HCAP FYSP GGPLOT/Data Analysis PDF

Brian Kim

17/07/2020

Portions of this workshop were drawn from Preceptor David Kane (Gov50) and Prof. Kevin Rader (STAT139) of Harvard University.

Workshop Initial Instructions

In this workshop we will use the 'sample_excel.csv' data set to perform simple linear regressions.

variables inside the dataset to note:

- `price`: the selling price of the home, in US $.
- `sqft`: the living area of the home as measured by floor space, in square feet
- `beds`: the number of bedrooms in the property.
- `baths`: the number of bathrooms in the property.
- `lotsize`: the size of the parcel lot, in square feet.
- `year`: the year the house was built.

Initial Setup

```
knitr::opts_chunk$set(echo = TRUE)

# This is where you do initial setups, such as installing packagesloading
# libraries which allow for us to do a lot of things in R. For the purposes of
# this activity, we will be learning how to make data visualizations and data
# analysis.

# Let's load the sample_excel csv. We can name this anything we want, I will
# name it sample_excel. The excel file should be in the same folder as your Rmd.

sample_excel <- read.csv("sample_excel.csv")

# To begin, we are going to install a package called "ggplot2" to make data
# visualizations. "Tidyverse" will be installed as well to clean up data.
# "Dplyr" will be used to looking at data on R.
```

```
install.packages("ggplot2")
install.packages("tidyverse")
install.packages("dplyr")
```

Once you have those packages installed, you don't really have to install them again.

Once things are successfully installed, we will then load all those libraries so we can use it.

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.3

## -- Conflicts ------------------------------------------------------------------------------------ tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
```

By the way, to actually run these codes, you need to have them in what's called a code chunk. Any gray area is considered a code chunk, and making them are very simple.

If I were to leave the code like `library(ggplot2)`, this wouldn't run. I'm doing this to show the text itself, but not necessarily to execute on that code.

Let's play with the data! Viewing and filtering

```
# We loaded up the dataset, we are going to use correlation to see how things
# fit into each other. But before, we can look at the dataset by clicking on the
# data file in the environment tab. We can also use the following command to
# view our dataset:
```

```
View(sample_excel)
```

Per Yinyu's request, we want to see the first 5 lines of the dataset so we know that this isn't faulty. We can do it by doing this:

```
head(sample_excel)
```

Let's begin our correlation. Let's just play around with the correlation, using the cor command. We can subset the data by making a new dataset altogether. For the purposes of this workshop, we will only subset for columns that are numeric.

```
data2 <- sample_excel %>%
  select_if(is.numeric)

# Let's view the subsetted dataset. There's two columns 'lotsize' and 'hoa' with
# NA values, and we do not want to include them because this could hinder what
# we want to do in the end.

# We could filter out all the NA observations in the dataset.
```

```r
data_na_play <- data2 %>%
  filter(!is.na(lotsize)) %>%
  filter(!is.na(hoa))

# There's only 11 observations with all the values intact, which isn't a lot. So
# what else can we do?

# We can remove the select columns.

data_without_na <- data2 %>%
  select(!lotsize) %>%
  select(!hoa)
```

Let's try doing some regressions now.

```r
# Let's run the correlation of the entire dataset and see what happens. This
# shows you correlation coefficients of how each variable would run against the
# other.

result <- cor(data_without_na)
```

```r
View(result)
```

```r
# So how about if we want to look at individual regressions between different
# variables? What can we do?

# We can try to print out specific correlation coefficients of different
# relationships.

data_without_na %>%
  summarise("Correlation Coefficient" = cor(sqft, price)) %>%
  print()
```

```
##   Correlation Coefficient
## 1               0.9257261
```

```r
# Let's try to change up the data. Let's say if your house was built before
# 2000, the value automatically increases by 100000 in USD. Sounds like a great
# proposition, right?

# So what can we do? Let's make a specific column that does this.

modified_data <- data_without_na %>%
  mutate(newprice = ifelse(year < 2000, price + 100000, price))

# The syntax is - I want to 'mutate' the original dataset by making a new column
# 'newprice', and the conditions are IF the year < 2000, then add 100000 to
# price and if not keep it as price value
```

```r
# while we're at it, let's make a categorical variable that shows which houses
# are made before 2000 and after.
modified_data <- modified_data%>%
  mutate(whenmade = ifelse(year < 2000, "Old", "New"))

# We will remember that 0 means old, 1 is new. And just in case this thinks it's
# a numerical variable, we will categorize it as a factor (either yes or no).



# Fun stuff. Does this impact correlation coefficient between sqft and newprice?

modified_data %>%
  summarise("Correlation coeffieicent" = cor(sqft, newprice)) %>%
  print()
```

```
##   Correlation coeffieicent
## 1                0.9264549
```

```r
# Slightly weakens it, but still there!
```

How would this work in a graph?

We are going to use the ggplot package for this portion.

```r
# Let's say you wanted to graph the relationship between new price and square
# footage.

