Lecture 12: Exploratory Data Analysis - Part 2

STAT 385 - James Balamuta

July 14, 2016

On the Agenda

1. Administrative Issues
   - Midterm (Part 2)
   - Group Project Update due on 7/19/16 at 11:59 PM CDT.
2. Graphing Systems in R
   - `Base R`, `lattice`, `ggplot2`
3. Exploratory Data Analysis
   - Visual Techniques

Midterm (Part 2)

- 10 minutes to do Midterm (Part 2)
- No notes or collaborating!
- 5 Points of E.C. up for grabs!

Group Project Update

- Help me, help you by letting me know about your group's project status
- Please answer:
    1. How is the project progressing?
    2. What has been accomplished thus far?
    3. What have you learned?
    4. What issues have arisen?
- *Avoid* showing me code in the report.
- Score for the progress report is based on how much work has been completed since the project proposal was initially submitted.

Moving along. . .

- We're going to cover **Graphing in *R*** next!

Graphing in *R*

- Before now, we never really focused on plotting.
- Instead, we aimed to understand the computing logic behind calculations in *R*.
- Now, to support visual *EDA*, we really need to start focusing on such features.

R and the Three Graphing Systems

- Dilemma: There are ***3*** graphing systems to chose from in *R*.
- Similar to the **Goldilocks and the Three Bears** problem.

Figure 1:

- Selecting the graphical system is important. . .

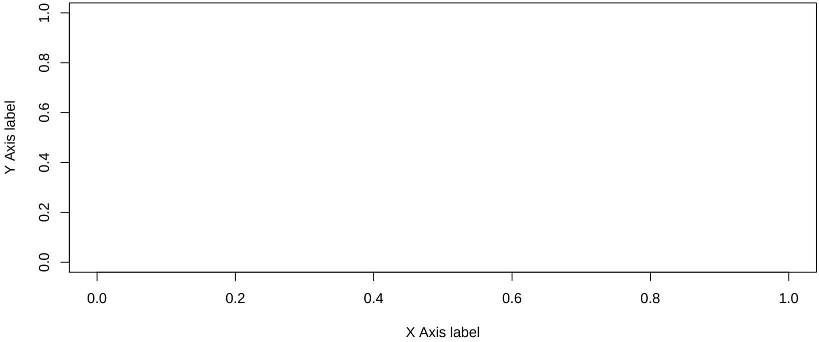
Different Plotting Systems in *R* - Overview

- R's `Base` plotting system.
  - `plot()`, `hist()`, `barplot()`
- `lattice` formulaic graphing in *R*.
  - `xyplot()`, `dotplot()`, `histogram()`, `*plot()`
- `ggplot2` rapid layered graphing approach
  - graphs start with `ggplot()` and add layers via + typically denoted by `geom_point()` , `geom_*()`

R's Base Plotting System - Example

- View it as an artists blank canvas

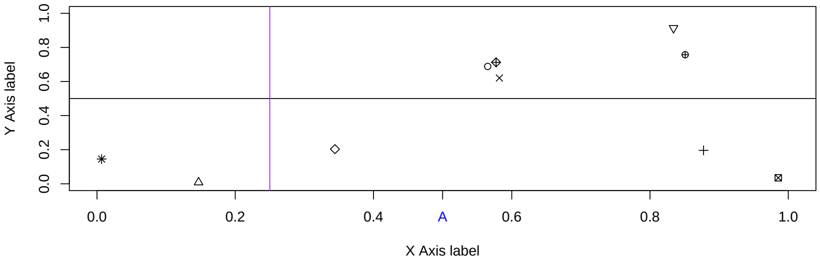
```
plot(NULL, xlim=c(0,1), ylim=c(0,1),
     ylab="Y Axis label", xlab="X Axis label")
```



R's Base Plotting System - Example

- Each subsequent function calls adds lines, points, axis, et cetera.

```
x = runif(10); y = runif(10)
abline(h = .5)                    # Horizontal Line
abline(v = .25, col="purple") # Vertical Line w/ color
points(x, y, pch = 1:10)          # Points w/ shapes
axis(1, .5, LETTERS[1], col.axis = "blue")
```



R's Base Plotting System - Verdict

- **Con:** No ability to change plot settings (e.g. `?par` settings) or draw content added once started.
- **Pros:** Easier custom graphs and higher quality graphs (e.g. AVLR using the `tikzDevice` package)
- **Verdict: Academics only**

Credit: The Global Warmers

lattice - A formulaic approach to graphs.

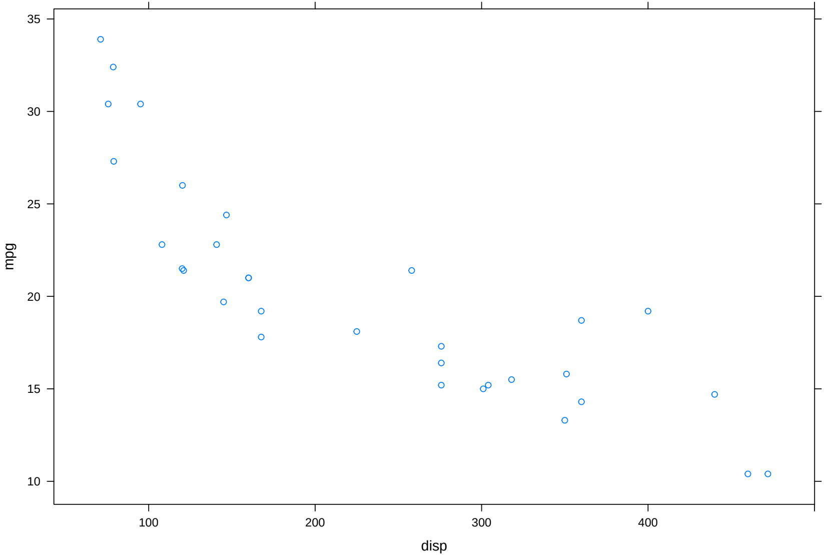
- The lattice package written by Deepayan Sarkar provides the ability to make graphs in *one* call vs. Base R's multiple calls.
- The call form is normally:

```
type_of_plot(formula, data=list())
```

