Lab 8: Reading in, Reordering, and Merging Data

Name: Rufus Petrie

This week's agenda: reading in data, cleaning data, reordering data, merging data, restructuring data.

Reading and cleaning data

- **1a.** Use `read.table()` to read into R the data sets found at http://www.stat.cmu.edu/~ryantibs/statcomp/data/sprint.m.dat and http://www.stat.cmu.edu/~ryantibs/statcomp/data/sprint.w.dat, 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. Also, make usre to set the argument `stringsAsFactors` to be `TRUE`. Verify that you end up with data frames of dimensions 2988 x 8 (for the men's data) and 2018 x 8 (for the women's data). Display the first five rows of both data frames.

```
sprint.m.df <- read.table("http://www.stat.cmu.edu/~ryantibs/statcomp/data/sprint.m.dat",
                          sep="\t", header=TRUE, quote="", stringsAsFactors=TRUE)
sprint.w.df <- read.table("http://www.stat.cmu.edu/~ryantibs/statcomp/data/sprint.w.dat",
                          sep="\t", header=TRUE, quote="", stringsAsFactors=TRUE)
head(sprint.m.df, 5)
```

```
##   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
```

```
head(sprint.w.df, 5)
```

```
##   Rank  Time Wind                    Name Country Birthdate         City
## 1    1 10.49  0,0 Florence Griffith-Joyner     USA  21.12.59 Indianapolis
## 2    2 10.61 +1,2 Florence Griffith-Joyner     USA  21.12.59 Indianapolis
## 3    3 10.62 +1,0 Florence Griffith-Joyner     USA  21.12.59        Seoul
## 4    4 10.64 +1,2          Carmelita Jeter     USA  24.11.79     Shanghai
## 5    5 10.65 +1,1             Marion Jones     USA  12.10.75 Johannesburg
##         Date
## 1 16.07.1988
## 2 17.07.1988
## 3 24.09.1988
## 4 20.09.2009
## 5 12.09.1998
```

- **1b.** Since we set `stringsAsFactors=TRUE` in the previous part, the function `read.table()` treated the values in the `Wind` column as factors. (The values would've been strings otherwise). We want to convert these factors to numerics. Converting factors to numerics can be an annoyingly frustrating task in general, so it's good to practice it. These next two questions will guide you on how to do this.

We provide a test string `input.value` below, which is "4,8". Use functions you have seen in previous weeks on text processing to convert `input.value` to contain the numeric `4.8` instead. Display the converted value and check its class to ensure it is a numeric. Hint: there are multiple ways to do the conversion; perhaps the most familiar way will be to use `strsplit()` to separate `input.value` by the comma, and then use `paste()` function to concatenate the "4" with the "8", separated by ".", and then finally use `as.numeric()`.

```
input.value = "4,8"
input.value <- strsplit(input.value, ",")
input.value <- paste(input.value[[1]][1], ".", input.value[[1]][2], sep="")
input.value <- as.numeric(input.value)
input.value
```

```
## [1] 4.8
```

- **1c.** Now we will write a function to repeatedly apply the strategy from the last part to elements of a vector. Below is a vector `wind.measurements` of *factors* (note: not strings) for you to play around with, as a testing ground. Write a function `factor.to.numeric()` that takes in a vector of factors and outputs a vector of corresponding numerics. Verify that `factor.to.numeric(wind.measurements)` returns `c(-2.0, 0.0, 0.6, 1.7)` (or equivalent numbers, e.g., 2 instead of `2.0` is fine).

```
wind.measurements = as.factor(c("-2,0", "0,0", "0,6", "+1,7"))
factor.to.numeric <- function(x){
  x <- sub(",", ".", x)
  x <- as.character(x)
  x <- as.numeric(x)
  return(x)
}
factor.to.numeric(wind.measurements)
```

```
## [1] -2.0  0.0  0.6  1.7
```

