

Data Visualization with R and ggplot2

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

2020/03/04 & 2020/03/11

 slides: bit.ly/berd_ggplot

 pdf: bit.ly/berd_ggplot_pdf

Load files for today's workshop

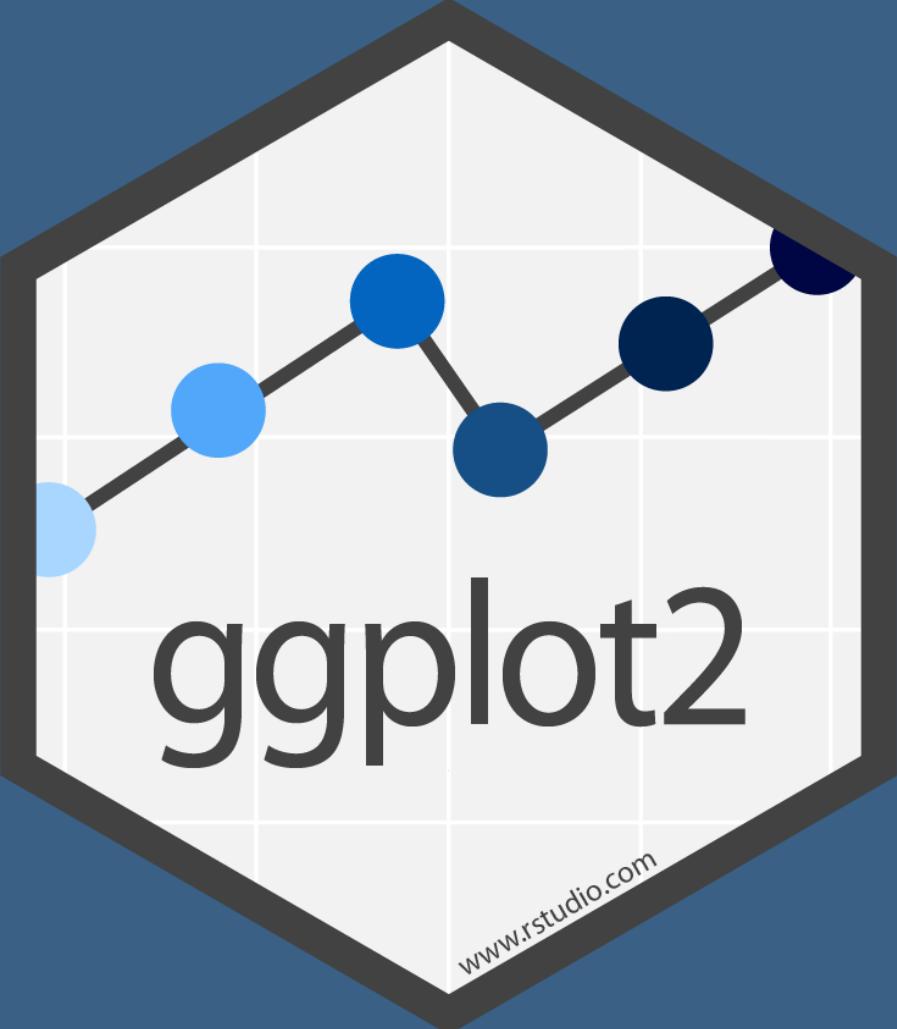
1. Open slides bit.ly/berd_ggplot
2. Get project folder
 - Download zip folder at bit.ly/berd_ggplot_zip
 - UNZIP completely (right click-> "extract all")
 - Open unzipped folder
 - Open (double click) `berd_ggplot_project.Rproj`
 - Inside RStudio 'Files' tab: click on file `00-install.R` and click "Run"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



Allison Horst

Learning objectives

- Understand the basic idea behind grammar of graphics
- Be able to map 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.

ggplot basics

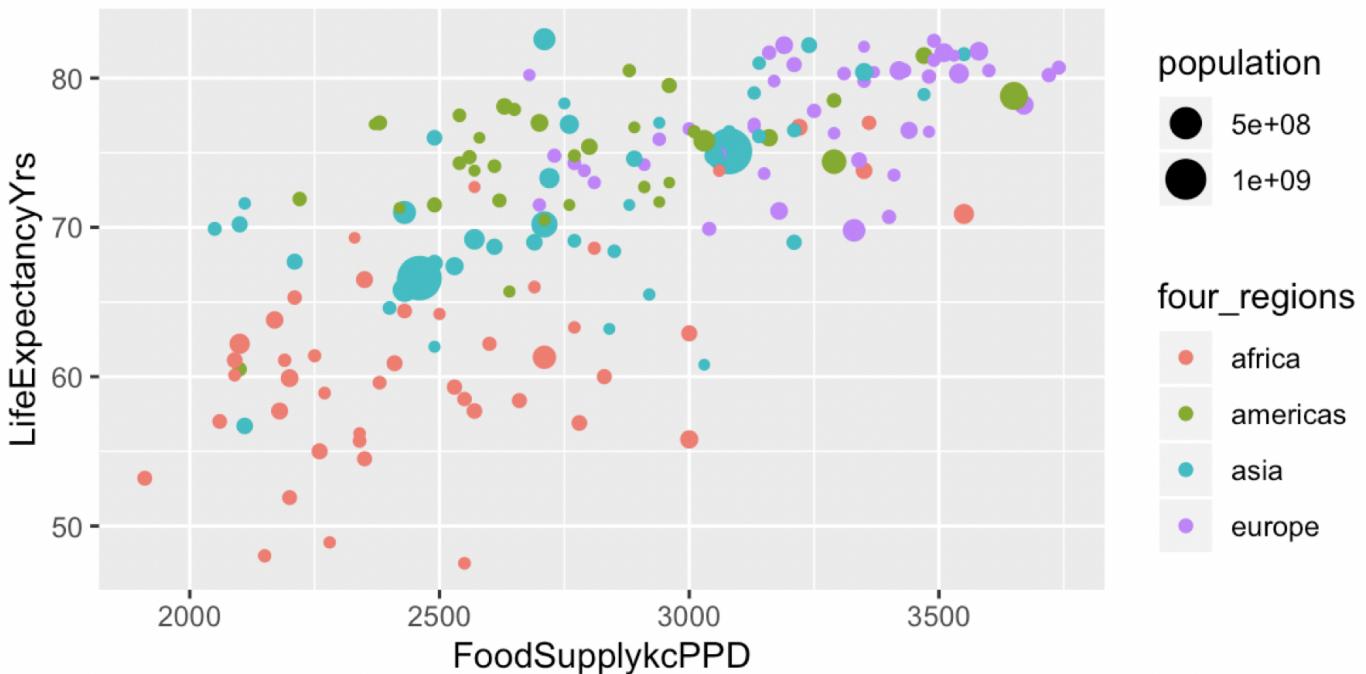
Function

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



Grammar of ggplot2

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

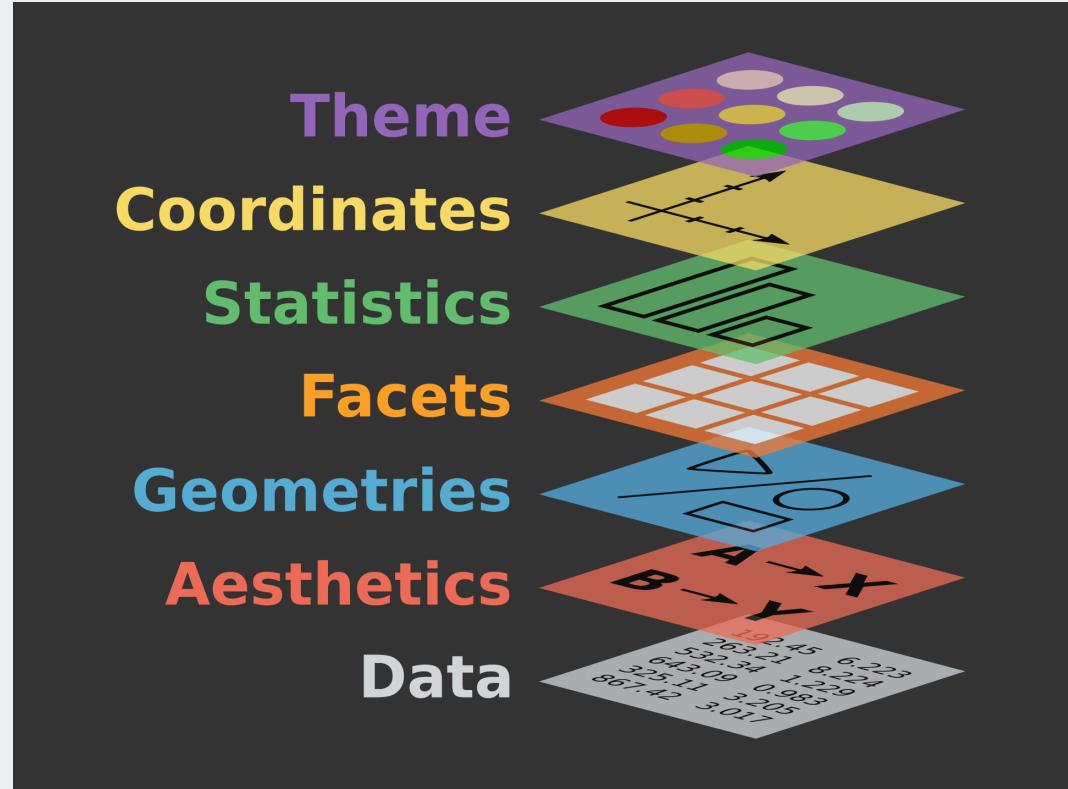


To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



ggplot cheatsheet

Theme
Coordinates
Statistics
Facets
Geometries
Aesthetics
Data



Grammar of ggplot2

1. Tidy Data

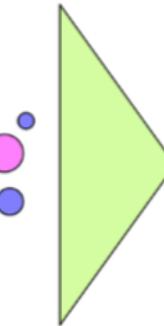
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

```
ggplot(data = gapminder, mapping =  
       aes(x = gdp,  
             y = lifespan,  
             color = continent,  
             size = pop))
```

