4.6 Spatial regression

Allow for spatial correlation in the residuals of the trend surface model on p. 187: Replace i.i.d. residual process \( \epsilon(s) \) by a stationary zero-mean process \( U(s) \)

\[
Y(s) = x^T(s)\beta + U(s),
\]

where \( \text{E}(U(s)) \equiv 0 \) and \( \text{Cov}(U(s), U(s')) = C(s - s') \).

Generalised least squares

Suppose we knew the true covariogram \( C \). For a given set of sample points \( s_1, s_2, \ldots, s_n \) this would then define the covariance matrix \( \Sigma \) of the residual vector \( U \) as explained on p. 237.

The generalised least squares (GLS) estimators of coefficients \( \hat{\beta} \) and their associated standard errors, taking into account spatial correlation, are then obtained from

\[
\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y \quad \text{Var}(\hat{\beta}) = (X^T \Sigma^{-1} X)^{-1},
\]

where \( y \) is the response vector and \( X \) the design matrix as on p. 187.
An example

Let us consider the following toy example. Data were created by simulating the responses $Y(s_i)$ at the given locations $s_i = (s_{i1}, s_{i2})$ from a stationary Gaussian process with $\mu = 0$, $\sigma^2 = 1$, $\tau^2 = 0$ and $\rho(h) = \exp(-h/\phi)^2$, the Gaussian correlation function with $\phi = 3$.

$$
\begin{array}{cccc}
  i & s_{i1} & s_{i2} & Y(s_i) \\
  1 & 0 & 0 & -0.39 \\
  2 & 1 & 0 & -0.85 \\
  3 & 0 & 1 & -0.99 \\
  4 & 1 & 1 & -1.21 \\
  5 & 10 & 10 & 0.05 \\
\end{array}
$$

```r
> sim4.coords <- rbind(expand.grid(0:1,0:1),c(10,10))
> library(geoR)
> sim4 <- grf(grid=sim4.coords, cov.model='gau', cov.pars=c(1,3))
> sim4$data <- round(sim4$data,2)
```

Suppose we wish to estimate $\mu$ from this data set (assuming stationarity). The OLS estimator, ignoring spatial correlation would be simply the mean of $Y(s_i)$.

```r
> mean(sim4$data)
[1] -0.678
```

Note that we could have computed this with `surf.ls` as well:

```r
> library(spatial)
> ssim4 <- data.frame(x=sim4$coords[,1],y=sim4$coords[,2],
+                     z=sim4$data)
> surf.ls(0,ssim4)$beta
[1] -0.678
```
The standard error of $\hat{\mu} = -0.678$, estimated under the i.i.d. assumption, is $\sqrt{\hat{\sigma}^2/n}$ (p. 184 & Math ex. V)

```r
> sqrt(var(sim4$data)/5)
[1] 0.2261504
```

This means that hypothesis $\mu = 0$ would be rejected: $\hat{\mu}$ is much further than 2 standard errors away from 0.

Of course, this might have been one of those 5% of cases, where the statistician is wrong, but if we make 100 replications,

```r
> nwrong <- 0
> for(i in 1:100){
+ sim <- grf(5,sim4.coords,cov.model='gau',cov.pars=c(1,3),
+ messages=F)
+ if(abs(mean(sim$data))>2*sqrt(var(sim$data)/5))
+ nwrong <- nwrong+1
+ }
> nwrong
[1] 60
```

we’re wrong 60% of times instead of 5%!

The reason is, that the i.i.d. assumption leads to too small standard error estimates.
Now let us assume that we knew the true covariance function. The covariance matrix $\Sigma$ and its inverse $\Sigma^{-1}$ are (approximately)

```
> Sigma <- gaucov(as.matrix(dist(sim4.coords)),d=3)
> round(Sigma,1)
                1   2   3   4   5
1 1.0 0.9 0.9 0.8 0
2 0.9 1.0 0.8 0.9 0
3 0.9 0.8 1.0 0.9 0
4 0.8 0.9 0.9 1.0 0
5 0.0 0.0 0.0 0.0 1
```

In this case ($X = (1, 1, \ldots, 1)^T$), the GLS estimator of $\beta = \mu$ is a weighted average of the observed responses $y(s_i)$ with weights proportional to the column sums of $\Sigma^{-1}$ (Math ex. VI)

```
> w <- apply(solve(Sigma),2,sum)
> round(w,1)
                   1   2   3   4   5
0.3 0.3 0.3 0.3 1.0
```

