

$$\mathbf{x} = \mathbf{x}_f - \mathbf{x}_i \quad \Delta \mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$$

$$= \frac{\Delta \vec{r}}{\Delta t} \quad \vec{a} = \frac{\Delta \vec{v}}{\Delta t}$$

$$v = |\mathbf{v}| = \sqrt{v_x^2 + v_y^2}$$

$$\theta = \tan^{-1}\left(\frac{v_y}{v_x}\right)$$



$$\omega = \frac{\Delta \theta}{\Delta t} \quad \alpha = \frac{\Delta \omega}{\Delta t}$$

$$= \mathbf{v}_0 + \mathbf{a}t$$

$$= \mathbf{x}_0 + \mathbf{v}_0 t + \frac{1}{2} \mathbf{a} t^2$$

$$-v_o^2 = 2a(x - x_0)$$

$$= \frac{v_f^2 - v_i^2}{2a}$$

# Data Visualization

$$\omega = 2\pi f \quad f = \frac{1}{T}$$

$$\omega = \omega_0 + \alpha t$$

$$\theta = \theta_0 + \omega_0 t + \frac{1}{2} \alpha t^2$$

$$\omega_o^2 = 2\alpha(\theta - \theta_0)$$

Curvefit Rankings for 0.5% SEV CO<sub>2</sub>

Rank	F-statistic
1	7.962883557
2	7.8601110639
3	7.5283512645
4	7.3357010958
5	6.3801158367
6	3.8079206858
7	3.742891358
8	3.5727028219
9	3.5546937052
10	3.0133372321
11	2.7408796673
12	2.6986270263
13	2.5801276758
14	2.5622800357
15	2.1798855221

**Prof. Dr. Javier Valdes**  
[Javier.valdes@th-deg.de](mailto:Javier.valdes@th-deg.de)

IAI - Institut für Angewandte Informatik  
 Institute for Applied Informatics

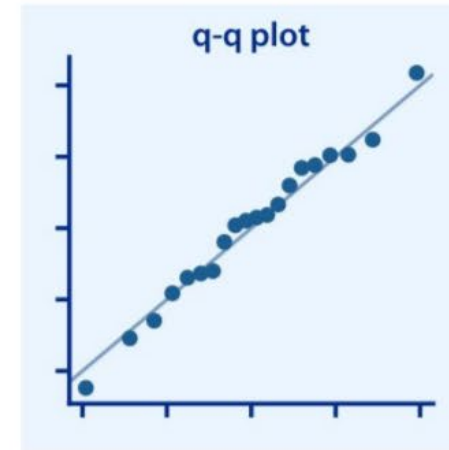
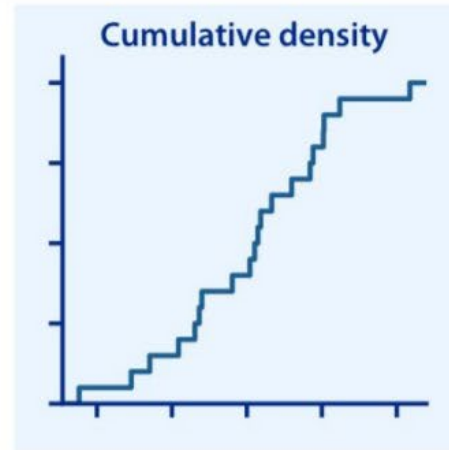
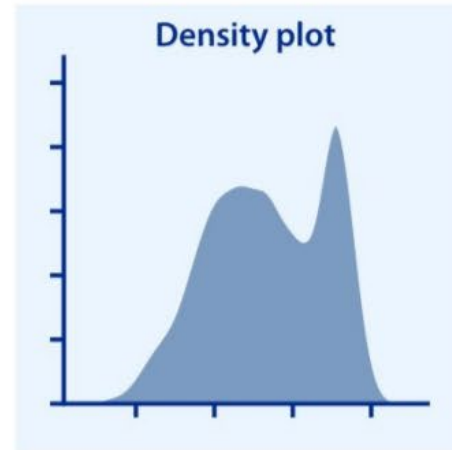
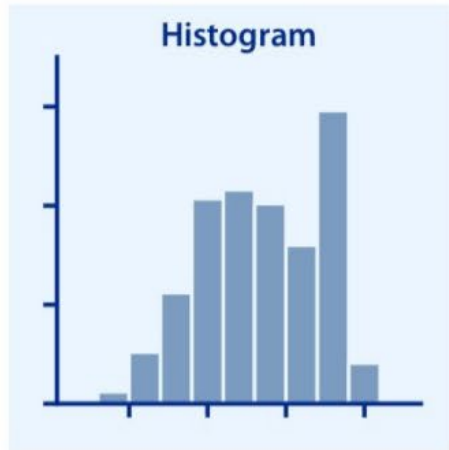
Technische Hochschule Deggendorf  
 Technologie Campus Freyung  
 Grafenauer Str. 22, D- 94078 Freyung,  
 Germany

Tel.: +49 8551 91764 40  
 Fax: +49 8551 91764 69

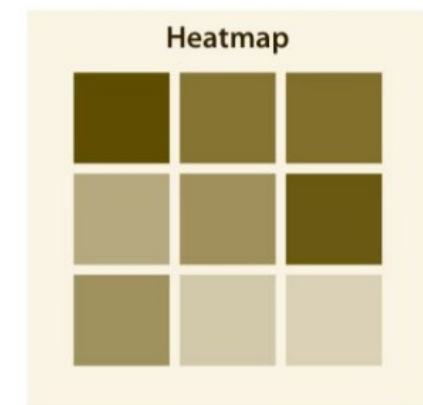
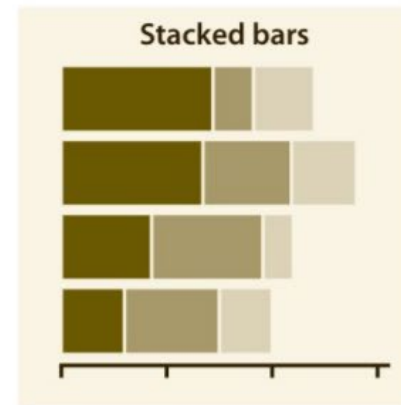
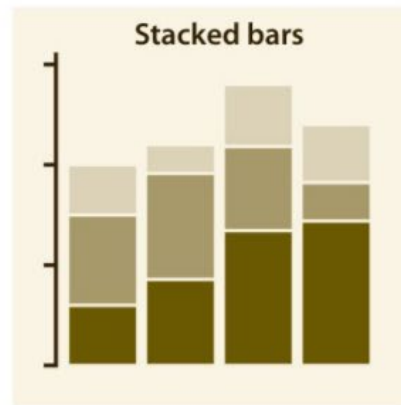
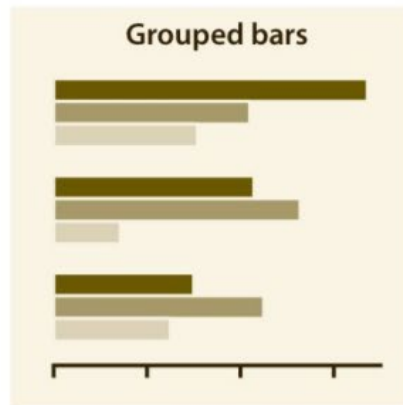
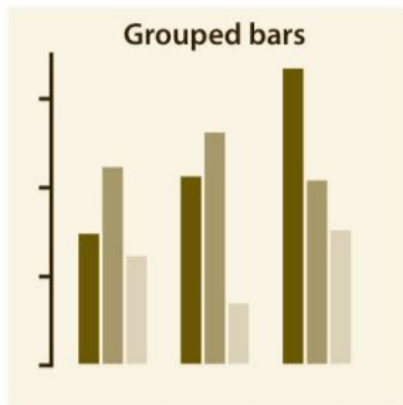
$$x = A \cos(\omega t + \phi) \quad v = -A \omega \sin(\omega t + \phi)$$



## Distributions

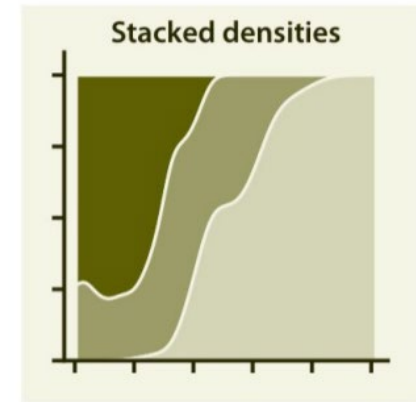
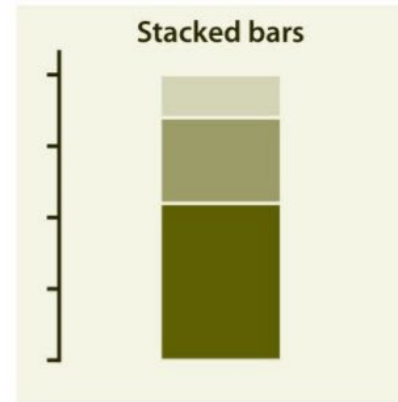
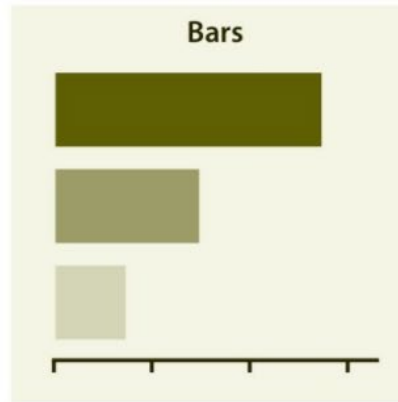
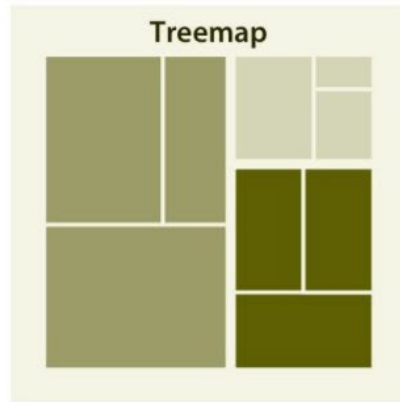
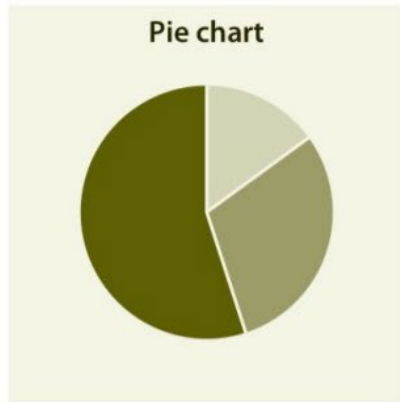


## Comparisons

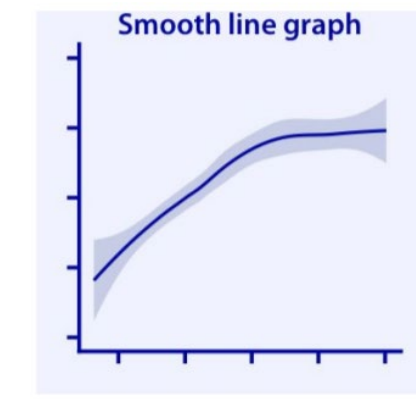
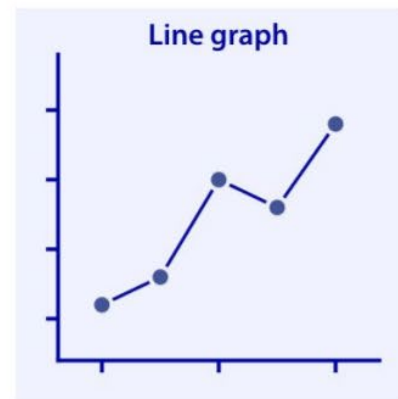
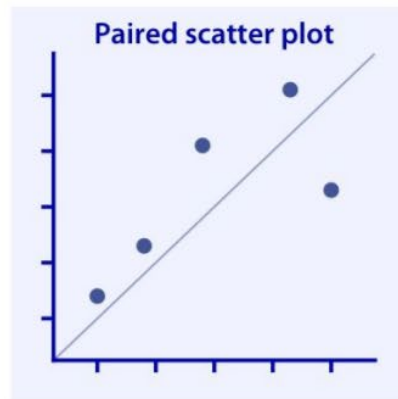
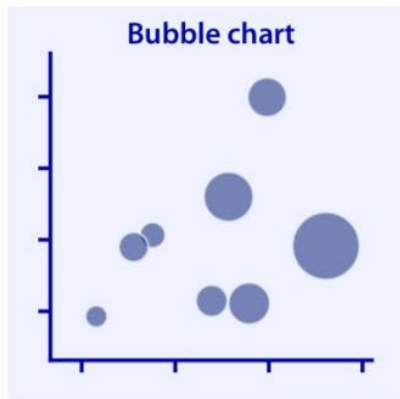
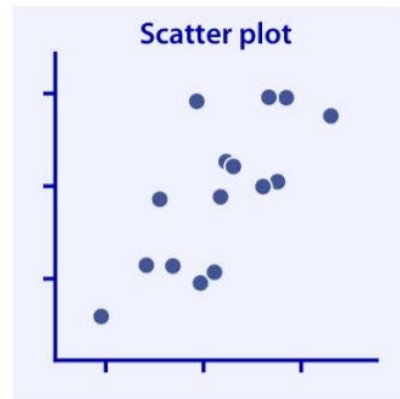




## Proportions



## x-y relationships





# Plotting with ggplot2

The **concept** behind **ggplot2** divides plot into three (even more) different **fundamental** parts:  
Plot = data + Aesthetics + Geometry

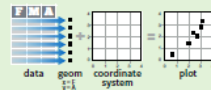
- **data** is a data frame
- **Aesthetics** is used to indicate x and y variables. It can also be used to control the **color**, the **size** or the **shape** of points, the height of bars, etc.....
- **Geometry** defines the type of graphics (histogram, box plot, line plot, density plot, dot plot, ....)
- All the rest....

## Data Visualization with ggplot2 Cheat Sheet

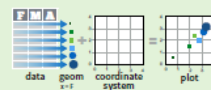


### Basics

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



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```

ggplot(data = <DATA>) +
  <GEOM_FUNCTION>
  mapping = aes(<MAPPINGS>),
  stat = <STAT>,
  position = <POSITION>
  +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
  
```

Required  
Not required, sensible defaults supplied

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

Begins a plot that you finish by adding layers to. Add one geom function per layer.

