

# Data Visualization with R & ggplot2

Karthik Ram

July 31, 2013

# Download this PDF

github.com/karthikram/ggplot-lecture

https://speakerdeck.com/karthik/

The screenshot shows the GitHub repository page for `karthikram / ggplot-lecture`. The repository is public. The main content area shows the repository name, a description "My lecture on ggplot at Cal (spring 2013) — [Read more](#)", and a list of clone options: "Clone in Mac", "ZIP" (highlighted with a red box), "HTTP", "SSH", "Git Read-Only", and a text input field containing `git@github.com:karthikram/ggplot-lecture.git` (also highlighted with a red box). Below the clone options, there are tabs for "branch: master", "Files", "Commits", and "Branches". The "Files" tab is selected, showing a list of files. The first file is "cropped slide image", which was authored by karthikram 6 minutes ago. The latest commit is `f171851480`. Below the file list, there is a commit message "Added new beamer theme and some images [karthikram]" from an hour ago.

PUBLIC **karthikram / ggplot-lecture** Pull Request Unwatch Star 0 Fork 0

Code Network Pull Requests 0 Issues 0 Wiki Graphs Settings

My lecture on ggplot at Cal (spring 2013) — [Read more](#)

Clone in Mac ZIP HTTP SSH Git Read-Only `git@github.com:karthikram/ggplot-lecture.git` Read+Write access

branch: master Files Commits Branches 1 Tags

**ggplot-lecture** 16 commits

cropped slide image

**karthikram** authored 6 minutes ago Latest commit `f171851480`

images an hour ago Added new beamer theme and some images [karthikram]

# Some housekeeping

Install some packages (make sure you also have recent copies of reshape2 and plyr)

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

# Base graphics

- ▶ Ugly, laborious, and verbose
- ▶ There are better ways to describe statistical visualizations.

# Why ggplot2?

- ▶ Follows a grammar, just like any language.
- ▶ It defines basic components that make up a sentence. In this case, the grammar defines components in a plot.
- ▶ Grammar of graphics originally coined by Lee Wilkinson

# Why ggplot2?

- ▶ Supports a continuum of expertise.
- ▶ Get started right away but with practice you can effortlessly build complex, publication quality figures.

# Section 1

## Basics

## Some terminology

- ▶ **ggplot** - The main function where you specify the dataset and variables to plot
- ▶ **geoms** - geometric objects
  - ▶ `geom_point()`, `geom_bar()`, `geom_density()`, `geom_line()`, `geom_area()`
- ▶ **aes** - aesthetics
  - ▶ shape, transparency (`alpha`), color, fill, linetype.
- ▶ **scales** Define how your data will be plotted
  - ▶ *continuous*, *discrete*, *log*



## Section 2

### Assembling your first ggplot

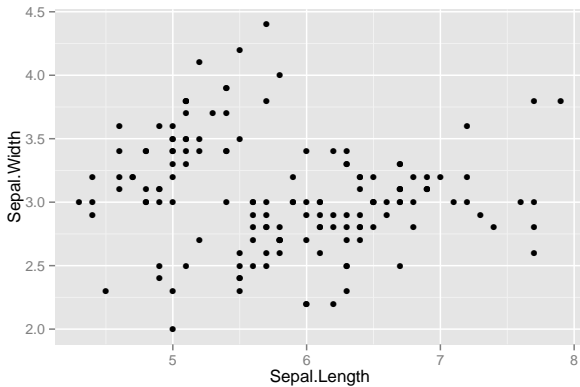
# The iris dataset

```
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

## Let's try an example

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



# 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()
```

