Lab 7

Amir ElTabakh

11:59PM April 22, 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.

```
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)
head(X)
```

```
##              [,1]       [,2]        [,3]        [,4]        [,5]        [,6]
## [1,]  0.064071254  2.3702098 -0.62569183  0.09888788 -0.18128948  0.49861552
## [2,]  0.007602619 -0.3504307 -0.30662858  0.02487729  0.07026957 -0.05487262
## [3,] -1.366606978 -1.0386020 -0.43562582  0.54696104 -0.22478256 -0.72850836
## [4,]  1.502859242 -1.2546661 -0.01168848 -0.32054404 -0.48503512  1.68411329
## [5,] -0.856949963  0.1613953 -0.08115485  0.09028681  0.39318678 -1.35158436
## [6,] -0.525506505  0.6917573 -0.32776881  1.00821468 -1.41934053 -1.82107162
##            [,7]       [,8]        [,9]       [,10]
## [1,] -1.8013142  0.6925615 -0.74413814  1.86696947
## [2,] -1.5420851  0.1715897 -0.14045166 -0.55197730
## [3,]  0.4384477  0.6947109 -1.31643707 -1.86342519
## [4,] -0.8570649 -0.3767673  1.61928132 -0.27343490
## [5,]  0.6006799 -1.8808094  1.40016174 -0.08253074
## [6,] -1.0807111 -0.7096775 -0.01671964  0.35804195
```

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)))
}

all_angles <- function(X) {
  A = matrix(NA, nrow = nrow(X), ncol = nrow(X))
  for(i in 1:(nrow(X) - 1)){
```

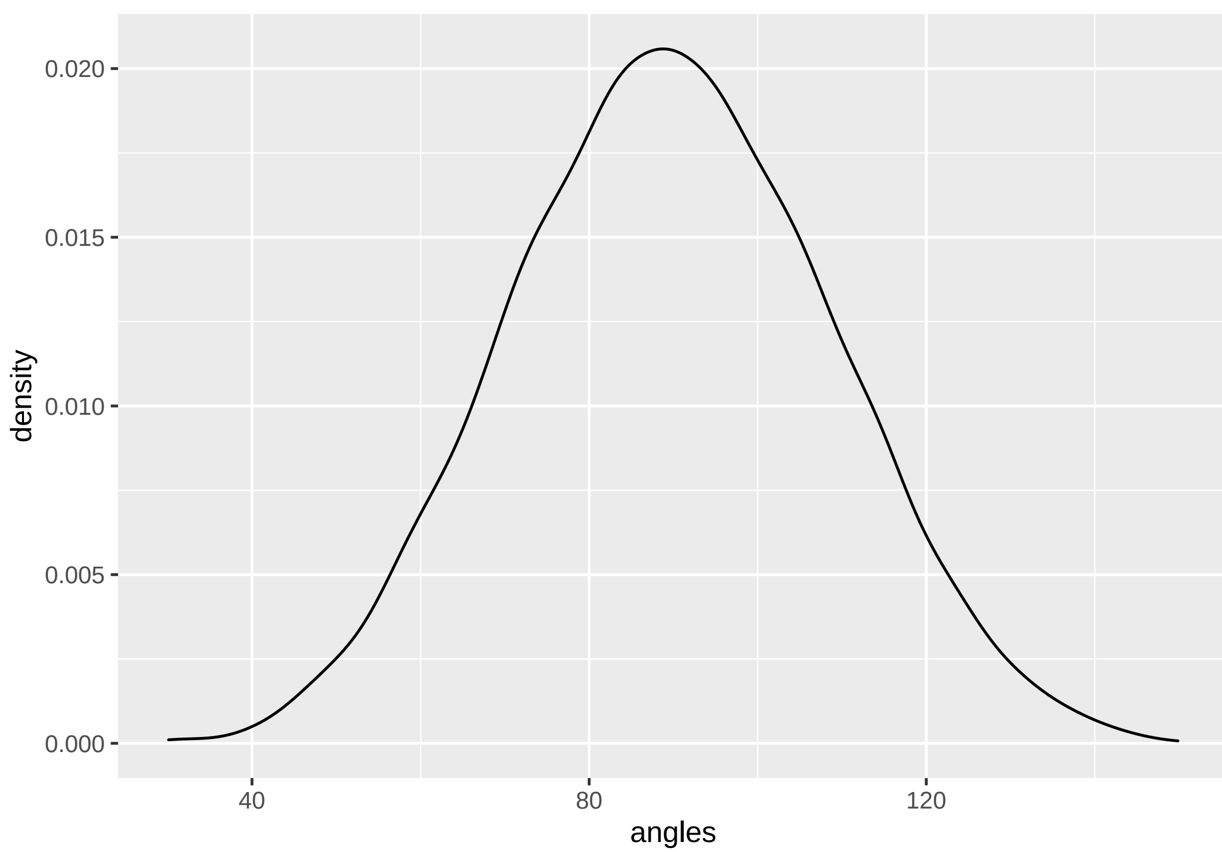
```r
    for(j in (i+1):nrow(X)){
      A[i,j] = angle(X[i,], X[j,]) * (180/pi)
    }
  }
  A
}

#all_angles(X)
```

Plot the density of these angles.

```r
pacman::p_load(ggplot2)
ggplot(data.frame(angles = c(all_angles(X)))) +
  aes(x = angles) +
  geom_density()
```

```
## 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.

```r
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++){
          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);
          //acos(sum(u * v)/sqrt(sum(u^2)*sum(v^2)))
          acos(sum_u_v/sqrt(sum_sqd_u * sum_sqd_v)) * (180/M_PI);
        }
        A(i_1, i_2) = acos(sum_u_v/sqrt(sum_sqd_u * sum_sqd_v)) * (180/M_PI);
      }
    }
    return A;
  }
  ')

#all_angles_cpp(X)
```

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.

