EnviroStat

Nhu D Le and James V Zidek

November 2013

1 The EnviroStat approach

EnviroStat provides functions for spatio-temporal modeling of environmental processes and designing monitoring networks for them based on an approach described in Le and Zidek (2006). That approach has a number of features.

- Conditional on knowing the process parameters the environmental process is assumed to have (after a suitable transformation if necessary) to be a Gaussian random field (GRF).
- At every spatial location, the process can yield a multiplicity of random responses such as air pollutant concentrations.
- The approach used in the package lies within a Bayesian hierarchical modeling framework. However for computational expediency empirical shortcuts are made at higher levels of the hierarchical setup. Thus for example most hyperparameters are fitted using a type II maximum likelihood approach, eliminating the need for the the user to specify them. Thus the package can handle large fields of monitoring networks, say with 600 or more spatial sites.
- The approach does not assume a stationary GRF. Instead it takes a nonparametric approach where the spatial covariance matrix is left completely unspecified and instead endowed with a prior distribution with a hypercovariance matrix that can be modeled at level two of the hierarchy, making the method quite robust against non-stationarity in the random field.
- It presents a approach for designing monitoring networks based on the well-known warping method of Sampson and Guttorp (1992) as developed with Wendy Meiring.
- It allows for missing data, providing that these data are missing in blocs of time, which after a regional trend is fitted, then become exchangeable. For then the blocs of residuals can be permuted the get a decreasing or increasing staircase pattern in the data matrix something that is required in the approach.

- It has been empirically assessed in a number of major applications and found to yield well calibrated prediction intervals. For example, a 95% interval will cover their predictands about 95% of the time.

This vignette describes the approach by means of example involving ozone concentrations over the state of New York. Ozone is a colorless, highly reactive gas that is one of a class of photooxidants. It is an air quality criteria pollutants since it has been strongly associated with adverse effects on human health and human welfare (e.g. crop yields). More specifically, the example illustrates how an hourly ozone concentration field with a staircase pattern of missing data can be interpolated through the hierarchical framework as described in Le & Zidek (2006, Chapter 10). It also shows how new stations could be optimally added to the network. This illustration is similar to that in Chapter 14 of the same book. Overall this package contains relevant R functions and help pages for their implementation.

2 Preliminary analysis

The data set (data) used in this example consists of hourly O_3 concentration levels (ppb) from nine stations in New York State numbered for our purposesfrom 1 to 9. These data were originally downloaded from the EPA's AIRS air quality site (Evironmental Protection Agency 2013). Furthermore preliminay analysis suggested a sqrt transformation of the original concentrations to symmetrize the data distribution.

Other information in the data set includes month (mm from 4 to 9), day within month (dd from 1 to 31), hour within day (hr from 0 to 23), weekday (wkday from 2–8), and sequential number of the week (wk from 1 to 27). Each row of the data set represents an hourly record starting at April 1, 1995, hour 0, and ending at September 30, 1995 hour 23; i.e., there are 4392 records (24 hours $\times$ 183 days). The last six stations have no missing observations while stations 1, 2, 3 have 2616, 2016, 72 missing hourly observations, respectively.

The first stage in any spatial data analysis is exploratory. That begins by loading the package and data files which, for convenience, me moved into our R Work subdirectory:

```
> library(EnviroStat)
> data(ozone.NY)
```

We thus extract the month, weekday, and sqrtO3 to get 36 columns (9 stations $\times$ 4 hours for the period 8am–12noon):

```
> month <- ozone.NY[,1]
> weekday <- ozone.NY[,2]
> sqO3 <- ozone.NY[,3:38]
```

All spatial analyses rely on the locations of the monitoring sites, so we now extract information about those locations in terms of their geographical coordinates, latitude and longitude, or (lat, long) for short.

```
> data(location.NY)
```

Not all of these sites started measuring ozone at the same time, so the data matrix has a lot of missing values. By permuting the order of the sites in the data set, we can make the missing data pattern monotone, in other words, look like a descending or ascending staircase, depending on how the permutation is done. This pattern simplifies the underlying technical analysis considerably.

The first step in our exploration, calculates the number of missing observations for each site.

```
> sitenumber <- apply(is.na(sqO3), 2, sum)
```

The counts tell us that the order of the nine monitoring sites (that are also called "stations") should be permuted to (2, 6, 5, 1, 3, 4, 7, 8, 9) to get the required monotone pattern. Our approach requires in the multiariate case where more than one response may be measured at each site, that every one of the responses has to be measured over the same number of time periods. This may entail deleting a few observations - in the end each tred in the staircase must be flat.

```
> norder <- c(2, 6, 5, 1, 3, 4, 7, 8, 9)
> tt <- NULL
> for (i in 1:9) tt <- c(tt, c(1:4) + 4 * (norder[i]-1))
> ndata <- sqO3[,tt]
> nloc <- location.NY[norder, ]
```

We now look at the locations of the stations:

```
> plot(nloc[,2], nloc[,1], type = "n",
+      xlab = "Long", ylab = "Lat")
> text(nloc[,2], nloc[,1], c(1:9))
```

