1. Correlation

install.packages("ggplot2")

library(ggplot2)

data("midwest")

midwest

### **Scatterplot**

# Scatterplot visulization

gg <- ggplot(midwest, aes(x=area, y=poptotal)) +

geom\_point(aes(col=state, size=popdensity)) +

geom\_smooth(method="loess", se=F) +

xlim(c(0, 0.1)) +

ylim(c(0, 500000)) +

labs(subtitle="Area Vs Population",

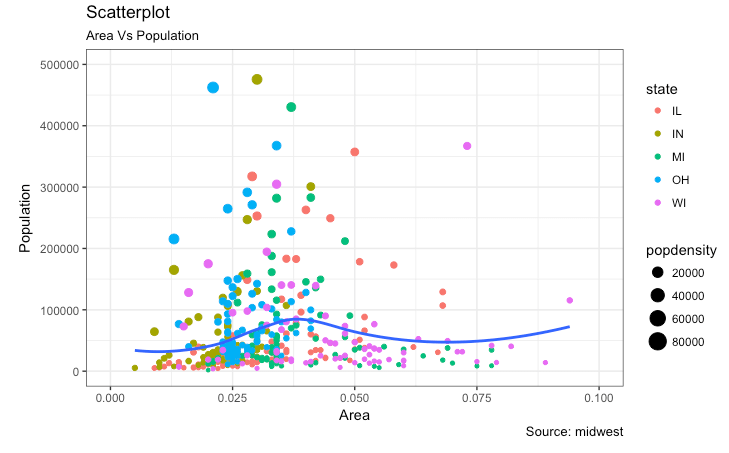
y="Population",

x="Area",

title="Scatterplot",

caption = "Source: midwest")

plot(gg)



### **Scatterplot With Encircling**

When presenting the results, sometimes I would encirlce certain special group of points or region in the chart so as to draw the attention to those peculiar cases. This can be conveniently done using the geom\_encircle() in ggalt package.

# Scatterplot With Encircling visulization

install.packages('ggalt')

library(ggalt)

# data gathering

midwest\_select <- midwest[midwest$poptotal > 350000 &

midwest$poptotal <= 500000 &

midwest$area > 0.01 &

midwest$area < 0.1, ]

# Plot

ggplot(midwest, aes(x=area, y=poptotal)) +

geom\_point(aes(col=state, size=popdensity)) + # draw points

geom\_smooth(method="loess", se=F) +

xlim(c(0, 0.1)) +

ylim(c(0, 500000)) + # draw smoothing line

geom\_encircle(aes(x=area, y=poptotal),

data=midwest\_select,

color="red",

size=2,

expand=0.08) + # encircle

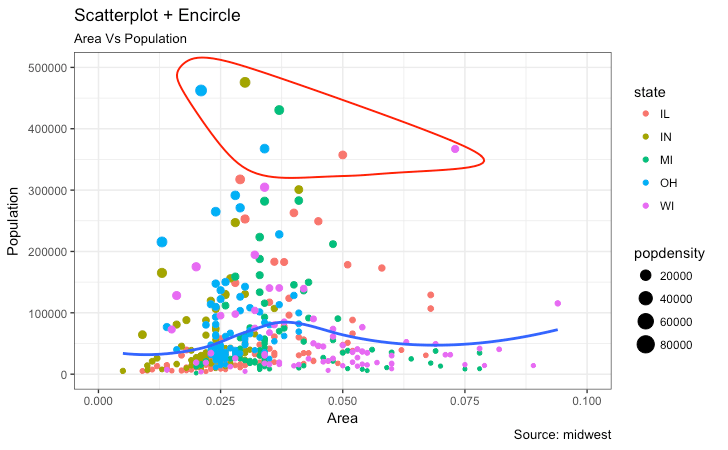
labs(subtitle="Area Vs Population",

y="Population",

x="Area",

title="Scatterplot + Encircle",

caption="Source: midwest")



### **Jitter Plot**

plot city mileage (cty) vs highway mileage (hwy).

# store data

# jitter plot visulization

data(mpg)

# store data

g <- ggplot(mpg, aes(cty, hwy))

# Scatterplot

g + geom\_point() +

geom\_smooth(method="lm", se=F) +

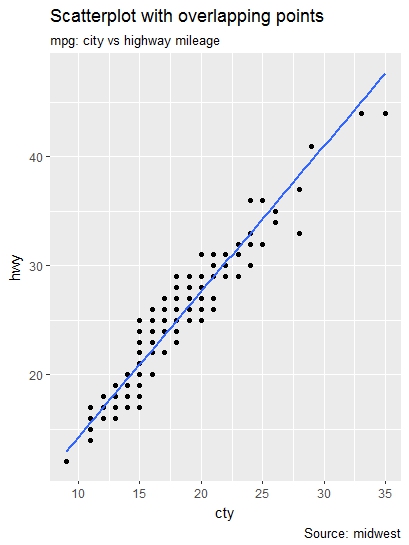
labs(subtitle="mpg: city vs highway mileage",

y="hwy",

x="cty",

title="Scatterplot with overlapping points",

caption="Source: midwest")



The original data has 234 data points but the chart seems to display fewer points. This is because there are many overlapping points appearing as a single dot.

