Lab 8

Tyron Samaroo

2AM April 26, 2020

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,skimr)
```

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

```
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   tropi~ -1          25     1013
## 2 Amy    1975     6    27     6  28.5 -79   tropi~ -1          25     1013
## 3 Amy    1975     6    27    12  29.5 -79   tropi~ -1          25     1013
## 4 Amy    1975     6    27    18  30.5 -79   tropi~ -1          25     1013
## 5 Amy    1975     6    28     0  31.5 -78.8 tropi~ -1          25     1012
## 6 Amy    1975     6    28     6  32.4 -78.7 tropi~ -1          25     1012
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

```
# Name and status need to factors

storms %<>%
  mutate(name = factor(name), status = factor(status))


orginal_col_names = colnames(storms)[1:3]
```

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

```
cat("Before Colnames Order ", orginal_col_names,"\n")
```

```
## Before Colnames Order  name year month
```

```
storms %<>%
  select(name, status, category, everything())
cat("After Colnames Order ", colnames(storms)[1:3])
```

```
## After Colnames Order  name status category
```

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

```
max(storms$year)
```

```
## [1] 2015
```

```
min(storms$year)
```

```
## [1] 1975
```

```
# smallest year is 1975 so no point to check between 1970 to 1979
storms %>%
  filter(year %in% 1975:1979)
```

```
## # A tibble: 546 x 13
```

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

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

```
storms %>%
  filter(category >= 4, wind >= 100)
```

```
## # A tibble: 416 x 13
##    name  status category  year month   day  hour   lat  long  wind pressure
##    <fct> <fct>  <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
##  1 Anita hurri~ 5         1977     9     2     0  24.6 -96.2   140      931
##  2 Anita hurri~ 5         1977     9     2     6  24.2 -97.1   150      926
##  3 Anita hurri~ 4         1977     9     2    12  23.7 -98     120      940
##  4 David hurri~ 4         1979     8    28     0  12.2 -52.9   115      947
##  5 David hurri~ 4         1979     8    28     6  12.5 -54.4   125      941
##  6 David hurri~ 4         1979     8    28    12  12.8 -55.7   130      938
##  7 David hurri~ 4         1979     8    28    18  13.2 -56.9   125      941
##  8 David hurri~ 4         1979     8    29     0  13.7 -58     120      944
##  9 David hurri~ 4         1979     8    29     6  14.2 -59.2   120      942
## 10 David hurri~ 4         1979     8    29    12  14.8 -60.3   125      938
## # ... with 406 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

Create a new feature `wind_speed_per_unit_pressure`.

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

head(storms$wind_speed_per_unit_pressure)
```

```
## [1] 0.02467917 0.02467917 0.02467917 0.02467917 0.02470356 0.02470356
```

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.

```
# All are missing in remaining data
#which(is.na(storms$ts_diameter))
#which(is.na(storms$hu_diameter))
storms %<>%
  mutate(average_diameter = (ts_diameter + hu_diameter) / 2)
```

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

```
#storms
storms %>%
  group_by(name) %>%
  summarize(max_wind_speed = max(wind))
```

```
## # A tibble: 198 x 2
##    name     max_wind_speed
##    <fct>             <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
```

```
head(storms)
```

```
## # A tibble: 6 x 15
##   name  status category  year month   day  hour   lat  long  wind pressure
##   <fct> <fct>  <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
## 1 Amy   tropi~ -1        1975     6    27     0  27.5 -79      25     1013
## 2 Amy   tropi~ -1        1975     6    27     6  28.5 -79      25     1013
## 3 Amy   tropi~ -1        1975     6    27    12  29.5 -79      25     1013
## 4 Amy   tropi~ -1        1975     6    27    18  30.5 -79      25     1013
## 5 Amy   tropi~ -1        1975     6    28     0  31.5 -78.8    25     1012
## 6 Amy   tropi~ -1        1975     6    28     6  32.4 -78.7    25     1012
## # ... with 4 more variables: ts_diameter <dbl>, hu_diameter <dbl>,
## #   wind_speed_per_unit_pressure <dbl>, average_diameter <dbl>
```

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 %<>%
 arrange(-wind, year)
storms$wind[1:50]
```

```
##  [1] 160 160 155 155 155 155 155 150 150 150 150 150 150 150 150 150 150 150 150 150
## [20] 150 150 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145
## [39] 145 140 140 140 140 140 140 140 140 140 140 140
```

Find the strongest storm by wind speed per year.

