

# Data Visualization with GGPlot

## Fundamental Techniques in Data Science



**Utrecht  
University**

Kyle M. Lang

Department of Methodology & Statistics  
Utrecht University

# Outline

---

Grammar of Graphics

Geometries & Statistics

Styling & Aesthetics

Multi-panel Figures



# GGPlot

---

Base R graphics are fine for quick-and-dirty visualizations, but for publication quality graphics, you should probably use GGPlot.

- GGPlot uses the "grammar of graphics" and "tidy data" to build up a figure from modular components.

Describes all the non-data ink	<b>Theme</b>
Plotting space for the data	<b>Coordinates</b>
Statistical models & summaries	<b>Statistics</b>
Rows and columns of sub-plots	<b>Facets</b>
Shapes used to represent the data	<b>Geometries</b>
Scales onto which data is mapped	<b>Aesthetics</b>
The actual variables to be plotted	<b>Data</b>



# Basic Setup

---

We start by calling the `ggplot()` function to initialize the object that will define our plot.

- We must specify a data source.
- We must also give some aesthetic via the `aes()` function.

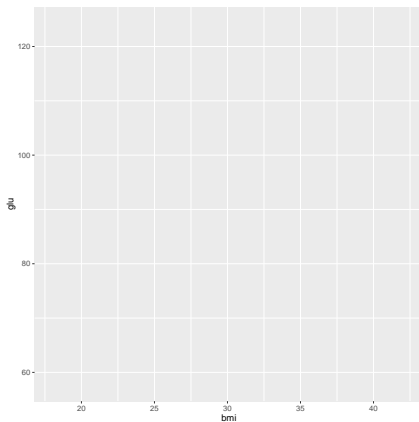
```
library(ggplot2)
p1 <- ggplot(data = diabetes, mapping = aes(x = bmi, y = glu))
```



# Basic Setup

At this point, our plot is pretty boring.

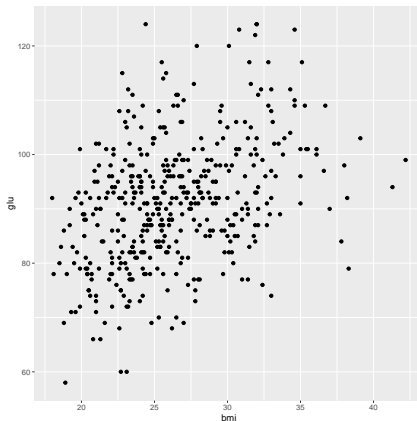
p1



# Geometries

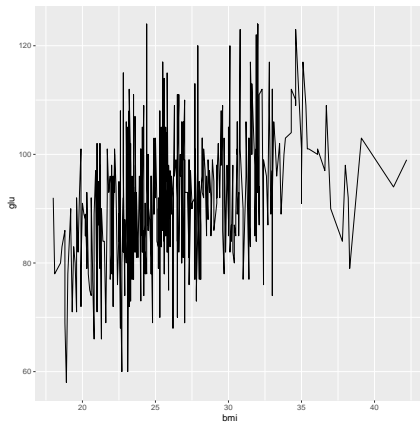
We need to define some geometry via a `geom_XXX()` function.

```
p1 + geom_point()
```



