poster

Zheng Wu

10/23/2017

Content

1.Introduction

We have learned and practiced the package “ggplot2” for a while in the lectures and labs. However, it occurred to me all the time that even though I succeeded getting the graphs I wanted everytime, I didn’t fully understand all the arguments and functions in each paranthesis. What’s more, the sample graph provided in the instruction always looked better than mine, which was frustrating.

Data visualization, as one of the major parts of data science, is as important as analyzing data per se. A good-looking graph will not only make the interface more beautiful, but will also increase the capability of the data to speak to the audience. Therefore, to have a well designed graph is never unnecessary.

With the above consideration, I started my research online to look for more supplemental knowledge of using the ggplot2 package. My first finding was a detailed itemized list of the basics of ggplot2. Based on this outline, I set my foot further by learning a little bit more about every aspects such as layers, legends, colours and so on. Lastly, I included several splendid graphs and their codes found online, hoping this could be an opportunity for us to appreciate the powerful function of graphs.

I hope you enjoy!

packages used in this Rmd

```
library(ggplot2)
```

2.ggplot

The package “ggplot2” is developed by Hadley Wickham for the programming language R. It implements the basis of “Grammar of Graphics”, which interprets the components of a graph into understandable elements, just as if making a sentence with linguistic grammar. The most outstanding feature ggplot2 employs that distinguishes it from other graph generator packages is its principle of addition of layers. By adding different layers, ggplot2 presents graphs that is capable of utilizing various grphic elements and displaying high-dimensional information.

One primary source that I found, an introductory walkthrough of ggplot2, outlines the construction of a ggplot function in the following:

> 1. data
> 2. mapping
> 3. geom
> 4. stats
> 5. scale
> 6. coord
> 7. facet
> 8. theme
> 9. storage and output

In the above sections, data is the source of input of ggplot2, while mapping and the following sections detail the establishment and manipulation of graphs, and some useful elements that better the presentation of the graphs with aesthetic concerns. The last section talks about storing the output of the plots.

In the rest of this part, we will go into some of the aspects and have a sense of how they function and corporate into the development of a graph. Considering that we have already learned ggplot2 in class, this post is not targeted to serve as a tutorial, but a notebook recording interesting tricks, confusing concepts and significant techniques. We are goint to use the embedded dataset “mtcars” as an example for demonstration. A summary and a glance of this “mtcars” is displayed below:

```
# attributes of mtcars
attributes(mtcars)
```

```
## $names
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
##
## $row.names
##  [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
##  [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
##  [7] "Duster 360"          "Merc 240D"           "Merc 230"
## [10] "Merc 280"            "Merc 280C"           "Merc 450SE"
## [13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetwood"
## [16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
## [19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
## [22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
## [25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
## [28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
## [31] "Maserati Bora"       "Volvo 142E"
##
## $class
## [1] "data.frame"
```

```
# a glance of the first three rows
head(mtcars, n = 3)
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

data

In ggplot2, the input dataset has to be in the type of dataframes(data.frame). This enables easier storage and operation on the input data for ggplot2.

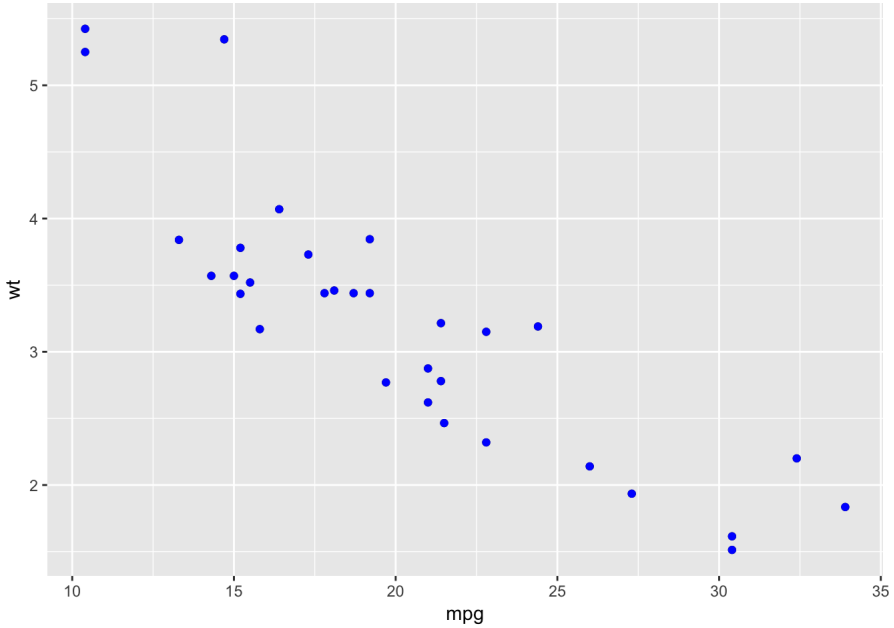
Alternation to the dataset can be made by using '%+%'. By doing this, we can preserve and use the ggplot function repeatatively. For example:

```
#the plot is specified using "+" a layer
plot1 <- ggplot(data = mtcars, aes(x = mpg, y = wt)) +
  geom_point()

