

Abstract

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 Simple examples

```
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_BAU <- auto_BAU(manifold=real_line(),data=sim_data,cellsize = c(0.01),type="grid")

## Warning in points2grid(.): cell size from constant coordinate 2
## possibly taken from other coordinate

grid_BAU$fs = 1
```

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

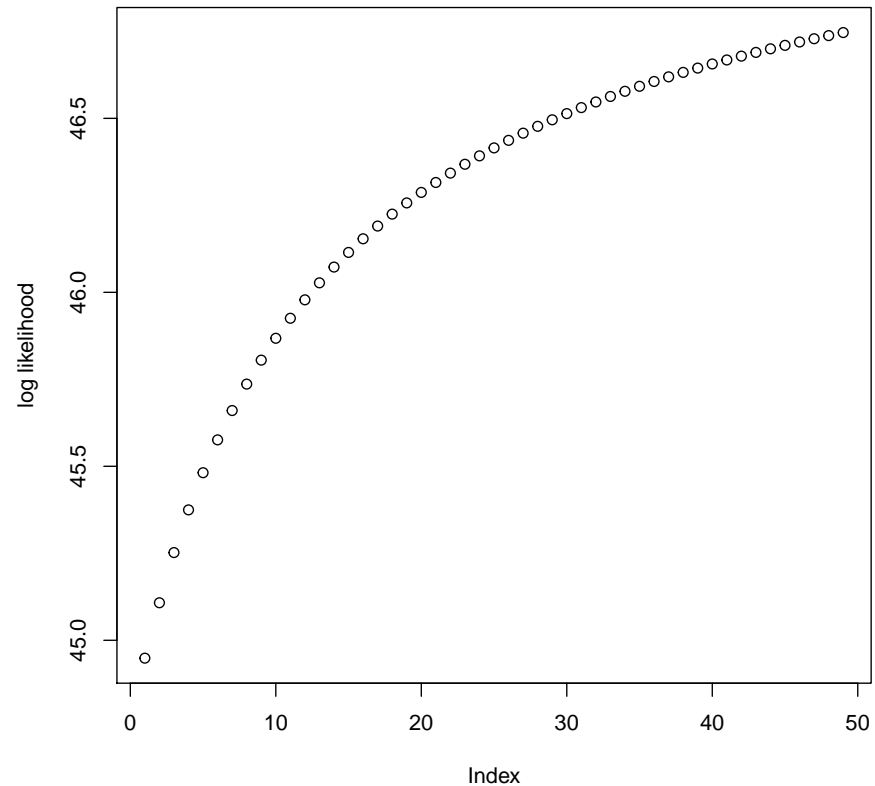
## [1] "Number of basis at resolution 1 = 6"
## [1] "Number of basis at resolution 2 = 12"
```

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

## [1] "Binned data in 0.0969999999999995 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)

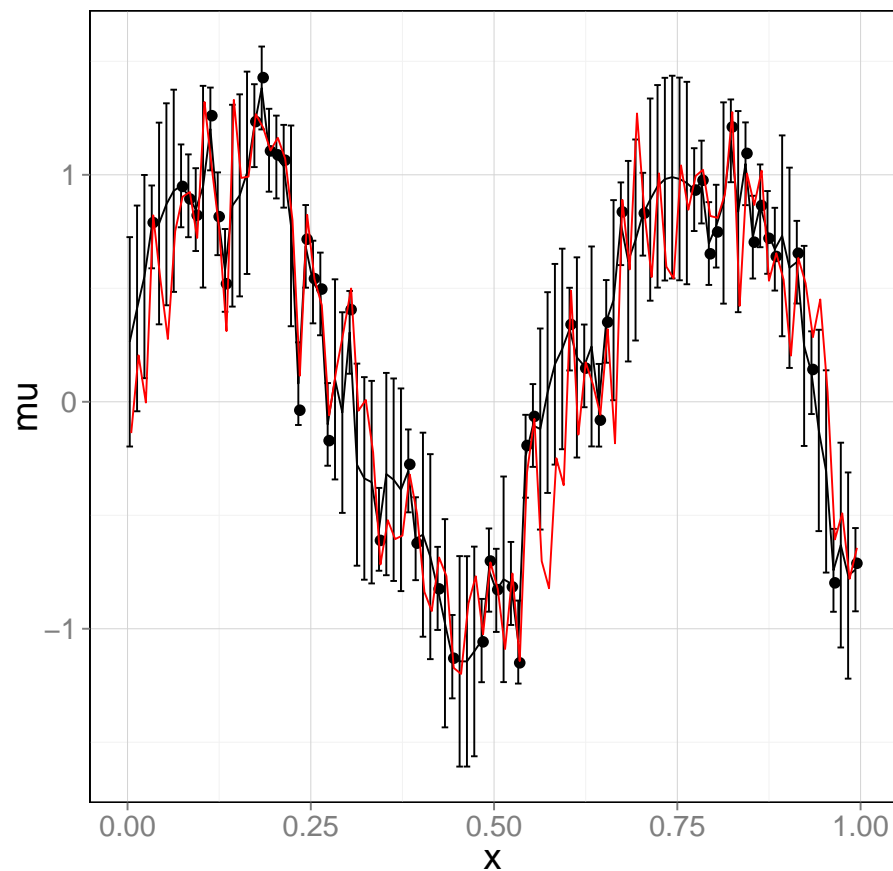
g1 <- LinePlotTheme() +
  geom_line(data=X,aes(x,y=mu)) +
  geom_errorbar(data=X,aes(x=x,ymax = mu + 2*sqrt(var), ymin= mu - 2*sqrt(var))) +
```

```

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)

