Lab 8

Loyd Flores

#Model Selection with Three Splits: Select from M models

We employ the diamonds dataset and specify M models nested from simple to more complex. We store the models as strings in a list (i.e. a hashset)

```
?ggplot2::diamonds
```

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

```
model_formulas = c(
  "carat",
  "carat + cut",
  "carat + cut + color",
  "carat + cut + color + clarity",
  "carat + cut + color + clarity + x + y + z",
  "carat + cut + color + clarity + x + y + z + depth",
  "carat + cut + color + clarity + x + y + z + depth + table",
  "carat * (cut + color + clarity) + x + y + z + depth + table",
  "(carat + x + y + z) * (cut + color + clarity) + depth + table",
  "(carat + x + y + z + depth + table) * (cut + color + clarity)",
  "(poly(carat, 2) + x + y + z + depth + table) * (cut + color + clarity)",
  "(poly(carat, 2) + poly(x, 2) + poly(y, 2) + poly(z, 2) + depth + table) * (cut + color + clarity)",
  "(poly(carat, 2) + poly(x, 2) + poly(y, 2) + poly(z, 2) + poly(depth, 2) + poly(table, 2)) * (cut + c
  "(poly(carat, 2) + poly(x, 2) + poly(y, 2) + poly(z, 2) + poly(depth, 2) + poly(table, 2) + log(carat
  "(poly(carat, 2) + poly(x, 2) + poly(y, 2) + poly(z, 2) + poly(depth, 2) + poly(table, 2) + log(carat
  "(poly(carat, 2) + poly(x, 2) + poly(y, 2) + poly(z, 2) + poly(depth, 2) + poly(table, 2) + log(carat
  "(poly(carat, 2) + poly(x, 2) + poly(y, 2) + poly(z, 2) + poly(depth, 2) + poly(table, 2) + log(carat
)
model_formulas = paste0("price ~ ", model_formulas)
M = length(model_formulas)
```

In order to use the formulas with logs we need to eliminate rows with zeros in those measurements:

```
diamonds_cleaned = ggplot2::diamonds
diamonds_cleaned = diamonds_cleaned[
  diamonds_cleaned$carat > 0 &
  diamonds_cleaned$x > 0 &
  diamonds_cleaned$y > 0 &
  diamonds_cleaned$z > 0 &
  diamonds_cleaned$depth > 0 &
  diamonds_cleaned$table > 0, #all columns
]
```

Split the data into train, select and test. Each set should have 1/3 of the total data.

```
n = nrow(diamonds_cleaned)
set.seed(1)
train_idx = sample(1 : n, round(n / 3))
select_idx = sample(setdiff(1 : n, train_idx), round(n / 3))
test_idx = setdiff(1 : n, c(train_idx, select_idx))
diamonds_train =  diamonds_cleaned[train_idx, ]
diamonds_select = diamonds_cleaned[select_idx, ]
diamonds_test =   diamonds_cleaned[test_idx, ]
```

Find the oosRMSE on the select set for each model. Save the number of df in each model while you're doing this as we'll need it for later.

```
#TO-DO
dfs = array(NA, M)
oosRMSEs = array(NA, M)
for(m in 1 : M){
  mod = lm(model_formulas[m], data = diamonds_train)
  dfs[m] = mod$rank
  y_hat = predict(mod, diamonds_select)
  oosRMSEs[m] = sqrt(mean((diamonds_select$price - y_hat)^2))

}
```

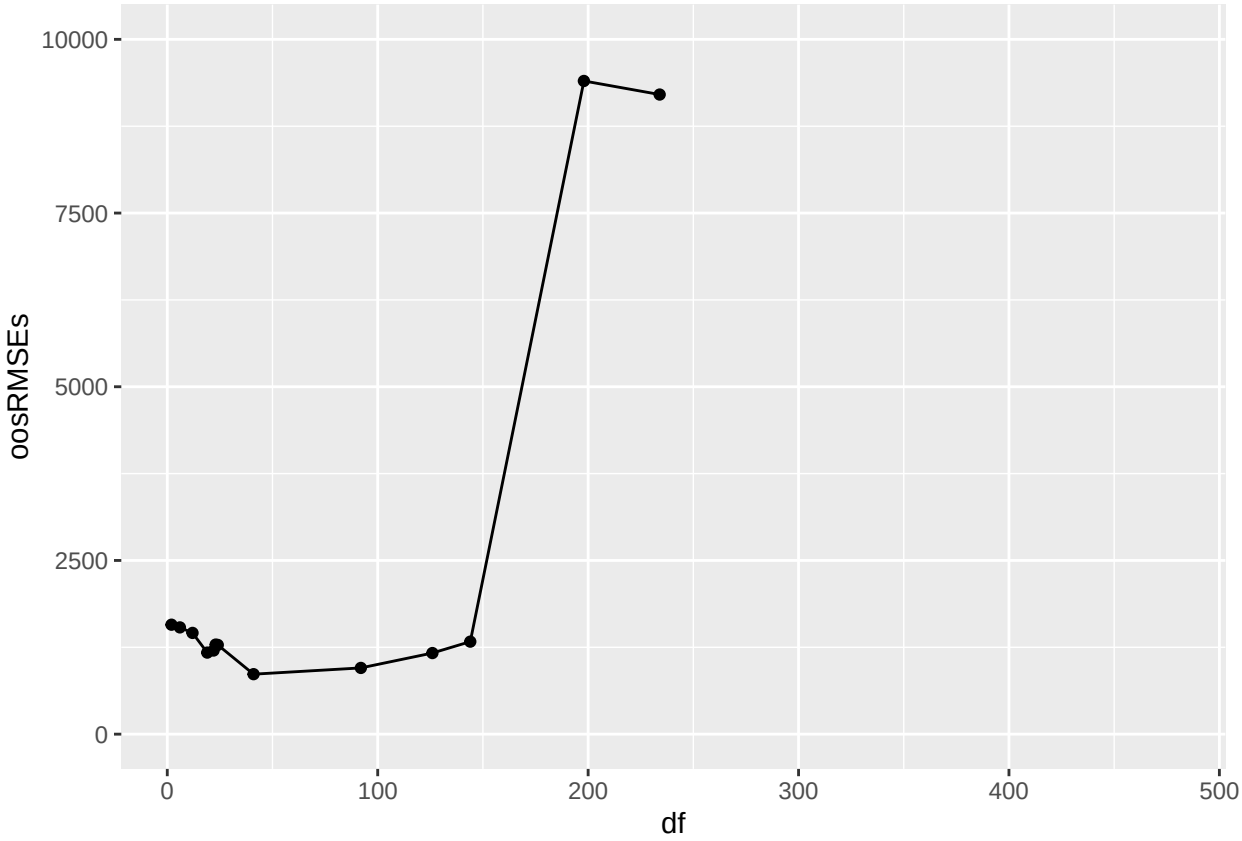
```
## Warning in predict.lm(mod, diamonds_select): prediction from rank-deficient
## fit; attr(*, "non-estim") has doubtful cases
```

Plot the oosRMSE by model complexity (df in model)

```
pacman::p_load(ggplot2)
#TO-DO
ggplot(data.frame(df=dfs, oosRMSE = oosRMSEs)) +
  aes(x = df, y = oosRMSEs) +
  geom_line() +
  geom_point() +
  ylim(0, 1e4)
```

```
## Warning: Removed 4 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



Select the best model by oosRMSE and find its oosRMSE on the test set.

