

# Fixed Rank Kriging

Andrew Zammit Mangion<sup>1</sup> and Noel Cressie<sup>1</sup>

<sup>1</sup>National Institute for Applied Statistics Research Australia (NIASRA), School of Mathematics and Applied Statistics, University of Wollongong, New South Wales 2522, Australia

January 5, 2016

## Abstract

Fixed Rank Kriging (FRK) is a spatial or spatio-temporal prediction method for large datasets. FRK hinges on the use of a spatial random effects (SRE) model, in which the spatially correlated random effect is decomposed using a set of basis functions. Dimensionality reduction ensures computationally efficient prediction, while the reconstructed random spatial field is, in general, non-stationary. Although FRK is relatively straightforward to implement, and despite its extensive use in remote sensing contexts, to date no software package has been developed to facilitate this task. The present article discusses the development of a new *R* package, **FRK**, that facilitates FRK on the most commonly used manifolds ( $\mathbb{R}^1$ ,  $\mathbb{R}^2$  and  $\mathbb{S}^2$ ), both for spatial and spatio-temporal fields. In particular, it provides automatic basis function generation, automatic basic areal unit construction, functionality to incorporate multiple datasets with different supports, an expectation maximisation algorithm for parameter estimation, and functionality for point- or area-prediction. The package also has the capability for carrying out FRK on massive datasets on a distributed file system using a Hadoop backend. Use of the package is illustrated on several small examples and a massive dataset, implemented on the Amazon Elastic MapReduce web service.

## 1 Introduction

## 2 Fixed Rank Kriging

### 2.1 SRE model

### 2.2 Estimation

### 2.3 Prediction

### 2.4 Application of sparse matrix inversion

## 3 Distributed computation with Hadoop

### 3.1 Hadoop and Rhipe for parallelising data and computations

### 3.2 FRK Operations to parallelise

## 4 Package layout

### 4.1 Classes and methods

### 4.2 Overall program structure

### 4.3 1D Simple example

```

library(sp)
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##   filter, lag
##
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

library(FRK)
#   devtools::load_all("~/Wollongong/pkgs/FRK",
#   export_all = FALSE)
opts_FRK$set("progress",FALSE)
opts_FRK$set("parallel",0L)

```

```

## Load data
set.seed(1)
sim_process <- data.frame(x = seq(0.005,0.995,by=0.01)) %>%
  mutate(y=0,proc = sin(x*10) + 0.3*rnorm(length(x)))

sim_data <- sample_n(sim_process,50) %>%
  mutate(z = proc + 0.1*rnorm(length(x)), std = 0.1)
coordinates(sim_data) = ~x + y# change into an sp object

```

```

## Prediction (BAU) grid
grid_BAUs <- auto_BAUs(manifold=real_line(),data=sim_data,cellsize = c(0.01),type="grid")
grid_BAUs$fs = 1

```

```

## Set up SRE model
G <- auto_basis(m = real_line(),data=sim_data,
  nres = 2,
  regular=6,
  type = "bisquare",
  subsamp = 20000)

## [1] "Number of basis at resolution 1 = 6"
## [1] "Number of basis at resolution 2 = 12"

