Department of Economics — Applied Econometrics and Data Analysis
Yale University
Prof. Edward Vytlacil — August 2020

Data Visualization in R

1 Data Visualization in R

One of R's strengths is it's powerful visualization ability. It provides both high and low level plotting commands and allows the proficient user to edit even the smallest detail of the graphs with relatively little code. In the context of data analysis, visualization techniques play an important role in the exploration of data sets, especially in detecting relationships between two or more variables. Moreover, thoughtful graphs can be powerful tools to communicate information. For these reasons, we encourage you to carefully study these notes and to take the visualization tasks on the problem sets as serious as any other question.

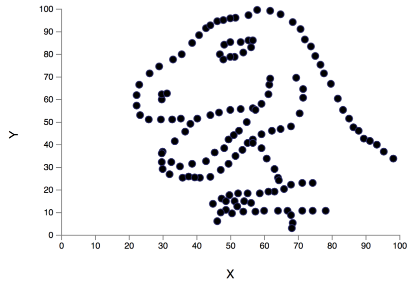


Figure 1: Visualize your data.

Graphs in R can be created using either the built-in base graphic functions or by using the functions of a graphing package, the most prominent being `ggplot2`. While the base graphic functions are in principle sufficient for our purposes, they do not seem to follow a consistent syntax and make the manipulation of details fairly difficult for beginners. As a consequence, we decided to teach data visualization using the `ggplot2` package. As you will see, the package is powerful in the sense that it enables you to produce sophisticated graphs with just a few lines of code. However, please note that you are free to submit Problem Set solutions using base graphic commands and output.

2 Base Graphics vs. ggplot2

There are two important differences between `ggplot` and base graphics. First, `ggplot` operates using data frames while base graphics uses vectors. Second, `ggplot` allows the user to add additional layers to an existing ggplot, something that is not generally possible for base graphic commands. That being said, it is not true that one strictly dominates the other. However, as mentioned above, one distinct advantage of `ggplot` is its consistent syntax due to the underlying *"Grammar of Graphics"* (Wilkinson, 2005). A fair description of the debate seems to be that base graphs is best for simple graphs, while ggplot, mostly because of its nicer default configuration, is better for complex graphs.

3 Understanding the ggplot Syntax

The syntax for creating graphs with ggplot can be confusing without some explanations. The basic idea of the *"Grammar of Graphics"* that underlies ggplot2 is to have a structure that allows you to independently specifiy building blocks of a plot and combine them to create any type of graph that you want. The most important building blocks are:

1. The data (`data`)
2. The geometric object (`geom`)
3. The aesthetic mapping (`aes`)

Unsurprisingly, the *data* block (1) specifies the dataframe that contains the data you want to visualize. *Geometric objects* (2) are the visual representations of the data. Examples include:

1. Data points (`geom_point`)
2. Lines (`geom_line`)
3. Histograms (`geom_hist`)
4. Densities (`geom_density`)

To get a list of all available geometric objects type:

```
help.search("geom_", package = "ggplot2")
```

While there is no upper limit, a plot must have at least one geom. You can add a geom to an existing ggplot using the `+` operator. You can thus plot multiple density plots in one figure, have a figure that contains both a histogram and a density plot, etc., by including multiple geoms in one ggplot. The end of this document contains a number of examples.

Aesthetic mappings (3) describe how variables in the data are mapped to visual properties (aesthetics) of geoms. Examples include the color of the graph, the color, shape and size of displayed objects, line types etc. Aesthetic mappings are set with the `aes()` function. Note that each `geom` only accepts a subset of all aesthetics. For example, `geom_point` accepts (among others) aesthethic mappings regarding the displayed data points, while `geom_hist` accepts (among others) aesthethic mappings regarding the size of the bins. Aesthetic mappings can be set in both, ggplot (at the level of the plot) and in individual layers.

If you are confused at this point, please don't worry. The best way to understand the data visualization techniques that are implemented in `ggplot2` is by examples as well as trial and error. The next subsections will provide examples that you will find useful for the Labs and Problem Sets. Please note that there are many more ways to visualize data and customize ggplots than we can discuss in this introduction. If you want to get an impression of the flexibility that `ggplot2` offers, take a look at the following link. The end of this document contains a number of examples.

3.1 A Simple Scatterplot

Let's start by initializing an empty ggplot based on the Chetty et. al (2017) data. You can download the data here.

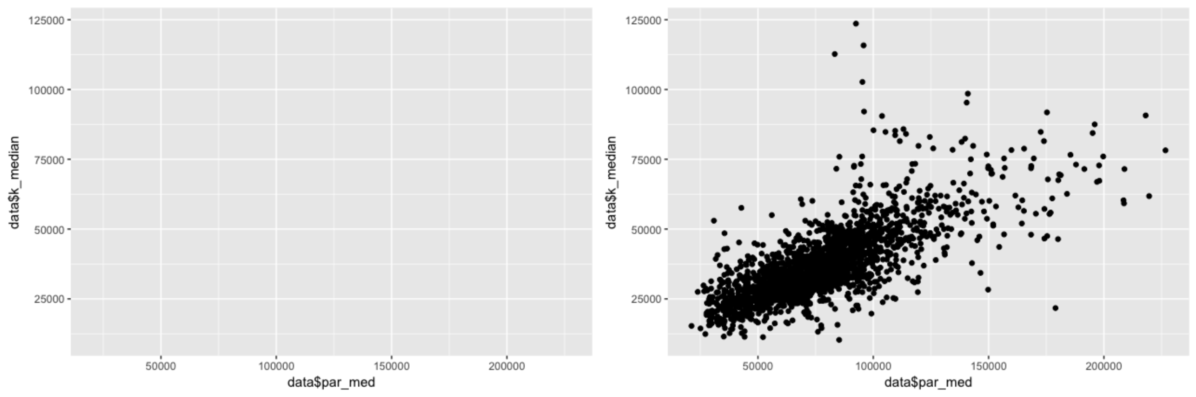
```
library(readstata13)    # Package to import .dta data
library(ggplot2)        # ggplot2

data <- read.dta13("/../mrc_table2.dta")    #Import Data
ggplot(data, aes(x=data$par_med, y=data$k_median)) # Figure (a)
```

The code above produces panel (a) of Figure 2. Note that we have set the aesthethic mapping at the level of the plot by specifying the x and y variable. However, an empty plot is generated since we have not yet specified the type of geometric object that we want to generate.

Suppose we want to get a first impression of the relationship between median parent and child income at the college level. The first step to do so visually is to create a scatterplot. This can be done by adding the appropriate geom to the ggplot using the `+` operator and the `geom_point()` function.

```
ggplot(data, aes(x=data$par_med, y=data$k_median)) + geom_point() # Figure (b)
```



(a) A basic ggplot (b) Adding a Scatterplot Layer

Figure 2: Adding a geom to a ggplot

We now got a basic scatterplot (Panel (b) of Figure 2), where each point represents a college in the median parent income – median child income space. However, it lacks basic features such as a title and axis labels. We can improve the plot by adding layers that contain these features, using the `xlab`, `ylab` and `ggtitle` functions. Note that these are only three of many options that can be used to customize the looks of your graph.

The code below adds axis labels and a title to panel (b) of Figure (2) and generates panel (a) of Figure 3.

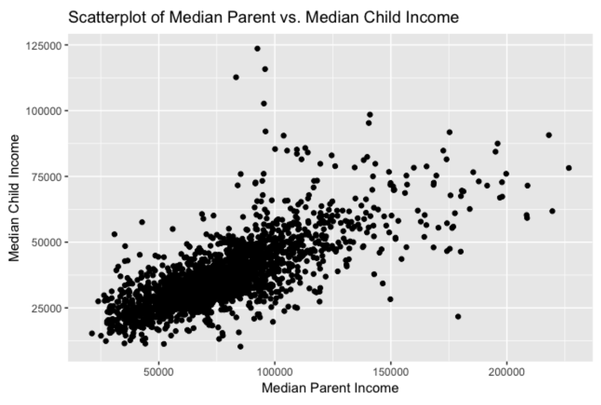
```
ggplot(data, aes(x=data$par_med, y=data$k_median)) + geom_point() +
xlab("Median Parent Income") +
ylab("Median Child Income") +
ggtitle("Scatterplot of Median Parent vs. Median Child Income")
```

Recall that we said earlier

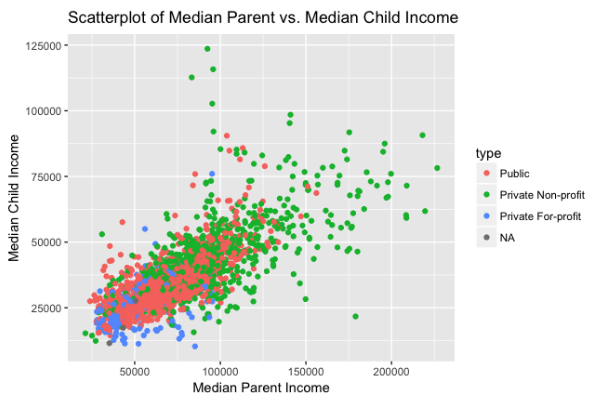
1. *Aesthetic mappings can be set in both, ggplot (at the level of the plot) and in individual layers,* and that
2. *...each `geom` only accepts a subset of all aesthetics.*

To see what these statements mean in practise suppose that you want to color each dot in the scatterplot by the type of college that it represents. Such an exercise is often helpful to test the empirical content of simple hypotheses that explain the patterns in the raw scatterplot. The code below does so by providing the aesthethic mapping `col=type` to the geom `geom_point`.

```
ggplot(data, aes(x=data$par_med, y=data$k_median)) + geom_point(aes(col=type)) +
  xlab("Median Parent Income") +
  ylab("Median Child Income") +
  ggtitle("Scatterplot of Median Parent vs. Median Child Income")
```



(a) Adding Titles and Legends



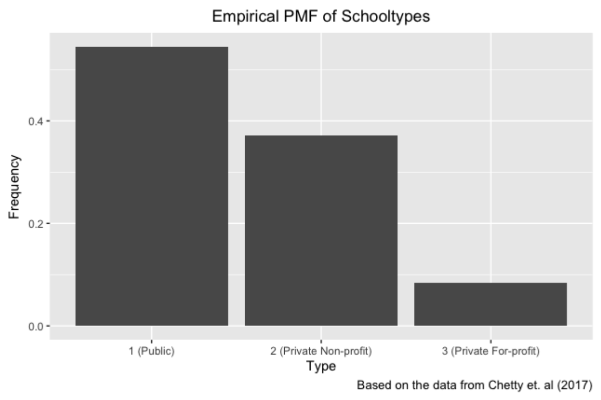
(b) Coloring by Types

Figure 3: Adding Layers and Aesthetic mappings

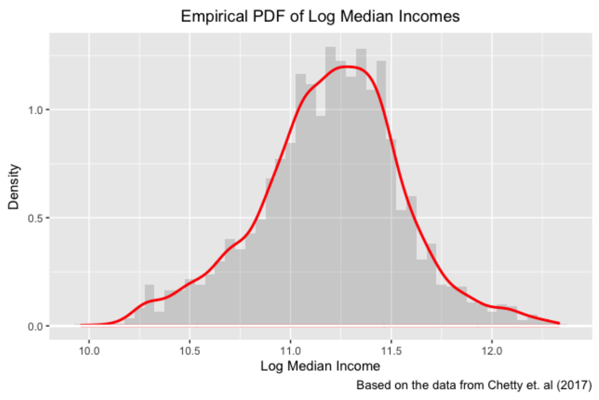
This is just one example of how you can customize a basic ggplot using additional layers and aesthethic mappings on different levels of the plot. Note that, as we pointed out in point 2 above, the set of feasible aesthetics depends on the respective geom. In the next subsections we will use our knowledge to demonstrate how to create different types of plots that you will be required to generate in the Labs and Problem Sets.

3.2 Histograms

Histograms are used to visualize the distribution of numerical data. A histogram divides the range of values into a series of intervals and displays the frequency of observations in each interval. The intervals are usually of equal length. You can think of histograms as estimates of PMFs or approximations of PDFs depending on the type of variable whose distribution is visualized.



(a) Histogram



(b) Histogram with Density Estimate Overlay

Figure 4: Histograms for Discrete and Continuous Variables

3.2.1 For Discrete Data

The appropriate geom for histograms of discrete data is `geom_bar`. The code below produces panel (a) of Figure 4. Note that in order to display frequencies instead of counts you have to provide the aesthethic `aes(y = (..count..)/sum(..count..))` to the geom. The `scale_x_discrete` command allows us to modify the values that are displayed on the x-axis.

```
ggplot(data.frame(na.omit(data)), aes(x=type)) + geom_bar(aes(y = (..count..)/sum(..
    count..))) +
  scale_x_discrete(labels=c("Public" = "1 (Public)", "Private Non-profit" = "2 (Private
      Non-profit)", "Private For-profit" = "3 (Private For-profit)")) +
  ylab("Frequency") +
  xlab("Type")  +
  ggtitle("Empirical PMF of Schooltypes") +
  labs(caption = "Based on the data from Chetty et. al (2017)")
```

3.2.2 For Continuous Data

In the continuous case the approriate geom is `geom_histogram`. Note the `binwidth` option provided to the geom. The value of this option determines the length of the intervals and can have a strong impact on the resulting graph. Panel (b) of Figure 4 produces a histogram of the natural logarithm of median parent income at the college level. The red function is a density estimate, a more approriate way to visualize PDFs that is presented in the next subsection.

```
ggplot(df, aes(lmi)) +
  geom_histogram(alpha=0.2,binwidth=0.05,aes(y = ..density..)) +
  geom_density(col = "red",lwd=1) +
  geom_hline(yintercept=0, colour="white", size=1)+
  ylab("Density")  +
  xlab("Log Median Income")  +
  ggtitle("Empirical PDF of Log Median Incomes") +
  labs(caption = "Based on the data from Chetty et. al (2017)")
```

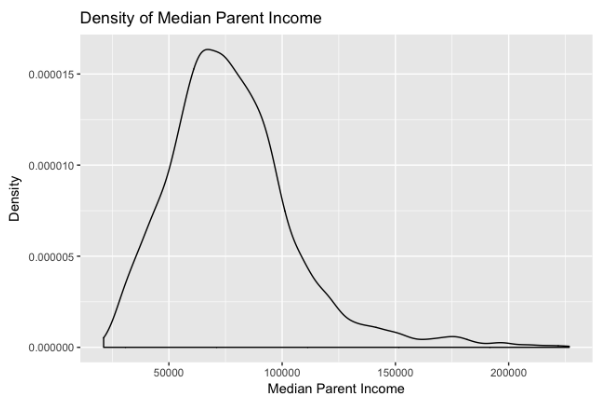
3.3 Density Plots

Density plots can be created using the `geom_density` function. Suppose we are interested in the distribution of median parent income. The first block of code below produces a simple graph of the estimate of the PDF displayed in panel (a) of Figure 5. Now suppose you are interested in comparing the distribution of median parent income across different type of colleges. This can be done by adding the `aes(fill=type)` option to the `geom_density()` layer. As you can see in panel (b) of Figure 5, `ggplot2` then assigns colors to each of the type categories and plots the density seperately for each type. The `alpha` option in `geom_density` regulates the transparency of the colored areas.

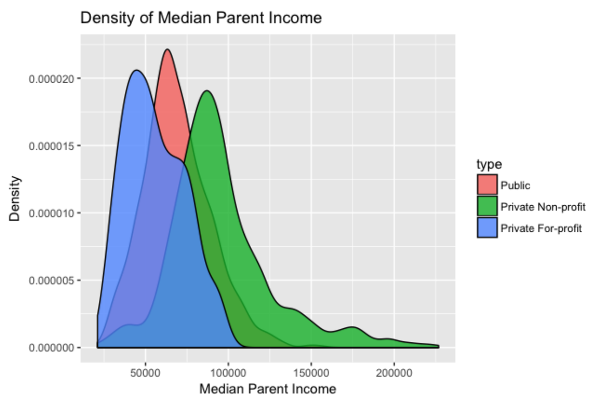
```
#1
na.omit.data <- subset(data,!is.na(data$type)) # Drop rows where type is missing

ggplot(data=na.omit.data,aes(x=na.omit.data$par_med)) +
           geom_density() +
           xlab("Median Parent Income") +
           ylab("Density") +
           ggtitle("Density of Median Parent Income")

#2
ggplot(data=na.omit.data,aes(x=na.omit.data$par_med)) +
           geom_density(aes(fill=type), alpha=0.8) +
           xlab("Median Parent Income") +
           ylab("Density") +
           ggtitle("Density of Median Parent Income")
```