- **1d.** Using `factor.to.numeric()`, convert the `Wind` column of `sprint.m.df` and `sprint.w.df` into numeric variables. However, you might get exactly one `NA` from this process in `sprint.w.df` (or get no `NA`s depending on you how wrote your function). If you do, what was the wind entry that failed to be converted into a numeric (hence becoming `NA`)? In words, can you describe why this `NA` occurred? (This will require you to reload the `sprint.w.df` from the beginning to see what certain values in the `Wind` column were before we used the `factor.to.numeric()`.) If needed, you should manually fix this `NA`. Then, display the first five rows of `sprint.m.df` and `sprint.w.df`.

```
sprint.w.df[sprint.w.df$Name=="Lyubov Perepelova",]
```

```
##      Rank  Time   Wind              Name Country Birthdate    City       Date
## 1226 1188 11.04 +2,0,0 Lyubov Perepelova     UZB  26.02.79 Bishkek 03.06.2000
```

```
sprint.m.df$Wind <- factor.to.numeric(sprint.m.df$Wind)
sprint.w.df$Wind <- factor.to.numeric(sprint.w.df$Wind)
```

```
## Warning in factor.to.numeric(sprint.w.df$Wind): NAs introduced by coercion
```

```
sprint.w.df[sprint.w.df$Name=="Lyubov Perepelova",]
```

```
##      Rank  Time Wind              Name Country Birthdate    City       Date
## 1226 1188 11.04   NA Lyubov Perepelova     UZB  26.02.79 Bishkek 03.06.2000
```

```
sprint.w.df[sprint.w.df$Name=="Lyubov Perepelova", "Wind"] <- 2.0
head(sprint.m.df, 5)
```

```
##   Rank Time Wind        Name Country Birthdate     City       Date
```

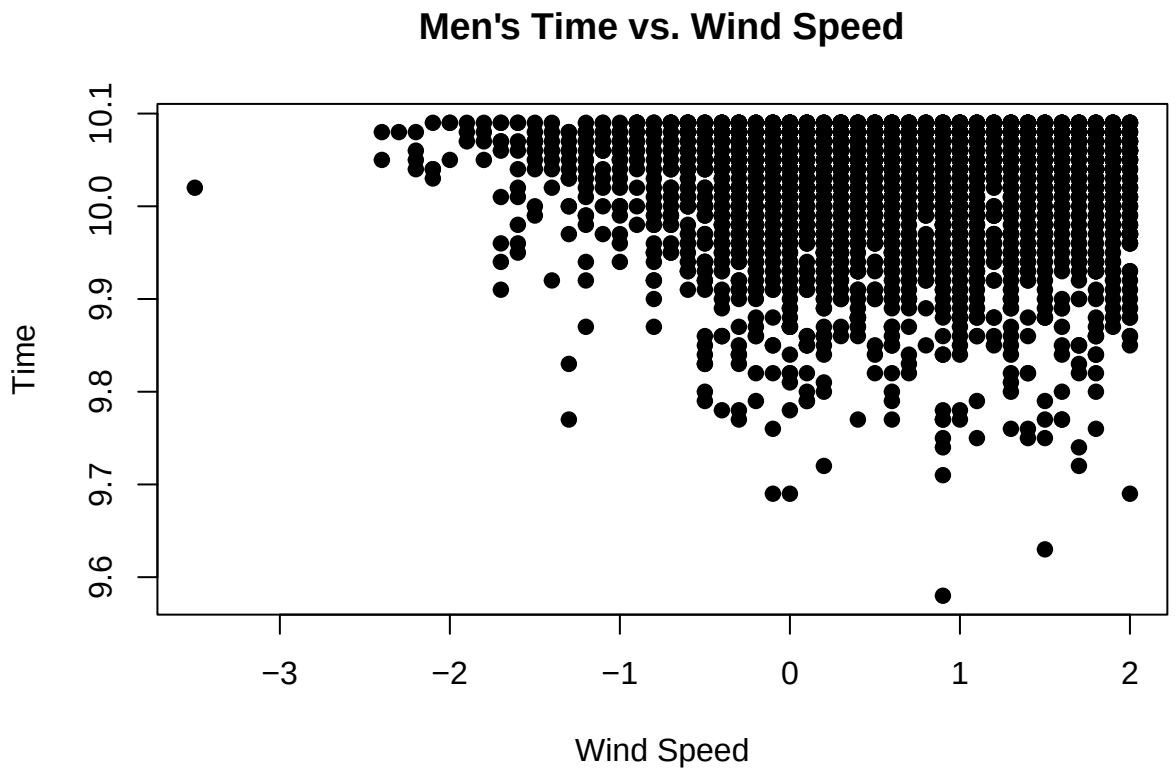
```
## 1    1 9.58  0.9  Usain Bolt     JAM  21.08.86   Berlin 16.08.2009
## 2    2 9.63  1.5  Usain Bolt     JAM  21.08.86   London 05.08.2012
## 3    3 9.69  0.0  Usain Bolt     JAM  21.08.86  Beijing 16.08.2008
## 4    3 9.69  2.0   Tyson Gay     USA  09.08.82 Shanghai 20.09.2009
## 5    3 9.69 -0.1 Yohan Blake     JAM  26.12.89 Lausanne 23.08.2012
```

```
head(sprint.m.df, 5)
```