```
storms %>%
  group_by(year) %>%
  summarize(max_wind_speed = max(wind)) %>%
  arrange(-max_wind_speed)
```

```
## # A tibble: 41 x 2
##     year max_wind_speed
##    <dbl>          <int>
##  1  1988            160
##  2  2005            160
```

```
##  3  1998            155
##  4  1977            150
##  5  1979            150
##  6  1992            150
##  7  2007            150
##  8  2003            145
##  9  2004            145
## 10  1989            140
## # ... with 31 more rows
```

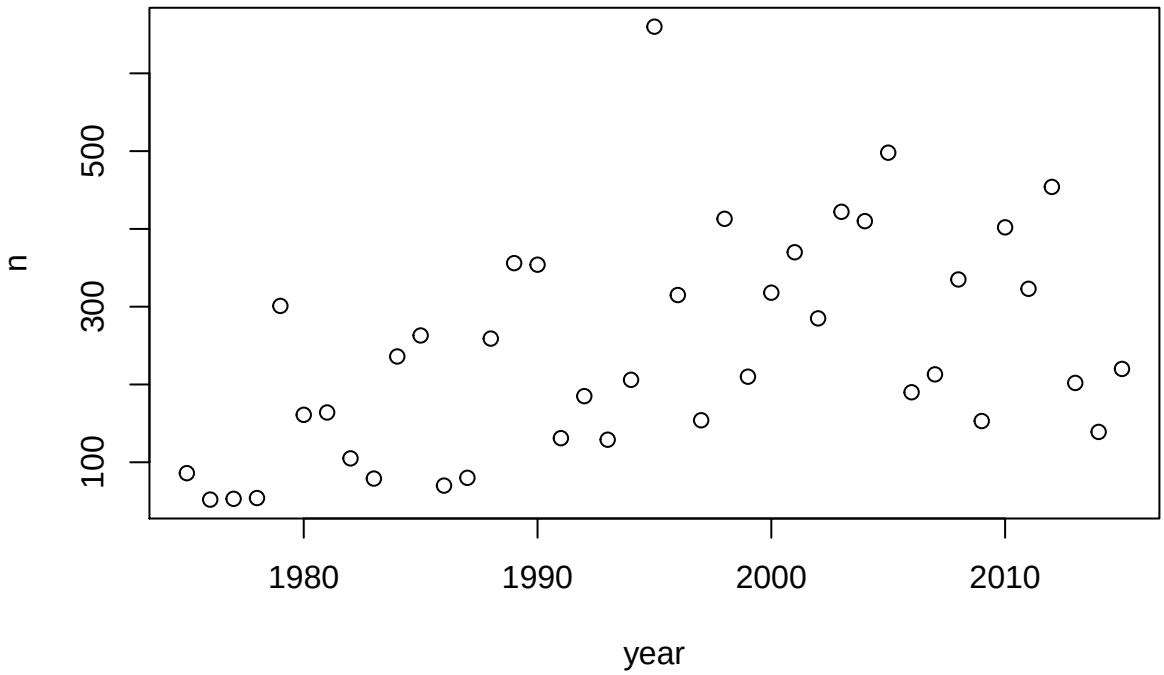
For each named storm, find its maximum category, wind speed, pressure and diameters. Do not allow the max to be NA (unless all the measurements for that storm were NA).

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

```
## # A tibble: 198 x 6
##    name  max_category max_wind_speed max_pressure max_ts_diameter
##    <fct> <ord>                 <int>        <int>           <dbl>
##  1 AL01~ -1                       30         1003            NA
##  2 AL01~ -1                       25         1010            NA
##  3 AL02~ -1                       30         1009            NA
##  4 AL02~ -1                       30         1017            NA
##  5 AL02~ -1                       30         1006            NA
##  6 AL02~ -1                       30         1010            NA
##  7 AL02~ -1                       25         1012            NA
##  8 AL02~ -1                       30         1010            NA
##  9 AL02~ 0                        45         1008            69.0
## 10 AL03~ 0                        40         1015            NA
## # ... with 188 more rows, and 1 more variable: max_hu_diameter <dbl>
```

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

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



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

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

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

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