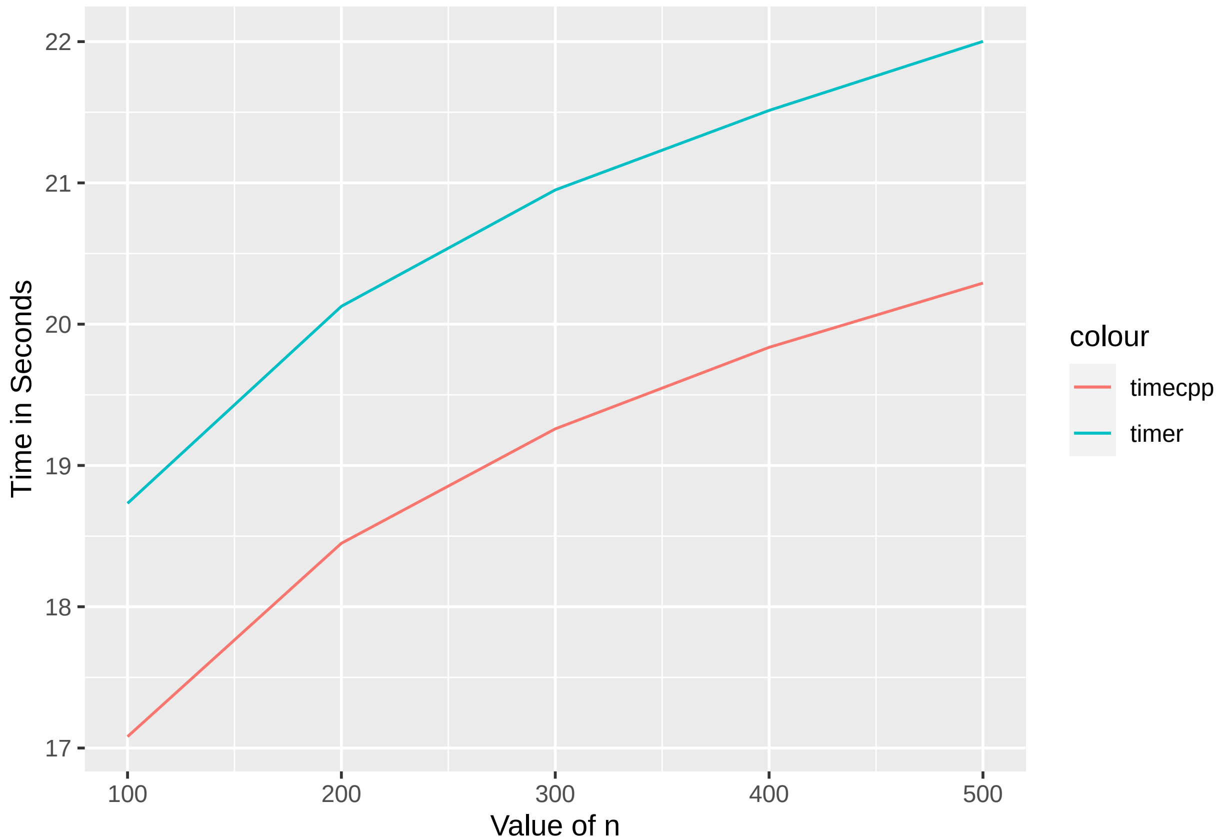
```
pacman::p_load(microbenchmark)

n <- 1000
Nvec <- c(100, 200, 300, 400, 500)
time_r <- c()
time_cpp <- c()
for (i in 1:length(Nvec)){
  X <- c()
  for (j in 1:n){
    x <- rnorm(Nvec[i])
    X <- cbind(X, x)
  }
  time_r <- c(time_r, mean(microbenchmark(angles_r = all_angles(X), times = 3, unit = "s")$time))
  time_cpp <- c(time_cpp, mean(microbenchmark(angles_cpp = all_angles_cpp(X), times = 3, unit = "s")$ti
}
```

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. We will 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_r), col = "timer")) +
  geom_line(aes(x = Nvec, y = log(time_cpp), col = "timecpp")) +
```

```r
  xlab("Value of n") +
  ylab("Time in 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.

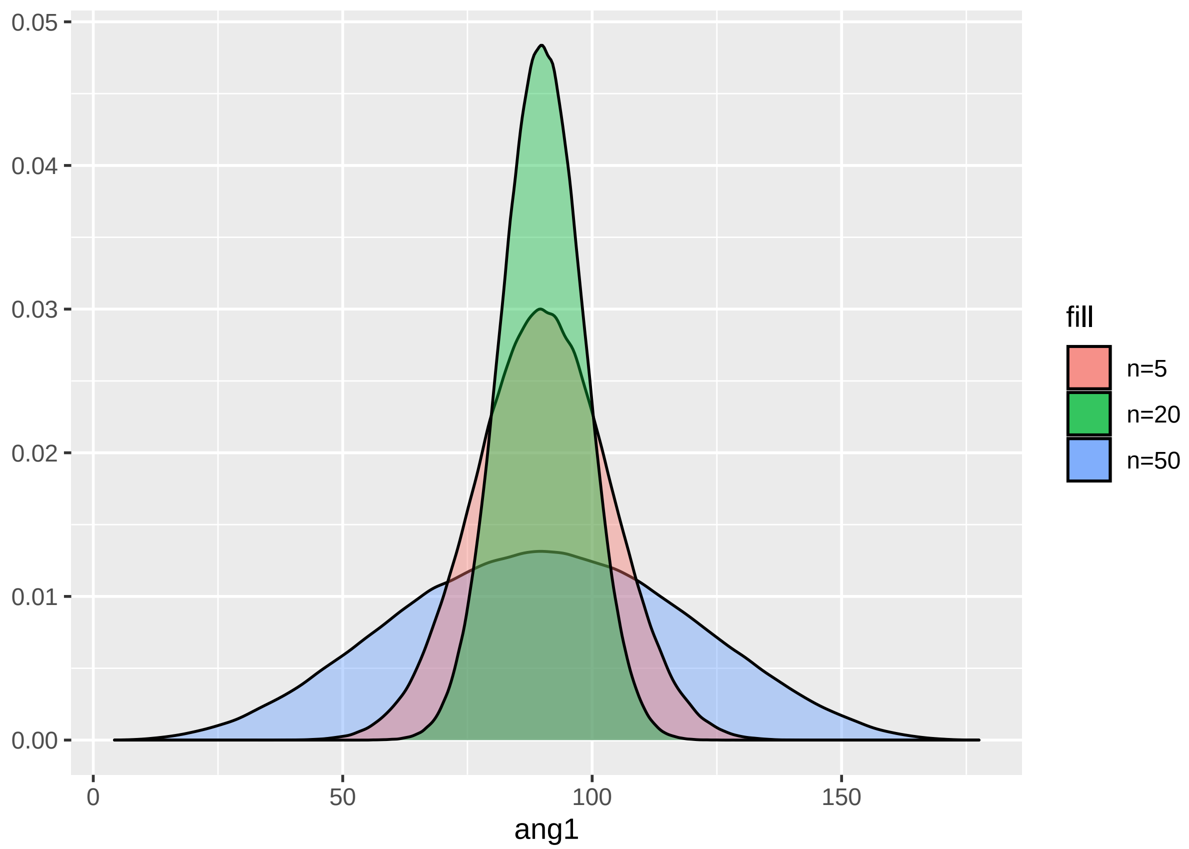
```r
Nvec = 1000
X <- c()
for (i in 1:5){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}
ang1 <- all_angles(X)
X <- c()
for (i in 1:20){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}
ang2 <- all_angles(X)
X <- c()
for (i in 1:50){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}
ang3 <- all_angles(X)
```

```r
ggplot() +
  geom_density(aes(x = ang1, fill = "red"), alpha = .4) +
  geom_density(aes(x = ang2, fill = "blue"), alpha = .4) +
  geom_density(aes(x = ang3, fill = "green"), alpha = .4) +
  scale_fill_discrete(labels = c("n=5", "n=20", "n=50")) +
  ylab("Density") +
  ylab("")
```

```
## 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 sequence 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.

