# Data Visualization with ggplot2

## Basics

### ggplot2

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

```
ggplot(data = mpg, aes(x = ct, y = hwy))
```

- To display data values, map variables in the data set to aesthetic properties of the geom like size, color, and x and y locations.

## Graphical Primitives

```
a <- ggplot(seals, aes(x = long, y = lat))
b <- ggplot(economics, aes(date, unemployment))
```

### Geoms

#### Use a geom to represent data points, use the geom’s aesthetic properties to represent variables. Each function returns a layer.

### One Variable

#### Continuous

```
c <- ggplot(mpg, aes(hwy))
```

#### Discrete

```
g <- ggplot(diamonds, aes(cut, color))
```

### Two Variables

#### Continuous

```
h <- ggplot(mpg, aes(cty, hwy))
```

#### Discrete

```
l <- ggplot(seals, aes(z = z))
```

## Geoms - A cheat sheet

### Graphical Primitives

```
a <- ggplot(data, aes(x = long, y = lat))
b <- ggplot(economics, aes(date, unemployment))
```

### Geoms

#### Use a geom to represent data points, use the geom’s aesthetic properties to represent variables. Each function returns a layer.

### One Variable

#### Continuous

```
c <- ggplot(mpg, aes(hwy))
```

#### Discrete

```
g <- ggplot(diamonds, aes(cut, color))
```

### Two Variables

#### Continuous

```
h <- ggplot(diamonds, aes(carat, price))
```

#### Discrete

```
l <- ggplot(seals, aes(z = z))
```

## Maps

```
d <- ggplot(data, aes(fill = z))
```

Learn more at [docs.ggplot2.org](http://docs.ggplot2.org) • ggplot2 2.0.0 • Updated: 12/15

RStudio® is a trademark of RStudio, Inc. • CC BY 3.0
Some plots visualize a transformation of the original data set. Use a `scale` to choose a common transformation to visualize, e.g., `aesthetics = "count"`.

Each stat creates additional variables to map aesthetics to. These variables use common `.name`, `.x`, `.y`, `.size`. `stat` and `geom` functions both combine a `stat` with a `geom` to make a layer, i.e., `stat_count(geom = "stat_bar")` does the same as `geom_bar(stat = "count")`.

### Stats - An alternative way to build a layer

Some plots visualize a transformation of the original data set. Use a `stat` to choose a common transformation to visualize, e.g., `aesthetics = "fill"`.

Each stat creates additional variables to map aesthetics to. These variables use common `.name`, `.x`, `.y`, `.size`. `stat` and `geom` functions both combine a `stat` with a `geom` to make a layer, i.e., `stat_count(geom = "stat_bar")` does the same as `geom_bar(stat = "count")`.

### Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

```
+ scale_<aesthetic> (data = ..., name = ..., range = ..., breaks = ..., labels = ..., ..., ...)  
```

- **aesthetic** to adjust
- **prepackaged** scale to use
- **scale specific arguments**

#### General Purpose scales

- **scale_*_continuous()** - map cont` values to visual values
- **scale_*_discrete()** - map discrete values to visual values
- **scale_*_identity()** - use data values as visual values
- **scale_*_manual()** - map discrete values to manually chosen visual values

X and Y location scales

Use with X or Y aesthetics (X shown here)

```
X or Y aesthetic
```

```
+ scale_<aesthetic> (data = ..., name = ..., range = ..., breaks = ..., labels = ..., ..., ...)  
```

- **aesthetic** to adjust
- **prepackaged** scale to use
- **scale specific arguments**

#### Color and fill scales

```
Discrete
```

```
+ scale_<aesthetic> (data = ..., name = ..., range = ..., breaks = ..., labels = ..., ..., ...)  
```

- **aesthetic** to adjust
- **prepackaged** scale to use
- **scale specific arguments**

#### Continuous

```
Continuous
```

```
+ scale_<aesthetic> (data = ..., name = ..., range = ..., breaks = ..., labels = ..., ..., ...)  
```

- **aesthetic** to adjust
- **prepackaged** scale to use
- **scale specific arguments**

### Coordinate Systems

```
X and Y location scales
```

```
+ scale_<aesthetic> (data = ..., name = ..., range = ..., breaks = ..., labels = ..., ..., ...)  
```

- **aesthetic** to adjust
- **prepackaged** scale to use
- **scale specific arguments**

### Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s + geom_<geom_name>(position = "<adjustment>")
```

- **geom_<geom_name>** - geom to make a layer, i.e. `geom_bar()`
- **position** - adjust the position of each element to avoid overplotting

### Faceting

```
f + facet_<facet_name>(vars = ..., title = ..., ..., ...)  
```

- **facet_<facet_name>** - facet into rows based on year
- **facet_wrap(<formula>)** - facet into a rectangular layout

### Labels

```
f + labs(name = "<label>", title = "<title>", subtitle = "<subtitle>")
```

- **name** - label each aesthetic: colorbar, legend, axis, title
- **title** - label each aesthetic: colorbar, legend, axis, title
- **subtitle** - label each aesthetic: colorbar, legend, axis, title

### Themes

```
f + theme_<theme_name>(...)  
```

- **theme_<theme_name>** - theme to change the overall appearance of a plot

### Zooming

```
f + zoom(<zoom_area>, <zoom_target>, <zoom_direction>)
```

- **zoom_area** - area to zoom into
- **zoom_target** - area to zoom out to
- **zoom_direction** - direction to zoom

Learn more at docs.ggplot2.org • ggplot2 2.0.0 • Updated: 12/15