- Uses the formula object associated with lm to specify: *response* (y~), *explanatory* (~x), *conditional relationships* (y~x|A).
- Great for viewing conditional relationships and multivariate data.

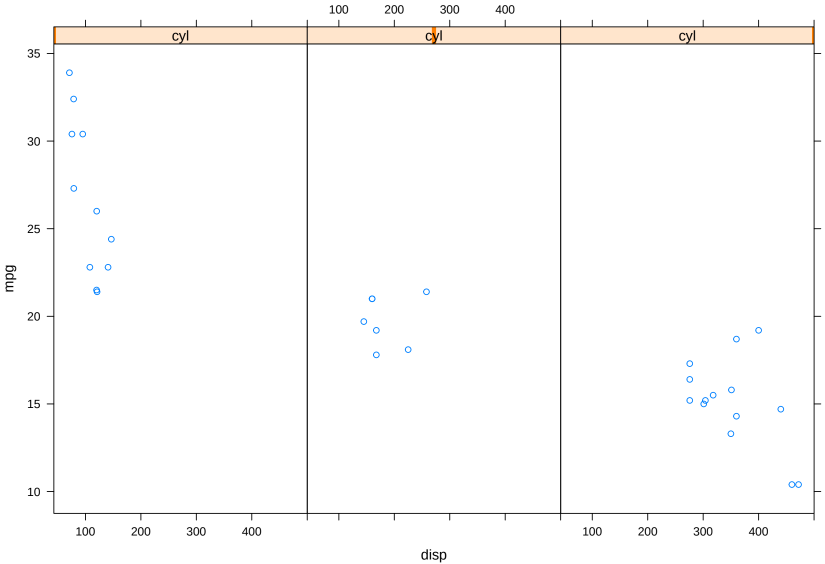
lattice - Example

```
xyplot(mpg ~ disp, data = mtcars)
```



lattice - Example with Condition

```
xyplot(mpg ~ disp|cyl, data = mtcars) # Note the |
```



`lattice` - Verdict

- **Cons:** Everything in 1 function call is *messy* and awkward.
- **Pros:** Handle all margin settings of multiple graphs and conditioning.
- **Verdict: Casual *R* users.**

Credit: The Global Warmers

ggplot2 - Grammar of Graphics

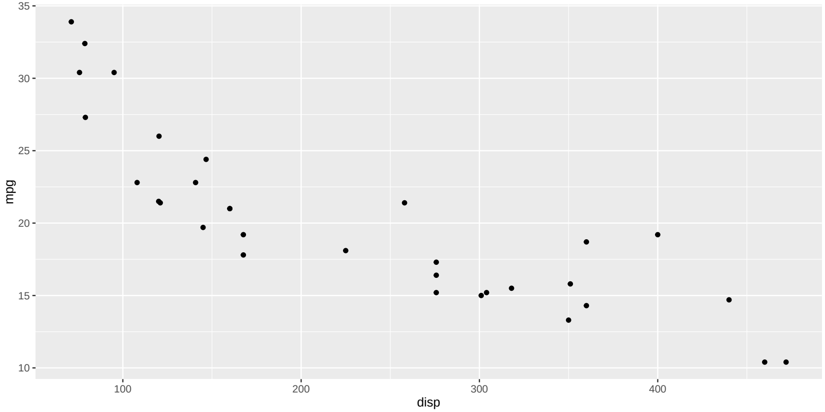
- ggplot2 is the implementation of the pivotal 1999 Book Grammar of Graphics by Leland Wilkinson.
  - Each Graph shares a common structure.
  - The difference between graphs is different component layers and rules.

ggplot2 - Grammar of Graphics

- Historical Information
  - ggplot1 written by Hadley Wickham as part of his PhD thesis.
  - ggplot2 released for ease of use alongside A Layered Grammar of Graphics
  - UseR 2016 Keynote: ggplot1 is better than ggplot2 API wise due to the piping operator
    - Time: 36:38 to 38:32

