Lab 5: Plotting Tools

Name: Rufus Petrie

This week's agenda: getting familiar with basic plotting tools; understanding the way layers work; recalling basic text manipulations; producing histograms and overlaid histograms; heatmaps.

Fastest 100m sprint times

Below, we read in a data set of the fastest times ever recorded for the 100m sprint, in men's track. (Usain Bolt may have slowed down now … but he was truly one of a kind!) We also read in a data set of the fastest times ever recorded for the 100m, in women's track. Both of these data sets were scraped from http://www.alltime-athletics.com/m_100ok.htm (we scraped it in spring 2018; this website may have been updated since).

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

Data frame and apply practice

- **1a.** Confirm that both `sprint.m.dat` and `sprint.w.dat` are data frames. Delete the `Rank` and `City` columns from each data frame. Then display the first and last 5 rows of each. **Challenge**: compute the ranks for the men's data set from the `Time` column and add them back as a `Rank` column to `sprint.m.dat`. Do the same for the women's data set.

```
class(sprint.m.dat)
```

```
## [1] "data.frame"
```

```
class(sprint.w.dat)
```

```
## [1] "data.frame"
```

```
head(sprint.m.dat, 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
```

```
tail(sprint.m.dat, 5)
```

```
##      Rank  Time Wind                Name Country Birthdate      City       Date
## 2984 2691 10.09  1.6       Daniel Bailey     ANT  09.09.86  Kingston 11.06.2016
## 2985 2691 10.09  0.7 Christophe Lemaitre     FRA  11.06.90    Angers 25.06.2016
## 2986 2691 10.09  0.1       Ramon Gittens     BAR  20.07.87 Waterford 26.06.2016
## 2987 2691 10.09  1.9        Ronnie Baker     USA  15.10.93    Eugene 02.07.2016
## 2988 2691 10.09 -0.1        Julian Forte     JAM  01.07.93  Warszawa 28.08.2016
```

```r
head(sprint.w.dat, 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
```

```r
tail(sprint.w.dat, 5)
```

```
##      Rank Time Wind                 Name Country Birthdate           City
## 2014    1 10.6  0,0         Zhanna Block     UKR  06.07.72           Kiev
## 2015    2 10.7 +1,1      Juliet Cuthbert     JAM  09.04.64       Kingston
## 2016    2 10.7  0,0         Zhanna Block     UKR  06.07.72           Kiev
## 2017    2 10.7 -0,2 Svetlana Goncharenko     RUS  26.05.71 Rostov-na-Donu
## 2018    2 10.7 +1,3     Blessing Okagbare     NGR  10.09.88        El Paso
##            Date
## 2014 12.06.1997
## 2015 04.07.1992
## 2016 12.06.1997
## 2017 30.05.1998
## 2018 10.04.2010
```

- **1b.** Using `table()`, compute for each unique country in the `Country` column of `sprint.m.dat`, the number of sprint times from this country that appear in the data set. Call the result `sprint.m.counts`. Do the same for the women, calling the result `sprint.w.counts`. What are the 5 most represented countries, for the men, and for the women? (Interesting side note: go look up the population of Jamaica, compared to that of the US. Pretty impressive, eh?)

```r
sprint.m.counts <- table(sprint.m.dat$Country)
sprint.w.counts <- table(sprint.w.dat$Country)
sort(sprint.m.counts, decreasing = TRUE)[1:5]
```

```
##
##  USA  JAM  GBR  TTO  CAN
## 1179  581  186  180  112
```

```r
sort(sprint.w.counts, decreasing = TRUE)[1:5]
```

```
##
## USA JAM BAH GDR NGR
## 754 438 110  83  75
```

- **1c.** Are there any countries that are represented by women but not by men, and if so, what are they?

Vice versa, represented by men and not women? Hint: you will want to use the `%in%` operator. If you're sure what it does you can read the documentation.

```
sprint.m.counts[!(rownames(sprint.m.counts) %in% rownames(sprint.w.counts))]
```

```
##
## AHO ANT AUS AZE BAR CUB GHA HUN IRI ITA JPN KOR KSA MAR NAM NOR OMA PAN POR QAT
##  17  25  10   1  36  10  35   2   1   3  38   1   1   1  62  17   1   2  29  22
## SKN SLE TUR ZAM ZIM
##  77   1  16   4   4
```

```
sprint.w.counts[!(rownames(sprint.w.counts) %in% rownames(sprint.m.counts))]
```

```
##
## BEL BLR BUL CMR CZE FRG GAB GEO GRE MEX RUS SLO SRI UZB VIN
##   6   9  42   2   1   6   4   1  31   1  70   1   2   1   2
```

- **1d.** Using some method for data frame subsetting, and then `table()`, recompute the counts of countries in `sprint.m.dat`, but now only counting sprint times that are faster than or equal to 10 seconds. Call the result `sprint.m.10.counts`. Recompute counts for women too, now only counting sprint times that are faster than or equal to 11 seconds, and call the result `sprint.w.11.counts`. What are the 5 most represented countries now, for men, and for women?

```
sprint.m.10.counts <- table(sprint.m.dat[sprint.m.dat["Time"]<10,]$Country)
sprint.w.11.counts <- table(sprint.w.dat[sprint.w.dat["Time"]<11,]$Country)
sort(sprint.m.10.counts, decreasing = TRUE)[1:5]
```

```
##
## USA JAM TTO CAN FRA
## 358 257  51  32  27
```

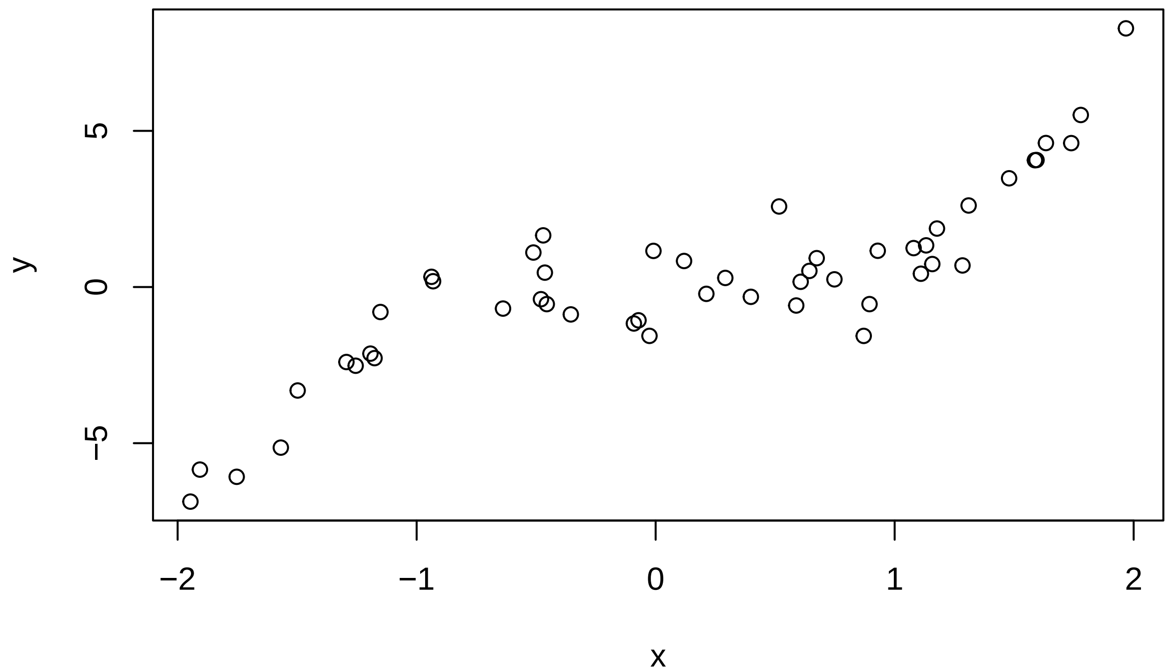
```
sort(sprint.w.11.counts, decreasing = TRUE)[1:5]
```

```
##
## USA JAM GDR TTO BAH
## 315 216  39  31  26
```

