2018

R and Data Visualization Workshop Using ggplot2

MARCH 15, 2018, MCGILL LIBRARY DATA LAB

BERENICA VEJVODA, NUMERIC DATA LIBRARIAN

Contents

What is Data Visualization? 2
R 2
Why R? 2
What is ggplot2 2
Why ggplot2? 2
Learning Objectives 3
Installing Packages 3
Importing Packages 3
Loading Data in Table Format 4
Creating a Subset 4
Removing Missing Values and Aggregating Data 4
Creating a Bar Graph 4
Flipping Coordinates 6
Creating a Scatter Plot (Bubble Chart) and gganimate 6
Creating a Scatter Plot with Jitter 7
Adding a logarithmic scale to Y axis 8
Creating Line Graphs 8
Adding Themes – There is One for APA 10
Saving and Printing Plots 11
Plotly is Cool 11
Further Resources 12

What is Data Visualization?

Data visualization is viewed by many disciplines as a modern equivalent of visual communication. A primary goal of data visualization is to communicate information clearly and efficiently via statistical graphics, plots and information graphics. In other words, data visualization communicates data by encoding it as visual objects.

R

R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing. 24 years old now!

Why R?

- Free
- Works on all operating systems
- **Excellent and efficient graphical capacities**
- Large and active community
- Many methods not available in other commercial software
- Good integration between programming language and statistical functions
- Good integration with a number of database systems and other language

What is ggplot2

Powerful R graphics package 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

Why ggplot2?

Tidyverse GitHub Wiki Post

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

# normally, R prints "1e+05" rather than "100000" because the former takes only 5 characters, while the latter takes 6, and 5 < 6.

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

# calculate mean for total income and gross rent

# | pipe means OR
```

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",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) + labs(title = "Average Income & Average Rent by Province", x="Province", y = "Average Income")
```

Write to New Dataframe

```
gg <- 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) + labs(title = "Average Income & Average Rent by Province", x="Province", y = "Average Income")
```

