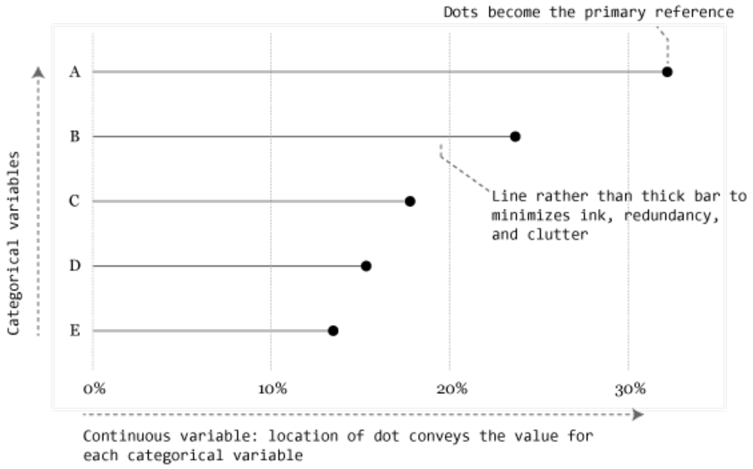
# Data Visualization: Lollipop Plot

## Introduction

### A. Background of lollipop plot.

A **lollipop plot** or **lollipop chart** is a hybrid between a **barplot** and a **scatter plot**. Usually, it displays the relationship between two variables, and they could be two numerical variables or a numerical variable and a categorical variable. The plot typically contains categorical variables on the y-axis measured against the second variable on the x-axis. The emphasis of the plot is on the dot used to draw viewers' attention to the specific value on x-axis achieved by each category. The line is meant to be a minimalistic approach to easily tie each category to its relative point. A lollipop plot is great for comparing multiple categories. In addition, we could also include **Cleveland dot plots** in our plot to compare values for each group.

Here is a graph to show the information about the lollipop plot



### B. Motivation of this post.

Recently we talked about data visualization and we can create variety graphs using R to help analyse data. In the previous post I showed how to make one type of plot with R, now I am going to show how to produce a hybrid between two type of plots with R.

## Data Preparation

Version of R:

```
version
```

```
##                _                           
## platform       x86_64-apple-darwin15.6.0   
## arch           x86_64                      
## os             darwin15.6.0                
## system         x86_64, darwin15.6.0        
## status                                     
## major          3                           
## minor          4.2                         
## year           2017                        
## month          09                          
## day            28                          
## svn rev        73368                       
## language       R                           
## version.string R version 3.4.2 (2017-09-28)
## nickname       Short Summer
```

Package required:

```
library(dplyr)      # for data manipulation
```

```
## 
## 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
```

```
library(tidyr)  # for data tidying
library(ggplot2)    # for creating plots
```