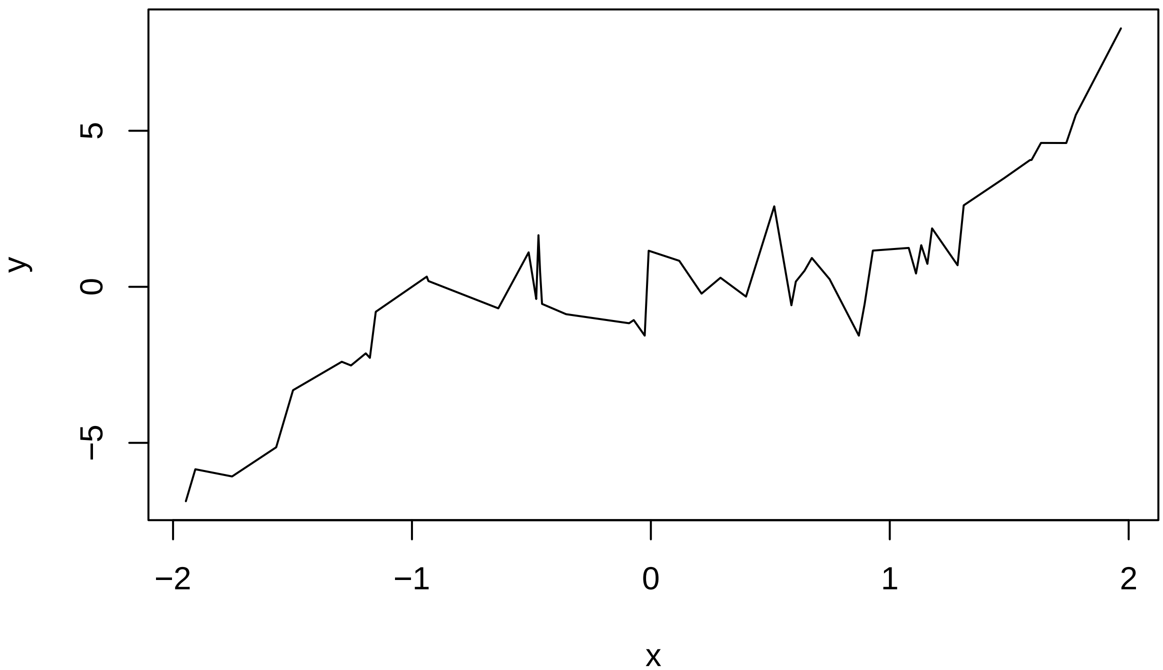
Plot basics

- **2a.** Below is some code that is very similar to that from the lecture, but with one key difference. Explain: why does the `plot()` result with with `type="p"` look normal, but the `plot()` result with `type="l"` look abnormal, having crossing lines? Then modify the code below (hint: modify the definition of `x`), so that the lines on the second plot do not cross.

```
n = 50
set.seed(0)
x = sort(runif(n, min=-2, max=2))
y = x^3 + rnorm(n)
plot(x, y, type="p")
```



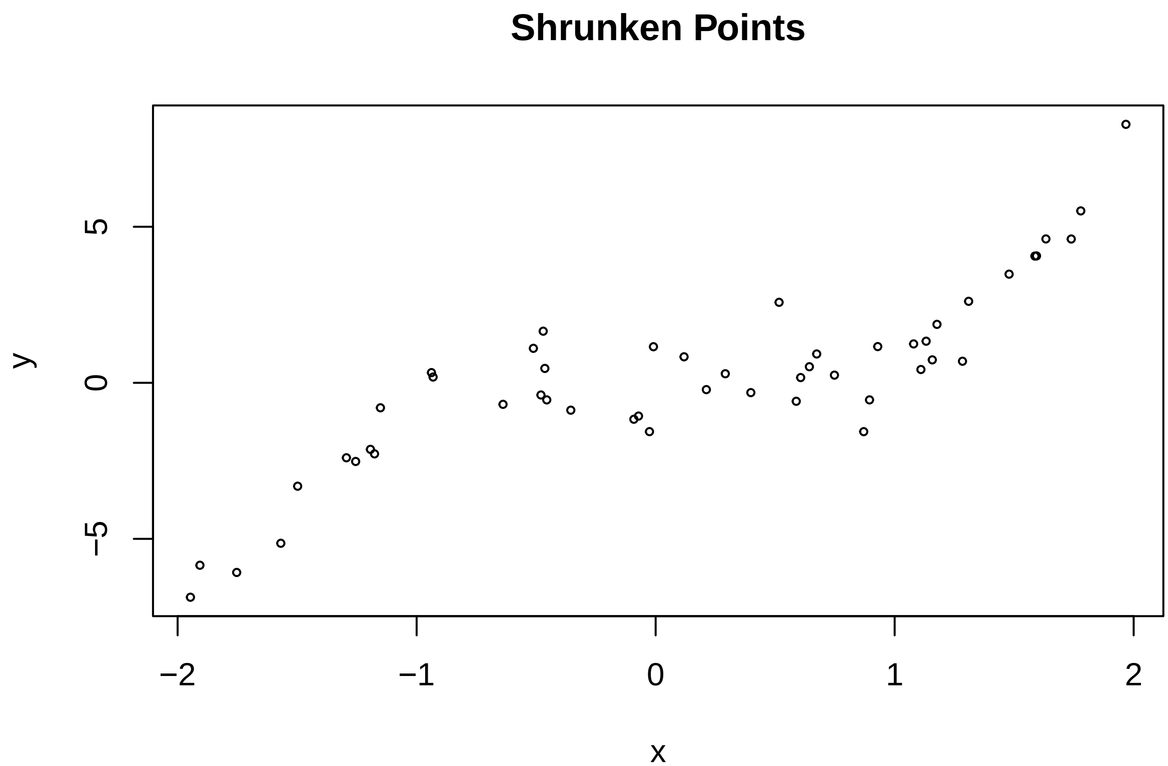

```
plot(x, y, type="l")
```



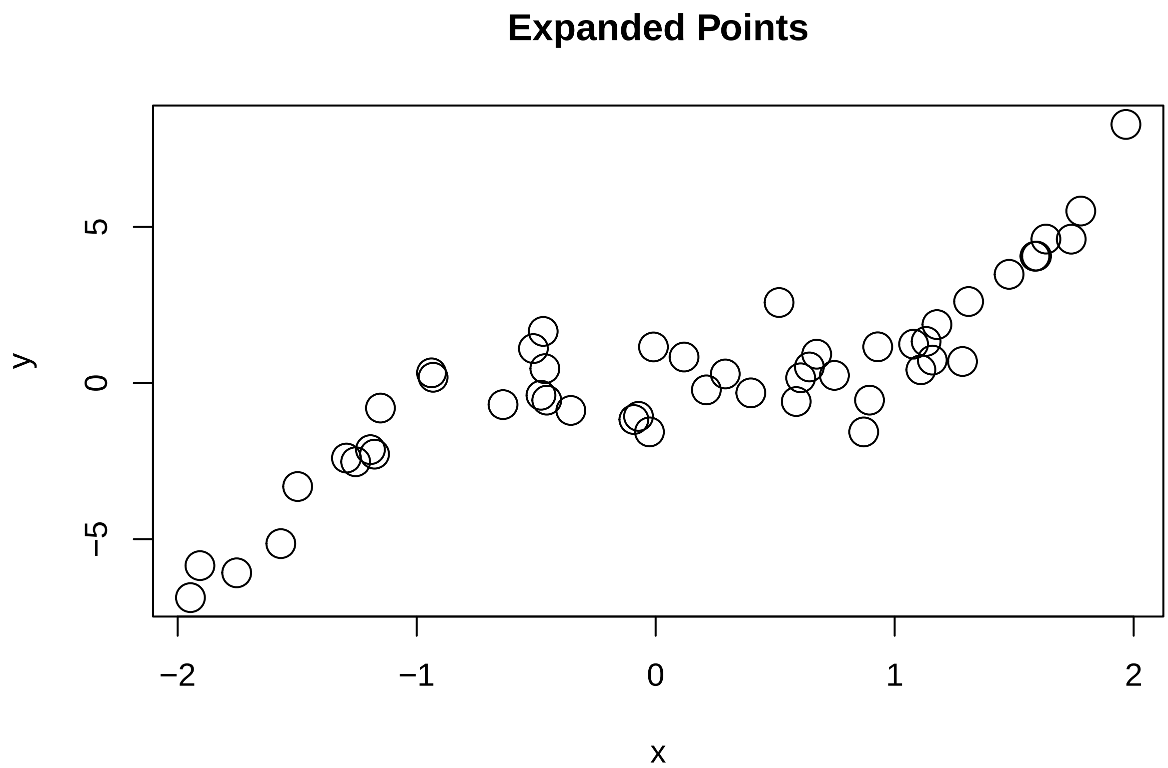
The second plot had crossing lines because x was not sorted.