Flipping Coordinates

# flipping coordinates (transpose x and y)

```
gg + coord_flip()
```

Creating a Scatter Plot (Bubble Chart) and gganimate

# select PR and HDGREE

# remove missing values

```
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

# remove missing values

```
NHSsubset.PR_HDGREE_TOTINC <- (df[!(df$HDGREE==88 | df$HDGREE==99), c("PR","HDGREE","TOTINC")])
```

# aggregate TOTINC on PR and HDGREE to get 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

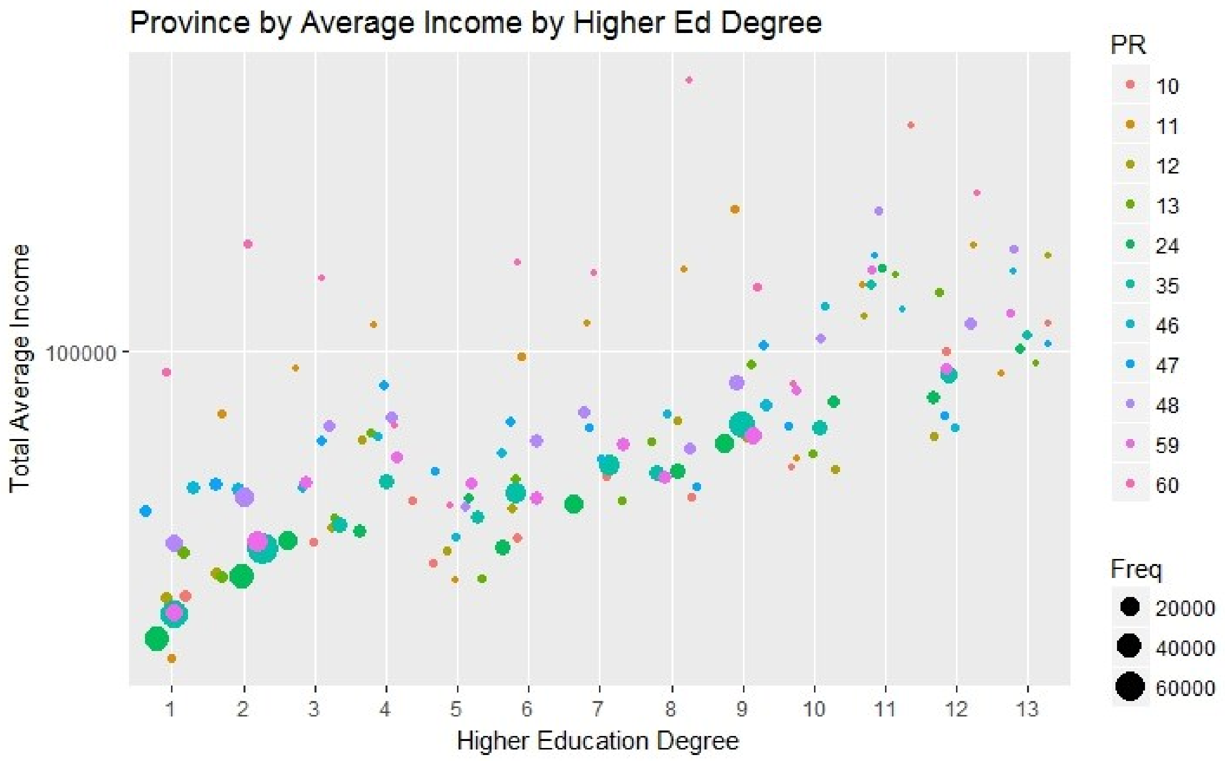
# not what happens when you consume too much coffee

# 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")

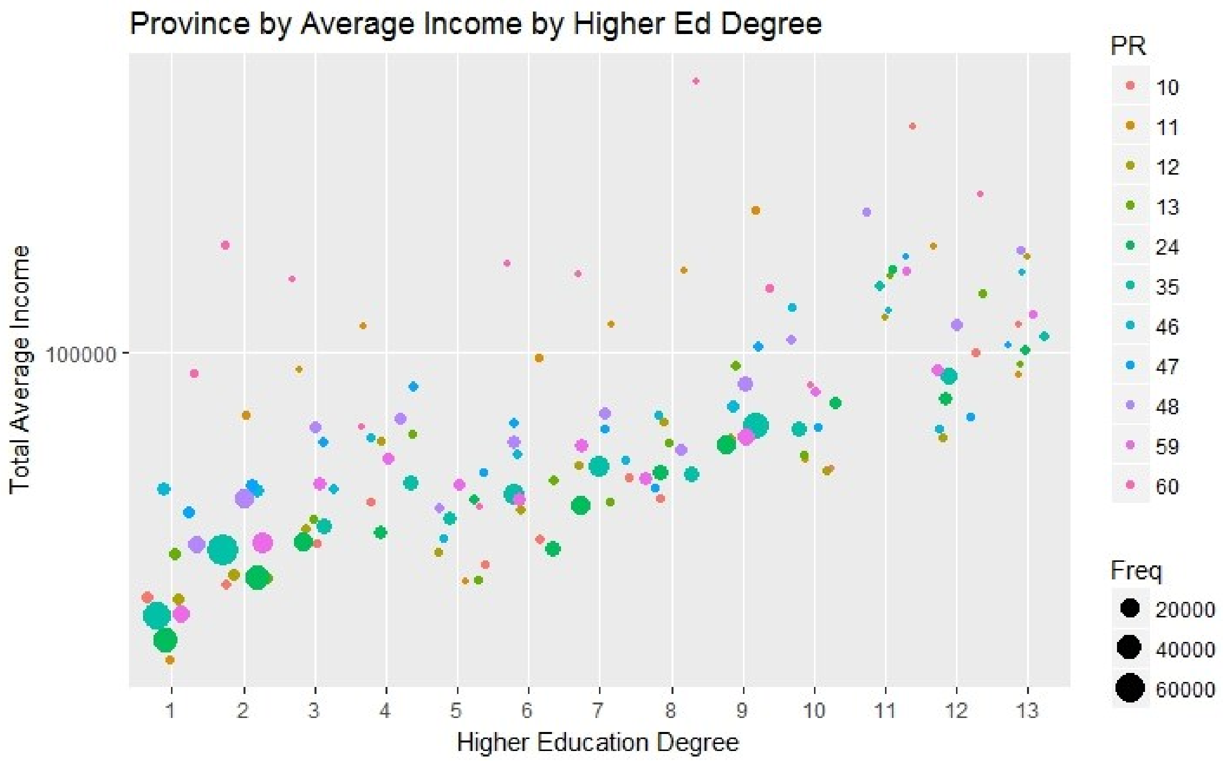


Adding a logarithmic scale to Y axis

# 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