Data set I choose for this post is the built-in data set `midwest`:

```
# display data
data("midwest")
head(midwest)
```

```
## # A tibble: 6 x 28
##     PID    county state  area poptotal popdensity popwhite popblack
##   <int>     <chr> <chr> <dbl>    <int>      <dbl>    <int>    <int>
## 1   561     ADAMS    IL 0.052    66090  1270.9615    63917     1702
## 2   562 ALEXANDER    IL 0.014    10626   759.0000     7054     3496
## 3   563      BOND    IL 0.022    14991   681.4091    14477      429
## 4   564     BOONE    IL 0.017    30806  1812.1176    29344      127
## 5   565     BROWN    IL 0.018     5836   324.2222     5264      547
## 6   566    BUREAU    IL 0.050    35688   713.7600    35157       50
## # ... with 20 more variables: popamerindian <int>, popasian <int>,
## #   popother <int>, percwhite <dbl>, percblack <dbl>, percamerindan <dbl>,
## #   percasian <dbl>, percother <dbl>, popadults <int>, perchsd <dbl>,
## #   percollege <dbl>, percprof <dbl>, poppovertyknown <int>,
## #   percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

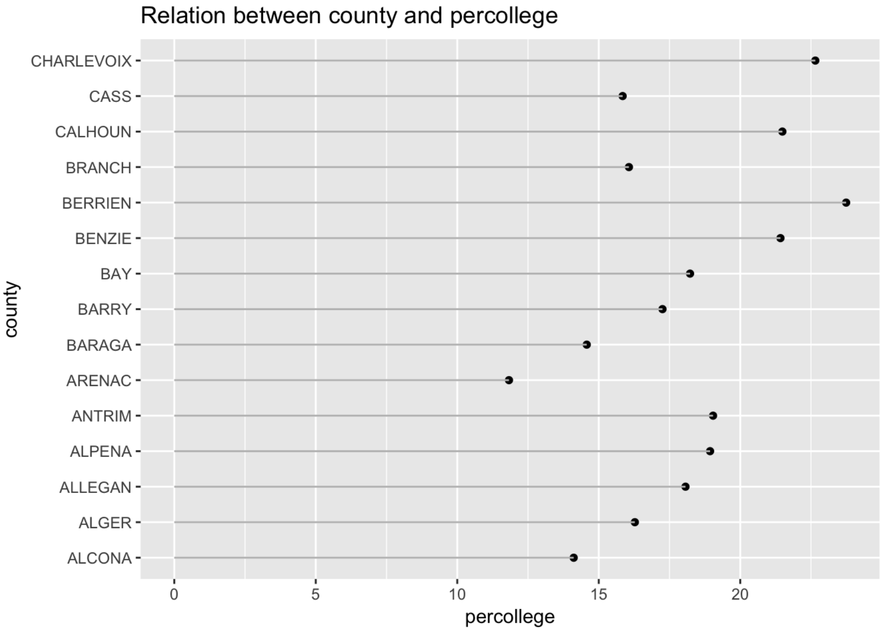
# Some examples of lollipop plots

## 1. Basic Lollipop Plots

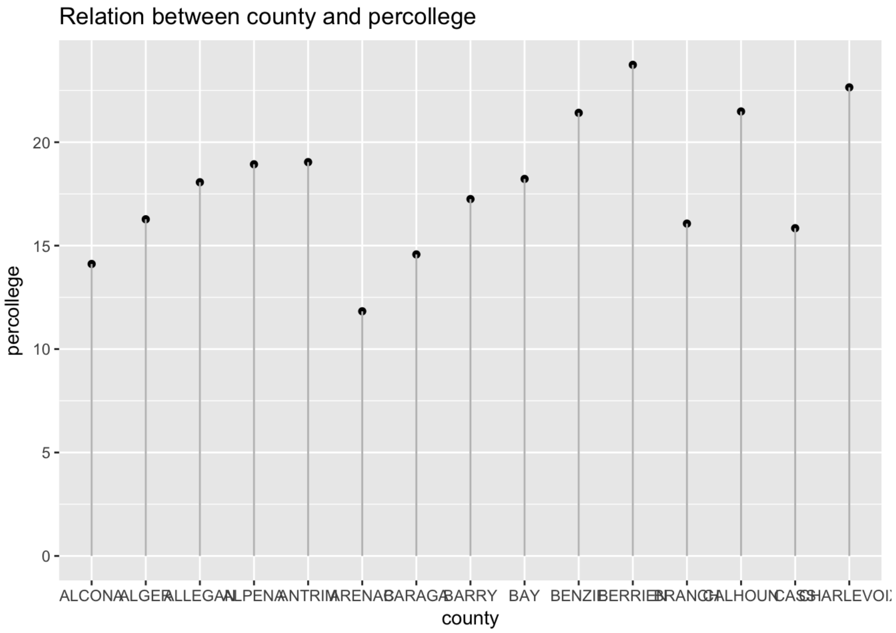
Firstly, I am going to introduce how to produce the basic lollipop plots. I will show the relationship between the first 15 counties in Michigan and the percentage of college educated population by using variables `county` and `percollege`. We use `ggplot`, `geom_point`, and `geom_segment` in our code for generating the plots:

```
# create dataset for first 15 counties in Michigan
michigan_top15 <- midwest %>%
  filter(state == "MI") %>%
  select(county, percollege) %>%
  slice(1:15) %>%
  mutate(county = factor(county, levels = .$county))
```

