title: “post01-sophia-cheng” author: “Sophia Cheng” date: “October 18, 2017” output:html_document

Introduction:

Being an advocate of visual data, learning ggplot2 in R was a major breakthrough in understanding how to process data in this introductory data analysis class. Since I am new to R and any computer language in general, this course has been more about how to get commands to produce an output rather than gaining an clearer understanding of the data using R. However, the graphing lessons stood out to me in its unparalleled practicality compared to the series of commands and data structures that preceded this. Through ggplot2, I saw the value of R language in decoding and displaying mass amounts of data in an easily digestible format with a short, stackable command. Although base graphics was a great introductory tool, it serves more as a springboard to ggplot2 rather than a main graphing device. I appreciate and see the value of both functions, but will most likely choose ggplot2 for graphing.

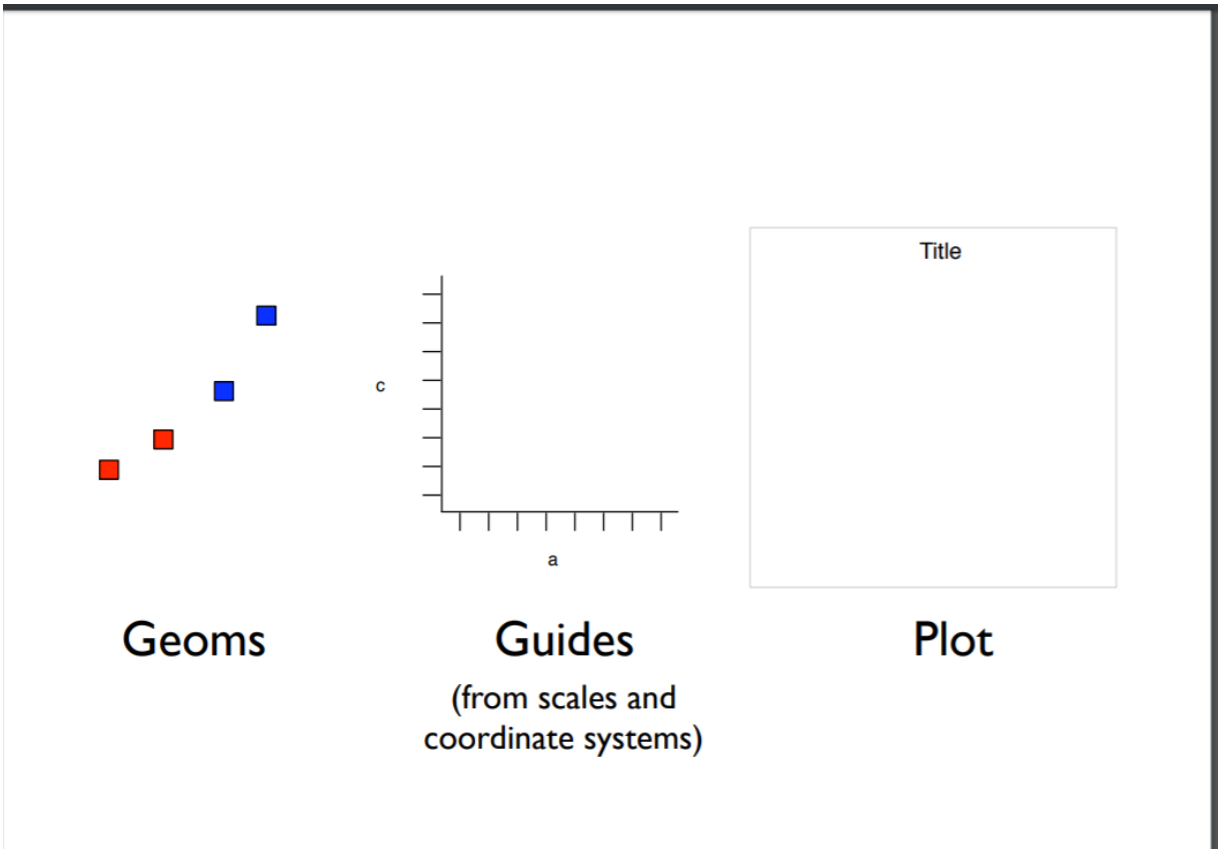
What is this post about?

