MVST applied to a 1-D bivariate problem

Andrew Zammit-Mangion, University of Wollongong, Australia

October 30, 2014

Abstract

MVST is an *R Software* package which facilitates the construction of Gaussian multi-variate spatio-temporal fields. These fields can be a-priori independent or co-regionalised. The package also allows the variates to be spatial-only, in which case they are assumed to be repeatedly observed in time. MVST is designed so that spatio-temporal fields can be decomposed using arbitrary basis functions. To date only finite elements are implemented, but in principle the package allows for any arbitrary decomposition, which can be different for each field.

In this vignette, we show the basic functionality of the package by considering a bivariate system, where two signals need to be separated from a single observation data set. This problem replicates that shown in the Annals of Glaciology paper.

1 Toy problem

First we load some essential packages. These include the plotting package `ggplot2` and a handy data-frame manipulation packages `dplyr`, followed by the in-house developed package `MVST`:

```
library(ggplot2)
library(dplyr)
library(MVST)
```

In this toy problem we consider two fields, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ on a one-dimensional grid of $n = 99$ cells. We index the spatial domain using the variable s.

```
set.seed(4)
n <- 99
s <- 1:n
```

We assume that $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are drawn from independent Gaussian processes with Matérn correlation functions. This class of correlation functions is fairly rich, and is defined by three parameters, ν, σ^2 and κ. ν defines the smoothness of the process, σ^2 defines the marginal variance and κ the scale (related to the process inverse correlation length). We set these processes to have different correlation lengths.

The covariance matrix of the 99 grid points is found by applying the respective Matérn function to the distance matrix computed from these 99 points (points closer to each other will have a higher covariance). The following covariance matrices can be plotted using the function `image` if needed.

```
S1 <- Matern(as.matrix(dist(s)), nu = 3/2,var=4, kappa = 0.1)
S2 <- Matern(as.matrix(dist(s)), nu = 3/2,var=4, kappa = 0.4)
```

The package `MVST` requires the construction of Gaussian Markov random fields (GMRFs) which are canonically defined using precision matrices (inverse covariance matrices). These GMRFs are defined as follows:

```
G1 <- GMRF(Q=as(chol2inv(chol(S1)),"dgCMatrix"))
G2 <- GMRF(Q=as(chol2inv(chol(S2)),"dgCMatrix"))
```

where `chol2inv(chol())` is an efficient way of computing the inverse of a matrix through the Cholesky factor (assuring matrix symmetry is preserved). Now that we have defined our processes we can construct a data frame containing all the information we require – namely the grid cells, two samples from these processes, the raw observations and the noisy observations (where the error has variance $\sigma_v^2 = 1$):

```
sigmav <- 1
df <- data.frame(s=s) %>%
      mutate(x1=sample_GMRF(G1),
             x2=sample_GMRF(G2),
             y_clean = x1 + x2,
             y = y_clean + sigmav*rnorm(n),
             x1_model = x1 + 2 + sample_GMRF(G1)*0.2,
             x2_est = y - x1_model)
```

Note that we have also included `x1_model` and `x2_est`. These are the (invented) biased model output of $\boldsymbol{x}_1$ ($\hat{\boldsymbol{x}}_1$) and the estimate of $\boldsymbol{x}_2$ which would be obtained if we simply subtracted this estimate from the observations. These will be visualised later on.

The utility of `MVST` is in constructing a graph where the user identifies what observation is measuring what signal. This is particulary useful in the context of the mass-balance assessment in the Antarctic ice sheet where

we have multiple observations and processes. Here we simply assume that the data is affected by both processes and to equal effect. Hence

$$\boldsymbol{y} = \boldsymbol{C}_1\boldsymbol{x}_1 + \boldsymbol{C}_2\boldsymbol{x}_2 + \boldsymbol{e} \tag{1}$$

where both $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are defined to be the identity matrix. This observation equation is set up using the `link` functions.

```
obs <- Obs(df=data.frame(x=s,z=df$y,std=sigmav))
C <- Imat(n)
L1 <- link(G1,obs,Cmat = C)
L2 <- link(G2,obs,Cmat = C)
```