```

```

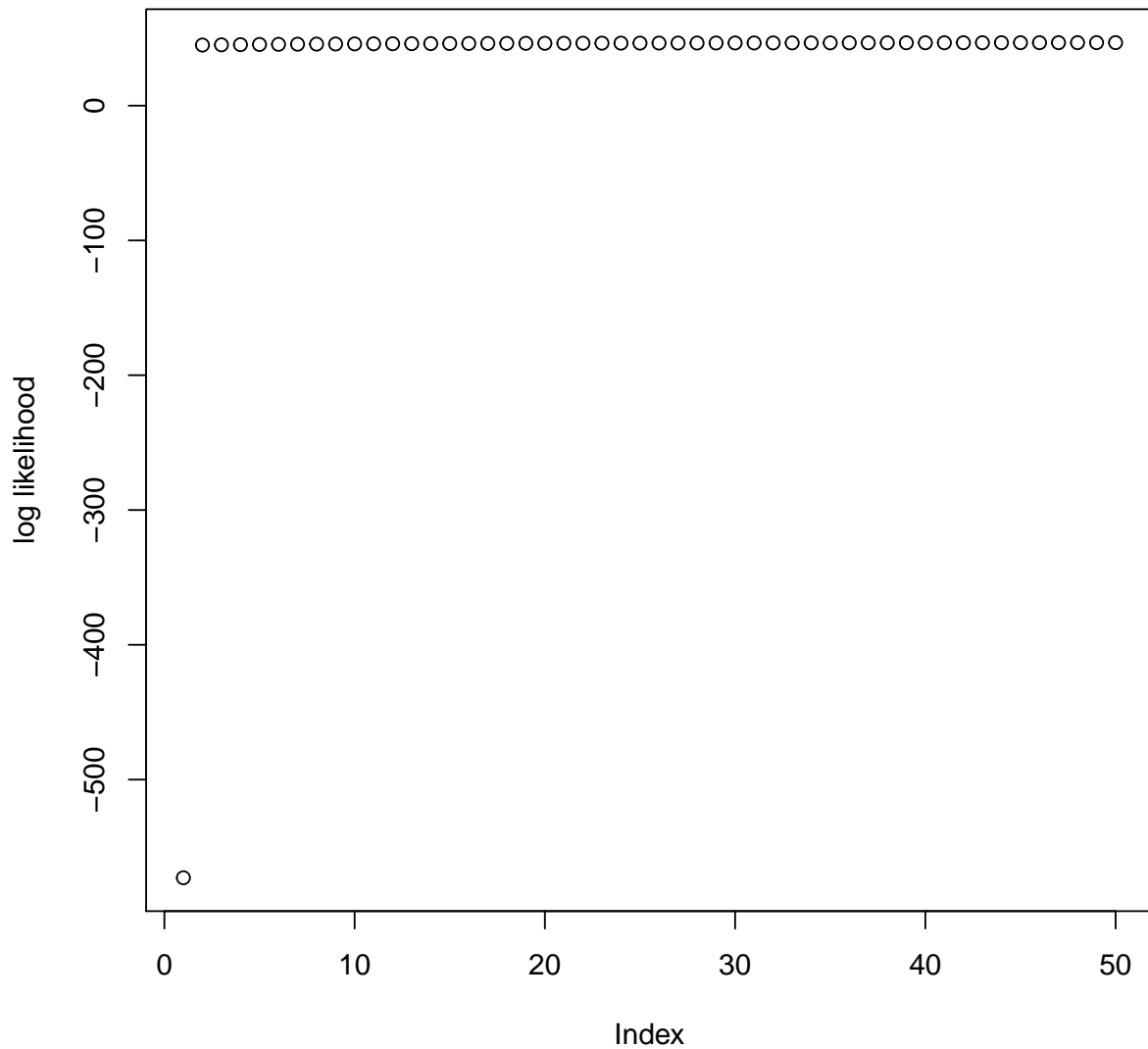
f <- z ~ 1
S <- SRE(f,list(sim_data),G,
  grid_BAUs,
  est_error = FALSE)

## [1] "Binned data in 0.0709999999999997 seconds"

```

```
S <- SRE.fit(S,n_EM = 50,tol = 1e-5,print_lik=TRUE)
```

```
## [1] "Maximum EM iterations reached"
```



```
## [1] "Warning: Ignoring constants in log-likelihood computation"
```

```
grid_BAUs <- SRE.predict(S,pred_locs = grid_BAUs,use_centroid = TRUE)
```

```
X <- grid_BAUs@data %>%  
  filter(x >= 0 & x <= 1)
```

