Lab 8

Tziporah Horowitz

2AM April 26, 2020

Data Wrangling / Munging / Carpentry

Throughout this assignment you can use either `dplyr` or `data.table` to answer but not base R.

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.

```
pacman::p_load(dplyr, magrittr)
data("storms")
str(storms)
```

```
## tibble [10,010 x 13] (S3: tbl_df/tbl/data.frame)
##  $ name       : chr [1:10010] "Amy" "Amy" "Amy" "Amy" ...
##  $ year       : num [1:10010] 1975 1975 1975 1975 1975 ...
##  $ month      : num [1:10010] 6 6 6 6 6 6 6 6 6 6 ...
##  $ day        : int [1:10010] 27 27 27 27 28 28 28 28 29 29 ...
##  $ hour       : num [1:10010] 0 6 12 18 0 6 12 18 0 6 ...
##  $ lat        : num [1:10010] 27.5 28.5 29.5 30.5 31.5 32.4 33.3 34 34.4 34 ...
##  $ long       : num [1:10010] -79 -79 -79 -79 -78.8 -78.7 -78 -77 -75.8 -74.8 ...
##  $ status     : chr [1:10010] "tropical depression" "tropical depression" "tropical depression" "tro
##  $ category   : Ord.factor w/ 7 levels "-1"<"0"<"1"<"2"<..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ wind       : int [1:10010] 25 25 25 25 25 25 25 30 35 40 ...
##  $ pressure   : int [1:10010] 1013 1013 1013 1013 1012 1012 1011 1006 1004 1002 ...
##  $ ts_diameter: num [1:10010] NA NA NA NA NA NA NA NA NA NA ...
##  $ hu_diameter: num [1:10010] NA NA NA NA NA NA NA NA NA NA ...
```

```
summary(storms)
```

```
##      name                year          month             day        
##  Length:10010       Min.   :1975   Min.   : 1.000   Min.   : 1.00  
##  Class :character   1st Qu.:1990   1st Qu.: 8.000   1st Qu.: 8.00  
##  Mode  :character   Median :1999   Median : 9.000   Median :16.00  
##                     Mean   :1998   Mean   : 8.779   Mean   :15.86  
##                     3rd Qu.:2006   3rd Qu.: 9.000   3rd Qu.:24.00  
##                     Max.   :2015   Max.   :12.000   Max.   :31.00  
##                                                                    
##       hour             lat             long            status         
##  Min.   : 0.000   Min.   : 7.20   Min.   :-109.30   Length:10010      
##  1st Qu.: 6.000   1st Qu.:17.50   1st Qu.: -80.70   Class :character  
##  Median :12.000   Median :24.40   Median : -64.50   Mode  :character  
```

```
##  Mean   : 9.114   Mean   :24.76   Mean   : -64.23
##  3rd Qu.:18.000   3rd Qu.:31.30   3rd Qu.: -48.60
##  Max.   :23.000   Max.   :51.90   Max.   :  -6.00
##
##  category       wind           pressure       ts_diameter       hu_diameter
##  -1:2545   Min.   : 10.00   Min.   : 882.0   Min.   :   0.00   Min.   :  0.00
##  0 :4373   1st Qu.: 30.00   1st Qu.: 985.0   1st Qu.:  69.05   1st Qu.:  0.00
##  1 :1685   Median : 45.00   Median : 999.0   Median : 138.09   Median :  0.00
##  2 : 628   Mean   : 53.49   Mean   : 992.1   Mean   : 166.76   Mean   : 21.41
##  3 : 363   3rd Qu.: 65.00   3rd Qu.:1006.0   3rd Qu.: 241.66   3rd Qu.: 28.77
##  4 : 348   Max.   :160.00   Max.   :1022.0   Max.   :1001.18   Max.   :345.23
##  5 :  68                                     NA's   :6528      NA's   :6528
```

```
head(storms)
```

```
## # A tibble: 6 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 Amy    1975     6    27     0  27.5 -79   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>
```

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

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

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

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

```
storms %>%
  filter(year %in% 1970:1979)
```

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)
```

Create a new feature `wind_speed_per_unit_pressure`.

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

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

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

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

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

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, year) %>%
  summarise(max_wind_speed = max(wind)) %>%
  arrange(-max_wind_speed, year)
```