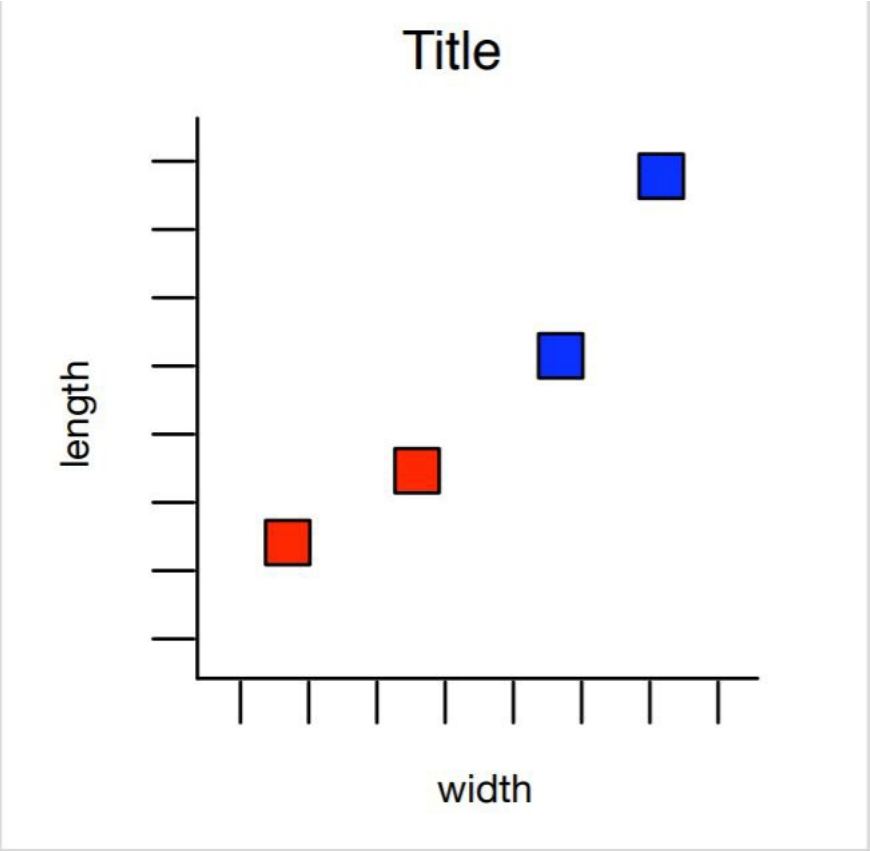
To show how an understanding of base graphics segue into using ggplot2 as the main tool for more complex functions.

Beginning to plot: exchanging lines for names

In R, the first step to plotting is understanding the base components. In my experience, it mirrors learning to graph for the first time. However, sketched lines are exchanged for the actual names of these graphing components. For example, “geoms” replace coordinates (2,2) and (5,3) that would have been counted by boxes on paper, and “plot” delineates a frame rather than an a square place allotted to draw a graph.

The Breakdown of an R Graph





With R, graphing is like building a lego tower- a vertical construction of dissimilar pieces, represented by individual graphing components, create a single unit, a graph.

Base components:

1. Data
2. Geometric object (geom.)
   - ☒Ex. Histogram, bar, scattterplot
3. Statistical transformation (stat) □* Ex. geom_smooth = stat_smooth + geom_ribbon
   - p + stat_bin(geom="area")
   - p + stat_bin(geom="point")
   - p + stat_bin(geom="line")
4. Scales: default, but I have found transforming data for a nicer-looking graph not particularly troublesome
5. Coordinate system: Control how the two positions aesthetics work together
   - (default: Cartesian)
   - Others of note: coord_flip(), coord_map(), coord_polar()
6. position adjustment/faceting
   - facet_grid(row ~ col, margins = TRUE) ### Graph example: scatterplot □#### Data: players, salary □#### Geom.: point □#### Stat: identity ☒#### Scales: linear ####☒ Coord: Cartesian Basic graphics:

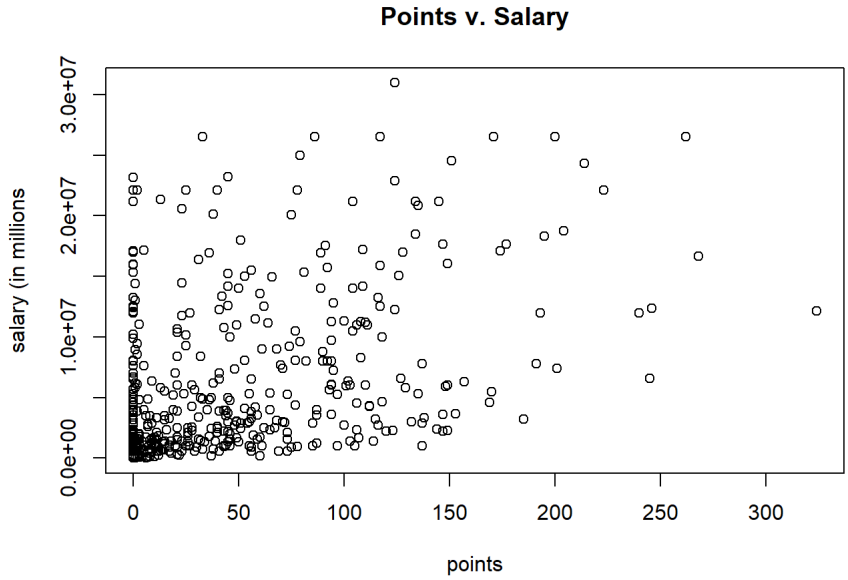
```
#load data file
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

```
load("data/nba2017-salary-points.RData")
dat <- read.csv("C:\\Users\\Sophia\\stat133-hws-fall17\\stat133-hws-fall17\\nba2017-player-statistics.csv", string
sAsFactors = FALSE)
str(dat)
```

```
## 'data.frame':    441 obs. of  24 variables:
##  $ Player      : chr  "Al Horford" "Amir Johnson" "Avery Bradley" "Demetrius Jackson" ...
##  $ Team        : chr  "BOS" "BOS" "BOS" "BOS" ...
##  $ Position    : chr  "C" "PF" "SG" "PG" ...
##  $ Experience  : chr  "9" "11" "6" "R" ...
##  $ Salary      : num  26540100 12000000 8269663 1450000 1410598 ...
##  $ Rank        : int  4 6 5 15 11 1 3 13 8 10 ...
##  $ Age         : int  30 29 26 22 31 27 26 21 20 29 ...
##  $ GP          : int  68 80 55 5 47 76 72 29 78 78 ...
##  $ GS          : int  68 77 55 0 0 76 72 0 20 6 ...
##  $ MIN         : int  2193 1608 1835 17 538 2569 2335 220 1341 1232 ...
##  $ FGM         : int  379 213 359 3 95 682 333 25 192 114 ...
##  $ FGA         : int  801 370 775 4 232 1473 720 58 423 262 ...
##  $ Points3     : int  86 27 108 1 39 245 157 12 46 45 ...
##  $ Points3_atts: int  242 66 277 1 111 646 394 35 135 130 ...
##  $ Points2     : int  293 186 251 2 56 437 176 13 146 69 ...
##  $ Points2_atts: int  559 304 498 3 121 827 326 23 288 132 ...
##  $ FTM         : int  108 67 68 3 33 590 176 6 85 26 ...
##  $ FTA         : int  135 100 93 6 41 649 217 9 124 37 ...
##  $ OREB        : int  95 117 65 2 17 43 48 6 45 60 ...
##  $ DREB        : int  369 248 269 2 68 162 367 20 175 213 ...
##  $ AST         : int  337 140 121 3 33 449 155 4 64 71 ...
##  $ STL         : int  52 52 68 0 9 70 72 10 35 26 ...
##  $ BLK         : int  87 62 11 0 7 13 23 2 18 17 ...
##  $ TO          : int  116 77 88 0 25 210 79 4 68 39 ...
```

```
plot(dat$Points3, dat$Salary, xlab = 'points', ylab = 'salary (in millions', main = "Points v. Salary")
```



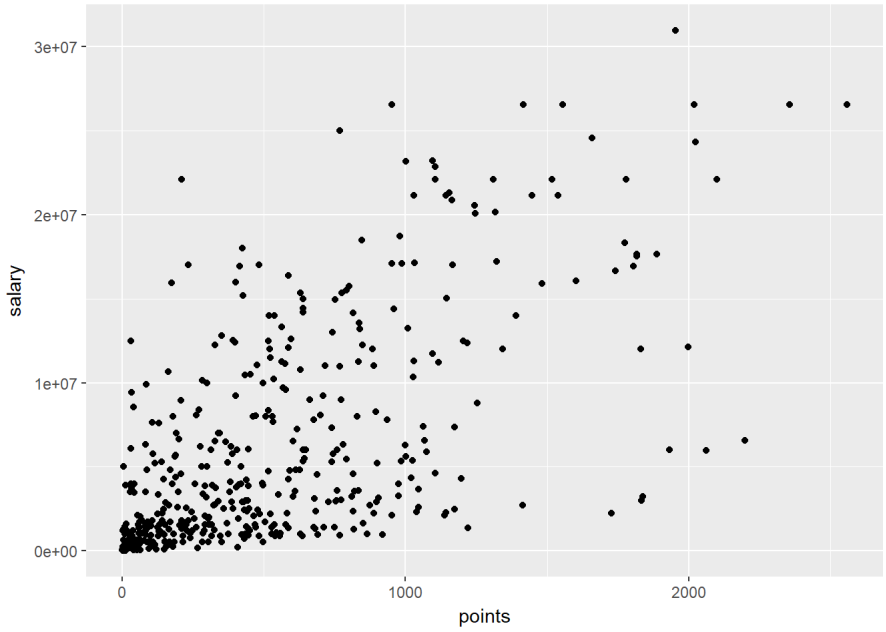
Post-first impressions: Once each component is identified, it becomes easy to piece the whole graph together. The output is unquestionably much neater and faster than hand-drawn graphs, and can be shared digitally. It can also support much more data than I am capable of processing by myself.

*However, compared to ggplot, base graphics contains smaller, more fractured pieces of the graph linked in a haphazard manner.

Transitioning to ggplot

```
ggplot example:
```

```
ggplot(data = dat) +
  geom_point(aes(x = points, y = salary))
```

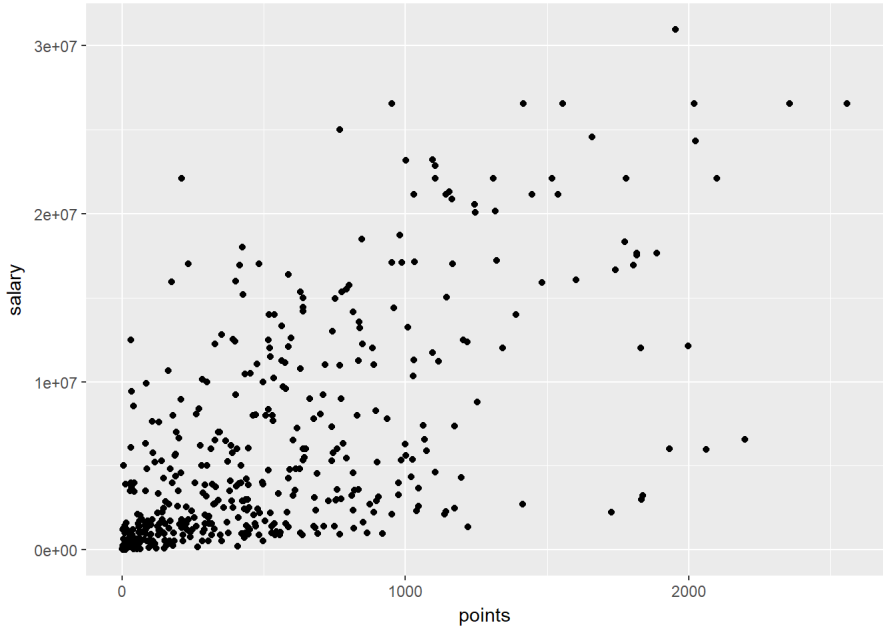


###At this point of the learning process, once components are clarified (labels, stats, shape..), transitioning from plotting on base graphics to ggplot2 is smooth sailing. Even from a simple graph, it is easy to see why ggplot2 would be preferred. The code itself is clearer even on the most rudimentary of graphs. Everything has its own label, and there are sub-levels of components that make the entire graphing process more organized. More chunky plotting- blocks of lego rather than separate lego pieces- easier to piece together.

> ## An essential ggplot2 auxiliary -> Ggplot2 cheatsheet
>
> It lays out a lot of options like a catalog of graphing functions and embellishments. The variety of options exhibits the detail and complexity that can be achieved with ggplot2. Ggplot retains an organized structure that is as easy to read as its visual output. For a person new to coding, the equation-like format and logical labeling is less daunting to attempt and takes less time to master. https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

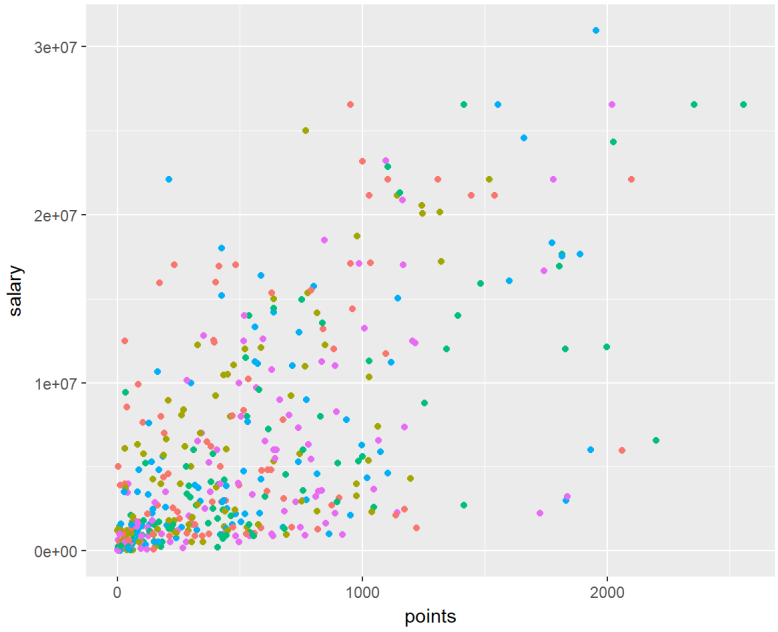
Here's an example that demonstrates the clarity and ease plotting with ggplot2 provides due to its "building blocks" feature. Having a cheatsheet facilitates this: 1. # scatterplot (option 1)

```
ggplot(data = dat) +
  geom_point(aes(x = points, y = salary))
```



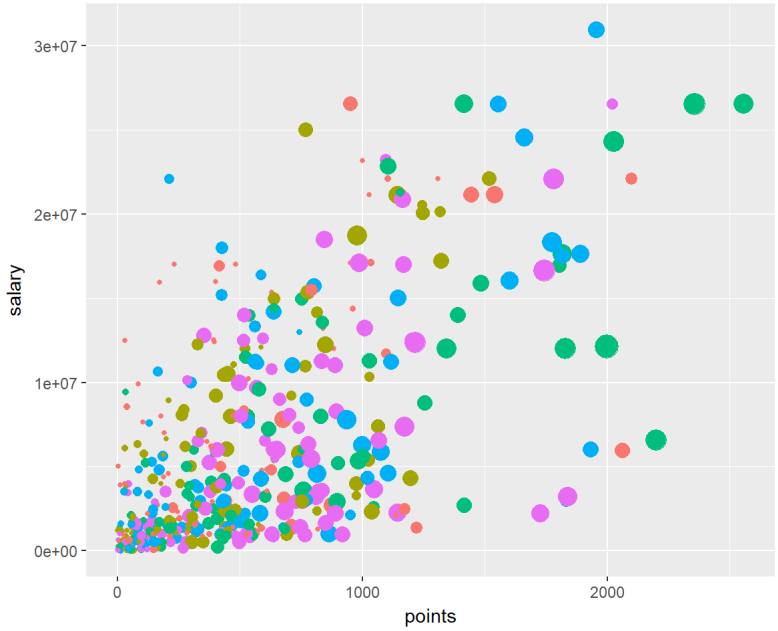
2. # **colored** scatterplot

```
ggplot(data = dat, aes(x = points, y = salary)) +
    geom_point(aes(color = position))
```



(added color to make a more visually appealing graph. This function would be useful for presentations and provide an extra aesthetic factor) 3. # sized and colored scatterplot

```
ggplot(data = dat, aes(x = points, y = salary)) +
    geom_point(aes(color = position, size = points3))
```



(to add the extra aesthetic, size, the size is literally added into geom._point)

Additional Points to Consider for Ggplot2

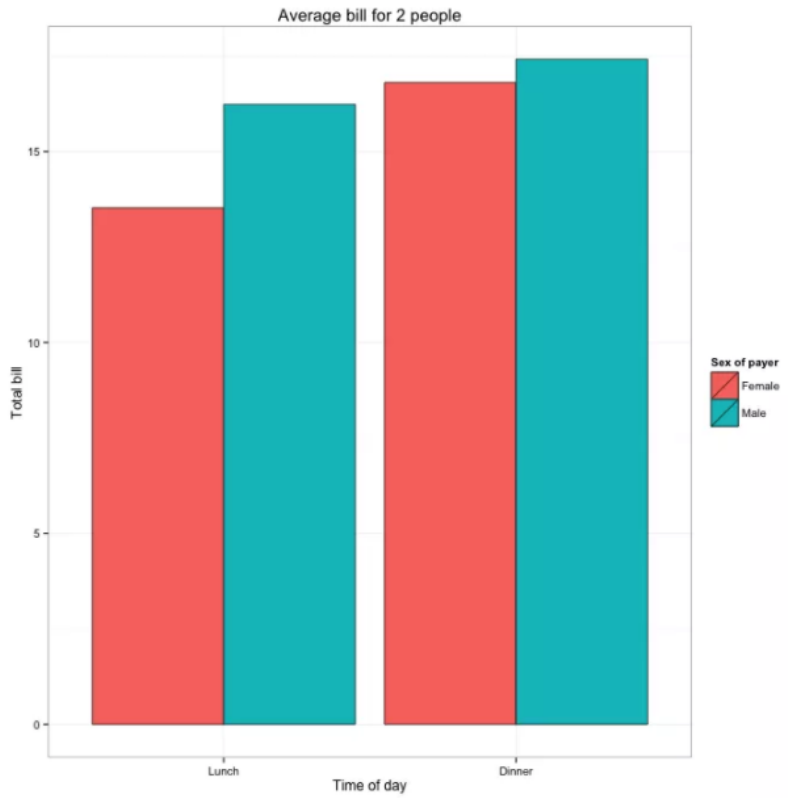
McClelland Kemp made a post expousing ggplot2 on Gihub that puts into words my impression of ggplot2- “makes doing the right thing easy, while keeping harder things possible.”

(Moving on to more complicated graphs)

Ggplot2 becomes more efficient. While the author of the article (and of the graphs) still prefers base graphic’s **customizable factor**, she admits that ggplot2 offers **clean, ready-to-use** packages. Loops, additional subsetting, and lengthiness that allow room for error are all avoided in ggplot2. However, base graphics allow for complete control of the output. On the other hand, this requires high proficiency of graphing in R, which for beginner-level purposes, do not apply.

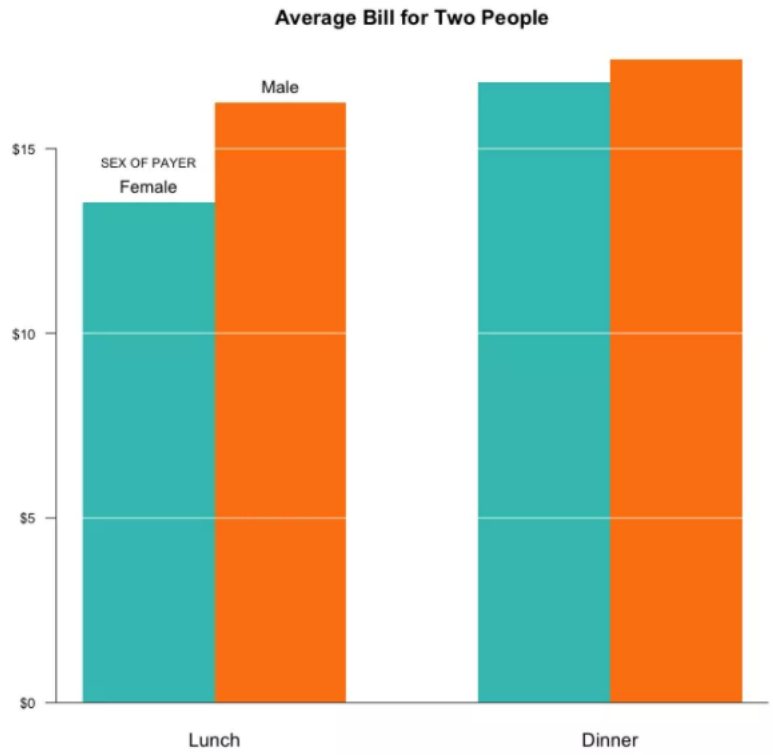
Looking at

ggplot2