```
#TO-DO
m_star = which.min(oosRMSEs)

mod_star = lm(model_formulas[m_star], data = rbind(diamonds_train, diamonds_select)) # Train the best m
y_hat = predict(mod_star, diamonds_test)
sqrt(mean((diamonds_test$price - y_hat)^2))
```

```
## [1] 776.811
```

Did we overfit the select set? Discuss why or why not.

No because the RMSE on the select is not less than the select on the test

```
oosRMSEs[m_star]
```

```
## [1] 863.366
```

```
mod_star = lm(model_formulas[m_star], data = diamonds_train )
y_hat = predict(mod_star, diamonds_test)
sqrt(mean((diamonds_test$price - y_hat)^2))
```

```
## [1] 779.3248
```

Create the final model object `g_final`.

```
#TO-DO
mod_star = lm(model_formulas[m_star], diamonds_cleaned)
g_final = function(x_star){
  predict(mod_star, x_star)
}
```

#Model Selection with Three Splits: Hyperparameter selection

We will use an algorithm that I historically taught in 324W but now moved to 343 so I can teach it more deeply using the Bayesian topics from 341. The regression algorithm is called "ridge" and it involves solving for the slope vector via:

b_ridge := (X^T X + lambda I_(p+1))^-1 X^T y

Note how if lambda = 0, this is the same algorithm as OLS. If lambda becomes very large then b_ridge is pushed towards all zeroes. So ridge is good at weighting only features that matter.

However, lambda is a hyperparameter >= 0 that needs to be selected.

We will work with the boston housing dataset except we will add 250 garbage features consisting of iid N(0,1) realizations. We will also standardize the columns so they're all xbar = 0 and s_x = 1. This is shown to be important in 343.

```
rm(list = ls())
?MASS::Boston
y = MASS::Boston$medv
X = model.matrix(medv ~ ., MASS::Boston)
n = nrow(X)
p_garbage = 250
set.seed(1)
X = cbind(X, matrix(rnorm(n * p_garbage), nrow = n))
X = apply(X, 2, function(x_dot_j){
                  (x_dot_j - mean(x_dot_j)) / sd(x_dot_j)
                })
X[, 1] = 1
dim(X)
```

```
## [1] 506 264
```

Now we split it into 300 train, 100 select and 106 test.

```
set.seed(1)
train_idx = sample(1 : n, 300)
select_idx = sample(setdiff(1 : n, train_idx), 100)
test_idx = setdiff(1 : n, c(train_idx, select_idx))
#TO-DO : train test split

# Train
X_train = X[train_idx,]
y_train = y[train_idx]
# Select
X_select = X[select_idx,]
y_select = y[select_idx]
```

```
# Test
X_test = X[test_idx,]
y_test = y[test_idx]
```

We now create a grid of M = 200 models indexed by lambda. The lowest lambda should be zero (which is OLS) and the highest lambda can be 100.

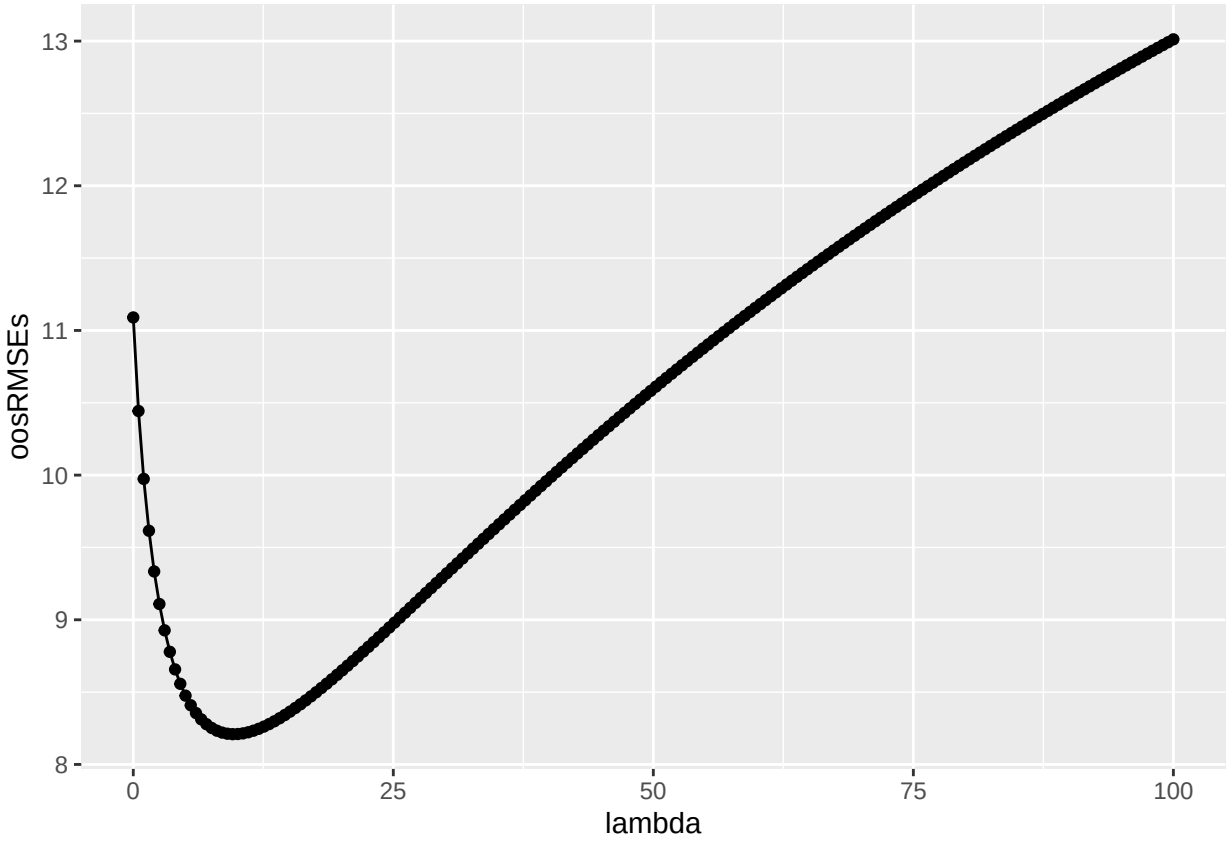
```
M = 200
lambda_grid = seq(from = 0, to = 100, length.out = M)
XtX = t(X_train) %*% X_train
Xty = t(X_train) %*% y_train
I_p_plus_one = diag(ncol(X_train))
oosRMSEs = array(NA, M)
for(m in 1 : M){
  b_ridge = solve(XtX + lambda_grid[m] * I_p_plus_one) %*% Xty # FIT ON X_train
  y_hat = X_select %*% b_ridge # Predict on X_select
  oosRMSEs[m] = sqrt(mean((y_select - y_hat)^2))
}
```

Now find the oosRMSE on the select set on all models each with their own lambda value.

