Lab 7

Janine Lim

11:59PM April 15, 2021

#Rcpp

We will get some experience with speeding up R code using C++ via the `Rcpp` package.

First, clear the workspace and load the `Rcpp` package.

```
rm(list=ls())
pacman::p_load(Rcpp)
```

Create a variable `n` to be 10 and a vaiable `Nvec` to be 100 initially. Create a random vector via `rnorm` `Nvec` times and load it into a `Nvec` x `n` dimensional matrix.

```
n=10
Nvec = 100
X= matrix(data = rnorm(Nvec*n), nrow = Nvec)
```

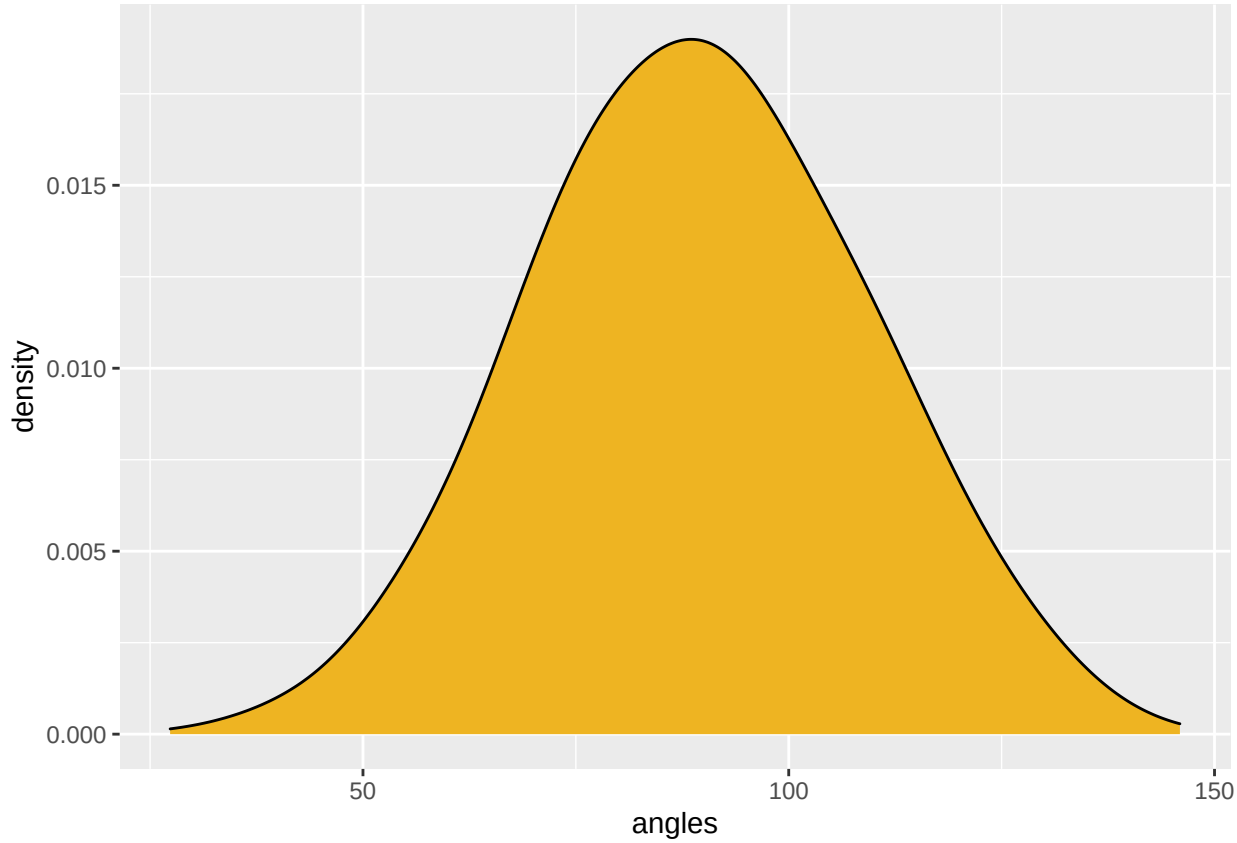
Write a function `all_angles` that measures the angle between each of the pairs of vectors. You should measure the vector on a scale of 0 to 180 degrees with negative angles coerced to be positive.

```
angle = function(u,v){
  acos(sum(u*v)/sqrt(sum(u^2)*sum(v^2))) * (180/pi)
}
all_angles = function(X){
  A = matrix(NA, nrow=nrow(X), ncol=nrow(X))
  for(i in 1:(nrow(X)-1)){
    for(j in (i+1):nrow(X)){
      A[i,j] = angle(X[i,],X[j,])
    }
  }
  A
}
```

Plot the density of these angles.

```
pacman::p_load(ggplot2)
ggplot(data.frame(angles=c(all_angles(X)))) +
  aes(x = angles) +
  geom_density(adjust = 2, fill = "goldenrod2")
```

```
## Warning: Removed 5050 rows containing non-finite values (stat_density).
```



Write an Rcpp function `all_angles_cpp` that does the same thing. Use an IDE if you want, but write it below in-line.

```
cppFunction('
  NumericMatrix all_angles_cpp(NumericMatrix X) {
    int n = X.nrow();
    int p = X.ncol();
    NumericMatrix A(n, n);
    std::fill(A.begin(), A.end(), NA_REAL);
    for (int i_1 = 0; i_1 < (n - 1); i_1++){
      //Rcout << "computing for row #: " << (i_1 + 1) << "\\n";
      for (int i_2 = i_1 + 1; i_2 < n; i_2++){
        double sum_sqd_u = 0;
        double sum_sqd_v = 0;
        double sum_u_v = 0;

        for (int j = 0; j < p; j++){

          //sqd_diff += pow(X(i_1, j) - X(i_2, j), 2); //by default the cmath library in std is loaded

          sum_sqd_u += pow(X(i_1, j), 2);
          sum_sqd_v += pow(X(i_2, j), 2);
          sum_u_v += X(i_1, j) * X(i_2, j);


        }
```

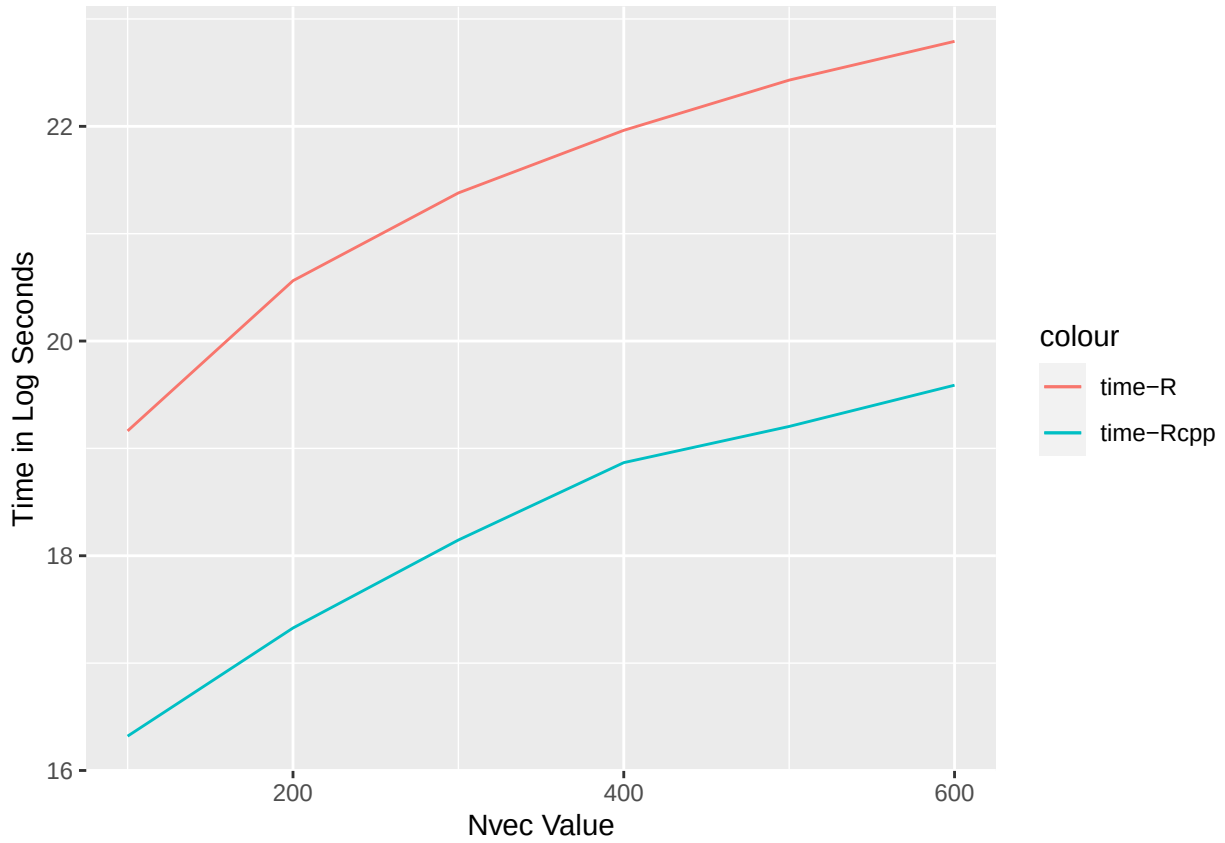
```
        A(i_1, i_2) = acos(sum_u_v / sqrt(sum_sqd_u * sum_sqd_v)) * (180 / M_PI); //by default the cmath
      }
    }
    return A;
  }
')
```