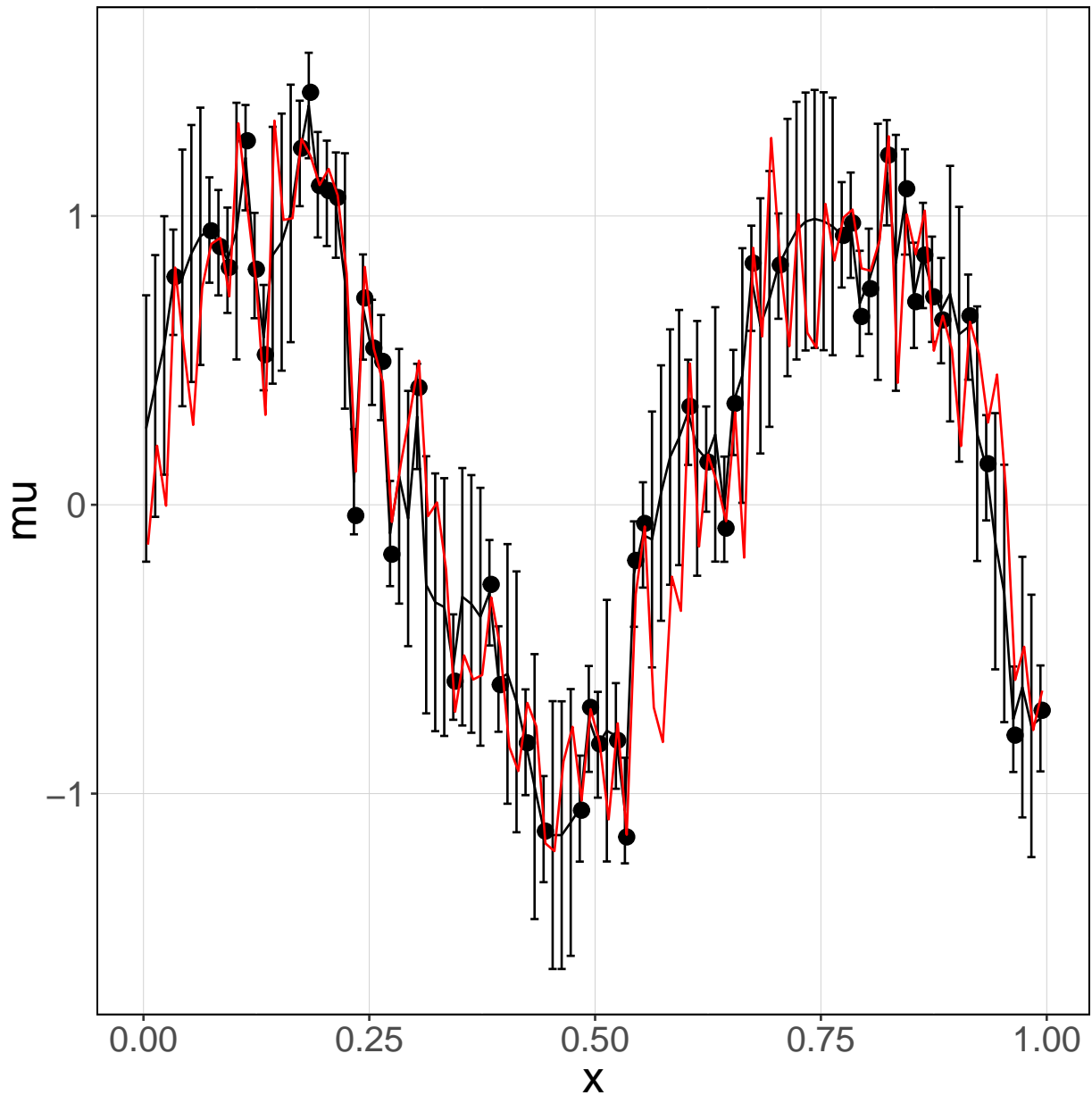
```
g1 <- LinePlotTheme() +
```

```

geom_line(data=X,aes(x,y=mu)) +
geom_errorbar(data=X,aes(x=x,ymax = mu + 2*sqrt(var), ymin= mu - 2*sqrt(var))) +
geom_point(data = data.frame(sim_data),aes(x=x,y=z),size=3) +
geom_line(data=sim_process,aes(x=x,y=proc),col="red")

print(g1)

```



#### 4.4 2D Simple example

```

library(sp)
library(ggplot2)
library(dplyr)

```

```

library(FRK)
# devtools::load_all("~/Wollongong/pkgs/FRK",
# export_all = FALSE)

opts_FRK$set("progress",FALSE)
opts_FRK$set("parallel",0L)
set.seed(1)

## Get data
data(meuse)
meuse$fs <- 1
coordinates(meuse) = ~x+y # change into an sp object

# Generate observations with large spatial support
data(meuse.grid)
gridded(meuse.grid) = ~x + y

HexPts2 <- spsample(meuse.grid,
                    type = "hexagonal",
                    cellsize = 100)
HexPols2 <- HexPoints2SpatialPolygons(HexPts2)
HexPols_df2 <- SpatialPolygonsDataFrame(HexPols2,
                                       over(HexPols2,meuse) %>%
                                       select(zinc)) %>%

  subset(!is.na(zinc))

if(0) { ## Need INLA from here on

  # Generate BAUs
  HexPols_df <- auto_BAUs(manifold = plane(),cellsize = c(50,50),type = "grid",data = meuse,conver
  HexPols_df$fs <- 1

  ## Generate basis functions
  G <- auto_basis(m = plane(),data=meuse,nres = 2,
                 prune=10,type = "Gaussian")

  ## Setup SRE model
  f <- log(zinc) ~ 1
  S <- SRE(f,list(meuse,HexPols_df2),BAUs = HexPols_df, G,est_error=T)
  S <- SRE.fit(S,n_EM = 10,print_lik=TRUE)

  ## Point predict
  HexPols_df <- SRE.predict(S,use_centroid = TRUE)

  X <- SpatialPolygonsDataFrame_to_df(sp_polys = HexPols_df,
                                       vars = c("mu","var"))

  g1 <- EmptyTheme() +
    geom_polygon(data=X,aes(x,y,fill=mu,group=id),
                 colour="light grey") +
    scale_fill_distiller(palette="Spectral",trans="reverse") +
    geom_point(data=data.frame(meuse),
               aes(x,y,fill=log(zinc))),

```

```

        colour="black",
        pch=21, size=3) +
    coord_fixed()

g2 <- EmptyTheme() +
  geom_polygon(data=X, aes(x,y, fill=sqrt(var), group=id),
    colour="light grey") +
  scale_fill_distiller(palette="Spectral", trans="reverse") +
  #geom_point(data=data.frame(meuse),
  #  aes(x,y), colour="black", pch=21, size=3) +
  coord_fixed()

print(g1)
print(g2)
}

```

4.5 Modifying the distance measure - a 1D space-time example

5 Global prediction of global mid-tropospheric CO2

6 Global prediction of sea-surface temperatures using Hadoop

7 Conclusion