Data Visualization with R and ggplot2

Jessica Minnier, PhD & Meike Niederhausen, PhD
OCTRI Biostatistics, Epidemiology, Research & Design (BERD) Workshop

2020/03/04 & 2020/05/20

slides: bit.ly/berd_ggplot
pdf: bit.ly/berd_ggplot_pdf

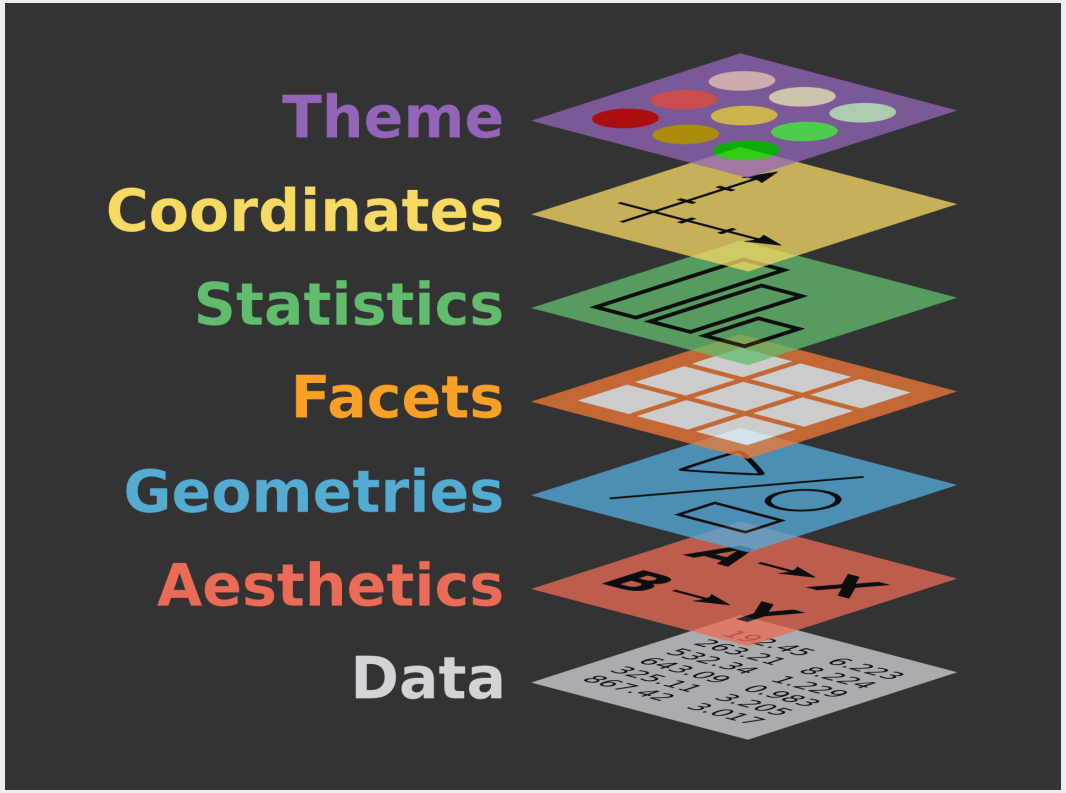
Load files for today's workshop

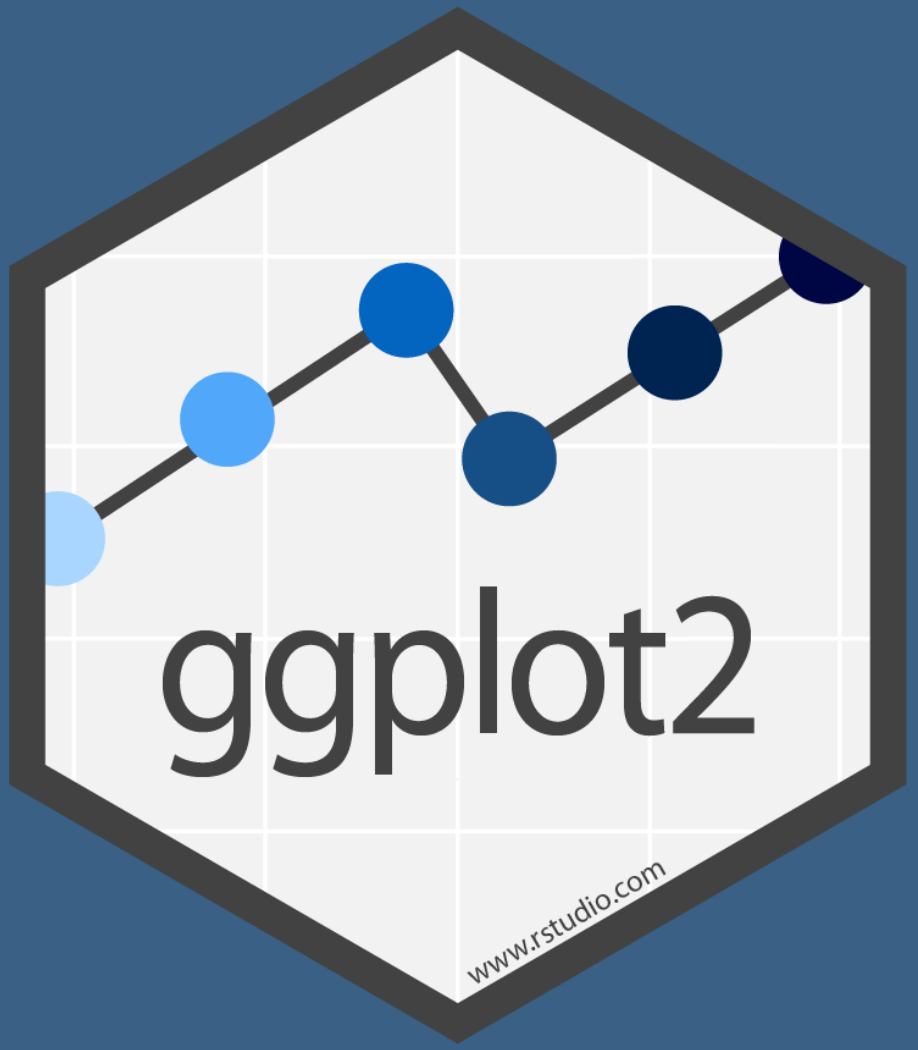
1. Open slides bit.ly/berd_ggplot
2. Get project folder (detailed instructions: bit.ly/berd_ggplot_instructions)
    - Download zip folder at bit.ly/berd_ggplot_zip
    - UNZIP completely (right click-> "extract all")
    - Open unzipped folder
    - Open (double click) `berd_ggplot_project.Rproj`
    - Inside RStudio 'Files' tab: click on file `00-install.R` and click "Run" to run all lines of code.
3. Open google doc for asking questions: https://bit.ly/berd_doc

Allison Horst

Learning objectives

- Understand the basic idea behind grammar of graphics
- Be able to data to visual elements
- Be able to customize plots in various ways
- Use ggplot extensions to make even more plots!




ggplot2
www.rstudio.com

Grammar of Graphics

- The "The Grammar of Graphics," is the theoretical basis for the ggplot2 package.
  - Much like how we construct sentences in any language by using a linguistic grammar (nouns, verbs, etc.), the grammar of graphics allows us to specify the components of a statistical graphic.

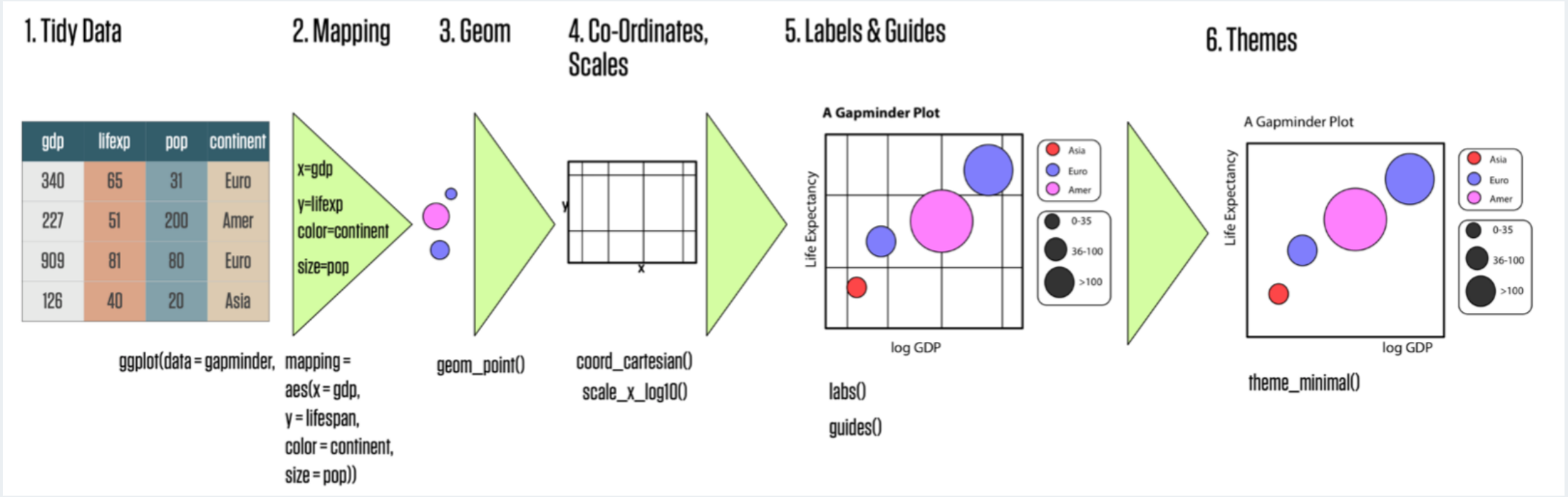
In short, the grammar tells us that:

> A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.

3 **essential** components to a graphic:

- data: the data-set comprised of variables that we plot
- geom: this refers to our type of geometric objects we see in our plot (points, lines, bars, etc.)
- aes: aesthetic attributes of the geometric object that we can perceive on a graphic. For example, x/y position, color, shape, and size. Each assigned aesthetic attribute can be mapped to a variable in our data-set.

Grammar of ggplot2



Kieran Healy

ggplot basics

Function

Dataset

```
ggplot(data = gapminder2011,
       aes(x = FoodSupplykcPPD, y = LifeExpectancyYrs,
           color = four_regions, size = population)) +
       geom_point()
```

Which variables to plot

What kind of plot to make

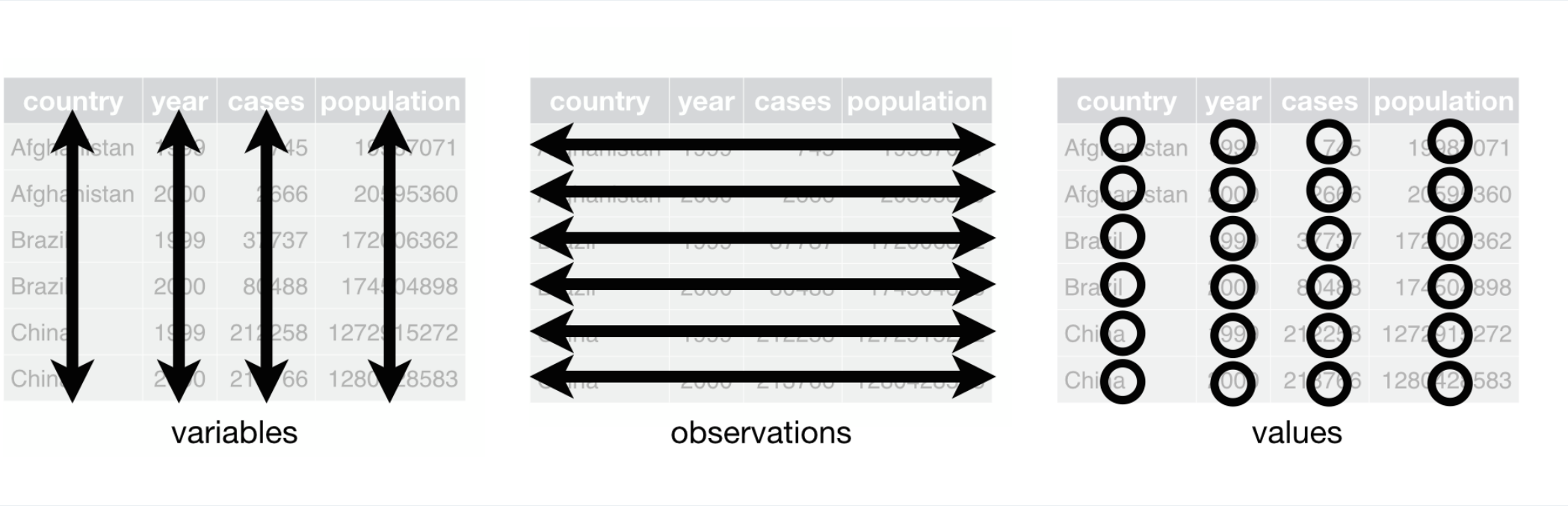
Tidy Data

Allison Horst

Ggplot needs tidy data

What are **tidy** data?

1. Each variable forms a column
2. Each observation forms a row
3. Each value has its own cell

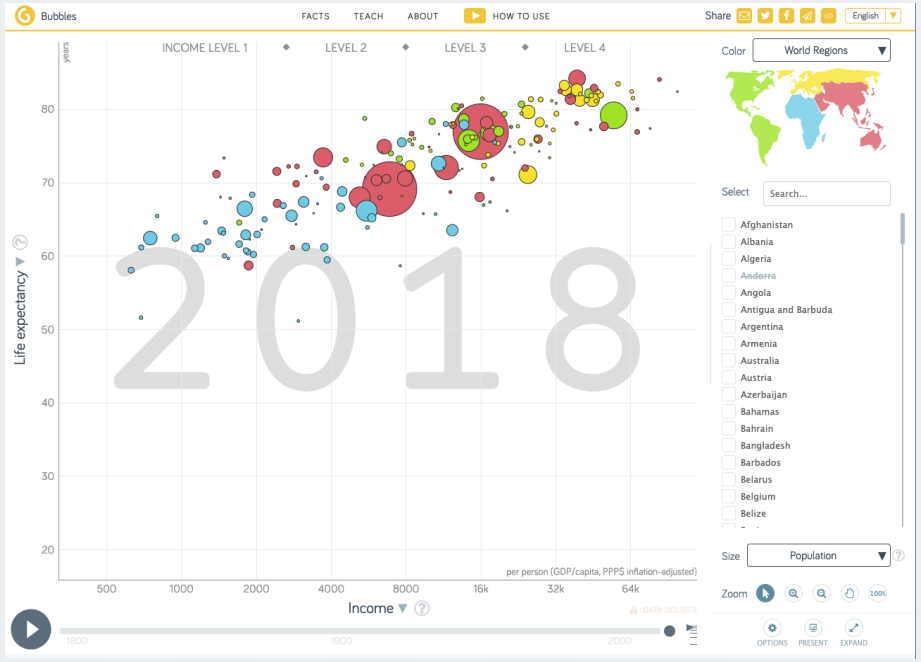


G. Grolemond & H. Wickham's R for Data Science

See BERD workshop Data Wrangling Part 1 slides for more info.