Test the time difference between these functions for `n = 1000` and `Nvec = 100, 500, 1000, 5000` using the package `microbenchmark`. Store the results in a matrix with rows representing `Nvec` and two columns for base R and Rcpp.

```
pacman::p_load(microbenchmark)
n <- 1000
Nvec <- c(100, 200, 300, 400, 500, 600)
time_for_r <- c()
time_for_cpp <- c()
for (i in 1:length(Nvec)){
  X <- c()
  for (j in 1:n){
    x <- rnorm(Nvec[i])
    X <- cbind(X, x)
  }
  time_for_r <- c(time_for_r, mean(microbenchmark(all_angles_r = all_angles(X), times = 3, unit = "s")$
  time_for_cpp <- c(time_for_cpp, mean(microbenchmark(all_angles_cpp = all_angles_cpp(X), times = 3, uni
}
```

Plot the divergence of performance (in log seconds) over Nvec using a line geometry. Use two different colors for the R and CPP functions. Make sure there's a color legend on your plot. We wil see later how to create "long" matrices that make such plots easier.

```
pacman::p_load(ggplot2)
ggplot() +
  geom_line(aes(x = Nvec, y = log(time_for_r), col = "time-R")) +
  geom_line(aes(x = Nvec, y = log(time_for_cpp), col = "time-Rcpp")) +
  xlab("Nvec Value") +
  ylab("Time in Log Seconds")
```



Let `Nvec = 10000` and vary `n` to be 10, 100, 1000. Plot the density of angles for all three values of `n` on one plot using color to signify `n`. Make sure you have a color legend. This is not easy.

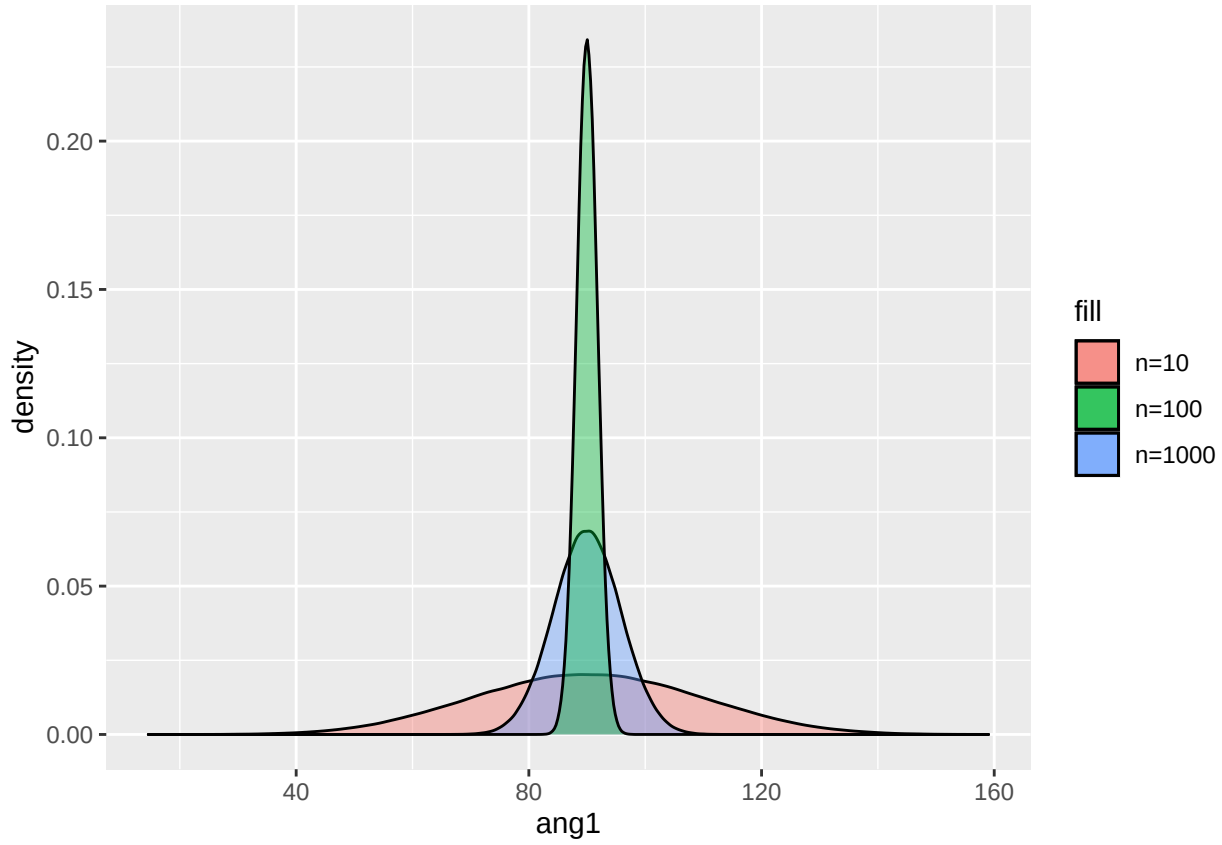
```
Nvec = 1000 #Nvec=10000 took too long to run i'm sorry
X <- c()
for (i in 1:10){
  y <- rnorm(Nvec)
  X <- cbind(X, y)
}
ang1 <- all_angles(X)
X <- c()
for (i in 1:100){
  y <- rnorm(Nvec)
  X <- cbind(X, y)
}
ang2 <- all_angles(X)
X <- c()
for (i in 1:1000){
  y <- rnorm(Nvec)
  X <- cbind(X, y)
}
ang3 <- all_angles(X)

ggplot() +
  geom_density(aes(x = ang1, fill = "darkorchid3"), alpha = .4) +
  geom_density(aes(x = ang2, fill = "goldenrod"), alpha = .4) +
```

```
  geom_density(aes(x = ang3, fill = "darkseagreen2"), alpha = .4) +
  scale_fill_discrete(labels = c("n=10", "n=100", "n=1000"))
```

```
## Warning: Removed 500500 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 500500 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 500500 rows containing non-finite values (stat_density).
```



Write an R function `nth_fibonnaci` that finds the nth Fibonnaci number via recursion but allows you to specify the starting number. For instance, if the sequency started at 1, you get the familiar 1, 1, 2, 3, 5, etc. But if it started at 0.01, you would get 0.01, 0.01, 0.02, 0.03, 0.05, etc.

```
nth_fibonacci = function(x,n){
  fib_sequence = array(data = NA, n)
  for (i in 2:n){
    fib_sequence[1] = x
    fib_sequence[2] = x
    fib_sequence[i+1] = sum(fib_sequence[i],fib_sequence[i-1])
  }
  fib_sequence[i]
}

nth_fibonacci(1, 21)
```

```
## [1] 10946
```

Write an Rcpp function `nth_fibonnaci_cpp` that does the same thing. Use an IDE if you want, but write it below in-line.

```
cppFunction(
  'double nth_fibonacci_cpp(int n, double start){
    if (n == 1 || n == 2) return start;
    else return (nth_fibonacci_cpp(n-1, start) + nth_fibonacci_cpp(n-2, start));
  }'
)
nth_fibonacci_cpp(21, 1)
```