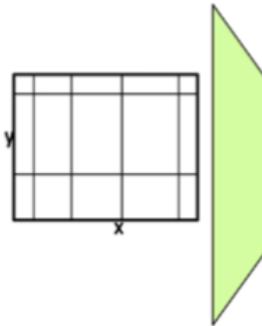
2. Mapping



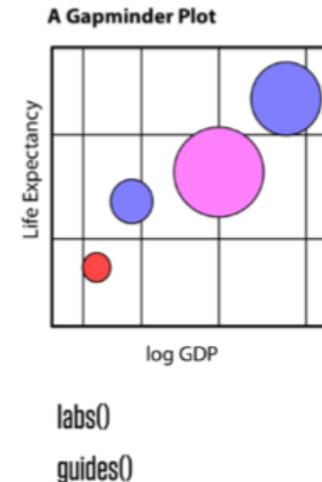
3. Geom



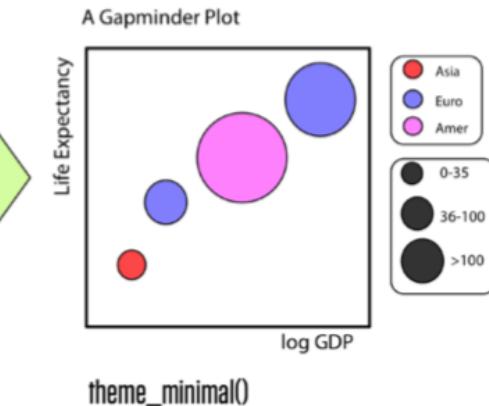
4. Co-ordinates, Scales



5. Labels & Guides



6. Themes



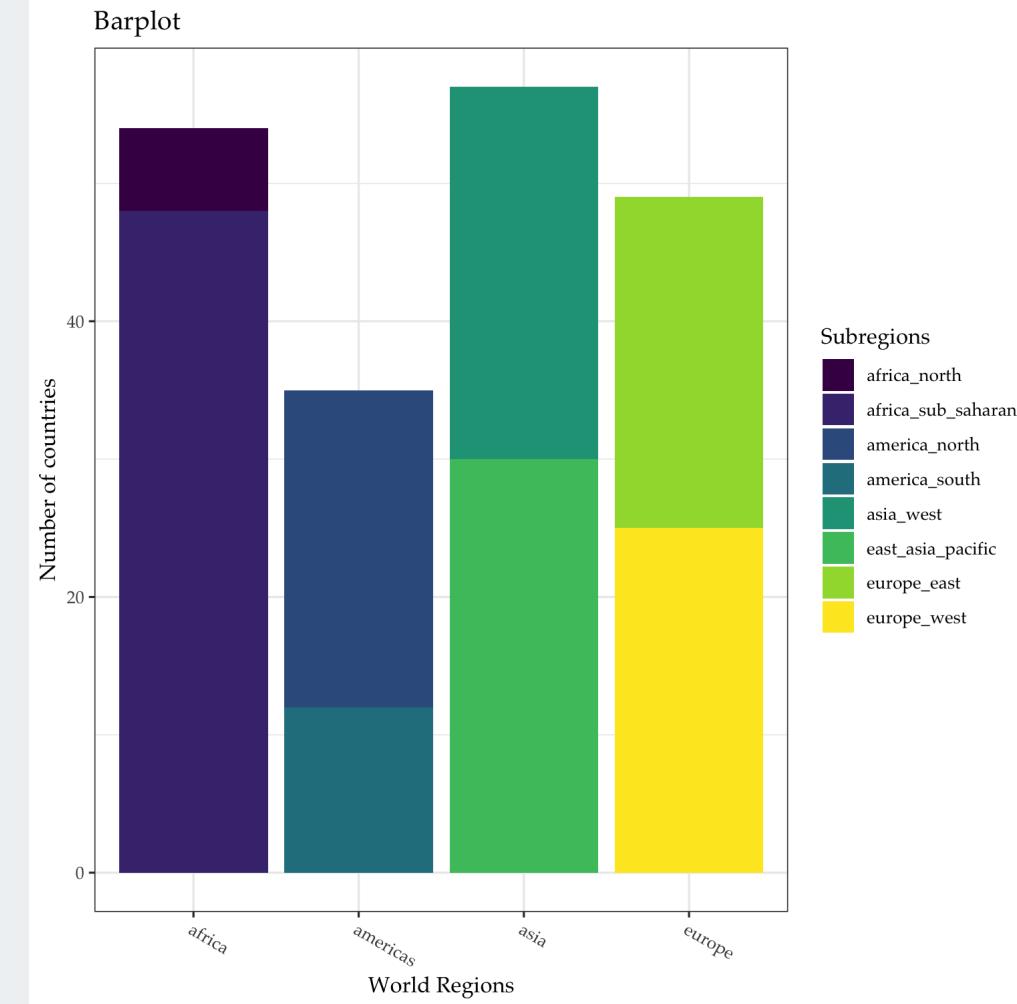
Kieran Healy

Visual Table of Contents

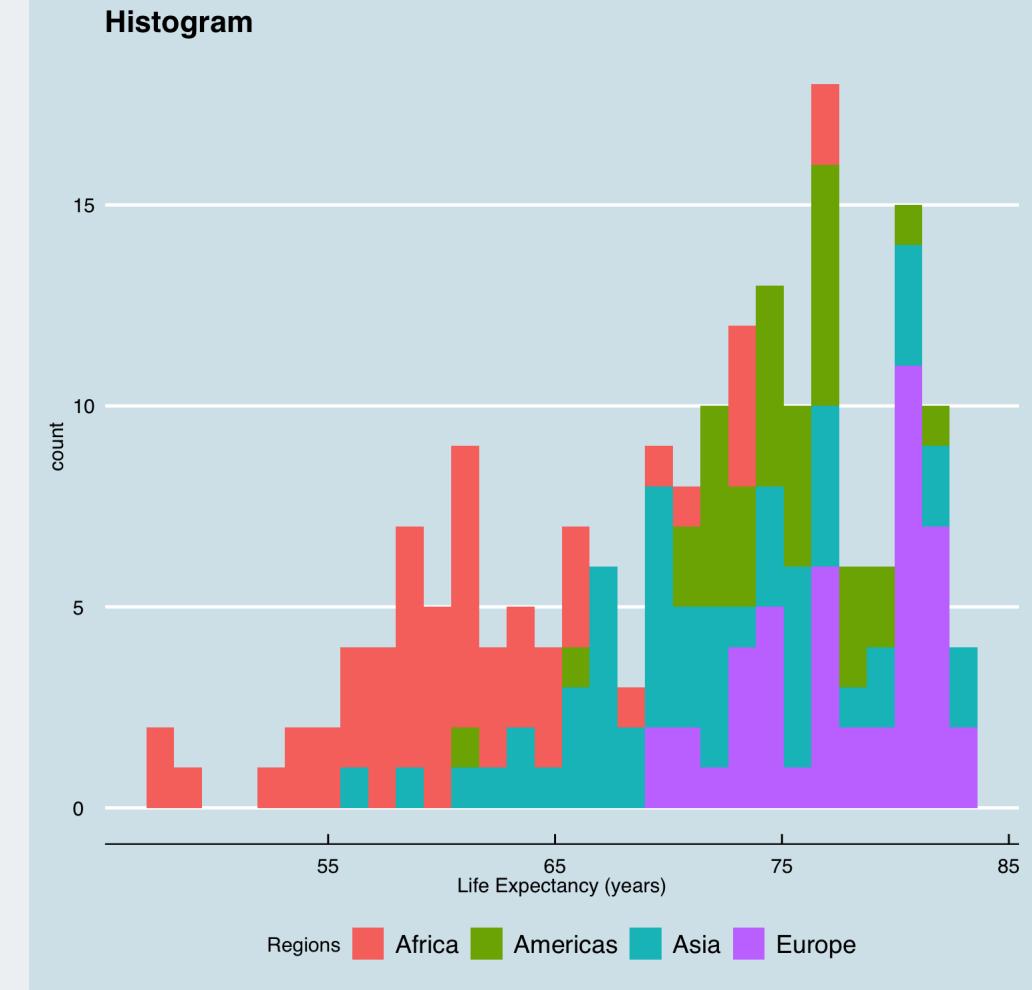


Inspired by [EvaMaeRey \(Gina Reynolds\)](#), author of the amazing `flipbookr` package.

```
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"))
```



