Homework 3

Advanced Statistical Computing (STAT 6984)

Sumin Shen

Problem 2: Asset returns (25 pts)

Return to the `dow30` data from Yahoo! Finance that we collected in class.

a. Consider the `getSymbols` function from the `quantmod` package for R.
   i. How can it be used in a fashion similar to our `get.quotes` and `get.multiple.quotes` functions?

   ii. By inspecting the source, describe how it handles the "crumb issue" we encountered?

   iii. Provide the commands (using `getSymbols`) which duplicate the result of our `get.multiple.quotes` on the `dow30.tickers` from class.

The commands for the function getSymbols are shown below:

```
library(quantmod)
GE = getSymbols(Symbols = 'GE', verbose=TRUE, src='yahoo', auto.assign = FALSE
                , from = (Sys.Date()-365), to=(Sys.Date()), periodicity=c("monthly")
                #, data.type = "csv"
               , return.class = "xts" )

dow30_tickers <-
  c("MMM", "AXP", "AAPL", "BA", "CAT", "CVX","CSCO", "KO",
    "DIS", "XOM", "GE", "GS", "HD", "IBM", "INTC", "JNJ",
    "JPM", "MCD", "MRK", "MSFT", "NKE", "PFE", "PG", "TRV",
    "UTX", "UNH", "VZ", "V", "WMT", "DWDP")

dow30 = getSymbols(Symbols = dow30_tickers, verbose=TRUE, src='yahoo'
                   , auto.assign = TRUE, from = (Sys.Date()-365), to=(Sys.Date())
                   , periodicity=c("monthly"))
# MMM
```

In the src file, the way it handles the crumb issue is that :

It uses a random query to avoid cache, e.g., the random URL is https://finance.yahoo.com?dogORcat , then use this random query to download by the function curl_download through a new curl_handle, get the crumb through the function curl_fetch_memory in the downloaded file, and pass the crumb to the URL.

b. With the data you downloaded, via the old `get.quotes` version or the new `getSymbols` (as you prefer), use the `transform` function to create a new `"Mid"` column tabulating the average between the `"Low"` and `"High"` quotes.

```
source("./quotesYahooF.R")
## get the tickers
dow30.tickers <-
  c("MMM", "AXP", "AAPL", "BA", "CAT", "CVX","CSCO", "KO",
    "DIS", "XOM", "GE", "GS", "HD", "IBM", "INTC", "JNJ",
    "JPM", "MCD", "MRK", "MSFT", "NKE", "PFE", "PG", "TRV",
    "UTX", "UNH", "VZ", "V", "WMT", "DWDP")
Sys.Date()

## you need to get the crumb from the browser
dow30 <- get.multiple.quotes(dow30.tickers, crumb="26SqzcW0nJE")

saveRDS(dow30,"./dow30.RDS")
```

c. Calculate *returns* $r_{a,t}$ for each asset a based on the `"Mid"` price $p_{a,t}$ on each day t: $r_{a,t} = (p_{a,t} - p_{a,t-1})/p_{a,t-1}$.

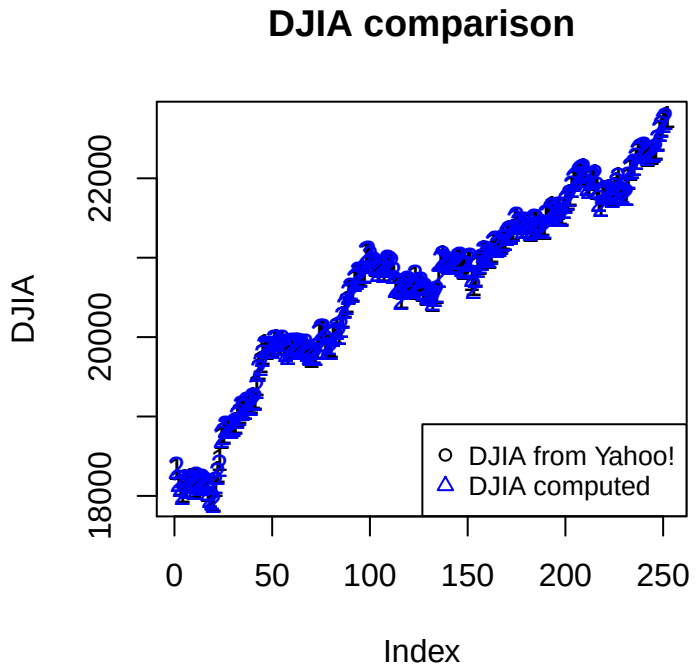
The summary of the calculated return is given:

