Bivariate conditional spatial models: Simulation example in Section 3.2

Noel Cressie and Andrew Zammit-Mangion

Sunday, March 01, 2015

Setting up

In this vignette we show the *R Software* code used in Section 3.2 of Cressie & Zammit-Mangion (2015). Code for the application study in Section 5 is available in a separate vignette.

In order to run this code, a few packages are needed. The first, `Matrix`, is needed for algebraic operations while `dplyr` and `tidyr` are needed for data manipulation.

```
library(Matrix)
library(dplyr)
library(tidyr)
```

The second set of packages are needed for plotting, and for arranging the figures into panels for publication.

```
library(ggplot2)
library(gridExtra)
library(grid)
library(extrafont)
loadfonts()
```

Finally, the package `bicon` provides the machinery for bivariate modelling using the conditional approach with (i) bisquare interaction functions and (ii) Matérn covariance functions for $C_{11}(\cdot)$ and $C_{2|1}(\cdot)$.

```
library(bicon)
```

We start off by setting up some parameters in the program – these are described in-line.

```
###---------------
### Setup
###---------------
img_path <- "../paper/art"   ## Where to save the figures
show_figs <- 1               ## Show the figures in document
print_figs <-  1             ## Print figures to file
```

Now we set up the simulation domain. We choose $D = [-1, 1]$, and a spacing $\eta_i = 0.01, i = 1, \ldots, 200$. We collect the grid information in a data frame `df`, to which extra columns will be added further on in the program. We also define `n1` as the number of grid cells for Y_1 and `n2` as the number of grid cells for Y_2. In this study, `n1` $=$ `n2` $= 200$ and we define `n` $=$ `n1` $+$ `n2` $= 400$.

```
###---------------
### Construct grid
###---------------
ds <- 0.01
```

```
df <- data.frame(s=seq(-1+ds/2,1-ds/2,by=ds),
                 areas = ds)
n1 <- n2 <- nrow(df)
n <- n1 + n2
```

Both covariance functions, $C_{11}(s,u)$ and $C_{2|1}(s,u)$, are Matérn covariance functions. That is,

$$C_{11}(s,u) \equiv \frac{\sigma_{11}^2}{2^{\nu_{11}-1}\Gamma(\nu_{11})}(\kappa_{11}|u-s|)^{\nu_{11}}K_{\nu_{11}}(\kappa_{11}|u-s|),$$
$$C_{2|1}(s,u) \equiv \frac{\sigma_{2|1}^2}{2^{\nu_{2|1}-1}\Gamma(\nu_{2|1})}(\kappa_{2|1}|u-s|)^{\nu_{2|1}}K_{\nu_{2|1}}(\kappa_{2|1}|u-s|),$$

where $\sigma_{11}^2, \sigma_{2|1}^2$ denote the marginal variances, $\kappa_{11}, \kappa_{2|1}$ are scale parameters, $\nu_{11}, \nu_{2|1}$ are smoothness parameters, and K_ν is the Bessel function of the second kind of order ν. The interaction function $b(s,u)$ is a bisquare function given by

$$b(s,v) \equiv \begin{cases} A\{1-(|v-s-\Delta|/r)^2\}^2, & |v-s-\Delta| \le r \\ 0, & \text{otherwise.} \end{cases}$$

In the simulation study we fix $\nu_{11} = \nu_{2|1} = 1.5$ and set the other parameters (including the standard deviation of the observation error) as follows:

```
###---------------------------------------
### True process and observation parameters
###---------------------------------------
kappa1 = 25          ## Scale of C_{11}(.)
kappa21 = 75         ## Scale of C_{2|1}(.)

sigma2_1 <- 1        ## Variance of C_{11}(.)
sigma2_21 <- 0.2     ## Variance of C_{2|1}(.)

A <- 5               ## Amplitude of b(.)
delta = -0.3         ## Shift of b(.)
r = 0.3              ## Aperture of b(.)

sigmav <- 0.5        ## Observation error std
```