So those observations that are ‘more correlated’ with others get smaller weights.
The GLS estimate then is

\[ \text{weighted.mean(sim4$data, w)} \]

\[
> \text{weighted.mean(sim4$data, w)} \\
[1] -0.4295521
\]

a bit closer to zero than the OLS estimate, and its estimated standard error, taking into account spatial correlation

\[ \sqrt{1/\text{sum(w)}} \]

\[
> \text{sqrt(1/sum(w))} \\
[1] 0.6877643
\]

So, 0 will be well within confidence intervals.

\[ \text{surf.gls of spatial library computes the GLS-estimate} \]

\[ > \text{surf.gls(0, gau cov, ssim4, d=3)$beta} \]

\[
[1] -0.429553
\]

but I haven’t figured out how to extract the standard errors from its output (apparently, \text{surf.gls} is designed mainly to serve spatial prediction, see p. 296).
Repeating GLS-estimation for 100 simulations from the same model:

```r
> glsse <-
+ sqrt(1/sum(solve(gaucov(as.matrix(dist(sim4.coords)),d=3))))
> nwrong <- 0
> for(i in 1:100){
+ sim <- grf(5,sim4.coords,cov.model='gau',cov.pars=c(1,3),
+ messages=F)
+ ssim <- data.frame(x=sim$coords[,1],y=sim$coords[,2],z=sim$data)
+ if(abs(surf.gls(0,gaucov,ssim,d=3)$beta)>2*glsse)
+ nwrong <- nwrong+1
+ }
> nwrong
[1] 5
```

we’re wrong 5% of the time, as we should be (getting `nwrong`=5 exactly is just a lucky incidence; we might as well have got 4 or 7).

So the big problem with OLS in spatial regression is that the standard errors are under-estimated in the presence of spatial correlation (of residuals). The OLS estimator of $\mu$, itself, is unbiased, but in this case GLS gives also a more efficient estimator (smaller RMSE = $\sqrt{\sum(\hat{\mu} - \mu)^2/n}$), in addition to appropriate standard errors:

```r
> ols <- gls <- numeric(100)
> for(i in 1:100){
+ sim <- grf(5,sim4.coords,cov.model='gau',cov.pars=c(1,3),messages=F)
+ ols[i] <- mean(sim$data)
+ ssim <- data.frame(x=sim$coords[,1],y=sim$coords[,2],z=sim$data)
+ gls[i] <- surf.gls(0,gaucov,ssim,d=3)$beta
+ }
> sqrt(mean(ols^2))
[1] 0.7708407
> sqrt(mean(gls^2))
[1] 0.6366853
```
Practical solution to spatial regression problem

1. Fit model on p. 187 using OLS regression
2. Estimate a (valid) variogram model, \( \hat{\gamma}(\cdot) \), from the residuals of OLS fit, giving rise to an equivalent covariogram model, \( \hat{C}(\cdot) \)
3. Estimate elements of covariance matrix \( \Sigma \) by \( \hat{C}(s_i, s_j) \)
4. Fit model on p. 275 with GLS (p. 276) using \( \hat{\Sigma} \)

This corrects parameter estimates and standard errors for second order effects:

- Possibly get a more efficient estimator, but most importantly
- standard errors are appropriately derived.

Spatial regression estimates from geoR

The corresponding likelihood-based estimates of \( \beta \) and \( \text{Var}(\hat{\beta}) \) are given in components \texttt{beta} and \texttt{beta.var} of the object returned by \texttt{likfit}.

For example, for the Lapland elevations, the coefficients of the linear trend surface, estimated by REML are (as we saw on p. 268)

```r
> rml$beta
intercept x y
0.4754206 -0.1063534 0.0699861
```
and their covariance matrix is

\[
\begin{pmatrix}
V1 & V2 & V3 \\
V1 & 1.334475e-02 & 2.452937e-05 & 2.445249e-05 \\
V2 & 2.452937e-05 & 3.538624e-05 & 9.630330e-07 \\
V3 & 2.445249e-05 & 9.630330e-07 & 3.698626e-05
\end{pmatrix}
\]

The standard errors are the square roots of the diagonal elements

\[
\sqrt{ \text{diag}(rml$beta.var)} 
\]

\[
\begin{pmatrix}
V1 & V2 & V3 \\
0.115519480 & 0.005948633 & 0.006081633
\end{pmatrix}
\]

The whole covariance matrix is needed when using the model for spatial prediction, for example (see later).

Ignoring spatial correlation:

\[
\begin{pmatrix}
V1 & V2 & V3 \\
0.49126016 & -0.10493160 & 0.07405326
\end{pmatrix}
\]

leads in this case to rather similar coefficients

\[
\begin{pmatrix}
V1 & V2 & V3 \\
0.033161734 & 0.002257049 & 0.002225683
\end{pmatrix}
\]

but to substantial under-estimation of standard errors
4.7 Spatial prediction

If we have model

\[ Y(s) = \mu(s) + U(s), \]

where covariance structure of \( U(s) \) known through a variogram model, can do better than predict \( Y(s) \) by \( \hat{\mu}(s) \)?