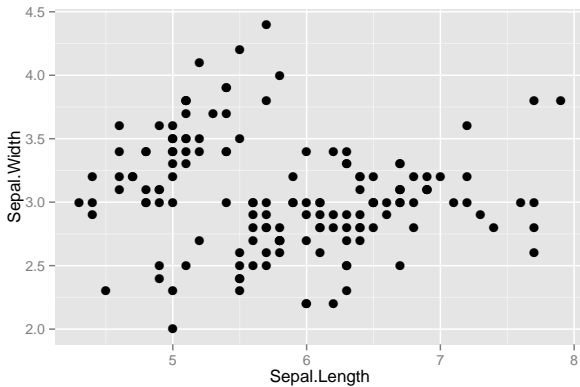
- ▶ Specify the data and variables inside the ggplot function.
- ▶ Anything else that goes in here becomes a global setting.
- ▶ Then add layers of geometric objects, statistical models, and panels.

## Quick note

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

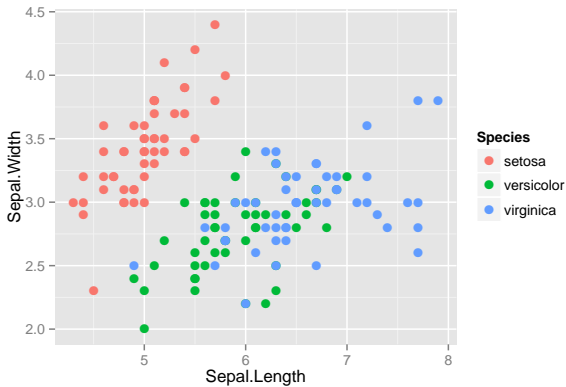
# Increase the size of points

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



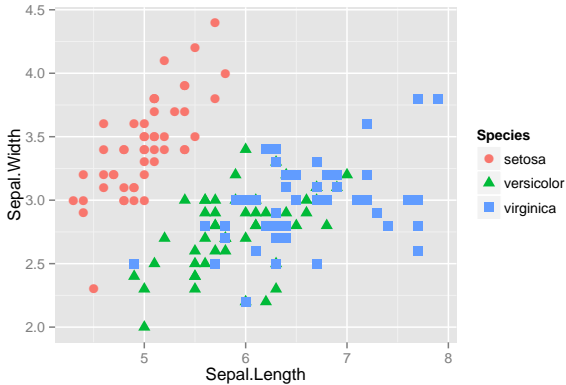
## Add some color

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



# Differentiate points by shape

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

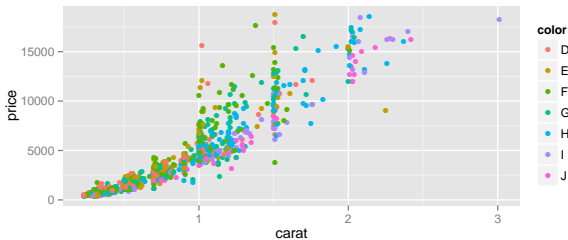




# Exercise 1

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

Then generate this plot below.