We can make a jitter plot with jitter\_geom(). As the name suggests, the overlapping points are randomly jittered around its original position based on a threshold controlled by the width argument.

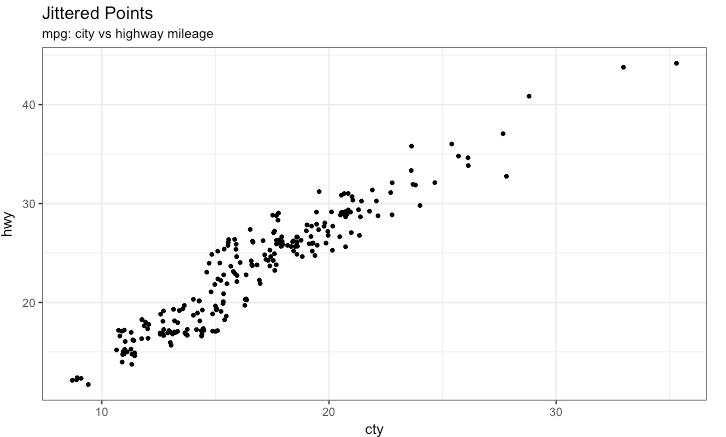
g + geom\_jitter(width = .5, size=1) +

labs(subtitle="mpg: city vs highway mileage",

y="hwy",

x="cty",

title="Jittered Points")



More the width, more the points are moved jittered from their original position.

### **Counts Chart**

The second option to overcome the problem of data points overlap is to use what is called a counts chart. Whereever there is more points overlap, the size of the circle gets bigger.

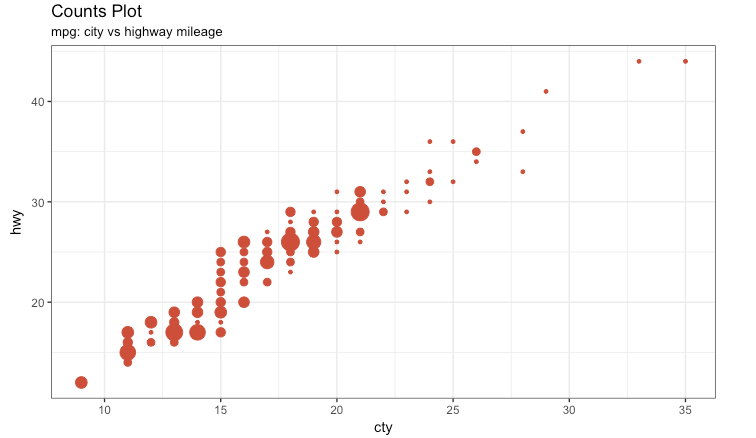
g + **geom\_count**(col="tomato3", show.legend=F) +

**labs**(subtitle="mpg: city vs highway mileage",

y="hwy",

x="cty",

title="Counts Plot")



### **Bubble plot**

While scatterplot lets you compare the relationship between 2 continuous variables, bubble chart serves well if you want to understand relationship within the underlying groups based on:

1. A Categorical variable (by changing the color) and
2. Another continuous variable (by changing the size of points).

if you have 4-Dimensional data where two of them are numeric (X and Y) and one other categorical (color) and another numeric variable (size).

mpg\_select <- mpg[mpg$manufacturer %in% c("audi", "ford", "honda", "hyundai"), ]

g <- ggplot(mpg\_select, aes(displ, cty)) +

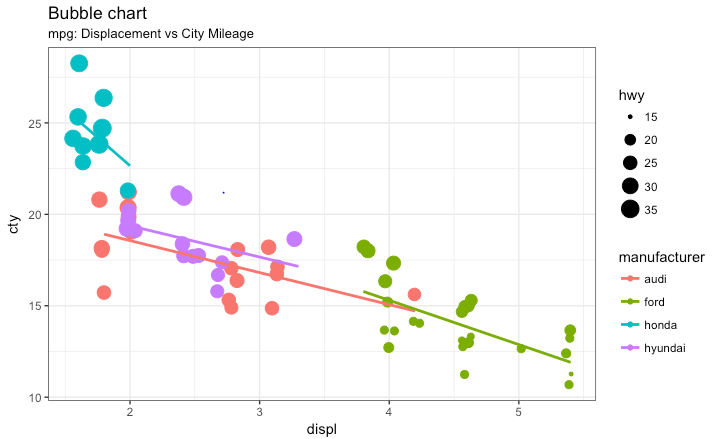
geom\_jitter(aes(col=manufacturer, size=hwy)) +

geom\_smooth(aes(col=manufacturer), method="lm", se=F) +

labs(subtitle="mpg: Displacement vs City Mileage",

title="Bubble chart")

plot(g)



### **Animated Bubble chart**

An animated bubble chart can be implemented using the gganimate package. It is same as the bubble chart, but, you have to show how the values change over a fifth dimension (typically time).

