Lab 1

Conie O'Malley

2025-01-21

```r
# libraries
library(reticulate)
```

```
Warning: package 'reticulate' was built under R version 4.3.3
```

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(ggplot2)
#use_condaenv("datascience", required = FALSE) # set my environment
```

Part 1: Hello World and Beyond

Issuing Interactive Commands & Adding Comments

```r
print("Hello World!") # first line r of code
```

```
[1] "Hello World!"
```

```r
print("Hello Stats!") # second line r of code
```

```
[1] "Hello Stats!"
```

```python
print("Hello World!") # frist line of python code
```

```
Hello World!
```

```python
print("Hello Stats!") # second line of python code
```

```
Hello Stats!
```

Doing Simple Math Calculations

```r
1 + 2 + 3 + 4 + 5
```

```
[1] 15
```

```r
sum(1:5) # alternate way to write the code
```

```
[1] 15
```

Creating and Using Vectors and Operations

```r
c(1, 2, 3, 4, 5) # concatenate
```

```
[1] 1 2 3 4 5
```

```r
1:5 # sequence operator
```

```
[1] 1 2 3 4 5
```

```r
sum(1:5) # addition using the sequence operator
```

```
[1] 15
```

Storing and Calculating Values

```r
x <- 1:5 # vector assignment
y <- 10 # vector assignment
x+y # vector addition
```

```
[1] 11 12 13 14 15
```

```r
z <- x+y # vector assignment
z
```

```
[1] 11 12 13 14 15
```

```r
h <- "Hello" # vector assignment
h
```

```
[1] "Hello"
```

```r
hw <- c("Hello", "World!") # vector concatenation
print(hw) # print view
```

```
[1] "Hello"  "World!"
```

```r
paste(hw) # paste view
```

```
[1] "Hello"  "World!"
```

Navigating the RStudio Workspace

```
ls() # view created objects
```

```
[1] "h"  "hw" "x"  "y"  "z"
```

```
rm("z") # remove "z" object
ls() # confirm removal
```

```
[1] "h"  "hw" "x"  "y"
```

More Practice Vectorizing & Vectors of Unequal Length

```
baskets.of.granny <- c(12, 4, 4, 6, 9, 3)
sum(baskets.of.granny)
```

```
[1] 38
```

```
firstnames <- c("John", "Jacqueline", "Robert") # create vector list
lastname <- "Kennedy" # create vector
paste(firstnames, lastname) # paste vectors
```

```
[1] "John Kennedy"       "Jacqueline Kennedy" "Robert Kennedy"
```

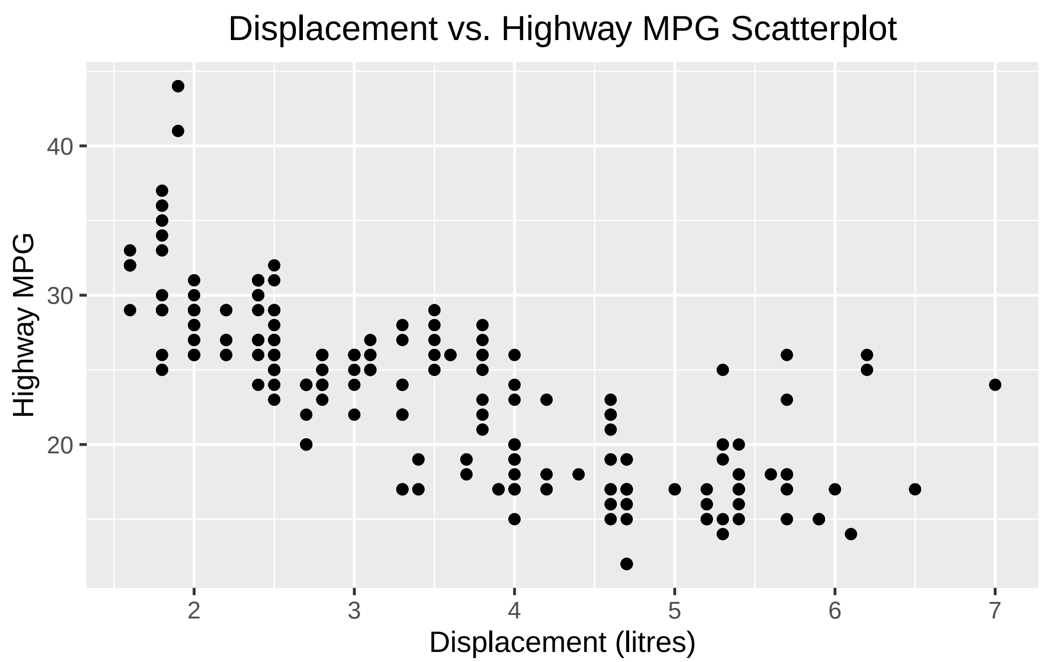
```
lastnames <- c("Kennedy", "Kennedy-Onnasis") # create vector list
paste(firstnames, lastnames) # paste vectors
```

