Composition

Pie Chart

Pie chart, a classic way of showing the compositions is equivalent to the waffle chart in terms of the information conveyed. But is a slightly tricky to implement in ggplot2 using the coord\_polar().

**library**(ggplot2)

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

*# Source: Frequency table*

df <- **as.data.frame**(**table**(mpg$class))

**colnames**(df) <- **c**("class", "freq")

pie <- **ggplot**(df, **aes**(x = "", y=freq, fill = **factor**(class))) +

**geom\_bar**(width = 1, stat = "identity") +

**theme**(axis.line = **element\_blank**(),

plot.title = **element\_text**(hjust=0.5)) +

**labs**(fill="class",

x=NULL,

y=NULL,

title="Pie Chart of class",

caption="Source: mpg")

pie + **coord\_polar**(theta = "y", start=0)

*# Source: Categorical variable.*

*# mpg$class*

pie <- **ggplot**(mpg, **aes**(x = "", fill = **factor**(class))) +

**geom\_bar**(width = 1) +

**theme**(axis.line = **element\_blank**(),

plot.title = **element\_text**(hjust=0.5)) +

**labs**(fill="class",

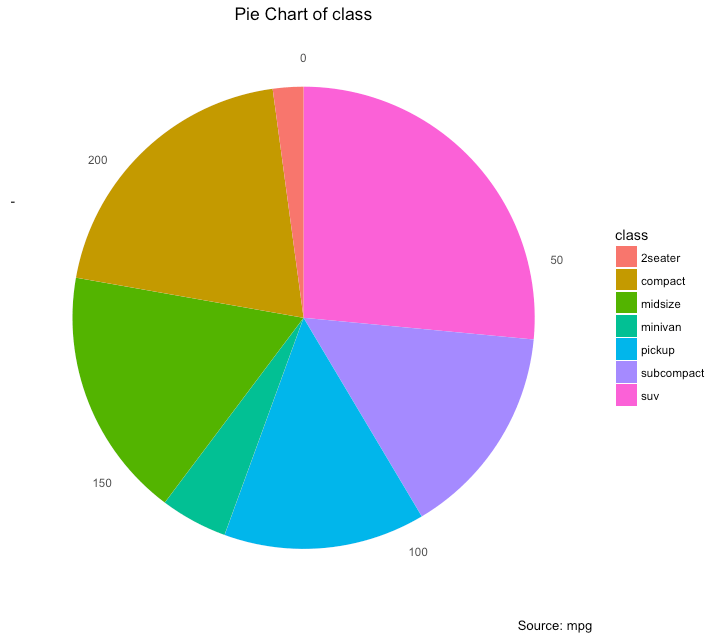
x=NULL,

y=NULL,

title="Pie Chart of class",

caption="Source: mpg")

pie + **coord\_polar**(theta = "y", start=0)



Treemap

Treemap is a nice way of displaying hierarchical data by using nested rectangles. The treemapifypackage provides the necessary functions to convert the data in desired format (treemapify) as well as draw the actual plot (ggplotify).

In order to create a treemap, the data must be converted to desired format using treemapify(). The important requirement is, your data must have one variable each that describes the area of the tiles, variable for fill color, variable that has the tile’s label and finally the parent group.

Once the data formatting is done, just call ggplotify() on the treemapified data.

**library**(ggplot2)

**library**(treemapify)

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

*# plot*

treeMapCoordinates <- **treemapify**(proglangs,

area = "value",

fill = "parent",

label = "id",

group = "parent")