Matrix construction and simulation

After setting the required parameters, we are now in a position to construct the covariance matrix Σ, which is given by

$$\begin{bmatrix} \Sigma_{11} & \Sigma_{11}B^{\mathrm{T}} \\ B\Sigma_{11} & \Sigma_{2|1} + B\Sigma_{11}B^{\mathrm{T}} \end{bmatrix}.$$

To facilitate this we have provided a function `makeSY` which takes a vector of grid distances, the parameters of the Matérn function, and the matrix B as input arguments. First, we construct the matrix B that, recall, is simply the interaction function evaluated over the grid cells multiplied by the grid spacing (when using the rectangular rule to approximate the integration). That is,

$$B^{(j,k)} = \eta_k b(s_j, v_k).$$

```
###---------------------------
### Construct required matrices
###---------------------------
H <- t(outer(df$s,df$s,FUN = "-"))              ## Find displacement
B <- A*bisquare_1d(H,delta = delta,r = r)*ds   ## Find Bmat
```

We can now construct the required covariance matrix as follows:

```
D <- abs(H)
Dvec <- as.double(c(D))                  ## Find distances
Sigma <- makeSY(r = Dvec,
                var1 = sigma2_1,
                var2 = sigma2_21,
                kappa1 = kappa1,
                kappa2 = kappa21,
                B = B)                 ## Build covariance matrix
```

The covariance functions are given as

```
Cov11 <- Sigma[n1/2,1:n1]
Cov12 <- Sigma[n1/2,(n1+1):n]
Cov21 <- Sigma[n1+n2/2,1:n1]
Cov22 <- Sigma[n1+n2/2,(n1+1):n]

Cov_df <- expand.grid(s=df$s,proc1=c("Y1","Y2"),proc2=c("Y1","Y2"))
Cov_df$cov <- c(Cov11,Cov21,Cov12,Cov22)

g_cov <- LinePlotTheme() + geom_line(data=Cov_df,aes(s,cov)) + facet_grid(proc1 ~ proc2)
if(print_figs) ggsave(g_cov,
                      filename = file.path(img_path,"cov_functions.png"),
                      width=12,height=10,family="Arial")
if(show_figs) print(g_cov)
```

Given the covariance matrix, we can simulate from the bivariate field *jointly*. Observations are simulated from this field by simply adding Gaussian error to the generated fields. These simulations are all added to the data frame `df`:

```
###--------------------
### Generate data
###--------------------
set.seed(50)                                        ## Fix seed
samp <- t(chol(Sigma)) %*% rnorm(2*nrow(df))        ## Simulate Y1 and Y2
df <- df %>%
  mutate(samp1 = samp[1:n1],
         samp2 = samp[-(1:n1)],
         Z1 = samp1 + sigmav*rnorm(n1),
         Z2 = samp2 + sigmav*rnorm(n2))             ## Add simulations to df
Z <- matrix(c(df$Z1,df$Z2))                         ## Save concatenated observations in Z
```

To demonstrate the benefits of cokriging, we choose to keep only half of the observations of Y_1, those appearing in the right half of the domain. Inferences on Y_1 in the left half of the domain will be facilitated through observations on Y_2:

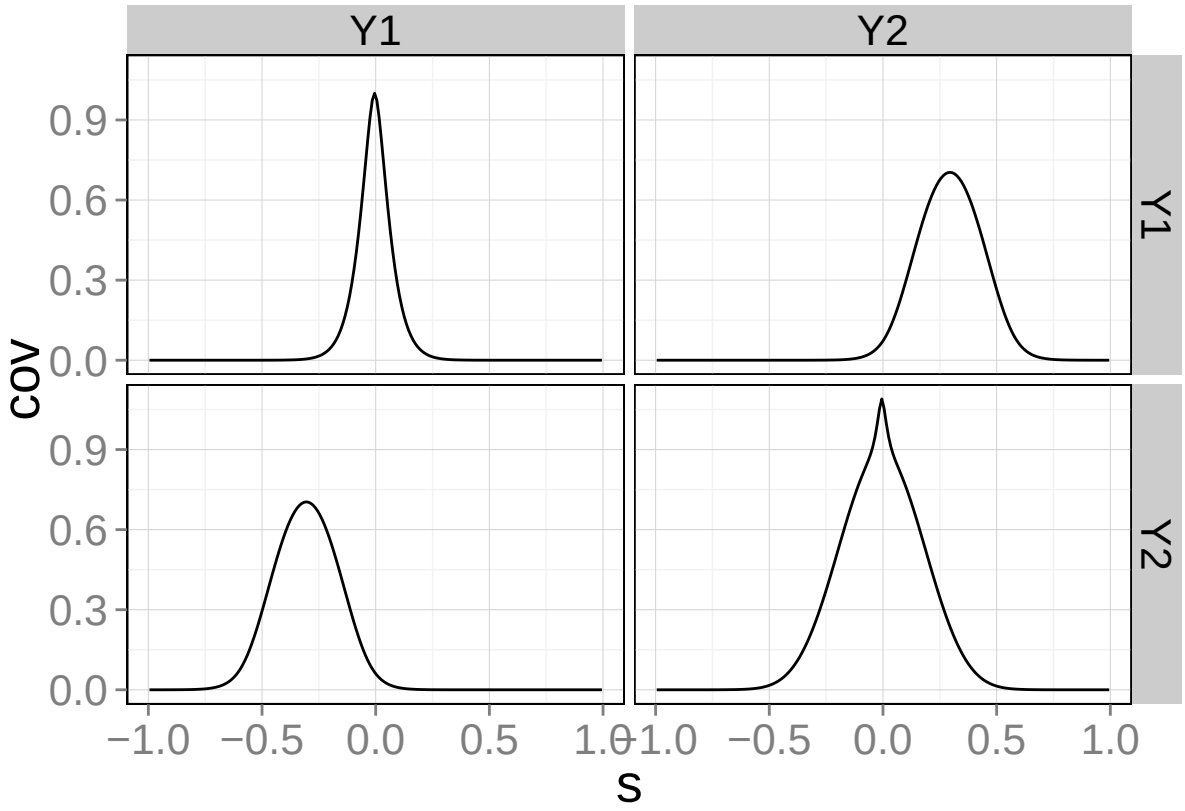

Figure 1: The correlation and cross-correlation functions estimated using Model 4, depicted as a function of displacement h in degrees latitude/longitude, at the point $s = (-123, 45)$. Contour lines denote intervals of 0.2.

```
keep_Z1 <- 101:200 ## Keep Z1 only in the right half of the domain
keep_Z2 <- 1:200   ## Keep Z2 everywhere
```

Cokriging

Since we are fixing both processes to have zero mean, cokriging of $Y_1(s_0), s_0 \in D$ proceeds through the *simple* cokriging equations. These are given through

$$\hat{Y}_1(s_0) \equiv E(Y_1(s_0) \mid Z_1, Z_2) = \begin{bmatrix} c_{11}^{\mathrm{T}} & c_{12}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} C_{11} + \sigma_{\varepsilon_1}^2 I_{m_1} & C_{12} \\ C_{21} & C_{22} + \sigma_{\varepsilon_2}^2 I_{m_2} \end{bmatrix}^{-1} \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix},$$

where for $q, r = 1, 2$,

$$c_{1r}^{\mathrm{T}} \equiv (C_{1r}(s_0, s_{ri}) : i = 1, \ldots, m_r);\ r = 1, 2, \tag{1}$$
$$C_{qr} \equiv (C_{qr}(s_{qi}, s_{rj}) : i = 1, \ldots, m_q,\ j = 1, \ldots, m_r);\ q, r = 1, 2, \tag{2}$$

and m_1, m_2 are the number of observations of Y_1, Y_2, respectively.

