R and Data Visualization Workshop Using ggplot2

march 15, 2018, McGill library Data Lab

Berenica Vejvoda, Numeric Data Librarian

2018

Contents

[What is ggplot2 2](#_Toc508623168)

[Learning Objectives 2](#_Toc508623169)

[Installing Packages 2](#_Toc508623170)

[Importing Packages 3](#_Toc508623171)

[Loading Data in Table Format 3](#_Toc508623172)

[Creating a Subset 3](#_Toc508623173)

[Removing Missing Values and Aggregating Data 3](#_Toc508623174)

[Creating a Bar Graph 4](#_Toc508623175)

[Flipping Coordinates 4](#_Toc508623176)

[Creating a Scatter Plot (Bubble Chart) and gganimate 5](#_Toc508623177)

[Creating a Scatter Plot with Jitter 5](#_Toc508623178)

[Adding a logarithmic scale to Y axis 6](#_Toc508623179)

[Creating Line Graphs 6](#_Toc508623180)

[Adding Themes – There is One for APA 8](#_Toc508623181)

[Saving and Printing Plots 8](#_Toc508623182)

[Plotly is Cool 8](#_Toc508623183)

[Further Resources 9](#_Toc508623184)

# What is ggplot2

Powerful graphics language for producing complex plots from data in a data frame (data frame=a two dimensional data structure in R for storing data tables)

Based on a Grammar of Graphics (as developed by Hadley Wickham) it allows one to concisely describe the components of a graphic.

Supports plotting both univariate and multivariate datasets (univariate omits y)

Currently cannot create 3D graphs

# Learning Objectives

* To understand the logic of ggplot building blocks and layering and the basic structure of a plot
* Show how to produce a bar graph, a scatter plot and a line graph, including animated plotting using gganimate and plotly
* Cover some of the data cleaning functions necessary in order to get raw data into the right shape and format for the visualizations desired

# Installing Packages

# install.packages("ggplot2", dependencies =TRUE)

# install.packages("devtools" ,dependencies = TRUE)

# install.packages("jtools", dependencies = TRUE)

# devtools::install\_github("dgrtwo/gganimate") (pkg::name returns the value of the exported variable name in namespace pkg)

To find out what functions mean try using ?”::”

# install ImageMagick from http://www.imagemagick.org/script/download.php

# while installing change application path to C: (IMPORTANT)

# select - install legacy tools

# install.packages("plotly",dependencies = TRUE)

# Plotly for R allows for interactive plotting

# turn off the scientific notation

options(scipen = 999)

# Importing Packages

library(ggplot2)

library(scales)

library(plotly)

library(gganimate)

# Loading Data in Table Format

# read in NHS dataset in table format and create a data frame from it

* df <- read.csv("C:\\RViz\\NHS.99M001X.E.2011.pumf.individuals\_F1.csv", header = TRUE)

# Creating a Subset

# create a subset choosing the variables TOTINC, GROSRT and PR

* NHS\_subset <- df[, c("PR","TOTINC","GROSRT")]

# Removing Missing Values and Aggregating Data

# remove missing values

# aggregate TOTINC and GROSRT by PR

This code will show how to aggregate multiple variables simultaneously as well as remove the missing values for TOTINC and GROSRT and calculate the mean for TOTINC and GROSRT

* NHS\_subset.agg <- aggregate(cbind(TOTINC,GROSRT)~PR, NHS\_subset[!(NHS\_subset$TOTINC==9999999 | NHS\_subset$GROSRT==9999 | NHS\_subset$GROSRT==8888),], mean)

# create a new variable called ProvinceName with the names of the provinces (so they can be used as labels in our plot)

* NHS\_subset.agg$ProvinceName <- c("Newfoundland & Labrador","Prince Edward Island","Nova Scotia","New Brunswick","Quebec","Ontario","Manitoba","Saskatchewn","Alberta","British Columbia","Northern Canada")

# Creating a Bar Graph

# plot a bar graph of PR versus TOTINC stacked with GROSRT

# add **ggplot()** function to initialize a ggplot object

# can invoke ggplot three ways: 1) ggplot(df, aes(x, y, )) 2) ggplot(df) 3) ggplot()

# this example uses ggplot() which initializes a skeleton ggplot object which is fleshed out when layers are added. This third method is almost always followed by + to add components (layers) to a plot

# add a **geom layer** (geoms are geometric objects which are the visual representation of observations. In other words they define the graph type.)