# Geometries

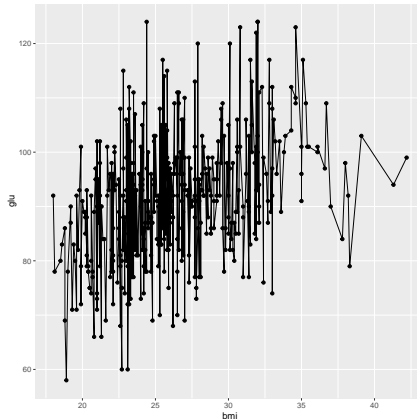
```
p1 + geom_line()
```



# Geometries

We can also combine different geometries into a single figure

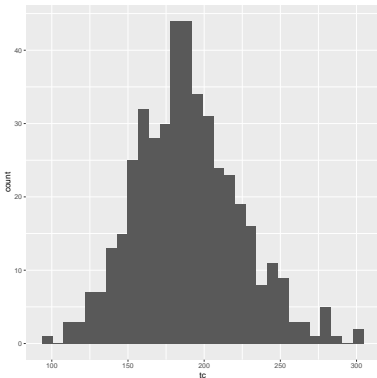
```
p1 + geom_point() + geom_line()
```



# Geometries

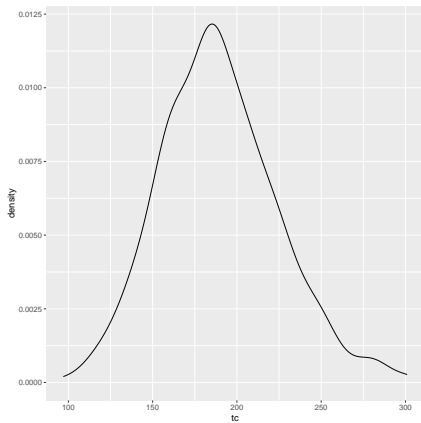
We can use different flavors of geometry for different types of data.

```
p2 <- ggplot(diabetes, aes(tc))  
p2 + geom_histogram()
```



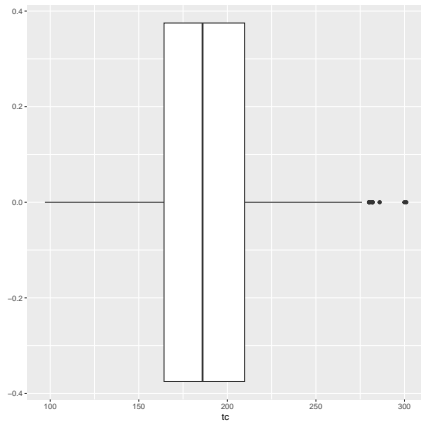
# Geometries

```
p2 + geom_density()
```



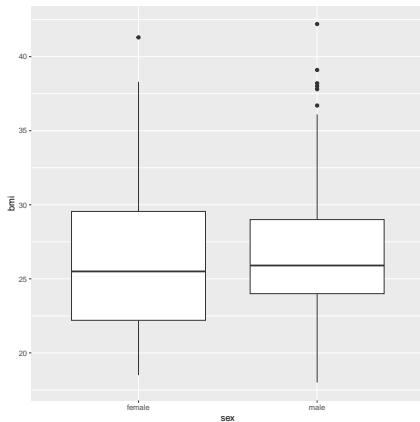
# Geometries

```
p2 + geom_boxplot()
```



# Geometries

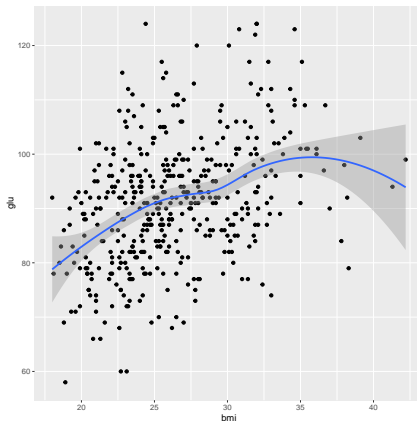
```
p3 <- ggplot(diabetes, aes(sex, bmi))  
p3 + geom_boxplot()
```



# Statistics

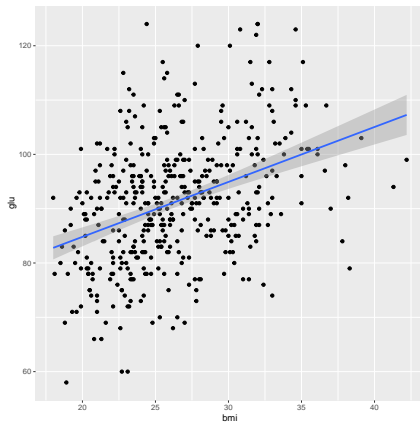
We can also add statistical summaries of the data.

```
p1 + geom_point() + geom_smooth()
```