Preliminary analysis (not included here) suggests 'month' and 'weekday' are important covariates and the sqrt transformed O3 concentration levels are approximately normal. The results show consistent patterns of hourly and weekday effects for all stations. Except for the first few weeks in April the weekly effects show little temporal trend. Autoregression of order 2 was also seen in the preliminary analysis. One could attempt to model that autocorrelation. But to demonstrate the full power of our multivariate approach we use a different approach that avoids altogether the need to model that dependence at the fine temporal scale of one hour. This would be a challenging problem since the strength of that autocorrelation varies over a day, being fairly weak at night for example but strong in the middle of the day. Instead we allow it to be arbitrary by exploiting the power of our multivariate approach by stacking the hourly responses in a single vector of responses for each day as described in the next section.

We conclude this section by setting up the model matrix for the analysis done in the next.

```
> month[1:5]

[1] 4 4 4 4 4

> weekday[1:5]

[1] 7 8 2 3 4
```

So we set the design matrix for the covariates using the 'model.matrix' function:

```
> ZZ <- model.matrix(~as.factor(month) + as.factor(weekday))
```

3 Model fitting

Suppose one were interested in interpolating the hourly O_3 levels at unobserved locations for a specific hour of the day, say the one we designate for definiteness as 11AM, meaning the period from 11AM to 12noon. The method described in Le and Zidek (2006, Chapter 10) allows us to use daily multivariate response vectors consisting of the hourly O_3 levels at 11AM as well as several of the preceeding hours. This approach to the spatial interpolation of unmonitored sites would then borrow strength not only from the measured hourly concentrations at monitoring sites at 11AM, but also from the measurements made at those sites in the preceeding hours.

For this vignette, we use just four consecutive hours in the multivariate response vector for each day, the last being the one ending with the 11AM hour. Our choice of four hours is based on expediency given the available data (183 days) as well as to insure that the response vectors are stochastically independent at least to a reasonable approximation. More specifically, given the AR(2) structure observed in the preliminary analysis, the 18-hour gap from one daily response vector to that in the next day would reduce the temporal correlation substantially. However it should be exphasized that our approach does rely on a covariance separability assumption that the covariance for those four hours is the same from one day to the next, an assumption that may not hold in some applications.

The daily four-hour response is now assumed to follow the model assumptions in Le and Zidek (2006). The predictive distribution for the unobserved locations and times, conditional on the observed data and the hyperparameters, can then be found by applying the theory in the book. In particular the hyperparameters associated with the monitoring sites can be estimated using the EM algorithm and those associated with the unmonitored sites can be estimated via the Sampson–Guttorp method for which a function is provided in this package - see below.

In this section, we see how the multivariate response model is fitted and the hyperparameters are estimated using functions provided in the package. In particular, the EM staircase fit uses the 'staircase' function. Note that missing

data are allowed only at the beginning of the series and as noted above, within a station all responses must have the same number of missing observations.

Here the staircase steps have a (default) block structure

```
> em.fit <- staircase.EM(ndata, p = 4, covariate = ZZ,
+                        maxit = 200, tol = .000001)
```

There are just four blocks, the first having only one station and the last, six stations.

```
> em.fit$block
```

```
1 2 3 4
1 1 1 6
```

The between–hours covariance matrix can be examined.

```
> em.fit$Omega
```

```
          [,1]      [,2]      [,3]      [,4]
[1,] 0.7614744 0.5240384 0.3524858 0.2638697
[2,] 0.5240384 0.5615223 0.4379186 0.3361430
[3,] 0.3524858 0.4379186 0.4752547 0.4019699
[4,] 0.2638697 0.3361430 0.4019699 0.4755399
```

So can the residual covariance matrix although for brevity it is not displayed.

```
> em.fit$Lambda
```

The current version of the algorithm allows only an exchangeable structure for each element in the regression model for the coefficients. In other works, it assumes the same expected (baseline) vector β_0 for each hour across all stations but it does allow different β_0's for different hours. We can see the fitted baseline values by using the following command but the display is omitted for brevity. Note that the four columns correspond to the four hours while the 12 rows represent the various model covariates (e.g. 'month').

```
> em.fit$Beta0[,1:4]
```

Next we see the marginal correlation matrices ('corr.est': spatial and 'omega': between hours) again not presented for brevity.

```
> cov.est <- em.fit$Psi[[1]]
> dim1 <- nrow(cov.est)
> dim2 <- nrow(em.fit$Omega)
> corr.est <- cov.est /
+     sqrt( matrix(diag(cov.est), dim1, dim1) *
+           t(matrix(diag(cov.est), dim1, dim1)) )
```

The all important spatial correlation matrix can be seen with the following command.

```
> round(corr.est, 2)
```

Surprisingly, considering that ozone is a secondary air pollutant meaning that it is generated in the atmosphere rather than coming from point sources, these spatial correlations are fairly small, the largest being around 0.31. But it has to be remembered that our multivariate approach distributes the spatial correlation over not only the 11AM hour, but also over its three previous hours. Thus the overall strength of association of between sites is stronger than it appears in just the spatial correlation matrix alone.