```
storms %>%
  group_by(year,status) %>%
  summarise(max_wind_speed = max(wind))
```

```
## # A tibble: 123 x 3
## # Groups:   year [41]
##     year status              max_wind_speed
##    <dbl> <fct>                        <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) %>%
  summarise(average_latitude = mean(lat), average_longitude = mean(long))
```

```
## # A tibble: 198 x 3
##    name     average_latitude average_longitude
##    <fct>               <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) %>%
  summarise(duration = round(sum(hour) / 6))
```

```
## # A tibble: 198 x 2
##    name     duration
##    <fct>       <dbl>
##  1 AL011993       12
##  2 AL012000        6
##  3 AL021992        8
##  4 AL021994        9
##  5 AL021999        5
##  6 AL022000       18
##  7 AL022001        9
##  8 AL022003        6
##  9 AL022006        7
## 10 AL031987       48
## # ... with 188 more rows
```

Convert year, month, day, hour into the variable `timestamp` using the `lubridate` package.

```
pacman::p_load(lubridate)

storms %<>%
  unite(timestamp, year, month, day, hour,sep = "-", remove = FALSE)

storms
```

```
## # A tibble: 10,010 x 16
##    name  status category timestamp  year month   day  hour   lat  long  wind
##    <fct> <fct>  <ord>    <chr>     <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>
##  1 Gilb~ hurri~ 5        1988-9-1~  1988     9    14     0  19.7 -83.8   160
##  2 Wilma hurri~ 5        2005-10-~  2005    10    19    12  17.3 -82.8   160
##  3 Gilb~ hurri~ 5        1988-9-1~  1988     9    14     6  19.9 -85.3   155
##  4 Mitch hurri~ 5        1998-10-~  1998    10    26    18  16.9 -83.1   155
##  5 Mitch hurri~ 5        1998-10-~  1998    10    27     0  17.2 -83.8   155
##  6 Rita  hurri~ 5        2005-9-2~  2005     9    22     3  24.7 -87.3   155
##  7 Rita  hurri~ 5        2005-9-2~  2005     9    22     6  24.8 -87.6   155
##  8 Anita hurri~ 5        1977-9-2~  1977     9     2     6  24.2 -97.1   150
##  9 David hurri~ 5        1979-8-3~  1979     8    30    18  16.6 -66.2   150
## 10 David hurri~ 5        1979-8-3~  1979     8    31    18  17.9 -69.7   150
## # ... with 10,000 more rows, and 5 more variables: pressure <int>,
## #   ts_diameter <dbl>, hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_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(day_of_week = wday(timestamp,label = TRUE), week_of_year = week(timestamp))
head(storms)
```

```
## # A tibble: 6 x 18
##   name  status category timestamp  year month   day  hour   lat  long  wind
##   <fct> <fct>  <ord>    <chr>     <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>
## 1 Gilb~ hurri~ 5        1988-9-1~  1988     9    14     0  19.7 -83.8   160
## 2 Wilma hurri~ 5        2005-10-~  2005    10    19    12  17.3 -82.8   160
## 3 Gilb~ hurri~ 5        1988-9-1~  1988     9    14     6  19.9 -85.3   155
## 4 Mitch hurri~ 5        1998-10-~  1998    10    26    18  16.9 -83.1   155
## 5 Mitch hurri~ 5        1998-10-~  1998    10    27     0  17.2 -83.8   155
## 6 Rita  hurri~ 5        2005-9-2~  2005     9    22     3  24.7 -87.3   155
## # ... with 7 more variables: pressure <int>, ts_diameter <dbl>,
## #   hu_diameter <dbl>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>, day_of_week <ord>, week_of_year <dbl>
```

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

```
storms %>%
  group_by(name)  %>%
  summarise(data = min(timestamp))
```

```
## # A tibble: 198 x 2
##    name     data
##    <fct>    <chr>
##  1 AL011993 1993-5-31-12
##  2 AL012000 2000-6-7-18
##  3 AL021992 1992-6-25-12
```

