01-spConjNNGP_Random

Frances Lin

2022-12-05

Load packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(pander)
library(here)
```

```
## here() starts at /Users/franceslin/GitHub/NNGP-with-Spatial-data-2022
```

```
library(spNNGP)
```

```
## Loading required package: coda
```

```
## Loading required package: Formula
```

```
## Loading required package: RANN
```

Load model object

```
bcef.c <- readRDS("../hpc/NNGP/real_data/bcef.c.rds")
```

Summary output

```
summary(bcef.c)
```

```
##
## Call:
## spConjNNGP(formula = FCH ~ PTC, data = BCEF.mod, coords = c("x",
##     "y"), n.neighbors = 10, theta.alpha = theta.alpha, sigma.sq.IG = c(2,
##     40), cov.model = "exponential", k.fold = 2, score.rule = "crps",
##     n.omp.threads = 40, fit.rep = TRUE, n.samples = 200)
##
## Model class is NNGP, method conjugate, family gaussian.
##
##             Estimate Variance
```

```
## (Intercept) 10.1422  0.0469
## PTC         0.0362   0.0000
## sigma.sq    34.7285  0.0241
## phi         8.7857   0.0000
## alpha       0.1000   0.0000
##
## samples size = 200
##             2.5%    25%     50%     75%     97.5%
## (Intercept) 9.7212  9.9831  10.1382 10.2753 10.5365
## PTC         0.0333  0.0352  0.0363  0.0373  0.0392
## sigma.sq    34.4394 34.6505 34.7309 34.8211 34.9960
## tau.sq      3.4439  3.4651  3.4731  3.4821  3.4996
```

Put it in a df

```
names(bcef.c)
```

```
##  [1] "beta.hat"             "ab"                   "bB.inv"
##  [4] "bb"                   "run.time"             "theta.alpha"
##  [7] "sigma.sq.hat"         "sigma.sq.var"         "beta.var"
## [10] "sigma.sq.IG"          "n.neighbors"          "cov.model"
## [13] "cov.model.indx"       "search.type"          "call"
## [16] "k.fold.scores"        "neighbor.info"        "p.beta.theta.samples"
## [19] "y.hat.samples"        "y.hat.quants"         "y.rep.samples"
## [22] "y.rep.quants"         "sub.sample"           "s.indx"
## [25] "coords"               "y"                    "X"
## [28] "type"
```

Notice that `theta.alpha` contains phi and alpha, instead of theta and alpha. Notice also that in `names(bcef.c)` there is no variance for neither phi and alpha. Need to come back!

```
# Create a df
df.bcef.c <- tibble(
  "b_0" = c(bcef.c$beta.hat[1], bcef.c$beta.var[1, 1]),
  "b_PTC" = c(bcef.c$beta.hat[2], bcef.c$beta.var[2, 2]),
  "sigma_2" = c(bcef.c$sigma.sq.hat, bcef.c$sigma.sq.var),
  "phi" = c(bcef.c$theta.alpha[1], 0),
  "alpha" = c(bcef.c$theta.alpha[2], 0)
)
#df.bcef.c
```

```
# Convert alpha to tau^2 OR
df.bcef.c <- df.bcef.c %>% mutate(
  "tau_2" = alpha * sigma_2
)
df.bcef.c
```

```
## # A tibble: 2 x 6
##       b_0      b_PTC sigma_2   phi alpha tau_2
##     <dbl>      <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 10.1    0.0362     34.7     8.79   0.1  3.47
## 2  0.0469 0.00000306  0.0241  0      0    0
```

```
# Transpose a tibble following [this post](https://stackoverflow.com/questions/42790219/how-do-i-transp
transpose_df <- function(df) {
  t_df <- data.table::transpose(df)
```

```
  colnames(t_df) <- rownames(df)
  rownames(t_df) <- colnames(df)
  t_df <- t_df %>%
    tibble::rownames_to_column(.data = .) %>%
    tibble::as_tibble(.)
  return(t_df)
}
```

```
df.bcef.c <- df.bcef.c %>% transpose_df()
df.bcef.c #%>% pander()
```

```
## # A tibble: 6 x 3
##   rowname     `1`         `2`
##   <chr>     <dbl>       <dbl>
## 1 b_0     10.1     0.0469
## 2 b_PTC    0.0362 0.00000306
## 3 sigma_2 34.7     0.0241
## 4 phi      8.79    0
## 5 alpha    0.1     0
## 6 tau_2    3.47    0
```

