Lab 7

Brendan Gubbins

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, ncol = 10)
head(X)
```

```
##            [,1]        [,2]        [,3]        [,4]       [,5]        [,6]
## [1,] -0.8152725 -0.63259671 -0.50198283 -0.22145917  0.3509743  0.25711620
## [2,]  0.8049438  0.60339866  0.06659846 -0.01867782 -0.1086336 -0.08726104
## [3,] -0.1006844  0.06929083 -2.22394007  0.60918906  0.7964786 -0.20418887
## [4,]  1.2583773 -0.03840441  0.68529101 -0.40937347 -1.3736660  0.14812750
## [5,] -1.5808925 -1.15736510 -0.56322975  1.54701288  0.2891235  0.64309735
## [6,]  0.9072673  2.08983220  0.83083998  1.07430024  0.7216358 -0.15002761
##            [,7]       [,8]       [,9]       [,10]
## [1,] -0.1037063  0.7821686 -0.9160984  2.12187654
## [2,]  0.5841189 -0.5022414  1.1692070 -0.24450607
## [3,] -0.9126133  0.9317149  0.1972460 -2.32017248
## [4,]  0.9240186 -1.5377753 -0.5943596 -0.75963943
## [5,]  0.2552241  0.2743324 -1.3902003  0.61341617
## [6,]  0.2118700 -0.4198976  1.3553721 -0.08173049
```

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.

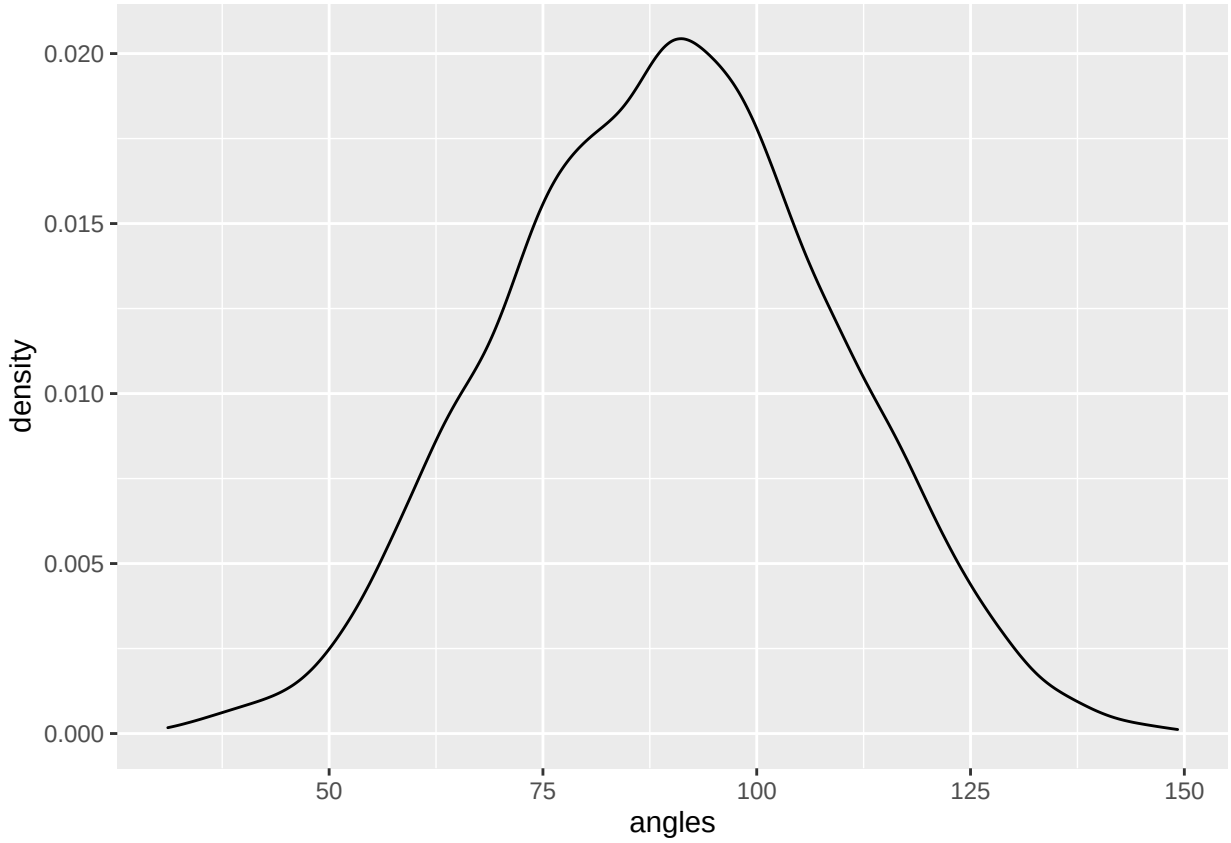
```
angle = function(u, v) {
  (acos(sum(u*v)/sqrt(sum(u^2)*sum(v^2)))) * (180/pi)
}