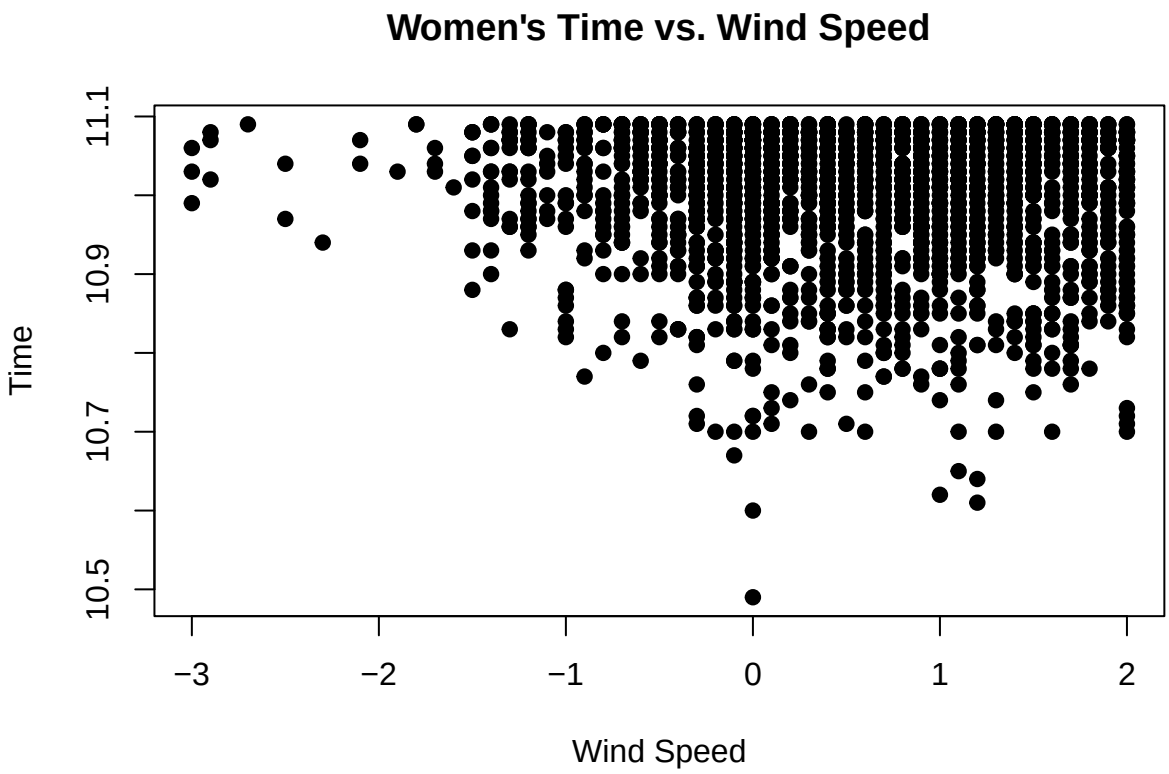
```
##   Rank Time Wind        Name Country Birthdate     City       Date
## 1    1 9.58  0.9  Usain Bolt     JAM  21.08.86   Berlin 16.08.2009
## 2    2 9.63  1.5  Usain Bolt     JAM  21.08.86   London 05.08.2012
## 3    3 9.69  0.0  Usain Bolt     JAM  21.08.86  Beijing 16.08.2008
## 4    3 9.69  2.0   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
```