```
#TO-DO
oosRMSEs = array(NA, M)
XtX = t(X_train)%*%X_train
Xty = t(X_train)%*%y_train
I_p_plus_one = diag(ncol(X_train))
for (m in 1:M) {
  b_ridge = solve(XtX + lambda_grid[m] * I_p_plus_one) %*% Xty
  yhat = X_select %*% b_ridge
  oosRMSEs[m] = sqrt(mean((y_select - yhat)^2))
}
```

Plot the oosRMSE by the value of lambda.

```
#TO-DO
ggplot(data.frame(lambda = lambda_grid, oosRMSE = oosRMSEs)) +
  aes(x = lambda, y = oosRMSEs) +
  geom_line() +
  geom_point()
```



Select the model with the best oosRMSE on the select set and find its oosRMSE on the test set.

```
#TO-DO

# Identify the lambda value corresponding to the minimum oosRMSE
best_lambda = lambda_grid[which.min(oosRMSEs)]

# Train the final model
final_model = solve(XtX + best_lambda * I_p_plus_one) %*% Xty

# Make predictions
yhat_test = X_test %*% final_model

# Calculate oosRMSE on the test set
oosRMSE_test = sqrt(mean((y_test - yhat_test)^2))

# Print the oosRMSE on the test set
print(paste("oosRMSE on test set:", oosRMSE_test))
```

```
## [1] "oosRMSE on test set: 7.31499748805904"
```

Create the final model object `g_final`.

```
#TO-DO
g_final = final_model
```

#Model Selection with Three Splits: Forward stepwise modeling

We will use the adult data

```
rm(list = ls())
pacman::p_load_gh("coatless/ucidata") #load from github
data(adult)
adult = na.omit(adult) #remove any observations with missingness
n = nrow(adult)
?adult
#let's remove "education" as its duplicative with education_num
adult$education = NULL
```

To implement forward stepwise, we need a "full model" that contains anything and everything we can possible want to use as transformed predictors. Let's first create log features of all the numeric features. Instead of pure log, use log(value + 1) to handle possible zeroes.

```
skimr::skim(adult)
```

Table 1: Data summary

Name	adult
Number of rows	30161
Number of columns	14
Column type frequency:	
factor	8
numeric	6
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
workclass	0	1	FALSE	7	Pri: 22286, Sel: 2499, Loc: 2067, Sta: 1278
marital_status	0	1	FALSE	7	Mar: 14065, Nev: 9725, Div: 4214, Sep: 939
occupation	0	1	FALSE	14	Pro: 4038, Cra: 4030, Exe: 3992, Adm: 3720
relationship	0	1	FALSE	6	Hus: 12463, Not: 7725, Own: 4466, Unm: 3212
race	0	1	FALSE	5	Whi: 25932, Bla: 2817, Asi: 895, Ame: 286
sex	0	1	FALSE	2	Mal: 20379, Fem: 9782
native_country	0	1	FALSE	41	Uni: 27503, Mex: 610, Phi: 188, Ger: 128
income	0	1	FALSE	2	<=5: 22653, >50: 7508

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	38.44	13.13	17	28	37	47	90	
fnlwgt	0	1	189797.56	105652.74	13769	117628	178429	237630	1484705	
education_num	0	1	10.12	2.55	1	9	10	13	16	
capital_gain	0	1	1091.97	7406.47	0	0	0	0	99999	
capital_loss	0	1	88.38	404.30	0	0	0	0	4356	
hours_per_week	0	1	40.93	11.98	1	40	40	45	99	

```
#this gives us the list of numeric features to create logs
adult$log_age = log(adult$age + 1)
adult$log_fnlwgt = log(adult$fnlwgt + 1)
adult$log_education_num = log(adult$education_num + 1)
adult$log_capital_gain = log(adult$capital_gain + 1)
adult$log_capital_loss = log(adult$capital_loss + 1)
adult$log_hours_per_week = log(adult$hours_per_week + 1)
```

Now let's create a model matrix Xfull that contains all first order interactions. How many degrees of freedom in this "full model"?

```
#TO-DO
Xfull = model.matrix(income ~ .*., adult)
p_plus_one = ncol(Xfull)
```

```
Now let's split it into train, select and test sets. Because this will be a glm, model-building (trainin
```

```
y = ifelse(adult$income == ">50K", 1, 0)
train_idx = sample(1:n,2000)
select_idx = sample(setdiff(1:n, train_idx), 14000)
test_idx = setdiff(1:n, c(train_idx, select_idx))

Xfull_train =  Xfull[train_idx, ]
Xfull_select = Xfull[select_idx, ]
Xfull_test =   Xfull[test_idx, ]
y_train =      y[train_idx]
y_select =     y[select_idx]
y_test =       y[test_idx]
```

Now let's use the code from class to run the forward stepwise modeling. As this is binary classification, let's use logistic regression and to measure model performance, let's use the Brier score. Compute the Brier score in-sample (on training set) and oos (on selection set) for every iteration of j, the number of features selected from the greedy selection procedure.

