Lab 9

Adriana Sham

11:59PM April 14, 2019

"data wrangling / munging / carpentry" with dplyr.

First load `dplyr`, `tidyr`, `magrittr` and `lubridate` in one line.

```
pacman::p_load(dplyr, tidyr, magrittr, lubridate)
```

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 using the `mutate` and the overwrite pipe operator `%<>%`. Verify.

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

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10010 obs. of  13 variables:
##  $ name       : chr  "Amy" "Amy" "Amy" "Amy" ...
##  $ year       : num  1975 1975 1975 1975 1975 ...
##  $ month      : num  6 6 6 6 6 6 6 6 6 6 ...
##  $ day        : int  27 27 27 27 28 28 28 28 29 29 ...
##  $ hour       : num  0 6 12 18 0 6 12 18 0 6 ...
##  $ lat        : num  27.5 28.5 29.5 30.5 31.5 32.4 33.3 34 34.4 34 ...
##  $ long       : num  -79 -79 -79 -79 -78.8 -78.7 -78 -77 -75.8 -74.8 ...
##  $ status     : chr  "tropical depression" "tropical depression" "tropical depression" "tropical depr
##  $ category   : Ord.factor w/ 7 levels "-1"<"0"<"1"<"2"<..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ wind       : int  25 25 25 25 25 25 25 30 35 40 ...
##  $ pressure   : int  1013 1013 1013 1013 1012 1012 1011 1006 1004 1002 ...
##  $ ts_diameter: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hu_diameter: num  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      
##  -1:2545   Min.   : 10.00   Min.   : 882.0   Min.   :   0.00  
```

```
##  0 :4373   1st Qu.: 30.00   1st Qu.: 985.0   1st Qu.:  69.05
##  1 :1685   Median : 45.00   Median : 999.0   Median : 138.09
##  2 : 628   Mean   : 53.49   Mean   : 992.1   Mean   : 166.76
##  3 : 363   3rd Qu.: 65.00   3rd Qu.:1006.0   3rd Qu.: 241.66
##  4 : 348   Max.   :160.00   Max.   :1022.0   Max.   :1001.18
##  5 :  68                                     NA's   :6528
##   hu_diameter
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 21.41
##  3rd Qu.: 28.77
##  Max.   :345.23
##  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))
str(storms)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10010 obs. of  13 variables:
##  $ name       : Factor w/ 198 levels "AL011993","AL012000",..: 44 44 44 44 44 44 44 44 44 44 ...
##  $ year       : num  1975 1975 1975 1975 1975 ...
##  $ month      : num  6 6 6 6 6 6 6 6 6 6 ...
##  $ day        : int  27 27 27 27 28 28 28 28 29 29 ...
##  $ hour       : num  0 6 12 18 0 6 12 18 0 6 ...
##  $ lat        : num  27.5 28.5 29.5 30.5 31.5 32.4 33.3 34 34.4 34 ...
##  $ long       : num  -79 -79 -79 -79 -78.8 -78.7 -78 -77 -75.8 -74.8 ...
##  $ status     : Factor w/ 3 levels "hurricane","tropical depression",..: 2 2 2 2 2 2 2 2 3 3 ...
##  $ category   : Ord.factor w/ 7 levels "-1"<"0"<"1"<"2"<..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ wind       : int  25 25 25 25 25 25 25 30 35 40 ...
##  $ pressure   : int  1013 1013 1013 1013 1012 1012 1011 1006 1004 1002 ...
##  $ ts_diameter: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hu_diameter: num  NA NA NA NA NA NA NA NA NA NA ...
```

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

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

Sort the dataframe by year (most recent first) then category of the storm (most severe first). Verify.

```
storms %<>%
  arrange(desc(year), desc(category))
```

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 diameters.

```
storms %<>%
  mutate(average_diameter = (ts_diameter + hu_diameter) / 2)
```

Calculate the distance from each storm observation to Miami in a new variable `distance_to_miami`.

