# Lab 7

Jacob Minkin

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.

```
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( rnorm(Nvec*n), nrow = Nvec)
```

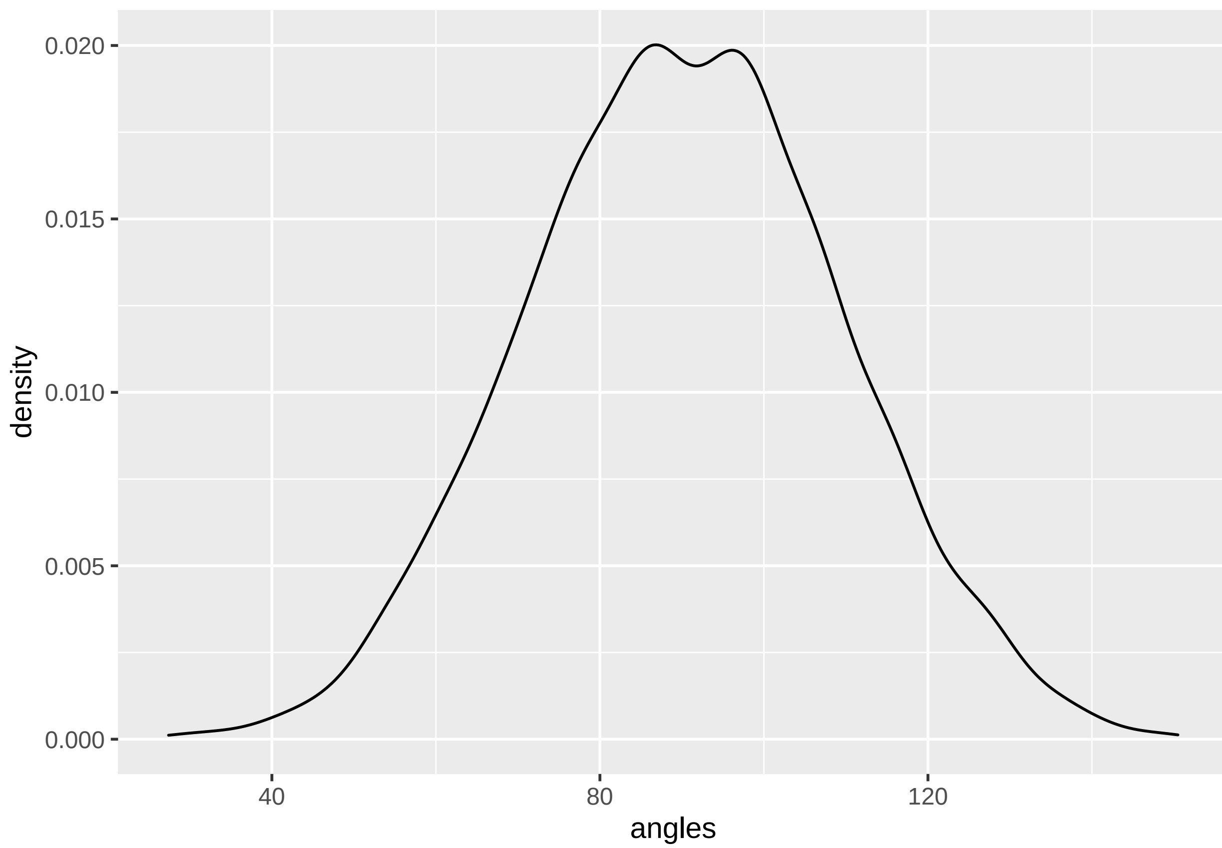
Write a function `all_angles` that measures the angle between each of the pairs of vectors. You should measure the vector on a scale of 0 to 180 degrees with negative angles coerced to be positive.

```
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
}

#all_angles(X)
```

Plot the density of these angles.

```
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.