In fact, the correlation between hours is much stronger as seen below.

```
> omega <- em.fit$Omega /
+     sqrt( matrix(diag(em.fit$Omega), dim2, dim2) *
+           t(matrix(diag(em.fit$Omega), dim2, dim2)) )
> round(omega, 2)
```

```
     [,1] [,2] [,3] [,4]
[1,] 1.00 0.80 0.59 0.44
[2,] 0.80 1.00 0.85 0.65
[3,] 0.59 0.85 1.00 0.85
[4,] 0.44 0.65 0.85 1.00
```

In spatial interpolation, the relationship between spatial correlation distance and the distance between sites has always played a key role. However since the earth's surface is curved and the lines of longitude are, unlike those of latitude, are parallel, the geographical coordinate system cannot easily be used to find the distance between sites except for small domains such as cities. Various projections of (lat, long) onto a flat plane have been developed to enable easy calculaton and the Lambert project incorporated into the package is quite popular.

The next plot will show the spatial correlation against inter-distances based on the projection of geographic coordinates onto a rectangular plane using the Lambert projection. The projection function comes from the package developed for the Sampson–Guttorp method seen below. So first we need the projected coordinates:

```
> coords <- Flamb2(abs(nloc))
```

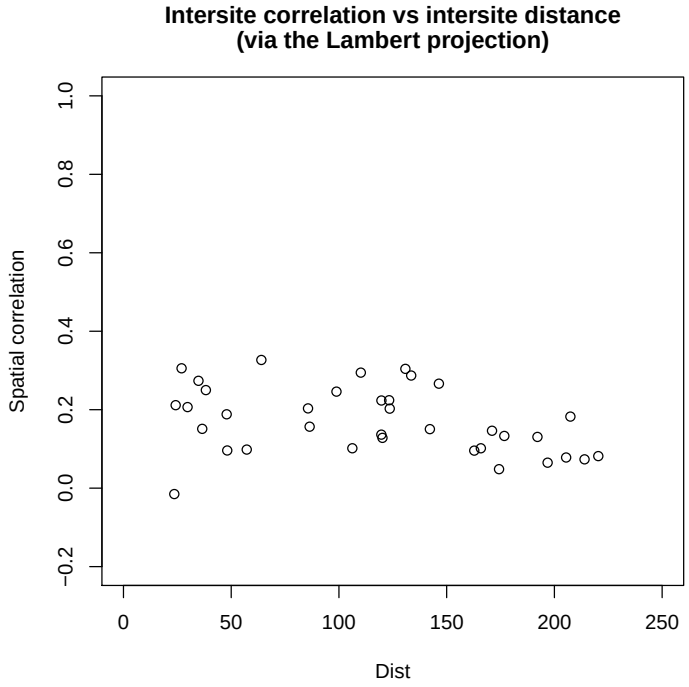
We can then calculate the distance between the locations using the 'Fdist' function that is provided in the package.

```
> dist <- Fdist(coords$xy)
```

Finally we can then plot the spatial correlation vs intersite–distances

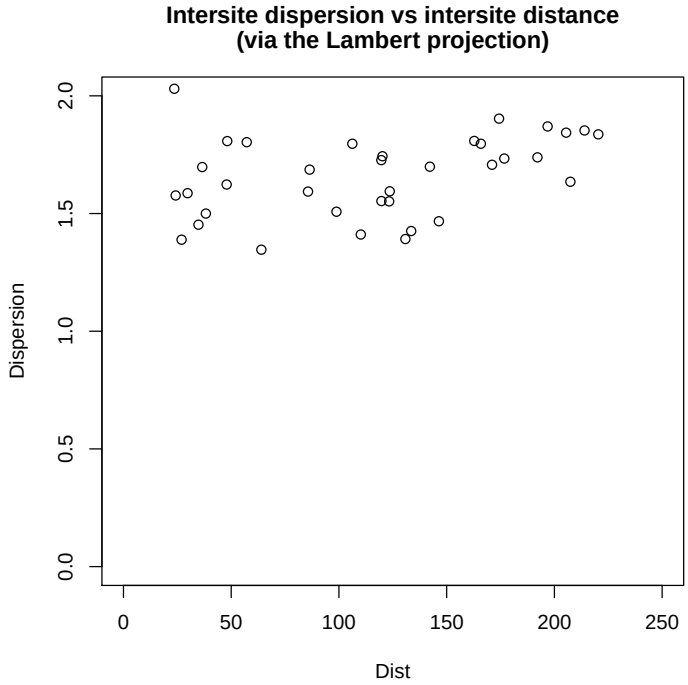
```
> par(mfrow = c(1, 1))
> plot(-.2, 0, xlim = c(0, 250), ylim = c(-.2, 1),
+      xlab = "Dist", ylab = "Spatial correlation", type = "n",
+      main = c("Intersite correlation vs intersite distance",
```

```
+             "(via the Lambert projection)"))
> for (i in 1:8)
+     for (j in (i+1):9)
+         points(dist[i, j], corr.est[i, j])
```