The key thing to do is to set the aes(frame) to the desired column on which you want to animate. Rest of the procedure related to plot construction is the same. Once the plot is constructed, you can animate it using gganimate() by setting a chosen interval.

# Animated Bubble chart visulization

install.packages("cowplot")

install.packages("gganimate")

library(cowplot)

library(gganimate)

library(gapminder)

View(gapminder)

g <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, frame = year)) +

geom\_point() +

geom\_smooth(aes(group = year),

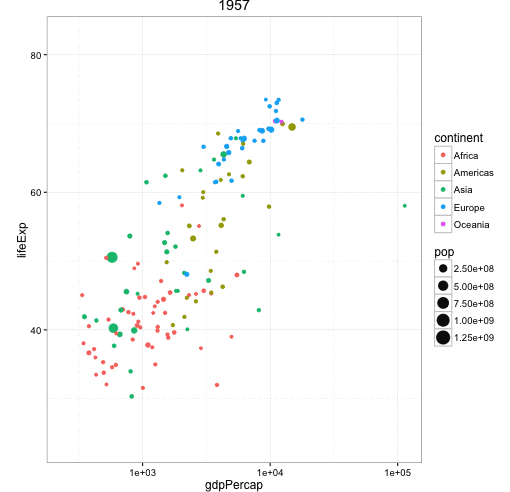
method = "lm",

show.legend = FALSE) +

facet\_wrap(~continent, scales = "free") +

scale\_x\_log10() # convert to log scale

gganimate(g, interval=0.2)



### **Marginal Histogram / Boxplot**

If you want to show the relationship as well as the distribution in the same chart, use the marginal histogram. It has a histogram of the X and Y variables at the margins of the scatterplot.

This can be implemented using the ggMarginal() function from the ‘ggExtra’ package. Apart from a histogram, you could choose to draw a marginal boxplot or density plot by setting the respective typeoption.

# Marginal Histogram / Boxplot visulization

install.packages(ggExtra)

library(ggplot2)

library(ggExtra)

library(ggMarginal)

mpg\_select <- mpg[mpg$hwy >= 35 & mpg$cty > 27, ]

g <- ggplot(mpg, aes(cty, hwy)) +

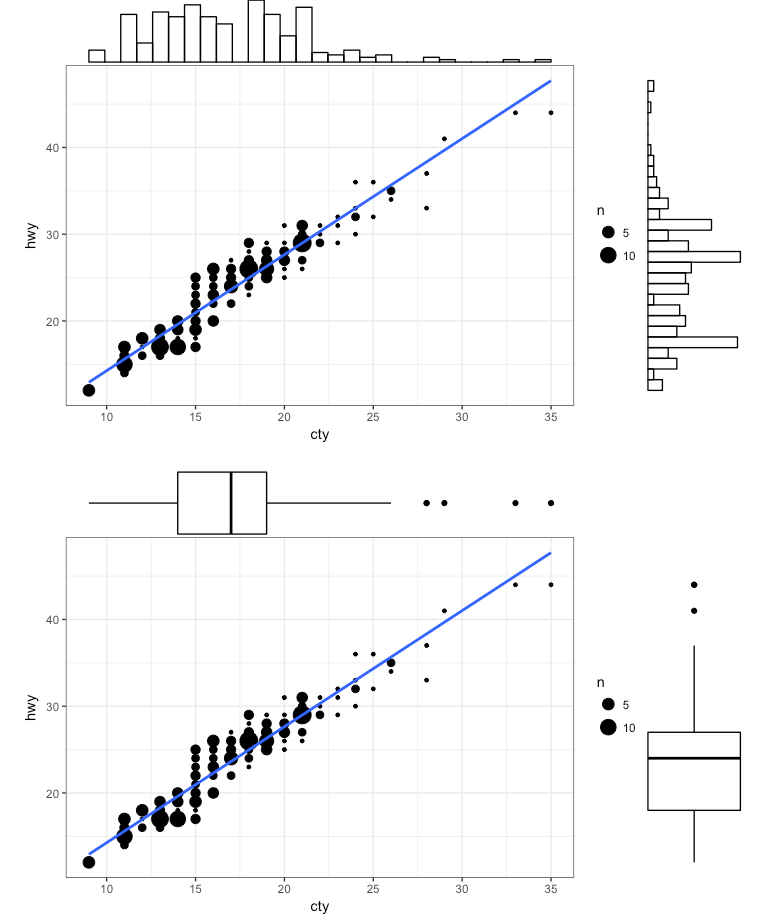
geom\_count() +

geom\_smooth(method="lm", se=F)

ggMarginal(g, type = "histogram", fill="transparent")

ggMarginal(g, type = "boxplot", fill="transparent")