```



Need an end comment

```

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(require(INLA)) { ## Need INLA from here on

  # Generate BAUs
  HexPols_df <- auto_BAUs(manifold = plane(), cellsize = c(50,50), type = "grid", data =
  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)
  }

## Loading required package: INLA
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:base':
##
##   crossprod, tcrossprod
##
## Loading required package: splines
## Loading required package: splancs
##
## Spatial Point Pattern Analysis Code in S-Plus
##
## Version 2 - Spatial and Space-Time analysis
## [1] "Number of basis at resolution 1 = 1"
## [1] "Number of basis at resolution 2 = 4"
## [1] "Binned data in 1.471 seconds"

## Warning in map_data_to_BAUs(data[[i]], BAUs, av_var = av_var, variogram.formula
= f, : Not accounting for multiple data in the same grid box during
variogram estimation. Need to see how to do this with gstat

## [1] "sigma2e estimate = 0.031397973693542"
## Called from: map_data_to_BAUs(data[[i]], BAUs, av_var = av_var, variogram.formula = f,
##   est_error = est_error)
## debug: if (is(variogram.formula, "formula") & (est_error == TRUE)) {
##   g <- gstat::gstat(formula = variogram.formula, data = new_sp_pts)
##   v <- gstat::variogram(g, cressie = T)
##   warning("Not accounting for multiple data in the same grid box during variogram estim