```
##  4 AL021994 1994-7-20-12
##  5 AL021999 1999-7-2-18
##  6 AL022000 2000-6-23-0
##  7 AL022001 2001-7-11-18
##  8 AL022003 2003-6-11-0
##  9 AL022006 2006-7-17-12
## 10 AL031987 1987-8-10-0
## # ... with 188 more rows
```

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

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

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

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

```
storms %>%
  group_by(category) %>%
  summarise(avg_wind_speed = mean(wind),
            avg_pressure = mean(pressure),
            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.
```

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_COORDS = c(25.7617, -80.1918)
#TO-DO
```

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

```
#TO-DO
```

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.

```
#TO-DO
```

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

```
#TO-DO
```

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

```
#TO-DO
```

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

```
#TO-DO
```

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.

```
model = lm(wind ~ . ,data = storms)
summary(model)$sigma
```

```
## [1] 0.1938434
```

```
summary(model)$r.squared
```

```
## [1] 0.9999899
```

Fit your model. Validate it. Assess your level of success at this endeavor.

```
my_storm = copy(storms)
train_size = 8000
train_indices = sample(1 : nrow(my_storm), train_size)
my_storm_train = my_storm[train_indices, ]
y_train = my_storm_train$wind
X_train = my_storm_train %>% select(-wind)
test_indices = setdiff(1 : nrow(my_storm), train_indices)
my_storm_test = my_storm[test_indices, ]
y_test = my_storm_test$wind
X_test = my_storm_test %>% select(-wind)

model_val = lm(y_train ~ . ,data = data.frame(X_train))
summary(model_val)$sigma
```

```
## [1] 0.181668
```

```
summary(model_val)$r.squared
```

```
## [1] 0.999993
```

```
#y_hat_test = predict(model_val, X_test)
```

Interactions in linear models

Load the Boston Housing Data from package `MASS` and use `str` and `summary` to remind yourself of the features and their types and then use `?MASS::Boston` to read an English description of the features.

```
data(Boston, package = "MASS")
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
summary(Boston)
```

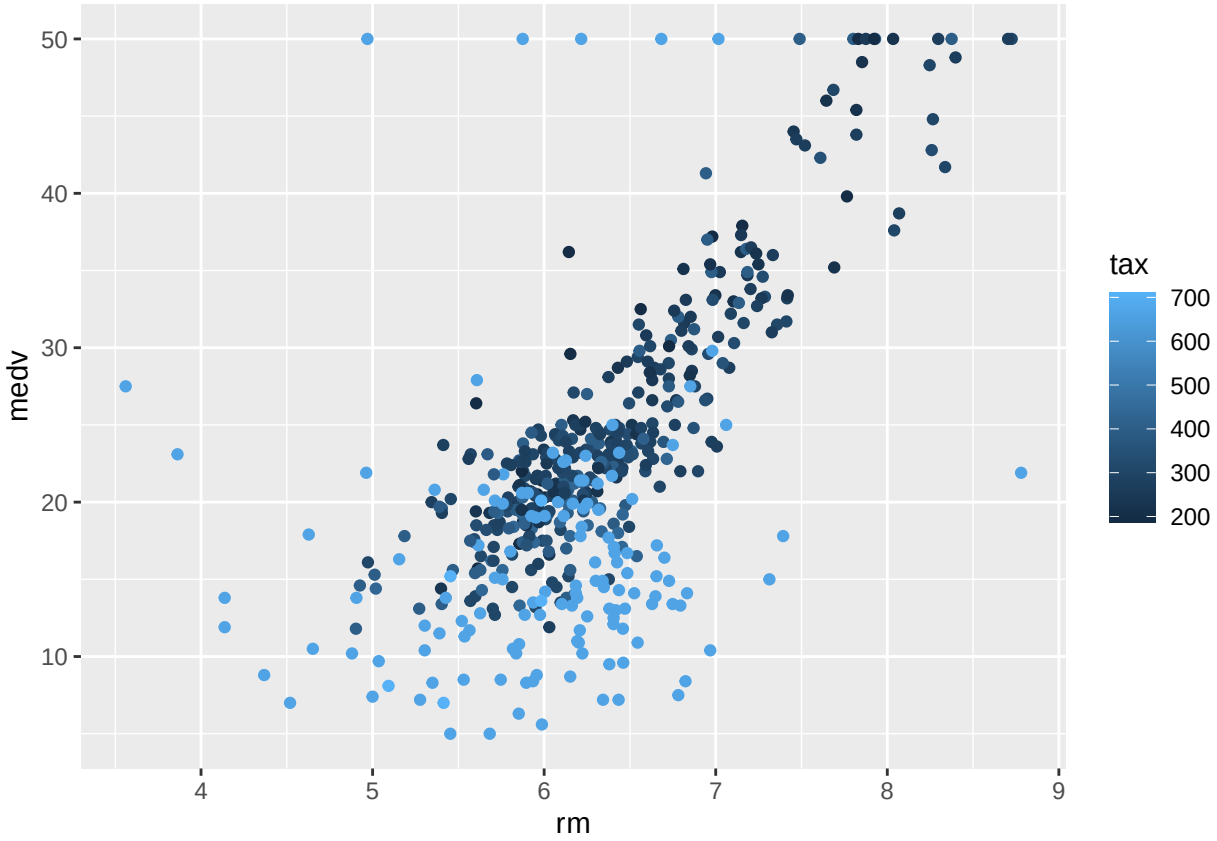
```
##       crim                 zn               indus             chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```
?MASS::Boston
```