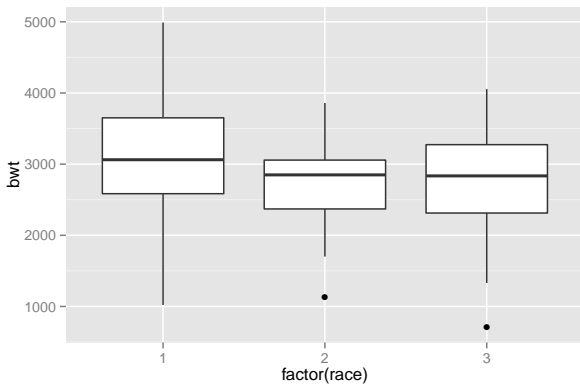


## Section 3

### Box plots

See ?geom\_boxplot for list of options

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

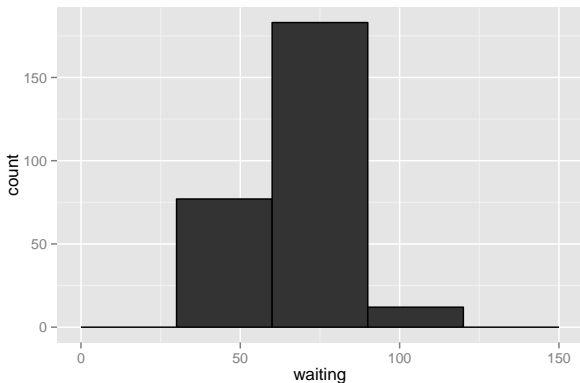


## Section 4

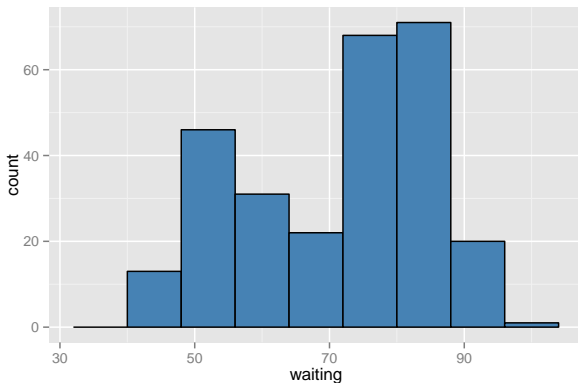
# Histograms

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")
```

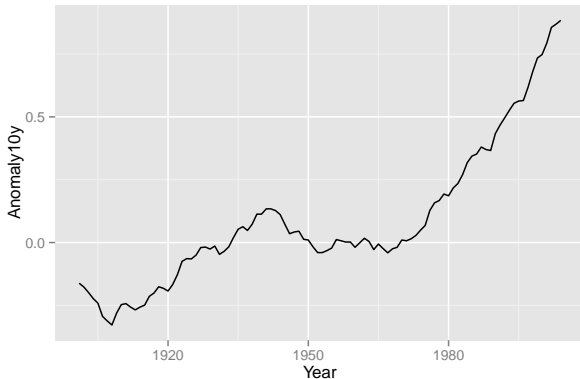


# Section 5

## Line plots

## ## Error: cannot change working directory

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

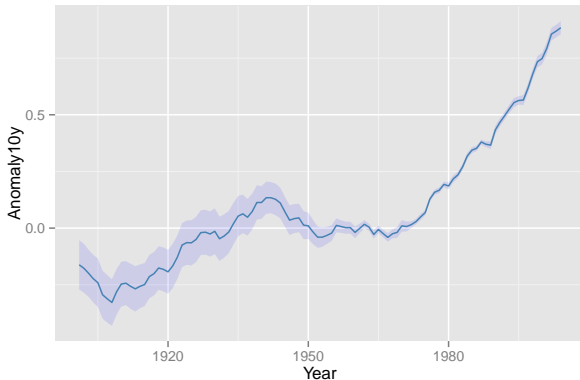