```r
nth_fibonacci <- function(n, start){
  if (n == 1 | n == 2) return(start)
  else return(nth_fibonacci(n-1, start) + nth_fibonacci(n-2, start))
}
nth_fibonacci(8, 1)
```

```
## [1] 21
```

Write an Rcpp function `nth_fibonnaci_cpp` that does the same thing. Use an IDE if ou 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(8, 1)
```

```
## [1] 21
```

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 = 20

time_r <- c()
time_cpp <- c()

for (i in 1:n){
  time_r <- c(time_r, mean(microbenchmark(fib_r = nth_fibonacci(i, .Machine$double.xmin), times = 3, uni
  time_cpp <- c(time_cpp, mean(microbenchmark(fib_cpp = nth_fibonacci_cpp(i, .Machine$double.xmin), time
}
```

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.

```
ggplot() +
  geom_line(aes(x = 1:n, y = log(time_r), col = "time_r")) +
  geom_line(aes(x = 1:n, y = log(time_cpp), col = "time_cpp")) +
  xlab("Number of iterations in Fib function") +
  ylab("Time in Seconds")
```

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)
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())
```

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

```
storms %>%
  filter(year >= 1970 & year < 1980)
```

```
## # A tibble: 546 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 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 %>%
  filter(category >= 4 & wind >= 100)
```

```
## # A tibble: 416 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 Anita hurricane 5         1977     9     2     0  24.6 -96.2   140      931
##  2 Anita hurricane 5         1977     9     2     6  24.2 -97.1   150      926
##  3 Anita hurricane 4         1977     9     2    12  23.7 -98     120      940
##  4 David hurricane 4         1979     8    28     0  12.2 -52.9   115      947
##  5 David hurricane 4         1979     8    28     6  12.5 -54.4   125      941
##  6 David hurricane 4         1979     8    28    12  12.8 -55.7   130      938
##  7 David hurricane 4         1979     8    28    18  13.2 -56.9   125      941
##  8 David hurricane 4         1979     8    29     0  13.7 -58     120      944
##  9 David hurricane 4         1979     8    29     6  14.2 -59.2   120      942
## 10 David hurricane 4         1979     8    29    12  14.8 -60.3   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)
```

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 %<>%
  mutate(average_diameter = ifelse(!is.na(ts_diameter) & !is.na(hu_diameter), (ts_diameter + hu_diamete
                                   ifelse(is.na(ts_diameter), hu_diameter,
                                   ifelse(is.na(hu_diameter), ts_diameter, NA))))
```

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(maximum_wind_speed = max(wind, na.rm = TRUE))
```

```
## # A tibble: 198 x 2
##    name     maximum_wind_speed
##    <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(maximum_wind_by_storm = max(wind, na.rm = TRUE)) %>%
  select(name, maximum_wind_by_storm, everything()) %>%
  arrange(maximum_wind_by_storm, year, month, day, hour)
```

```
## # A tibble: 10,010 x 16
## # Groups:   name [198]
##    name   maximum_wind_by_~ status  category  year month   day  hour   lat  long
##    <chr>              <int> <chr>   <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 AL101~                25 tropic~ -1        1991    10    24    12  13.4 -42.3
##  2 AL101~                25 tropic~ -1        1991    10    24    18  13.7 -43.6
##  3 AL101~                25 tropic~ -1        1991    10    25     0  13.8 -44.9
##  4 AL101~                25 tropic~ -1        1991    10    25     6  14   -46.4
##  5 AL101~                25 tropic~ -1        1991    10    25    12  14.1 -47.7
```

```
##   6 AL012~                25 tropic~ -1        2000     6     7    18  21   -93
##   7 AL012~                25 tropic~ -1        2000     6     8     0  20.9 -92.8
##   8 AL012~                25 tropic~ -1        2000     6     8     6  20.7 -93.1
##   9 AL012~                25 tropic~ -1        2000     6     8    12  20.8 -93.5
##  10 AL022~                25 tropic~ -1        2001     7    11    18  10.9 -42.1
## # ... with 10,000 more rows, and 6 more variables: wind <int>, pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_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).

```
storms %>%
  group_by(name) %>%
  summarize(max_category = max(category),
            max_wind_speed = max(wind),
            max_pressure = max(pressure),
            max_ts_diam = max(ts_diameter),
            max_hu_diam = max(hu_diameter)) %>%
  na.omit()
```