```
[1] "John Kennedy"                "Jacqueline Kennedy-Onnasis"
[3] "Robert Kennedy"
```

Part 2: Statistical Analysis with R

Scatter Plot

```
ggplot(mpg, aes(displ, hwy)) + # create a scatter plot
  geom_point() +
  labs(title = "Displacement vs. Highway MPG Scatterplot", x = "Displacement (litres)",
       y = "Highway MPG") + # apply labels
  theme(plot.title = element_text(hjust = 0.5))
```

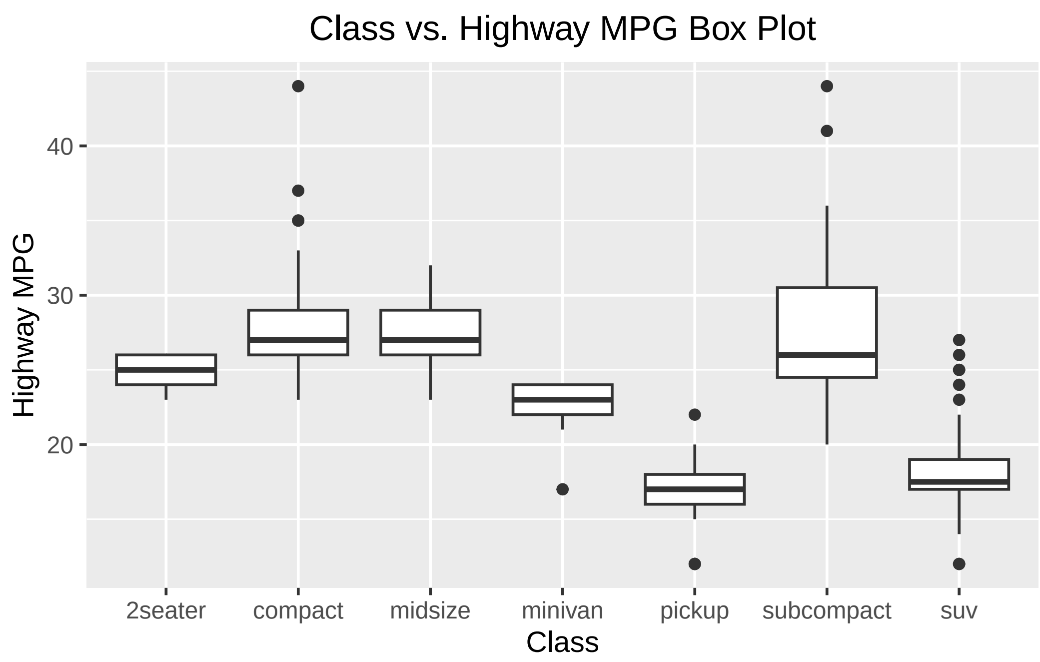


Analysis

There is a inverse relationship between the volume of engine displacement and highway miles per gallon. The relationship appears to be slightly curvilinear, but cannot be confirmed without a residuals plot.

Box Plot

```
ggplot(mpg, aes(class, hwy)) + # create a box plot
  geom_boxplot() +
  labs(title = "Class vs. Highway MPG Box Plot", x = "Class", y = "Highway MPG") + # apply la
  theme(plot.title = element_text(hjust = 0.5))
```



Analysis

There seems to be some commonality among the types *2seater*, *compact*, *midsize*, and *subcompact*, because they all have overlapping box plot ranges. *pickup* and *suv* also fall into their own common area because of overlapping ranges, while the *minivan* class has no overlaps. There are a considerable amount of outliers in the *suv* class - meaning that, at first glance, this is the most varied class of vehicles, where as *2seater* has no outliers and a very tight range - meaning that, at first glance, this is likely the most homogeneous class of vehicles.

Computing Basic Statistics

```
mean(economics$unemploy) # calculate mean
```

```
[1] 7771.31
```

```
var(economics$unemploy) # calculate variance
```

```
[1] 6979948
```

```r
sd(economics$unemploy) # calculate standard deviation
```

```
[1] 2641.959
```

```r
min(economics$unemploy) # calculate min
```

```
[1] 2685
```

```r
max(economics$unemploy) # calculate max
```

```
[1] 15352
```

```r
median(economics$unemploy) # calculate median
```

```
[1] 7494
```

```r
cor(economics$pce, economics$psavert) # calculate correlation
```

```
[1] -0.7928546
```

```r
pce_psavert_cor <- round(cor(economics$pce, economics$psavert),4)
# assign correlation to vector variable
```

Analysis

There is a strong, negative correlation between *pce* and *psavert* (-0.7929).

Conducting a t-test

```r
data(tips, package = 'reshape2') # attach data
t.test(tips$tip, alternative = 'two.sided', mu=2.50) # conduct two tail t-test
```

```
    One Sample t-test

data:  tips$tip
t = 5.6253, df = 243, p-value = 5.08e-08
alternative hypothesis: true mean is not equal to 2.5
95 percent confidence interval:
 2.823799 3.172758
sample estimates:
mean of x
 2.998279
```

Analysis

t is our t-value, which is the standardized test statistic for this data set. Our t-value should be greater than our t statistic we are testing against, so we can reject $H0 : mu = 2.50$. *df* are the degrees of freedom in this data set. *p-value* is the probability that we will get an sample mean under the $H0$. Our p-value (5.08e-08) < 0.05, meaning we can reject $H0$. *confidence interval* is 95% - meaning that if we sampled the data randomly, our sample mean would be within the range 95% of the time. *sample mean of x* is the mean of our current sample from the data. We can reject $H0 : mu = 2.50$ because our p-value < 0.05.

Building a Linear Regression Model

```r
head(mpg) # view first 6 rows of data
```

```
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
3 audi         a4      2     2008     4 manual(m6) f        20    31 p     compa~
4 audi         a4      2     2008     4 auto(av)   f        21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

```r
tail(mpg) # view last 6 rows of data
```

```
# A tibble: 6 x 11
  manufacturer model  displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr>  <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 volkswagen   passat   1.8  1999     4 auto(l5)   f        18    29 p     mids~
2 volkswagen   passat   2    2008     4 auto(s6)   f        19    28 p     mids~
3 volkswagen   passat   2    2008     4 manual(m6) f        21    29 p     mids~
4 volkswagen   passat   2.8  1999     6 auto(l5)   f        16    26 p     mids~
5 volkswagen   passat   2.8  1999     6 manual(m5) f        18    26 p     mids~
6 volkswagen   passat   3.6  2008     6 auto(s6)   f        17    26 p     mids~
```