```
climate <- read.csv(text =
  RCurl::getURL('https://raw.githubusercontent.com/karthikram/ggplot-lecture/master/climate.csv'))
```



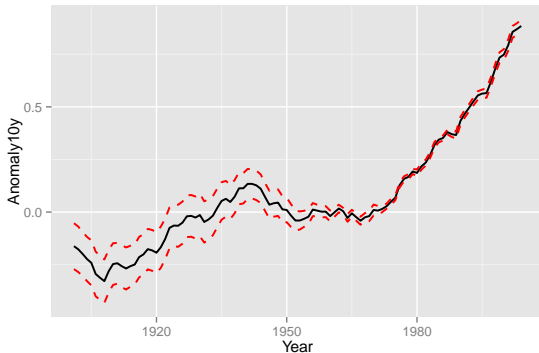
## We can also plot confidence regions

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



## Exercise 2

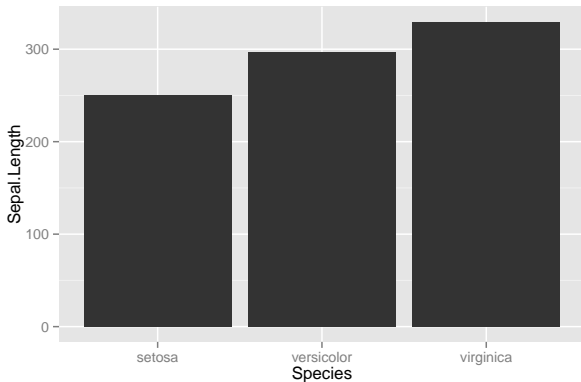
- Modify the previous plot and change it such that there are three lines instead of one with a confidence band.



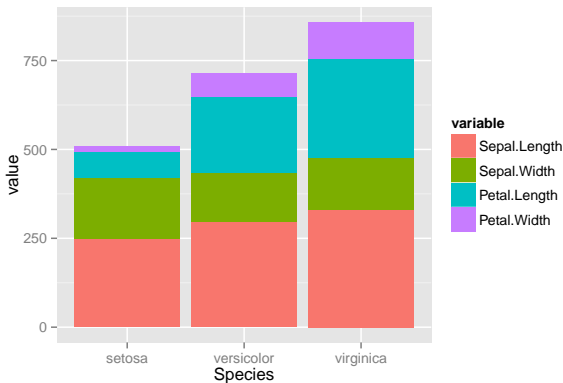
## Section 6

### Bar plots

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



```
df <- melt(iris, id.vars = "Species")
ggplot(df, aes(Species, value, fill = variable)) +
  geom_bar(stat = "identity")
```



## Section 7

plyr and reshape are key for using R

# plyr and reshape

These two packages are the swiss army knives of R.

- ▶ plyr
  1. dply
  2. llply
  3. join
- ▶ reshape.
  1. melt
  2. dcast
  3. acast

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

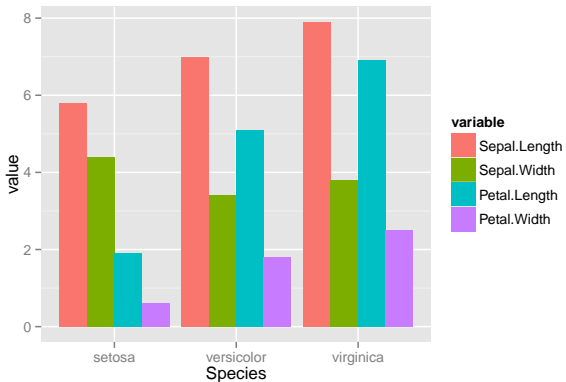
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1          3.5          1.4          0.2   setosa
## 2          4.9          3.0          1.4          0.2   setosa
```

```
df <- melt(iris, id.vars = "Species")
df[1:2, ]
```

```
##   Species      variable value
## 1   setosa Sepal.Length    5.1
## 2   setosa Sepal.Length    4.9
```

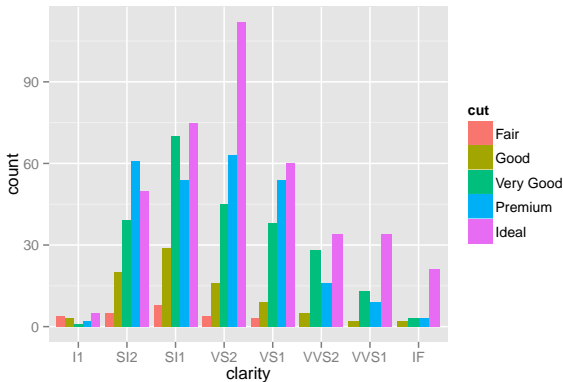


```
ggplot(df, aes(Species, value, fill = variable)) +  
geom_bar(stat = "identity", position = "dodge")
```



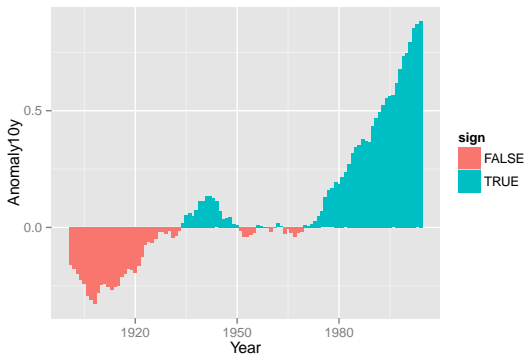
## Exercise 3

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

- ▶ 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.

