Bivariate conditional spatial models: Simulation example in Section 3.2

Noel Cressie and Andrew Zammit-Mangion

Setting up

In this document we show the *R Software* (R Core Team, 2014) code used to reproduce the results shown in Section 3.2 of Cressie & Zammit-Mangion (2016). The code for the application study in Section 5 is available in a separate document. Both these documents are available as vignettes in the package `bicon` which can be installed by first installing and loading the package `devtools` and then running

```
install_github("andrewzm/bicon")
```

In order to run this code, a few packages are needed. For the package versions used here, please refer to the end of the document. The first, `Matrix`, is needed for algebraic operations while `dplyr` and `tidyr` are needed for data manipulation.

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

The other packages, listed below, 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 <-  0             ## 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 $b(s,v)$ is a bisquare function given by

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

where Δ is a shift parameter, r is the aperture, and A is a scaling parameter. 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 now can construct the full covariance matrix Σ,

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

To facilitate this construction we have provided a function `makeSY` in the package `bicon`, 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 displacements
B <- A*bisquare_1d(H,delta = delta,r = r)*ds   ## Find B
```

Above, the function `bisquare_1d` is also provided in `bicon`. 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 individual marginal and cross-covariance functions can then be illustrated by extracting individual rows from the block matrix Σ corresponding to the location $s = 0$ (the mid-point of D).

```
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]
```

The following code plots the covariance functions shown in Fig. 1.

```
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,width=12,height=10)
```

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 %>%
```

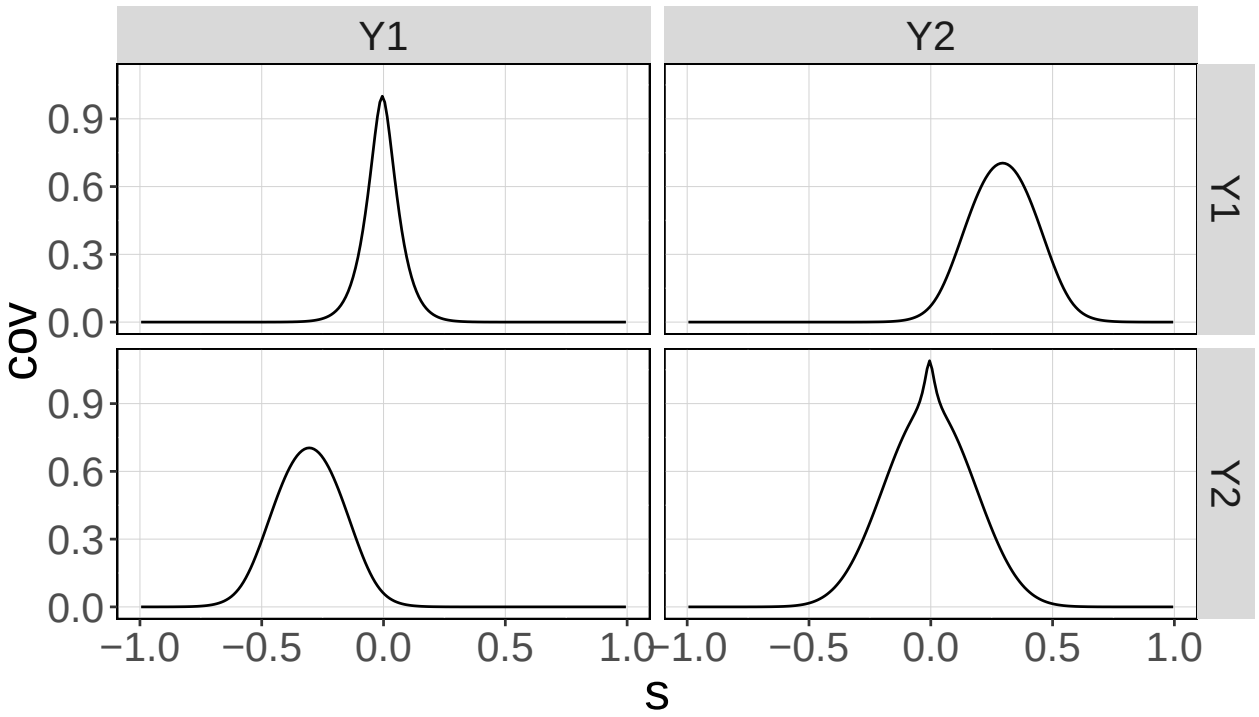

Fig. 1: The correlation and cross-correlation functions for the model of Section 3.2.

```
  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))                     ## Store 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.

```
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. Predictions and prediction errors are stored in the data frame `df`.

```
###--------------------
### 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,                         ## Construct Sigma
                  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] +               ## Compute precision
               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
}
```

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 used the cokriging equations to implement three different predictors

1. Auto-kriging predictor ($\widetilde{Y}_1$): A set to 0.
2. Cokriging predictor under misspecified model ($Y_1^\dagger$): A and r found using maximum likelihood with Δ fixed to zero.
3. Cokriging predictor under true model ($\hat{Y}_1$): A and r fixed to their true values.

Note that for predictor 1., cokriging with `A = 0` is identical to simple kriging on Y_1 using only Z_1, since under independence the system is *autokrigeable* (see Wackernagel, 1995, p. 149). For predictor 2. we need to define the log-likelihood and find the maximum likelihood parameters using an optimisation routine (`optim` in `R`). This is given by the following code, following which the maximum likelihood parameters are stored in `non_symm_par`.

```
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 displacements
  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) {                   ## Do not allow aperture to get too small
    return(Inf)
  } else {
    B <- theta[1]*bisquare_1d(H2,delta =0,
                              r = theta[2])*ds   ## Find B
    Sigma <- makeSY(r = Dvec2,
                    var1 = sigma2_1,
                    var2 = sigma2_21,
                    kappa1 = kappa1,
                    kappa2 = kappa21,
                    B = B) +                     ## Construct Sigma
      sigmav^2 * Imat(nrow(df2)*2)               ## Add on Meas. cov. matrix
    cholZ <- chol(Sigma)
    loglik <-                                    ## Compute log-likelihood
      -(-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),      ## init. conditions
        fn = loglik_Model,              ## log-likelihood
        hessian=FALSE,                  ## do not compute Hessian
        control=list(trace=6,           ## optim. options
                     pgtol=0,
                     maxit=3000))$par
```

Now that we have all the parameters we need to carry out (co)kriging, we can now simply pipe our original `df` through `co_krige` using differing values of `A`, `delta` and `r`.

```
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 relies on knowledge of the packages `ggplot2`, `dplyr`, and `tidyr`, the latter needed 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)

obs_plot <- LinePlotTheme() +
  geom_point(data=df_obs,
             aes(x=s,y=z,shape=obs),
             size=3,alpha=1) +
  theme(legend.title=element_blank(),
        plot.margin = grid::unit(c(3, 0, 0, 0),units = "mm"))+
  scale_shape_manual(values=c(1,20),
                     labels=c(expression(Z[1]),expression(Z[2]))) +
  ylab("Z")

df_estY1$process <- as.factor(df_estY1$process)
df_estY1$process <- relevel(df_estY1$process,2)
est_plotY1_no_CIs <- LinePlotTheme() +
  geom_line(data=df_estY1,
            aes(x=s,y=z,colour=process,linetype=process,size=process)) +
  theme(legend.title=element_blank(),
        plot.margin = grid::unit(c(3, 0, 0, 0),units = "mm"))+
  scale_linetype_manual(values=c("solid","dashed","dotted","dotdash"),
                        labels=c(expression(Y[1]),
                                 expression(tilde(Y)[1]),
                                 expression(Y[1]^"\u2020"),#"\u2020",
                                 expression(hat(Y)[1]))) +
  scale_size_manual(values=c(0.4,1.3,1.3,1.3),guide=F) +
  scale_colour_manual(values=c("black","black","black","black"),guide=F,name="") +
  ylab("Y")

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("")
```

The following code prints Fig. 2.

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

The following code prints Fig. 3.

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

```
if(print_figs) {
  g_all <- grid.arrange(Sigma_plot,
                    arrangeGrob(obs_plot,est_plotY1_no_CIs,ncol=1),
```

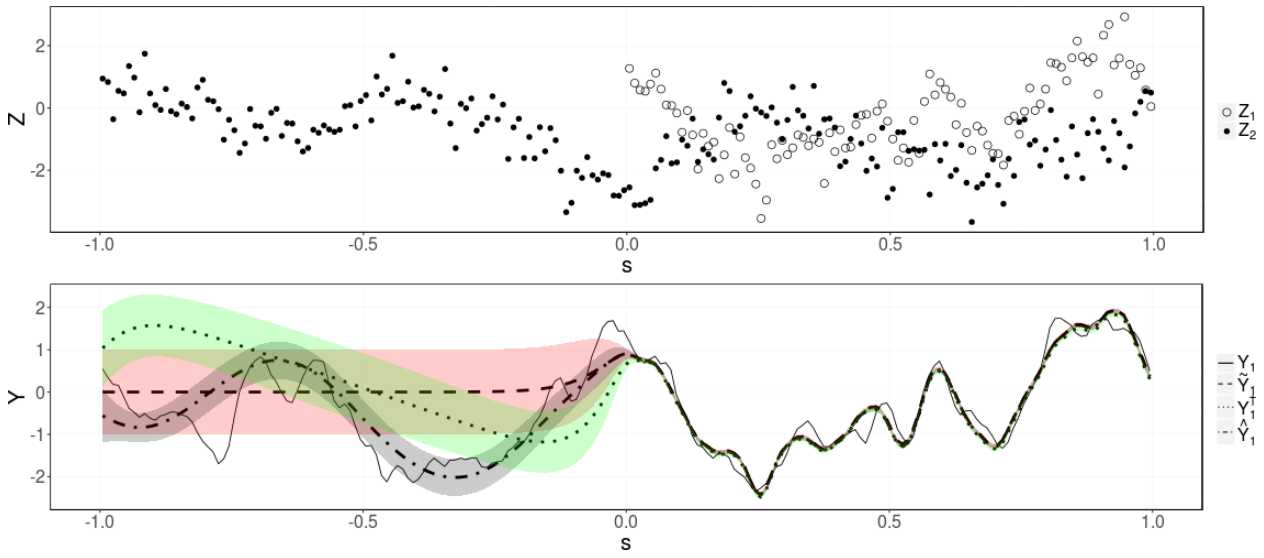

Fig. 2: Cokriging using spatial covariances defined by the conditional approach. Top panel: The simulated observations Z_1 (open circles) and Z_2 (dots). Bottom panel: The hidden value Y_1 (solid line), the kriging predictor $\widetilde{Y}_1$ (dashed line), the misspecified cokriging predictor $Y_1^{\dagger}$ (dotted line), and the cokriging predictor $\hat{Y}_1$ (dotted-dashed line). Prediction-error intervals are shown using the different shadings.

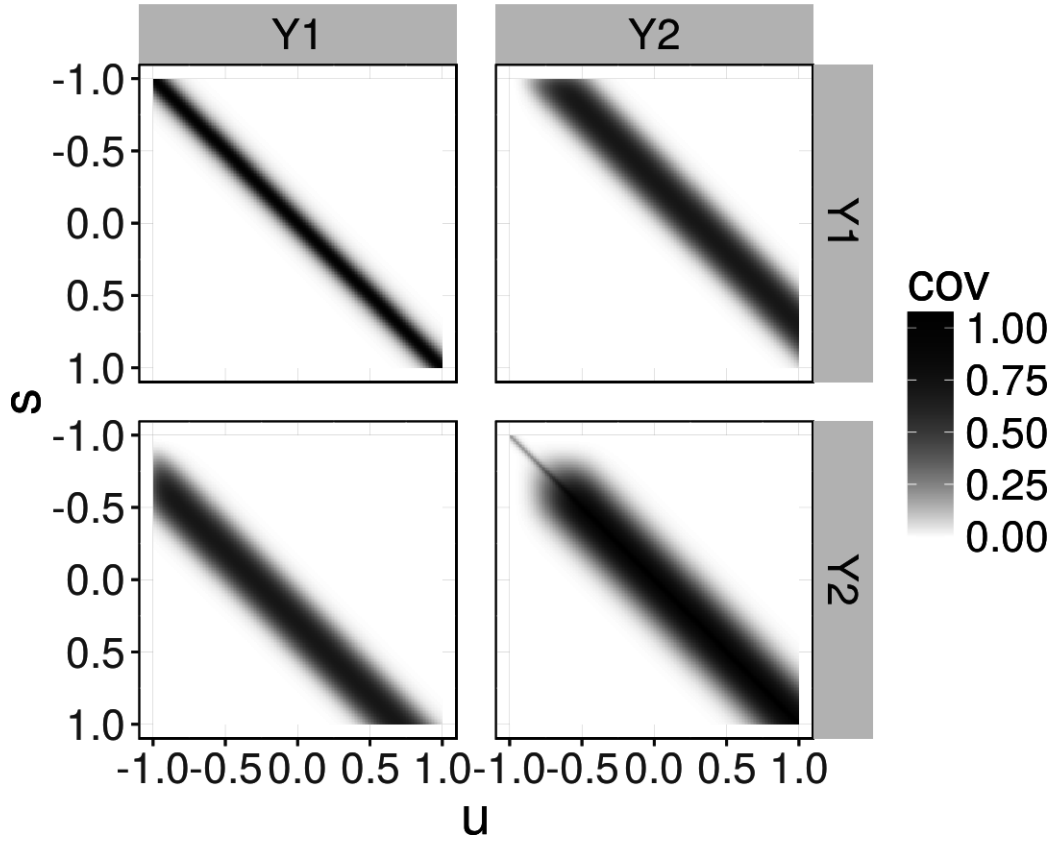

Fig. 3: The covariance and cross-covariance matrix obtained using the function `makeSY`.

```
                    ncol=2,widths=c(1,1))

  cairo_ps(filename = file.path(img_path,"Fig1.eps"),
                     width=16,height=7,family="Arial")
   grid.draw(g_all)
    dev.off()
}
```

Package versions

If you find that the code above is not reproducing the figures precisely, it is highly likely that this is due to some new, updated package implementing things differently. The package versions used to construct this document are listed below.

```
sessionInfo()
```

```
## R version 3.2.0 (2015-04-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.2 LTS
##
## locale:
##  [1] LC_CTYPE=en_AU.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_AU.UTF-8        LC_COLLATE=en_AU.UTF-8
##  [5] LC_MONETARY=en_AU.UTF-8    LC_MESSAGES=en_AU.UTF-8
##  [7] LC_PAPER=en_AU.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_AU.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] bicon_0.1.0     extrafont_0.17  gridExtra_2.0.0 ggplot2_2.1.0
## [5] tidyr_0.5.1     dplyr_0.5.0     Matrix_1.2-0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.7      Rttf2pt1_1.3.3   knitr_1.13       magrittr_1.5
##  [5] network_1.13.0   munsell_0.4.3    colorspace_1.2-6 lattice_0.20-31
##  [9] R6_2.1.3         gpclib_1.5-5     stringr_1.1.0    plyr_1.8.4
## [13] tools_3.2.0      gtable_0.2.0     DBI_0.5-1        deldir_0.1-12
## [17] extrafontdb_1.0  htmltools_0.3.5  lazyeval_0.2.0   yaml_2.1.13
## [21] assertthat_0.1   digest_0.6.10    tibble_1.2       akima_0.5-12
## [25] reshape2_1.4.1   formatR_1.4      evaluate_0.9     rmarkdown_0.8.1
## [29] labeling_0.3     sp_1.2-3         stringi_1.1.1    scales_0.4.0
```

References

Cressie, N., & Zammit-Mangion, A. (2016). Multivariate spatial covariance models: A conditional approach. *Biometrika, in press.*

R Core Team. (2014). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/.

Wackernagel, H. (1995). *Multivariate Geostatistics: An Introduction with Applications*. Berlin: Springer.