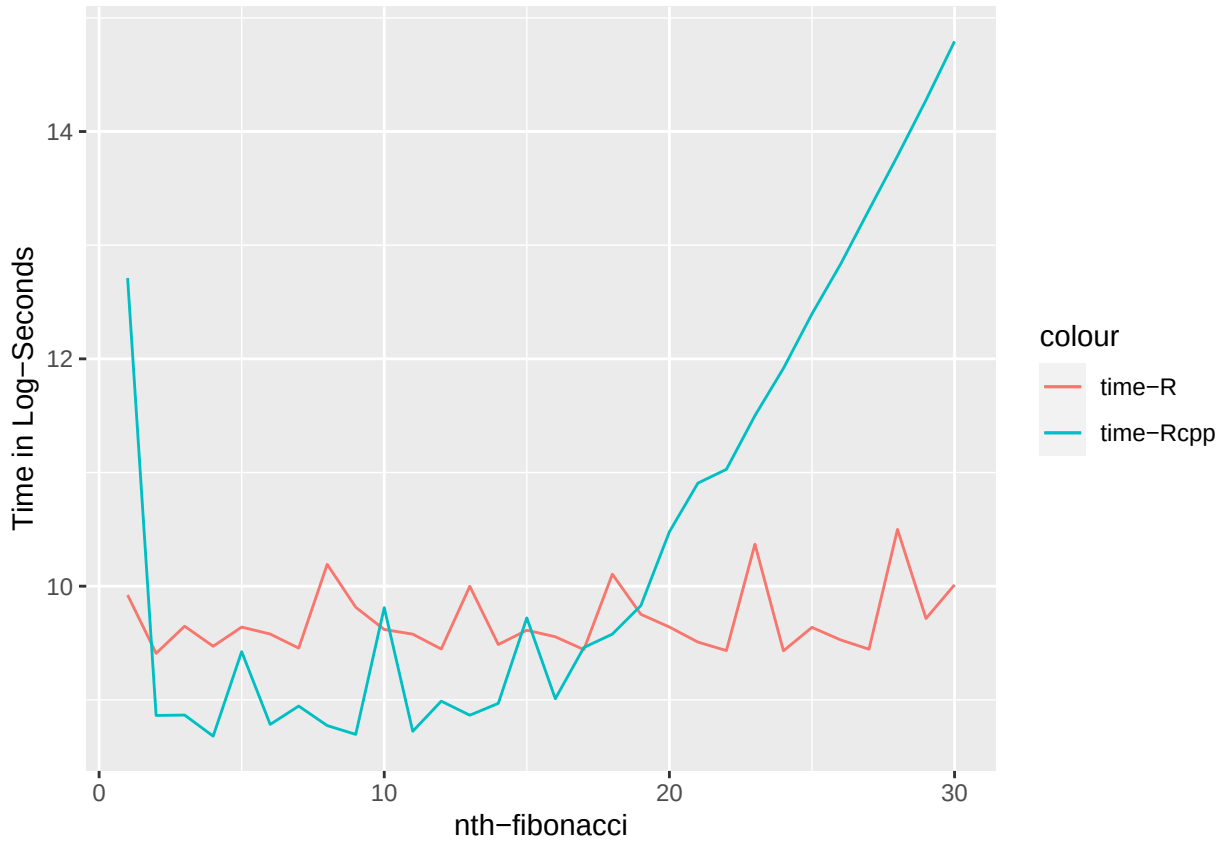
```
## [1] 10946
```

Time the difference in these functions for n = 100, 200, ..., 1500 while starting the sequence at the smallest possible floating point value in R. Store the results in a matrix.

```
n = 30 #could not run n higher
time_fib_r <- c()
time_fib_cpp <- c()
for (i in 1:n){
  time_fib_r <- c(time_fib_r, mean(microbenchmark(fib_r = nth_fibonacci(i, .Machine$double.xmin), times
  time_fib_cpp <- c(time_fib_cpp, mean(microbenchmark(fib_cpp = nth_fibonacci_cpp(i, .Machine$double.xm
}
```

Plot the divergence of performance (in log seconds) over n using a line geometry. Use two different colors for the R and CPP functions. Make sure there's a color legend on your plot.

```
pacman::p_load(ggplot2)
ggplot() +
  geom_line(aes(x = 1:n, y = log(time_fib_r), col = "time-R")) +
  geom_line(aes(x = 1:n, y = log(time_fib_cpp), col = "time-Rcpp")) +
  ylab("Time in Log-Seconds") +
  xlab("nth-fibonacci")
```



Data Wrangling / Munging / Carpentry

Throughout this assignment you can use either the `tidyverse` package suite or `data.table` to answer but not base R. You can mix `data.table` with `magrittr` piping if you wish but don't go back and forth between `tbl_df`'s and `data.table` objects.

```
pacman::p_load(tidyverse, magrittr, data.table)
```

Load the `storms` dataset from the `dplyr` package and investigate it using `str` and `summary` and `head`. Which two columns should be converted to type factor? Do so below.

```
data(storms)
str(storms)
```

```
## tibble[,13] [10,010 x 13] (S3: tbl_df/tbl/data.frame)
##  $ name       : chr [1:10010] "Amy" "Amy" "Amy" "Amy" ...
##  $ year       : num [1:10010] 1975 1975 1975 1975 1975 ...
##  $ month      : num [1:10010] 6 6 6 6 6 6 6 6 6 6 ...
##  $ day        : int [1:10010] 27 27 27 27 28 28 28 28 29 29 ...
##  $ hour       : num [1:10010] 0 6 12 18 0 6 12 18 0 6 ...
##  $ lat        : num [1:10010] 27.5 28.5 29.5 30.5 31.5 32.4 33.3 34 34.4 34 ...
##  $ long       : num [1:10010] -79 -79 -79 -79 -78.8 -78.7 -78 -77 -75.8 -74.8 ...
##  $ status     : chr [1:10010] "tropical depression" "tropical depression" "tropical depression" "tro
```

```
##  $ category   : Ord.factor w/ 7 levels "-1"<"0"<"1"<"2"<..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ wind       : int [1:10010] 25 25 25 25 25 25 25 30 35 40 ...
##  $ pressure   : int [1:10010] 1013 1013 1013 1013 1012 1012 1011 1006 1004 1002 ...
##  $ ts_diameter: num [1:10010] NA NA NA NA NA NA NA NA NA NA ...
##  $ hu_diameter: num [1:10010] NA NA NA NA NA NA NA NA NA NA ...
```

```
summary(storms)
```

```
##      name                year          month             day
##  Length:10010       Min.   :1975   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:1990   1st Qu.: 8.000   1st Qu.: 8.00
##  Mode  :character   Median :1999   Median : 9.000   Median :16.00
##                     Mean   :1998   Mean   : 8.779   Mean   :15.86
##                     3rd Qu.:2006   3rd Qu.: 9.000   3rd Qu.:24.00
##                     Max.   :2015   Max.   :12.000   Max.   :31.00
##
##       hour             lat             long            status
##  Min.   : 0.000   Min.   : 7.20   Min.   :-109.30   Length:10010
##  1st Qu.: 6.000   1st Qu.:17.50   1st Qu.: -80.70   Class :character
##  Median :12.000   Median :24.40   Median : -64.50   Mode  :character
##  Mean   : 9.114   Mean   :24.76   Mean   : -64.23
##  3rd Qu.:18.000   3rd Qu.:31.30   3rd Qu.: -48.60
##  Max.   :23.000   Max.   :51.90   Max.   :  -6.00
##
##  category       wind           pressure       ts_diameter       hu_diameter
##  -1:2545   Min.   : 10.00   Min.   : 882.0   Min.   :   0.00   Min.   :  0.00
##  0 :4373   1st Qu.: 30.00   1st Qu.: 985.0   1st Qu.:  69.05   1st Qu.:  0.00
##  1 :1685   Median : 45.00   Median : 999.0   Median : 138.09   Median :  0.00
##  2 : 628   Mean   : 53.49   Mean   : 992.1   Mean   : 166.76   Mean   : 21.41
##  3 : 363   3rd Qu.: 65.00   3rd Qu.:1006.0   3rd Qu.: 241.66   3rd Qu.: 28.77
##  4 : 348   Max.   :160.00   Max.   :1022.0   Max.   :1001.18   Max.   :345.23
##  5 :  68                                     NA's   :6528      NA's   :6528
```

```
head(storms)
```

```
## # A tibble: 6 x 13
##   name   year month   day  hour   lat  long status       category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>        <ord>    <int>    <int>
## 1 Amy    1975     6    27     0  27.5 -79   tropical de~ -1          25     1013
## 2 Amy    1975     6    27     6  28.5 -79   tropical de~ -1          25     1013
## 3 Amy    1975     6    27    12  29.5 -79   tropical de~ -1          25     1013
## 4 Amy    1975     6    27    18  30.5 -79   tropical de~ -1          25     1013
## 5 Amy    1975     6    28     0  31.5 -78.8 tropical de~ -1          25     1012
## 6 Amy    1975     6    28     6  32.4 -78.7 tropical de~ -1          25     1012
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

Reorder the columns so name is first, status is second, category is third and the rest are the same.

```
storms %>%
  select(name, status, category, everything())
```

```
## # A tibble: 10,010 x 13
```

```
##    name  status      category  year month   day  hour   lat  long  wind pressure
##    <chr> <chr>       <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
##  1 Amy   tropical d~ -1        1975     6    27     0  27.5 -79      25     1013
##  2 Amy   tropical d~ -1        1975     6    27     6  28.5 -79      25     1013
##  3 Amy   tropical d~ -1        1975     6    27    12  29.5 -79      25     1013
##  4 Amy   tropical d~ -1        1975     6    27    18  30.5 -79      25     1013
##  5 Amy   tropical d~ -1        1975     6    28     0  31.5 -78.8    25     1012
##  6 Amy   tropical d~ -1        1975     6    28     6  32.4 -78.7    25     1012
##  7 Amy   tropical d~ -1        1975     6    28    12  33.3 -78      25     1011
##  8 Amy   tropical d~ -1        1975     6    28    18  34   -77      30     1006
##  9 Amy   tropical s~ 0         1975     6    29     0  34.4 -75.8    35     1004
## 10 Amy   tropical s~ 0         1975     6    29     6  34   -74.8    40     1002
## # ... with 10,000 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

Find a subset of the data of storms only in the 1970's.

```
storms %>%
  dplyr::filter(year >= 1970 & year <= 1979)
```

```
## # A tibble: 546 x 13
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 536 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

Find a subset of the data of storm observations only with category 4 and above and wind speed 100MPH and above.