Find the strongest storm by wind speed per year.

```
storms %>%
  group_by(year) %>%
  summarize(name = first(name), max_wind = max(wind)) %>%
  arrange(-max_wind)
```

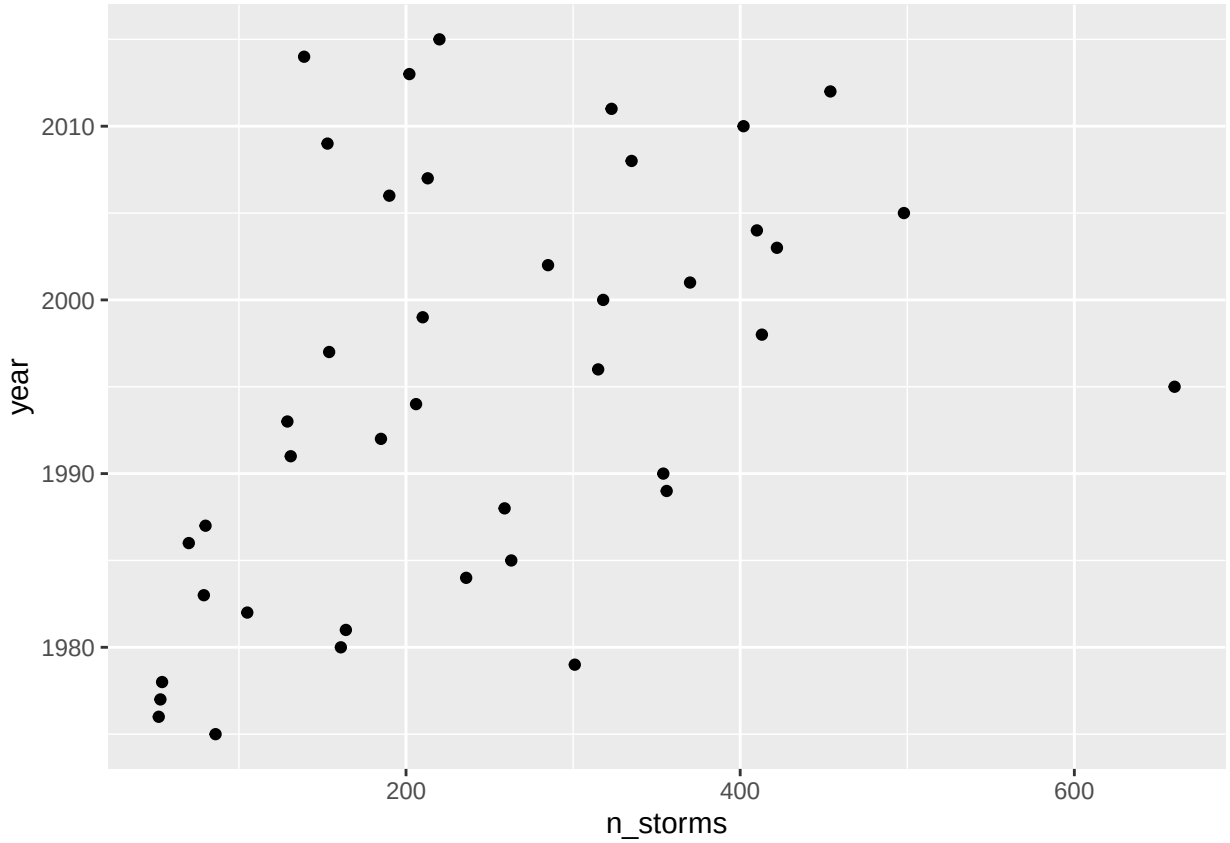
For each named storm, find its maximum category, wind speed, pressure and diameters. Do not allow the max to be NA (unless all the measurements for that storm were NA).

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

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?

```
pacman::p_load(ggplot2)

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



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

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

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

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

For each storm, summarize its average location in latitude / longitude coordinates.