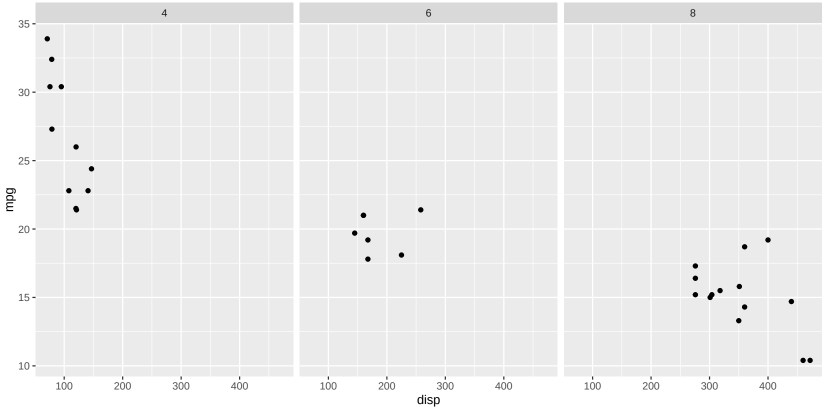
ggplot2 - Scatterplot

```
ggplot(mtcars) +                    # Supply data.frame
  geom_point(aes(disp, mpg))    # Add points to plot
```



ggplot2 - Scatterplot Conditioned

```
ggplot(mtcars) +                    # Supply data.frame
  geom_point(aes(disp, mpg)) + # Add points to plot
  facet_wrap(~cyl)                  # Write conditioning
```



ggplot2 vs. Base R

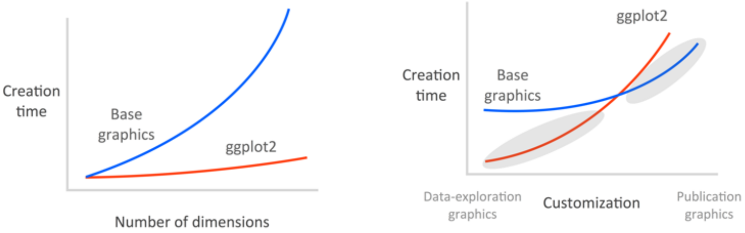


Figure 2: Credit: Sean C. Anderson

`ggplot2` - Verdict

- **Cons:** Data must be in a `data.frame`, global scoping of variables, data copies, and simple things might be *complex*.
- **Pros:** Rapidly iterate visualizations, grammatical structure, and extendable graphing system.
- **Verdict: Data Scientists, Researchers, and Causal *R* users.**

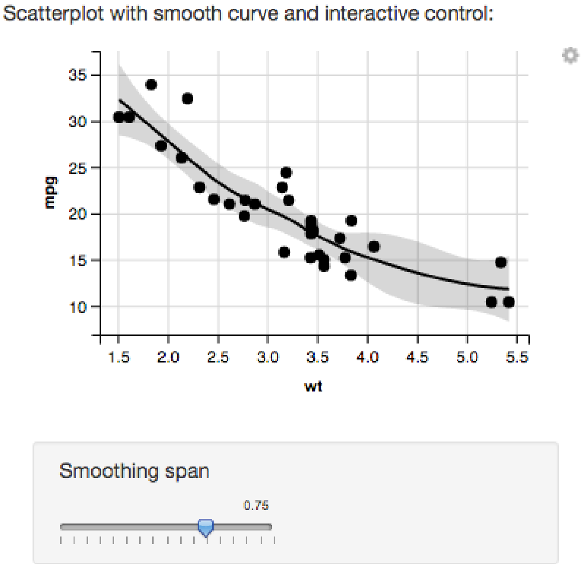
Credit: The Global W

I lied. . .

- As is the case with technology, there always a new graphing system around the corner.
- Coming Soon a 4th Graphical System for *R* using the parts of `ggplot2`. . . .

ggvis - Coming Soon (TM)

- Introducing ggvis, the successor to ggplot2...



Scatterplot with smooth curve and interactive control:

Smoothing span

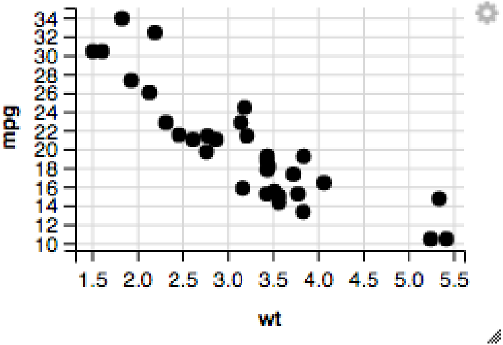
0.75

ggvis - Coming Soon (TM)

- ggvis will usher in a new way of interactive graphics (think identify()). Keep an eye on this project.
- Alas, as the API is currently in flux, we will not dedicate any time to it. However, note that ggvis replaces ggplot2's concatenation with the %>% operator.

ggvis - Example

```
# install.packages("ggvis")
library("ggvis")
mtcars %>% ggvis(~wt, ~mpg) %>% layer_points()
```



Note: This code can only be run in *HTML* rich environments. No LaTeX environments need apply.

Moving Along . . .

- Any questions on **R's Plotting Systems?**
- Next up. . . **Visual EDA**!

Visual EDA

"Use a picture. It's worth a thousand words"
— Tess Flanders in Speakers Give Sound Advice

Visual EDA

Data Wrangling `birth` from `msos` package.

For the next section, I'll aim to use the `birth` data from `msos`. Note, the data is in **wide** form in a `matrix`. The below script sets up the data for graphing by converting it to **long** form and class `data.frame`.

```r
# Extract hospital birth dates
data(births, package="msos")

library("tidyr")
df_births = as.data.frame(births)
df_births$time = seq_len(nrow(df_births))
long_births = gather(df_births, hospital, value, -time)
```

Looking into long_births

Let's peek at what the data in long_births looks like.