#the dataset is replaced with "%+%"
plot2 <- ggplot(data = mtcars, aes(x = mpg, y = wt)) +
  geom_point()
plot2 <- plot2 %+% transform(mtcars, mpg = mpg^2)
```

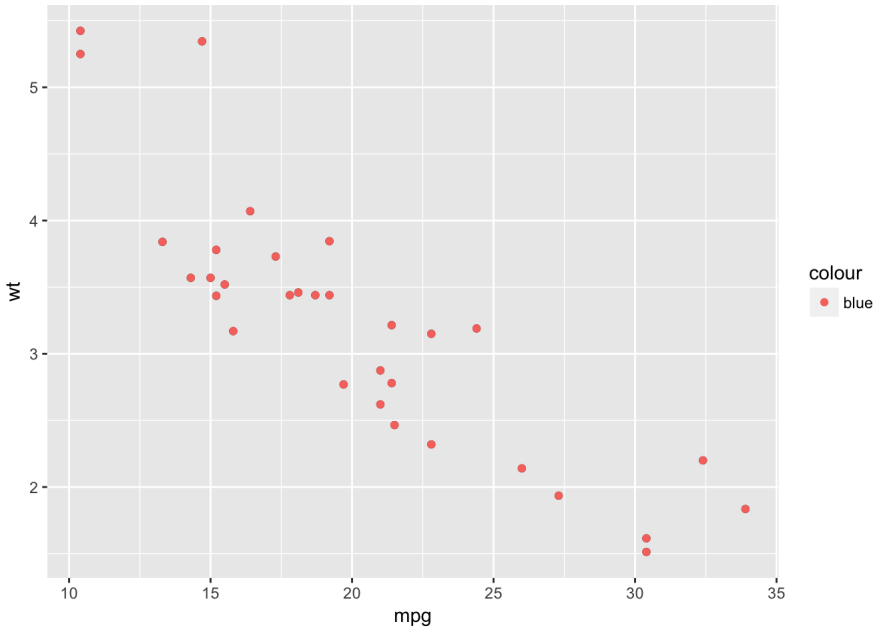
mapping

Mapping is the corelation between datas and conresponding graphic factors. The function 'aes()' establishes the mapping rules of the data from the dataset to corresponding graphic factors. This may sound tedious, however, the point is better illustrated here:

```
# original plot
plot1 <- ggplot(data = mtcars, aes(x = mpg, y = wt)) +
  geom_point()
# standard blue point plot
plot1 + geom_point(color = "blue")
```



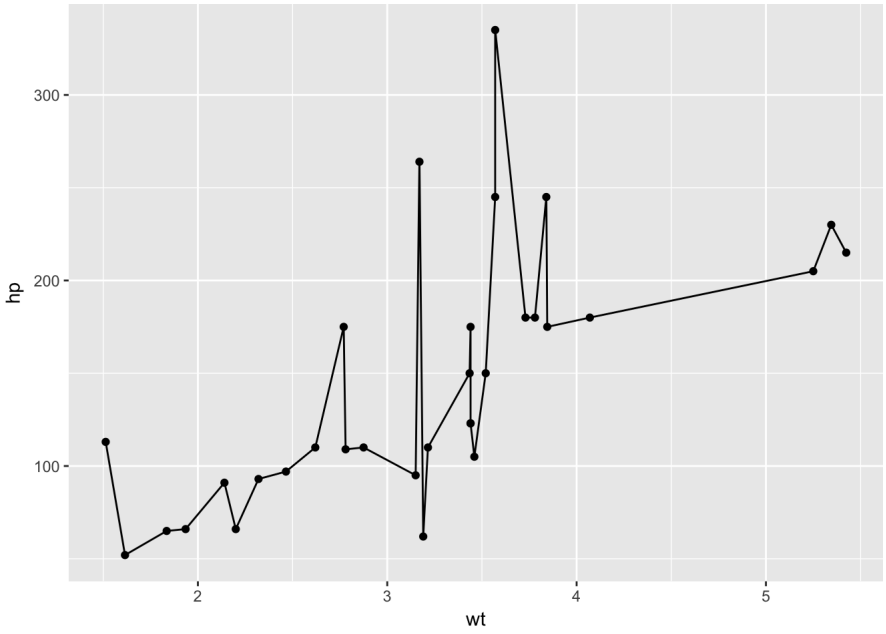
```
# the points turn out to be red
plot1 + geom_point(aes(color = "blue"))
```



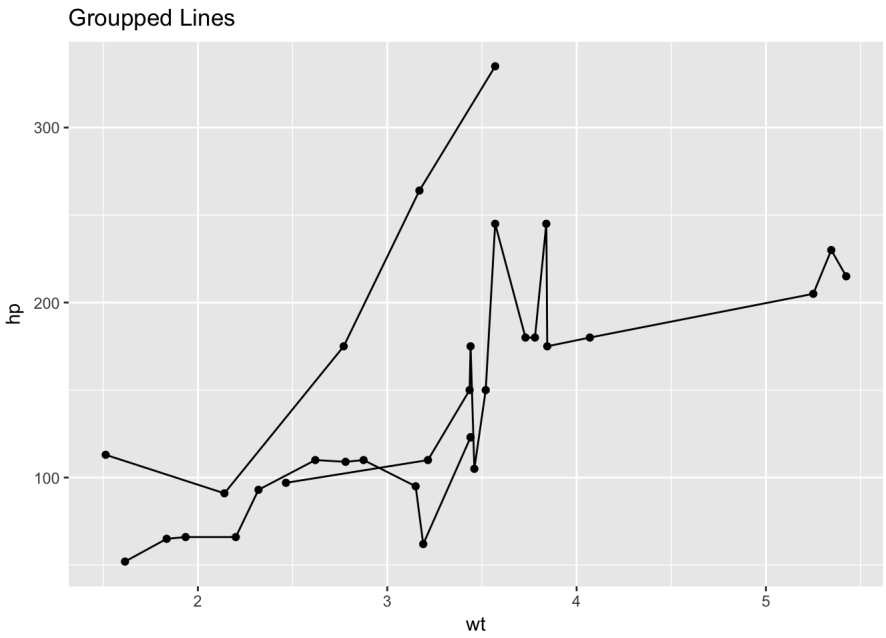
In the latter graph, what 'aes()' did was to connect the variable "blue" to the attribute 'color', instead of treating "blue" as a string input. With only the legnth of 1, "blue" failed to be interpreted by ggplot and thus the color was red by defult.

Another concept to be noticed is grouping:

```
# the original ungroupped plot
ggplot(data = mtcars, mapping = aes(x = wt, y = hp)) +
  geom_point() +
  geom_line()
```




```
# groupped by the factor "gear"
ggplot(data = mtcars, mapping = aes(x = wt, y = hp, group = factor(gear))) + geom_point() + geom_line() +ggtitle("
Groupped Lines")
```



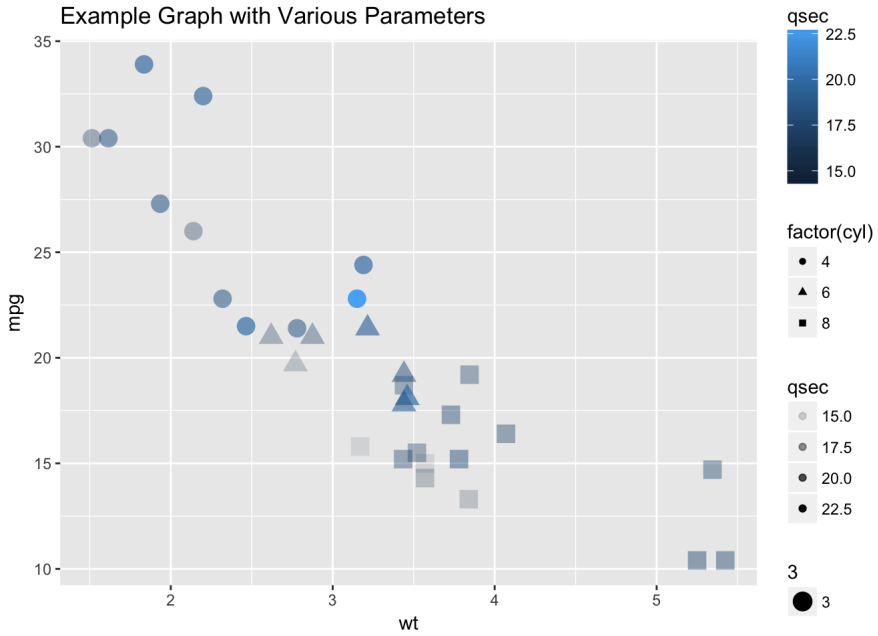
Groupping arranges data by their common attributes to display the data in a more organized and insightful way.

geom & stat

The two major sets of functions of ggplot2 are “geom” and “stat”: the former implements the geometrical fundation of the graph, while the latter analyzes the data and create visual information in addition to the raw data.

Some of the most frequently used geom shapes include scatterplot, histogram, smooth and so on. The shape of the graph should be chosen based on the consideration of best presenting the characteristcs of the data. When a graph is outlined, we can polish the graph by modifying some parameters of the graph. For example, when “geom_point” function draws a scatterplot, we can further modify the size, color, shape, and transparency of the individual points. One example code is here:

```
ggplot(data = mtcars, aes(wt, mpg)) +
  geom_point(aes(color = qsec, alpha = qsec, size = 3, shape = factor(cyl))) +
  ggtitle("Example Graph with Various Parameters")
```



As shown in the graph above, a legend is attached to the right of the graph for reference.

Another useful trick is the various ways of establishing a histogram plot. There are five ways to position the bars: dodge, fill, identity, jitter, and stack. More examples will be displayed in the practice section.

colors

It is unnecessary to address the importance of colors to a lively and comprehensible graph anymore. The package ggplot2 provides flexible options of manipulations around colors. In addition, some preset palettes are implanted within the package for users to choose from. Users are free to set their own color patterns using the hexadecimal code for colors, with a format of `#FF9999`. The colors are used in ggplots in two ways: the lines and points with `color =`, as well as the color-filled objects with `fill =`. The luminance and sturation of the colors are also subject to customized changes.

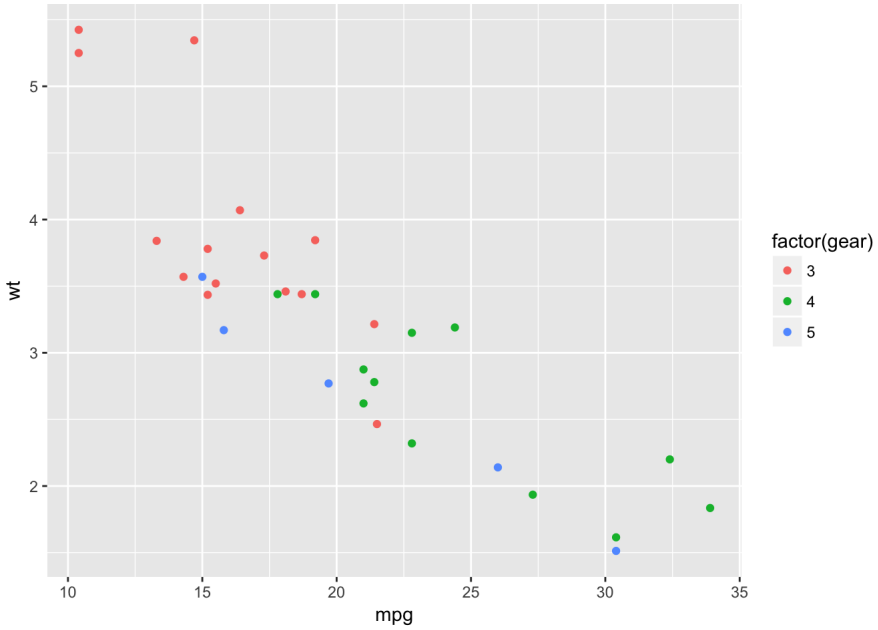
The functions that are used to set the parameter of the colors follow the similar pattern of `scale_fill_manual`.

facet

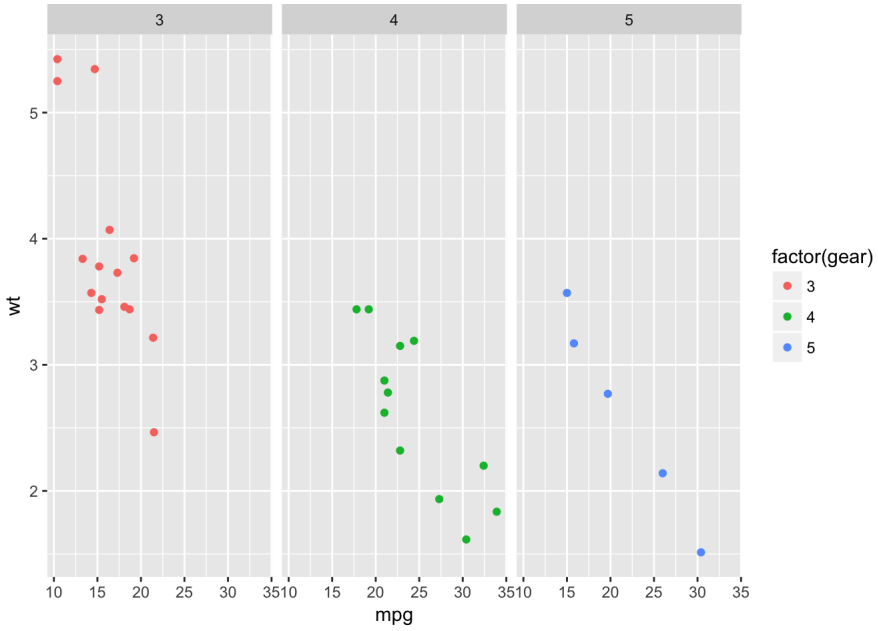
Facet is used to display multiple graphs on one page. By facetting a dataset, ggplot2 first distributes the data into subdivided sets, than maps

them into different facets. There are two types of faceting: (1) `facet_grid`, which develops a two-dimension tabular network with the row and column defined by varibales, and (2) `facet_wrap`, which first develops an one-dimension cordinate then wrap the data onto the second dimension. To better understand the distinctions, one example can be:

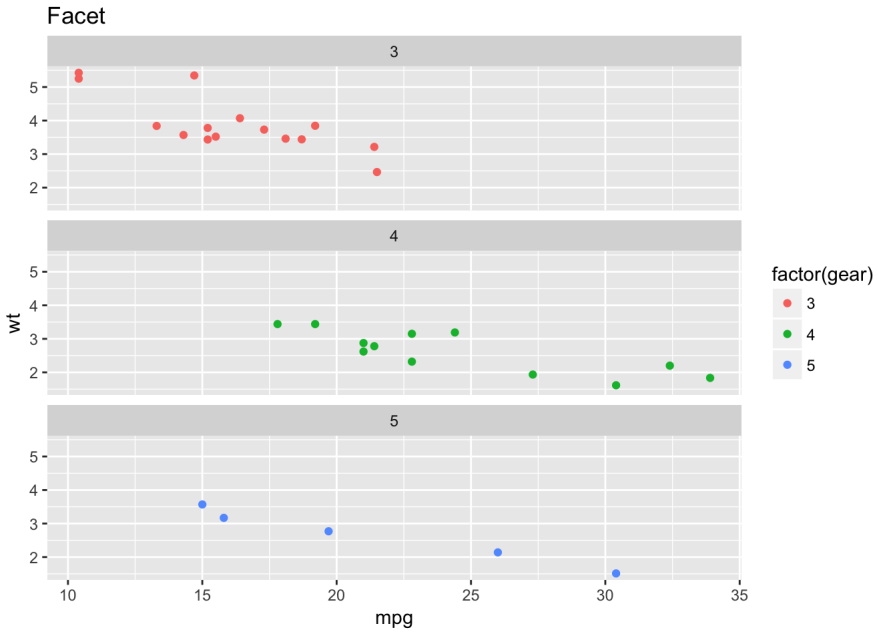
```
# the original plot, groupped by "gear"
ggplot(data = mtcars, aes(x = mpg, y = wt, color = factor(gear))) +
  geom_point()
```




```
# to display the same graph in three graphs, based on the group
ggplot(data = mtcars, aes(x = mpg, y = wt, color = factor(gear))) + geom_point() + facet_grid(~ gear)
```




```
# to display the graph, now with wrapping
ggplot(data = mtcars, aes(x = mpg, y = wt, color = factor(gear))) + geom_point() + facet_wrap(~ gear, nrow = 3) +
  ggtitle("Facet")
```

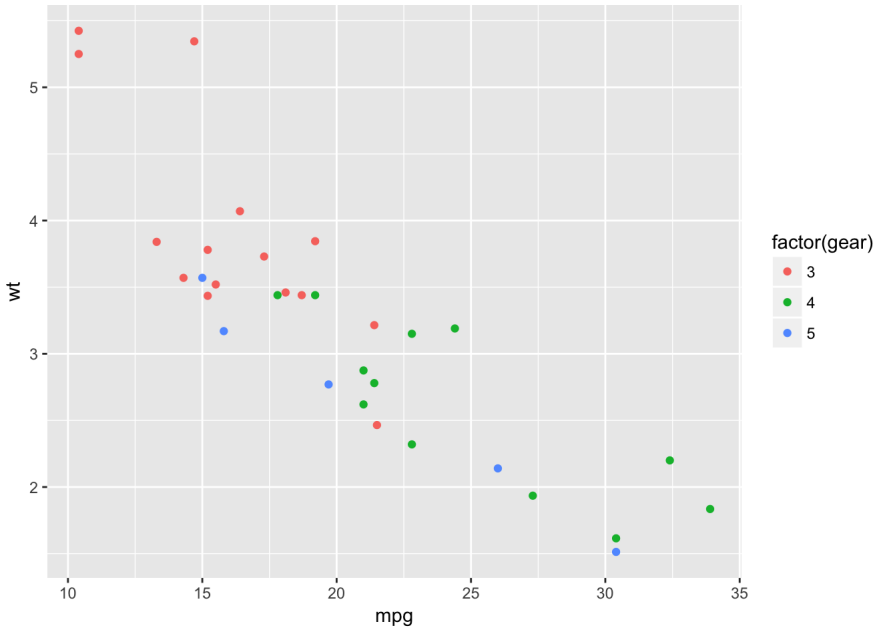


As can be seen in the above two graphs, `facet_grid` cannot be added with an arguement to control the number of rows/colums. In addition, `facet_grid` displays all graph matrix even if in some of the facets no data exists. In contrast, `facet_wrap` will eliminate those where no data is presented.

ggsave

ggsave is the specialized writing function for ggplot2. It automatically detects and stores the last plot into designated path. The format `ggsave` supports includes png, jpg, pdf and more. It can be simply used in the following way:

```
ggplot(data = mtcars, aes(x = mpg, y = wt, color = factor(gear))) +
  geom_point()
```




```
ggsave(file = "./temporary_plot.png", width = 5, height = 5)
```

Resources and Others

When working on this post, I encountered a lot of new knowledge and difficulties in codings, as I'm sure a lot of my fellow classmates struggled at. The biggest suggestion I want to give, as simple and unnecessary it may sound, is never hesitate to search for help online. There are plenty of resources online, no matter they are official documents published by Rstudio and authors of the packages, or forums like stackoverflow, or simply other people's blogs. I can't be more grateful to the detailed reference and cookbooks, the anonymous helpers on forums and a diverse variety of sharings on personal sites. I've included some of the reference that are most significant and organized in the "Reference section" at the end of the post. The ones that I used most ended up being the most original and official tutorials, such as reference 1, 2 and 5.

To be experienced and skillful on plot drawing, constant practice is inavoidable. One way I use to find the sample graphs to learn from is to look at the reports of some big companies, especially those of consulting industries. They usually have the most beautiful and comprehensible graphs with professional organizations such as the color selection, the shape, the processing of the data, etc. They may have created these graphs with other softwares like excel or powerpoint, therefore it might be challenging to use R to mimic the same graph, however, they are still excellent templates. Use R to attempt to reestablish the same templates, and you will find your ability of solving unexpected error as well as your aesthetics of the graphs steadily increasing.

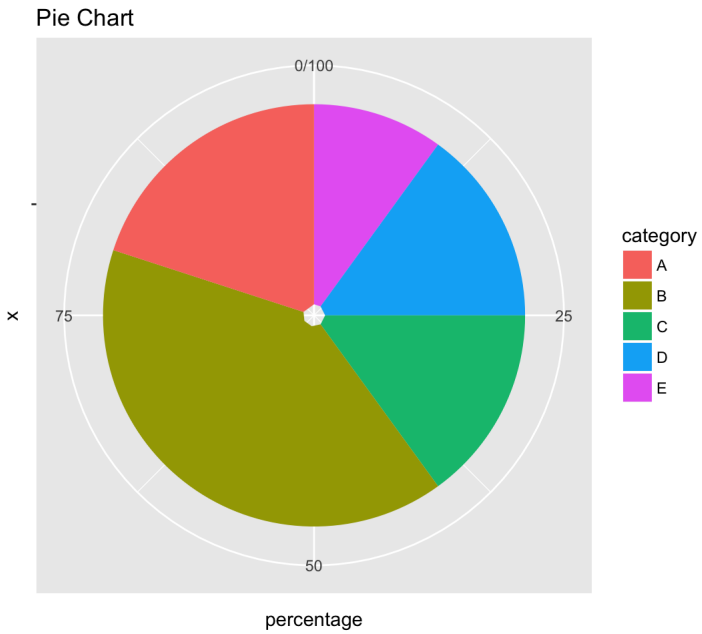
3.Practices

In this section, I'm going to present some examples of graphs drawn with ggplot2. Most of the graphs are found online, with or without original codes. The goal of this section is to have more exposure to the challenging yet surprisingly beautiful graphs made possible by ggplot2. Let's begin!

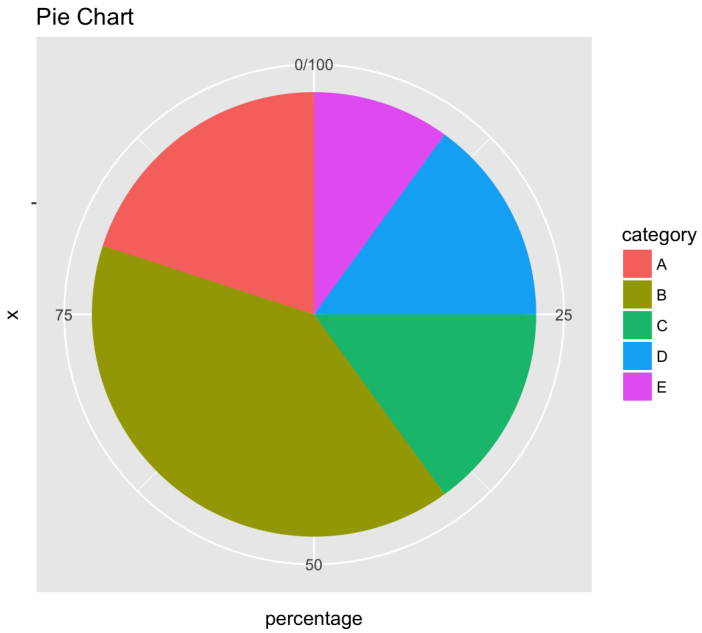
(1) Pie and Ring and More

ggplot2 can also be used to draw a pie chart, or some derived forms of a pie chart, to better display the proportion each variable takes up.

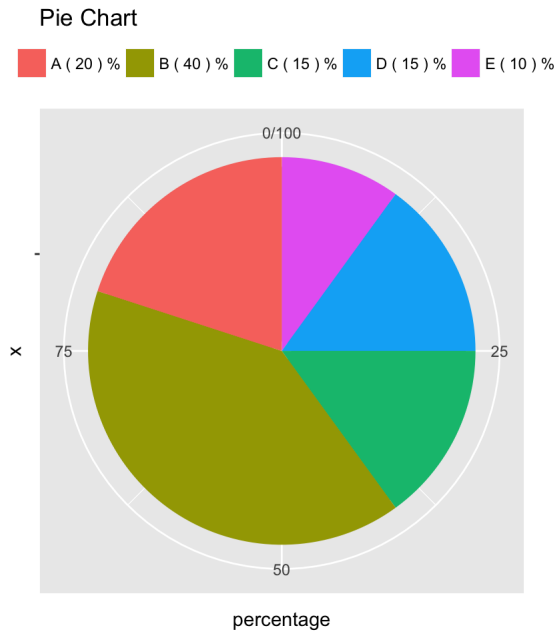
```
# dummy dataframe
df1 <- data.frame(percentage = c(20, 40, 15, 15, 10), category = c("A", "B", "C", "D", "E"))
# first to creat a histogram that stacks all percentages on one bin
step1_1 <- ggplot(data = df1, aes(x = "", y = percentage, fill = category)) +
  geom_bar(stat = "identity") +
  ggtitle("Pie Chart")
# then convert the cordinate into polar cordinate
step1_2 <- step1_1 + coord_polar(theta = "y")
step1_2
```



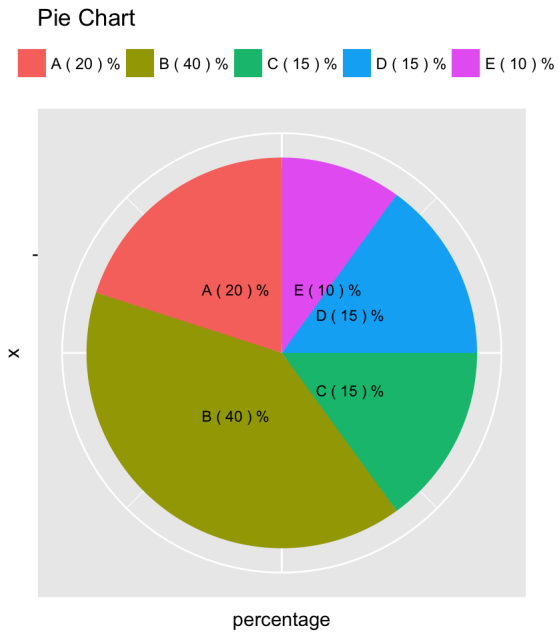

```
# to eliminate the blank in the middle
step1_3 <- step1_2 + geom_bar(stat = "identity",width = 1)
step1_3
```



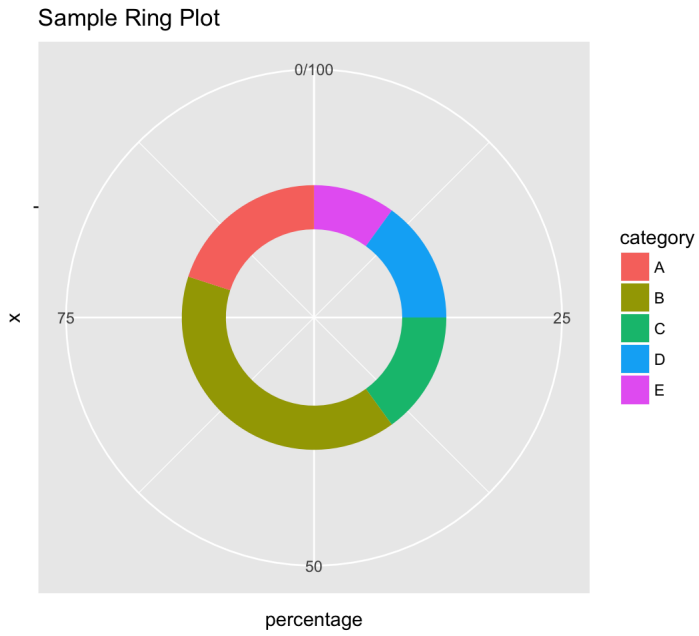
```
# to move the legend to the top, and add percentage
Percentage_label <- as.vector(df1$category)
Percentage_label <- paste(Percentage_label, "(", df1$percentage, ")", "%")

step1_4 <-  step1_3 + theme(legend.title = element_blank(), legend.position = "top") + scale_fill_discrete(breaks
= df1$category, labels = Percentage_label)
step1_4
```




```
# to remove the white line around the pie but add to the graph
step1_5 <- step1_4 + theme(axis.text.x = element_blank()) + geom_text(aes(x = 0.9, y = c(90,60,33,17,10), label =
Percentage_label), size = 3)
step1_5
```

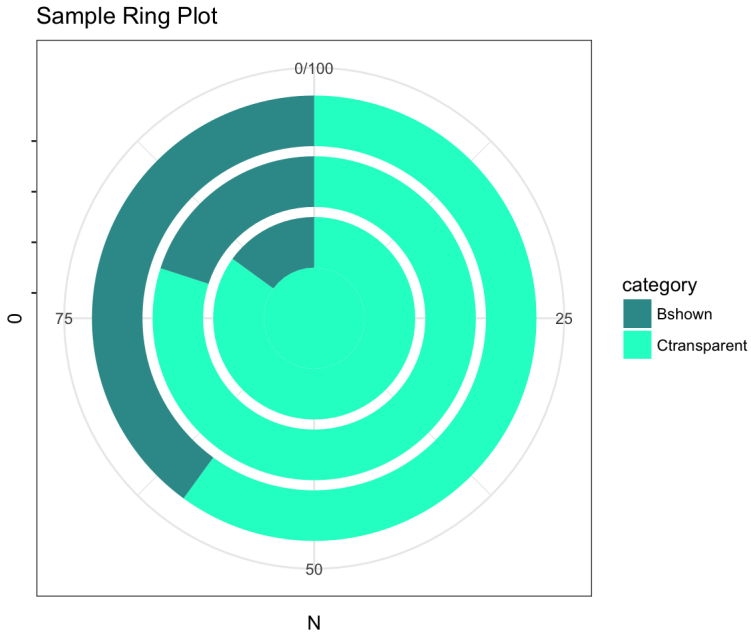



```
# make a ring
ggplot(data = df1, aes(x = "", y = percentage, fill = category)) +
  geom_bar(stat = "identity", width = 0.2) +
  coord_polar(theta = "y") +
  ggtitle("Sample Ring Plot")
```



2. Another Ring

```
N <- c(0,100)
B <- c(20,80)
C <- c(40,60)
A <- c(15,85)
category <- c("Bshown","Ctransparent")
df1_1 <- data.frame(A, category)
df1_2 <- data.frame(B, category)
ggplot()+
  geom_bar(aes(x = 0, y = N, fill = category), stat= "identity", width = 0.05) +
  geom_bar(data = df1_1, aes(x = 0.05, y = A, fill = category), stat = "identity", width = 0.05) +
  geom_bar(data = df1_2, aes(x = 0.11, y = B, fill = category), stat = "identity", width = 0.05) +
  geom_bar(aes(x = 0.17, y = C, fill = category), stat = "identity", width = 0.05) +
  coord_polar(theta = "y") +
  theme_bw() +
  scale_fill_manual(values = c("#339999", "#00FFCC")) +
  ggtitle("Sample Ring Plot") +
  theme(axis.text.y = element_blank())
```



(3) Stacked Barchart

This is another simplistic yet pretty barchart I found online to present the result of a Likert-matrix survey. What's outstanding in this plot is how it arranges some data into the negative part of x-cordinate, making the distribution more comprehensible. Let's take a look.

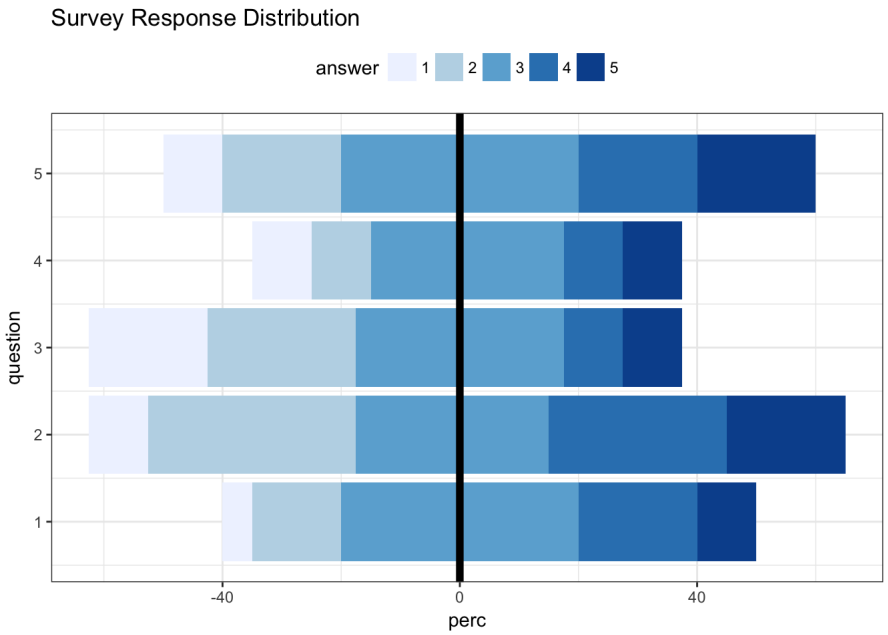
```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#dummy data
question <- rep(c("1","2","3","4","5"), each = 5)
answer <- rep(c(1,2,3,4,5))
perc <- c(10,20,40,20,10,10,10,30,30,20,20,25,35,10,10,10,35,35,10,10,5,15,40,20,20)
df2 <- data.frame(question, answer, perc)
df2$answer <- as.factor(df2$answer)
df2$question <- as.numeric(df2$question)
#split the dataframe into two parts into positive and negative x-cordinate
df2_2 <- df2 %>% filter(answer == 3) %>% mutate(perc = perc/2)

df2_1 <- df2 %>% filter(answer == 1 | answer == 2) %>% rbind(df2_2) %>% arrange(answer)
df2_1$answer <- rev(df2_1$answer)
df2_1$perc <- rev(df2_1$perc)
df2_1$perc <- df2_1$perc * (-1)

df2_3 <- df2 %>% filter(answer == 5 | answer == 4) %>% rbind(df2_2)