Then now plot the dispersion curve vs inter-distances. Note that the term dispersion is the term applied by Sampson and Guttorp (1992) to distinquish the curve obtained from the spatial correlation, which does not assume stationarity from the variogram. This curve should start close to zero when the intersite distance is near zero and rise to around 2, its maximum possible value if the process variances are equal over all sites.

```
> disp <- 2 - 2 * corr.est
> plot(-.2, 0, xlim = c(0, 250), ylim = c(0, 2),
+      xlab = "Dist", ylab = "Dispersion", type = "n",
+      main = c("Intersite dispersion vs intersite distance",
+             "(via the Lambert projection)"))
> for (i in 1:8)
+     for (j in (i+1):9)
+         points(dist[i, j], disp[i, j])
```



We do not have have monitoring sites sufficiently close together as to get a good picture of what is going on at the left hand side of the plot close to zero on the horizontal distance scale. However we demonstrate how to fit an exponential variogram using the 'Fvariogfit3' function in the Sampson–Guttorp–Meiring (SGM) package included in EnviroStat. Here model = 1 means the Exponential (default) variogram and model = 2 for the Gaussian one. To be more specific:

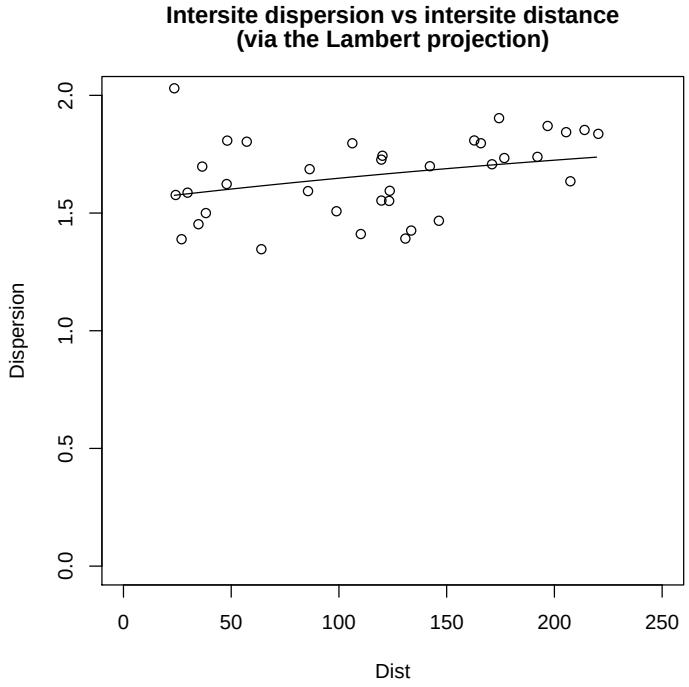
Exponential variogram: `a[1] + (2 - a[1]) * (1 - exp(-t0 * h));`

Gaussian variogram: `a[1] + (2 - a[1]) * (1 - exp(-t0 * h^2)).`

The fitting sequence for the exponential variogram is as follows:

```
> h.lt <- dist[row(dist) < col(dist)]
> disp.lt <- disp[row(disp) < col(disp)]
> variogfit <- Fvariogfit3(disp.lt, h.lt, a0 = 1.5, t0 = .1)
> x <- seq(min(h.lt), max(h.lt), 1)
> a0 <- variogfit$a[1]
> t0 <- variogfit$t0
> disp <- 2 - 2 * corr.est
> plot(-.2, 0, xlim = c(0, 250), ylim = c(0, 2),
+      xlab = "Dist", ylab = "Dispersion", type = "n",
+      main = c("Intersite dispersion vs intersite distance",
```

```
+             "(via the Lambert projection)"))
> for (i in 1:8)
+     for (j in (i+1):9)
+         points(dist[i, j], disp[i, j])
> lines(x, a0 + (2 - a0) * (1 - exp(-(t0 * x))))
```



While nonstationarity seen in this plot is mild, we can improve the fit by applying the Sampson–Guttorp method as we see below. We save the fit for a later comparison!

4 The Sampson–Guttorp approach: Thin-plate spline mapping

Sampson and Guttorp (1992) suggest a method called "warping" by which non–stationary random fields can be rendered into stationary ones. The method warps "geographical space" or G–space into what is called "dispersion space" or D–space through smooth transformations of the latitude and longitude coordinates into dispersion space coordinates in such a way that pairs of sites that are highly correlated are moved closer together in the warped space, while those less well correlated are moved further apart. In particular they extend the estimated unconditional intersite spatial covariance matrix Ψ using a nonparametric technique to generate the spatial correlations between unmonitored of interest

and other sites.The procedure also estimates critical parameters in the method However, the SG–estimation procedure is not not fully automated and has to be done in several sequential steps as described below. The SG method does not assume a constant variance, since that is not required for the SG method. Hence the approach starts with the estimation of the correlation and then any estimate of the variance field can be incorporated. Key to success of the method is the smooth bijective link functions that takes us between the two spaces.

