Lab 4.1: Spatio-Temporal Kriging with gstat

In this Lab we go through the process of carrying out spatio-temporal universal kriging using the semivariogram with the package **gstat**. We focus on the maximum temperature data in the NOAA data set (`Tmax`) in July 1993. In addition to the packages used earlier for data wrangling, we need **RColorBrewer** to color some of the surfaces that will be produced.

```
library("sp")
library("spacetime")
library("ggplot2")
library("dplyr")
library("gstat")
library("RColorBrewer")
library("STRbook")
library("tidyr")
```

For S-T kriging of the maximum-temperature data set in July 1993, we need to fit a parametric function to the empirical semivariogram `vv` computed in Lab 2.3. The code is reproduced below for completeness.

```
data("STObj3", package = "STRbook")
STObj4 <- STObj3[, "1993-07-01::1993-07-31"]
vv <- variogram(object = z ~ 1 + lat, # fixed effect component
                data = STObj4,        # July data
                width = 80,           # spatial bin (80 km)
                cutoff = 1000,        # consider pts < 1000 km apart
                tlags = 0.01:6.01)    # 0 days to 6 days
```

A number of covariance-function models are available with the package **gstat**; see the **gstat** vignette "spatio-temporal-kriging" for details by typing

```
vignette("spatio-temporal-kriging")
```

The first semivariogram we consider here corresponds to the spatio-temporal separable covariance function

$$c^{(sep)}(\mathbf{h};\tau) \equiv c^{(s)}(\|\mathbf{h}\|)\cdot c^{(t)}(|\tau|), \tag{1}$$

in which we let both covariance functions, $c^{(s)}(\cdot)$ and $c^{(t)}(\cdot)$, take the form

$$c(d) = b_1 \exp(-\phi d) + b_2 I(d=0), \tag{2}$$

where b_1 and b_2 are parameters that need to be estimated and $I(\cdot)$ is the indicator function. Observe from the vignette that a separable covariance function (1) corresponds to a semivariogram of the form

$$\gamma^{\text{sep}}(\mathbf{h};\tau) = \text{sill}\cdot\left(\bar{\gamma}^{(s)}(\|\mathbf{h}\|) + \bar{\gamma}^{(t)}(|\tau|) - \bar{\gamma}^{(s)}(\|\mathbf{h}\|)\bar{\gamma}^{(t)}(|\tau|)\right),$$

where the "standardized" semivariograms $\bar{\gamma}^{(s)}$ and $\bar{\gamma}^{(t)}$ have separate nugget effects and sills equal to 1.

A spatio-temporal semivariogram is constructed with **gstat** using the function `vgmST`. The argument `stModel = "separable"` is used to define a separable model, while the function `vgm` is used to construct the individual semivariograms (one for space and one for time). Several arguments can be passed to `vgm`. The first four, which we use below, correspond to the partial sill, the model type, the range, and the nugget, respectively. The argument `sill` that is supplied to `vgmST` defines the joint spatio-temporal sill. The numbers used in their definition are initial values supplied to the optimization routine used for fitting in the function `fit.StVariogram`, which fits `sepVgm` to `vv`. These initial values should be reasonable – for example, the length scale ϕ can be set to a value that spans 10% of the spatial/temporal domain, and the variances/sills can be set such that they have similar orders of magnitude to the total variance of the measurements.

```
sepVgm <- vgmST(stModel = "separable",
                space = vgm(10, "Exp", 400, nugget = 0.1),
                time = vgm(10, "Exp", 1, nugget = 0.1),
                sill = 20)
sepVgm <- fit.StVariogram(vv, sepVgm)
```

The second model we fit has a joint covariance function over space and time, in which the temporal separation is scaled to account for the different nature (anisotropy) of space and time. This model is given by

$$c^{(st)}(\|\mathbf{v}\|) \equiv b_3 \exp(-\phi\|\mathbf{v}\|) + b_4 I(\|\mathbf{v}\| = 0), \tag{3}$$

where, for any two spatio-temporal coordinates $(\mathbf{s}_i', t_i)'$ and $(\mathbf{s}_j', t_j)'$, $\mathbf{v} \equiv (\mathbf{s}_i', at_i)' - (\mathbf{s}_j', at_j)'$. Here, a is the scaling factor used for generating space-time anisotropy, and is also the argument `stAni` supplied to `vgmST`. This parameter can be initially set by considering orders of magnitudes – if the spatial field is evolving on scales of the order of hundreds of kilometers and the temporal evolution is on scales of the order of days, then an initial value of `stAni = 100` is reasonable. This covariance function is sometimes referred to as a *metric* covariance function.

```
metricVgm <- vgmST(stModel = "metric",
                   joint = vgm(100, "Exp", 400, nugget = 0.1),
                   sill = 10,
                   stAni = 100)
metricVgm <- fit.StVariogram(vv, metricVgm)
```