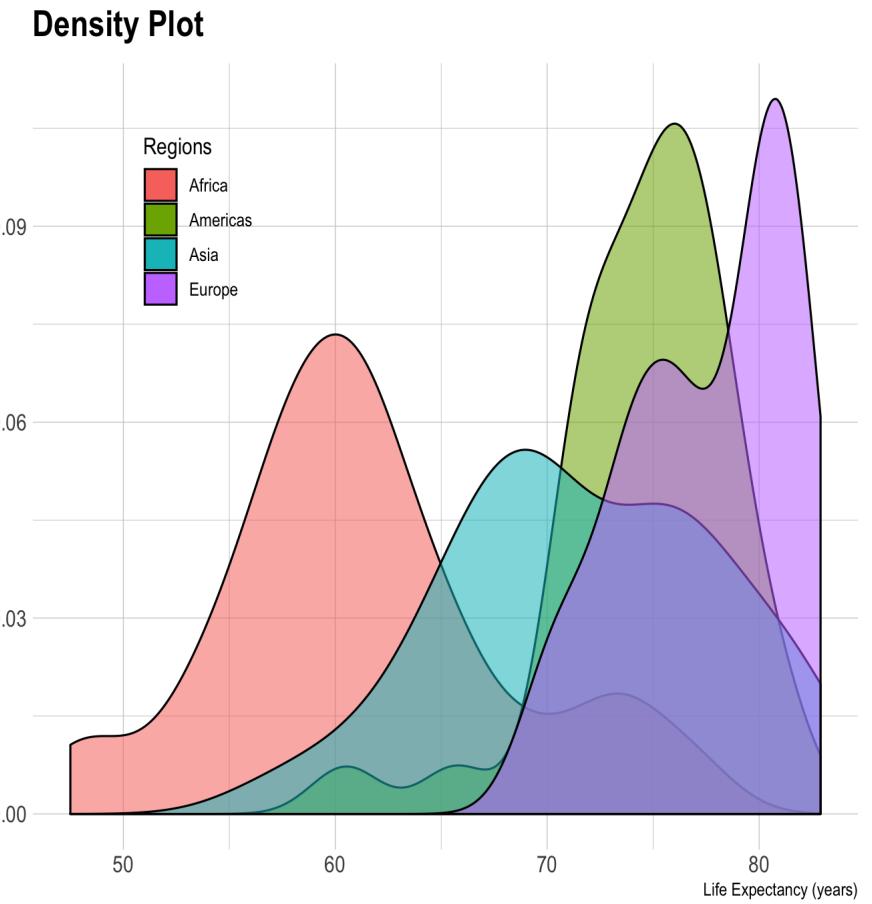
```
ggplot(data = gapminder2011) +  
  aes(x = LifeExpectancyYrs) +  
  geom_histogram() +  
  aes(fill = four_regions) +  
  scale_fill_discrete(  
    name = "Regions",  
    labels = c("Africa", "Americas",  
              "Asia", "Europe"))  
  ) +  
  labs(  
    x = "Life Expectancy (years)", 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



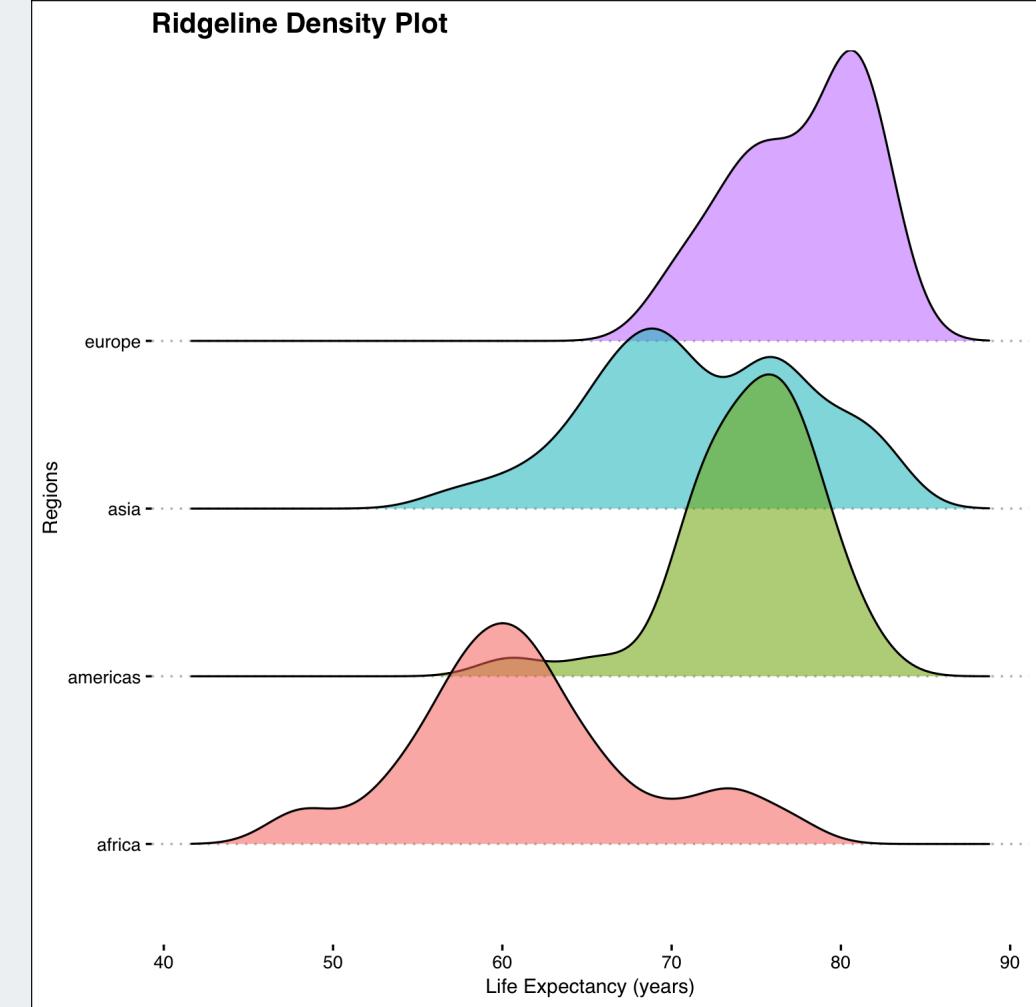
Legend position

- "Generic" positions
 - `legend.position = "left"`
 - Other options: "top", "right", "bottom", "none"
- Specified 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**

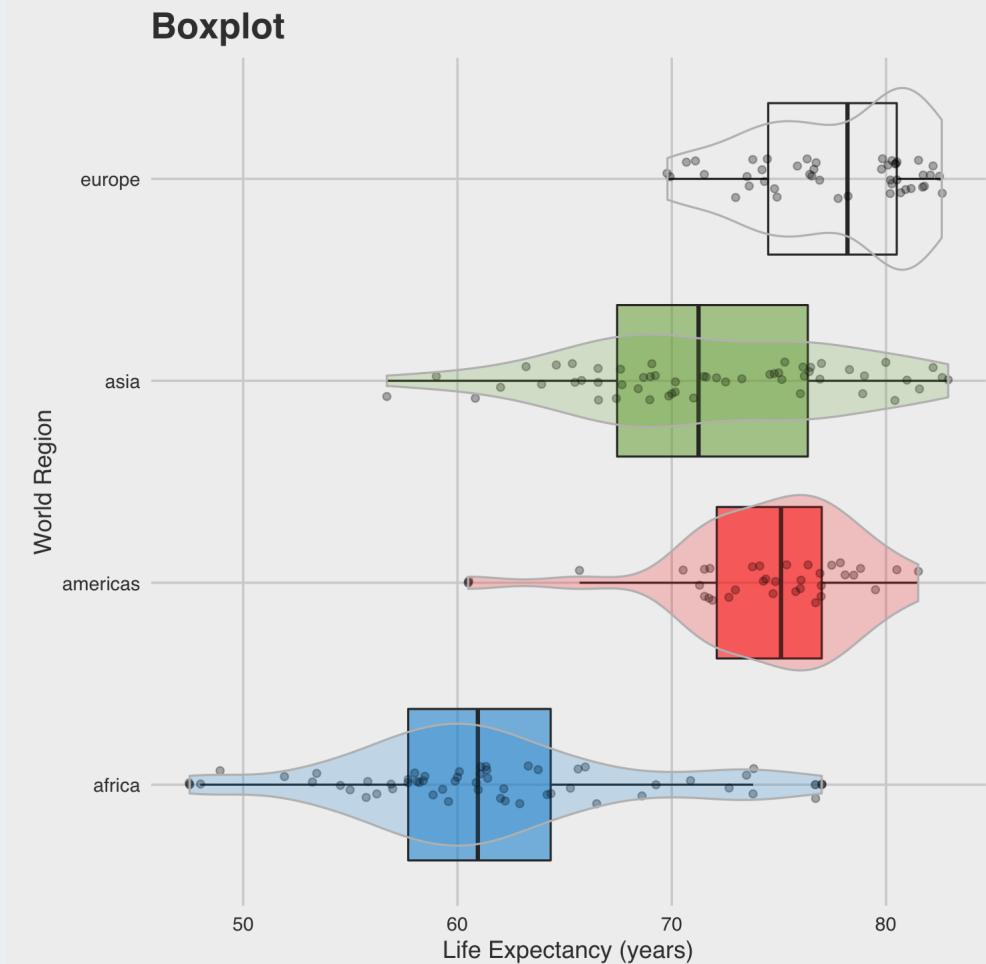
```
ggplot(data = gapminder2011) +  
  aes(x = LifeExpectancyYrs) +  
  geom_density() +  
  aes(fill = four_regions) +  
  aes(alpha=.4) +  
  scale_fill_discrete(  
    name = "Regions",  
    labels = c("Africa", "Americas",  
              "Asia", "Europe"))  
  ) +  
  scale_alpha(guide = "none") +  
  hrbrthemes::theme_ipsum() +  
  theme(legend.position=c(.2,.8)) +  
  labs(  
    x = "Life Expectancy (years)",  
    title = "Density Plot"  
  )
```



```
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(y = LifeExpectancyYrs) +  
  geom_boxplot() +  
  aes(x = four_regions) +  
  aes(fill = four_regions) +  
  aes(alpha=.3) +  
  coord_flip() +  
  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"  
  )
```



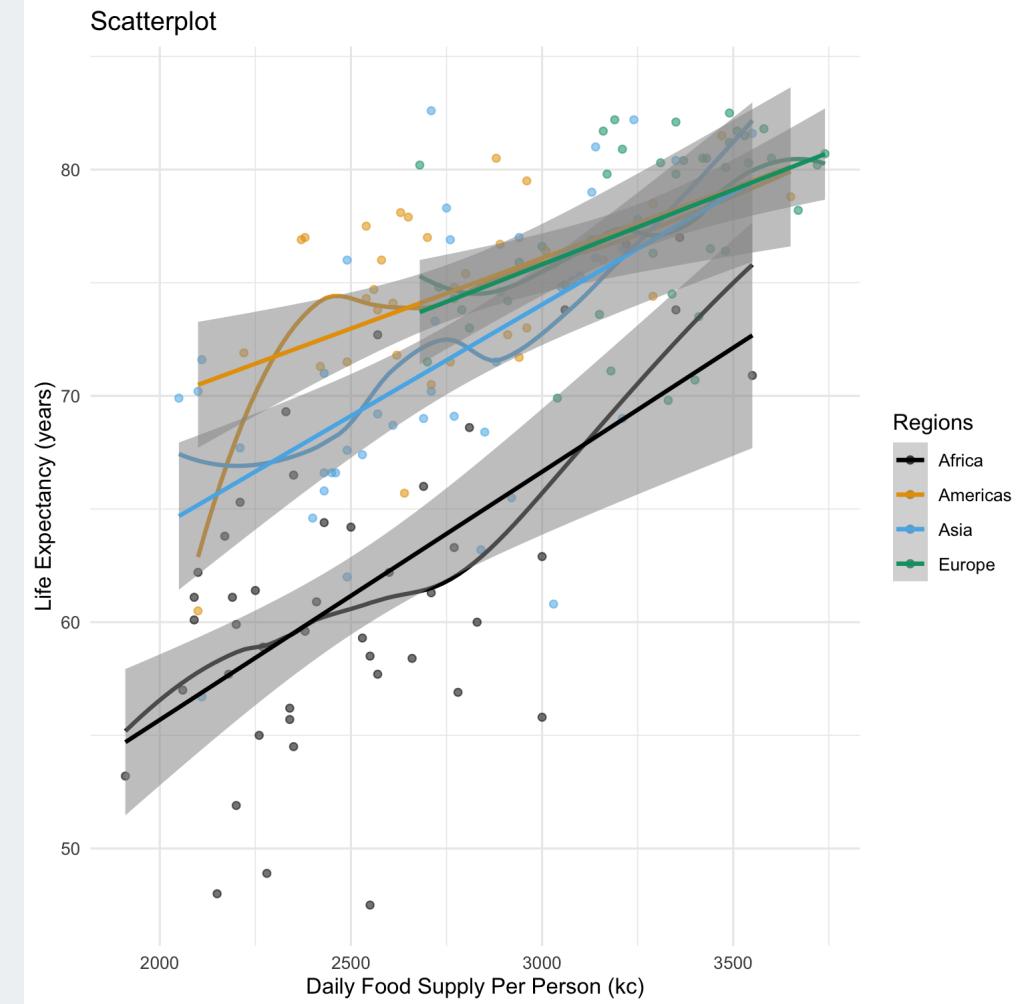
Exercise

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

