

Data Visualization Using R & ggplot2

Naupaka Zimmerman (@naupakaz)
Andrew Tredennick (@ATredennick)

Hat tip to Karthik Ram (@_inundata) for original slides

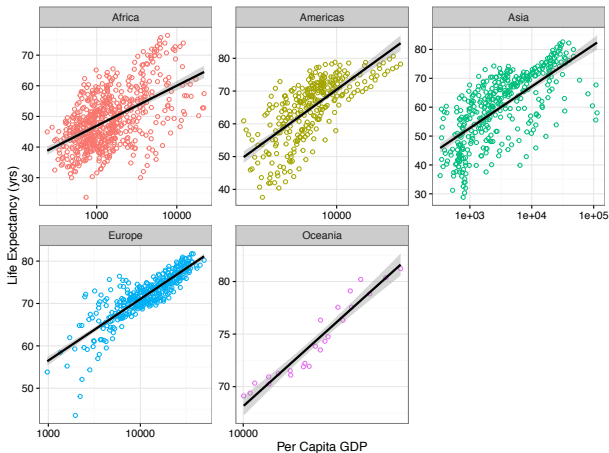
August 6, 2016

Some housekeeping

Install some packages

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

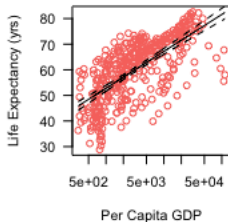
1 minute



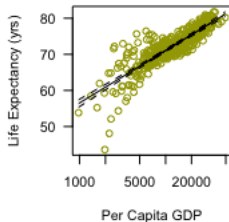
```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(shape = 1, aes(color = continent)) +  
  stat_smooth(method = "lm", size = 1, color = "black") +  
  scale_x_log10() +  
  xlab("Per Capita GDP") +  
  ylab("Life Expectancy (yrs)") +  
  facet_wrap(~continent, scales = "free") +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

30 minutes

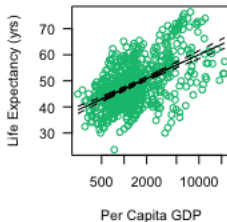
Asia



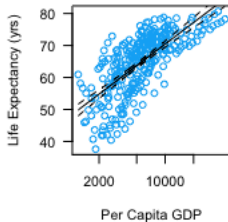
Europe



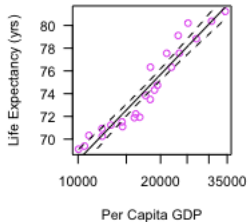
Africa



Americas



Oceania



```
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"]),  
  sort(pred[,1]))  
lines(sort(gapminder[which(gapminder$continent == i),  
  "gdpPercap"]),  
  sort(pred[,2]), lty = 2)  
lines(sort(gapminder[which(gapminder$continent == i),  
  "gdpPercap"]),  
  sort(pred[,3]), lty = 2)  
counter <- counter + 1  
}
```

Section 1

Why ggplot2?

Why ggplot2?

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

Why ggplot2?

- ▶ '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

Section 2

The Grammar

Some terminology

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

The iris dataset

```
head(iris)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 6	5.4	3.9	1.7	0.4	setosa

tidyr

Help your data play nice with ggplot.

tidyr

```
iris[1:2, ]
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1           3.5           1.4           0.2  setosa
## 2           4.9           3.0           1.4           0.2  setosa
```

```
library("tidyr")
```

```
df <- gather(iris, key = flower_attribute, value = measurement,
             -Species)
```

```
df[1:2, ]
```

```
##      Species flower_attribute measurement
## 1  setosa      Sepal.Length           5.1
## 2  setosa      Sepal.Length           4.9
```


Section 3

Aesthetics

Some terminology

- ▶ **data**
- ▶ **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

Let's try an example

```
myplot <- ggplot(data = iris, aes(x = Sepal.Length,  
                                   y = Sepal.Width))  
summary(myplot)  
  
## data: Sepal.Length, Sepal.Width, Petal.Length,  
##       Petal.Width, Species [150x5]  
## mapping: x = Sepal.Length, y = Sepal.Width  
## faceting: facet_null()
```

Section 4

Geoms

Some terminology

- ▶ data
 - ▶ aesthetics
 - ▶ **geometry**
- ▶ **The geometric objects in the plot**
 - ▶ points, lines, polygons, etc
 - ▶ shortcut functions: `geom_point()`,
`geom_bar()`, `geom_line()`

Basic structure

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

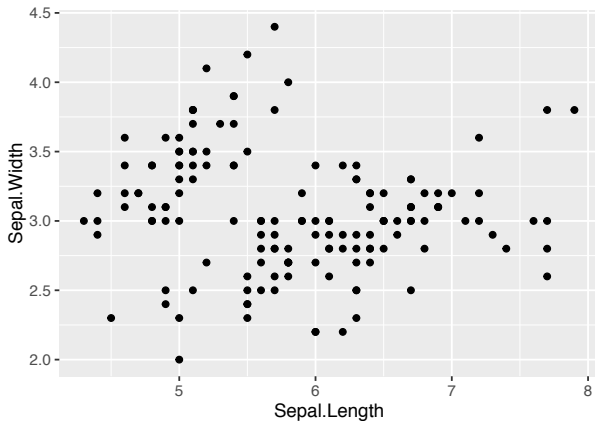
- ▶ Specify the data and variables inside the `ggplot` function.
- ▶ Anything else that goes in here becomes a global setting.
- ▶ Then add layers: geometric objects, statistical models, and facets.