```

```

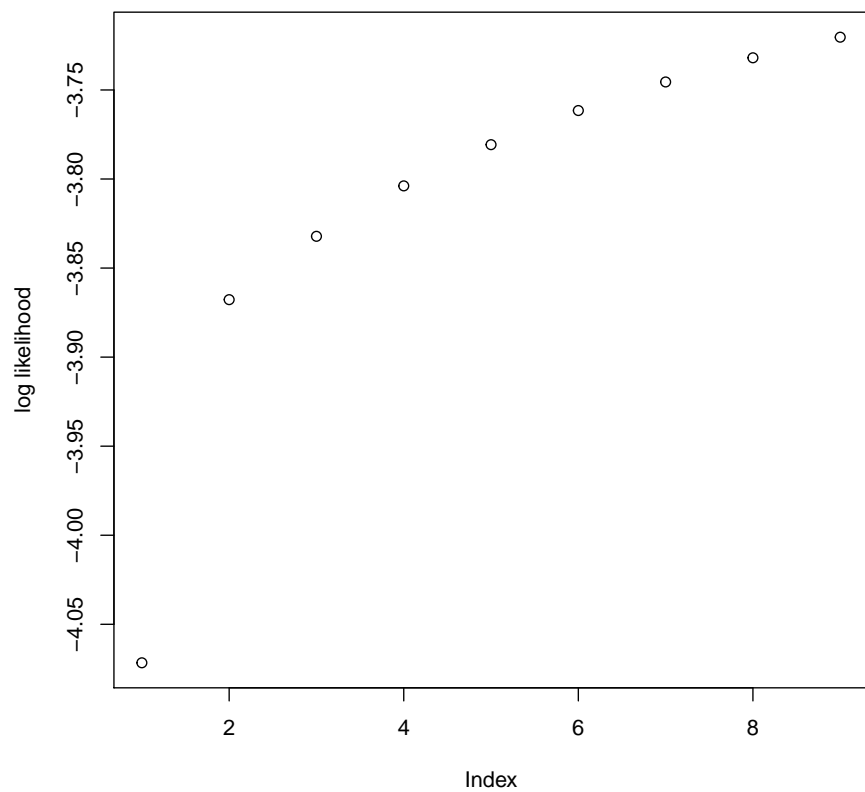
##      vgm.fit = gstat::fit.variogram(v, model = gstat::vgm(1, "Lin",
##      mean(v$dist), 1))
##      plot(v, vgm.fit)
##      print(paste0("sigma2e estimate = ", vgm.fit$psill[1]))
##      if (vgm.fit$psill[1] == 0)
##          stop("Observational error estimated to be zero. Please consider using finer BAUs")
##      new_sp_pts$std <- sqrt(vgm.fit$psill[1]/new_sp_pts$Nobs)
##  }
## debug: g <- gstat::gstat(formula = variogram.formula, data = new_sp_pts)
## debug: v <- gstat::variogram(g, cressie = T)
## debug: warning("Not accounting for multiple data in the same grid box during variogram es

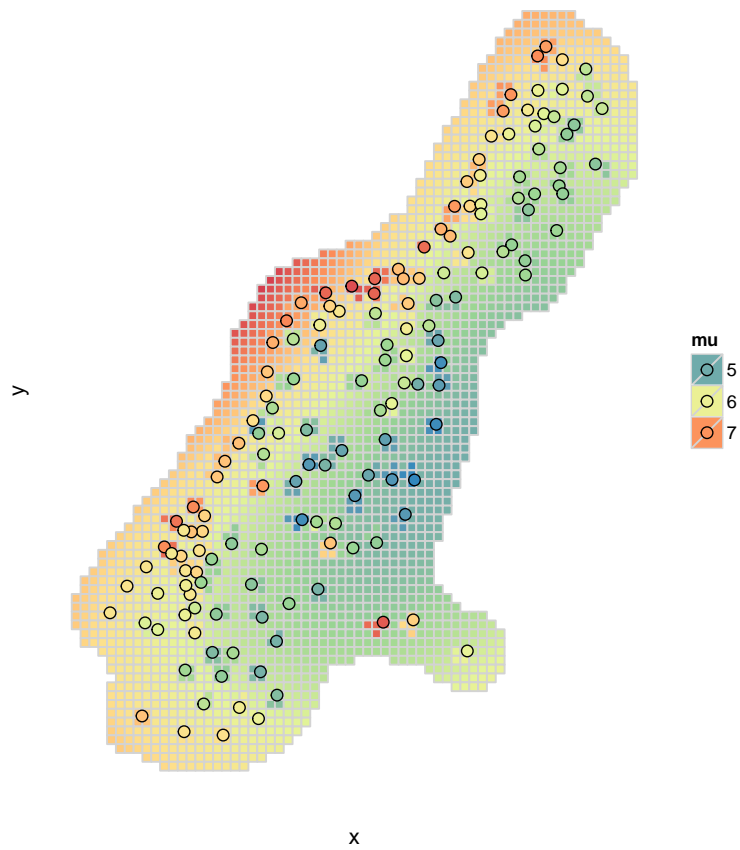
## Warning in map_data_to_BAUs(data[[i]], BAUs, av_var = av_var, variogram.formula
= f, : Not accounting for multiple data in the same grid box during
variogram estimation. Need to see how to do this with gstat

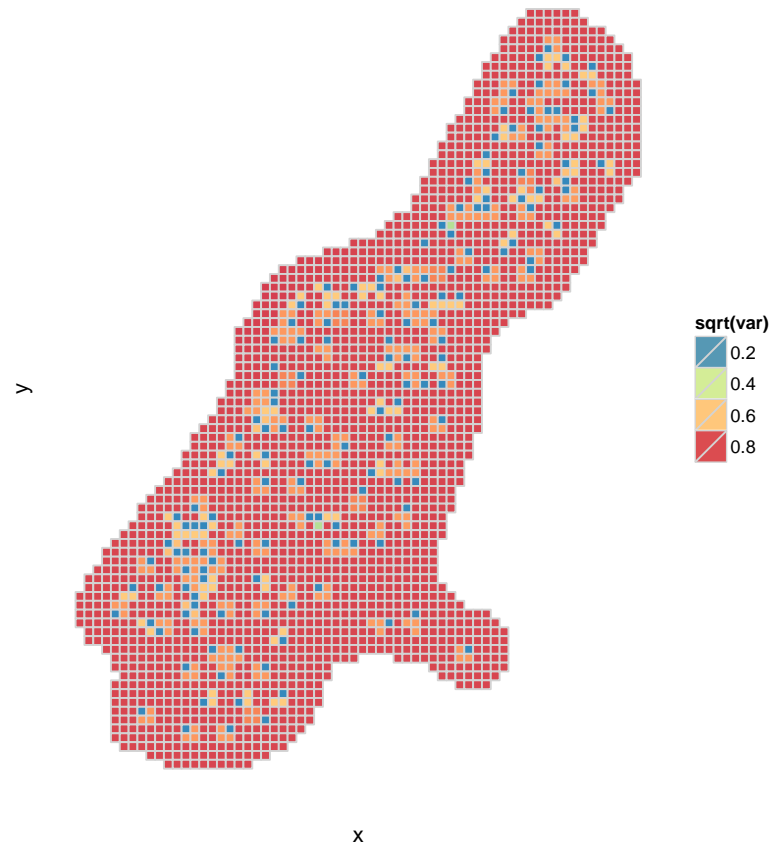
## debug: vgm.fit = gstat::fit.variogram(v, model = gstat::vgm(1, "Lin",
##      mean(v$dist), 1))
## debug: plot(v, vgm.fit)
## debug: print(paste0("sigma2e estimate = ", vgm.fit$psill[1]))
## [1] "sigma2e estimate = 0.00973076660686084"
## debug: if (vgm.fit$psill[1] == 0) stop("Observational error estimated to be zero. Please
## debug: new_sp_pts$std <- sqrt(vgm.fit$psill[1]/new_sp_pts$Nobs)
## debug: new_sp_pts
## [1] "Averaging over polygons"
## [1] "Maximum EM iterations reached"
## [1] "Warning: Ignoring constants in log-likelihood computation"

## Joining by: "id"