```
MIAMI_COORDS = c(25.7617, -80.1918)
RAD_EARTH = 3958.8
degrees_to_radians = function(angle_degrees){
  for(i in 1:length(angle_degrees))
    angle_degrees[i] = angle_degrees[i]*pi/180
  return(angle_degrees)
}
compute_globe_distance = function(destination, origin){
  destination_rad = degrees_to_radians(destination)
  origin_rad = degrees_to_radians(origin)
  delta_lat = destination_rad[1] - origin_rad[1]
  delta_lon = destination_rad[2] - origin_rad[2]
  h = (sin(delta_lat/2))^2 + cos(origin_rad[1]) * cos(destination_rad[1]) * (sin(delta_lon/2))^2
  central_angle = 2 * asin(sqrt(h))
  return(RAD_EARTH * central_angle)
}
storms %<>%
  rowwise() %>%
  mutate(distance_to_miami = compute_globe_distance(MIAMI_COORDS, c(lat, long))) %>%
  select(lat, long, distance_to_miami, everything())
```

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

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

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)) %<>%
  mutate(week_of_the_year = week(timestamp))
```

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 = " / ")
```

Back to the main dataframe `storms`, create a new feature `decile_windspeed` by binning wind speed into 10 bins.

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

Let's summarize some data. Find the strongest storm by wind speed per year.

```
storms %>%
  separate(timestamp, c("year", "month","day", "hour"), sep = " - ") %>%
  group_by(year) %>%
  summarize(max_wind_speed = max(wind))
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 10010
## rows [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20, ...].
```

```
## # A tibble: 7,958 x 2
##    year                max_wind_speed
##    <chr>                        <dbl>
##  1 1975-06-27 00:00:00             25
##  2 1975-06-27 06:00:00             25
##  3 1975-06-27 12:00:00             25
##  4 1975-06-27 18:00:00             25
##  5 1975-06-28 00:00:00             25
##  6 1975-06-28 06:00:00             25
##  7 1975-06-28 12:00:00             25
##  8 1975-06-28 18:00:00             30
##  9 1975-06-29 00:00:00             35
## 10 1975-06-29 06:00:00             40
## # ... with 7,948 more rows
```

For each status, find the average category, wind speed, pressure and diameters (do not allow the average to be NA).

```
storms %>%
  group_by(status) %>%
  summarise(avg_category = mean(as.numeric(as.character(category))), avg_wind_speed = mean(wind), avg_p
```

```
## # A tibble: 3 x 6
##   status avg_category avg_wind_speed avg_pressure avg_ts_diameter
##   <fct>         <dbl>          <dbl>        <dbl>           <dbl>
## 1 hurri~     1.86               86.0         969.            288.
## 2 tropi~    -1                  27.3        1008.              0
## 3 tropi~     0.000229           45.8         999.            160.
## # ... with 1 more variable: avg_hu_diameter <dbl>
```

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

```
storms %>%
  group_by(name) %>%
  summarize(max_category = max(category), max_wind_speed = max(wind), max_pressure = max(pressure), max
```

```
## # A tibble: 198 x 5
##    name     max_category max_wind_speed max_pressure max_diameter
##    <fct>    <ord>                 <dbl>        <dbl>        <dbl>
##  1 AL011993 -1                       30         1003           NA
##  2 AL012000 -1                       25         1010           NA
##  3 AL021992 -1                       30         1009           NA
```

```
##  4 AL021994 -1                       30         1017          NA
##  5 AL021999 -1                       30         1006          NA
##  6 AL022000 -1                       30         1010          NA
##  7 AL022001 -1                       25         1012          NA
##  8 AL022003 -1                       30         1010          NA
##  9 AL022006 0                        45         1008           0
## 10 AL031987 0                        40         1015          NA
## # ... with 188 more rows
```

For each category, find its average wind speed, pressure and diameters (do not allow the max to be NA).

```
storms %<>%
  group_by(category, average_wind_speed = wind / category)
```

```
## Warning in Ops.ordered(wind, category): '/' is not meaningful for ordered
## factors
```

for each named storm, find its duration in hours.