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_times_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_times_v += X(i_1, j) * X(i_2, j);
        }
        A(i_1, i_2) = acos(sum_u_times_v / sqrt(sum_sqd_u * sum_sqd_v)) * (180/M_PI);
      }
    }
    return A;
  }
  "
)
#all_angles_cpp(X)
```

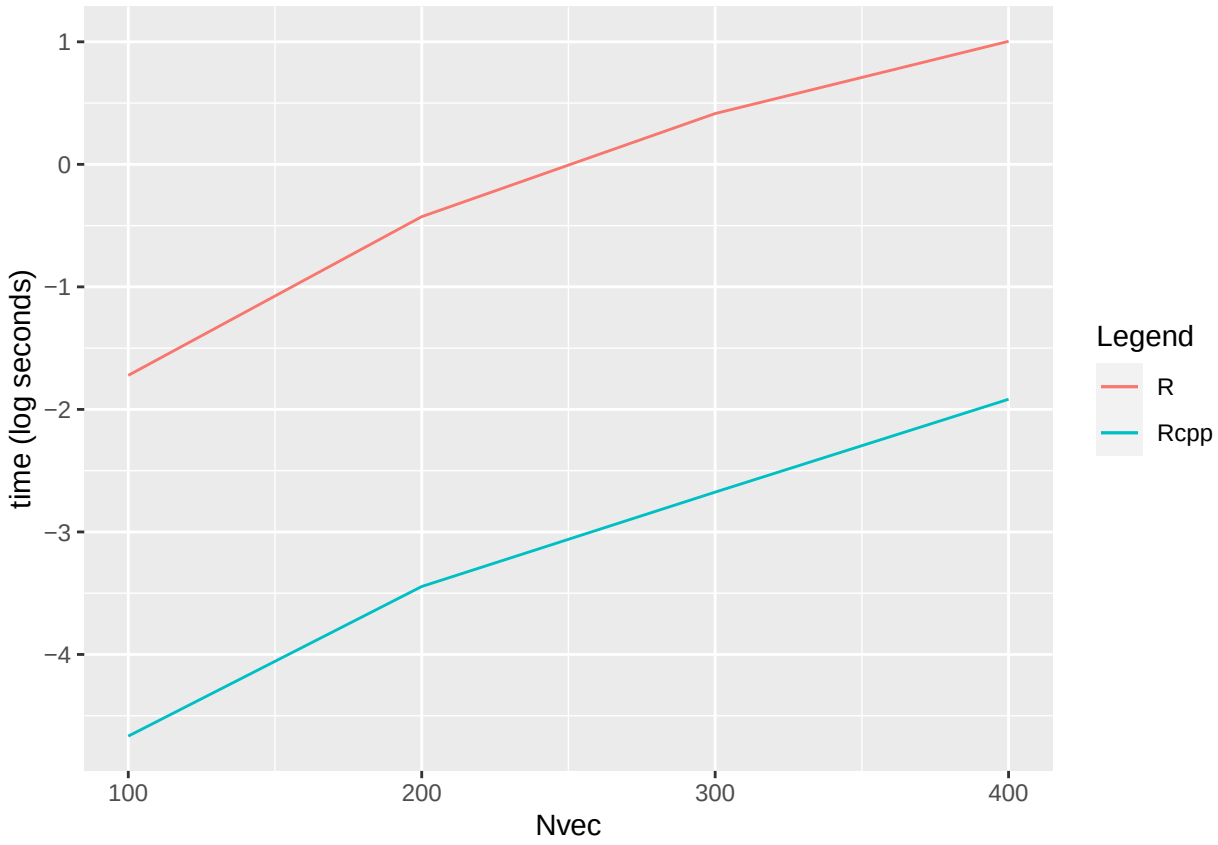
Test the time difference between these functions for `n = 1000` and `Nvec = 100, 500, 1000, 5000` using the package `microbenchmark`. Store the results in a matrix with rows representing `Nvec` and two columns for base R and Rcpp.

```
pacman::p_load(microbenchmark)
n = 1000

# Nvec = 500+ runtime too long...
Nvec = c(100, 200, 300, 400)
results_Nvec = matrix(data = NA, nrow = length(Nvec), ncol = 2)

for (i in 1:length(Nvec)) {
  X = matrix(data = rnorm(Nvec[i] * n), nrow = Nvec[i])
  bench_Nvec = summary(microbenchmark(all_angles(X), all_angles_cpp(X), times = 10, unit = "s"))
  results_Nvec[i,1] = bench_Nvec[1,4] # grabs the mean
  results_Nvec[i,2] = bench_Nvec[2,4]
}

results_Nvec
```

```
##           [,1]        [,2]
## [1,] 0.1786682 0.009404333
## [2,] 0.6525344 0.031895922
## [3,] 1.5137918 0.068895111
## [4,] 2.7296734 0.147025723
```

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.