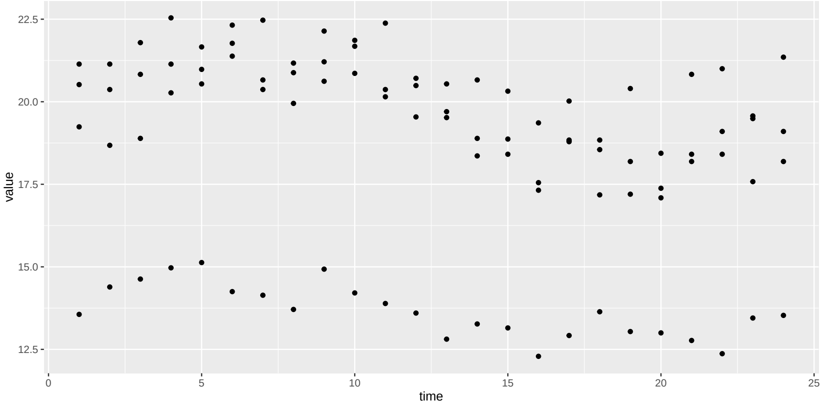
```
head(long_births)
```

```
##   time  hospital value
## 1    1 Hospital1 13.56
## 2    2 Hospital1 14.39
## 3    3 Hospital1 14.63
## 4    4 Hospital1 14.97
## 5    5 Hospital1 15.13
## 6    6 Hospital1 14.25
```

Note: long_births is of class data.frame!!

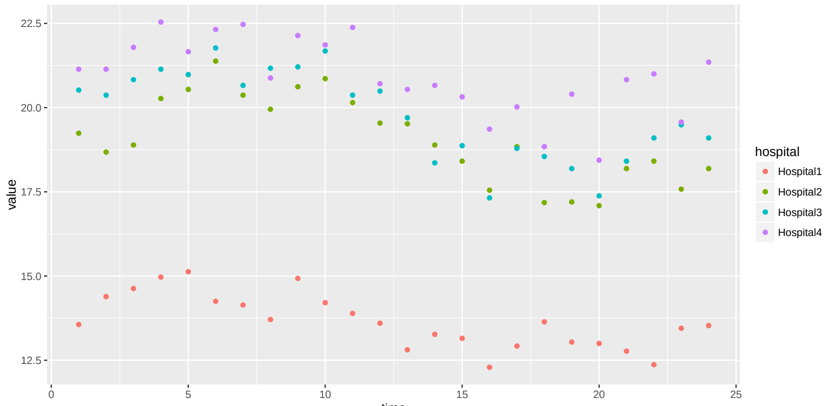
Your first ggplot!

```
ggplot(long_births) +       # Initialize ggplot w/ data
  geom_point(               # Add a point layer
    aes(x = time, y = value)# Add an aesthetic mapping
             )
```



Your second ggplot!

```
ggplot(long_births) +    # Initialize ggplot w/ data
  geom_point(            # Add a point layer
    aes(x = time,        # Add an aesthetic mapping
        y = value,
        color = hospital)# Added color
             )
```



Key Terms and Ideas with `ggplot2`

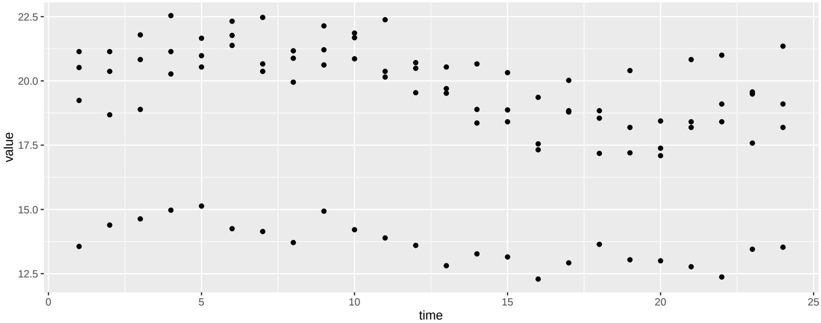
- **ggplot**: Initialization function of the graph
- **geom_**: Geometric (shape) objects
- **aes**: Provides the aesthetic options the `geom` should take.
  - Examples: color, fill, transparency (alpha), linetype, and point shape.
- **scales**: Axis kind
  - Examples: Continuous, Discrete, $\log$, $\sqrt{}$, and so on.
- **facet**: Panel layout
  - Examples: Grid ($x \times y$) or Wrapped

Reusing ggplot2 base objects

Each ggplot2 object can be saved individually and added to in the future

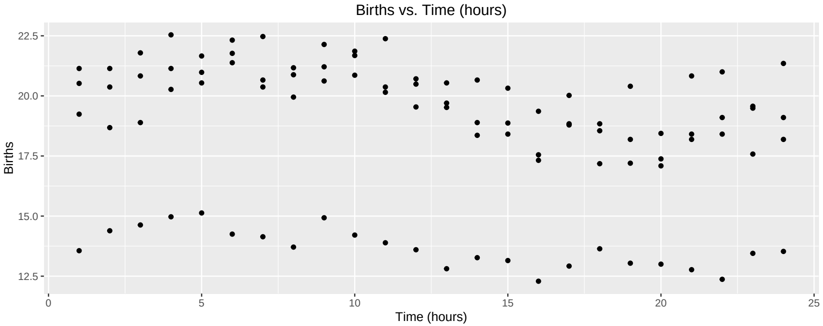
```
g = ggplot(long_births) +
  geom_point(aes(x = time, y = value))

g
```