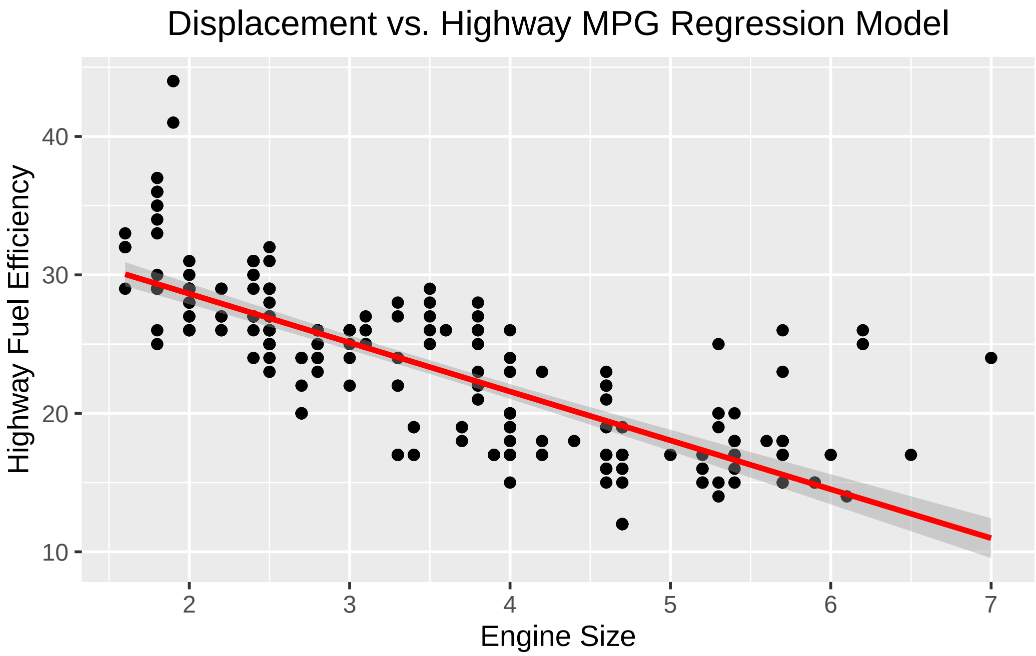
```r
lm(hwy ~ displ, mpg) # build a basic linear regression model
```

```
Call:
lm(formula = hwy ~ displ, data = mpg)

Coefficients:
(Intercept)        displ
     35.698       -3.531
```

```r
ggplot(mpg, aes(displ, hwy)) + # build a linear regression model scatter plot
  geom_point() +
  labs(title = "Displacement vs. Highway MPG Regression Model", x = "Engine Size",
       y = "Highway Fuel Efficiency") +
  geom_smooth(method = "lm", color = "red") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
fuelILM <- lm(displ ~ hwy, mpg) # assign new model to vector variable
fuelILM
```

```

Call:
lm(formula = displ ~ hwy, data = mpg)

Coefficients:
(Intercept)          hwy
     7.3676      -0.1662
```

```
summary(fuelILM) # review summary statistics
```

```

Call:
lm(formula = displ ~ hwy, data = mpg)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4126 -0.5710 -0.1105  0.4571  3.6212

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.367570   0.221422   33.27   <2e-16 ***
hwy         -0.166201   0.009157  -18.15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8323 on 232 degrees of freedom
Multiple R-squared:  0.5868,    Adjusted R-squared:  0.585
F-statistic: 329.5 on 1 and 232 DF,  p-value: < 2.2e-16
```

Part 3: Basic Importing and Wrangling of Data

Inspecting and Cleaning the Data

```r
housing <- read.table("http://www.jaredlander.com/data/housing.csv", sep = ",",
                      header = TRUE, stringsAsFactors = FALSE) # read in outside data
names(housing) <- c("Neighborhood", "Class", "Units", "YearsBuilt", "SqFt", "Income",
                    "IncomePer-SqFt", "Expense", "ExpensePerSqFt", "NetIncome", "Value",
                    "ValuePerSqFt", "Boro") # rename columns
head(housing)
```

```
  Neighborhood           Class Units YearsBuilt   SqFt   Income IncomePer-SqFt
1    FINANCIAL R9-CONDOMINIUM    42       1920  36500  1332615          36.51
2    FINANCIAL R4-CONDOMINIUM    78       1985 126420  6633257          52.47
3    FINANCIAL RR-CONDOMINIUM   500         NA 554174 17310000          31.24
4    FINANCIAL R4-CONDOMINIUM   282       1930 249076 11776313          47.28
5      TRIBECA R4-CONDOMINIUM   239       1985 219495 10004582          45.58
6      TRIBECA R4-CONDOMINIUM   133       1986 139719  5127687          36.70
  Expense ExpensePerSqFt NetIncome    Value ValuePerSqFt      Boro
1  342005           9.37    990610  7300000       200.00 Manhattan
2 1762295          13.94   4870962 30690000       242.76 Manhattan
3 3543000           6.39  13767000 90970000       164.15 Manhattan
4 2784670          11.18   8991643 67556006       271.23 Manhattan
5 2783197          12.68   7221385 54320996       247.48 Manhattan
6 1497788          10.72   3629899 26737996       191.37 Manhattan
```

Building a Linear Regression Model

```
house1 <- lm(ValuePerSqFt ~ Units + SqFt + Boro, housing) # build a linear regression model
summary(house1) # view summary statistics
```

```
Call:
lm(formula = ValuePerSqFt ~ Units + SqFt + Boro, data = housing)

Residuals:
     Min       1Q   Median       3Q      Max
-164.418  -22.692    1.416   26.972  261.122

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        4.329e+01  5.330e+00   8.122 6.97e-16 ***
Units             -1.881e-01  2.210e-02  -8.511  < 2e-16 ***
SqFt               2.103e-04  2.087e-05  10.079  < 2e-16 ***
BoroBrooklyn       3.456e+01  5.535e+00   6.244 4.95e-10 ***
BoroManhattan      1.310e+02  5.385e+00  24.327  < 2e-16 ***
BoroQueens         3.299e+01  5.663e+00   5.827 6.35e-09 ***
BoroStaten Island -3.630e+00  9.993e+00  -0.363    0.716
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 43.35 on 2619 degrees of freedom
Multiple R-squared:  0.6009,    Adjusted R-squared:    0.6
F-statistic: 657.2 on 6 and 2619 DF,  p-value: < 2.2e-16
```

Part 4: Hello World, Data, Statistics and Beyond in Python

Hello World in Python

```
print("Hello World!") # print function practice
```

```
Hello World!
```

```
print("Hello Stats!") # print function practice
```

```
Hello Stats!
```

```python
print("Hello", "World!") # print function practice
```

```
Hello World!
```

Doing Simple Math Calculations

```python
1+2
```

```
3
```

```python
1+2+3+4+5
```

```
15
```

Installing and importing packages

```python
from matplotlib import pyplot as plt # import libraries
from statsmodels.formula.api import ols
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn import datasets
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import seaborn as sns
```

Accessing a Built-In Dataset with Python

```python
housing = fetch_california_housing() # assign data to matrix vector
X,y = housing.data, housing.target # assign data subsets to variable vectors
print("The size of the dataset is {}".format(X.shape))
```

```
The size of the dataset is (20640, 8)
```

```python
print("The names of the data columns are {}", housing.feature_names)
```

```
The names of the data columns are {} ['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Popula
```

```python
print(housing.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESCR'])
```

```python
hypothesis = LinearRegression() # set hypothesis model type
hypothesis.fit(X,y) # fit the model
```

```
LinearRegression()
```

```python
print(hypothesis.coef_) # print coefficients
```

```
[ 4.36693293e-01  9.43577803e-03 -1.07322041e-01  6.45065694e-01
 -3.97638942e-06 -3.78654265e-03 -4.21314378e-01 -4.34513755e-01]
```

Accessing and Exploring Another Built-in Dataset in Python

```python
iris = datasets.load_iris() # assign data set to matrix vector
iris.keys() # view keys
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename',
```

```python
iris['data'] # view data
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
```

```
[4.9, 3.1, 1.5, 0.1],
[5.4, 3.7, 1.5, 0.2],
[4.8, 3.4, 1.6, 0.2],
[4.8, 3. , 1.4, 0.1],
[4.3, 3. , 1.1, 0.1],
[5.8, 4. , 1.2, 0.2],
[5.7, 4.4, 1.5, 0.4],
[5.4, 3.9, 1.3, 0.4],
[5.1, 3.5, 1.4, 0.3],
[5.7, 3.8, 1.7, 0.3],
[5.1, 3.8, 1.5, 0.3],
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 1.9, 0.2],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
```

```
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
```

```
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
```

```
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

```python
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names) # create a df
iris_df['species'] = pd.Categorical.from_codes(iris.target, # add species column
                                              iris.target_names)
print(iris_df.head())
```

```
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  species
0                5.1               3.5  ...               0.2   setosa
1                4.9               3.0  ...               0.2   setosa
2                4.7               3.2  ...               0.2   setosa
3                4.6               3.1  ...               0.2   setosa
4                5.0               3.6  ...               0.2   setosa

[5 rows x 5 columns]
```

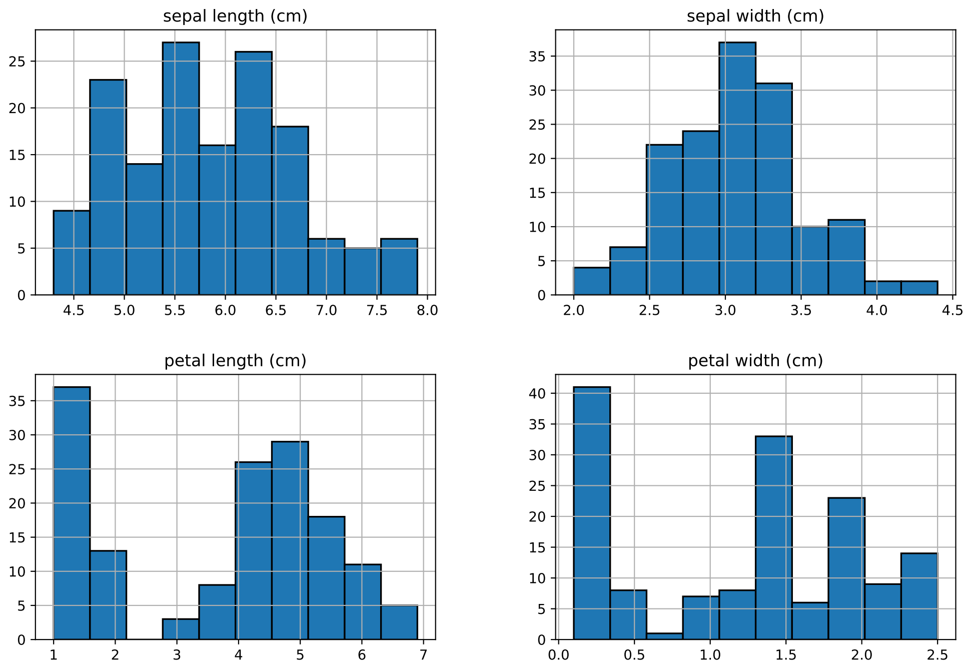
```python
print(iris_df.describe())
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
count         150.000000        150.000000         150.000000        150.000000
mean            5.843333          3.057333           3.758000          1.199333
std             0.828066          0.435866           1.765298          0.762238
min             4.300000          2.000000           1.000000          0.100000
25%             5.100000          2.800000           1.600000          0.300000
50%             5.800000          3.000000           4.350000          1.300000
75%             6.400000          3.300000           5.100000          1.800000
max             7.900000          4.400000           6.900000          2.500000
```

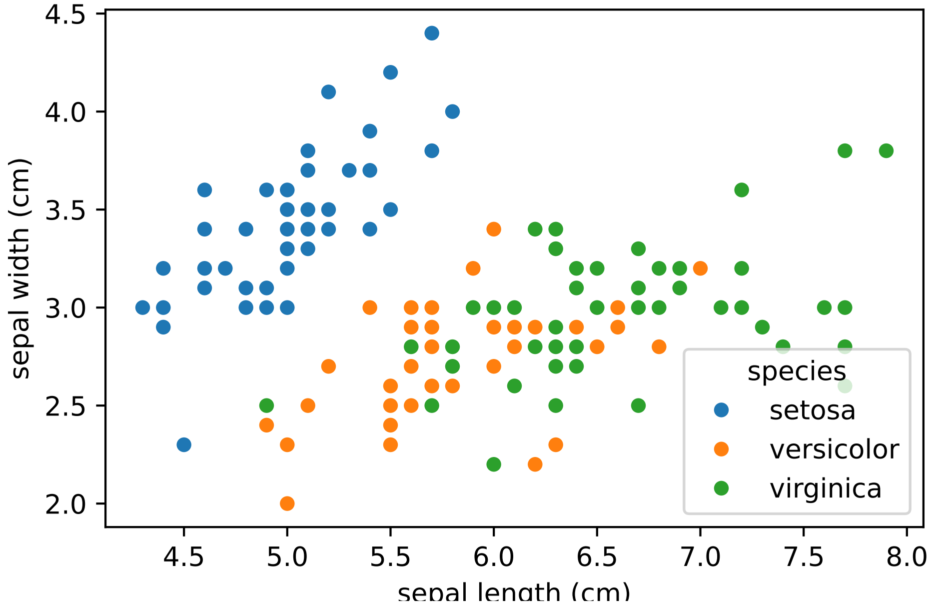
```
iris_df.hist(edgecolor = 'black', linewidth = 1.2, figsize=(12,8)) # histograms
```

```
array([[<Axes: title={'center': 'sepal length (cm)'}>,
        <Axes: title={'center': 'sepal width (cm)'}>],
       [<Axes: title={'center': 'petal length (cm)'}>,
        <Axes: title={'center': 'petal width (cm)'}>]], dtype=object)
```

```
plt.show()
```




```
sns.scatterplot(x='sepal length (cm)', y = 'sepal width (cm)', hue = 'species',
                data = iris_df)
plt.savefig('iris_scatter.png')
plt.show()
```




```
scaler = StandardScaler()
iris_scaled = scaler.fit_transform(iris_df.iloc[:, :-1]) # scale data frame
```

```
X_train, X_test, y_train, y_test = train_test_split(iris_scaled, iris.target,
                                                    test_size = 0.3,
                                                    random_state = 42)
model = LogisticRegression()
model.fit(X_train, y_train) # fit the model
```

```
LogisticRegression()
```

```
y_pred = model.predict(X_test) # test the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 1.00
```