Microsoft

Data Science Essentials

Lab 2 – Working with Summary Statistics

Overview

In this lab, you will learn how to use either R or Python to compute and understand the basics of descriptive statistics. Descriptive statistics aid in understanding a data set. Often your first step after receiving a new data set is to compute, examine and understand simple summary statistics.

What You'll Need

To complete this lab, you will need the following:

- The files for this lab
- An R or Python development environment. The lab steps assume you are using R Studio (for R) or Spyder (for Python).

Note: To set up the required environment for the lab, follow the instructions in the Setup Guide for this course.

Computing and Visualizing Summary Statistics with R

Summary statistics generally include the mean, the median and quartiles of the data. This gives you a first quick look at the distribution of data values. In this exercise, you will compute and interpret the computed summary statistics for the prices of automobiles from the Automotive Price Dataset.

Note: If you prefer to work with Python, skip this exercise and complete the next exercise, Computing and Visualizing Summary Statistics with Python.

Compute Summary Statistics for Price

1. Verify that the directory containing the lab files for this module contains files named **Automobile price data _Raw_.csv** and **autostats.R**; and note the full path to this directory.
2. In RStudio, open **autostats.R** from the directory containing the lab files for this module.
3. Verify that the code in the code editor window contains the following function named **read.auto**:

```
read.auto <- function(path = 'c:/dat203.1x/mod2'){
  ## Read the csv file
  filePath <- file.path(path, 'Automobile price data _Raw_.csv')
  auto.price <- read.csv(filePath, header = TRUE,
                         stringsAsFactors = FALSE)
```

```
  ## Coerce some character columns to numeric
  cols <- c('price', 'bore', 'stroke',
          'horsepower', 'peak.rpm')
  auto.price[, cols] <- lapply(auto.price[, cols], as.numeric)

  ## remove rows with NAs
  auto.price <- auto.price[complete.cases(auto.price), ]

  ## Add a log transformed column for price
  auto.price$lnprice <- log(auto.price$price)

  ## Consolidate the number of cylinders
  auto.price$num.cylinders <-
    ifelse(auto.price$num.of.cylinders %in% c("four", "three"), "three-
four",
           ifelse(auto.price$num.of.cylinders %in% c("five", "six"),
"five-six", "eight-twelve"))

  auto.price
}
```

Examine this code while reading the comments to understand the operation of this function.

4. Click the **Source** icon in the upper right above the code editor window, and in the console window, note that RStudio generates a **source** command to reference the **autostats.R** script file.
5. In the console window type the following code, substituting the path to the lab files directory for this module in place of *PATH-TO-YOUR-LAB-FILES* – for example, *C:/DAT203x/Mod2*.

```
auto.price <- read.auto(path = 'PATH-TO-YOUR-LAB-FILES')
str(auto.price)
```

Note: You can safely ignore warning messages concerning the handling of NA values which may appear.

6. Review the output, which shows the columns in the source data along with the data type and first few values for each column, as shown below:
7. Examine the feature names (column names) in this dataset, noting the various properties of the automobiles that might be useful in predicting the price.
8. In RStudio verify that the **autostats.R** script in the code editor contains the following function named **auto.describe**:

```
auto.describe <- function(df, col){
  tmp <- df[, col]
  sumry <- summary(tmp)
  nms <- names(sumry)
  nms <- c(nms, 'std')
  out <- c(sumry, sd(tmp))
  names(out) <- nms
  out
}
```

9. Review this code and note that it performs the following operations:
    - Computes the summary statistics for the selected column.
    - Computes the standard deviation of the column.

- Concatenates the computed results into a vector.
- Assigns names to the elements of the vector.

10. Type the following line of code in the RStudio console to compute and view summary statistics for the automobile price.

```
auto.describe(auto.price, 'price')
```

11. Review the output, which should look like this:

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.      std
 5118.00  7756.00 10240.00 13250.00 16510.00 45400.00  8056.33
```

12. Note the following properties of these data from the computed summary statistics:
    - The mean is noticeably larger than the median (midpoint or 50% quartile), indicating the distribution is skewed toward large values.
    - The first quartile is closer to the median than the third quartile, adding support to the hypothesis that these distributions are skewed toward large values.
    - The standard deviation is small relative to the range between the minimum and the maximum indicating the distribution has 'long tails'. This is likely to be especially the case toward the larger values.

Visualize Summary Statistics for Price

1. In RStudio, verify that the **autostats.R** script in the code editor contains the following function named **plot.auto**:

```
plot.auto <- function(df = auto.price, val = 'price', num.bin = 30){
   require(ggplot2)
   require(gridExtra)
   dat <- as.factor('')

   ## Compute bin width
   bin.width <- (max(df[, val]) - min(df[, val]))/num.bin

   ## Plot a histogram
   p1 <- ggplot(df, aes_string(val)) +
     geom_histogram(binwidth = bin.width)

   ## A simple boxplot
   p2 <- ggplot(df, aes_string(dat, val)) +
     geom_boxplot() + coord_flip() + ylab('')

   ## Now stack the plots
   grid.arrange(p2, p1, nrow = 2)
}
```

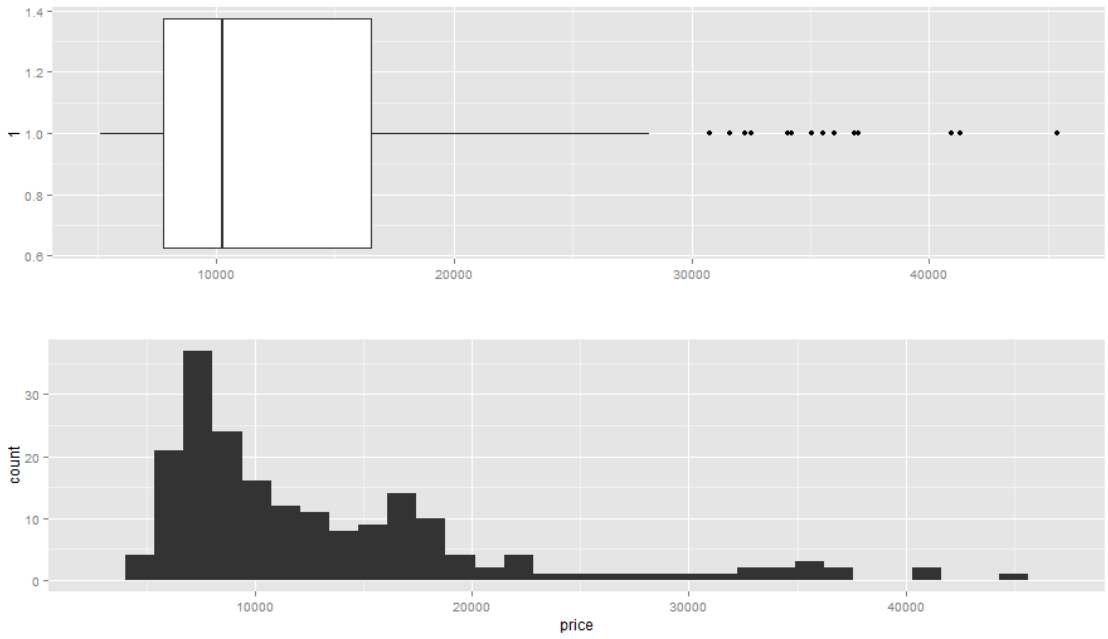
2. Read the code listing above, and note the comments to understand the operations.

Note: Plotting with R is discussed in another module of this course.

3. In the console window, type the following line of code to create plots of the **price** column in the **auto.price** data frame.

```
plot.auto(auto.price, 'price')
```

4. Review the output, which should look similar to this:



Note: The box plot is plotted horizontally so it can be compared directly with the histogram plotted below.

5. Examine these plots and note the following observations, which are consistent with the analysis of the summary statistics in the preceding exercise:
    - The dark line in the box of the box plot is at the median value and is to the right of the mode (the highest peak on the histogram).
    - The box of the box plot contains the middle two quartiles of the data. A considerable number of data values can be seen to the right (high value side) of the box and in the right 'tail' of the histogram.
    - The box plot exhibits a considerably longer whisker and dots showing several outliers on the right side.

Compute and Visualize Summary Statistics for lnprice

The autos dataset also includes a column named **lnprice**, which contains the log of the **price** value. In this procedure you will view statistics for this column.

1. Use the **auto.describe** function to compute summary statistics for the **lnprice** column.
2. Use the **plot.auto** function to create a box chart and histogram for the **lnprice** column.

Computing and Visualizing Summary Statistics with Python

Summary statistics generally include the mean, the median and quartiles of the data. This gives you a first quick look at the distribution of data values. In this exercise, you will compute and interpret the computed summary statistics for the prices of automobiles from the Automotive Price Dataset.

Note: If you prefer to work with Python, skip this exercise and complete the previous exercise, Computing and Visualizing Summary Statistics with R.

Compute Summary Statistics for Price

1. Verify that the directory containing the lab files for this module contains files named **Automobile price data _Raw_.csv** and **autostats.py**; and note the full path to this directory.
2. In Spyder, change the directory of your IPython session by typing the following commands in the IPython console, substituting the path to the lab files directory for this module in place of *PATH-TO-YOUR-LAB-FILES* – for example, *C:/DAT203x/Mod2*.

```
import os
os.chdir('PATH-TO-YOUR-LAB-FILES')
```

3. In Spyder, open **autostats.py** from the directory containing the lab files for this module.
4. Verify that the code in the code editor contains the following function named **read_auto**.

```
def read_auto(pathName = "c:\dat203.1x\mod2", fileName = "Automobile
price data _Raw_.csv"):
    ## Load the data
    import pandas as pd
    import numpy as np
    import os

    ## Read the .csv file
    pathName = pathName
    fileName = fileName
    filePath = os.path.join(pathName, fileName)
    auto_price = pd.read_csv(filePath)

    ## Convert some columns to numerica values
    cols = ['price', 'bore', 'stroke',
          'horsepower', 'peak-rpm']
    auto_price[cols] = auto_price[cols].convert_objects(convert_numeric
= True)

    ## Remove rows with missing values
    auto_price.dropna(axis = 0, inplace = True)

    ## Compute the log of the auto price
    auto_price['lnprice'] = np.log(auto_price.price)

    ## Create a column with new levels for the number of cylinders
    auto_price['num-cylinders'] = ['four-or-less' if x in ['two',
'three', 'four'] else
                                   ('five-six' if x in ['five', 'six']
else
                                   'eight-twelve') for x in
auto_price['num-of-cylinders']]
    return auto_price
```

5. Read this code, note the comments to understand the operations performed.
6. To load the code in the **autostats.py** file and then read the data in the **Automobile price data _Raw_.csv** file as a Pandas data frame, type the following lines in the IPython console:

```
import autostats as astats
auto_price = astats.read_auto('.')
```

Note: Using the Pandas package to manipulate data is discussed in a later module.

7. In the IPython console, enter the following code to display column names and first few data values in the data:

```
auto_price.head()
```

8. Verify that the **autostats.py** script in the code in code editor contains the following function named **auto_describe**:

```
def auto_describe(df, col):
    ## Compute the summary stats
    desc = df[col].describe()

    ## Change the name of the 50% index to median
    idx = desc.index.tolist()
    idx[5] = 'median'
    desc.index = idx
    return desc
```

9. Examine this code, note the comments, to understand the operation.

10. Type the following line of code in the IPython console to compute and view summary statistics for the **price** column in the **auto_price** data frame.

```
astats.auto_describe(auto_price, 'price')
```

11. Review the output, which should look like this:

```
count       195.000000
mean      13248.015385
std        8056.330093
min        5118.000000
25%        7756.500000
median    10245.000000
75%       16509.000000
max       45400.000000
Name: price, dtype: float64
```

12. Note the following properties of these data from the computed summary statistics:
    - The mean is noticeably larger than the median (midpoint or 50% quartile), indicating the distribution is skewed toward large values.
    - The first quartile is closer to the median than the third quartile adding confirmation to the hypothesis that these distributions are skewed toward large values.
    - The standard deviation is small relative to the range between the minimum and the maximum indicating the distribution has 'long tails'. This is likely to be especially the case toward the larger values.

Visualize Summary Statistics for Price

1. In Spyder, verify that the **autostats.py** script in the code editor contains the following function named **plot_auto**:

```
def plot_auto(df, col):
    import matplotlib.pyplot as plt
```

```
    ## Setup for ploting two charts one over the other
    fig, ax = plt.subplots(2, 1, figsize = (12,8))

    ## First a box plot
    df.dropna().boxplot(col, ax = ax[0], vert=False,
return_type='dict')

    ## Plot the histogram
    temp = df[col].as_matrix()
    ax[1].hist(temp, bins = 30, alpha = 0.7)
    plt.ylabel('Number of Cars')
    plt.xlabel(col)
    return [col]
```

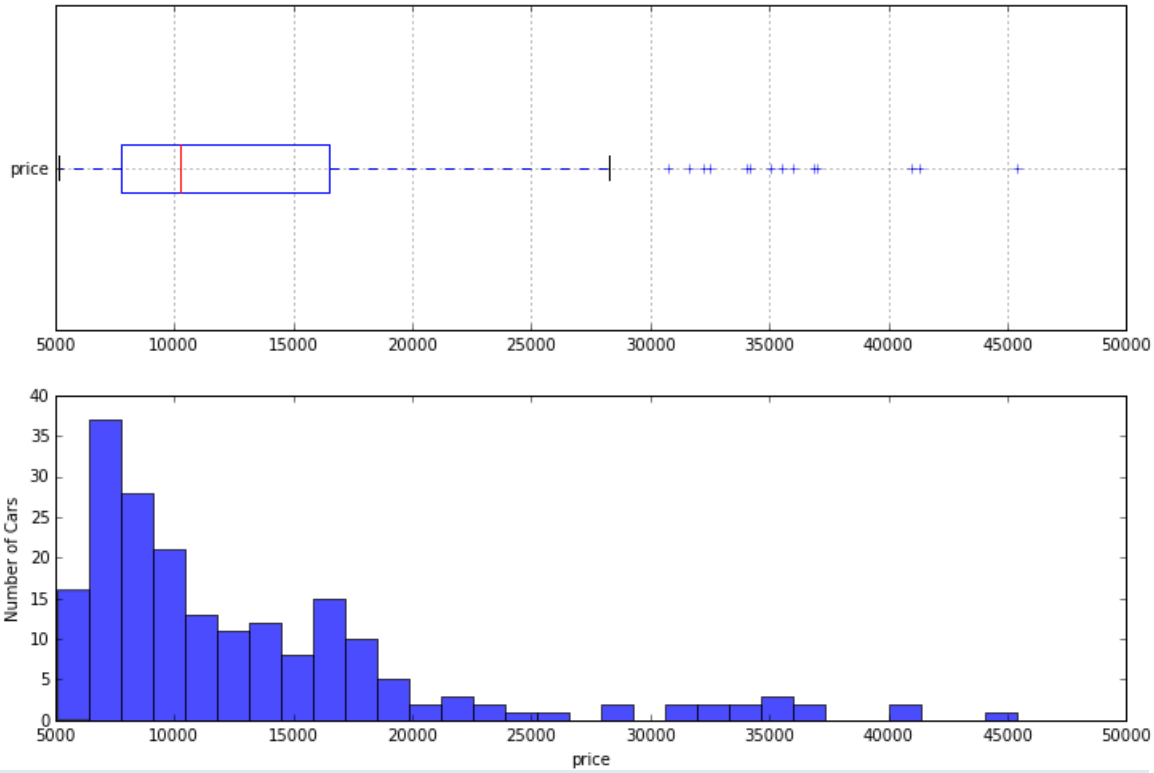
2. Read the code listing above, and note the comments to understand the operations.

Note: Plotting with Python is discussed in another module of this course.

3. At the IPython prompt, type the following line of code to create plots of the **price** column.

```
astats.plot_auto(auto_price, 'price')
```

4. Review the output, which should look similar to this:



Note: The box plot is plotted horizontally so it can be compared directly with the histogram plotted below.

5. Examine these plots and note the following observations which are consistent with the analysis of the summary statistics in the preceding exercise:

- The red line in in the box of the box plot is at the median value and is to the right of the mode (the highest peak on the histogram).
- The box of the box plot contains the middle two quartiles of the data. A considerable number of data values can be seen to the right (high value side) of the box and in the right 'tail' of the histogram.
- The box plot exhibits a considerably longer whisker and '+' symbols show several outliers on the right side.

Compute and Visualize Summary Statistics for lnprice

The autos dataset also includes a column named **lnprice**, which contains the log of the **price** value. In this procedure you will view statistics for this column.

1. Use the **auto_describe** function to compute summary statistics for the **lnprice** column.
2. Use the **plot_auto** function to create a box chart and histogram for the **lnprice** column.

Summary

This lab you have used either R or Python to compute and examine some descriptive statistics for the automobile price data set. You also created some simple exploratory charts to aid in understanding this dataset. You will apply this simple, yet useful, techniques many times in your career as a data scientist.