Those functions are fitted splines - they map images of the points s = (lat, long) in G–space into points as d = f(s) in D–space so that, indexed by the points in D–space, the process becomes isotropic – the intersite correlation between the process at any two point vectors, d1 and d2, in D–space is a monotone function of the Euclidean distance between them.

The points in D–space and the spline transformations are found iteratively by warping geographic in a stepwise fashion, each step improving the fit of the variogram. The work is done by the function Falternate3 in the package developed by Sampson, Guttorp and Meiring. We now demonstate their algorithm in our example.

Step 1: Identify a thin-plate spline mapping transformation with no smoothing ($\lambda = 0$) in the following steps

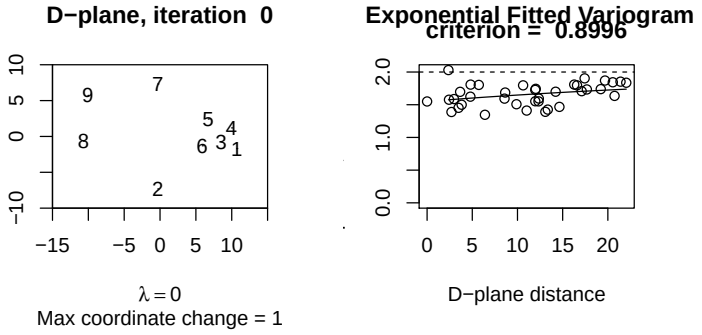
1. start with the spatial correlation
2. use the Falternate3 function – an SGM function

Smaller distances seem to work better in the multidimensional scaling method used to fit the splines.

```
> coords.lamb  <- coords$xy / 10
```

The exponential variogram is used by default, but if if Gaussian one is preferred, set model = 2.

```
> sg.est <- Falternate3(disp, coords.lamb, max.iter = 100,
+                       alter.lim = 100, model = 1)
```



The output `sg.est` has the following elements: the fitted variogram is in element `variogfit` (with subelements `variogfit$objf`, `variogfit$t0`, `variogfit$a`, `variogfit$fit`), and coordinates of new locations is in element `ncoords`.

4.1 The smoothing parameter

Step 2: Select a smoothing parameter for the identified spline mapping through the 'Ftransdraw' function.

Do this by first creating a coordinate grid for examining the mapping transformation.

```
> apply(coords.lamb, 2, range)
> coords.grid <- Fmgrid(range(coords.lamb[,1]),
+                       range(coords.lamb[,2]))
> par(mfrow = c(1, 2))
> temp <- setplot(coords.lamb, axis = TRUE)
> deform  <- Ftransdraw(disp = disp, Gcrds = coords.lamb,
+                       MDScrds = sg.est$ncoords,
+                       gridstr = coords.grid)
```

Note: This program is interative. The user will be asked to point and click on the emerging graphical image to register the curser before proceeding. This function will then post the number 1. and ask the user to enter a smoothing

parameter λ value after which the program will re–run. The user will be asked examine the plot and enter a new of λ in the R console window if necessary.

Lambda = 50 may well prove to be acceptable. The fitted variogram is quite similar to the one without any transformation (above) The mapping is almost linear! In general the goal is to trade–off the quest for stationarity in the model against the need to maintain a surface that is not so warped that interpretability is lost.

4.2 Optimizing the thin-plate spline

Step 3: Get an optimal thin-plate spline using the the 'sinterp' function & results stored in Tspline.

```
> Tspline <- sinterp(coords.lamb, sg.est$ncoords, lam = 50 )
```

The function below allows the user to plot the biorthogonal grid characterizing the warping of G-space. A grid of curves will be seen in the graphic, the intersection of any two curves being at right angles. Heavy dark curves indicate warping contractions that tend to draw points in G–space together, while the lighter, broken curves mean that surface is being stretched or expanded. This device is a very useful diagnostic tool once the user has learned to interpret it (see Sampson and Guttorp (1992).

```
> par(mfrow = c(1, 1))
> Tgrid  <- bgrid(start = c(0, 0), xmat = coords.lamb,
+                 coef = Tspline$sol)
> tempplot <- setplot(coords.lamb, axis = TRUE)
> text(coords.lamb, labels = 1:nrow(coords.lamb))
> draw(Tgrid, fs = TRUE)
```

4.3 Estimating station dispersion

Step 4: Estimate the dispersion between new locations and the stations using the SGM fit from Steps 1–3 above In this step we use the thin–plate spline from the previous step and the corresponding fitted variogram to estimate the dispersions between the stations and the new locations of interest, the latter being created using a grid of 100 points between the stations

```
> lat10 <- seq(min(nloc[,1]), max(nloc[,1]), length = 10)
> long10 <- seq(max(abs(nloc[,2])), min(abs(nloc[,2])), length = 10)
> llgrid <- cbind(rep(lat10, 10), c(outer(rep(1, 10), long10)))
```