```
#TO-DO
# Load necessary libraries
# pacman::p_load(glmnet)

# Function to calculate Brier score
# brier_score <- function(y, y_pred) {
#   mean((y_true - y_pred)^2)
# }
```

```
# # Initialize variables to store Brier scores
# # Initialize variables to store Brier scores
# brier_train <- numeric(p_plus_one)
# brier_select <- numeric(p_plus_one)

# # Loop over different numbers of features selected
# for (j in 1:p_plus_one) {
#   # Check if the number of columns in Xfull_train is sufficient
#   if (j <= ncol(Xfull_train)) {
#     # Fit logistic regression model
#     model <- glm(y_train ~ ., family = binomial(link = "logit"),
#                  data = as.data.frame(Xfull_train[, 1:j]))

#     # Predict probabilities on training and selection sets
#     y_pred_train <- predict(model, newdata = as.data.frame(Xfull_train[, 1:j]), type = "response")
#     y_pred_select <- predict(model, newdata = as.data.frame(Xfull_select[, 1:j]), type = "response")

#     # Calculate Brier score on training and selection sets
#     brier_train[j] <- brier_score(y_train, y_pred_train)
#     brier_select[j] <- brier_score(y_select, y_pred_select)
#   } else {
    # If j exceeds the number of columns in Xfull_train, set Brier scores to NA
#     brier_train[j] <- NA
#     brier_select[j] <- NA
#   }
# }
```

Plot the in-sample Brier score (in red) and oos Brier score (in blue) by the number of features used.

```
#ggplot(brier_df, aes(x = Features)) +
  #geom_line(aes(y = Train_Brier, color = "In-sample"), size = 1) +
  #geom_line(aes(y = Select_Brier, color = "Out-of-sample"), size = 1) +
  #labs(x = "Number of Features", y = "Brier Score", title = "Brier Score vs. Number of Features") +
  #scale_color_manual(values = c("In-sample" = "red", "Out-of-sample" = "blue")) +
  #theme_minimal()
```

Select the model with the best oos Brier score on the select set and find its oos Brier score on the test set.

```
#TO-DO
```

Create the final model object `g_final`.

```
#TO-DO
```

Data Wrangling / Munging / Carpentry

Throughout this assignment you should use `dplyr` with `magrittr` piping. I'll be writing the data.table code for you after you're done so you can see it as it may be useful for your future.

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

Load the `storms` dataset from the `dplyr` package and read about it using `?storms` and summarize its data via `skimr::skim`.

```
storms = dplyr::storms
?storms
skimr::skim(storms)
```

Table 4: Data summary

Name	storms
Number of rows	19537
Number of columns	13
Column type frequency:	
character	1
factor	1
numeric	11
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
name	0	1	3	9	0	260	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
status	0	1	FALSE	9	tro: 6830, hur: 4803, tro: 3569, ext: 2151

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
year	0	1.00	2002.75	12.77	1975.0	1994.0	2004.0	2013.0	2022.0	
month	0	1.00	8.71	1.35	1.0	8.0	9.0	9.0	12.0	
day	0	1.00	15.73	8.90	1.0	8.0	16.0	24.0	31.0	
hour	0	1.00	9.10	6.74	0.0	5.0	12.0	18.0	23.0	
lat	0	1.00	27.01	10.47	7.0	18.3	26.6	33.8	70.7	
long	0	1.00	-61.56	21.17	-136.9	-78.8	-62.3	-45.5	13.5	
category	14734	0.25	1.90	1.15	1.0	1.0	1.0	3.0	5.0	
wind	0	1.00	50.05	25.46	10.0	30.0	45.0	65.0	165.0	
pressure	0	1.00	993.48	18.75	882.0	986.0	1000.0	1007.0	1024.0	
tropicalstorm_force_diameter	9512	0.51	147.87	157.49	0.0	0.0	110.0	220.0	1440.0	
hurricane_force_diameter	9512	0.51	14.92	34.18	0.0	0.0	0.0	0.0	300.0	

```
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> <fct>           <dbl> <int>    <int>
## 1 Amy    1975     6    27     0  27.5 -79   tropical de~       NA    25     1013
## 2 Amy    1975     6    27     6  28.5 -79   tropical de~       NA    25     1013
## 3 Amy    1975     6    27    12  29.5 -79   tropical de~       NA    25     1013
## 4 Amy    1975     6    27    18  30.5 -79   tropical de~       NA    25     1013
## 5 Amy    1975     6    28     0  31.5 -78.8 tropical de~       NA    25     1012
## 6 Amy    1975     6    28     6  32.4 -78.7 tropical de~       NA    25     1012
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

To make the modeling exercise easier, let's eliminate rows that have missingness in `tropicalstorm_force_diameter` or `hurricane_force_diameter`.

```
storms <- storms[!(is.na(storms$tropicalstorm_force_diameter) | is.na(storms$hurricane_force_diameter)),]
skimr::skim(storms)
```

Table 8: Data summary

Name	storms
Number of rows	10025
Number of columns	13
Column type frequency:	
character	1
factor	1
numeric	11
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
name	0	1	3	9	0	162	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
status	0	1	FALSE	9	tro: 3571, hur: 2170, oth: 1317, tro: 1312

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
year	0	1.00	2013.15	5.71	2004.0	2008.0	2013.0	2018.0	2022.0	
month	0	1.00	8.63	1.47	1.0	8.0	9.0	9.0	12.0	
day	0	1.00	15.52	8.82	1.0	8.0	15.0	23.0	31.0	
hour	0	1.00	9.07	6.73	0.0	5.0	12.0	18.0	23.0	
lat	0	1.00	26.91	10.69	7.0	18.0	26.2	33.8	69.0	
long	0	1.00	-60.65	21.66	-136.9	-78.5	-61.9	-43.7	13.5	
category	7855	0.22	2.00	1.20	1.0	1.0	2.0	3.0	5.0	
wind	0	1.00	49.29	25.22	10.0	30.0	40.0	60.0	160.0	
pressure	0	1.00	993.30	19.21	882.0	987.0	1000.0	1007.0	1021.0	
tropicalstorm_force_diameter	0	1.00	147.87	157.49	0.0	0.0	110.0	220.0	1440.0	
hurricane_force_diameter	0	1.00	14.92	34.18	0.0	0.0	0.0	0.0	300.0	

```
head(storms, 100)
```

```
## # A tibble: 100 x 13
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>          <dbl> <int>    <int>
##  1 Alex   2004     7    31    18  30.3 -78.3 tropical d~       NA    25     1010
##  2 Alex   2004     8     1     0  31   -78.8 tropical d~       NA    25     1009
##  3 Alex   2004     8     1     6  31.5 -79   tropical d~       NA    25     1009
##  4 Alex   2004     8     1    12  31.6 -79.1 tropical d~       NA    30     1009
##  5 Alex   2004     8     1    18  31.6 -79.2 tropical s~       NA    35     1009
##  6 Alex   2004     8     2     0  31.5 -79.3 tropical s~       NA    35     1007
##  7 Alex   2004     8     2     6  31.4 -79.4 tropical s~       NA    40     1005
##  8 Alex   2004     8     2    12  31.3 -79   tropical s~       NA    50      992
##  9 Alex   2004     8     2    18  31.8 -78.7 tropical s~       NA    50      993
## 10 Alex   2004     8     3     0  32.4 -78.2 tropical s~       NA    60      987
## # i 90 more rows
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

Which column(s) should be converted to type factor? Do the conversion:

```
#TO-DO : It should be the the different status of the storm. It could be converted into leveled factors

# Check levels of status
unique(storms$status)
```

```
## [1] tropical depression    tropical storm         hurricane
## [4] extratropical          tropical wave          other low
## [7] subtropical storm      subtropical depression disturbance
## 9 Levels: disturbance extratropical hurricane ... tropical wave
```

```
storms$status = as.integer(factor(storms$status, levels = c("other low", "disturbance", "tropical wave"

head(storms, 50)
```

```
## # A tibble: 50 x 13
```

```
##    name   year month   day  hour   lat  long status category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl>  <int>    <dbl> <int>    <int>
##  1 Alex   2004     7    31    18  30.3 -78.3      4       NA    25     1010
##  2 Alex   2004     8     1     0  31   -78.8      4       NA    25     1009
##  3 Alex   2004     8     1     6  31.5 -79        4       NA    25     1009
##  4 Alex   2004     8     1    12  31.6 -79.1      4       NA    30     1009
##  5 Alex   2004     8     1    18  31.6 -79.2      6       NA    35     1009
##  6 Alex   2004     8     2     0  31.5 -79.3      6       NA    35     1007
##  7 Alex   2004     8     2     6  31.4 -79.4      6       NA    40     1005
##  8 Alex   2004     8     2    12  31.3 -79        6       NA    50      992
##  9 Alex   2004     8     2    18  31.8 -78.7      6       NA    50      993
## 10 Alex   2004     8     3     0  32.4 -78.2      6       NA    60      987
## # i 40 more rows
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

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

```
#TO-DO
# Reorder columns
storms = storms[, c("name", "status", "category", setdiff(names(storms), c("name", "status", "category"


head(storms, 100)
```

```
## # A tibble: 100 x 13
##    name  status category  year month   day  hour   lat  long  wind pressure
##    <chr>  <int>    <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
##  1 Alex       4       NA  2004     7    31    18  30.3 -78.3    25     1010
##  2 Alex       4       NA  2004     8     1     0  31   -78.8    25     1009
##  3 Alex       4       NA  2004     8     1     6  31.5 -79      25     1009
##  4 Alex       4       NA  2004     8     1    12  31.6 -79.1    30     1009
##  5 Alex       6       NA  2004     8     1    18  31.6 -79.2    35     1009
##  6 Alex       6       NA  2004     8     2     0  31.5 -79.3    35     1007
##  7 Alex       6       NA  2004     8     2     6  31.4 -79.4    40     1005
##  8 Alex       6       NA  2004     8     2    12  31.3 -79      50      992
##  9 Alex       6       NA  2004     8     2    18  31.8 -78.7    50      993
## 10 Alex       6       NA  2004     8     3     0  32.4 -78.2    60      987
## # i 90 more rows
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

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

```
#TO-DO
storm_1970 = storms[storms$year >= 1970 & storms$year < 1980, ]

storm_1970
```

```
## # A tibble: 0 x 13
## # i 13 variables: name <chr>, status <int>, category <dbl>, year <dbl>,
## #   month <dbl>, day <int>, hour <dbl>, lat <dbl>, long <dbl>, wind <int>,
## #   pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

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

```
#TO-DO
storm_category4_above100 = storms[storms$category == 4 & storms$wind >= 100, ]


storm_category4_above100
```

```
## # A tibble: 298 x 13
##    name    status category  year month   day  hour   lat  long  wind pressure
##    <chr>    <int>    <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
##  1 Charley      7        4  2004     8    13    18  26.1 -82.4   125      947
##  2 Frances      7        4  2004     8    28    18  17.7 -52.3   115      948
##  3 Frances      7        4  2004     8    29     0  18.1 -52.9   115      948
##  4 Frances      7        4  2004     8    29     6  18.4 -53.6   115      948
##  5 Frances      7        4  2004     8    29    12  18.6 -54.4   115      948
##  6 Frances      7        4  2004     8    31     6  19.8 -62.1   115      950
##  7 Frances      7        4  2004     8    31    12  20   -63.5   120      949
##  8 Frances      7        4  2004     8    31    18  20.3 -65     125      942
##  9 Frances      7        4  2004     9     1     0  20.6 -66.4   120      941
## 10 Frances      7        4  2004     9     1     6  21   -67.9   120      939
## # i 288 more rows
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

Create a new feature `wind_speed_per_unit_pressure`.

```
#TO-DO
storms$wind_speed_per_unit_pressure = storms$wind / storms$pressure

head(storms)
```

```
## # A tibble: 6 x 14
##   name  status category  year month   day  hour   lat  long  wind pressure
##   <chr>  <int>    <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
## 1 Alex       4       NA  2004     7    31    18  30.3 -78.3    25     1010
## 2 Alex       4       NA  2004     8     1     0  31   -78.8    25     1009
## 3 Alex       4       NA  2004     8     1     6  31.5 -79      25     1009
## 4 Alex       4       NA  2004     8     1    12  31.6 -79.1    30     1009
## 5 Alex       6       NA  2004     8     1    18  31.6 -79.2    35     1009
## 6 Alex       6       NA  2004     8     2     0  31.5 -79.3    35     1007
## # i 3 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, 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.

```
#TO-DO
storms$average_diameter <- rowMeans(storms[, c("tropicalstorm_force_diameter", "hurricane_force_diameter
# View the updated dataset
head(storms)
```

```
## # A tibble: 6 x 15
##   name  status category  year month   day  hour   lat  long  wind pressure
##   <chr>  <int>    <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
## 1 Alex       4       NA  2004     7    31    18  30.3 -78.3    25     1010
## 2 Alex       4       NA  2004     8     1     0  31   -78.8    25     1009
## 3 Alex       4       NA  2004     8     1     6  31.5 -79      25     1009
## 4 Alex       4       NA  2004     8     1    12  31.6 -79.1    30     1009
## 5 Alex       6       NA  2004     8     1    18  31.6 -79.2    35     1009
## 6 Alex       6       NA  2004     8     2     0  31.5 -79.3    35     1007
## # i 4 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, wind_speed_per_unit_pressure <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.