```
ggplot(data.frame(log(results_Nvec))) +
  aes(x = Nvec) +
  geom_line(aes(y = log(results_Nvec[,1]), color = "R")) +
  geom_line(aes(y = log(results_Nvec[,2]), color = "Rcpp")) +
  labs(y = "time (log seconds)", color = "Legend")
```

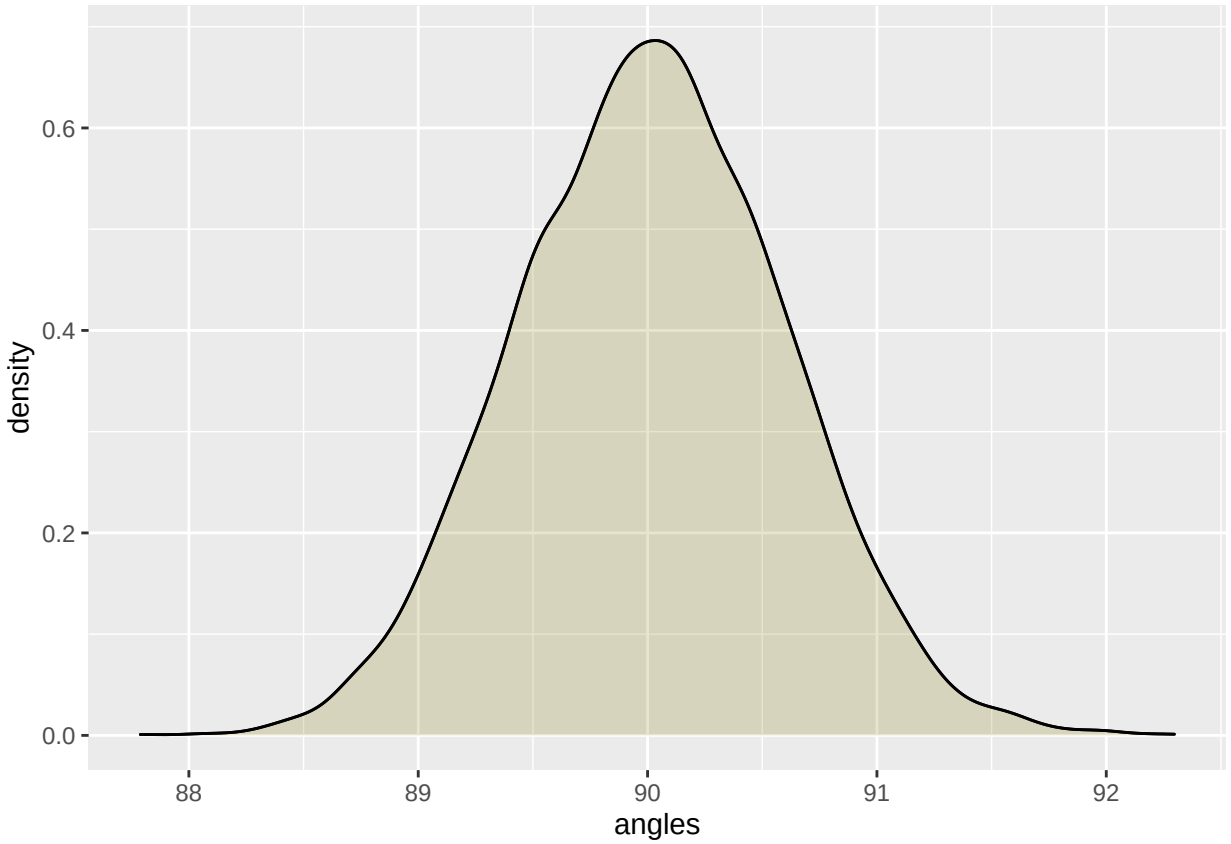


Let `Nvec = 10000` and vary `n` to be 10, 100, 1000. Plot the density of angles for all three values of `n` on one plot using color to signify `n`. Make sure you have a color legend. This is not easy.

