STATS 506 HW 6

Calder Moore

Problem 1 - Rcpp

```
library(Rcpp)

cppFunction("
            double C_moment(NumericVector v, int k) {
              double sum = 0;
              for (int i = 0; i < v.length(); ++i){
                sum += v[i];
              }

              double mu = 0.0;
              mu = (sum/v.length());

              double acc = 0.0;
              for (int i = 0; i < v.length(); i++) {
                acc += pow(v[i] - mu, k);
              }

              return acc / v.length();

            }")

C_moment(v = c(1,0,2,3,4.5,6.7,9,1.2), k = 3)
```

```
[1] 17.65247
```

```
e1071::moment(x = c(1,0,2,3,4.5,6.7,9,1.2), order = 3, center = TRUE)
```

```
[1] 17.65247
```

They match when the `center` argument in `e1071::moment` is set to `TRUE`. They won't match otherwise, since the `C_moment` function is calculating moments relative to the mean (`mu` in the function), so we need the other function to also compute centered moments.

##Problem 2 - Expanding on waldCI

a)

Read in waldCI using `source()`.

```
source("C:/Users/moore/Desktop/School/Master/STATS 506/STATS-506-HW-5/hw5_code.R")
```

```
Creating a new generic function for 'mean' in the global environment

Warning in transformCI(ci1, sqrt): Only monotonic transformations are sensible.

Warning in transformCI(ci2, log): Only monotonic transformations are sensible.

Warning: package 'plotly' was built under R version 4.5.2

Loading required package: ggplot2


Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout


Attaching package: 'dplyr'
```

```
The following object is masked _by_ '.GlobalEnv':

    contains

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(parallel)

setClass(
  "bootstrapWaldCI",
  contains = "waldCI",
  slots = c(
    data = "ANY",
    reps = "numeric",
    fun = "function",
    compute = "character",
    boot_vals = "numeric"
  )
)

#function
boot <- function(fun, data, reps, compute){

  n <- nrow(data)

  one_boot <- function(i){
    idx <- sample(seq_len(n), n, replace = TRUE)
    fun(data[idx, , drop = FALSE])
  }
  #no parallel
  if (compute == "serial") {

    replicate(reps, one_boot(1))

  }
  #parallel
```

```
  else {

    cores <- max(1, detectCores() - 1)
    cl <- makeCluster(cores)

    clusterExport(cl,
      varlist = c("data","fun","n"),
      envir = environment()
    )

    clusterEvalQ(cl, {
      one_boot <- function(i){
        idx <- sample(seq_len(n), n, replace = TRUE)
        fun(data[idx, , drop = FALSE])
      }
    })

    out <- parSapply(cl, 1:reps, one_boot)

    stopCluster(cl)
    out
  }
}



#constructor
makeBootstrapCI <- function(fun, data, reps = 1000, level = 0.95,
                            compute = c("serial","parallel")) {

  compute <- match.arg(compute)

  boot_vals <- boot(fun, data, reps, compute)

  est <- mean(boot_vals)
  se <- sd(boot_vals)

  alpha <- 1 - level
  z <- qnorm(1 - alpha/2)

  lb <- est - z*se
  ub <- est + z*se
```

```
  new("bootstrapWaldCI",
      level = level,
      mean = est,
      sterr = se,
      lb = lb,
      ub = ub,
      data = data,
      reps = reps,
      fun = fun,
      compute = compute,
      boot_vals = boot_vals)
}


#show method
setMethod("show", "bootstrapWaldCI", function(object){
  cat(paste0("(", object@lb, ", ", object@ub, ")"), "\n")
})


#accessors
setGeneric("rebootstrap", function(object) standardGeneric("rebootstrap"))
```

```
[1] "rebootstrap"
```

```
setMethod("rebootstrap", "bootstrapWaldCI", function(object){

  boot_vals <- boot(object@fun, object@data, object@reps, object@compute)

  est <- mean(boot_vals)
  se <- sd(boot_vals)

  alpha <- 1 - object@level
  z <- qnorm(1 - alpha/2)

  object@mean <- est
  object@sterr <- se
  object@lb <- est - z*se
  object@ub <- est + z*se
  object@boot_vals <- boot_vals
```

```
  validObject(object)
  return(object)
})
```

b)

```
ci1 <- makeBootstrapCI(function(x) mean(x$y),
                       ggplot2::diamonds,
                       reps = 1000)

ci1
```

```
(5.72496658671744, 5.74403039808049)
```

```
rebootstrap(ci1)
```

```
(5.7246496049563, 5.74417872207374)
```

```
library(microbenchmark)

ci12 <- makeBootstrapCI(function(x) mean(x$y),
                       ggplot2::diamonds,
                       reps = 1000,
                       compute = "serial")

ci13 <- makeBootstrapCI(function(x) mean(x$y),
                       ggplot2::diamonds,
                       reps = 1000,
                       compute = "parallel")

microbenchmark(ci12)
```

```
Warning in microbenchmark(ci12): Could not measure a positive execution time
for 43 evaluations.

Unit: nanoseconds
 expr min lq mean median uq  max neval
 ci12   0  0   22      0  0 2200   100
```

```
microbenchmark(ci13)
```

```
Unit: nanoseconds
 expr min lq mean median uq  max neval
 ci13   0  0   15      0  0 1500   100
```

`parallel` method appears to be faster.

c)

```
dispCoef <- function(dat) {
  mod <- lm(mpg ~ cyl + disp + wt, data = dat)
  return(coef(mod)[["disp"]])
}

ci2 <- makeBootstrapCI(dispCoef,
                       mtcars,
                       reps = 1000)
ci2
```

```
(-0.0103715422004717, 0.0250499177460242)
```

```
rebootstrap(ci2)
```

```
(-0.0119955391658191, 0.0263014520822257)
```

```
ci22 <- makeBootstrapCI(dispCoef,
                       mtcars,
                       reps = 1000,
                       compute = "serial")

ci23 <- makeBootstrapCI(dispCoef,
                       mtcars,
                       reps = 1000,
                       compute = "parallel")

microbenchmark(ci22)
```

```
Warning in microbenchmark(ci22): Could not measure a positive execution time
for 16 evaluations.
```

```
Unit: nanoseconds
 expr min lq mean median uq  max neval
 ci22   0  0   28      0  0 2800   100
```

```
microbenchmark(ci23)
```

```
Warning in microbenchmark(ci23): Could not measure a positive execution time
for 23 evaluations.
```

```
Unit: nanoseconds
 expr min lq mean median uq  max neval
 ci23   0  0   31      0  0 3100   100
```

`series` is the faster method here.

Problem 3 - Large Data

a)

```
source("https://dept.stat.lsa.umich.edu/~jerrick/courses/stat506_f25/ps06q3.R")
```

```
Data.frame `df` is 251.8 Mb
```

```
#rescale
for (i in unique(df$country)) {

  id <- (df$country == i)
  n <- sum(id)

  if(n > 1) {
    df$gpa_std[id] <- as.vector(scale(df$prior_gpa[id]))
    df$forum_posts_std[id] <- as.vector(scale(df$forum_posts[id]))
    df$quiz_attempts_std[id] <- as.vector(scale(df$quiz_attempts[id]))
  }
  else {
```

```
    df$GPA_std[id] <- 0
    df$forum_posts_std[id] <- 0
    df$quiz_attempts_std[id] <- 0
  }
}

#make models
library(lme4)
```

```
Loading required package: Matrix
```

```
countries <- unique(df$country)
models <- list()
runtimes <- numeric(length(countries))

for (i in seq_along(countries)) {
  data_i <- df[df$country == countries[i], ]

  #run time
  runtimes[i] <- system.time({
    models[[i]] <- glmer(completed_course ~ gpa_std + forum_posts_std + quiz_attempts_std +
      data = data_i,
      family = binomial
    )
  })["elapsed"]
}

# Display running times
runtimes
```

```
[1] 231.96 434.53 177.15 117.67   3.39   7.53
```

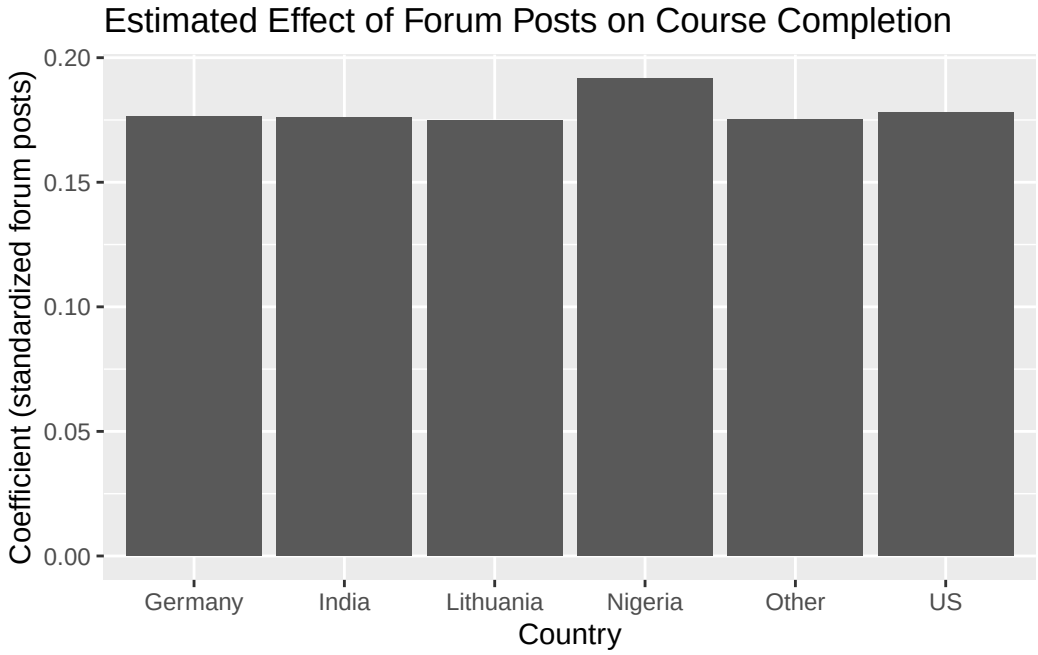
Visualization:

```
library(ggplot2)

coef_forum <- sapply(models, function(mod) {
  fixef(mod)["forum_posts_std"]
})
names(coef_forum) <- countries
```

```r
coef_df <- data.frame(
  country = names(coef_forum),
  forum_coef = coef_forum
)

ggplot(coef_df, aes(x = country, y = forum_coef)) +
  geom_col() +
  labs(
    title = "Estimated Effect of Forum Posts on Course Completion",
    y = "Coefficient (standardized forum posts)",
    x = "Country"
  )
```



b)

Use parallel processing to try and improve the run time. Part a) took forever.

```r
library(parallel)

countries <- unique(df$country)
```

```r
#model per country
fit_model <- function(c) {
  data_i <- df[df$country == c, ]
  glmer(completed_course ~ gpa_std + forum_posts_std + quiz_attempts_std + (1 | device_type)
        data = data_i, family = binomial)
}

#run time
time_total <- system.time({


  cl <- makeCluster(detectCores() - 1)
  clusterExport(cl, c("df", "fit_model"))
  clusterEvalQ(cl, library(lme4))

  models <- parLapply(cl, countries, fit_model)
  stopCluster(cl)

  names(models) <- countries

  coef_forum <- sapply(models, function(mod) fixef(mod)["forum_posts_std"])
  names(coef_forum) <- countries
})

coef_df_par <- data.frame(
  country = names(coef_forum),
  forum_coef = as.numeric(coef_forum)
)

# Show results
time_total
```

```
   user  system elapsed
   5.33    4.47  516.72
```

```r
coef_df
```

```
            country forum_coef
US               US  0.1781708
Other         Other  0.1751446
India         India  0.1761594
```

```
Germany     Germany  0.1765869
Lithuania Lithuania  0.1747332
Nigeria     Nigeria  0.1916254
```

```
coef_df_par
```

```
    country forum_coef
1        US  0.1781708
2     Other  0.1751446
3     India  0.1761594
4   Germany  0.1765869
5 Lithuania  0.1747332
6   Nigeria  0.1916254
```

The results match. The total run time for this script was 518.76 seconds.

Problem 4 - data.table

a)

```
library(data.table)
```

```
Attaching package: 'data.table'

The following objects are masked from 'package:dplyr':

    between, first, last
```

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.1     v stringr   1.5.1
v lubridate 1.9.4     v tibble    3.3.0
v purrr     1.1.0     v tidyr     1.3.1
v readr     2.1.5
```

```
-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x data.table::between() masks dplyr::between()
x tidyr::expand()       masks Matrix::expand()
x dplyr::filter()       masks plotly::filter(), stats::filter()
x data.table::first()   masks dplyr::first()
x lubridate::hour()     masks data.table::hour()
x lubridate::isoweek()  masks data.table::isoweek()
x dplyr::lag()          masks stats::lag()
x data.table::last()    masks dplyr::last()
x lubridate::mday()     masks data.table::mday()
x lubridate::minute()   masks data.table::minute()
x lubridate::month()    masks data.table::month()
x tidyr::pack()         masks Matrix::pack()
x lubridate::quarter()  masks data.table::quarter()
x lubridate::second()   masks data.table::second()
x purrr::transpose()    masks data.table::transpose()
x tidyr::unpack()       masks Matrix::unpack()
x lubridate::wday()     masks data.table::wday()
x lubridate::week()     masks data.table::week()
x lubridate::yday()     masks data.table::yday()
x lubridate::year()     masks data.table::year()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
tennis <- fread("atp_matches_2019.txt")

#deal with the Davis Cup issue
tennis$tourney_name_davis <- ifelse(grepl("Davis Cup", tennis$tourney_name), "Davis Cup", ten


tennis_2019 <- tennis[substr(tourney_date, 1, 4) == 2019,]
num_tournaments_2019 <- tennis_2019[, uniqueN(tourney_name_davis)]
num_tournaments_2019
```

```
[1] 66
```

I seem to be missing three tournaments compared to the solution.

b)

```
tourney_wins <- tennis[, .(tournaments_won = uniqueN(tourney_name)), by = winner_name]

multi_winners <- tourney_wins[tournaments_won > 1]
multi_winners <- multi_winners[order(-tournaments_won)]
multi_winners
```

```
              winner_name tournaments_won
                   <char>           <int>
  1:     Diego Schwartzman              21
  2: Roberto Bautista Agut              20
  3:         Andrey Rublev              20
  4:    Jan Lennard Struff              20
  5:    Stefanos Tsitsipas              20
 ---
151:       Kamil Majchrzak               2
152:    Thanasi Kokkinakis               2
153:        Norbert Gombos               2
154:            Tommy Paul               2
155:         Zhizhen Zhang               2
```

```
nrow(multi_winners)
```

```
[1] 155
```

```
max_wins <- max(multi_winners$tournaments_won)
multi_winners[tournaments_won == max_wins]
```

```
         winner_name tournaments_won
              <char>           <int>
1: Diego Schwartzman              21
```

155 players have won more than one tournament and Diego Schwartzman has won the most at 21.

c)

```r
aces <- tennis[!is.na(w_ace) & !is.na(l_ace)]

aces_long <- melt(aces,
                  measure.vars = c("w_ace", "l_ace"),
                  variable.name = "player_type",
                  value.name = "aces") %>%
  mutate(player_type = if_else(player_type == "w_ace", "winner", "loser"))

aces_long %>%
  group_by(player_type) %>%
  summarise(mean_aces = mean(aces), .groups = "drop")
```

```
# A tibble: 2 x 2
  player_type mean_aces
  <chr>           <dbl>
1 loser            5.79
2 winner           7.50
```

```r
t.test(aces$w_ace, aces$l_ace, paired = TRUE)
```

```
    Paired t-test

data:  aces$w_ace and aces$l_ace
t = 12.373, df = 2693, p-value < 2.2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 1.434712 1.975088
sample estimates:
mean difference
         1.7049
```

There is evidence of a difference, the p-value from the t-test is very small.

d)

```r
players <- rbind(
  tennis[, .(player = winner_name, result = 1)],
  tennis[, .(player = loser_name, result = 0)]
)

players <- players[, .(matches = .N, wins = sum(result)), by = player]
players <- players[matches >= 5]
players[, win_rate := wins / matches]

players[players$win_rate == max(players$win_rate)]
```

```
         player matches  wins  win_rate
         <char>   <int> <num>     <num>
1: Rafael Nadal      69    60 0.8695652
```

Again we find Nadal, with an 86.96% win rate.