```
#TO-DO
storm_max_wind = aggregate(wind ~ name, data = storms, FUN = max)

# View the summary dataframe
head(storm_max_wind)
```

```
##       name wind
## 1 AL022006   45
## 2 AL102004   30
## 3 Al202011   40
## 4  Alberto   60
## 5     Alex  105
## 6    Alpha   45
```

Order your dataset by maximum wind speed storm but within the rows of storm show the observations in time order from early to late.

```
#TO-DO

# Order the dataset by maximum wind speed for each storm
storm_ordered = storms %>%
  group_by(name) %>%
  arrange(desc(wind)) %>%
  ungroup() %>%
  arrange(name, hour)

# View the ordered dataset
head(storm_ordered)
```

```
## # A tibble: 6 x 15
##   name     status category  year month   day  hour   lat  long  wind pressure
##   <chr>     <int>    <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>    <int>
## 1 AL022006      6       NA  2006     7    18     0  42.4 -62.1    40      999
## 2 AL022006      8       NA  2006     7    17     0  38.3 -67.6    30     1009
## 3 AL022006      1       NA  2006     7    19     0  48.6 -52.9    25     1012
## 4 AL022006      6       NA  2006     7    18     6  43.7 -60.1    35     1004
## 5 AL022006      4       NA  2006     7    17     6  39.1 -66.4    30     1008
```

```
## 6 AL022006      1       NA  2006     7    19     6  49.2 -49.4    25     1012
## # i 4 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, wind_speed_per_unit_pressure <dbl>,
## #   average_diameter <dbl>
```

Find the strongest storm by wind speed per year.

```
#distinct(storms[, max_wind_by_year := max(wind), by = year][wind == max_wind_by_year, .(year, name, wi

#storms %>%
#  group_by(year) %>%
#  filter(wind == max(wind)) %>%
#  select(year, name, wind) %>%
#  distinct %>%
 # select(year, name)
```

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).

```
#TO-DO
#colnames(storms)
#storms_filtered = storms %>%
#  group_by(name) %>%
#  filter(!all(is.na(category), is.na(wind), is.na(pressure), is.na(average_diameter)))
#
# For each named storm, find its maximum category, wind speed, pressure, and diameter
#storm_summary = storms_filtered %>%
#  group_by(name) %>%
#  summarise(
#    max_category = max(category, na.rm = TRUE),
#    max_wind = max(wind, na.rm = TRUE),
#    max_pressure = min(pressure, na.rm = TRUE),
#    max_diameter = max(average_diameter, na.rm = TRUE)
#  )

# View the summary
#print(storm_summary)
```

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) %>%
  summarize(num_storms = n_distinct(name))
```

```
## # A tibble: 48 x 2
##     year num_storms
##    <dbl>      <int>
##  1  1975          8
##  2  1976          7
##  3  1977          6
```

```
##  4  1978         11
##  5  1979          8
##  6  1980         11
##  7  1981         11
##  8  1982          5
##  9  1983          4
## 10  1984         12
## # i 38 more rows
```

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

```
#TO-DO
storm_tally = storms %>%
  group_by(year, category) %>%
  summarise(storm_count = n())
```

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

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

```
#TO-DO
max_wind_per_status = storms %>%
  group_by(year, status) %>%
  summarise(max_wind_speed = max(wind, na.rm = TRUE))
```

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

```
print(max_wind_per_status)
```

```
## # A tibble: 290 x 3
## # Groups:   year [48]
##     year status                  max_wind_speed
##    <dbl> <fct>                            <int>
##  1  1975 extratropical                       75
##  2  1975 hurricane                          120
##  3  1975 subtropical depression              30
##  4  1975 subtropical storm                   45
##  5  1975 tropical depression                 30
##  6  1975 tropical storm                      60
##  7  1976 extratropical                       55
##  8  1976 hurricane                          105
##  9  1976 tropical depression                 30
## 10  1976 tropical storm                      60
## # i 280 more rows
```

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

```r
#TO-DO
colnames(storms)
```

```
##  [1] "name"                          "year"
##  [3] "month"                         "day"
##  [5] "hour"                          "lat"
##  [7] "long"                          "status"
##  [9] "category"                      "wind"
## [11] "pressure"                      "tropicalstorm_force_diameter"
## [13] "hurricane_force_diameter"
```

```r
storm_location_summary <- storms %>%
  group_by(name) %>%
  summarise(avg_latitude = mean(lat, na.rm = TRUE),
            avg_longitude = mean(long, na.rm = TRUE))


print(storm_location_summary)
```

```
## # A tibble: 260 x 3
##    name     avg_latitude avg_longitude
##    <chr>           <dbl>         <dbl>
##  1 AL011993        25.7          -75.2
##  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        43.1          -60.3
## 10 AL031987        30.8          -88.7
## # i 250 more rows
```

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

```r
colnames(storms)
```

```
##  [1] "name"                          "year"
##  [3] "month"                         "day"
##  [5] "hour"                          "lat"
##  [7] "long"                          "status"
##  [9] "category"                      "wind"
## [11] "pressure"                      "tropicalstorm_force_diameter"
## [13] "hurricane_force_diameter"
```

```r
# Convert time to POSIXct format
storms$hour <- as.POSIXct(storms$hour)

# Calculate the duration of each storm in hours
storm_duration <- storms %>%
```

```
  group_by(name) %>%
  summarise(duration_hours = round(diff(range(hour), units = "hours") / 6) * 6)

# View the summary
print(storm_duration)
```

```
## # A tibble: 260 x 2
##    name     duration_hours
##    <chr>    <drtn>
##  1 AL011993 18 secs
##  2 AL012000 18 secs
##  3 AL021992 18 secs
##  4 AL021994 18 secs
##  5 AL021999 18 secs
##  6 AL022000 18 secs
##  7 AL022001 18 secs
##  8 AL022003 18 secs
##  9 AL022006 18 secs
## 10 AL031987 18 secs
## # i 250 more rows
```

