boldsymbol{s}_i), \widehat{Z}(\boldsymbol{s}_j)])\},\end{aligned} \tag{12}$$

where $\widehat{Z}$ and $\widehat{U}$ denote the log- and back-transformed predictions of the signal, and

$$\mu_{\hat\theta}(\boldsymbol{s}) \approx \exp(\boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\widehat{\boldsymbol{\beta}} + 1/2\mathrm{Var}_{\hat\theta}[Z(\boldsymbol{s})]). \tag{13}$$

The expressions for the required covariance terms can be found in the Appendices of Nussbaum *et al.* (2012, 2014). Instead of the signal $Z(\boldsymbol{s})$, predictions of the log-transformed response $Y(\boldsymbol{s})$ or the estimated trend $\boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\widehat{\boldsymbol{\beta}}$ of the log-transformed data can be back-transformed. The above transformations are used if the `object` passed as first argument to `lgnpp()` contains point Kriging predictions and if the argument `is.block = FALSE` and the argument `all.pred` is missing.

6.2.2 Back-transformation of block Kriging predictions of a log-transformed response

Block Kriging predictions of a log-transformed response variable are back-transformed by the approximately unbiased transformation proposed by Cressie (2006, Appendix C)

$$\widehat{U}(A) = \exp(\widehat{Z}(A) + 1/2\{\mathrm{Var}_{\hat\theta}[Z(\boldsymbol{s})] + \widehat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{M}(A)\widehat{\boldsymbol{\beta}} - \mathrm{Var}_{\hat\theta}[\widehat{Z}(A)]\}), \tag{14}$$

$$\begin{aligned}\mathrm{E}_{\hat\theta}[\{U(A) - \widehat{U}(A))^2] \;=\; &\mu_{\hat\theta}(A)^2\{\exp(\mathrm{Var}_{\hat\theta}[Z(A)]) \\ &-2\exp(\mathrm{Cov}_{\hat\theta}[\widehat{Z}(A), Z(A)]) + \exp(\mathrm{Var}_{\hat\theta}[\widehat{Z}(A)])\}\end{aligned} \tag{15}$$

where $\widehat{Z}(A)$ and $\widehat{U}(A)$ are the log- and back-transformed predictions of the block mean $U(A)$, respectively, $\boldsymbol{M}(A)$ is the spatial covariance matrix of the covariates

$$\boldsymbol{M}(A) = 1/|A| \int_A (\boldsymbol{x}(\boldsymbol{s}) - \boldsymbol{x}(A))(\boldsymbol{x}(\boldsymbol{s}) - \boldsymbol{x}(A))^{\mathrm{T}}\, d\boldsymbol{s} \tag{16}$$

within the block A, where

$$\boldsymbol{x}(A) = 1/|A| \int_A \boldsymbol{x}(\boldsymbol{s})\, d\boldsymbol{s} \tag{17}$$

and

$$\mu_{\hat{\theta}}(A) \approx \exp(\boldsymbol{x}(A)^{\mathrm{T}}\widehat{\boldsymbol{\beta}} + 1/2\mathrm{Var}_{\hat{\theta}}[Z(A)]). \tag{18}$$

This back-transformation is based on the assumption that both the point data $U(\boldsymbol{s})$ and the block means $U(A)$ follow log-normal laws, which strictly cannot hold. But for small blocks the assumption works well as the bias and the loss of efficiency caused by this assumption are small (Cressie, 2006; Hofer *et al.*, 2013).

The above formulae are used by `lgnpp()` if `object` contains block Kriging predictions in the form of a `SpatialPolygonsDataFrame`. To approximate $\boldsymbol{M}(A)$, one needs the covariates on a fine grid for the whole study domain in which the blocks lie. The covariates are passed to `lgnpp()` as argument `newdata`, where `newdata` can be any spatial data frame accepted by `predict.georob`. For evaluating $\boldsymbol{M}(A)$ the geometry of the blocks is taken from the `polygons` slot of the `SpatialPolygonsDataFrame` passed as `object` to `lgnpp()`.