#TO-DO

Using what you learned about the Boston Housing Data in the previous question, try to guess which features are interacting. Confirm using plots in `ggplot` that illustrate three (or more) features.

```
ggplot(Boston, aes(x= rm, y= medv)) + geom_point(aes(col=tax))
```



Once an interaction has been located, confirm the "non-linear linear" model with the interaction term does better than just the vanilla linear model by demonstrating a lower RMSE. In Econ 382 you would test this explicitly using a hypothesis test. We know in this class than increasing p yields alower RMSE. But the exercise is still a good one.

```
model_non_linear = lm(medv ~ rm * tax ,Boston)
model_linear = lm(medv ~ rm + tax,Boston)

summary(model_linear)$r.squared
```

```
## [1] 0.5605639
```

```
summary(model_linear)$sigma
```

```
## [1] 6.108867
```

```
#Does better
summary(model_non_linear)$r.squared
```

```
## [1] 0.6511582
```

```
summary(model_non_linear)$sigma
```

```
## [1] 5.448277
```

Repeat this procedure for another interaction with two different features (not used in the previous interaction you found) and verify.

```
colnames(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```
skim(Boston)
```

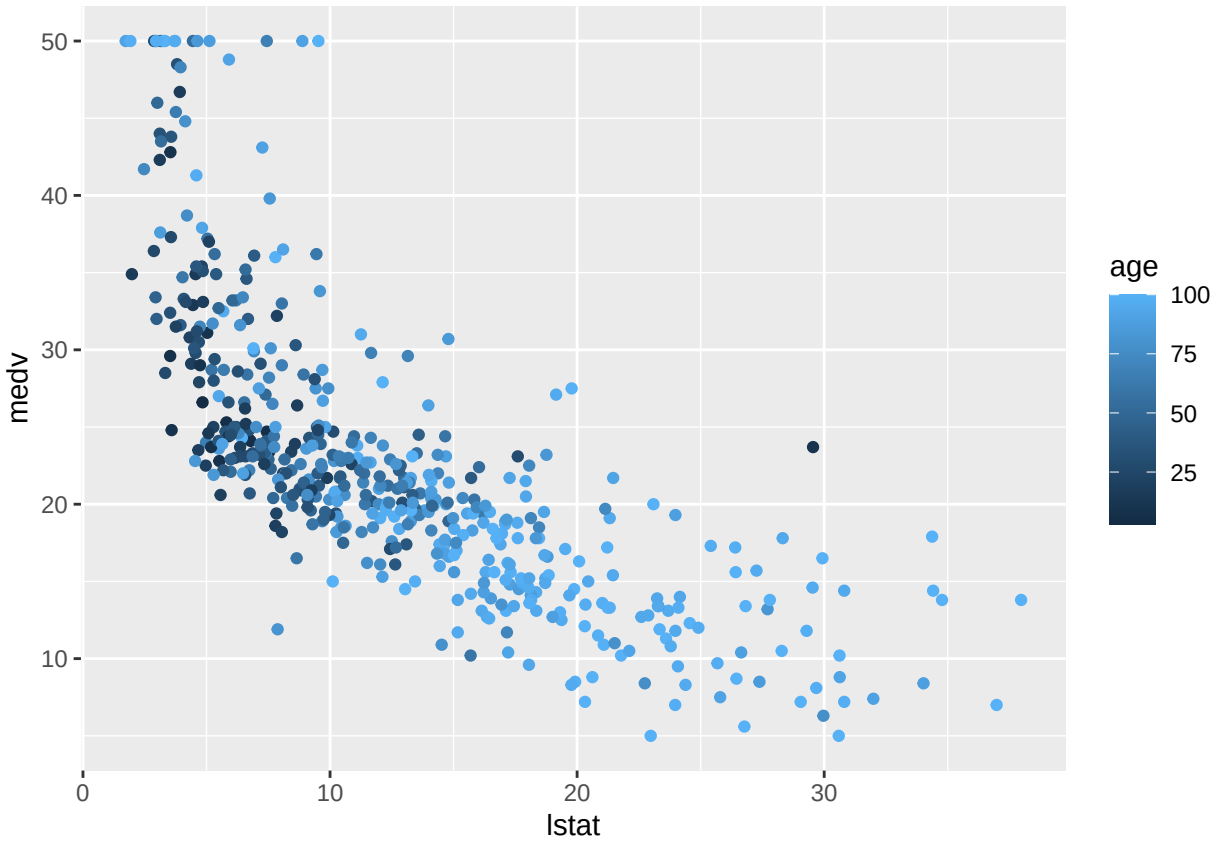
Table 1: Data summary

Name	Boston
Number of rows	506
Number of columns	14
Column type frequency:	
numeric	14
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
crim	0	1	3.61	8.60	0.01	0.08	0.26	3.68	88.98	
zn	0	1	11.36	23.32	0.00	0.00	0.00	12.50	100.00	