# ggMarginal(g, type = "density", fill="transparent")



### **Correlogram**

 let’s you examine the corellation of multiple continuous variables present in the same dataframe. This is conveniently implemented using the ggcorrplot package

**library**(ggplot2)

**library**(ggcorrplot)

*# Correlation matrix*

install.packages("ggcorrplot")

**data**(mtcars)

corr <- **round**(**cor**(mtcars), 1)

*# Plot*

**ggcorrplot**(corr, hc.order = TRUE,

type = "lower",

lab = TRUE,

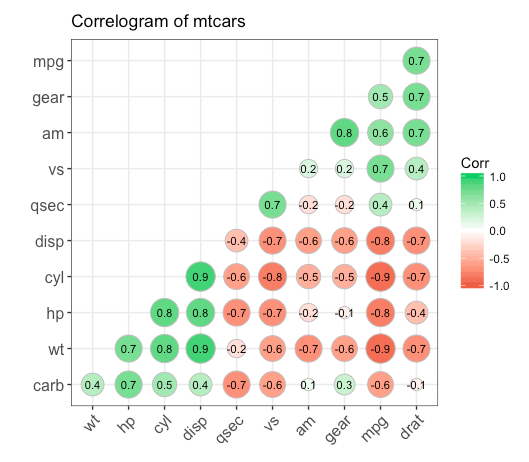
lab\_size = 3,

method="circle",

colors = **c**("tomato2", "white", "springgreen3"),

title="Correlogram of mtcars",

ggtheme=theme\_bw)



## 2. Deviation

### **Diverging bars**

Diverging Bars is a bar chart that can handle both negative and positive values. This can be implemented by a smart tweak with geom\_bar(). But the usage of geom\_bar() can be quite confusing. Thats because, it can be used to make a bar chart as well as a histogram. Let me explain.

By default, geom\_bar() has the stat set to count. That means, when you provide just a continuous X variable (and no Y variable), it tries to make a histogram out of the data.

In order to make a bar chart create bars instead of histogram, you need to do two things.

1. Set stat=identity
2. Provide both x and y inside aes() where, x is either character or factor and y is numeric.

 In below example, the mpg from mtcars dataset is normalised by computing the z score. Those vehicles with mpg above zero are marked green and those below are marked red

**library**(ggplot2)

**theme\_set**(**theme\_bw**())

*# Data Prep*

**data**("mtcars") *# load data*

mtcars$`car name` <- **rownames**(mtcars) *# create new column for car names*

mtcars$mpg\_z <- **round**((mtcars$mpg - **mean**(mtcars$mpg))/**sd**(mtcars$mpg), 2) *# compute normalized mpg*

mtcars$mpg\_type <- **ifelse**(mtcars$mpg\_z < 0, "below", "above") *# above / below avg flag*

mtcars <- mtcars[**order**(mtcars$mpg\_z), ] *# sort*

mtcars$`car name` <- **factor**(mtcars$`car name`, levels = mtcars$`car name`) *# convert to factor to retain sorted order in plot.*

*# Diverging Barcharts*

**ggplot**(mtcars, **aes**(x=`car name`, y=mpg\_z, label=mpg\_z)) +

**geom\_bar**(stat='identity', **aes**(fill=mpg\_type), width=.5) +

**scale\_fill\_manual**(name="Mileage",

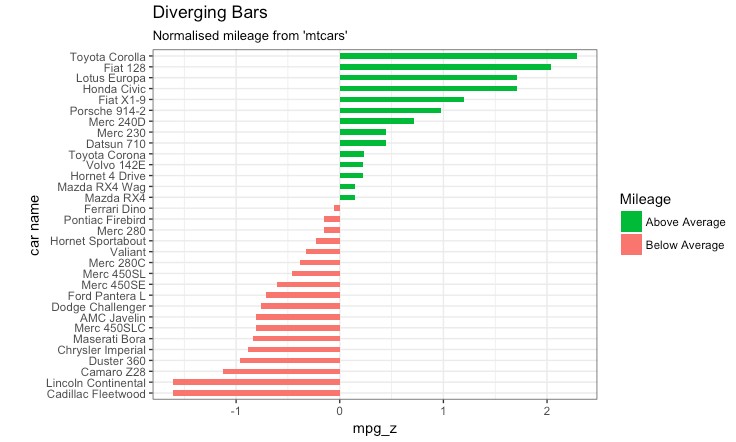
labels = **c**("Above Average", "Below Average"),