```
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++){
        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);
          }

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

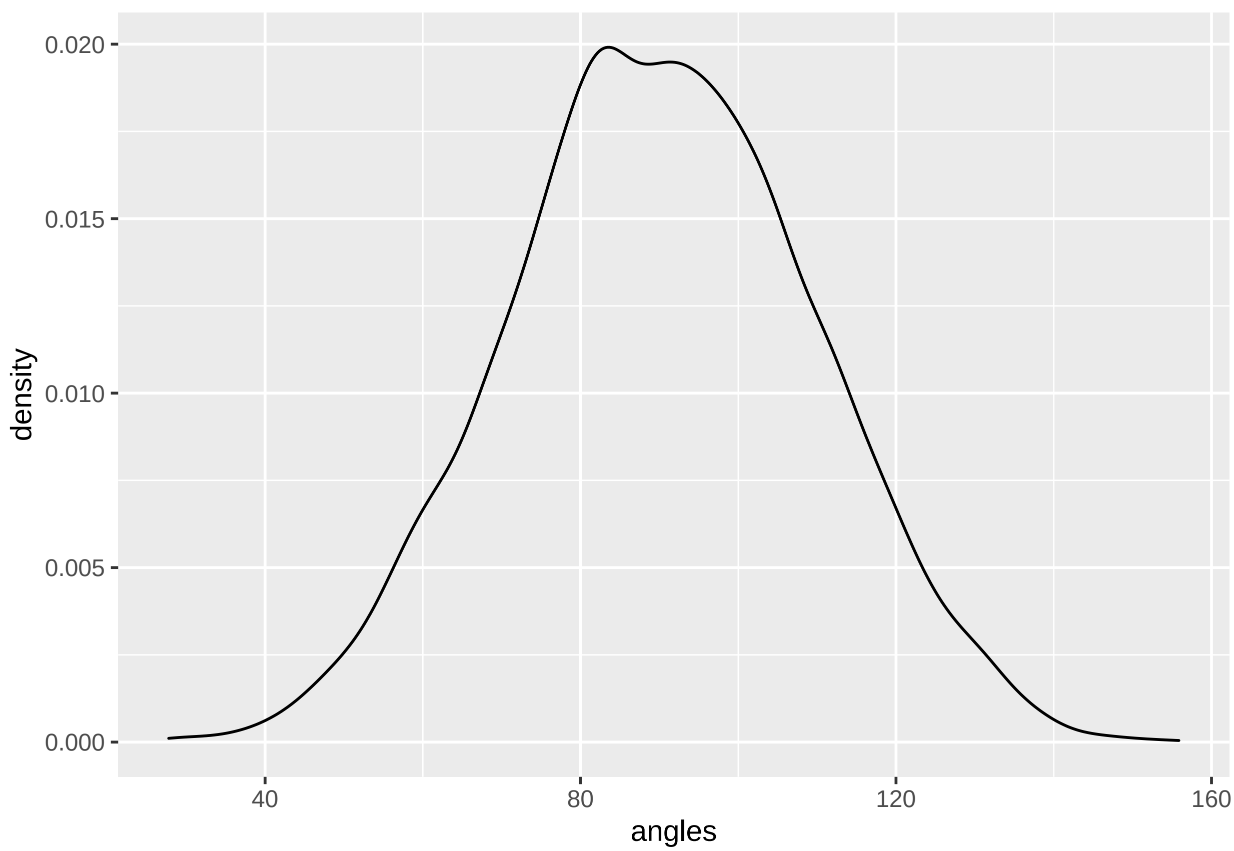
Test the time difference between these functions for `n = 1000` and `Nvec = 100, 500, 1000, 5000` using the package `microbenchmark`. Store the results in a matrix with rows representing `Nvec` and two columns for base R and Rcpp.

```
pacman::p_load(microbenchmark)

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

benchmark_data = microbenchmark(all_angles(X),all_angles_cpp(X),times = 10)
```

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 wil see later how to create "long" matrices that make such plots easier.

```
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).
```