```

ggplot(data = gapminder2011) +
  aes(x = FoodSupplykcPPD) +
  aes(y = LifeExpectancyYrs) +
  geom_point() +
  aes(color = four_regions) +
  aes(alpha=.4) +
  scale_color_colorblind(
    name = "Regions",
    labels = c("Africa", "Americas",
              "Asia", "Europe"))
  ) +
  scale_alpha(guide = "none") +
  geom_smooth(se = FALSE) +
  geom_smooth(method = lm) +
  theme_minimal() +
  labs(
    x = "Daily Food Supply Per Person (kc)",
    y = "Life Expectancy (years)",
    title = "Scatterplot"
  )

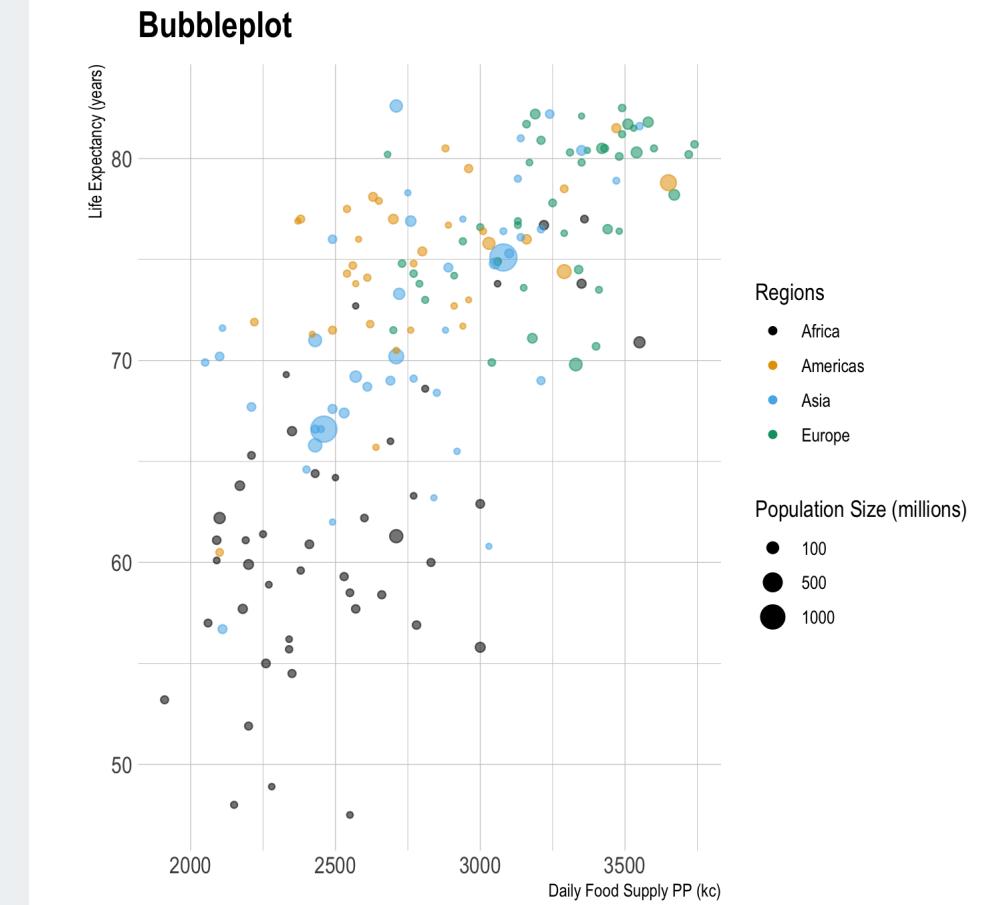
```



```

ggplot(data = gapminder2011) +
  aes(x = FoodSupplykcPPD) +
  aes(y = LifeExpectancyYrs) +
  geom_point() +
  aes(color = four_regions) +
  aes(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)
) +
  scale_alpha(guide = "none") +
  hrbrthemes::theme_ipsum() +
  labs(
    x = "Daily Food Supply PP (kc)",
    y = "Life Expectancy (years)",
    title = "Bubbleplot"
  )

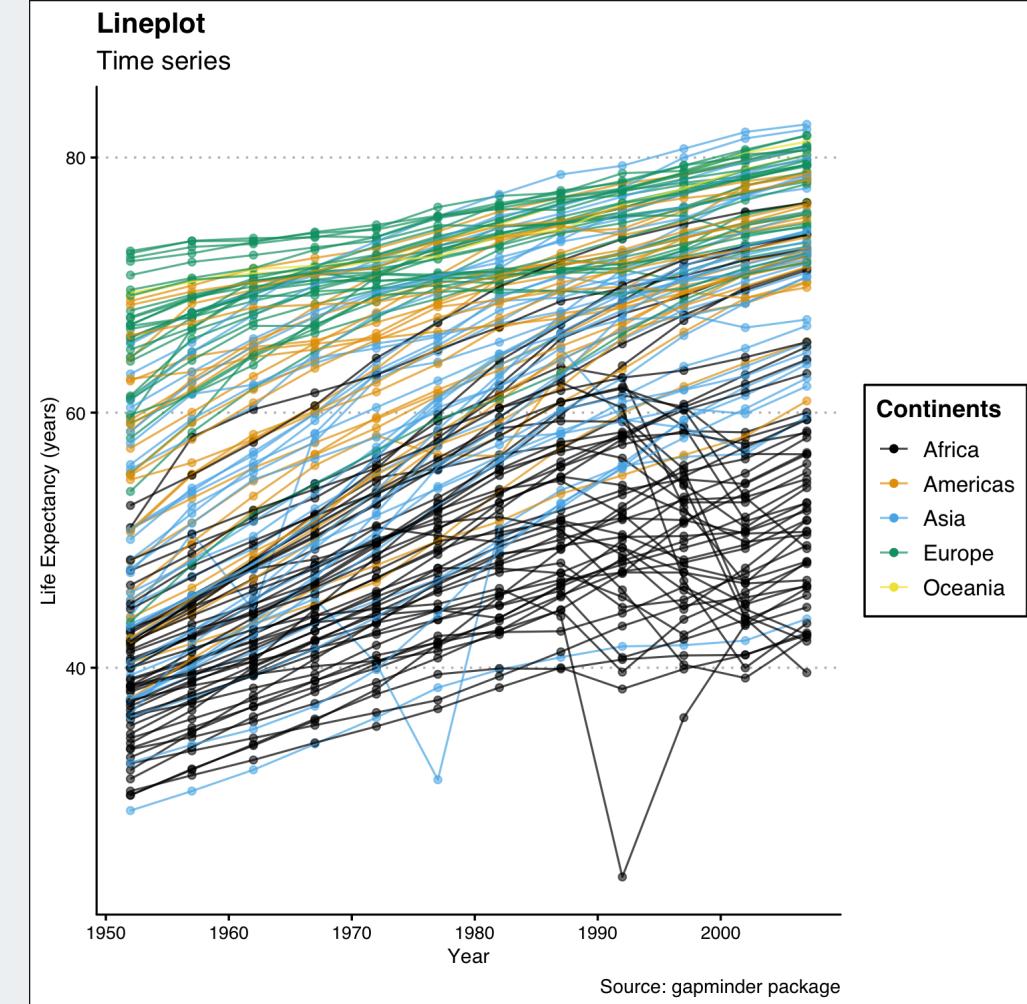
```



Exercise

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