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
## -0.091470 -0.003758  0.000564  0.000704  0.004932  0.089946         30
```

Note: NA is used for the first day return.

d. Use those same `"Mid"` prices to calculate the Dow Jones Industrial Average (DJIA) using the single "divisor" provided on that page. Compare your calculation to a DJIA index downloaded from Yahoo!

The Dow Divisor was 0.14523396877348 on September 1, 2017. Figure show the DJIA downloaded from Yahoo! and DJIA computed from mid price. These two lines almost overlap each other.



Summary for the return calculated from the DJIA is given below:

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
## -0.010519 -0.001493  0.000685  0.000868  0.002953  0.020305         30
```

e. The DJIA is often criticized for being a simple "price-weighted" index, meaning that the divisor is common to all assets. Create an alternative market-capitalization weighted version, more like the S&P500. For bonus points, write a script to scrape the "market caps" from Google or Yahoo. (Otherwise do it by hand.) Calculate returns. Finally, put a `data.frame` together with your all of your DJIA prices and returns and provide a visualization comparing the two.

The market caps from Yahoo is calculated by the multiplication of mid stock price and the shares outstanding, where the shares outstanding is copied from Yahoo!.

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
## -0.091470 -0.003758  0.000564  0.000704  0.004932  0.089946         30
```

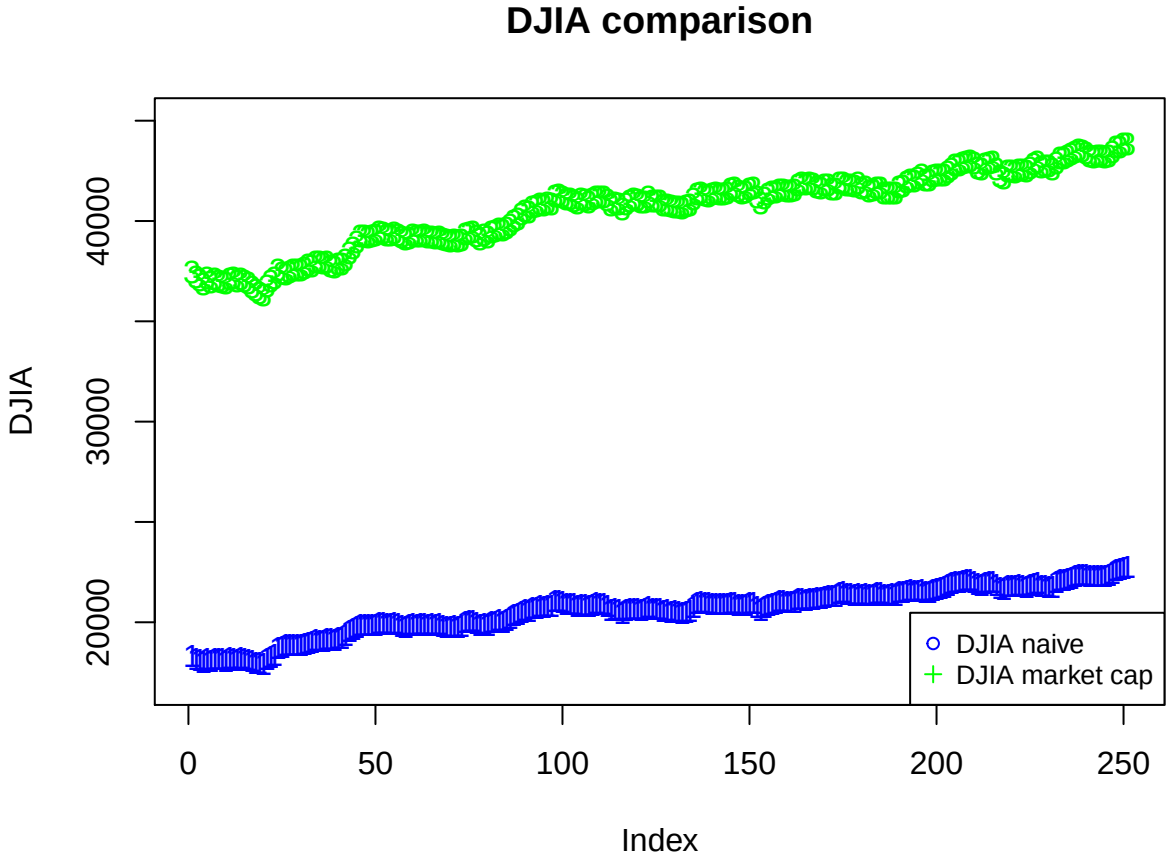
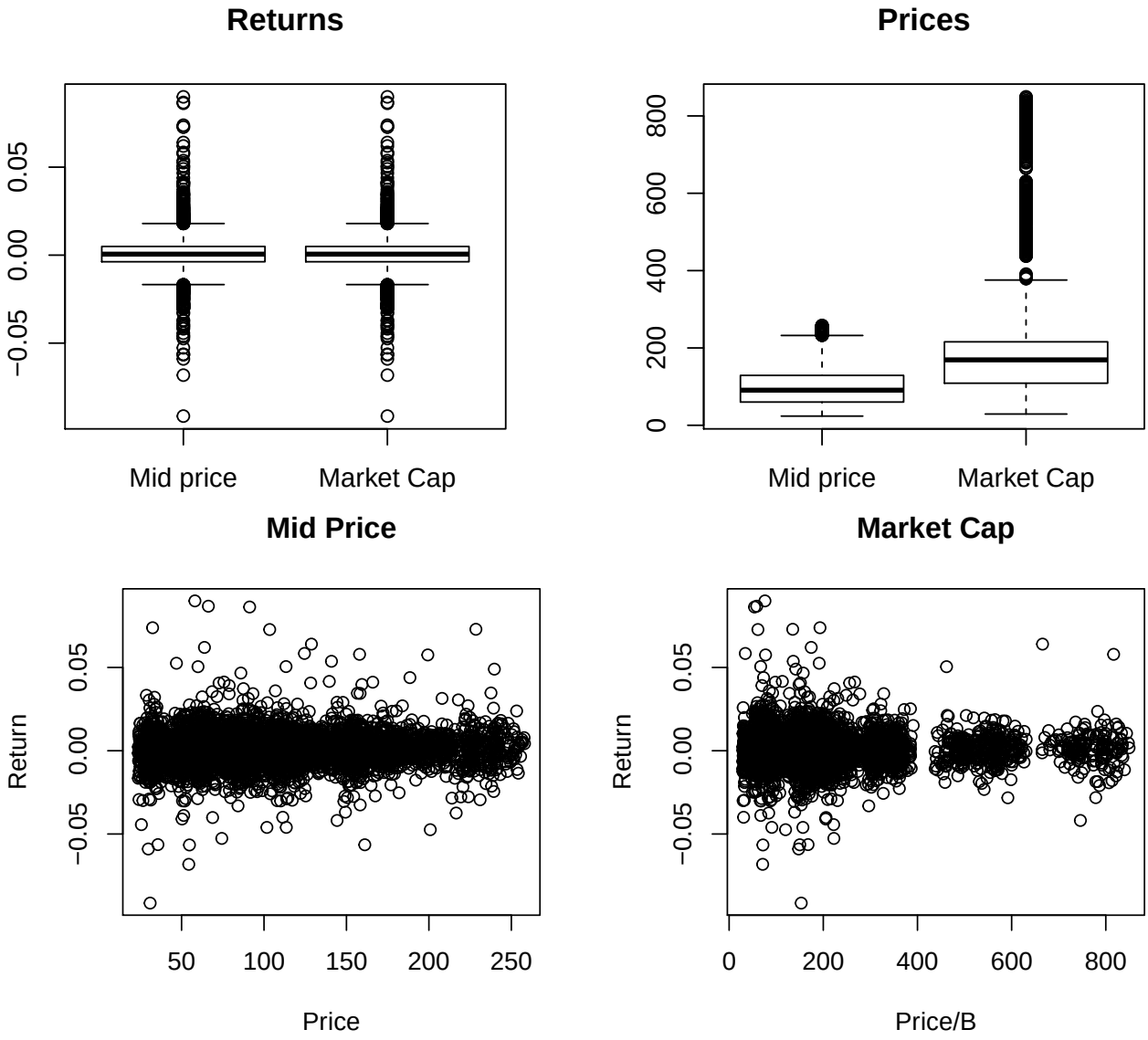


Figure shows that the alternative market-capitalization weighted version DJIA is much larger than the naive DJIA.



Note: the unit of Prices in Market Cap are Billion.

The returns from the mid price and market cap is the same because the market cap is computed from the multiplication of the mid price and the constant share outstanding.

f. Calculate the correlation of each stock's returns to those of the market, as measured (separately) by your two DJIA return calculations. Create a new data set where the stocks are presented after sorting on these correlations.