```
storms %>%
  dplyr::filter(category >= 4 & wind >=100)
```

```
## # A tibble: 416 x 13
##    name   year month   day  hour   lat  long status    category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>     <ord>    <int>    <int>
##  1 Anita  1977     9     2     0  24.6 -96.2 hurricane 5          140      931
##  2 Anita  1977     9     2     6  24.2 -97.1 hurricane 5          150      926
##  3 Anita  1977     9     2    12  23.7 -98   hurricane 4          120      940
##  4 David  1979     8    28     0  12.2 -52.9 hurricane 4          115      947
##  5 David  1979     8    28     6  12.5 -54.4 hurricane 4          125      941
##  6 David  1979     8    28    12  12.8 -55.7 hurricane 4          130      938
##  7 David  1979     8    28    18  13.2 -56.9 hurricane 4          125      941
```

```
##  8 David  1979     8    29     0  13.7 -58   hurricane 4          120      944
##  9 David  1979     8    29     6  14.2 -59.2 hurricane 4          120      942
## 10 David  1979     8    29    12  14.8 -60.3 hurricane 4          125      938
## # ... with 406 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

Create a new feature `wind_speed_per_unit_pressure`.

```
storms %>%
  mutate(wind_speed_per_unit_pressure = wind/pressure)
```

```
## # A tibble: 10,010 x 14
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 10,000 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>
```

Create a new feature: `average_diameter` which averages the two diameter metrics. If one is missing, then use the value of the one that is present. If both are missing, leave missing.

```
storms %>%
  rowwise() %>%
  arrange(desc(year)) %>%
  mutate(average_diamter = mean(c(ts_diameter, hu_diameter), na.rm=TRUE)) %>%
  mutate(average_diamter = ifelse(average_diamter ==0, NA, average_diamter)) #turned the 0's into NA va
```

```
## # A tibble: 10,010 x 14
## # Rowwise:
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Ana    2015     5     9     6  32.2 -77.5 tropical s~ 0           50      998
##  2 Ana    2015     5     9    12  32.5 -77.8 tropical s~ 0           50     1001
##  3 Ana    2015     5     9    18  32.7 -78   tropical s~ 0           45     1001
##  4 Ana    2015     5    10     0  33.1 -78.3 tropical s~ 0           45     1001
##  5 Ana    2015     5    10     6  33.5 -78.6 tropical s~ 0           40     1002
##  6 Ana    2015     5    10    10  33.8 -78.8 tropical s~ 0           40     1002
##  7 Ana    2015     5    10    12  33.9 -78.8 tropical s~ 0           35     1002
##  8 Ana    2015     5    10    18  34.3 -78.7 tropical d~ -1          30     1006
##  9 Ana    2015     5    11     0  34.7 -78.5 tropical d~ -1          30     1009
## 10 Ana    2015     5    11     6  35.5 -78   tropical d~ -1          30     1010
## # ... with 10,000 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, average_diamter <dbl>
```

For each storm, summarize the maximum wind speed. "Summarize" means create a new dataframe with only the summary metrics you care about.

```
storms %>%
  group_by(name) %>%
  summarize(max_windspeed = max(wind, na.rm= TRUE))
```

```
## # A tibble: 198 x 2
##    name     max_windspeed
##    <chr>            <int>
##  1 AL011993            30
##  2 AL012000            25
##  3 AL021992            30
##  4 AL021994            30
##  5 AL021999            30
##  6 AL022000            30
##  7 AL022001            25
##  8 AL022003            30
##  9 AL022006            45
## 10 AL031987            40
## # ... with 188 more rows
```

Order your dataset by maximum wind speed storm but within the rows of storm show the observations in time order from early to late.

```
storms %>%
  group_by(name) %>%
  mutate(max_wind_by_storm = max(wind, na.rm= TRUE)) %>%
  select(name, max_wind_by_storm, everything()) %>%
  arrange(desc(max_wind_by_storm), year, month, day, hour)
```

```
## # A tibble: 10,010 x 14
## # Groups:   name [198]
##    name   max_wind_by_sto~  year month   day  hour   lat  long status   category
##    <chr>             <int> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>    <ord>
##  1 Gilbe~              160  1988     9     8    18  12   -54   tropica~ -1
##  2 Gilbe~              160  1988     9     9     0  12.7 -55.6 tropica~ -1
##  3 Gilbe~              160  1988     9     9     6  13.3 -57.1 tropica~ -1
##  4 Gilbe~              160  1988     9     9    12  14   -58.6 tropica~ -1
##  5 Gilbe~              160  1988     9     9    18  14.5 -60.1 tropica~ 0
##  6 Gilbe~              160  1988     9    10     0  14.8 -61.5 tropica~ 0
##  7 Gilbe~              160  1988     9    10     6  15   -62.8 tropica~ 0
##  8 Gilbe~              160  1988     9    10    12  15.3 -64.1 tropica~ 0
##  9 Gilbe~              160  1988     9    10    18  15.7 -65.4 tropica~ 0
## 10 Gilbe~              160  1988     9    11     0  15.9 -66.8 hurrica~ 1
## # ... with 10,000 more rows, and 4 more variables: wind <int>, pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>
```

Find the strongest storm by wind speed per year.

```
storms %>%
  group_by(year) %>%
```

```
  arrange(year, desc(wind)) %>%
  slice(1) %>%
  select(name, year, wind)
```

```
## # A tibble: 41 x 3
## # Groups:   year [41]
##    name      year  wind
##    <chr>    <dbl> <int>
##  1 Caroline  1975   100
##  2 Belle     1976   105
##  3 Anita     1977   150
##  4 Cora      1978    80
##  5 David     1979   150
##  6 Ivan      1980    90
##  7 Harvey    1981   115
##  8 Debby     1982   115
##  9 Alicia    1983   100
## 10 Diana     1984   115
## # ... with 31 more rows
```

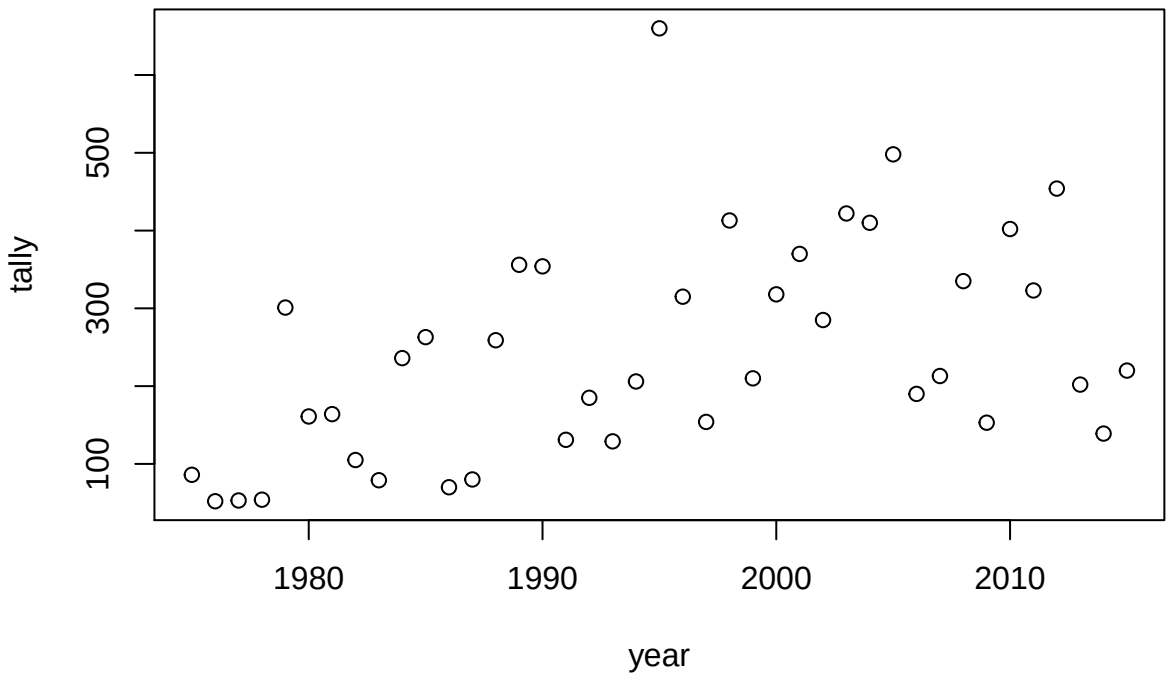
For each named storm, find its maximum category, wind speed, pressure and diameters. Do not allow the max to be NA (unless all the measurements for that storm were NA).

```
suppressWarnings(storms %>%
  group_by(name)  %>%
  summarise(max_category = max(category),
            max_wind_speed = max(wind),
            max_pressure = max(pressure),
            max_ts_diameter = max(ts_diameter, na.rm = TRUE),
            max_hu_diameter = max(hu_diameter, na.rm = TRUE)))
```

```
## # A tibble: 198 x 6
##    name     max_category max_wind_speed max_pressure max_ts_diameter
##    <chr>    <ord>                <int>        <int>           <dbl>
##  1 AL011993 -1                      30         1003          -Inf
##  2 AL012000 -1                      25         1010          -Inf
##  3 AL021992 -1                      30         1009          -Inf
##  4 AL021994 -1                      30         1017          -Inf
##  5 AL021999 -1                      30         1006          -Inf
##  6 AL022000 -1                      30         1010          -Inf
##  7 AL022001 -1                      25         1012          -Inf
##  8 AL022003 -1                      30         1010          -Inf
##  9 AL022006 0                       45         1008           69.0
## 10 AL031987 0                       40         1015          -Inf
## # ... with 188 more rows, and 1 more variable: max_hu_diameter <dbl>
```

For each year in the dataset, tally the number of storms. "Tally" is a fancy word for "count the number of". Plot the number of storms by year. Any pattern?

```
storms %>%
  group_by(year) %>%
  summarize(tally=n()) %>%
  plot
```



For each year in the dataset, tally the storms by category.

```
storms %>%
  group_by(year) %>%
  count(category)
```

```
## # A tibble: 233 x 3
## # Groups:   year [41]
##     year category     n
##    <dbl> <ord>    <int>
##  1  1975 -1          30
##  2  1975 0           33
##  3  1975 1           12
##  4  1975 2            9
##  5  1975 3            2
##  6  1976 -1          10
##  7  1976 0           20
##  8  1976 1           10
##  9  1976 2            9
## 10  1976 3            3
## # ... with 223 more rows
```

For each year in the dataset, find the maximum wind speed per status level.

```
storms %>%
  group_by(year, status) %>%
  mutate(max_wind = max(wind, na.rm = TRUE)) %>%
  arrange(year, status, desc(max_wind)) %>%
  select(year, status, max_wind) %>%
  distinct
```

```
## # A tibble: 123 x 3
## # Groups:   year, status [123]
##     year status              max_wind
##    <dbl> <chr>                  <int>
##  1  1975 hurricane                100
##  2  1975 tropical depression       30
##  3  1975 tropical storm            60
##  4  1976 hurricane                105
##  5  1976 tropical depression       30
##  6  1976 tropical storm            60
##  7  1977 hurricane                150
##  8  1977 tropical depression       30
##  9  1977 tropical storm            60
## 10  1978 hurricane                 80
## # ... with 113 more rows
```

For each storm, summarize its average location in latitude / longitude coordinates.

```
storms %>%
  group_by(name) %>%
  summarize(avg_lat = mean(lat), avg_long = mean(long)) %>%
  mutate(location = paste(avg_lat, avg_long, sep = ", "))
```

```
## # A tibble: 198 x 4
##    name     avg_lat avg_long location
##    <chr>      <dbl>    <dbl> <chr>
##  1 AL011993   24.7     -78.0 24.6875, -78.05
##  2 AL012000   20.8     -93.1 20.85, -93.1
##  3 AL021992   26.7     -84.5 26.66, -84.48
##  4 AL021994   33.6     -79.7 33.6166666666667, -79.7333333333333
##  5 AL021999   20.4     -96.4 20.425, -96.4
##  6 AL022000    9.9     -28.5 9.9, -28.5416666666667
##  7 AL022001   11.9     -45.3 11.94, -45.32
##  8 AL022003    9.62    -43.4 9.625, -43.35
##  9 AL022006   41.3     -63.5 41.26, -63.48
## 10 AL031987   30.8     -88.7 30.784375, -88.7
## # ... with 188 more rows
```

For each storm, summarize its duration in number of hours (to the nearest 6hr increment).

```
storms %>%
  group_by(name) %>%
  mutate(duration = (n()-1)*6) %>%
  select(name, duration) %>%
  distinct
```

```
## # A tibble: 198 x 2
## # Groups:   name [198]
##    name     duration
##    <chr>       <dbl>
##  1 Amy           174
##  2 Caroline      192
##  3 Doris         132
##  4 Belle         102
##  5 Gloria        744
##  6 Anita         114
##  7 Clara         138
##  8 Evelyn         48
##  9 Amelia         30
## 10 Bess           72
## # ... with 188 more rows
```

For storm in a category, create a variable `storm_number` that enumerates the storms 1, 2, ... (in date order).

```
storms %>%
  mutate(storm_number = dense_rank(paste(year, month, day)))
```

```
## # A tibble: 10,010 x 14
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 10,000 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, storm_number <int>
```

Convert year, month, day, hour into the variable `timestamp` using the `lubridate` package. Although the new package `clock` just came out, `lubridate` still seems to be standard. Next year I'll probably switch the class to be using `clock`.

```
pacman::p_load(lubridate)
storms %>%
  mutate(timestamp = make_datetime(year, month, day, hour)) %>%
  select(timestamp, everything())
```

```
## # A tibble: 10,010 x 14
##    timestamp           name   year month   day  hour   lat  long status category
##    <dttm>              <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>
##  1 1975-06-27 00:00:00 Amy    1975     6    27     0  27.5 -79   tropi~ -1
##  2 1975-06-27 06:00:00 Amy    1975     6    27     6  28.5 -79   tropi~ -1
##  3 1975-06-27 12:00:00 Amy    1975     6    27    12  29.5 -79   tropi~ -1
```

```
##  4 1975-06-27 18:00:00 Amy    1975     6    27    18  30.5 -79   tropi~ -1
##  5 1975-06-28 00:00:00 Amy    1975     6    28     0  31.5 -78.8 tropi~ -1
##  6 1975-06-28 06:00:00 Amy    1975     6    28     6  32.4 -78.7 tropi~ -1
##  7 1975-06-28 12:00:00 Amy    1975     6    28    12  33.3 -78   tropi~ -1
##  8 1975-06-28 18:00:00 Amy    1975     6    28    18  34   -77   tropi~ -1
##  9 1975-06-29 00:00:00 Amy    1975     6    29     0  34.4 -75.8 tropi~ 0
## 10 1975-06-29 06:00:00 Amy    1975     6    29     6  34   -74.8 tropi~ 0
## # ... with 10,000 more rows, and 4 more variables: wind <int>, pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>
```

Using the `lubridate` package, create new variables `day_of_week` which is a factor with levels "Sunday", "Monday", ... "Saturday" and `week_of_year` which is integer 1, 2, ..., 52.

```
storms %>%
  mutate(timestamp = make_datetime(year, month, day),
         day_of_week = wday(ymd(timestamp), label = TRUE, abbr = FALSE),
         week_of_year = week(ymd(timestamp)))
```

```
## # A tibble: 10,010 x 16
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 10,000 more rows, and 5 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, timestamp <dttm>, day_of_week <ord>, week_of_year <dbl>
```

For each storm, summarize the day in which is started in the following format "Friday, June 27, 1975".

```
storms %>%
  group_by(name) %>%
  arrange(day, hour) %>%
  slice(1) %>%
   mutate(timestamp = make_datetime(year, month, day),
         day_of_week = wday(ymd(timestamp), label = TRUE, abbr = FALSE)) %>%
  summarize(start_date = paste(day_of_week, paste(month(month, label = TRUE, abbr = FALSE), day), year,
```

```
## # A tibble: 198 x 2
##    name     start_date
##    <chr>    <chr>
##  1 AL011993 Tuesday, June 1, 1993
##  2 AL012000 Wednesday, June 7, 2000
##  3 AL021992 Thursday, June 25, 1992
##  4 AL021994 Wednesday, July 20, 1994
##  5 AL021999 Friday, July 2, 1999
```

```
##  6 AL022000 Friday, June 23, 2000
##  7 AL022001 Wednesday, July 11, 2001
##  8 AL022003 Wednesday, June 11, 2003
##  9 AL022006 Monday, July 17, 2006
## 10 AL031987 Sunday, August 9, 1987
## # ... with 188 more rows
```

Create a new factor variable `decile_windspeed` by binning wind speed into 10 bins.

```
storms %>%
  mutate(decile_windspeed = factor(ntile(wind, 10)))
```

```
## # A tibble: 10,010 x 14
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 10,000 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <fct>
```

Create a new data frame `serious_storms` which are category 3 and above hurricanes.

```
serious_storms = storms %>%
  dplyr::filter(category >=3)
```

In `serious_storms`, merge the variables lat and long together into `lat_long` with values `lat / long` as a string.

```
serious_storms %>%
  unite(lat_long, lat, long, sep = "/")
```

```
## # A tibble: 779 x 12
##    name      year month   day  hour lat_long   status    category  wind pressure
##    <chr>    <dbl> <dbl> <int> <dbl> <chr>      <chr>     <ord>    <int>    <int>
##  1 Caroline  1975     8    31     0 24/-97     hurricane 3          100      973
##  2 Caroline  1975     8    31     6 24.1/-97.5 hurricane 3          100      963
##  3 Belle     1976     8     8    18 29.5/-75.3 hurricane 3          100      958
##  4 Belle     1976     8     9     0 30.9/-75.3 hurricane 3          105      957
##  5 Belle     1976     8     9     6 32.5/-75.2 hurricane 3          105      959
##  6 Anita     1977     9     1    18 25.2/-95.5 hurricane 3          110      945
##  7 Anita     1977     9     2     0 24.6/-96.2 hurricane 5          140      931
##  8 Anita     1977     9     2     6 24.2/-97.1 hurricane 5          150      926
##  9 Anita     1977     9     2    12 23.7/-98   hurricane 4          120      940
## 10 David     1979     8    28     0 12.2/-52.9 hurricane 4          115      947
## # ... with 769 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

Let's return now to the original storms data frame. For each category, find the average wind speed, pressure and diameters (do not count the NA's in your averaging).

```
storms %>%
  group_by(category) %>%
  summarize(avg_wind_speed = mean(wind),
            avg_pressure = mean(pressure),
            avg_hu = mean(hu_diameter, na.rm = TRUE),
            avg_ts = mean(ts_diameter, na.rm = TRUE)
            )
```

```
## # A tibble: 7 x 5
##   category avg_wind_speed avg_pressure avg_hu avg_ts
##   <ord>             <dbl>        <dbl>  <dbl>  <dbl>
## 1 -1                 27.3        1008.    0       0
## 2 0                  45.8         999.    0     160.
## 3 1                  70.9         982.   57.3   278.
## 4 2                  89.4         967.   78.8   282.
## 5 3                 105.          954.   91.4   307.
## 6 4                 122.          940.  102.    315.
## 7 5                 145.          916.  120.    317.
```

For each named storm, find its maximum category, wind speed, pressure and diameters (do not allow the max to be NA) and the number of readings (i.e. observations).

```
suppressWarnings(storms %>%
  group_by(name) %>%
  summarize(max_category = max(category),
            max_wind = max(wind),
            max_pressure = max(pressure),
            max_hu = max(hu_diameter, na.rm = TRUE),
            max_ts = max(ts_diameter, na.rm = TRUE),
            count = n()))
```

```
## # A tibble: 198 x 7
##    name     max_category max_wind max_pressure max_hu max_ts count
##    <chr>    <ord>           <int>        <int>  <dbl>  <dbl> <int>
##  1 AL011993 -1                 30         1003   -Inf -Inf       8
##  2 AL012000 -1                 25         1010   -Inf -Inf       4
##  3 AL021992 -1                 30         1009   -Inf -Inf       5
##  4 AL021994 -1                 30         1017   -Inf -Inf       6
##  5 AL021999 -1                 30         1006   -Inf -Inf       4
##  6 AL022000 -1                 30         1010   -Inf -Inf      12
##  7 AL022001 -1                 25         1012   -Inf -Inf       5
##  8 AL022003 -1                 30         1010   -Inf -Inf       4
##  9 AL022006 0                  45         1008      0   69.0     5
## 10 AL031987 0                  40         1015   -Inf -Inf      32
## # ... with 188 more rows
```

Calculate the distance from each storm observation to Miami in a new variable `distance_to_miami`. This is very challenging. You will need a function that computes distances from two sets of latitude / longitude coordinates.

```
MIAMI_LAT_LONG_COORDS = c(25.7617, -80.1918)

find_distance = function(lat1, long1, lat2, long2){
  lat1 = lat1*180/pi
  long1 = long1*180/pi
  lat2 = lat2*180/pi
  long2 = long2*180/pi
  x_1 = sin(lat2 - lat1 / 2)^2 + (cos(lat2) * cos(lat1)) * sin(long2 - long1 / 2)^2
  x_2 = 2 * atan2(sqrt(x_1), sqrt(1-x_1))
  distance = 6373.0 * x_2 #multiply by radius of earth in km
  distance
}

suppressWarnings(storms %>%
  mutate(distance_to_miami = find_distance(lat,long, MIAMI_LAT_LONG_COORDS[1], MIAMI_LAT_LONG_COORDS[2])
  mutate(distance_to_miami = ifelse(is.na(distance_to_miami), 0, distance_to_miami)))
```

```
## # A tibble: 10,010 x 14
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 10,000 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, distance_to_miami <dbl>
```

For each storm observation, use the function from the previous question to calculate the distance it moved since the previous observation.

```
suppressWarnings(storms %<>%
  group_by(name) %>%
  mutate(dist_from_previous = ifelse(name != lag(name), 0, find_distance(lat, long, lag(lat), lag(long))
  mutate(dist_from_previous = ifelse(is.na(dist_from_previous), 0, dist_from_previous)))
```

For each storm, find the total distance it moved over its observations and its total displacement. "Distance" is a scalar quantity that refers to "how much ground an object has covered" during its motion. "Displacement" is a vector quantity that refers to "how far out of place an object is"; it is the object's overall change in position.

```
storms %<>%
  group_by(name) %>%
  mutate(distance = sum(dist_from_previous)) %>%
  mutate(displacement = last(dist_from_previous) - first(dist_from_previous))
```

For each storm observation, calculate the average speed the storm moved in location.

```
storms %<>%
  group_by(name) %>%
  mutate(speed = dist_from_previous/6) ##dividing by the 6 hour increments
```

For each storm, calculate its average ground speed (how fast its eye is moving which is different from windspeed around the eye).

```
storms %<>%
  group_by(name) %>%
  mutate(avg_ground = mean(speed))
```

Is there a relationship between average ground speed and maximum category attained? Use a dataframe summary (not a regression).

```
X = storms %>%
  group_by(name) %>%
  summarize(max_category = max(category), avg_ground) %>%
  unique()
```

```
## `summarise()` has grouped output by 'name'. You can override using the `.groups` argument.
```

```
head(X)
```

```
## # A tibble: 6 x 3
## # Groups:   name [6]
##   name     max_category avg_ground
##   <chr>    <ord>             <dbl>
## 1 AL011993 -1                 601.
## 2 AL012000 -1                1079.
## 3 AL021992 -1                 886.
## 4 AL021994 -1                1705.
## 5 AL021999 -1                1757.
## 6 AL022000 -1                1409.
```

```
cor(as.numeric(X$max_category), X$avg_ground)
```

```
## [1] 0.08382671
```

Now we want to transition to building real design matrices for prediction. This is more in tune with what happens in the real world. Large data dump and you convert it into X and y how you see fit.

Suppose we wish to predict the following: given the first three readings of a storm, can you predict its maximum wind speed? Identify the `y` and identify which features you need $x_1, ...x_p$ and build that matrix with `dplyr` functions. This is not easy, but it is what it's all about. Feel free to "featurize" as creatively as you would like. You aren't going to overfit if you only build a few features relative to the total 198 storms.

```
X = storms %>%
  group_by(name) %>%
  mutate(max_wind = max(wind), ##this is our y
            max_category = max(category),
```

```
            avg_pressure = mean(pressure),
            avg_distance = mean(distance)) %>%
  slice(1:3) %>%
  ungroup() %>%
  select(max_category, avg_pressure, avg_distance,max_wind)
X
```

```
## # A tibble: 593 x 4
##    max_category avg_pressure avg_distance max_wind
##    <ord>               <dbl>        <dbl>    <int>
##  1 -1                  1000.       28845.       30
##  2 -1                  1000.       28845.       30
##  3 -1                  1000.       28845.       30
##  4 -1                  1009.       25894.       25
##  5 -1                  1009.       25894.       25
##  6 -1                  1009.       25894.       25
##  7 -1                  1007.       26594.       30
##  8 -1                  1007.       26594.       30
##  9 -1                  1007.       26594.       30
## 10 -1                  1016.       61383.       30
## # ... with 583 more rows
```

Fit your model. Validate it.

```
n = nrow(X)
K = 5
test_indices = sample(1 : n, size = n * 1 / K)
master_train_indices = setdiff(1 : n, test_indices) ##overall train
select_indices = sample(master_train_indices, size = n * 1 / K)
subtrain_indices = setdiff(master_train_indices, select_indices)

storms_train = X[master_train_indices,]

storms_subtrain = X[subtrain_indices, ]
y_subtrain = storms_subtrain$max_wind

storms_select = X[select_indices, ]
y_select = storms_select$max_wind
storms_select$max_wind= NULL

storms_test = X[test_indices, ]
y_test = storms_test$max_wind
storms_test$max_wind = NULL

mod = lm(max_wind~ ., data = storms_subtrain)
mod2 = lm(max_wind~.*., data = storms_subtrain)
length(coef(mod)) #9 features
```

```
## [1] 9
```

```
length(coef(mod2)) #22 features
```

```
## [1] 22
```

```
yhat_mod = predict(mod, storms_select)
yhat_mod2 = predict(mod2, storms_select)
se_select_mod = sd(y_select - yhat_mod)
se_select_mod2 = sd(y_select - yhat_mod2)

c(se_select_mod, se_select_mod2) #pick mod2 with more interactions
```

```
## [1] 4.964288 4.569563
```

```
g_final = lm(max_wind ~.*., data = storms_train)
yhat_final = predict(g_final, storms_test)
se_final = sd(y_test - yhat_final)
se_final
```

```
## [1] 4.303091
```

Assess your level of success at this endeavor.

The final model had an oos SE of 3.915, meaning that the model is able to predict the max wind of a storm within plus or minus, 8 mph. This seems to do pretty well considering the range of max_wind goes from 25 to 160 mph and the model is within a range of approximately 16 mph.

The Forward Stepwise Procedure for Probability Estimation Models

Set a seed and load the `adult` dataset and remove missingness and randomize the order.

```
set.seed(119)
pacman::p_load_gh("coatless/ucidata")
data(adult)
adult = na.omit(adult)
adult = adult[sample(1 : nrow(adult)), ]
```

Copy from the previous lab all cleanups you did to this dataset.

```
adult$fnlwgt = NULL
adult$marital_status=as.character(adult$marital_status)
adult$marital_status = ifelse(adult$marital_status == "Married-AF-spouse" | adult$marital_status== "Mar
adult$marital_status = as.factor(adult$marital_status)

adult$education=as.character(adult$education)
adult$education = ifelse(adult$education == "1st-4th" | adult$education== "Preschool", "<=4th", adult$e
adult$education = as.factor(adult$education)
tab = sort(table(adult$native_country))
adult$native_country=as.character(adult$native_country)
adult$native_country = ifelse(adult$native_country %in% names(tab[tab<50]), "other", adult$native_count
adult$native_country = as.factor(adult$native_country)
```

```
adult$occupation = as.character(adult$occupation)
adult$workclass = as.character(adult$workclass)
adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tab_worktype = sort(table(adult$worktype))
adult$occupation = NULL
adult$workclass = NULL

adult$worktype=as.character(adult$worktype)
adult$worktype = ifelse(adult$worktype %in% names(tab_worktype[tab_worktype<100]), "other", adult$workty
adult$worktype = as.factor(adult$worktype)

adult$marital_status = as.character(adult$marital_status)
adult$relationship = as.character(adult$relationship)
adult$relationship_status = paste(adult$marital_status, adult$relationship, sep = ":")
adult$relationship_status = as.factor(adult$relationship_status)
tab_relationship_status = sort(table(adult$relationship_status))
adult$marital_status = NULL
adult$relationship = NULL

adult$relationship_status=as.character(adult$relationship_status)
adult$relationship_status = ifelse(adult$relationship_status %in% names(tab_relationship_status[tab_rela
adult$relationship_status = as.factor(adult$relationship_status)
```

We will be doing model selection. We will split the dataset into 3 distinct subsets. Set the size of our splits here. For simplicitiy, all three splits will be identically sized. We are making it small so the stepwise algorithm can compute quickly. If you have a faster machine, feel free to increase this.

```
Nsplitsize = 1000
```

Now create the following variables: `Xtrain`, `ytrain`, `Xselect`, `yselect`, `Xtest`, `ytest` with `Nsplitsize` observations. Binarize the y values.

```
Xtrain = adult[1 : Nsplitsize, ]
Xtrain$income = NULL
ytrain = ifelse(adult[1 : Nsplitsize, "income"] == ">50K", 1, 0)
Xselect = adult[(Nsplitsize + 1) : (2 * Nsplitsize), ]
Xselect$income = NULL
yselect = ifelse(adult[(Nsplitsize + 1) : (2 * Nsplitsize), "income"] ==">50K", 1, 0)
Xtest = adult[(2 * Nsplitsize + 1) : (3 * Nsplitsize), ]
Xtest$income = NULL
ytest = ifelse(adult[(2 * Nsplitsize + 1) : (3 * Nsplitsize), "income"] == ">50K", 1, 0)
```

Fit a vanilla logistic regression on the training set.

```
logistic_mod = glm(ytrain ~ ., Xtrain, family = binomial(link = logit))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

and report the log scoring rule, the Brier scoring rule.

```
brier_score = function(prob_est_mod, X, y){
  phat = predict(prob_est_mod, X, type = "response")
  mean (- (y-phat)^2)
}

brier_score(logistic_mod, Xtrain, ytrain)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## [1] -0.08061693
```

```
log_score = function(prob_est_mod, X, y){
  phat = predict(prob_est_mod, X, type = "response")
  mean(y*log(phat)+ (1-y)*log(1-phat))
}

log_score(logistic_mod, Xtrain, ytrain)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## [1] -0.2504639
```

We will be doing model selection using a basis of linear features consisting of all first-order interactions of the 14 raw features (this will include square terms as squares are interactions with oneself).

Create a model matrix from the training data containing all these features. Make sure it has an intercept column too (the one vector is usually an important feature). Cast it as a data frame so we can use it more easily for modeling later on. We're going to need those model matrices (as data frames) for both the select and test sets. So make them here too (copy-paste). Make sure their dimensions are sensible.

```
Xmm_train = data.frame(model.matrix(~., data = Xtrain))
Xmm_select = data.frame(model.matrix(~., data = Xselect))
Xmm_test = data.frame(model.matrix(~., data = Xtest))
dim(Xmm_train)
```

```
## [1] 1000  103
```

```
dim(Xmm_select)
```

```
## [1] 1000  103
```

```
dim(Xmm_test)
```

```
## [1] 1000  103
```

Write code that will fit a model stepwise. You can refer to the chunk in the practice lecture. Use the negative Brier score to do the selection. The negative of the Brier score is always positive and lower means better making this metric kind of like s_e so the picture will be the same as the canonical U-shape for oos performance.

Run the code and hit "stop" when you begin to the see the Brier score degrade appreciably oos. Be patient as it will wobble.

```
#try to break at 100
pacman::p_load(Matrix)
p_plus_one = ncol(Xmm_train)
predictor_by_iteration = c() #keep a growing list of predictors by iteration
in_sample_brier_by_iteration = c() #keep a growing list of briers by iteration
oos_brier_by_iteration = c() #keep a growing list of briers by iteration
i = 1

repeat {

  #get all predictors left to try
  all_briers = array(NA, p_plus_one) #record all possibilities
  for (j_try in 1 : p_plus_one){
    if (j_try %in% predictor_by_iteration){
      next
    }
    Xmm_sub = Xmm_train[, c(predictor_by_iteration, j_try), drop = FALSE]
    logistic_mod = suppressWarnings(glm(ytrain ~ ., Xmm_sub, family = "binomial"))
    phat_train = suppressWarnings(predict(logistic_mod, Xmm_sub, type = 'response'))
    all_briers[j_try] = -mean(-(ytrain - phat_train)^2)
  }
  j_star = which.max(all_briers)
  predictor_by_iteration = c(predictor_by_iteration, j_star)
  in_sample_brier_by_iteration = c(in_sample_brier_by_iteration, all_briers[j_star])

  #now let's look at oos
  Xmm_sub = Xmm_train[, predictor_by_iteration, drop = FALSE]

    logistic_mod = suppressWarnings(glm(ytrain ~ ., Xmm_sub, family = "binomial"))
    phat_train = suppressWarnings(predict(logistic_mod, Xmm_sub, type = 'response'))
    all_briers[j_try] = -mean(-(ytrain - phat_train)^2)

    phat_select = suppressWarnings(predict(logistic_mod, Xmm_select[, predictor_by_iteration, drop = FA

    oos_brier = -mean(-(yselect - phat_select)^2)
    oos_brier_by_iteration = c(oos_brier_by_iteration, oos_brier)

  cat("i =", i, "in-sample_brier =", all_briers[j_star], "oos_brier =", oos_brier, "\n   predictor added

  i = i + 1

  if (i > Nsplitsize || i > p_plus_one){
    break
  }
}
```

```
## i = 1 in-sample_brier = 0.1875 oos_brier = 0.193
```

```
##    predictor added: worktypeProtective.serv.Private
## i = 2 in-sample_brier = 0.1875 oos_brier = 0.193
##    predictor added: X.Intercept.
## i = 3 in-sample_brier = 0.1875 oos_brier = 0.193
##    predictor added: native_countryJamaica
## i = 4 in-sample_brier = 0.1875 oos_brier = 0.193
##    predictor added: native_countryPoland
## i = 5 in-sample_brier = 0.1874914 oos_brier = 0.1930585
##    predictor added: educationAssoc.acdm
## i = 6 in-sample_brier = 0.1874827 oos_brier = 0.1931395
##    predictor added: native_countryGermany
## i = 7 in-sample_brier = 0.1874774 oos_brier = 0.192884
##    predictor added: worktypeSales.Self.emp.inc
## i = 8 in-sample_brier = 0.1874648 oos_brier = 0.192767
##    predictor added: native_countryCuba
## i = 9 in-sample_brier = 0.1874441 oos_brier = 0.1926326
##    predictor added: native_countryEngland
## i = 10 in-sample_brier = 0.1874233 oos_brier = 0.1927827
##    predictor added: native_countryVietnam
## i = 11 in-sample_brier = 0.1874004 oos_brier = 0.1927051
##    predictor added: worktypeTech.support.Private
## i = 12 in-sample_brier = 0.1873554 oos_brier = 0.1924547
##    predictor added: worktypeProf.specialty.State.gov
## i = 13 in-sample_brier = 0.187305 oos_brier = 0.1921257
##    predictor added: worktypeExec.managerial.Federal.gov
## i = 14 in-sample_brier = 0.1872545 oos_brier = 0.1920954
##    predictor added: native_countryItaly
## i = 15 in-sample_brier = 0.1871929 oos_brier = 0.1920315
##    predictor added: native_countryChina
## i = 16 in-sample_brier = 0.1871312 oos_brier = 0.1917824
##    predictor added: relationship_statusother
## i = 17 in-sample_brier = 0.1870699 oos_brier = 0.1919997
##    predictor added: worktypeSales.Private
## i = 18 in-sample_brier = 0.1869867 oos_brier = 0.1917508
##    predictor added: raceOther
## i = 19 in-sample_brier = 0.1869021 oos_brier = 0.1914944
##    predictor added: worktypeAdm.clerical.State.gov
## i = 20 in-sample_brier = 0.1868107 oos_brier = 0.1914385
##    predictor added: worktypeFarming.fishing.Self.emp.not.inc
## i = 21 in-sample_brier = 0.1867043 oos_brier = 0.1918837
##    predictor added: worktypeTransport.moving.Self.emp.not.inc
## i = 22 in-sample_brier = 0.1865949 oos_brier = 0.1921209
##    predictor added: native_countryIndia
## i = 23 in-sample_brier = 0.186472 oos_brier = 0.1915765
##    predictor added: relationship_statusDivorced.Own.child
## i = 24 in-sample_brier = 0.1863423 oos_brier = 0.1914852
##    predictor added: native_countryDominican.Republic
## i = 25 in-sample_brier = 0.186212 oos_brier = 0.191349
##    predictor added: native_countryJapan
## i = 26 in-sample_brier = 0.1860811 oos_brier = 0.1910807
##    predictor added: relationship_statusMarried.spouse.absent.Unmarried
## i = 27 in-sample_brier = 0.1859424 oos_brier = 0.1913287
##    predictor added: native_countryother
## i = 28 in-sample_brier = 0.185799 oos_brier = 0.1914366
```

```
##    predictor added: worktypeExec.managerial.Self.emp.not.inc
## i = 29 in-sample_brier = 0.1856481 oos_brier = 0.1915811
##    predictor added: native_countryGuatemala
## i = 30 in-sample_brier = 0.1854932 oos_brier = 0.1914397
##    predictor added: worktypeOther.service.State.gov
## i = 31 in-sample_brier = 0.1853329 oos_brier = 0.1913469
##    predictor added: native_countryColumbia
## i = 32 in-sample_brier = 0.1851701 oos_brier = 0.1913229
##    predictor added: native_countrySouth
## i = 33 in-sample_brier = 0.1849907 oos_brier = 0.1911184
##    predictor added: relationship_statusWidowed.Not.in.family
## i = 34 in-sample_brier = 0.1848102 oos_brier = 0.1912467
##    predictor added: native_countryPhilippines
## i = 35 in-sample_brier = 0.1846171 oos_brier = 0.1914852
##    predictor added: worktypeCraft.repair.Local.gov
## i = 36 in-sample_brier = 0.1844225 oos_brier = 0.1914933
##    predictor added: worktypeCraft.repair.Self.emp.not.inc
## i = 37 in-sample_brier = 0.1842163 oos_brier = 0.1910224
##    predictor added: relationship_statusSeparated.Not.in.family
## i = 38 in-sample_brier = 0.184003 oos_brier = 0.1908809
##    predictor added: worktypeTransport.moving.Local.gov
## i = 39 in-sample_brier = 0.1837877 oos_brier = 0.1907309
##    predictor added: relationship_statusMarried.Own.child
## i = 40 in-sample_brier = 0.1835577 oos_brier = 0.1906811
##    predictor added: native_countryPuerto.Rico
## i = 41 in-sample_brier = 0.1833129 oos_brier = 0.1902605
##    predictor added: education7th.8th
## i = 42 in-sample_brier = 0.1830615 oos_brier = 0.1899138
##    predictor added: worktypePriv.house.serv.Private
## i = 43 in-sample_brier = 0.182846 oos_brier = 0.1901682
##    predictor added: native_countryEl.Salvador
## i = 44 in-sample_brier = 0.1826043 oos_brier = 0.190608
##    predictor added: relationship_statusMarried.spouse.absent.Not.in.family
## i = 45 in-sample_brier = 0.1823424 oos_brier = 0.1901003
##    predictor added: worktypeFarming.fishing.Private
## i = 46 in-sample_brier = 0.1820864 oos_brier = 0.1896127
##    predictor added: worktypeExec.managerial.Self.emp.inc
## i = 47 in-sample_brier = 0.1818181 oos_brier = 0.1894936
##    predictor added: native_countryUnited.States
## i = 48 in-sample_brier = 0.1816359 oos_brier = 0.1893227
##    predictor added: education5th.6th
## i = 49 in-sample_brier = 0.1813595 oos_brier = 0.1891713
##    predictor added: relationship_statusSeparated.Own.child
## i = 50 in-sample_brier = 0.1810828 oos_brier = 0.1905475
##    predictor added: raceAsian.Pac.Islander
## i = 51 in-sample_brier = 0.1808007 oos_brier = 0.1903948
##    predictor added: relationship_statusSeparated.Other.relative
## i = 52 in-sample_brier = 0.180485 oos_brier = 0.1906098
##    predictor added: worktypeother
## i = 53 in-sample_brier = 0.1801984 oos_brier = 0.1909189
##    predictor added: worktypeCraft.repair.Private
## i = 54 in-sample_brier = 0.1798251 oos_brier = 0.190723
##    predictor added: worktypeSales.Self.emp.not.inc
## i = 55 in-sample_brier = 0.1794652 oos_brier = 0.1902039
```

```
##    predictor added: worktypeMachine.op.inspct.Private
## i = 56 in-sample_brier = 0.1790928 oos_brier = 0.1891132
##    predictor added: worktypeProf.specialty.Self.emp.not.inc
## i = 57 in-sample_brier = 0.1786954 oos_brier = 0.1891712
##    predictor added: relationship_statusDivorced.Other.relative
## i = 58 in-sample_brier = 0.1782868 oos_brier = 0.1887587
##    predictor added: worktypeOther.service.Local.gov
## i = 59 in-sample_brier = 0.1778711 oos_brier = 0.1884657
##    predictor added: education9th
## i = 60 in-sample_brier = 0.1774272 oos_brier = 0.1887331
##    predictor added: worktypeProf.specialty.Local.gov
## i = 61 in-sample_brier = 0.1769939 oos_brier = 0.189252
##    predictor added: worktypeAdm.clerical.Local.gov
## i = 62 in-sample_brier = 0.1765207 oos_brier = 0.1886085
##    predictor added: worktypeOther.service.Self.emp.not.inc
## i = 63 in-sample_brier = 0.1760158 oos_brier = 0.1913734
##    predictor added: worktypeProtective.serv.State.gov
## i = 64 in-sample_brier = 0.1754713 oos_brier = 0.1920718
##    predictor added: relationship_statusMarried.Other.relative
## i = 65 in-sample_brier = 0.1749069 oos_brier = 0.1935001
##    predictor added: worktypeExec.managerial.Local.gov
## i = 66 in-sample_brier = 0.1742923 oos_brier = 0.1916799
##    predictor added: worktypeHandlers.cleaners.Private
## i = 67 in-sample_brier = 0.1735638 oos_brier = 0.1921958
##    predictor added: relationship_statusWidowed.Unmarried
## i = 68 in-sample_brier = 0.1727953 oos_brier = 0.1895753
##    predictor added: raceWhite
## i = 69 in-sample_brier = 0.172594 oos_brier = 0.1899023
##    predictor added: raceBlack
## i = 70 in-sample_brier = 0.17193 oos_brier = 0.1891318
##    predictor added: relationship_statusSeparated.Unmarried
## i = 71 in-sample_brier = 0.1712842 oos_brier = 0.1905047
##    predictor added: educationAssoc.voc
## i = 72 in-sample_brier = 0.170445 oos_brier = 0.1899959
##    predictor added: education12th
## i = 73 in-sample_brier = 0.1695559 oos_brier = 0.1880339
##    predictor added: relationship_statusNever.married.Other.relative
## i = 74 in-sample_brier = 0.1686675 oos_brier = 0.187446
##    predictor added: worktypeProf.specialty.Self.emp.inc
## i = 75 in-sample_brier = 0.1676971 oos_brier = 0.1887105
##    predictor added: education10th
## i = 76 in-sample_brier = 0.1666957 oos_brier = 0.1876612
##    predictor added: worktypeProf.specialty.Federal.gov
## i = 77 in-sample_brier = 0.1656881 oos_brier = 0.1874898
##    predictor added: worktypeTransport.moving.Private
## i = 78 in-sample_brier = 0.1646454 oos_brier = 0.189744
##    predictor added: worktypeExec.managerial.State.gov
## i = 79 in-sample_brier = 0.163276 oos_brier = 0.1890693
##    predictor added: education11th
## i = 80 in-sample_brier = 0.1617077 oos_brier = 0.1897186
##    predictor added: worktypeProtective.serv.Local.gov
## i = 81 in-sample_brier = 0.1600879 oos_brier = 0.1870476
##    predictor added: relationship_statusDivorced.Unmarried
## i = 82 in-sample_brier = 0.1584822 oos_brier = 0.1902353
```

```
##    predictor added: native_countryMexico
## i = 83 in-sample_brier = 0.1568702 oos_brier = 0.188998
##    predictor added: educationDoctorate
## i = 84 in-sample_brier = 0.1550575 oos_brier = 0.1867599
##    predictor added: educationSome.college
## i = 85 in-sample_brier = 0.1532508 oos_brier = 0.1866456
##    predictor added: worktypeProf.specialty.Private
## i = 86 in-sample_brier = 0.1517495 oos_brier = 0.1849195
##    predictor added: worktypeAdm.clerical.Private
## i = 87 in-sample_brier = 0.1501224 oos_brier = 0.1864301
##    predictor added: capital_loss
## i = 88 in-sample_brier = 0.1484982 oos_brier = 0.1863653
##    predictor added: educationProf.school
## i = 89 in-sample_brier = 0.1461234 oos_brier = 0.1825821
##    predictor added: relationship_statusNever.married.Unmarried
## i = 90 in-sample_brier = 0.1435705 oos_brier = 0.1756891
##    predictor added: educationBachelors
## i = 91 in-sample_brier = 0.1399805 oos_brier = 0.1736974
##    predictor added: worktypeExec.managerial.Private
## i = 92 in-sample_brier = 0.1380136 oos_brier = 0.1716672
##    predictor added: worktypeOther.service.Private
## i = 93 in-sample_brier = 0.1353454 oos_brier = 0.1690406
##    predictor added: hours_per_week
## i = 94 in-sample_brier = 0.1306897 oos_brier = 0.1627436
##    predictor added: sexMale
## i = 95 in-sample_brier = 0.1262455 oos_brier = 0.1575297
##    predictor added: educationHS.grad
## i = 96 in-sample_brier = 0.125764 oos_brier = 0.1568466
##    predictor added: educationMasters
## i = 97 in-sample_brier = 0.125764 oos_brier = 0.1568466
##    predictor added: education_num
## i = 98 in-sample_brier = 0.1189196 oos_brier = 0.1523416
##    predictor added: age
## i = 99 in-sample_brier = 0.114518 oos_brier = 0.1493713
##    predictor added: relationship_statusNever.married.Own.child
## i = 100 in-sample_brier = 0.1053176 oos_brier = 0.1423397
##    predictor added: relationship_statusNever.married.Not.in.family
## i = 101 in-sample_brier = 0.1010409 oos_brier = 0.1418943
##    predictor added: relationship_statusMarried.Husband
## i = 102 in-sample_brier = 0.09150601 oos_brier = 0.1371742
##    predictor added: relationship_statusMarried.Wife
## i = 103 in-sample_brier = 0.08061693 oos_brier = 0.1343893
##    predictor added: capital_gain
```

Plot the in-sample and oos (select set) Brier score by p. Does this look like what's expected?

```
simulation_results = data.frame(
  iteration = 1 : length(in_sample_brier_by_iteration),
  in_sample_briers_by_iteration = in_sample_brier_by_iteration,
  oos_brier_by_iteration = oos_brier_by_iteration
)
pacman::p_load(latex2exp)
ggplot(simulation_results) +
  geom_line(aes(x = iteration, y = in_sample_brier_by_iteration), col = "red") +
```

```
  geom_line(aes(x = iteration, y = oos_brier_by_iteration), col = "blue") +
  ylab(TeX("$brier$"))
```