# plot
ggplot() +
  geom_bar(data = df2_1, aes(x = question, y = perc, fill = answer), stat = "identity") +
  geom_bar(data = df2_3, aes(x = question, y = perc, fill = forcats::fct_rev(answer)), position = "stack", stat =
"identity") +
  coord_flip() +
  geom_hline(yintercept = 0, color = "black", size = 2) +
  theme_bw() +
  theme(legend.position = "top") +
  scale_fill_brewer(palette = "Blues") +
  ggtitle("Survey Response Distribution")
```



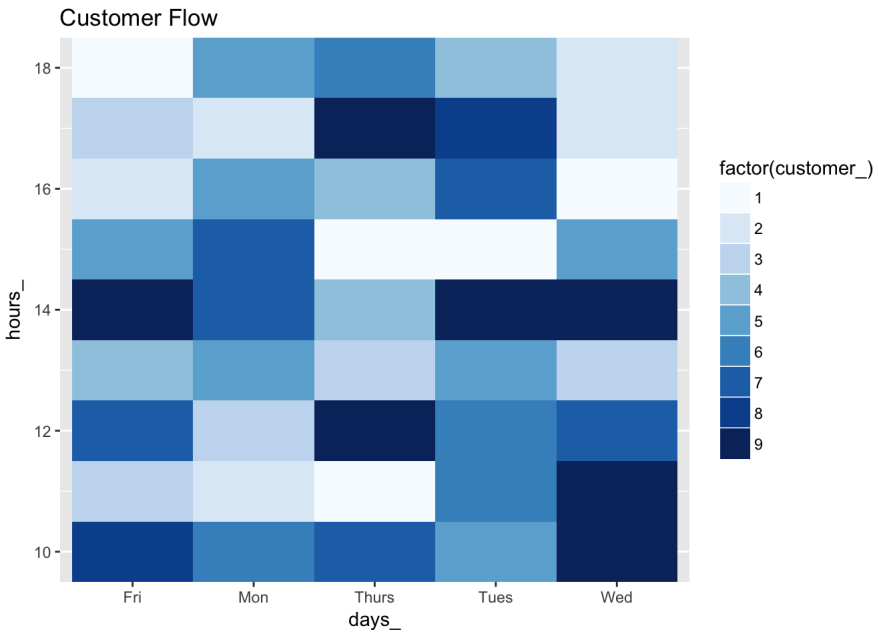
(4) Heat Map

I came across this plot when searching for professional plots for business operations. The heat map is powerful in displaying the comparative distribution in all brackets, so that the viewer can get an intuitive idea of which area the most frequencies fall in–just as the name indicates the plot shows where the heat is. One perfect usage of heat map is recording the customer flow during a certain period of time. In the following graph, an example heat map is presented:

```
# create a dummy dataset; in this table, each cell shows the corresponding customer flow in this time period
days <- c("Mon", "Tues", "Wed", "Thurs", "Fri")
hours <- c(10, 11, 12, 1, 2, 3, 4, 5, 6)
df3 <- data.frame(matrix(sample(1:9, 45, replace = TRUE),ncol = 9, nrow = 5))
colnames(df3) <- hours
rownames(df3) <- days
df3
```

```
##       10 11 12 1 2 3 4 5 6
## Mon    6  2  3 5 7 7 5 2 5
## Tues   5  6  6 5 9 1 7 8 4
## Wed    9  9  7 3 9 5 1 2 2
## Thurs  7  1  9 3 4 1 4 9 6
## Fri    8  3  7 4 9 5 2 3 1
```

```
# first, to rearrange the dataframe
days <- rep(days, each = 9)
hours <- rep(hours, 5)
df3_1 <- data.frame(days_ = days, hours_ = hours)

customer <- NULL
for (i in 1:5) {
  for (j in 1:9) {
    customer <- c(customer, df3[i,j])
  }
}

df3_1 <- mutate(df3_1, customer_ = customer)

# convert the time into navy time
df3_1$hours_ <-  as.numeric(df3_1$hours_)
df3_1[which(df3_1[,"hours_"] < 10),2] <- df3_1[which(df3_1[,"hours_"] < 10),2] + 12

df3_1$customer_ <- as.factor(df3_1$customer_)

ggplot(data = df3_1, aes(x = days_, y = hours_, fill = factor(customer_))) +
  geom_tile(stat = "identity") +
  scale_fill_brewer(palette = "Blues") +
  scale_y_continuous(expand = c(0,0)) +
  ggtitle("Customer Flow")
```



Notice the data is displayed in the order of Friday, Monday, Thursday, Tuesday and Wednesday. This is because ggplot2 automatically rearranged them by the alphabetical order. To fix this, one solution is to change the order by adding an index letter, say, A_Monday, B_Tuesday and so on, then change the label of x-axis later.

(5) Maps

One powerful function ggplot2 has is the drawing on maps. With external packages, ggplot2 can conduct data presentation on maps with points, colors and more. Releated packages include `maptools`, `ggmap`, and specific map of the country or rigions that you choose. In particular, the U.S. map is implanted within the package already.

Due to the overloaded installation of the packages, this post is not going to detail on the codes and pckages of drawing a map graph. However, a sample graph is attached here to provide a glance of the options.

Map of Africa with “prevalence”

4.Conclusion

In conclusion, the potentiality of ggplot2 may exceed what you have imagined. With the powerful manipulation of the data, and the assistance of external packages, there are more wonderful graphs ggplot2 is capable of making. It may seem difficult to be really experienced at ggplot2, as there are numerous functions and parameters and always unexpected error messages and challenges. However, with more and more practice, you will find ggplot2 more handy and the plots more and more splendid.

5.Reference

1. http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/
2. http://www.cookbook-r.com/Graphs/Axes_(ggplot2)/
3. http://www.cellyse.com/how_to_use_gggplot2_part1/
4. http://www.cellyse.com/how_to_use_gggplot2_part2/
5. http://www.cookbook-r.com/Graphs/Legends_(ggplot2)/
6. http://statistics.ohlsen-web.de/stacked-barchart-for-likert-items/
7. https://stackoverflow.com/questions/20220424/ggplot2-bar-plot-no-space-between-bottom-of-geom-and-x-axis-keep-space-above
8. https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
9. http://ggplot2.tidyverse.org/reference/
10. http://www.sthda.com/english/wiki/ggplot2-axis-ticks-a-guide-to-customize-tick-marks-and-labels
11. http://datartisan.com/article/detail/57.html

I realize that somehow the .md file on github is not showing the reference, while the links are shown without problem in the .Rmd file and local knitted file. I'm seeking for solutions now. Please refer to the .Rmd file in the same folder for links.