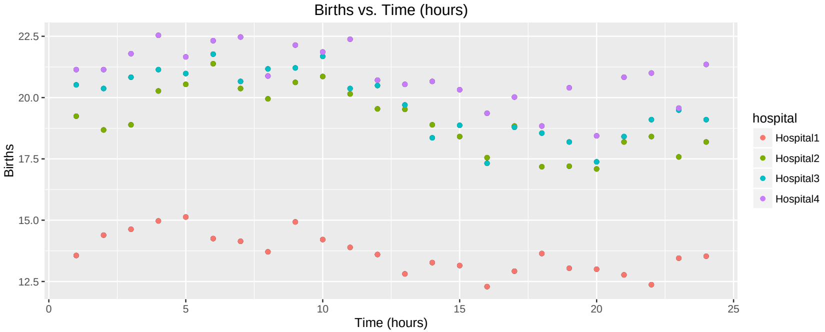
Adding Label Information to ggplot2

```
(g = g + xlab("Time (hours)") + ylab("Births") +
  ggtitle("Births vs. Time (hours)"))
```



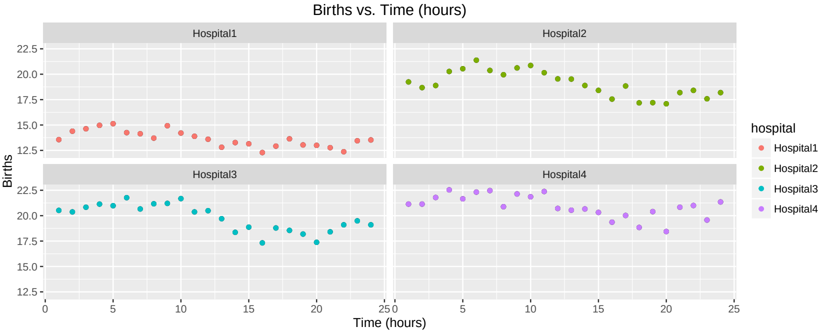
Changing aes for geom_point

```
(g = g + geom_point(aes(x = time, y = value, color = hospit
```



Adding a `facet_wrap` to distinguish variables

```
(g = g + facet_wrap(~hospital))
```



Graphing with ggplot2

- ggplot2 makes available various geometric objects via geom_.
- These objects determine how the data is rendered on the plot.
- Some of the geoms_*() typically used:

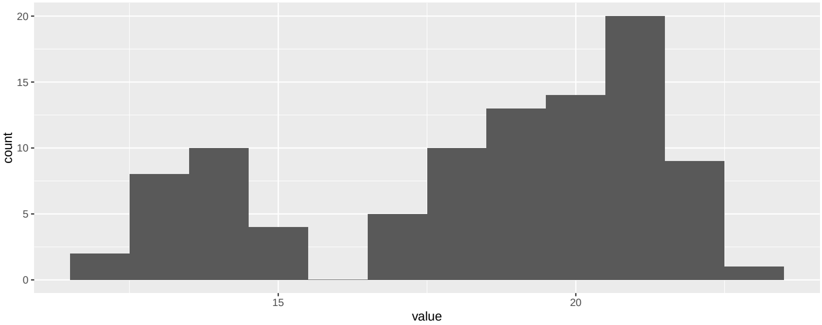
geoms_*()	Description
geom_point()	Adds data points to plot
geom_line()	Adds connected lines to the plot
geom_histogram()	Makes a histogram
geom_bar()	Creates a bar chart
geom_text()	Adds text annotations
geom_violin()	Makes a violin plot

- Many more geoms_*() exists and can be found at docs.ggplot2.org with graphing examples!

Making a histogram

Histograms typically provide count or frequency values on the y-axis

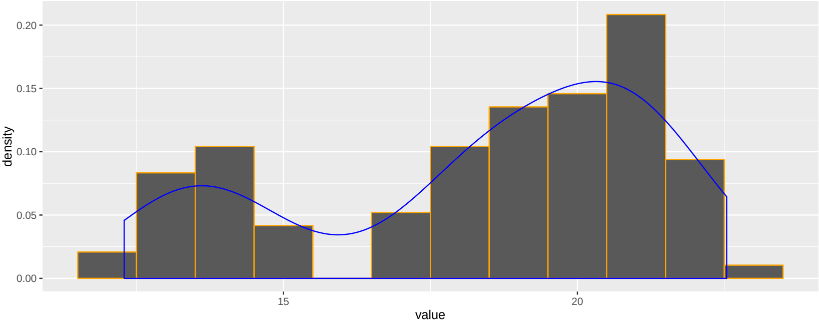
```
ggplot(long_births) +
  geom_histogram(aes(value), binwidth = 1)
```



Making a histogram with a density plot

Density plots alongside a histogram require `density` (bounded between 0 and 1) to be on the y-axis. If `count` is on the y-axis, then results are not valid.