values = **c**("above"="#00ba38", "below"="#f8766d")) +

**labs**(subtitle="Normalised mileage from 'mtcars'",

title= "Diverging Bars") +

**coord\_flip**()



### **Diverging Lollipop Chart**

Instead of geom\_bar, I use geom\_point and geom\_segment to get the lollipops right

**library**(ggplot2)

**theme\_set**(**theme\_bw**())

**ggplot**(mtcars, **aes**(x=`car name`, y=mpg\_z, label=mpg\_z)) +

**geom\_point**(stat='identity', fill="black", size=6) +

**geom\_segment**(**aes**(y = 0,

x = `car name`,

yend = mpg\_z,

xend = `car name`),

color = "black") +

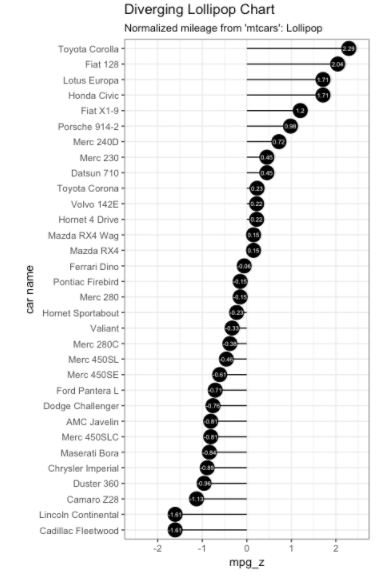
**geom\_text**(color="white", size=2) +

**labs**(title="Diverging Lollipop Chart",

subtitle="Normalized mileage from 'mtcars': Lollipop") +

**ylim**(-2.5, 2.5) +

**coord\_flip**()



### **Diverging Dot Plot**

Dot plot conveys similar information. The principles are same as what we saw in Diverging bars, except that only point are used. Below example uses the same data prepared in the [diverging bars example](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Diverging%20Bars).

**library**(ggplot2)

**theme\_set**(**theme\_bw**())

*# Plot*

**ggplot**(mtcars, **aes**(x=`car name`, y=mpg\_z, label=mpg\_z)) +

**geom\_point**(stat='identity', **aes**(col=mpg\_type), size=6) +

**scale\_color\_manual**(name="Mileage",

labels = **c**("Above Average", "Below Average"),

values = **c**("above"="#00ba38", "below"="#f8766d")) +

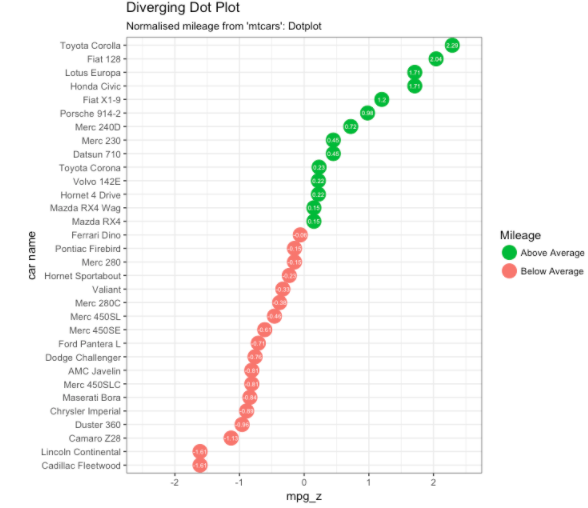
**geom\_text**(color="white", size=2) +

**labs**(title="Diverging Dot Plot",

subtitle="Normalized mileage from 'mtcars': Dotplot") +

**ylim**(-2.5, 2.5) +

**coord\_flip**()



### **Area Chart**

Area charts are typically used to visualize how a particular metric (such as % returns from a stock) performed compared to a certain baseline.

 Other types of %returns or %change data are also commonly used. The geom\_area() implements this.