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.

```
n = c( 10, 100, 1000)
Nvec = 10000

for (i in n){

  X = matrix( rnorm(Nvec*i), nrow = Nvec)

}
```

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_fibonnaci = function(n){
  if (n <= 1){
    return (n)
  }

  else{
    return (nth_fibonnaci(n - 1) + nth_fibonnaci(n - 2))
  }
}

nth_fibonnaci (25)
```

```
## [1] 75025
```

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('
  int nth_fibonnaci_cpp(int n)
  {
     if (n <= 1)
        return n;
    return nth_fibonnaci_cpp(n - 1) + nth_fibonnaci_cpp(n - 2);
  }
')
```

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.

```
pacman::p_load(microbenchmark)
n = seq(100, 1500, 100)
#benchmark_data_fib = microbenchmark(nth_fibonnaci(n),nth_fibonnaci_cpp(n),times = 10)
#benchmark_data_fib
```

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.

## 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)
```

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" "trop
##  $ 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 ...
```

```
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 %>%
  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 %>%
  filter(category <= 4 & wind >= 100)
```

```
## # A tibble: 711 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 Caroline  1975     8    31     0  24   -97   hurrica~ 3          100      973
##  2 Caroline  1975     8    31     6  24.1 -97.5 hurrica~ 3          100      963
##  3 Belle     1976     8     8    18  29.5 -75.3 hurrica~ 3          100      958
##  4 Belle     1976     8     9     0  30.9 -75.3 hurrica~ 3          105      957
##  5 Belle     1976     8     9     6  32.5 -75.2 hurrica~ 3          105      959
##  6 Anita     1977     9     1    18  25.2 -95.5 hurrica~ 3          110      945
##  7 Anita     1977     9     2    12  23.7 -98   hurrica~ 4          120      940
##  8 David     1979     8    28     0  12.2 -52.9 hurrica~ 4          115      947
##  9 David     1979     8    28     6  12.5 -54.4 hurrica~ 4          125      941
## 10 David     1979     8    28    12  12.8 -55.7 hurrica~ 4          130      938
## # ... with 701 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_diameter = mean( c(ts_diameter, hu_diameter), ra.rm = TRUE ))
```

```
## # 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_diameter <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_wind = max(wind, na.rm = TRUE))
```

```
## # A tibble: 198 x 2
##    name     max_wind
##    <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.

```r
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.

```r
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
```

