Homework 5 - Drew Kearny - Due November 3

Name: Drew Kearny  
netID: 231005935  
Collaborated with:

Your homework **must be submitted in Word or PDF format, created by calling “Knit Word” or “Knit PDF” from RStudio on your R Markdown document.**  
Submission in other formats may receive a grade of 0**. Your responses must be supported by both textual explanations and the code you generate to produce your result. Note that all R code used to produce your results must be shown in your knitted file.**  
Reminder\*\* do not include excessive output and use the head() function where appropriate. For reference, my HW 5 solution is 16 pages.

## Q1 Reading and cleaning data

1. Use read.table() to read into R the data sets found at <https://raw.githubusercontent.com/schafert/stat404-data/main/sprint.m.df.txt> and <https://raw.githubusercontent.com/schafert/stat404-data/main/sprint.w.df.txt>, and call the resulting data frames sprint.m.df and sprint.w.df, respectively. Make sure to use appropriate arguments in read.table(), you can check the lecture for what is needed. Verify that you end up with data frames of dimensions 1000 x 8. Display the first six rows of both data frames.

sprint.m.df = read.table(file = "https://raw.githubusercontent.com/schafert/stat404-data/main/sprint.m.df.txt", sep = "\t", header = TRUE, quote="")  
sprint.w.df = read.table(file = "https://raw.githubusercontent.com/schafert/stat404-data/main/sprint.w.df.txt", sep = "\t", header = TRUE, quote="")  
  
dim(sprint.m.df)

## [1] 1000 8

dim(sprint.w.df)

## [1] 1000 8

head(sprint.m.df)

## Rank Time Wind Name Country Birthdate City Date  
## 1 1 9.58 0.9 Usain Bolt JAM 21.08.86 Berlin 16.08.2009  
## 2 2 9.63 1.5 Usain Bolt JAM 21.08.86 London 05.08.2012  
## 3 3 9.69 0.0 Usain Bolt JAM 21.08.86 Beijing 16.08.2008  
## 4 3 9.69 2.0 Tyson Gay USA 09.08.82 Shanghai 20.09.2009  
## 5 3 9.69 -0.1 Yohan Blake JAM 26.12.89 Lausanne 23.08.2012  
## 6 6 9.71 0.9 Tyson Gay USA 09.08.82 Berlin 16.08.2009

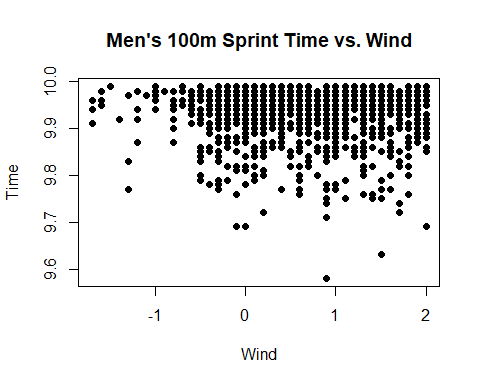
head(sprint.w.df)

## Rank Time Wind Name Country Birthdate City  
## 1 1 10.49 0.0 Florence Griffith-Joyner USA 21.12.59 Indianapolis  
## 2 2 10.54 0.9 Elaine Thompson-Herah JAM 28.06.92 Eugene  
## 3 3 10.60 1.7 Shelly-Ann Fraser-Pryce JAM 27.12.86 Lausanne  
## 4 4 10.61 1.2 Florence Griffith-Joyner USA 21.12.59 Indianapolis  
## 5 4 10.61 -0.6 Elaine Thompson-Herah JAM 28.06.92 Tokyo  
## 6 6 10.62 1.0 Florence Griffith-Joyner USA 21.12.59 Seoul  
## Date  
## 1 16.07.1988  
## 2 21.08.2021  
## 3 26.08.2021  
## 4 17.07.1988  
## 5 31.07.2021  
## 6 24.09.1988

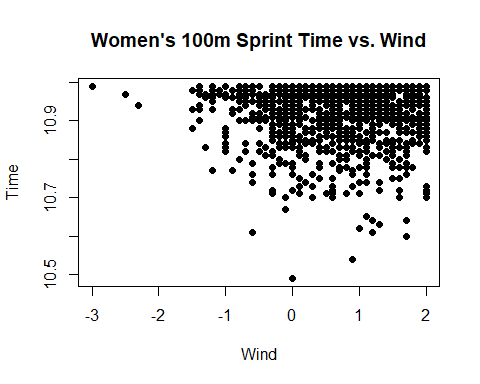
1. For each of the men’s and women’s data frames, plot the the 100m sprint time versus the wind measurements in Wind variable, setting the pch appropriately so that the points are solid small black dots. Label the axes and title the plot appropriately. Do you notice a trend—does more wind assistance mean faster sprint times? Where do the fastest men’s time, and the fastest women’s time, lie among this trend? (Remark: there’s an interesting story behind the wind measurement that was recorded for the fastest women’s time, you might enjoy reading about it online …)

