13\_homework3

Randy

5/12/2021

Kindly Matt helps me with this homework, However it is really hard for me to understand his codes.

## question1

1. when any variables or outcomes can be treated as a smooth curve or function
2. when the rates of change, the harmonic pattern, the phases and amplitude, or the derivatives of the curves also provide information
3. when the current discrete scale hypothetically can be extend to a infinitely dimensional but impossible to use linear regressions, and an flexible approximation is good enough to summarized the overall pattern

library("tidyverse")

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.0.4

## Warning: package 'tidyr' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library("mgcv")

## Loading required package: nlme

##   
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':  
##   
## collapse

## This is mgcv 1.8-33. For overview type 'help("mgcv-package")'.

library("refund")

## Warning: package 'refund' was built under R version 4.0.4

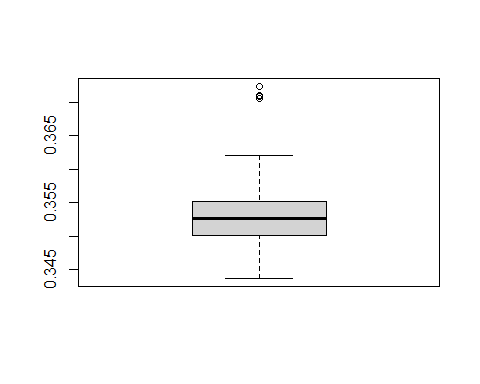
# number of participants (functions)  
N <- 200  
  
# number of observations per function  
J <- 50  
  
# functional domain of observed functions  
sind <- seq(0, 1, len = J)  
  
## fixed effects  
f0 <- function(s) 0.5 + (s - 0.5)^2 # f\_0(s)  
f1 <- function(s) sin(2 \* pi \* s) # f\_1(s)  
x <- rnorm(N) # x\_i  
  
## random effects  
# number of \phi  
K <- 4  
  
# true eigenvalues (defines variance of \xi\_i)  
lambda <- 0.5^(0:(K - 1))  
  
# \phi\_k  
Phi <- sqrt(2) \* cbind(  
 sin(2 \* pi \* sind), cos(2 \* pi \* sind),  
 sin(4 \* pi \* sind), cos(4 \* pi \* sind))  
  
# \xi\_{ik}  
xi\_raw <- matrix(rnorm(N \* K), N, K)  
xi <- xi\_raw %\*% diag(sqrt(lambda))  
  
# simulate b\_i(s) as \sum\_k \xi\_ik \phi\_k(t)  
bi <- xi %\*% t(Phi)  
  
# simulate linear predictor  
# by adding the fixed effects  
# to the random effects  
# note that vapply will return the results  
# in matrix format with one column per function  
# instead of the more common 1 row per function  
fixed\_eff <- vapply(1:N,   
 function(i, sind)   
 f0(sind) + f1(sind) \* x[i],   
 numeric(J),   
 sind = sind)  
  
# because the fixed effect matrix is transposed,  
# need to transpose the ranom effect matrix b\_i(s)  
eta <- fixed\_eff + t(bi)  
  
# simulate outcome by applying g^{-1}  
expit <- function(x) 1 / (1 + exp(-x))  
ptrue <- expit(eta)  
  
# simulate the outcome.  
Y <- vapply(ptrue, function(x) rbinom(n = 1, size = 1, prob = x), numeric(1))   
Y <- matrix(Y, N, J, byrow = TRUE)

## 2a

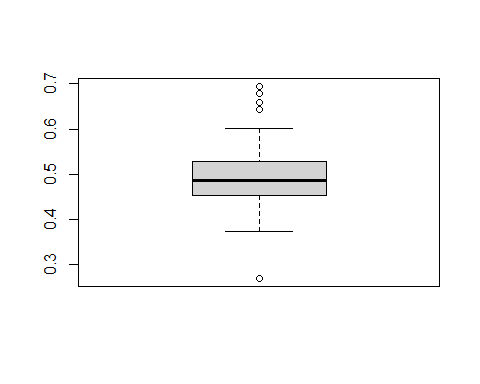
set.seed(666)  
# write a function that simulates the dataset  
simulation <- function(iteration, p = ptrue, J = 50, N = 200) {  
 sind <- seq(0, 1, len = J)  
 x <- rnorm(N)  
   