```
storms %>%
  group_by(name) %>%
  summarise(avg_lat = mean(lat), avg_long = mean(long))
```

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

```
storms %>%
  group_by(name) %>%
  summarise(duration = ifelse(sum(hour) %% 6 == 0, sum(hour),
                              ifelse(sum(hour) %% 6 < 3, sum(hour) - (sum(hour) %% 6),
                                     sum(hour) + (6-(sum(hour) %% 6)))))
```

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

```
pacman::p_load(lubridate)

storms %<>%
  mutate(timestamp = ymd_h(paste(year, month, day, hour, sep = "-"))) %>%
  select(-year, -month, -day, -hour)
```

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

```
storms %<>%
  mutate(day_of_week = weekdays(timestamp),
         week_of_year = week(timestamp))
```

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

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

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

```
bins <- 0:10

storms %<>%
  mutate(decile_windspeed = factor(cut(wind, breaks = quantile(wind, bins/10), labels = FALSE)))
```

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

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

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

```
serious_storms %<>%
  mutate(lat_long = paste(lat, long, sep = " / ")) %>%
  select(-lat, -long)
```

For each category in storms, 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_diam = mean(ts_diameter, na.rm = TRUE),
            avg_hu_diam = mean(hu_diameter, na.rm = TRUE))
```

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

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

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

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

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

  # Haversine Formular for Distance
  a <- (sin(REdaS::deg2rad(lat2 - lat1) / 2)^2) +
    (cos(REdaS::deg2rad(lat2)) * cos(REdaS::deg2rad(lat1)) * (sin(REdaS::deg2rad(long2 - long1) / 2)^2))
  c <- 2 * atan2(sqrt(a), sqrt(1 - a))
  R <- 3958.8 # in mi
  d <- (R*c)

  return(d)
}

storms %<>%
  mutate(distance_to_miami = d_coords(lat, long, MIAMI_COORDS[1], MIAMI_COORDS[2]))
```

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

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

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

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

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

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

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

```
cor(as.numeric(storms$category), storms$speed)
```

```
## [1] 0.05758702
```

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.

```
storms_m <- storms %>%
  group_by(name) %>%
  summarise(y = max(wind),
            avg_pressure = mean(pressure),
            avg_distance = mean(dist_from_prev),
            final_status = last(status)) %>%
  select(-name)
```

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

```
mod <- lm(y ~ 0 + ., data = storms_m)
summary(mod)
```

```
##
## Call:
## lm(formula = y ~ 0 + ., data = storms_m)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.108 -14.362  -1.649  14.145  47.372
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## avg_pressure                    -2.80984    0.12907 -21.771   <2e-16 ***
## avg_distance                     0.13515    0.05501   2.457   0.0149 *
## final_statushurricane         2851.01985  127.23561  22.407   <2e-16 ***
```

```
## final_statustropical depression 2869.44988  129.35662  22.182   <2e-16 ***
## final_statustropical storm      2870.29345  128.42260  22.350   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.55 on 193 degrees of freedom
## Multiple R-squared:  0.9516, Adjusted R-squared:  0.9503
## F-statistic: 758.5 on 5 and 193 DF,  p-value: < 2.2e-16
```

```
n <- nrow(storms_m)
K <- 5
test_indices <- sample(1 : n, 1 / K * n)
train_indices <- setdiff(1 : n, test_indices)

X <- select(storms_m, -y)
y <- storms_m$y

X_train <- X[train_indices, ]
y_train <- y[train_indices]
X_test <- X[test_indices, ]
y_test <- y[test_indices]

modv <- lm(y_train ~ ., data.frame(X_train))
yhat_oos <- predict(mod, data.frame(X_test))
oos_residuals <- y_test - yhat_oos

sd(modv$residuals) - sd(oos_residuals)
```

```
## [1] -1.362131
```

```
head(cbind(y_test, yhat_oos))
```

```
##   y_test yhat_oos
## 1     60 58.12646
## 2     80 76.09101
## 3     85 61.36628
## 4     30 46.77496
## 5     45 64.92360
## 6     55 74.72554
```

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)
```