- **2b.** The `cex` argument can used to shrink or expand the size of the points that are drawn. Its default value is 1 (no shrinking or expansion). Values between 0 and 1 will shrink points, and values larger than 1 will expand points. Plot `y` versus `x`, first with `cex` equal to 0.5 and then 2 (so, two separate plots). Give titles "Shrunken points", and "Expanded points", to the plots, respectively.

```
plot(x, y, type="p", cex=0.5, main="Shrunken Points")
```

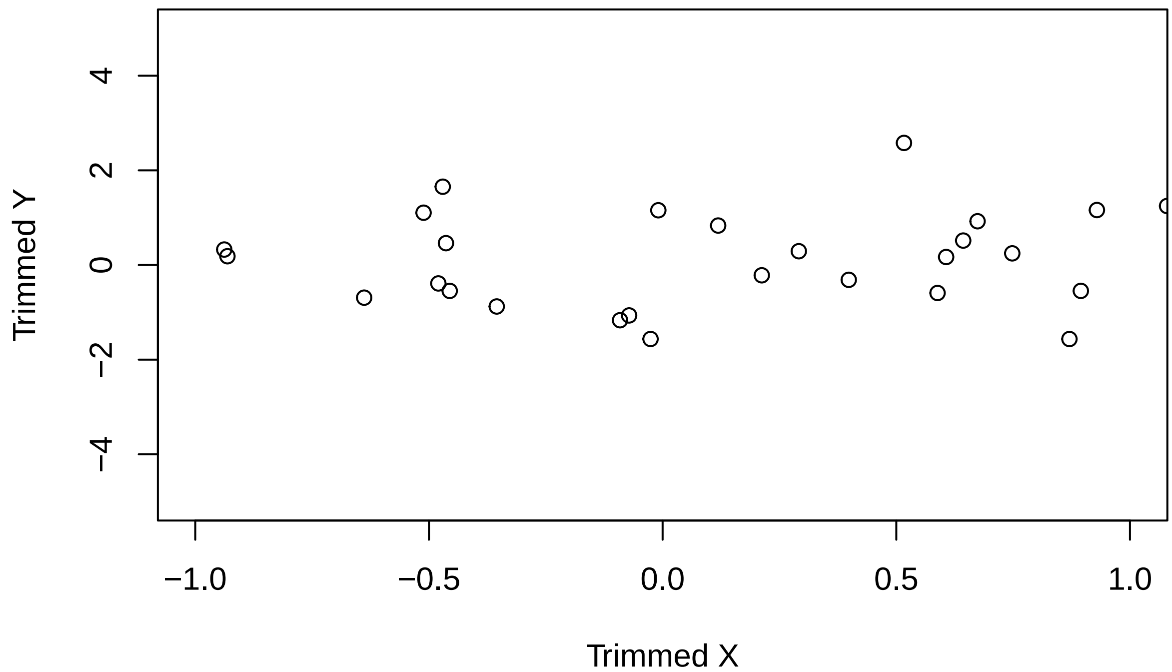



```r
plot(x, y, type="p", cex=2, main="Expanded Points")
```



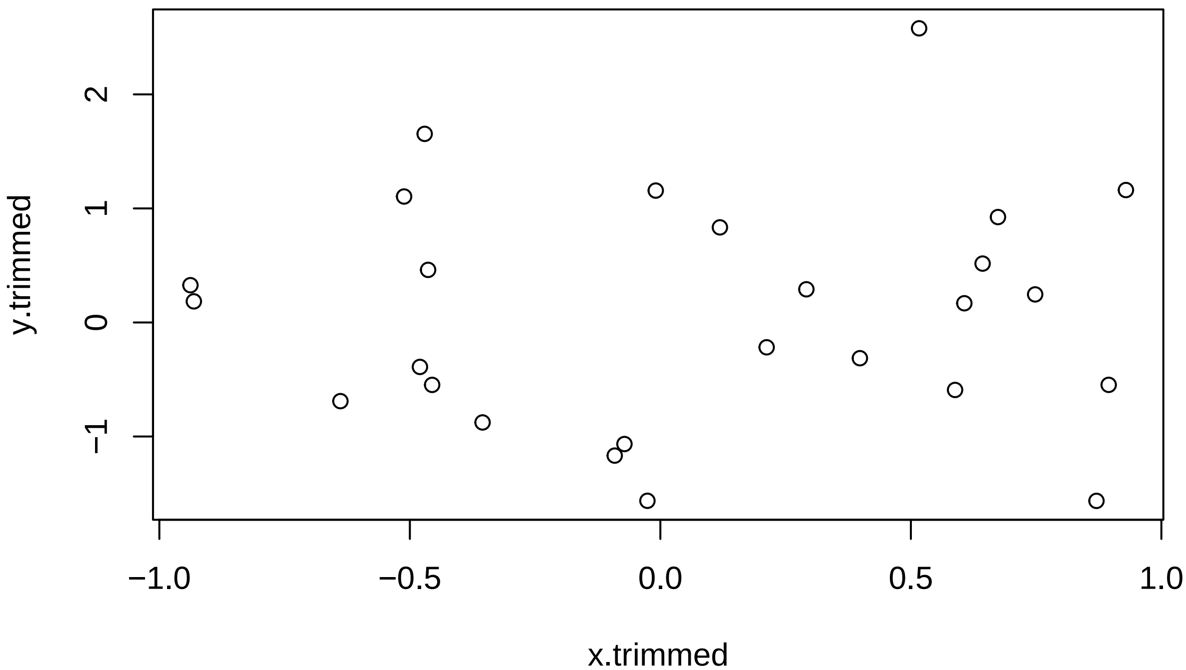
- **2c.** The `xlim` and `ylim` arugments can be used to change the limits on the x-axis and y-axis, repsectively. Each argument takes a vector of length 2, as in `xlim = c(-1, 0)`, to set the x limit to be from -1 to 0. Plot `y` versus `x`, with the x limit set to be from -1 to 1, and the y limit set to be from -5 to 5. Assign x and y labels “Trimmed x” and “Trimmed y”, respectively.

```
plot(x, y, type="p", xlim = c(-1,1), ylim = c(-5,5), xlab="Trimmed X", ylab="Trimmed Y")
```



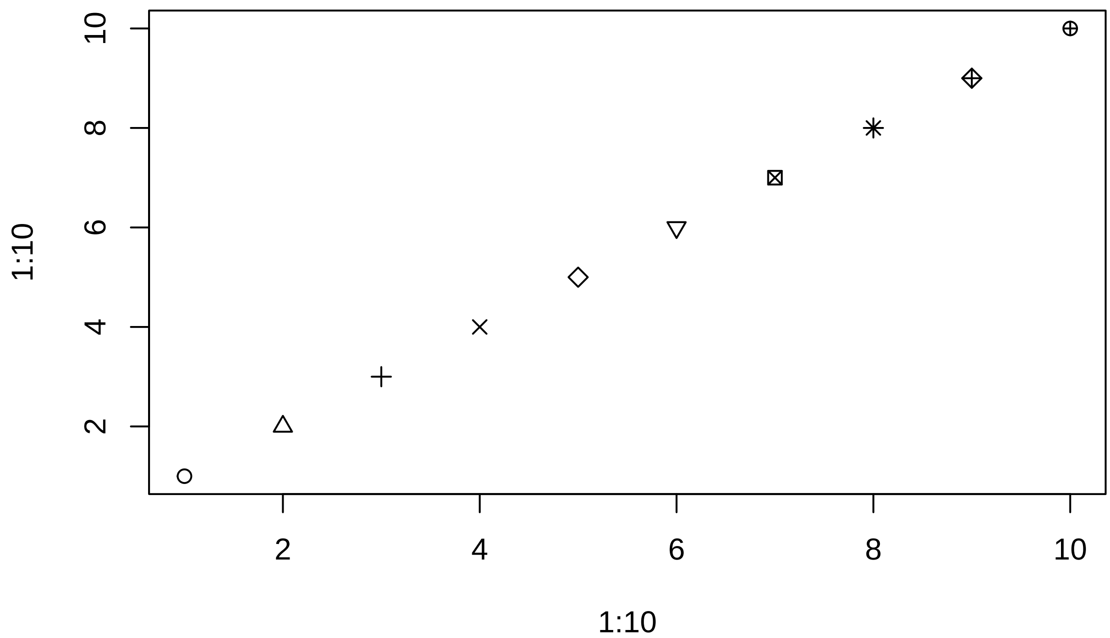
- **2d.** Again plot `y` versus `x`, only showing points whose x values are between -1 and 1. But this time, define `x.trimmed` to be the subset of `x` between -1 and 1, and define `y.trimmed` to be the corresponding subset of `y`. Then plot `y.trimmed` versus `x.trimmed` without setting `xlim` and `ylim`: now you should see that the y limit is (automatically) set as "tight" as possible. Hint: use logical indexing to define `x.trimmed`, `y.trimmed`.

```
x.trimmed <- x[x<=1 & x>=-1]
y.trimmed <- y[x<=1 & x>=-1]
plot(x.trimmed, y.trimmed)
```



- **2e.** The `pch` argument, recall, controls the point type in the display. In the lecture examples, we set it to a single number. But it can also be a vector of numbers, with one entry per point in the plot. So, e.g.,

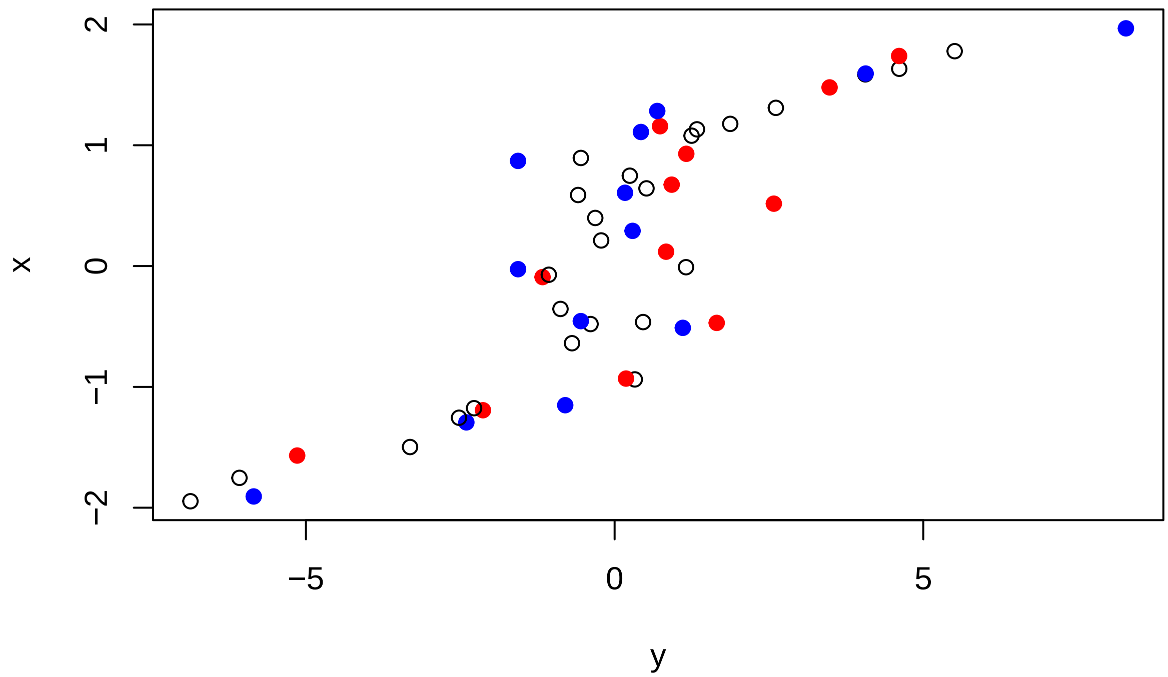
```
plot(1:10, 1:10, pch=1:10)
```



displays the first 10 point types. If `pch` is a vector whose length is shorter than the total number of points to be plotted, then its entries are recycled, as appropriate. Plot `y` versus `x`, with the point type alternating in between an empty circle and a filled circle.

- **2f.** The `col` argument, recall, controls the color the points in the display. It operates similar to `pch`, in the sense that it can be a vector, and if the length of this vector is shorter than the total number of points, then it is recycled appropriately. Plot `y` versus `x`, and repeat the following pattern for the displayed points: a black empty circle, a blue filled circle, a black empty circle, a red filled circle.

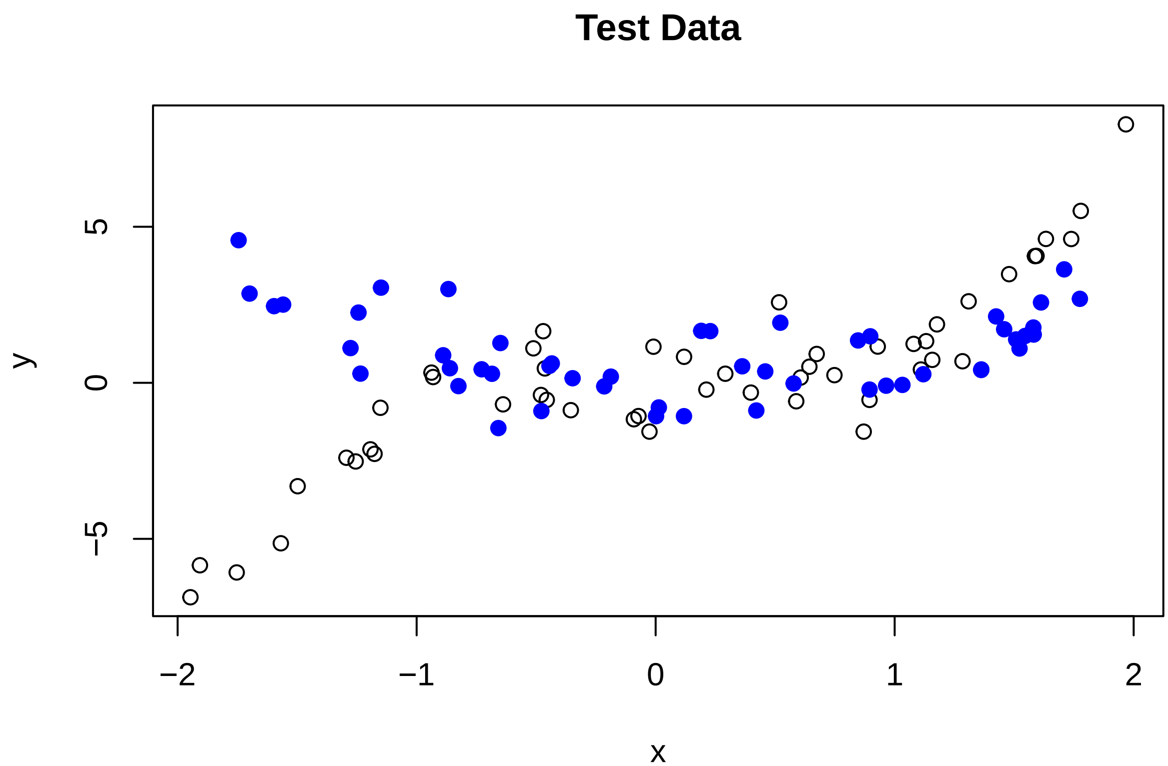
```
plot(y, x, col=c("black", "blue", "black", "red"), pch=c(21, 19))
```



Adding to plots

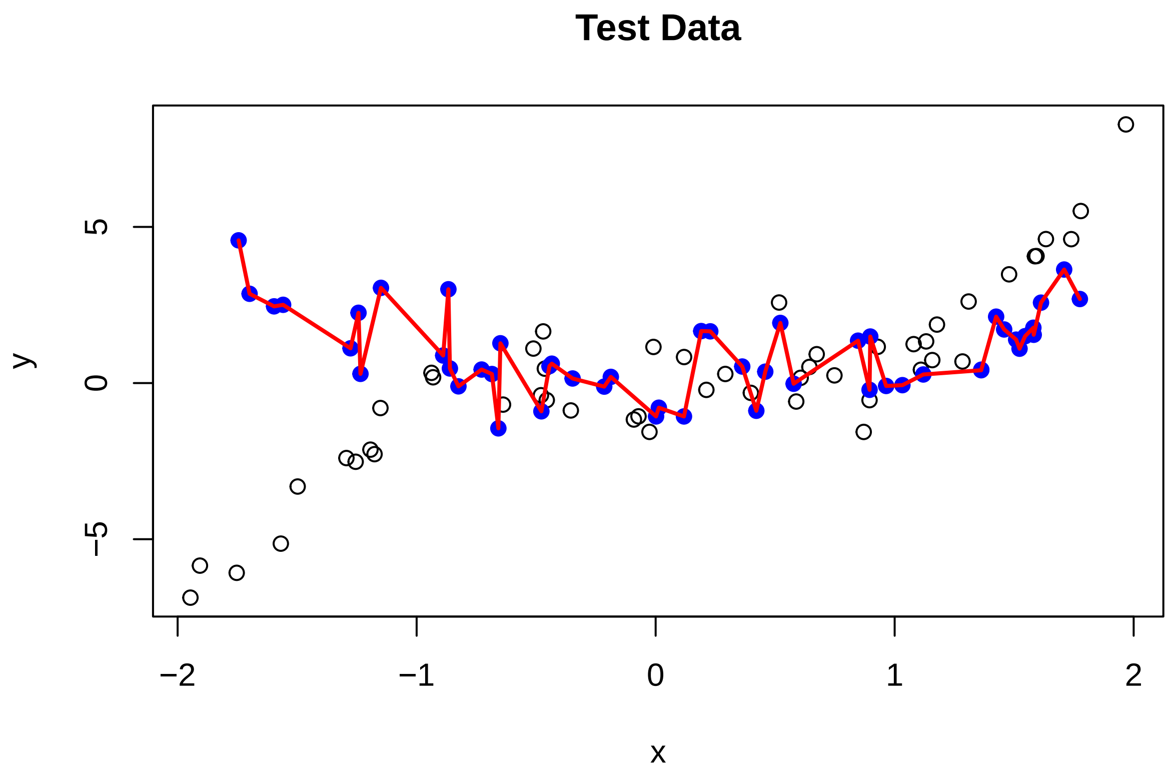
- **3a.** Produce a scatter plot of `y` versus `x`, and set the title and axes labels as you see fit. Then overlay on top a scatter plot of `y2` versus `x2`, using the `points()` function, where `x2` and `y2` are as defined below. In the call to `points()`, set the `pch` and `col` arguments appropriately so that the overlaid points are drawn as filled blue circles.

```
x2 = sort(runif(n, min=-2, max=2))
y2 = x^2 + rnorm(n)
plot(x, y, main = "Test Data")
points(x2, y2, col="blue", pch=19)
```



- **3b.** Starting with your solution code from the last question, overlay a line plot of `y2` versus `x2` on top of the plot (which contains empty black circles of `y` versus `x`, and filled blue circles of `y2` versus `x2`), using the `lines()` function. In the call to `lines()`, set the `col` and `lwd` arguments so that the line is drawn in red, with twice the normal thickness. Look carefully at your resulting plot. Does the red line pass overtop of or underneath the blue filled circles? What do you conclude about the way R *layers* these additions to your plot?

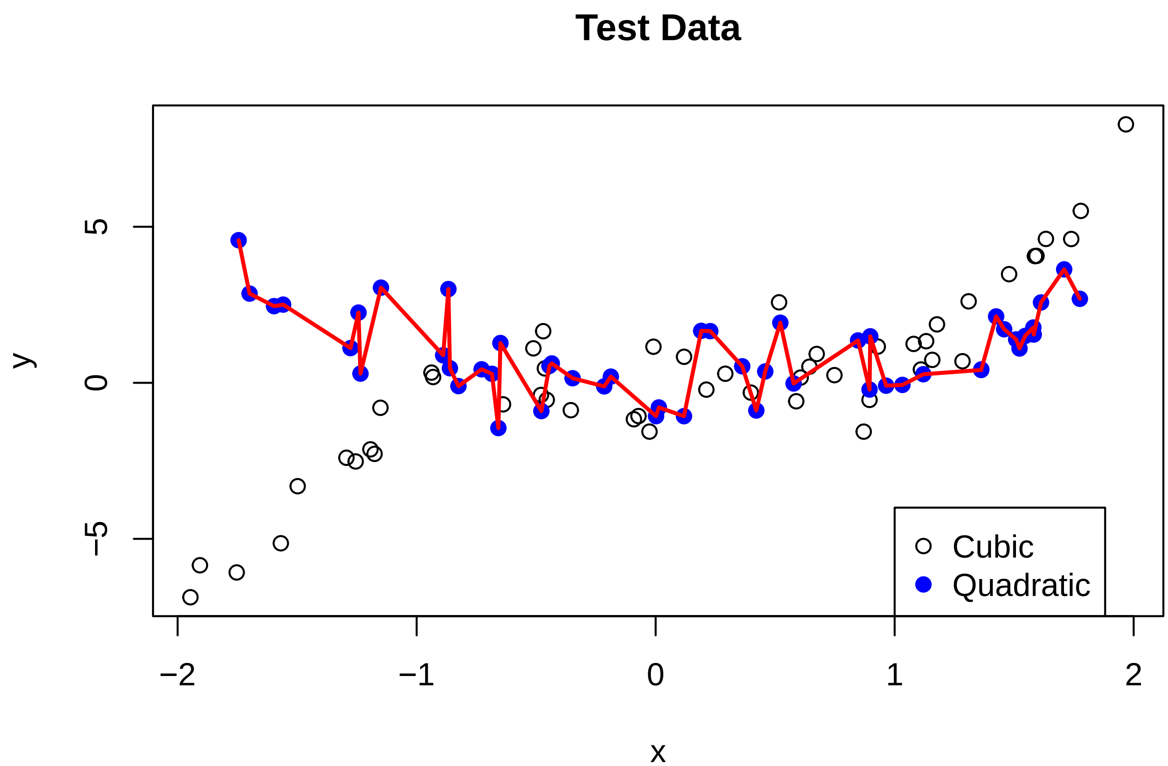
```
plot(x, y, main = "Test Data")
points(x2, y2, col="blue", pch=19)
lines(x2, y2, col="red", lwd=2)
```



The lines pass over the new points, so newer layers take priority.

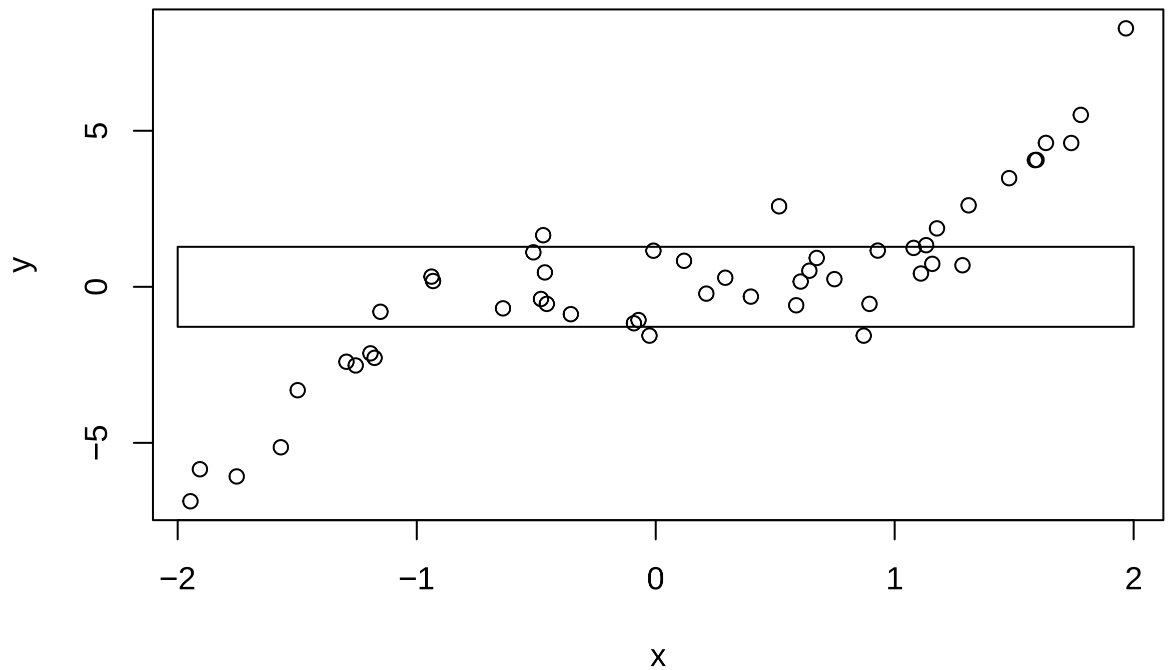
- **3c.** Starting with your solution code from the last question, add a legend to the bottom right corner of the the plot using `legend()`. The legend should display the text: "Cubic" and "Quadratic", with corresponding symbols: an empty black circle and a filled blue circle, respectively. Hint: it will help to look at the documentation for `legend()`.

```
plot(x, y, main = "Test Data")
points(x2, y2, col="blue", pch=19)
lines(x2, y2, col="red", lwd=2)
legend(1, -4, legend=c("Cubic", "Quadratic"), col=c("black", "blue"), pch=c(21,19))
```



- **3d.** Produce a plot of `y` versus `x`, but with a gray rectangle displayed underneath the points, which runs has a lower left corner at `c(-2, qnorm(0.1))`, and an upper right corner at `c(2, qnorm(0.9))`. Hint: use `rect()` and consult its documentation. Also, remember how layers work; call `plot()`, with `type="n"` or `col="white"` in order to refrain from drawing any points in the first place, then call `rect()`, then call `points()`.

```
plot(x, y, col="white")
rect(-2, qnorm(0.1), 2, qnorm(0.9))
points(x, y)
```



Text manipulations, and layered plots

- **4a.** Back to the sprinters data set: define `sprint.m.times` to be the `Time` column of `sprint.m.dat`. Define `sprint.m.dates` to be the `Date` column of `sprint.m.dat`, converted into a character vector. Define a character vector `sprint.m.years` to contain the last 4 characters of an entry of `sprint.m.dates`. Hint: use `substr()`. Finally, convert `sprint.m.years` into a numeric vector. Display its first 10 entries.

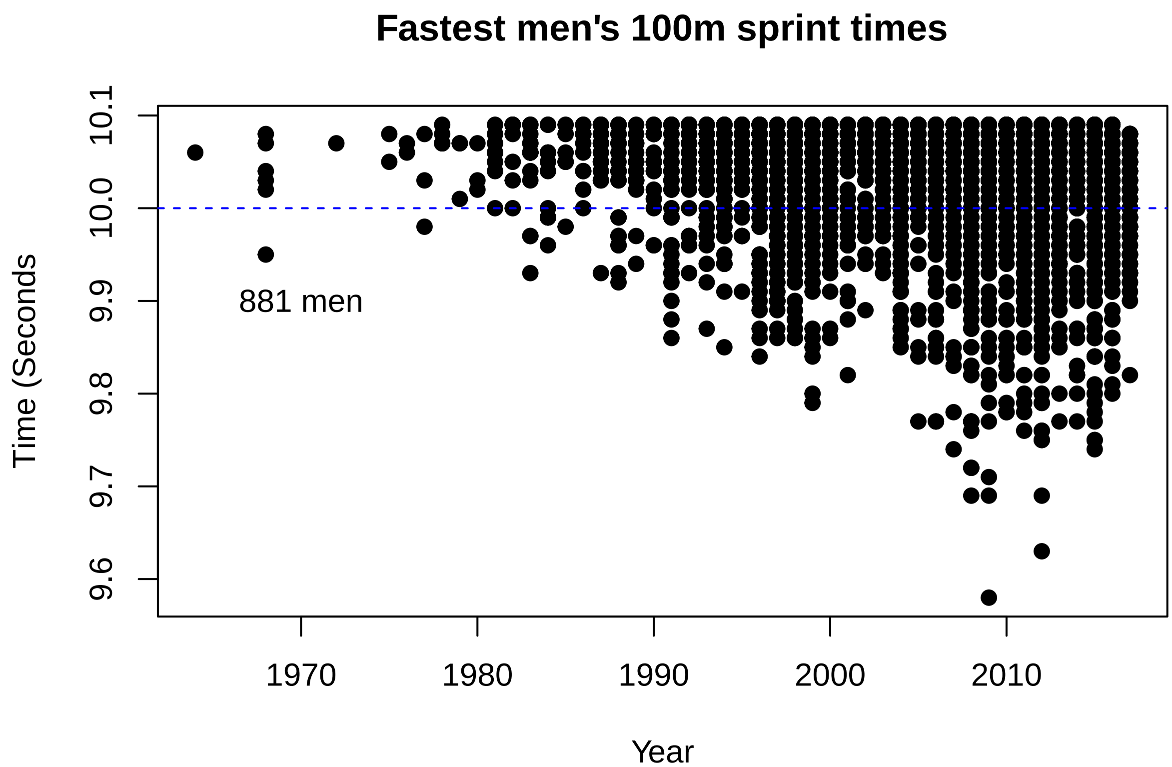
```
sprint.m.times <- sprint.m.dat$Time
sprint.m.dates <- as.character(sprint.m.dat$Date)
sprint.m.years <- substr(sprint.m.dates, nchar(sprint.m.dates)-3, nchar(sprint.m.dates))
sprint.m.years <- as.numeric(sprint.m.years)
sprint.m.years[1:10]
```

```
##  [1] 2009 2012 2008 2009 2012 2009 2008 2008 2007 2015
```

- **4b.** Plot `sprint.m.times` versus `sprint.m.years`. For the point type, use small, filled black circles. Label the x-axis "Year" and the y-axis "Time (seconds)". Title the plot "Fastest men's 100m sprint times". Using `abline()`, draw a dashed blue horizontal line at 10 seconds. Using `text()`, draw below this line, in text on the plot, the string "N men", replacing "N" here by the number of men who have run under 10 seconds. Your code should programmatically determine the correct number here, and use `paste()` to form the string. Comment on what you see visually, as per the sprint times across the years. What does the trend look like for the fastest time in any given year?

```
plot(sprint.m.years, sprint.m.times, pch=19, col="black",
     xlab="Year", ylab="Time (Seconds", main="Fastest men's 100m sprint times")
```

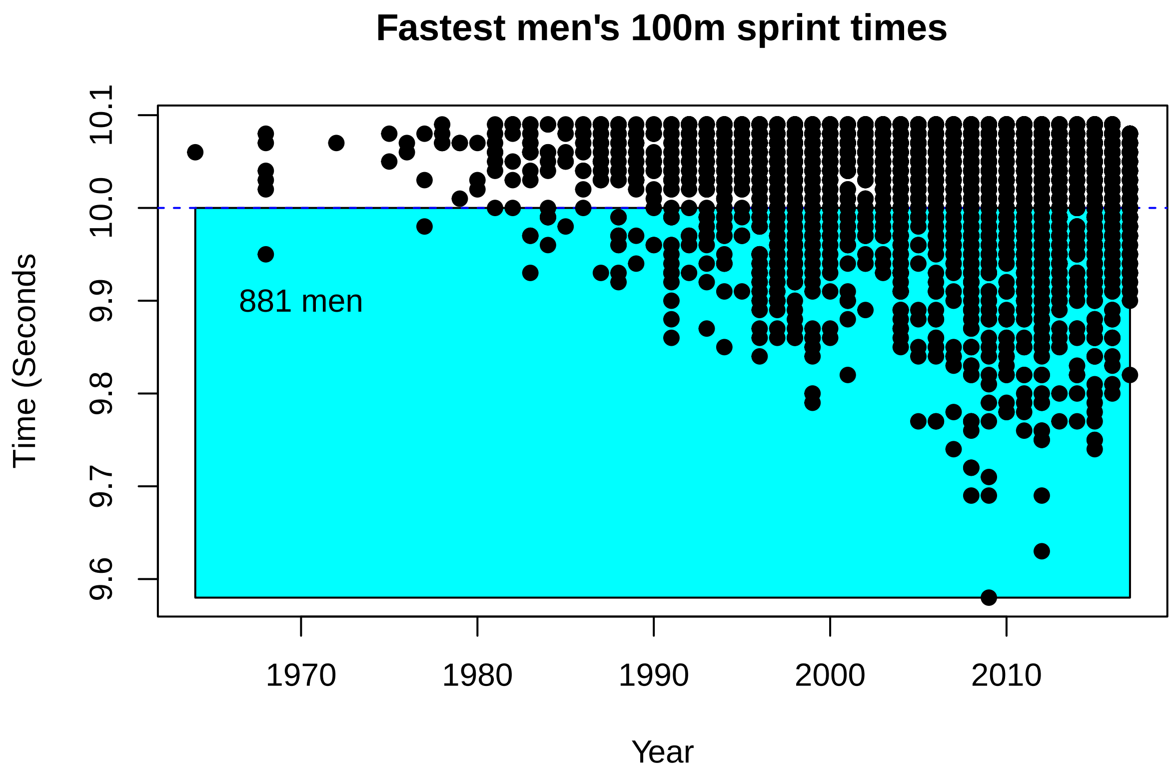
```r
abline(h=10, col="blue", lty=2)
text(paste(length(sprint.m.times[sprint.m.times<10]), "men"), x=1970, y=9.9)
```



The number of sprint times below 10 seconds has been steadily increasing over the years. The fastest time also steaily decreased until about 2010.

- **4c.** Reproduce the previous plot, but this time, draw a light blue rectangle underneath all of the points below the 10 second mark. The rectangle should span the entire region of the plot below the horizontal line at $y = 10$. And not only the points of sprint times, but the blue dashed line, and the text "N men" (with "N" replaced by the appropriate number) should appear *on top* of the rectangle. Hint: use `rect()` and layering as appropriate.

```r
plot(sprint.m.years, sprint.m.times, pch=19, col="white",
     xlab="Year", ylab="Time (Seconds", main="Fastest men's 100m sprint times")
rect(1964, 9.58, 2017, 10, col="cyan")
abline(h=10, col="blue", lty=2)
points(sprint.m.years, sprint.m.times, pch=19)
text(paste(length(sprint.m.times[sprint.m.times<10]), "men"), x=1970, y=9.9)
```