```
ggplot(NHS_subset.HdegreePRTOTINC, aes(x=HDGREE, y=TOTINC)) + geom_jitter(aes(col=PR, size=Freq))  + labs(x="Higher Ed Degree", y="Total Average Income", title ="Province by Total Average Income by Higher Ed Degree") + scale_y_log10()
```



Creating Line Graphs

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

# read the Real 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))
```

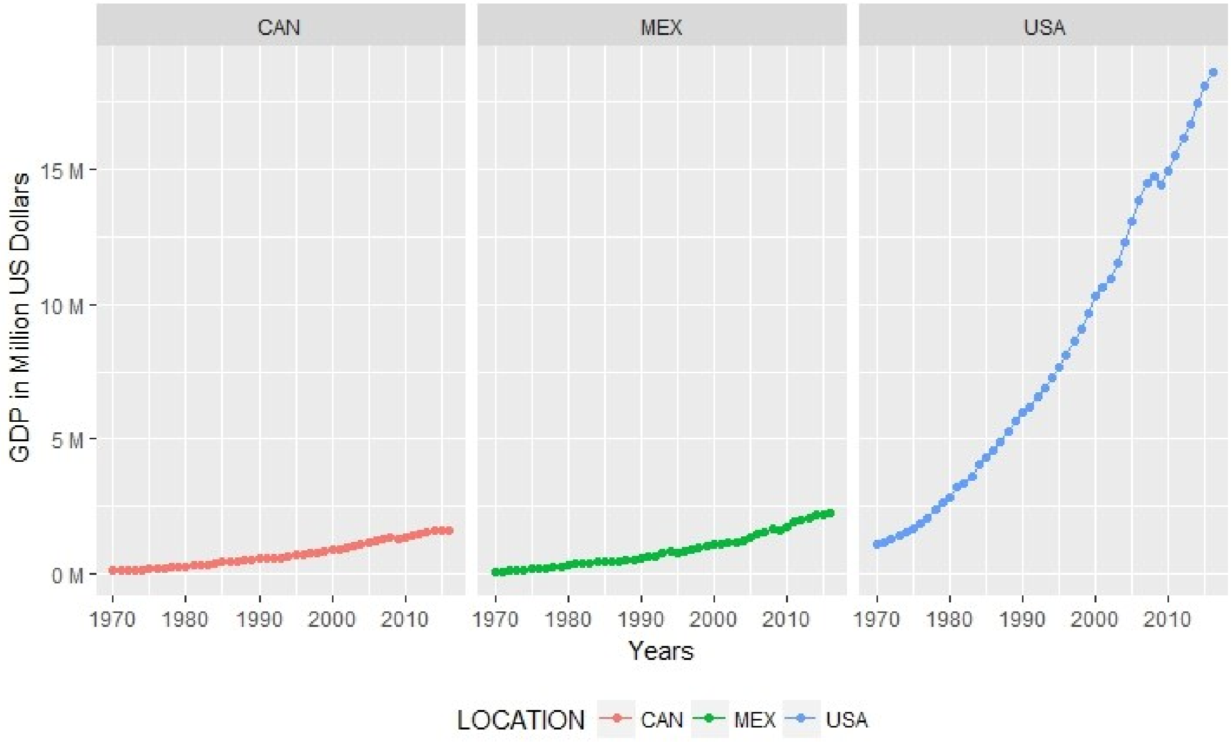
```
# replace the year with its abbvreviated form

# 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 “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") + labs(x="Year", y="GDP (in Million USD)", title ="GDP
      growth in Million US Dollars")

# 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") + labs(x="Year", y="GDP (USD per Capita)", title ="GDP growth in US Dollars per Capita")

# 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 = " GDP in Million US Dollars", labels = unit_format(unit = 'M',scale = 1e-6)) + xlab("Years")



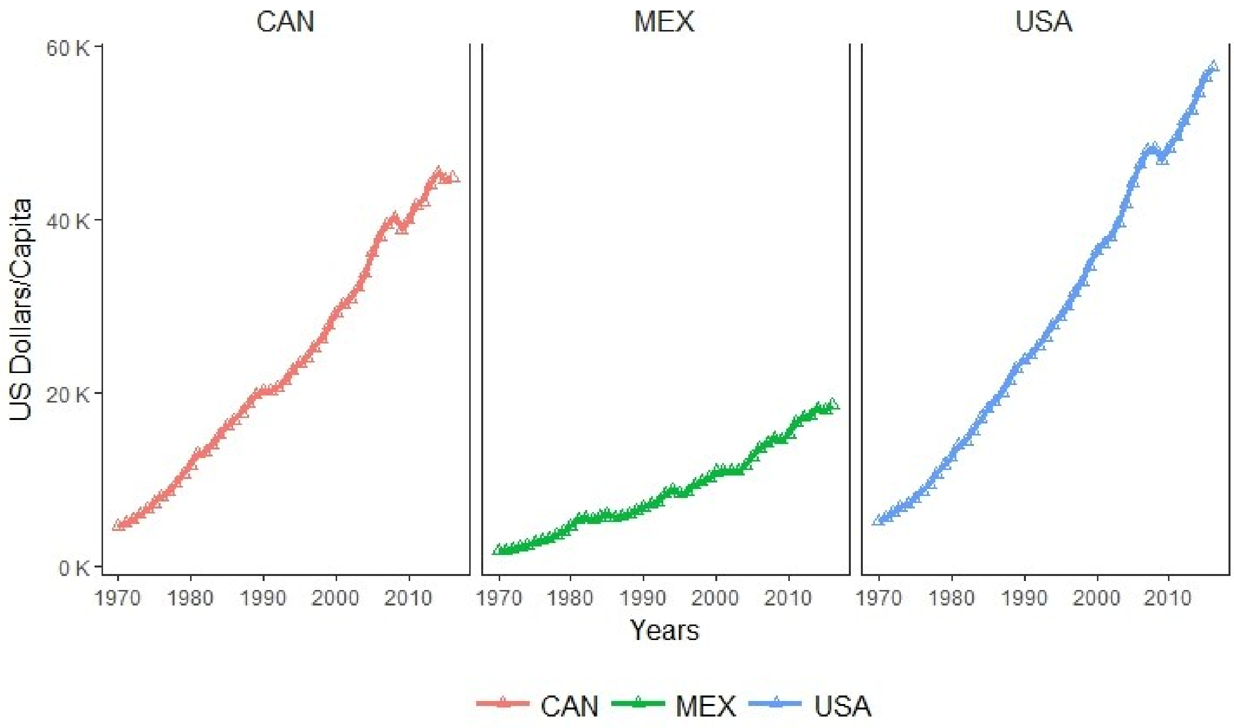
Adding Themes – There is One for APA

# 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 = " GDP in 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.

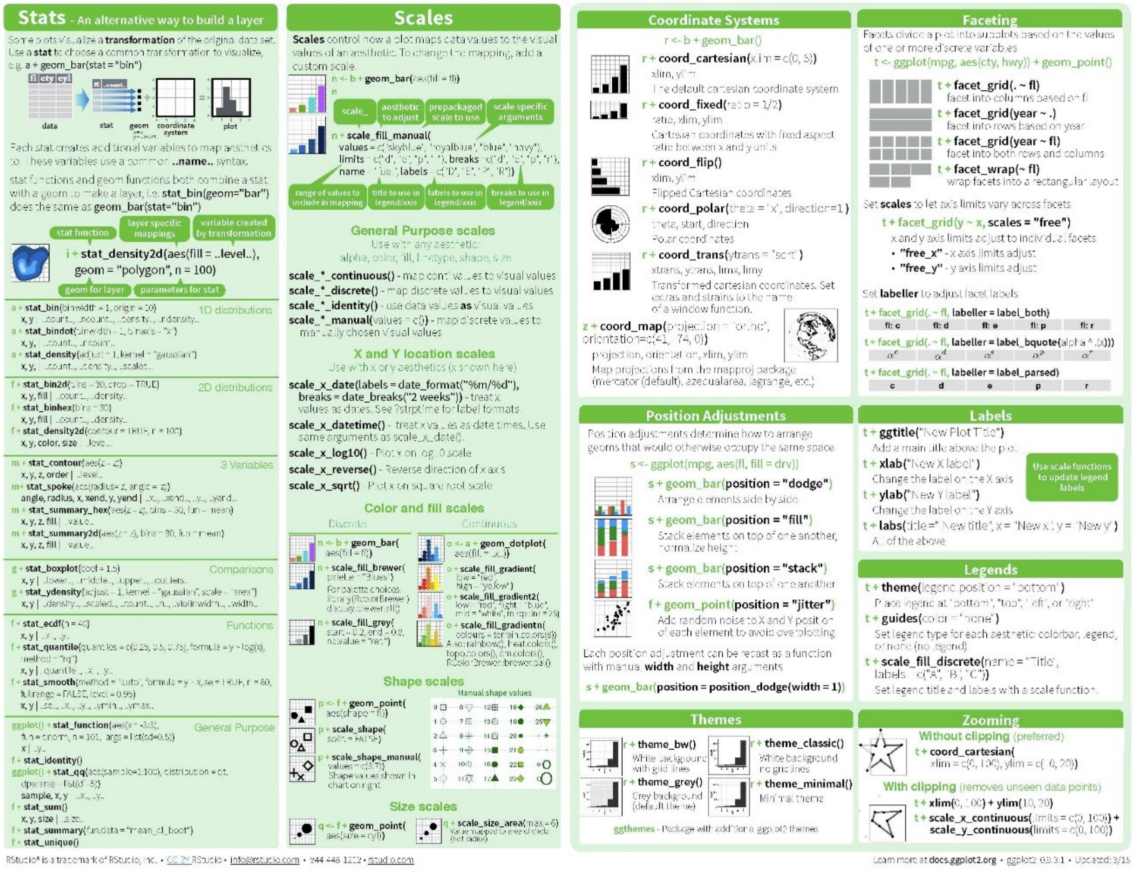
Plotly is Cool

# 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



Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. a + geom_bar(stat = "bin")

Each stat creates additional variables to map aesthetics to. These variables use a common **..name..** syntax.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. **stat_bin(geom="bar")** does the same as **geom_bar(stat="bin")**

stat function · layer specific mappings · variable created by transformation

i + **stat_density2d**(aes(fill = ..level..), geom = "polygon", n = 100)

geom for layer · parameters for stat

1D distributions

a + **stat_bin**(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + **stat_bindot**(binwidth = 1, binaxis = "x")
x, y, | ..count.., ..ncount..
a + **stat_density**(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

2D distributions

f + **stat_bin2d**(bins = 30, drop = TRUE)