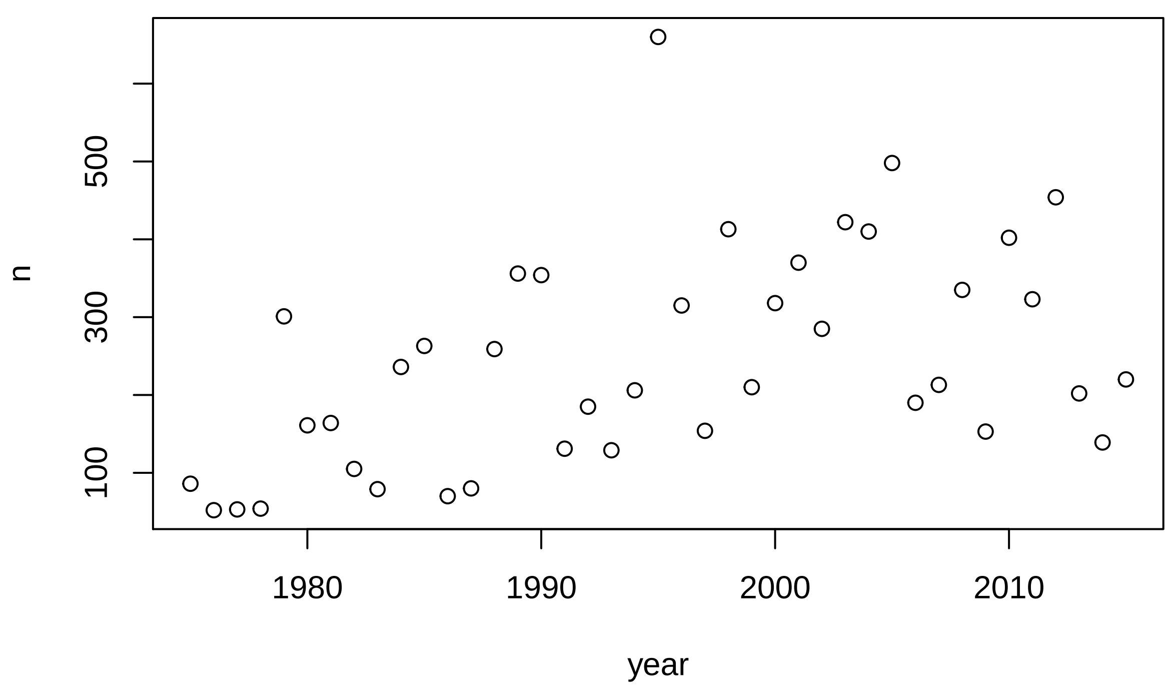
For each named storm, find its maximum category, wind speed, pressure and diameters. Do not allow the max to be NA (unless all the measurements for that storm were NA).

```r
storms %>%
  group_by(name) %>%
  arrange(name, desc(category),  desc(wind) ) %>%
  slice(1) %>%
  mutate(average_diameter = mean( c(ts_diameter, hu_diameter), ra.rm = TRUE )) %>%
  select(name, category, wind, average_diameter, hu_diameter, ts_diameter)
```

```
## # A tibble: 198 x 6
## # Groups:   name [198]
##    name     category  wind average_diameter hu_diameter ts_diameter
##    <chr>    <ord>    <int>            <dbl>       <dbl>       <dbl>
##  1 AL011993 -1          30             NA            NA        NA
##  2 AL012000 -1          25             NA            NA        NA
##  3 AL021992 -1          30             NA            NA        NA
##  4 AL021994 -1          30             NA            NA        NA
##  5 AL021999 -1          30             NA            NA        NA
##  6 AL022000 -1          30             NA            NA        NA
##  7 AL022001 -1          25             NA            NA        NA
##  8 AL022003 -1          30             NA            NA        NA
##  9 AL022006 0           45             34.5           0        69.0
## 10 AL031987 0           40             NA            NA        NA
## # ... with 188 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?

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




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

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

```
  tally (name = "number of storms by category")
```

```
## # A tibble: 233 x 3
## # Groups:   year [41]
##     year category `number of storms by category`
##    <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) %>%
  arrange(desc(status), desc(wind)) %>%
  slice(1) %>%
  select(year, status, wind)
```

```
## # A tibble: 123 x 3
## # Groups:   year, status [123]
##     year status               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) %>%
  mutate(average_lat = mean(c (lat)) ) %>%
  mutate(average_long = mean(c (long)) ) %>%
  slice(1) %>%
  select(name, average_lat, average_long)
```

```
## # A tibble: 198 x 3
## # Groups:   name [198]
##    name     average_lat average_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) %>%
  add_tally () %>%
  mutate (duration = c(n) * 6) %>%
  slice (1) %>%
  select (name, duration)
```

```
## # A tibble: 198 x 2
## # Groups:   name [198]
##    name     duration
##    <chr>       <dbl>
##  1 AL011993       48
##  2 AL012000       24
##  3 AL021992       30
##  4 AL021994       36
##  5 AL021999       24
##  6 AL022000       72
##  7 AL022001       30
##  8 AL022003       24
##  9 AL022006       30
## 10 AL031987      192
## # ... 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 ,desc(year)) %>%
  mutate (storm_num = row_number()) %>%
  select (name, category, storm_num)
```

```
## Adding missing grouping variables: `desc(year)`
```

```
## # A tibble: 10,010 x 4
## # Groups:   category, desc(year) [233]
##    `desc(year)` name  category storm_num
##           <dbl> <chr> <ord>        <int>
##  1        -1975 Amy   -1               1
##  2        -1975 Amy   -1               2
##  3        -1975 Amy   -1               3
##  4        -1975 Amy   -1               4
##  5        -1975 Amy   -1               5
##  6        -1975 Amy   -1               6
##  7        -1975 Amy   -1               7
##  8        -1975 Amy   -1               8
```

```
## 9        -1975 Amy   0                 1
## 10       -1975 Amy   0                 2
## # ... with 10,000 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 = ymd_h(paste(year, month, day, hour))) %>%
  select(name, timestamp)
```

```
## # A tibble: 10,010 x 2
##    name  timestamp
##    <chr> <dttm>
##  1 Amy   1975-06-27 00:00:00
##  2 Amy   1975-06-27 06:00:00
##  3 Amy   1975-06-27 12:00:00
##  4 Amy   1975-06-27 18:00:00
##  5 Amy   1975-06-28 00:00:00
##  6 Amy   1975-06-28 06:00:00
##  7 Amy   1975-06-28 12:00:00
##  8 Amy   1975-06-28 18:00:00
##  9 Amy   1975-06-29 00:00:00
## 10 Amy   1975-06-29 06:00:00
## # ... with 10,000 more rows
```

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.

```
days_of_week = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
storms %>%
  mutate(timestamp = ymd_h(paste(year, month, day, hour))) %>%
  mutate(day_of_week = days_of_week [wday(timestamp)]) %>%
  mutate(week_of_year = week(timestamp)) %>%
  select(name, timestamp, day_of_week, week_of_year)
```

```
## # A tibble: 10,010 x 4
##    name  timestamp           day_of_week week_of_year
##    <chr> <dttm>              <chr>              <dbl>
##  1 Amy   1975-06-27 00:00:00 Friday                26
##  2 Amy   1975-06-27 06:00:00 Friday                26
##  3 Amy   1975-06-27 12:00:00 Friday                26
##  4 Amy   1975-06-27 18:00:00 Friday                26
##  5 Amy   1975-06-28 00:00:00 Saturday              26
##  6 Amy   1975-06-28 06:00:00 Saturday              26
##  7 Amy   1975-06-28 12:00:00 Saturday              26
##  8 Amy   1975-06-28 18:00:00 Saturday              26
##  9 Amy   1975-06-29 00:00:00 Sunday                26
## 10 Amy   1975-06-29 06:00:00 Sunday                26
## # ... with 10,000 more rows
```

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

```
storms %>%
  mutate(timestamp = ymd_h(paste(year, month, day, hour))) %>%
  mutate(day_of_week = days_of_week [wday(timestamp)])
```

```
## # A tibble: 10,010 x 15
##    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 4 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, timestamp <dttm>, day_of_week <chr>
```

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

```
storms %>%
   mutate(decile_windspeed = cut(wind, breaks = 10,labels = FALSE))
```

```
## # 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 <int>
```

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

```
serious_storms = storms %>%
  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 %>%
  mutate(lat_long = paste(lat, "/", long))
```

```
## # A tibble: 779 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 Caroline  1975     8    31     0  24   -97   hurrica~ 3          100      973
##  2 Caroline  1975     8    31     6  24.1 -97.5 hurrica~ 3          100      963
```

```
##  3 Belle     1976     8     8    18  29.5 -75.3 hurrica~ 3          100      958
##  4 Belle     1976     8     9     0  30.9 -75.3 hurrica~ 3          105      957
##  5 Belle     1976     8     9     6  32.5 -75.2 hurrica~ 3          105      959
##  6 Anita     1977     9     1    18  25.2 -95.5 hurrica~ 3          110      945
##  7 Anita     1977     9     2     0  24.6 -96.2 hurrica~ 5          140      931
##  8 Anita     1977     9     2     6  24.2 -97.1 hurrica~ 5          150      926
##  9 Anita     1977     9     2    12  23.7 -98   hurrica~ 4          120      940
## 10 David     1979     8    28     0  12.2 -52.9 hurrica~ 4          115      947
## # ... with 769 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, lat_long <chr>
```

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) %>%
  mutate(avg_wind = mean(wind), avg_pressure = mean(pressure)) %>%
  mutate(avg_diameter = mean( c(ts_diameter, hu_diameter), ra.rm = TRUE )) %>%
  slice(1) %>%
  select(category, avg_wind, avg_pressure, avg_diameter)
```

```
## # A tibble: 7 x 4
## # Groups:   category [7]
##   category avg_wind avg_pressure avg_diameter
##   <ord>       <dbl>        <dbl>        <dbl>
## 1 -1           27.3        1008.           NA
## 2 0            45.8         999.           NA
## 3 1            70.9         982.           NA
## 4 2            89.4         967.           NA
## 5 3           105.          954.           NA
## 6 4           122.          940.           NA
## 7 5           145.          916.           NA
```

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 %>%
  group_by(name) %>%
  mutate(max_category = max(category), max_wind = max(wind), max_pressure = max(pressure)) %>%
  slice(1) %>%
  select(name, max_category, max_wind, max_pressure)
```

```
## # A tibble: 198 x 4
## # Groups:   name [198]
##    name     max_category max_wind max_pressure
##    <chr>    <ord>           <int>        <int>
##  1 AL011993 -1                 30         1003
##  2 AL012000 -1                 25         1010
##  3 AL021992 -1                 30         1009
##  4 AL021994 -1                 30         1017
##  5 AL021999 -1                 30         1006
##  6 AL022000 -1                 30         1010
##  7 AL022001 -1                 25         1012
##  8 AL022003 -1                 30         1010
##  9 AL022006 0                  45         1008
## 10 AL031987 0                  40         1015
```

```
## # ... 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)

pacman::p_load(geosphere)

dist_function = function(lat1, long1, lat2 =  25.7617, long2 = -80.1918){
    dist = sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(long1 - long2);
    dist = acos(dist);
    dist = (6371 * pi * dist) / 180;
}

storms %>%
  mutate(distance_to_miami = dist_function(lat, long))
```

```
## # 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.

```
storms %>%
  group_by(name) %>%
  mutate(distance_last = dist_function(lat, long, lag(lat), lag(long))) %>%
  select(name,distance_last)
```

```
## Warning in acos(dist): NaNs produced
```

```
## # A tibble: 10,010 x 2
## # Groups:   name [198]
##    name  distance_last
##    <chr>         <dbl>
##  1 Amy            NA
##  2 Amy           111.
##  3 Amy           111.
##  4 Amy           111.
##  5 Amy           113.
##  6 Amy           100.
##  7 Amy            94.3
##  8 Amy            96.8
```

```
##  9 Amy           131.
## 10 Amy           112.
## # ... with 10,000 more rows
```

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 = dist_function(last(lat), last(long), first(lat), first(long))) %>%
  mutate( displacement = paste(last(lat)- first(lat), ",", last(long)- first(long))) %>%
  slice(1) %>%
  select(name, distance, displacement)
```

```
## # A tibble: 198 x 3
## # Groups:   name [198]
##    name     distance displacement
##    <chr>       <dbl> <chr>
##  1 AL011993     36.1 6.3 , 12.2
##  2 AL012000     33.4 -0.199999999999999 , -0.5
##  3 AL021992     70.5 4 , 2.59999999999999
##  4 AL021994    190.  3 , -2.09999999999999
##  5 AL021999     26.1 0.199999999999999 , -2.3
##  6 AL022000     53.9 0.199999999999999 , -18.4
##  7 AL022001    245.  2.2 , -6.4
##  8 AL022003    131.  0.199999999999999 , -5.1
##  9 AL022006    187.  4.6 , 6.3
## 10 AL031987    122.  5.5 , 11.3
## # ... with 188 more rows
```

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

```
storms %>%
  group_by(name) %>%
  mutate(distance = dist_function(last(lat), last(long), first(lat), first(long))) %>%
  mutate(average_speed = distance / row_number() ) %>%
  slice(1) %>%
  select(name, average_speed)
```

```
## # A tibble: 198 x 2
## # Groups:   name [198]
##    name     average_speed
##    <chr>            <dbl>
##  1 AL011993          36.1
##  2 AL012000          33.4
##  3 AL021992          70.5
##  4 AL021994         190.
##  5 AL021999          26.1
##  6 AL022000          53.9
##  7 AL022001         245.
##  8 AL022003         131.
##  9 AL022006         187.
## 10 AL031987         122.
## # ... with 188 more rows
```

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(distance = dist_function(last(lat), last(long), first(lat), first(long))) %>%
  mutate(average_speed = distance / row_number() ) %>%
  slice(1) %>%
  select(name, average_speed)
```

```
## # A tibble: 198 x 2
## # Groups:   name [198]
##    name     average_speed
##    <chr>            <dbl>
##  1 AL011993          36.1
##  2 AL012000          33.4
##  3 AL021992          70.5
##  4 AL021994         190.
##  5 AL021999          26.1
##  6 AL022000          53.9
##  7 AL022001         245.
##  8 AL022003         131.
##  9 AL022006         187.
## 10 AL031987         122.
## # ... with 188 more rows
```

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

```
storms %>%
  group_by(name) %>%
  mutate(distance = dist_function(last(lat), last(long), first(lat), first(long))) %>%
  mutate(average_speed = distance / row_number() ) %>%
  group_by(category) %>%
  mutate(avg_ground_speed = mean(average_speed)) %>%
  slice(1) %>%
  select(category, avg_ground_speed)
```

```
## # A tibble: 7 x 2
## # Groups:   category [7]
##   category avg_ground_speed
##   <ord>               <dbl>
## 1 -1                  26.5
## 2 0                   10.9
## 3 1                    7.22
## 4 2                    5.02
## 5 3                    4.48
## 6 4                    4.09
## 7 5                    5.12
```

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.

```
storm_train = storms %>%
  group_by(name) %>%
  mutate(max_wind_by_storm = max(wind, na.rm = TRUE)) %>%
  slice_head(n = 3)
```

Fit your model. Validate it.

```
wind_model = lm(max_wind_by_storm ~ ., storm_train)
```

Assess your level of success at this endeavor.

NOT GOOD

## 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$income = ifelse(adult$income== ">50K",1,0)

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

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

adult$native_country = as.character(adult$native_country)
tab = sort(table(adult$native_country))
adult$native_country = ifelse(adult$native_country%in% names(tab[tab<50]),"Other",adult$native_country)
adult$native_country = as.factor(adult$native_country)

adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tab = (table(adult$worktype))
adult$worktype = ifelse(adult$worktype%in% names(tab[tab<50]),"Other",adult$worktype)
adult$worktype = as.factor(adult$worktype)

adult$relmarried = paste(adult$relationship, adult$marital_status, sep = ":")
adult$relmarried = ifelse(adult$relmarried%in% names(tab[tab<50]),"Other",adult$relmarried)
adult$relmarried = as.factor(adult$relmarried)

adult$log_capital_gain = log( 1 + adult$capital_gain )
adult$log_capital_loss = log( 1 + adult$capital_loss )
```

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.

```r
Nsplitsize = 1000
```

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

```r
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.

```r
#logistic_mod = glm(ytrain ~ ., Xtrain, family = "binomial")
```

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

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.

```r
#dim(Xmm_train)
#dim(Xmm_select)
#dim(Xmm_test)
```

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.

```r
#
```

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

```r
#
```