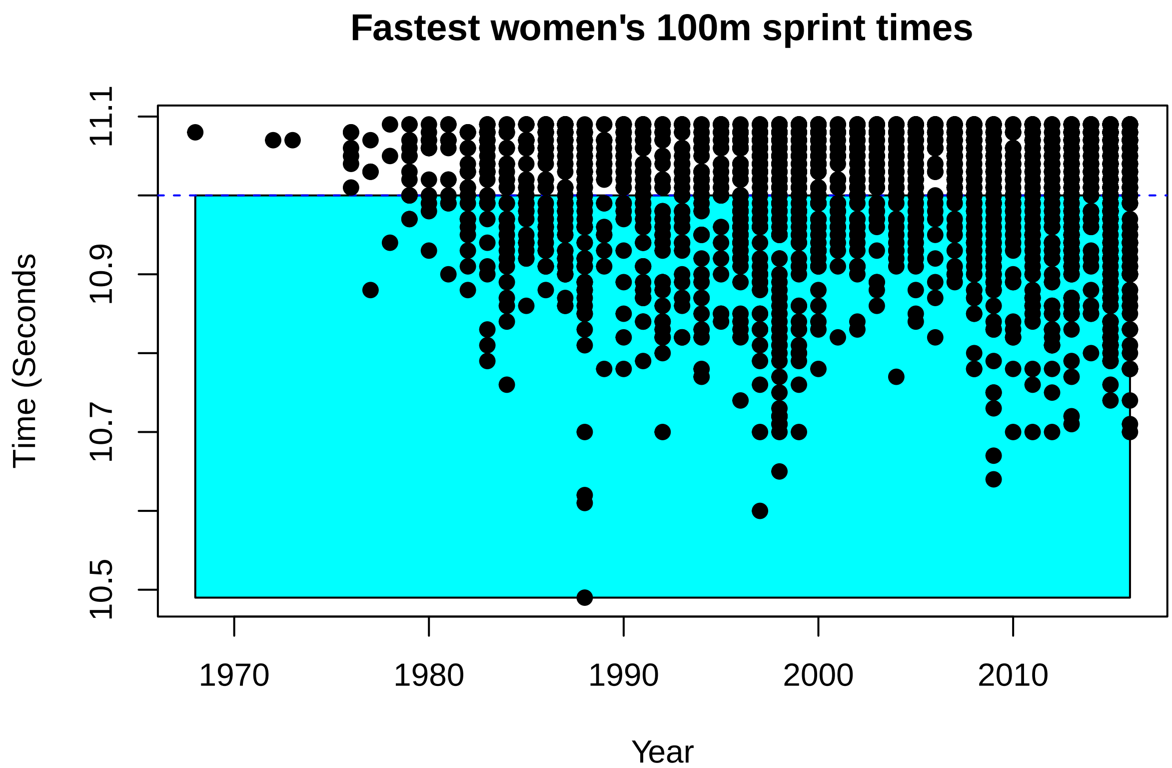


- **4d.** Repeat Q4a but for the women's sprint data, arriving at vectors `sprint.w.times` and `sprint.w.years`. Then repeat Q4c for this data, but with the 10 second cutoff being replaced by 11 seconds, the rectangle colored pink, and the dashed line colored red. Comment on the differences between this plot for the women and your plot for the men, from Q4c. In particular, is there any apparent difference in the trend for the fastest sprint time in any given year?

```
sprint.w.times <- sprint.w.dat$Time
sprint.w.dates <- as.character(sprint.w.dat$Date)
sprint.w.years <- substr(sprint.w.dates, nchar(sprint.w.dates)-3, nchar(sprint.w.dates))
sprint.w.years <- as.numeric(sprint.w.years)
sprint.w.years[1:10]
```

```
##  [1] 1988 1988 1988 2009 1998 2009 1988 1999 2011 2012
```

```
plot(sprint.w.years, sprint.w.times, pch=19, col="white",
     xlab="Year", ylab="Time (Seconds", main="Fastest women's 100m sprint times")
rect(min(sprint.w.years), min(sprint.w.times), max(sprint.w.years), 11, col="cyan")
abline(h=11, col="blue", lty=2)
points(sprint.w.years, sprint.w.times, pch=19)
text(paste(length(sprint.m.times[sprint.m.times<11]), "women"), x=1970, y=9.9)
```



Although the number of women with times below 11 seconds has been steadily increased, the maximum time was set in ~1988 and hasn't been matched since.

More text manipulations, and histograms

- **5a.** Extract the birth years of the sprinters from the data frame `sprint.m.dat`. To do so, define `sprint.m.bdates` to be the `Birthdate` column of `sprint.m.dat`, converted into a character vector. Then define a character vector `sprint.m.byears` to contain the last 2 characters of each entry of `sprint.m.bdates`. Convert `sprint.m.byears` into a numeric vector, add 1900 to each entry, and redefine `sprint.m.byears` to be the result. Finally, compute a vector `sprint.m.ages` containing the age (in years) of each sprinter when their sprint time was recorded. Hint: use `sprint.m.byears` and `sprint.m.years`.

```
sprint.m.bdates <- sprint.m.dat[,"Birthdate"]
sprint.m.byears <- substr(sprint.m.bdates, nchar(sprint.m.bdates)-1, nchar(sprint.m.bdates))
sprint.m.byears <- as.numeric(sprint.m.byears)
sprint.m.byears <- (sprint.m.byears+1900)
sprint.m.ages <- sprint.m.years - sprint.m.byears
```

- **5b.** Repeat the last question, but now for the data `sprint.w.dat`, arriving at a vector of ages called `sprint.w.ages`.

```
sprint.w.bdates <- sprint.w.dat[,"Birthdate"]
sprint.w.byears <- substr(sprint.w.bdates, nchar(sprint.w.bdates)-1, nchar(sprint.w.bdates))
sprint.w.byears <- as.numeric(sprint.w.byears)
```

```r
sprint.w.byears <- (sprint.w.byears+1900)
sprint.w.ages <- sprint.w.years - sprint.w.byears
```

- **5c.** Using one of the apply functions, compute the average sprint time for each age in `sprint.m.ages`, calling the result `time.m.avg.by.age`. Similarly, compute the analogous quantity for the women, calling the result `time.w.avg.by.age`. Are there any ages for which the men's average time is faster than 10 seconds, and if so, which ones? Are there any ages for which the women's average time is faster than 10.98 seconds, and if so, which ones?

```r
time.m.avg.by.age <- tapply(sprint.m.dat[,"Time"], sprint.m.ages, mean)
time.w.avg.by.age <- tapply(sprint.w.dat[,"Time"], sprint.w.ages, mean)
time.m.avg.by.age[time.m.avg.by.age<10]
```

```
##       33
## 9.969444
```

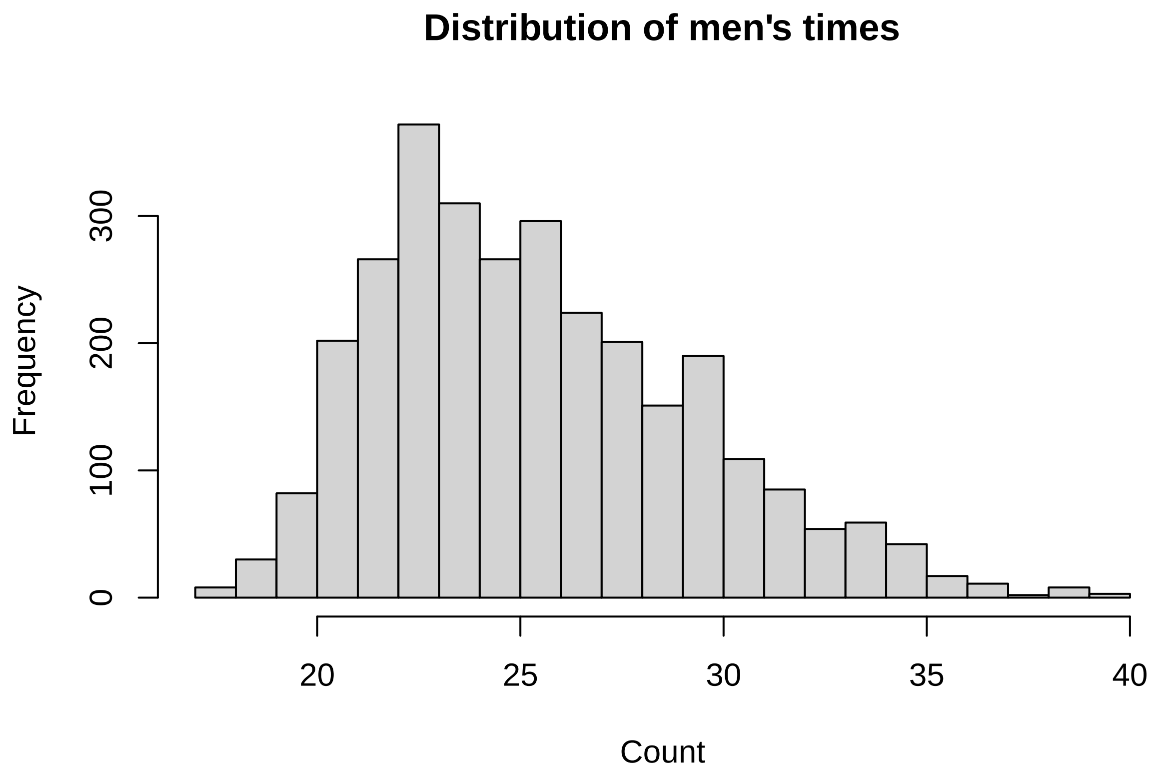
```r
time.w.avg.by.age[time.w.avg.by.age<10.98]
```

```
##       29       33       36       37
## 10.97696 10.97500 10.97056 10.95778
```

Men's average age is faster than 10 seconds for 33 year olds, and women's average time is faster than 10.98 for 29, 33, 36, and 37 year olds.

- **5d.** Plot a histogram of `sprint.m.ages`, with break locations occuring at every age in between 17 and 40. Color the histogram to your liking; label the x-axis, and title the histogram appropriately. What is the mode, i.e., the most common age? Also, describe what you see around the mode: do we see more sprinters who are younger, or older?

```r
hist(sprint.m.ages, breaks = 17:40, xlab = "Count", main = "Distribution of men's times")
```

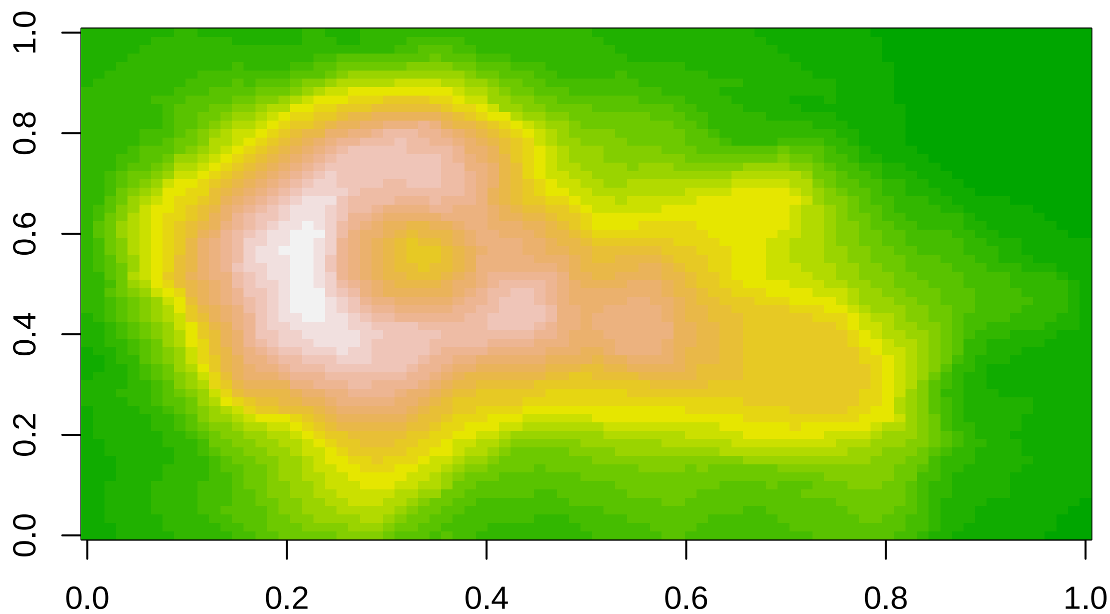


The mode of the distribution appears to happen at age 23. Note that the mass of the distribution is concentrated after the mode, as this graph has a long right tail. Therefore, we see more older sprinters.