```
# Create lb and ub using base R
lb <- df.bcef.c$`1` - df.bcef.c$`2`
ub <- df.bcef.c$`1` + df.bcef.c$`2`
```

```
# Recreate df using base R
bcef_small <- df.bcef.c[c(1,2)]
df.bcef.c <- cbind(bcef_small, lb, ub)
colnames(df.bcef.c) <- c("rowname", "estimate", "lb", "ub")
```

```
# Fix tau^2
df.bcef.c[df.bcef.c$rowname == "sigma_2", c(2:4)] * df.bcef.c[df.bcef.c$rowname == "alpha", c(2)] -> df
df.bcef.c #%>% pander
```

```
##   rowname    estimate          lb          ub
## 1     b_0 10.14221937 10.09536762 10.18907111
## 2   b_PTC  0.03620543  0.03620236  0.03620849
## 3 sigma_2 34.72850130 34.70437992 34.75262267
## 4     phi  8.78571429  8.78571429  8.78571429
## 5   alpha  0.10000000  0.10000000  0.10000000
## 6   tau_2  3.47285013  3.47043799  3.47526227
```

```
# Remove alpha
df.bcef.c[df.bcef.c$rowname == "sigma_2", c(2:4)] * df.bcef.c[df.bcef.c$rowname == "alpha", c(2)] -> df
df.bcef.c.remove <- df.bcef.c[-5, ]
df.bcef.c.remove
```

```
##   rowname    estimate          lb          ub
## 1     b_0 10.14221937 10.09536762 10.18907111
## 2   b_PTC  0.03620543  0.03620236  0.03620849
## 3 sigma_2 34.72850130 34.70437992 34.75262267
## 4     phi  8.78571429  8.78571429  8.78571429
## 6   tau_2  3.47285013  3.47043799  3.47526227
```

```
# Save output
saveRDS(df.bcef.c, here("results", "df.bcef.c.rds"))
```

Consider write the above functions into a class.

β

```
bcef.c$beta.hat
```

```
##      (Intercept)        PTC
## [1,]    10.14222 0.03620543
```

```
bcef.c$beta.var
```

```
##                (Intercept)           PTC
## (Intercept)  0.0468517423 -2.192628e-04
## PTC         -0.0002192628  3.064511e-06
```

Theta and Alpha

```
# Extract theta & alpha
bcef.c$theta.alpha
```

```
##           phi alpha
## [1,] 8.785714   0.1
```

```
# Just to make sure
k.fold.scores.df <- bcef.c$k.fold.scores %>% as.data.frame()
k.fold.scores.df[which.min(k.fold.scores.df$crps), ]
```