```
ggplot(data=dat1, aes(x=time, y=total_bill,
fill=sex)) +
  geom_bar(colour="black", stat="identity",
           position=position_dodge(),
           size=.3) +
  scale_fill_hue(name="Sex of payer") +
  xlab("Time of day") + ylab("Total bill") +
  ggtitle("Average bill for 2 people") +
  theme_bw()
```

Base Graphics



##Now it also becomes

```
par(cex=1.2, cex.axis=1.1)
barplot(dat1mat, beside = TRUE, border=NA,
col=mf_col,
    main="Average Bill for Two People",
yaxt="n")
axis(2, at=axTicks(2), labels=sprintf("$%s",
axTicks(2)),
    las=1, cex.axis=0.8)
grid(NA, NULL, lwd=1, lty=1, col="#ffffff")
abline(0, 0)
text(1.5, dat1mat["Female", "Lunch"], "Female",
pos=3)
text(2.5, dat1mat["Male", "Lunch"], "Male",
pos=3)
text(1.5, dat1mat["Female", "Lunch"]+0.7, "SEX
OF PAYER",
    pos=3, cex=0.75)
```

apparent the additional merits of ggplot2 * based on grid, the preferred and recommended way to develop new graphics in R
* powerful layer system that makes it easy to combine different sources of data
* superpose several layers
* let your plots evolve (or devolve) with minimal changes to code (ex. Faceting, coloring, smoother) * Presentation
* The ultimate ggplot2 advantage: faceting and multiple categorical variables
* **Practically, comparing variables is an integral part of data analysis when looking at entire data sets. More often than not, multiple variables need to be considered in any real-world situation. While basic graphing is capable of matching ggplot2, the work required to imitate these functions requires more time and skill, which is not ideal, especially for an R novice.**

Conclusion

An understanding of base graphics is fundamental to graphing in R. However, the next step is to transition to ggplot, which is more user-friendly and much less cryptic for an R novice like myself. Trying to master base graphics requires a considerable

amount of skill that is not worth the time and effort when to proceed with a data set, a simple visual is needed. For practical and aesthetic reasons, and at this point in my R progress, ggplot2 is my preferred graphing tool. It is more rational, and easier to adjust to. Rather than a new language, ggplot2 is comparable to lego blocks- you just have to identify the right pieces, and from there, construct as you wish. The variety and flexibility of these geom's or "lego blocks" is sufficient for all my graphing needs thus far. Additionally, I can see the faceting functioning coming into frequent use with real data. Most importantly, the handy ggplot2 cheat sheet provides the means for me to focus on what I would like to display visually using R instead of trying to figure out how to get the correct output with base graphics.

Resources

1. http://ggplot2.org/resources/2007-vanderbilt.pdf
2. http://www.cookbook-r.com/Graphs/Bar_and_line_graphs_(ggplot2)/
3. https://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/
4. https://github.com/tidyverse/ggplot2/wiki/Why-use-ggplot2
5. http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html#scales
6. https://mandymejia.wordpress.com/2013/11/13/10-reasons-to-switch-to-ggplot-7/
7. https://github.com/ucb-stat133/stat133-fall-2017/blob/master/slides/14-ggplot-lecture.pdf