In the following cokriging function, we require four input variables. These are:

- `df`: The original dataframe with information on grid spacings, locations and observations.
- `A`: The amplitude of the bisquare function. If `A` $= 0$, then the two fields are independent.
- `obs_ind`: A vector with values equal to 1 for observations which are kept, and 0 for observations which are omitted.
- `name`: The name to be associated with the cokriging results

The function first constructs the required Σ (through `makeSY`), and then implements the above equations. The function stores the results in the data frame `df`.

To call the function `co_krige`, we first specify which observations to keep in the variable `obs_ind`:

```
df$keep_Z1 <- 1:nrow(df) %in% keep_Z1   ## Create vector of indices marking which
df$keep_Z2 <- 1:nrow(df) %in% keep_Z2   ## observations are kept and which are discarded
obs_ind <- c(keep_Z1,keep_Z2 + n1)
```

We can then pipe our original `df` through `co_krige` using differing values of `A`: (i) `A = 0` (independent variates) and (ii) `A = A` (true model). Note that case (i) is identical to simple kriging on Y_1 using only Z_1, since under independence the system is *autokrigeable* (see Wackernagel, 1995, p. 149).

```
loglik_Model <- function(theta,model_num,i=NULL) {
  # theta1: A
  # theta2: r
  df2 <- subset(df, s > 0)
  H2 <- t(outer(df2$s,df2$s,FUN = "-"))              ## Find displacement
  D2 <- abs(H2)
  Dvec2 <- as.double(c(D2))                 ## Find distances
  Z2 <- matrix(c(df2$Z1,df2$Z2))                     ## Save concatenated observations in Z

  if(theta[2] < 0.0005) {
    return(Inf)
```

```
  } else {
    B <- theta[1]*bisquare_1d(H2,delta =0,r = theta[2])*ds   ## Find Bmat
    Sigma <- makeSY(r = Dvec2,
                    var1 = sigma2_1,
                    var2 = sigma2_21,
                    kappa1 = kappa1,
                    kappa2 = kappa21,
                    B = B) +
      sigmav^2 * Imat(nrow(df2)*2)
    cholZ <- chol(Sigma)
    loglik <-
      -(-0.5 * logdet(cholZ) -
          0.5 * t(Z2) %*% chol2inv(cholZ) %*% Z2 -
          0.5 * nrow(Z2)*log(2*pi)) %>% as.numeric()
       return(loglik)
     }
}
non_symm_par <-  optim(par=c(1,1),
        fn = loglik_Model,
        hessian=FALSE,
        control=list(trace=6,
                     pgtol=0,
                     maxit=3000))$par
```

```
##   Nelder-Mead direct search function minimizer
## function value for initial parameters = 262.792339
##   Scaled convergence tolerance is 3.91591e-06
## Stepsize computed as 0.100000
## BUILD              3 262.792339 260.721990
## EXTENSION          5 261.806068 258.148582
## EXTENSION          7 260.721990 255.202841
## EXTENSION          9 258.148582 249.078490
## EXTENSION         11 255.202841 243.526698
## EXTENSION         13 249.078490 235.382955
## REFLECTION        15 243.526698 235.036893
## REFLECTION        17 235.382955 232.584968
## LO-REDUCTION      19 235.036893 232.584968
## LO-REDUCTION      21 233.493373 232.584968
## EXTENSION         23 232.626744 