```
## # A tibble: 54 x 6
##    name     max_category max_wind_speed max_pressure max_ts_diam max_hu_diam
##    <chr>    <ord>                 <int>        <int>       <dbl>       <dbl>
##  1 AL022006 0                        45         1008        69.0           0
##  2 AL102004 -1                       30         1013         0             0
##  3 Al202011 0                        40         1011        69.0           0
##  4 Andrea   0                        55         1006       207.            0
```

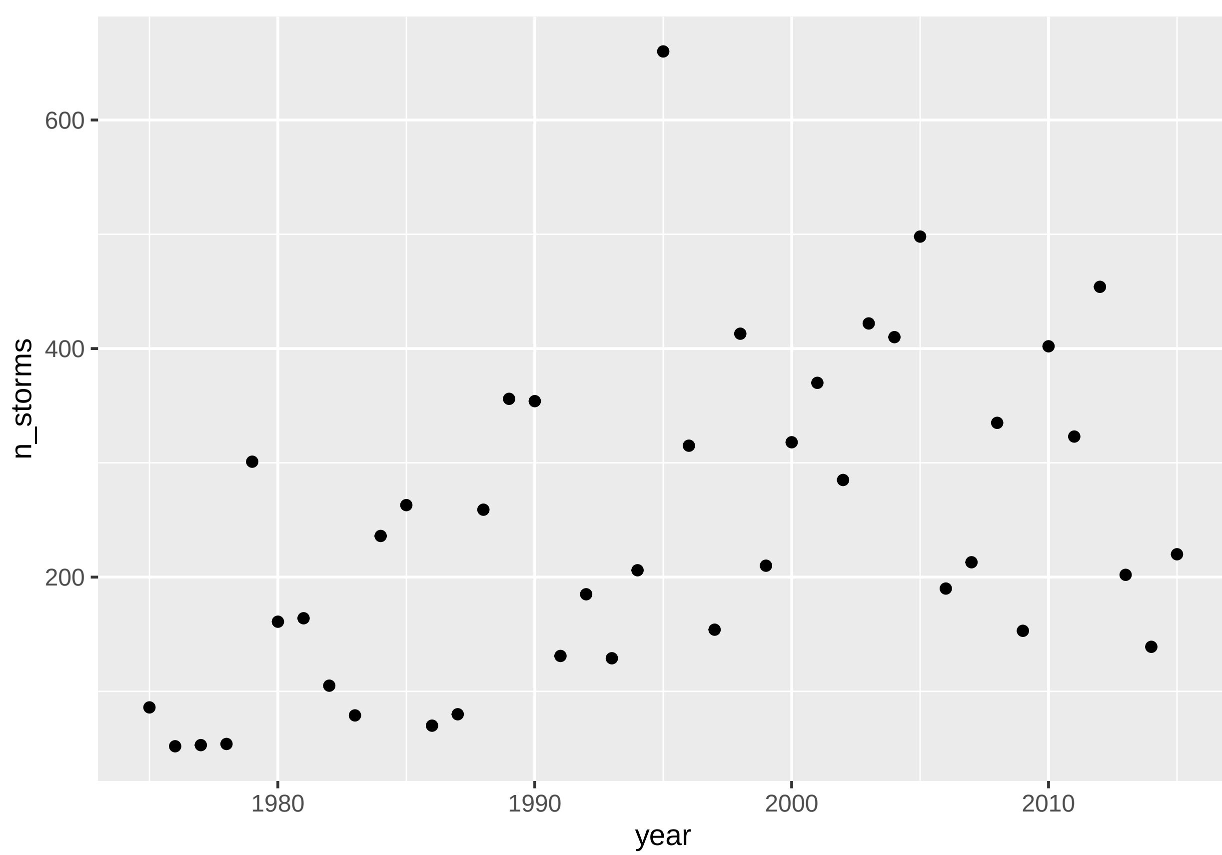
```
##   5 Beta      3                              100         1007        127.          34.5
##   6 Colin     0                               50         1013        104.           0
##   7 Don       0                               45         1007         69.0          0
##   8 Dorian    0                               50         1013         80.6          0
##   9 Eight     -1                              30         1009          0            0
## 10 Epsilon   1                               75         1005        276.          63.3
## # ... with 44 more rows
```

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?

There have been more storms more recently.

```
pacman::p_load(ggplot2)

storms %>%
  group_by(year) %>%
  summarize(n_storms = n()) %>%
  ggplot() +
  geom_point(aes(y = n_storms, x = year))
```



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

```
storms %>%
  group_by(year, category) %>%
  summarize(num_storms = n_distinct(name))
```

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

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

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

```
storms %>%
  group_by(year, status) %>%
  summarize(max_wind = max(wind))
```

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

```
## # A tibble: 123 x 3
## # Groups:   year [41]
##     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))
```

```
## # A tibble: 198 x 3
##    name     avg_lat avg_long
##    <chr>      <dbl>    <dbl>
##  1 AL011993    24.7    -78.0
```

```
##  2 AL012000   20.8     -93.1
##  3 AL021992   26.7     -84.5
##  4 AL021994   33.6     -79.7
##  5 AL021999   20.4     -96.4
##  6 AL022000    9.9     -28.5
##  7 AL022001   11.9     -45.3
##  8 AL022003    9.62    -43.4
##  9 AL022006   41.3     -63.5
## 10 AL031987   30.8     -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 %>%
  group_by(category, name) %>%
  slice(1) %>%
  group_by(category) %>%
  mutate(storm_number = dense_rank(paste(year, as.numeric(month), day))) %>%
  select(category, storm_number, year, month, day, name) %>%
  distinct %>%
  arrange(category, storm_number)
```

```
## # A tibble: 687 x 6
## # Groups:   category [7]
##    category storm_number  year month   day name
##    <ord>           <int> <dbl> <dbl> <int> <chr>
##  1 -1                  1  1975     6    27 Amy
##  2 -1                  2  1975     8    24 Caroline
##  3 -1                  3  1976     8     6 Belle
```

```
##   4 -1                    4  1976     9     26 Gloria
##   5 -1                    5  1977    10     13 Evelyn
##   6 -1                    6  1977     8     29 Anita
##   7 -1                    7  1977     9      5 Clara
##   8 -1                    8  1978    10      7 Juliet
##   9 -1                    9  1978     7     30 Amelia
##  10 -1                   10  1978     8      5 Bess
## # ... with 677 more rows
```

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())
```

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(day_of_week = weekdays(timestamp),
         week_of_year = week(timestamp))
```

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

```
storms %>%
  group_by(name) %>%
  summarize(start_date = min(timestamp)) %>%
  mutate(start_date = paste(weekdays(start_date),
                          paste(months(start_date), day(start_date), sep = " "),
                          year(start_date), sep = ", "))
```

```
## # A tibble: 198 x 2
##    name     start_date
##    <chr>    <chr>
##  1 AL011993 Monday, May 31, 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.

```
bins <- 0:10

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

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

```
serious_storms <- storms %>%
  filter(category >= 3)

serious_storms
```

```
## # A tibble: 779 x 19
##    timestamp           name  status category  year month   day  hour   lat  long
##    <dttm>              <chr> <chr>  <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 1975-08-31 00:00:00 Caro~ hurri~ 3         1975     8    31     0  24   -97
##  2 1975-08-31 06:00:00 Caro~ hurri~ 3         1975     8    31     6  24.1 -97.5
##  3 1976-08-08 18:00:00 Belle hurri~ 3         1976     8     8    18  29.5 -75.3
##  4 1976-08-09 00:00:00 Belle hurri~ 3         1976     8     9     0  30.9 -75.3
##  5 1976-08-09 06:00:00 Belle hurri~ 3         1976     8     9     6  32.5 -75.2
##  6 1977-09-01 18:00:00 Anita hurri~ 3         1977     9     1    18  25.2 -95.5
##  7 1977-09-02 00:00:00 Anita hurri~ 5         1977     9     2     0  24.6 -96.2
##  8 1977-09-02 06:00:00 Anita hurri~ 5         1977     9     2     6  24.2 -97.1
##  9 1977-09-02 12:00:00 Anita hurri~ 4         1977     9     2    12  23.7 -98
## 10 1979-08-28 00:00:00 David hurri~ 4         1979     8    28     0  12.2 -52.9
## # ... with 769 more rows, and 9 more variables: wind <int>, pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>, day_of_week <chr>, week_of_year <dbl>,
## #   decile_windspeed <fct>
```

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 = " / ")
```