Gapminder data

Gapminder is a foundation "fact tank" that collects reliable global statistics on many different measures, such as average life expectancy, population size, food supply, water source, etc. for individual countries



Gapminder

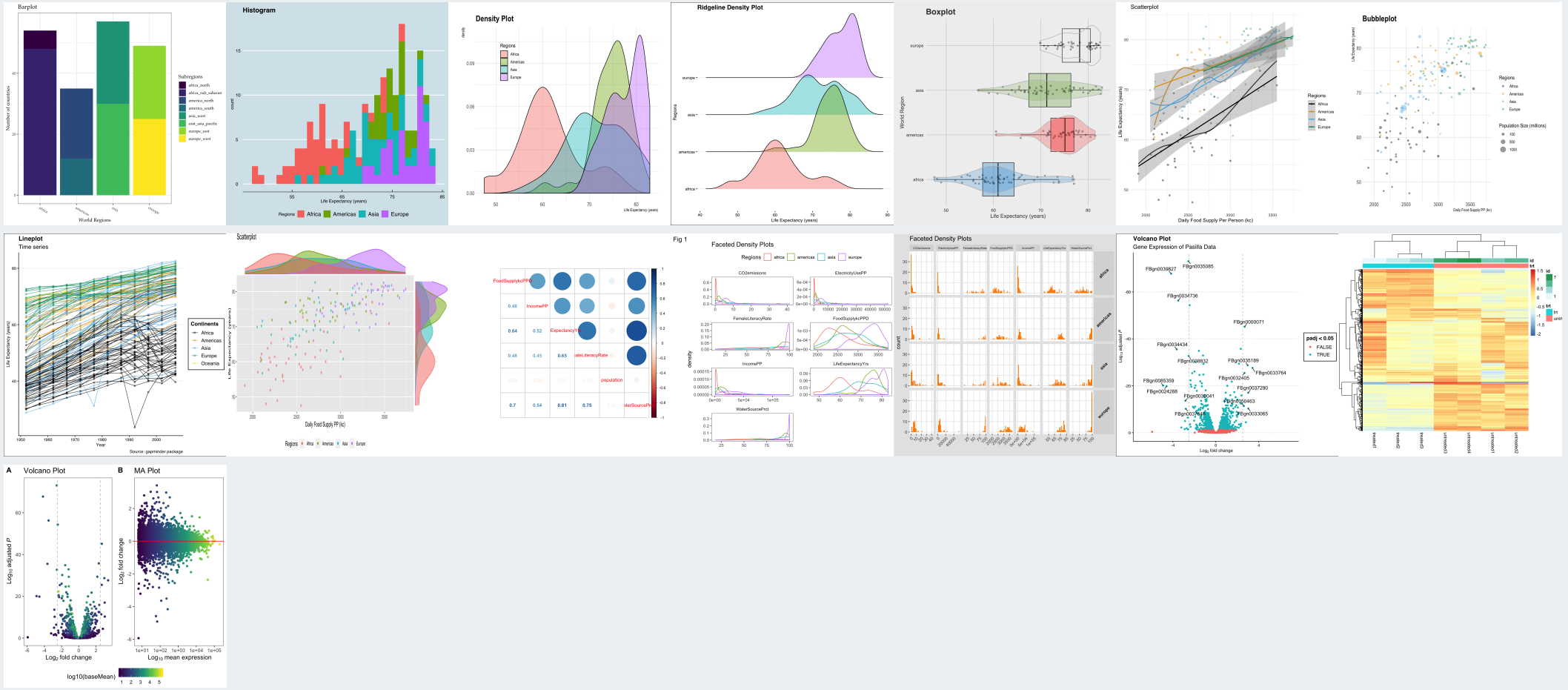
- `Gapminder_vars_2011.csv` contains select measures restricted to the year 2011.
- `Gapminder_vars_2011_long.csv` is the same data as in `Gapminder_vars_2011.csv`, but in a *long* format
  - Instead of individual columns for `CO2emissions`, `ElectricityUsePP`, … `WaterSourcePrct`,
    - there is a column called `Measures` which contains these variables names and
    - a column called `Values` with the actual values for these measures.
  - This means the dataset contains multiple rows per country to account for each of these measures.

A look at the Gapminder_vars_2011.csv dataset

```
gapminder2011 <- read_csv("data/Gapminder_vars_2011.csv")
glimpse(gapminder2011)
```

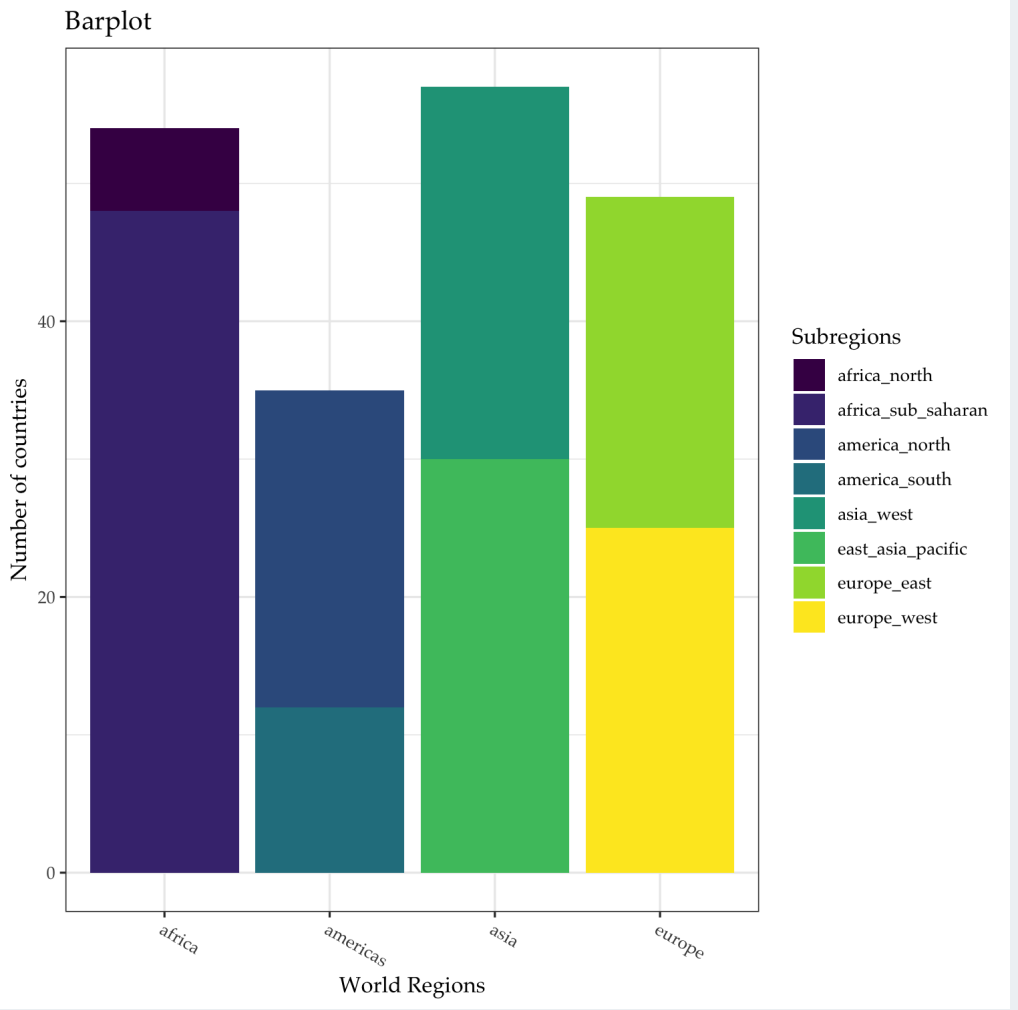
```
Observations: 195
Variables: 19
$ country                          <chr> "Afghanistan", "Albania", "Algeria", …
$ CO2emissions                     <dbl> 0.412, 1.790, 3.290, 5.870, 1.250, 5.…
$ ElectricityUsePP                 <dbl> NA, 2210.0, 1120.0, NA, 207.0, NA, 29…
$ FoodSupplykcPPD                  <dbl> 2110, 3130, 3220, NA, 2410, 2370, 316…
$ IncomePP                         <dbl> 1660, 10200, 13000, 42000, 5910, 1860…
$ LifeExpectancyYrs                <dbl> 56.7, 76.7, 76.7, 82.6, 60.9, 76.9, 7…
$ FemaleLiteracyRate               <dbl> 13.0, 95.7, NA, NA, 58.6, 99.4, 97.9,…
$ population                       <dbl> 2.97e+07, 2.93e+06, 3.68e+07, 8.38e+0…
$ WaterSourcePrct                  <dbl> 52.6, 88.1, 92.6, 100.0, 40.3, 97.0, …
$ WaterSourcePrct_2011_quart       <chr> "Q1", "Q2", "Q2", "Q4", "Q1", "Q3", "…
$ geo                              <chr> "afg", "alb", "dza", "and", "ago", "a…
$ four_regions                     <chr> "asia", "europe", "africa", "europe",…
$ eight_regions                    <chr> "asia_west", "europe_east", "africa_n…
$ six_regions                      <chr> "south_asia", "europe_central_asia", …
$ members_oecd_g77                 <chr> "g77", "others", "g77", "others", "g7…
$ latitude                         <dbl> 33.00000, 41.00000, 28.00000, 42.5077…
$ longitude                        <dbl> 66.00000, 20.00000, 3.00000, 1.52109,…
$ world_bank_region                <chr> "South Asia", "Europe & Central Asia"…
$ world_bank_4_income_groups_2017 <chr> "Low income", "Upper middle income", …
```

Visual Table of Contents

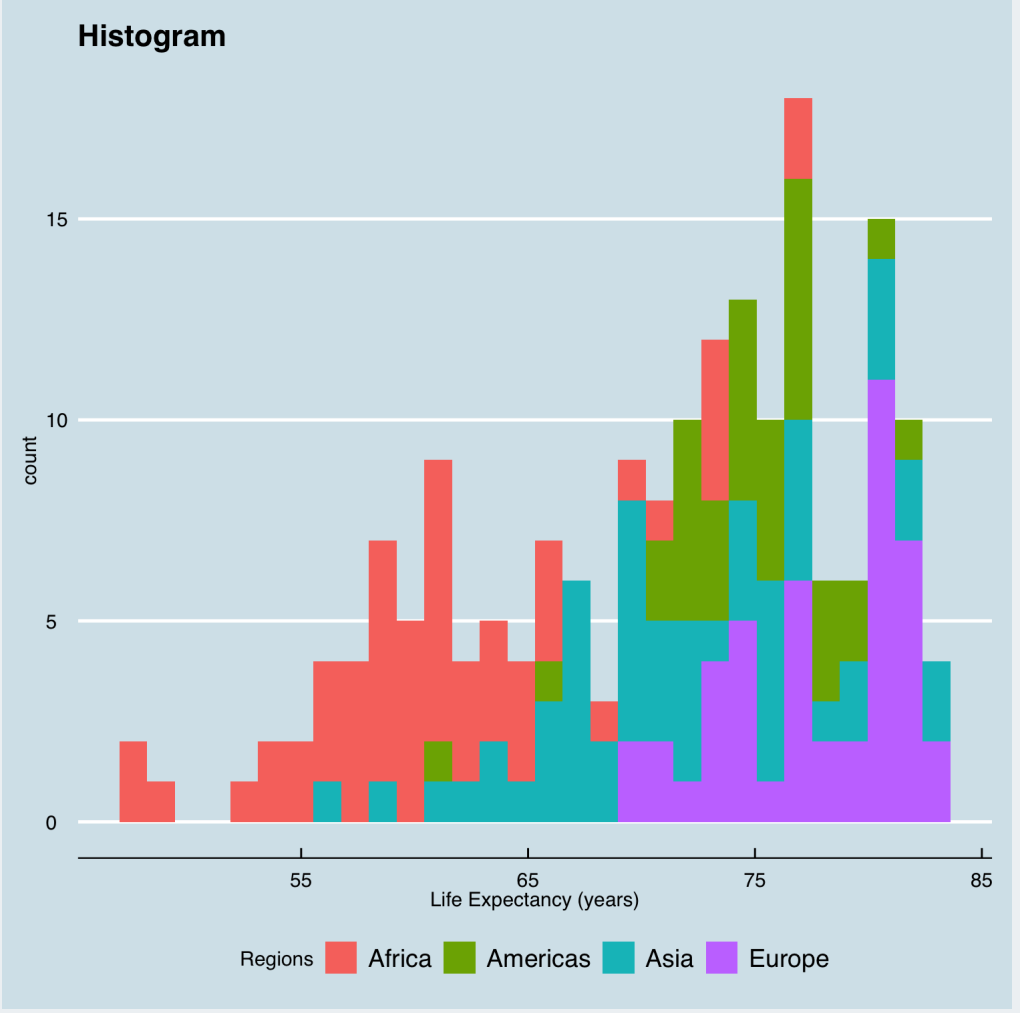


Inspired by EvaMaeRey (Gina Reynolds), author of the amazing `flipbookr` package.

```r
ggplot(data = gapminder2011) +
  aes(x = four_regions) +
  geom_bar() +
  aes(fill = eight_regions) +
  scale_fill_discrete(
    name = "Subregions"
    ) +
  labs(x = "World Regions",
       y = "Number of countries",
       title = "Barplot") +
  theme_bw() +
  theme(axis.text.x=element_text(
    angle = -30, hjust = 0)) +
  scale_fill_viridis_d(name = "Subregions")
  theme(
    text = element_text(family = "Palatino")
```