```
##    unique.dow30tt.symbol. cor_DJIA_price cor_DJIAw_price
## 12                     GS      0.7093911               1
## 17                    JPM      0.6889896               1
## 30                   DWDP      0.6020057               1
## 4                      BA      0.5683948               1
## 25                    UTX      0.5570112               1
## 5                     CAT      0.5508904               1
## 1                     MMM      0.4975067               1
```

```
## 14          IBM      0.4972420             1
## 2           AXP      0.4887704             1
## 13           HD      0.4843609             1
## 11           GE      0.4819619             1
## 7          CSCO      0.4816337             1
## 10          XOM      0.4718848             1
## 24          TRV      0.4476057             1
## 28            V      0.4431586             1
## 15         INTC      0.4168199             1
## 6           CVX      0.4141318             1
## 26          UNH      0.4006677             1
## 22          PFE      0.3825417             1
## 9           DIS      0.3565549             1
## 20         MSFT      0.3356186             1
## 21          NKE      0.3243568             1
## 3          AAPL      0.3220325             1
## 19          MRK      0.3024810             1
## 18          MCD      0.2769896             1
## 16          JNJ      0.2520662             1
## 29          WMT      0.2322019             1
## 27           VZ      0.1830734             1
## 8            KO      0.1707775             1
## 23           PG      0.1427670             1
```

Problem 3: Data visualization (30 pts)

Use charts to explore the 1970s UK teacher's pay survey data in the `teach.csv` available on the course web page.

- Info on the data columns is available in the CSV file.

What to show and how to show it is up to you. However, you may wish to (at the very least) consider the following.

i. Plot `salary` v. the number of `months` in service, while also indicating `sex`. What do you find?

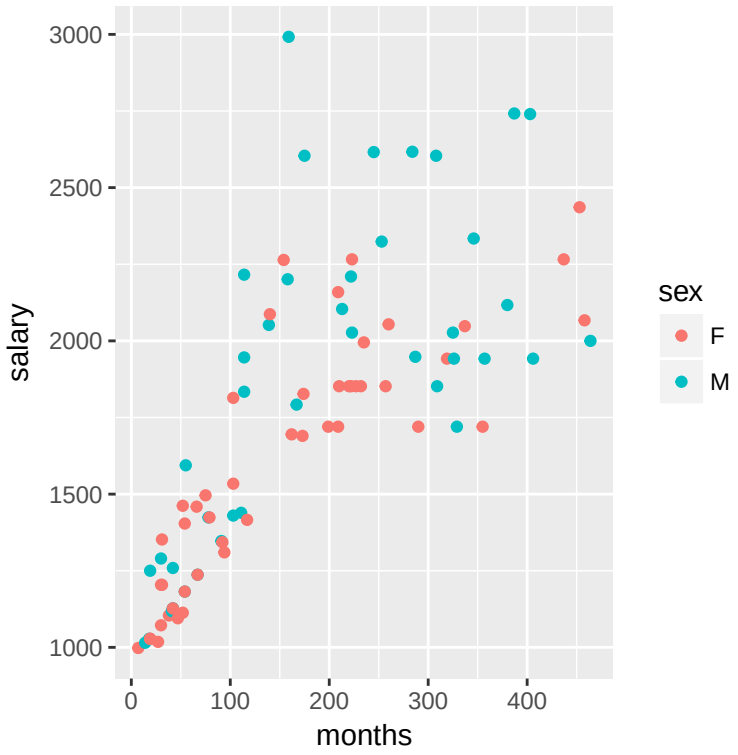
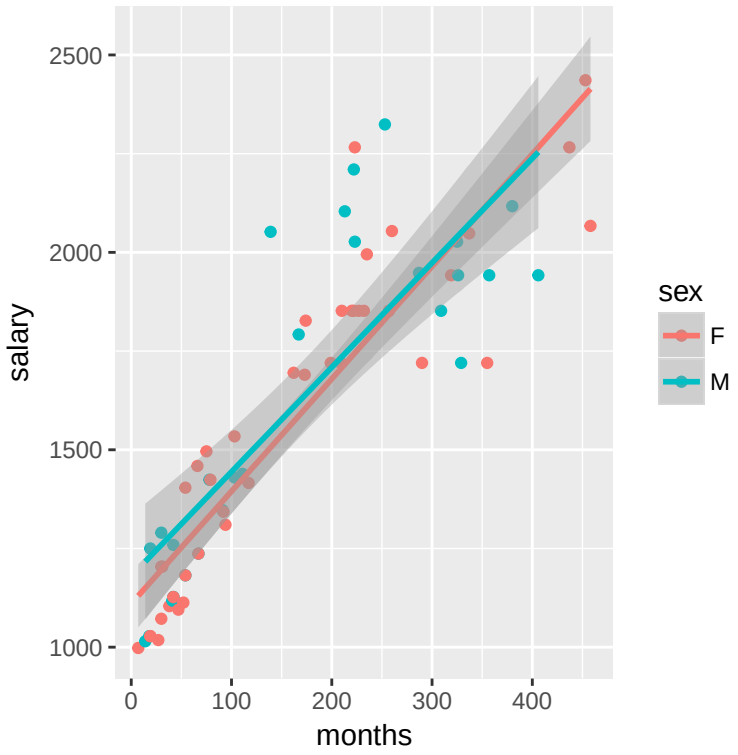


Figure shows that salary increases as the months in service increases. Females have shorter months in service than males.

ii. Ignore `months` and use `boxplots` to explore the distribution of each of the levels of `sex`, `marry`, `degree`, `type`, `train`, and `break`.

iii. Consider the part of the data for teachers whose school offers a `degree` of type `"0"`, and revisit i. Plot and try a augmenting with the best fitting regression line/intervals.



Note that this question is about visualization, not statistical modeling. For full credit your plots must be properly annotated with informative axes, labels, legends, main titles, etc., with accompanying verbal descriptions and interpretations in prose.