Maungawhau volcano and heatmaps

- **6a.** The `volcano` object in R is a matrix of dimension 87 x 61. It is a digitized version of a topographic map of the Maungawhau volcano in Auckland, New Zealand. Plot a heatmap of the volcano using `image()`, with 25 colors from the terrain color palette.

```
image(volcano, col = terrain.colors(25))
```

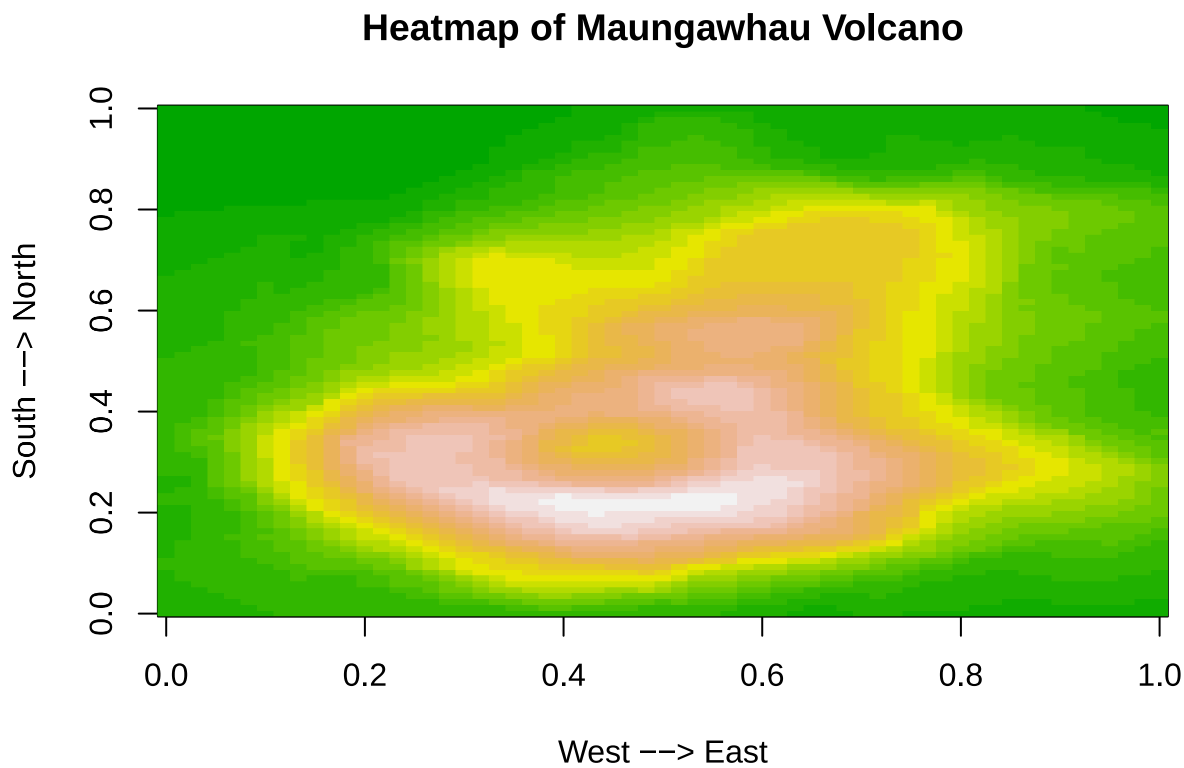


- **6b.** Each row of `volcano` corresponds to a grid line running east to west. Each column of `volcano` corresponds to a grid line running south to north. Define a matrix `volcano.rev` by reversing the order of the rows, as well as the order of the columns, of `volcano`. Therefore, each row `volcano.rev` should now correspond to a grid line running west to east, and each column of `volcano.rev` a grid line running north to south.

```
volcano <- volcano[order(nrow(volcano):1),]
volcano <- volcano[,order(ncol(volcano):1)]
```

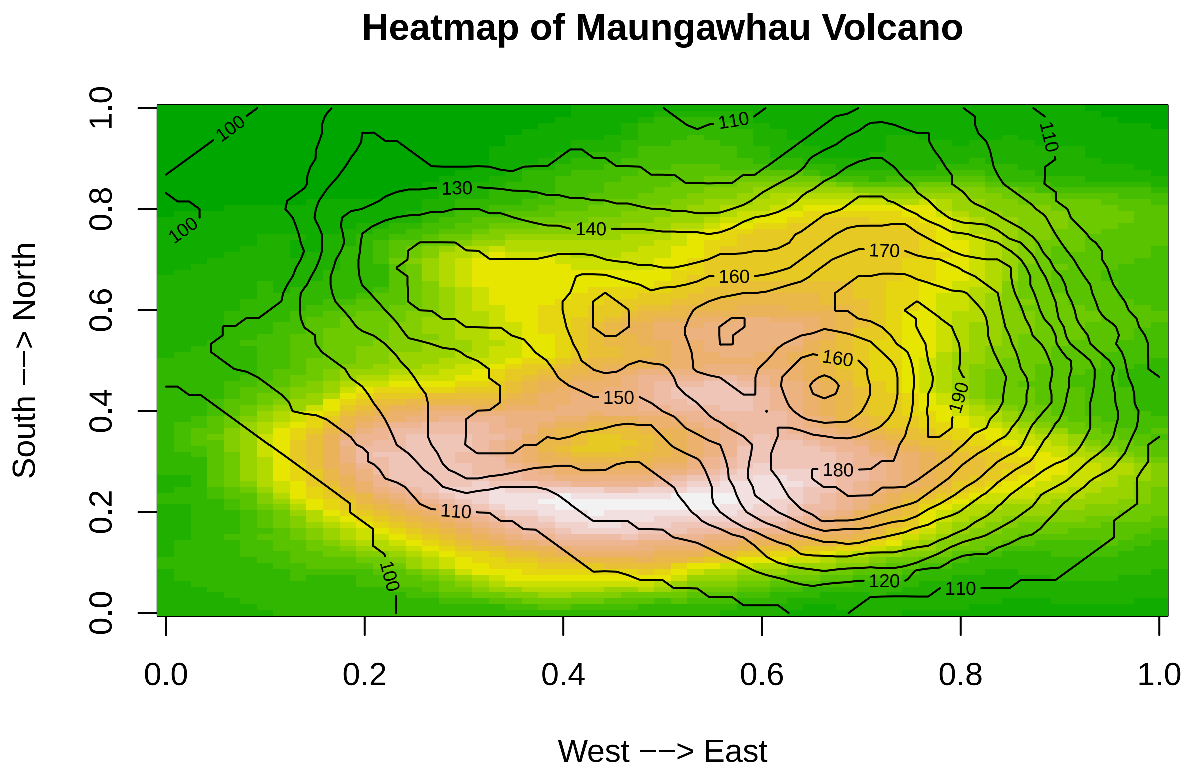
- **6c.** If we printed out the matrix `volcano.rev` to the console, then the elements would follow proper geographic order: left to right means west to east, and top to bottom means north to south. Now, produce a heatmap of the volcano that follows the same geographic order. Hint: recall that the `image()` function rotates a matrix 90 degrees counterclockwise before displaying it; and recall the function `clockwise90()` from the lecture, which you can copy and paste into your code here. Label the x-axis "West –> East", and the y-axis "South –> North". Title the plot "Heatmap of Maungawhau volcano".

```
clockwise90 = function(a) { t(a[nrow(a):1,]) }
image(clockwise90(volcano), col = terrain.colors(25),
      xlab = "West --> East", ylab = "South --> North", main = "Heatmap of Maungawhau Volcano")
```



- **6d.** Reproduce the previous plot, and now draw contour lines on top of the heatmap.

```
image(clockwise90(volcano), col = terrain.colors(25),
      xlab = "West --> East", ylab = "South --> North", main = "Heatmap of Maungawhau Volcano")
contour(volcano, add=TRUE)
```



Heatmap of Maungawhau Volcano
South --> North
West --> East
0.0
0.2
0.4
0.6
0.8
1.0
100
110
120
130
140
150
160
170
180
190