232.084235
## EXTENSION         25 232.584968 231.545326
## LO-REDUCTION      27 232.084235 231.545326
## LO-REDUCTION      29 231.610555 231.420849
## HI-REDUCTION      31 231.545326 231.420849
## REFLECTION        33 231.446089 231.381155
## REFLECTION        35 231.420849 231.352172
## LO-REDUCTION      37 231.381155 231.336869
## LO-REDUCTION      39 231.352172 231.331323
## LO-REDUCTION      41 231.336869 231.322556
## LO-REDUCTION      43 231.331323 231.322460
## LO-REDUCTION      45 231.322556 231.317145
## HI-REDUCTION      47 231.322460 231.316870
## HI-REDUCTION      49 231.317145 231.315857
## HI-REDUCTION      51 231.316870 231.315857
```

```
## REFLECTION        53 231.315931 231.315247
## HI-REDUCTION      55 231.315857 231.315121
## EXTENSION         57 231.315247 231.313948
## HI-REDUCTION      59 231.315121 231.313948
## EXTENSION         61 231.314655 231.312498
## EXTENSION         63 231.313948 231.311172
## EXTENSION         65 231.312498 231.307405
## LO-REDUCTION      67 231.311172 231.307405
## EXTENSION         69 231.307451 231.301045
## LO-REDUCTION      71 231.307405 231.301045
## EXTENSION         73 231.301146 231.293742
## LO-REDUCTION      75 231.301045 231.293742
## REFLECTION        77 231.295808 231.292624
## REFLECTION        79 231.293742 231.291563
## LO-REDUCTION      81 231.292624 231.291563
## LO-REDUCTION      83 231.292331 231.291563
## EXTENSION         85 231.291765 231.291058
## LO-REDUCTION      87 231.291563 231.291058
## LO-REDUCTION      89 231.291235 231.291058
## LO-REDUCTION      91 231.291224 231.291058
## REFLECTION        93 231.291114 231.291015
## LO-REDUCTION      95 231.291058 231.291015
## LO-REDUCTION      97 231.291045 231.291015
## REFLECTION        99 231.291029 231.291011
## LO-REDUCTION     101 231.291015 231.291011
## Exiting from Nelder Mead minimizer
##     103 function evaluations used
```

```
###--------------------
### Cokriging function
###-------------------

co_krige <- function(df,A,delta,r,obs_ind,name=NULL) {

  B <- A*bisquare_1d(H,delta=delta,r=r)*ds          ## Form B matrix
  Sigma <- makeSY(r = Dvec,
                  var1 = sigma2_1,
                  var2 = sigma2_21,
                  kappa1 = kappa1,
                  kappa2 = kappa21,
                  B = B)
  Zobs <- Z[obs_ind,]                               ## Subset the observations
  Q <- solve(Sigma[obs_ind,obs_ind] +
               sigmav^2 * Imat(length(obs_ind)))
  mu <- Sigma[,obs_ind] %*% Q %*% Zobs              ## Cokriging equations
  sd <- diag(Sigma -  Sigma[,obs_ind] %*% Q %*% t(Sigma[,obs_ind]))

  df[paste0(name,"_mu1")] <- mu[1:n1]               ## Save results
  df[paste0(name,"_mu2")] <- mu[-(1:n1)]
  df[paste0(name,"_sd1")] <- sd[1:n1]
  df[paste0(name,"_sd2")] <- sd[-(1:n1)]
  df
}
```

```
df <- df %>%
  co_krige(A=0,delta= 0, r = r, obs_ind = obs_ind,name="ind_model") %>%
  co_krige(A=non_symm_par[1],delta=0,r = non_symm_par[2],obs_ind = obs_ind,name="symm_model") %>%
  co_krige(A=A,delta=delta,r = r,obs_ind = obs_ind,name="true_model")
```

