Lab 7

Hubert Majewski

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.

```
#Turn off warnings
options(warn = -1)

pacman::p_load(Rcpp, microbenchmark)
```

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 = 100)
head(X)
```

```
##            [,1]        [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] -1.4984689  0.40497106 -1.1467178 -0.6838356  0.6264661  0.6461078
## [2,] -1.1689006  1.27104230  0.2051682  0.6111950  0.9808982  0.5790121
## [3,]  0.6653255 -0.53591734 -1.7308344 -0.2581449  0.2433777  1.2044406
## [4,]  1.0858043  0.02587649  1.0257913 -0.4597439  0.6269530  1.0102600
## [5,]  0.7735014  1.88744569  0.3846406 -1.0758949 -0.3274697 -0.4340552
## [6,] -0.6516318 -1.39176292 -0.9566813  1.3142714  1.9038905  0.2134195
##            [,7]       [,8]        [,9]      [,10]
## [1,] -0.9030426 -1.1301065 -0.56339319  0.8433010
## [2,]  0.3320956 -0.2936879  2.00143586 -0.2322360
## [3,] -0.3779455 -0.5095187 -0.69286503  1.1987303
## [4,] -0.1006343  0.1521106  1.02606491 -0.7781927
## [5,] -1.3134929  1.3799228  1.19465006 -0.2444820
## [6,] -0.8016641 -1.3604655 -0.05860679  0.4688876
```

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) {

  return(acos(sum(u*v) / sqrt(sum(u^2)*sum(v^2))) * (180/pi))
```

```r
}

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,])
    }

  }

  return(A)
}

head(
all_angles(X)
)
```

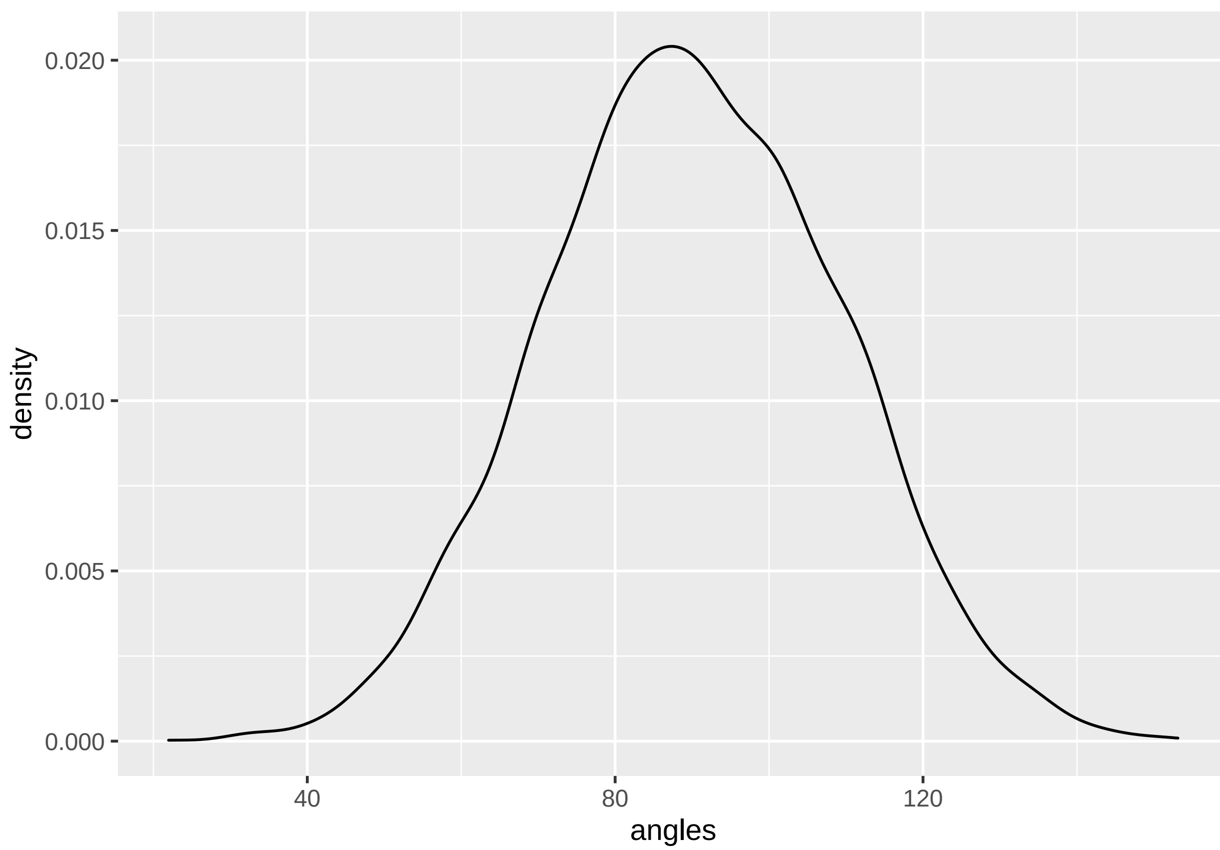
```
##      [,1]     [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,]   NA 81.16117  57.77033 114.16515 101.27661  61.21775  89.58055 137.85245
## [2,]   NA       NA 108.90859  73.09477  78.60266  81.46916  67.69329  89.42244
## [3,]   NA       NA        NA 101.19089 108.00721  64.93008 116.51686 107.18585
## [4,]   NA       NA        NA        NA  68.14430 100.66390  66.32368  66.34431
## [5,]   NA       NA        NA        NA        NA 126.60591  87.69528  97.57780
## [6,]   NA       NA        NA        NA        NA        NA 100.90942 102.97756
##           [,9]     [,10]     [,11]     [,12]     [,13]    [,14]    [,15]
## [1,]  73.40466  75.28467 100.88988 125.37764  83.54642 71.56929 83.63185
## [2,] 109.53841  95.43691  92.88879  74.45398 111.11593 62.98008 62.11412
## [3,]  71.12173 114.21100  93.81506 142.76435  49.46919 77.77102 69.99038
## [4,] 108.14151 110.27679 117.37660  68.57413  88.34409 73.00381 71.52302
## [5,]  70.58512  77.88864  69.24211  96.98414  77.19271 65.39729 59.12058
## [6,] 114.72779  90.64767 106.79555  97.85607  98.20962 88.28579 99.66352
##          [,16]     [,17]     [,18]     [,19]     [,20]     [,21]     [,22]
## [1,] 108.15029  80.47386  69.88627  95.81957  72.24109 116.39859 104.24731
## [2,] 127.67331 100.49852  90.49341 105.00398  56.42260  92.97630  76.67629
## [3,]  86.03953  80.00854  84.13791 104.09054  86.77594  94.48884 115.91251
## [4,]  86.40649 109.42637 144.44547  94.93526 102.14206  95.39209  41.63351
## [5,]  90.48768  87.07616 120.76184 109.13398 101.82663 100.99935  79.60515
## [6,]  83.70447  78.89719  53.22518  82.48579  79.23761 105.55959  89.04257
##          [,23]     [,24]     [,25]     [,26]     [,27]     [,28]     [,29]
## [1,]  85.88755 121.59733  96.36770  59.09311  88.92134 117.94577 112.31973
## [2,]  94.76953 127.56282 108.54355  82.53601 109.22306  93.88584  87.86614
## [3,] 104.79080  79.87742 106.68748  80.87283  63.62968 100.51085  79.78635
## [4,] 114.23611 105.67893  91.08844 105.49746  78.39736  74.89934  53.69837
## [5,]  84.47240 102.35731  76.79643  68.41108  87.06096  78.49255  91.27065
## [6,] 103.05157  86.97943 121.60471  81.66763 110.98841 115.90122  87.43417
##          [,30]    [,31]     [,32]     [,33]     [,34]     [,35]     [,36]
## [1,]  86.73382 94.68297  91.01225  76.98852 118.55744  58.05465  97.11273
## [2,]  68.34984 88.23935 110.74406  91.29933  64.46751 119.99194  79.38676
## [3,] 115.33571 77.74512  88.18129 104.05808 130.41612  62.41351  80.29929
## [4,] 102.97036 72.17007  91.39582  70.05170  77.16326 112.72763  78.39601
```

```
## [5,] 116.61000 84.33400 122.96943  72.45947  65.10680 113.11310  81.97488
## [6,]  92.37110 92.49430  71.41056  93.31917  96.82635  57.27462 101.80407
##          [,37]    [,38]     [,39]    [,40]     [,41]     [,42]    [,43]
## [1,]  92.17583 50.51956 113.57451 75.62228  46.12789  91.94107 61.45017
## [2,]  49.11674 50.08688  82.62431 79.48008 112.09166 106.66696 86.60908
## [3,] 118.71304 76.49055 112.14754 98.02088  45.08447  74.41584 48.92171
## [4,]  89.37382 82.16688  94.78296 99.78112 106.13172  68.12420 80.08935
## [5,]  90.34651 91.04105  75.27478 54.66155 101.69273 118.88243 78.15780
## [6,]  68.24650 69.22944 117.76135 94.01005  63.86170  63.03104 74.84821
##          [,44]     [,45]     [,46]     [,47]     [,48]     [,49]    [,50]
## [1,]  96.66162  86.35206  95.27949  91.22103 105.91160  69.95696 69.64706
## [2,]  86.98889  76.33683  84.85278  84.39393  69.46542 132.10142 87.95990
## [3,] 114.84665  83.47199  98.43183  82.33443 138.53587  60.10103 39.96819
## [4,] 102.37792 109.18869  80.63793  65.90388  56.18213 100.86258 76.09546
## [5,]  83.68637 111.76222  41.84077 110.20866  66.61775  97.24386 85.28210
## [6,]  90.40472  80.62756 103.80985  65.64481 103.34726  95.26936 77.85898
##         [,51]     [,52]     [,53]     [,54]     [,55]     [,56]    [,57]
## [1,] 69.94873  66.01345  79.19810  91.50286  87.01856 112.44672 122.0939
## [2,] 85.63129 111.24241  99.22366  71.01450 127.45684  94.26655 121.1606
## [3,] 74.04693  78.77332  87.83848 107.17439  72.41524  73.03061 100.9795
## [4,] 79.71828 116.43861 115.69739  89.75133  91.13868  81.90057 113.7462
## [5,] 78.88008  84.50401  93.83161  61.71240 103.68994 106.73150 103.6196
## [6,] 68.44734  82.94422  84.46213  95.86719  85.35943  72.89358 100.2575
##          [,58]    [,59]    [,60]     [,61]     [,62]     [,63]     [,64]
## [1,]  96.36896 112.2220 83.79947  71.06491 114.69837  74.17291  85.21497
## [2,]  62.15468 115.5782 98.82253  73.68237  84.55303 102.82393  92.43322
## [3,] 113.00395 121.6394 91.51500  91.58702 105.16874  78.96232  67.52929
## [4,]  94.42646 109.1338 85.41433 120.81435  61.78053 106.82460  89.83049
## [5,]  81.06355  85.5995 95.00029  80.64964 116.68958  77.07958 142.38615
## [6,]  82.44775 122.5116 76.06591  86.21313  79.40569  89.53046  57.70157
##          [,65]     [,66]     [,67]     [,68]     [,69]     [,70]     [,71]
## [1,] 123.53302  75.09636 105.05361  46.72012  87.61689  92.51821  85.47438
## [2,] 106.69894  56.29514  74.87655  74.26861 119.83519  90.01411  89.91050
## [3,] 103.08250  89.80932 106.54392  88.75480  75.70439 113.67311  87.40837
## [4,]  93.97414 101.43004  69.19722 112.16747  87.74624  81.47290 110.80544
## [5,] 117.72531 100.89407 126.31802  76.72892 105.22161  68.70101  93.13085
## [6,]  98.20378  60.20179  71.04590 101.66802 104.07440 100.24870  78.04857
##         [,72]     [,73]     [,74]     [,75]     [,76]     [,77]     [,78]
## [1,] 79.93236 105.90522  72.85408  74.91134 135.95439  95.70393  71.06373
## [2,] 75.75241  90.02719 103.16196  85.27746 109.04085  80.96346  90.71469
## [3,] 92.62988  99.59608  74.06661  65.79395 128.71984  95.45473  76.65384
## [4,] 88.60826  70.14203  67.44904  69.29276  94.53355 110.24893 104.15923
## [5,] 74.08749  76.64789  98.89351 106.89527  83.47693 113.45381  96.77625
## [6,] 98.75957 102.23073  82.39267  74.12137 109.06406  81.75541  84.00753
##          [,79]     [,80]     [,81]    [,82]    [,83]     [,84]     [,85]
## [1,]  89.32663 104.25145  86.37346 76.56883 54.35930  80.38250  61.71893
## [2,]  70.77344  82.38640  92.07069 79.09800 67.75098  80.28355  99.66121
## [3,]  91.40880 107.37741 106.96528 94.42488 73.99136  79.66071  53.83543
## [4,]  96.51351  80.18419 132.75648 93.28466 84.01030  88.99518 109.95652
## [5,]  92.14475  99.96386  88.07508 60.78771 89.66794 106.33168  86.25794
## [6,] 111.15906 102.65876  90.07584 82.82322 89.93502  67.99338  62.15562
##          [,86]    [,87]     [,88]     [,89]     [,90]     [,91]    [,92]
## [1,] 100.23167 66.86240 101.94611  82.71912  50.16177  75.54637 95.07555
## [2,]  67.16279 77.65732 123.78264  63.90098  97.80255  88.72315 54.10255
```

```
## [3,] 103.06101 76.69950  79.82957  82.67994  73.44536  80.99827 97.74294
## [4,]  76.46541 71.28070  82.98728  85.75574 135.91902 103.59858 59.46221
## [5,] 104.61860 79.87440 112.82368  81.60003 108.65166  71.93510 90.69838
## [6,]  72.14240 57.05908  66.92854 101.12745  87.04630  85.22077 95.80026
##           [,93]     [,94]     [,95]     [,96]     [,97]     [,98]     [,99]
## [1,] 115.69716  96.89901 117.22828  82.73116  55.55069 100.35862 113.93078
## [2,]  91.03181 106.14587 109.56255 103.81450  94.05802 108.89880 115.16095
## [3,]  95.16952  93.07092  98.32585  80.92704  73.34555 110.93921 110.99403
## [4,]  61.19565  93.75019  93.84321 132.63659 106.89373 100.01239  84.66612
## [5,]  65.30884 106.13911  90.60121 123.43871  93.59106  80.15386  56.87254
## [6,] 123.19375  89.15349 115.91111  87.80924  82.94939  92.11696 126.34135
##          [,100]
## [1,] 108.63270
## [2,] 104.56106
## [3,]  72.99134
## [4,]  91.21688
## [5,] 108.32559
## [6,]  95.01004
```

