

# Data Visualization Using R & ggplot2

---

Naupaka Zimmerman (@naupakaz) and Andrew Tredennick (@ATredennick)  
(and Karthik Ram (@\_inundata))

August 6, 2017

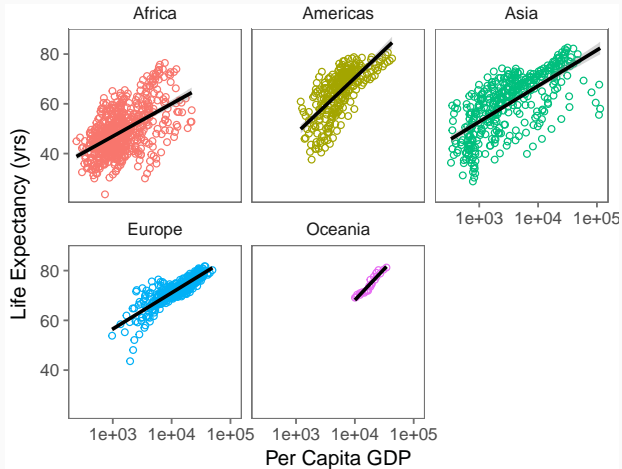
# Some housekeeping

Install some packages

```
install.packages("ggplot2", dependencies = TRUE)
install.packages("ggthemes")
install.packages("tidyr")
install.packages("dplyr")
```

# Why ggplot?

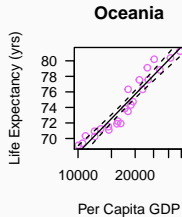
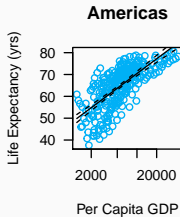
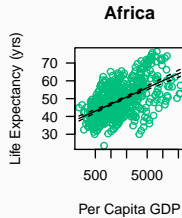
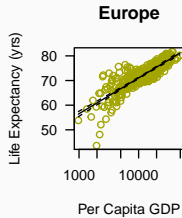
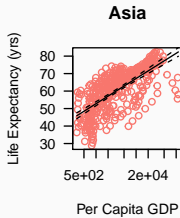
---



# 1 minute

```
library(ggplot2)
library(gapminder)
library(ggthemes)

ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(shape = 1, aes(color = continent)) +
  stat_smooth(method = "lm", size = 1, color = "black") +
  scale_x_log10() +
  xlab("Per Capita GDP") +
  ylab("Life Expectancy (yrs)") +
  facet_wrap(~continent) +
  theme_few() +
  guides(color = FALSE)
```



```
library(scales)
library(gapminder)
gapminder <- as.data.frame(gapminder)
conts <- unique(gapminder[, "continent"])
cols <- scales::hue_pal()(length(conts))
par(mfrow = c(2,3))
counter <- 1
for (i in conts) {
  plot(gapminder[which(gapminder$continent == i), "gdpPercap"],
       gapminder[which(gapminder$continent == i), "lifeExp"], col = cols[counter],
       xlab = "Per Capita GDP", ylab = "Life Expectancy (yrs)",
       main = i, las = 1, log = "x")
  fit <- lm(gapminder[which(gapminder$continent == i), "lifeExp"] ~
```

But wait, there's more...

```
log(gapminder[which(gapminder$continent == i), "gdpPercap"]))  
  pred <- predict(fit, interval = "confidence")  
  lines(sort(gapminder[which(gapminder$continent == i), "gdpPercap"]),  
        lines(sort(gapminder[which(gapminder$continent == i), "gdpPercap"]),  
        lines(sort(gapminder[which(gapminder$continent == i), "gdpPercap"]),  
        counter <- counter + 1  
}
```



# Why ggplot?

---

# Why ggplot?

- More elegant and compact code than with base graphics
- More aesthetically pleasing defaults than lattice
- Very powerful for exploratory data analysis

# Why ggplot?

- `gg` is for **grammar of graphics** (term by Lee Wilkinson)
- A set of terms that defines the basic components of a plot
- Used to produce figures using coherent, consistent syntax

# Why ggplot2?

- Supports a continuum of expertise
- Easy to get started, plenty of power for complex figures

# The Grammar

---

### Data

- Must be a data frame (`data.frame()`, `as.data.frame()`)
- Gets pulled into the `ggplot()` object

Helps your data play nice with ggplot

# Aesthetics

---



## aesthetics

- How your data are represented visually
- a.k.a. mapping
- which data on the x
- which data on the y
- but also: **color**, **size**, shape, transparency

# Geometries

---

### geometry

- The geometric objects in the plot
- points, lines, polygons, etc.
- functions: `geom_point()`, `geom_bar()`, `geom_line()`

# Basic structure

```
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point()
```

*# Equivalently...*

```
myplot <- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))  
myplot + geom_point()
```

- Specify the data and variables inside the `ggplot()` function.
- Anything else that goes in here becomes a *global* setting.
- Then add layers: geometric objects, statistical models, and facets.