```







1-d and 2-d

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

5 Global prediction of global mid-tropospheric CO₂

```
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)
```

```
## Load data
data(AIRS_05_2003)
AIRS_05_2003 <- filter(AIRS_05_2003,day %in% 1:1) %>%
  mutate(std=co2std) %>%
  select(lon,lat,co2avgret,std)
coordinates(AIRS_05_2003) = ~lon+lat # change into an sp object
proj4string(AIRS_05_2003)=CRS("+proj=longlat")
```

```
## Prediction (BAU) grid
isea3h_sp_poldf <- auto_BAUs(manifold=sphere(),res=6,data=NULL,cellsize = c(5,5),type="grid")
isea3h_sp_poldf$fs = 1
```

```
## Set up SRE model
G <- auto_basis(m = sphere(),data=AIRS_05_2003,
  nres = 2,prune=15,
  type = "bisquare",
  subsamp = 20000)

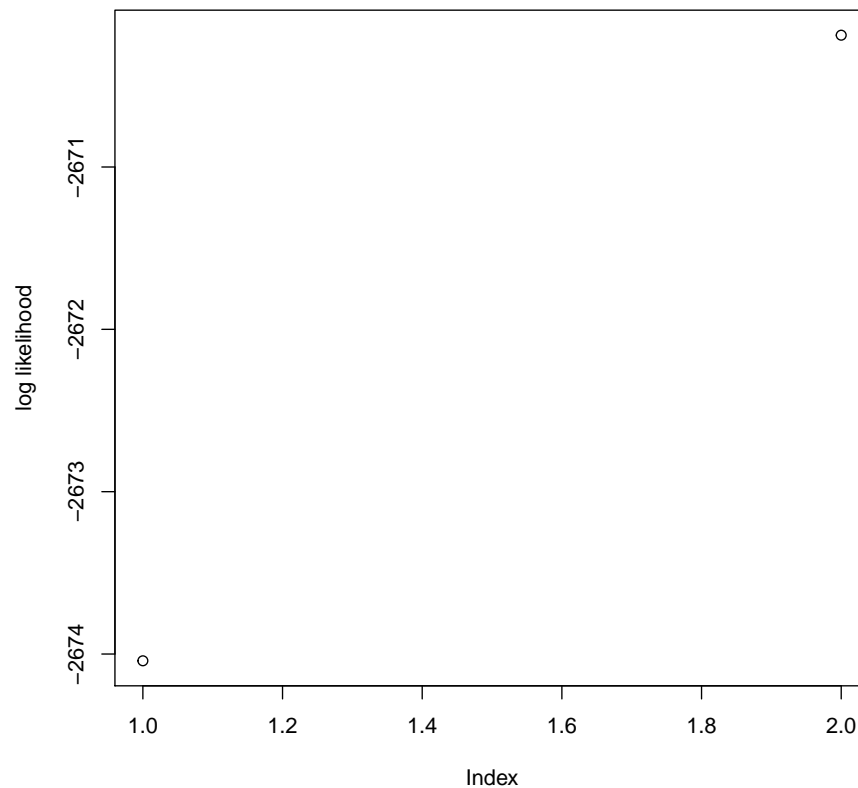
## [1] "Number of basis at resolution 1 = 32"
## [1] "Number of basis at resolution 2 = 90"
```

```
f <- co2avgret ~ lat + 1
S <- SRE(f,list(AIRS_05_2003),G,
  isea3h_sp_poldf,
  est_error = FALSE)

## [1] "Binned data in 1.308 seconds"
```

```
S <- SRE.fit(S,n_EM = 3,tol = 1e-5,print_lik=TRUE)

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



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