6.2.3 Back-transformation and averaging of point Kriging predictions of a log-transformed response

`lgnpp()` allows as a further option to back-transform and *average* point Kriging predictions passed as `object` to the function (optimal log-normal block Kriging, see Cressie, 2006). One then assumes that the predictions in `object` refer to points that lie in *a single* block. Hence, one uses the approximation

$$\widehat{U}(A) \approx \frac{1}{K}\sum_{s_i \in A}\widehat{U}(\boldsymbol{s}_i) \tag{19}$$

to predict the block mean $U(A)$, where K is the number of points in A. The mean squared error of prediction can be approximated by

$$\mathrm{E}_{\hat{\theta}}[\{U(A) - \widehat{U}(A)\}^2] \approx \frac{1}{K^2}\sum_{s_i \in A}\sum_{s_j \in A}\mathrm{Cov}_{\hat{\theta}}[U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i), U(\boldsymbol{s}_j) - \widehat{U}(\boldsymbol{s}_j)]. \tag{20}$$

In most instances, the evaluation of the above double sum is not feasible because a large number of points is used to discretize the block A. `lgnpp()` then uses the following approximations to compute the mean squared error (see Nussbaum *et al.*, 2012, 2014, Appendices):

- Point prediction results are passed as `object` to `lgnpp()` only for a *random sample of points in* A (of size k), for which the evaluation of the above double sum is feasible.
- The prediction results for the *complete set of points* within the block are passed as argument `all.pred` to `lgnpp`. These results are used to compute $\widehat{U}(A)$.
- The mean squared error is then approximated by

$$\begin{aligned}\mathrm{E}_{\hat{\theta}}[\{U(A) - \widehat{U}(A)\}^2] \approx{}& \frac{1}{K^2}\sum_{s_i \in A}\mathrm{E}_{\hat{\theta}}[\{U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i)\}^2] \\ &+ \frac{K-1}{Kk(k-1)}\sum_{s_i \in \text{sample}}\sum_{s_j \in \text{sample}, s_j \neq s_i}\mathrm{Cov}_{\hat{\theta}}[U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i), U(\boldsymbol{s}_j) - \widehat{U}(\boldsymbol{s}_j)]. \end{aligned} \tag{21}$$

The first term of the RHS (and $\widehat{U}(A)$) can be computed from the point Kriging results contained in `all.pred`, and the double sum is evaluated from the full covariance matrices of the predictions and the respective targets, passed to `lgnpp()` as `object` (one has to use the arguments `control=control.predict.georob(full.covmat=TRUE)` for `predict.georob()` when computing the point Kriging predictions stored in `object`).

- If the prediction results are not available for the complete set of points in A then `all.pred` may be equal to K. The block mean is then approximated by

$$\widehat{U}(A) \approx \frac{1}{k} \sum_{s_i \in \text{sample}} \widehat{U}(\boldsymbol{s}_i) \tag{22}$$

and the first term of the RHS of the expression for the mean squared error by

$$\frac{1}{kK} \sum_{s_i \in \text{sample}} \mathrm{E}_{\hat{\theta}}[\{U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i)\}^2]. \tag{23}$$

- By drawing samples repeatedly and passing the related Kriging results as `object` to `lgnpp()`, one can reduce the error of the approximation of the mean squared error.

7 Building models and assessing fitted models

7.1 Model building

Wald tests The `waldtest` method for class `georob` can be used to test hypotheses about the fixed effects of a model. Note that this function uses *conditional* F- or χ^2-tests (Pinheiro and Bates, 2000, section 2.4.2), i.e. it fixes the variogram parameters at the values of the more general model of each comparison (see help page of function `waldtest()` of package **lmtest** for details).

Besides `waldtest.georob()` the functions of the package **multcomp** can be used to test general linear hypotheses about the fixed effects of the model.

Log-likelihood and AIC The `deviance` method for class `georob` returns for Gaussian (RE)ML fits the *residual deviance*

$$(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}})^{\mathrm{T}}(\widehat{\tau}^2\boldsymbol{I} + \boldsymbol{\Gamma}_{\widehat{\theta}})^{-1}(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}),$$

and the `logLik` and `extractAIC` methods extract for class `georob` the respective goodness-of-fit criteria, depending on the argument `REML` either for ML (default) or REML.

For a robust REML fit the deviance is not defined because there is no robustified log-likelihood. `deviance.georob()` then computes (with a warning) the residual deviance of an equivalent Gaussian model with heteroscedastic nugget effect $\widehat{\tau}^2/\boldsymbol{w}$, where $\boldsymbol{w}$ are the "robustness weights"

$$w_i = \psi_c(\widehat{\varepsilon}_i/\widehat{\tau})/(\widehat{\varepsilon}_i/\widehat{\tau}).$$

For robust REML, `logLik()` and `extractAIC()` return the respective criteria also for the equivalent Gaussian model with heteroscedastic nugget effect.