- **1e.** For each of the men's and women's data frames, plot the the 100m sprint time versus the wind measurements, 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 for 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 . . .)

```
plot(sprint.m.df$Wind, sprint.m.df$Time, pch=19,
     main="Men's Time vs. Wind Speed", xlab="Wind Speed", ylab="Time")
```




```
plot(sprint.w.df$Wind, sprint.w.df$Time, pch=19,
     main="Women's Time vs. Wind Speed", xlab="Wind Speed", ylab="Time")
```



In general, higher windspeeds seem to correlate with faster times. However, it appears that the fastest women's time had no wind assistance.

Reordering data

- **2a.** Notice that the `Birthdate` and `Date` columns in both data frames `sprint.m.df` and `sprint.w.df` are currently factors that follow the format `DAY.MONTH.YEAR`. Write a function called `date.to.numeric()` that takes in a factor 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(as.factor("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 five 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.

```
date.to.numeric <- function(x){
  x <- as.character(x)
  x <- unlist(strsplit(x, split="\\."))
  if(nchar(x[3])==2){
    if(x[3]>=22){
      x[3] <- paste("19", x[3])
    }
    else{
      x[3] <- paste("20", x[3])
```

```
    }
  }
  x <- paste(x[3], x[2], x[1])
  x <- gsub(" ", "", x)
  return(as.numeric(x))
}

sprint.m.df$Date <- sapply(sprint.m.df$Date, FUN = date.to.numeric)
sprint.m.df$Birthdate <- sapply(sprint.m.df$Birthdate, FUN = date.to.numeric)
sprint.w.df$Date <- sapply(sprint.w.df$Date, FUN = date.to.numeric)
sprint.w.df$Birthdate <- sapply(sprint.w.df$Birthdate, FUN = date.to.numeric)
head(sprint.m.df, 5)
```

```
##   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
```

```
head(sprint.w.df, 5)
```

```
##   Rank  Time Wind                     Name Country Birthdate         City
## 1    1 10.49  0.0 Florence Griffith-Joyner     USA  19591221 Indianapolis
## 2    2 10.61  1.2 Florence Griffith-Joyner     USA  19591221 Indianapolis
## 3    3 10.62  1.0 Florence Griffith-Joyner     USA  19591221        Seoul
## 4    4 10.64  1.2          Carmelita Jeter     USA  19791124     Shanghai
## 5    5 10.65  1.1             Marion Jones     USA  19751012 Johannesburg
##       Date
## 1 19880716
## 2 19880717
## 3 19880924
## 4 20090920
## 5 19980912
```

- **2b.** 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 five lines of `sprint.m.df` and `sprint.w.df` afterwards.

```
i.slow <-  order(sprint.m.df$Time, decreasing=FALSE)
sprint.m.df <-  sprint.m.df[i.slow,]
i.slow <-  order(sprint.w.df$Time, decreasing=FALSE)
sprint.w.df <-  sprint.w.df[i.slow,]
head(sprint.m.df, 5)
```

```
##   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
```

```
head(sprint.w.df, 5)
```

```
##      Rank  Time Wind                     Name Country Birthdate         City
## 1       1 10.49  0.0 Florence Griffith-Joyner     USA  19591221 Indianapolis
## 2014    1 10.60  0.0             Zhanna Block     UKR  19720706         Kiev
```

```
## 2       2 10.61  1.2 Florence Griffith-Joyner     USA  19591221 Indianapolis
## 3       3 10.62  1.0 Florence Griffith-Joyner     USA  19591221        Seoul
## 4       4 10.64  1.2          Carmelita Jeter     USA  19791124     Shanghai
##           Date
## 1     19880716
## 2014  19970612
## 2     19880717
## 3     19880924
## 4     20090920
```

- **2c.** 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 five 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, 5)
```

```
##   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
##           City.Date
## 1   Berlin.20090816
## 2   London.20120805
## 3  Beijing.20080816
## 4 Shanghai.20090920
## 5 Lausanne.20120823
```