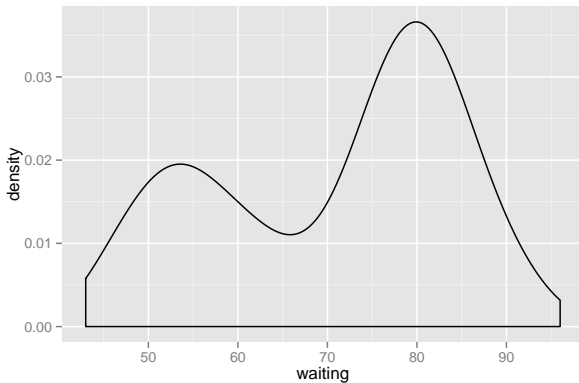


## Section 8

### Density Plots

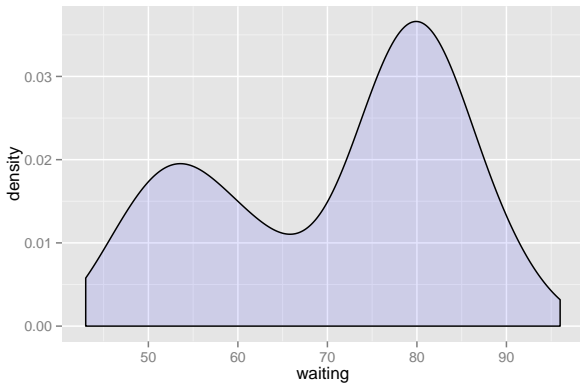
# Density plots

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

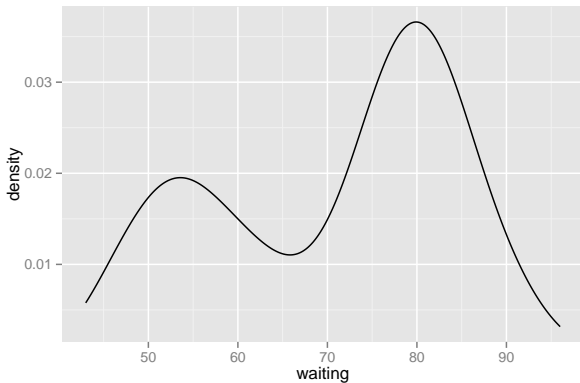


# Density plots

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



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



## Section 9

### Mapping Variables to colors

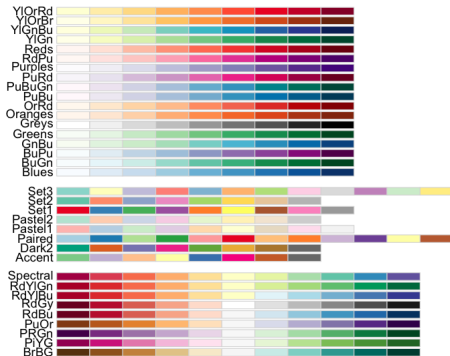


# Colors

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

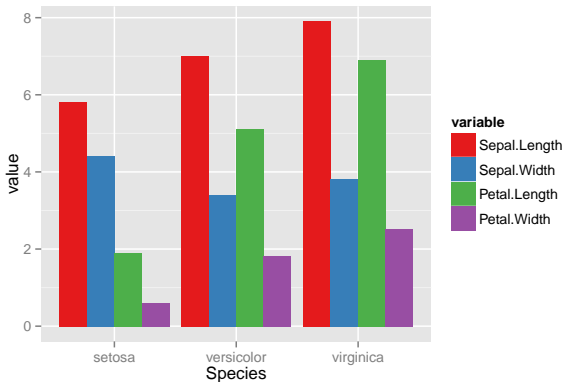
# The RColorBrewer package

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



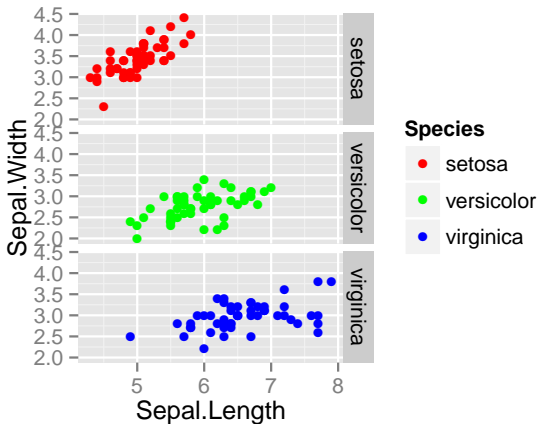
## Using a color brewer palette

```
df <- melt(iris, id.vars = "Species")
ggplot(df, aes(Species, value, fill = variable)) +
  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) ▾