Quick note

- ▶ Never use `qplot` - short for quick plot.
- ▶ You'll end up unlearning and relearning a good bit.

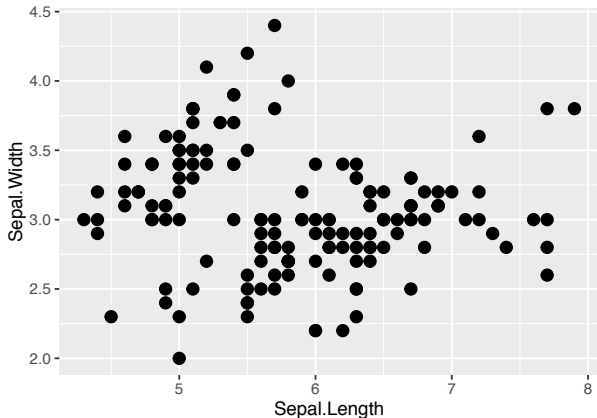
Let's try an example

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



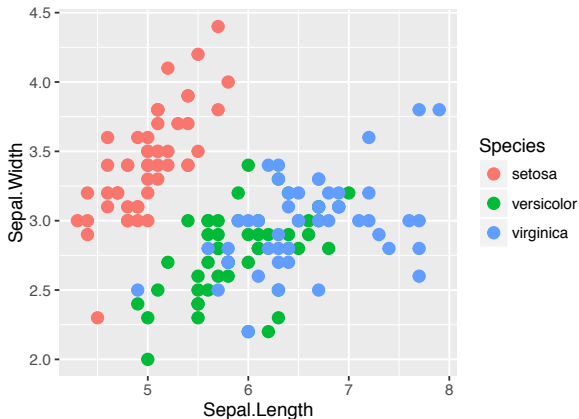
Changing the aesthetics of a geom: Increase the size of points

```
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point(size = 3)
```



Changing the aesthetics of a geom: Add some color

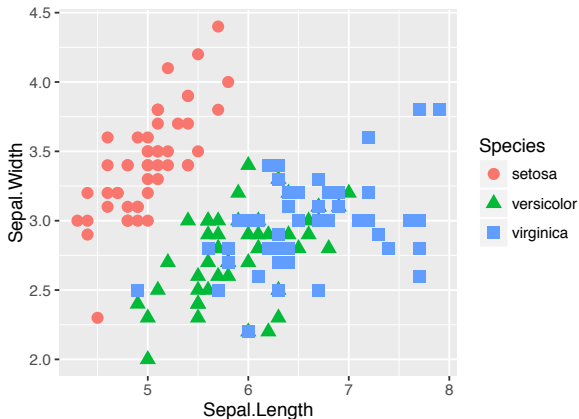
```
ggplot(iris, aes(Sepal.Length, Sepal.Width,  
                 color = Species)) +  
  geom_point(size = 3)
```



Changing the aesthetics of a geom:

Differentiate points by shape

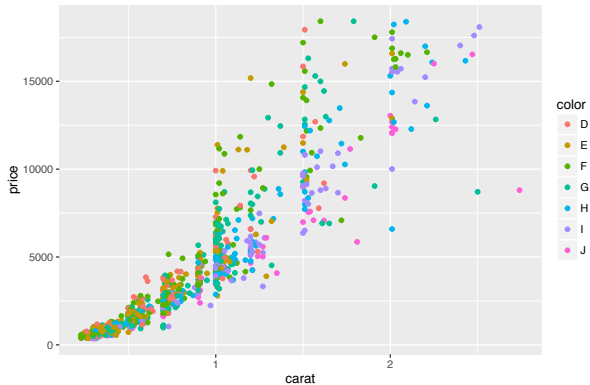
```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point(aes(shape = Species), size = 3)  
# Why aes(shape = Species)?
```



Exercise 1

```
# Make a small sample of the diamonds dataset  
d2 <- diamonds[sample(1:dim(diamonds)[1], 1000), ]
```

Then generate this plot below.



Section 5

Stats

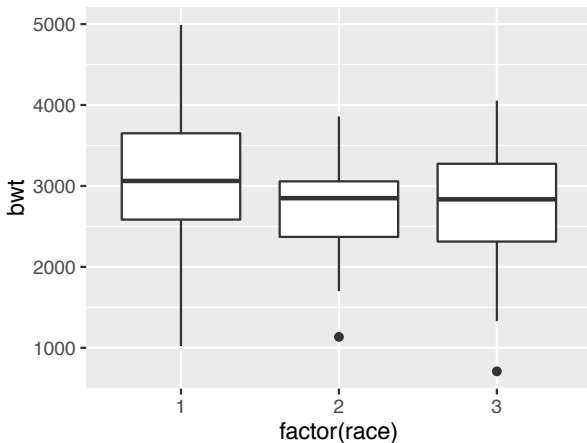
Some terminology

- ▶ data
 - ▶ aesthetics
 - ▶ geometry
 - ▶ stats
- ▶ **Statistical transformations and data summary**
 - ▶ All geoms have associated default stats, and vice versa
 - ▶ e.g. binning for a histogram or fitting a linear model