# We probably want to see a scatterplot to show individual datapoints, correct?
# So let's do that.

sampleplot <- ggplot(modified_data, aes(x = sqft, y = newprice, color = whenmade)) +

  # To specify which kind of graph we want, we use geom_ - for the purpose of
  # this, we will be using geom_point

  geom_point() +

  # This is where we will insert a line of best fit to see how the relationship
  # looks like.

  geom_smooth(formula = y ~ x,
              method = "lm", se = FALSE, color = "black") +

  # label

  labs(title = "Relationship Between Sq. Ft and Price",
       subtitle = "100000 added to New Homes",
       y = "Price", x = "Sq. Footage", color = "Year Made") +

  # scale the color manually
```

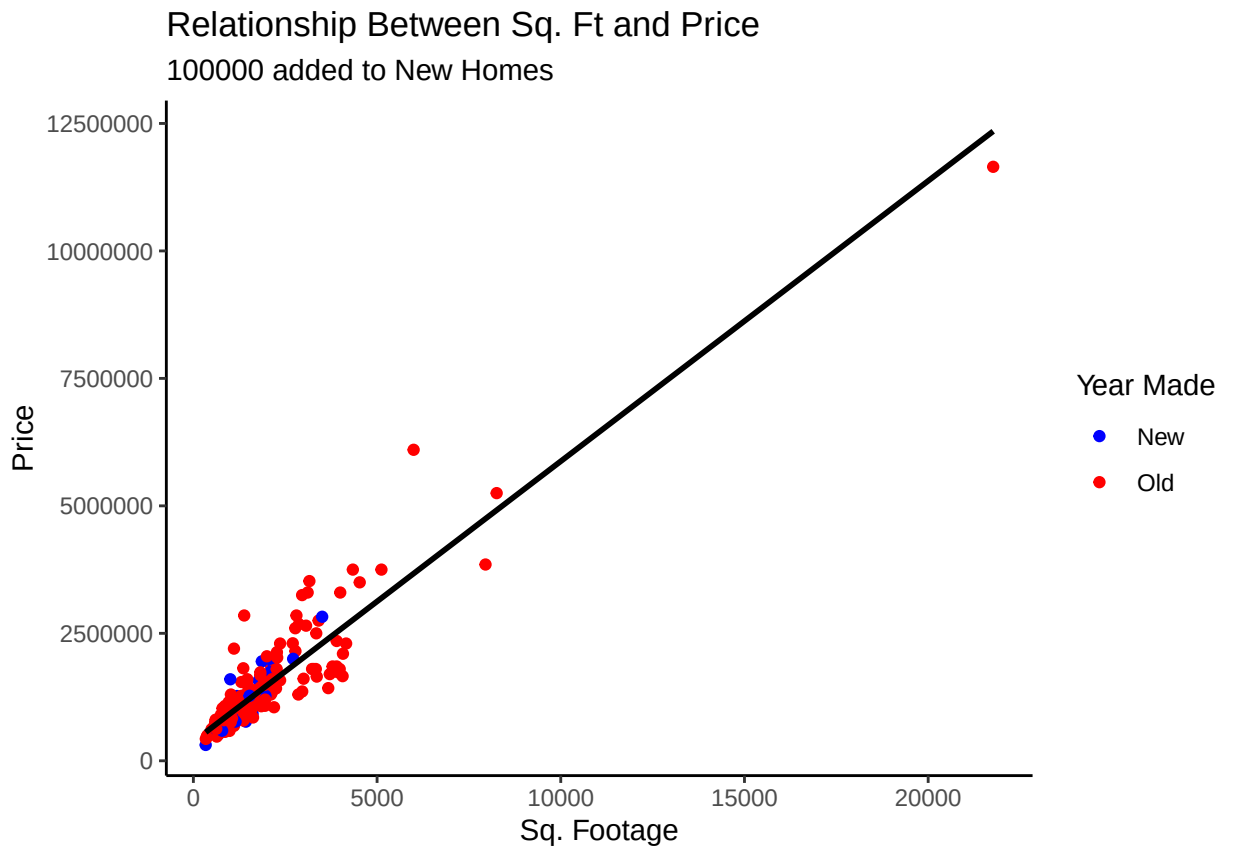
```
  scale_color_manual(values = c("blue", "red")) +

  # make it theme_classic (lots of themes if you search online)

  theme_classic()


# print the plot

sampleplot
```



Hmm, it seems like we have an outlier. We can subset the data to push the outlier out by manually setting the conditions.

```
# There's a chance that the numbers could come out in Scientific notation. What
# we can do is this:

# disable scientific notation
options(scipen = 999)

# moving onto our graph

sampleplot2 <- modified_data %>%
  subset(sqft < 5500) %>%
```

```r
  ggplot(aes(x = sqft, y = newprice, color = whenmade)) +

  # To specify which kind of graph we want, we use geom_ - for the purpose of
  # this, we will be using geom_point

  geom_point() +

  # This is where we will insert a line of best fit to see how the relationship
  # looks like.

  geom_smooth(formula = y ~ x,
              method = "lm", se = FALSE, color = "black") +

  # label

  labs(title = "Relationship Between Sq. Ft and Price",
       subtitle = "100000 added to New Homes",
       y = "Price", x = "Sq. Footage", color = "Year Made") +

  # scale the color manually

  scale_color_manual(values = c("red", "blue")) +

  # change x scale

  scale_x_continuous(
    breaks = c(0, 2500, 5000),
    labels = c("0", "2,5K", "5K")
    )+

  # make it theme_classic (lots of themes if you search online)

  theme_classic()



# print the plot

sampleplot2
```

And that is how we print out graphs! Onto our process of Shiny.