Stepwise selection of covariates The `add1` and `drop1` methods for class `georob` compute all the single terms that can be added or dropped from the model according to their `scope` argument. By default, the variogram parameters are kept fixed at the values fitted for `object`. Use the argument `fixed = FALSE` if the variogram parameters should be re-estimate afresh for each evaluated term. Then either the variogram parameters in `object$initial.objects` (`use.fitted.param = FALSE`) or the fitted parameters of `object` (`use.fitted.param = TRUE`) are used as initial values.

The `step` method for class `georob`[1] allows even finer control whether the variogram parameters are kept fixed or re-estimated when evaluating single terms. Two argument of `step.georob()` control the behaviour:

`fixed.add1.drop1`
: logical controlling whether the variogram parameters are *not* adjusted when `add`ing or `drop`ping model terms by `add1.georob()` and `drop1.georob()` (default `TRUE`).

`fixed.step`
: logical controlling whether the variogram parameters are *not* adjusted after having called `add1.georob()` and `drop1.georob()` in a cycle of `step.georob()` (default `TRUE`). For `fixed.step = FALSE` the parameters are estimated afresh for the new model that was chosen in the previous cycle.

[1] The package **georob** provides a generic `step` function and re-defines the function `step()` of the package **stats** as `default` method.

Of course, model building based on AIC is only sound for Gaussian (RE)ML as there is no log-likelihood for robust REML. For robust REML fits `add1.georob()`, `drop1.georob()` and `step.georob()` use AIC values evaluated for an equivalent Gaussian model with heteroscedastic nugget effect, see above, and it is currently not known whether this is a valid approach for model building.

A last remark: Use the argument `ncores` to let `add1.georob()` and `drop1.georob()` evaluate single terms in parallel.

7.2 Assessing fitted models

7.2.1 Model diagnostics

The methods required for residual diagnostics (`fitted`, `residuals`, `rstandard`, `ranef`) work for objects of class `georob` (either as `default` or specific `georob` method).

Note that the are two kind of residuals: `residuals.georob()` extracts either the estimated independent errors $\widehat{\varepsilon}(\boldsymbol{s})$ or the sum of the latter quantities and the spatial random effects $\widehat{B}(\boldsymbol{s})$ (regression residuals). `rstandard.georob()` does the same but standardizes the residuals to unit variance. `ranef.georob()` extracts $\widehat{B}(\boldsymbol{s})$ with the option to standardize them as well (by argument `standard`).

Diagnostics plots are created by the `plot` method for class `georob`: Depending on the value of the argument `type` the following plots are created:

- `"variogram"`: the estimated variogram (default),
- `"covariance"`: the estimated covariance function,
- `"correlation"`: the estimated correlation function,
- `"scale-location"`: square root of absolute regression residuals plotted against fitted values (Scale-Location plot),
- `"ta"`: regression residuals plotted against fitted values (Tukey-Anscombe plot),
- `"qq.res"`: normal QQ plot of standardized errors $\hat{\varepsilon}$,
- `"qq.ranef"`: normal QQ plot of standardized random effects $\hat{B}$.

7.2.2 Log-likelihood profiles

The function `profilelogLik()` computes for an array of fixed values of variogram parameters the profile log-likelihood by maximizing the (restricted) log-likelihood with respect to the remaining variogram parameters, the fixed and random effects. Of course, the maximized profile log-likelihood values are meaningful only for Gaussian (RE)ML fits (for robust fits the calculated values refer to the equivalent Gaussian model with heteroscedastic nugget effect, see above). Use the argument `ncores` to fit multiple models in parallel.

`profilelogLik()` uses the function `update` to re-estimated the model with partly fixed variogram parameters. Therefore, any argument accepted by `georob()` (except `data`) can be changed when fitting the model. Some of them (e.g. `verbose`) are explicit arguments of `profilelogLik`, but also the remaining ones can be passed by `...` to the function.

`profilelogLik()` returns its results as a data frame. Customary graphics functions can be used for display of log-likelihood profiles and dependence of estimated parameters on each other.