Built-in stat example: Boxplots

See `?geom_boxplot` for list of options

```
library(MASS)
ggplot(birthwt, aes(factor(race), bwt)) + geom_boxplot()
```



Built-in stat example: Boxplots

```
myplot <- ggplot(birthwt, aes(factor(race), bwt)) +  
  geom_boxplot()  
summary(myplot)  
  
## data: low, age, lwt, race, smoke, ptl, ht, ui, ftv,  
##    bwt [189x10]  
## mapping:  x = factor(race), y = bwt  
## faceting: facet_null()  
## -----  
## geom_boxplot: outlier.colour = NULL, outlier.shape = 19, outlier.size  
## stat_boxplot: na.rm = FALSE  
## position_dodge
```


Section 6

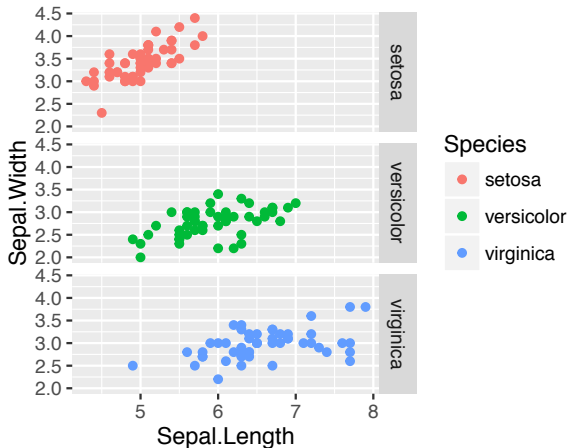
Facets

Some terminology

- ▶ data
 - ▶ aesthetics
 - ▶ geometry
 - ▶ stats
 - ▶ facets
- ▶ **Subsetting data to make lattice plots**
 - ▶ Really powerful

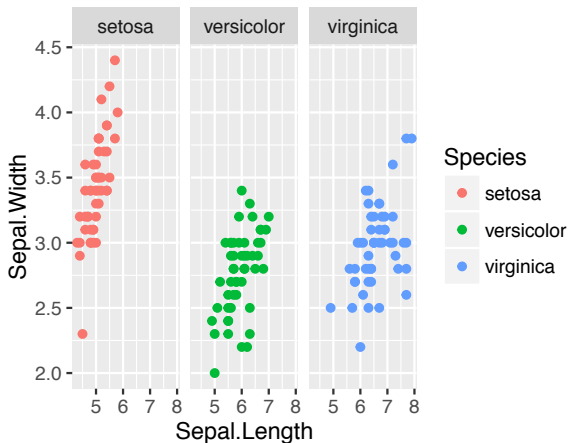
Faceting: single column, multiple rows

```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point() +  
  facet_grid(Species ~ .)
```



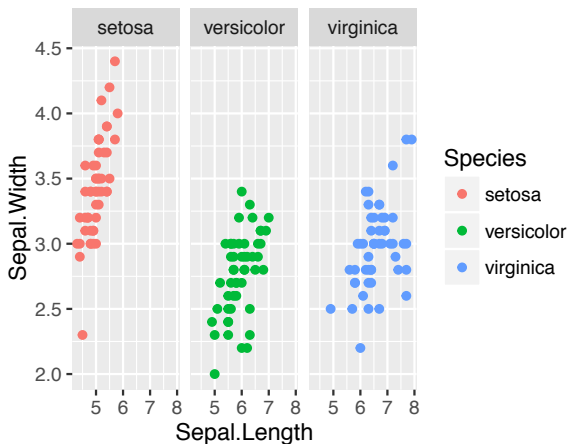
Faceting: single row, multiple columns

```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point() +  
  facet_grid(. ~ Species)
```



or just wrap your facets

```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point() +  
  facet_wrap(~ Species) # notice lack of .
```



Section 7

Scales

Some terminology

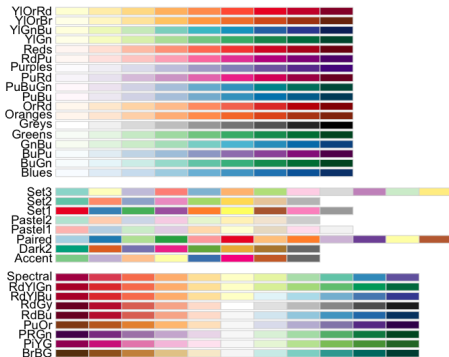
- ▶ data
 - ▶ aesthetics
 - ▶ geometry
 - ▶ stats
 - ▶ facets
 - ▶ scales
- ▶ **Control the mapping from data to aesthetics**
 - ▶ Often used for adjusting color mapping

Colors

```
aes(color = variable) # mapping  
color = "black" # setting  
  
# Or add it as a scale  
scale_fill_manual(values = c("color1", "color2"))
```