Plotting

The rest of the code (and the biggest part!) is devoted to plotting. Since this is terse, we do not discuss it in detail. It highly relies on knowledge of the package `ggplot2` and `tidyr`, the latter for putting the data into an appropriate format.

```
###--------------------
### Plotting
###--------------------
df_obs <- df %>%
  dplyr::select(s,Z1,Z2,keep_Z1,keep_Z2) %>%
  gather(obs,z,Z1:Z2) %>%
  filter((keep_Z2 == TRUE & obs == "Z2") | (keep_Z1 == TRUE & obs == "Z1"))

df_estY1 <- df %>%
  dplyr::select(s,samp1,ind_model_mu1,symm_model_mu1,true_model_mu1) %>%
  gather(process,z,samp1,ind_model_mu1,symm_model_mu1,true_model_mu1) %>%
  mutate(group = substr(process,1,3))



df_estY2 <- df %>%
  dplyr::select(s,samp2,ind_model_mu2,symm_model_mu2,true_model_mu2) %>%
  gather(process,z,samp2,ind_model_mu2,symm_model_mu2,true_model_mu2)

# df_estY2 <- df %>%
#  dplyr::select(-areas,-samp1,-Z1,-Z2,-keep_Z1,-keep_Z2) %>%
#  gather(process,z,-s)

obs_plot <- LinePlotTheme() +
  geom_point(data=df_obs,
             aes(x=s,y=z,shape=obs),
             size=3,alpha=1,guide=F) +
  theme(legend.title=element_blank(),
        plot.margin = grid::unit(c(3, 0, 0, 0),units = "mm"))+
  scale_shape_manual(values=c(1,20),guide=F) +
  ylab("")

est_plotY1_no_CIs <- LinePlotTheme() +
  geom_line(data=df_estY1,
            aes(x=s,y=z,colour=process,linetype=process,size=process)) +
  scale_linetype_manual(values=c("solid","dashed","dotted","dotdash"),
                        guide=FALSE) +
  scale_size_manual(values=c(0.4,1.3,1.3,1.3),guide=F) +
  scale_colour_manual(values=c("black","black","black","black"),
                      labels=c("Y1",
                               expression(paste(tilde(Y),1)),
```

```
                              expression(paste(hat(Y),1))),
                      name="",
                      guide=F) +
  ylab("")

est_plotY1 <- est_plotY1_no_CIs +
  geom_ribbon(data=df,aes(s,ymax=ind_model_mu1 + ind_model_sd1,
                          ymin = ind_model_mu1 - ind_model_sd1),alpha=0.2,fill="red") +
  geom_ribbon(data=df,aes(s,ymax=true_model_mu1 + true_model_sd1,
                          ymin = true_model_mu1 - true_model_sd1),alpha=0.2,fill="black") +
  geom_ribbon(data=df,aes(s,ymax=symm_model_mu1 + symm_model_sd1,
                          ymin = symm_model_mu1 - symm_model_sd1),alpha=0.2,fill="green")


est_plotY2 <- LinePlotTheme() +
  geom_line(data=df_estY2,
            aes(x=s,y=z,colour=process,linetype=process,size=process)) +
  scale_linetype_manual(values=c("solid","dashed","dotted","dotdash"),
                        guide=FALSE) +
  scale_size_manual(values=c(1,1.3,1.3,1.3),guide=F) +
  scale_colour_manual(values=c("black","orange","blue","red"),
                      labels=c("Y2","IM","TM"),
                      name="") +
  ylab("")
```

```
if(print_figs) ggsave(obs_plot,
                      filename = file.path(img_path,"sim_obs.eps"),
                      width=7,height=4,family="Arial")
if(print_figs) ggsave(est_plotY1_no_CIs,
                      filename = file.path(img_path,"sim_est_no_CIs.eps"),
                      width=7,height=4,family="Arial")
if(print_figs) ggsave(est_plotY1,
                      filename = file.path(img_path,"sim_est.eps"),
                      width=7,height=4,family="Arial")
```

```
## Warning in grid.Call.graphics(L_polygon, x$x, x$y, index): semi-
## transparency is not supported on this device: reported only once per page
```

```
if(print_figs) ggsave(est_plotY1,
                      filename = file.path(img_path,"sim_est.png"),
                      width=7,height=4,family="Arial")
if(show_figs) print(arrangeGrob(obs_plot,est_plotY1,ncol=1),
                    width=16,height=7,family="Arial")
```

```
Sigma_df <- expand.grid(s1 = df$s,comp1 = c("Y1","Y2"),s2 = df$s,comp2 = c("Y1","Y2")) %>%
  mutate(cov = c(Sigma))
Sigma_plot <- LinePlotTheme() +
    geom_tile(data=Sigma_df,aes(x=s2,y=s1,fill=cov)) +
    facet_grid(comp1 ~ comp2) +
  scale_fill_gradient2(low="white",high="black",mid="white") +
    coord_fixed() +
```

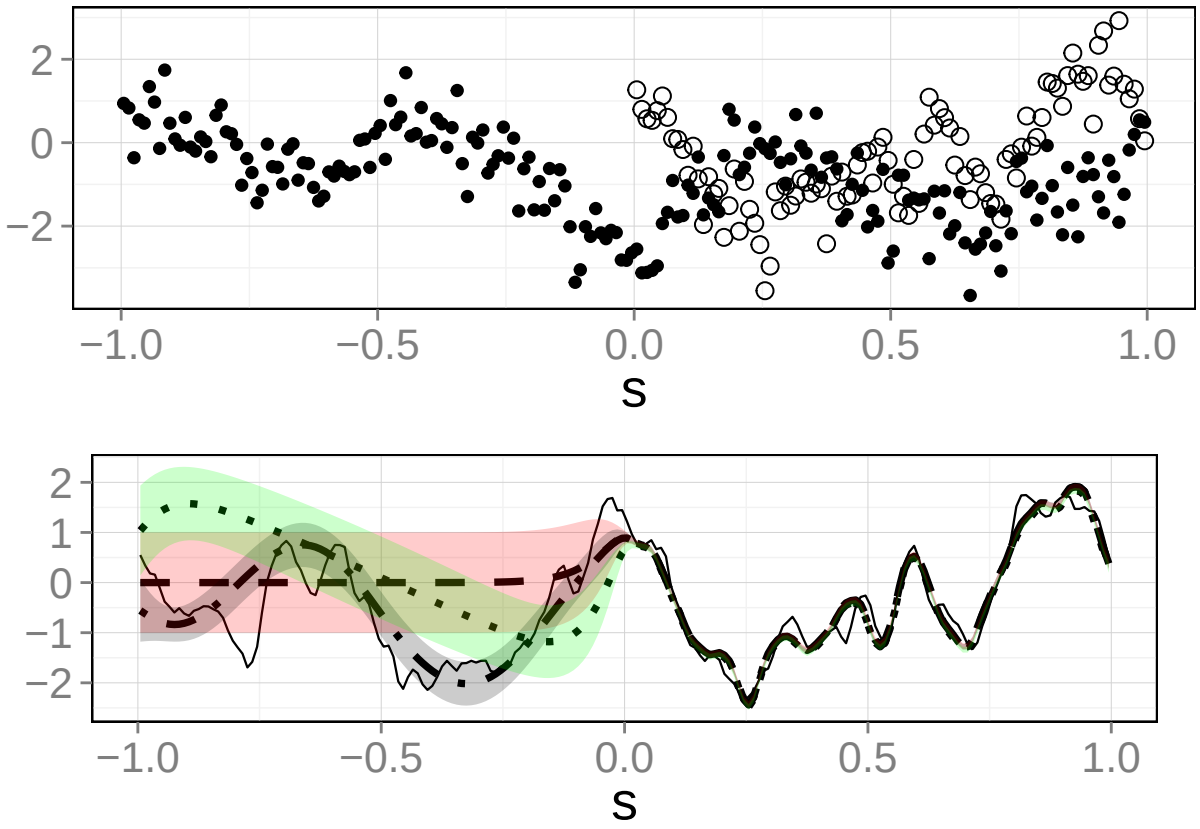

Figure 2: Cokriging using spatial covariances defined by the conditional approach. Top panel: The simulated observations Z_1 (open circles) and Z_2 (dots). Bottom panel: The hidden value Y_1 (solid line), the kriging predictor of Y_1 (dashed line), and the cokriging predictor of Y_1 (dotted line).

```
  ylab("s") + xlab("u") + scale_y_reverse() +
    theme(panel.margin = grid::unit(1, "lines"))
```

```
## Warning: Non Lab interpolation is deprecated
```

```
if(print_figs) ggsave(Sigma_plot,
                      filename = file.path(img_path,"Sigma.eps"),
                      width=8,height=7,family="Arial")
if(show_figs) print(Sigma_plot,width=16,height=7,family="Arial")
```

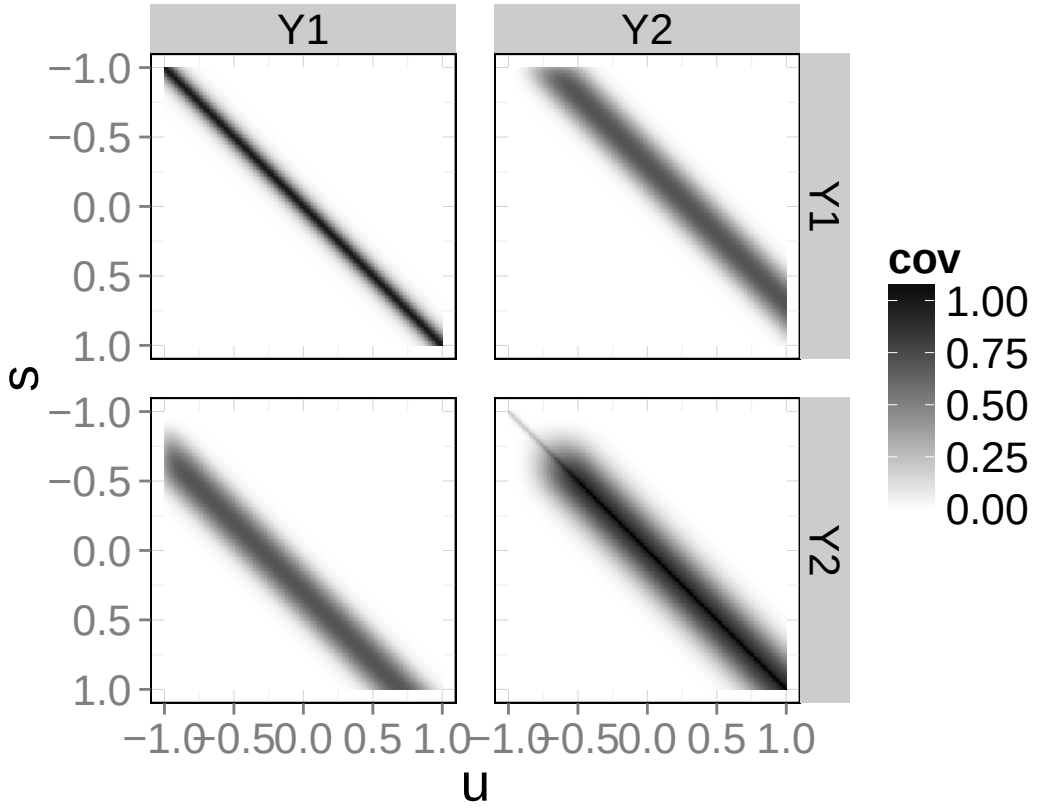

Figure 3: The covariance and cross-covariance matrix obtained using the function `makeSY`

```
g_all <- arrangeGrob(Sigma_plot,
                     arrangeGrob(obs_plot,est_plotY1_no_CIs,ncol=1),
                     ncol=2,widths=c(1,1))
if(print_figs) {
  cairo_ps(filename = file.path(img_path,"Fig1.eps"),
                     width=16,height=7,family="Arial")
   print(g_all)
    dev.off()
}
```

```
## pdf
##   2
```

References

Cressie, N., & Zammit-Mangion, A. (2015). Multivariate spatial covariance models: A conditional approach. *Submitted*.

Wackernagel, H. (1995). *Multivariate Geostatistics*. Berlin, DE: Springer.