The locations are ordered as (lat1, long1), (lat2, long1), ..., (latn, long1), (lat1, long1), ... We project the new locations using the same Lambert projection as was used for the stations above, ie. using the same reference point. Note the same scale factor of 10 is used as was used before. We then map these points into D–space using the thin–plate splines computed in the previous steps. Finally the intersite correlations are computed on D–space using the fitted variogram parameters and distances in D–space. We now show the steps involved.

```
> z <- coords
> newcrds.lamb <- Flamb2(llgrid, latrf1 = z$latrf1, latrf2 = z$latrf2,
+                        latref = z$latref, lngref = z$lngref)$xy / 10
```

Combining the new locations and stations, begining with the new locations:

```
> allcrds <- rbind(newcrds.lamb, coords.lamb)
```

Using the 'corrfit' function to estimate the correlations between the stations:

```
> corr.est <- corrfit(allcrds, Tspline = Tspline,
+                     sg.fit  = sg.est, model = 1)
> round(corr.est$cor[1:5, 1:5], 2)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,] 1.00 0.44 0.40 0.36 0.32
[2,] 0.44 1.00 0.44 0.40 0.36
[3,] 0.40 0.44 1.00 0.44 0.40
[4,] 0.36 0.40 0.44 1.00 0.44
[5,] 0.32 0.36 0.40 0.44 1.00
```

4.4 Interpolating the variance field

Step 5: Interpolate the variance field.

This step estimates the variances of the random field at all locations and then combines with them with the etimated correlation matrix in the previous step to estimate the covariance matrix.

Looking at the diagonal elements of that matrix first

```
> diag(cov.est)
```

```
[1] 0.5074834 1.3915584 1.6447338 0.6772847 0.8095887 2.3221759 1.0494045
[8] 0.6954894 1.2333390
```

suggests non-homogeneity in the field that could be smoothed using the same thin-plate splines.

```
> Tspline.var <- sinterp(allcrds[101:109,],
+                        matrix(diag(cov.est), ncol = 1),
+                        lam = 50)
```

The 'seval' function is used to obtain variance estimates at the locations. Using the thin-plate spline and then arranged in to a matrix we get

```
> varfit <- seval(allcrds, Tspline.var)$y
> temp <- matrix(varfit, length(varfit), length(varfit))
```

Combine the results of these computations to get the covariance matrix for all stations

```
> covfit <- corr.est$cor * sqrt(temp * t(temp))
```

That completes our use of the SG-method to extend the covariance matrix to the ungauged sites stored in `covfit`.

5 Estimating hyperparameters at the new locations

We can now extend the results to estimate hyperparameters associated with the new locations through the 'staircase.hyper.est' function

```
> u <- 100 # number of new locations
> p <- 4  # dimension of the multivariate response
> hyper.est <- staircase.hyper.est(emfit = em.fit,
+                                  covfit = covfit, u = u, p = p)
```

This completes the estimation of all hyperparameters. The predictive distribution can now be used for spatial interpolation. For example, let us get the predictive mean and covariance for Day 183.

```
> x <- hyper.est
> tpt <- 183
> Z <- x$covariate[tpt,]
> y <- x$data[tpt,]
```

The beta0 for the 100 ungauged locations is

```
> b0 <- matrix(rep(c(x$Beta0[,1:4]), 100), nrow = 12)
```

So the predictive mean for hours 8–11 on Day 183 are:

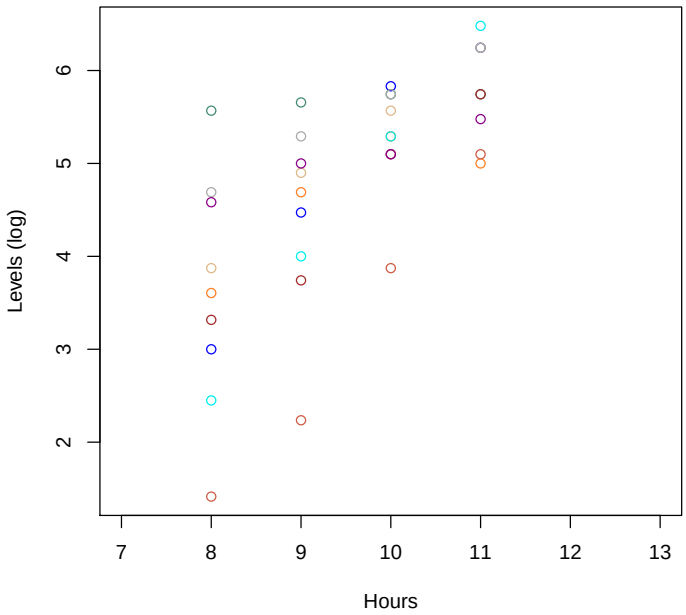
```
> mu.u <- Z %*% b0 + (as.matrix(y)- Z %*% x$Beta0) %*%
+                    kronecker(x$Xi0.0, diag(4))
```