Adding the prediction of \( U(s) \) is known as kriging.

Linear prediction

Intuitively sensible to predict \( U(s) \) by a weighted linear combination of observed residuals

\[ \hat{U}(s) = \sum_{i=1}^{n} \lambda_i(s) U(s_i), \]

where \( \lambda_i(s) \) large when \( |s_i - s| \) small.

Many smoothing and interpolation methods (moving averages, nearest neighbour methods, kernel methods) based on such predictors.

Kriging allows for choosing weights \( \lambda_i(s) \) according to the degree of spatial correlation, which can be estimated from the data.

Furthermore, accuracy assessments are automatically available.
**Simple kriging**

Assumes $\mu(s)$ and $C()$ known (rather than estimated); $U(s_i) = Y(s_i) - \mu(s_i)$.

Best linear predictor is obtained (mean squared prediction error minimised) by choosing

$$\lambda(s) = \Sigma^{-1} c(s),$$

where $\lambda(s)$ is the vector of kriging weights $\lambda_i(s)$ and $c(s)$ the vector of covariances $C(s, s_i)$.

Note that weight of $U(s_i)$ depends both on Cov$(s, s_i)$ and, through $\Sigma^{-1}$, on correlation of $U(s_i)$ with other observations.

The minimised mean squared prediction error, or the kriging variance,

$$E\{(\hat{U}(s) - U(s))^2\} = C(0) - c(s)^T \Sigma^{-1} c(s)$$

gives an estimate of prediction accuracy.

See Math. ex. VI for the derivation of formulae on this and previous page.
Kriging with GLS estimates

1. Estimate $C()$ from OLS residuals and $\mu(s)$ by GLS as explained on p. 287.

2. Plug estimates into the simple kriging formulae.

3. Adjust mean square prediction error by adding an extra component arising from estimating trend from the data.

This is implemented in functions `surf.gls`, `prmat` and `semat` of the `spatial` library.

`surf.gls` can work with exponential, Gaussian and spherical covariograms and requires the values of $\phi$ and $\alpha$ (p. 246) as input (see p. 284).

`prmat` takes the fitted model, an object produced by `surf.gls`, as input and computes kriging predictions in a regular grid.

`semat` computes the associated standard errors (square roots of kriging variances).

Predictions do not depend on $\sigma^2$, but variances do. Square root of $\sigma^2$ can be given to `semat` as argument `se`; otherwise `semat` estimates it.
Kriging topo data

Try different models fitted in Comp. ex. VI.

```r
gls.wls0gau <- surf.gls(0, gauccov, topo,
 + d=3.6179, alpha=87.7717/(87.7717+6835.6169))
gls.rml0gau <- surf.gls(0, gauccov, topo,
 + d=1.687, alpha=96.069/(96.069+3049.956))
gls.wls2exp <- surf.gls(2, expcov, topo, d=1.0443)
gls.wls2gau <- surf.gls(2, gauccov, topo, d=1.2398)
> points(geotopo)
> contour(
  + prmat(gls.rml0gau, 0, 6.5, 0, 6.5, 50),
  + levels=seq(600, 1000, 25), add=T)
> title('No trend, Gaussian, REML')
> contour(prmat(gls.wls0gau, 0, 6.5, 0, 6.5, 50),
  + levels=seq(600, 1000, 25), add=T)
> title('No trend, Gaussian, WLS')
```

etc.
Prediction errors

> points(geotopo,cex.min=.7,cex.max=.7,col='blue')
> contour(
  + semat(gls.rml0gau,
  + 0,6.5,0,6.5,50),
  + levels=seq(0,25,by=5),
  + add=T)
> title(
  + ’No trend, Gaussian, REML’)
> points(geotopo,
  + cex.min=.7,cex.max=.7,
  + col=’blue’)
> contour(
  + semat(gls.wls0gau,
  + 0,6.5,0,6.5,50),
  + levels=seq(0,25,by=5),
  + add=T)

etc.

NFI data for course work

Pointwise estimates of mean volume of growing stock (m³/ha) and estimated quadratic trend (left); kriged mean volume surface (right).
• Full data contains mean volume estimates in most points of a grid with $x$-coordinate ranging from 0 to 196 km with 7 km intervals and $y$ from 0 to 288 km with 8 km intervals.

• For the course work, subsets of 100 random points are sampled from the full data.

• Basic idea is to compare kriging-predicted mean volume surfaces and their prediction errors between different trend surface and variogram models.