**library**(ggplot2)

**library**(quantmod)

**data**("economics", package = "ggplot2")

*# Compute % Returns*

economics$returns\_perc <- **c**(0, **diff**(economics$psavert)/economics$psavert[-**length**(economics$psavert)])

*# Create break points and labels for axis ticks*

brks <- economics$date[**seq**(1, **length**(economics$date), 12)]

lbls <- lubridate::**year**(economics$date[**seq**(1, **length**(economics$date), 12)])

*# Plot*

**ggplot**(economics[1:100, ], **aes**(date, returns\_perc)) +

**geom\_area**() +

**scale\_x\_date**(breaks=brks, labels=lbls) +

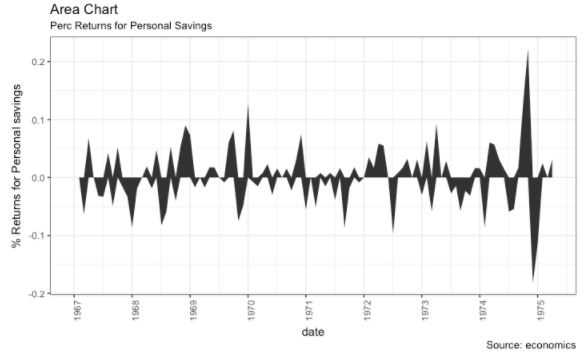
**theme**(axis.text.x = **element\_text**(angle=90)) +

**labs**(title="Area Chart",

subtitle = "Perc Returns for Personal Savings",

y="% Returns for Personal savings",

caption="Source: economics")



3. Ranking

Used to compare the position or performance of multiple items with respect to each other. Actual values matters somewhat less than the ranking.

### **Ordered Bar Chart**

Ordered Bar Chart is a Bar Chart that is ordered by the Y axis variable. Just sorting the dataframe by the variable of interest isn’t enough to order the bar chart. In order for the bar chart to retain the order of the rows, the X axis variable (i.e. the categories) has to be converted into a factor.

*# Prepare data: group mean city mileage by manufacturer.*

cty\_mpg <- **aggregate**(mpg$cty, by=**list**(mpg$manufacturer), FUN=mean) *# aggregate*

**colnames**(cty\_mpg) <- **c**("make", "mileage") *# change column names*

cty\_mpg <- cty\_mpg[**order**(cty\_mpg$mileage), ] *# sort*

cty\_mpg$make <- **factor**(cty\_mpg$make, levels = cty\_mpg$make) *# to retain the order in plot.*

**head**(cty\_mpg, 4)

*#> make mileage*

*#> 9 lincoln 11.33333*

*#> 8 land rover 11.50000*

*#> 3 dodge 13.13514*

*#> 10 mercury 13.25000*

The X variable is now a factor, let’s plot.

**library**(ggplot2)

**theme\_set**(**theme\_bw**())

*# Draw plot*

**ggplot**(cty\_mpg, **aes**(x=make, y=mileage)) +

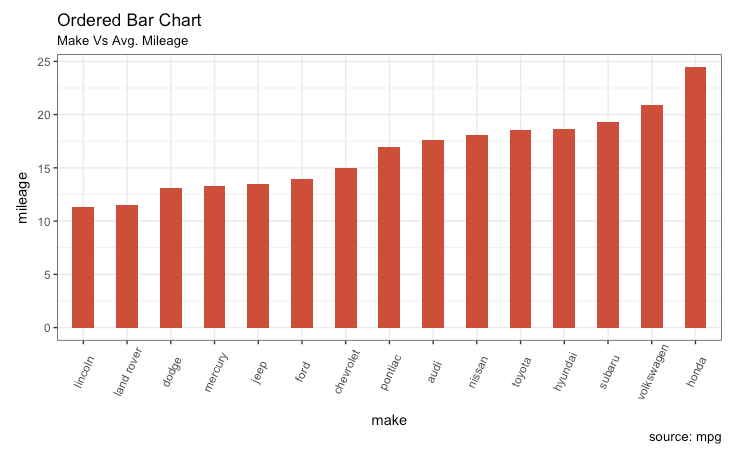
**geom\_bar**(stat="identity", width=.5, fill="tomato3") +