# Statistics

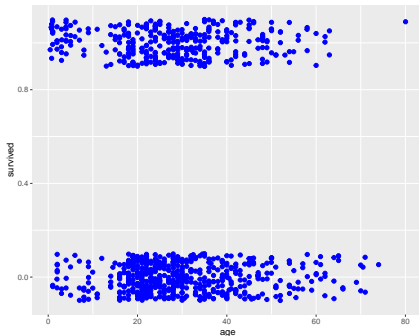
```
p1 + geom_point() + geom_smooth(method = "lm")
```



# Styling

Changing style options outside of the `aes()` function applies the styling to the entire plot.

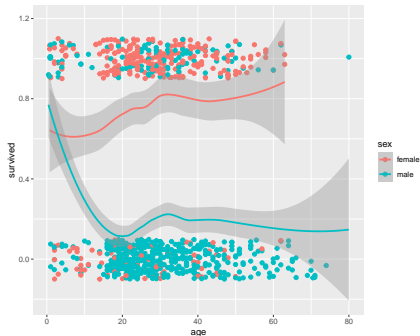
```
p5 <- ggplot(titanic, aes(age, survived))  
p5 + geom_jitter(color = "blue", size = 2, height = 0.1)
```



# Styling

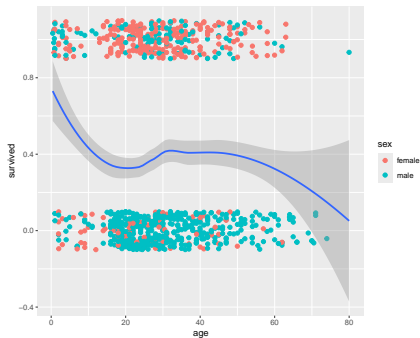
We can also apply styles as a function of variables by defining the style within the `aes()` function.

```
p6.1 <- ggplot(titanic, aes(age, survived, color = sex))  
p6.1 + geom_jitter(size = 2, height = 0.1) + geom_smooth()
```



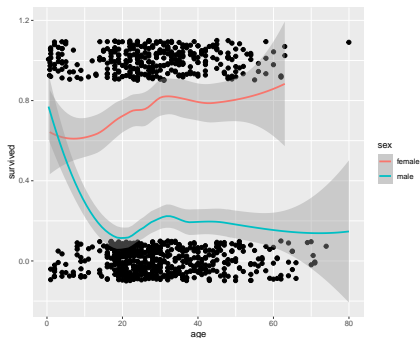
# Styling

```
p6.2 <- ggplot(titanic, aes(age, survived))  
p6.2 + geom_jitter(aes(color = sex), size = 2, height = 0.1) +  
  geom_smooth()
```



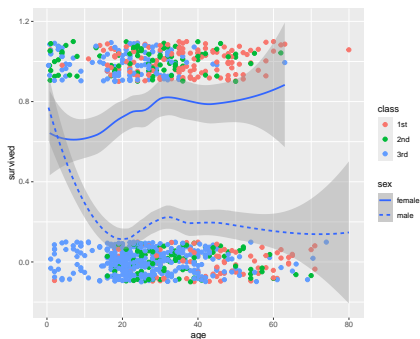
# Styling

```
p6.2 + geom_jitter(size = 2, height = 0.1) +  
  geom_smooth(aes(color = sex))
```



# Styling

```
p6.2 + geom_jitter(aes(color = class), size = 2, height = 0.1) +  
  geom_smooth(aes(linetype = sex))
```



# Themes

---

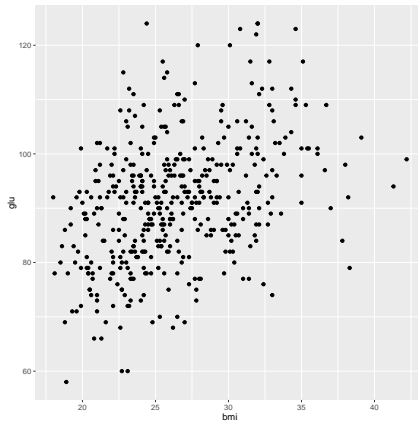
We can apply canned themes to adjust a plot's overall appearance.

```
p1.1 <- p1 + geom_point()  
p1.1
```



# Themes

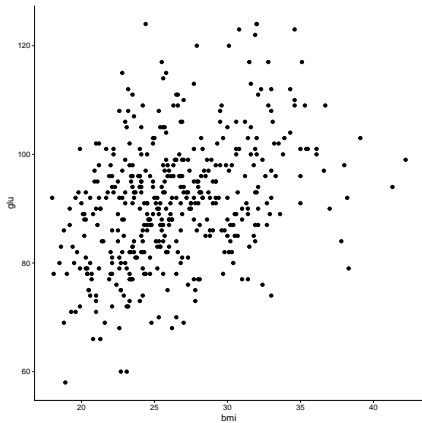
---



# Themes

---

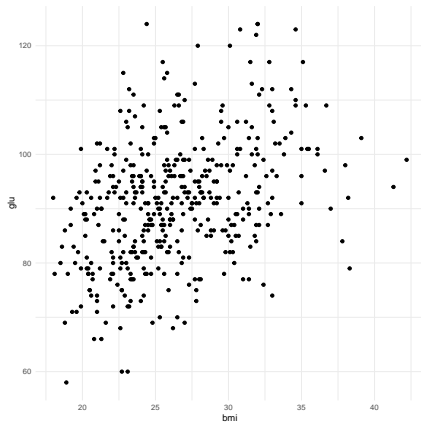
```
p1.1 + theme_classic()
```



# Themes

---

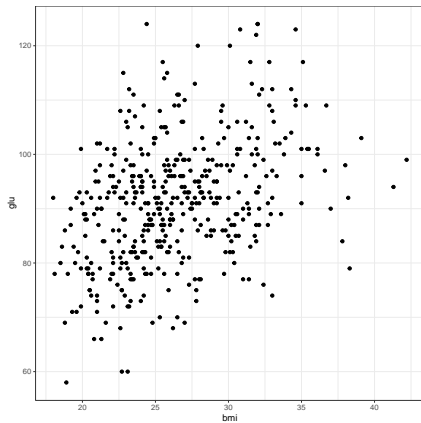
```
p1.1 + theme_minimal()
```



# Themes

---

```
p1.1 + theme_bw()
```



# Themes

---

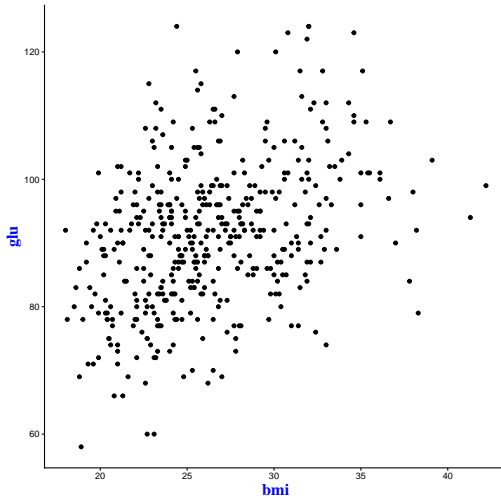
We can also modify individual theme elements.

```
p1.1 + theme_classic() +  
  theme(axis.title = element_text(size = 16,  
                                   family = "serif",  
                                   face = "bold",  
                                   color = "blue"),  
        aspect.ratio = 1)
```