 # generate the outcomes  
 Y <- vapply(p, function(x) rbinom(n = 1, size = 1, prob = x), numeric(1))  
 Y <- matrix(Y, N, J, byrow = TRUE)  
   
 # make a dataframe (or matrix) with this data  
 data <- data.frame(  
 y = as.vector(t(Y)),  
 x = rep(x, each = J),  
 id = factor(rep(1:N, each = J)),  
 sind = rep(sind, N))  
   
 return(data)  
}  
  
# df <- map(1:100, simulation)  
# save(df, file = "13\_homework3\_df.Rdata")  
  
load("13\_homework3\_df.Rdata")  
  
# Warning in doTryCatch(return(expr), name, parentenv, handler) :  
# restarting interrupted promise evaluation  
# Warning in doTryCatch(return(expr), name, parentenv, handler) :  
# restarting interrupted promise evaluation  
# Warning in chol.default(qrx$R, pivot = TRUE) :  
# the matrix is either rank-deficient or indefinite

# models <- map(df, ~ bam(y ~ s(sind, k = 10) +  
# s(sind, by = x, bs = "cr", k = 10) +  
# ti(id, sind, bs = c("re", "cr"), k = c(5, 5), mc = c(TRUE, FALSE)),  
# data = .x,  
# method = "fREML",  
# family = binomial,  
# discrete = TRUE))  
#   
# save(models, file = "13\_homework3\_models.Rdata")

# data <- data.frame(sind = sind,  
# x = 1,  
# id = factor(1))  
#   
# f0\_s <- f0(sind)  
# f1\_s <- f1(sind)  
#   
# int\_mse <- var\_mse <-   
# int\_cover <- var\_cover <-   
# int\_lower <- int\_upper <-   
# var\_lower <- var\_upper <- rep(NA, 100)  
# prediction <- list()  
#   
# for (i in 1:100){  
# prediction[[i]] <- predict(models[[i]],  
# data,  
# type = "terms",  
# se.fit = T)  
#   
# int <- prediction[[i]]$fit[, "s(sind)"]  
# var <- prediction[[i]]$fit[, "s(sind):x"]  
#   
#   
# int\_mse[i] <- mean((int - f0\_s)^2)  
# var\_mse[i] <- mean((var - f1\_s)^2)  
#   
#   
# int\_se <- prediction[[1]]$se.fit[, "s(sind)"]  
# var\_se <- prediction[[1]]$se.fit[, "s(sind):x"]  
#   
# int\_lower[i] <- int - 2 \* int\_se  
# int\_upper[i] <- int + 2 \* int\_se  
# var\_lower[i] <- var - 2 \* var\_se  
# var\_upper[i] <- var + 2 \* var\_se  
#   
# cover\_int <- ifelse((f0\_s > int\_lower) & (f0\_s < int\_upper), 1, 0)  
# cover\_var <- ifelse((f1\_s > var\_lower) & (f1\_s < var\_upper), 1, 0)  
#   
# int\_cover[i] <- mean(cover\_int)  
# var\_cover[i] <- mean(cover\_var)  
# }  
  
  
load("homework3\_predict.Rdata")  
load("homework3\_int\_mse.Rdata")  
load("homework3\_var\_mse.Rdata")  
  
load("homework3\_int\_cover.Rdata")  
load("homework3\_var\_cover.Rdata")  
  
load("homework3\_int\_upper.Rdata")  
load("homework3\_int\_lower.Rdata")  
load("homework3\_var\_uper.Rdata")  
load("homework3\_var\_loer.Rdata")  
  