x, y, fill | ..count.., ..density..
f + **stat_binhex**(bins = 30)
x, y, fill | ..count.., ..density..
f + **stat_density2d**(contour = TRUE, n = 100)
x, y, color, size | ..level..

3 Variables

m + **stat_contour**(aes(z = z))
x, y, z, order | ..level..
m+ **stat_spoke**(aes(radius= z, angle = z))
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + **stat_summary_hex**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..
m + **stat_summary2d**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

Comparisons

g + **stat_boxplot**(coef = 1.5)
x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + **stat_ydensity**(adjust = 1, kernel = "gaussian", scale = "area")
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Functions

f + **stat_ecdf**(n = 40)
x, y | ..x.., ..y..
f + **stat_quantile**(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")
x, y | ..quantile.., ..x.., ..y..
f + **stat_smooth**(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

General Purpose

ggplot() + **stat_function**(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5))
x | ..y..
f + **stat_identity**()
ggplot() + **stat_qq**(aes(sample=1:100), distribution = qt, dparams = list(df=5))
sample, x, y | ..x.., ..y..
f + **stat_sum**()
x, y, size | ..size..
f + **stat_summary**(fun.data = "mean_cl_boot")
f + **stat_unique**()

Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

n <- b + **geom_bar**(aes(fill = fl))
n

scale_ · aesthetic to adjust · prepackaged scale to use · scale specific arguments

n + **scale_fill_manual**(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in mapping · title to use in legend/axis · labels to use in legend/axis · breaks to use in legend/axis

General Purpose scales

Use with any aesthetic:
alpha, color, fill, linetype, shape, size

scale_*_continuous() - map cont' values to visual values
scale_*_discrete() - map discrete values to visual values
scale_*_identity() - use data values **as** visual values
scale_*_manual(values = c()) - map discrete values to manually chosen visual values

X and Y location scales

Use with x or y aesthetics (x shown here)

scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks")) - treat x values as dates. See ?strptime for label formats.
scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

Color and fill scales

Discrete

n <- b + **geom_bar**(aes(fill = fl))
n + **scale_fill_brewer**(palette = "Blues")
For palette choices: library(RColorBrewer) display.brewer.all()
n + **scale_fill_grey**(start = 0.2, end = 0.8, na.value = "red")

Continuous

o <- a + **geom_dotplot**(aes(fill = ..x..))
o + **scale_fill_gradient**(low = "red", high = "yellow")
o + **scale_fill_gradient2**(low = "red", hight = "blue", mid = "white", midpoint = 25)
o + **scale_fill_gradientn**(colours = terrain.colors(6))
Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

Shape scales

p <- f + **geom_point**(aes(shape = fl))
p + **scale_shape**(solid = FALSE)
p + **scale_shape_manual**(values = c(3:7))
Shape values shown in chart on right

Manual shape values

Size scales

q <- f + **geom_point**(aes(size = cyl))
q + **scale_size_area**(max = 6) Value mapped to area of circle (not radius)

Coordinate Systems

r <- b + geom_bar()

r + **coord_cartesian**(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system

r + **coord_fixed**(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + **coord_flip**()
xlim, ylim
Flipped Cartesian coordinates

r + **coord_polar**(theta = "x", direction=1)
theta, start, direction
Polar coordinates

r + **coord_trans**(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set extras and strains to the name of a window function.

z + **coord_map**(projection = "ortho", orientation=c(41, -74, 0))
projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + **facet_grid(. ~ fl)**
facet into columns based on fl
t + **facet_grid(year ~ .)**
facet into rows based on year
t + **facet_grid(year ~ fl)**
facet into both rows and columns
t + **facet_wrap(~ fl)**
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(y ~ x, **scales = "free"**)
x and y axis limits adjust to individual facets
- **"free_x"** - x axis limits adjust
- **"free_y"** - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(. ~ fl, **labeller = label_both**)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(. ~ fl, **labeller = label_bquote**(alpha ^ .(x)))