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_ts_diam = mean(ts_diameter, na.rm = TRUE),
            avg_hu_diam = mean(hu_diameter, na.rm = TRUE))
```

```
## # A tibble: 7 x 5
##   category avg_wind_speed avg_pressure avg_ts_diam avg_hu_diam
##   <ord>             <dbl>        <dbl>       <dbl>       <dbl>
## 1 -1                 27.3        1008.          0           0
## 2 0                  45.8         999.        160.          0
```

```
## 3 1                  70.9         982.        278.        57.3
## 4 2                  89.4         967.        282.        78.8
## 5 3                 105.          954.        307.        91.4
## 6 4                 122.          940.        315.       102.
## 7 5                 145.          916.        317.       120.
```

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).

```
storms %<>%
  filter(!is.na(ts_diameter), !is.na(hu_diameter)) %>%
  group_by(name) %>%
  mutate(max_category = max(category),
            max_wind = max(wind),
            max_pressure = max(pressure),
            max_ts_diam = max(ts_diameter),
            max_hu_diam = max(hu_diameter))
```

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)

distance <- function(lat1, long1, lat2, long2){

  lat1 = lat1 * 180/pi
  lat2 = lat2 * 180/pi
  long1 = long1 * 180/pi
  long2 = long2 * 180/pi
  # Haversine formula
  a = sin(lat2 - lat1 / 2)^2 + (cos(lat2) * cos(lat1)) * sin(long2 - long1 / 2)^2
  b = 2 * atan2(sqrt(a), sqrt(1 - a))
  distance = 6373.0 * b # Multiplying by radius of earth in KM
  return(distance)
}



storms %>%
  mutate(distance_to_miami = distance(lat, long, MIAMI_LAT_LONG_COORDS[1], MIAMI_LAT_LONG_COORDS[2]))
```

```
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
```

```
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
```

```
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## # A tibble: 3,482 x 25
## # Groups:   name [114]
##    timestamp           name  status category  year month   day  hour   lat  long
##    <dttm>              <chr> <chr>  <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 2004-07-31 18:00:00 Alex  tropi~ -1        2004     7    31    18  30.3 -78.3
##  2 2004-08-01 00:00:00 Alex  tropi~ -1        2004     8     1     0  31   -78.8
##  3 2004-08-01 06:00:00 Alex  tropi~ -1        2004     8     1     6  31.5 -79
##  4 2004-08-01 12:00:00 Alex  tropi~ -1        2004     8     1    12  31.6 -79.1
##  5 2004-08-01 18:00:00 Alex  tropi~ 0         2004     8     1    18  31.6 -79.2
##  6 2004-08-02 00:00:00 Alex  tropi~ 0         2004     8     2     0  31.5 -79.3
##  7 2004-08-02 06:00:00 Alex  tropi~ 0         2004     8     2     6  31.4 -79.4
##  8 2004-08-02 12:00:00 Alex  tropi~ 0         2004     8     2    12  31.3 -79
##  9 2004-08-02 18:00:00 Alex  tropi~ 0         2004     8     2    18  31.8 -78.7
## 10 2004-08-03 00:00:00 Alex  tropi~ 0         2004     8     3     0  32.4 -78.2
## # ... with 3,472 more rows, and 15 more variables: wind <int>, pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>, day_of_week <chr>, week_of_year <dbl>,
## #   decile_windspeed <fct>, max_category <ord>, max_wind <int>,
## #   max_pressure <int>, max_ts_diam <dbl>, max_hu_diam <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.

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

```
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
```

```
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(1 - a): NaNs produced
## Warning in sqrt(a): NaNs produced
## Warning in sqrt(a): NaNs produced
```

```
## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced

## Warning in sqrt(a): NaNs produced

## Warning in sqrt(1 - a): NaNs produced
```

```
head(storms)
```

```
## # A tibble: 6 x 25
## # Groups:   name [1]
##   timestamp           name  status  category  year month   day  hour   lat  long
##   <dttm>              <chr> <chr>   <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 2004-07-31 18:00:00 Alex  tropic~ -1        2004     7    31    18  30.3 -78.3
## 2 2004-08-01 00:00:00 Alex  tropic~ -1        2004     8     1     0  31   -78.8
## 3 2004-08-01 06:00:00 Alex  tropic~ -1        2004     8     1     6  31.5 -79
## 4 2004-08-01 12:00:00 Alex  tropic~ -1        2004     8     1    12  31.6 -79.1
## 5 2004-08-01 18:00:00 Alex  tropic~ 0         2004     8     1    18  31.6 -79.2
## 6 2004-08-02 00:00:00 Alex  tropic~ 0         2004     8     2     0  31.5 -79.3
## # ... with 15 more variables: wind <int>, pressure <int>, ts_diameter <dbl>,
## #   hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>, day_of_week <chr>, week_of_year <dbl>,
## #   decile_windspeed <fct>, max_category <ord>, max_wind <int>,
## #   max_pressure <int>, max_ts_diam <dbl>, max_hu_diam <dbl>,
## #   dist_from_prev <dbl>
```

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) %>%
  summarize(Distance = sum(dist_from_prev),
            Displacement = paste(round(last(lat) - first(lat), 2), round(last(long) - first(long), 2), s
```

```
## # A tibble: 114 x 3
##    name     Distance Displacement
##    <chr>       <dbl> <chr>
##  1 AL022006   24361. 4.6 / 6.3
##  2 AL102004   36454. 4.7 / 5.4
##  3 Al202011   29375. 1.7 / 2.1
##  4 Alberto   216936. 11.5 / 8.9
##  5 Alex      389785. -7.1 / -23.6
```

```
##   6 Ana       220407. 22.6 / -48.4
##   7 Andrea     30050. 8.4 / 6.4
##   8 Arthur    204244. 24.8 / 19.9
##   9 Barry     168924. -2.7 / -11.2
##  10 Beryl     182274. 1.4 / -5.6
## # ... with 104 more rows
```