(a) Density



(b) Conditional Densities

Figure 5: Plotting PDFs using `geom_density()`

3.4 CDFs

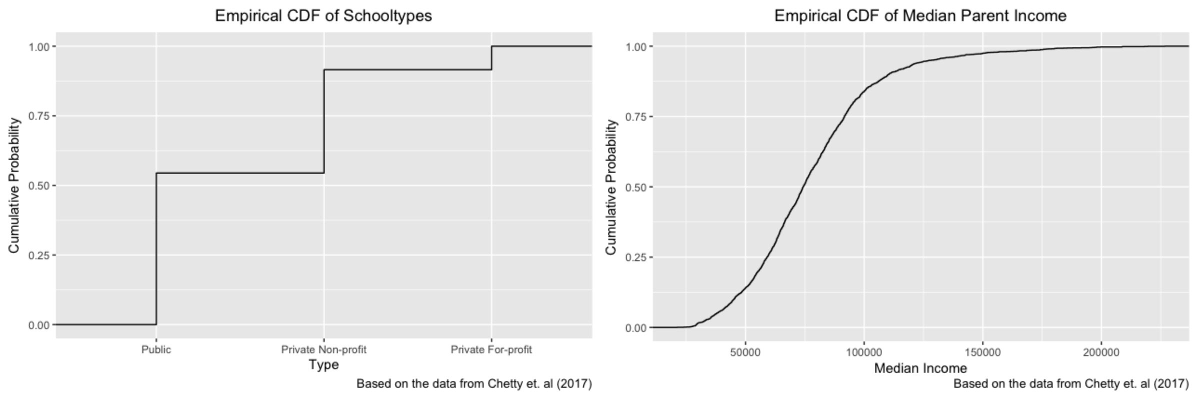
Empirical CDFs can be visualized using the `stat_ecdf` command, which handles both continuous and categorical variables. For example, the code below produces the two example empirical CDF graphs from Lecture 6.

```
# CDF - Type

ggplot(na.omit(data), aes(as.numeric(type))) + stat_ecdf(geom = "step") +
scale_x_discrete(name= "Type", limits= c("1","2","3"),
labels=c("1" = "Public", "2" = "Private Non-profit", "3" = "Private For-profit")) +
ylab("Cumulative Probability")  +
ggtitle("Empirical CDF of Schooltypes",subtitle = NULL) +
labs(caption = "Based on the data from Chetty et. al (2017)")

# CDF - Median Income

ggplot(data, aes(data$par_med)) + stat_ecdf(geom = "step") +
ggtitle("Empirical CDF of Median Parent Income",subtitle = NULL) +
ylab("Cumulative Probability")  +
xlab("Median Income") +
labs(caption = "Based on the data from Chetty et. al (2017)")
```



(a) Schooltypes (b) Median Parent Income

Figure 6: Plotting CDFs using `stat_ecdf()`

Note that the first lines of code are sufficient to create the basic graphs. All other commands, most of whom should look familiar to you by now, add features that make the graphs look better. As in the histogram example, the `scale_x_discrete()` line modifies the x-axis labels of the "Schooltypes" CDF, while the `labs()` lines add the captions at the bottom right of the graph.

```r

# if not installed, you will need to install the ggplot2 package which you can install
    by running the commented out line below. You only need to install it once.
# install.packages("ggplot2")

# iris is data.frame in R, type help(iris) in R for more information on data frame.

# loading ggplot2
library(ggplot2)

# A simple histogram with ggplot2
ggplot(data = iris, aes(x = Sepal.Length)) +
    geom_histogram( bins = 10, fill = "blue", color = "black") +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot.png")

# A simple ecdf plot with ggplot2
ggplot(data = iris,  aes(x = Sepal.Length)) +
    stat_ecdf(, color = "black") +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot2.png")

# histogram by Species on the same plot (blending overlapping colors)
ggplot(data = iris, aes(x = Sepal.Length, fill = Species)) +
    geom_histogram(position="identity", alpha = .5, color = "black") +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot3.png")

# ECDF by Species on the same plot (blending overlapping colors)
ggplot(data = iris, aes(x = Sepal.Length, color = Species)) +
    stat_ecdf() +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot4.png")

# histogram by Species on different plots (but same x and y axis)
ggplot(data = iris, aes(x = Sepal.Length, fill = Species)) +
    geom_histogram(  color = "black") +
    facet_wrap(~ Species) +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot5.png")

# ECDF  by Species on different plots (but same x and y axis)
ggplot(data = iris, aes(x = Sepal.Length, color = Species)) +
    stat_ecdf() +
    facet_wrap(~ Species) +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot6.png")


# Smoothed density by Species on the same plot (blending overlapping colors)
ggplot(data = iris, aes(x = Sepal.Length, fill = Species)) +
    geom_density( alpha = .5, color = "black") +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot7.png")

# Histogram of two different variables on the same plot
ggplot(data = iris) +
    geom_histogram(aes(x = Sepal.Length, fill = "blue"), alpha = .4, color = "black") +
    geom_histogram(aes(x = Petal.Length, fill = "red"), alpha = .4, color = "black") +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot8.png")

# Smoothed density of two different variables on the same plot
ggplot(data = iris) +
    geom_density(aes(x = Sepal.Length, fill = "blue"), alpha = .4, color = "black") +
    geom_density(aes(x = Petal.Length, fill = "red"), alpha = .4, color = "black") +
    theme_bw() +
    xlab("X axis label") +
    ylab("Y axis label") +
    ggtitle("Plot title") +
    ggsave("exampleplot9.png")
```