```
aesthetic mappings | data | geom
```

```
qplot(x = cty, y = hwy, data = mpg, geom = "point")
```

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

```
last_plot()
```

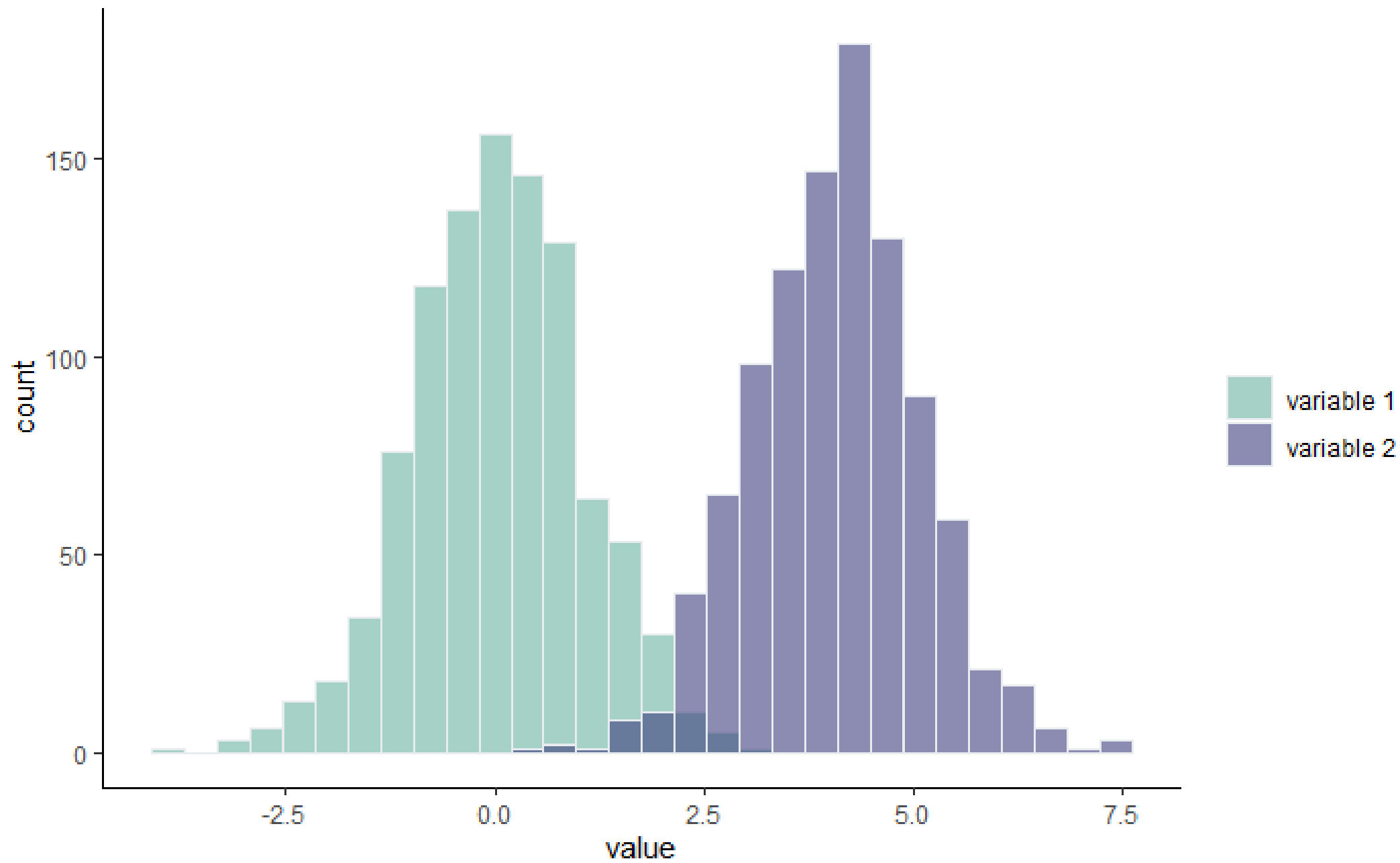
Returns the last plot

```
ggsave("plot.png", width = 5, height = 5)
```

Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

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

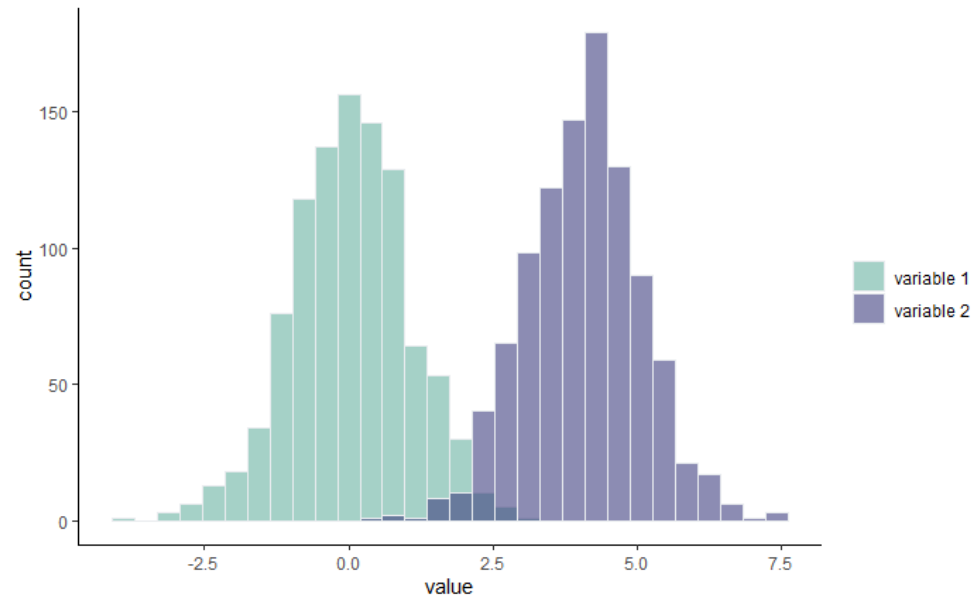
Graphical Primitives	Two Variables
<p><b>a</b> &lt;- ggplot(economics, aes(date, unemployment))</p> <p><b>b</b> &lt;- ggplot(seals, aes(x = long, y = lat))</p> <p>(Useful for expanding limits)</p> <p><b>a</b> + <b>geom_blank()</b></p> <p><b>b</b> + <b>geom_curve</b>(aes(yend = lat + 1, xend = long + 1, curvature = z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size</p> <p><b>a</b> + <b>geom_path</b>(lineend = "butt", linejoin = "round", linemitre = 1)</p> <p>x, y, alpha, color, group, linetype, size</p> <p><b>a</b> + <b>geom_polygon</b>(aes(group = group))</p> <p>x, y, alpha, color, fill, group, linetype, size</p> <p><b>b</b> + <b>geom_rect</b>(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmin, xmax, ymax, ymin, alpha, color, fill, linetype, size</p> <p><b>a</b> + <b>geom_ribbon</b>(aes(ymin = unemployment - 900, ymax = unemployment + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size</p>	<p><b>Continuous X, Continuous Y</b></p> <p><b>e</b> &lt;- ggplot(mpg, aes(cty, hwy))</p> <p><b>e</b> + <b>geom_label</b>(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)</p> <p>x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust</p> <p><b>e</b> + <b>geom_jitter</b>(height = 2, width = 2)</p> <p>x, y, alpha, color, fill, shape, size</p> <p><b>e</b> + <b>geom_point()</b></p> <p>x, y, alpha, color, fill, shape, size, stroke</p> <p><b>e</b> + <b>geom_quantile()</b></p> <p>x, y, alpha, color, group, linetype, size, weight</p> <p><b>e</b> + <b>geom_rug</b>(sides = "bl")</p> <p>x, y, alpha, color, linetype, size</p> <p><b>e</b> + <b>geom_smooth</b>(method = lm)</p> <p>x, y, alpha, color, fill, group, linetype, size, weight</p> <p><b>e</b> + <b>geom_text</b>(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)</p> <p>x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust</p>
<p><b>Line Segments</b></p> <p>common aesthetics: x, y, alpha, color, linetype, size</p> <p><b>b</b> + <b>geom_abline</b>(aes(intercept = 0, slope = 1))</p> <p><b>b</b> + <b>geom_hline</b>(aes(yintercept = lat))</p> <p><b>b</b> + <b>geom_vline</b>(aes(xintercept = long))</p> <p><b>b</b> + <b>geom_segment</b>(aes(yend = lat + 1, xend = long + 1))</p> <p><b>b</b> + <b>geom_spoke</b>(aes(angle = 1:1155, radius = 1))</p>	<p><b>Continuous Bivariate Distribution</b></p> <p><b>h</b> &lt;- ggplot(diamonds, aes(carat, price))</p> <p><b>h</b> + <b>geom_bin2d</b>(binwidth = c(0.25, 500))</p> <p>x, y, alpha, color, fill, linetype, size, weight</p> <p><b>h</b> + <b>geom_density2d()</b></p> <p>x, y, alpha, colour, group, linetype, size</p> <p><b>h</b> + <b>geom_hex()</b></p> <p>x, y, alpha, colour, fill, size</p> <p><b>Continuous Function</b></p> <p><b>i</b> &lt;- ggplot(economics, aes(date, unemployment))</p> <p><b>i</b> + <b>geom_area()</b></p> <p>x, y, alpha, color, fill, linetype, size</p> <p><b>i</b> + <b>geom_line()</b></p> <p>x, y, alpha, color, group, linetype, size</p> <p><b>i</b> + <b>geom_step</b>(direction = "hv")</p> <p>x, y, alpha, color, group, linetype, size</p>
<p><b>One Variable</b></p> <p><b>Continuous</b></p> <p><b>c</b> &lt;- ggplot(mpg, aes(hwy)); <b>c2</b> &lt;- ggplot(mpg)</p> <p><b>c</b> + <b>geom_area</b>(stat = "bin")</p> <p>x, y, alpha, color, fill, linetype, size</p> <p><b>c</b> + <b>geom_density</b>(kernel = "gaussian")</p> <p>x, y, alpha, color, fill, group, linetype, size, weight</p> <p><b>c</b> + <b>geom_dotplot()</b></p> <p>x, y, alpha, color, fill</p> <p><b>c</b> + <b>geom_freqpoly()</b></p> <p>x, y, alpha, color, group, linetype, size</p> <p><b>c</b> + <b>geom_histogram</b>(binwidth = 5)</p> <p>x, y, alpha, color, fill, linetype, size, weight</p> <p><b>c2</b> + <b>geom_qq</b>(aes(sample = hwy))</p> <p>x, y, alpha, color, fill, linetype, size, weight</p> <p><b>Discrete</b></p> <p><b>d</b> &lt;- ggplot(mpg, aes(fill))</p> <p><b>d</b> + <b>geom_bar()</b></p> <p>x, alpha, color, fill, linetype, size, weight</p>	<p><b>Discrete X, Continuous Y</b></p> <p><b>f</b> &lt;- ggplot(mpg, aes(class, hwy))</p> <p><b>f</b> + <b>geom_col()</b></p> <p>x, y, alpha, color, fill, group, linetype, size</p> <p><b>f</b> + <b>geom_boxplot()</b></p> <p>x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight</p> <p><b>f</b> + <b>geom_dotplot</b>(binaxis = "y", stackdir = "center")</p> <p>x, y, alpha, color, fill, group</p> <p><b>f</b> + <b>geom_violin</b>(scale = "area")</p> <p>x, y, alpha, color, fill, group, linetype, size, weight</p> <p><b>Visualizing error</b></p> <p><b>df</b> &lt;- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)</p> <p><b>j</b> &lt;- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))</p> <p><b>j</b> + <b>geom_crossbar</b>(fatten = 2)</p> <p>x, y, ymax, ymin, alpha, color, fill, group, linetype, size</p> <p><b>j</b> + <b>geom_errorbar()</b></p> <p>x, ymax, ymin, alpha, color, group, linetype, size, width (also <b>geom_errorbarh()</b>)</p> <p><b>j</b> + <b>geom_linerange()</b></p> <p>x, ymin, ymax, alpha, color, group, linetype, size</p> <p><b>j</b> + <b>geom_pointrange()</b></p> <p>x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size</p> <p><b>Maps</b></p> <p><b>k</b> &lt;- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests)))</p> <p><b>map</b> &lt;- map_data("state")</p> <p><b>k</b> &lt;- ggplot(data, aes(fill = murder))</p> <p><b>k</b> + <b>geom_map</b>(aes(map_id = state), map = map) + <b>expand_limits</b>(x = map\$long, y = map\$lat)</p> <p>map_id, alpha, color, fill, linetype, size</p>
<p><b>Three Variables</b></p> <p><b>seals</b> &lt;- with(seals, sqrt(delta_long^2 + delta_lat^2))</p> <p><b>l</b> &lt;- ggplot(seals, aes(long, lat))</p> <p><b>l</b> + <b>geom_raster</b>(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)</p> <p>x, y, alpha, fill</p> <p><b>l</b> + <b>geom_tile</b>(aes(fill = z))</p> <p>x, y, alpha, color, fill, linetype, size, width</p>	<p><b>Discrete X, Discrete Y</b></p> <p><b>g</b> &lt;- ggplot(diamonds, aes(cut, color))</p> <p><b>g</b> + <b>geom_count()</b></p> <p>x, y, alpha, color, fill, shape, size, stroke</p> <p><b>l</b> + <b>geom_contour</b>(aes(z = z))</p> <p>x, y, z, alpha, colour, group, linetype, size, weight</p>



# Example

```
data <- data.frame(  
  type = c( rep("variable 1", 1000), rep("variable 2", 1000) ),  
  value = c( rnorm(1000), rnorm(1000, mean=4) ) ) # Represent it
```

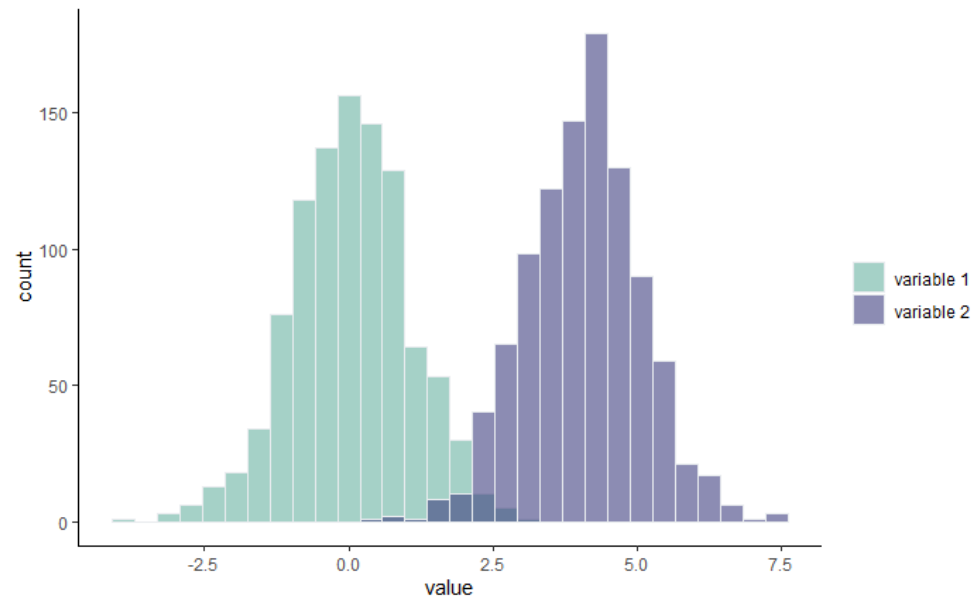
```
ggplot(data, aes(x=value, fill=type)) +  
  geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +  
  scale_fill_manual(values=c("#69b3a2", "#404080")) + theme_classic()  
+ labs(fill="")
```



1. Generate a dataframe with two variables and 1000 observations
2. Generate random values based on a normal distribution

# Example