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

```
storms %<>%
  mutate(speed = dist_from_prev / 6)

head(storms)
```

```
## # A tibble: 6 x 26
## # Groups:   name [1]
##   timestamp           name  status  category  year month   day  hour   lat  long
##   <dttm>              <chr> <chr>   <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 2004-07-31 18:00:00 Alex  tropic~ -1        2004     7    31    18  30.3 -78.3
## 2 2004-08-01 00:00:00 Alex  tropic~ -1        2004     8     1     0  31   -78.8
## 3 2004-08-01 06:00:00 Alex  tropic~ -1        2004     8     1     6  31.5 -79
## 4 2004-08-01 12:00:00 Alex  tropic~ -1        2004     8     1    12  31.6 -79.1
## 5 2004-08-01 18:00:00 Alex  tropic~ 0         2004     8     1    18  31.6 -79.2
## 6 2004-08-02 00:00:00 Alex  tropic~ 0         2004     8     2     0  31.5 -79.3
## # ... with 16 more variables: wind <int>, pressure <int>, ts_diameter <dbl>,
## #   hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>, day_of_week <chr>, week_of_year <dbl>,
## #   decile_windspeed <fct>, max_category <ord>, max_wind <int>,
## #   max_pressure <int>, max_ts_diam <dbl>, max_hu_diam <dbl>,
## #   dist_from_prev <dbl>, speed <dbl>
```

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) %>%
  summarize(avg_ground_speed = mean(speed))
```

```
## # A tibble: 114 x 2
##    name     avg_ground_speed
##    <chr>               <dbl>
##  1 AL022006             812.
##  2 AL102004             759.
##  3 Al202011             612.
##  4 Alberto             1205.
##  5 Alex                1249.
##  6 Ana                 1361.
##  7 Andrea               556.
##  8 Arthur              1174.
##  9 Barry               1224.
## 10 Beryl               1125.
## # ... with 104 more rows
```

```r
head(storms)
```

```
## # A tibble: 6 x 26
## # Groups:   name [1]
##   timestamp           name  status  category  year month   day  hour   lat  long
##   <dttm>              <chr> <chr>   <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 2004-07-31 18:00:00 Alex  tropic~ -1        2004     7    31    18  30.3 -78.3
## 2 2004-08-01 00:00:00 Alex  tropic~ -1        2004     8     1     0  31   -78.8
## 3 2004-08-01 06:00:00 Alex  tropic~ -1        2004     8     1     6  31.5 -79
## 4 2004-08-01 12:00:00 Alex  tropic~ -1        2004     8     1    12  31.6 -79.1
## 5 2004-08-01 18:00:00 Alex  tropic~ 0         2004     8     1    18  31.6 -79.2
## 6 2004-08-02 00:00:00 Alex  tropic~ 0         2004     8     2     0  31.5 -79.3
## # ... with 16 more variables: wind <int>, pressure <int>, ts_diameter <dbl>,
## #   hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>, day_of_week <chr>, week_of_year <dbl>,
## #   decile_windspeed <fct>, max_category <ord>, max_wind <int>,
## #   max_pressure <int>, max_ts_diam <dbl>, max_hu_diam <dbl>,
## #   dist_from_prev <dbl>, speed <dbl>
```

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

```r
speed_and_category <- storms %>%
  group_by(name) %>%
  summarize(avg_ground_speed = mean(speed), maximum_category = as.numeric(max(category)))

cor(speed_and_category[,2], speed_and_category[,3])
```

```
##                  maximum_category
## avg_ground_speed        0.2531993
```

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.

```r
storms_data = storms %>%
  group_by(name) %>% # Because we only want one data point per storm
  summarize(y = max(wind), # Dependent variable
            max_category = max(category), # Independent variables
            pressure = max(pressure),
            speed = max(speed),
            total_distance_traveled = sum(dist_from_prev),
            status = last(status)) %>%
  select(-name) # Remove name feature because it is meaningless

head(storms_data)
```

```
## # A tibble: 6 x 6
```

```
##        y max_category pressure speed total_distance_traveled status
##    <int> <ord>           <int> <dbl>                   <dbl> <chr>
## 1    45 0                 1008 2210.                  24361. tropical storm
## 2    30 -1                1013 2907.                  36454. tropical depression
## 3    40 0                 1011 1728.                  29375. tropical storm
## 4    60 0                 1008 2603.                 216936. tropical storm
## 5   105 3                 1010 3184.                 389785. tropical depression
## 6    50 0                 1012 2672.                 220407. tropical depression
```

Fit your model. Validate it.

```r
# OLS model
n = nrow(storms_data)
K = 5 # 1/5 of data will be set to the testing set
test_indices = sample(1 : n, 1 / K * n)
train_indices = setdiff(1:n, test_indices)

X = select(storms_data, -y)
y = storms_data$y

X_test = X[test_indices,]
y_test = y[test_indices]
X_train = X[train_indices,]
y_train = y[train_indices]

modv = lm(y_train ~ ., data.frame(X_train))
yhat = predict(modv, data.frame(X_train))

y_bar = sum(y)/n
e = y - yhat
```

```
## Warning in y - yhat: longer object length is not a multiple of shorter object
## length
```

```r
SSE = sum(e^2)
SST = sum((y - y_bar)^2)
MSE = SSE / (n-2)
RMSE = sqrt(MSE)
Rsq = 1 - (SSE/SST)

metrics = list("y_bar" = y_bar, "SSE" = SSE, "SST" = SST, "RMSE" = RMSE, "Rsq" = Rsq)
metrics
```

```
## $y_bar
## [1] 75.61404
##
## $SSE
## [1] 240121.4
##
## $SST
## [1] 133257
##
## $RMSE
```

```
## [1] 46.30271
##
## $Rsq
## [1] -0.8019417
```

Assess your level of success at this endeavor.

$y_bar [1] 75.61404

$SSE [1] 230023.3

$SST [1] 133257

$RMSE [1] 45.31864

$Rsq [1] -0.7261624

These are the metrics the model achieved after a first run. The objective was to predict the maximum wind speed of a storm based on a few causal features. The model could predict the maximum wind speed of a storm with a margin of +- 45.3 miles per hour. Considering the storms can range with wind speeds between 15 and 160 miles per hour, for the model to make predictions with such a margin makes this an inadequate model. The model achieved an Rsq score of -0.72 with only 5 features, meaning the features are not explaining the variance within y well at all. This model is clearly poor at its task.

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(1)
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$marital_status == "Ma
adult$marital_status = as.factor(adult$marital_status)

adult$education = as.character(adult$education)
adult$education = ifelse(adult$education == "1st-4th" | adult$education == "Preschool", "<=4th", adult$
adult$education = as.factor(adult$education)
adult$education = NULL

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_countr
adult$native_country= as.factor(adult$native_country)

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

```
p_hat_train = predict(logistic_mod, Xtrain, type = 'response')
```

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

```
#in sample log scoring rule
mean(ytrain * log(p_hat_train) + (1 - ytrain) * log(1 - p_hat_train))
```