```
ggplot(data = gapminder) +  
  aes(x = year) +  
  aes(y = lifeExp) +  
  geom_point() +  
  aes(color = continent) +  
  aes(alpha=.4) +  
  geom_line(alpha = .7) +  
  aes(group = country) +  
  scale_color_colorblind(  
    name = "Continents"  
  ) +  
  scale_alpha(guide = "none") +  
  ggthemes::theme_clean() +  
  labs(  
    x = "Year",  
    y = "Life Expectancy (years)",  
    title = "Lineplot",  
    subtitle = "Time series",  
    caption = "Source: gapminder package"  
  )
```



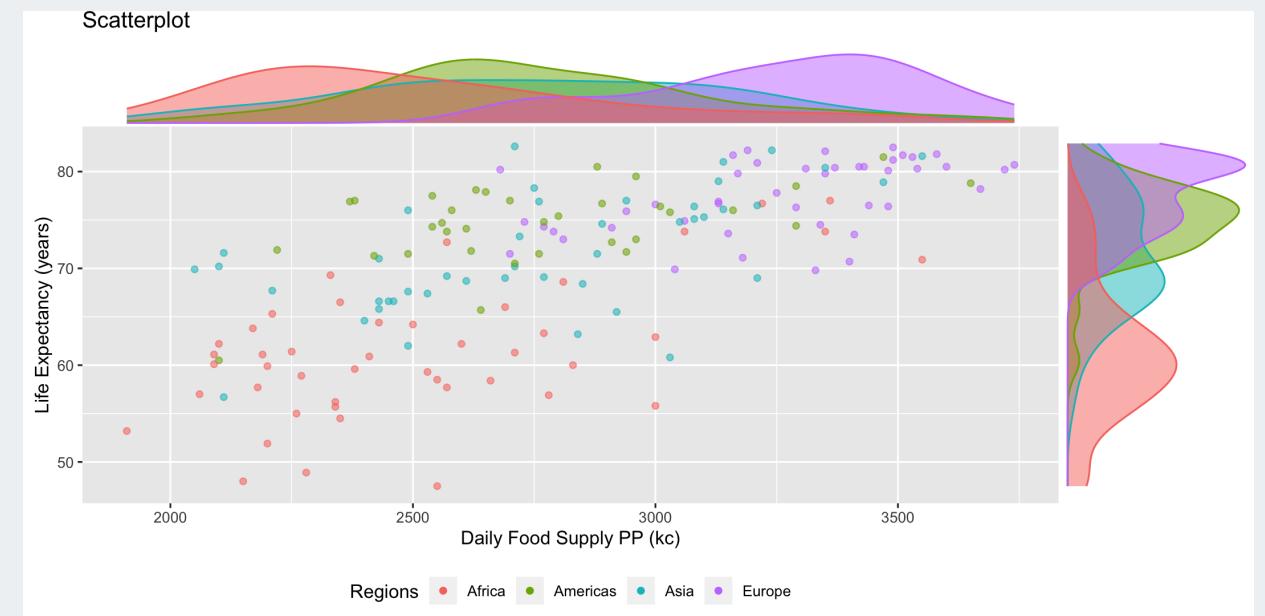
ggmarginal

<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,
                  alpha=.4)
            ) +
  geom_point() +
  scale_color_discrete(
    name = "Regions",
    labels = c("Africa", "Americas",
              "Asia", "Europe")
  ) +
  scale_alpha(guide = "none") +
  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
)
```



Corroleograms

Correlation matrix

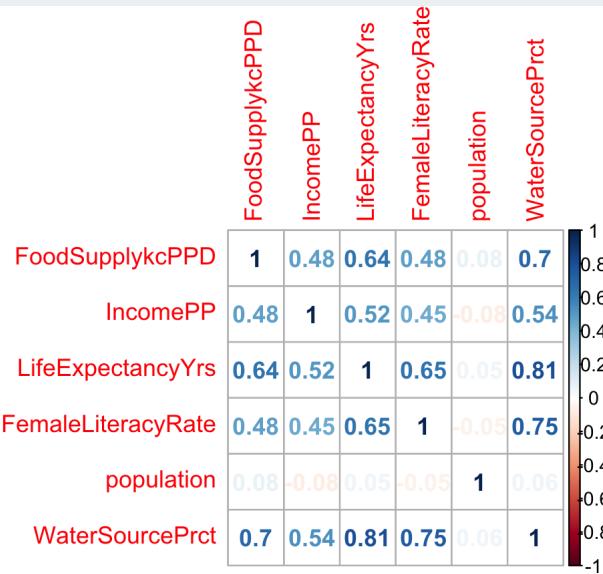
```
M <- cor(gapminder2011 %>%
  select(FoodSupplykcPPD:WaterSourcePrct),
  use = "complete.obs" # specified since there are missing values
)
M
```

	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

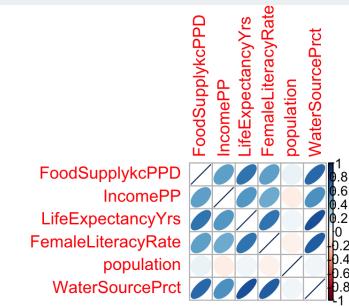
corrplot::corrplot()

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

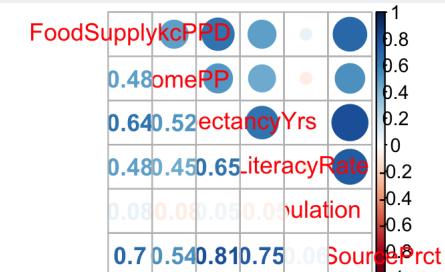
```
library(corrplot)
corrplot(M, method = "number")
```



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



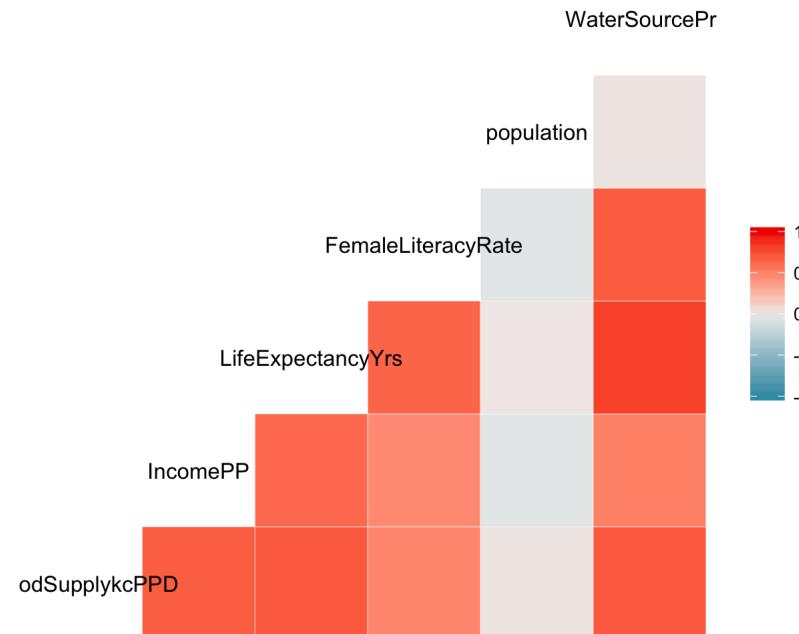
```
corrplot.mixed(M)
```



GGally::ggcorr()

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

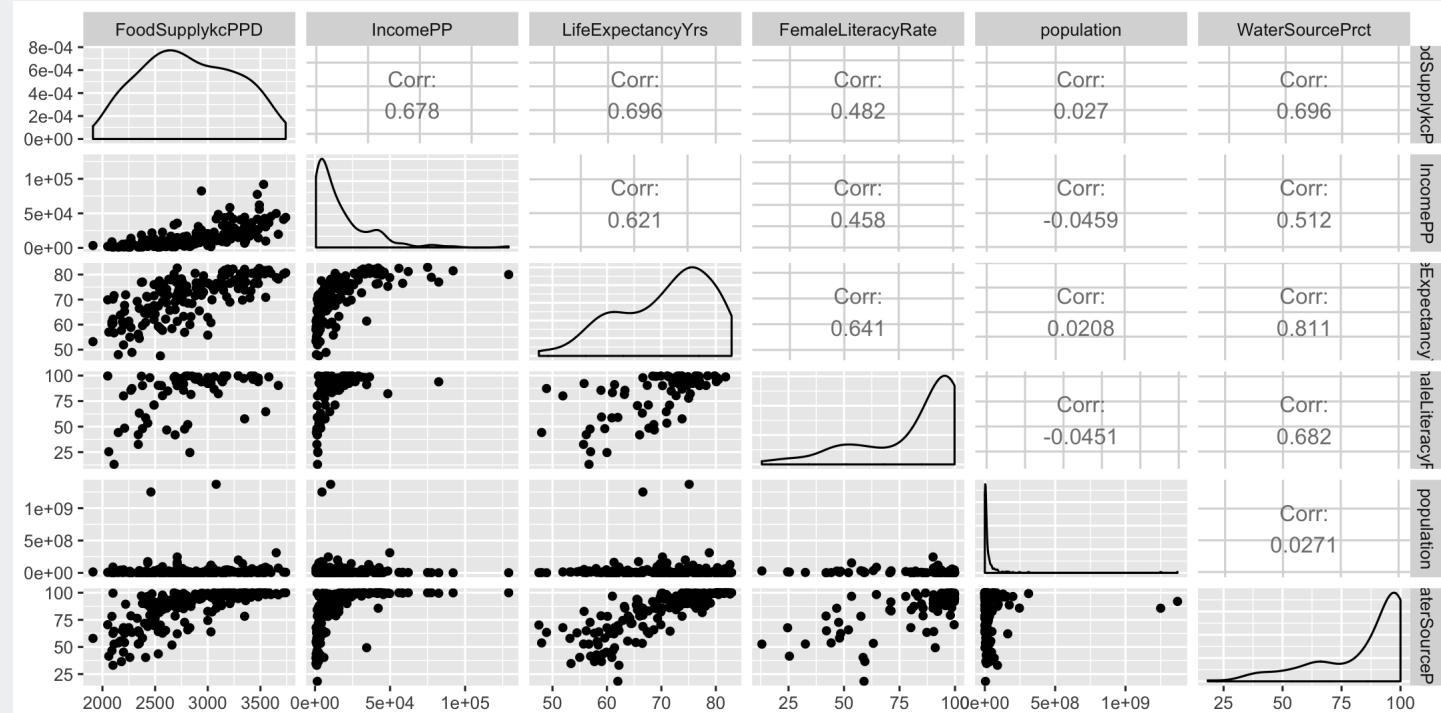
```
# library(GGally)
gapminder2011 %>%
  select(FoodSupplykcPPD:WaterSourcePrct) %>% # specifying which columns to use
  ggcorr()
```



GGally::ggpairs()