```
data <- data.frame(  
  type = c( rep("variable 1", 1000), rep("variable 2", 1000) ),  
  value = c( rnorm(1000), rnorm(1000, mean=4) ) ) # Represent it  
  
ggplot(data, aes(x=value, fill=type)) +  
  geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +  
  scale_fill_manual(values=c("#69b3a2", "#404080")) + theme_classic()  
+ labs(fill="")
```

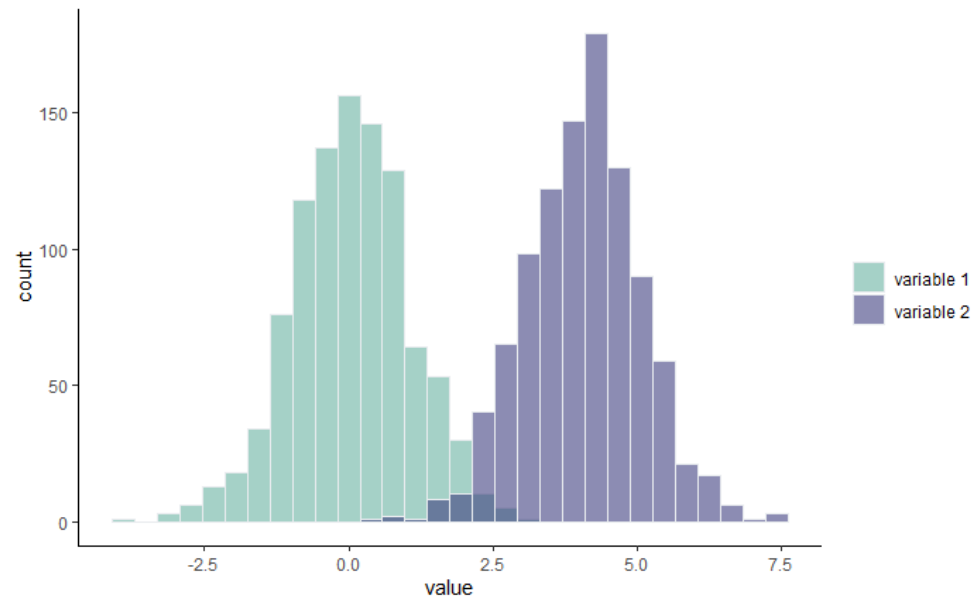


1. Generate a dataframe with two variables and 1000 observations
2. Generate random values based on a normal distribution
3. Select the data and variables we want to plot



# Example

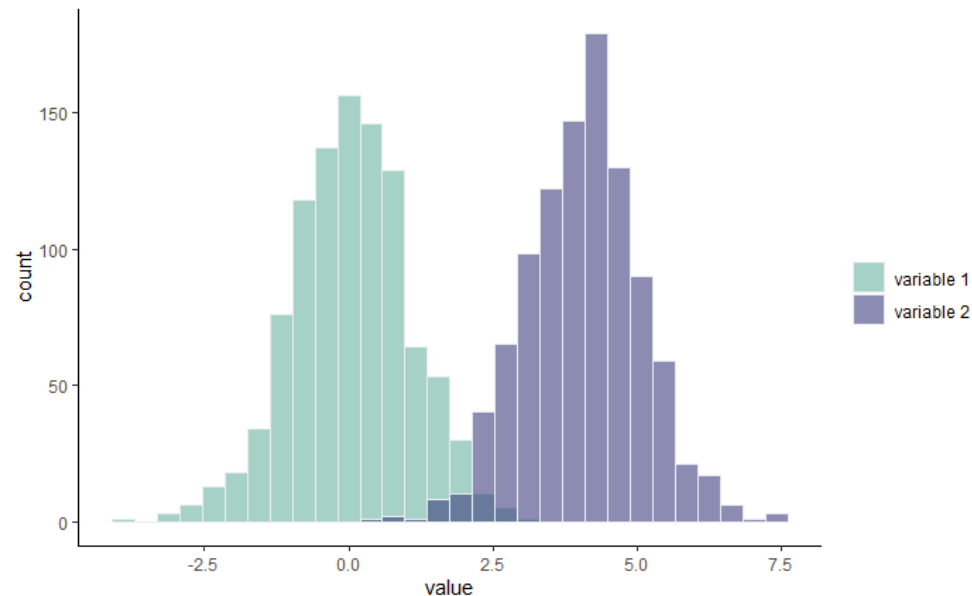
```
data <- data.frame(  
  type = c( rep("variable 1", 1000), rep("variable 2", 1000) ),  
  value = c( rnorm(1000), rnorm(1000, mean=4) ) ) # Represent it  
  
ggplot(data, aes(x=value, fill=type)) +  
  geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +  
  scale_fill_manual(values=c("#69b3a2", "#404080")) + theme_classic()  
+ labs(fill="")
```



1. Generate a dataframe with two variables and 1000 observations
2. Generate random values based on a normal distribution
3. Select the data and variables we want to plot
4. Select the type of visualization, identify, color sets the lines, alpha for transparency and identify overlaps the bar

# Example

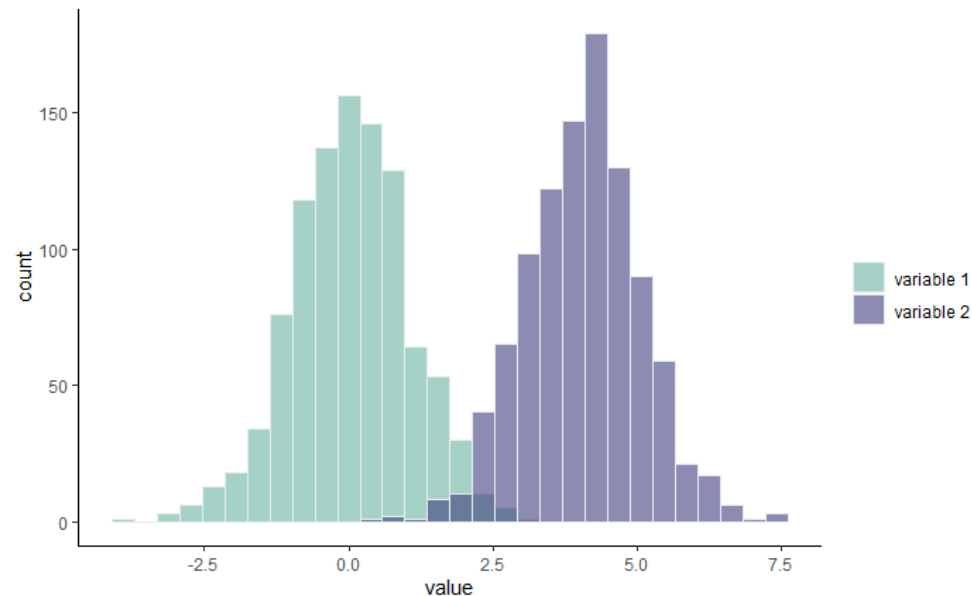
```
data <- data.frame(  
  type = c( rep("variable 1", 1000), rep("variable 2", 1000) ),  
  value = c( rnorm(1000), rnorm(1000, mean=4) ) ) # Represent it  
  
ggplot(data, aes(x=value, fill=type)) +  
  geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +  
  scale_fill_manual(values=c("#69b3a2", "#404080")) + theme_classic()  
  + labs(fill="")
```



1. Generate a dataframe with two variables and 1000 observations
2. Generate random values based on a normal distribution
3. Select the data and variables we want to plot
4. Select the type of visualization, identify, color sets the lines, alpha for transparency and identify overlaps the bar
5. We fill each histogram with a different colour

# Example

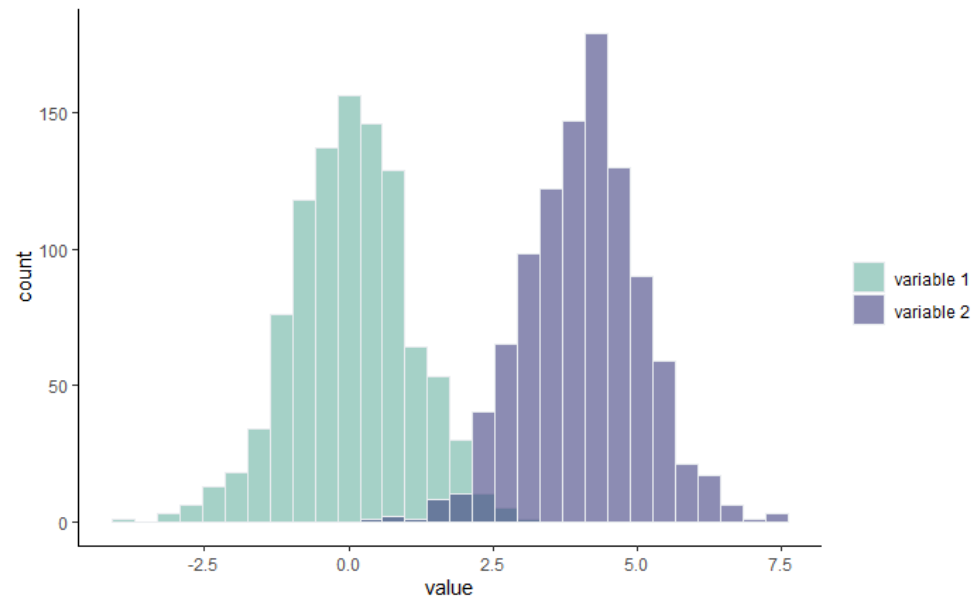
```
data <- data.frame(  
  type = c( rep("variable 1", 1000), rep("variable 2", 1000) ),  
  value = c( rnorm(1000), rnorm(1000, mean=4) ) ) # Represent it  
  
ggplot(data, aes(x=value, fill=type)) +  
  geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +  
  scale_fill_manual(values=c("#69b3a2", "#404080")) + theme classic()  
  + labs(fill="")
```



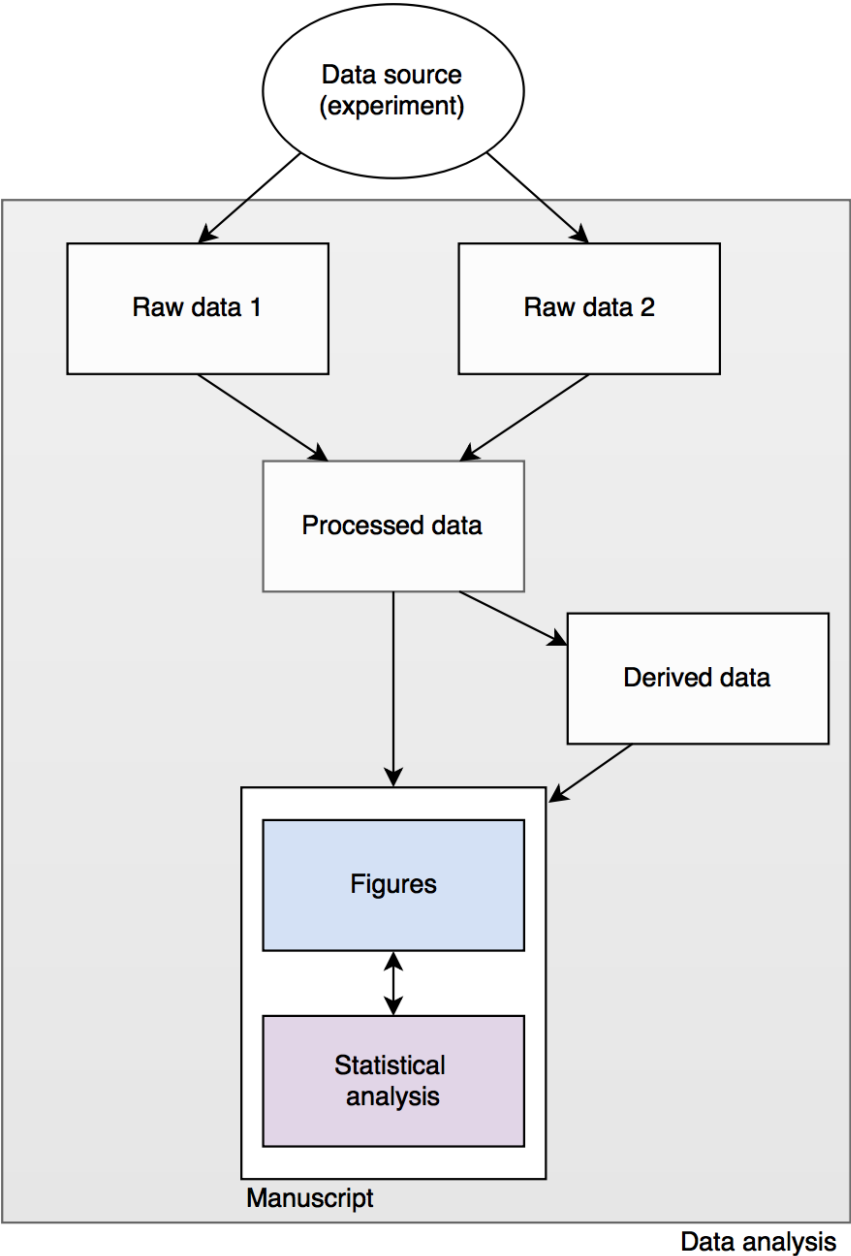
1. Generate a dataframe with two variables and 1000 observations
2. Generate random values based on a normal distribution
3. Select the data and variables we want to plot
4. Select the type of visualization, identify, color sets the lines, alpha for transparency and identify overlaps the bar
5. We fill each histogram with a different colour
6. Choose a preset theme

# Example