# **geom\_bar** is a bar graph representation. height of bars proportional to number of cases

# add aesthetic mappings (**aes function**) which describe how variables in the data are mapped to visual properties of geoms

# use **factor** when using discrete categories (vs continuous variables)

# **fill** aesthetic = fill colour of object

# **data** specifies a data frame

# **stat** function is your statistical transformation

# geom\_bar uses stat\_count by default so we will want to use **stat-=”identity”** to leave data as is since we want to plot the calculated means and not have ggplot count the numbers of cases at each x position (i.e. add up the mean values)

# **geom\_text** is a label argument that draws text labels

# label=paste0 – paste0 forces string

# **size** = how large objects appear (nsmall and size affect size of text

# x axis is discrete

# abbreviate built into label function (not necessarily ideal)

* ggplot() + geom\_bar(aes(y = TOTINC, x = factor(ProvinceName), fill = GROSRT), data = NHS\_subset.agg, stat = "identity") + geom\_text(data = NHS\_subset.agg, aes(x = factor(ProvinceName), y = TOTINC, label=paste0(format(round(GROSRT, 2), nsmall = 2))), size = 3.5) + scale\_x\_discrete(labels = abbreviate)

# Flipping Coordinates

# flipping coordinates (transpose x and y)

* ggplot() + geom\_bar(aes(y = TOTINC, x = factor(ProvinceName), fill = GROSRT), data = NHS\_subset.agg,stat = "identity", position = "stack") + geom\_text(data = NHS\_subset.agg,aes(x = factor(ProvinceName), y = TOTINC, label=paste0(format(round(GROSRT, 2), nsmall = 2))), size = 3.5) + scale\_x\_discrete(labels = abbreviate) + coord\_flip()

# Creating a Scatter Plot (Bubble Chart) and gganimate

# select PR and HDGREE

* NHS\_subset.degreePR <-df[!(df$HDGREE==88 | df$HDGREE==99), c("PR","HDGREE")]

# aggregate HDGREE by PR

* NHS\_subset.HdegreePRwise <- table(NHS\_subset.degreePR)

# select PR & HDGREE & TOTINC

* NHSsubset.PR\_HDGREE\_TOTINC <- (df[!(df$HDGREE==88 | df$HDGREE==99), c("PR","HDGREE","TOTINC")])

# aggregate TOTINC on PR and HDGREE to find out average total income for each HDGREE in each province

* AGG.PR\_HDGREE\_TOTINC <- aggregate(TOTINC ~ PR + HDGREE, NHSsubset.PR\_HDGREE\_TOTINC, mean)

# merge

# scale\_y\_log10() – converts y axis to a logarithmic scale (power of 10) to make TOTINC values display better (condense the numbers)

* NHS\_subset.HdegreePRTOTINC <- merge (NHS\_subset.HdegreePRwise, AGG.PR\_HDGREE\_TOTINC, all.y=TRUE)
* g <-ggplot(NHS\_subset.HdegreePRTOTINC, aes(x=HDGREE, y=TOTINC, size=Freq, frame=PR, color = HDGREE)) + geom\_point() + scale\_y\_log10() + labs(x="Higher Education Degree", y="Total Income", title ="Total Income by Higher Education Degree by Province)")

# interval = 1 shows frame for an interval of 1 second

* gganimate(g, interval = 1)

# Creating a Scatter Plot with Jitter

# scatter plot with jittering

# not what happens when you consume too much coffee

# jittering adds random noise to prevent over plotting

# often an issue with large samples and conveniently rounded values for continuous variables (e.g. cases=88000, many people have same income range)

# size of the circles is determined by the frequency count

* ggplot(NHS\_subset.HdegreePRTOTINC, aes(x=HDGREE, y=TOTINC)) + geom\_jitter(aes(col=PR, size=Freq)) + labs(x="Higher Education Degree", y="Total Income", title ="Province by Average Total Income by Higher Education Degree (Frequency wise)")

# Adding a logarithmic scale to Y axis

# log10() – converts y axis to logarithmic scale with base 10 (labels in powers of 10)

# [Further information on using logarithmic scales in charts and graphs](http://datadrivenjournalism.net/resources/when_should_i_use_logarithmic_scales_in_my_charts_and_graphs)

* ggplot(NHS\_subset.HdegreePRTOTINC, aes(x=HDGREE, y=TOTINC)) + geom\_jitter(aes(col=PR, size=Freq)) + labs(x="Higher Education Degree", y="Total Income", title ="Province Wise Total Income by Higher Education Degree (Frequency wise)") + scale\_y\_log10()

# Creating Line Graphs

# Lets use OECD open data and look at global GDP measures for Canada, the U.S. and Mexico

# read the ForecastGDP dataset

* df\_forecastgdp <- read.csv(file = "C:\\RViz\\RealGDP.csv", header = TRUE, strip.white = TRUE, stringsAsFactors = FALSE)