Plot the density of these angles.

```
pacman::p_load(ggplot2)

ggplot(data.frame(angles = c(all_angles(X)) )) +
  aes(x = angles) +
  geom_density()
```



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

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

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

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

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


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

head(all_angles_cpp(X))
```

```
##      [,1]     [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,]   NA 81.16117  57.77033 114.16515 101.27661  61.21775  89.58055 137.85245
## [2,]   NA       NA 108.90859  73.09477  78.60266  81.46916  67.69329  89.42244
## [3,]   NA       NA        NA 101.19089 108.00721  64.93008 116.51686 107.18585
## [4,]   NA       NA        NA        NA  68.14430 100.66390  66.32368  66.34431
## [5,]   NA       NA        NA        NA        NA 126.60591  87.69528  97.57780
## [6,]   NA       NA        NA        NA        NA        NA 100.90942 102.97756
##           [,9]     [,10]     [,11]     [,12]     [,13]    [,14]    [,15]
## [1,]  73.40466  75.28467 100.88988 125.37764  83.54642 71.56929 83.63185
## [2,] 109.53841  95.43691  92.88879  74.45398 111.11593 62.98008 62.11412
## [3,]  71.12173 114.21100  93.81506 142.76435  49.46919 77.77102 69.99038
## [4,] 108.14151 110.27679 117.37660  68.57413  88.34409 73.00381 71.52302
## [5,]  70.58512  77.88864  69.24211  96.98414  77.19271 65.39729 59.12058
## [6,] 114.72779  90.64767 106.79555  97.85607  98.20962 88.28579 99.66352
##          [,16]     [,17]     [,18]     [,19]     [,20]     [,21]     [,22]
## [1,] 108.15029  80.47386  69.88627  95.81957  72.24109 116.39859 104.24731
## [2,] 127.67331 100.49852  90.49341 105.00398  56.42260  92.97630  76.67629
## [3,]  86.03953  80.00854  84.13791 104.09054  86.77594  94.48884 115.91251
## [4,]  86.40649 109.42637 144.44547  94.93526 102.14206  95.39209  41.63351
```