```
data <- data.frame(  
  type = c( rep("variable 1", 1000), rep("variable 2", 1000) ),  
  value = c( rnorm(1000), rnorm(1000, mean=4) ) ) # Represent it  
  
ggplot(data, aes(x=value, fill=type)) +  
  geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +  
  scale_fill_manual(values=c("#69b3a2", "#404080")) + theme_classic()  
+ labs(fill="")
```



1. Generate a dataframe with two variables and 1000 observations
2. Generate random values based on a normal distribution
3. Select the data and variables we want to plot
4. Select the type of visualization, identify, color sets the lines, alpha for transparency and identify overlaps the bar
5. We fill each histogram with a different colour
6. Choose a preset theme
7. Names for labs added



# Univariate Graphs



Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

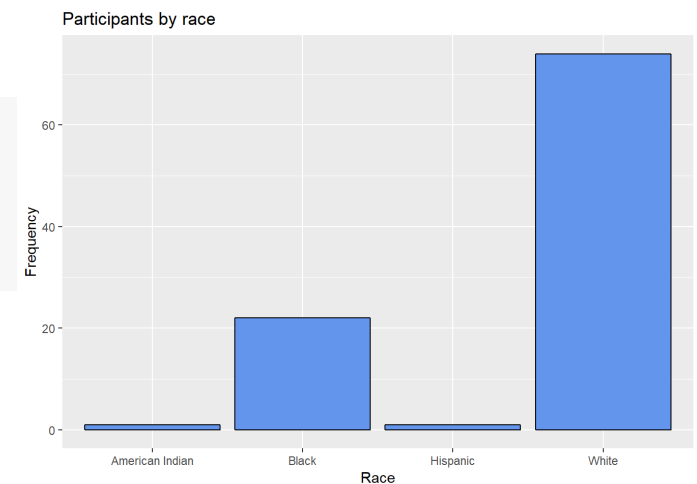
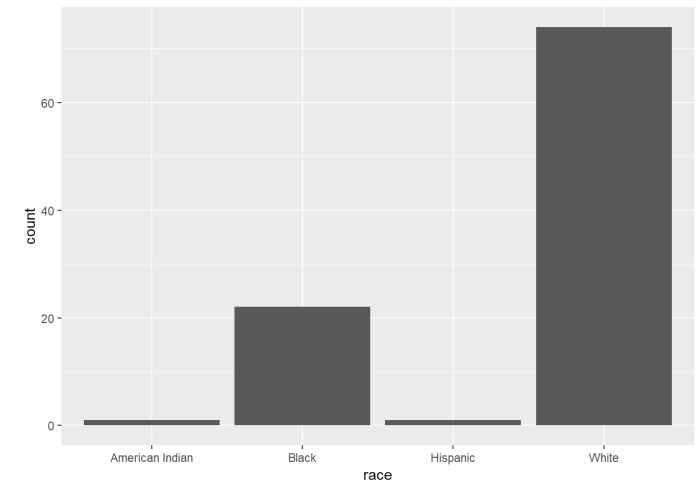
- **Categorical**
- Quantitative

```
library(ggplot2)
```

```
data(Marriage, package = "mosaicData") # plot the distribution of race
ggplot(Marriage, aes(x = race)) +
  geom_bar()
```

```
# plot the distribution of race with modified colors and labels
```

```
ggplot(Marriage, aes(x = race)) +
  geom_bar(fill = "cornflowerblue", color="black") +
  labs(x = "Race", y = "Frequency", title = "Participants by race")
```



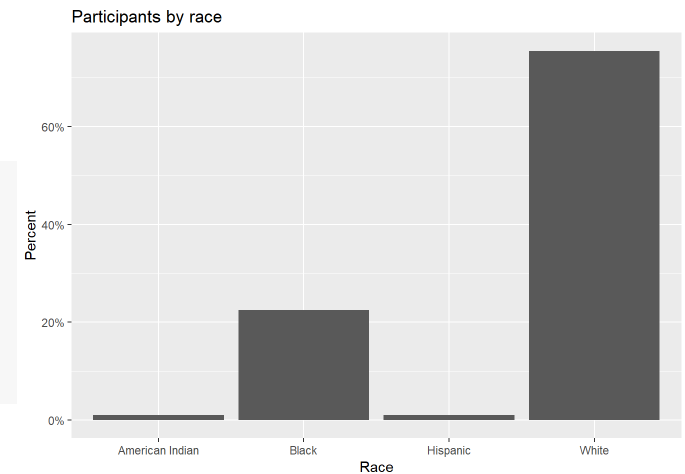
# Univariate Graphs



Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

## Percents

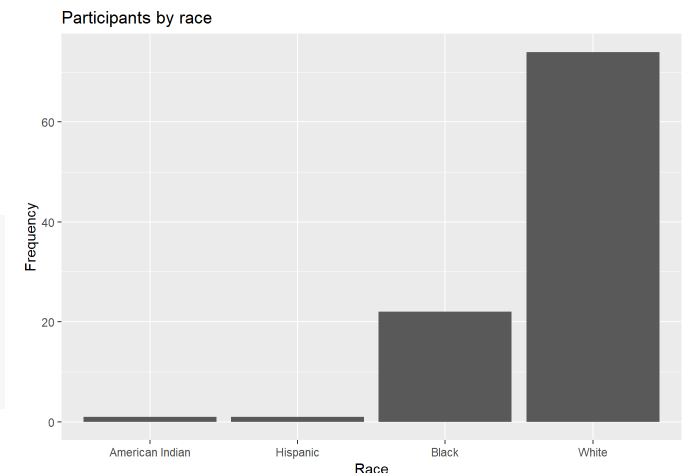
```
# plot the distribution as percentages  
ggplot(Marriage, aes(x = race, y = ..count.. / sum(..count..))) +  
  geom_bar() +  
  labs(x = "Race", y = "Percent", title = "Participants by race") +  
  scale_y_continuous(labels = scales::percent)
```



## Sorting categories

```
# calculate number of participants in # each race category  
library(dplyr)  
plotdata <- Marriage %>% count(race)
```

```
# plot the bars in ascending order  
ggplot(plotdata, aes(x = reorder(race, n), y = n)) +  
  geom_bar(stat = "identity") +  
  labs(x = "Race", y = "Frequency", title = "Participants by race")
```



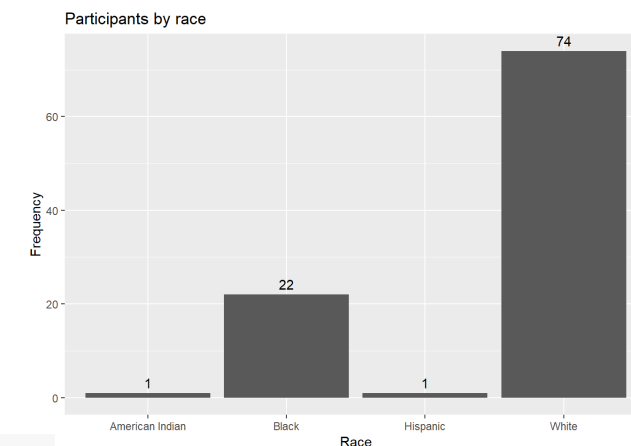
# Univariate Graphs



Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

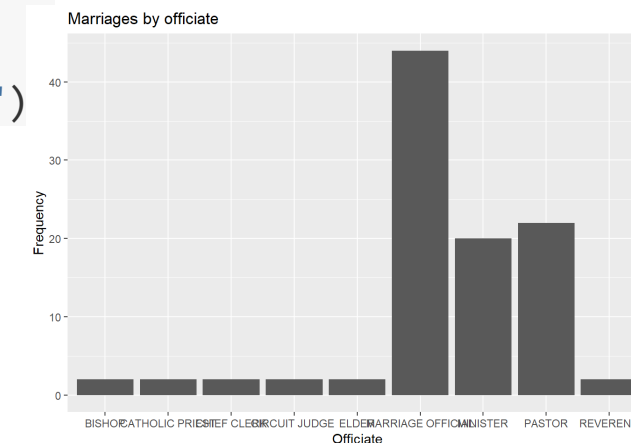
## Labeling

```
# plot the bars with numeric labels
ggplot(plotdata, aes(x = race, y = n)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = n), vjust=-0.5) +
  labs(x = "Race", y = "Frequency", title = "Participants by race")
```



## Overlapping labels

```
# basic bar chart with overlapping labels
ggplot(Marriage, aes(x = officialTitle)) +
  geom_bar() +
  labs(x = "Officiate", y = "Frequency", title = "Marriages by officiate")
```



```
# horizontal bar chart
ggplot(Marriage, aes(x = officialTitle)) +
  geom_bar() +
  labs(x = "", y = "Frequency", title = "Marriages by officiate") +
  coord_flip()
```



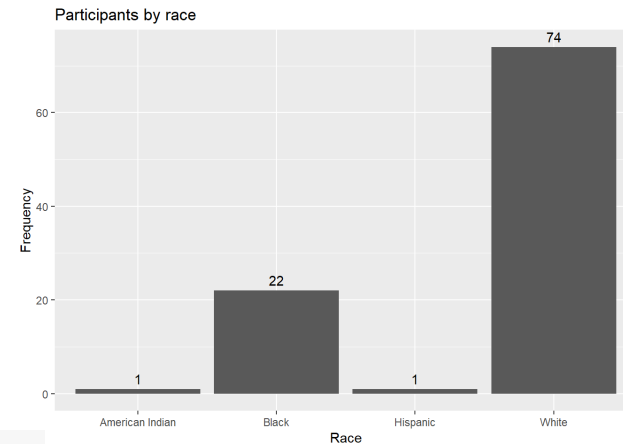
# Univariate Graphs



Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

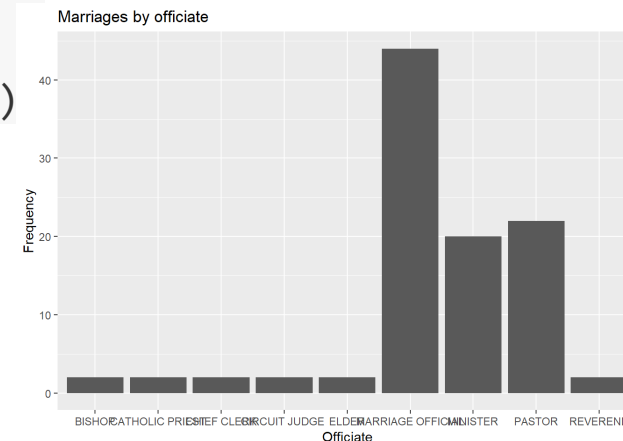
## Labeling

```
# plot the bars with numeric labels
ggplot(plotdata, aes(x = race, y = n)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = n), vjust=-0.5) +
  labs(x = "Race", y = "Frequency", title = "Participants by race")
```



## Overlapping labels

```
# basic bar chart with overlapping labels
ggplot(Marriage, aes(x = officialTitle)) +
  geom_bar() +
  labs(x = "Officiate", y = "Frequency", title = "Marriages by officiate")
```



```
# horizontal bar chart
ggplot(Marriage, aes(x = officialTitle)) +
  geom_bar() +
  labs(x = "", y = "Frequency", title = "Marriages by officiate") +
  coord_flip()
```

# Univariate Graphs



Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

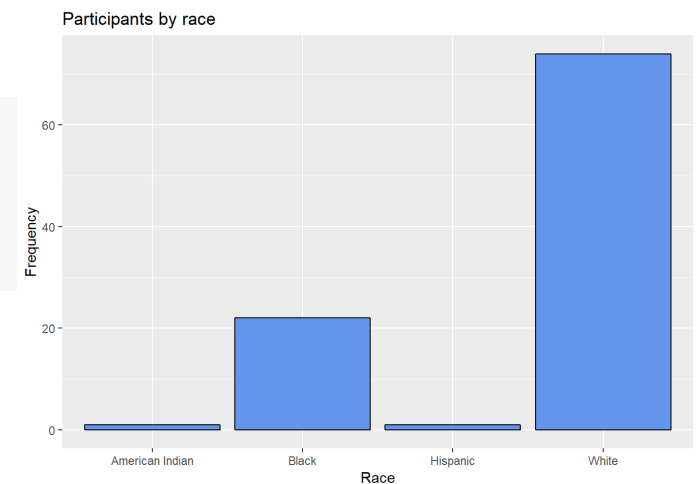
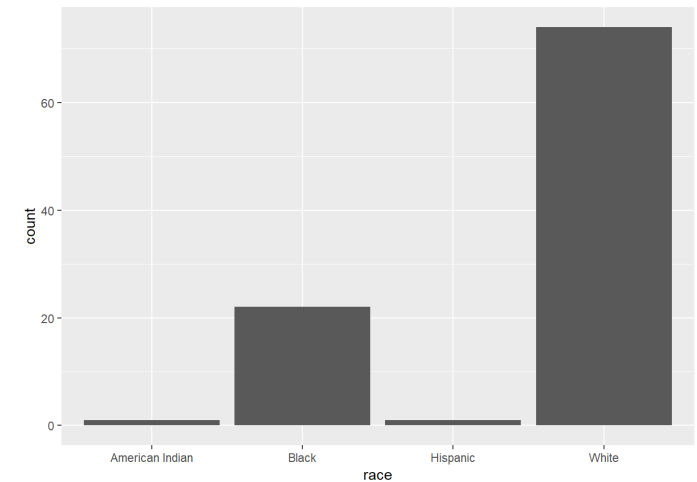
- Categorical
- Quantitative

```
library(ggplot2)
```

```
data(Marriage, package = "mosaicData") # plot the distribution of race
ggplot(Marriage, aes(x = race)) +
  geom_bar()
```

```
# plot the distribution of race with modified colors and labels
```

```
ggplot(Marriage, aes(x = race)) +
  geom_bar(fill = "cornflowerblue", color="black") +
  labs(x = "Race", y = "Frequency", title = "Participants by race")
```



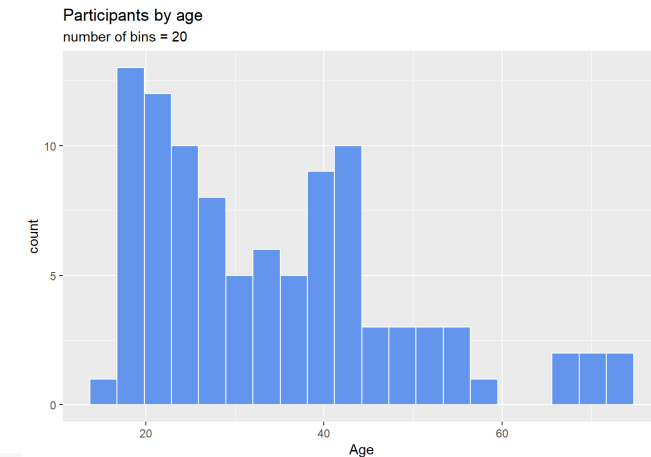
# Univariate Graphs



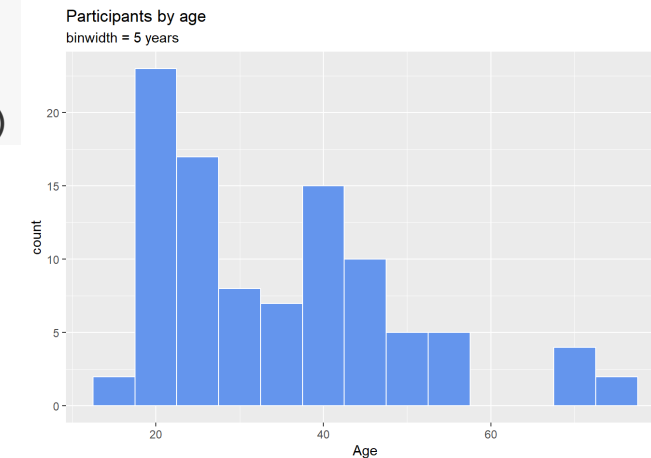
Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

- Categorical
- **Quantitative**