The RColorBrewer package

```
library("RColorBrewer")  
display.brewer.all()
```



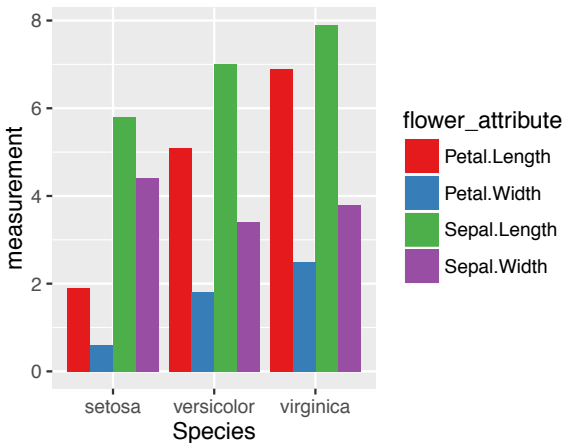
Using a color brewer palette

```
library("RColorBrewer")
df <- gather(iris, key = flower_attribute,
              value = measurement, -Species)
df[1:2,]
```

```
##   Species flower_attribute measurement
## 1  setosa      Sepal.Length          5.1
## 2  setosa      Sepal.Length          4.9
```

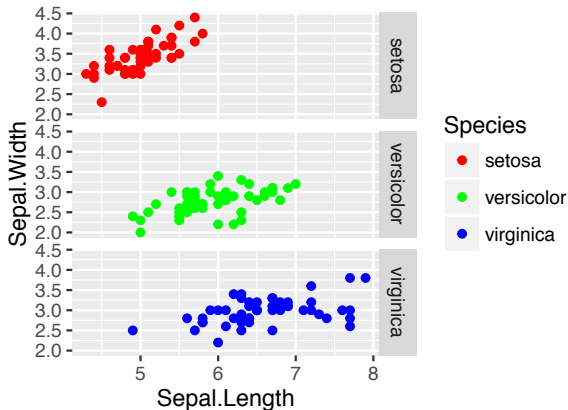
Using a color brewer palette

```
ggplot(df, aes(Species, measurement, fill = flower_attribute)) +  
  geom_bar(stat = "identity", position = "dodge") +  
  scale_fill_brewer(palette = "Set1")
```



Manual color scale


```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point() +  
  facet_grid(Species ~ .) +  
  scale_color_manual(values = c("red", "green", "blue"))
```



Refer to a color chart for beautiful visualizations

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

[i want hue](#) [Tutorials](#) [Examples](#) [Theory](#) [Experiment](#) [Old version](#) [GitHub](#) [Issues](#) [+ Medialab Tools](#)

 i want hue

Colors for data scientists. Generate and refine palettes of optimally distinct colors.

Color space

Intense

H

0

360

C

0.6


1

L

0.2

1.1

☐ Dark background




Palette

5 colors

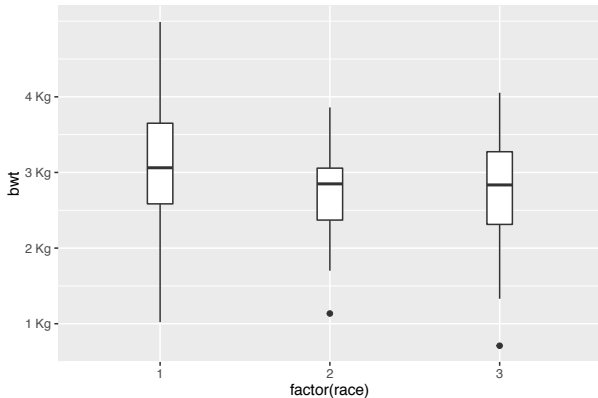
soft (k-Means)

Reset palette



Adding a continuous scale to an axis

```
library(MASS)
ggplot(birthwgt, aes(factor(race), bwt)) +
  geom_boxplot(width = .2) +
  scale_y_continuous(labels = (paste0(1:4, " Kg")),
    breaks = seq(1000, 4000, by = 1000))
```



Commonly used 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()
```

Section 8

Coordinates

Some terminology

- ▶ data
 - ▶ aesthetics
 - ▶ geometry
 - ▶ stats
 - ▶ facets
 - ▶ scales
 - ▶ **coordinates**
- ▶ Not going to cover this in detail
 - ▶ e.g. polar coordinate plots

Section 9

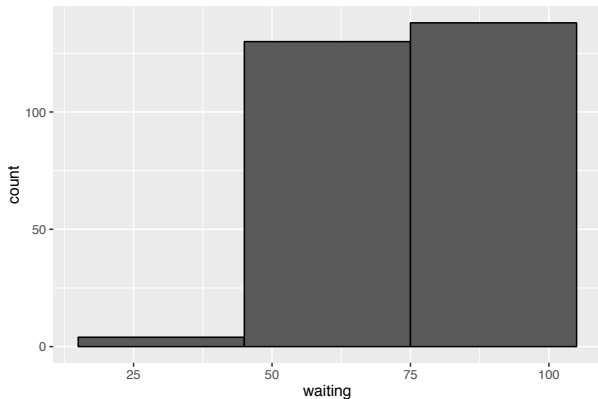
Putting it all together with more examples

Section 10

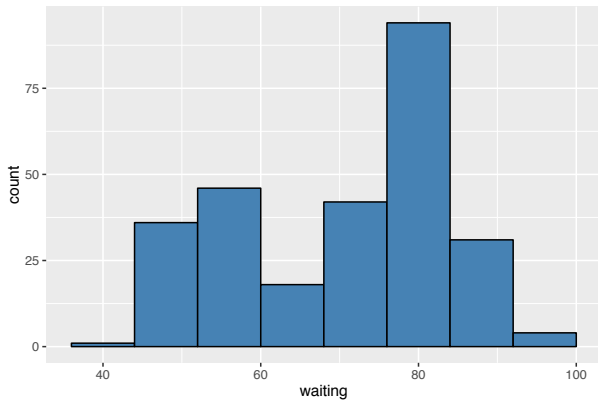
Histograms

See `?geom_histogram` for list of options