```
# plot the relationship between county and percollege
ggplot(data = michigan_top15, aes(x = percollege, y = county)) +
  geom_point() +
  geom_segment(aes(x = 0, xend = percollege, y = county, yend = county), color = "grey") +
  ggtitle("Relation between county and percollege")
```



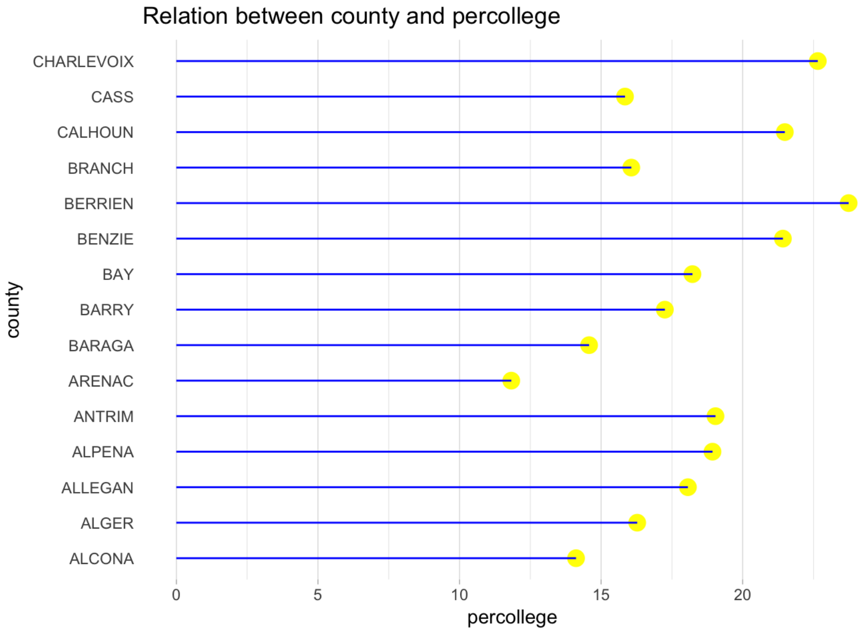
```
# flip the plot by exchange the inputs for x and y:
ggplot(data = michigan_top15, aes(x = county, y = percollege)) +
  geom_point() +
  geom_segment(aes(x = county, xend = county, y = 0, yend = percollege), color = "grey") +
  ggtitle("Relation between county and percollege")
```



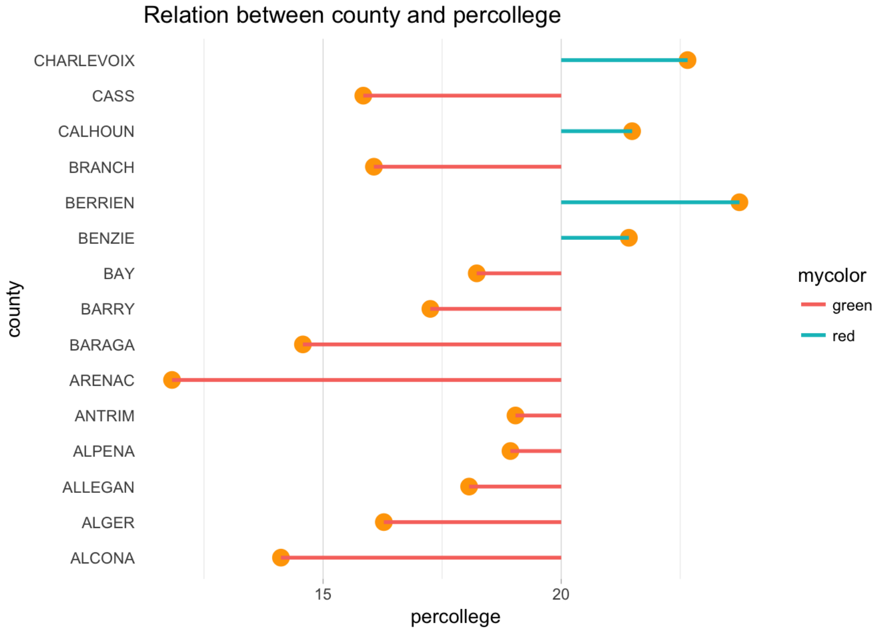
## 2. Custom Lollipop Plots

With the basic plots we can custom the general layout using the `theme()` function. By changing the input `color` and `size` we can custom the color of the plots. The arguments in the `theme()` function allow us to custom the background of the plots:

```
# blank background
ggplot(data = michigan_top15, aes(x = percollege, y = county)) +
  geom_point(color = "yellow", size = 4) +
  geom_segment(aes(x = 0, xend = percollege, y = county, yend = county), color = "blue") +
  theme_light() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()
  ) +
  ggtitle("Relation between county and percollege")
```



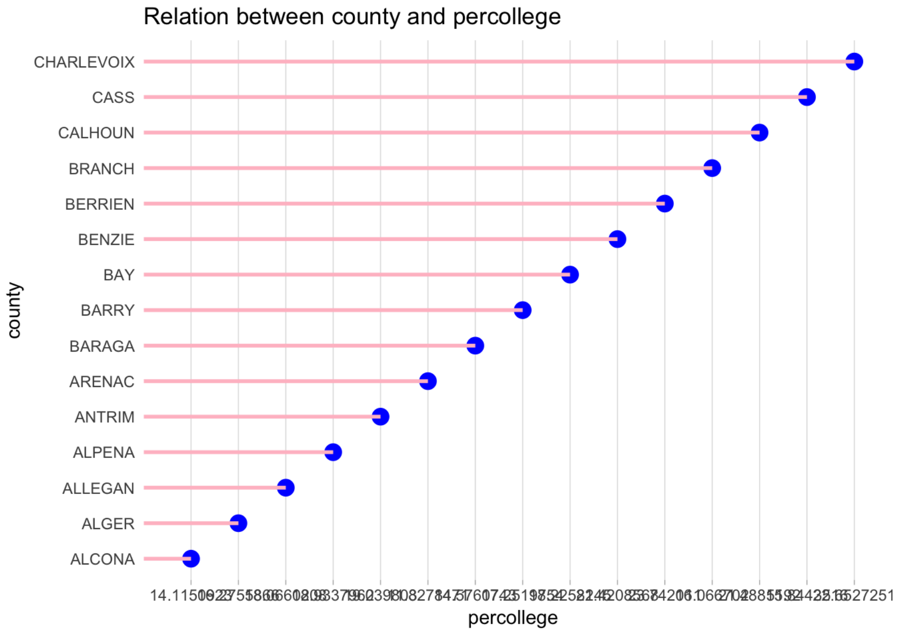
```r
# create a condition for the colors you want
michigan_top15 = michigan_top15 %>%
  mutate(mycolor = ifelse(percollege > 20, "red", "green"))


# add the conditioned colors to the input aes() in function geom_segment() to control the colors
ggplot(data = michigan_top15, aes(x = percollege, y = county)) +
  geom_point(color = "orange", size = 4) +
  geom_segment(aes(x = 20, xend = percollege, y = county, yend = county, color = mycolor), size = 1) +
  theme_light() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()
  ) +
  ggtitle("Relation between county and percollege")
```



## 3. Reorder Lollipop Plots

With the data operation in package `dplyr`, we could do the reorder and then produce the plots. Function I used here are : `arrange()` and `mutate()`, to reorder rows and add new variables respectively.