Once we have the observations, the processes and the links all defined, we can construct the graph. `MVST` requires that a list of links and a list of edges is defined before constructing the graph, which is then compressed into a simple two-node network. Results are then obtained by a simple Gaussian update using the `Infer` function.

```
e <- new("link_list",list(L1,L2))
v <- new("block_list",list(G1=G1,G2=G2,O=obs))
G <- new("Graph",e=e,v=v)
G_reduced <- compress(G)
Results <- Infer(G_reduced)

## Doing Cholesky and Takahashi
```

`Results` contains all the information required to obtain inferences. The key object is `Results$Post_GMRF` which contains the posterior precision matrix and the posterior expectation `mu`. These can be added to our previous data frame as follows:

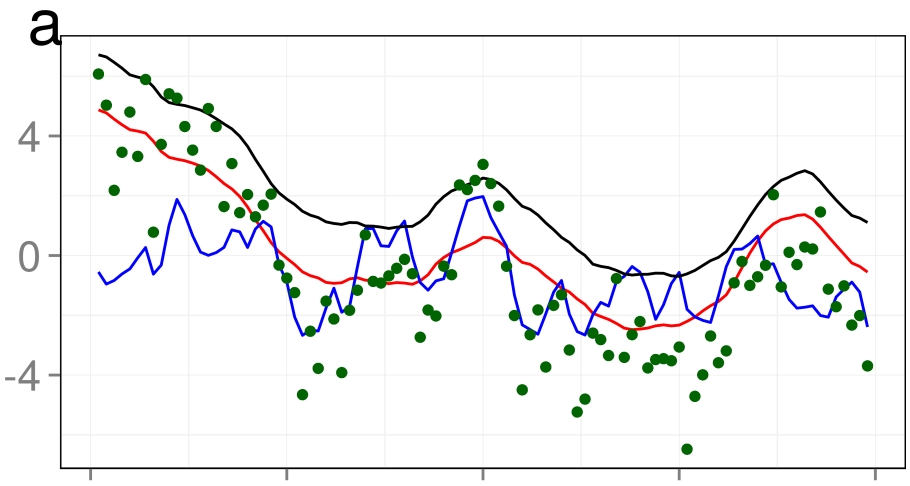
```
Sigma <- solve(Results$Post_GMRF@Q)
df <- mutate(df,
             x1_mean = Results$Post_GMRF@mu[1:n] ,
             x2_mean = Results$Post_GMRF@mu[-(1:n)] ,
             x1_std = sqrt(diag(Sigma))[(1:n)],
             x2_std = sqrt(diag(Sigma))[-(1:n)])
```

Now all that is left is plotting the results. Plotting is probably the most tedious part of the exercise and thus we include all the code below. Note, from the following figures, how using the model output blindly (without additional modelling effort) may lead to biases. The result we obtain from this approach will be unbiased if the prior assumption of the signals being centred around 0 is unbiased.