```
storms %>%
  group_by(name) %>%
  mutate(duration_in_hours = 6*n())
```

```
## # A tibble: 10,010 x 18
## # Groups:   name [198]
##      lat  long distance_to_mia~ name  status category timestamp
##    <dbl> <dbl>            <dbl> <fct> <fct>  <ord>    <dttm>
##  1  23.1 -73.7             448. Joaq~ hurri~ 4        2015-10-01 12:00:00
##  2  23   -74.2             423. Joaq~ hurri~ 4        2015-10-01 18:00:00
##  3  22.9 -74.4             415. Joaq~ hurri~ 4        2015-10-02 00:00:00
##  4  23   -74.7             395. Joaq~ hurri~ 4        2015-10-02 06:00:00
##  5  23.4 -74.8             376. Joaq~ hurri~ 4        2015-10-02 12:00:00
##  6  24.3 -74.3             382. Joaq~ hurri~ 4        2015-10-03 00:00:00
##  7  24.8 -73.6             417. Joaq~ hurri~ 4        2015-10-03 06:00:00
##  8  25.4 -72.6             474. Joaq~ hurri~ 4        2015-10-03 12:00:00
##  9  26.3 -71               572. Joaq~ hurri~ 4        2015-10-03 18:00:00
## 10  27.4 -69.5             670. Joaq~ hurri~ 4        2015-10-04 00:00:00
## # ... with 10,000 more rows, and 11 more variables: wind <int>,
## #   pressure <int>, ts_diameter <dbl>, hu_diameter <dbl>,
## #   wind_speed_per_unit_pressure <dbl>, average_diameter <dbl>,
## #   day_of_week <ord>, week_of_the_year <dbl>, decile_windspeed <fct>,
## #   average_wind_speed <lgl>, duration_in_hours <dbl>
```

For each named storm, find the distance from its starting position to ending position in kilometers.

```
storms %>%
  group_by(name) %>%
  arrange(desc(timestamp)) %>%
  summarize(distance_from_start = 1.61 * compute_globe_distance( c(last(lat), last(long)) , c(first(lat
```

```
## # A tibble: 198 x 2
##    name     distance_from_start
##    <fct>                  <dbl>
##  1 AL011993              1417.
##  2 AL012000                56.5
##  3 AL021992               515.
##  4 AL021994               386.
##  5 AL021999               241.
```

```
##  6 AL022000              2018.
##  7 AL022001               738.
##  8 AL022003               560.
##  9 AL022006               733.
## 10 AL031987              1257.
## # ... with 188 more rows
```

Now we want to transition to building real design matrices for prediction. We want 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 Dana Chandler spoke about) 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.

```
y = storms %>%
  group_by(name) %>%
  summarize(max_wind_speed = max(wind))
y = y %>%
  arrange(desc(name))
X = storms %>%
  group_by(name) %>%
  arrange(desc(timestamp)) %>%
  filter(timestamp <= nth(timestamp, n()-2)) %>%
  summarize(ave_pressure = mean(pressure), ave_category = mean(as.numeric(as.character(category))),
   distance_from_start = compute_globe_distance( c(last(lat),last(long)) , c(first(lat),first(long)) ),
    ave_ts_diameter = mean(ts_diameter, na.rm = TRUE), ave_hu_diameter = mean(hu_diameter, na.rm = TRUE)
    pressure_by_ts_diameter = ave_pressure * ave_ts_diameter, pressure_by_hu_diameter = ave_pressure * a
    category_by_ts_diameter = ave_category * ave_ts_diameter, category_by_hu_diameter = ave_category * a
#Arrange by descending time to get the three earliest observations.
#Take average pressure, average category, how far the storms traveled in 18 hours.
#I included interactions with the diameters because sometimes they were zero.
edge_case = storms %>%
  group_by(name) %>%
  mutate (observations = n()) %>%
  filter(observations < 3) %>%
  summarize(ave_pressure = mean(pressure), ave_category = mean(as.numeric(as.character(category))),
    distance_from_start = compute_globe_distance( c(last(lat),last(long)) , c(first(lat),first(long)) )
    ave_ts_diameter = mean(ts_diameter, na.rm = TRUE), ave_hu_diameter = mean(hu_diameter, na.rm = TRUE)
    pressure_by_ts_diameter = ave_pressure * ave_ts_diameter, pressure_by_hu_diameter = ave_pressure * a
    category_by_ts_diameter = ave_category * ave_ts_diameter, category_by_hu_diameter = ave_category * a
X = rbind.data.frame(X, edge_case)
X = X %>%
  arrange(desc(name))
#We check for an edge case, where a storm does not have three observations.
#Then append it to our design matrix then also reorder the names.
y = y %>%
  select(-name)
X = X %>%
  select(-name)
mod = lm(as.matrix(y) ~ as.matrix(X))
summary(mod)$r.squared
```

```
## [1] 0.1113474
```

```
summary(mod)$sigma
```

```
## [1] 35.66259
```

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 ...
```