indus	0	1	11.14	6.86	0.46	5.19	9.69	18.10	27.74	
chas	0	1	0.07	0.25	0.00	0.00	0.00	0.00	1.00	
nox	0	1	0.55	0.12	0.38	0.45	0.54	0.62	0.87	
rm	0	1	6.28	0.70	3.56	5.89	6.21	6.62	8.78	
age	0	1	68.57	28.15	2.90	45.02	77.50	94.07	100.00	
dis	0	1	3.80	2.11	1.13	2.10	3.21	5.19	12.13	
rad	0	1	9.55	8.71	1.00	4.00	5.00	24.00	24.00	
tax	0	1	408.24	168.54	187.00	279.00	330.00	666.00	711.00	
ptratio	0	1	18.46	2.16	12.60	17.40	19.05	20.20	22.00	
black	0	1	356.67	91.29	0.32	375.38	391.44	396.23	396.90	
lstat	0	1	12.65	7.14	1.73	6.95	11.36	16.96	37.97	
medv	0	1	22.53	9.20	5.00	17.02	21.20	25.00	50.00	

```
ggplot(Boston, aes(x= lstat, y= medv)) + geom_point(aes(col=age))
```



Fit a model using all possible first-order interactions. Verify it is "better" than the linear model. Do you think you overfit? Why or why not?

```
#Better Model yes overfit because Im trying find all possible relationship/ interaction hence trying fi
first_order_inter = lm(medv ~ .*., data = Boston)
summary(first_order_inter)$r.squared
```

```
## [1] 0.9211876
```

```
summary(first_order_inter)$sigma
```

```
## [1] 2.851634
```

CV

Use 5-fold CV to estimate the generalization error of the model with all interactions.

```
pacman::p_load(caret)
#train method makes this easy from caret libary
cv_model = train(medv ~ .*., data = Boston, method = "lm", trControl = trainControl(method = "cv", numbe


summary(cv_model)$r.squared
```

```
## [1] 0.9211876
```

```
summary(cv_model)$sigma
```

```
## [1] 2.851634
```