HW4_wgeither

Warren Geither

10/12/2020

Problem 2

```
set.seed(1256)
theta <- as.matrix(c(1,2),nrow=2)
X <- cbind(1,rep(1:10,10))
h <- X%*%theta+rnorm(100,0,0.2)

# solve for Y
Y <- X%*%solve((t(X)%*%X))%*%t(X)

# lm model
lm_theta0 <- lm(h~0+X)$coefficients[1]

lm_theta1 <- lm(h~0+X)$coefficients[2]

# implement
for(i in 1:100){
theta_0 <- theta[1,1] - 0.05*(1/i)*(h[i] - Y[i])

theta_1 <- theta[2,1] - 0.05*(1/i)*((h[i] - Y[i])%*%X[i])
}

# create dataframe with results
results <- data.frame(theta_0 = c(lm_theta0, theta_0)
                      , theta_1 = c(lm_theta1, theta_1))

# add rownames
rownames(results) <- c("lm", "loop")

# print table
knitr::kable(results)
```

Problem 3

Part a.

Part b.

Part c.

Problem 4

You just need to transpose and invert the X matrices like so

```
#beta <- X%*%solve((t(X)%*%X))%*%t(X)
```

Problem 5: Need for speed challenge

In this problem, we are looking to compute the following:

$$y = p + AB^{-1}(q - r) \tag{1}$$

Where A, B, p, q and r are formed by:

```
    set.seed(12456)

    G <- matrix(sample(c(0,0.5,1),size=16000,replace=T),ncol=10)
    R <- cor(G) # R: 10 * 10 correlation matrix of G
    C <- kronecker(R, diag(1600)) # C is a 16000 * 16000 block diagonal matrix
    id <- sample(1:16000,size=932,replace=F)
    q <- sample(c(0,0.5,1),size=15068,replace=T) # vector of length 15068
    A <- C[id, -id] # matrix of dimension 932 * 15068
    B <- C[-id, -id] # matrix of dimension 15068 * 15068
    p <- runif(932,0,1)
    r <- runif(15068,0,1)
    C<-NULL #save some memory space
```

Part a.)

It takes way too long to calculate Y

```
# get size of A & B
size_of_a <- object.size(A)
size_of_b <- object.size(B)

# print sizes
print(paste0("size of A: ", size_of_a))
```

```
## [1] "size of A: 112347224"
```

```
print(paste0("size of B: ", size_of_b))
```

```
## [1] "size of B: 1816357208"
```

```
# would not terminate so commenting out
#microbenchmark(y = p + (A%*%inv(B)) %*% (q - r))
```

Part b.

Take the inverse of B first, then find q - r, then multiple A by B's inverse, then multiple by q-r and finally add p

Part c.

Use ANY means (ANY package, ANY trick, etc) necessary to compute the above, fast. Wrap your code in "system.time({})", everything you do past assignment "C <- NULL".

Problem 3

a.)

```
# create function
prob_successes <- function(x){

  # sum where x is logically equivalent to 1 meaning success and divide by length of vector
  return <- sum(x == 1) / length(x)
  return(return)
}
```

b.)

```
set.seed(12345)
P4b_data <- matrix(rbinom(10, 1, prob = (31:40)/100), nrow = 10, ncol = 10, byrow = FALSE)
P4b_data
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    1    1    1    1    1    1    1    1     1
##  [2,]    1    1    1    1    1    1    1    1    1     1
##  [3,]    1    1    1    1    1    1    1    1    1     1
##  [4,]    1    1    1    1    1    1    1    1    1     1
##  [5,]    0    0    0    0    0    0    0    0    0     0
##  [6,]    0    0    0    0    0    0    0    0    0     0
##  [7,]    0    0    0    0    0    0    0    0    0     0
##  [8,]    0    0    0    0    0    0    0    0    0     0
##  [9,]    1    1    1    1    1    1    1    1    1     1
## [10,]    1    1    1    1    1    1    1    1    1     1
```

```
# prob of success by columns
apply(P4b_data, 2, prob_successes)
```

```
##  [1] 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6
```

```
# prob of success by rows
apply(P4b_data, 1, prob_successes)
```

```
##  [1] 1 1 1 1 0 0 0 0 1 1
```

c.) the column proportions are all 0.6. And the row propotions are either 1 or 0. Which tells us that some of the row have no successes.