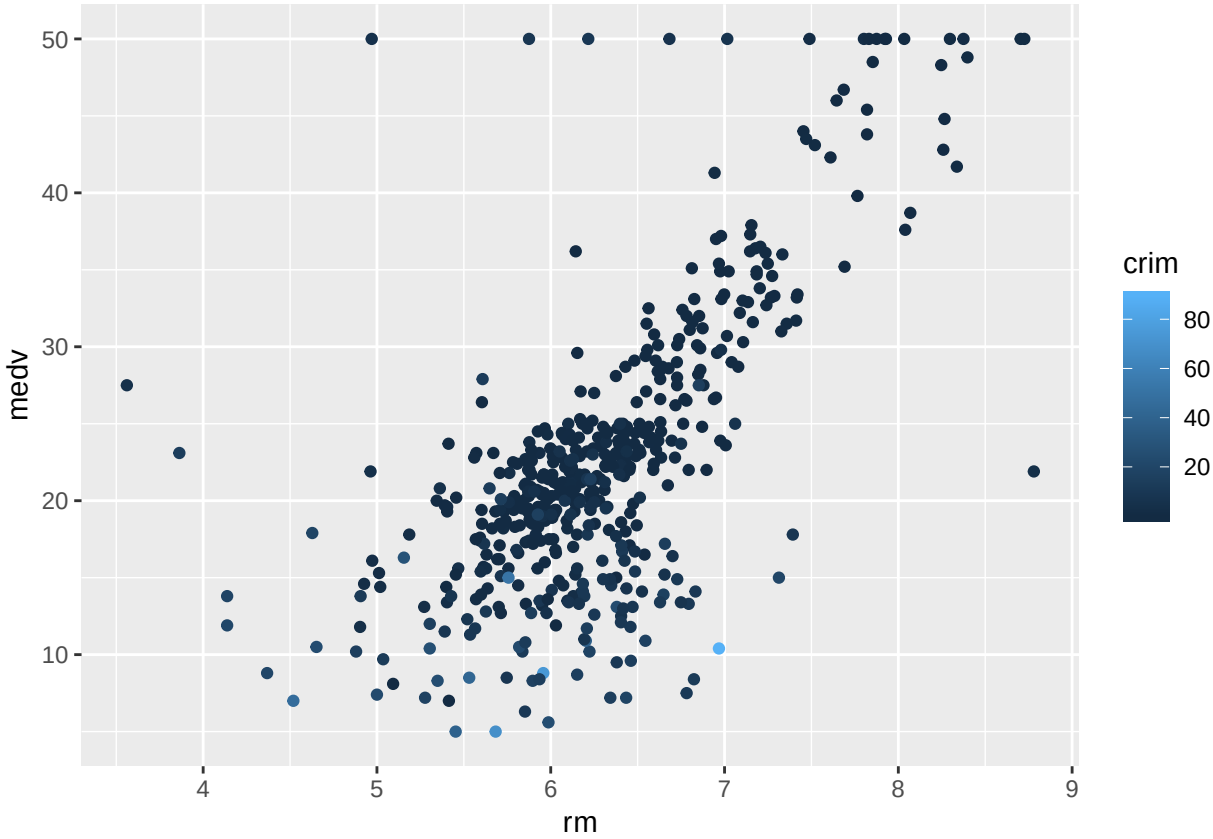
```
summary(Boston)
```

```
##       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
```

Using your knowledge of the modeling problem, try to guess which features are interacting. Confirm using

plots in ggplot that illustrate three (or more) features.

```
pacman::p_load(ggplot2)
base = ggplot(Boston, aes(x = rm, y = medv))
base + geom_point(aes(col = crim))
```



Once an interaction has been located, confirm the "non-linear linear" model with the interaction term does better than just the vanilla linear model.

```
mod = lm(medv ~ rm * crim, Boston)
coef(mod)
```

```
## (Intercept)          rm        crim     rm:crim
##  -37.257338    9.651470    1.462943   -0.287657
```

```
mod_vanilla = lm(medv ~ rm + crim, Boston)
summary(mod_vanilla)$r.squared
```

```
## [1] 0.5419592
```

```
summary(mod_vanilla)$sigma
```

```
## [1] 6.236844
```