```
## [1] -0.2671121
```

```
#in sample Brier scoring rule
mean(-(ytrain - p_hat_train)^2)
```

```
## [1] -0.08715781
```

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

dim(Xmm_train)
```

```
## [1] 1000   93
```

```
dim(Xmm_select)
```

```
## [1] 1000   93
```

```
dim(Xmm_test)
```

```
## [1] 1000   93
```

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.

```
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 adde

  i = i + 1

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

```
## i = 1 in-sample_brier = 0.181356 oos_brier = 0.185548
##    predictor added: X.Intercept.
## i = 2 in-sample_brier = 0.181356 oos_brier = 0.185548
##    predictor added: native_countryPoland
## i = 3 in-sample_brier = 0.181356 oos_brier = 0.185548
##    predictor added: statusNot.in.family.Married
## i = 4 in-sample_brier = 0.181356 oos_brier = 0.185548
##    predictor added: statusOther.relative.Separated
## i = 5 in-sample_brier = 0.181356 oos_brier = 0.185548
##    predictor added: statusOther.relative.Widowed
## i = 6 in-sample_brier = 0.181356 oos_brier = 0.185548
##    predictor added: statusOwn.child.Widowed
## i = 7 in-sample_brier = 0.1813554 oos_brier = 0.1855417
##    predictor added: worktypeTransport.moving.Self.emp.not.inc
## i = 8 in-sample_brier = 0.1813548 oos_brier = 0.1855661
##    predictor added: statusUnmarried.Married.spouse.absent
## i = 9 in-sample_brier = 0.1813542 oos_brier = 0.1855927
##    predictor added: worktypeSales.Self.emp.not.inc
## i = 10 in-sample_brier = 0.181353 oos_brier = 0.1856649
##    predictor added: statusUnmarried.Widowed
## i = 11 in-sample_brier = 0.1813499 oos_brier = 0.1856563
##    predictor added: worktypeCraft.repair.Private
```

```
## i = 12 in-sample_brier = 0.1813447 oos_brier = 0.1856134
##    predictor added: native_countryIndia
## i = 13 in-sample_brier = 0.1813373 oos_brier = 0.1856355
##    predictor added: native_countryPuerto.Rico
## i = 14 in-sample_brier = 0.1813246 oos_brier = 0.1859607
##    predictor added: worktypeFarming.fishing.Private
## i = 15 in-sample_brier = 0.1813123 oos_brier = 0.1857883
##    predictor added: worktypeFarming.fishing.Self.emp.not.inc
## i = 16 in-sample_brier = 0.1812982 oos_brier = 0.1856838
##    predictor added: statusNot.in.family.Separated
## i = 17 in-sample_brier = 0.1812717 oos_brier = 0.1852927
##    predictor added: worktypeProf.specialty.Federal.gov
## i = 18 in-sample_brier = 0.1812449 oos_brier = 0.1853558
##    predictor added: native_countryGuatemala
## i = 19 in-sample_brier = 0.181218 oos_brier = 0.1857469
##    predictor added: worktypeCraft.repair.Local.gov
## i = 20 in-sample_brier = 0.1811902 oos_brier = 0.1856173
##    predictor added: raceOther
## i = 21 in-sample_brier = 0.1811586 oos_brier = 0.1855962
##    predictor added: worktypeExec.managerial.State.gov
## i = 22 in-sample_brier = 0.1811215 oos_brier = 0.1859505
##    predictor added: worktypeAdm.clerical.Local.gov
## i = 23 in-sample_brier = 0.1810644 oos_brier = 0.185881
##    predictor added: native_countryDominican.Republic
## i = 24 in-sample_brier = 0.1810644 oos_brier = 0.185881
##    predictor added: statusOwn.child.Married.spouse.absent
## i = 25 in-sample_brier = 0.1810073 oos_brier = 0.1858114
##    predictor added: native_countryVietnam
## i = 26 in-sample_brier = 0.1809499 oos_brier = 0.1860419
##    predictor added: statusOwn.child.Married
## i = 27 in-sample_brier = 0.1808553 oos_brier = 0.1860526
##    predictor added: native_countryOther
## i = 28 in-sample_brier = 0.1807887 oos_brier = 0.1862179
##    predictor added: native_countryUnited.States
## i = 29 in-sample_brier = 0.180699 oos_brier = 0.1868485
##    predictor added: worktypeTech.support.Private
## i = 30 in-sample_brier = 0.1805934 oos_brier = 0.1864382
##    predictor added: worktypeOther.service.Local.gov
## i = 31 in-sample_brier = 0.1804642 oos_brier = 0.1848996
##    predictor added: worktypeExec.managerial.Self.emp.inc
## i = 32 in-sample_brier = 0.1803137 oos_brier = 0.1846994
##    predictor added: native_countryJapan
## i = 33 in-sample_brier = 0.1801419 oos_brier = 0.1849772
##    predictor added: worktypeProtective.serv.State.gov
## i = 34 in-sample_brier = 0.1799592 oos_brier = 0.1847671
##    predictor added: statusOther.relative.Divorced
## i = 35 in-sample_brier = 0.179768 oos_brier = 0.1846089
##    predictor added: worktypeProtective.serv.Private
## i = 36 in-sample_brier = 0.1795723 oos_brier = 0.1842935
##    predictor added: worktypeProf.specialty.Local.gov
## i = 37 in-sample_brier = 0.179356 oos_brier = 0.1841564
##    predictor added: native_countryChina
## i = 38 in-sample_brier = 0.1791469 oos_brier = 0.1840683
##    predictor added: native_countryColumbia
```

```
## i = 39 in-sample_brier = 0.1789191 oos_brier = 0.1840311
##    predictor added: worktypeOther.service.State.gov
## i = 40 in-sample_brier = 0.1786884 oos_brier = 0.1838212
##    predictor added: statusOwn.child.Divorced
## i = 41 in-sample_brier = 0.1784501 oos_brier = 0.1838435
##    predictor added: native_countryEl.Salvador
## i = 42 in-sample_brier = 0.1782627 oos_brier = 0.1844303
##    predictor added: statusOther.relative.Married.spouse.absent
## i = 43 in-sample_brier = 0.1780273 oos_brier = 0.1841625
##    predictor added: worktypeTransport.moving.Local.gov
## i = 44 in-sample_brier = 0.1777802 oos_brier = 0.1838986
##    predictor added: worktypeCraft.repair.Self.emp.not.inc
## i = 45 in-sample_brier = 0.1775394 oos_brier = 0.1839145
##    predictor added: worktypeSales.Self.emp.inc
## i = 46 in-sample_brier = 0.1772784 oos_brier = 0.184464
##    predictor added: worktypeAdm.clerical.State.gov
## i = 47 in-sample_brier = 0.1770012 oos_brier = 0.1848479
##    predictor added: native_countryEngland
## i = 48 in-sample_brier = 0.1766289 oos_brier = 0.1852858
##    predictor added: native_countryItaly
## i = 49 in-sample_brier = 0.1762576 oos_brier = 0.1850986
##    predictor added: worktypeTransport.moving.Private
## i = 50 in-sample_brier = 0.1759073 oos_brier = 0.185645
##    predictor added: statusOther.relative.Married
## i = 51 in-sample_brier = 0.1755777 oos_brier = 0.1855656
##    predictor added: worktypePriv.house.serv.Private
## i = 52 in-sample_brier = 0.1752024 oos_brier = 0.1858937
##    predictor added: worktypeOther
## i = 53 in-sample_brier = 0.1748781 oos_brier = 0.1858285
##    predictor added: native_countryGermany
## i = 54 in-sample_brier = 0.1744952 oos_brier = 0.1864225
##    predictor added: native_countryCuba
## i = 55 in-sample_brier = 0.1741871 oos_brier = 0.186287
##    predictor added: statusOwn.child.Separated
## i = 56 in-sample_brier = 0.1737656 oos_brier = 0.1862193
##    predictor added: native_countrySouth
## i = 57 in-sample_brier = 0.1733164 oos_brier = 0.1853527
##    predictor added: worktypeOther.service.Self.emp.not.inc
## i = 58 in-sample_brier = 0.1728051 oos_brier = 0.1853208
##    predictor added: worktypeProf.specialty.Self.emp.inc
## i = 59 in-sample_brier = 0.1722497 oos_brier = 0.1846987
##    predictor added: worktypeSales.Private
## i = 60 in-sample_brier = 0.1717164 oos_brier = 0.1863781
##    predictor added: worktypeProtective.serv.Local.gov
## i = 61 in-sample_brier = 0.1711044 oos_brier = 0.1860013
##    predictor added: statusNot.in.family.Widowed
## i = 62 in-sample_brier = 0.1705002 oos_brier = 0.1857051
##    predictor added: worktypeExec.managerial.Self.emp.not.inc
## i = 63 in-sample_brier = 0.1698833 oos_brier = 0.1865027
##    predictor added: native_countryJamaica
## i = 64 in-sample_brier = 0.1693691 oos_brier = 0.1866908
##    predictor added: raceWhite
## i = 65 in-sample_brier = 0.1686613 oos_brier = 0.1859704
##    predictor added: statusUnmarried.Separated
```

```
## i = 66 in-sample_brier = 0.1678313 oos_brier = 0.1864843
##    predictor added: raceBlack
## i = 67 in-sample_brier = 0.1671104 oos_brier = 0.1841216
##    predictor added: worktypeMachine.op.inspct.Private
## i = 68 in-sample_brier = 0.1664096 oos_brier = 0.1846154
##    predictor added: raceAsian.Pac.Islander
## i = 69 in-sample_brier = 0.165671 oos_brier = 0.1834925
##    predictor added: worktypeProf.specialty.Self.emp.not.inc
## i = 70 in-sample_brier = 0.164799 oos_brier = 0.1839977
##    predictor added: native_countryPhilippines
## i = 71 in-sample_brier = 0.1639532 oos_brier = 0.1829634
##    predictor added: statusOther.relative.Never.married
## i = 72 in-sample_brier = 0.1630177 oos_brier = 0.1798843
##    predictor added: worktypeProf.specialty.Private
## i = 73 in-sample_brier = 0.161836 oos_brier = 0.178388
##    predictor added: worktypeHandlers.cleaners.Private
## i = 74 in-sample_brier = 0.1604635 oos_brier = 0.1780931
##    predictor added: worktypeExec.managerial.Local.gov
## i = 75 in-sample_brier = 0.1590754 oos_brier = 0.1803847
##    predictor added: native_countryMexico
## i = 76 in-sample_brier = 0.1576239 oos_brier = 0.18131
##    predictor added: statusNot.in.family.Married.spouse.absent
## i = 77 in-sample_brier = 0.1561724 oos_brier = 0.1814974
##    predictor added: worktypeExec.managerial.Federal.gov
## i = 78 in-sample_brier = 0.154877 oos_brier = 0.1792748
##    predictor added: worktypeAdm.clerical.Private
## i = 79 in-sample_brier = 0.1530984 oos_brier = 0.1792153
##    predictor added: worktypeProf.specialty.State.gov
## i = 80 in-sample_brier = 0.1512046 oos_brier = 0.1803241
##    predictor added: statusUnmarried.Divorced
## i = 81 in-sample_brier = 0.1486265 oos_brier = 0.1798221
##    predictor added: statusUnmarried.Never.married
## i = 82 in-sample_brier = 0.1455114 oos_brier = 0.1793399
##    predictor added: statusWife.Married
## i = 83 in-sample_brier = 0.141789 oos_brier = 0.179233
##    predictor added: statusNot.in.family.Divorced
## i = 84 in-sample_brier = 0.1375809 oos_brier = 0.1772499
##    predictor added: capital_loss
## i = 85 in-sample_brier = 0.1330105 oos_brier = 0.1663411
##    predictor added: hours_per_week
## i = 86 in-sample_brier = 0.1290151 oos_brier = 0.1591097
##    predictor added: worktypeExec.managerial.Private
## i = 87 in-sample_brier = 0.1283621 oos_brier = 0.1569123
##    predictor added: worktypeOther.service.Private
## i = 88 in-sample_brier = 0.1242607 oos_brier = 0.1476126
##    predictor added: education_num
## i = 89 in-sample_brier = 0.1209538 oos_brier = 0.1422338
##    predictor added: statusOwn.child.Never.married
## i = 90 in-sample_brier = 0.1133092 oos_brier = 0.1362918
##    predictor added: sexMale
## i = 91 in-sample_brier = 0.1027663 oos_brier = 0.1329848
##    predictor added: statusNot.in.family.Never.married
## i = 92 in-sample_brier = 0.09516563 oos_brier = 0.1313902
##    predictor added: age
```

```
## i = 93 in-sample_brier = 0.08715781 oos_brier = 0.1264595
##    predictor added: capital_gain
```

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

```r
simulation_results = data.frame(
  iteration = 1 : length(in_sample_brier_by_iteration),
  in_sample_brier_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), color = "red") +
  geom_line(aes(x = iteration, y = oos_brier_by_iteration), color = "blue") +
  #ylim(0, max(c(simulation_results$in_sample_brier_by_iteration, simulation_results$oos_brier_by_itera
  ylab(TeX("$brier score$"))
```