Next plot the observed levels - Day 183. Then select a set of colors - color = colors() if the default condition is preferred.

```
> color <- colors()[c(12, 26, 32, 37, 53, 60, 70, 80, 84, 88, 94, 101,
+                     116, 142, 366, 371, 376, 386, 392, 398, 400:657)]
```

Now we plot the observed values for Day 183 by hours:

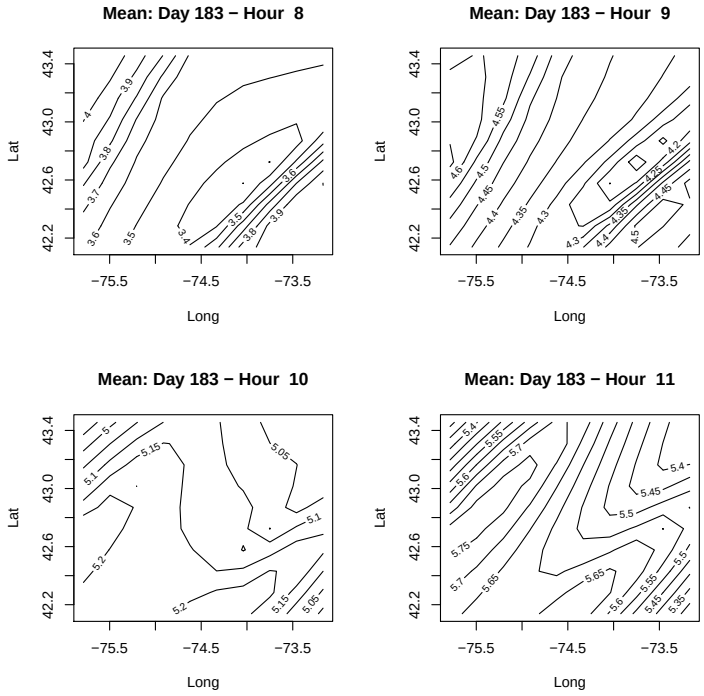
```
> par(mfrow = c(1, 1))
> plot(c(7, 13), range(y), type = "n",
+      xlab = "Hours", ylab = "Levels (log)")
> for (i in 1:4)
+     points(rep(i+7, 9), y[i + 4 * 0:8], col = color)
```



Next we can plot the contours for those hours.

```
> par(mfrow = c(2, 2))
> for (i in 1:4) {
+     tt <- i + 4 * 0:99
+     mu <- mu.u[tt]
+     hr <- matrix(mu, byrow = TRUE, ncol = 10)
+     print(range(hr))
+     contour(-long10, lat10, hr, xlab = "Long", ylab = "Lat",
+             main = paste("Mean: Day 183 - Hour ", 7+i))
+ }

[1] 3.297850 4.094019
[1] 4.113260 4.666897
[1] 4.910713 5.245441
[1] 5.266389 5.773970
```



Note that the predictive covariance may have to be obtained recursively (ie. when steps are involved). Generally the derivation is not simple in this case. A simpler approach is to simulate realizations from the predictive distribution and estimate the mean and covariance from the simulated data as demonstrated next.

5.1 Interpolation by Simulation

Since the package provides a predictive distribution, we can use it to spatially interpolate the field. Although the distribution is nonstandard, its mean and covariance can be derived in analytical form. For quantiles explicit forms can be hard to work out. So instead we can use the predictive distributions to generate realizations of the field and the spatial interpolation can be done empirically. In the next step we use the 'pred.dist.simul' function to get N = 1000 realizations:

```
> simu <- pred.dist.simul(hyper.est, tpt = 183, N = 1000)
```

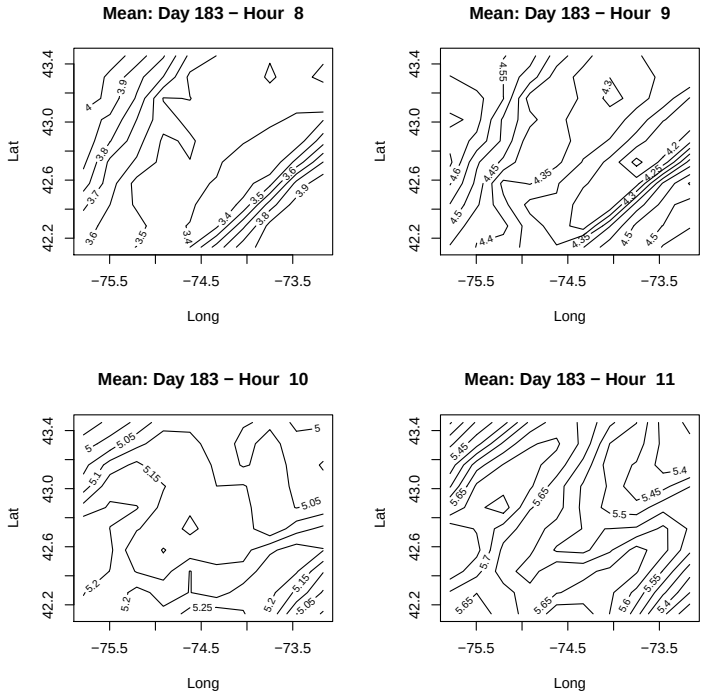
We can then extract the simulated data at the gauged stations and plot the contours of the mean in a new graphics window.

```
> x <- apply(simu, 2, mean)[1:400]
```