7.3 Cross-validation

7.3.1 Computing cross-validation predictions

The function `cv.georob()` assesses the goodness-of-fit of a spatial linear model by K-fold cross-validation. In more detail, the model is fitted K times by robust (or Gaussian) (RE)ML, excluding each time *1/K*th of the data. The fitted models are used to compute robust (or customary) external Kriging predictions for the omitted observations. If the response variable is log-transformed then the Kriging predictions can be optionally transformed back to the original scale of the measurements. Use the argument `lgn = TRUE` for this.
Practitioners in geostatistics commonly cross-validate a fitted model without re-estimating the model parameters with the reduced data sets. This is clearly an unsound practice (Hastie *et al.*, 2009, section 7.10). Therefore, the argument `re.estimate` should always be set to `TRUE`. The alternative is provided only for historic reasons. The argument `return.fit` and `reduced.output` control whether results of the model fit are returned by `cv.georob()`.

By default, `cv.georob()` fits the models in parallel to the cross-validation sets. Use the argument `ncores` to control parallelized computations.

Defining the cross-validation subsets The argument `method` controls how the data set is partitioned into the K subsets: For `method = "block"` (default) the function `kmeans()` is used to form geographically compact subsets of data locations and for `method = "random"` simple random sampling is used to form the subsets. In analogy to the block bootstrap in time series analysis (Künsch, 1989), the first method should be preferred for model assessment, while the latter might be more informative for assessing prediction precision. Instead of using `method` (along with the argument `seed`) to form the subsets, the argument `sets` can be used to pass the definition of a partition to `cv.georob()`.
Irrespective of the method used to define the subsets, coincident sampling locations are assigned to the same subset, except when one uses the argument `duplicate.in.same.set = FALSE`.

Further control When the external drift model contains factors it may happen that observations are missing for some factor levels in some of the subsets. The argument `mfl.action` controls what then happens: For `mfl.action = "stop"` `cv.georob()` stops with an error message. For `mfl.action = "offset"` the effect of the respective factor (estimated with all the data) is treated as an `offset` term and `cv.georob()` estimates only the remaining terms of the external drift model.

`cv.georob()` uses the function `update()` to re-estimated the model with the subsets. Therefore, any argument accepted by `georob()` except `data` can be changed when refitting the model. Some of them (e.g. `formula`, `subset`, etc.) are explicit arguments of `cv.georob()`, but also the remaining ones can be passed by `...` to the function.

Sometimes, the estimated variograms differ considerably between the cross-validation subsets. Using common initial values for estimating the model is then numerically inefficient. Therefore, the arguments `param` and `aniso` accept in addition to vectors of initial variogram parameters matrices with K rows of initial values. The ith row of the matrix then contains the initial variogram parameters that are used to fit the model to the ith cross-validation subset.

7.3.2 Criteria for assessing (cross-)validation prediction errors

The function `validate.predictions()` computes the items required to evaluate (and plot) the diagnostic criteria proposed by Gneiting *et al.* (2007) for assessing the *calibration* and the *sharpness* of probabilistic predictions of (cross-)validation data. To this aim, `validate.predictions()` uses the assumption that the prediction errors $Y(\boldsymbol{s}) - \widehat{Y}(\boldsymbol{s})$ follow normal distributions with zero mean and standard deviations equal to the Kriging standard errors. This assumption is an approximation if the errors ε come from a long-tailed distribution. Furthermore, for the time being, the Kriging variance of the *response* Y is approximated by adding the estimated nugget $\widehat{\tau}^2$ to the Kriging variance of the signal Z. This approximation likely underestimates the mean squared prediction error of the response if the errors come from a long-tailed distribution. Hence, for robust Kriging, the standard errors of the (cross-)validation errors are likely too small.

Notwithstanding these difficulties and imperfections, `validate.predictions()` computes

- the *probability integral transform* (PIT),

$$\mathrm{PIT}_i = F_i(y_i), \tag{24}$$

  where $F_i(y_i)$ denotes the (plug-in) predictive CDF evaluated at y_i, the value of the ith (cross-)validation datum,

- the *average predictive CDF*

$$\bar{F}_n(y) = 1/n \sum_{i=1}^{n} F_i(y), \tag{25}$$

  where n is the number of (cross-)validation observations and the F_i are evaluated at N quantiles equal to the set of distinct y_i (or a subset of size N of them),

- the *Brier Score*

$$\mathrm{BS}(y) = 1/n \sum_{i=1}^{n} \left(F_i(y) - I(y_i \leq y)\right)^2, \tag{26}$$

  where $I(x)$ is the indicator function for the event x, and the Brier score is again evaluated at the unique values of the (cross-)validation observations (or a subset of size N of them),