  
## this is the idea of getting all the things together in one data frame.   
## I just do not have the energy and time to do it.  
# fit\_sind <- prediction %>%  
# as.data.frame() %>%  
# select(contains("fit.s.sind")) %>%  
# select(!contains("se"))  
#   
# fit\_se <- prediction %>%  
# as.data.frame() %>%  
# select(contains("fit.s.sind")) %>%  
# select(contains("se"))  
  
dfs <- rbind(df)  
# save(predict, file = "homework3\_predict.Rdata")  
# save(int\_mse, file = "homework3\_int\_mse.Rdata")  
# save(var\_mse, file = "homework3\_var\_mse.Rdata")  
# save(int\_lower, file = "homework3\_int\_upper.Rdata")  
# save(int\_upper, file = "homework3\_int\_lower.Rdata")  
# save(var\_lower, file = "homework3\_var\_uper.Rdata")  
# save(var\_upper, file = "homework3\_var\_loer.Rdata")  
# save(int\_cover, file = "homework3\_int\_cover.Rdata")  
# save(var\_cover, file = "homework3\_var\_cover.Rdata")  
boxplot(int\_mse)



boxplot(var\_mse)



mean(int\_cover)

## [1] 0

mean(var\_cover)

## [1] 0.113

The mses for the two fs are plotted, which I have no idea what is the reason behind it. We did no center no scale on the error terms, which will not be a problem if the assumption of normal distribution holds. the predicts are build on the level of random effect as factor[1]. I am not sure whether that is the correct way for population level mse and coverage calculation.

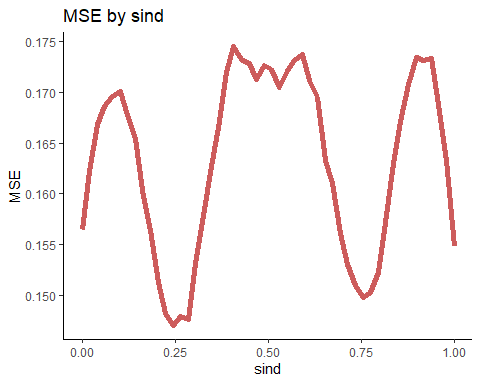
There is probably something going on the coverage for the f0 is 0, probably due to the shift or rescale on the smooth function. I never full catch the idea of identifiability problem, and you just told me to ignore the too complicated part. Sometime there is identifiability problem, sometimes not. for f1, the coverage is pretty bad too.

for approximation the 2 times of se interval is used to avoid the normality assumption, but it did not validate the results that much.

Sorry I just cannot get what do you want from this question. it does not make any sense based on what I get, I do not see what is the big picture here, and I am not sure what should I summarize.

## 2b

# mse <- rep(NA, 100)  
# mse\_sind <- matrix(NA, nrow = 50, ncol = 100)  
#   
# for (i in 1:100){  
# predictions <- predict(models[[i]], df[[i]], type = "value")  
# prob <- plogis(predictions)  
#   
# df[[i]]$sq\_diff <- (df[[i]]$y - prob)^2  
# mse[i] <- mean(df[[i]]$sq\_diff)  
#   
# mse\_df <- df[[i]] %>%  
# group\_by(sind) %>%  
# summarise(mse\_sind = mean(sq\_diff))   
#   
# mse\_sind[, i] <- as.vector(mse\_df$mse\_sind)  
# }  
#   
# save(mse\_sind, file = "13\_homework\_question2\_mse\_sind.Rdata")  
# save(mse, file = "13\_homework\_question2\_mse.Rdata")  
  
load("13\_homework\_question2\_mse\_sind.Rdata")  
load("13\_homework\_question2\_mse.Rdata")  
  
mse\_sind <- mse\_sind %>%  
 as.data.frame() %>%   
 mutate(mse = rowMeans(.),  
 sind = sind)%>%  
 select(sind, mse, everything())  
  
  
mse\_sind %>%   
 ggplot(aes(x = sind, y = mse)) +  
 geom\_line(color = "indianred", size = 2) +  
 ggtitle("MSE by sind") +  
 ylab("MSE") +  
 theme\_classic()



1. the variability reaches to the peak around 0.1, 0.5, and 0.9;
2. the variability is lower at 0.25 and 0.75;
3. the start and beginning are the same, because using the cyclic cubic spline.