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

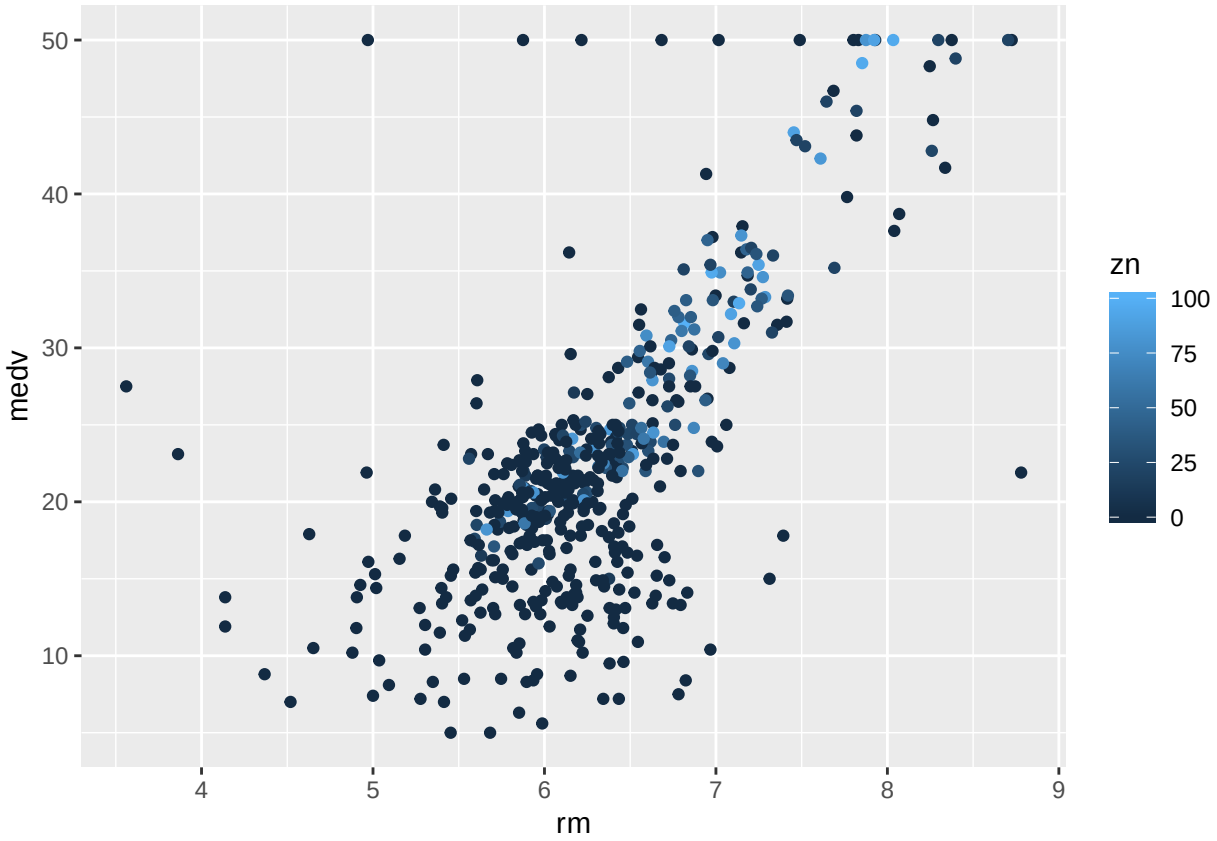
```
## [1] 0.5814763
```

```
summary(mod)$sigma
```

```
## [1] 5.967672
```

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

```
base + geom_point(aes(col = zn))
```




```
mod = lm(medv ~ rm * zn, Boston)
coef(mod)
```

```
## (Intercept)          rm          zn       rm:zn
## -26.9934476   7.7661501  -0.4697937   0.0791624
```

```
mod_vanilla = lm(medv ~ rm + zn, Boston)
summary(mod_vanilla)$r.squared
```

```
## [1] 0.5063381
```

```
summary(mod_vanilla)$sigma
```

```
## [1] 6.474818
```