```
h <- ggplot(faithful, aes(x = waiting))  
h + geom_histogram(binwidth = 30, colour = "black")
```



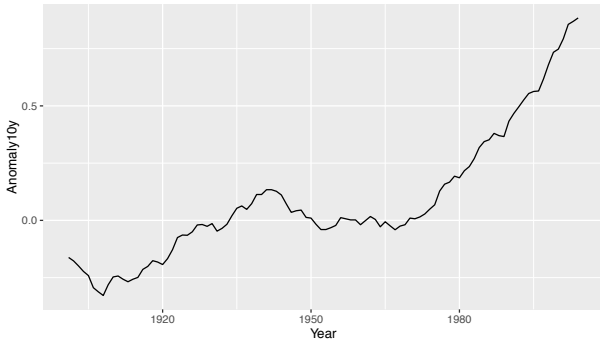
```
h <- ggplot(faithful, aes(x = waiting))  
h + geom_histogram(binwidth = 8, fill = "steelblue",  
colour = "black")
```



Section 11

Line plots

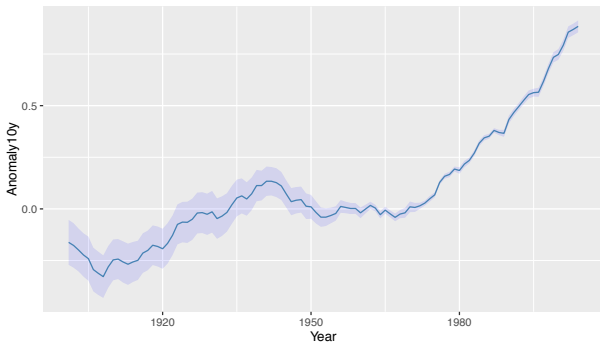
```
climate <- read.csv("../data/climate.csv", header = TRUE)
ggplot(climate, aes(Year, Anomaly10y)) +
  geom_line()
```



```
climate <- read.csv(text =
Rcurl::getURL('https://raw.githubusercontent.com/karthikram/ggplot-lecture/master/climate.csv'))
```

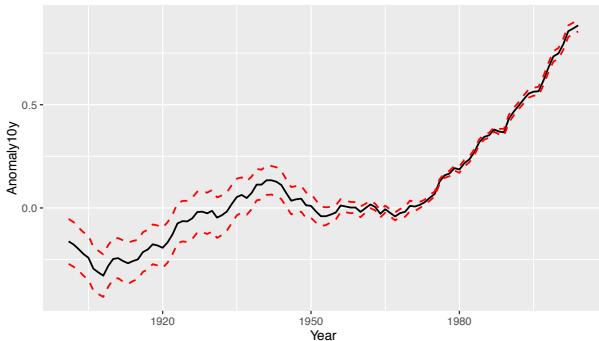
We can also plot confidence regions

```
ggplot(climate, aes(Year, Anomaly10y)) +  
  geom_ribbon(aes(ymin = Anomaly10y - Unc10y,  
    ymax = Anomaly10y + Unc10y),  
    fill = "blue", alpha = .1) +  
  geom_line(color = "steelblue")
```



Exercise 2

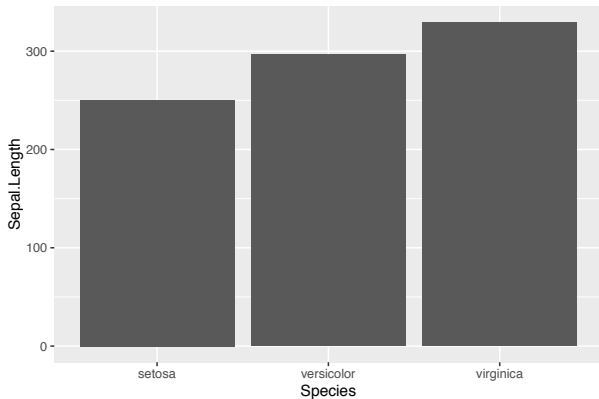
- Modify the previous plot and change it such that there are three lines instead of one with a confidence band.



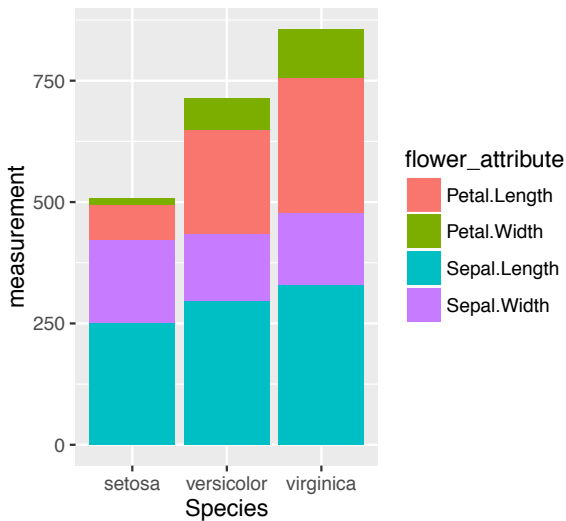
Section 12

Bar plots

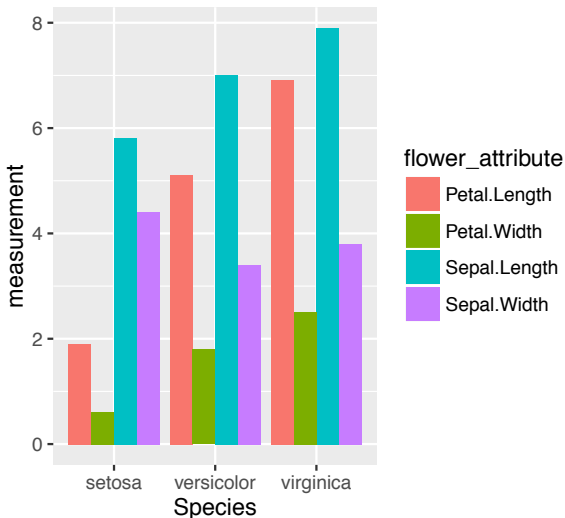
```
ggplot(iris, aes(Species, Sepal.Length)) +  
  geom_bar(stat = "identity")
```