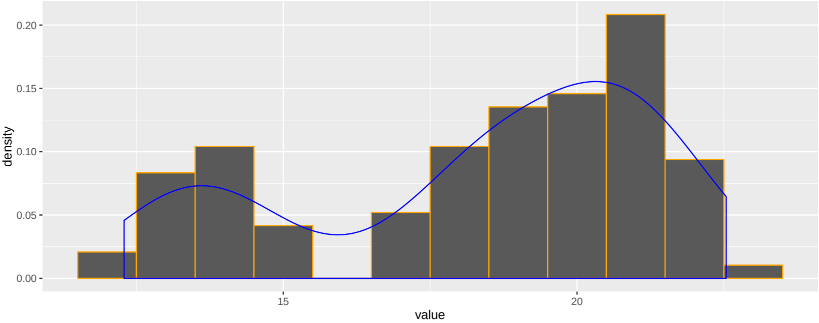
```
ggplot(long_births) +
  geom_histogram(aes(x = value, y = ..density..),
                 binwidth = 1, color = "orange") +
  geom_density(aes(value), color = "blue")
```



Storing aes in construction

Specify aes() does not necessarily have to be done in the geom_*() call. Some users prefer to specify the relationship in the ggplot() creation.

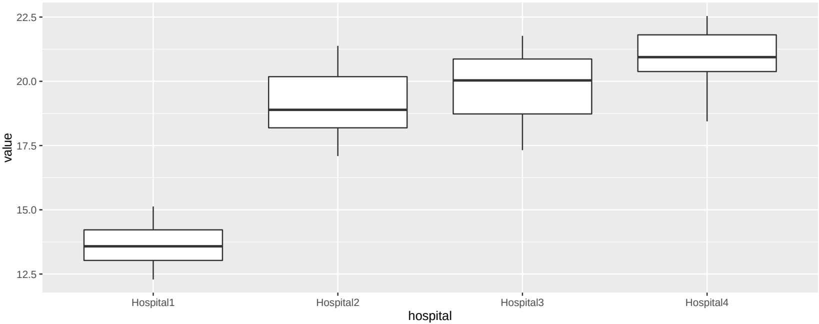
```
ggplot(long_births, aes(x = value)) +
  geom_histogram(aes(y = ..density..), # Notice no `x=`
                 binwidth = 1, color = "orange") +
  geom_density(color = "blue")         # Notice no `x=`
```



Boxplot

Boxplot are a helpful way to visualize Q1, Q2, Q3, and outlier information. They are may be referred to as a *box and whiskers* plot.

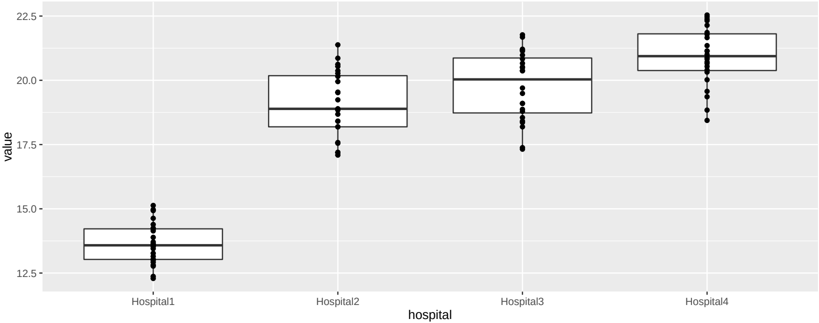
```r
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot()
```



Boxplot with Points

What is nice, is instead of only seeing outliers, you can also see where all the points lie just by adding `geom_point()`

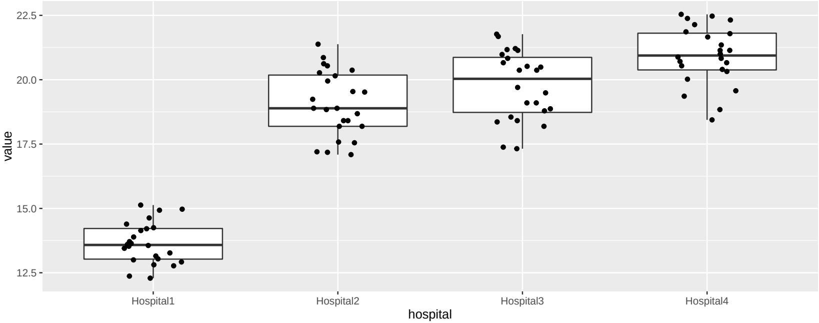
```r
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot() + geom_point()
```



Boxplot with Points Redux

However, adding points without *jittering* them will lead to non-informative clumping. To avoid this, use a jitter: `geom_jitter()`

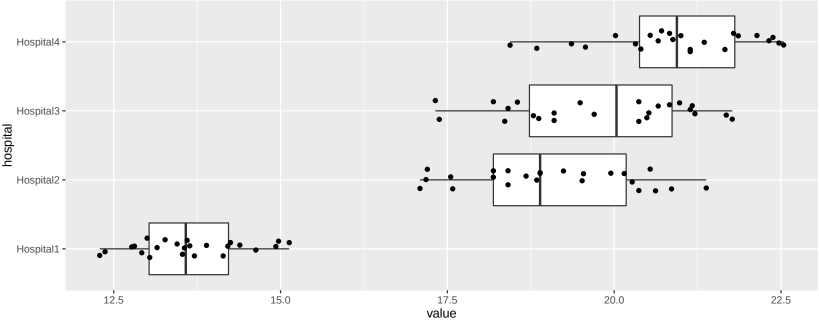
```r
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot() + geom_jitter(height = 0, width = 0.4)
```



Flipping My Box

The coordinate system can also change from being y-based to x-based via `coord_flip()`.