For storm in a category, create a variable `storm_number` that enumerates the storms 1, 2, … (in date order).

```
#TO-DO
# Ensure time is in POSIXct format and combine all components to become date
storms$date = as.Date(paste(storms$year, storms$month, storms$day, sep = "-"))

# Group by category and arrange by time within each category
storms = storms %>%
  group_by(category) %>%
  arrange(date) %>%
  mutate(storm_number = row_number())

# View the updated dataset
print(storms)
```

```
## # A tibble: 19,537 x 15
## # Groups:   category [6]
##    name   year month   day hour                 lat  long status category  wind
##    <chr> <dbl> <dbl> <int> <dttm>              <dbl> <dbl> <fct>     <dbl> <int>
##  1 Amy    1975     6    27 1969-12-31 19:00:00  27.5 -79   tropi~       NA    25
##  2 Amy    1975     6    27 1969-12-31 19:00:06  28.5 -79   tropi~       NA    25
##  3 Amy    1975     6    27 1969-12-31 19:00:12  29.5 -79   tropi~       NA    25
##  4 Amy    1975     6    27 1969-12-31 19:00:18  30.5 -79   tropi~       NA    25
##  5 Amy    1975     6    28 1969-12-31 19:00:00  31.5 -78.8 tropi~       NA    25
##  6 Amy    1975     6    28 1969-12-31 19:00:06  32.4 -78.7 tropi~       NA    25
##  7 Amy    1975     6    28 1969-12-31 19:00:12  33.3 -78   tropi~       NA    25
##  8 Amy    1975     6    28 1969-12-31 19:00:18  34   -77   tropi~       NA    30
##  9 Amy    1975     6    29 1969-12-31 19:00:00  34.4 -75.8 tropi~       NA    35
## 10 Amy    1975     6    29 1969-12-31 19:00:06  34   -74.8 tropi~       NA    40
## # i 19,527 more rows
## # i 5 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, 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`.

```
#TO-DO : I converted the date and added a new column date
head(storms)
```

```
## # A tibble: 6 x 15
## # Groups:   category [1]
##   name   year month   day hour                 lat  long status  category  wind
##   <chr> <dbl> <dbl> <int> <dttm>              <dbl> <dbl> <fct>      <dbl> <int>
## 1 Amy    1975     6    27 1969-12-31 19:00:00  27.5 -79   tropic~       NA    25
## 2 Amy    1975     6    27 1969-12-31 19:00:06  28.5 -79   tropic~       NA    25
## 3 Amy    1975     6    27 1969-12-31 19:00:12  29.5 -79   tropic~       NA    25
## 4 Amy    1975     6    27 1969-12-31 19:00:18  30.5 -79   tropic~       NA    25
## 5 Amy    1975     6    28 1969-12-31 19:00:00  31.5 -78.8 tropic~       NA    25
## 6 Amy    1975     6    28 1969-12-31 19:00:06  32.4 -78.7 tropic~       NA    25
## # i 5 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, storm_number <int>
```

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.

```
#TO-DO
storms$day_of_week <- factor(weekdays(storms$date), levels = c("Sunday", "Monday", "Tuesday", "Wednesday
storms$week_of_year <- isoweek(storms$date)

# View the updated dataset
print(storms)
```

```
## # A tibble: 19,537 x 17
## # Groups:   category [6]
##    name   year month   day hour                 lat  long status category  wind
##    <chr> <dbl> <dbl> <int> <dttm>              <dbl> <dbl> <fct>     <dbl> <int>
##  1 Amy    1975     6    27 1969-12-31 19:00:00  27.5 -79   tropi~       NA    25
##  2 Amy    1975     6    27 1969-12-31 19:00:06  28.5 -79   tropi~       NA    25
##  3 Amy    1975     6    27 1969-12-31 19:00:12  29.5 -79   tropi~       NA    25
##  4 Amy    1975     6    27 1969-12-31 19:00:18  30.5 -79   tropi~       NA    25
##  5 Amy    1975     6    28 1969-12-31 19:00:00  31.5 -78.8 tropi~       NA    25
##  6 Amy    1975     6    28 1969-12-31 19:00:06  32.4 -78.7 tropi~       NA    25
##  7 Amy    1975     6    28 1969-12-31 19:00:12  33.3 -78   tropi~       NA    25
##  8 Amy    1975     6    28 1969-12-31 19:00:18  34   -77   tropi~       NA    30
##  9 Amy    1975     6    29 1969-12-31 19:00:00  34.4 -75.8 tropi~       NA    35
## 10 Amy    1975     6    29 1969-12-31 19:00:06  34   -74.8 tropi~       NA    40
## # i 19,527 more rows
## # i 7 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, storm_number <int>,
## #   day_of_week <fct>, week_of_year <int>
```

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

```r
#TO-DO
storms$start_day = format(storms$timestamp, "%A, %B %d, %Y")
```

```
## Warning: Unknown or uninitialised column: `timestamp`.
```

```r
# View the updated dataset
print(storms)
```

```
## # A tibble: 19,537 x 18
## # Groups:   category [6]
##    name   year month   day hour                 lat  long status category  wind
##    <chr> <dbl> <dbl> <int> <dttm>              <dbl> <dbl> <fct>     <dbl> <int>
##  1 Amy    1975     6    27 1969-12-31 19:00:00  27.5 -79   tropi~       NA    25
##  2 Amy    1975     6    27 1969-12-31 19:00:06  28.5 -79   tropi~       NA    25
##  3 Amy    1975     6    27 1969-12-31 19:00:12  29.5 -79   tropi~       NA    25
##  4 Amy    1975     6    27 1969-12-31 19:00:18  30.5 -79   tropi~       NA    25
##  5 Amy    1975     6    28 1969-12-31 19:00:00  31.5 -78.8 tropi~       NA    25
##  6 Amy    1975     6    28 1969-12-31 19:00:06  32.4 -78.7 tropi~       NA    25
##  7 Amy    1975     6    28 1969-12-31 19:00:12  33.3 -78   tropi~       NA    25
##  8 Amy    1975     6    28 1969-12-31 19:00:18  34   -77   tropi~       NA    30
##  9 Amy    1975     6    29 1969-12-31 19:00:00  34.4 -75.8 tropi~       NA    35
## 10 Amy    1975     6    29 1969-12-31 19:00:06  34   -74.8 tropi~       NA    40
## # i 19,527 more rows
## # i 8 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, storm_number <int>,
## #   day_of_week <fct>, week_of_year <int>, start_day <chr>
```

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

```r
#TO-DO
# Calculate the deciles of wind speed
deciles = quantile(storms$wind, probs = seq(0, 1, by = 0.1), na.rm = TRUE)

# Create a new factor variable decile_windspeed
storms$decile_windspeed = cut(storms$wind, breaks = deciles, labels = 1:10, include.lowest = TRUE)