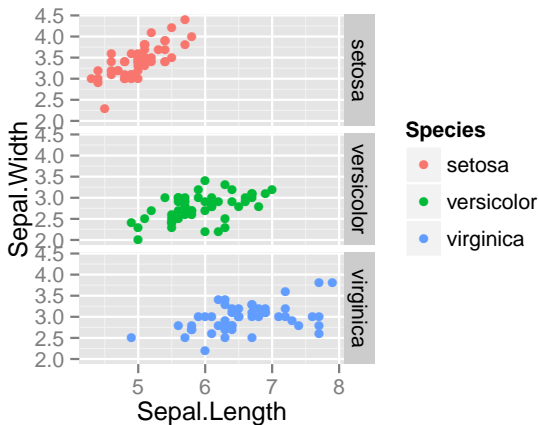


## Section 10

### Faceting

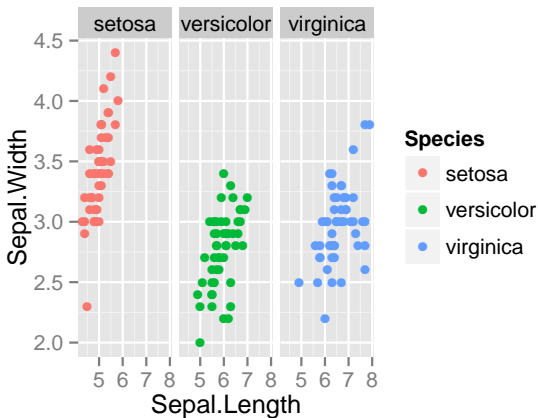
# Faceting along columns

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



and along rows

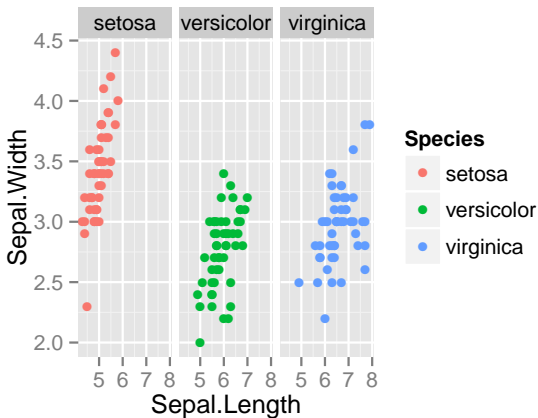
```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
geom_point() +  
facet_grid(. ~ Species)
```





## or just wrap your panels

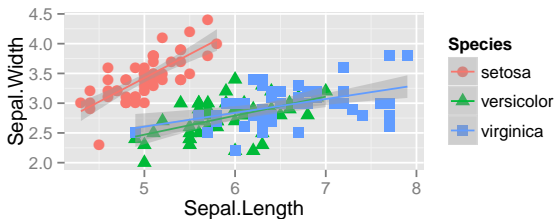
```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +  
geom_point() +  
facet_wrap(~ Species)
```



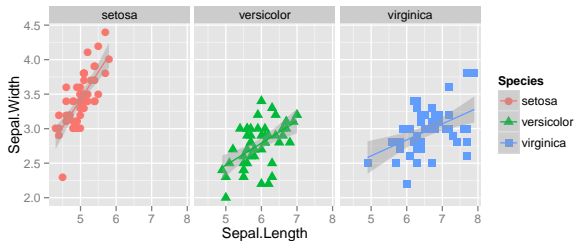
# Section 11

## 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 12

## Themes

# Adding themes

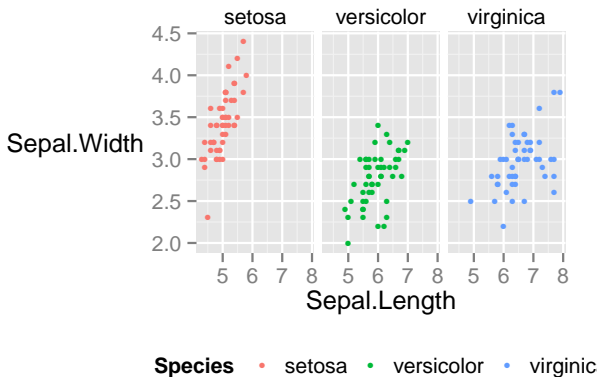
Themes are a great way to define custom plots.

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

## 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))
```