```r
# reorder the plot
michigan_top15 %>%
  arrange(county) %>%
  mutate(percollege = factor(percollege, percollege)) %>%
  ggplot(aes(x = percollege, y = county)) +
  geom_point(color = "blue", size = 4) +
  geom_segment(aes(x = 0, xend = percollege, y = county, yend = county), color = "pink", size = 1) +
  theme_light() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()
  ) +
  ggtitle("Relation between county and percollege")
```

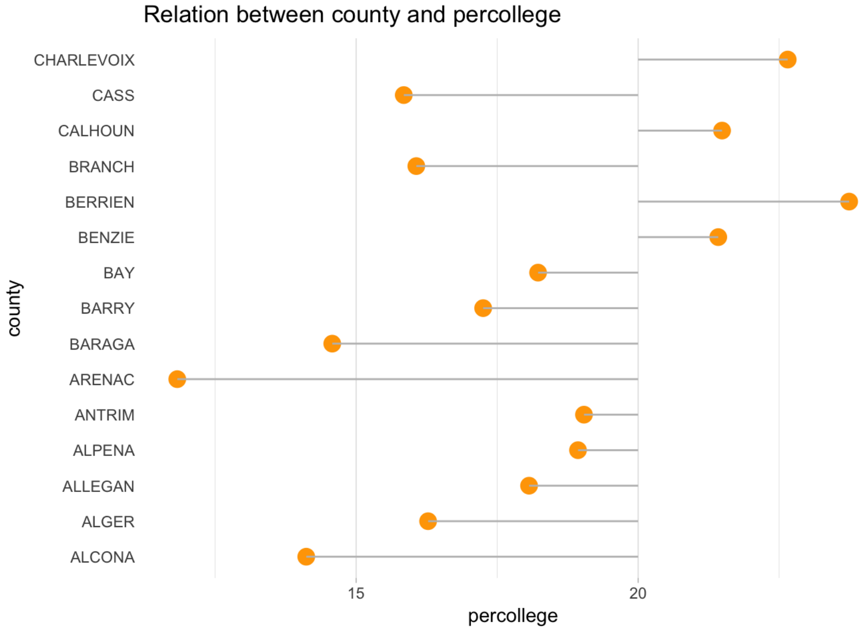


## 4. Change Baseline

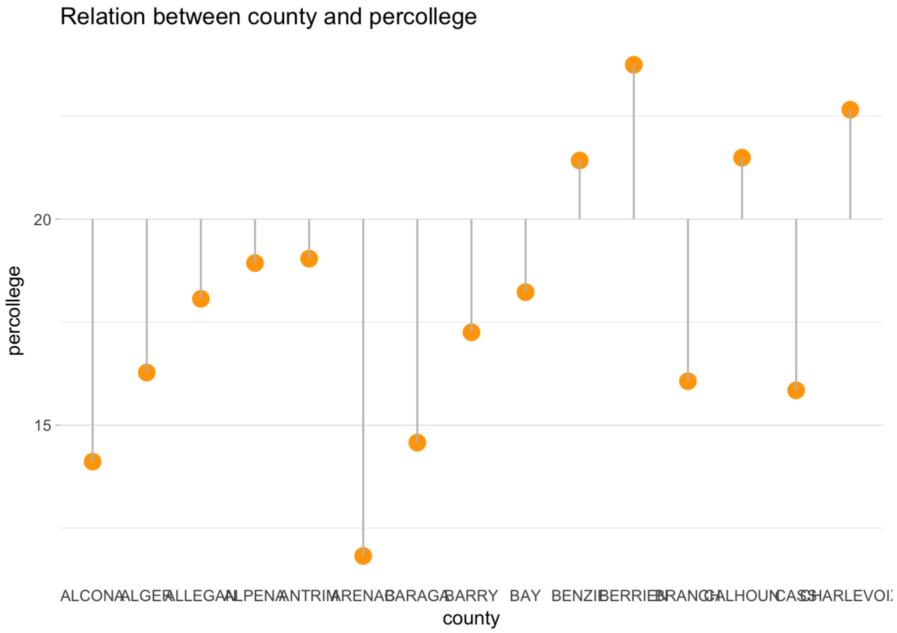
By changing the value of `x` in the function `geom_segment()` we can change the baseline of the graph to anywhere we want. For the vertical plot, the value we need to change is `y` in the function `geom_segment()`.

Here are the examples:

```
ggplot(data = michigan_top15, aes(x = percollege, y = county)) +
  geom_point(color = "orange", size = 4) +
  geom_segment(aes(x = 20, xend = percollege, y = county, yend = county), color = "grey") +
  theme_light() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()
  ) +
  ggtitle("Relation between county and percollege")
```



```
ggplot(data = michigan_top15, aes(x = county, y = percollege)) +
  geom_point(color = "orange", size = 4) +
  geom_segment(aes(x = county, xend = county, y = 20, yend = percollege), color = "grey") +
  theme_light() +
  theme(
    panel.grid.major.x = element_blank(),
    panel.border = element_blank(),
    axis.ticks.x = element_blank()
  ) +
  ggtitle("Relation between county and percollege")
```



## Conclusion

The purpose of this post is to show how to produce lollipop plot with R and it is useful in data analysis. In spite of the examples I showed in this post, there are also variety of complex plots we could produce. After reading my post, I hope this will be useful for you and have the basic knowledge of lollipop plots.

## Reference:

- Detailed information about `geom_segment()` function usage: geom_segment
- Detailed information about `theme()` function in `ggplot2` : theme
- For the package `dplyr` , see Introduction to dplyr for more information.
- Information about Lollipop charts: Lollipop chart
- More examples about Lollipop chart:

  Beating lollipops into dumbbells

  examples

  Lollipop Charts