```
head(sprint.w.df, 5)
```

```
##      Rank  Time Wind                     Name Country Birthdate         City
## 1       1 10.49  0.0 Florence Griffith-Joyner     USA  19591221 Indianapolis
## 2014    1 10.60  0.0             Zhanna Block     UKR  19720706         Kiev
## 2       2 10.61  1.2 Florence Griffith-Joyner     USA  19591221 Indianapolis
## 3       3 10.62  1.0 Florence Griffith-Joyner     USA  19591221        Seoul
## 4       4 10.64  1.2          Carmelita Jeter     USA  19791124     Shanghai
##           Date             City.Date
## 1     19880716 Indianapolis.19880716
## 2014  19970612         Kiev.19970612
## 2     19880717 Indianapolis.19880717
## 3     19880924        Seoul.19880924
## 4     20090920     Shanghai.20090920
```

- **2d.** 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 1253 x 9, while `sprint.w.df` should be 921 x 9. Display the first five 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.

```
require(data.table)
```

```
## Loading required package: data.table
```

```
cleaner <- function(df){
  return(setDT(df)[, .SD[which.min(Time)], by=City.Date])
}
sprint.m.df <- cleaner(sprint.m.df)
sprint.w.df <- cleaner(sprint.w.df)
head(sprint.m.df, 5)
```

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

```
head(sprint.w.df, 5)
```

```
##                City.Date Rank  Time Wind                     Name Country
## 1: Indianapolis.19880716    1 10.49  0.0 Florence Griffith-Joyner     USA
## 2:         Kiev.19970612    1 10.60  0.0             Zhanna Block     UKR
## 3: Indianapolis.19880717    2 10.61  1.2 Florence Griffith-Joyner     USA
## 4:        Seoul.19880924    3 10.62  1.0 Florence Griffith-Joyner     USA
## 5:     Shanghai.20090920    4 10.64  1.2          Carmelita Jeter     USA
##    Birthdate         City     Date
## 1:  19591221 Indianapolis 19880716
## 2:  19720706         Kiev 19970612
## 3:  19591221 Indianapolis 19880717
## 4:  19591221        Seoul 19880924
## 5:  19791124     Shanghai 20090920
```

- **2e.** 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).

```
length(unique(sprint.m.df$City.Date))
```

```
## [1] 1253
```

```
length(unique(sprint.w.df$City.Date))
```

```
## [1] 921
```

Merging data

- **3a.** In preparation of merging `sprint.m.df` and `sprint.w.df`, we first wwant 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 377 rows and 7 columns. Print out the first five lines of `sprint.m.df` and `sprint.w.df` afterwards. Hint: you might find the `%in%` operator useful; try looking it its help file.

```
sprint.m.df <- sprint.m.df[sprint.m.df$City.Date %in% sprint.w.df$City.Date,]
sprint.w.df <- sprint.w.df[sprint.w.df$City.Date %in% sprint.m.df$City.Date,]
head(sprint.m.df, 5)
```

```
##                 City.Date Rank Time Wind        Name Country Birthdate
## 1:        Berlin.20090816    1 9.58  0.9  Usain Bolt     JAM  19860821
## 2:       Beijing.20080816    3 9.69  0.0  Usain Bolt     JAM  19860821
## 3:      Shanghai.20090920    3 9.69  2.0   Tyson Gay     USA  19820809
## 4:      Lausanne.20120823    3 9.69 -0.1 Yohan Blake     JAM  19891226
## 5: New York City.20080531    7 9.72  1.7  Usain Bolt     JAM  19860821
##             City     Date
## 1:        Berlin 20090816
## 2:       Beijing 20080816
## 3:      Shanghai 20090920
## 4:      Lausanne 20120823
## 5: New York City 20080531
```

```
head(sprint.w.df, 5)
```

```
##            City.Date Rank  Time Wind                      Name Country Birthdate
## 1:    Seoul.19880924    3 10.62  1.0 Florence Griffith-Joyner     USA  19591221
## 2: Shanghai.20090920    4 10.64  1.2          Carmelita Jeter     USA  19791124
## 3:  Sevilla.19990822    7 10.70 -0.1             Marion Jones     USA  19751012
## 4:   Eugene.20110604    7 10.70  2.0          Carmelita Jeter     USA  19791124
## 5: Kingston.20120629    7 10.70  0.6  Shelly-Ann Fraser-Pryce     JAM  19861227
##        City     Date
## 1:    Seoul 19880924
## 2: Shanghai 20090920
## 3:  Sevilla 19990822
## 4:   Eugene 20110604
## 5: Kingston 20120629
```

- **3b.** 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 five lines of `sprint.df` afterwards, and verify that its dimensions are 377 x 13.

```
i.city <-  order(sprint.m.df$City.Date, decreasing=FALSE)
sprint.m.df <- sprint.m.df[i.city]
i.city <-  order(sprint.w.df$City.Date, decreasing=FALSE)
sprint.w.df <- sprint.w.df[i.city,]

sprint.df <- cbind(sprint.m.df$City.Date,
                   sprint.m.df[, c("Rank", "Time", "Wind", "Name", "Country", "Birthdate")],
                   sprint.w.df[, c("Rank", "Time", "Wind", "Name", "Country", "Birthdate")])
head(sprint.df, 5)
```

```
##                    V1 Rank  Time Wind               Name Country Birthdate
## 1: Ad-Dawhah.19980507 2145 10.07  0.4     Donovan Bailey     CAN  19671216
## 2: Ad-Dawhah.20020515 2422 10.08  0.0   Bernard Williams     USA  19780119
```

```
## 3: Ad-Dawhah.20080509 1011 10.01  1.4 Jaysuma Saidy Ndure     NOR  19840101
## 4: Ad-Dawhah.20090508  882 10.00  1.4      Travis Padgett     USA  19861213
## 5: Ad-Dawhah.20120511  132  9.87  0.4       Justin Gatlin     USA  19820210
##    Rank  Time Wind             Name Country Birthdate
## 1:  722 10.99  0.7 Beverly McDonald     JAM  19700215
## 2:  888 11.01  0.4  Chandra Sturrup     BAH  19710912
## 3:  377 10.93  1.5    Allyson Felix     USA  19851118
## 4:  377 10.93  1.3   Kerron Stewart     JAM  19840416
## 5:  333 10.92  0.7    Allyson Felix     USA  19851118
```

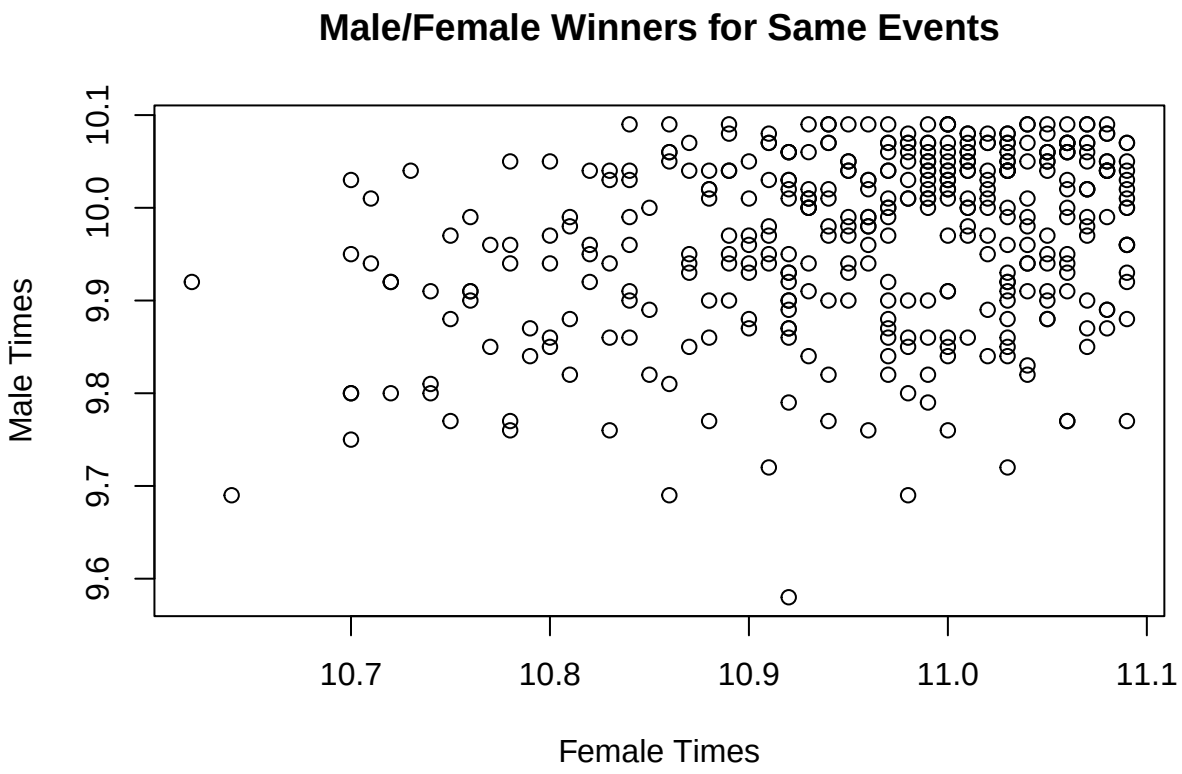
- **3c.** Ue the `merge()` function to recreate the merge in the previous part. This should require only one line of code; call the result `sprint.df.2`. In the call to `merge()`, make sure to set the argument `suffixes=c(".m",".w")`, which will help appropriately distinguish the column names after merging (a convenience of using the `merge()` function). The merged data frame `sprint.df2` should be of dimension 377 x 13; display its first five 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 arugment you should set in order to turn off this behavior; then check again the first five lines of the output `sprint.df2`, and compare to those from `sprint.df`.

```
sprint.df.2 <- merge(sprint.m.df[, c("City.Date","Rank","Time","Wind","Name","Country","Birthdate")],
                     sprint.w.df[, c("City.Date","Rank","Time","Wind","Name","Country","Birthdate")],
                     by.x = "City.Date", by.y = "City.Date", suffixes = c(".m", ".w"))
head(sprint.df.2, 5)
```

```
##             City.Date Rank.m Time.m Wind.m              Name.m Country.m
## 1: Ad-Dawhah.19980507   2145  10.07    0.4      Donovan Bailey       CAN
## 2: Ad-Dawhah.20020515   2422  10.08    0.0    Bernard Williams       USA
## 3: Ad-Dawhah.20080509   1011  10.01    1.4 Jaysuma Saidy Ndure       NOR
## 4: Ad-Dawhah.20090508    882  10.00    1.4      Travis Padgett       USA
## 5: Ad-Dawhah.20120511    132   9.87    0.4       Justin Gatlin       USA
##    Birthdate.m Rank.w Time.w Wind.w           Name.w Country.w Birthdate.w
## 1:    19671216    722  10.99    0.7 Beverly McDonald       JAM    19700215
## 2:    19780119    888  11.01    0.4  Chandra Sturrup       BAH    19710912
## 3:    19840101    377  10.93    1.5    Allyson Felix       USA    19851118
## 4:    19861213    377  10.93    1.3   Kerron Stewart       JAM    19840416
## 5:    19820210    333  10.92    0.7    Allyson Felix       USA    19851118
```

- **3d.** 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. Then, use the `cor.test()` function to determine if there is a significant correlation between `Time.w` and `Time.m`—specifically, report the `p.value` from its output. In the call to `cor.test()`, use all default arguments.

```
plot(sprint.df.2$Time.w, sprint.df.2$Time.m,
     main="Male/Female Winners for Same Events", xlab="Female Times", ylab="Male Times")
```




```
cor.test(sprint.df.2$Time.w, sprint.df.2$Time.m)
```

```
##
##  Pearson's product-moment correlation
##
## data:  sprint.df.2$Time.w and sprint.df.2$Time.m
## t = 5.9721, df = 375, p-value = 5.441e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1996416 0.3842650
## sample estimates:
##      cor
## 0.294701
```

It appears that there might be a slight track effect. Although the graph doesn't reveal much, the correlation test has very low p-value and indicates the correlation is somewhere around 0.3. Looking at the graph, this correlation seems feasible.