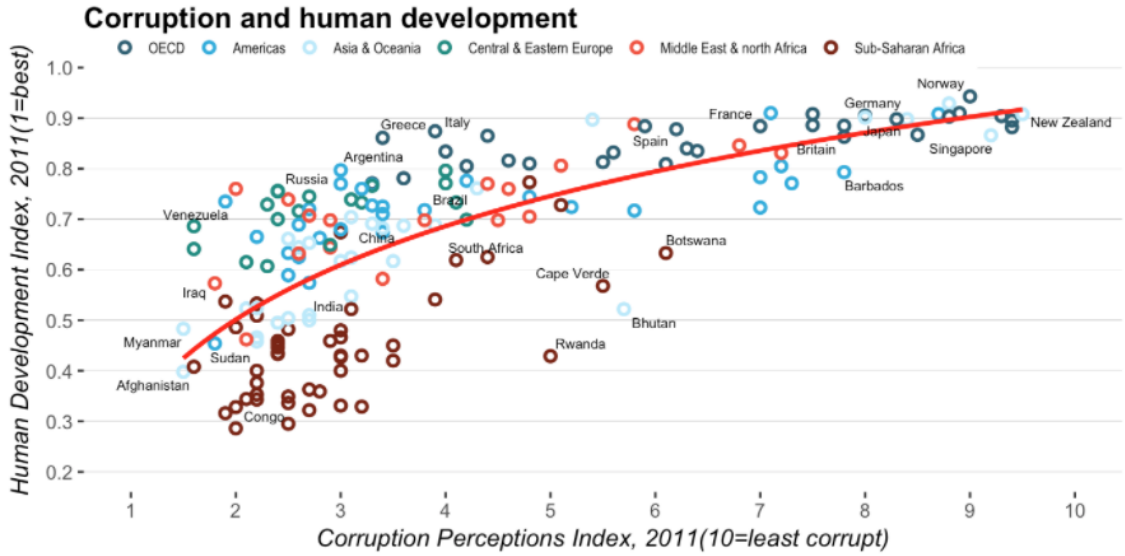
Data visualization with R & ggplot2

Introduction — Motivation, Background and Audience

Have you ever thought about what software package *The Economist* group uses to create charts and graphs for the magazine? Yes, that's right. R and ggplot2 (a specific data visualization package for R) are the typical tools people use to generate complex graphs and statistical visualizations. Here is a quick screenshot showcasing the great aesthetic power of ggplot2:

```
econ <- ggplot(data = dat, aes(x=CPI, y=HDI)) +
  geom_point(aes(color=Region), size=1.5, fill=4, shape=1, stroke = 1.2) +
  geom_smooth(method = "lm", formula = y ~ log(x), se=FALSE, color="red",
linetype=1, weight=2) +
  geom_text_repel(aes(label=Country), data = dat[dat$Country %in% countryList,], s
ize=2.2) +
  scale_x_continuous(name="Corruption Perceptions Index, 2011(10=least corrupt)",
breaks=seq(1,10,by=1), limits=c(1,10)) +
  scale_y_continuous(name="Human Development Index, 2011(1=best)", breaks=seq(0,
1.0, by=0.1), limits= c(0.2,1.0)) +
  scale_color_manual(name = "", values = c("#2d6074", "#29adde", "#bbe9fb", "#188c
81", "#f2523a", "#7c2510"), labels=c("EU W. Europe"="OECD", "Americas" = "America
s", "Asia Pacific"="Asia & Oceania", "East EU Cemt Asia"="Central & Eastern Europ
e", "MENA"="Middle East & north Africa", "SSA"="Sub-Saharan Africa")) +
  labs(title="Corruption and human development") +
  guides(color=guide_legend(nrow=1)) +
  econ_theme
econ
```



As the most elegant and aesthetically pleasing graphics framework available in R, ggplot2 has been used in many places and for different objectives. Therefore, it is critical to dig deeper with ggplot2 to understand what type of visualization to use for a certain type of question and how to implement it in R using ggplot2. The other thing that sets ggplot2 apart from other plots is that the way we make plots in ggplot2 is very different from base graphics in R. Therefore, it's best to leave what we know about those base graphics behind for now. This interactive post is divided into three sections:

- **Section 1: A Quick Recap of ggplot2. This section covers the basic set-up of ggplots and simple modifications for the components and aesthetics.**
- **Section 2: Advanced Modification and Customization. This section covers advanced techniques and aesthetic features like styling legend title, text, key, and removing and changing legend positions that haven't been emphasized a lot in the lecture.**
- **Section 3: A List of Most Frequently Used ggplot2 Visualizations. Based on the previous two sections, this section covers different types of ggplots, including advanced counts chart, marginal histogram / boxplot and even spatial visualization using google maps that haven't been discussed in class before.**

Given the level of complexity in section 2 and 3, this post is primarily geared towards Stats 133 students at Berkeley and those who have some knowledge of the R programming language and want to make advanced graphs based on complex data sets.