There seems to be a slight trend in the data in that when there is more wind assistance the times for the 100m sprint seem to be lower (faster) on average for both men and women. The fastest men’s time came with a wind assistance of just under 1 and the fastest women’s time was when the wind was at 0. In general, the fastest times for both all occur when the wind assistance is greater than or equal to 0.

plot(sprint.m.df$Wind, sprint.m.df$Time, pch = 16, col = "black", xlab = "Wind", ylab = "Time",   
 main = "Men's 100m Sprint Time vs. Wind")



plot(sprint.w.df$Wind, sprint.w.df$Time, pch = 16, col = "black", xlab = "Wind", ylab = "Time",  
 main = "Women's 100m Sprint Time vs. Wind")



1. Notice that the Birthdate and Date columns in both data frames sprint.m.df and sprint.w.df are currently strings that follow the format DAY.MONTH.YEAR. Write a function called date.to.numeric() that takes in a string from either the Birthdate or Date columns, and outputs a numeric of the form DAY + (MONTH)\*10^2 + (YEAR)\*10^4. For example, date.to.numeric("16.08.2009") should return the numeric 20090816. Then, use one of the apply functions to iteratively use date.to.numeric() on both the Birthdate and Date columns in both the sprint.m.df and sprint.w.df data frames, converting these columns to numerics as appropriate. Print out the first six lines of sprint.m.df and sprint.w.df afterwards. Note: the dates in Birthdate have only the last two numbers of the year, while Date has all four numbers of the year (e.g., 86 vs. 1986). Your code should handle this appropriately using conditional execution. Use the function description to determine the arguments.

# date.to.numeric: converts a date string of the format DAY.MONTH.YEAR to numeric DAY + (MONTH)\*10^2 + (YEAR)\*10^4  
# Inputs:  
# - x\_str: string of the format DAY.MONTH.YEAR  
# - x\_split: default "\\.": splitting string for x\_str  
# Output: numeric calculated as DAY + (MONTH)\*10^2 + (YEAR)\*10^4  
  
date.to.numeric <- function(x\_str, x\_split = "\\.") {  
 date\_split <- strsplit(x\_str, x\_split)[[1]]  
 day <- as.numeric(date\_split[1])  
 month <- as.numeric(date\_split[2])  
 year <- as.numeric(date\_split[3])  
   
 if (year < 100 && year > 20) {  
 year <- year + 1900  
 } else if (year <= 20){  
 year <- year + 2000  
 }  
   
 numeric\_date <- day + (month \* 10^2) + (year \* 10^4)  
 return(numeric\_date)  
}

sprint.m.df$Birthdate <- sapply(sprint.m.df$Birthdate, date.to.numeric)  
sprint.m.df$Date <- sapply(sprint.m.df$Date, date.to.numeric)  
sprint.w.df$Birthdate <- sapply(sprint.w.df$Birthdate, date.to.numeric)  
sprint.w.df$Date <- sapply(sprint.w.df$Date, date.to.numeric)  
  
head(sprint.m.df)

## Rank Time Wind Name Country Birthdate City Date  
## 1 1 9.58 0.9 Usain Bolt JAM 19860821 Berlin 20090816  
## 2 2 9.63 1.5 Usain Bolt JAM 19860821 London 20120805  
## 3 3 9.69 0.0 Usain Bolt JAM 19860821 Beijing 20080816  
## 4 3 9.69 2.0 Tyson Gay USA 19820809 Shanghai 20090920  
## 5 3 9.69 -0.1 Yohan Blake JAM 19891226 Lausanne 20120823  
## 6 6 9.71 0.9 Tyson Gay USA 19820809 Berlin 20090816

head(sprint.w.df)

## Rank Time Wind Name Country Birthdate City  
## 1 1 10.49 0.0 Florence Griffith-Joyner USA 19591221 Indianapolis  
## 2 2 10.54 0.9 Elaine Thompson-Herah JAM 19920628 Eugene  
## 3 3 10.60 1.7 Shelly-Ann Fraser-Pryce JAM 19861227 Lausanne  
## 4 4 10.61 1.2 Florence Griffith-Joyner USA 19591221 Indianapolis  
## 5 4 10.61 -0.6 Elaine Thompson-Herah JAM 19920628 Tokyo  
## 6 6 10.62 1.0 Florence Griffith-Joyner USA 19591221 Seoul  
## Date  
## 1 19880716  
## 2 20210821  
## 3 20210826  
## 4 19880717  
## 5 20210731  
## 6 19880924

1. Reorder both data frames sprint.m.df and sprint.w.df so that their rows are in increasing order of Date. Print out the first six lines of sprint.m.df and sprint.w.df afterwards.

sprint.m.df <- sprint.m.df[order(sprint.m.df$Date), ]  
  
sprint.w.df <- sprint.w.df[order(sprint.w.df$Date), ]  
  
head(sprint.m.df)