```
library(ggplot2) # plot the age distribution using a histogram
ggplot(Marriage, aes(x = age)) +
  geom_histogram() +
  labs(title = "Participants by age", x = "Age")
```



```
# plot the histogram with a binwidth of 5
ggplot(Marriage, aes(x = age)) +
  geom_histogram(fill = "cornflowerblue", color = "white", binwidth = 5) +
  labs(title="Participants by age", subtitle = "binwidth = 5 years", x = "Age")
```



# Univariate Graphs

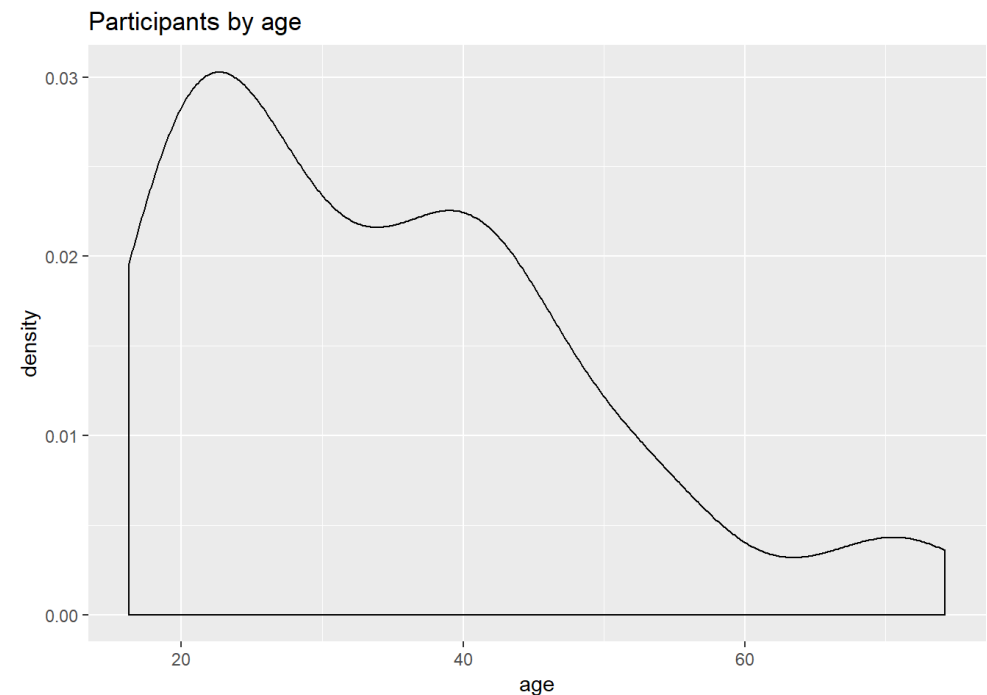


Univariate graphs plot the distribution of data from a single variable. The variable can be categorical (e.g., race, sex) or quantitative (e.g., age, weight).

- Categorical
- **Quantitative**

Computes and draws kernel density estimate, which is a smoothed version of the histogram. This is a useful alternative to the histogram for continuous data that comes from an underlying smooth distribution.

(What??) Basically, we are trying to draw a smoothed histogram, where the area under the curve equals one.



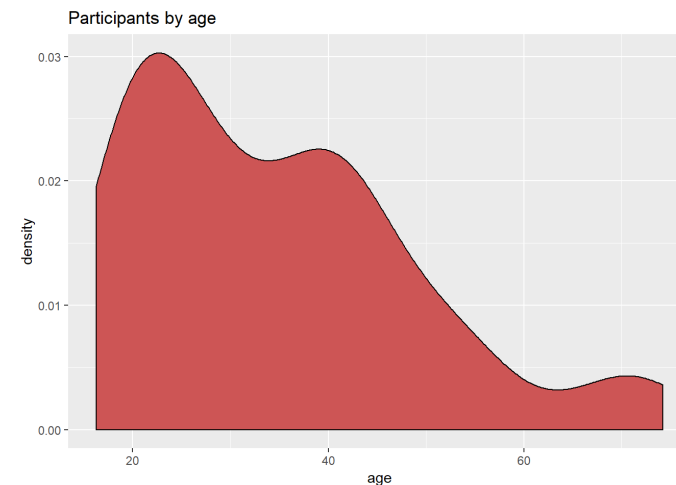
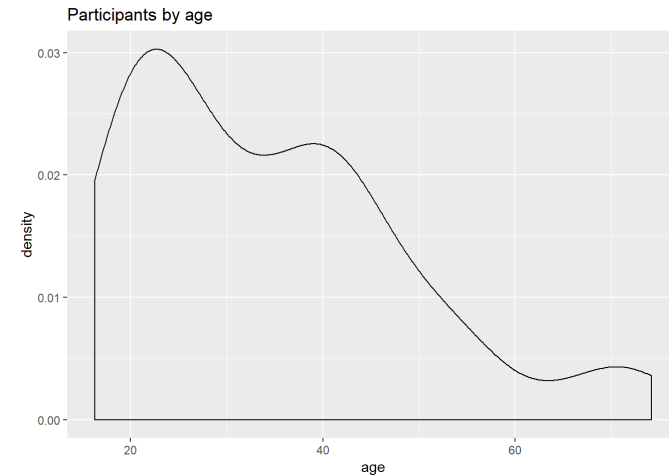
# Univariate Graphs



Smoothing parameter

```
# Create a kernel density plot of age  
ggplot(Marriage, aes(x = age)) +  
  geom_density() +  
  labs(title = "Participants by age")
```

```
# Create a kernel density plot of age  
ggplot(Marriage, aes(x = age)) +  
  geom_density(fill = "indianred3") +  
  labs(title = "Participants by age")
```



# Bivariate Graphs

## Categorical vs. Categorical



Stacked bar chart

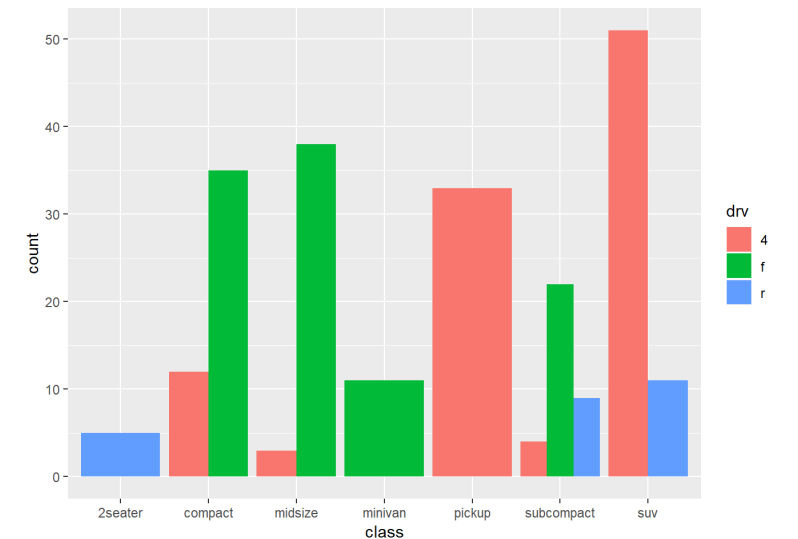
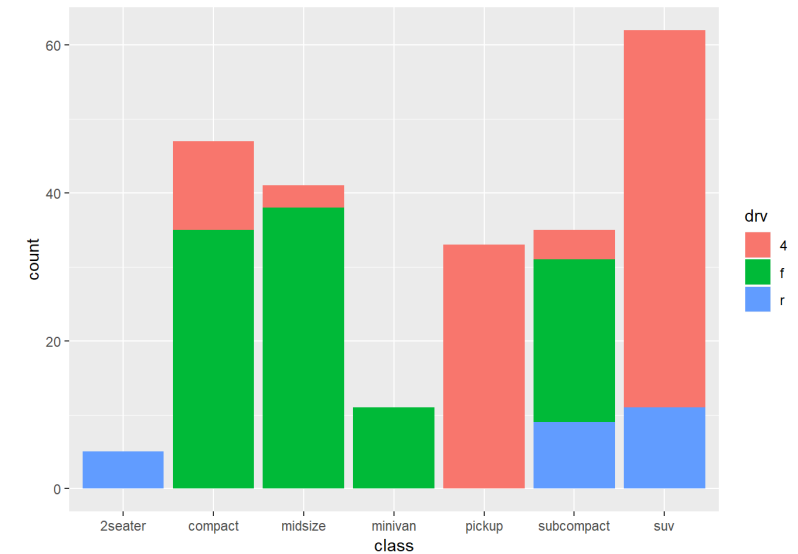
```
library(ggplot2)

# stacked bar chart

ggplot(mpg, aes(x = class, fill = drv)) +
  geom_bar(position = "stack")
```

*# grouped bar plot*

```
ggplot(mpg, aes(x = class, fill = drv)) +
  geom_bar(position = "dodge")
```



# Bivariate Graphs



Categorical vs. Categorical

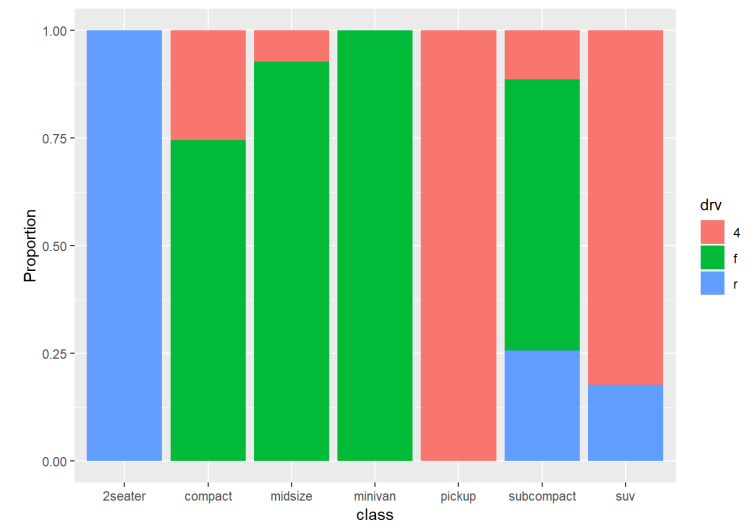
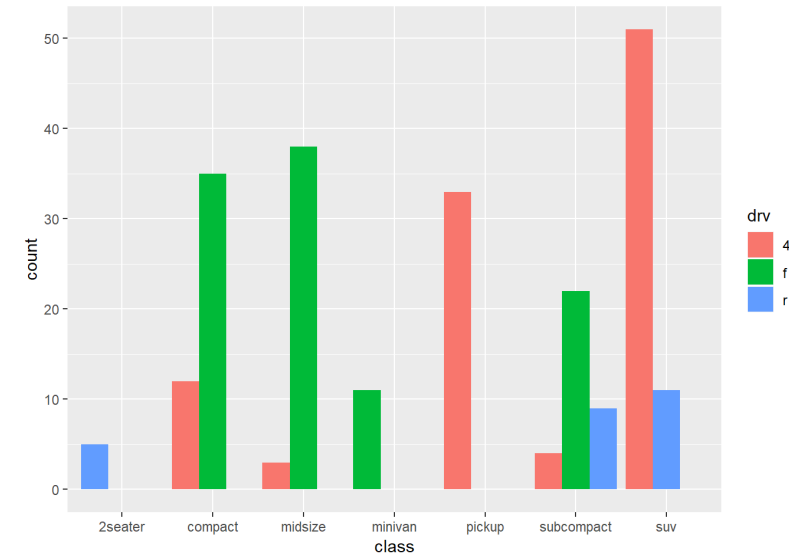
Stacked bar chart

```
# grouped bar plot preserving zero count bars
```

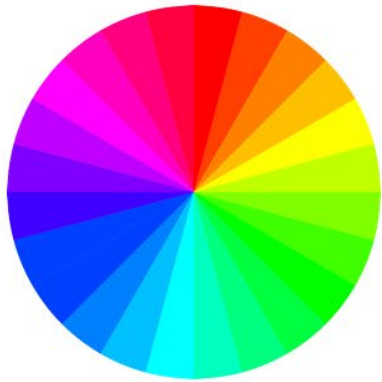
```
ggplot(mpg, aes(x = class, fill = drv)) +  
  geom_bar(position = position_dodge(preserve = "single"))
```

```
# bar plot, with each bar representing 100%
```

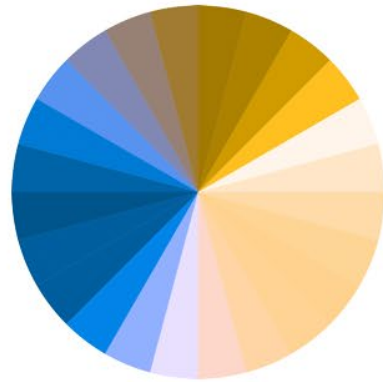
```
ggplot(mpg, aes(x = class, fill = drv)) +  
  geom_bar(position = "fill") +  
  labs(y = "Proportion")
```



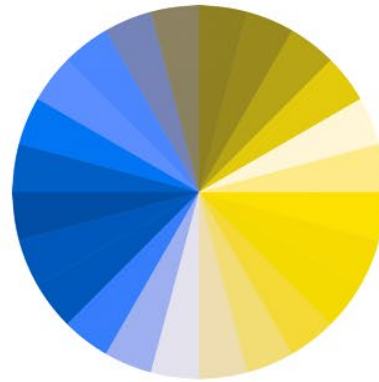
# Effective communication through visualization



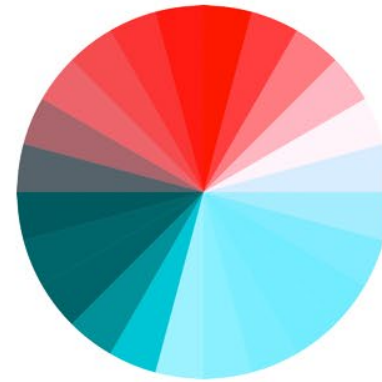
Regular vision



Deuteranopia



Protanopia



Tritanopia



Monochromacy





# Bivariate Graphs



Categorical vs. Categorical

## Improving the color and labeling

You can use additional options to improve color and labeling.

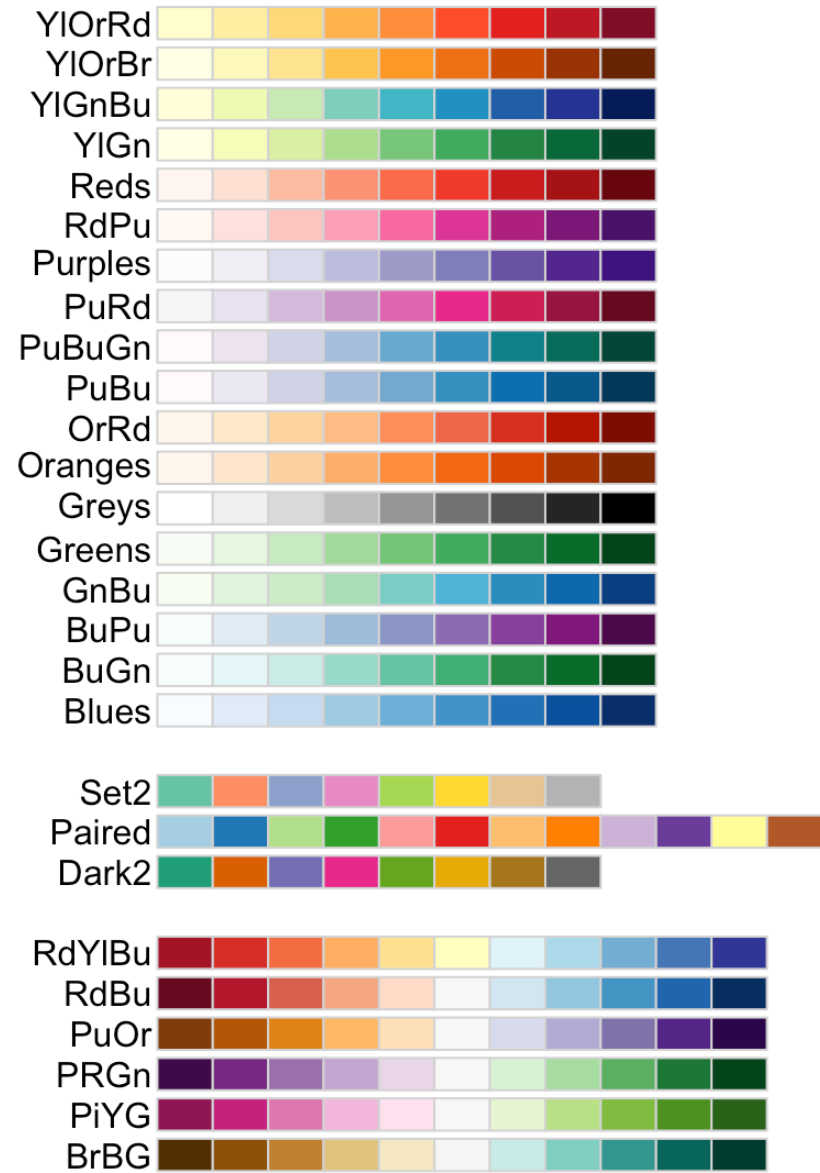
- **factor** modifies the order of the categories for the class variable and both the order and the labels for the drive variable
- **scale\_y\_continuous** modifies the y-axis tick mark labels
- **labs** provides a title and changed the labels for the x and y axes and the legend
- **scale\_fill\_brewer** changes the fill color scheme
- **theme\_minimal** removes the grey background and changed the grid color



# Bivariate Graphs

Categorical vs. Categorical

**Improving the color and labeling**



# Bivariate Graphs

Categorical vs. Categorical

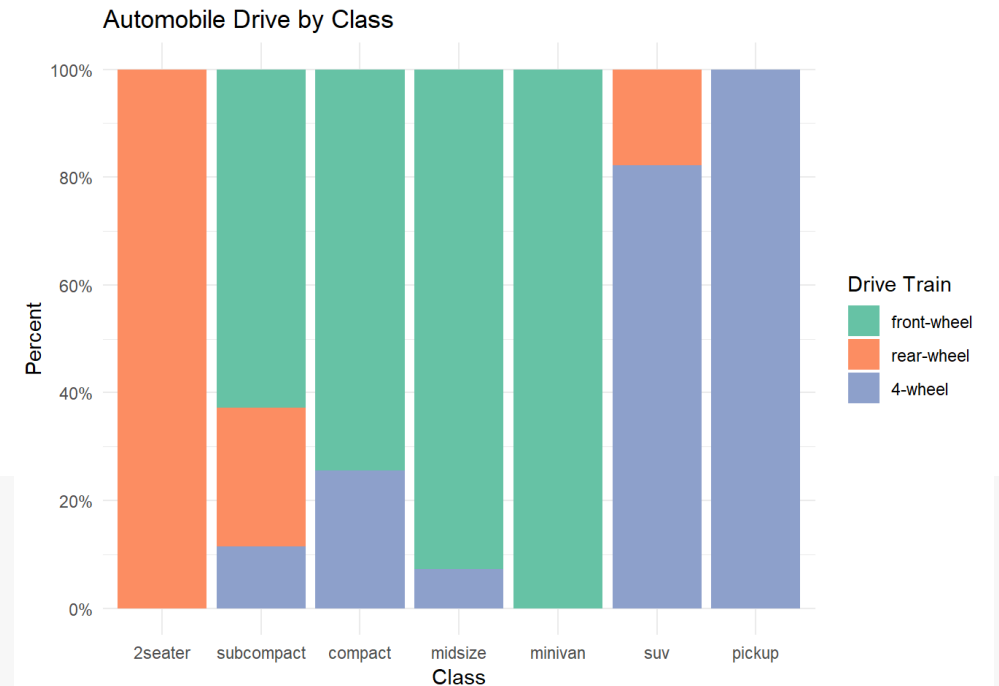
**Improving the color and labeling**

```
# bar plot, with each bar representing 100%,  
# reordered bars, and better labels and colors
```

```
library(scales)
```

```
ggplot(mpg,
```

```
aes(x = factor(class, levels = c("2seater", "subcompact", "compact", "midsize", "minivan", "suv", "pickup")),  
fill = factor(drv, levels = c("f", "r", "4"), labels = c("front-wheel", "rear-wheel", "4-wheel")))) +  
geom_bar(position = "fill") +  
scale_y_continuous(breaks = seq(0, 1, .2), label = percent) +  
scale_fill_brewer(palette = "Set2") +  
labs(y = "Percent", fill = "Drive Train", x = "Class", title = "Automobile Drive by Class") +  
theme_minimal()
```



# Bivariate Graphs

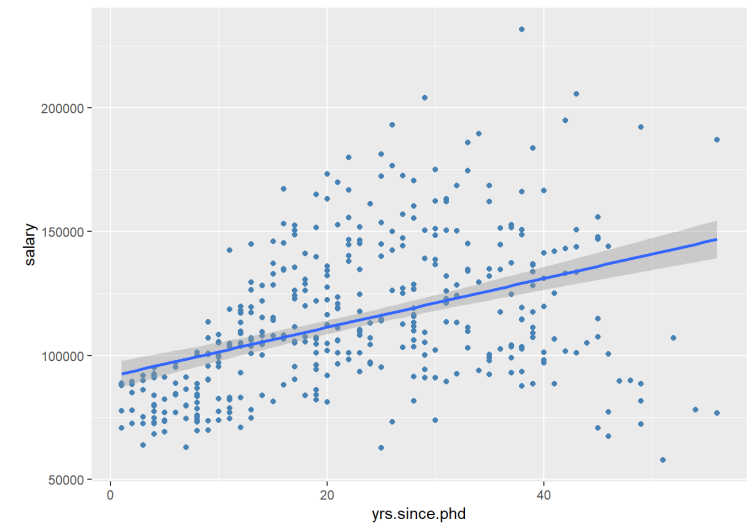
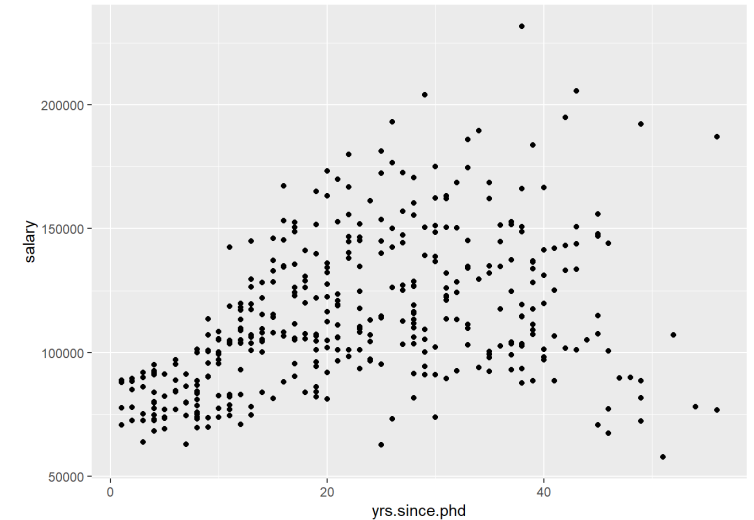
## Quantitative vs. Quantitative

### Scatterplot

```
data(Salaries, package="carData")  
  
# simple scatterplot  
ggplot(Salaries, aes(x = yrs.since.phd, y = salary)) +  
geom_point()
```

```
# scatterplot with linear fit line
```

```
ggplot(Salaries, aes(x = yrs.since.phd, y = salary)) +  
geom_point(color= "steelblue") +  
geom_smooth(method = "lm")
```



# Bivariate Graphs

Quantitative vs. Quantitative

## Line plot

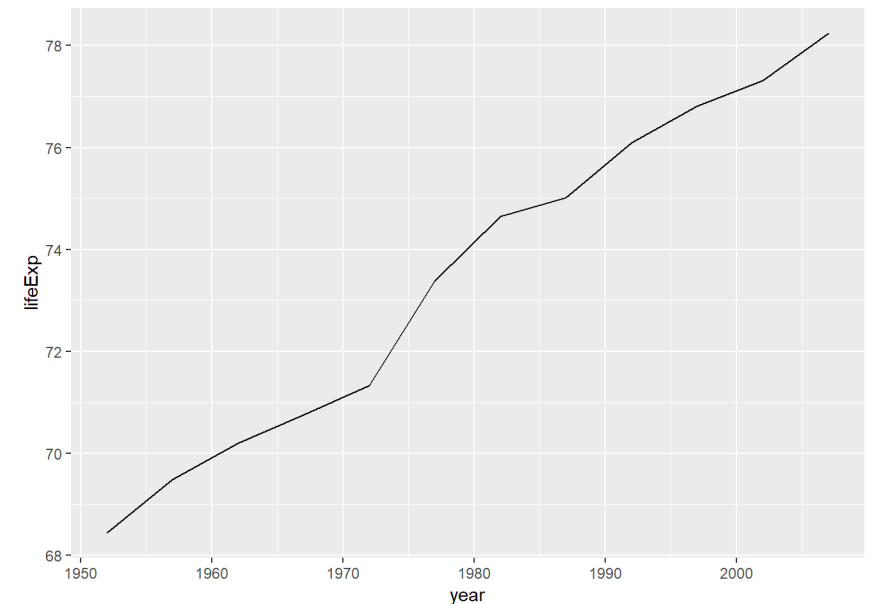
```
data(gapminder, package="gapminder") # Select US cases

library(dplyr)

plotdata <- filter(gapminder, country == "United States")

# simple line plot

ggplot(plotdata, aes(x = year, y = lifeExp)) +
  geom_line()
```



# Bivariate Graphs

## Categorical vs. Quantitative

Bar chart (on summary statistics)

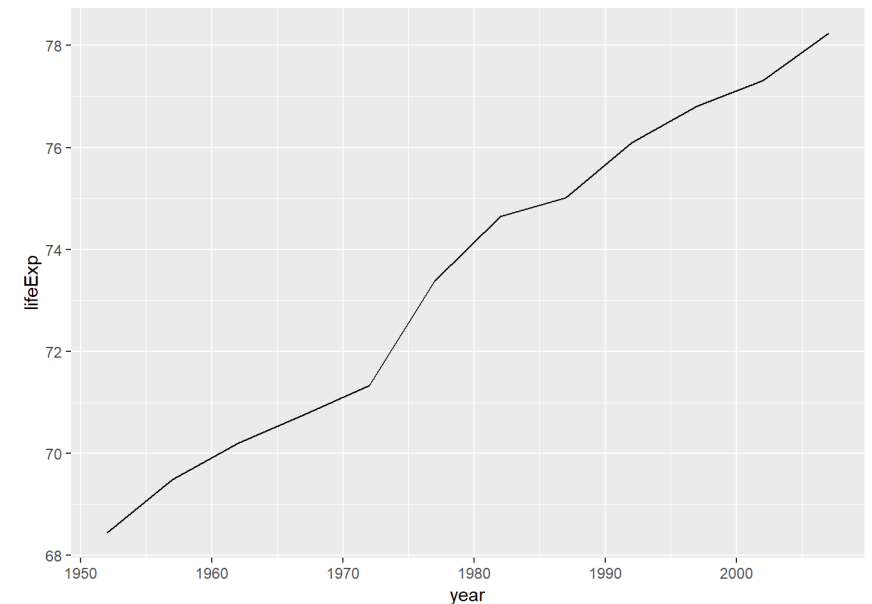
```
data(gapminder, package="gapminder") # Select US cases

library(dplyr)

plotdata <- filter(gapminder, country == "United States")

# simple line plot

ggplot(plotdata, aes(x = year, y = lifeExp)) +
  geom_line()
```





# Bivariate Graphs

## Categorical vs. Quantitative

Bar chart (on summary statistics)

```
# plot mean salaries
library(scales)

ggplot(plotdata, aes(x = factor(rank, labels =
c("Assistant\nProfessor", "Associate\nProfessor",
"Full\nProfessor")), y = mean_salary)) +
geom_bar(stat = "identity", fill = "cornflowerblue") +
geom_text(aes(label = dollar(mean_salary)), vjust = -0.25) +
scale_y_continuous(breaks = seq(0, 130000, 20000), label =
dollar) +
labs(title = "Mean Salary by Rank", subtitle = "9-month
academic salary for 2008-2009", x = "", y = "")
```



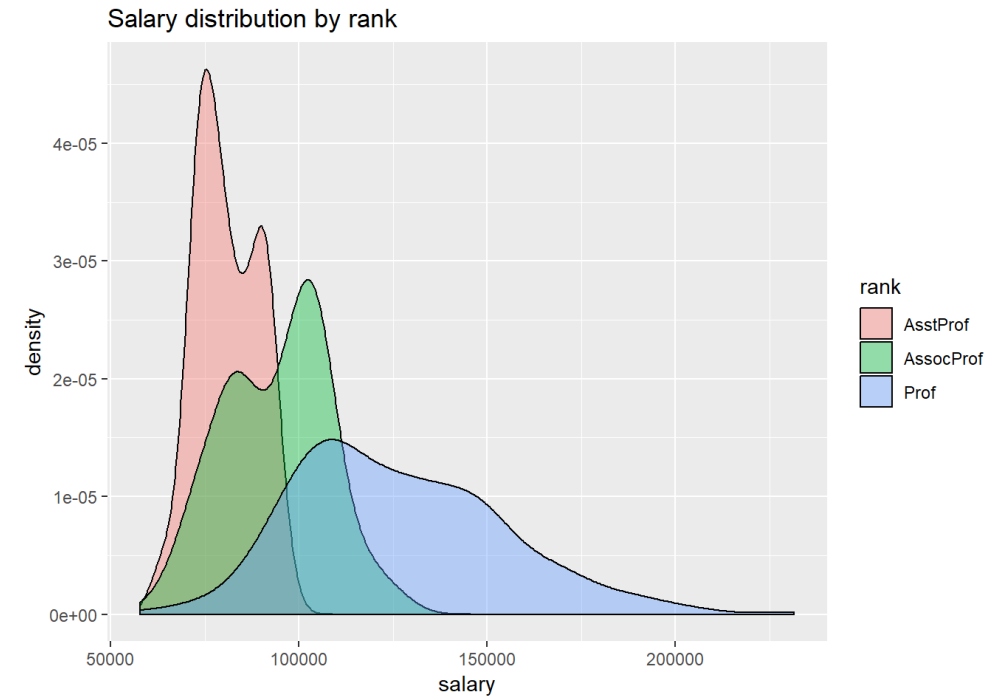
# Bivariate Graphs

## Categorical vs. Quantitative

Grouped kernel density plots

```
# plot the distribution of salaries # by rank using kernel density plots
```

```
ggplot(Salaries, aes(x = salary, fill = rank)) +  
geom_density(alpha = 0.4) +  
labs(title = "Salary distribution by rank")
```





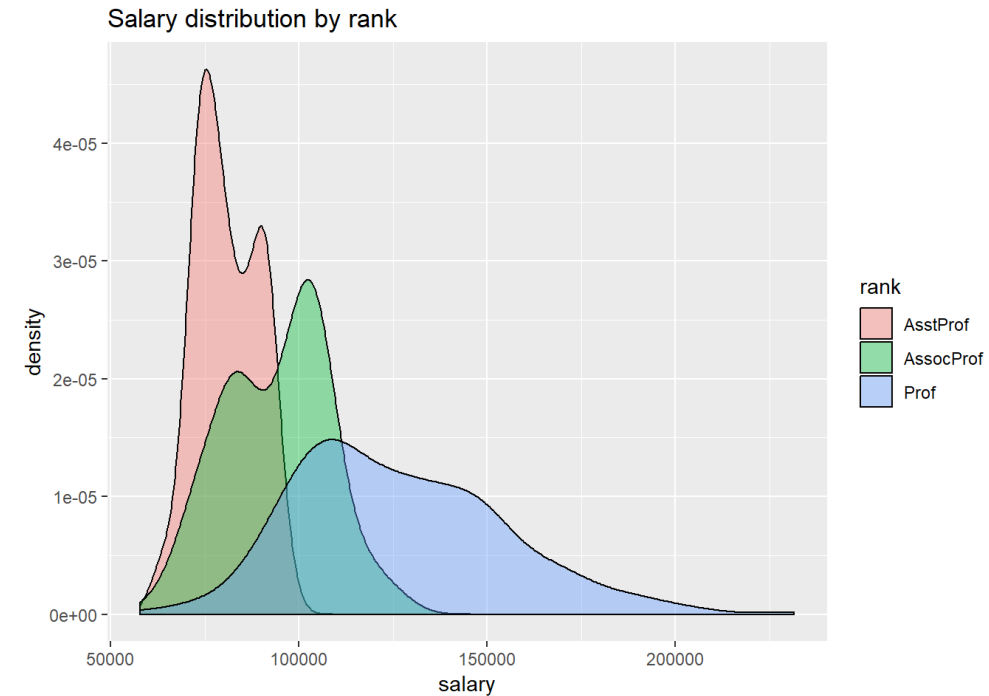
# Bivariate Graphs

## Categorical vs. Quantitative

Grouped kernel density plots

```
# plot the distribution of salaries # by rank using kernel density plots
```

```
ggplot(Salaries, aes(x = salary, fill = rank)) +  
geom_density(alpha = 0.4) +  
labs(title = "Salary distribution by rank")
```

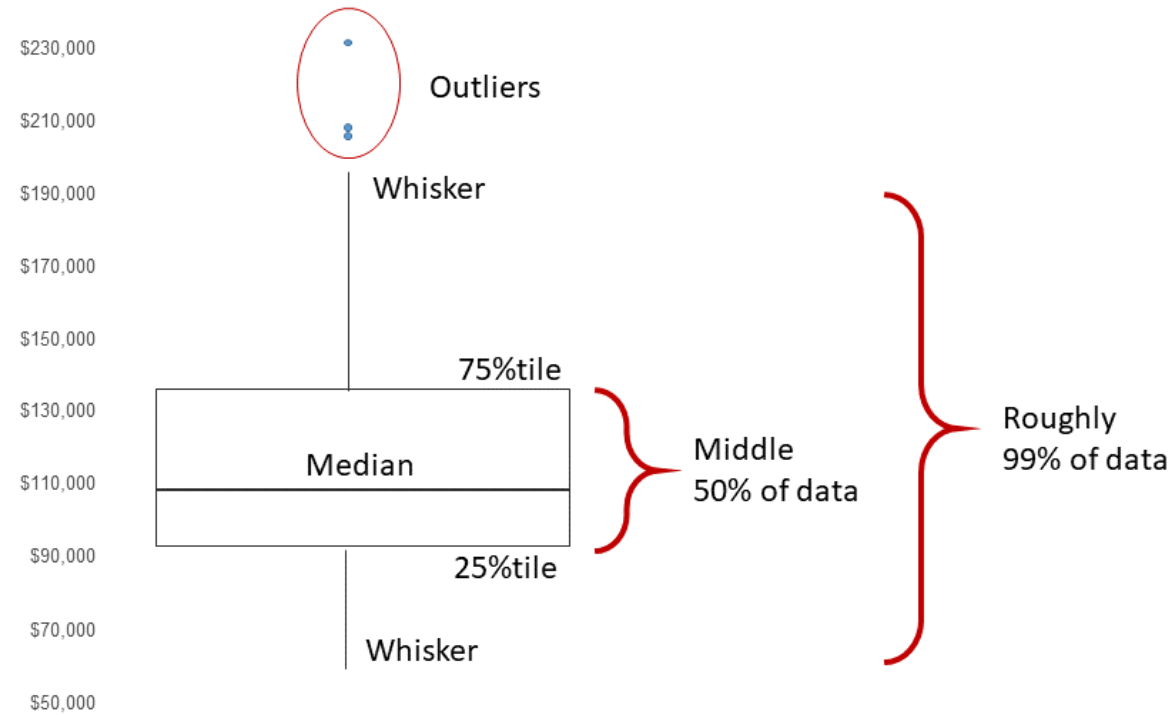




# Bivariate Graphs

## Categorical vs. Quantitative

### Boxplots





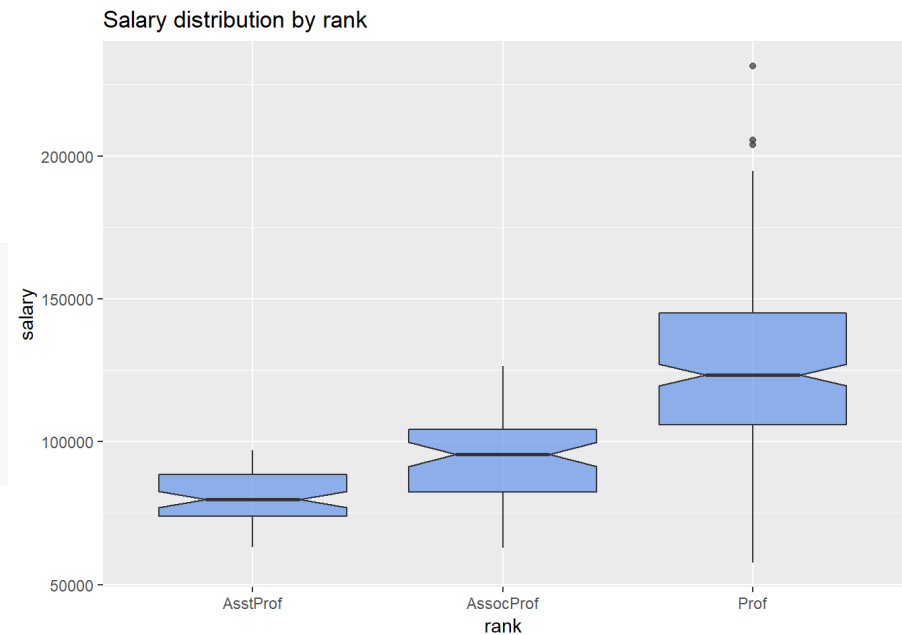
# Bivariate Graphs

## Categorical vs. Quantitative

### Boxplots

*# plot the distribution of salaries by rank using boxplots*

```
ggplot(Salaries, aes(x = rank, y = salary)) +  
geom_boxplot(notch = TRUE, fill = "cornflowerblue", alpha = .7) +  
labs(title = "Salary distribution by rank")
```





# Bivariate Graphs

## Categorical vs. Quantitative

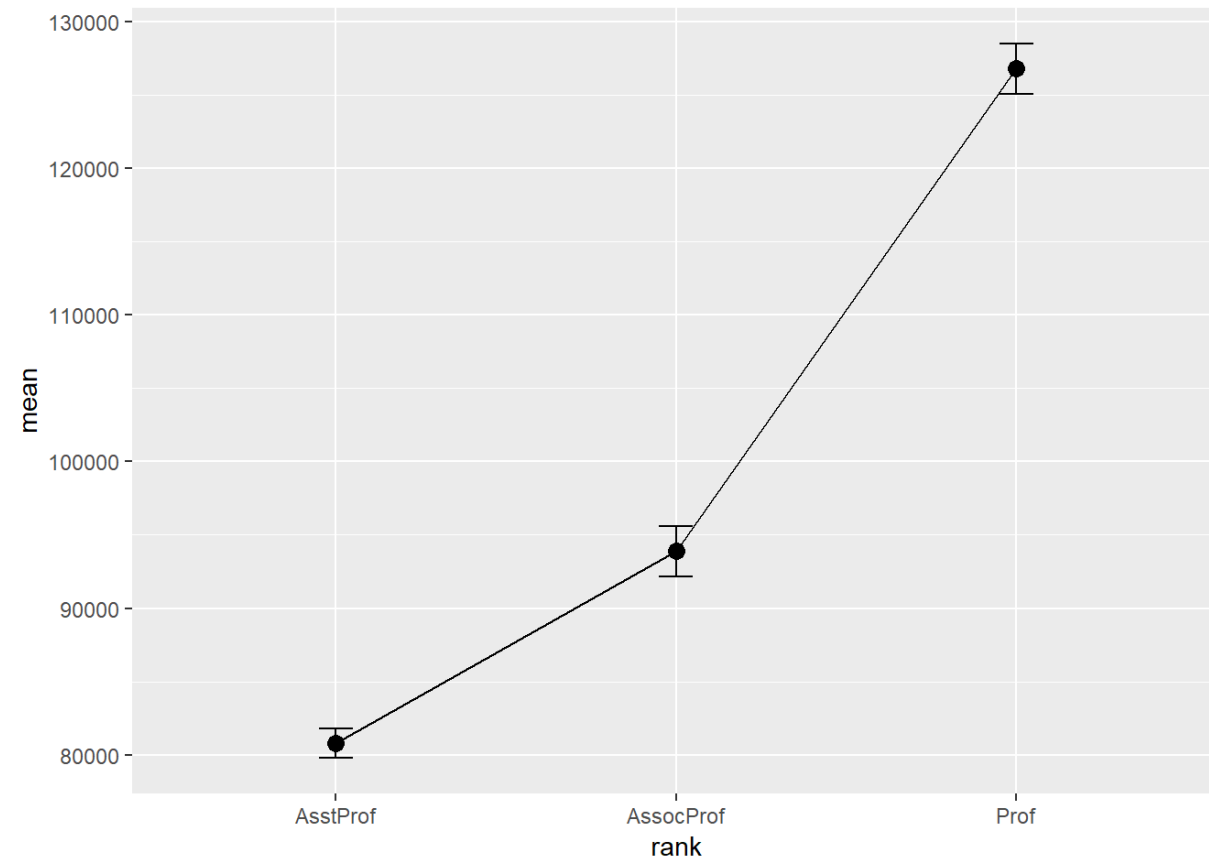
Mean/SEM plots

```
# calculate means, standard deviations, # standard errors, and 95% confidence # intervals by rank
```

```
library(dplyr)
plotdata <- Salaries %>% group_by(rank) %>%
  summarize(n = n(), mean = mean(salary), sd =
  sd(salary), se = sd / sqrt(n), ci = qt(0.975, df =
  n - 1) * sd / sqrt(n))
```

```
# plot the means and standard errors
```

```
ggplot(plotdata, aes(x = rank, y = mean, group =
1)) + geom_point(size = 3) + geom_line() +
geom_errorbar(aes(ymin = mean - se, ymax = mean +
se), width = .1)
```





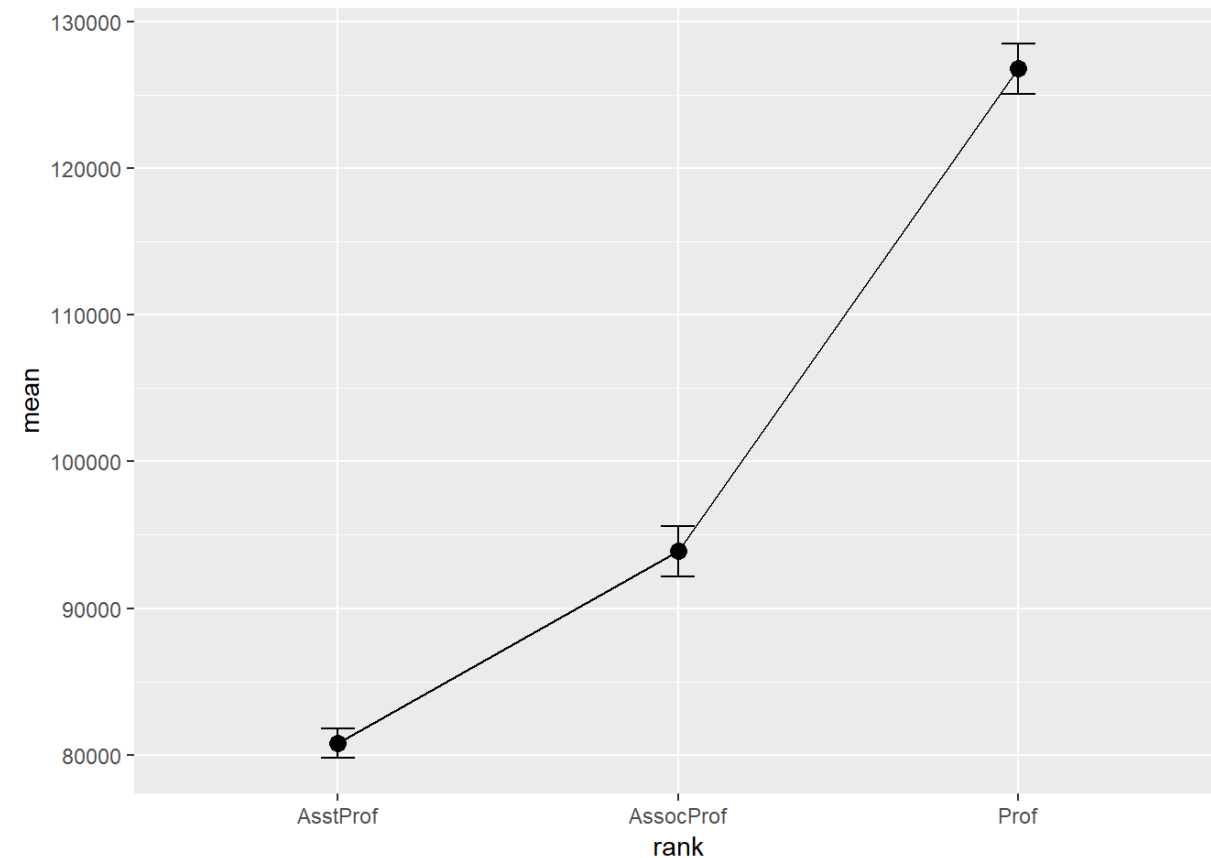
# Bivariate Graphs

## Categorical vs. Quantitative

Mean/SEM plots

```
# calculate means and standard errors by  
rank and sex
```

```
plotdata <- Salaries %>% group_by(rank, sex)  
%>% summarize(n = n(), mean = mean(salary),  
sd = sd(salary), se = sd/sqrt(n))
```





# Bivariate Graphs

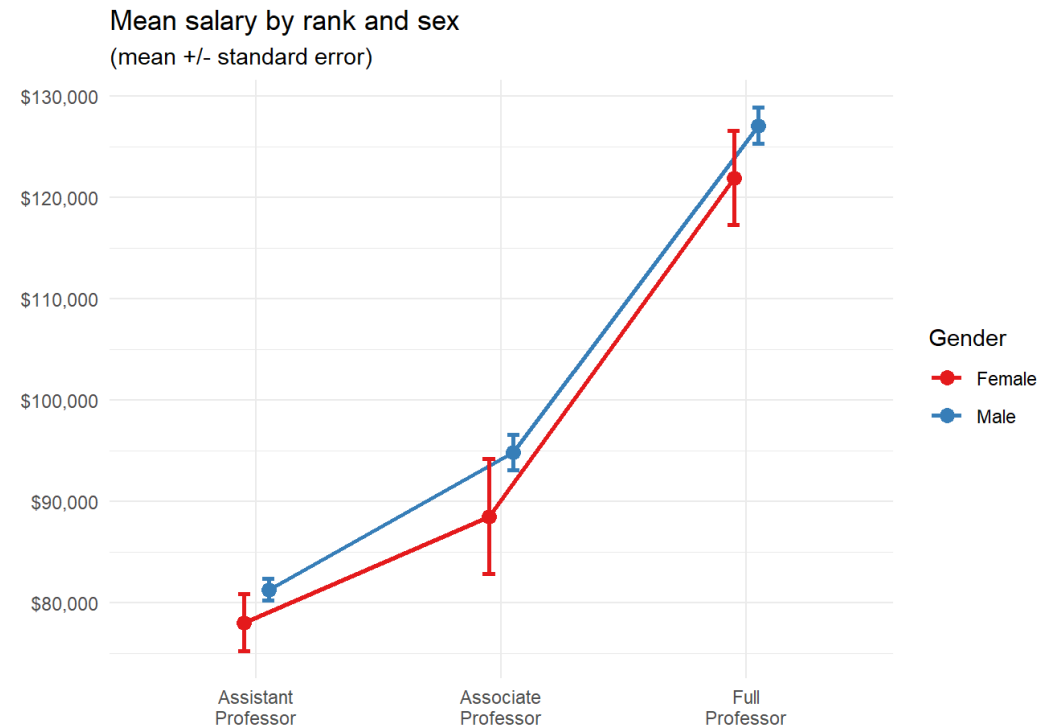
## Categorical vs. Quantitative

Mean/SEM plots

```
# improved means/standard error plot

pd <- position_dodge(0.2)

ggplot(plotdata,
  aes(x = factor(rank, labels = c("Assistant\nProfessor",
  "Associate\nProfessor", "Full\nProfessor")), y = mean, group=sex,
  color=sex)) +
  geom_point(position=pd, size = 3) + geom_line(position = pd, size = 1) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se), width = .1,
  position = pd, size = 1) + scale_y_continuous(label = scales::dollar) +
  scale_color_brewer(palette="Set1") + theme_minimal() +
  labs(title = "Mean salary by rank and sex", subtitle = "(mean +/-
  standard error)", x = "", y = "", color = "Gender")
```



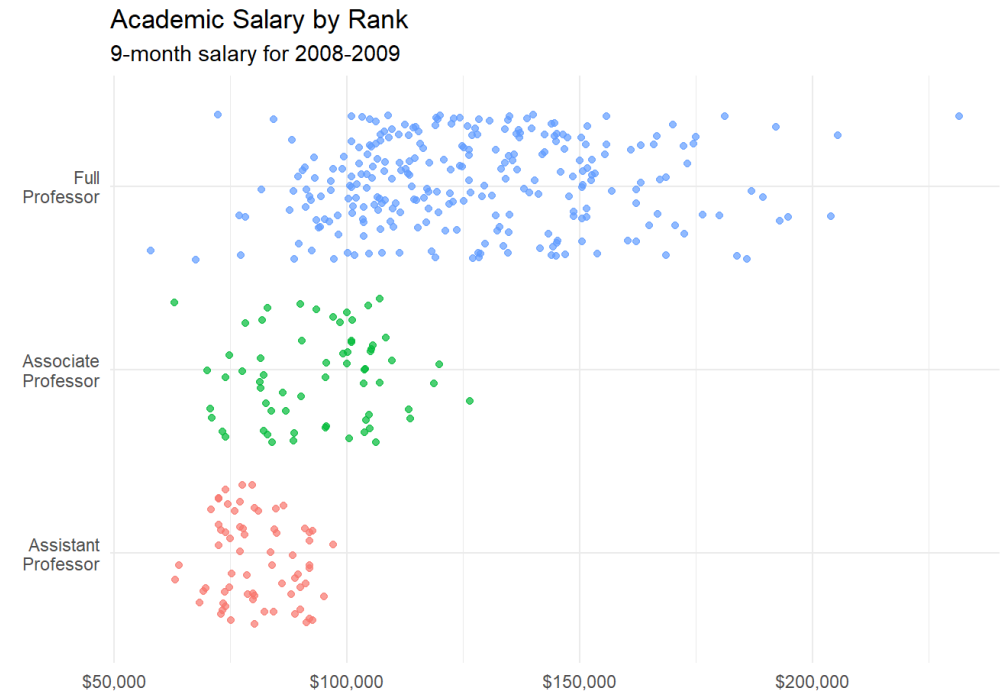
# Bivariate Graphs



```
# plot the distribution of salaries # by rank using jittering
```

```
library(scales)
```

```
ggplot(Salaries, aes(y = factor(rank, labels = c("Assistant\nProfessor", "Associate\nProfessor",  
"Full\nProfessor")), x = salary, color = rank)) +  
geom_jitter(alpha = 0.7, size = 1.5) + scale_x_continuous(label = dollar) +  
labs(title = "Academic Salary by Rank", subtitle = "9-month salary for 2008-2009", x = "", y = "") +  
theme_minimal() +  
theme(legend.position = "none")
```



$$\mathbf{x} = \mathbf{x}_f - \mathbf{x}_i \quad \Delta \mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$$

$$= \frac{\Delta \vec{r}}{\Delta t} \quad \vec{a} = \frac{\Delta \vec{v}}{\Delta t}$$

$$v = |\mathbf{v}| = \sqrt{v_x^2 + v_y^2}$$

$$\theta = \tan^{-1}\left(\frac{v_y}{v_x}\right)$$



$$\omega = \frac{\Delta \theta}{\Delta t} \quad \alpha = \frac{\Delta \omega}{\Delta t}$$

$$= \mathbf{v}_0 + \mathbf{a}t$$

$$= \mathbf{x}_0 + \mathbf{v}_0 t + \frac{1}{2} \mathbf{a} t^2$$

$$-v_0^2 = 2a(x - x_0)$$

$$= \frac{v_f^2 - v_i^2}{2a}$$

# Statistical Models

$$\omega = \omega_0 + \alpha t$$

$$\theta = \theta_0 + \omega_0 t + \frac{1}{2} \alpha t^2$$

$$\omega_0^2 = 2\alpha(\theta - \theta_0)$$

Curvfit Rankings for 0.5% SEV CO<sub>2</sub>

Rank	F-statistic
1	7.962883557
2	7.8601110639
3	7.5283512645
4	7.3357010958
5	6.3801158367
6	3.8079206858
7	3.742891358
8	3.5727028219
9	3.5546937052
10	3.0133372321
11	2.7408796673
12	2.6986270263
13	2.5801276758
14	2.5622800357
15	2.1798855221

**Prof. Dr. Javier Valdes**  
[Javier.valdes@th-deg.de](mailto:Javier.valdes@th-deg.de)

IAI - Institut für Angewandte Informatik  
 Institute for Applied Informatics

Technische Hochschule Deggendorf  
 Technologie Campus Freyung  
 Grafenauer Str. 22, D- 94078 Freyung,  
 Germany

Tel.: +49 8551 91764 40  
 Fax: +49 8551 91764 69

$$x = A \cos(\omega t + \phi) \quad v = -A \omega \sin(\omega t + \phi)$$





# Linear Regression



Linear regression allows us to explore the relationship between a quantitative response variable and an explanatory variable while other variables are held constant.

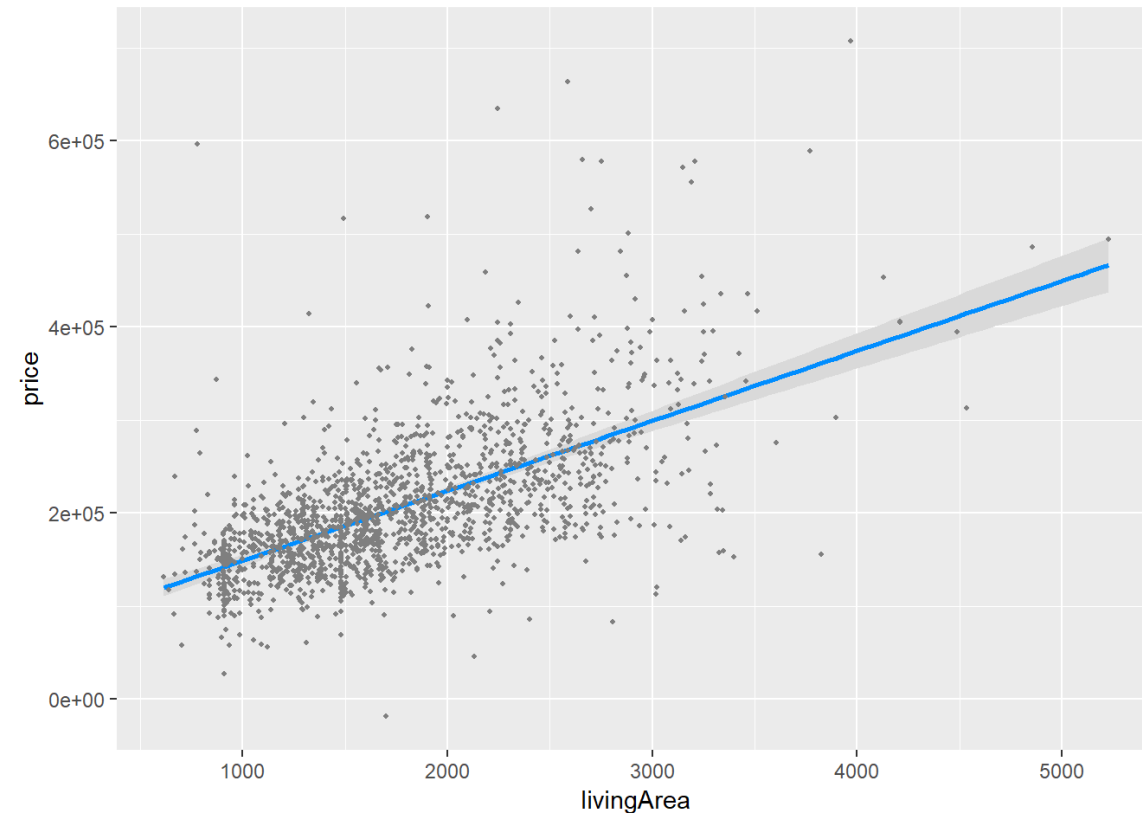
Consider the prediction of home prices in the [Saratoga](#) dataset from lot size (square feet), age (years), land value (1000s dollars), living area (square feet), number of bedrooms and bathrooms and whether the home is on the waterfront or not.

```
data(SaratogaHouses, package="mosaicData")

houses_lm <- lm(price ~ lotSize + age + landValue +
  livingArea + bedrooms + bathrooms + waterfront, data =
  SaratogaHouses)
```

```
# conditional plot of price vs. living area
```

```
library(ggplot2)
library(visreg)
visreg(houses_lm, "livingArea", gg = TRUE)
```



# Survival plots



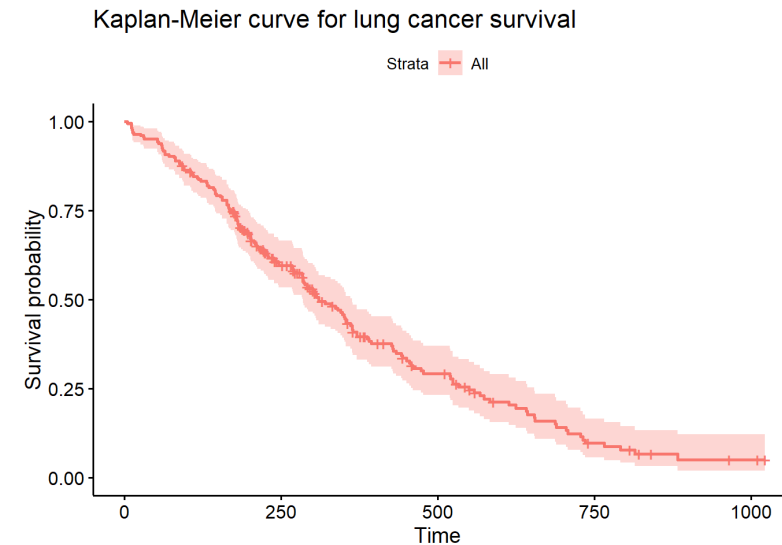
In many research settings, the response variable is the time to an event. This is frequently true in healthcare research, where we are interested in time to recovery, time to death, or time to relapse.

If the event has not occurred for an observation (either because the study ended or the patient dropped out) the observation is said to be *censored*.

```
# plot survival curve
```

```
library(survival)  
library(survminer)  
data(lung)
```

```
sfit <- survfit(Surv(time, status) ~ 1, data=lung)  
ggsurvplot(sfit, title="Kaplan-Meier curve for lung  
cancer survival")
```



The outcome for each patient is measured by two variables

- *time* - survival time in days
- *status* - 1=censored, 2=dead