```
##         phi alpha   rmspe     crps
## 61 8.785714   0.1 3.16315 1.653082
```

Plot theta.alpha

Check to see whether ϕ in the package spNNGP paper is ϕ from Matern covariance function or ϕ in R correlation function. Notice that ϕ in the the plot in the package spNNGP paper is not 4.93!

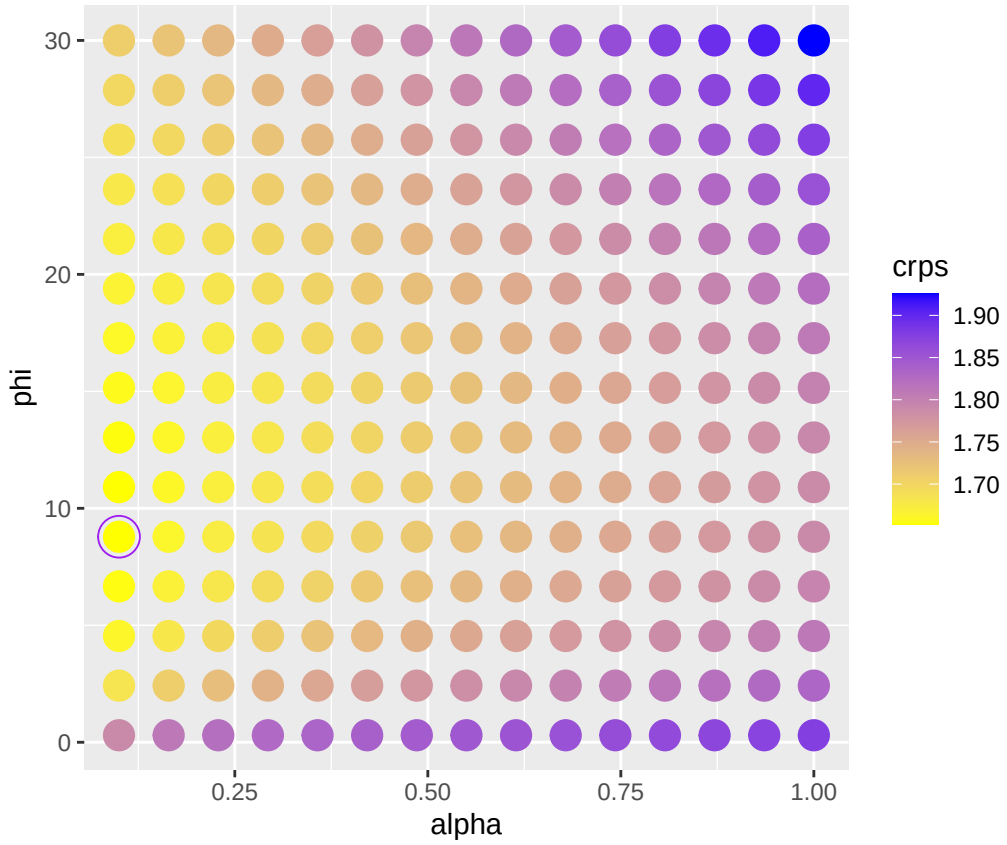
```
k.fold.scores <- bcef.c$k.fold.scores %>% as_tibble()
k.fold.scores %>% head()
```

```
## # A tibble: 6 x 4
##     phi alpha rmspe  crps
##   <dbl> <dbl> <dbl> <dbl>
## 1   0.3 0.1    3.35  1.79
## 2   0.3 0.164  3.38  1.81
## 3   0.3 0.229  3.39  1.82
## 4   0.3 0.293  3.40  1.83
## 5   0.3 0.357  3.41  1.83
## 6   0.3 0.421  3.42  1.84
```

Differ a lot from the plot in the package spNNGP paper? crps range is still off.

```
k.fold.scores %>% ggplot(aes(x = alpha, y = phi, color = crps)) +
  geom_point(size = 5) +
  geom_point(data = k.fold.scores %>% filter(crps == min(crps)), # Hello!
             pch = 21,
             size = 7,
             colour = "purple") +
  scale_color_gradient(low = "yellow", high = "blue") +
```

```
  theme(aspect.ratio=1) -> p_crps
p_crps
```




```
# Save output
saveRDS(p_crps, here("results", "p_crps.rds"))
```

Load model diagnostics

```
bcef.c.diag <- readRDS("../hpc/NNGP/real_data/bcef.c.diag.rds")
#bcef.c.diag
```

```
names(bcef.c.diag)
```

```
## [1] "y.hat.samples" "y.rep.samples" "s.indx"        "sub.sample"
## [5] "GP"            "GRS"
```

`GPD` a data frame holding the values needed to compute the predictive criterion D = G + P defined by Gelfand and Ghosh (1998). The GPD data frame includes rows labeled G a goodness of fit, P a penalty term, and D the criterion (lower is better).

Paper has

G = 6403563

P = 3502314

D = 9905877

Still off but not terrible!

```
bcef.c.diag$GP
```

```
##     value
## G 6490429
## P 3499399
## D 9989828
```

`GRS` a scoring rule, see Equation 27 in Gneiting and Raftery (2007) for details, with larger values of GRS indicating better model fit.

Paper has

GRS = -539539.6

Still off but not terrible!

```
bcef.c.diag$GRS
```

```
##         value
## GRS -541984.9
```

```
bcef.c.diag
```

```
## Chain sub.sample:
## start = 1
## end = 200
## thin = 1
## samples size = 200
##     value
## G 6490429
## P 3499399
## D 9989828
##
##         value
## GRS -541984.9
```