We can compare the fits of the two semivariograms by checking the mean squared error of the fits. These can be found by directly accessing the final function value of the optimizer used by `fit.StVariogram`.

```
metricMSE <- attr(metricVgm, "optim")$value
sepMSE <- attr(sepVgm, "optim")$value
```

Here the variable `metricMSE` is 2.1 while `sepMSE` is 1.4, indicating that the separable semivariogram gives a better fit to the empirical semivariogram in this case. The fitted semivariograms can be plotted using the standard **`plot`** function.

```
plot(vv, list(sepVgm, metricVgm), main = "Semi-variance")
```

Next, we use the fitted S-T covariance models for prediction using S-T kriging, in this case *universal* S-T kriging since we are treating the latitude coordinate as a covariate. First, we need to create a space-time prediction grid. For our spatial grid, we consider 20 spatial locations between 100°W and 80°W, and 20 spatial locations between 32°N and 46°N. In the code below, when converting to `SpatialPoints`, we ensure that the coordinate reference system (CRS) of the prediction grid is the same as that of the observations.

```
spat_pred_grid <- expand.grid(
                      lon = seq(-100, -80, length = 20),
                      lat = seq(32, 46, length = 20)) %>%
            SpatialPoints(proj4string = CRS(proj4string(STObj3)))
gridded(spat_pred_grid) <- TRUE
```

For our temporal grid, we consider six equally spaced days in July 1993.

```
temp_pred_grid <- as.Date("1993-07-01") + seq(3, 28, length = 6)
```

We can then combine `spat_pred_grid` and `temp_pred_grid` to construct an `STF` object for our space-time prediction grid.

```
DE_pred <- STF(sp = spat_pred_grid,     # spatial part
               time = temp_pred_grid)   # temporal part
```

Since there are missing observations in `STObj4`, we first need to cast `STObj4` into either an `STSDF` or an `STIDF`, and remove the data recording missing observations. For simplicity here, we consider the `STIDF` (considering `STSDF` would be around twice as fast). Also, in order to show the capability of S-T kriging to predict across time, we omitted data on 14 July 1993 from the data set.

```
STObj5 <- as(STObj4[, -14], "STIDF")         # convert to STIDF
STObj5 <- subset(STObj5, !is.na(STObj5$z))   # remove missing data
```

Now we can call **`krigeST`** using `STObj5` as our data.

```
pred_kriged <- krigeST(z ~ 1 + lat,          # latitude trend
                       data = STObj5,        # data set w/o 14 July
                       newdata = DE_pred,    # prediction grid
                       modelList = sepVgm,   # semivariogram
                       computeVar = TRUE)    # compute variances
```

To plot the predictions and accompanying prediction standard errors, it is straightforward to use the function `stplot`. First, we define our color palette using the function `brewer.pal` and the function `colorRampPalette` (see help files for details on what these functions do).

```
color_pal <- rev(colorRampPalette(brewer.pal(11, "Spectral"))(16))
```

Second, we call the `stplot` function with the object containing the results.

```
stplot(pred_kriged,
       main = "Predictions (degrees Fahrenheit)",
       layout = c(3, 2),
       col.regions = color_pal)
```

The prediction (kriging) standard errors can be plotted in a similar way.

```
pred_kriged$se <- sqrt(pred_kriged$var1.var)
stplot(pred_kriged[, , "se"],
       main = "Prediction std. errors (degrees Fahrenheit)",
       layout = c(3, 2),
       col.regions = color_pal)
```

Spatio-temporal kriging as shown in this Lab is relatively quick and easy to implement for small data sets, but it starts to become prohibitive as data sets grow in size, unless some approximation is used. For example, the function `krigeST` allows one to use the argument `nmax` to determine the maximum number of observations to use when doing prediction. The predictor is no longer optimal, but it is close enough to the optimal predictor in many cases of practical interest.