- the *averaged continuous ranked probability score*, CRPS, a strictly proper scoring criterion to rank predictions, which is related to the Brier score by

$$\mathrm{CRPS} = \int_{-\infty}^{\infty} \mathrm{BS}(y)\, dy. \tag{27}$$

Gneiting *et al.* (2007) proposed the following plots to validate probabilistic predictions:

- A histogram (or a plot of the empirical CDF) of the PIT values. For ideal predictions, with observed coverages of prediction intervals matching nominal coverages, the PIT values have an uniform distribution.
- Plots of $\bar{F}_n(y)$ and of the empirical CDF of the data, say $\widehat{G}_n(y)$, and of their difference, $\bar{F}_n(y) - \widehat{G}_n(y)$ vs. y. The forecasts are said to be *marginally calibrated* if $\bar{F}_n(y)$ and $\widehat{G}_n(y)$ match.
- A plot of $\mathrm{BS}(y)$ vs. y. Probabilistic predictions are said to be *sharp* if the area under this curve, which equals CRPS, is minimized.

The `plot()` method for class `cv.georob` allows to create these plots, along with scatterplots of observations and predictions, Tukey-Anscombe and normal QQ plots of the standardized prediction errors, and `summary.cv.georob()` computes the mean and dispersion statistics of the (standardized) (cross-)validation prediction errors.

References

Bivand, R. S., Pebesma, E. J., and Gómez-Rubio, V. (2013). *Applied Spatial Data Analysis with R*. Springer, New York, second edition.

Clifford, D. (2005). Computation of spatial covariance matrices. *Journal of Computational and Graphical Statistics*, **14**(1), 155–167.

Cressie, N. (2006). Block kriging for lognormal spatial processes. *Mathematical Geology*, **38**(4), 413–443.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley & Sons, New York, revised edition.

Diggle, P. J. and Ribeiro, Jr., P. J. (2007). *Model-based Geostatistics*. Springer, New York.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B*, **69**(2), 243–268.

Gomez, M. and Hazen, K. (1970). Evaluating sulfur and ash distribution in coal seams by statistical response surface regression analysis. Report of investigations 7377, United States Department of the Interior, Bureau of Mines, [Washington, D.C.]. http://hdl.handle.net/2027/mdp.39015078532879.

Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, **72**, 320–340.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer, New York, second edition.

Hofer, C. and Papritz, A. (2011). constrainedKriging: An R-package for customary, constrained and covariance-matching constrained point or block kriging. *Computers & Geosciences*, **37**(10), 1562–1569.

Hofer, C., Borer, F., Bono, R., Kayser, A., and Papritz, A. (2013). Predicting topsoil heavy metal content of parcels of land: An empirical validation of customary and constrained lognormal block kriging and conditional simulations. *Geoderma*, **193–194**, 200–212.

Künsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, **17**(3), 1217–1241.

Künsch, H. R., Papritz, A., Schwierz, C., and Stahel, W. A. (2011). Robust estimation of the external drift and the variogram of spatial data. Proceedings of the ISI 58th World Statistics Congress of the International Statistical Institute.

Künsch, H. R., Papritz, A., Schwierz, C., and Stahel, W. A. (in prep.). Robust geostatistics.

Lark, R. M. (2000). A comparison of some robust estimators of the variogram for use in soil survey. *European Journal of Soil Science*, **51**, 137–157.

Maronna, R. A., Martin, R. D., and Yohai, V. J. (2006). *Robust Statistics Theory and Methods*. John Wiley & Sons, Chichester.

Nussbaum, M., Papritz, A., Baltensweiler, A., and Walthert, L. (2012). Organic carbon stocks of swiss forest soils. Final report, Institute of Terrestrial Ecosystems, ETH Zürich and Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Zürich and Birmensdorf. http://e-collection.library.ethz.ch/eserv/eth:6027/eth-6027-01.pdf.

Nussbaum, M., Papritz, A., Baltensweiler, A., and Walthert, L. (2014). Estimating soil organic carbon stocks of swiss forest soils by robust external-drift kriging. *Geoscientific Model Development*, **7**(3), 1197–1210.

Pebesma, E. J. (2004). Multivariable geostatistics in S: the gstat package. *Computers & Geosciences*, **30**, 683–691.

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer Verlag.