# Themes

---



# Facets

---

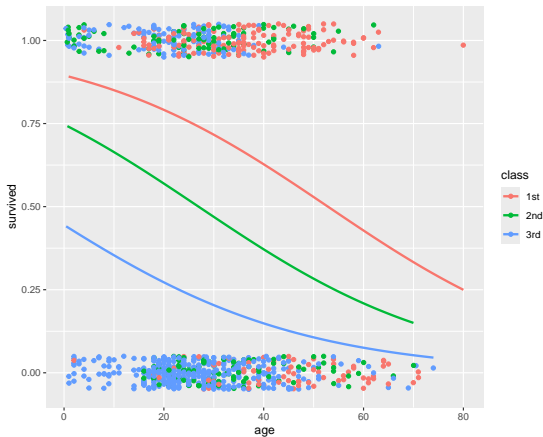
Faceting allow us to make arrays of conditional plots.

```
p7 <- ggplot(titanic, aes(age, survived, color = class)) +  
  geom_jitter(height = 0.05) +  
  geom_smooth(method = "glm",  
             method.args = list(family = "binomial"),  
             se = FALSE)
```

p7

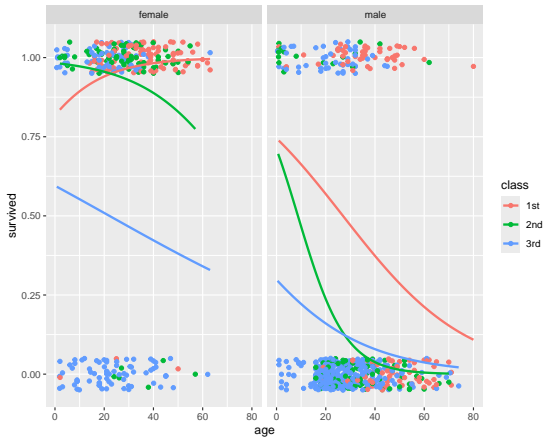


# Facets



# Facets

```
p7 + facet_wrap(vars(sex))
```



# Facets

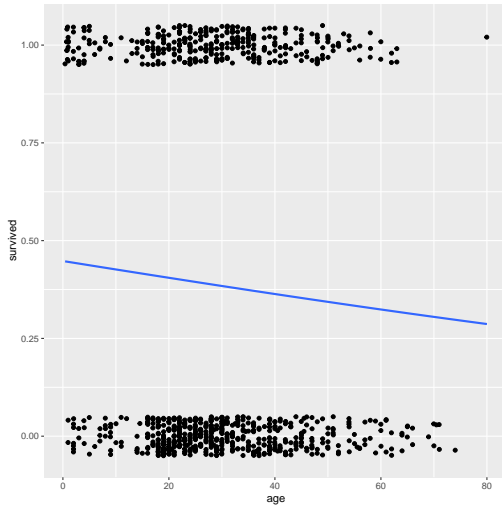
---

```
p8 <- ggplot(titanic, aes(age, survived)) +  
  geom_jitter(height = 0.05) +  
  geom_smooth(method = "glm",  
             method.args = list(family = "binomial"),  
             se = FALSE)  
p8
```



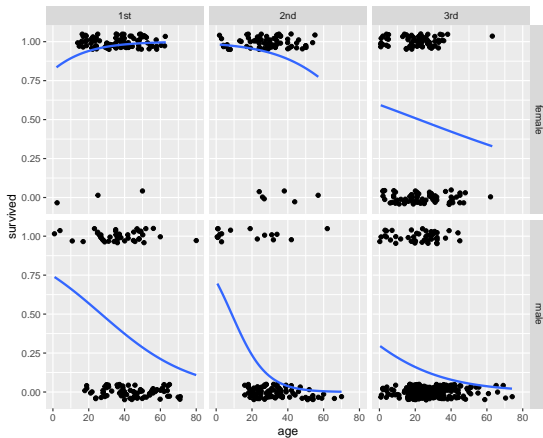
# Facets

---



# Facets

```
p8 + facet_grid(vars(sex), vars(class))
```



# Joining Multiple Figures

---

If we want to paste several different plots into a single figure (without faceting), we can use the utilities in the **gridExtra** package.

```
library(gridExtra)

grid.arrange(
  p1 + geom_point(),
  p3 + geom_boxplot(),
  p1 + geom_line(),
  p8 + facet_grid(vars(sex), vars(class)),
  ncol = 2
)
```

# Joining Multiple Figures