**labs**(title="Ordered Bar Chart",

subtitle="Make Vs Avg. Mileage",

caption="source: mpg") +

**theme**(axis.text.x = **element\_text**(angle=65, vjust=0.6))



### **Lollipop Chart**

Lollipop charts conveys the same information as in bar charts

**library**(ggplot2)

**theme\_set**(**theme\_bw**())

*# Plot*

**ggplot**(cty\_mpg, **aes**(x=make, y=mileage)) +

**geom\_point**(size=3) +

**geom\_segment**(**aes**(x=make,

xend=make,

y=0,

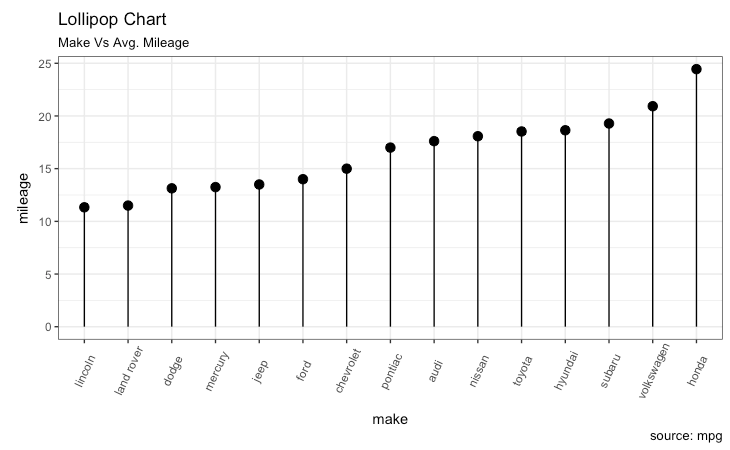
yend=mileage)) +

**labs**(title="Lollipop Chart",

subtitle="Make Vs Avg. Mileage",

caption="source: mpg") +

**theme**(axis.text.x = **element\_text**(angle=65, vjust=0.6))



Dot Plot

Dot plots are very similar to lollipops, but without the line and is flipped to horizontal position. It emphasizes more on the rank ordering of items with respect to actual values and how far apart are the entities with respect to each other.

**library**(ggplot2)

**library**(scales)

**theme\_set**(**theme\_classic**())

*# Plot*

**ggplot**(cty\_mpg, **aes**(x=make, y=mileage)) +

**geom\_point**(col="tomato2", size=3) + *# Draw points*

**geom\_segment**(**aes**(x=make,

xend=make,

y=**min**(mileage),

yend=**max**(mileage)),

linetype="dashed",

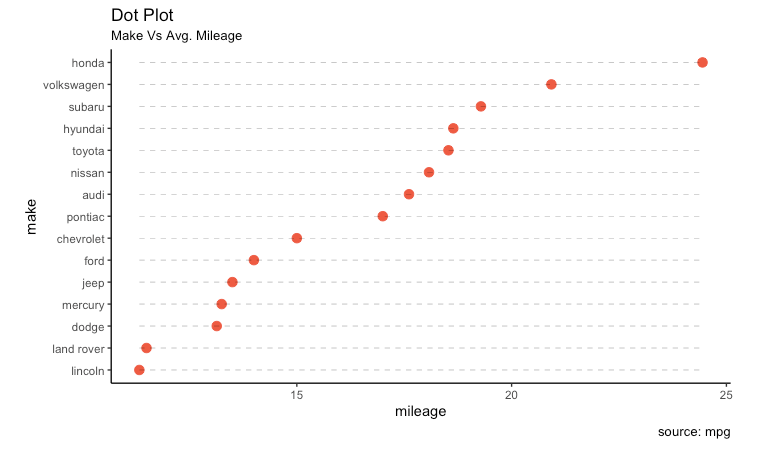
size=0.1) + *# Draw dashed lines*

**labs**(title="Dot Plot",

subtitle="Make Vs Avg. Mileage",

caption="source: mpg") +

**coord\_flip**()



### **Slope Chart**

Slope charts are an excellent way of comparing the positional placements between 2 points on time

**library**(ggplot2)

**library**(scales)

**theme\_set**(**theme\_classic**())

*# prep data*

df <- **read.csv**("https://raw.githubusercontent.com/selva86/datasets/master/gdppercap.csv")

**colnames**(df) <- **c**("continent", "1952", "1957")

left\_label <- **paste**(df$continent, **round**(df$`1952`),sep=", ")

right\_label <- **paste**(df$continent, **round**(df$`1957`),sep=", ")

df$class <- **ifelse**((df$`1957` - df$`1952`) < 0, "red", "green")

*# Plot*

p <- **ggplot**(df) + **geom\_segment**(**aes**(x=1, xend=2, y=`1952`, yend=`1957`, col=class), size=.75, show.legend=F) +

**geom\_vline**(xintercept=1, linetype="dashed", size=.1) +

**geom\_vline**(xintercept=2, linetype="dashed", size=.1) +

**scale\_color\_manual**(labels = **c**("Up", "Down"),

values = **c**("green"="#00ba38", "red"="#f8766d")) + *# color of lines*

**labs**(x="", y="Mean GdpPerCap") + *# Axis labels*

**xlim**(.5, 2.5) + **ylim**(0,(1.1\*(**max**(df$`1952`, df$`1957`)))) *# X and Y axis limits*

*# Add texts*

p <- p + **geom\_text**(label=left\_label, y=df$`1952`, x=**rep**(1, **NROW**(df)), hjust=1.1, size=3.5)

p <- p + **geom\_text**(label=right\_label, y=df$`1957`, x=**rep**(2, **NROW**(df)), hjust=-0.1, size=3.5)

p <- p + **geom\_text**(label="Time 1", x=1, y=1.1\*(**max**(df$`1952`, df$`1957`)), hjust=1.2, size=5) *# title*

p <- p + **geom\_text**(label="Time 2", x=2, y=1.1\*(**max**(df$`1952`, df$`1957`)), hjust=-0.1, size=5) *# title*