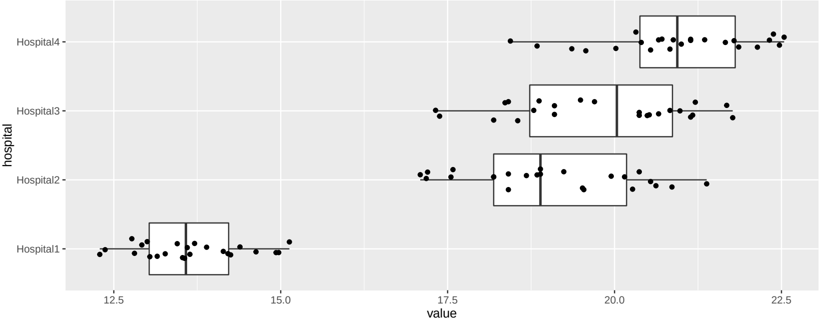
```
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot() + geom_jitter(height = 0, width = 0.4) +
  coord_flip()
```



Flipping My Box

The coordinate system can also change from being y-based to x-based via `coord_flip()`.

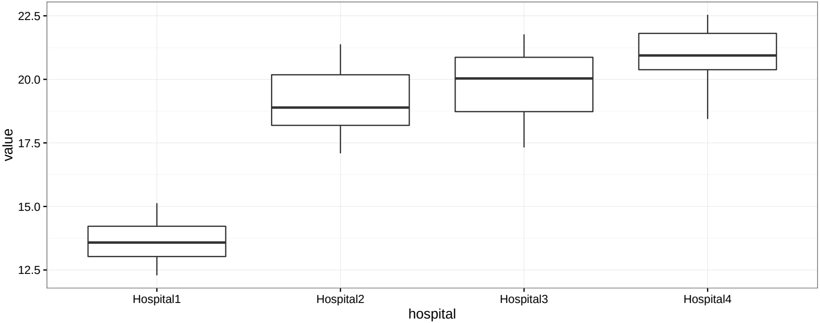
```
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot() + geom_jitter(height = 0, width = 0.4) +
  coord_flip()
```



Theming - Black & White

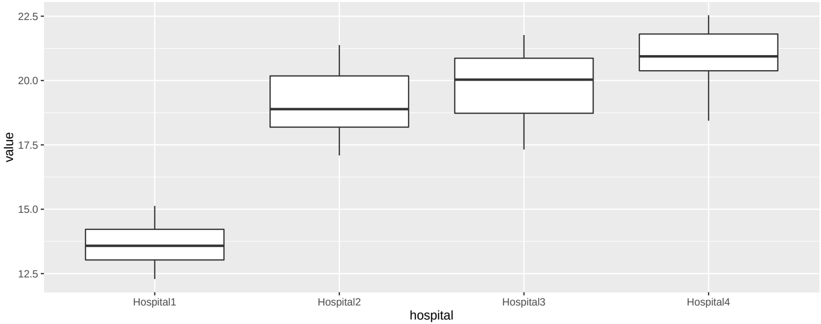
ggplot2 can easily switch to different color themes. By default, the theme_gray() is used. Some prefer the theme_bw().

```
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot()  + theme_bw()
```



Theming - Original

```r
ggplot(long_births, aes(x = hospital, y = value)) +
  geom_boxplot()
```



The main question:

> *Are you team `theme_gray()` or `theme_bw()`?*
>
> Twitter Question

Exercises

1. Load `sportsranks` from `msos` and transform it to long form. Make sure to add an indicator. Try to create boxplots of the different ratings.
2. Open `states` in the `msos` data set. Explore the different levels of school enrollment and crime. Try out other dimensions as well!
3. Last but not least, try to explore the `SAheart` data in `msos`.

Practical Comparison

For the next example, we are going to craft a Q-Q Plot in `Base R` and `ggplot2`. The Q-Q plot is normally used to check to see if the residuals of a model follow a normal distribution.

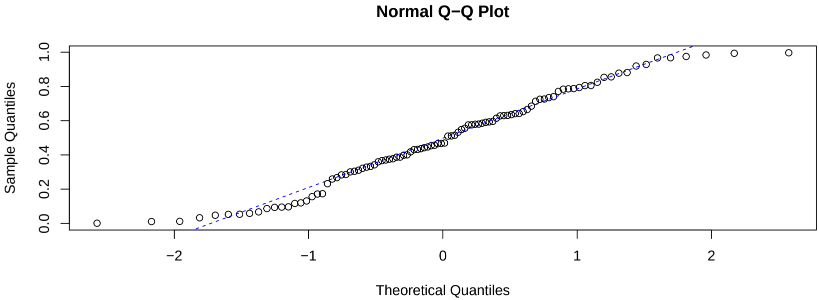
```r
# Set seed for reproducibility
set.seed(111)

# Generate data
x = runif(100,0,1)
```

Traditional q–q plot in R

Included in *Base R*, without any modification is `qqnorm()`, which provides the Q-Q Plot. Though, it is missing the traditional line connecting the first and third quartiles.

```r
qqnorm(x)                          # normal q-q plot
qqline(x,lty=2,col="blue") # line through the Q1 and Q3 qua
```