d.)

```
# create function for coin flips
coin_flips <- function(x){
  # input prob into rbinom and create 10 column vector
  vector <- matrix(rbinom(10, 1, prob = x), nrow = 1, ncol = 10, byrow = FALSE)
  return(vector)
}

# get probablities ready
prob_matrix <- c(rep(0.6,10))

# apply function to our prob matrix
matrix_we_want <- sapply(prob_matrix, coin_flips)

# prob of success by columns
apply(matrix_we_want, 2, prob_successes)
```

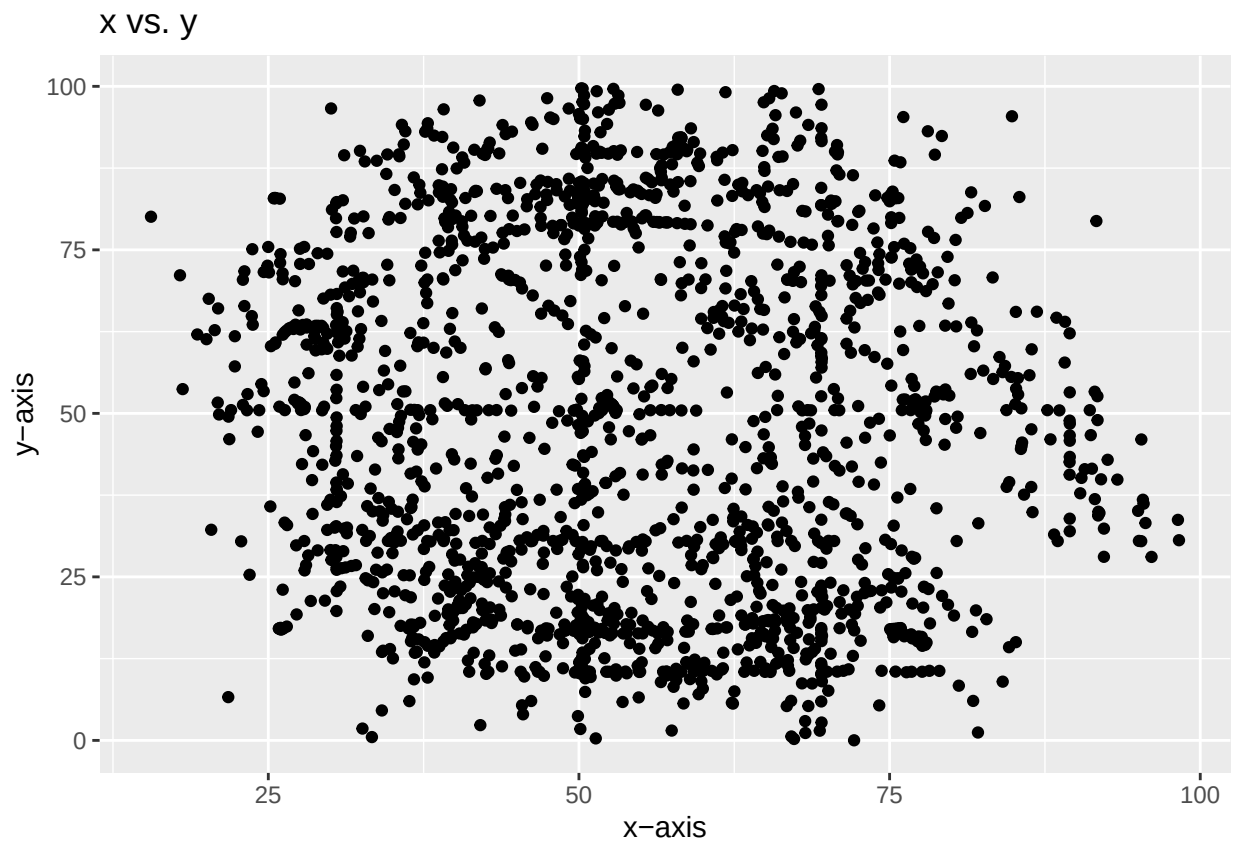
```
##  [1] 0.8 0.6 0.5 0.6 0.6 0.3 0.6 0.7 0.4 0.4
```

```
# prob of success by rows
apply(matrix_we_want, 1, prob_successes)
```

```
##  [1] 0.3 0.7 0.5 0.4 0.6 0.8 0.2 0.6 0.7 0.7
```

Problem 4

```
# load in data
hw3_data_df <- readRDS("HW3_data (2).rds")

# set column names
colnames(hw3_data_df) <- c("Observer", "x", "y")

plot_function <- function(df,title="title", x_lab="x-axis", y_lab="y-axis"){
  # create scatterplot
  plot <- ggplot(df, aes(x = x, y = y)) +
            geom_point() +
            ggtitle(title) +
            xlab(x_lab) +
            ylab(y_lab)
  return(plot)

}

# plot for whole dataframe
plot_function(hw3_data_df, "x vs. y", "x-axis", "y-axis")
```




```
# plot each one by observer
#gapply(hw3_data_df, "Observer", plot_function)

# did not get apply function to work
#ggplot(hw3_data_df, aes(x=x,y=y)) + geom_point() + facet_wrap(Observer~.)
```

Problem 5

```
# download files
download("http://www.farinspace.com/wp-content/uploads/us_cities_and_states.zip"
         ,dest="us_cities_states.zip")
unzip("us_cities_states.zip", exdir=".")

#read in data, looks like sql dump, blah
states <- fread(input = "us_cities_and_states/states.sql"
                ,skip = 23
                ,sep = "'"
                , sep2 = ","
                , header = F
                , select = c(2,4))

# adjust everything to lower so we can do analysis later
states$V2 <- tolower(states$V2)
```

```
# read in cities
cities <- fread(input = "us_cities_and_states/cities.sql"
                ,skip = 23
                ,sep = "'"
                , sep2 = ","
                , header = F
                , select = c(2,4))

# get first 50 cities
cities <- cities[1:50,]
```

Part b.) Create a summary table of the number of cities included by state.

```
# number of cities by state
knitr::kable(head(dplyr::count(cities, V4)))
```

V4	n
AL	2
CA	1
CT	1
GA	1
ID	1
IL	1

Part c.)

```
# create function to count occurances of letter
letter_counter <- function(letter, state_name){
  # use regex to find the matching letter in state name
  lengths(regmatches(state_name, gregexpr(letter, state_name)))
}
```

Create a for loop to loop through the state names imported in part a. Inside the for loop, use an apply family function to iterate across a vector of letters and collect the occurance count as a vector.

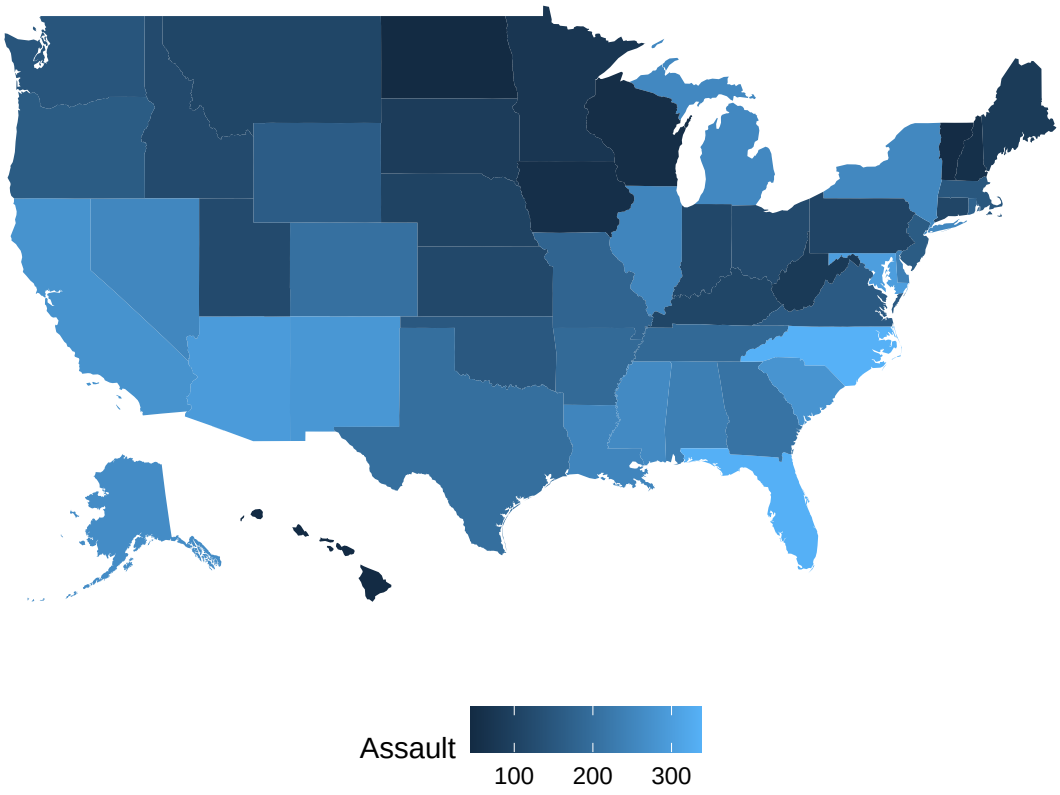
```
letter_count <- data.frame(matrix(NA, nrow=50, ncol=51))

# loop to count occurances of letters in state names
for(i in c("a")){
 vector <- letter_counter(i,states$V2)
}

# print vector of counts
vector
```

```
##  [1] 3 4 3 2 2 1 0 1 2 1 1 2 1 1 0 2 2 0 2 2 2 1 1 1 0 0 2 2 2 2 1 0 0 2 0 0 2 0
## [39] 2 1 2 2 0 1 1 1 0 1 0 1 0
```

```
# grab states with 3 or more "a"s
a_states <- states[1:3,]
```

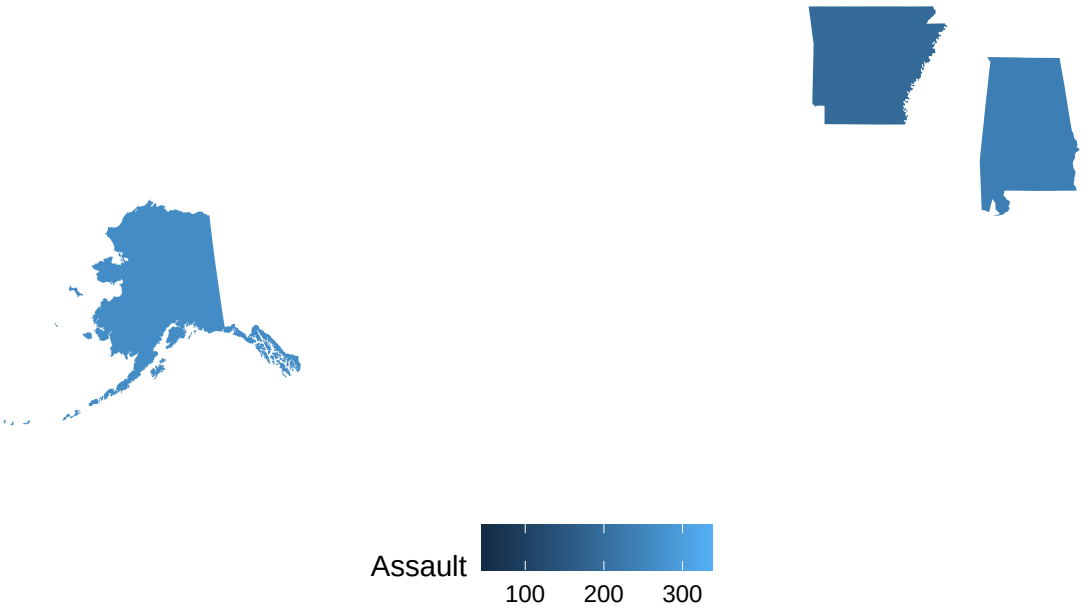
Part d.)

```
data("fifty_states") # this line is optional due to lazy data loading
crimes <- data.frame(state = tolower(rownames(USArrests)), USArrests)

# map_id creates the aesthetic mapping to the state name column in your data
p <- ggplot(crimes, aes(map_id = state)) +
  # map points to the fifty_states shape data
  geom_map(aes(fill = Assault), map = fifty_states) +
  expand_limits(x = fifty_states$long, y = fifty_states$lat) +
  coord_map() +
  scale_x_continuous(breaks = NULL) +
  scale_y_continuous(breaks = NULL) +
  labs(x = "", y = "") +
  theme(legend.position = "bottom",
        panel.background = element_blank())
p
```




```
# grab a states found in vector
a_states_data <- fifty_states %>% filter(fifty_states$id == "alaska"
                                        | fifty_states$id == "alabama"
                                        | fifty_states$id == "arkansas")
```

```
# plot only a states
p2 <- ggplot(crimes, aes(map_id = state)) +
  # map points to the fifty_states shape data
  geom_map(aes(fill = Assault), map = a_states_data) +
  expand_limits(x = a_states_data$long, y = a_states_data$lat) +
  coord_map() +
  scale_x_continuous(breaks = NULL) +
  scale_y_continuous(breaks = NULL) +
  labs(x = "", y = "") +
  theme(legend.position = "bottom",
        panel.background = element_blank())

p2
```



Problem 2

Part a.

He forgot to name the columns in df08.

```
library(quantreg)
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve
```

```
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## The following objects are masked from 'package:data.table':
##
##     first, last

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
#1)fetch data from Yahoo
#AAPL prices
apple08 <- getSymbols('AAPL', auto.assign = FALSE, from = '2008-1-1', to =
"2008-12-31")[,6]
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
#market proxy
rm08<-getSymbols('^ixic', auto.assign = FALSE, from = '2008-1-1', to =
"2008-12-31")[,6]

#log returns of AAPL and market
logapple08<- na.omit(ROC(apple08)*100)
logrm08<-na.omit(ROC(rm08)*100)

#OLS for beta estimation
beta_AAPL_08<-summary(lm(logapple08~logrm08))$coefficients[2,1]

#create df from AAPL returns and market returns
df08<-cbind(logapple08,logrm08)
# added this line
colnames(df08) <- c("logapple08", "logrm08")

set.seed(666)
Boot=1000
sd.boot=rep(0,Boot)
for(i in 1:Boot){
# nonparametric bootstrap
bootdata=df08[sample(nrow(df08), size = 251, replace = TRUE),]
sd.boot[i]= coef(summary(lm(logapple08~logrm08, data = bootdata)))[2,1]
}
```

Part b.

Bootstrap the analysis to get the parameter estimates using 100 bootstrapped samples. Make sure to use system.time to get total time for the analysis. You should probably make sure the samples are balanced across operators, ie each sample draws for each operator.

```
url <- "http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat"

#sensory_data <- fread(url, fill=TRUE, header=TRUE)
#saveRDS(sensory_data, "sensory_data_raw.RDS")
sensory_df <- readRDS("sensory_data_raw.RDS")

# remove first row
tidyverse_sensory_df <- slice(sensory_df[2:31])

# Remove column full of na's
tidyverse_sensory_df <- select(tidyverse_sensory_df, -Operator)

# seperate values
tidyverse_sensory_df <- tidyverse_sensory_df %>%
                        separate(col=V1
                                 ,into=c("Item", "1", "2", "3", "4", "5")
                                 , sep=" "
                                 , fill="left")

# remove first column since its missing values
tidyverse_sensory_df <- select(tidyverse_sensory_df, -Item)
```

```r
# add item column
tidyverse_sensory_df <- tidyverse_sensory_df %>%
                                add_column("Item" = c(1,1,1,2,2,2,3,3,3,4,4,4
                                                      ,5,5,5,6,6,6,7,7,7,8,8,8
                                                      ,9,9,9,10,10,10)
                                           , .before = "1")

# gather all the year values into one column
tidy_tidyverse_sensory_df <- tidyverse_sensory_df %>% gather(key="Operator"
                                                             , value="Value", "1","2"
                                                             , "3", "4", "5")

# show tidy data
knitr::kable(tidy_tidyverse_sensory_df[1:5,])
```

Item	Operator	Value
1	1	4.3
1	1	4.3
1	1	4.1
2	1	6.0
2	1	4.9

```r
# get rid of item value
test_df <- tidy_tidyverse_sensory_df[,2:3]

set.seed(54161)
Boot <- 1000
sd.boot <- rep(0,Boot)

time_results1 <- system.time(
    for(i in 1:Boot){
    # nonparametric bootstrap
    bootdata=test_df[sample(nrow(test_df), size = 150, replace = TRUE),]
    sd.boot[i]= coef(summary(lm(Operator~Value, data = bootdata)))[2,1]
    })

# store results
result_set1 <- t(data.matrix(time_results1))
rownames(result_set1) <- c("org")
```

Part c.

This can be done because each sample in the bootstrap is independent of the each other. The code runs significantly faster.

```r
# set cores
cores <- detectCores() - 1

# create cluster
c1 <- makeCluster(cores)
```

```r
# register cluster
registerDoParallel(c1)

# run in parallel
time_results2 <- system.time(
    foreach(i=1:Boot, combine = 'c') %dopar% {
    # nonparametric bootstrap
    bootdata=test_df[sample(nrow(test_df), size = 150, replace = TRUE),]
    sd.boot[i]= coef(summary(lm(Operator~Value, data = bootdata)))[2,1]
    })

# stopping cluster
stopCluster(c1)

# store results in a data frame and rename row
result_set2 <- t(data.matrix(time_results2))
rownames(result_set2) <- c("bootstrap")

# combine results
final_results <- rbind(result_set1, result_set2)

# print table
knitr::kable(final_results)
```

	user.self	sys.self	elapsed	user.child	sys.child
org	3.14	0.04	3.22	NA	NA
bootstrap	0.03	0.00	0.13	NA	NA

Problem 3

Part a.

```r
# create function for newtons method
newtons_method_func <- function(x){

  # input the equation
  eq <- (3^x) - sin(x) + cos(5*x)

  # store the derivative
  derivative_eq <- -5*sin(5*x)-cos(x)+log(3)*3^x

  # calculate the next term
  x_n1 <- x - (eq/derivative_eq)

  return(x_n1)
}

# create vector of values
long_vector <- seq(from = -500, to = 500, by=1)
```

```r
# apply function to vector
system.time(sapply(long_vector, newtons_method_func))
```

```
##    user  system elapsed
##    0.01    0.00    0.02
```

Part b.

Parallel is much faster.

```r
# set cores
cores <- detectCores() - 1

# create cluster
c1 <- makeCluster(cores)

# use Sapply
system.time(parSapply(c1,long_vector, newtons_method_func))
```

```
##    user  system elapsed
##    0.00    0.02    0.02
```

```r
# stopping cluster
stopCluster(c1)
```

Finish this homework by pushing your changes to your repo.

Only submit the .Rmd and .pdf solution files. Names should be formatted HW4_pid.Rmd and HW4_pid.pdf