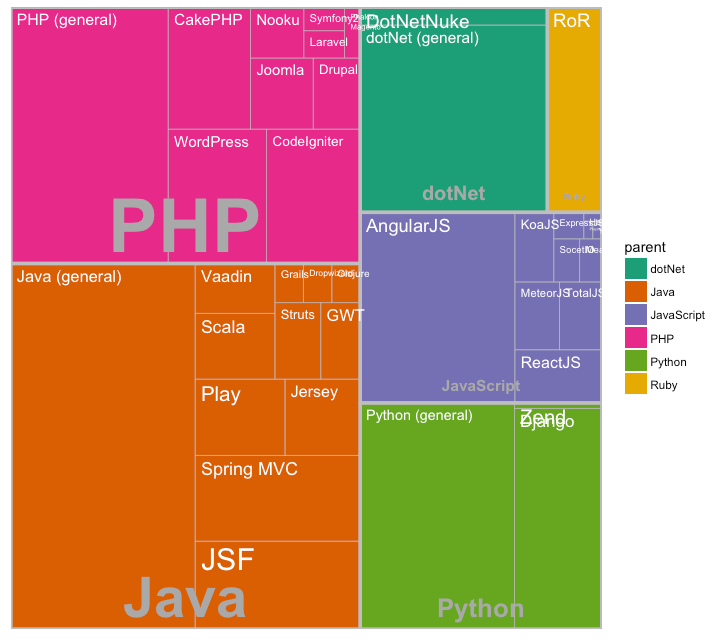
treeMapPlot <- **ggplotify**(treeMapCoordinates) +

**scale\_x\_continuous**(expand = **c**(0, 0)) +

**scale\_y\_continuous**(expand = **c**(0, 0)) +

**scale\_fill\_brewer**(palette = "Dark2")

**print**(treeMapPlot)



Bar Chart

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.

A bar chart can be drawn from a categorical column variable or from a separate frequency table. By adjusting width, you can adjust the thickness of the bars. If your data source is a frequency table, that is, if you don’t want ggplot to compute the counts, you need to set the stat=identity inside the geom\_bar().

*# prep frequency table*

freqtable <- **table**(mpg$manufacturer)

df <- **as.data.frame.table**(freqtable)

**head**(df)

*#> Var1 Freq*

*#> 1 audi 18*

*#> 2 chevrolet 19*

*#> 3 dodge 37*

*#> 4 ford 25*

*#> 5 honda 9*

*#> 6 hyundai 14*

*# plot*

**library**(ggplot2)

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

*# Plot*

g <- **ggplot**(df, **aes**(Var1, Freq))

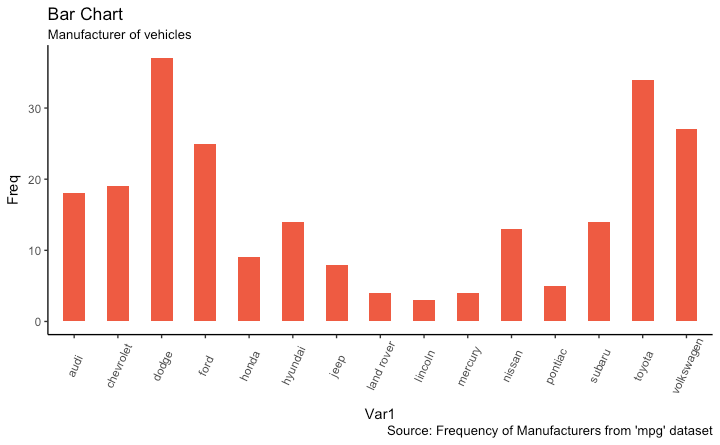
g + **geom\_bar**(stat="identity", width = 0.5, fill="tomato2") +

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

subtitle="Manufacturer of vehicles",

caption="Source: Frequency of Manufacturers from 'mpg' dataset") +

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



It can be computed directly from a column variable as well. In this case, only X is provided and stat=identity is not set.

*# From on a categorical column variable*

g <- **ggplot**(mpg, **aes**(manufacturer))

g + **geom\_bar**(**aes**(fill=class), width = 0.5) +

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

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

subtitle="Manufacturer of vehicles",

caption="Source: Manufacturers from 'mpg' dataset")