Examples and Discussion — Section 1

Since the reason why people choose ggplot2 over base graphs in R has already been thoroughly discussed in the lecture, I'll instead go over a quick summary of basic terminologies for ggplot2 to refresh your mind and get you started.

- ggplot: the main function where you specify the dataset and variables to plot
- geoms: geometric objects
    - geom point(), geom bar(), geom density(), geom line(), geom area()

- aes: aesthetics
    - shape, transparency (alpha), color, fill, linetype
- scales: determines how the data will be plotted
    - continuous, discrete, log

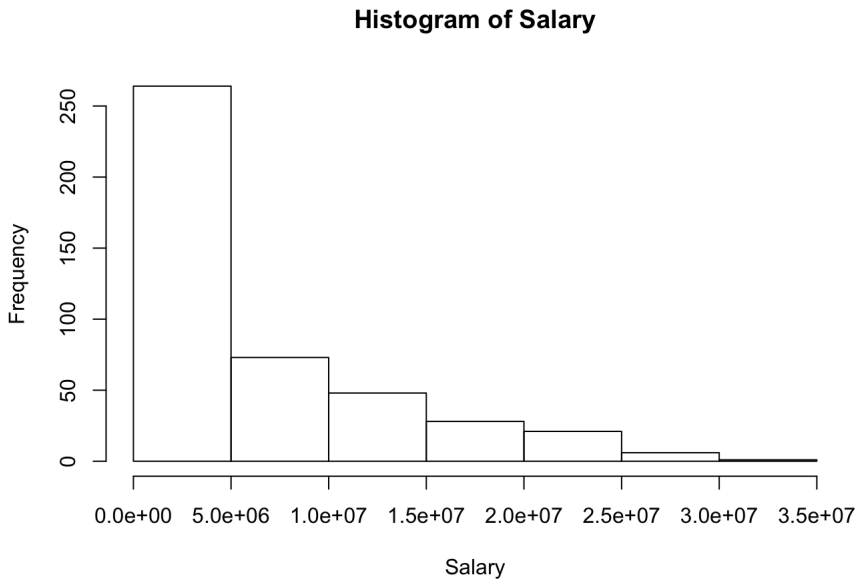
During class, we've briefly touched upon the comparison between ggplot2 and base graphics-as we mentioned, people choose ggplot2 over base graphics because the former tends to be less laborious and verbose than the latter. Let's dig in a little deeper and try to understand the dynamic relationship between ggplot2 and base graphics.

ggplot2 VS Base for **simple** graphs Basic set-up:

```
# Import NBA player's data for this illustration
data <- read.csv('/Users/zoey/Desktop/stat133/stat133-hws-fall17/post01/data/nba2017-player-statistics.csv', colCl
asses = c("character", "character", "factor", "character", "double", rep("integer", 19)), stringsAsFactors = FALSE
)
# Only want first six rows and first five columns
data1 <- data[1:5]
# Set up new experience variable
data1$Experience[data1$Experience == "R"] <- "0"
data1$Experience <- as.integer(data1$Experience)
```

For simple graphs, say we want to construct a histogram out of `data1`. Using base graphics, we only need one line of code:

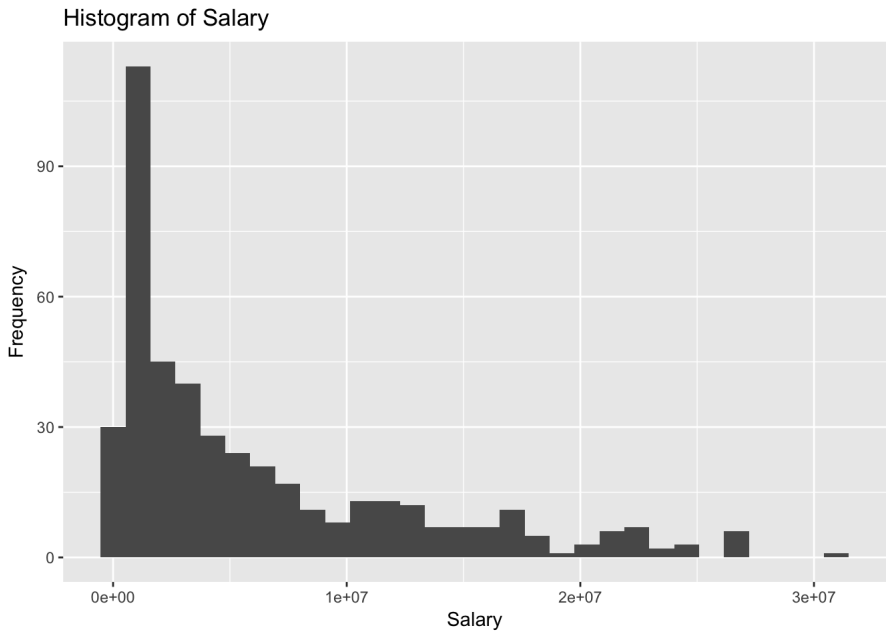
```
# We want to see the frequency for players' salaries using a histogram
hist(data1$Salary, xlab = "Salary", main = "Histogram of Salary")
```



Using ggplot2, however, we'll need a lot more:

```
library(ggplot2)
  ggplot(data1, aes(x = Salary)) +
  geom_histogram() +
  ggtitle("Histogram of Salary") +
  labs(x = "Salary", y = "Frequency")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

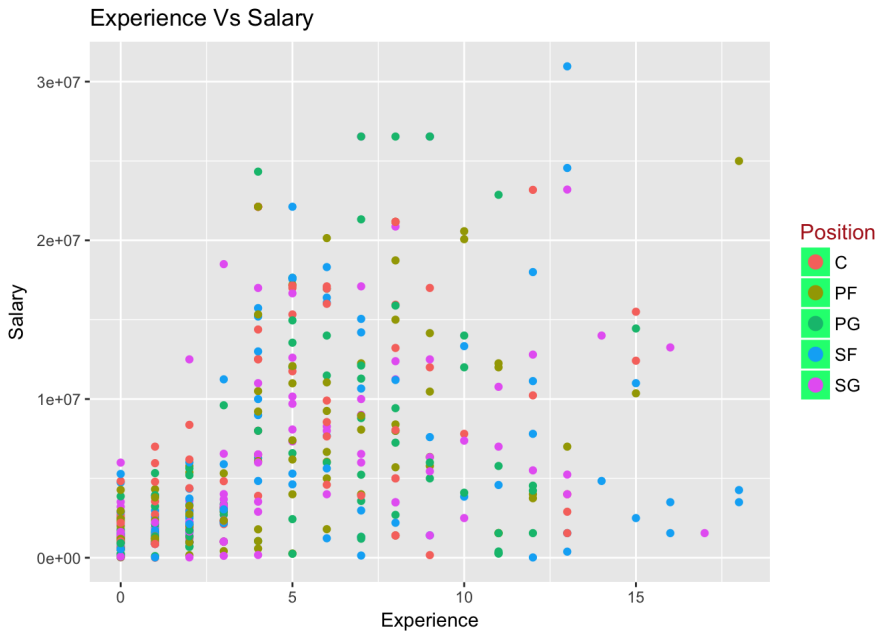


Therefore, it's not always true that we should choose ggplot2 over base graphics because, for simple graphs, base graphics does win in terms of efficiency. Feel free to go on to https://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/ if you want to know more about nuances of ggplot2 and base r graphics.

Examples and Discussion — Section 2

To style legend title, text and key, we need to use `element_text()` and `element_rect()` functions. See the example below, where we not only have appropriate legend, but also **a new style**.

```
library(ggplot2)
gg1 <- ggplot(data1, aes(x=Experience, y=Salary)) +
  geom_point(aes(col=Position)) +
  labs(title="Experience Vs Salary", y="Salary", x="Experience")

gg1 + theme(legend.title = element_text(size=12, color = "firebrick"),
           legend.text = element_text(size=10),
           legend.key=element_rect(fill='springgreen')) +
  guides(colour = guide_legend(override.aes = list(size=2, stroke=1.5)))
```

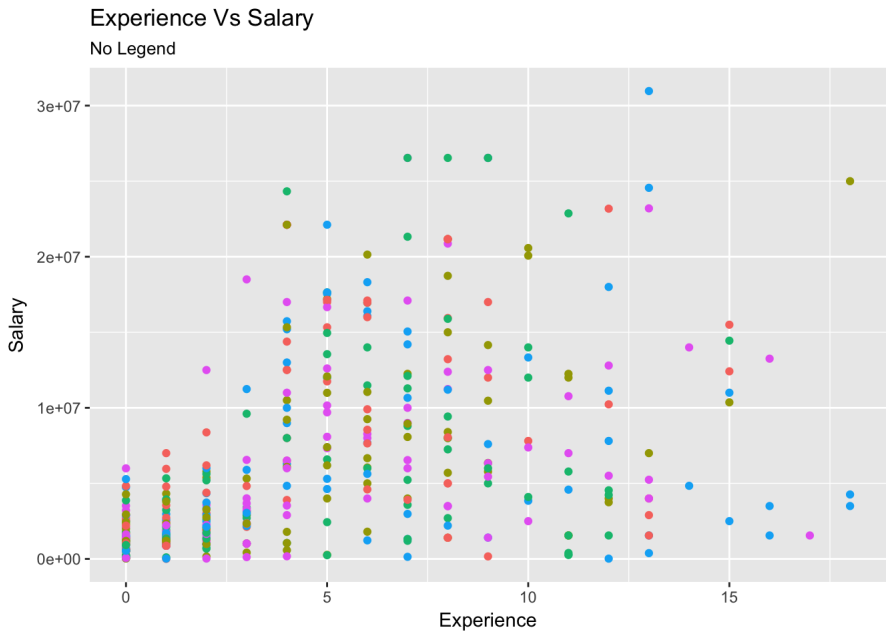


Now, we know how to make the legend prettier. However, what if we have to remove or change legend positions because of certain layout issues during actual publications? Therein lies the importance of knowing how to dynamically manipulate the legend in ggplot2 in case we need to make any adjustments. With different lines of code below, we can **move things around easily**.

Note that the legend's position inside the plot is modified using the `theme()` function. A quick list below specifying different functions we need move the legend around:

- `legend.position` represents the x and y axis position in chart area
  - (0,0) is bottom left of the chart
  - (1,1) is top right of the chart
- `legend.justification` represents the hinge point inside the legend. To move the legend inside the plot, we need to additionally control the hinge point of the legend using `legend.justification`.

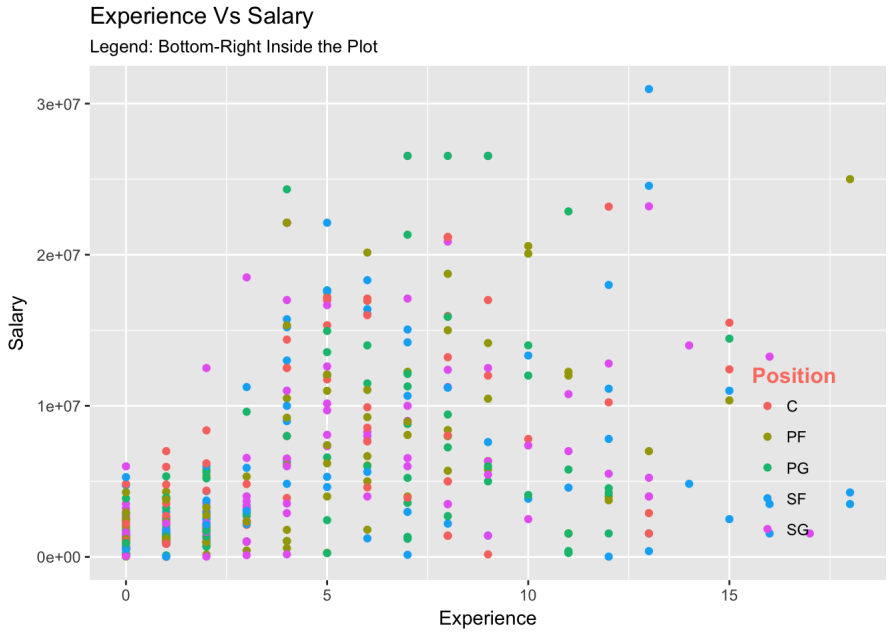
```
library(ggplot2)
gg1 <- ggplot(data1, aes(x=Experience, y=Salary)) +
  geom_point(aes(col=Position)) +
  labs(title="Experience Vs Salary", y="Salary", x="Experience")

# No legend
gg1 + theme(legend.position="None") + labs(subtitle="No Legend")
```




```
# Legend to the left
gg1 + theme(legend.position="left") + labs(subtitle="Legend on the Left")
```


```
# legend at the bottom and horizontal
gg1 + theme(legend.position="bottom", legend.box = "horizontal") + labs(subtitle="Legend at Bottom")
```


```
# legend at bottom-right, inside the plot.
gg1 + theme(legend.title = element_text(size=12, color = "salmon", face="bold"),
            legend.justification=c(1,0),
            legend.position=c(0.95, 0.05),
            legend.background = element_blank(),
            legend.key = element_blank()) +
  labs(subtitle="Legend: Bottom-Right Inside the Plot")
```




```
# legend at top-left, inside the plot
gg1 + theme(legend.title = element_text(size=12, color = "salmon", face="bold"),
            legend.justification=c(0,1),
            legend.position=c(0.05, 0.95),
            legend.background = element_blank(),
            legend.key = element_blank()) +
  labs(subtitle="Legend: Top-Left Inside the Plot")
```

For more manipulation on theme and legend, watch "How to change the themes of ggplot2 graphs in R and RStudio" by Sheffield Methods Institute at https://www.youtube.com/watch?v=DuSBUAFPjxg

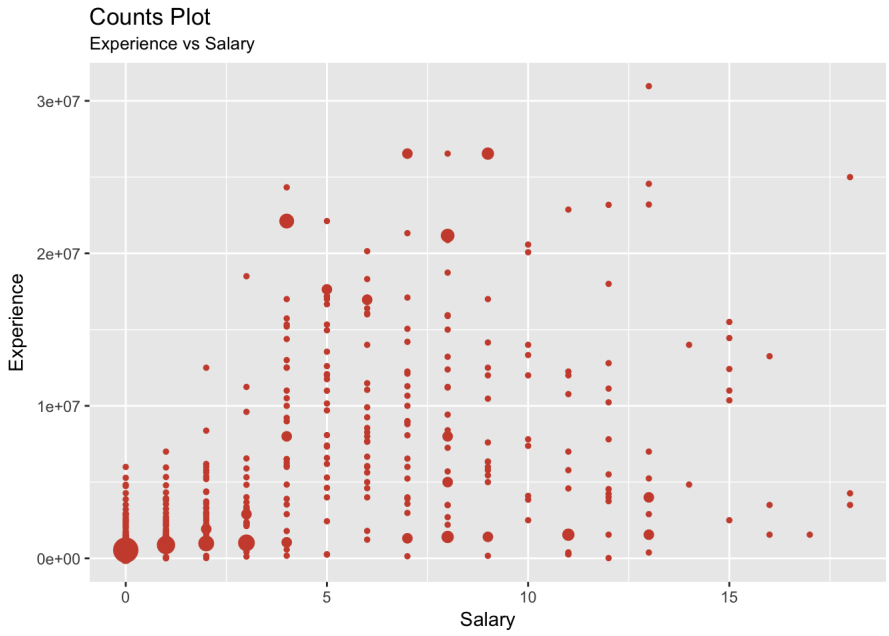
Examples and Discussion — Section 3

Combining what we learned in section 1 and 2, we can take a step further and summarize different types of ggplots in this final section, including advanced counts chart, marginal histogram / boxplot, and real application of google maps.

1. Count plot

As is known to all Stats students, scatterplot is the most widely used plot. Whenever we want to scrutinize the relationship between two variables we set up the scatterplot. **However, problem comes when data points accidentally overlap with each other. The best option for this case is counts plot** in which the size of the circle will get larger as more points overlap. We use `geom_count` to accomplish this.

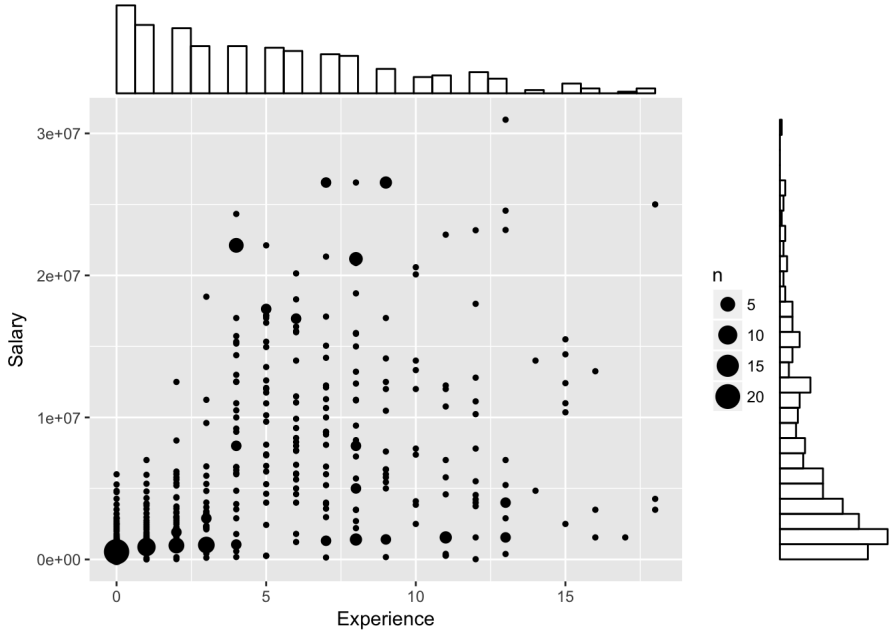
```
# Scatterplot
gg2 <- ggplot(data1, aes(x=Experience, y=Salary))
gg2 + geom_count(col="tomato3", show.legend=F) +
  labs(subtitle="Experience vs Salary",
       y="Experience",
       x="Salary",
       title="Counts Plot")
```



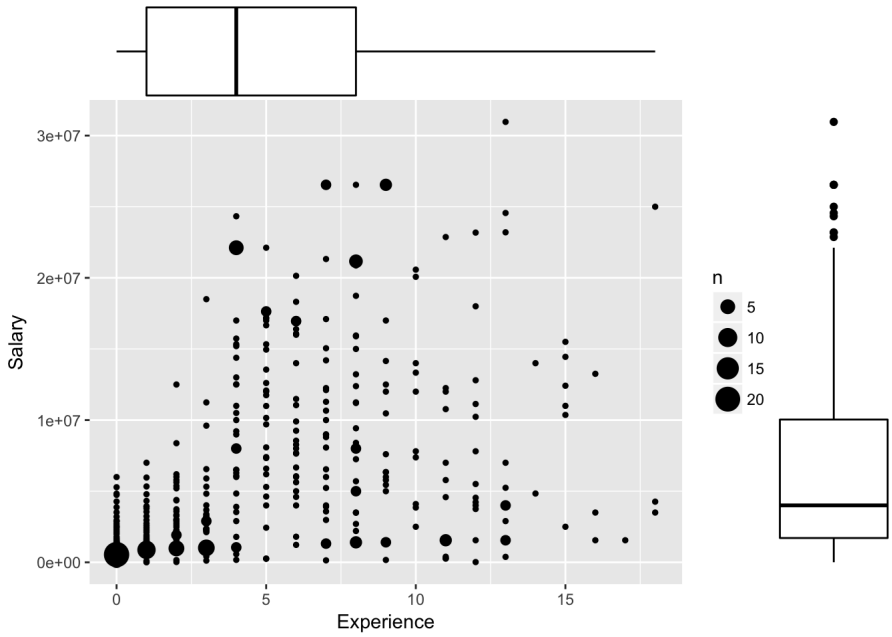
2. Marginal Histogram / Boxplot

We use scatterplot when we want to see the relationship between two variables histogram / boxplot when we want to see the distribution between two variables. **But what if we want to demonstrate both relationship and distribution in the same chart? That's when marginal histogram / boxplot comes into play.**

```r
# load package and data
library(ggExtra)
gg2 <- ggplot(data1, aes(x=Experience, y=Salary)) +
  geom_count()
  ggMarginal(gg2, type = "histogram", fill="transparent")
```




```r
# load package and data
library(ggExtra)
gg2 <- ggplot(data1, aes(x=Experience, y=Salary)) +
  geom_count()
ggMarginal(gg2, type = "boxplot", fill="transparent")
```



3. Spatial visualization — ggmap

Besides the statistical relationship between variables, ggplot2 is also helpful for real-life spatial visualization. The ggmap package is able to interact with the google maps API (application programming interface) and get the latitude and longitude of places we want to pinpoint. The simple example of the road map for Berkeley shows the useful application of ggplot2 in real life. Here is a list of important functions in the code chunk below:

- `geocode()` is used to get the coordinates of these places
- `qmap()` is used to get the maps.
    - the value of `maptype` determines the type of map we get.
- the default zoom-in factor is 10, which is suitable for large cities.

```r
# load packages
library(ggmap)
library(ggalt)
# Get Berkeley 's Coordinates
# get longitude and latitude
berkeley <-  geocode("Berkeley")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Berkeley&sensor=false
```

```
# Get the Map
# Google Road Map
berkeley_ggl_road_map <- qmap("Berkeley", zoom=12, source = "google", maptype="roadmap")
```

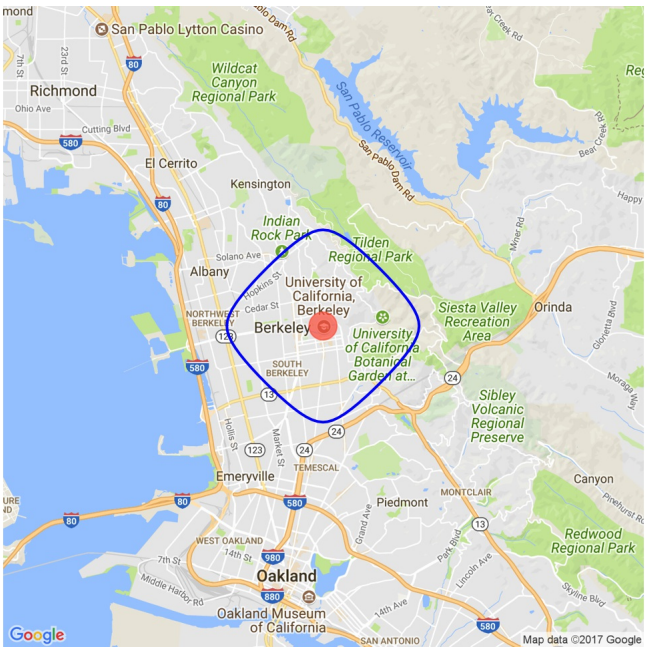
```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Berkeley&zoom=12&size=640x640&scale=2&mapty
pe=roadmap&language=en-EN&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Berkeley&sensor=false
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```

```
# get longitudes and latitudes
places_loc <- geocode("Berkeley")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Berkeley&sensor=false
```

```
# Plot Google Road Map
berkeley_ggl_road_map + geom_point(aes(x=lon, y=lat),
                                   data = places_loc,
                                   alpha = 0.7,
                                   size = 7,
                                   color = "tomato") +
                        geom_encircle(aes(x=lon, y=lat),
                                      data = places_loc, size = 2, color = "blue")
```



It's pretty cool, isn't it? For those who want to dig deeper into ggmap, I highly recommend looking into this pdf file at http://stat405.had.co.nz/ggmap.pdf

Conclusion — Take-home Message

> Although base graphics in R offers a good set of plots for simple data visualizations, some of them still require a lot of work. By contrast, ggplot2 is extremely flexible and efficient. With this three-part post on ggplot2, I hope you not only got a more comprehensive understanding of the aesthetic power of ggplot2, but also started to know what type of plot to use for what sort of problem. For example, like what was mentioned before in section 3 — what if data points overlap with each other when we are trying to make the scatterplot? What if we want to demonstrate both relationship and distribution in the same chart? What if we want to generate a map pinpointing target cities? You'll encounter all kinds of different situations when visualizing data, so it is critical to choose the right type of plot for the specific objectives.

References

- http://www.economist.com/node/21541178
- http://media.economist.com/sites/default/files/imagecache/original-size/20111210_WOC210.gif
- https://www.quora.com/What-software-package-does-The-Economist-group-use-to-create-charts-and-graphs-for-the-magazine
- https://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/
- https://www.youtube.com/watch?v=DuSBUAFPjxg
- http://stat405.had.co.nz/ggmap.pdf

- https://www.youtube.com/watch?v=LHb8eYXMuwU