# Convert to factor
storms$decile_windspeed = as.factor(storms$decile_windspeed)

# View the updated dataset
print(storms)
```

```
## # A tibble: 19,537 x 19
## # Groups:   category [6]
##    name   year month   day hour                 lat  long status category  wind
##    <chr> <dbl> <dbl> <int> <dttm>              <dbl> <dbl> <fct>     <dbl> <int>
##  1 Amy    1975     6    27 1969-12-31 19:00:00  27.5 -79   tropi~       NA    25
##  2 Amy    1975     6    27 1969-12-31 19:00:06  28.5 -79   tropi~       NA    25
##  3 Amy    1975     6    27 1969-12-31 19:00:12  29.5 -79   tropi~       NA    25
##  4 Amy    1975     6    27 1969-12-31 19:00:18  30.5 -79   tropi~       NA    25
##  5 Amy    1975     6    28 1969-12-31 19:00:00  31.5 -78.8 tropi~       NA    25
##  6 Amy    1975     6    28 1969-12-31 19:00:06  32.4 -78.7 tropi~       NA    25
```

```
##  7 Amy    1975     6    28 1969-12-31 19:00:12  33.3 -78   tropi~       NA    25
##  8 Amy    1975     6    28 1969-12-31 19:00:18  34   -77   tropi~       NA    30
##  9 Amy    1975     6    29 1969-12-31 19:00:00  34.4 -75.8 tropi~       NA    35
## 10 Amy    1975     6    29 1969-12-31 19:00:06  34   -74.8 tropi~       NA    40
## # i 19,527 more rows
## # i 9 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, storm_number <int>,
## #   day_of_week <fct>, week_of_year <int>, start_day <chr>,
## #   decile_windspeed <fct>
```

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

```
#TO-DO

serious_storms = storms[storms$category >= 3, ]

print(serious_storms)
```

```
## # A tibble: 15,996 x 19
## # Groups:   category [4]
##    name   year month   day hour     lat  long status category  wind
##    <chr> <dbl> <dbl> <int> <dttm> <dbl> <dbl> <fct>     <dbl> <int>
##  1 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  2 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  3 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  4 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  5 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  6 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  7 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  8 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
##  9 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
## 10 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
## # i 15,986 more rows
## # i 9 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, storm_number <int>,
## #   day_of_week <fct>, week_of_year <int>, start_day <chr>,
## #   decile_windspeed <fct>
```

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

```
#TO-DO
serious_storms$lat_long <- paste(serious_storms$lat, serious_storms$long, sep = " / ")


print(serious_storms)
```

```
## # A tibble: 15,996 x 20
## # Groups:   category [4]
##    name   year month   day hour     lat  long status category  wind
##    <chr> <dbl> <dbl> <int> <dttm> <dbl> <dbl> <fct>     <dbl> <int>
##  1 <NA>     NA    NA    NA NA         NA    NA <NA>         NA    NA
```

```
##  2 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  3 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  4 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  5 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  6 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  7 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  8 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
##  9 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
## 10 <NA>     NA    NA    NA NA        NA    NA <NA>         NA    NA
## # i 15,986 more rows
## # i 10 more variables: pressure <int>, tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>, date <date>, storm_number <int>,
## #   day_of_week <fct>, week_of_year <int>, start_day <chr>,
## #   decile_windspeed <fct>, lat_long <chr>
```

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

```
#TO-DO
colnames(storms)
```

```
##  [1] "name"                          "year"
##  [3] "month"                         "day"
##  [5] "hour"                          "lat"
##  [7] "long"                          "status"
##  [9] "category"                      "wind"
## [11] "pressure"                      "tropicalstorm_force_diameter"
## [13] "hurricane_force_diameter"      "date"
## [15] "storm_number"                  "day_of_week"
## [17] "week_of_year"                  "start_day"
## [19] "decile_windspeed"
```

```
category_averages <- storms %>%
  group_by(category) %>%
  summarise(avg_wind_speed = mean(wind, na.rm = TRUE),
            avg_pressure = mean(pressure, na.rm = TRUE),
            avg_diameter = mean(hurricane_force_diameter, na.rm = TRUE))

# View the result
print(category_averages)
```

```
## # A tibble: 6 x 4
##   category avg_wind_speed avg_pressure avg_diameter
##      <dbl>          <dbl>        <dbl>        <dbl>
## 1        1           71.0         981.        49.7
## 2        2           89.5         967.        70.7
## 3        3          104.          955.        75.0
## 4        4          122.          940.        81.5
## 5        5          146.          918.        90.7
## 6       NA           38.1        1002.         1.67
```

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).

```
#TO-DO
storm_summary <- storms %>%
  group_by(name) %>%
  summarise(max_category = max(category, na.rm = TRUE),
            max_wind_speed = max(wind, na.rm = TRUE),
            max_pressure = min(pressure, na.rm = TRUE),
            max_diameter = max(hurricane_force_diameter, na.rm = TRUE),
            readings = sum(!is.na(wind)))
```

```
## Warning: There were 182 warnings in `summarise()`.
## The first warning was:
## i In argument: `max_category = max(category, na.rm = TRUE)`.
## i In group 1: `name = "AL011993"`.
## Caused by warning in `max()`:
## ! no non-missing arguments to max; returning -Inf
## i Run `dplyr::last_dplyr_warnings()` to see the 181 remaining warnings.
```

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)
#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.

Fit your model. Validate it.

```
#TO-DO
```

Assess your level of success at this endeavor.

#TO-DO

More data munging with table joins

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

We will be using the `storms` dataset from the `dplyr` package. Filter this dataset on all storms that have no missing measurements for the two diameter variables, "ts_diameter" and "hu_diameter". Zeroes count as missing as well.

```
#TO-DO
#storms_filtered <- storms %>%
#  filter(complete.cases(ts_diameter, hu_diameter))

# View the filtered dataset
#print(storms_filtered)
```

From this subset, create a data frame that only has storm name, observation period number for each storm (i.e., 1, 2, ..., T) and the "ts_diameter" and "hu_diameter" metrics.

```
#TO-DO
```

Create a data frame in long format with columns "diameter" for the measurement and "diameter_type" which will be categorical taking on the values "hu" or "ts".

```
#TO-DO
```

Using this long-formatted data frame, use a line plot to illustrate both "ts_diameter" and "hu_diameter" metrics by observation period for four random storms using a 2x2 faceting. The two diameters should appear in two different colors and there should be an appropriate legend.