# Don't be tempted!

- **NEVER** use `qplot()` - short for quick plot.
- You'll end up unlearning and relearning a good bit.

## Exercise 1

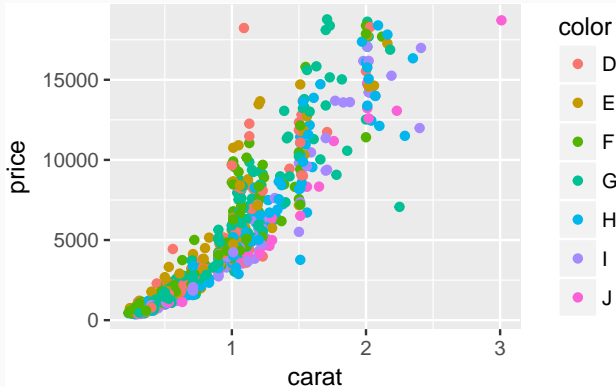
---

# Exercise 1

*# Make a small sample of the diamonds dataset*

```
d2 <- diamonds[sample(x = 1:nrow(diamonds), size = 1000), ]
```

The make this plot:



# Stats

---



## stats

- Statistical transformations and data summary
- All geoms have associated default stats, and vice versa
- e.g. binning (the stat) for a histogram or fitting a linear model

# Facets

---

## facets

- Subsetting data to make lattice plots
- Really powerful
- I use in almost every publication

# Scales

---

## scales

- Control the *mapping* from data to aesthetics
- Often used for adjusting color mapping (i.e., setting colors manually)

## Refer to a color chart for beautiful visualizations

<http://tools.medialab.sciences-po.fr/iwanthue/>

## Some common scales

```
scale_fill_discrete(); scale_colour_discrete()  
scale_fill_hue(); scale_color_hue()  
scale_fill_manual(); scale_color_manual()  
scale_fill_brewer(); scale_color_brewer()  
scale_linetype(); scale_shape_manual()
```

# Coordinates

---

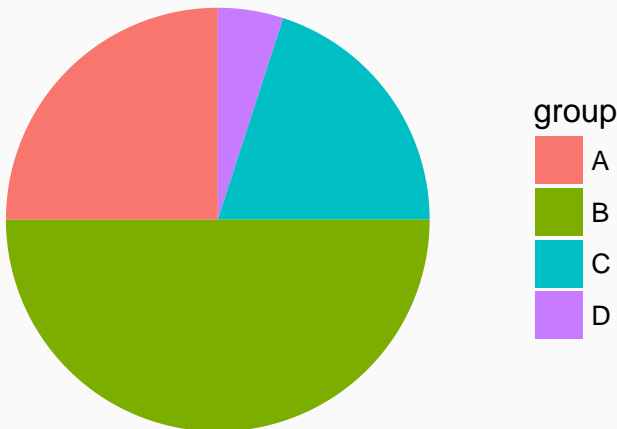


## coordinates

- Not going to cover this in detail
- e.g. polar coordinate plots

# Don't try this at home... please!

```
ggplot(mydata, aes(x = "", y = proportion, fill = group)) +  
  geom_bar(stat = "identity", width = 1) +  
  coord_polar("y", start=0) +  
  theme_void()
```



## Putting it all together with more examples

---

# Histograms

---

## Line plots

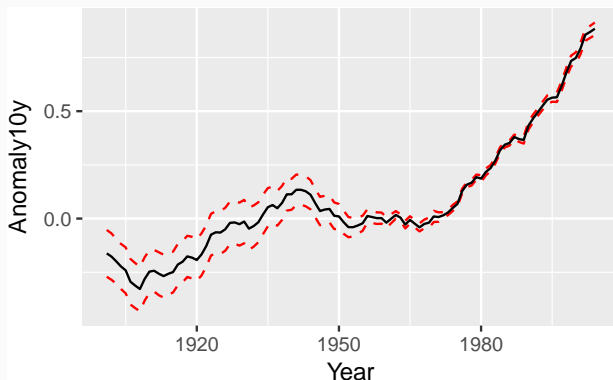
---

## Exercise 2

---

## Exercise 2

- Modify the last plot and change the ribbon to three separate lines, like below:



## Bar plots

---

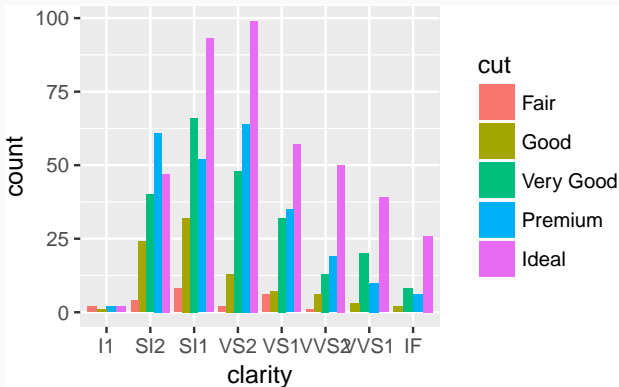


## Exercises 3, 4, & 5

---

## Exercise 3

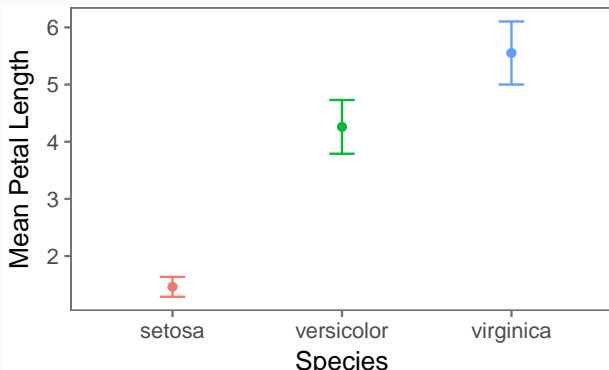
Using the d2 dataset you created earlier, make this plot:



## Exercise 4

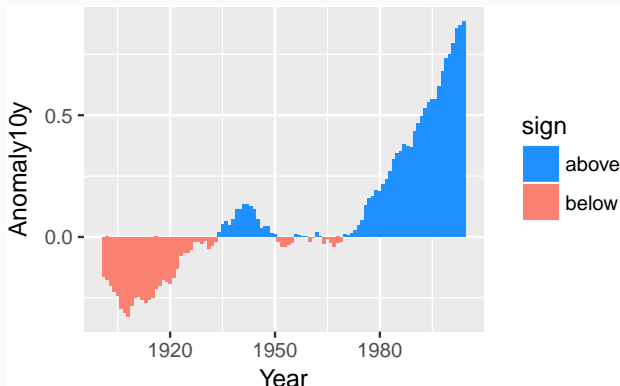
Use dplyr to summarize the iris data, then make the plot below.

```
iris_agg <- iris %>%  
  group_by(Species) %>%  
  summarise(mean_petal_len = mean(Petal.Length),  
            sd_petal_len = sd(Petal.Length))  
# ?geom_errorbar()
```



## Exercise 5

- Using the `climate` dataset, create a new variable called `sign`. Make it categorical (above/below) based on the sign of `Anomaly10y`.
- Plot a bar plot and use `sign` variable as the fill.



## Density plots

---

## Adding smoothers

---

# Themes!

---

- Everything can be customized using `theme()` settings
- <http://ggplot2.tidyverse.org/reference/theme.html>

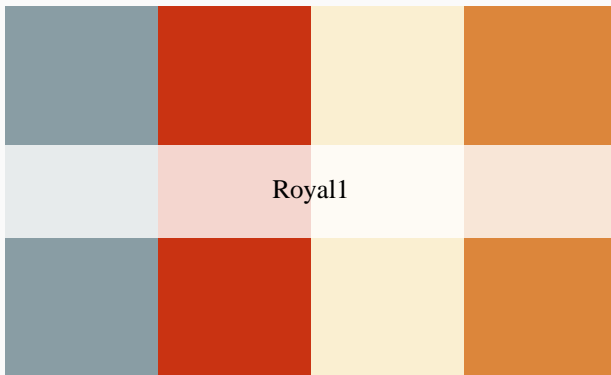


# What about Wes Anderson?



# There's a theme for that!

```
# install.packages('wesanderson')  
library("wesanderson")  
# display a palette  
wes_palette("Roya11")
```



# Save your beautiful plot

---

- If the plot is on the screen

```
ggsave('~ /path/to/figure/filename.png')
```

- If your plot is assigned to a named object

```
ggsave(plot1, '~ /path/to/figure/filename.png')
```

- Specify size

```
ggsave(file = '~ /path/figure.png', width = 4, height = 3, units = "in")
```

- And format

```
ggsave(file = "/path/to/figure/filename.eps")
```

```
ggsave(file = "/path/to/figure/filename.jpg")
```

```
ggsave(file = "/path/to/figure/filename.pdf")
```

## Further help

- We've only scratched the surface
- Practice
- Read: <http://ggplot2.tidyverse.org/index.html>
- Work together