# subset the annual data for Canada, Mexico & USA

* df\_forecastgdp.subset <- subset(df\_forecastgdp, subset = ((LOCATION == "MEX" & FREQUENCY =="A") | (LOCATION == "CAN" & FREQUENCY =="A") | (LOCATION == "USA" & FREQUENCY =="A")), select = c(LOCATION, TIME, Value))

# replace the year with its abbvreviated form

# gsuv = string replacement function

df\_forecastgdp.subset$TIME <- gsub('^19',"'",df\_forecastgdp.subset$TIME)

df\_forecastgdp.subset$TIME <- gsub('^20',"'",df\_forecastgdp.subset$TIME)

df\_forecastgdp.subset$TIME <- factor(df\_forecastgdp.subset$TIME, levels = df\_forecastgdp.subset$TIME)

# plot forecasted real GDP by country

* ggplot(data = df\_forecastgdp.subset,aes(x = TIME, y=Value, group = LOCATION)) + geom\_line(aes(color = LOCATION)) + geom\_point(aes(color = LOCATION)) + theme(legend.position = "bottom")

# read in GDP dataset for Millions U.S. Dollars

* df\_gdp <- read.csv(file = "C:\\RViz\\GDP.csv",header = TRUE, strip.white = TRUE, stringsAsFactors = FALSE)

# subset the data for Canada, Mexico & U.S. and select only “Million US Dollars”

* df\_gdp.subset\_MLNUSD <- subset(df\_gdp, subset = ((ï..LOCATION == "MEX" & MEASURE =="MLN\_USD") | (ï..LOCATION == "CAN" & MEASURE =="MLN\_USD") | (ï..LOCATION == "USA" & MEASURE =="MLN\_USD")), select = c(ï..LOCATION, TIME, Value))
* colnames(df\_gdp.subset\_MLNUSD) <- c("LOCATION","TIME","Value")

# subset the data for Canada, Mexico & USA and select only the US Dollars/Capita

* df\_gdp.subset\_USDCAP <- subset(df\_gdp, subset = ((ï..LOCATION == "MEX" & MEASURE =="USD\_CAP") | (ï..LOCATION == "CAN" & MEASURE =="USD\_CAP") | (ï..LOCATION == "USA" & MEASURE =="USD\_CAP")), select = c(ï..LOCATION, TIME, Value))
* colnames(df\_gdp.subset\_USDCAP) <- c("LOCATION","TIME","Value")

# plot GDP by country in MLNUSD

* ggplot(data = df\_gdp.subset\_MLNUSD, aes(x = TIME, y=Value, group = LOCATION)) + geom\_line(aes(color = LOCATION)) + geom\_point(aes(color = LOCATION)) + theme(legend.position = "bottom")

# plot GDP by country in USDCAP

* ggplot(data = df\_gdp.subset\_USDCAP, aes(x = TIME, y=Value, group = LOCATION)) + geom\_line(aes(color = LOCATION)) + geom\_point(aes(color = LOCATION)) + theme(legend.position = "bottom")

# use facets to show 3 different plots for Canada, Mexico & USA, respectively

# transform the numbers in their abbreviated form (eg. 30000 as 30K)

* ggplot(data = df\_gdp.subset\_MLNUSD, aes(x = TIME, y=Value, group = LOCATION)) + geom\_line(aes(color = LOCATION)) + geom\_point(aes(color = LOCATION)) + theme(legend.position = "bottom") + facet\_wrap( ~ LOCATION, ncol =3) + scale\_y\_continuous(name = "Million US Dollars", labels = unit\_format(unit = 'M',scale = 1e-6)) + xlab("Years")

# Adding Themes – There is One for APA

# using APA theme to format the legend and labels to be consistent with the APA Style guide specifications for charts

* ggplot(data = df\_gdp.subset\_USDCAP,aes(x = TIME, y=Value, group = LOCATION)) + geom\_line(aes(color = LOCATION)) + geom\_point(aes(color = LOCATION)) + theme(legend.position = "bottom") + facet\_wrap( ~ LOCATION, ncol =3) + scale\_y\_continuous(name = "US Dollars/Capita", labels =unit\_format(unit = 'K',scale = 1e-3)) + xlab("Years") +jtools::theme\_apa(legend.pos = "bottom")

# change the shape of the points to triangle & make the line graph thicker

* Newplot <- ggplot(data = df\_gdp.subset\_USDCAP, aes(x = TIME, y=Value, group = LOCATION)) + geom\_line(aes(color = LOCATION),size =1.2) + geom\_point(aes(color = LOCATION),shape =2) + theme(legend.position = "bottom") + facet\_wrap( ~ LOCATION, ncol =3) + scale\_y\_continuous(name = "US Dollars/Capita",labels =unit\_format(unit = 'K',scale = 1e-3)) + xlab("Years") +jtools::theme\_apa(legend.pos = "bottom")

# Saving and Printing Plots

# prints the plot in the plot window

* print(Newplot)

# saves the plot in the desired format

* ggsave("C:\\myggplot.pdf", plot=Newplot)

# for an interactive ggplot we can use the package plotly.

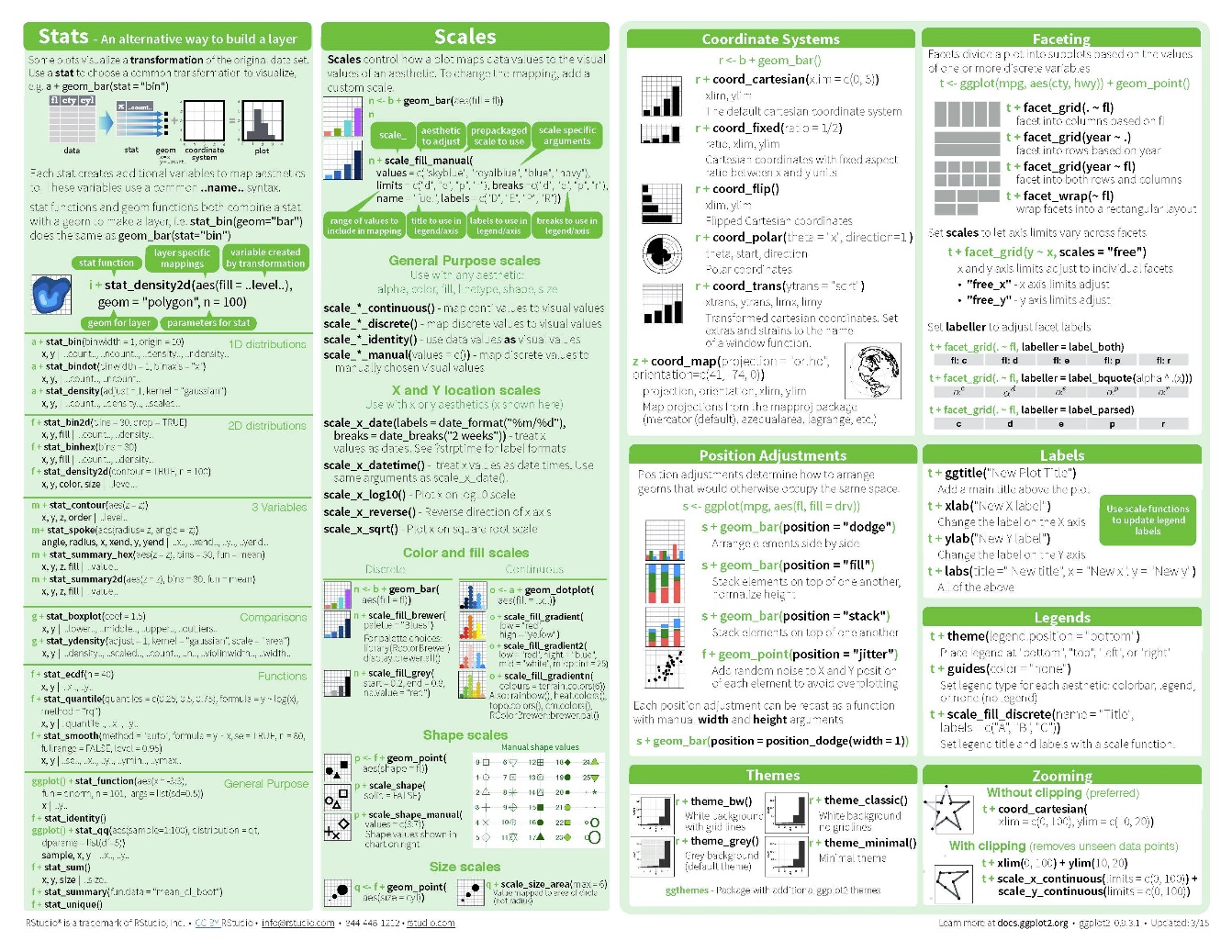
# Plotly is Cool

# we can zoom and see actual values on mouse-over and similar abilities using plotly

* ggplotly(newplot)

# Further Resources

RStudio cheat sheet for ggplot2 <https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>

[](https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf)

[QuickR DataCamp](https://www.statmethods.net/advgraphs/ggplot2.html)

[Data Visualization with ggplot2 (DataCamp)](https://www.datacamp.com/courses/data-visualization-with-ggplot2-1)

McGill Library eBooks by Hadley Wickham (ggplot2 creator)

[](http://mcgill.worldcat.org/oclc/656394958)[](http://mcgill.worldcat.org/oclc/905902760)