```
# n = 100+ runtime too long...
n = c(10, 50, 100)
Nvec = 10000
results_n = list()

for (i in 1:length(n)) {
  X = matrix(data = rnorm(n[i] * Nvec), nrow = n[i])

  results_n = all_angles(X)
  results_n = all_angles_cpp(X)
}

ggplot() +
  geom_density(data = data.frame(angles = c(all_angles(X))), aes(x = angles), fill = "green", alpha = 0
  geom_density(data = data.frame(angles_cpp = c(all_angles_cpp(X))), aes(x = angles_cpp), fill = "red",
```

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

```
## Warning: Removed 5050 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.

```
nth_fibonnaci = function(num, start) {
  if (num == 1 | num == 2) return(start)

  return(nth_fibonnaci(num - 1, start) + nth_fibonnaci(num - 2, start))
}
```

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

```
cppFunction(
  "
  double nth_fibonnaci_cpp(int num, double start) {
    if (num == 1 || num == 2) return start;

    return nth_fibonnaci_cpp(num - 1, start)
    + nth_fibonnaci_cpp(num - 2, start);
  }
  "
)
```

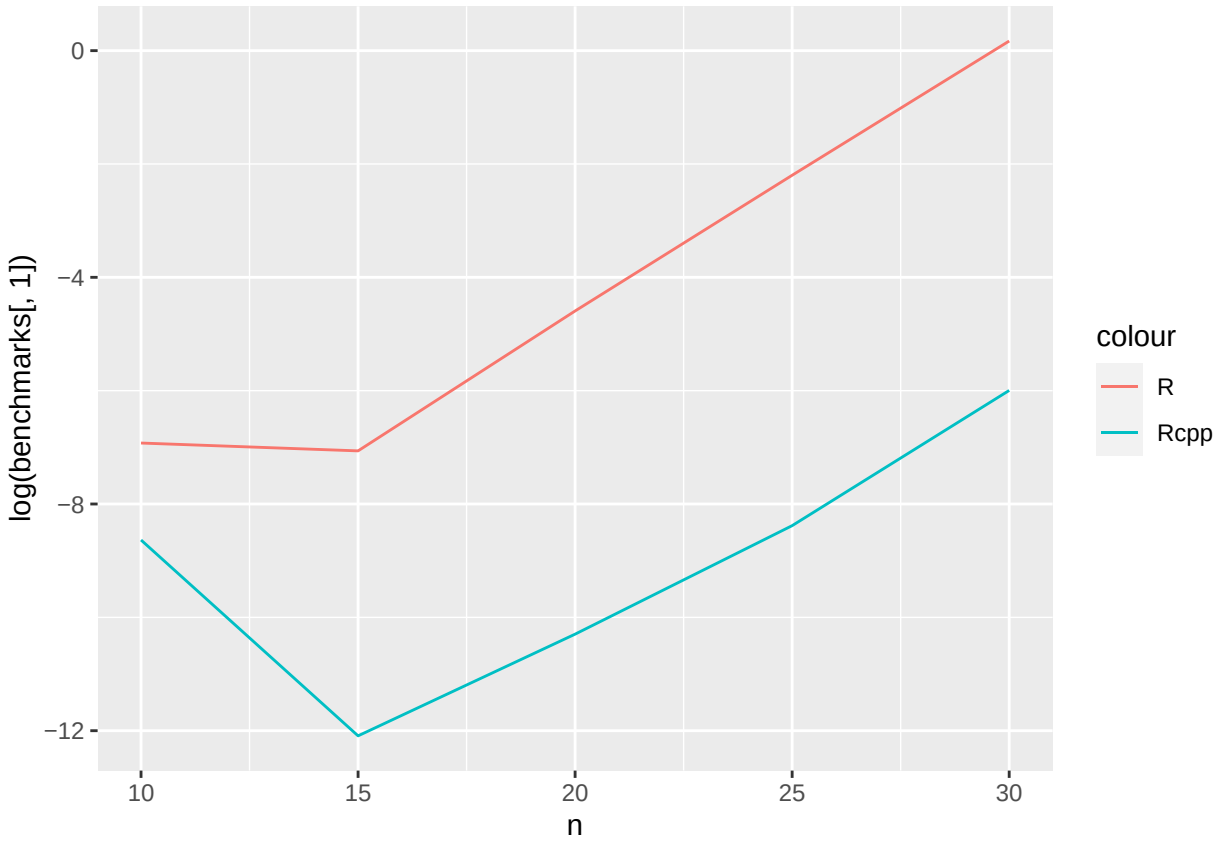
Time the difference in these functions for n = 100, 200, . . . , 1500 while starting the sequence at the smallest possible floating point value in R. Store the results in a matrix.

```
# fibonnaci is O(2^n), can't run 1500...
n = seq(10, 30, by = 5)
benchmarks = matrix(data = NA, nrow = length(n), ncol = 2)
min = .Machine$double.xmin

for (i in 1 : length(n)) {
    bench = summary(microbenchmark(nth_fibonnaci(n[i], min), nth_fibonnaci_cpp(n[i], min), times = 10, u
    benchmarks[i,1] = bench[1,4] # grabbing the mean
    benchmarks[i,2] = bench[2,4]
}

benchmarks
```

```
##              [,1]         [,2]
## [1,] 0.0009838406 0.0001777247
## [2,] 0.0008569106 0.0000056094
## [3,] 0.0101174298 0.0000337524
## [4,] 0.1109311127 0.0002289951
## [5,] 1.1853948639 0.0024879899
```

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(data.frame(log(benchmarks))) +
  aes(x = n) +
  geom_line(aes(y = log(benchmarks[,1]), color = "R")) +
  geom_line(aes(y = log(benchmarks[,2]), color = "Rcpp"))
```



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(dplyr, magrittr, data.table)
```

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

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

```
## tibble [10,010 x 13] (S3: tbl_df/tbl/data.frame)
##  $ name       : chr [1:10010] "Amy" "Amy" "Amy" "Amy" ...
##  $ year       : num [1:10010] 1975 1975 1975 1975 1975 ...
##  $ month      : num [1:10010] 6 6 6 6 6 6 6 6 6 6 ...
##  $ day        : int [1:10010] 27 27 27 27 28 28 28 28 29 29 ...
##  $ hour       : num [1:10010] 0 6 12 18 0 6 12 18 0 6 ...
##  $ lat        : num [1:10010] 27.5 28.5 29.5 30.5 31.5 32.4 33.3 34 34.4 34 ...
##  $ long       : num [1:10010] -79 -79 -79 -79 -78.8 -78.7 -78 -77 -75.8 -74.8 ...
##  $ status     : chr [1:10010] "tropical depression" "tropical depression" "tropical depression" "tro
##  $ category   : Ord.factor w/ 7 levels "-1"<"0"<"1"<"2"<..: 1 1 1 1 1 1 1 1 2 2 ...
```

```
##  $ wind       : int [1:10010] 25 25 25 25 25 25 25 30 35 40 ...
##  $ pressure   : int [1:10010] 1013 1013 1013 1013 1012 1012 1011 1006 1004 1002 ...
##  $ ts_diameter: num [1:10010] NA NA NA NA NA NA NA NA NA NA ...
##  $ hu_diameter: num [1:10010] NA NA NA NA NA NA NA NA NA NA ...
```

```
summary(storms)
```

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

```
head(storms)
```

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

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

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

```
## # A tibble: 10,010 x 13
##    name  status      category  year month   day  hour   lat  long  wind pressure
```

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

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

```
storms %>%
  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: 416 x 13
##    name   year month   day  hour   lat  long status    category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>     <ord>    <int>    <int>
##  1 Anita  1977     9     2     0  24.6 -96.2 hurricane 5          140      931
##  2 Anita  1977     9     2     6  24.2 -97.1 hurricane 5          150      926
##  3 Anita  1977     9     2    12  23.7 -98   hurricane 4          120      940
##  4 David  1979     8    28     0  12.2 -52.9 hurricane 4          115      947
##  5 David  1979     8    28     6  12.5 -54.4 hurricane 4          125      941
##  6 David  1979     8    28    12  12.8 -55.7 hurricane 4          130      938
##  7 David  1979     8    28    18  13.2 -56.9 hurricane 4          125      941
##  8 David  1979     8    29     0  13.7 -58   hurricane 4          120      944
```

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

Create a new feature `wind_speed_per_unit_pressure`.

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

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

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

```
storms %>%
  rowwise() %>%
    arrange(desc(year)) %>%
      mutate(average_diameter = mean(c(ts_diameter, hu_diameter), 0, na.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_speed = max(wind, na.rm = TRUE))
```

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

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

Find the strongest storm by wind speed per year.

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

```
## # A tibble: 41 x 2
## # Groups:   year [41]
##    name      year
##    <chr>    <dbl>
##  1 Caroline  1975
##  2 Belle     1976
##  3 Anita     1977
##  4 Cora      1978
##  5 David     1979
##  6 Ivan      1980
##  7 Harvey    1981
##  8 Debby     1982
##  9 Alicia    1983
## 10 Diana     1984
## # ... 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_wind_speed = max(wind, na.rm = TRUE), max_category = max(category, na.rm = TRUE),
              max_pressure = max(pressure, na.rm = TRUE), max_diameter_ts = max(ts_diameter, 0, na.rm =
              max_diameter_hu = max(hu_diameter, 0, na.rm = TRUE))
```

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

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

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

```
## # A tibble: 41 x 2
##     year num_storms
##  * <dbl>      <int>
##  1  1975          3
##  2  1976          2
```

```
##  3  1977          3
##  4  1978          4
##  5  1979          7
##  6  1980          8
##  7  1981          5
##  8  1982          5
##  9  1983          4
## 10  1984         10
## # ... with 31 more rows
```

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

```
storms %>%
  group_by(year, category) %>%
    distinct(name) %>%
      count(category)
```

```
## # A tibble: 233 x 3
## # Groups:   year, category [233]
##     year category     n
##    <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_speed = max(wind, na.rm = TRUE))
```

```
## `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_speed
##    <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(average_lat = mean(lat), average_long = mean(long), average_location = mean(c(lat, long)))
```

```
## # A tibble: 198 x 4
##    name     average_lat average_long average_location
##  * <chr>          <dbl>        <dbl>            <dbl>
##  1 AL011993       24.7         -78.0           -26.7
##  2 AL012000       20.8         -93.1           -36.1
##  3 AL021992       26.7         -84.5           -28.9
##  4 AL021994       33.6         -79.7           -23.1
##  5 AL021999       20.4         -96.4           -38.0
##  6 AL022000        9.9         -28.5            -9.32
##  7 AL022001       11.9         -45.3           -16.7
##  8 AL022003        9.62        -43.4           -16.9
##  9 AL022006       41.3         -63.5           -11.1
## 10 AL031987       30.8         -88.7           -29.0
## # ... with 188 more rows
```

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

```
storms %>%
  group_by(name) %>%
    arrange(year, month, day, hour) %>%
      count(name) %>%
        summarize(duration = n * 6)
```

```
## # A tibble: 198 x 2
##    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) %>%
    arrange(year, month, day, hour) %>%
      summarize(storm_number = row_number(category))
```

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

```
## # A tibble: 10,010 x 2
## # Groups:   category [7]
##    category storm_number
##    <ord>           <int>
##  1 -1                  1
##  2 -1                  2
##  3 -1                  3
##  4 -1                  4
##  5 -1                  5
##  6 -1                  6
##  7 -1                  7
##  8 -1                  8
##  9 -1                  9
## 10 -1                 10
## # ... 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, sep = '.')))
```

```
## # 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>, timestamp <dttm>
```

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(ymd_h(paste(year, month, day, hour, sep = "."))),
         week_of_year = week(ymd_h(paste(year, month, day, hour, sep = "."))))
```

```
## # 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>, day_of_week <chr>, week_of_year <dbl>
```

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

```
storms %>%
  group_by(name) %>%
    mutate(timestamp = ymd_h(paste(year, month, day, hour, sep = "."))) %>%
      arrange(timestamp) %>%
        slice(1)
```

```
## # A tibble: 198 x 14
## # Groups:   name [198]
##    name    year month   day  hour   lat  long status     category  wind pressure
##    <chr>  <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>      <ord>    <int>    <int>
##  1 AL011~  1993     5    31    12  21.5 -84   tropical ~ -1          25     1003
##  2 AL012~  2000     6     7    18  21   -93   tropical ~ -1          25     1008
##  3 AL021~  1992     6    25    12  24.5 -85.5 tropical ~ -1          25     1009
##  4 AL021~  1994     7    20     6  32.2 -78.9 tropical ~ -1          25     1017
##  5 AL021~  1999     7     2    18  20.2 -95   tropical ~ -1          30     1006
##  6 AL022~  2000     6    23     0   9.5 -19.8 tropical ~ -1          25     1010
##  7 AL022~  2001     7    11    18  10.9 -42.1 tropical ~ -1          25     1011
##  8 AL022~  2003     6    11     0   9.5 -40.8 tropical ~ -1          30     1009
##  9 AL022~  2006     7    17     6  39.1 -66.4 tropical ~ -1          30     1008
## 10 AL031~  1987     8     9    12  26.3 -93.6 tropical ~ -1          30     1010
## # ... with 188 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, timestamp <dttm>
```

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

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

```
## # A tibble: 10,010 x 14
```

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

```
storms$decile_windspeed = ntile("wind", 10)
```

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$lat_long = paste(serious_storms$lat, serious_storms$long, sep = " / ")

serious_storms$lat = NULL
serious_storms$long = NULL

serious_storms
```

```
## # A tibble: 779 x 13
##    name      year month   day  hour status   category  wind pressure ts_diameter
##    <chr>    <dbl> <dbl> <int> <dbl> <chr>    <ord>    <int>    <int>       <dbl>
##  1 Caroline  1975     8    31     0 hurrica~ 3          100      973          NA
##  2 Caroline  1975     8    31     6 hurrica~ 3          100      963          NA
##  3 Belle     1976     8     8    18 hurrica~ 3          100      958          NA
##  4 Belle     1976     8     9     0 hurrica~ 3          105      957          NA
##  5 Belle     1976     8     9     6 hurrica~ 3          105      959          NA
##  6 Anita     1977     9     1    18 hurrica~ 3          110      945          NA
##  7 Anita     1977     9     2     0 hurrica~ 5          140      931          NA
##  8 Anita     1977     9     2     6 hurrica~ 5          150      926          NA
##  9 Anita     1977     9     2    12 hurrica~ 4          120      940          NA
## 10 David     1979     8    28     0 hurrica~ 4          115      947          NA
## # ... with 769 more rows, and 3 more variables: hu_diameter <dbl>,
## #   decile_windspeed <int>, 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) %>%
    summarize(avg_wind_speed = mean(wind), avg_pressure = mean(pressure), avg_diameters = mean(c(ts_dia
```

```
## # A tibble: 7 x 4
##   category avg_wind_speed avg_pressure avg_diameters
## * <ord>             <dbl>        <dbl>         <dbl>
## 1 -1                 27.3        1008.           0
## 2 0                  45.8         999.          79.8
## 3 1                  70.9         982.         168.
## 4 2                  89.4         967.         180.
## 5 3                 105.          954.         199.
## 6 4                 122.          940.         209.
## 7 5                 145.          916.         219.
```

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) %>%
    summarize(max_category = max(category),
      max_wind_speed = max(wind),
      max_pressure = max(pressure),
      max_ts_diameter = max(ts_diameter, 0, na.rm = TRUE),
      max_hu_diameter = max(hu_diameter, 0, na.rm = TRUE),
      readings = (storms %>% group_by(name) %>% count(name))$n)
```

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

```
## # A tibble: 39,204 x 7
## # Groups:   name [198]
##    name     max_category max_wind_speed max_pressure max_ts_diameter
##    <chr>    <ord>                <int>        <int>           <dbl>
##  1 AL011993 -1                      30         1003               0
##  2 AL011993 -1                      30         1003               0
##  3 AL011993 -1                      30         1003               0
##  4 AL011993 -1                      30         1003               0
##  5 AL011993 -1                      30         1003               0
##  6 AL011993 -1                      30         1003               0
##  7 AL011993 -1                      30         1003               0
##  8 AL011993 -1                      30         1003               0
##  9 AL011993 -1                      30         1003               0
## 10 AL011993 -1                      30         1003               0
## # ... with 39,194 more rows, and 2 more variables: max_hu_diameter <dbl>,
## #   readings <int>
```

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)

#' Coordinate Distance
#'
#' Given two latitude and longitude coordinates, compute the distance
#' apart in kilometers using the Haversine formula.
#'
#' @param coords1     vector of lat long coordinates
#' @param coords2     vector of lat long coordinates
#'
#' @return            returns the distance in km
coord_distance = function(lat1, long1, lat2, long2) {
  R = 6371e3
  phi1 = lat1 * pi/180
  phi2 = lat2 * pi/180
  delta_phi = (lat2 - lat1) * pi/180
  delta_lambda = (long2 - long1) * pi/180

  a = sin(delta_phi / 2)^2 + cos(phi1) * cos(phi2) * sin(delta_lambda / 2)^2
  c = 2 * atan2(sqrt(a), sqrt(1-a))

  distance = R * c / 1000 # in km
  distance
}

coord_distance(47.23, 92.22, MIAMI_LAT_LONG_COORDS[1], MIAMI_LAT_LONG_COORDS[2])
```

```
## [1] 11863.13
```

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

```
storms = storms %>%
  group_by(name) %>%
    mutate(distance_from_previous = if_else(name != lag(name), 0, coord_distance(lag(lat), lag(long) , l
    mutate(distance_from_previous = if_else(is.na(distance_from_previous), 0, distance_from_previous))

storms
```

```
## # A tibble: 10,010 x 15
## # Groups:   name [198]
##    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>, decile_windspeed <int>, distance_from_previous <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(distance_from_previous),
              displacement_lat = last(lat) - first(lat),
              displacement_long = last(long) - first(long))
```

```
## # A tibble: 198 x 4
##    name     distance displacement_lat displacement_long
##  * <chr>       <dbl>            <dbl>             <dbl>
##  1 AL011993    1431.            6.3               12.2
##  2 AL012000     105.           -0.200             -0.5
##  3 AL021992     539.            4                  2.60
##  4 AL021994     395.            3                 -2.10
##  5 AL021999     250.            0.200             -2.30
##  6 AL022000    2028.            0.200            -18.4
##  7 AL022001     739.            2.20              -6.40
##  8 AL022003     560.            0.200             -5.1
##  9 AL022006     734.            4.6                6.3
## 10 AL031987    1934.            5.5               11.3
## # ... with 188 more rows
```

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

```
storms = storms %>%
  group_by(name) %>%
    mutate(avg_speed_km_per_hour = distance_from_previous / 6)

storms
```

```
## # A tibble: 10,010 x 16
## # Groups:   name [198]
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s~ 0           40     1002
## # ... with 10,000 more rows, and 5 more variables: ts_diameter <dbl>,
```

```
## #   hu_diameter <dbl>, decile_windspeed <int>, distance_from_previous <dbl>,
## #   avg_speed_km_per_hour <dbl>
```

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

```
storms = storms %>%
  group_by(name) %>%
    mutate(ground_speed = mean(avg_speed_km_per_hour))

storms
```

```
## # A tibble: 10,010 x 17
## # Groups:   name [198]
##    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 6 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <int>, distance_from_previous <dbl>,
## #   avg_speed_km_per_hour <dbl>, ground_speed <dbl>
```

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

```
storms %>%
  group_by(name) %>%
    summarize(max_category = max(category), ground_speed)
```

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

```
## # A tibble: 10,010 x 3
## # Groups:   name [198]
##    name     max_category ground_speed
##    <chr>    <ord>               <dbl>
##  1 AL011993 -1                   29.8
##  2 AL011993 -1                   29.8
##  3 AL011993 -1                   29.8
##  4 AL011993 -1                   29.8
##  5 AL011993 -1                   29.8
##  6 AL011993 -1                   29.8
##  7 AL011993 -1                   29.8
##  8 AL011993 -1                   29.8
##  9 AL012000 -1                    4.37
## 10 AL012000 -1                    4.37
## # ... with 10,000 more rows
```

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

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

```
X = storms %>%
  group_by(name) %>%
  mutate(max_wind_speed = max(wind, na.rm = TRUE)) %>%
  filter(n() >= 3) %>%
  slice(1:3)

y = X %>%
  group_by(name) %>%
  select(max_wind_speed)
```

```
## Adding missing grouping variables: `name`
```

```
X = X %>%
  select(-max_wind_speed, -lat, -long, -ts_diameter, -hu_diameter)
```

Fit your model. Validate it.

```
n = nrow(X)
K = 5
n_test = round(n * 1 / K)
n_train = n - n_test

test_indices = sample(1 : n, n_test)
train_indices = setdiff(1 : n, test_indices)

X_train = X[train_indices,]
y_train = y[train_indices,]$max_wind_speed

X_test = X[test_indices,]
y_test = y[test_indices,]$max_wind_speed

ols_mod = lm(y_train ~ ., X_train)
y_oos = predict(ols_mod, X_test)
residuals = y_test - y_oos
ooss_e = sd(residuals)
```

Assess your level of success at this endeavor.

This is probably not very successful because predictions can be on features observations that have not been trained. Additionally, there isn't really any interacting between the first three observations of a storm.

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 == "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)

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<100]), "other", adult$native_coun

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

adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tabulate = sort(table(adult$worktype))

adult = subset(adult, select = -c(occupation, workclass))

adult$worktype = ifelse(adult$worktype %in% names(tabulate[tabulate<100]), "other", adult$worktype)

adult$worktype = as.factor(adult$worktype)

adult$relationship_status = paste(adult$relationship, adult$marital_status, sep = ":")

adult$relationship_status = as.factor(adult$relationship_status)

tabul = sort(table(adult$relationship_status))

adult$relationship_status = as.factor(adult$relationship_status)

adult$relationship_status = ifelse(adult$relationship_status %in% names(tabul[tabul<100]), "other", adu

adult$relationship_status = as.factor(adult$relationship_status)

adult = subset(adult, select = -c(relationship, marital_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", maxit = 100)
# doesn't converge without maxit
```

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

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

brier_score(logistic_mod, Xtrain, ytrain)
```

```
## [1] -2.085639e-26
```

```
brier_score(logistic_mod, Xtest, ytest)
```

```
## [1] -2.085639e-26
```

```
brier_score(logistic_mod, Xselect, yselect)
```

```
## [1] -2.085639e-26
```

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(~ . * . + 0, Xtrain))
Xmm_select = data.frame(model.matrix(~ . * . + 0, Xselect))
Xmm_test = data.frame(model.matrix(~ . * . + 0, Xtest))

dim(Xmm_train)
```

```
## [1] 1000 2953
```

```
dim(Xmm_select)
```

```
## [1] 1000 2953
```

```
dim(Xmm_test)
```

```
## [1] 1000 2953
```

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 {
  #TO-DO
  #wrap glm and predict calls with use suppressWarnings() so the console is clean during run

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

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

```
#TO-DO
```