*# Minify theme*

p + **theme**(panel.background = **element\_blank**(),

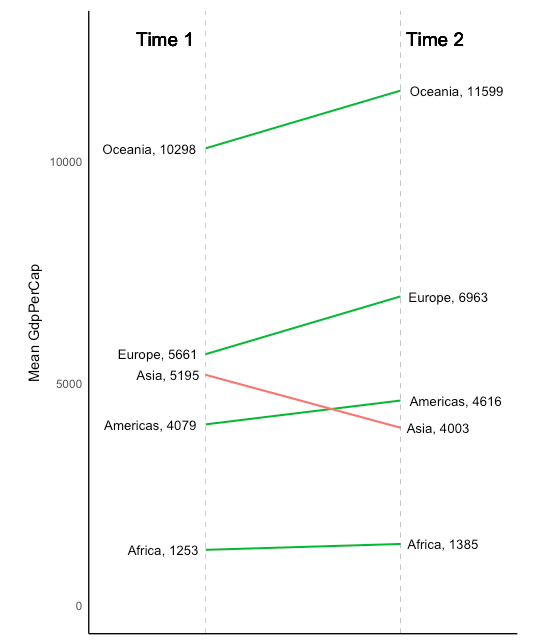
panel.grid = **element\_blank**(),

axis.ticks = **element\_blank**(),

axis.text.x = **element\_blank**(),

panel.border = **element\_blank**(),

plot.margin = **unit**(**c**(1,2,1,2), "cm"))



Dumbbell Plot

Dumbbell charts are a great tool if you wish to:

1. Visualize relative positions (like growth and decline) between two points in time.

2. Compare distance between two categories.

In order to get the correct ordering of the dumbbells, the Y variable should be a factor and the levels of the factor variable should be in the same order as it should appear in the plot.

**library**(ggplot2)

**library**(ggalt)

**theme\_set**(**theme\_classic**())

health <- **read.csv**("https://raw.githubusercontent.com/selva86/datasets/master/health.csv")

health$Area <- **factor**(health$Area, levels=**as.character**(health$Area)) *# for right ordering of the dumbells*

*# health$Area <- factor(health$Area)*

gg <- **ggplot**(health, **aes**(x=pct\_2013, xend=pct\_2014, y=Area, group=Area)) +

**geom\_dumbbell**(color="#a3c4dc",

size=0.75,

point.colour.l="#0e668b") +

**scale\_x\_continuous**(label=percent) +

**labs**(x=NULL,

y=NULL,

title="Dumbbell Chart",

subtitle="Pct Change: 2013 vs 2014",

caption="Source: https://github.com/hrbrmstr/ggalt") +

**theme**(plot.title = **element\_text**(hjust=0.5, face="bold"),

plot.background=**element\_rect**(fill="#f7f7f7"),

panel.background=**element\_rect**(fill="#f7f7f7"),

panel.grid.minor=**element\_blank**(),

panel.grid.major.y=**element\_blank**(),

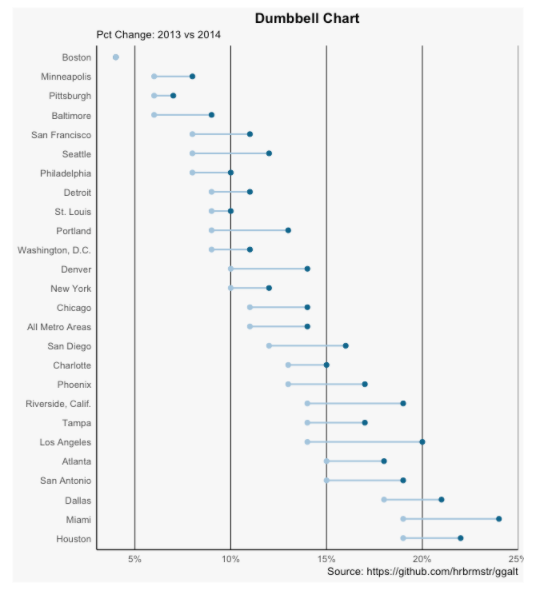
panel.grid.major.x=**element\_line**(),

axis.ticks=**element\_blank**(),

legend.position="top",

panel.border=**element\_blank**())

**plot**(gg)



## Time Series Analysis

> births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")

Read 168 items

> birthstimeseries <- ts(births, frequency=12, start=c(1946,1))

> birthstimeseries

> plot.ts(birthstimeseries)