```
isea3h_sp_poldf <- SRE.predict(S,pred_locs = isea3h_sp_poldf,use_centroid = TRUE)
```

```
X <- SpatialPolygonsDataFrame_to_df(sp_polys = isea3h_sp_poldf,
                                     vars = c("mu","var"))

## Joining by: "id"

g1 <- (EmptyTheme() +
      geom_polygon(data=X,
                   aes(lon,lat,fill=mu,group=id))+
      scale_fill_distiller(palette="Spectral",trans="reverse") +
      coord_map("mollweide")) %>%
```

```

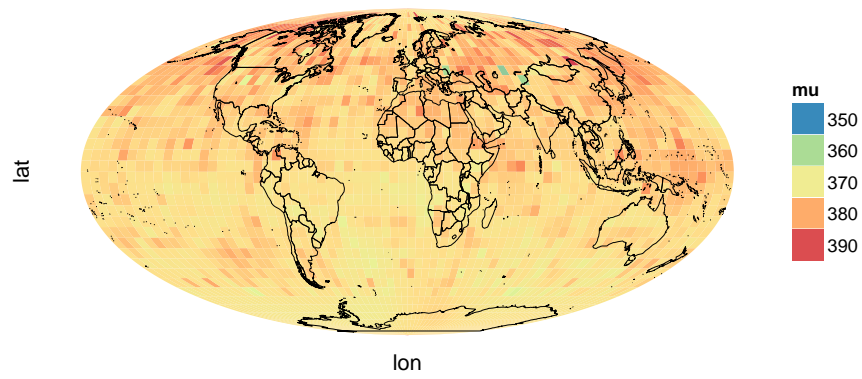
    draw_world(inc_border=FALSE)

mumin <- min(X$mu)
mumax <- max(X$mu)

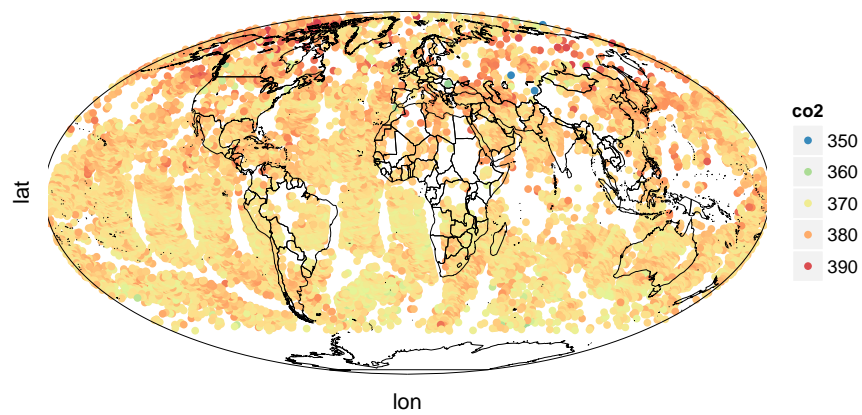
g2 <- (EmptyTheme() +
      geom_point(data=data.frame(AIRS_05_2003),
                 aes(lon,lat,
                     colour=pmin(pmax(
                                   co2avgret,mumin),
                                   mumax)),
                 size=2) +
      scale_colour_distiller(palette="Spectral",
                             trans="reverse",
                             guide_legend(title="co2")
      ) +
      coord_map("mollweide")) %>%
  draw_world(inc_border=TRUE)

print(g1)

```



```
print(g2)
```



6 Global prediction of sea-surface temperatures using Hadoop

7 Conclusion