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

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



Faceting

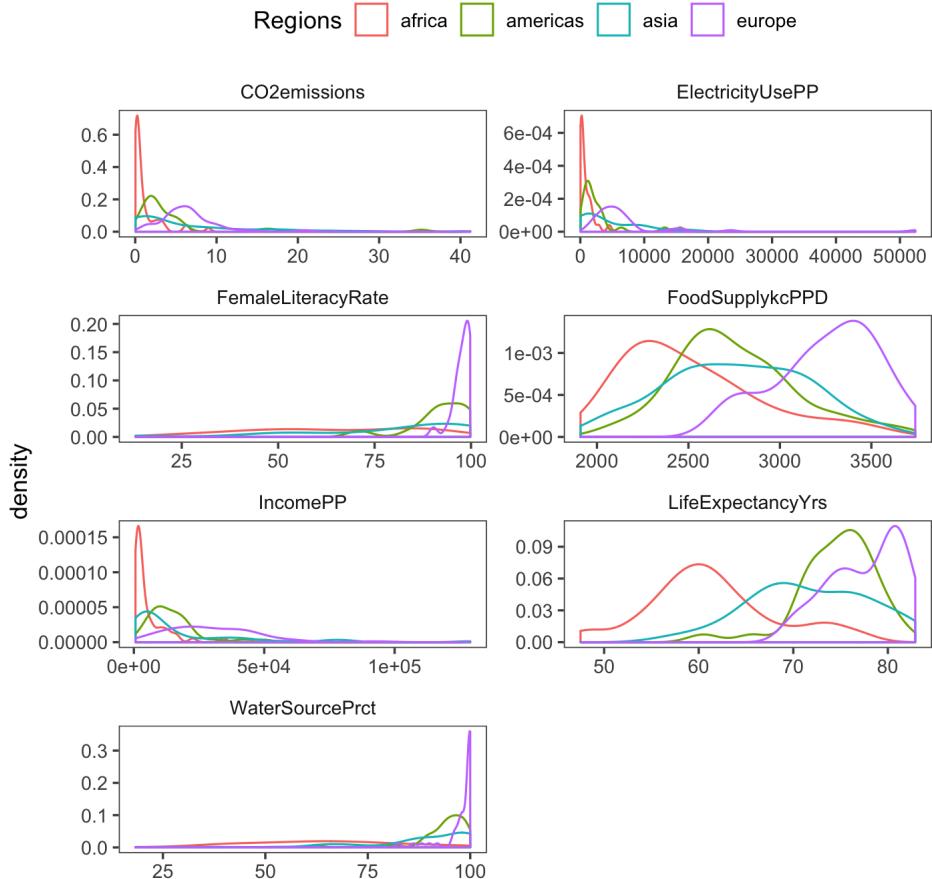
```

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

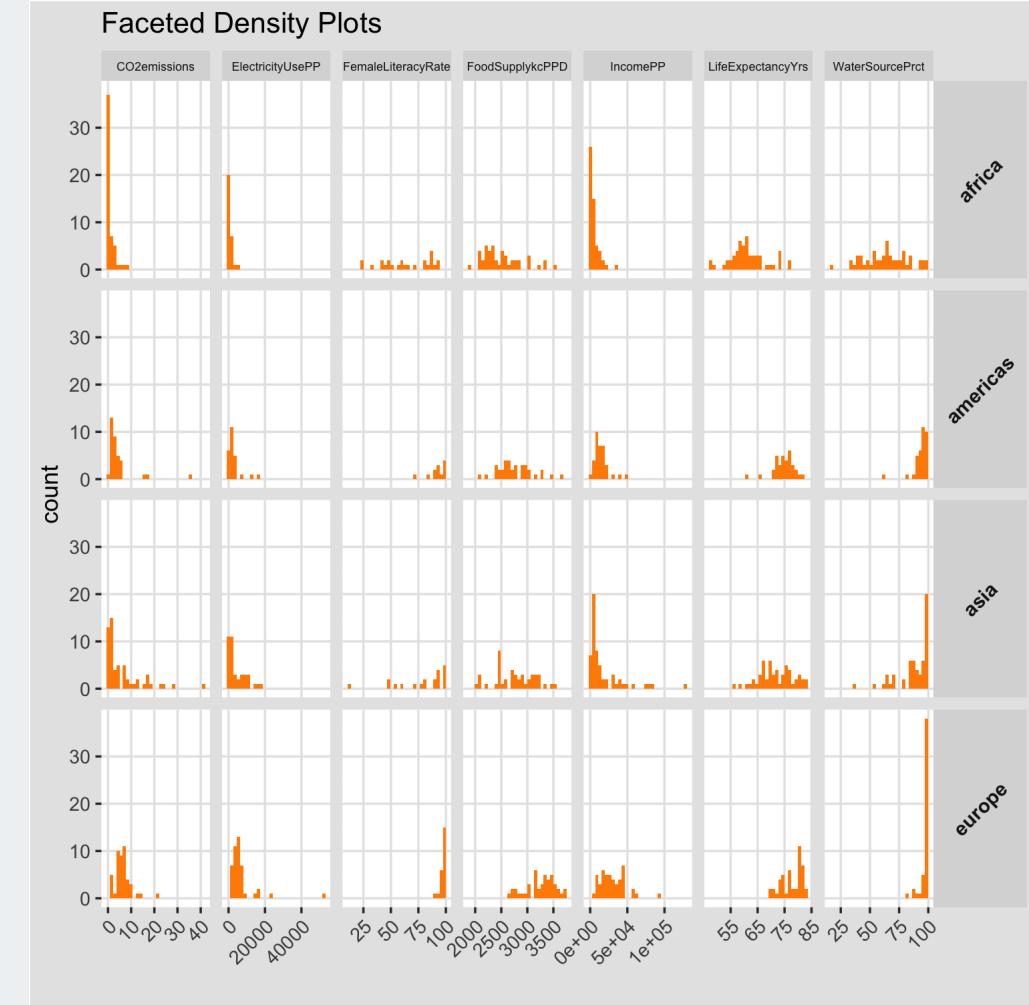
Faceted Density Plots



```

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)
```

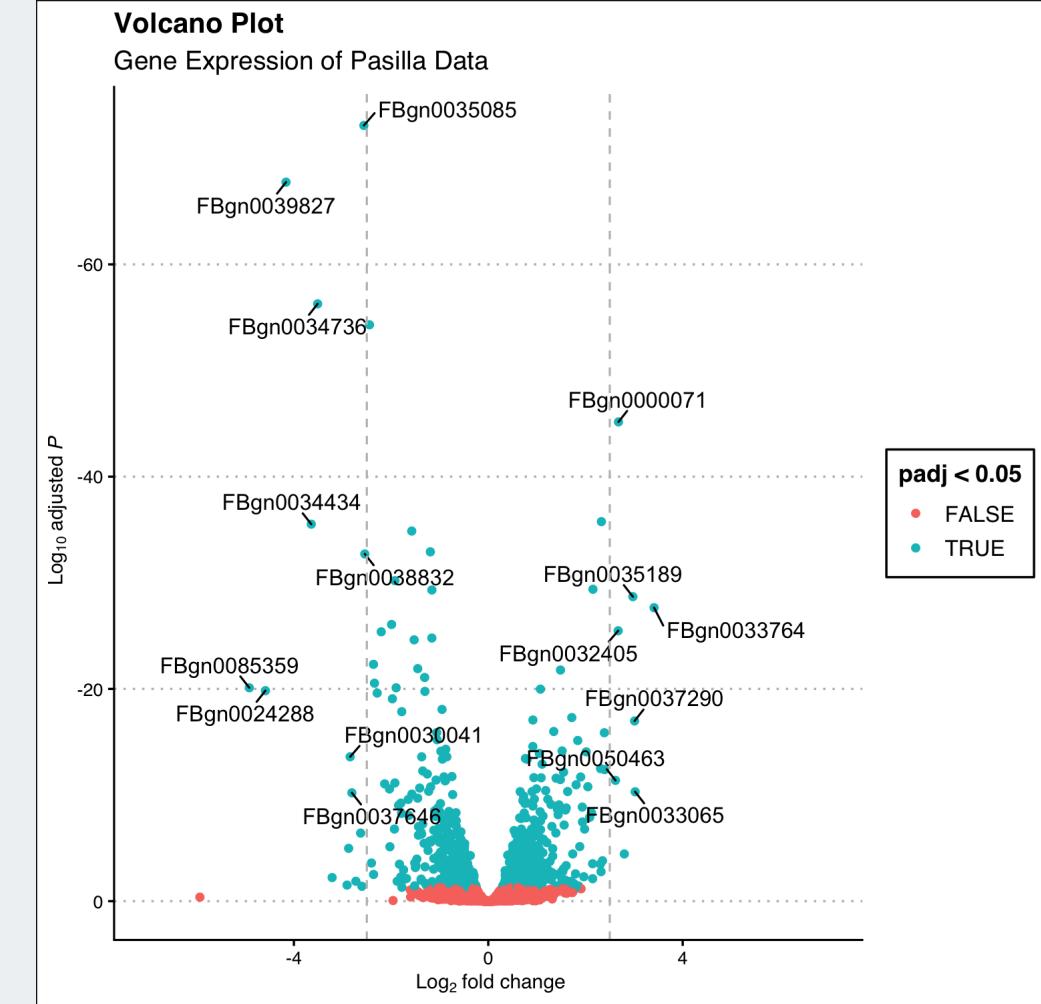
Observations: 8,377

Variables: 15