Problem 4: Processing web data (35 pts)

Hockey season is starting and it is time to get excited! There are many outlets on the web that let you keep track league standings. We will use hockey-reference.com. Click on that link and have a look.

- The regular season starts next week, so it is pretty empty at the moment.
- Swap in 2017 for 2018 in the URL to have a look at last season's version.
- To read these table I suggest looking into `htmltab` in a package of the same name on CRAN.

Your task is to produce league standings tables from this data. For an example see the middle panel of the main page. However you must augment tables like those with columns to include goals for, goals against, and goal differential.

- You will get something very similar, augmented with that extra goals information, if you Google "NHL standings".
- A file linked from the course web page provides conference and division information for each team, including a three-letter team name abbreviation. Use the `merge` function to merge them with the games data.

Here are some details on how you must present your standings.

a. You must provide a version of the table within your PDF homework solutions submitted on Canvas, which may be rendered (i.e., is up-to-date) at the time of submission.

b. More importantly, you must also provide a bash script called 'nhlstandings.sh' which I can call at any time after the time of submission and which will provide an up-to-date summary. That bash script should print a text version of the standings to the terminal and nothing else (i.e., no other R or bash messages).

In addition to being segmented by conference and league (you might find `aggregate` to be of assistance here), your teams must be properly sorted, first by points (higher is better) and then by losses (lower is better), then by wins (higher is better), then by goal differential (higher better).

- Teams get 2 points for a win, whether in regulation or overtime (OT), or via shoot-out (SO), and 1 point for an OT or SO loss. Otherwise zero points.
- Note that the goals number reported for the winning team in SO win is one greater than the actual number of goals scored for the purposes of counting goals in your table.
- You do not need anything else, i.e., no indicators of playoff clinch, etc.

For extra credit, design your script `nhlstandings.sh` to take an optional argument allowing the specification of one of the following

- three-letter team abbreviation
- conference name ("eastern", "western")
- division name ("atlantic", "metropolitan", "central", "pacific")

which will show only the relevant portion of the table. Include the entire devision, but not the whole conference, if a team abbreviation is provided.