## Rank Time Wind Name Country Birthdate City  
## 514 514 9.95 0.3 Jim Hines USA 19460910 Ciudad de M&eacute;xico  
## 813 813 9.98 0.6 Silvio Leonard CUB 19550920 Guadalajara  
## 689 689 9.97 1.5 Carl Lewis USA 19610701 Modesto  
## 366 366 9.93 1.4 Calvin Smith USA 19610108 Air Force Academy  
## 690 689 9.97 1.6 Calvin Smith USA 19610108 Z&uuml;rich  
## 588 588 9.96 0.1 Mel Lattany USA 19590810 Athens GA  
## Date  
## 514 19681014  
## 813 19770811  
## 689 19830514  
## 366 19830703  
## 690 19830824  
## 588 19840505

head(sprint.w.df)

## Rank Time Wind Name Country Birthdate City Date  
## 272 272 10.88 2.0 Marlies G&ouml;hr GDR 19580321 Dresden 19770701  
## 535 535 10.94 1.4 Marlies G&ouml;hr GDR 19580321 Dresden 19780812  
## 741 741 10.97 0.8 Marlies G&ouml;hr GDR 19580321 Dresden 19790613  
## 742 741 10.97 0.9 Evelyn Ashford USA 19570415 Walnut 19790616  
## 824 824 10.98 0.9 Marlies G&ouml;hr GDR 19580321 Potsdam 19800510  
## 480 480 10.93 2.0 Marlies G&ouml;hr GDR 19580321 Dresden 19800524

1. Create a column in both sprint.m.df and sprint.w.df called City.Date, given by concatenating the entries in the City and Date columns, separated by “.”. For example, if the City is Tokyo and Date is 19641015, then City.Date should be Tokyo.19641015. Print out the first six lines of sprint.m.df and sprint.w.df afterwards.

sprint.m.df$City.Date <- paste(sprint.m.df$City, sprint.m.df$Date, sep = ".")  
  
sprint.w.df$City.Date <- paste(sprint.w.df$City, sprint.w.df$Date, sep = ".")  
  
head(sprint.m.df)

## Rank Time Wind Name Country Birthdate City  
## 514 514 9.95 0.3 Jim Hines USA 19460910 Ciudad de M&eacute;xico  
## 813 813 9.98 0.6 Silvio Leonard CUB 19550920 Guadalajara  
## 689 689 9.97 1.5 Carl Lewis USA 19610701 Modesto  
## 366 366 9.93 1.4 Calvin Smith USA 19610108 Air Force Academy  
## 690 689 9.97 1.6 Calvin Smith USA 19610108 Z&uuml;rich  
## 588 588 9.96 0.1 Mel Lattany USA 19590810 Athens GA  
## Date City.Date  
## 514 19681014 Ciudad de M&eacute;xico.19681014  
## 813 19770811 Guadalajara.19770811  
## 689 19830514 Modesto.19830514  
## 366 19830703 Air Force Academy.19830703  
## 690 19830824 Z&uuml;rich.19830824  
## 588 19840505 Athens GA.19840505

head(sprint.w.df)

## Rank Time Wind Name Country Birthdate City Date  
## 272 272 10.88 2.0 Marlies G&ouml;hr GDR 19580321 Dresden 19770701  
## 535 535 10.94 1.4 Marlies G&ouml;hr GDR 19580321 Dresden 19780812  
## 741 741 10.97 0.8 Marlies G&ouml;hr GDR 19580321 Dresden 19790613  
## 742 741 10.97 0.9 Evelyn Ashford USA 19570415 Walnut 19790616  
## 824 824 10.98 0.9 Marlies G&ouml;hr GDR 19580321 Potsdam 19800510  
## 480 480 10.93 2.0 Marlies G&ouml;hr GDR 19580321 Dresden 19800524  
## City.Date  
## 272 Dresden.19770701  
## 535 Dresden.19780812  
## 741 Dresden.19790613  
## 742 Walnut.19790616  
## 824 Potsdam.19800510  
## 480 Dresden.19800524

1. We now want to remove all duplicated sprints in each of sprint.m.df and sprint.w.df. Specifically, if multiple sprints (rows) in sprint.m.df occur on the same City.Date, we will only keep the fastest sprint and discard the rest. Do the same with sprint.w.df. Make sure at the end, all the rows in sprint.m.df and sprint.w.df are still sorted in order of Date, and if multiple sprints occur on the same date, then sort those sprints alphabetically by City. Your final sprint.m.df should have dimension 509 x 9, while sprint.w.df should be 514 x 9. Display the first six lines of sprint.m.df and sprint.w.df afterwards. Hint: write a function to do the cleaning; then apply this function to each of the two data frames. The split-apply-combine concept may be useful here.

# clean.data: Converts a data frame by ordering it by 'Date' and 'City' and then removing all duplicated entries and only keeping the fastest time on each date and removing all other data points.  
# Inputs:  
# - df: A data frame that contains a 'City' and 'Date' column  
# Output: cleaned data frame with duplicates removed  
# Define a function to clean the data  
  
clean.sprint.data <- function(df) {  
 df <- df[order(df$Date, df$City), ]  
   
 cleaned.df <- data.frame()  
   
 unique.city.dates <- unique(df$City.Date)  
   
 for (city.date in unique.city.dates) {  
 subset.df <- df[df$City.Date == city.date, ]  
   
  
 if (nrow(subset.df) > 1) {  
 fastest.sprint <- subset.df[which.min(subset.df$Time), ]  
 cleaned.df <- rbind(cleaned.df, fastest.sprint)  
 } else {  
   
 cleaned.df <- rbind(cleaned.df, subset.df)  
 }  
 }  
   
 return(cleaned.df)  
}  
  
sprint.m.df <- clean.sprint.data(sprint.m.df)  
sprint.w.df <- clean.sprint.data(sprint.w.df)  
  
sprint.m.df <- sprint.m.df[order(sprint.m.df$Date, sprint.m.df$City), ]  
sprint.w.df <- sprint.w.df[order(sprint.w.df$Date, sprint.w.df$City), ]  
  
dim(sprint.m.df)

## [1] 509 9

dim(sprint.w.df)

## [1] 514 9

head(sprint.m.df)

## Rank Time Wind Name Country Birthdate City  
## 514 514 9.95 0.3 Jim Hines USA 19460910 Ciudad de M&eacute;xico  
## 813 813 9.98 0.6 Silvio Leonard CUB 19550920 Guadalajara  
## 689 689 9.97 1.5 Carl Lewis USA 19610701 Modesto  
## 366 366 9.93 1.4 Calvin Smith USA 19610108 Air Force Academy  
## 690 689 9.97 1.6 Calvin Smith USA 19610108 Z&uuml;rich  
## 588 588 9.96 0.1 Mel Lattany USA 19590810 Athens GA  
## Date City.Date  
## 514 19681014 Ciudad de M&eacute;xico.19681014  
## 813 19770811 Guadalajara.19770811  
## 689 19830514 Modesto.19830514  
## 366 19830703 Air Force Academy.19830703  
## 690 19830824 Z&uuml;rich.19830824  
## 588 19840505 Athens GA.19840505

head(sprint.w.df)

## Rank Time Wind Name Country Birthdate City Date  
## 272 272 10.88 2.0 Marlies G&ouml;hr GDR 19580321 Dresden 19770701  
## 535 535 10.94 1.4 Marlies G&ouml;hr GDR 19580321 Dresden 19780812  
## 741 741 10.97 0.8 Marlies G&ouml;hr GDR 19580321 Dresden 19790613  
## 742 741 10.97 0.9 Evelyn Ashford USA 19570415 Walnut 19790616  
## 824 824 10.98 0.9 Marlies G&ouml;hr GDR 19580321 Potsdam 19800510  
## 480 480 10.93 2.0 Marlies G&ouml;hr GDR 19580321 Dresden 19800524  
## City.Date  
## 272 Dresden.19770701  
## 535 Dresden.19780812  
## 741 Dresden.19790613  
## 742 Walnut.19790616  
## 824 Potsdam.19800510  
## 480 Dresden.19800524

1. Verify that in both sprint.m.df and sprint.w.df, each of the values in the City.Date column appear exactly once (i.e., there are no duplicated values). No duplicates.

any(duplicated(sprint.m.df$City.Date))

## [1] FALSE

any(duplicated(sprint.w.df$City.Date))

## [1] FALSE

## Q2 Merging data

1. In preparation of merging sprint.m.df and sprint.w.df, we first want to find all the sprints that occur in the same race in both data frames. Specifically, remove all the rows in sprint.m.df that have a City.Date that does not occur in sprint.w.df. Likewise, remove all the rows in sprint.w.df that have a City.Date that does not occur in sprint.m.df. Then, remove the City and Date columns in both data frames. In the end, both sprint.m.df and sprint.w.df should have 150 rows and 7 columns. Print out the first six lines of sprint.m.df and sprint.w.df afterwards. Hint: you might find the %in% operator useful; try looking it its help file.

common.dates <- sprint.m.df$City.Date %in% sprint.w.df$City.Date  
  
sprint.m.df <- sprint.m.df[common.dates, ]  
  
common.dates <- sprint.w.df$City.Date %in% sprint.m.df$City.Date  
  
sprint.w.df <- sprint.w.df[common.dates, ]  
  
sprint.m.df <- sprint.m.df[, !(names(sprint.m.df) %in% c("City", "Date"))]  
sprint.w.df <- sprint.w.df[, !(names(sprint.w.df) %in% c("City", "Date"))]  
  
head(sprint.m.df)

## Rank Time Wind Name Country Birthdate City.Date  
## 932 930 9.99 0.9 Carl Lewis USA 19610701 Z&uuml;rich.19840822  
## 814 813 9.98 1.6 Carl Lewis USA 19610701 Modesto.19850511  
## 367 366 9.93 1.0 Carl Lewis USA 19610701 Roma.19870830  
## 317 317 9.92 1.1 Carl Lewis USA 19610701 Seoul.19880924  
## 436 436 9.94 0.8 Leroy Burrell USA 19670221 Houston.19890616  
## 232 232 9.90 1.9 Leroy Burrell USA 19670221 New York City.19910614

head(sprint.w.df)

## Rank Time Wind Name Country Birthdate  
## 49 49 10.76 1.7 Evelyn Ashford USA 19570415  
## 538 535 10.94 0.6 Merlene Ottey JAM 19600510  
## 334 330 10.90 -0.5 Silke M&ouml;ller GER 19640620  
## 6 6 10.62 1.0 Florence Griffith-Joyner USA 19591221  
## 385 375 10.91 1.6 Dawn Sowell USA 19660327  
## 544 535 10.94 1.0 Carlette Guidry USA 19680904  
## City.Date  
## 49 Z&uuml;rich.19840822  
## 538 Modesto.19850511  
## 334 Roma.19870830  
## 6 Seoul.19880924  
## 385 Houston.19890616  
## 544 New York City.19910614

1. We will now complete the manual merge of sprint.m.df and sprint.w.df. First, check the order of values in City.Date in sprint.m.df match exactly with those in sprint.w.df. Then, use cbind() to create a new data frame sprint.df that has 13 columns. The first column should be City.Date, the next 6 columns should contain all the remaining columns from sprint.m.df, and the last 6 columns should contain all the remaining columns form sprint.w.df. Of course, each row should correspond to sprints from the same City.Date. Print out the first six lines of sprint.df afterwards, and verify that its dimensions are 150 x 13.

identical(sprint.m.df$City.Date, sprint.w.df$City.Date)

## [1] TRUE

sprint.df <- cbind(sprint.m.df[, "City.Date", drop = FALSE], sprint.m.df[, -1, drop = FALSE], sprint.w.df[, -1, drop = FALSE])  
  
head(sprint.df)

## City.Date Time Wind Name Country Birthdate  
## 932 Z&uuml;rich.19840822 9.99 0.9 Carl Lewis USA 19610701  
## 814 Modesto.19850511 9.98 1.6 Carl Lewis USA 19610701  
## 367 Roma.19870830 9.93 1.0 Carl Lewis USA 19610701  
## 317 Seoul.19880924 9.92 1.1 Carl Lewis USA 19610701  
## 436 Houston.19890616 9.94 0.8 Leroy Burrell USA 19670221  
## 232 New York City.19910614 9.90 1.9 Leroy Burrell USA 19670221  
## City.Date Time Wind Name Country  
## 932 Z&uuml;rich.19840822 10.76 1.7 Evelyn Ashford USA  
## 814 Modesto.19850511 10.94 0.6 Merlene Ottey JAM  
## 367 Roma.19870830 10.90 -0.5 Silke M&ouml;ller GER  
## 317 Seoul.19880924 10.62 1.0 Florence Griffith-Joyner USA  
## 436 Houston.19890616 10.91 1.6 Dawn Sowell USA  
## 232 New York City.19910614 10.94 1.0 Carlette Guidry USA  
## Birthdate City.Date  
## 932 19570415 Z&uuml;rich.19840822  
## 814 19600510 Modesto.19850511  
## 367 19640620 Roma.19870830  
## 317 19591221 Seoul.19880924  
## 436 19660327 Houston.19890616  
## 232 19680904 New York City.19910614

dim(sprint.df)

## [1] 150 13

1. Use the merge() function to recreate the merge in the previous part. This should require only one line of code; call the result sprint.df.2. In the call to merge(), make sure to set the argument suffixes=c(".m",".w"), which will help appropriately distinguish the column names after merging (a convenience of using the merge() function). The merged data frame sprint.df2 should be of dimension 150 x 13; display its first six lines. Do these match those of sprint.df from the last part? They shouldn’t match, and this is because the merge() function sorts according to the by column, by default. Take a look at the help file for merge() to see what argument you should set in order to turn off this behavior; then check again the first six lines of the output sprint.df2, and compare to those from sprint.df. The first run sprin.df.2 did not match and the first City.Date was instead Ad-Dawhah.20120511. After setting sort = FALSE the two data frames did match.

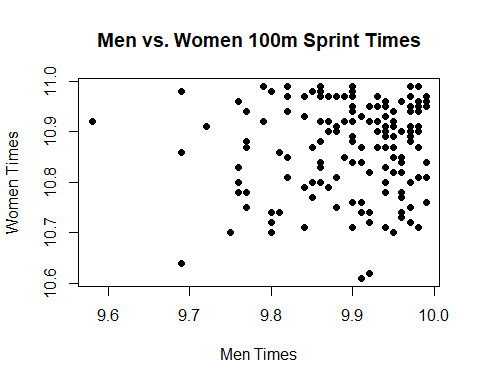
sprint.df.2 <- merge(sprint.m.df, sprint.w.df, by="City.Date", suffixes=c(".m", ".w"), sort = FALSE)  
  
head(sprint.df.2)

## City.Date Rank.m Time.m Wind.m Name.m Country.m  
## 1 Z&uuml;rich.19840822 930 9.99 0.9 Carl Lewis USA  
## 2 Modesto.19850511 813 9.98 1.6 Carl Lewis USA  
## 3 Roma.19870830 366 9.93 1.0 Carl Lewis USA  
## 4 Seoul.19880924 317 9.92 1.1 Carl Lewis USA  
## 5 Houston.19890616 436 9.94 0.8 Leroy Burrell USA  
## 6 New York City.19910614 232 9.90 1.9 Leroy Burrell USA  
## Birthdate.m Rank.w Time.w Wind.w Name.w Country.w  
## 1 19610701 49 10.76 1.7 Evelyn Ashford USA  
## 2 19610701 535 10.94 0.6 Merlene Ottey JAM  
## 3 19610701 330 10.90 -0.5 Silke M&ouml;ller GER  
## 4 19610701 6 10.62 1.0 Florence Griffith-Joyner USA  
## 5 19670221 375 10.91 1.6 Dawn Sowell USA  
## 6 19670221 535 10.94 1.0 Carlette Guidry USA  
## Birthdate.w  
## 1 19570415  
## 2 19600510  
## 3 19640620  
## 4 19591221  
## 5 19660327  
## 6 19680904

1. Plot the Time.w versus Time.m columns in sprint.df2, with appropriately labeled axes and an appropriate title. Looking at the the women’s versus men’s times from the common track meets—is there a positive correlation here, i.e., is there a “track meet effect”? This might suggest that there is something about the track meet itself (e.g., the weather, the atmosphere, the crowd, the specific way the track has been constructed/set up, etc.) that helps sprinters run faster.

There appear to be very small correlation between men’s times and women’s times based on the track meet they were at. There is somewhat of a positive pattern relationship but the correlation coefficient looks like it would only be about .1 so there appear to be a small track meet effect.

plot(x = sprint.df.2$Time.m, y = sprint.df.2$Time.w, pch = 16, col = "black",   
 xlab = "Men Times", ylab = "Women Times",  
 main = "Men vs. Women 100m Sprint Times")



## Q3 Prostate cancer data - Simple exploration and linear modeling

Recall the data set on 97 men who have prostate cancer (from the book [The Elements of Statistical Learning](http://statweb.stanford.edu/~tibs/ElemStatLearn/)). Reading it into our R session:

pros.df =   
 read.table("http://www.stat.cmu.edu/~ryantibs/statcomp/data/pros.dat")  
dim(pros.df)

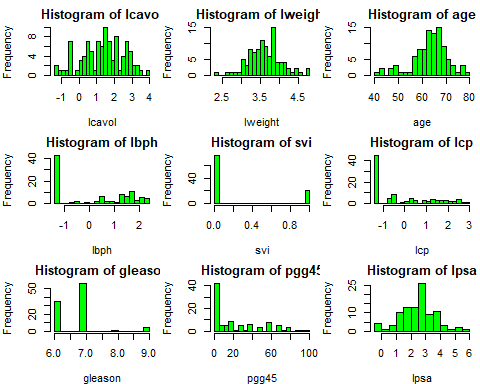
## [1] 97 9

head(pros.df, 3)

## lcavol lweight age lbph svi lcp gleason pgg45 lpsa  
## 1 -0.5798185 2.769459 50 -1.386294 0 -1.386294 6 0 -0.4307829  
## 2 -0.9942523 3.319626 58 -1.386294 0 -1.386294 6 0 -0.1625189  
## 3 -0.5108256 2.691243 74 -1.386294 0 -1.386294 7 20 -0.1625189

1. Define pros.df.subset to be the subset of observations (rows) of the prostate data set such the lcp measurement is greater than the minimum value (the minimum value happens to be log(0.25), but you should not hardcode this value and should work it out from the data). As in lecture, plot histograms of all of the variables in pros.df.subset. Comment on any differences you see between these distributions and the ones in lecture. I didn’t have any differences in the histograms when I looked at mine vs the ones that came from the code in the lecture.

min\_lcp <- min(pros.df$lcp)  
  
pros.df.subset <- subset(pros.df, subset = lcp > min\_lcp)  
  
par(mfrow = c(3, 3),  
 mar = c(4, 4, 2, 0.5))  
  
for(i in 1:ncol(pros.df)){  
 hist(pros.df[[i]], xlab = colnames(pros.df)[i],  
 main = paste("Histogram of", colnames(pros.df)[i]),  
 breaks = 20, col = "green")  
}



1. Also as in lecture, compute and display correlations between all pairs of variables in pros.df.subset. Report the two highest correlations between pairs of (distinct) variables, and also report the names of the associated variables. Are these different from answers that were computed on the full data set?

The highest correlation was .8049 between lcp and ‘lcavol’ and the second highest correlation was .6246 between ‘lcp’ and ‘svi’. Yes, the highest correlations on the subset of the data were different than the highest correlations that were computed on the full data set in lecture, because for the full data set the top correlation was between ‘pgg45’ and ‘gleason’.

pros.cor <- cor(pros.df.subset)  
pros.cor

## lcavol lweight age lbph svi lcp  
## lcavol 1.00000000 0.220935164 0.01342238 -0.24074053 0.56751340 0.80497276  
## lweight 0.22093516 1.000000000 0.39791553 0.33819930 0.14536906 0.11345014  
## age 0.01342238 0.397915527 1.00000000 0.31072815 0.08621023 -0.02542962  
## lbph -0.24074053 0.338199299 0.31072815 1.00000000 -0.23893338 -0.17943183  
## svi 0.56751340 0.145369055 0.08621023 -0.23893338 1.00000000 0.62463822  
## lcp 0.80497276 0.113450137 -0.02542962 -0.17943183 0.62463822 1.00000000  
## gleason 0.28515349 0.001325445 0.17297082 0.04101308 0.16980339 0.36706046  
## pgg45 0.31679928 0.057510270 0.26630159 0.06478598 0.32513404 0.42533870  
## lpsa 0.62417072 0.243466312 -0.05929751 -0.06325458 0.60088451 0.52967466  
## gleason pgg45 lpsa  
## lcavol 0.285153491 0.31679928 0.62417072  
## lweight 0.001325445 0.05751027 0.24346631  
## age 0.172970825 0.26630159 -0.05929751  
## lbph 0.041013079 0.06478598 -0.06325458  
## svi 0.169803395 0.32513404 0.60088451  
## lcp 0.367060460 0.42533870 0.52967466  
## gleason 1.000000000 0.61844087 0.15237932  
## pgg45 0.618440867 1.00000000 0.26751083  
## lpsa 0.152379320 0.26751083 1.00000000

pros.cor[lower.tri(pros.cor,diag=TRUE)] = 0   
pros.cor.sorted = sort(abs(pros.cor),decreasing=T)  
  
vars.big.cor <- which(abs(pros.cor) == pros.cor.sorted[1],  
 arr.ind = TRUE)  
vars.next.big.cor <- which(abs(pros.cor) == pros.cor.sorted[2],  
 arr.ind = TRUE)  
vars.big.cor

## row col  
## lcavol 1 6

colnames(pros.df)[vars.big.cor]

## [1] "lcavol" "lcp"

pros.cor.sorted[1]

## [1] 0.8049728

vars.next.big.cor

## row col  
## svi 5 6

colnames(pros.df)[vars.next.big.cor]

## [1] "svi" "lcp"

pros.cor.sorted[2]

## [1] 0.6246382

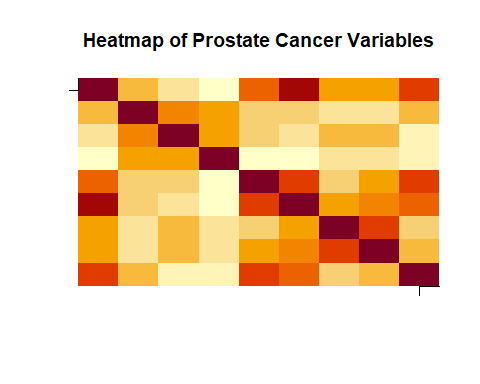
1. Produce a heatmap of the correlation matrix (which contains correlations of all pairs of variables) of pros.df.subset. For this heatmap, use the full matrix (not just its upper triangular part). Makes sure your heatmap is displayed in a sensible way. For full points, create your heatmap using base R graphics (hint: the clockwise90() function from the “Plotting tools” lecture will be handy); for partial points, use an R package. **Optional extra credit**: Using base R graphsics, label the axes with the variable names in the plot (hint: look back at split-apply-combine notes for example on using axis())

clockwise90 <- function(a) { t(a[nrow(a):1,]) }

pros.cor <- cor(pros.df.subset)  
col.names <- colnames(pros.cor)  
  
image(clockwise90(pros.cor),  
 main = "Heatmap of Prostate Cancer Variables",  
 axes = FALSE  
   
)  
  
axis(1, at = 1:ncol(pros.cor), labels = FALSE)  
mtext(variable\_names, side = 1, at = 1:ncol(pros.cor), line = 2, las = 1)

## Error in as.graphicsAnnot(text): object 'variable\_names' not found

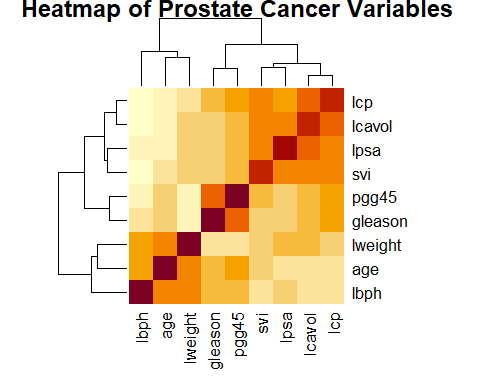
axis(2, at = 1:nrow(pros.cor), labels = FALSE)



mtext(variable\_names, side = 2, at = 1:nrow(pros.cor), line = 2, las = 1)