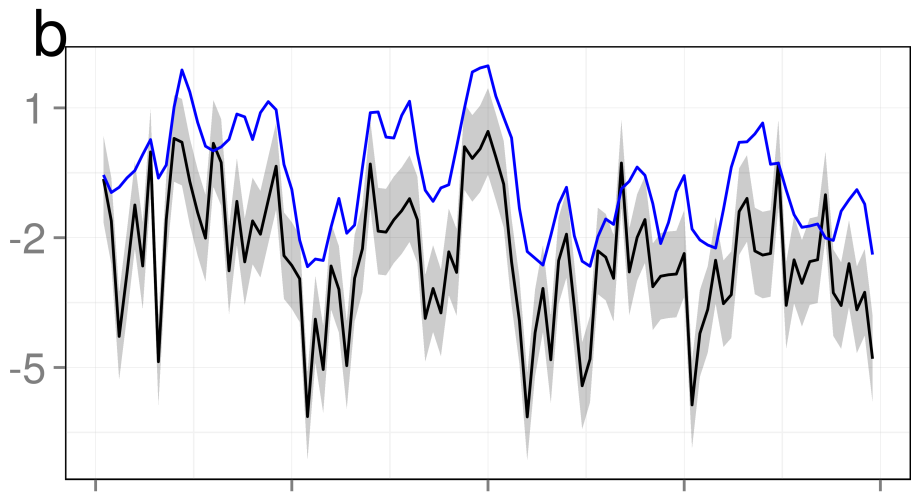
```
g <- LinePlotTheme() + geom_line(data=df,aes(x=s,y=x1),colour="red") +
     geom_line(data=df,aes(x=s,y=x2),colour="blue") + ylab("") +
     geom_line(data=df,aes(x=s,y=x1_model),colour="black") + ylab("") + xlab("") +
     geom_point(data=df,aes(x=s,y=y),colour="dark green",ize=2) +
     guides(colour=guide_legend(""))  +
     guides(colour=F) +
     ggtitle("a") +
     theme(plot.title = element_text(hjust = -0.04,vjust=-0.5)) +
     #theme(plot.margin=grid::unit(c(2,2,2,2),"mm")) +
     theme(axis.text.x=element_blank())

g1 <- LinePlotTheme() + geom_line(data=df,aes(x=s,y=x2_est,colour=bquote("x2")),colour="black") +
   geom_ribbon(data=df,aes(x=s, ymax = x2_est + sigmav, ymin = x2_est-sigmav),
               alpha=0.2,fill="black") + ylab("") +xlab("") +
    geom_line(data=df,aes(x=s,y=x2,colour="x2"),colour="blue") +
  guides(colour=guide_legend("",))  +
  guides(colour=F) +
  ggtitle("b") +
  theme(plot.title = element_text(hjust = -0.04,vjust=-0.5)) +
  theme(axis.text.x=element_blank()) +
  scale_y_continuous(breaks=seq(-8,1,3))

g2 <- LinePlotTheme() + geom_line(data=df,aes(x=s,y=x2_mean),colour="black") +
  geom_ribbon(data=df,aes(x=s, ymax = x2_mean + x2_std, ymin = x2_mean-x2_std),
              alpha=0.2,fill="black") + ylab("") +xlab("") +
  geom_line(data=df,aes(x=s,y=x2,colour="x2"),colour="blue") +
  guides(colour=guide_legend("")) +
  guides(colour=F) +
```

```
  ggtitle("c") +
  theme(plot.title = element_text(hjust = -0.04,vjust=-0.5)) +
  theme(axis.text.x=element_blank()) +
  scale_y_continuous(breaks=seq(-3,3,2))

g3 <- LinePlotTheme() + geom_line(data=df,aes(x=s,y=x1_mean),colour="black") +
  geom_ribbon(data=df,aes(x=s, ymax = x1_mean + x1_std, ymin = x1_mean-x1_std),
              alpha=0.2,fill="black") + ylab("") +
  guides(colour=guide_legend(""))  +
  guides(colour=F) +
  ggtitle("d") +
  theme(plot.title = element_text(hjust = -0.04,vjust=-0.5)) +
  geom_line(data=df,aes(x=s,y=x1,colour="x1"),colour="red") +
  xlab("n")
```

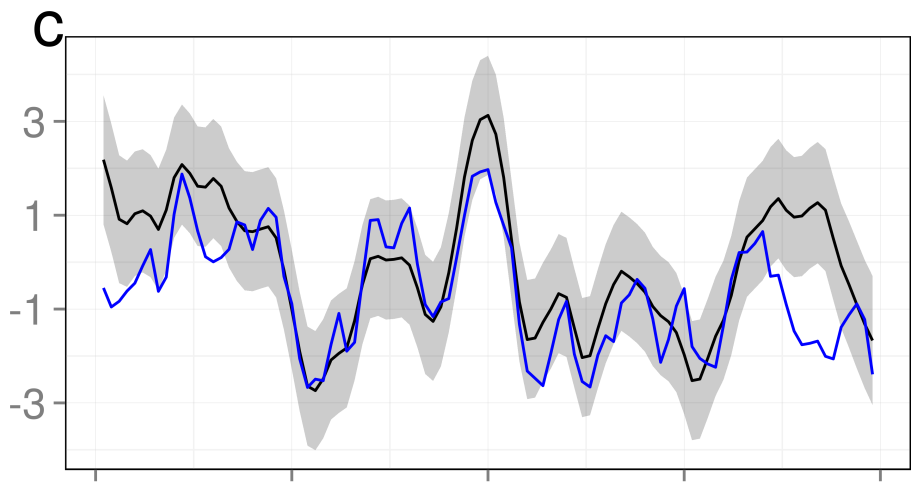
```
print(g)
```



```
print(g1)
```




```
print(g2)
```



```
print(g3)
```