Next we plot the contours for hours:

```
> par(mfrow = c(2, 2))
> for (i in 1:4) {
+     tt <- i + 4 * 0:99
+     x1 <- x[tt]
+     hr <- matrix(x1, byrow = TRUE, ncol = 10)
+     print(range(x1 ))
+     contour(-long10, lat10, hr, xlab = "Long", ylab = "Lat",
+             main = paste("Mean: Day 183 - Hour ", 7+i))
+ }
```

```
[1] 3.311693 4.095067
[1] 4.132001 4.660923
[1] 4.907434 5.282614
[1] 5.257624 5.782128
```



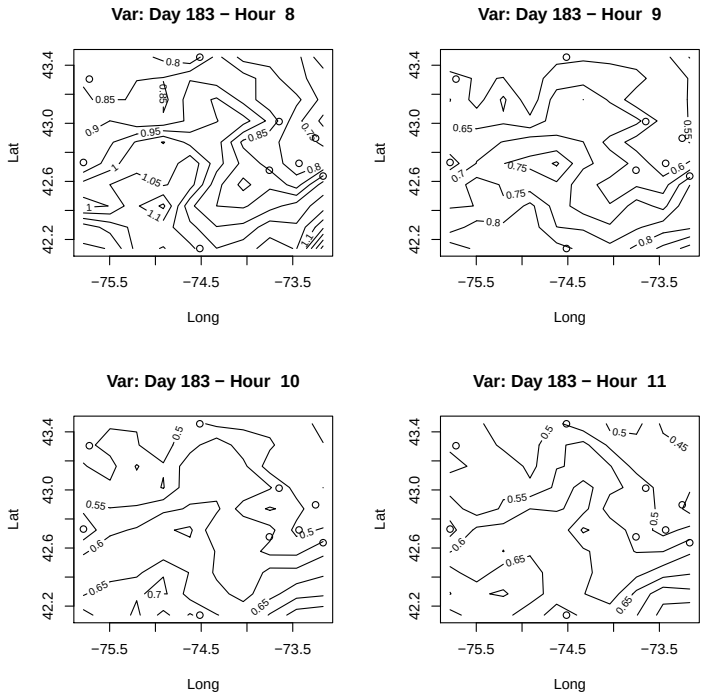
Finally we find the contours for the corresponding variance field. First we find

```
> x <- simu[,1:400]
```

and then the variance contours for each of the hour:

```
> par(mfrow = c(2, 2))
> for (i in 1:4) {
```

```
+     tt <- i + 4 * 0:99
+     x1 <- x[,tt]
+     x2 <- diag(var(x1))
+     vv <- matrix(x2, byrow = TRUE, ncol = 10)
+     contour(-long10, lat10, vv, xlab = "Long", ylab = "Lat",
+             main = paste("Var: Day 183 - Hour ", 7+i))
+ points(nloc[,2], nloc[,1])
+ }
```



6 Design Solution

Designing networks is an important element of spatio–temporal analysis. The package provides a way of doing this. A design solution can be obtained with the 'ldet.eval' function to select an additional say an 3 stations among the 100 new locations from the above grid. Their conditional covariance is in `hyper.est$Lambda.0`. The Entropy criterion selects the combination with the largest $log|det|$.

The 'ldet.eval' above evaluates the $log|det|$ for sub-covariance matrices of size 'nsel' and returns the combination with largest value using option 'all = FALSE'. Option 'all = TRUE' returns all combinations with corresponding values. This option needs a very large memory allocation if the number of combinations is big and so it should be used only for small number of potential sites, say < 15.

Note that when the number of combinations is large, this function could be very slow even with option 'all = FALSE'. For example, it could take about 30 minutes to select 5 out of 100. The following line of code produces in yy the determinant evaluations.

```
> nsel <- 3
> yy <- ldet.eval((hyper.est$Lambda.0 + t(hyper.est$Lambda.0)) / 2,
+                 nsel, all = FALSE)
```

Use the option 'all = TRUE' for a smaller matrix to get the ranked determinant evaluations in yy1 that for brevity we do not display:

```
> yy1 <- ldet.eval(((hyper.est$Lambda.0 +
+                    t(hyper.est$Lambda.0))/2)[1:10, 1:10],
+                  nsel, all = TRUE)
```

The output in yy1 enables us to see the best three additional sites.

References

[1] US Environmental Protection Agency (2013). Air Quality System Data Mart [internet database] available at http://www.epa.gov/ttn/airs/aqsdatamart.

[2] Le, N.D. and Zidek, J.V. (2006) *Statistical analysis of environmental space-time processes*, Springer, NY.

[3] Sampson, P.D. and P., Guttorp. (1992). Nonparametric Estimation of Nonstationary Spatial Covariance Structure. Journal of the American Statistical Association, 87, 108–119.