```

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 gene
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)
```

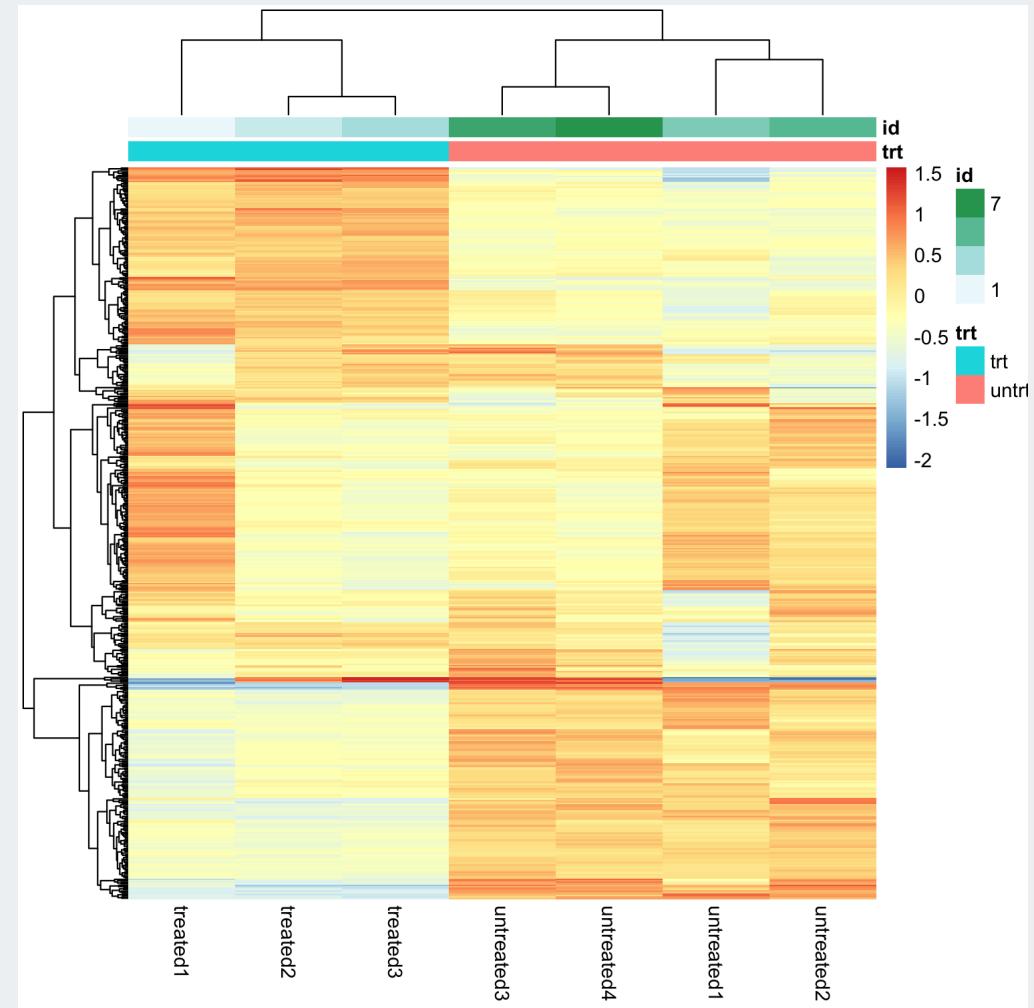
	treated1	treated2	treated3	unt
2390	-1.5997691	0.8713581	1.568570	-1
521	-1.3218267	0.9954861	1.278523	-1
7886	-0.5901012	0.8225366	1.339219	-1

```
pasilla_col
```

	trt	id
treated1	trt	1
treated2	trt	2
treated3	trt	3
untreated1	untrt	4
untreated2	untrt	5
untreated3	untrt	6
untreated4	untrt	7

Heatmap with pheatmap::pheatmap()

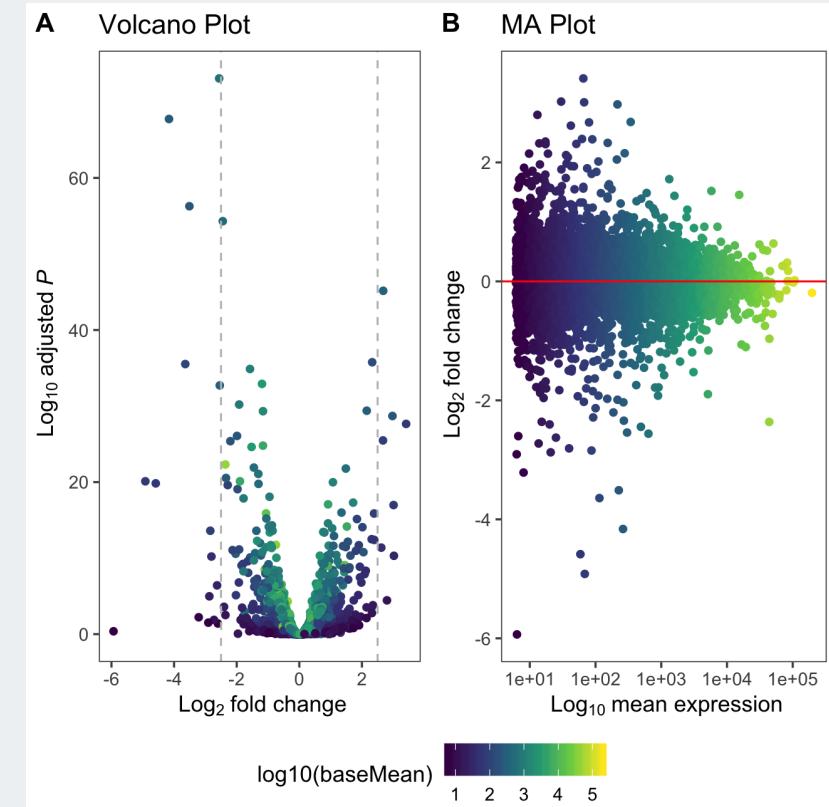
```
pheatmap::pheatmap(  
  mat = pasilla_heat,  
  show_rownames = FALSE,  
  annotation_col = pasilla_col  
)
```



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")
```

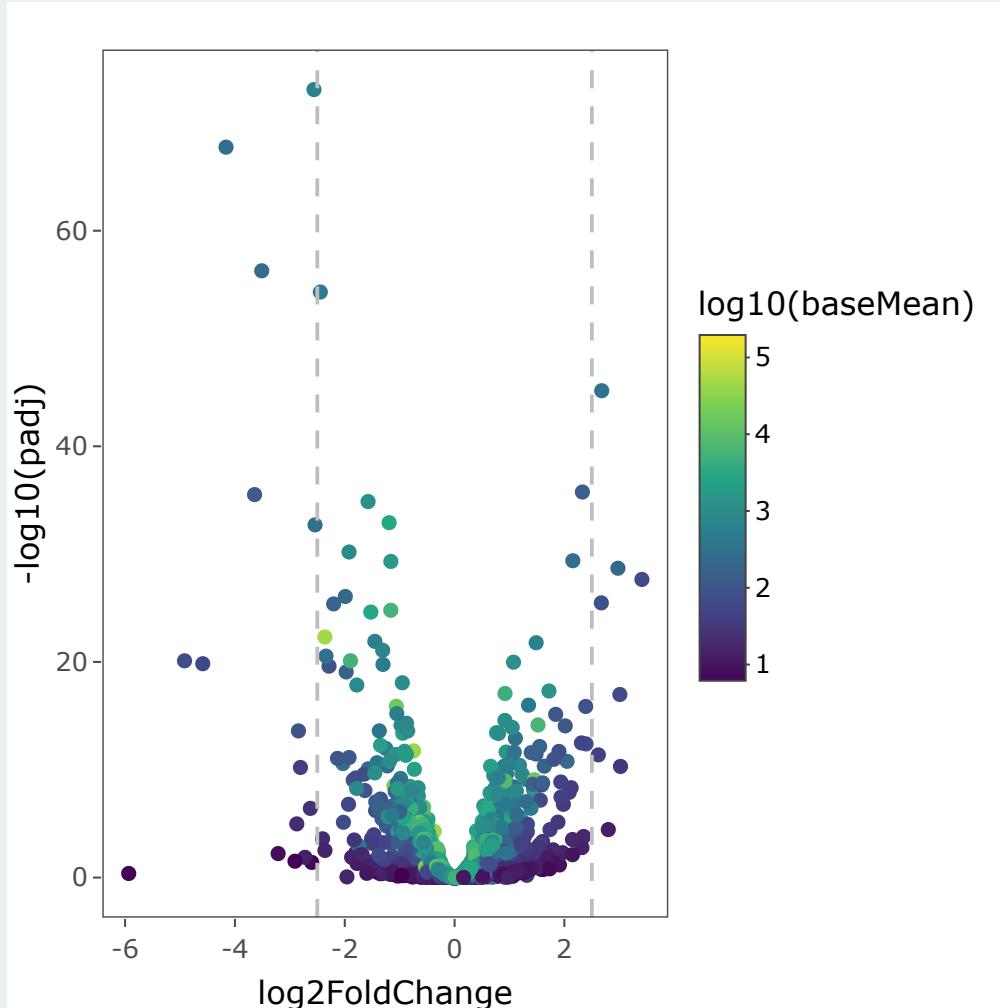


Other options: [cowplot](#) and [patchwork](#).

Interactive plotly graphs with ggplotly()

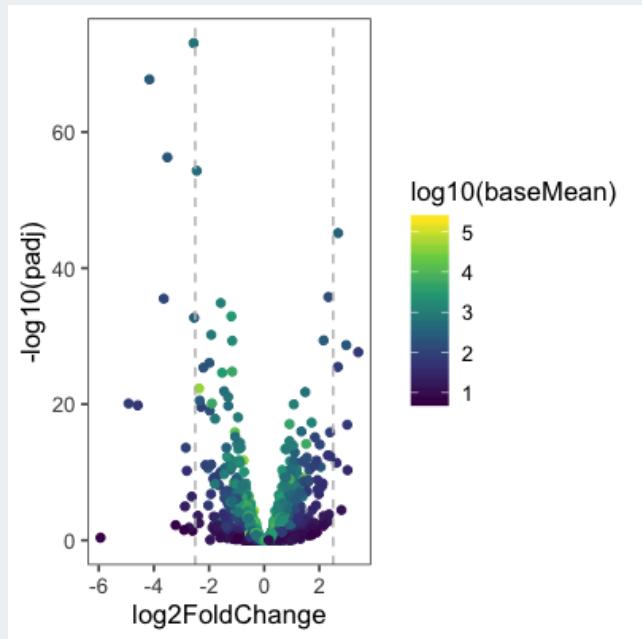
```
# Save ggplot
p1 <- ggplot(
  data = pasilla_data,
  aes(x = log2FoldChange,
      y = -log10(padj),
      color = log10(baseMean),
      key = gene)
) +
  geom_point() +
  geom_vline(
    xintercept = c(-2.5, 2.5),
    lty = 2, color="grey") +
  theme_few() +
  scale_color_viridis_c()
```

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

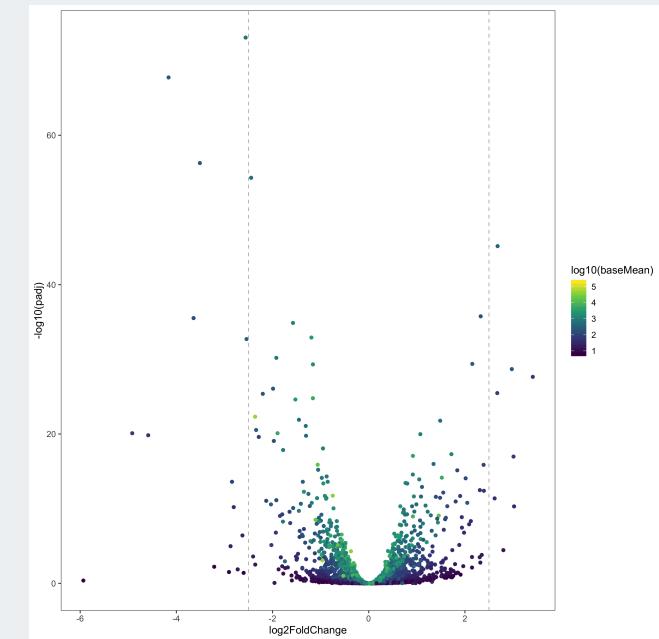


Saving plots

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

References and Links



from Data to Viz

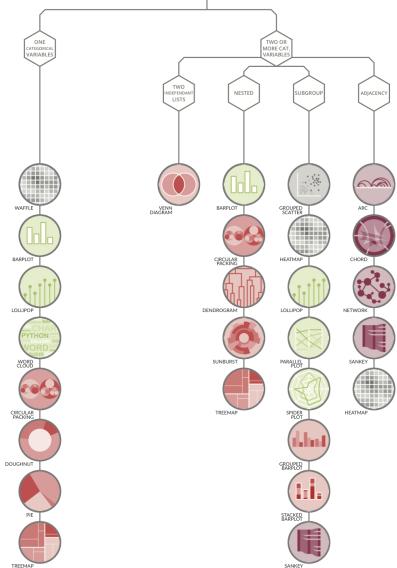
'From Data to Viz' is a classification of chart types based on input data format. It will help you find the perfect chart in three simple steps :

- 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.

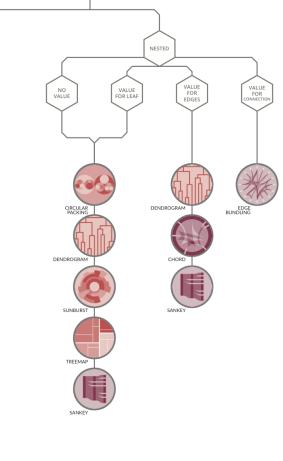
DataViz is a world with endless possibilities and this project does not claim to be exhaustive. However it should provide you with a good starting point. For an interactive version and much more, visit:

data-to-viz.com

CATEGORIC



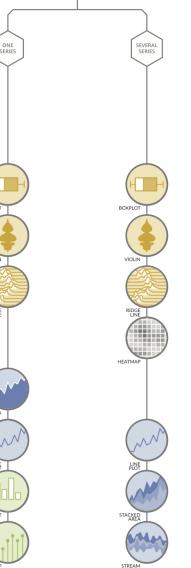
RELATIONAL



MAP



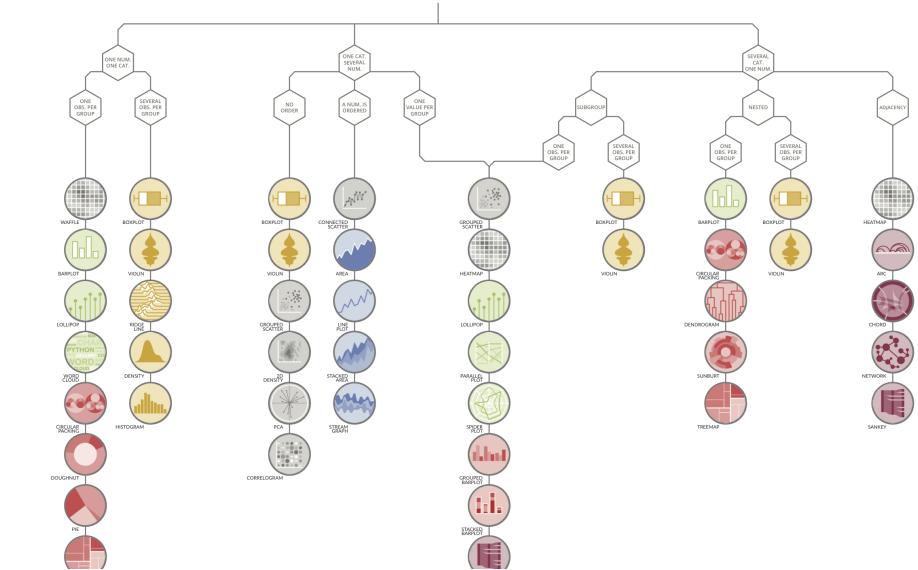
TIME SERIES



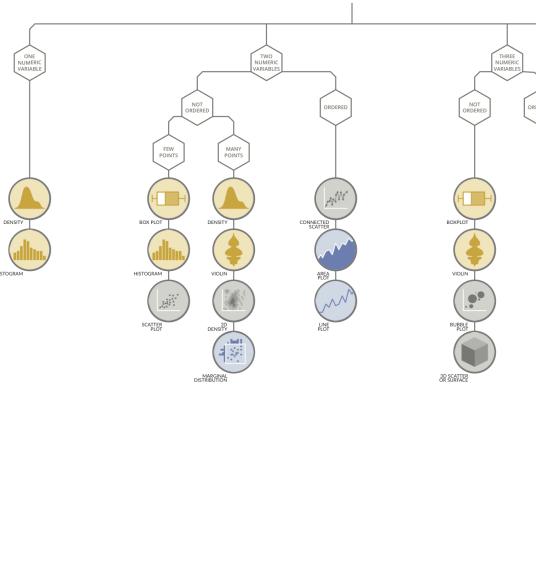
WHAT DO YOU WANT TO SHOW ?

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

CATEGORIC AND NUMERIC



NUMERIC

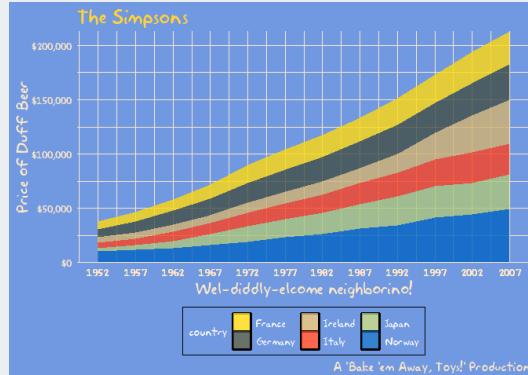


Many, many ggplot extensions!

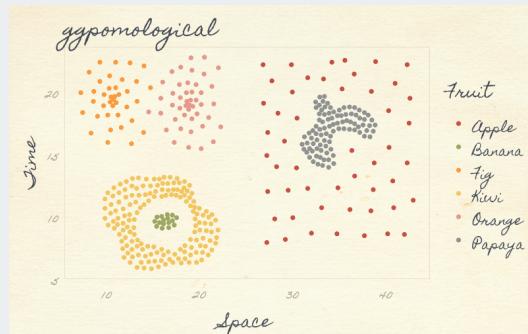
Some examples at the [ggplot2 extensions gallery](#)

Many, many themes and palettes/scales!

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

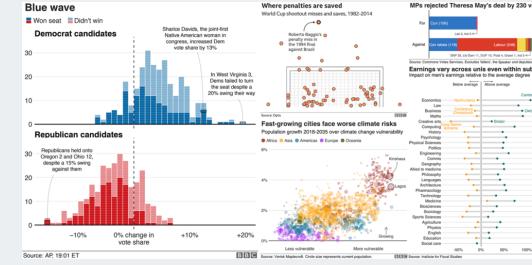


TV Themes

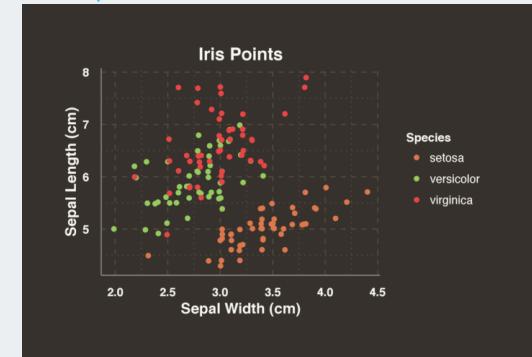


ggpomological

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



bbcplot for BBC themes

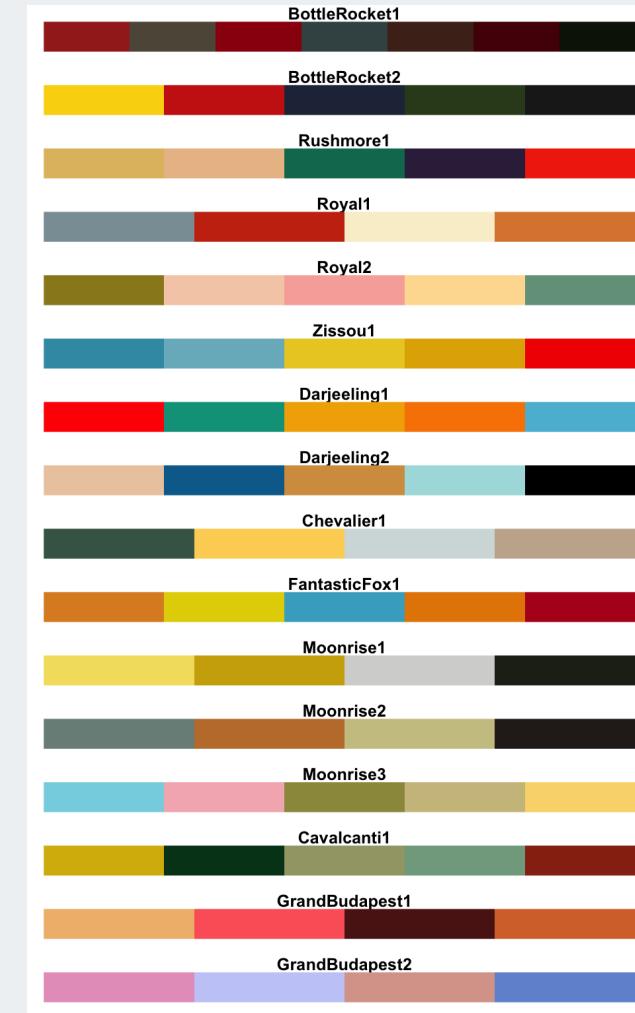


ggthemr

R colors and palettes



Built in R Colors



wesanderson package

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 `sols`/ folder.