```
ggplot(df, aes(Species, measurement, fill = flower_attribute)) +  
  geom_bar(stat = "identity")
```

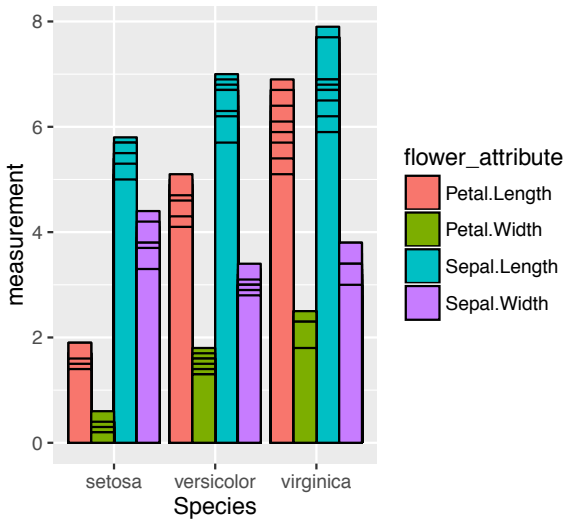


```
ggplot(df, aes(Species, measurement, fill = flower_attribute)) +  
  geom_bar(stat = "identity", position = "dodge")
```



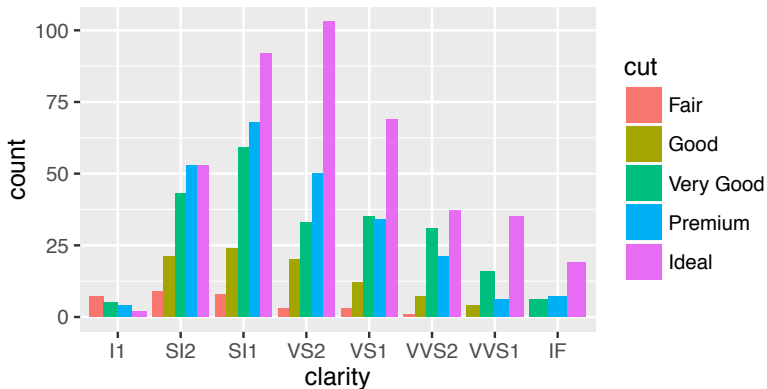
What's going on with the y axis?

```
ggplot(df, aes(Species, measurement, fill = flower_attribute)) +  
  geom_bar(stat = "identity", position="dodge", color="black")
```



Exercise 3

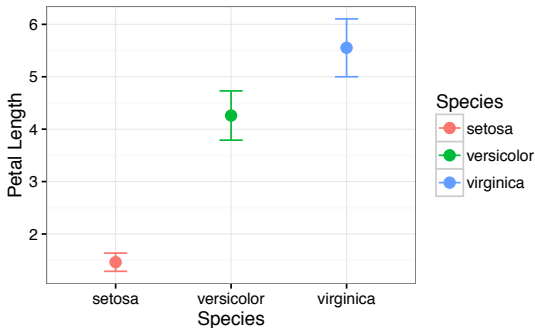
Using the d2 dataset you created earlier, generate this plot below. Take a quick look at the data first to see if it needs to be binned.



Exercise 4

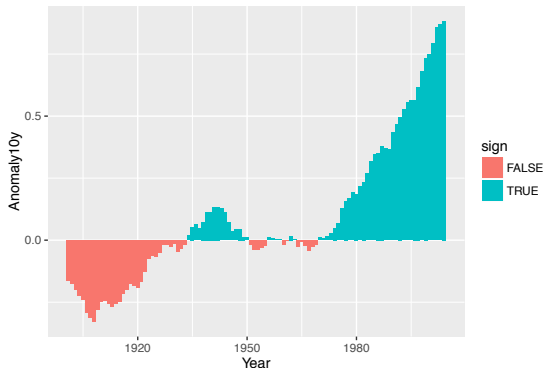
- Use dplyr to calculate the mean and standard deviation of petal length by species.