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

```
## [1] 0.5223732
```

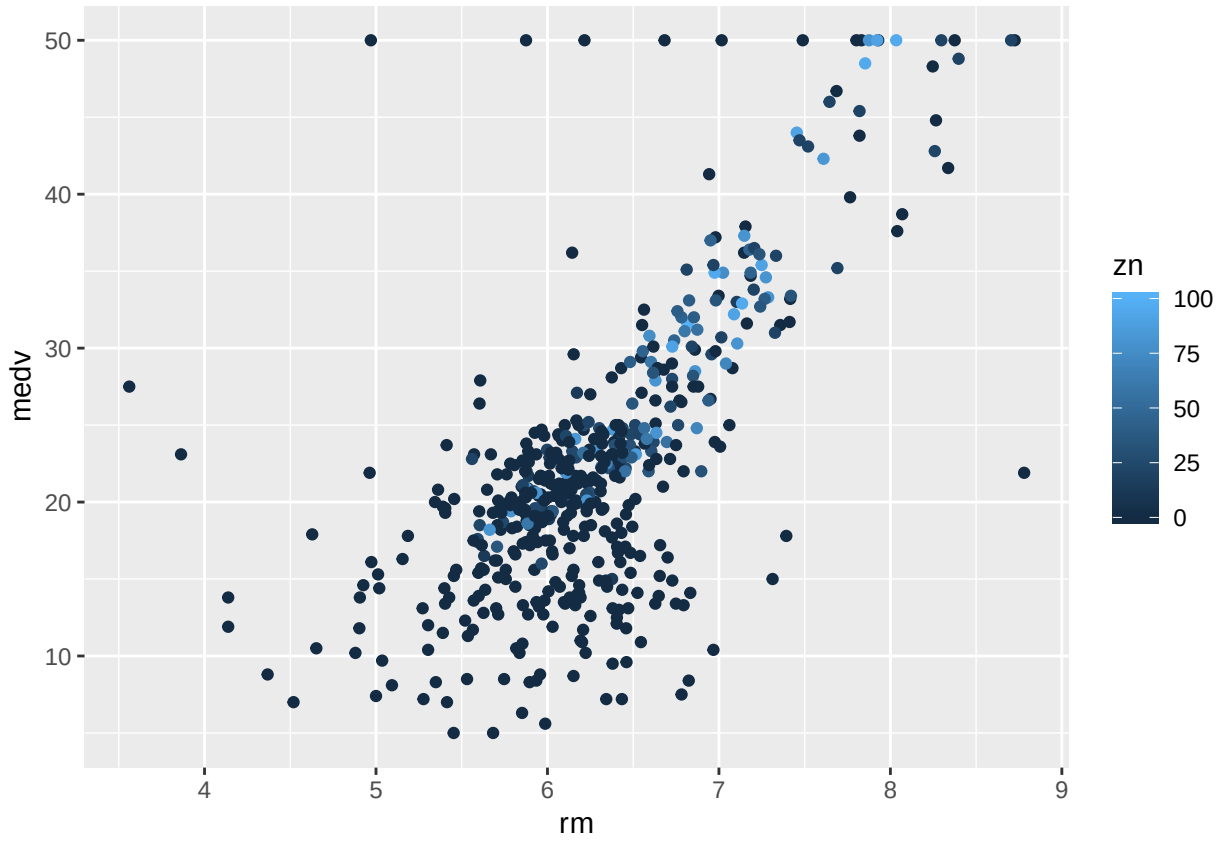
```
summary(mod)$sigma
```

```
## [1] 6.375133
```

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?

No, we did not overfit. The number of rows in the Boston housing data is 506 and we only used 91 features.

```
base + geom_point(aes(col = zn))
```




```
mod = lm(medv ~  (.)^2 , Boston)
mod_vanilla = lm(medv ~ rm + zn, Boston)
summary(mod_vanilla)$r.squared
```

```
## [1] 0.5063381
```

```
summary(mod_vanilla)$sigma
```

```
## [1] 6.474818
```

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

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

```
summary(mod)$sigma
```

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

CV

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

```
pacman::p_load(mlr)
modeling_task = makeRegrTask(data = Boston, target = "medv") #instantiate the task
algorithm = makeLearner("regr.lm") #instantiate the OLS learner algorithm on the diamonds dataset and s
```

```r
validation = makeResampleDesc("CV", iters = 5) #instantiate the 5-fold CV
resample(algorithm, modeling_task, validation)
```

```
## Resampling: cross-validation
## Measures:             mse
## [Resample] iter 1:    24.2821581
## [Resample] iter 2:    22.5152578
## [Resample] iter 3:    16.1875142
## [Resample] iter 4:    26.7891792
## [Resample] iter 5:    27.1957323
##
## Aggregated Result: mse.test.mean=23.3939683
##
## Resample Result
## Task: Boston
## Learner: regr.lm
## Aggr perf: mse.test.mean=23.3939683
## Runtime: 0.0580711
```