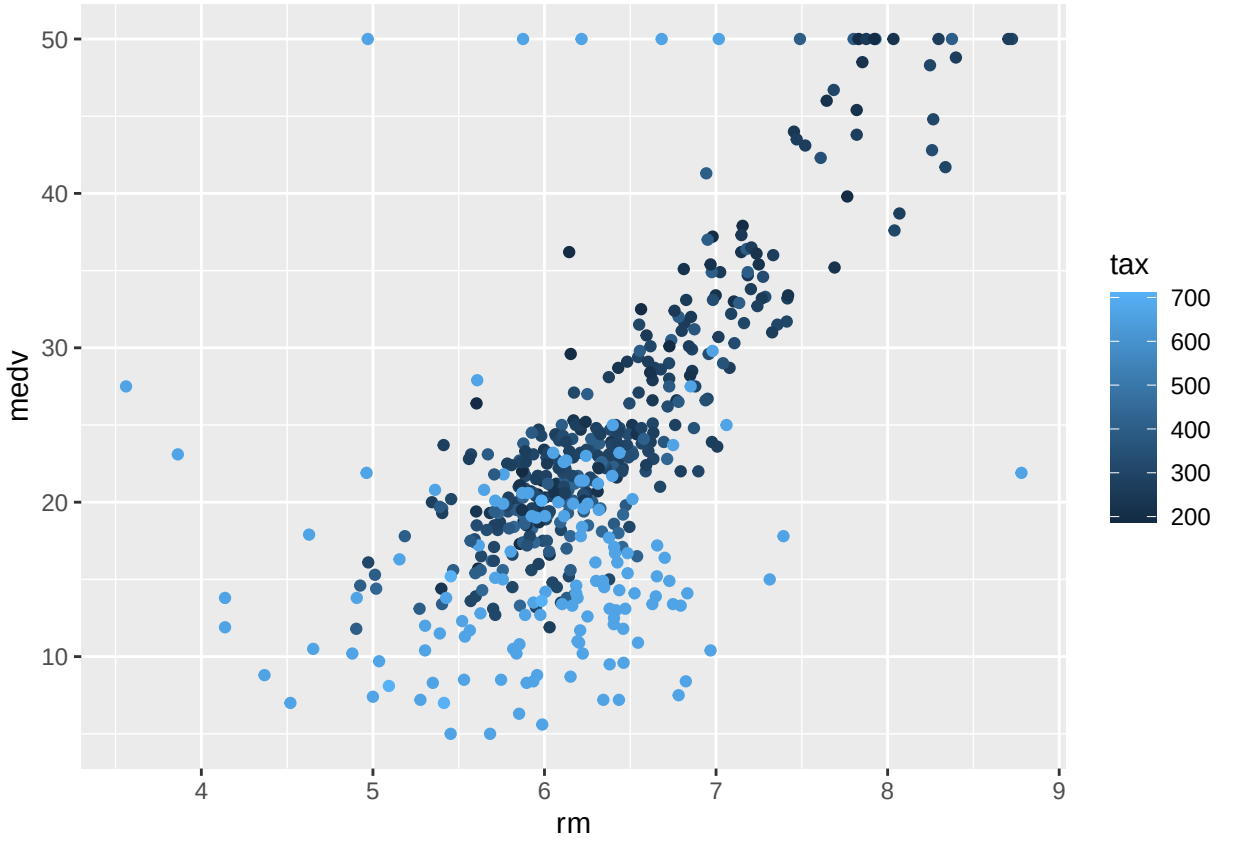
```
##       crim                zn              indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  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
```

```
## starting httpd help server ... done
```

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(data = Boston) +
  geom_point(aes(x = rm, y = medv, color = 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.

```
mod_linear <- lm(medv ~ ., data = Boston)
mod_interaction <- lm(medv ~ . + (rm * tax), data = Boston)

summary(mod_linear)$sigma
```

```
## [1] 4.745298
```

```
summary(mod_interaction)$sigma
```

```
## [1] 4.2299
```

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

```
mod_interaction2 <- lm(medv ~ . + (black * lstat), data = Boston)

summary(mod_linear)$sigma
```

```
## [1] 4.745298
```

```
summary(mod_interaction2)$sigma
```

```
## [1] 4.717432
```

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?

```
mod_all_interactions <- lm(medv ~ .*., data = Boston)

summary(mod_linear)$sigma
```

```
## [1] 4.745298
```

```
summary(mod_all_interactions)$sigma
```

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

This model will overfit because it increases the complexity unnecessarily.

CV

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

```
pacman::p_load(caret)
cv <- trainControl(method = "cv", number = 5)
mod <- train(medv ~ .*., data = Boston, method = "lm",
               trControl = cv)
print(mod)
```

```
## Linear Regression
##
## 506 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 406, 405, 404, 404, 405
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   3.323402  0.8732214  2.419459
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```