# Adding themes





# ggthemes library

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

## Section 13

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 14

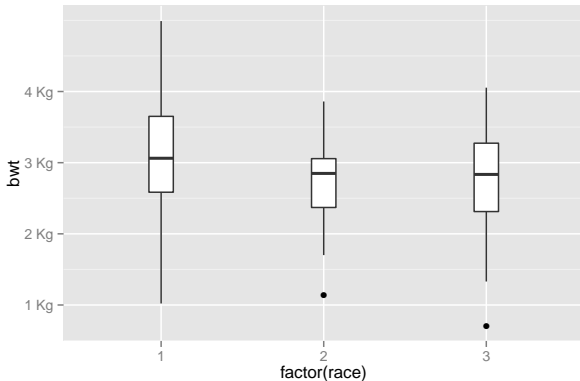
### Scales

## 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()
```

# Adding a continuous scale

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

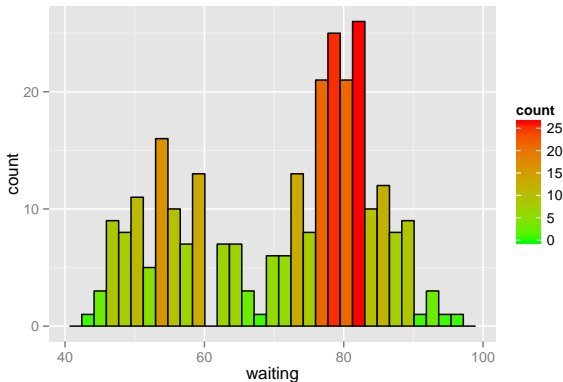


## Another continuous scale with custom labels

```
# Assign the plot to an object
dd <- ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point(size = 4, shape = 16) +
  facet_grid(. ~Species)
# Now add a scale
dd +
  scale_y_continuous(breaks = seq(2, 8, by = 1),
    labels = paste0(2:8, " cm"))
```

# gradients

```
h + geom_histogram( aes(fill = ..count..), color="black") +  
  scale_fill_gradient(low="green", high="red")
```





## Section 15

### 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