```
##                    Team acronym conference     division GP W L OTL Pts GF
## 3    Toronto Maple Leafs     tor    eastern     atlantic  1 1 0   0   2  7
## 7      Detroit Red Wings     det    eastern     atlantic  1 1 0   0   2  4
## 1     Montreal Canadiens     mtl    eastern     atlantic  1 1 0   0   2  3
## 4          Boston Bruins     bos    eastern     atlantic  1 1 0   0   2  4
## 8         Buffalo Sabres     buf    eastern     atlantic  1 0 1   0   1  2
## 2        Ottawa Senators     ott    eastern     atlantic  0 0 0   0   0  0
## 5    Tampa Bay Lightning     tbl    eastern     atlantic  0 0 0   0   0  0
## 6       Florida Panthers     fla    eastern     atlantic  0 0 0   0   0  0
## 14   Philadelphia Flyers     phi    eastern metropolitan  2 1 1   0   2  5
## 10   Pittsburgh Penguins     pit    eastern metropolitan  2 0 2   1   1  5
## 9    Washington Capitals     was    eastern metropolitan  0 0 0   0   0  0
## 11 Columbus Blue Jackets     cbj    eastern metropolitan  0 0 0   0   0  0
## 12      New York Rangers     nyr    eastern metropolitan  0 0 0   0   0  0
## 13    New York Islanders     nyi    eastern metropolitan  0 0 0   0   0  0
## 15   Carolina Hurricanes     car    eastern metropolitan  0 0 0   0   0  0
## 16     New Jersey Devils     njd    eastern metropolitan  0 0 0   0   0  0
## 17    Chicago Blackhawks     chi    western      central  1 1 0   0   2 10
## 19       St. Louis Blues     stl    western      central  1 1 0   0   2  5
## 22          Dallas Stars     dal    western      central  0 0 0   0   0  0
## 23    Colorado Avalanche     col    western      central  0 0 0   0   0  0
## 20   Nashville Predators     nas    western      central  1 0 1   0   0  3
## 18        Minnesota Wild     min    western      central  1 0 1   0   0  2
## 21         Winnipeg Jets     wpg    western      central  1 0 1   0   0  2
## 25       Edmonton Oilers     edm    western      pacific  1 1 0   0   2  3
## 28     Los Angeles Kings     lak    western      pacific  1 1 0   0   2  2
```

```
## 24         Anaheim Ducks     ana    western      pacific  1 1 0   0   2  5
## 30     Vancouver Canucks     van    western      pacific  0 0 0   0   0  0
## 31  Vegas Golden Knights     vgk    western      pacific  0 0 0   0   0  0
## 29       Arizona Coyotes     ari    western      pacific  1 0 1   0   0  4
## 26       San Jose Sharks     sjs    western      pacific  1 0 1   0   0  3
## 27        Calgary Flames     cal    western      pacific  1 0 1   0   0  0
##    GA Diff
## 3   2    5
## 7   2    2
## 1   2    1
## 4   3    1
## 8   3   -1
## 2   0    0
## 5   0    0
## 6   0    0
## 14  5    0
## 10 15  -10
## 9   0    0
## 11  0    0
## 12  0    0
## 13  0    0
## 15  0    0
## 16  0    0
## 17  1    9
## 19  4    1
## 22  0    0
## 23  0    0
## 20  4   -1
## 18  4   -2
## 21  7   -5
## 25  0    3
## 28  0    2
## 24  4    1
## 30  0    0
## 31  0    0
## 29  5   -1
## 26  5   -2
## 27  3   -3
```