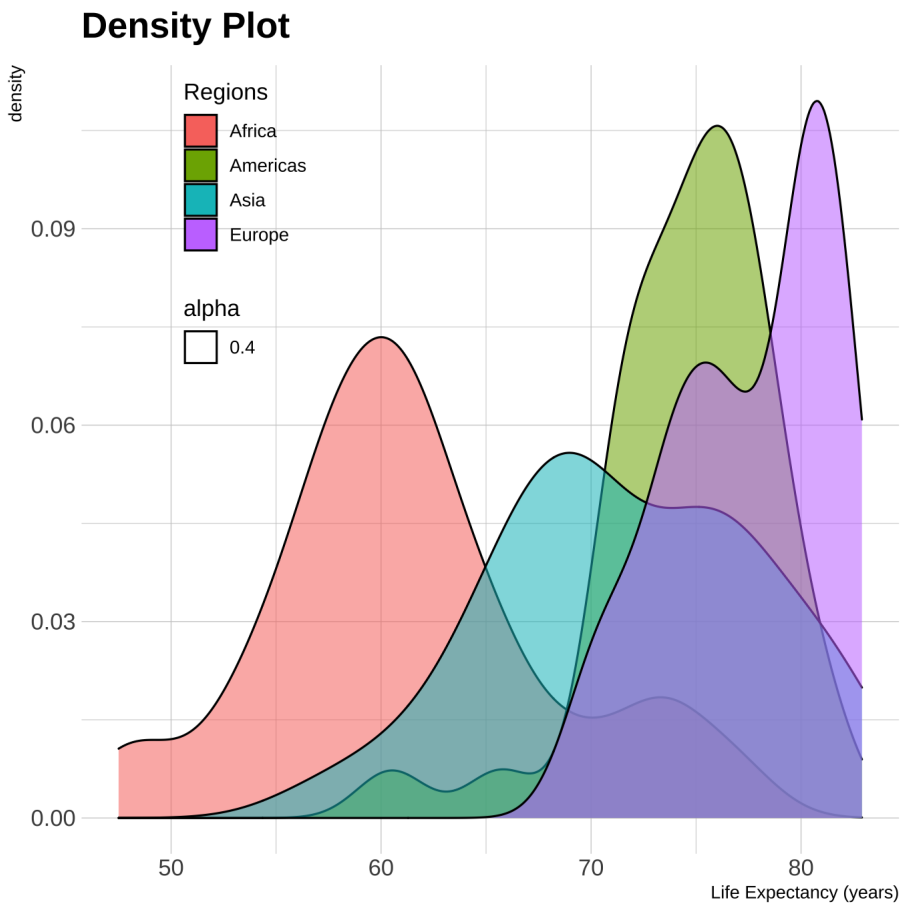
```r
ggplot(data = gapminder2011) +
  aes(x = LifeExpectancyYrs) +
  geom_histogram() +
  aes(fill = four_regions) +
  scale_fill_discrete(
    name = "Regions",
    labels = c("Africa", "Americas",
               "Asia", "Europe")
    ) +
  labs(
    x = "Life Expectancy (years)",
    title = "Histogram"
    ) +
  ggthemes::theme_economist() +
  theme(legend.position="bottom")
```



Legend position

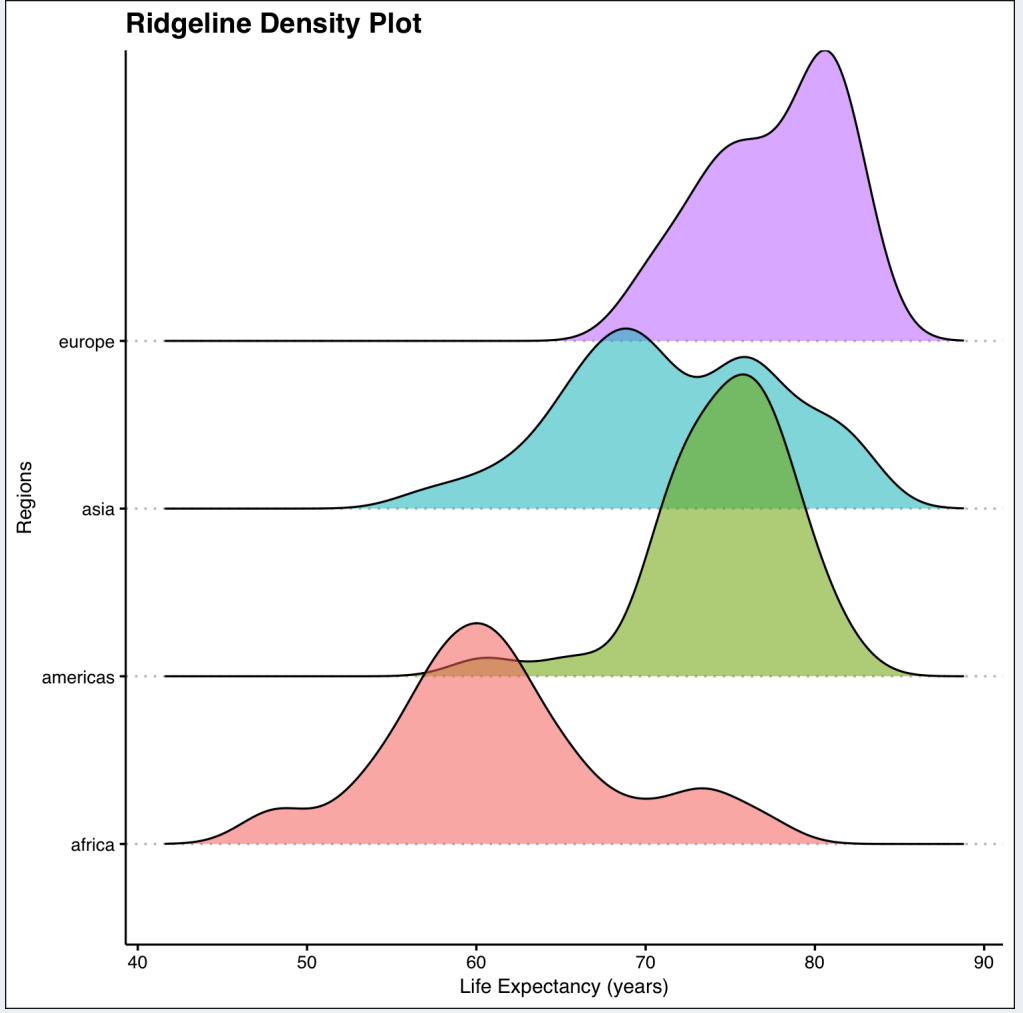
- "Generic" positions
  - `legend.position = "left"`
  - Other options: "top", "right", "bottom", "none"
- Specificied by location
  - `legend.position = c(x,y)`
  - Specify x and y coordinates of position
    - Values should be between 0 and 1
    - **c(0,0)** corresponds to the **bottom left**
    - **c(1,1)** corresponds to the **top right**

```r
ggplot(data = gapminder2011) +
  aes(x = LifeExpectancyYrs) +
  geom_density() +
  aes(fill = four_regions) +
  aes(alpha=.4) +
  scale_fill_discrete(
    name = "Regions",
    labels = c("Africa", "Americas",
               "Asia", "Europe")
    ) +
  hrbrthemes::theme_ipsum() +
  theme(legend.position=c(.2,.8)) +
  labs(
    x = "Life Expectancy (years)",
    title = "Density Plot"
    )
```

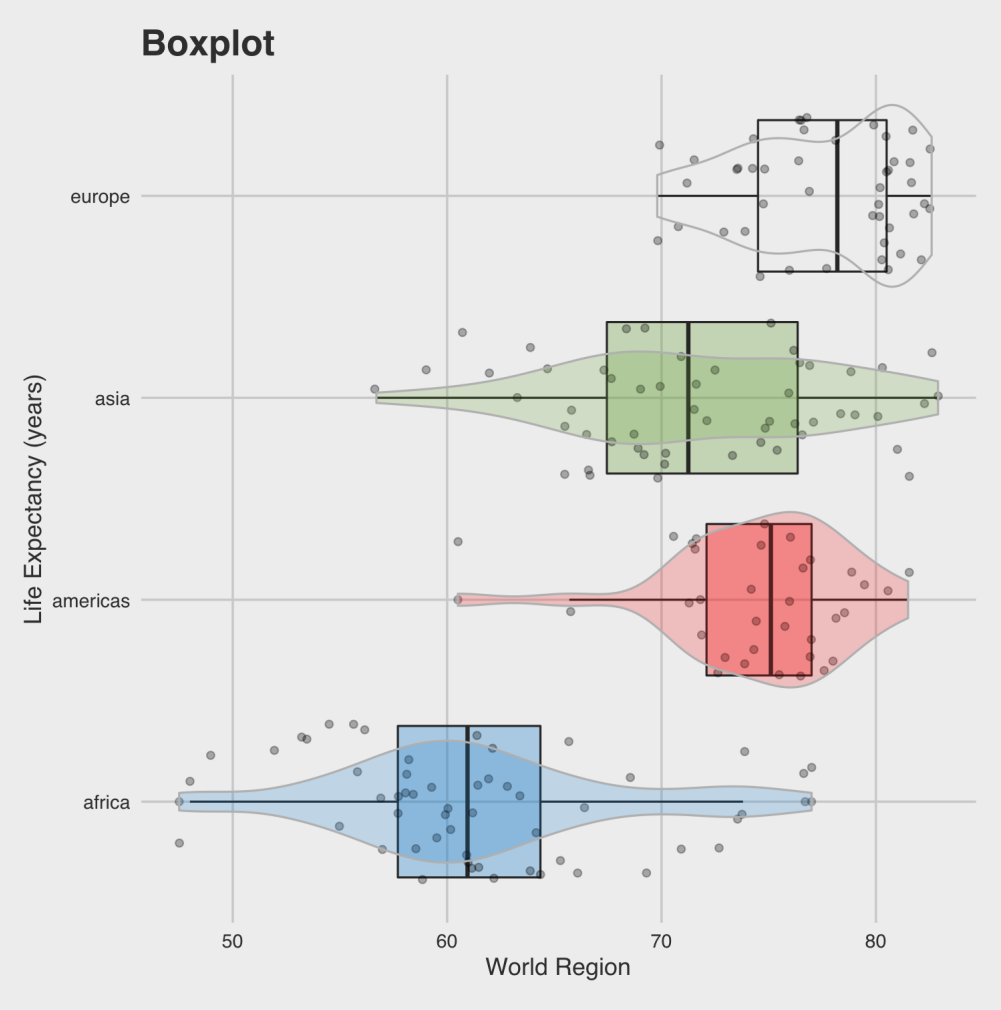


```r
library(ggridges)

ggplot(data = gapminder2011) +
  aes(x = LifeExpectancyYrs) +
  aes(y = four_regions) +
  geom_density_ridges() +
  aes(fill = four_regions) +
  aes(alpha = 0.4) +
  ggthemes::theme_clean() +
  theme(legend.position = "none") +
  labs(
    x = "Life Expectancy (years)",
    y = "Regions",
    title = "Ridgeline Density Plot"
    )
```



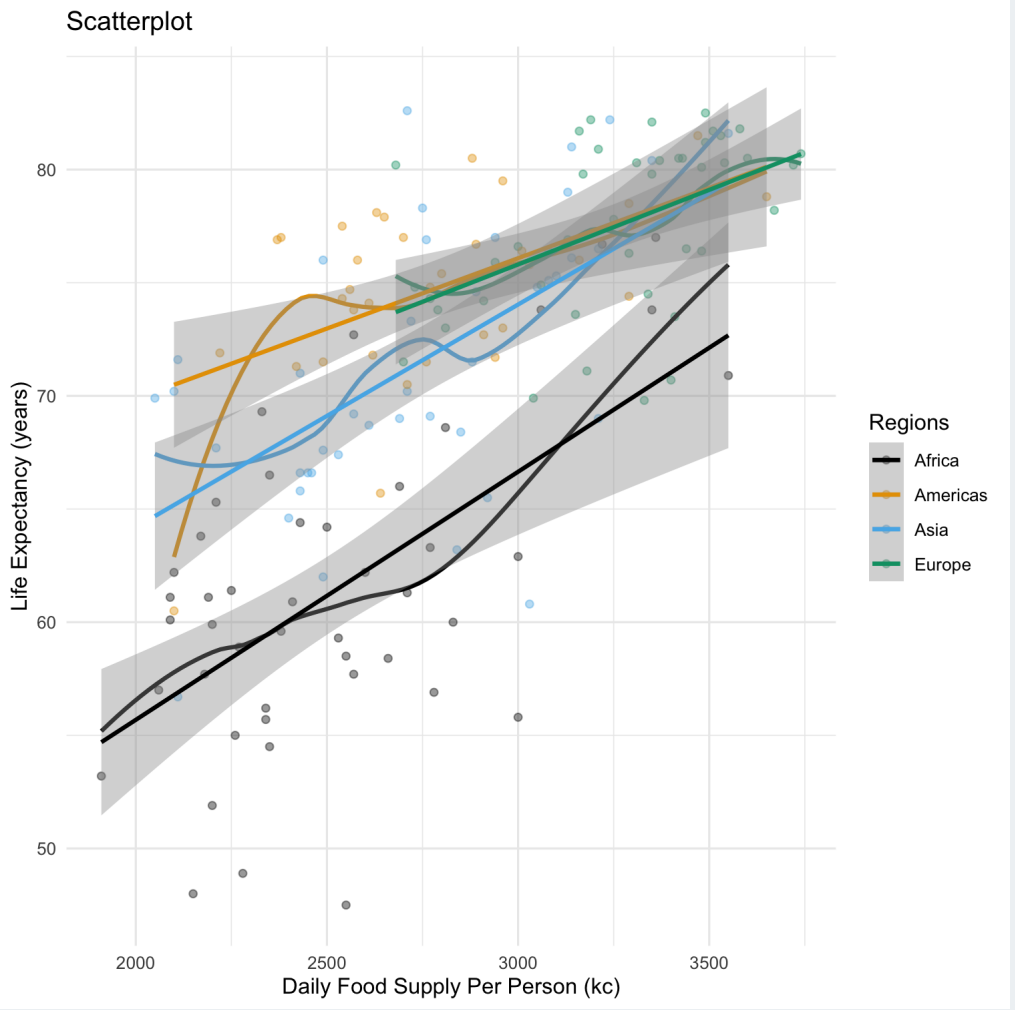
```
ggplot(data = gapminder2011) +
  aes(x = LifeExpectancyYrs) +  # New!
  geom_boxplot(alpha=.3) +
  aes(y = four_regions) +
  aes(fill = four_regions) +
  theme_fivethirtyeight() +
  scale_fill_fivethirtyeight() +
  theme(axis.title = element_text()) +
  theme(legend.position = "none") +
  geom_jitter(
    width = .1,
    alpha = 0.3
    ) +
  geom_violin(
    colour = "grey",
    alpha = .2
  ) +
  labs(
    x = "World Region",
    y = "Life Expectancy (years)",
    title = "Boxplot"
    )
```



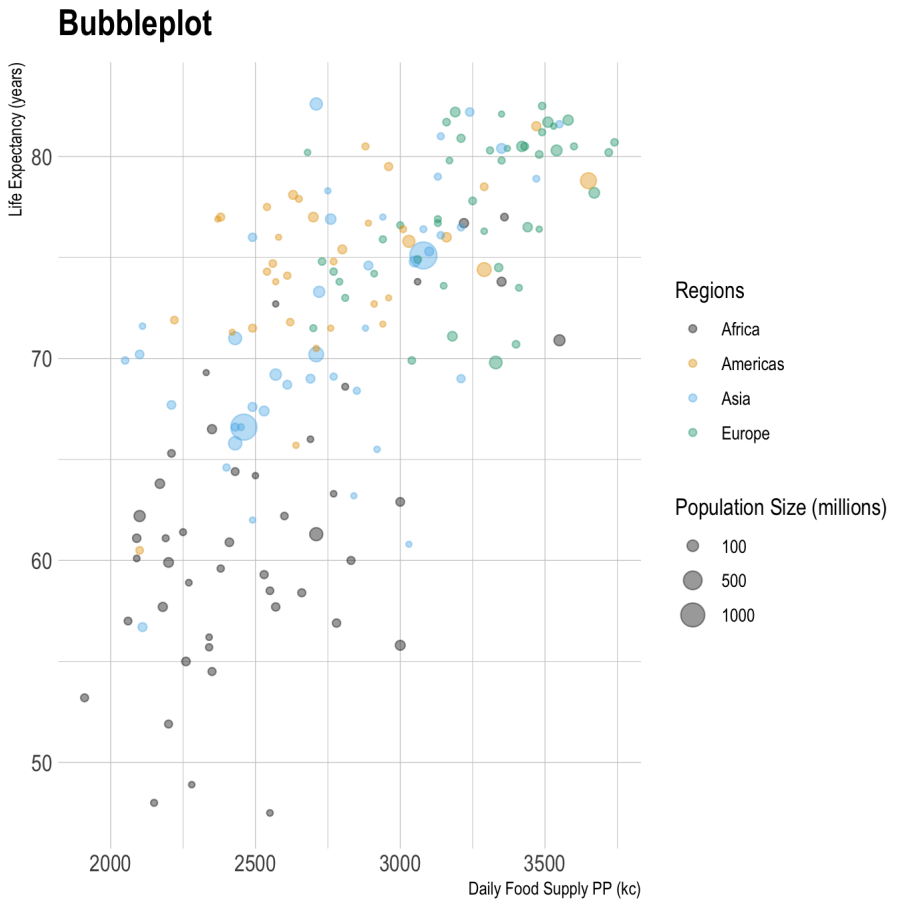
Exercise

Complete the third section of the `practice_ggplot.Rmd` file: "Bar plot".

```
ggplot(data = gapminder2011,
       aes(x = FoodSupplykcPPD,
           y = LifeExpectancyYrs)
       ) +
  geom_point(alpha=.4) +
  aes(color = four_regions) +
  scale_color_colorblind(
    name = "Regions",
    labels = c("Africa", "Americas",
               "Asia", "Europe")
    ) +
  geom_smooth(se = FALSE) +
  geom_smooth(method = lm) +
  theme_minimal() +
  labs(
    x = "Daily Food Supply Per Person (kc)",
    y = "Life Expectancy (years)",
    title = "Scatterplot"
    )
```



```r
ggplot(data = gapminder2011,
       aes(x = FoodSupplykcPPD,
           y = LifeExpectancyYrs,
           color = four_regions)
       ) +
  geom_point(alpha=.4) +
  aes(size = population) +
  scale_color_colorblind(
    name = "Regions",
    labels = c("Africa", "Americas",
               "Asia", "Europe")
  ) +
  scale_size(
    name = "Population Size (millions)",
    breaks = c(1e08,5e08,1e09),
    labels = c(100,500,1000)
  ) +
  hrbrthemes::theme_ipsum() +
  labs(
    x = "Daily Food Supply PP (kc)",
    y = "Life Expectancy (years)",
    title = "Bubbleplot"
    )
```



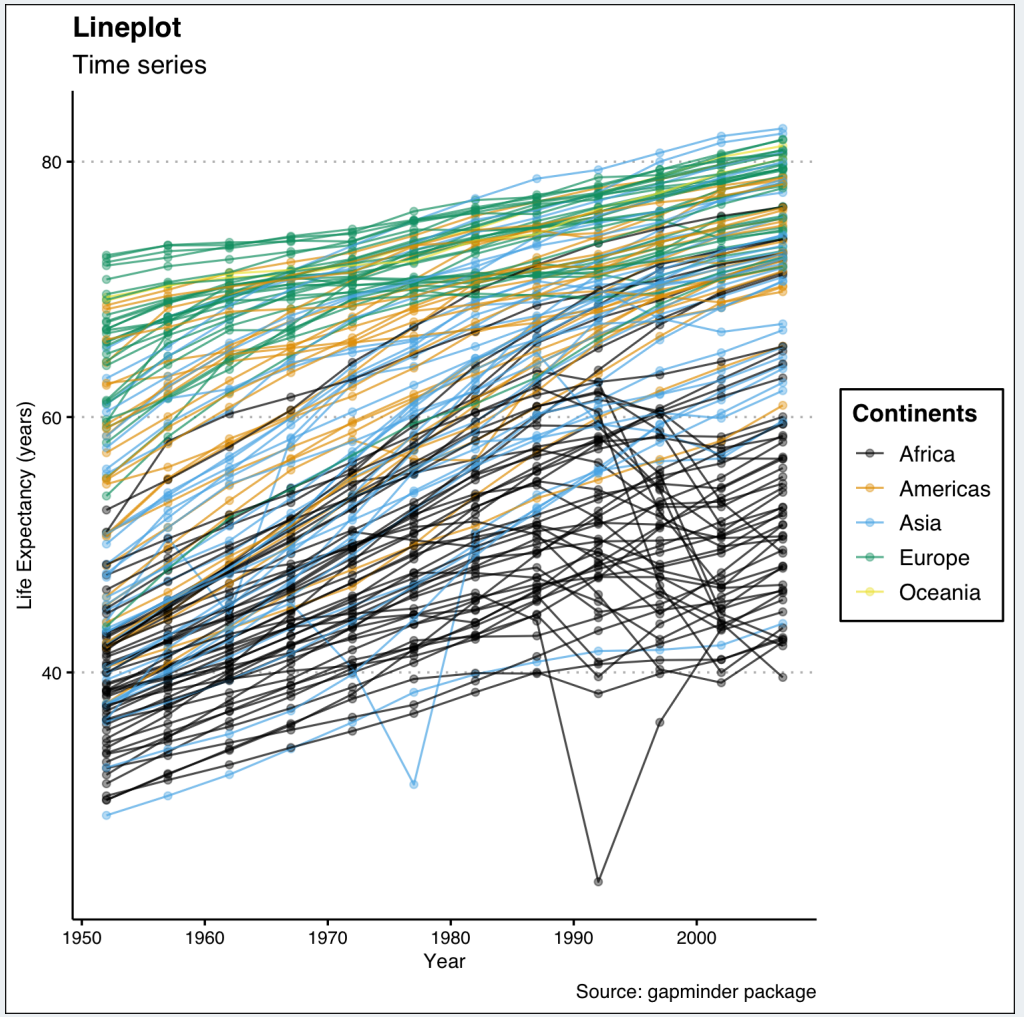
Exercise

Complete the fourth section of the `practice_ggplot.Rmd` file: "Bubbleplot"

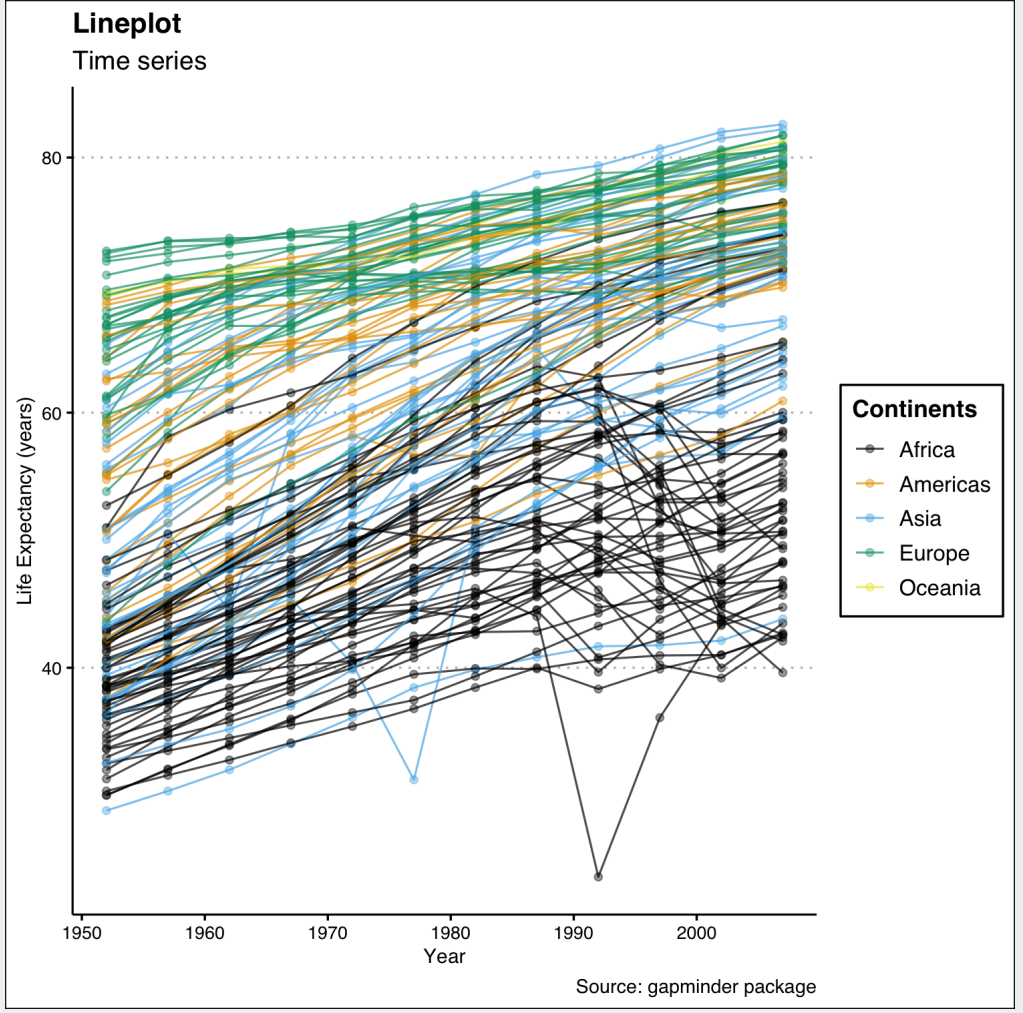
Lineplot

For the Lineplot example we are using the `gapminder::gapminder` dataset since it has longitudinal data across many years for life expectancy.

```
library(gapminder)
ggplot(data = gapminder,
       aes(x = year,
           y = lifeExp,
           color = continent,
           group = country)
       ) +
  geom_point(alpha = 0.4) +
  geom_line(alpha = 0.7) +
  scale_color_colorblind(name = "Continents") +
  ggthemes::theme_clean() +
  labs(
    x = "Year",
    y = "Life Expectancy (years)",
    title = "Lineplot",
    subtitle = "Time series",
    caption = "Source: gapminder package"
    )
```



```
ggplot(data = gapminder,
       aes(x = year,
           y = lifeExp,
           color = continent)
       ) +
  geom_point(alpha = .4) +
  geom_line(alpha = .7) +
  aes(group = country) +
  scale_color_colorblind(
    name = "Continents"
    ) +
  ggthemes::theme_clean() +
  labs(
    x = "Year",
    y = "Life Expectancy (years)",
    title = "Lineplot",
    subtitle = "Time series",
    caption = "Source: gapminder package"
    )
```



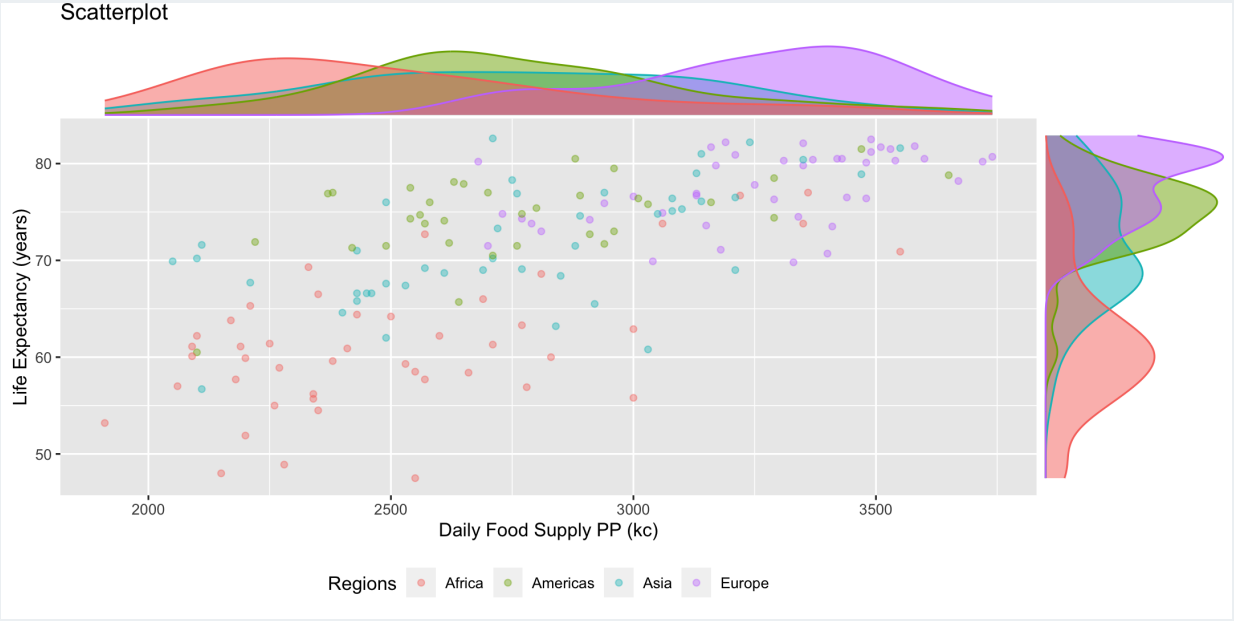
ggmarginal

https://cran.r-project.org/web/packages/ggExtra/vignettes/ggExtra.html

```
# library(ggExtra)

p <- ggplot(data = gapminder2011,
        aes(x = FoodSupplykcPPD,
            y = LifeExpectancyYrs,
            color = four_regions)
        ) +
  geom_point(alpha = .4) +
  scale_color_discrete(
    name = "Regions",
    labels = c("Africa", "Americas",
               "Asia", "Europe")
    ) +
  theme(legend.position="bottom") +
  labs(
    x = "Daily Food Supply PP (kc)",
    y = "Life Expectancy (years)",
    title = "Scatterplot"
    )
```

```
ggMarginal(p,
  type = "density",
  margins = "both",
  groupColour = TRUE,
  groupFill = TRUE
)
```



Corrolelograms

Correlation matrix

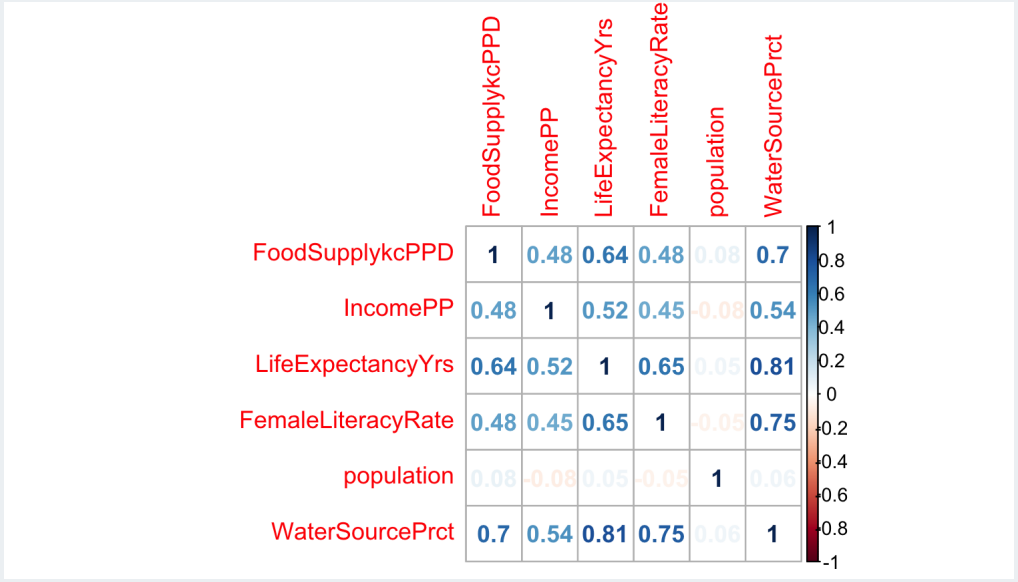
```
M <- cor(gapminder2011 %>%
           select(FoodSupplykcPPD:WaterSourcePrct),
         use = "complete.obs" # specified since there are missing values
         )
M
```

```
                  FoodSupplykcPPD    IncomePP LifeExpectancyYrs
FoodSupplykcPPD         1.0000000  0.48221951        0.64233437
IncomePP                0.4822195  1.00000000        0.51567562
LifeExpectancyYrs       0.6423344  0.51567562        1.00000000
FemaleLiteracyRate      0.4816309  0.44804036        0.64921874
population              0.0768498 -0.07838737        0.05467681
WaterSourcePrct         0.6980454  0.53687914        0.80693858
                  FemaleLiteracyRate  population WaterSourcePrct
FoodSupplykcPPD           0.48163092  0.07684980      0.69804539
IncomePP                  0.44804036 -0.07838737      0.53687914
LifeExpectancyYrs         0.64921874  0.05467681      0.80693858
FemaleLiteracyRate        1.00000000 -0.05188109      0.74980282
population               -0.05188109  1.00000000      0.05559188
WaterSourcePrct           0.74980282  0.05559188      1.00000000
```

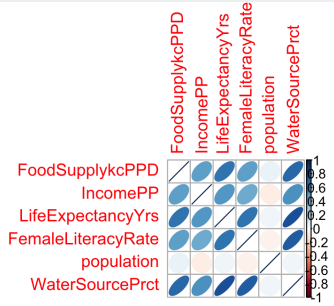
corrplot::corrplot()

https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html

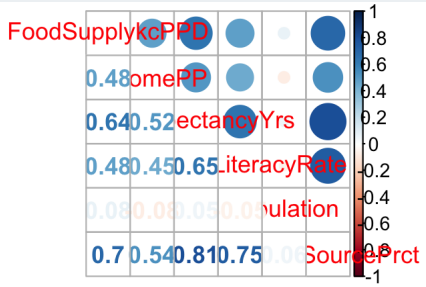
```
library(corrplot)
corrplot(M, method = "number")
```




```
corrplot(M, method = "ellipse")
```



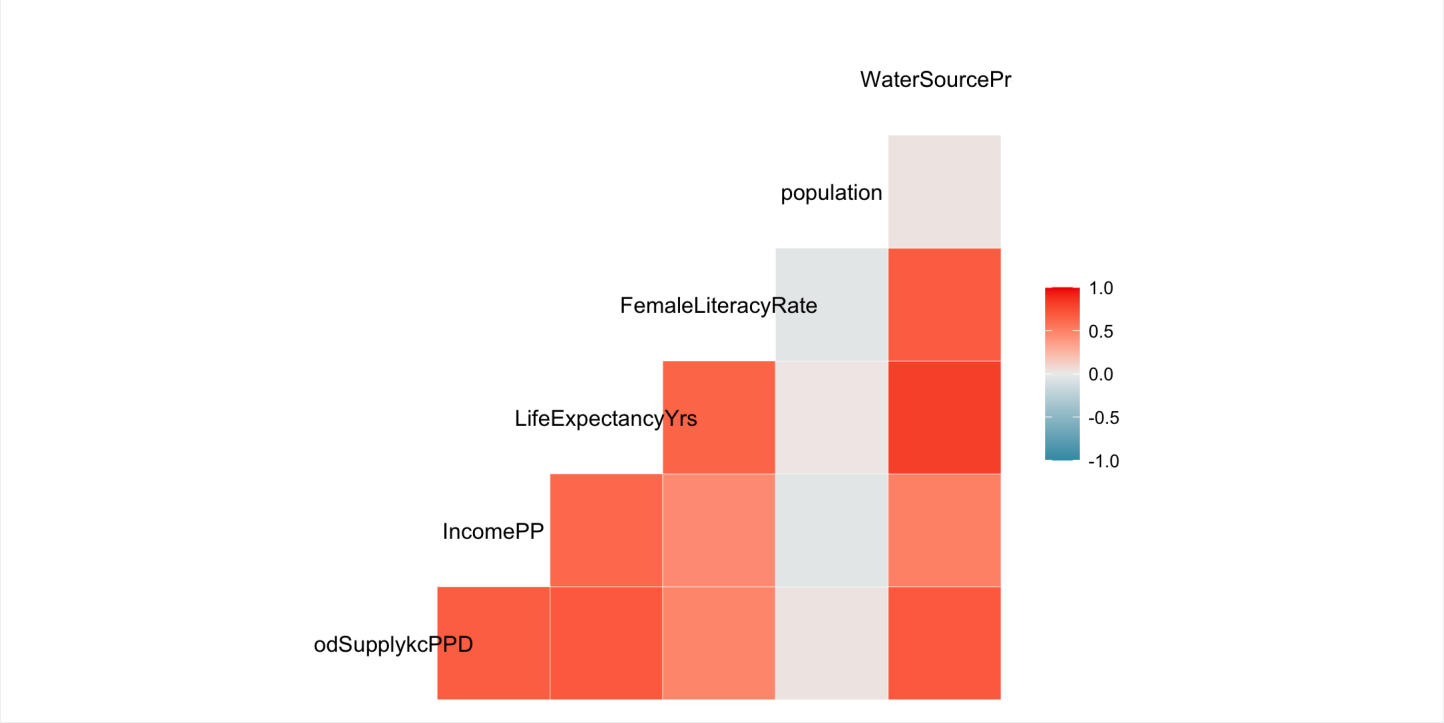

```
corrplot.mixed(M)
```



GGally::ggcorr()

https://ggobi.github.io/ggally/index.html

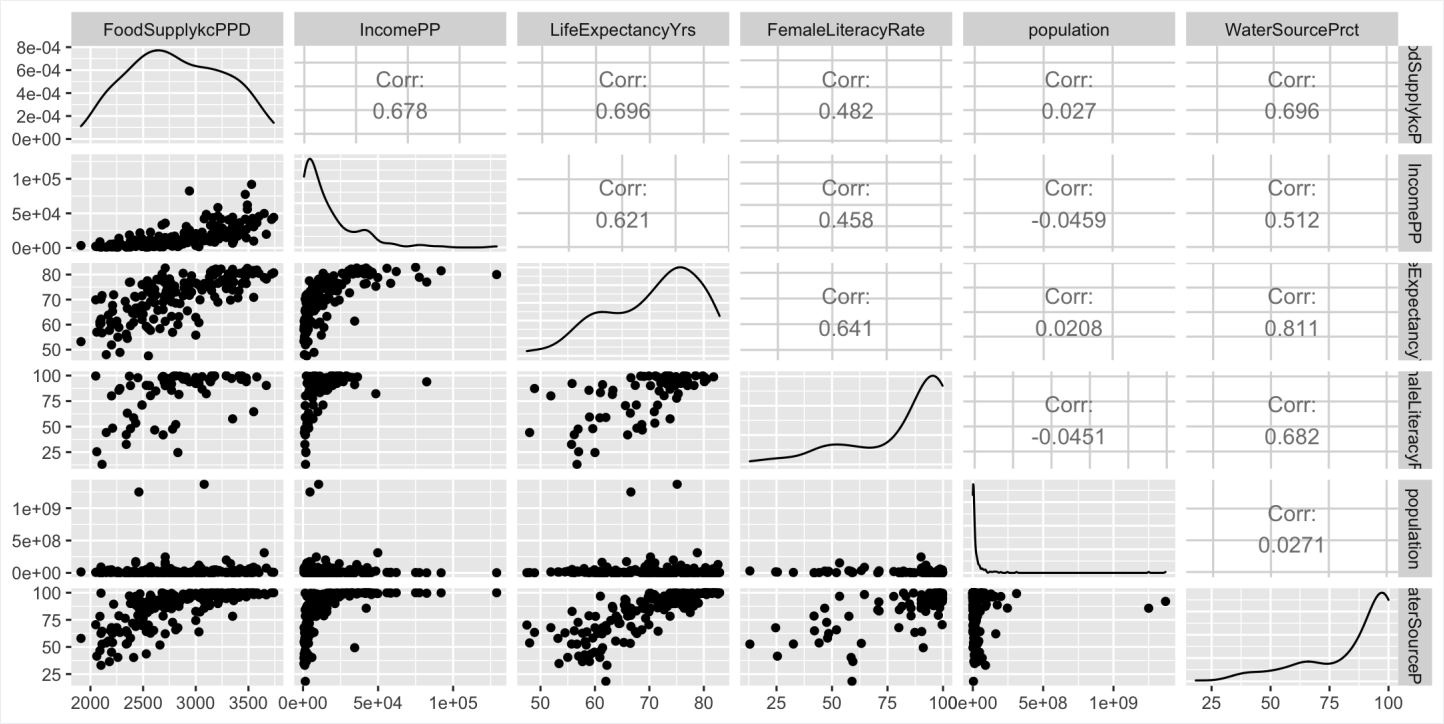
```
# library(GGally)
gapminder2011 %>%
  select(FoodSupplykcPPD:WaterSourcePrct) %>% # specifying which columns to use
  ggcorr()
```



GGally::ggpairs()

https://ggobi.github.io/ggally/index.html

```
# library(GGally)
gapminder2011 %>%
  select(FoodSupplykcPPD:WaterSourcePrct) %>% # specifying which columns to use
  ggpairs()
```

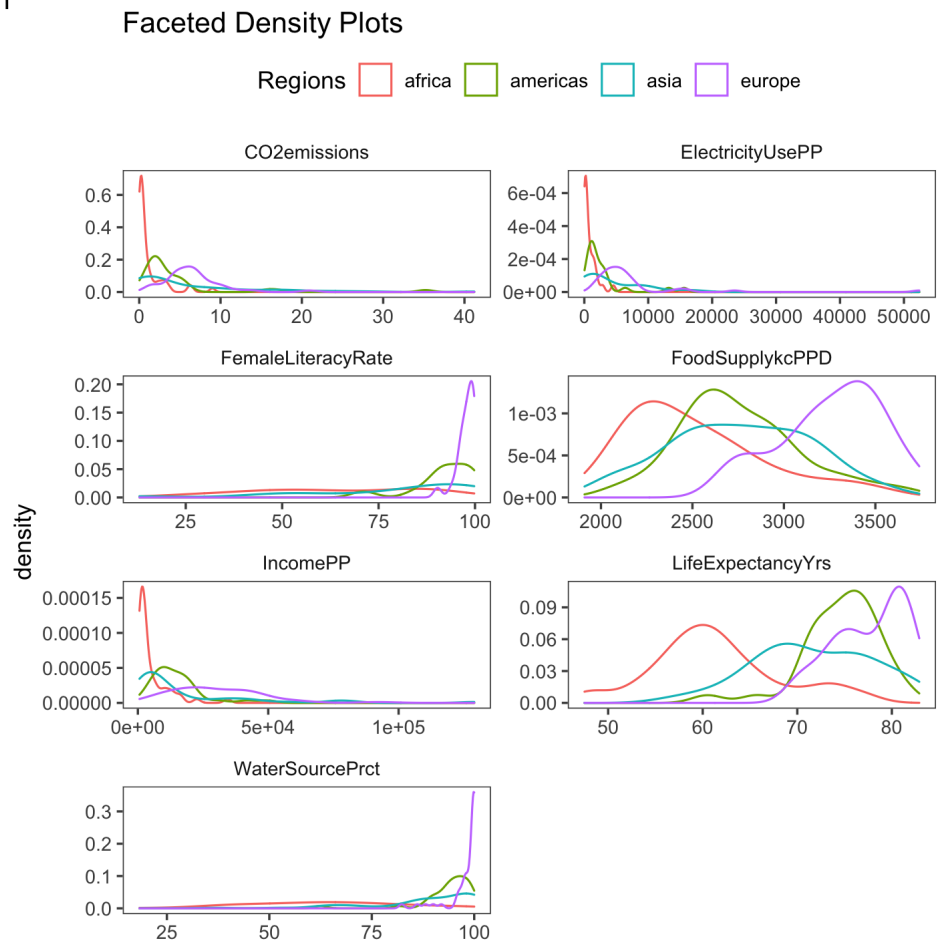


Faceting

Faceted Density Plot

```
ggplot(data = gapminder2011_long,
       aes(x = Values,
           color = four_regions)
       ) +
  facet_wrap(~ Measures,
             scales = "free",
             ncol = 2
             ) +
  geom_density() +
  ggthemes::theme_few() +
  theme(legend.position="top")  +
  labs(
    x = "",
    title = "Faceted Density Plots",
    # Add a figure number!
    tag = "Fig 1",
    # note that color is being
    # specified inside labs!
    color = "Regions"
    )
```

Fig 1



Wide vs. long data

- **Wide** data has one row per subject, with multiple columns for their repeated measurements
- **Long** data has multiple rows per subject, with one column for the measurement variable and another indicating from when/where the repeated measures are from

wide

id	SBP_visit1	SBP_visit2	SBP_visit3
a	130	110	112
b	120	116	122
c	130	136	138
d	119	106	118

See BERD workshop Data Wrangling Part 2 for slides on how to make wide data long.

long

id	visit	SBP
a	1	130
b	1	120
c	1	130
d	1	119
a	2	110
b	2	116
c	2	136
d	2	106
a	3	112
b	3	122
c	3	138
d	3	118

Dataset Gapminder_vars_2011_long.csv (1/2)

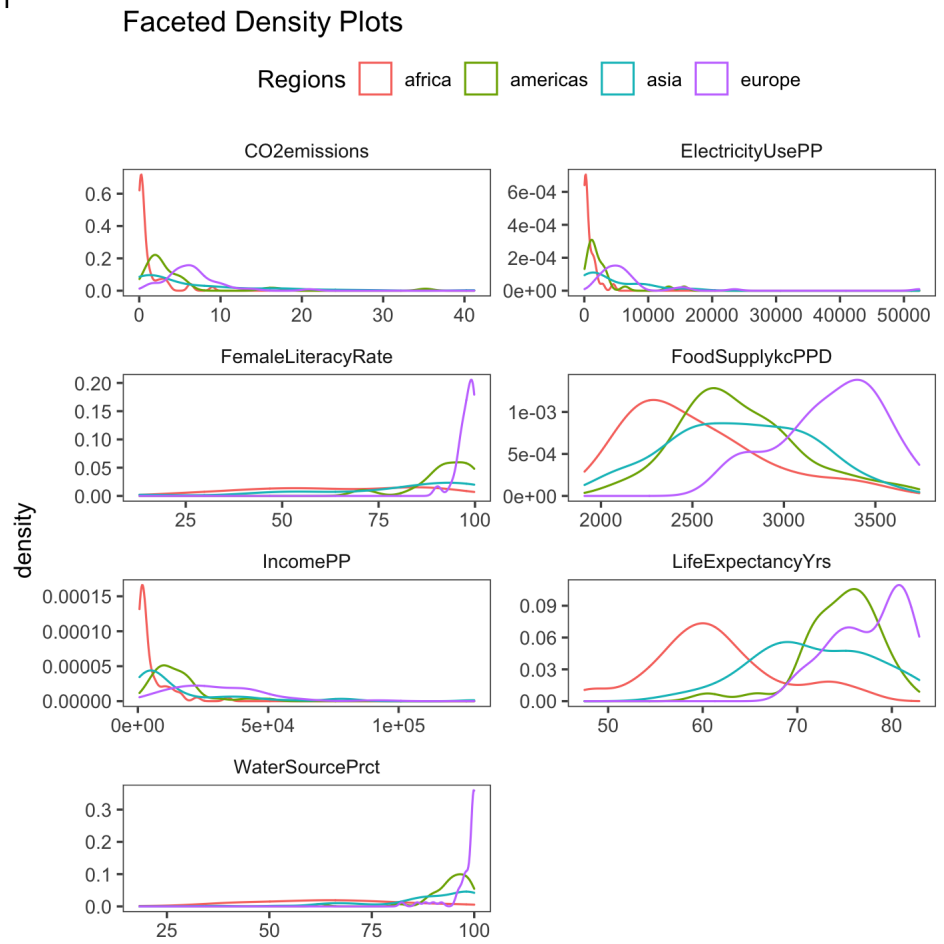
- This is the same 2011 Gapminder data we've been using thus far, but in a **long format** instead of wide.
  - Instead of individual columns for `CO2emissions`, `ElectricityUsePP`, ... `WaterSourcePrct`,
    - there is a column called `Measures` which contains these variables names and
    - a column called `Values` with the actual values for these measures.
  - This means the dataset contains multiple rows per country to account for each of these measures.

```
gapminder2011_long <- read_csv("data/Gapminder_vars_2011_long.csv")
glimpse(gapminder2011_long)
```

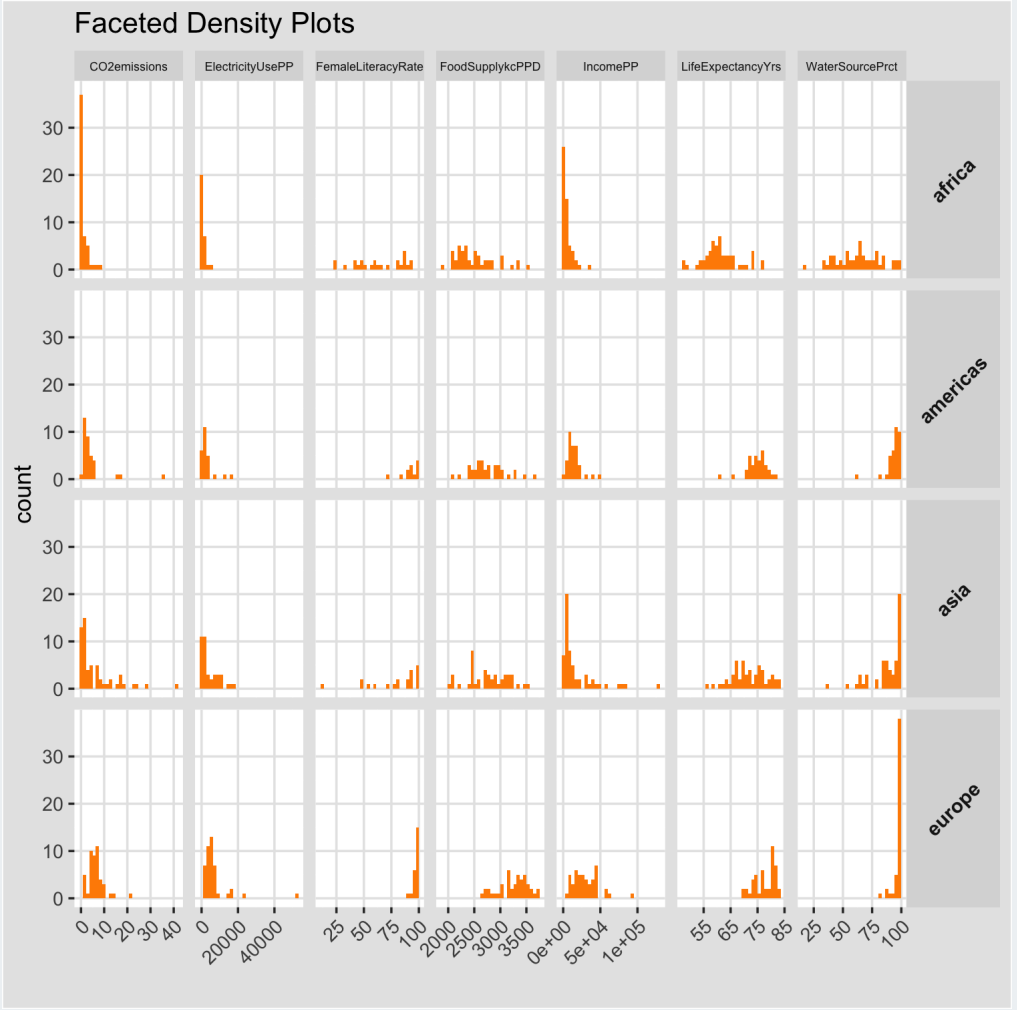
```
Rows: 1,365
Columns: 8
$ country       <chr> "Afghanistan", "Afghanistan", "Afghanista…
$ population    <dbl> 29700000, 29700000, 29700000, 29700000, 2…
$ four_regions  <chr> "asia", "asia", "asia", "asia", "asia", "…
$ eight_regions <chr> "asia_west", "asia_west", "asia_west", "a…
$ six_regions   <chr> "south_asia", "south_asia", "south_asia",…
$ WorldRegions  <chr> "Asia", "Asia", "Asia", "Asia", "Asia", "…
$ Measures      <chr> "CO2emissions", "ElectricityUsePP", "Food…
$ Values        <dbl> 4.12e-01, NA, 2.11e+03, 1.66e+03, 5.67e+0…
```

```
ggplot(data = gapminder2011_long,
       aes(x = Values)
       ) +
  geom_density() +
  facet_wrap(~ Measures,
             scales = "free",
             ncol = 2
             ) +
  aes(color = four_regions) +
  ggthemes::theme_few() +
  theme( legend.position="top")  +
  labs(
    x = "",
    title = "Faceted Density Plots",
    tag = "Fig 1",
    color = "Regions"
    )
```

Fig 1



```
ggplot(data = gapminder2011_long,
       aes(x = Values)
       ) +
  facet_grid(
    four_regions ~ Measures,
    scales = "free_x"
    ) +
  geom_histogram(fill = "darkorange") +
  ggthemes::theme_igray() +
  theme(
    strip.text.y =
      element_text(size=10,
                   angle=45,
                   face = "bold"),
    strip.text.x = element_text(size=6),
    axis.text.x = element_text(angle=45,
                               hjust=1)
    ) +
  labs(
    x = "",
    title = "Faceted Density Plots"
    )
```



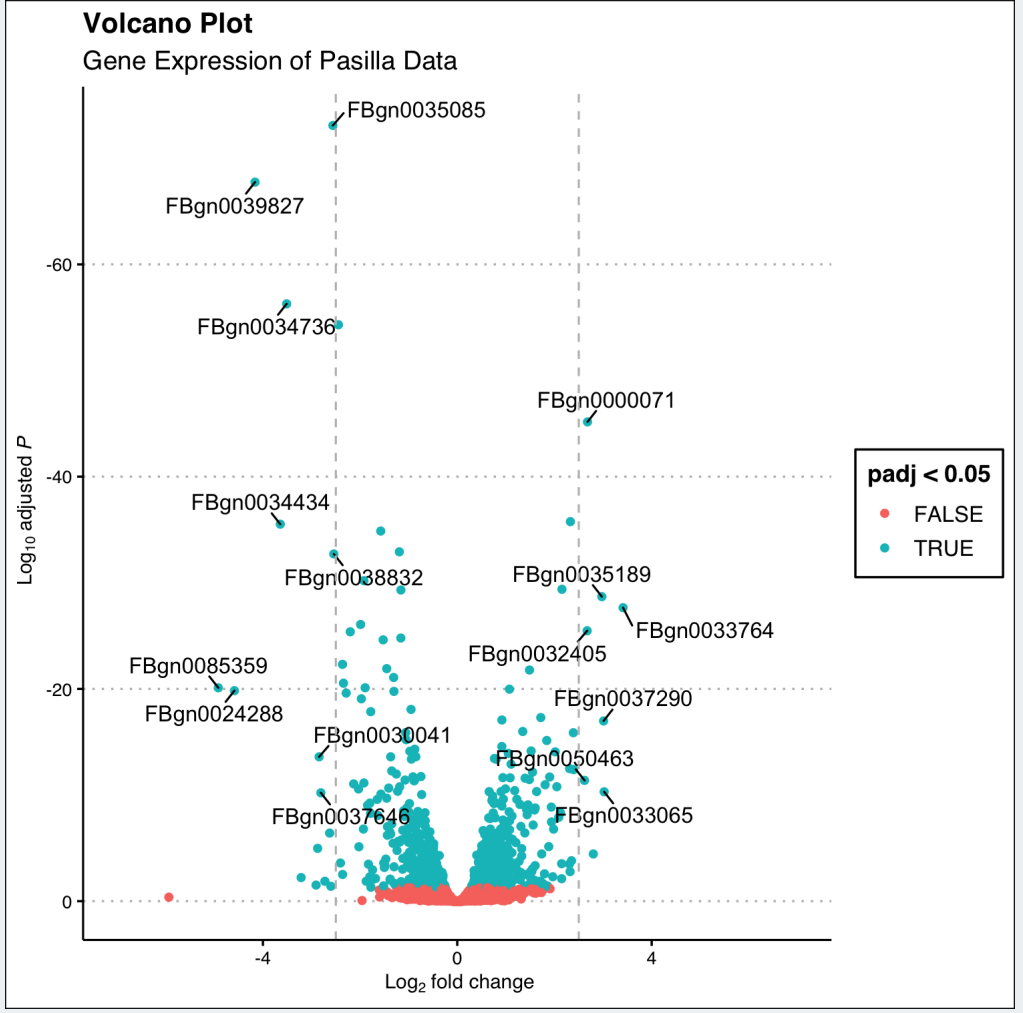
Gene Expression

Pasilla Data

```
glimpse(pasilla_data)
```

```
Rows: 8,377
Columns: 15
$ gene           <chr> "FBgn0000008", "FBgn0000017", "FBgn00000…
$ baseMean       <dbl> 95.144292, 4352.553569, 418.610484, 6.40…
$ fc             <dbl> 1.0015792, 0.8467929, 0.9300151, 1.15736…
$ log2FoldChange <dbl> 0.002276441, -0.239918944, -0.104673912,…
$ lfcSE          <dbl> 0.2237287, 0.1263369, 0.1484891, 0.68958…
$ stat           <dbl> 0.01017501, -1.89904084, -0.70492676, 0.…
$ pvalue         <dbl> 9.918817e-01, 5.755911e-02, 4.808558e-01…
$ padj           <dbl> 9.972108e-01, 2.880017e-01, 8.268337e-01…
$ treated1       <dbl> 7.607917, 11.938311, 9.143372, 6.479135,…
$ treated2       <dbl> 7.834912, 12.024557, 9.011505, 6.577240,…
$ treated3       <dbl> 7.595052, 12.013565, 8.944883, 6.475226,…
$ untreated1     <dbl> 7.567298, 12.045721, 9.315269, 6.565256,…
$ untreated2     <dbl> 7.642174, 12.284647, 9.098290, 6.479802,…
$ untreated3     <dbl> 7.844603, 12.455939, 8.966546, 6.422196,…
$ untreated4     <dbl> 7.669147, 12.077404, 9.066286, 6.395509,…
```

```
ggplot(data = pasilla_data,
       aes(x = log2FoldChange,
           y = log10(padj))) +
  geom_point() +
  scale_y_reverse() +
  aes(color = padj < 0.05) +
  ggrepel::geom_text_repel(
    data = pasilla_data_top,
    aes(label = gene), color = "black",
    box.padding = 0.5,
    min.segment.length = 0) +
  xlim(c(-7,7)) +
  geom_vline(xintercept = c(-2.5, 2.5),
             lty = "dashed", color="grey") +
  ggthemes::theme_clean() +
  labs(
    x = bquote(~Log[2]~ "fold change"),
    y = bquote(~Log[10]~adjusted~italic(P)),
    title = "Volcano Plot",
  subtitle="Gene Expression of Pasilla Data"
  )
```



Heatmap with pheatmap::pheatmap()

It's possible to make heatmaps in ggplot2 with `geom_tile()`, but there are many other better functions using base R that cluster and annotate the data. This is using **pheatmap** package.

We need to create the data:

```
# select expression data
pasilla_heat <- pasilla_data %>%
  select(treated1:untreated4)
# subtract off gene-specific means
pasilla_heat <- pasilla_heat - rowMeans(pasilla_heat)
# calculate standard deviation of each centered gened
sd_gene <- apply(pasilla_heat,1,sd)
# select top 500 most variable
pasilla_heat <-
  pasilla_heat[order(sd_gene, decreasing = TRUE)[1:500],]

# create annotation data
pasilla_col <- data.frame(
  trt = factor(c(rep("trt",3), rep("untrt",4))),
  id = 1:7,
  row.names=colnames(pasilla_heat))
```

```
head(pasilla_heat, n = 3)
```

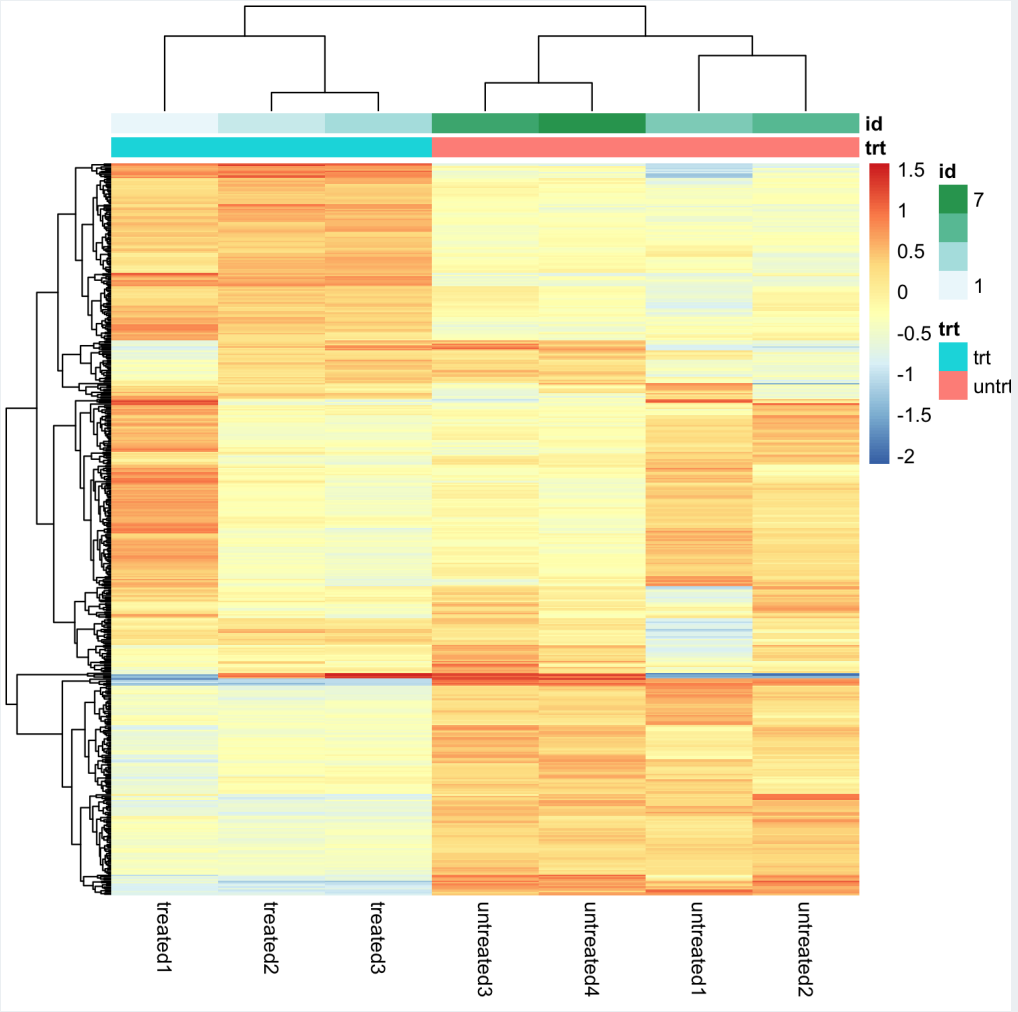
```
       treated1  treated2 treated3 unt
2390 -1.5997691 0.8713581 1.568570  -1
521  -1.3218267 0.9954861 1.278523  -1
7886 -0.5901012 0.8225366 1.339219  -1
     untreated3 untreated4
2390   1.338488  1.4253512
521    1.040472  0.9541077
7886   1.155933  0.7369965
```

```
pasilla_col
```

```
             trt id
treated1     trt  1
treated2     trt  2
treated3     trt  3
untreated1 untrt  4
```

Heatmap with pheatmap::pheatmap()

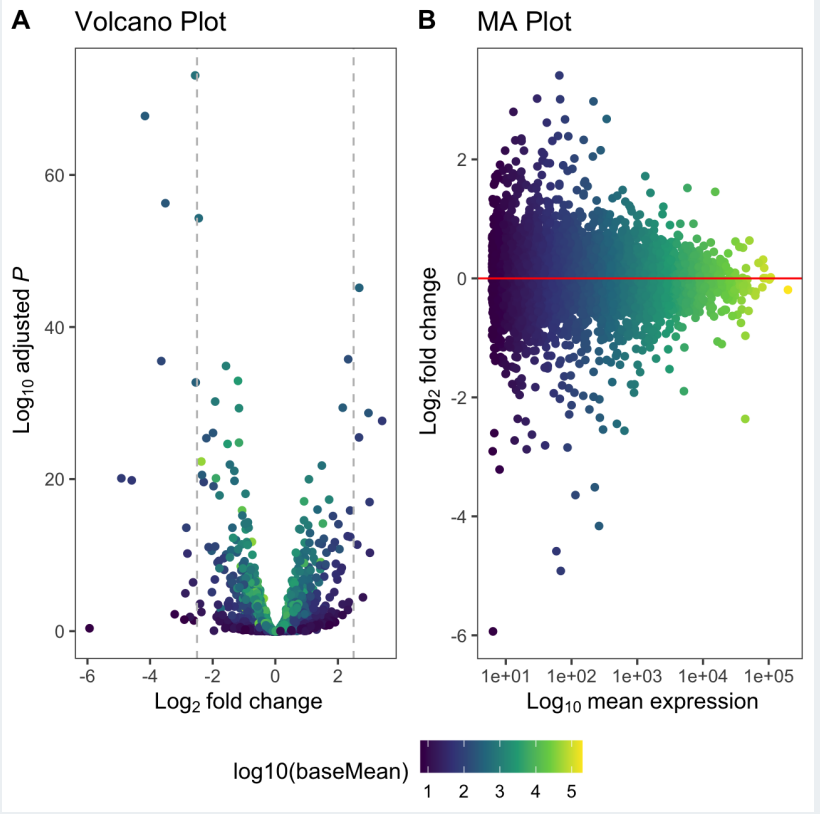
```
pheatmap::pheatmap(
  mat = pasilla_heat,
  show_rownames = FALSE,
  annotation_col = pasilla_col
)
```



Side by side plot with ggpubr

```
p1 <- ggplot(data = pasilla_data,
       aes(x = log2FoldChange,
           y = -log10(padj),
           color = log10(baseMean))) +
  geom_point() +
  geom_vline(xintercept = c(-2.5, 2.5),
             lty = 2, color="grey") +
  theme_few() + scale_color_viridis_c() +
  labs(x = bquote(~Log[2]~ "fold change"),
       y = bquote(~Log[10]~adjusted~italic(P)),
       title = "Volcano Plot")
p2 <- ggplot(data = pasilla_data,
       aes(x = baseMean,
           y = log2FoldChange,
           color = log10(baseMean))) +
  geom_point() +
  scale_x_log10() +
  geom_hline(yintercept = 0, color = "red") +
  theme_few() + scale_color_viridis_c() +
  labs(y = bquote(~Log[2]~ "fold change"),
       x = bquote(~Log[10]~ "mean expression"),
       title = "MA Plot")
```

```
ggpubr::ggarrange(p1, p2, labels = "AUTO",
  common.legend = TRUE, legend = "bottom")
```

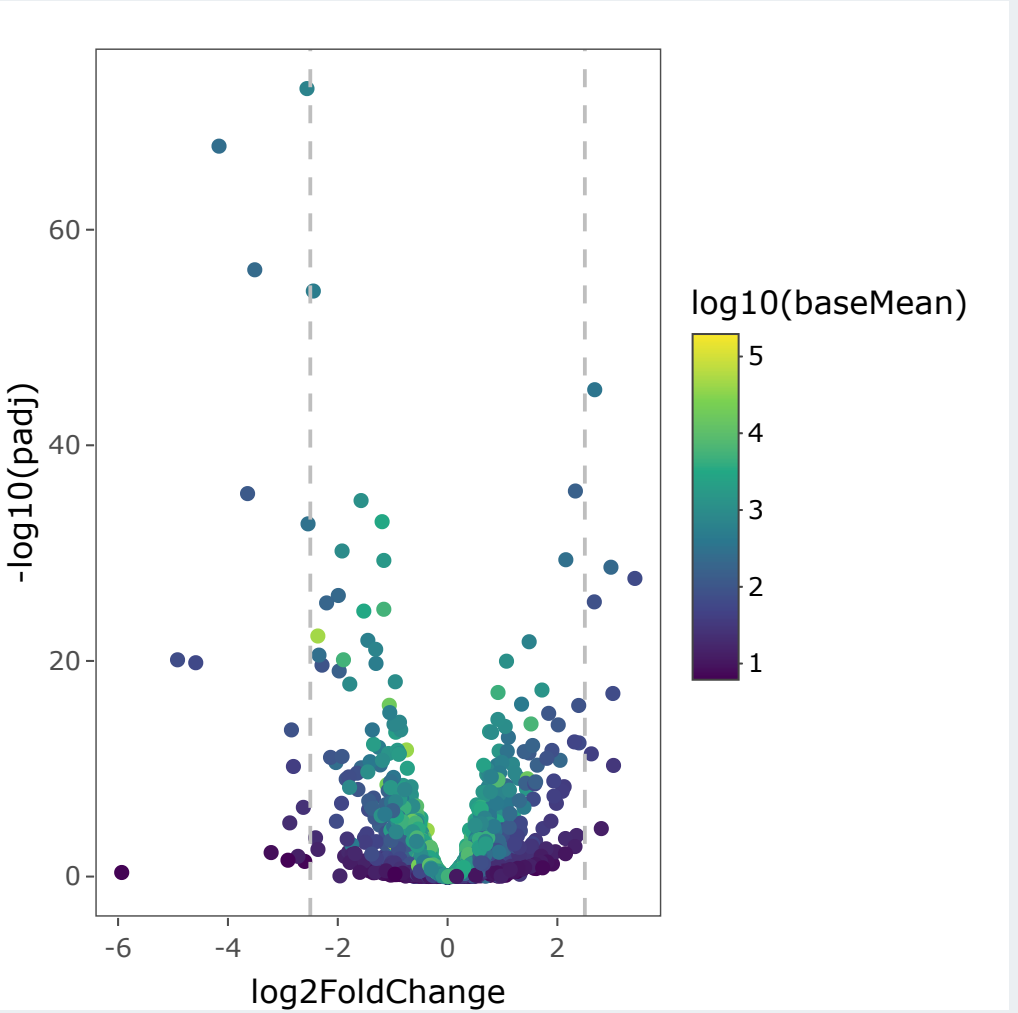


Other options: cowplot and patchwork.

Interactive plotly graphs with ggplotly()

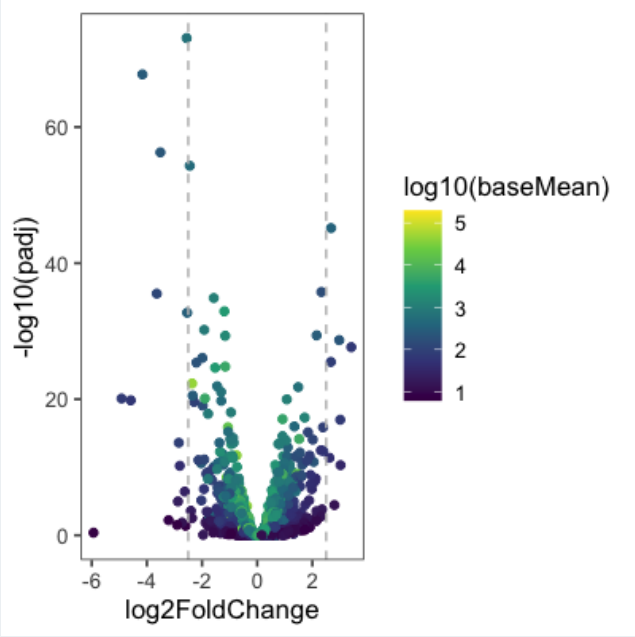
```
# Save ggplot
p1 <- ggplot(
  data = pasilla_data,
  aes(x = log2FoldChange,
      y = -log10(padj),
      color = log10(baseMean),
      key = gene)
) +
  geom_point() +
  geom_vline(
    xintercept = c(-2.5, 2.5),
    lty = 2, color="grey") +
  theme_few() +
  scale_color_viridis_c()
```

```
plotly::ggplotly(p1)
```

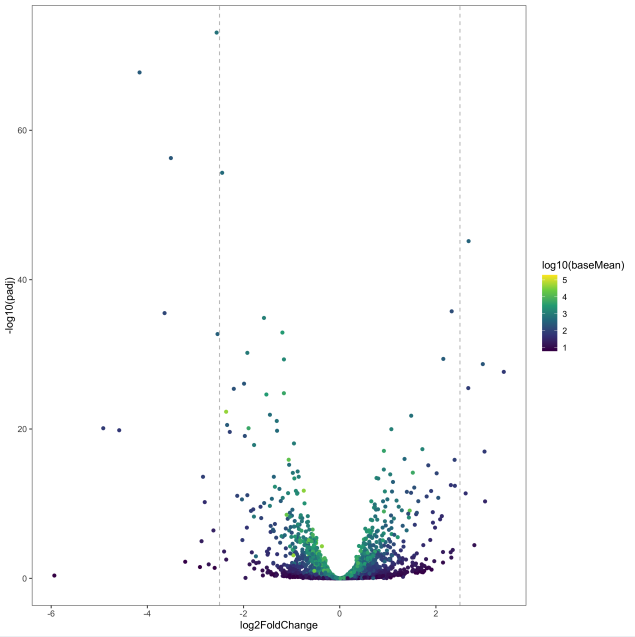


Saving plots

```
ggsave(plot = p1,
       filename = "figs/volcanoplot_small.png",
       height = 4,
       width = 4,
       units = "in",
       dpi = 100)
```




```
ggsave(plot = p1,
       filename = "figs/volcanoplot_large.png",
       height = 10,
       width = 10,
       units = "in",
       dpi = 300)
```



Exercise

Complete the fifth section of the `practice_ggplot.Rmd` file: "Histogram".

References and Links

from Data to Viz

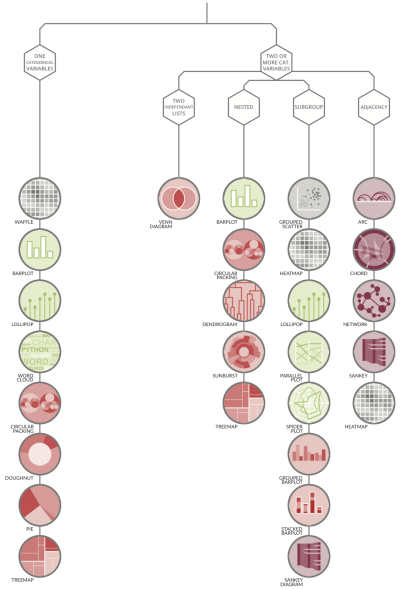
'From Data to Viz' is a classification of chart types based on input data format. It will help you find the perfect chart in three simple steps :

1. Identify what type of data you have.
2. Go to the corresponding decision tree and follow it down to a set of possible charts.
3. Choose the chart from the set that will suit your data and your needs best.

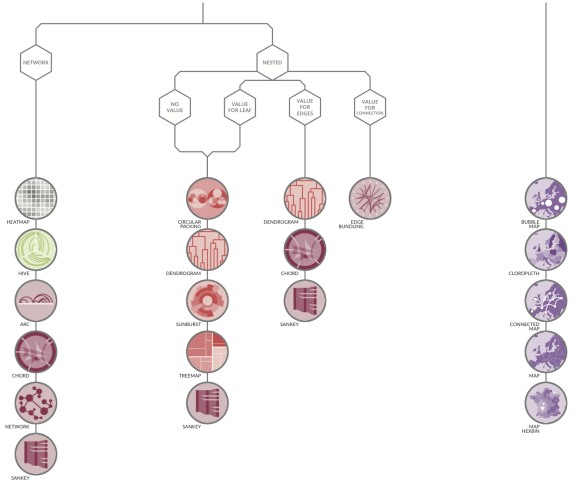
Dataviz is a world with endless possibilities and this project does not claim to be exhaustive. However it should provide you with a good starting point. For an interactive version and much more, visit:

data-to-viz.com

CATEGORIC

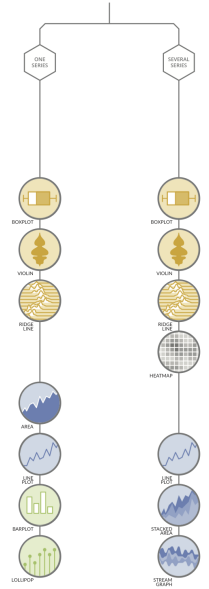


RELATIONAL

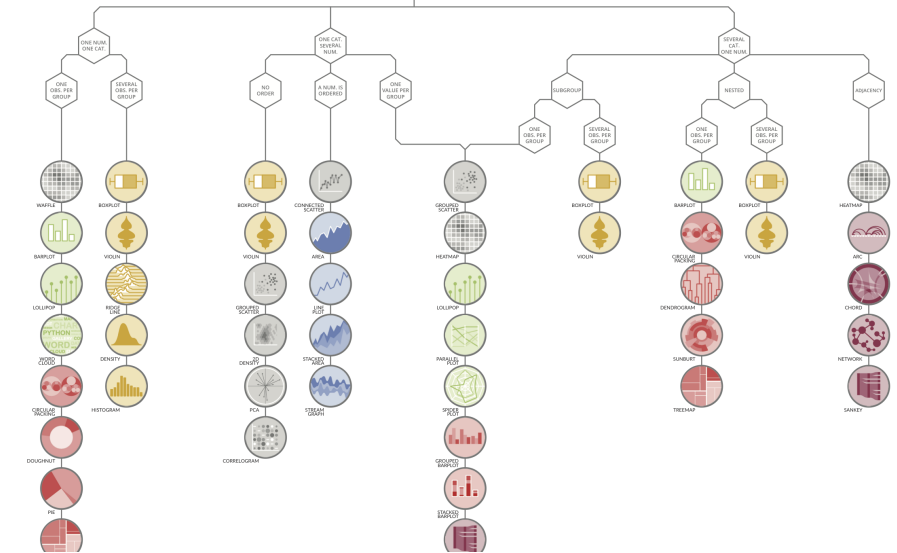


MAP

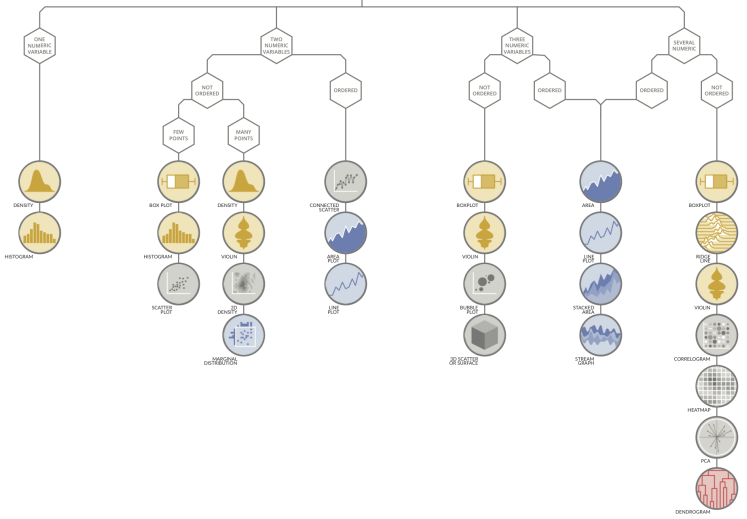
TIME SERIES



CATEGORIC AND NUMERIC



NUMERIC



WHAT DO YOU WANT TO SHOW ?

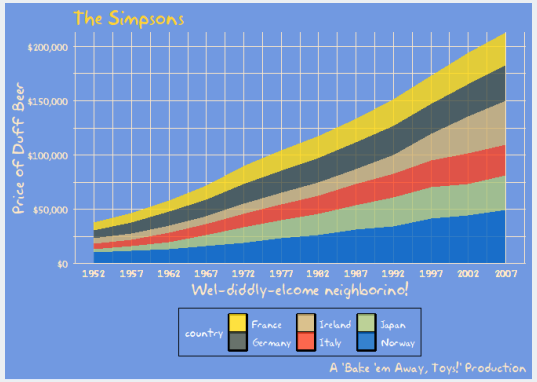
- Distribution
- Correlation
- Ranking
- Part of a whole
- Evolution
- Maps
- Flow

Many, many ggplot extensions!

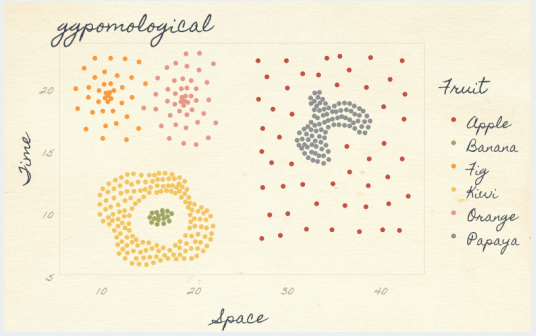
Some examples at the ggplot2 extensions gallery

Many, many themes and palettes/scales!

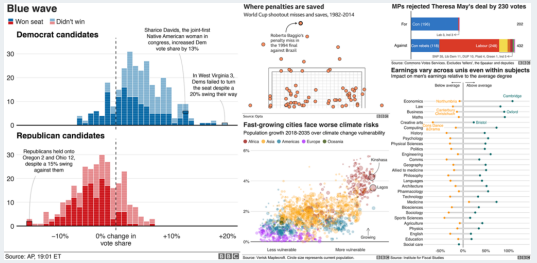
We used themes from `ggthemes` and `hrbrthemes` as well as built in themes, but there are many more:



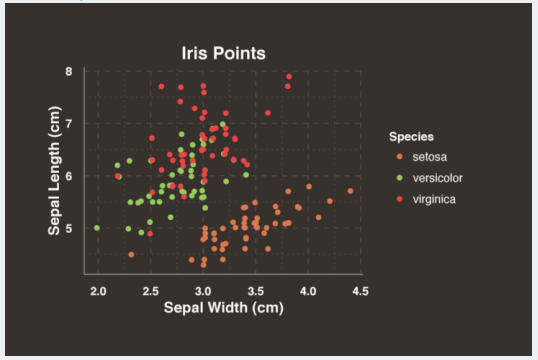
TV Themes



ggpomological



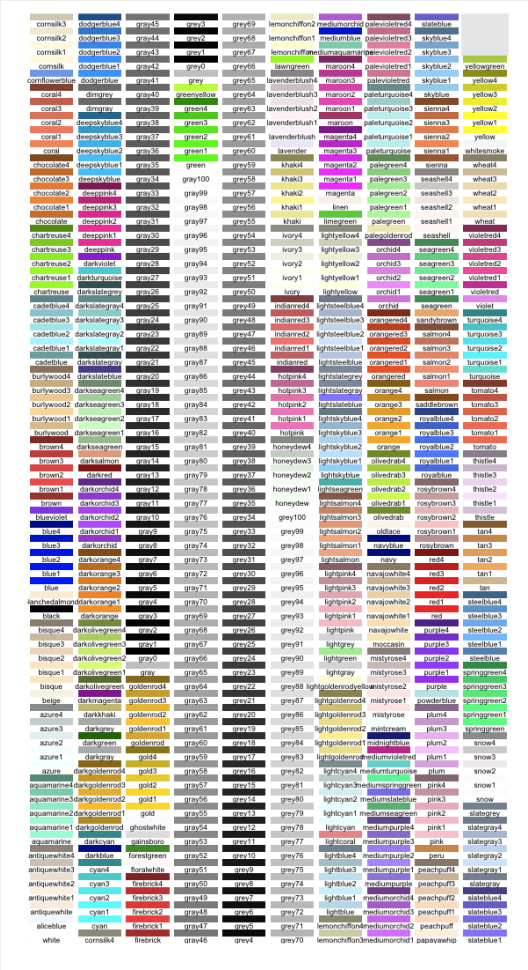
bbplot for BBC themes



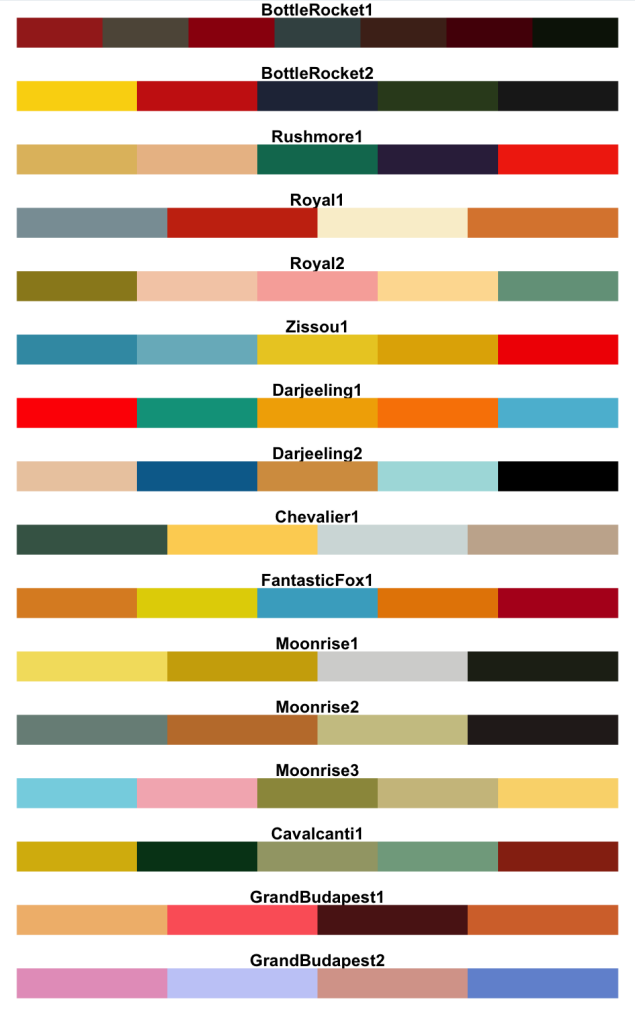
ggthemr

from "Themes to improve your ggplot figures" by David Keyes

R colors and palettes



Built in R Colors



wesanderson package

Changing the order of names (levels) within a categorical variable

- The default order of names within a categorical variable is alphanumeric
  - Such as `africa, americas, asia, europe` for the `four_regions` variable
- Often we want a different order when making plots though.

`factor` level variables

Do this by making the categorical variable a `factor` level variable in R.

- We can change the order of names within a `factor` level variable, and even rename the levels.
- The `forcats` package makes this easy to do. See https://forcats.tidyverse.org/.

References

- ggplot cheatsheet
- ggplot2 package reference
- ggplot2: Elegant Graphics for Data Analysis by Hadley Wickham
- *Data Visualizaton* online textbook by Kieran Healy
- R Graphics Cookbook by Winston Chang
- *R for Data Science* online textbook by Hadley Wickham
- *Introduction to Data Science* online textbook by Rafael A. Irizarry

Example plots and extensions:

- R Graph Gallery
- ggplot2 extension gallery
- All Your Figure Are Belong To Us
- from Data to Viz - beautiful flowcharts to help you decide on a plot based on the variable type(s); check out their poster
- Top 50 ggplot2 Visualizations - The Master List (With Full R Code)

OHSU class:

- CS 631 Data Visualization

Inspiration for this talk

- github/flipbookr
- Kieran Healy's rstudio::conf2020 data viz materials

Thank you!

Contact info:

- Jessica Minnier: *minnier@ohsu.edu*
- Meike Niederhausen: *niederha@ohsu.edu*

This workshop info:

- Code for these slides are on github, with links to other course materials: jminnier/berd_r_courses
- The `.Rmd` file that generated the slides is on github and can be downloaded here, though you need to download the whole R project to knit the file.
- The project folder of examples can be downloaded at github.com/jminnier/berd_ggplot_project & the solutions are in the `solns/` folder.