```
agg <- iris %>%  
  group_by(Species) %>%  
  summarise(mean.petal.length = mean(Petal.Length),  
            sd.petal.length = sd(Petal.Length))  
# ?geom_errorbar()
```



Exercise 5

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

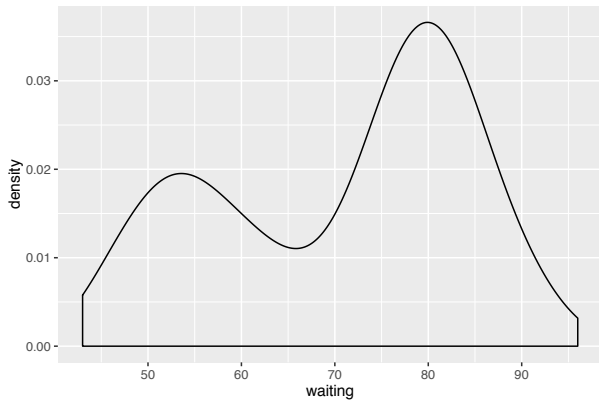


Section 13

Density Plots

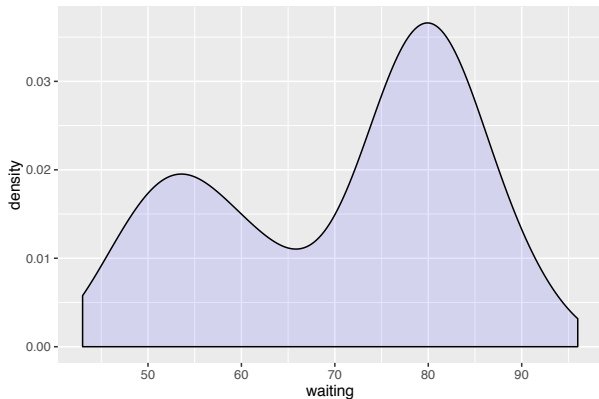
Density plots

```
ggplot(faithful, aes(waiting)) + geom_density()
```

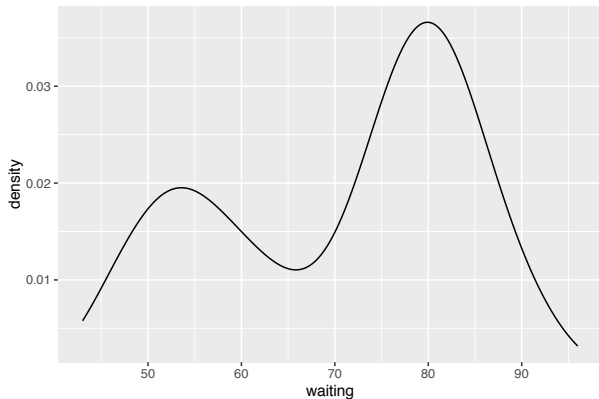


Density plots

```
ggplot(faithful, aes(waiting)) +  
  geom_density(fill = "blue", alpha = 0.1)
```



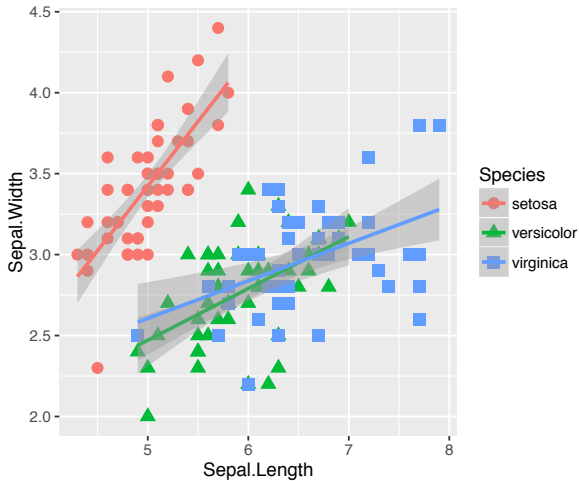
```
ggplot(faithful, aes(waiting)) +  
  geom_line(stat = "density")
```



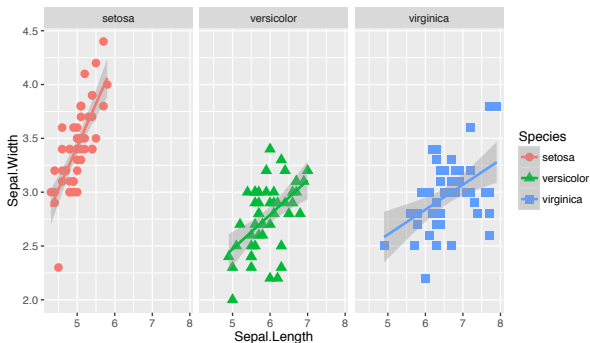
Section 14

Adding smoothers

```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point(aes(shape = Species), size = 3) +  
  geom_smooth(method = "lm")
```



```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point(aes(shape = Species), size = 3) +  
  geom_smooth(method = "lm") +  
  facet_grid(. ~ Species)
```



Section 15

Themes

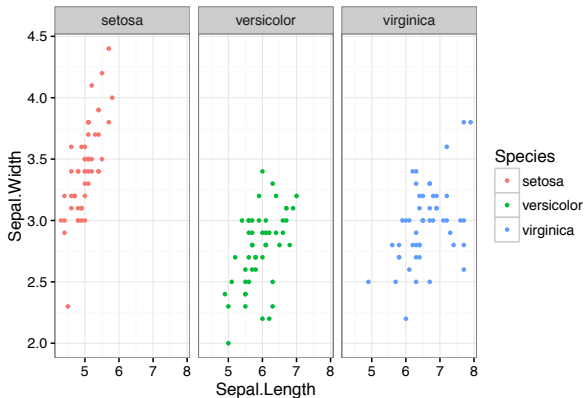
Adding themes

Themes are a great way to define custom plots.