```
## [1] "/home/shen/Documents/VT/Git/asc_wind/hwk/hw3"
```

```
## Session info -------------------------------------------------------------
```

```
##  setting  value
##  version  R version 3.4.1 (2017-06-30)
##  system   x86_64, linux-gnu
##  ui       X11
##  language en_US
##  collate  en_US.UTF-8
##  tz       America/New_York
##  date     2017-10-10
```

```
## Packages -----------------------------------------------------------------
```

```
##  package    * version  date       source
##  backports    1.1.0    2017-05-22 CRAN (R 3.4.1)
##  base       * 3.4.1    2017-07-08 local
##  colorspace   1.3-2    2016-12-14 CRAN (R 3.4.1)
##  compiler     3.4.1    2017-07-08 local
##  curl         2.8.1    2017-07-21 CRAN (R 3.4.1)
##  datasets   * 3.4.1    2017-07-08 local
##  devtools     1.13.3   2017-08-02 CRAN (R 3.4.1)
##  digest       0.6.12   2017-01-27 CRAN (R 3.4.1)
##  evaluate     0.10.1   2017-06-24 CRAN (R 3.4.1)
##  ggplot2    * 2.2.1    2016-12-30 CRAN (R 3.4.1)
##  graphics   * 3.4.1    2017-07-08 local
##  grDevices  * 3.4.1    2017-07-08 local
##  grid       * 3.4.1    2017-07-08 local
##  gtable       0.2.0    2016-02-26 CRAN (R 3.4.1)
##  htmltab    * 0.7.1    2016-12-29 CRAN (R 3.4.1)
##  htmltools    0.3.6    2017-04-28 CRAN (R 3.4.1)
##  httr         1.2.1    2016-07-03 CRAN (R 3.4.1)
##  knitr        1.16     2017-05-18 CRAN (R 3.4.1)
##  labeling     0.3      2014-08-23 CRAN (R 3.4.1)
##  lazyeval     0.2.0    2016-06-12 CRAN (R 3.4.1)
##  magrittr     1.5      2014-11-22 CRAN (R 3.4.1)
##  memoise      1.1.0    2017-04-21 CRAN (R 3.4.1)
##  methods    * 3.4.1    2017-07-08 local
##  munsell      0.4.3    2016-02-13 CRAN (R 3.4.1)
##  plyr         1.8.4    2016-06-08 CRAN (R 3.4.1)
##  R6           2.2.2    2017-06-17 CRAN (R 3.4.1)
##  Rcpp         0.12.12  2017-07-15 CRAN (R 3.4.1)
##  rlang        0.1.1    2017-05-18 CRAN (R 3.4.1)
##  rmarkdown    1.6      2017-06-15 CRAN (R 3.4.1)
##  rprojroot    1.2      2017-01-16 CRAN (R 3.4.1)
##  scales       0.4.1    2016-11-09 CRAN (R 3.4.1)
##  stats      * 3.4.1    2017-07-08 local
##  stringi      1.1.5    2017-04-07 CRAN (R 3.4.1)
##  stringr      1.2.0    2017-02-18 CRAN (R 3.4.1)
```

```
##  tibble       1.3.3    2017-05-28 CRAN (R 3.4.1)
##  tools        3.4.1    2017-07-08 local
##  utils      * 3.4.1    2017-07-08 local
##  withr        2.0.0    2017-07-28 CRAN (R 3.4.1)
##  XML          3.98-1.9 2017-06-19 CRAN (R 3.4.1)
##  yaml         2.1.14   2016-11-12 CRAN (R 3.4.1)
```