α^c	α^d	α^e	α^p	α^r

t + facet_grid(. ~ fl, **labeller = label_parsed**)

c	d	e	p	r

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(**position = "dodge"**)
Arrange elements side by side
s + geom_bar(**position = "fill"**)
Stack elements on top of one another, normalize height
s + geom_bar(**position = "stack"**)
Stack elements on top of one another
f + geom_point(**position = "jitter"**)
Add random noise to X and Y position of each element to avoid overplotting

Each position adjustment can be recast as a function with manual **width** and **height** arguments

s + geom_bar(**position = position_dodge(width = 1)**)

Labels

t + **ggtitle**("New Plot Title")
Add a main title above the plot
t + **xlab**("New X label")
Change the label on the X axis
t + **ylab**("New Y label")
Change the label on the Y axis
t + **labs**(title =" New title", x = "New x", y = "New y")
All of the above

Use scale functions to update legend labels

Legends

t + **theme**(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"
t + **guides**(color = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)
t + **scale_fill_discrete**(name = "Title", labels = c("A", "B", "C"))
Set legend title and labels with a scale function.

Themes

r + **theme_bw()**
White background with grid lines
r + **theme_classic()**
White background no gridlines
r + **theme_grey()**
Grey background (default theme)
r + **theme_minimal()**
Minimal theme

ggthemes - Package with additional ggplot2 themes

Zooming

Without clipping (preferred)

t + **coord_cartesian**(
xlim = c(0, 100), ylim = c(10, 20))

With clipping (removes unseen data points)

t + **xlim**(0, 100) + **ylim**(10, 20)
t + **scale_x_continuous**(limits = c(0, 100)) + **scale_y_continuous**(limits = c(0, 100))

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844 448 1212 • rstudio.com

Learn more at **docs.ggplot2.org** • ggplot2 0.9.3.1 • Updated: 3/15

QuickR DataCamp

Data Visualization with ggplot2 (DataCamp)

McGill Library eBooks by Hadley Wickham (ggplot2 creator)