```
+ theme()  
# see ?theme() for more options
```

A more basic theme

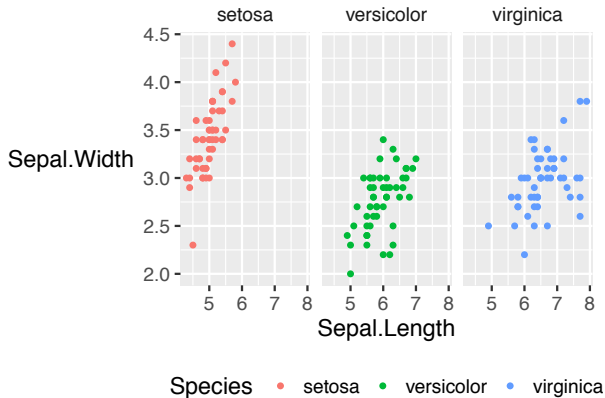
```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point(size = 1.2, shape = 16) +  
  facet_wrap(~ Species) +  
  theme_bw()
```



A themed plot

```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
  geom_point(size = 1.2, shape = 16) +  
  facet_wrap( ~ Species) +  
  theme(legend.key = element_rect(fill = NA),  
        legend.position = "bottom",  
        strip.background = element_rect(fill = NA),  
        axis.title.y = element_text(angle = 0))
```

A themed plot



ggthemes library

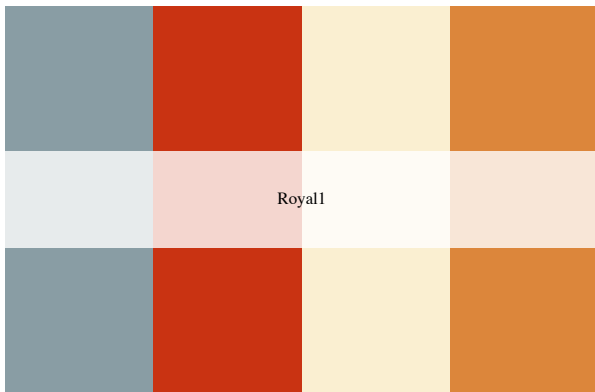
```
install.packages('ggthemes')  
library(ggthemes)  
# Then add one of these themes to your plot  
+ theme_stata()  
+ theme_excel()  
+ theme_wsj()  
+ theme_solarized()
```

Fan of Wes Anderson movies?



Yup, that's a thing

```
# install.packages('wesanderson')  
library("wesanderson")  
# display a palette  
wes_palette("Royal1")
```



Section 16

Create functions to automate your plotting

Write functions for day to day plots

```
my_custom_plot <- function(df, title = "", ...) {  
  ggplot(df, ...) +  
  ggtitle(title) +  
  whatever_geoms() +  
  theme(...)  
}
```

Then just call your function to generate a plot. It's a lot easier to fix one function that do it over and over for many plots

```
plot1 <- my_custom_plot(dataset1, title = "Figure 1")
```

Section 17

Publication quality figures

- ▶ If the plot is on your screen

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

- ▶ If your plot is assigned to an object

```
ggsave(plot1, file = "~/path/to/figure/filename.png")
```

- ▶ Specify a size

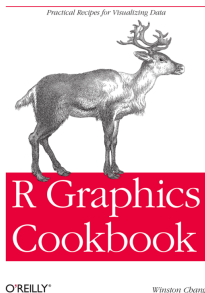
```
ggsave(file = "/path/to/figure/filename.png", width = 6,  
height = 4)
```

- ▶ or any format (pdf, png, eps, svg, jpg)

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

Further help

- ▶ You've just scratched the surface with ggplot2.
- ▶ Practice
- ▶ Read the docs (either locally in R or at <http://docs.ggplot2.org/current/>)
- ▶ Work together



ggplot2 Help Pages

ggplot2 0.9.3.1

[🏠 Index](#)

Help topics

Geoms

Geoms, short for geometric objects, describe the type of plot you will produce.

- [geom_abline](#)
Line specified by slope and intercept.
- [geom_area](#)
Area plot.
- [geom_bar](#)
Bars, rectangles with bases on x-axis
- [geom_bin2d](#)
Add heatmap of 2d bin counts.
- [geom_blank](#)
Blank, draws nothing.
- [geom_boxplot](#)
Box and whiskers plot.

[geom_contour](#)

StackOverflow

[Questions](#)[Tags](#)[Users](#)[Badges](#)

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free.

Greek and alpha numeric in ggplot2 axis labels

Work on work you love. From home.



 stackoverflowcareers



I would like to use ggplot2 to make a chart with a axis label of μL , where the 'u' is the greek 'mu'. to add just mu, I have found this to work

2



```
p <- ggplot(data.frame(x = 1:3, y = 1:3), aes(x= x, y=y)) + geom_point()  
p + ylab(expression(mu))
```



But I have not been able to place anything else alongside it. These do not work

```
p + ylab(paste(expression(mu), "foo"))  
p + ylab("expression(mu)~foo")
```

Thanks in advance

Sam