```
## [5,]  90.48768  87.07616 120.76184 109.13398 101.82663 100.99935  79.60515
## [6,]  83.70447  78.89719  53.22518  82.48579  79.23761 105.55959  89.04257
##           [,23]     [,24]     [,25]     [,26]     [,27]     [,28]     [,29]
## [1,]  85.88755 121.59733  96.36770  59.09311  88.92134 117.94577 112.31973
## [2,]  94.76953 127.56282 108.54355  82.53601 109.22306  93.88584  87.86614
## [3,] 104.79080  79.87742 106.68748  80.87283  63.62968 100.51085  79.78635
## [4,] 114.23611 105.67893  91.08844 105.49746  78.39736  74.89934  53.69837
## [5,]  84.47240 102.35731  76.79643  68.41108  87.06096  78.49255  91.27065
## [6,] 103.05157  86.97943 121.60471  81.66763 110.98841 115.90122  87.43417
##           [,30]    [,31]     [,32]     [,33]     [,34]     [,35]     [,36]
## [1,]  86.73382 94.68297  91.01225  76.98852 118.55744  58.05465  97.11273
## [2,]  68.34984 88.23935 110.74406  91.29933  64.46751 119.99194  79.38676
## [3,] 115.33571 77.74512  88.18129 104.05808 130.41612  62.41351  80.29929
## [4,] 102.97036 72.17007  91.39582  70.05170  77.16326 112.72763  78.39601
## [5,] 116.61000 84.33400 122.96943  72.45947  65.10680 113.11310  81.97488
## [6,]  92.37110 92.49430  71.41056  93.31917  96.82635  57.27462 101.80407
##           [,37]    [,38]     [,39]    [,40]     [,41]     [,42]    [,43]
## [1,]  92.17583 50.51956 113.57451 75.62228  46.12789  91.94107 61.45017
## [2,]  49.11674 50.08688  82.62431 79.48008 112.09166 106.66696 86.60908
## [3,] 118.71304 76.49055 112.14754 98.02088  45.08447  74.41584 48.92171
## [4,]  89.37382 82.16688  94.78296 99.78112 106.13172  68.12420 80.08935
## [5,]  90.34651 91.04105  75.27478 54.66155 101.69273 118.88243 78.15780
## [6,]  68.24650 69.22944 117.76135 94.01005  63.86170  63.03104 74.84821
##           [,44]     [,45]     [,46]     [,47]     [,48]     [,49]    [,50]
## [1,]  96.66162  86.35206  95.27949  91.22103 105.91160  69.95696 69.64706
## [2,]  86.98889  76.33683  84.85278  84.39393  69.46542 132.10142 87.95990
## [3,] 114.84665  83.47199  98.43183  82.33443 138.53587  60.10103 39.96819
## [4,] 102.37792 109.18869  80.63793  65.90388  56.18213 100.86258 76.09546
## [5,]  83.68637 111.76222  41.84077 110.20866  66.61775  97.24386 85.28210
## [6,]  90.40472  80.62756 103.80985  65.64481 103.34726  95.26936 77.85898
##          [,51]     [,52]     [,53]     [,54]     [,55]     [,56]    [,57]
## [1,] 69.94873  66.01345  79.19810  91.50286  87.01856 112.44672 122.0939
## [2,] 85.63129 111.24241  99.22366  71.01450 127.45684  94.26655 121.1606
## [3,] 74.04693  78.77332  87.83848 107.17439  72.41524  73.03061 100.9795
## [4,] 79.71828 116.43861 115.69739  89.75133  91.13868  81.90057 113.7462
## [5,] 78.88008  84.50401  93.83161  61.71240 103.68994 106.73150 103.6196
## [6,] 68.44734  82.94422  84.46213  95.86719  85.35943  72.89358 100.2575
##           [,58]    [,59]    [,60]     [,61]     [,62]     [,63]     [,64]
## [1,]  96.36896 112.2220 83.79947  71.06491 114.69837  74.17291  85.21497
## [2,]  62.15468 115.5782 98.82253  73.68237  84.55303 102.82393  92.43322
## [3,] 113.00395 121.6394 91.51500  91.58702 105.16874  78.96232  67.52929
## [4,]  94.42646 109.1338 85.41433 120.81435  61.78053 106.82460  89.83049
## [5,]  81.06355  85.5995 95.00029  80.64964 116.68958  77.07958 142.38615
## [6,]  82.44775 122.5116 76.06591  86.21313  79.40569  89.53046  57.70157
##           [,65]     [,66]     [,67]     [,68]     [,69]     [,70]     [,71]
## [1,] 123.53302  75.09636 105.05361  46.72012  87.61689  92.51821  85.47438
## [2,] 106.69894  56.29514  74.87655  74.26861 119.83519  90.01411  89.91050
## [3,] 103.08250  89.80932 106.54392  88.75480  75.70439 113.67311  87.40837
## [4,]  93.97414 101.43004  69.19722 112.16747  87.74624  81.47290 110.80544
## [5,] 117.72531 100.89407 126.31802  76.72892 105.22161  68.70101  93.13085
## [6,]  98.20378  60.20179  71.04590 101.66802 104.07440 100.24870  78.04857
##          [,72]     [,73]     [,74]     [,75]     [,76]     [,77]     [,78]
## [1,] 79.93236 105.90522  72.85408  74.91134 135.95439  95.70393  71.06373
## [2,] 75.75241  90.02719 103.16196  85.27746 109.04085  80.96346  90.71469
```

```
## [3,] 92.62988  99.59608  74.06661  65.79395 128.71984  95.45473  76.65384
## [4,] 88.60826  70.14203  67.44904  69.29276  94.53355 110.24893 104.15923
## [5,] 74.08749  76.64789  98.89351 106.89527  83.47693 113.45381  96.77625
## [6,] 98.75957 102.23073  82.39267  74.12137 109.06406  81.75541  84.00753
##           [,79]     [,80]     [,81]    [,82]    [,83]     [,84]     [,85]
## [1,]  89.32663 104.25145  86.37346 76.56883 54.35930  80.38250  61.71893
## [2,]  70.77344  82.38640  92.07069 79.09800 67.75098  80.28355  99.66121
## [3,]  91.40880 107.37741 106.96528 94.42488 73.99136  79.66071  53.83543
## [4,]  96.51351  80.18419 132.75648 93.28466 84.01030  88.99518 109.95652
## [5,]  92.14475  99.96386  88.07508 60.78771 89.66794 106.33168  86.25794
## [6,] 111.15906 102.65876  90.07584 82.82322 89.93502  67.99338  62.15562
##           [,86]    [,87]     [,88]     [,89]     [,90]     [,91]    [,92]
## [1,] 100.23167 66.86240 101.94611  82.71912  50.16177  75.54637 95.07555
## [2,]  67.16279 77.65732 123.78264  63.90098  97.80255  88.72315 54.10255
## [3,] 103.06101 76.69950  79.82957  82.67994  73.44536  80.99827 97.74294
## [4,]  76.46541 71.28070  82.98728  85.75574 135.91902 103.59858 59.46221
## [5,] 104.61860 79.87440 112.82368  81.60003 108.65166  71.93510 90.69838
## [6,]  72.14240 57.05908  66.92854 101.12745  87.04630  85.22077 95.80026
##           [,93]     [,94]     [,95]     [,96]     [,97]     [,98]     [,99]
## [1,] 115.69716  96.89901 117.22828  82.73116  55.55069 100.35862 113.93078
## [2,]  91.03181 106.14587 109.56255 103.81450  94.05802 108.89880 115.16095
## [3,]  95.16952  93.07092  98.32585  80.92704  73.34555 110.93921 110.99403
## [4,]  61.19565  93.75019  93.84321 132.63659 106.89373 100.01239  84.66612
## [5,]  65.30884 106.13911  90.60121 123.43871  93.59106  80.15386  56.87254
## [6,] 123.19375  89.15349 115.91111  87.80924  82.94939  92.11696 126.34135
##          [,100]
## [1,] 108.63270
## [2,] 104.56106
## [3,]  72.99134
## [4,]  91.21688
## [5,] 108.32559
## [6,]  95.01004
```

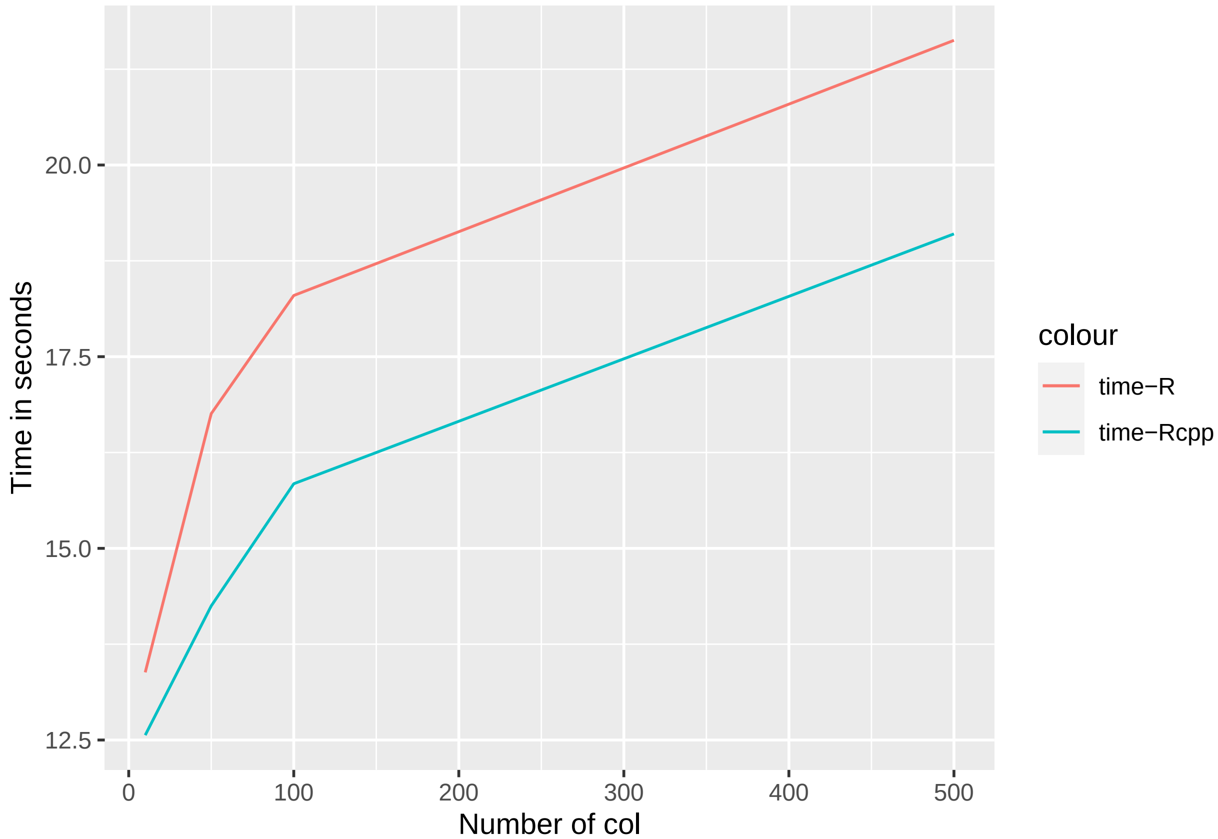
Test the time difference between these functions for `n = 1000` and `Nvec = 100, 500, 1000, 5000` using the package `microbenchmark`. Store the results in a matrix with rows representing `Nvec` and two columns for base R and Rcpp.

```r
Nvec = c(10, 50, 100, 500)

n <- 1000
time_for_R <- c()
time_for_cpp <- c()

for (i in 1:length(Nvec)) {
  X <- c()

  for (j in 1:n) {
    X <- cbind(X, rnorm(Nvec[i]))
  }

  time_for_R <- c(
    time_for_R,
    mean(microbenchmark(
      angles_r = all_angles(X),
```

```r
      times = 3,
      unit = "s"
    )$time)
  )

  time_for_cpp <- c(
    time_for_cpp,
    mean(microbenchmark(angles_cpp = all_angles_cpp(X),
      times = 3,
      unit = "s"
    )$time)
  )

}
```

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

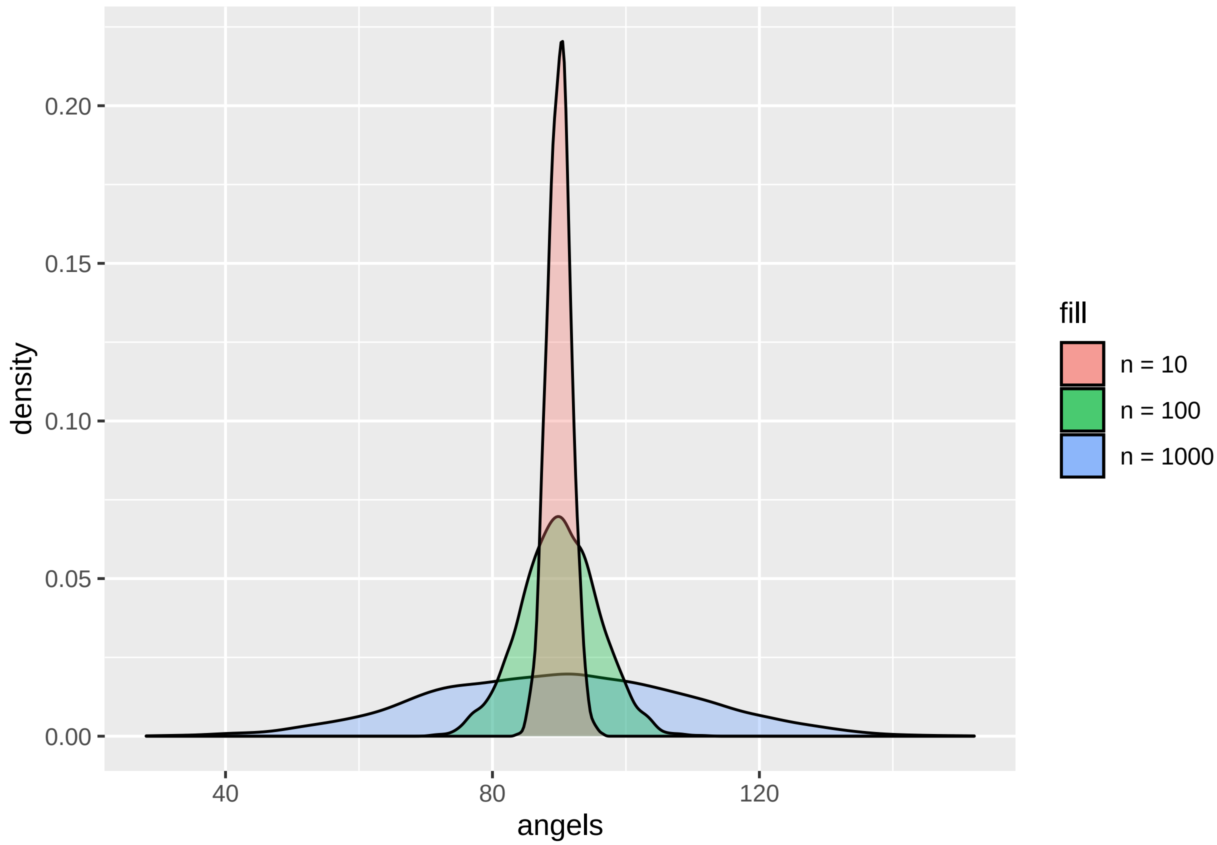
```r
ggplot() +
  geom_line(aes(x = Nvec, y = log(time_for_R), col = "time-R")) +

  geom_line(aes(x = Nvec, y = log(time_for_cpp), col = "time-Rcpp")) +

  xlab("Number of col") +

  ylab("Time in seconds")
```



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

```
Nvec = 100
X <- c()
for (i in 1:10) {
  X <- cbind(X, rnorm(Nvec))
}
a1 <- all_angles(X)

X <- c()
for (i in 1:100) {
  X <- cbind(X, rnorm(Nvec))
}
a2 <- all_angles(X)

X <- c()
for (i in 1:1000) {
  X <- cbind(X, rnorm(Nvec))
}
a3 <- all_angles(X)

ggplot() +
  geom_density(aes(x = a1, fill = "red"), alpha = .33) +
  geom_density(aes(x = a2, fill = "green"), alpha = .33) +
  geom_density(aes(x = a3, fill = "blue"), alpha = .33) +
```

```
  scale_fill_discrete(labels = c("n = 10", "n = 100", "n = 1000")) +
  xlab("angels")
```



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

```
nth_fibonacci <- function(n, start) {

  if (n == 1 | n == 2)
    return(start)
  else
    return(nth_fibonacci(n-1, start) + nth_fibonacci(n-2, start))

}
```

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

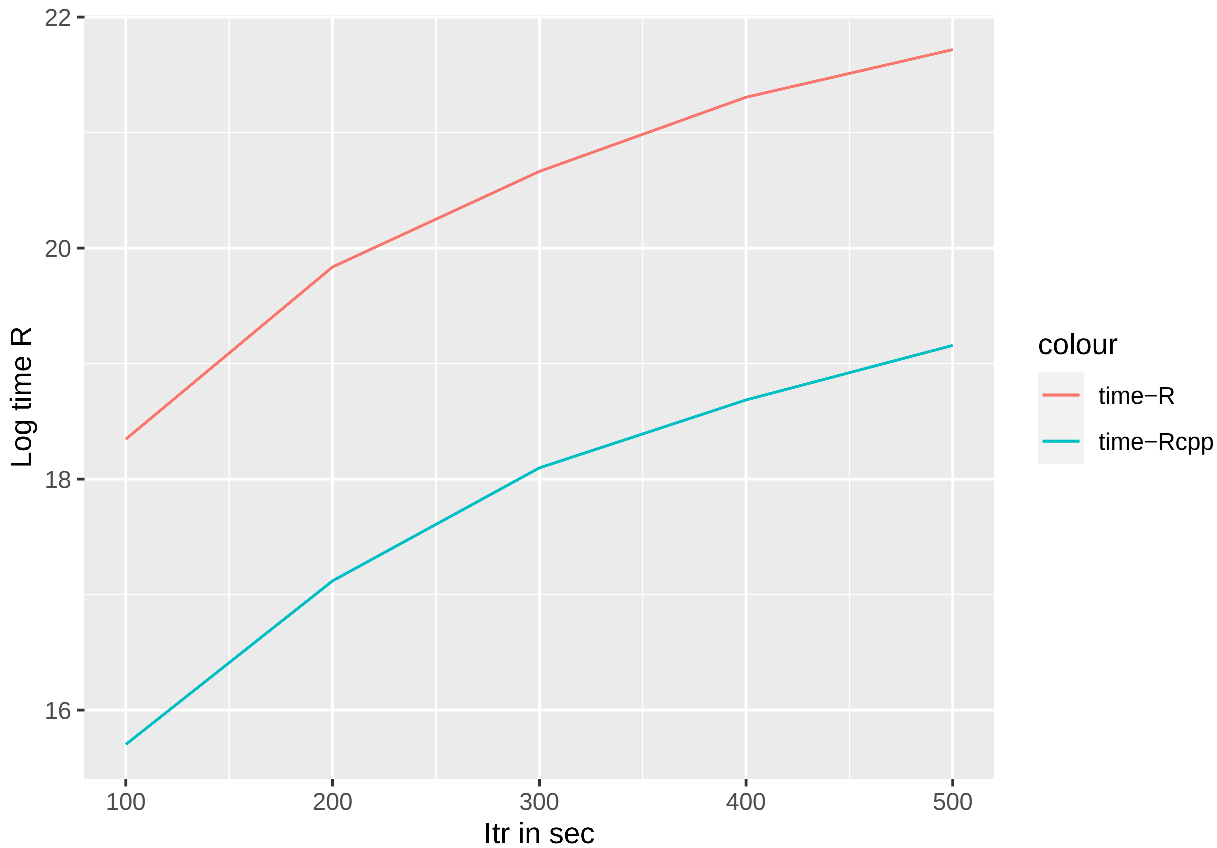
```
cppFunction("
  double nth_fibonnaci_cpp(int n, double start) {
     if (n - 1 <= 1)
        return start;
     return nth_fibonnaci_cpp(n-1, start) + nth_fibonnaci_cpp(n-2, start);
  }
")
```

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.

```
n <- 1000
Nvec <- c(100, 200, 300, 400, 500)

time_for_R <- c()

time_for_cpp <- c()

for (i in 1:length(Nvec)){
  X <- c()

  for (j in 1:n) {
    X <- cbind(X, rnorm(Nvec[i]))
  }

  time_for_R <- c(time_for_R, mean(microbenchmark(angles_r = all_angles(X), times = 3, unit = "s")$time

  time_for_cpp <- c(time_for_cpp, mean(microbenchmark(angles_cpp = all_angles_cpp(X), times = 3, unit =
}
```

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

```
ggplot() +
  geom_line(aes(x = Nvec, y = log(time_for_R), col = "time-R")) +
  geom_line(aes(x = Nvec, y = log(time_for_cpp), col = "time-Rcpp")) +
  xlab("Itr in sec") +
  ylab("Log time R")
```



Data Wrangling / Munging / Carpentry

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

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

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

```
data(storms)
```

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), 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) %>%
  summarise(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_storm = max(wind, na.rm = TRUE)) %>%
  select(name, max_wind_storm, everything()) %>%
  arrange(max_wind_storm, year, day, hour)
```

```
## # A tibble: 10,010 x 14
## # Groups:   name [198]
##    name   max_wind_storm  year month   day  hour   lat  long status     category
##    <chr>           <int> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>      <ord>
##  1 AL101~             25  1991    10    24    12  13.4 -42.3 tropical ~ -1
##  2 AL101~             25  1991    10    24    18  13.7 -43.6 tropical ~ -1
##  3 AL101~             25  1991    10    25     0  13.8 -44.9 tropical ~ -1
##  4 AL101~             25  1991    10    25     6  14   -46.4 tropical ~ -1
##  5 AL101~             25  1991    10    25    12  14.1 -47.7 tropical ~ -1
##  6 AL012~             25  2000     6     7    18  21   -93   tropical ~ -1
##  7 AL012~             25  2000     6     8     0  20.9 -92.8 tropical ~ -1
##  8 AL012~             25  2000     6     8     6  20.7 -93.1 tropical ~ -1
##  9 AL012~             25  2000     6     8    12  20.8 -93.5 tropical ~ -1
## 10 AL022~             25  2001     7    11    18  10.9 -42.1 tropical ~ -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(desc(wind)) %>%
  slice(1) %>% #Take the first row
  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).

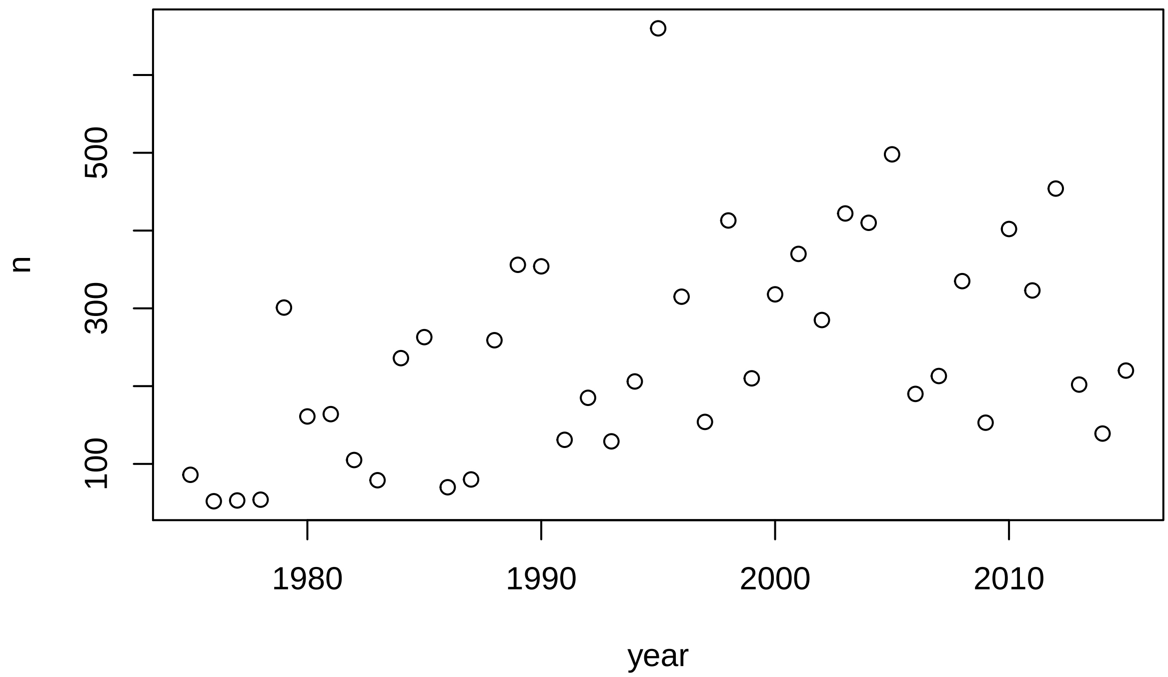
```
storms %>%
  group_by(name) %>%
  mutate(max_wind_speed = max(wind, na.rm = TRUE)) %>%
  mutate(max_pressure = max(pressure, na.rm = TRUE)) %>%
  mutate(max_hu_diameter = max(hu_diameter, na.rm = TRUE)) %>%
  mutate(max_ts_diameter = max(ts_diameter, na.rm = TRUE)) %>%
  select(max_pressure, max_wind_speed,  max_ts_diameter,max_hu_diameter) %>%
  ungroup() %>%
  distinct
```

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

```
## # A tibble: 198 x 5
##    name     max_pressure max_wind_speed max_ts_diameter max_hu_diameter
##    <chr>           <int>          <int>           <dbl>           <dbl>
##  1 Amy              1013             60            -Inf            -Inf
##  2 Caroline         1014            100            -Inf            -Inf
##  3 Doris            1005             95            -Inf            -Inf
##  4 Belle            1012            105            -Inf            -Inf
##  5 Gloria           1009            125            -Inf            -Inf
##  6 Anita            1012            150            -Inf            -Inf
##  7 Clara            1015             65            -Inf            -Inf
##  8 Evelyn           1010             70            -Inf            -Inf
##  9 Amelia           1010             45            -Inf            -Inf
## 10 Bess             1012             45            -Inf            -Inf
## # ... 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?

```
#data(storms)

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



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

```
storms %>%
  group_by(year,category) %>%
  summarise(tally = n())
```

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

```
## # A tibble: 233 x 3
## # Groups:   year [41]
##     year category tally
##    <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) %>%
  summarise(max_wind_speed = max(wind))
```

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

```
## # A tibble: 123 x 3
## # Groups:   year [41]
##     year status              max_wind_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_latitude = mean(lat), avrage_longitude = mean(long))
```

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

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

```
storms %>%
  group_by(name) %>%
  mutate(duration = (n()-1)*6) %>%
  summarise(neareast_6hr_increment = (round((n()-1)*6/6) * 6)) %>%
  distinct
```

```
## # A tibble: 198 x 2
##    name     neareast_6hr_increment
##    <chr>                     <dbl>
##  1 AL011993                     42
##  2 AL012000                     18
##  3 AL021992                     24
##  4 AL021994                     30
##  5 AL021999                     18
##  6 AL022000                     66
##  7 AL022001                     24
##  8 AL022003                     18
##  9 AL022006                     24
## 10 AL031987                    186
## # ... 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(name) %>%
  mutate(storm_number = dense_rank(paste(year, month, day)))
```

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

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

```
pacman::p_load(lubridate)

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

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

```
##  2 1975-06-27 06:00:00 Amy    1975     6    27     6  28.5 -79   tropi~ -1
##  3 1975-06-27 12:00:00 Amy    1975     6    27    12  29.5 -79   tropi~ -1
##  4 1975-06-27 18:00:00 Amy    1975     6    27    18  30.5 -79   tropi~ -1
##  5 1975-06-28 00:00:00 Amy    1975     6    28     0  31.5 -78.8 tropi~ -1
##  6 1975-06-28 06:00:00 Amy    1975     6    28     6  32.4 -78.7 tropi~ -1
##  7 1975-06-28 12:00:00 Amy    1975     6    28    12  33.3 -78   tropi~ -1
##  8 1975-06-28 18:00:00 Amy    1975     6    28    18  34   -77   tropi~ -1
##  9 1975-06-29 00:00:00 Amy    1975     6    29     0  34.4 -75.8 tropi~ 0
## 10 1975-06-29 06:00:00 Amy    1975     6    29     6  34   -74.8 tropi~ 0
## # ... with 10,000 more rows, and 4 more variables: wind <int>, pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>
```

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

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

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

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

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

```
                           abbr = FALSE),
                         day),
    year,
    sep = ", ")
  )
```

```
## # A tibble: 198 x 2
##    name     start_date
##    <chr>    <chr>
##  1 AL011993 Tuesday, June 1, 1993
##  2 AL012000 Wednesday, June 7, 2000
##  3 AL021992 Thursday, June 25, 1992
##  4 AL021994 Wednesday, July 20, 1994
##  5 AL021999 Friday, July 2, 1999
##  6 AL022000 Friday, June 23, 2000
##  7 AL022001 Wednesday, July 11, 2001
##  8 AL022003 Wednesday, June 11, 2003
##  9 AL022006 Monday, July 17, 2006
## 10 AL031987 Sunday, August 9, 1987
## # ... with 188 more rows
```

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

```
x = (1:10) / 10

storms <- storms %>%
  mutate(decile_windspeed = cut(wind, quantile(wind, x), labels =FALSE))

storms
```

```
## # 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)

serious_storms
```

```
## # 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>, decile_windspeed <int>
```

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

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

```
## # A tibble: 779 x 13
##    name      year month   day  hour lat_long    status   category  wind pressure
##    <chr>    <dbl> <dbl> <int> <dbl> <chr>       <chr>    <ord>    <int>    <int>
##  1 Caroline  1975     8    31     0 24 / -97    hurrica~ 3          100      973
##  2 Caroline  1975     8    31     6 24.1 / -97~ hurrica~ 3          100      963
##  3 Belle     1976     8     8    18 29.5 / -75~ hurrica~ 3          100      958
##  4 Belle     1976     8     9     0 30.9 / -75~ hurrica~ 3          105      957
##  5 Belle     1976     8     9     6 32.5 / -75~ hurrica~ 3          105      959
##  6 Anita     1977     9     1    18 25.2 / -95~ hurrica~ 3          110      945
##  7 Anita     1977     9     2     0 24.6 / -96~ hurrica~ 5          140      931
##  8 Anita     1977     9     2     6 24.2 / -97~ 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~ hurrica~ 4          115      947
## # ... with 769 more rows, and 3 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <int>
```

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, na.rm = TRUE),
    avg_pressure = mean(pressure, na.rm = TRUE),
    avg_ts_diameter = mean(ts_diameter, na.rm = TRUE),
    avg_hu_diameter = mean(hu_diameter, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 5
##   category avg_wind_speed avg_pressure avg_ts_diameter avg_hu_diameter
```

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

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

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

storms
```

```
## # A tibble: 3,482 x 19
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Alex   2004     7    31    18  30.3 -78.3 tropical d~ -1          25     1010
##  2 Alex   2004     8     1     0  31   -78.8 tropical d~ -1          25     1009
##  3 Alex   2004     8     1     6  31.5 -79   tropical d~ -1          25     1009
##  4 Alex   2004     8     1    12  31.6 -79.1 tropical d~ -1          30     1009
##  5 Alex   2004     8     1    18  31.6 -79.2 tropical s~ 0           35     1009
##  6 Alex   2004     8     2     0  31.5 -79.3 tropical s~ 0           35     1007
##  7 Alex   2004     8     2     6  31.4 -79.4 tropical s~ 0           40     1005
##  8 Alex   2004     8     2    12  31.3 -79   tropical s~ 0           50      992
##  9 Alex   2004     8     2    18  31.8 -78.7 tropical s~ 0           50      993
## 10 Alex   2004     8     3     0  32.4 -78.2 tropical s~ 0           60      987
## # ... with 3,472 more rows, and 8 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <int>, max_category <ord>,
## #   max_wind <int>, max_pressure <int>, max_ts_diameter <dbl>,
## #   max_hu_diameter <dbl>
```

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

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

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

  #Conversion for angles
```

```r
  lat1 = lat1 * 180/pi

  lat2 = lat2 * 180/pi

  long1 = long1 * 180/pi

  long2 = long2 * 180/pi


  #https://en.wikipedia.org/wiki/Haversine_formula
  a = sin(lat2 - lat1 / 2)^2 + (cos(lat2) * cos(lat1)) * sin(long2 - long1 / 2) ^ 2
  b = 2 * atan2(sqrt(a), sqrt(1 - a))

  distance = 6373.0 * b

  return(distance)
}

storms <- storms %>%
  mutate(distance_to_miami = distance_to_miami(lat, long, MIAMI_LAT_LONG_COORDS[1], MIAMI_LAT_LONG_COORI

storms
```

```
## # A tibble: 3,482 x 20
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Alex   2004     7    31    18  30.3 -78.3 tropical d~ -1          25     1010
##  2 Alex   2004     8     1     0  31   -78.8 tropical d~ -1          25     1009
##  3 Alex   2004     8     1     6  31.5 -79   tropical d~ -1          25     1009
##  4 Alex   2004     8     1    12  31.6 -79.1 tropical d~ -1          30     1009
##  5 Alex   2004     8     1    18  31.6 -79.2 tropical s~ 0           35     1009
##  6 Alex   2004     8     2     0  31.5 -79.3 tropical s~ 0           35     1007
##  7 Alex   2004     8     2     6  31.4 -79.4 tropical s~ 0           40     1005
##  8 Alex   2004     8     2    12  31.3 -79   tropical s~ 0           50      992
##  9 Alex   2004     8     2    18  31.8 -78.7 tropical s~ 0           50      993
## 10 Alex   2004     8     3     0  32.4 -78.2 tropical s~ 0           60      987
## # ... with 3,472 more rows, and 9 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <int>, max_category <ord>,
## #   max_wind <int>, max_pressure <int>, max_ts_diameter <dbl>,
## #   max_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.

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

storms
```

```
## # A tibble: 3,482 x 21
## # Groups:   name [114]
```

```
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Alex   2004     7    31    18  30.3 -78.3 tropical d~ -1          25     1010
##  2 Alex   2004     8     1     0  31   -78.8 tropical d~ -1          25     1009
##  3 Alex   2004     8     1     6  31.5 -79   tropical d~ -1          25     1009
##  4 Alex   2004     8     1    12  31.6 -79.1 tropical d~ -1          30     1009
##  5 Alex   2004     8     1    18  31.6 -79.2 tropical s~ 0           35     1009
##  6 Alex   2004     8     2     0  31.5 -79.3 tropical s~ 0           35     1007
##  7 Alex   2004     8     2     6  31.4 -79.4 tropical s~ 0           40     1005
##  8 Alex   2004     8     2    12  31.3 -79   tropical s~ 0           50      992
##  9 Alex   2004     8     2    18  31.8 -78.7 tropical s~ 0           50      993
## 10 Alex   2004     8     3     0  32.4 -78.2 tropical s~ 0           60      987
## # ... with 3,472 more rows, and 10 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <int>, max_category <ord>,
## #   max_wind <int>, max_pressure <int>, max_ts_diameter <dbl>,
## #   max_hu_diameter <dbl>, distance_to_miami <dbl>, dist_from_prev <dbl>
```

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

```
storms %>%
  group_by(name) %>%
  summarize(
    Distance = sum(dist_from_prev),
    Displacement = paste(round(last(lat) - first(lat), 2), round(last(long) - first(long), 2), sep = " /
  )
```

```
## # A tibble: 114 x 3
##    name     Distance Displacement
##  * <chr>       <dbl> <chr>
##  1 AL022006   24361. 4.6 / 6.3
##  2 AL102004   36454. 4.7 / 5.4
##  3 Al202011   29375. 1.7 / 2.1
##  4 Alberto   216936. 11.5 / 8.9
##  5 Alex      389785. -7.1 / -23.6
##  6 Ana       220407. 22.6 / -48.4
##  7 Andrea     30050. 8.4 / 6.4
##  8 Arthur    204244. 24.8 / 19.9
##  9 Barry     168924. -2.7 / -11.2
## 10 Beryl     182274. 1.4 / -5.6
## # ... with 104 more rows
```

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

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

storms
```

```
## # A tibble: 3,482 x 22
## # Groups:   name [114]
```

```
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Alex   2004     7    31    18  30.3 -78.3 tropical d~ -1          25     1010
##  2 Alex   2004     8     1     0  31   -78.8 tropical d~ -1          25     1009
##  3 Alex   2004     8     1     6  31.5 -79   tropical d~ -1          25     1009
##  4 Alex   2004     8     1    12  31.6 -79.1 tropical d~ -1          30     1009
##  5 Alex   2004     8     1    18  31.6 -79.2 tropical s~ 0           35     1009
##  6 Alex   2004     8     2     0  31.5 -79.3 tropical s~ 0           35     1007
##  7 Alex   2004     8     2     6  31.4 -79.4 tropical s~ 0           40     1005
##  8 Alex   2004     8     2    12  31.3 -79   tropical s~ 0           50      992
##  9 Alex   2004     8     2    18  31.8 -78.7 tropical s~ 0           50      993
## 10 Alex   2004     8     3     0  32.4 -78.2 tropical s~ 0           60      987
## # ... with 3,472 more rows, and 11 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>, decile_windspeed <int>, max_category <ord>,
## #   max_wind <int>, max_pressure <int>, max_ts_diameter <dbl>,
## #   max_hu_diameter <dbl>, distance_to_miami <dbl>, dist_from_prev <dbl>,
## #   speed <dbl>
```

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

```
storms %>%
  group_by(name) %>%
  summarize(avg_ground_speed = mean(speed))
```

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

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

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

#Depict relationship
cor(ret[,2], ret[,3])
```

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

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

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

```
storms2 <- storms %>%
  group_by(name) %>%
  summarize(
    y = max(wind),
    max_category = max(category),
    pressure = max(pressure),
    speed = max(speed),
    total_distance_traveled = sum(dist_from_prev),
    status = last(status)
  ) %>%
  select(-name) %>%
  ungroup()

head(storms2)
```

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

Fit your model. Validate it.

```
n = nrow(storms2)
K = 5 # 1/5 split
test_indices = sample(1 : n, 1 / K * n)
train_indices = setdiff(1:n, test_indices)
X = select(storms2, -y)
y = storms2$y

#Testing and training
X_test = X[test_indices,]
y_test = y[test_indices]
X_train = X[train_indices,]
y_train = y[train_indices]
model = lm(y_train ~ ., data.frame(X_train))

#In sample metrics
is_Rsq = summary(model)$r.squared
is_RMSE = sqrt(mean((model$residuals)^2))

#Out sample metrics
```

```
y_hat_oos = predict(model, data.frame(X_test))
oos_residuals = y_test - y_hat_oos
oos_Rsq = 1 - sum(oos_residuals^2) / sum((y_test - mean(y_test))^2)
oos_RMSE = sqrt(mean((oos_residuals)^2))

#Put validations in table
validations = data.frame(
  Metric = c("IS R^2:", "In SSE:", "OOS R^2:", "OOS SE:"),
  Value = c(is_Rsq, is_RMSE, oos_Rsq, oos_RMSE)
)

validations
```

```
##     Metric     Value
## 1  IS R^2: 0.9751654
## 2  In SSE: 5.4616791
## 3 OOS R^2: 0.9757663
## 4  OOS SE: 4.9770212
```

Assess your level of success at this endeavor.

We see from the sample metrics that the model can pridict the maximum speed of a storm with having a margin of up to 7. This model achieved a high out of sample R^2 when limited to using only 5 features. These metrics state that this will be a good model.

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.

```
#Copied from prev lab
adult$fnlwgt = NULL
adult$marital_status = as.character(adult$marital_status)
adult$marital_status = ifelse(adult$marital_status == "Married-AF-spouse" | adult$marital_status == "Ma
adult$marital_status = as.factor(adult$marital_status)
adult$education = as.character(adult$education)
adult$education = ifelse(adult$education == "1st-4th" | adult$education == "Preschool", "<=4th", adult$
adult$education = as.factor(adult$education)
adult$education = NULL
tab = sort(table(adult$native_country))
adult$native_country = as.character(adult$native_country)
adult$native_country= ifelse(adult$native_country %in% names(tab[tab<50]), "Other", adult$native_countr
adult$native_country= as.factor(adult$native_country)
```

```
adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tab_worktype = sort(table(adult$worktype))
adult$occupation = NULL
adult$workclass = NULL
adult$worktype = as.character(adult$worktype)
adult$worktype = ifelse(adult$worktype %in% names(tab_worktype[tab_worktype<100]), "Other", adult$workty
adult$worktype = as.factor(adult$worktype)
adult$status = paste(as.character(adult$relationship), as.character(adult$marital_status), sep = ":")
adult$status = as.character(adult$status)
tab_status = sort(table(adult$status))
adult$relationship = NULL
adult$marital_status = NULL
adult$status = as.factor(adult$status)
```

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

```
Nsplitsize = 1000
```

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

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

Fit a vanilla logistic regression on the training set.

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

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

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

#Scores log & Brier
mean(ytrain * log(p_hat_train) + (1 - ytrain) * log(1 - p_hat_train))
```

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

```
mean(-(ytrain - p_hat_train) ^ 2)
```

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

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

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

```
Xmm_train = data.frame(model.matrix( ~ . , Xtrain))

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

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

dim(Xmm_train)
```

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

```
dim(Xmm_select)
```

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

```
dim(Xmm_test)
```

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

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

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

```
#Turn off warnings
options(warn = -1)

pacman::p_load(Matrix)
p_plus_one = ncol(Xmm_train)

#Grow list with iterations
predictor_by_iteration = c()
in_sample_brier_by_iteration = c()
oos_brier_by_iteration = c()

i = 1
repeat {

  all_briers = array(NA, p_plus_one)

  for (j_try in 1 : p_plus_one) {
```

```
    if (j_try %in% predictor_by_iteration) {

      next

    }

    sub = Xmm_train[, c(predictor_by_iteration, j_try), drop = FALSE]

    logistic_mod = (glm(ytrain ~ ., sub, family = "binomial"))

    phat_t = (predict(logistic_mod, sub, type = 'response'))

    all_briers[j_try] = -mean(-(ytrain - phat_t)^2)
  }

  j_star = which.max(all_briers)

  predictor_by_iteration = c(predictor_by_iteration, j_star)

  in_sample_brier_by_iteration = c(in_sample_brier_by_iteration, all_briers[j_star])


  sub = Xmm_train[, predictor_by_iteration, drop = FALSE]

  logistic_mod = (glm(ytrain ~ ., sub, family = "binomial"))

  phat_t = (predict(logistic_mod, sub, type = 'response'))

  all_briers[j_try] = -mean( - (ytrain - phat_t) ^ 2)

  phat_s = (predict(logistic_mod, Xmm_select[, predictor_by_iteration, drop = FALSE], type = 'response')

  oos_brier = -mean(-(yselect - phat_s)^2)

  oos_brier_by_iteration = c(oos_brier_by_iteration, oos_brier)

  cat(
    "i =", i,
    "is_brier =", all_briers[j_star],
    "oos_brier =", oos_brier,
    "predictor =", colnames(Xmm_train)[j_star],
    "\n"
  )

  i = i + 1

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

```
## i = 1 is_brier = 0.181356 oos_brier = 0.185548 predictor = X.Intercept.
## i = 2 is_brier = 0.181356 oos_brier = 0.185548 predictor = native_countryPoland
```

```
## i = 3 is_brier = 0.181356 oos_brier = 0.185548 predictor = statusNot.in.family.Married
## i = 4 is_brier = 0.181356 oos_brier = 0.185548 predictor = statusOther.relative.Separated
## i = 5 is_brier = 0.181356 oos_brier = 0.185548 predictor = statusOther.relative.Widowed
## i = 6 is_brier = 0.181356 oos_brier = 0.185548 predictor = statusOwn.child.Widowed
## i = 7 is_brier = 0.1813554 oos_brier = 0.1855417 predictor = worktypeTransport.moving.Self.emp.not.in
## i = 8 is_brier = 0.1813548 oos_brier = 0.1855661 predictor = statusUnmarried.Married.spouse.absent
## i = 9 is_brier = 0.1813542 oos_brier = 0.1855927 predictor = worktypeSales.Self.emp.not.inc
## i = 10 is_brier = 0.181353 oos_brier = 0.1856649 predictor = statusUnmarried.Widowed
## i = 11 is_brier = 0.1813499 oos_brier = 0.1856563 predictor = worktypeCraft.repair.Private
## i = 12 is_brier = 0.1813447 oos_brier = 0.1856134 predictor = native_countryIndia
## i = 13 is_brier = 0.1813373 oos_brier = 0.1856355 predictor = native_countryPuerto.Rico
## i = 14 is_brier = 0.1813246 oos_brier = 0.1859607 predictor = worktypeFarming.fishing.Private
## i = 15 is_brier = 0.1813123 oos_brier = 0.1857883 predictor = worktypeFarming.fishing.Self.emp.not.in
## i = 16 is_brier = 0.1812982 oos_brier = 0.1856838 predictor = statusNot.in.family.Separated
## i = 17 is_brier = 0.1812717 oos_brier = 0.1852927 predictor = worktypeProf.specialty.Federal.gov
## i = 18 is_brier = 0.1812449 oos_brier = 0.1853558 predictor = native_countryGuatemala
## i = 19 is_brier = 0.181218 oos_brier = 0.1857469 predictor = worktypeCraft.repair.Local.gov
## i = 20 is_brier = 0.1811902 oos_brier = 0.1856173 predictor = raceOther
## i = 21 is_brier = 0.1811586 oos_brier = 0.1855962 predictor = worktypeExec.managerial.State.gov
## i = 22 is_brier = 0.1811215 oos_brier = 0.1859505 predictor = worktypeAdm.clerical.Local.gov
## i = 23 is_brier = 0.1810644 oos_brier = 0.185881 predictor = native_countryDominican.Republic
## i = 24 is_brier = 0.1810644 oos_brier = 0.185881 predictor = statusOwn.child.Married.spouse.absent
## i = 25 is_brier = 0.1810073 oos_brier = 0.1858114 predictor = native_countryVietnam
## i = 26 is_brier = 0.1809499 oos_brier = 0.1860419 predictor = statusOwn.child.Married
## i = 27 is_brier = 0.1808553 oos_brier = 0.1860526 predictor = native_countryOther
## i = 28 is_brier = 0.1807887 oos_brier = 0.1862179 predictor = native_countryUnited.States
## i = 29 is_brier = 0.180699 oos_brier = 0.1868485 predictor = worktypeTech.support.Private
## i = 30 is_brier = 0.1805934 oos_brier = 0.1864382 predictor = worktypeOther.service.Local.gov
## i = 31 is_brier = 0.1804642 oos_brier = 0.1848996 predictor = worktypeExec.managerial.Self.emp.inc
## i = 32 is_brier = 0.1803137 oos_brier = 0.1846994 predictor = native_countryJapan
## i = 33 is_brier = 0.1801419 oos_brier = 0.1849772 predictor = worktypeProtective.serv.State.gov
## i = 34 is_brier = 0.1799592 oos_brier = 0.1847671 predictor = statusOther.relative.Divorced
## i = 35 is_brier = 0.179768 oos_brier = 0.1846089 predictor = worktypeProtective.serv.Private
## i = 36 is_brier = 0.1795723 oos_brier = 0.1842935 predictor = worktypeProf.specialty.Local.gov
## i = 37 is_brier = 0.179356 oos_brier = 0.1841564 predictor = native_countryChina
## i = 38 is_brier = 0.1791469 oos_brier = 0.1840683 predictor = native_countryColumbia
## i = 39 is_brier = 0.1789191 oos_brier = 0.1840311 predictor = worktypeOther.service.State.gov
## i = 40 is_brier = 0.1786884 oos_brier = 0.1838212 predictor = statusOwn.child.Divorced
## i = 41 is_brier = 0.1784501 oos_brier = 0.1838435 predictor = native_countryEl.Salvador
## i = 42 is_brier = 0.1782627 oos_brier = 0.1844303 predictor = statusOther.relative.Married.spouse.abs
## i = 43 is_brier = 0.1780273 oos_brier = 0.1841625 predictor = worktypeTransport.moving.Local.gov
## i = 44 is_brier = 0.1777802 oos_brier = 0.1838986 predictor = worktypeCraft.repair.Self.emp.not.inc
## i = 45 is_brier = 0.1775394 oos_brier = 0.1839145 predictor = worktypeSales.Self.emp.inc
## i = 46 is_brier = 0.1772784 oos_brier = 0.184464 predictor = worktypeAdm.clerical.State.gov
## i = 47 is_brier = 0.1770012 oos_brier = 0.1848479 predictor = native_countryEngland
## i = 48 is_brier = 0.1766289 oos_brier = 0.1852858 predictor = native_countryItaly
## i = 49 is_brier = 0.1762576 oos_brier = 0.1850986 predictor = worktypeTransport.moving.Private
## i = 50 is_brier = 0.1759073 oos_brier = 0.185645 predictor = statusOther.relative.Married
## i = 51 is_brier = 0.1755777 oos_brier = 0.1855656 predictor = worktypePriv.house.serv.Private
## i = 52 is_brier = 0.1752024 oos_brier = 0.1858937 predictor = worktypeOther
## i = 53 is_brier = 0.1748781 oos_brier = 0.1858285 predictor = native_countryGermany
## i = 54 is_brier = 0.1744952 oos_brier = 0.1864225 predictor = native_countryCuba
## i = 55 is_brier = 0.1741871 oos_brier = 0.186287 predictor = statusOwn.child.Separated
## i = 56 is_brier = 0.1737656 oos_brier = 0.1862193 predictor = native_countrySouth
```

```
## i = 57 is_brier = 0.1733164 oos_brier = 0.1853527 predictor = worktypeOther.service.Self.emp.not.inc
## i = 58 is_brier = 0.1728051 oos_brier = 0.1853208 predictor = worktypeProf.specialty.Self.emp.inc
## i = 59 is_brier = 0.1722497 oos_brier = 0.1846987 predictor = worktypeSales.Private
## i = 60 is_brier = 0.1717164 oos_brier = 0.1863781 predictor = worktypeProtective.serv.Local.gov
## i = 61 is_brier = 0.1711044 oos_brier = 0.1860013 predictor = statusNot.in.family.Widowed
## i = 62 is_brier = 0.1705002 oos_brier = 0.1857051 predictor = worktypeExec.managerial.Self.emp.not.i
## i = 63 is_brier = 0.1698833 oos_brier = 0.1865027 predictor = native_countryJamaica
## i = 64 is_brier = 0.1693691 oos_brier = 0.1866908 predictor = raceWhite
## i = 65 is_brier = 0.1686613 oos_brier = 0.1859704 predictor = statusUnmarried.Separated
## i = 66 is_brier = 0.1678313 oos_brier = 0.1864843 predictor = raceBlack
## i = 67 is_brier = 0.1671104 oos_brier = 0.1841216 predictor = worktypeMachine.op.inspct.Private
## i = 68 is_brier = 0.1664096 oos_brier = 0.1846154 predictor = raceAsian.Pac.Islander
## i = 69 is_brier = 0.165671 oos_brier = 0.1834925 predictor = worktypeProf.specialty.Self.emp.not.inc
## i = 70 is_brier = 0.164799 oos_brier = 0.1839977 predictor = native_countryPhilippines
## i = 71 is_brier = 0.1639532 oos_brier = 0.1829634 predictor = statusOther.relative.Never.married
## i = 72 is_brier = 0.1630177 oos_brier = 0.1798843 predictor = worktypeProf.specialty.Private
## i = 73 is_brier = 0.161836 oos_brier = 0.178388 predictor = worktypeHandlers.cleaners.Private
## i = 74 is_brier = 0.1604635 oos_brier = 0.1780931 predictor = worktypeExec.managerial.Local.gov
## i = 75 is_brier = 0.1590754 oos_brier = 0.1803847 predictor = native_countryMexico
## i = 76 is_brier = 0.1576239 oos_brier = 0.18131 predictor = statusNot.in.family.Married.spouse.absen
## i = 77 is_brier = 0.1561724 oos_brier = 0.1814974 predictor = worktypeExec.managerial.Federal.gov
## i = 78 is_brier = 0.154877 oos_brier = 0.1792748 predictor = worktypeAdm.clerical.Private
## i = 79 is_brier = 0.1530984 oos_brier = 0.1792153 predictor = worktypeProf.specialty.State.gov
## i = 80 is_brier = 0.1512046 oos_brier = 0.1803241 predictor = statusUnmarried.Divorced
## i = 81 is_brier = 0.1486265 oos_brier = 0.1798221 predictor = statusUnmarried.Never.married
## i = 82 is_brier = 0.1455114 oos_brier = 0.1793399 predictor = statusWife.Married
## i = 83 is_brier = 0.141789 oos_brier = 0.179233 predictor = statusNot.in.family.Divorced
## i = 84 is_brier = 0.1375809 oos_brier = 0.1772499 predictor = capital_loss
## i = 85 is_brier = 0.1330105 oos_brier = 0.1663411 predictor = hours_per_week
## i = 86 is_brier = 0.1290151 oos_brier = 0.1591097 predictor = worktypeExec.managerial.Private
## i = 87 is_brier = 0.1283621 oos_brier = 0.1569123 predictor = worktypeOther.service.Private
## i = 88 is_brier = 0.1242607 oos_brier = 0.1476126 predictor = education_num
## i = 89 is_brier = 0.1209538 oos_brier = 0.1422338 predictor = statusOwn.child.Never.married
## i = 90 is_brier = 0.1133092 oos_brier = 0.1362918 predictor = sexMale
## i = 91 is_brier = 0.1027663 oos_brier = 0.1329848 predictor = statusNot.in.family.Never.married
## i = 92 is_brier = 0.09516563 oos_brier = 0.1313902 predictor = age
## i = 93 is_brier = 0.08715781 oos_brier = 0.1264595 predictor = capital_gain
```

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

```
simulation_results = data.frame(

  iteration = 1 : length(in_sample_brier_by_iteration),

  in_sample_brier_by_iteration = in_sample_brier_by_iteration,

  oos_brier_by_iteration = oos_brier_by_iteration

)

ggplot(simulation_results) +
  geom_line(aes(x = iteration, y = in_sample_brier_by_iteration), color = "green") +
  geom_line(aes(x = iteration, y = oos_brier_by_iteration), color = "red") +
  ylab("Brier score")
```

0.175
0.150
0.125
0.100
Brier score
0
25
50
75
iteration