## Error in as.graphicsAnnot(text): object 'variable\_names' not found

heatmap(clockwise90(pros.cor),  
 main = "Heatmap of Prostate Cancer Variables"  
   
)



1. Compute, using lm(), a linear regression model of lpsa (log PSA score) on lcavol (log cancer volume). Do this twice: once with the full data set, pros.df, and once with the subsetted data, pros.df.subset. Save the results as pros.lm. and pros.subset.lm, respectively. Using coef(), display the coefficients (intercept and slope) from each linear regression. Are they different? YES they are slightly different as seen from the output as the subset regression line has a higher intercept and smaller slope.

pros.lm <- lm(lpsa ~ lcavol, data = pros.df)  
  
pros.subset.lm <- lm(lpsa ~ lcavol, data = pros.df.subset)  
  
coef(pros.lm)

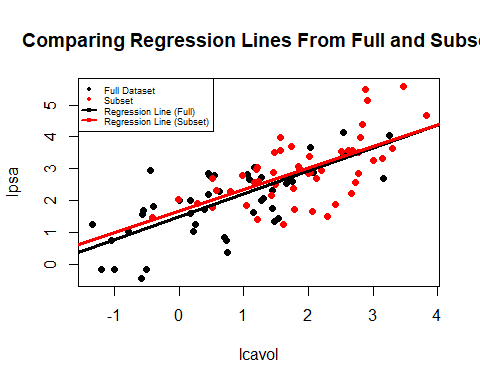
## (Intercept) lcavol   
## 1.5072975 0.7193204

coef(pros.subset.lm)

## (Intercept) lcavol   
## 1.6695707 0.6725807

1. Let’s produce a visualization to help us figure out how different these regression lines really are. Plot lpsa versus lcavol, using the full set of observations, in pros.df. Label the axes appropriately. Then, mark the observations in pros.df.subset by small filled red circles. Add a thick black line to your plot, displaying the fitted regression line from pros.lm. Add a thick red line, displaying the fitted regression line from pros.subset.lm. **Optional extra credit**: Add a legend that explains the color coding.

plot(pros.df$lcavol, pros.df$lpsa,   
 xlab = "lcavol", ylab = "lpsa",  
 main = "Comparing Regression Lines From Full and Subset", pch = 19)  
  
points(pros.df.subset$lcavol, pros.df.subset$lpsa, col = "red", pch = 19)  
  
abline(pros.lm, col = "black", lwd = 3)  
  
abline(pros.subset.lm, col = "red", lwd = 3)  
  
legend("topleft", legend = c("Full Dataset", "Subset", "Regression Line (Full)", "Regression Line (Subset)"),  
 col = c("black", "red", "black", "red"), lty = c(0, 0, 1, 1), lwd = c(1, 1, 2, 2), pch = c(19, 19), cex = .55)



1. Compute again a linear regression of lpsa on lcavol, but now on two different subsets of the data: the first consisting of patients with SVI, and the second consistent of patients without SVI. Display the resulting coefficients (intercept and slope) from each model, and produce a plot just like the one in the last question, to visualize the different regression lines on top of the data. Do these two regression lines differ, and in what way?

Yes these regression lines are different in that the regression line for data with patients who have SVI has a slope of about .65 while the regression line for patients with no SVI has a slope of about .58. Also, the regression line for patients with SVI has a higher intercept on the ‘lpsa’ (y) axis at about 2 while the line for patients without SVI has an intercept at about 1.5.

pros.svi.df <- subset(pros.df, svi == 1)  
pros.svi.lm <- lm(lpsa ~ lcavol, data = pros.svi.df)  
coef(pros.svi.lm)

## (Intercept) lcavol   
## 2.053552 0.651189

pros.no.svi.df <- subset(pros.df, svi == 0)  
pros.no.svi.lm <- lm(lpsa ~ lcavol, data = pros.no.svi.df)  
coef(pros.no.svi.lm)

## (Intercept) lcavol   
## 1.539704 0.586396

plot(pros.svi.df$lcavol, pros.svi.df$lpsa,   
 xlab = "lcavol", ylab = "lpsa",  
 main = "Comparing Regression Lines From SVI vs. No SVI", pch = 19)  
  
points(pros.no.svi.df$lcavol, pros.no.svi.df$lpsa, col = "red", pch = 19)  
  
abline(pros.svi.lm, col = "black", lwd = 3)  
  
abline(pros.no.svi.lm, col = "red", lwd = 3)  
  
legend("topleft", legend = c("SVI", "No SVI", "Regression Line (SVI)", "Regression Line (No SVI)"),  
 col = c("black", "red", "black", "red"), lty = c(0, 0, 1, 1), lwd = c(1, 1, 2, 2), pch = c(19, 19), cex = 0.55)