Crafting the q-q plot

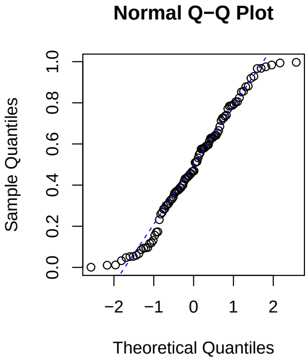
```
qqn = function(w) {
  n = length(w)
  nv = qnorm((1:n)/(n+1)) # Quantiles of Normal Dist.
  plot(nv, sort(w),       # X,Y
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles")
  title("Normal Q-Q Plot")
  m = (quantile(w,0.75)-quantile(w,0.25))/
      (qnorm(0.75)-qnorm(0.25))
  b = quantile(w,0.25) - m*qnorm(0.25)
  abline(b, m, lty=2, col="red") # Line through Q1 & Q3
}
```

Comparing Base implementations

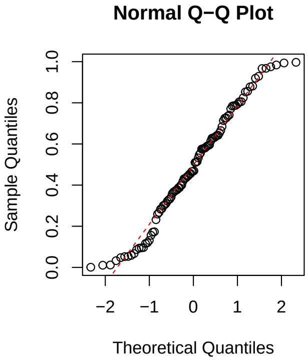
```
par(mfrow=c(1,2))       #  Two plots in one window
par(pty="s")            #  Square plots

qqnorm(x)               # Base R first
qqline(x,lty=2,col="blue")

qqn(x)                  # Our Plot
```



Normal Q−Q Plot

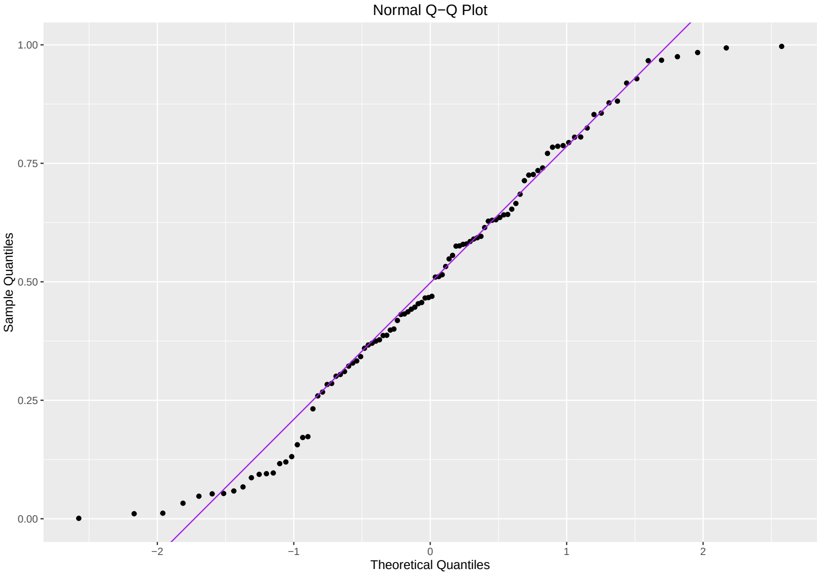


Normal Q−Q Plot

ggplot2 implementation

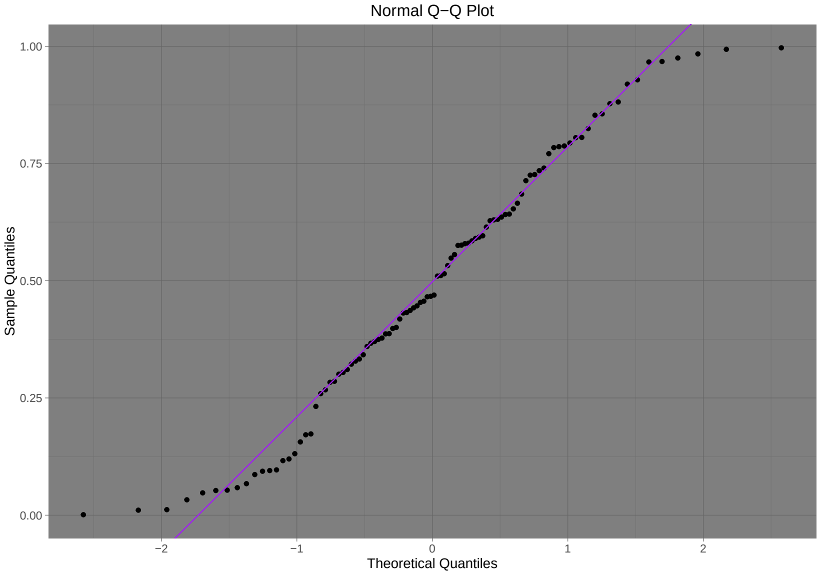
```
df_x = as.data.frame(x)
n = nrow(df_x)
m = (quantile(x,0.75)-quantile(x,0.25))/
      (qnorm(0.75)-qnorm(0.25))
b = quantile(x,0.25) - m*qnorm(0.25)
g = ggplot(df_x, aes(sample=x)) +
      stat_qq() +
      geom_abline(intercept = b,
                  slope = m,
                  color = "purple") +
      xlab("Theoretical Quantiles") +
      ylab("Sample Quantiles") +
      ggtitle("Normal Q-Q Plot")
```

ggplot2 implementation



ggplot2 implementation

```
g + theme_dark() # Welcome to the dark side!
```


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles