

**NC STATE UNIVERSITY**

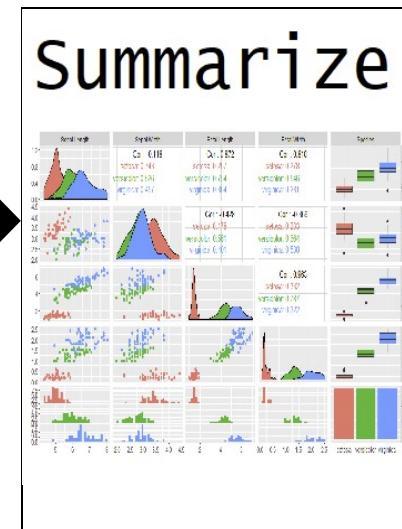
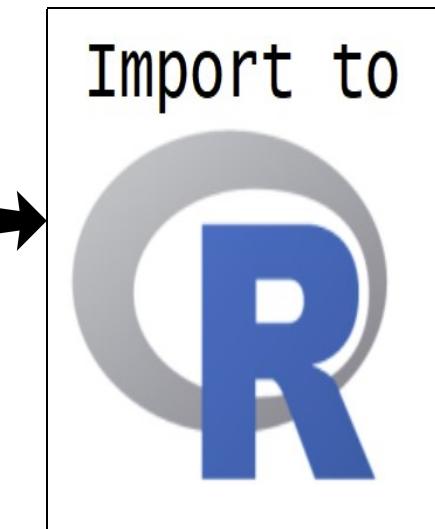
# Summarizing Data

Justin Post

# What is this course about?

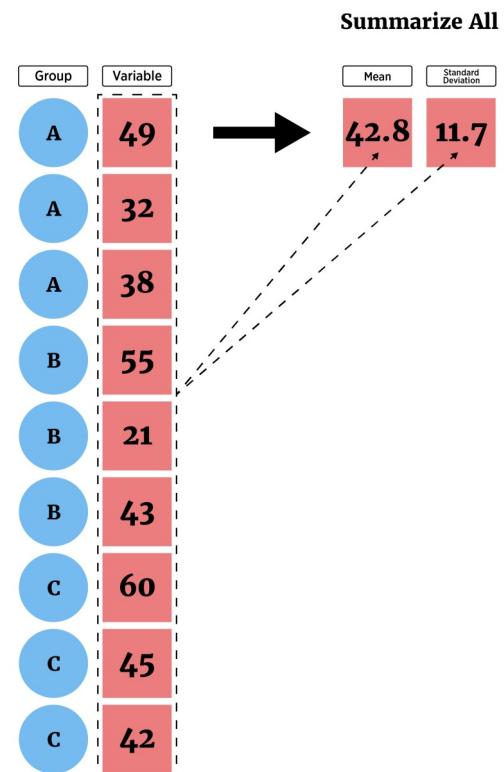
Basic use of R for reading, manipulating, and plotting data!

temp	conc	time	percent
-1	-1	-1	45.9
1	-1	-1	60.6
-1	1	-1	57.5
1	1	1	58
-1	1	1	58.8
1	1	1	52.4



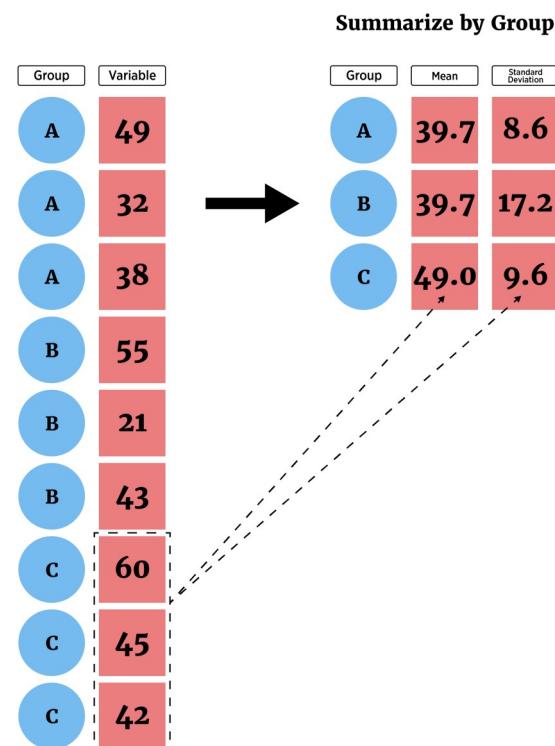
# Where do we start?

- Understand types of data and their distributions
- Numerical summaries



# Where do we start?

- Understand types of data and their distributions
- Numerical summaries (across subgroups)



# Where do we start?

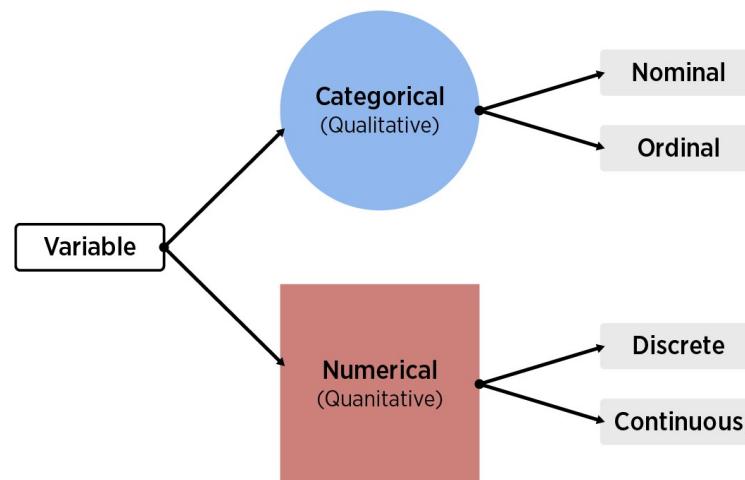
- Understand types of data and their distributions
- Numerical summaries (across subgroups)
  - Contingency Tables
  - Mean/Median
  - Standard Deviation/Variance/IQR
  - Quantiles/Percentiles

# Where do we start?

- Understand types of data and their distributions
- Numerical summaries (across subgroups)
  - Contingency Tables
  - Mean/Median
  - Standard Deviation/Variance/IQR
  - Quantiles/Percentiles
- Graphical summaries (across subgroups)
  - Bar plots
  - Histograms
  - Box plots
  - Scatter plots

# Understanding Data

- How to summarize data?
- Depends on data type:
  - Categorical (Qualitative) variable - entries are a label or attribute
  - Numeric (Quantitative) variable - entries are a numerical value where math can be performed



# Categorical Data

Common goal: Describe the **distribution** of the variable

- Distribution = pattern and frequency with which you observe a variable
- Categorical variable - entries are a label or attribute
- Try to describe relative frequency (or count) in each category
- Create contingency tables via `table`

# Contingency tables

- Consider data on titanic passengers in `titanic.csv`

```
titanicData <- read_csv("../datasets/titanic.csv")  
titanicData  
  
## # A tibble: 1,310 × 14  
##   pclass survived name       sex     age sibsp parch ticket    fare cabin embarked  
##   <dbl>     <dbl> <chr>     <chr>   <dbl> <dbl> <dbl> <chr>    <dbl> <chr> <chr>  
## 1     1       1 Allen, M~ fema~ 29        0      0 24160  211. B5     S  
## 2     1       1 Allison,~ male  0.917     1      2 113781  152. C22 ~ S  
## 3     1       0 Allison,~ fema~ 2          1      2 113781  152. C22 ~ S  
## 4     1       0 Allison,~ male  30         1      2 113781  152. C22 ~ S  
## 5     1       0 Allison,~ fema~ 25         1      2 113781  152. C22 ~ S  
## # ... with 1,305 more rows, and 3 more variables: boat <chr>, body <dbl>,  
## #   home.dest <chr>
```

# Contingency tables

- Create **one-way contingency tables** for each of three categorical variables:
  - embarked (where journey started)
  - survived (survive or not)
  - sex (Male or Female)

```
table(titanicData$embarked)
```

```
##  
##   C     Q     S  
## 270  123  914
```

```
table(titanicData$survived)
```

```
##  
##   0     1  
## 809  500
```

```
table(titanicData$sex)
```

```
##  
##   female    male  
##      466    843
```

# Two-way contingency tables

- Create **two-way contingency tables** for pairs of categorical variables

```
table(titanicData$survived,  
      titanicData$sex)
```

```
##  
##      female male  
## 0    127   682  
## 1    339   161
```

```
table(titanicData$survived,  
      titanicData$embarked)
```

```
##  
##          C     Q     S  
## 0    120   79  610  
## 1    150   44 304
```

```
table(titanicData$sex,  
      titanicData$embarked)
```

```
##  
##          C     Q     S  
## female 113   60 291  
## male   157   63 623
```

# Three-way contingency tables

- Create a **three-way contingency table** for three categorical variables

```
table(titanicData$sex, titanicData$embarked, titanicData$survived)
```

```
## , , = 0
##
##
##          C   Q   S
## female   11  23  93
## male    109  56 517
##
## , , = 1
##
##
##          C   Q   S
## female 102  37 198
## male    48   7 106
```

# Three-way contingency tables

- Create a **three-way contingency table** for three categorical variables (order matters for output!)
- Example of an array! 3 dimensions [ , , ]

```
tab <- table(titanicData$sex, titanicData$embarked, titanicData$survived)
```

```
str(tab)
```

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
##  - attr(*, "dimnames")=List of 3
##    ..$ : chr [1:2] "female" "male"
##    ..$ : chr [1:3] "C" "Q" "S"
##    ..$ : chr [1:2] "0" "1"
```

# Conditional contingency tables

- Can obtain **conditional** bivariate info!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
## - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

```
#returns embarked vs survived table for females
tab[1, , ]
```

```
##
##      0   1
## C 11 102
## Q 23 37
## S 93 198
```

# Conditional contingency tables

- Can obtain **conditional** bivariate info!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
## - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

*#returns embarked vs survived table for males*  
tab[2, , ]

```
##
##      0    1
## C 109  48
## Q  56   7
## S 517 106
```

# Conditional contingency tables

- Can obtain **conditional** bivariate info!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
## - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

```
#returns survived vs sex table for embarked "C"
tab[, 1, ]
```

```
##
##          0    1
## female  11 102
## male   109  48
```

# Conditional contingency tables

- Can obtain **conditional univariate** info too!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
## - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

*#Survived status for males that embarked at "Q"*  
tab[2, 2, ]

```
## 0 1
## 56 7
```

# Numerical summaries: Numeric variables

Numeric variable - entries are a numerical value where math can be performed

**Single variable:** describe the distribution via

- Shape: Histogram, Density plot, ...
- Measures of center: Mean, Median, ...
- Measures of spread: Variance, Standard Deviation, Quartiles, IQR, ...

# Numerical summaries: Numeric variables

Numeric variable - entries are a numerical value where math can be performed

**Single variable:** describe the distribution via

- Shape: Histogram, Density plot, ...
- Measures of center: Mean, Median, ...
- Measures of spread: Variance, Standard Deviation, Quartiles, IQR, ...

**Two Variables:**

- Shape: Scatter plot, ...
- Measures of linear relationship: Covariance, Correlation, ...

# Numerical summaries: Numeric variables

- Look at carbon dioxide (CO<sub>2</sub>) uptake data set
  - Response recorded: `uptake` CO<sub>2</sub> uptake rates in grass plants
  - Environment manipulated: Treatment - chilled/nonchilled
  - Ambient CO<sub>2</sub> specified and measured: `conc`

```
CO2 <- as_tibble(CO2)
CO2

## # A tibble: 84 x 5
##   Plant Type  Treatment conc  uptake
##   <ord> <fct>    <fct>   <dbl>  <dbl>
## 1 Qn1   Quebec nonchilled    95    16
## 2 Qn1   Quebec nonchilled   175   30.4
## 3 Qn1   Quebec nonchilled   250   34.8
## 4 Qn1   Quebec nonchilled   350   37.2
## 5 Qn1   Quebec nonchilled   500   35.3
## # ... with 79 more rows
```

# Measures of center

## Mean & Median

```
mean(CO2$uptake)
```

```
## [1] 27.2131
```

*#note you can easily get a trimmed mean*

```
mean(CO2$uptake, trim = 0.05) #5% trimmed mean
```

```
## [1] 27.25263
```

```
median(CO2$uptake)
```

```
## [1] 28.3
```

# Measures of spread

## Variance, Standard Deviation, Quartiles, & IQR

#quartiles and mean

```
summary(CO2$uptake)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.  
##    7.70    17.90   28.30    27.21   37.12   45.50
```

```
var(CO2$uptake)
```

```
IQR(CO2$uptake)
```

```
## [1] 116.9515
```

```
## [1] 19.225
```

```
sd(CO2$uptake)
```

```
quantile(CO2$uptake, probs = c(0.1, 0.2))
```

```
## [1] 10.81441
```

```
##    10%    20%
```

```
## 12.36 15.64
```

# Measures of linear relationship

## Covariance & Correlation

```
cov(CO2$conc, CO2$uptake)
```

```
## [1] 1552.687
```

```
cor(CO2$conc, CO2$uptake)
```

```
## [1] 0.4851774
```

# Numerical summaries: Numeric variables

Usually want summaries for different subgroups of data

- Ex: Get similar uptake summaries for each **Treatment**

# Numerical summaries: Numeric variables

Usually want summaries for different subgroups of data

- Ex: Get similar uptake summaries for each **Treatment**
- `dplyr` easy to use but can only return one value

# Numerical summaries: Numeric variables

Usually want summaries for different subgroups of data

- Ex: Get similar uptake summaries for each **Treatment**
- `dplyr` easy to use (although it can only return one value)

Idea:

- Use `group_by` to create subgroups associated with the data frame
- Use `summarize` to create basic summaries for each subgroup

# Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**

```
CO2 %>%  
  group_by(Treatment) %>%  
  summarise(avg = mean(uptake), med = median(uptake), var = var(uptake))  
  
## # A tibble: 2 x 4  
##   Treatment     avg     med     var  
##   <fct>       <dbl>   <dbl>   <dbl>  
## 1 nonchilled  30.6    31.3   94.2  
## 2 chilled     23.8    19.7  118.
```

# Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment** and **Concentration**

```
CO2 %>%  
  group_by(Treatment, conc) %>%  
    summarise(avg = mean(uptake), med = median(uptake), var = var(uptake))
```

```
## # A tibble: 14 x 5  
## # Groups: Treatment [2]  
##   Treatment   conc     avg     med     var  
##   <fct>     <dbl> <dbl> <dbl> <dbl>  
## 1 nonchilled     95  13.3  12.8  5.75  
## 2 nonchilled    175  25.1  24.6 32.6  
## 3 nonchilled    250  32.5  32.7 35.1  
## 4 nonchilled    350  35.1  34.5 37.4  
## 5 nonchilled    500  35.1  33.8 31.9  
## 6 nonchilled    675  36.0  35.8 40.2  
## 7 nonchilled   1000  37.4  37.6 49.8  
## 8 chilled        95  11.2  10.6  8.18  
## 9 chilled       175  19.4  19.5 34.7  
## 10 chilled      250  25.3  24.2 112.  
## 11 chilled      350  26.2  26.4 117.  
## 12 chilled      500  26.6  26   131.  
## 13 chilled      675  27.9  28.8 120.  
## 14 chilled     1000  29.8  30.3 154.
```

# Summarizing across groups

dplyr has variations on summarise that can be used:

- summarise\_all() - Apply functions to every column
- summarise\_at() - Apply functions to specific columns
- summarise\_if() - Apply functions to all columns of one type

# Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**
- Built-in `aggregate()` function more general

# Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**
- Built-in `aggregate()` function more general
- Basic use gives response (`x`) and a list of variables to group by

```
aggregate(x = CO2$uptake, by = list(CO2$Treatment), FUN = summary)
```

```
##      Group.1    x.Min.   x.1st Qu.   x.Median   x.Mean   x.3rd Qu.   x.Max.
## 1 nonchilled 10.60000 26.47500 31.30000 30.64286 38.70000 45.50000
## 2 chilled    7.70000 14.52500 19.70000 23.78333 34.90000 42.40000
```

# Summarizing across groups

- `aggregate()` is commonly used with `formula` notation!

`uptake ~ Treatment` - is an example of formula notation

- Idea: `uptake` (LHS) modeled by `Treatment levels` (RHS)

# Summarizing across groups

- `aggregate()` is commonly used with formula notation!

`uptake ~ Treatment` - is an example of formula notation

- Idea: `uptake` (LHS) modeled by `Treatment levels` (RHS)

```
aggregate(uptake ~ Treatment, data = CO2, FUN = summary)
```

# Summarizing across groups

- `aggregate()` is commonly used with formula notation!

`uptake ~ Treatment + conc` model uptake by levels of Treatment and conc

```
aggregate(uptake ~ Treatment + conc, data = CO2, FUN = summary)
```

```
##      Treatment conc uptake.Min. uptake.1st Qu. uptake.Median uptake.Mean
## 1    nonchilled   95    10.60000    11.47500    12.80000    13.28333
## 2      chilled   95     7.70000     9.60000    10.55000    11.23333
## 3    nonchilled  175    19.20000    20.05000    24.65000    25.11667
## 4      chilled  175    11.40000    15.67500    19.50000    19.45000
## 5    nonchilled  250    25.80000    27.30000    32.70000    32.46667
## 6      chilled  250    12.30000    17.95000    24.20000    25.28333
## 7    nonchilled  350    27.90000    30.45000    34.50000    35.13333
## 8      chilled  350    13.00000    18.15000    26.45000    26.20000
## 9    nonchilled  500    28.50000    31.27500    33.85000    35.10000
## 10     chilled  500    12.50000    18.30000    26.00000    26.65000
## 11    nonchilled  675    28.10000    31.42500    35.80000    36.01667
## 12     chilled  675    13.70000    19.72500    28.80000    27.88333
## 13    nonchilled 1000    27.80000    32.50000    37.60000    37.38333
## 14     chilled 1000    14.40000    20.40000    30.30000    29.78333
##      uptake.3rd Qu. uptake.Max.
## 1          15.40000    16.20000
## 2          13.30000    15.10000
```

# Recap

- Understand types of data and their distributions
- Numerical summaries
  - Contingency Tables: `table`
  - Mean/Median: `mean`, `median`
  - Standard Deviation/Variance/IQR: `sd`, `var`, `IQR`
  - Quantiles/Percentiles: `quantile` for more general quantiles
- Across subgroups with `dplyr::group_by` and `dplyr::summarize` or `aggregate`

# Quick Examples

# Where are we at?

- Understand types of data and their distributions
- Numerical summaries (across subgroups)
  - Contingency Tables
  - Mean/Median
  - Standard Deviation/Variance/IQR
  - Quantiles/Percentiles
- Graphical summaries (across subgroups)
  - Bar plots (categorical data)
  - Histograms
  - Box plots
  - Scatter plots

# Graphical Summaries

Three major systems for plotting:

- Base R (built-in functions)
- Lattice
- ggplot2 (sort of part of the tidyverse - [Cheat Sheet](#))
  - `ggplot(data = data_frame)` creates a plot instance
  - Add “layers” to the system (geoms or stats)

Great [reference book here!](#)

# ggplot2 Plotting

## ggplot2 basics ([Cheat Sheet](#))

- `ggplot(data = data_frame)` creates a plot instance
- Add “layers” to the system (geoms or stats)
  - Creates a visualization of the data

# ggplot2 Plotting

## ggplot2 basics ([Cheat Sheet](#))

- `ggplot(data = data_frame)` creates a plot instance
- Add “layers” to the system (geoms or stats)
  - Creates a visualization of the data
- Modify layer “mapping” args (aes)
  - Ex: size, color, and x, y location(s)
- Coordinate system (mostly use Cartesian plane)
- Optional: Titles, etc.

# factors

- factor - special class of vector with a `levels` attribute
- Levels define all possible values for that variable
  - Great for variable like `Day` (Monday, Tuesday, ...)
  - Not great for variable like `Name` where new values may come up
- Quite useful with plotting
  - Allows for easy labeling of subgroups

# factors

- Consider data on titanic passengers in `titanic.csv`

```
#convert survival status to a factor
titanicData$survived <- as.factor(titanicData$survived)
levels(titanicData$survived) #R knows it isn't numeric now

## [1] "0" "1"
```

- Can't add value unless it is a level

```
titanicData$survived[1] <- "5"

## Warning in `[<-.factor`(`*tmp*`, 1, value = structure(c(NA, 2L, 1L, 1L, :
## invalid factor level, NA generated
```

# factor levels

- Useful if you want to create better labels (or change the ordering)

```
levels(titanicData$survived) <- c("Died", "Survived")
```

```
levels(titanicData$survived)
```

```
## [1] "Died"      "Survived"
```

# ggplot2 Plotting: Categorical variables

Categorical variable - entries are a label or attribute

Generally, describe distribution using a barplot!

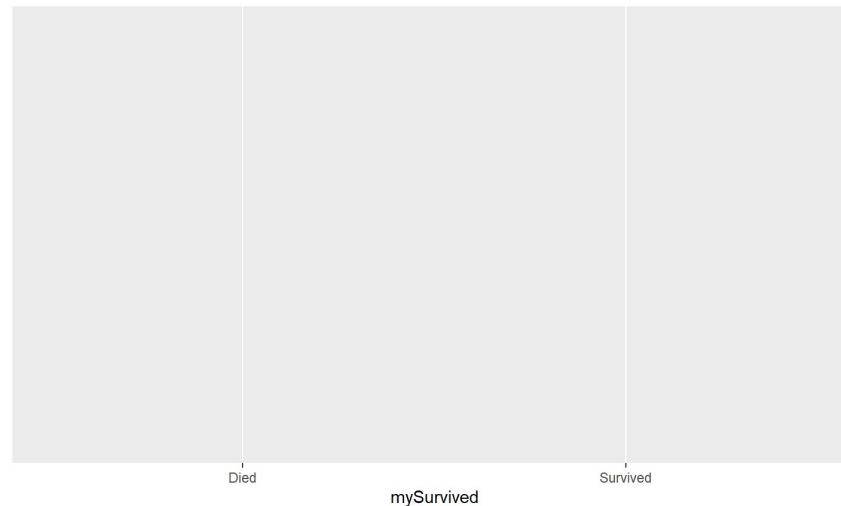
- Barplots via ggplot + geom\_bar

```
titanicData <- read_csv("../datasets/titanic.csv")
titanicData$mySurvived <- as.factor(titanicData$survived)
levels(titanicData$mySurvived) <- c("Died", "Survived")
titanicData$myEmbarked <- as.factor(titanicData$embarked)
levels(titanicData$myEmbarked) <- c("Cherbourg", "Queenstown", "Southampton")
titanicData <- titanicData %>% drop_na(mySurvived, sex, myEmbarked)
```

# ggplot2 barplots

- Barplots via `ggplot + geom_bar`
- Across x-axis we want our categories - specify with `aes(x = ...)`

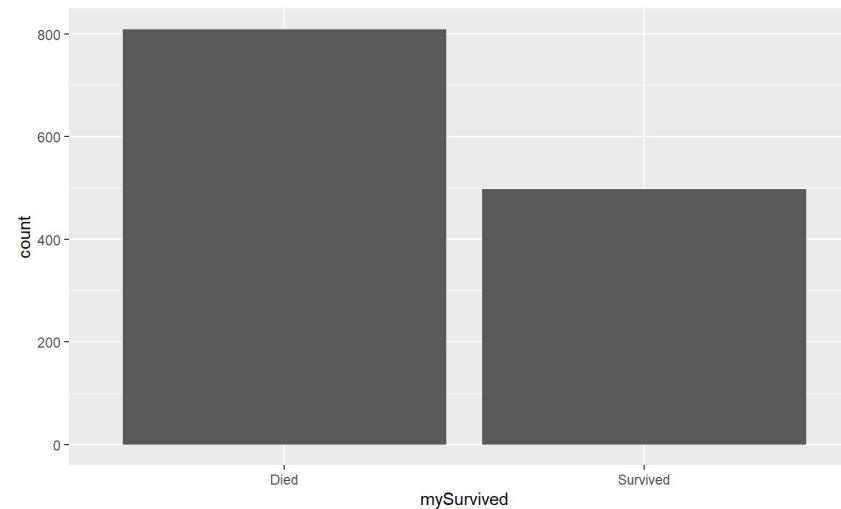
```
ggplot(data = titanicData, aes(x = mySurvived) )
```



# ggplot2 barplots

- Barplots via `ggplot + geom_bar`
- Must add `geom` (or `stat`) layer!

```
ggplot(data = titanicData, aes(x = mySurvived)) + geom_bar()
```



# ggplot2 barplots

- Generally: Save base object, then “add layers”

```
g <- ggplot(data = titanicData, aes(x = mySurvived))  
g + geom_bar()
```

# ggplot2 barplots

- Generally: Save base object, then “add layers”

```
g <- ggplot(data = titanicData, aes(x = mySurvived))  
g + geom_bar()
```

- `aes()` defines visual properties of objects in the plot

`x = , y = , size = , shape = , color = , alpha = , ...`

- [Cheat Sheet](#) gives most common properties for a given `geom`

# ggplot2 barplots

- Generally: Save base object, then “add layers”

```
g <- ggplot(data = titanicData, aes(x = mySurvived))  
g + geom_bar()
```

- `aes()` defines visual properties of objects in the plot

`x = , y = , size = , shape = , color = , alpha = , ...`

- Cheat Sheet gives most common properties for a given `geom`

```
d + geom_bar()
```

`x, alpha, color, fill, linetype, size, weight`

# ggplot2 global and local aesthetics

data and aes can be set in two ways;

- 'globally' (for all layers) via the `ggplot` statement
- 'locally' (for just that layer) via the `geom`, `stat`, etc. layer

# ggplot2 global and local aesthetics

data and aes can be set in two ways;

- 'globally' (for all layers) via the `ggplot` statement
- 'locally' (for just that layer) via the `geom`, `stat`, etc. layer

```
#global  
ggplot(data = titanicData, aes(x = mySurvived)) + geom_bar()  
#local  
ggplot() + geom_bar(data = titanicData, aes(x = mySurvived))
```

# ggplot2 global and local aesthetics

data and aes can be set in two ways;

- 'globally' (for all layers) via the `ggplot` statement
- 'locally' (for just that layer) via the `geom`, `stat`, etc. layer

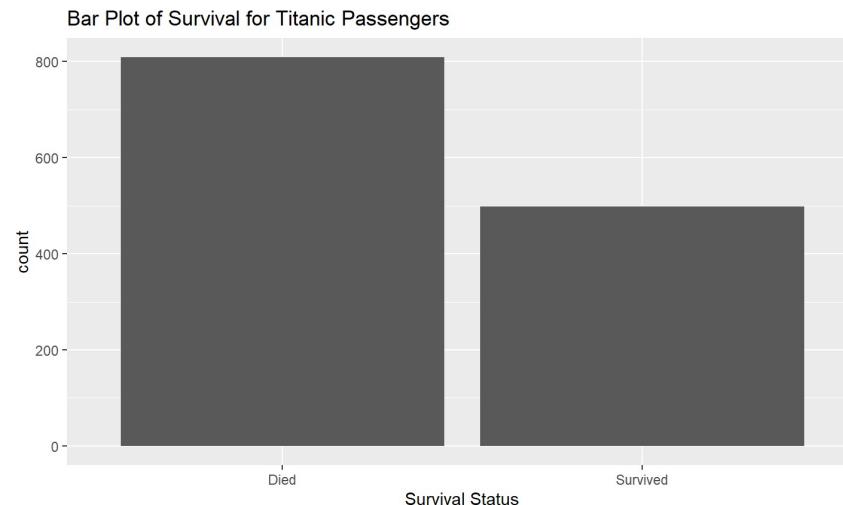
```
#global  
ggplot(data = titanicData, aes(x = mySurvived)) + geom_bar()  
#local  
ggplot() + geom_bar(data = titanicData, aes(x = mySurvived))
```

- To set an attribute that doesn't depend on the data (i.e. `color = 'blue'`), generally place these outside of the `aes`

# ggplot2 barplots

- Add better labels and a title (new layers, see cheat sheet!)

```
ggplot(data = titanicData, aes(x = mySurvived)) +  
  geom_bar() +  
  labs(x = "Survival Status", title = "Bar Plot of Survival for Titanic Passengers")
```



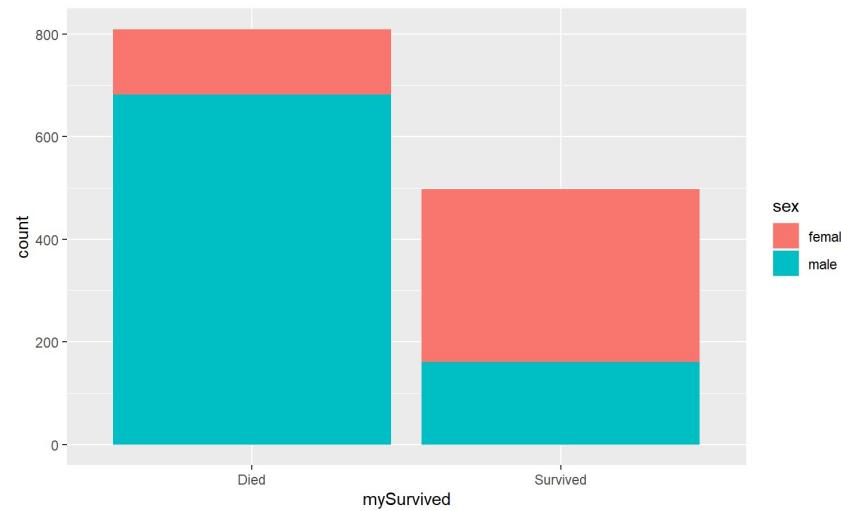
# ggplot2 stacked barplots

- Stacked barplot created by via `fill` aesthetic and same process
  - Create base object
  - Add geoms
  - Use `aes` to specify aspects of the plot

# ggplot2 stacked barplots

- Stacked barplot created by via `fill` aesthetic
- Automatic assignment of colors, creation of legends, etc. for `aes` elements (except with `group`)

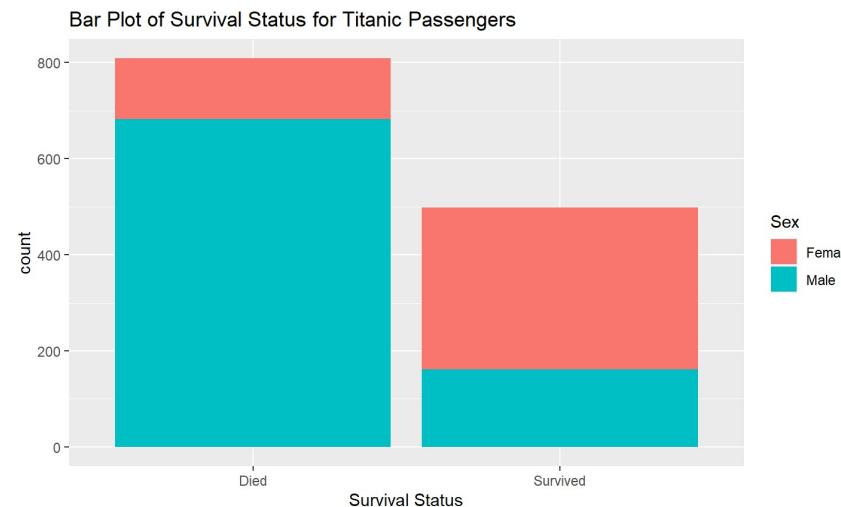
```
ggplot(data = titanicData, aes(x = mySurvived, fill = sex)) + geom_bar()
```



# ggplot2 labeling

- Add custom labels by adding more layers

```
ggplot(data = titanicData, aes(x = mySurvived, fill = sex)) +  
  geom_bar() +  
  labs(x = "Survival Status",  
       title = "Bar Plot of Survival Status for Titanic Passengers") +  
  scale_fill_discrete(name = "Sex", labels = c("Female", "Male"))
```



# ggplot2 labeling

- Adjusting appropriate labeling via `scale_*_discrete`

```
aes(x = survived, fill = sex)

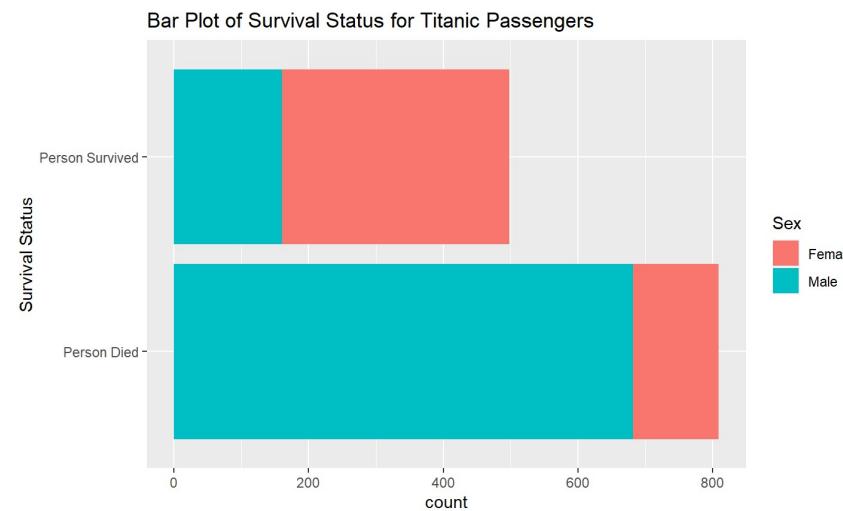
scale_x_discrete(labels = c("Person Died", "Person Survived"))

scale_fill_discrete(name = "Sex", labels = c("Female", "Male"))
```

# ggplot2 horizontal barplots

- Easy to rotate a plot with `coord_flip`

```
ggplot(data = titanicData, aes(x = mySurvived, fill = sex)) + geom_bar() +  
  labs(x = "Survival Status",  
       title = "Bar Plot of Survival Status for Titanic Passengers") +  
  scale_x_discrete(labels = c("Person Died", "Person Survived")) +  
  scale_fill_discrete(name = "Sex", labels = c("Female", "Male")) +  
  coord_flip()
```



# ggplot2 stat vs geom layers

Note: Most geoms have a corresponding stat that can be used

```
geom_bar(mapping = NULL, data = NULL, stat = "count", position =  
"stack", ..., width = NULL, binwidth = NULL, na.rm = FALSE,  
show.legend = NA, inherit.aes = TRUE)
```

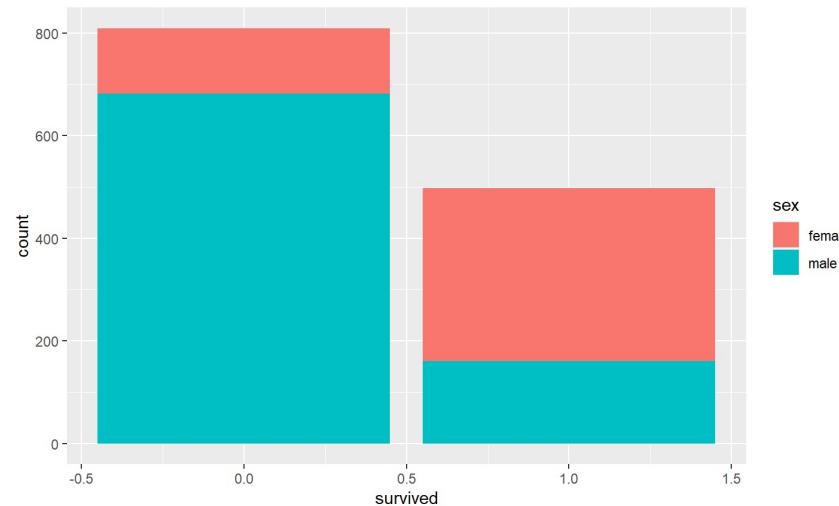
- Equivalent plots via:

```
ggplot(data = titanicData, aes(x = survived, fill = sex)) + geom_bar()  
ggplot(data = titanicData, aes(x = survived, fill = sex)) + stat_count()
```

# ggplot2 stat vs geom layers

- Can modify the stat: if you have summary data, use `identity`

```
sumData <- titanicData %>%
  group_by(survived, sex) %>%
  summarize(count = n())
ggplot(sumData, aes(x = survived, y = count, fill = sex)) +
  geom_bar(stat = "identity")
```



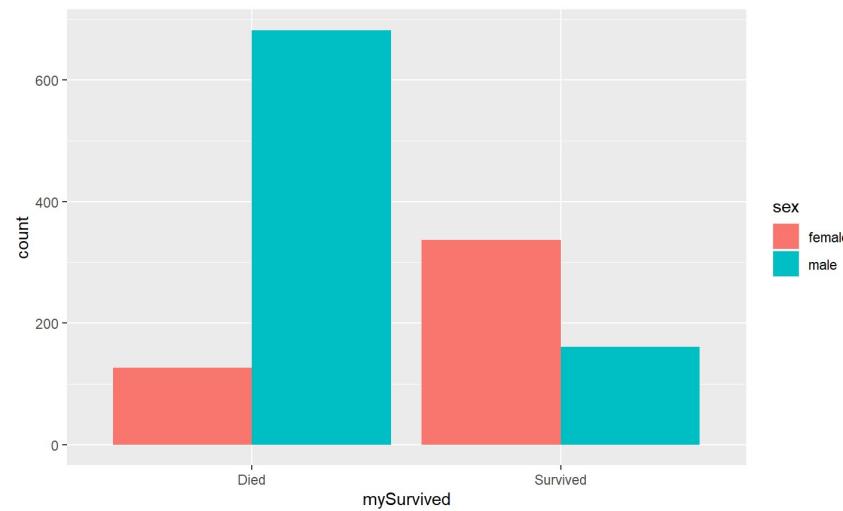
# ggplot2 side-by-side barplots

- **Side-by-side barplot** created by via position aesthetic
  - `dodge` for side-by-side bar plot
  - `jitter` for continuous data with many points at same values
  - `fill` stacks bars and standardises each stack to have constant height
  - `stack` stacks bars on top of each other

# ggplot2 side-by-side barplots

- Side-by-side barplot created by via position aesthetic

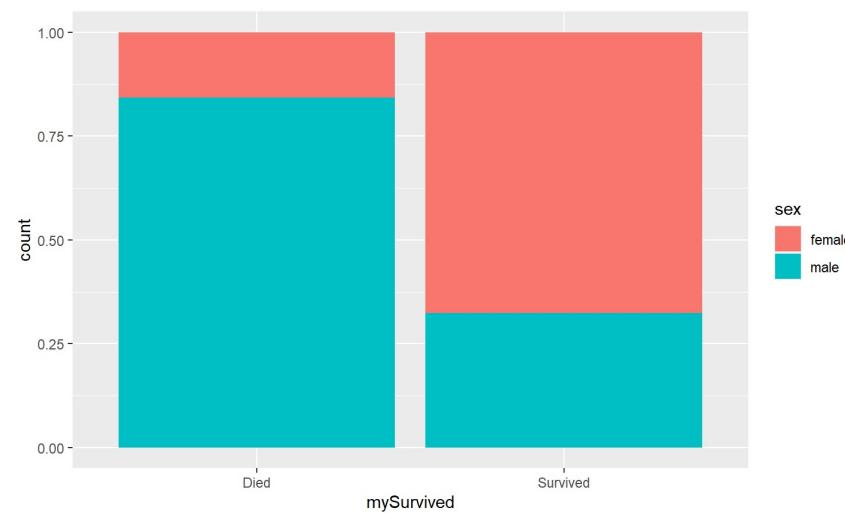
```
ggplot(data = titanicData, aes(x = mySurvived, fill = sex)) +  
  geom_bar(position = "dodge")
```



# ggplot2 filled barplots

- `position = "fill"` stacks bars and standardises each stack to have constant height (especially useful with equal group sizes)

```
ggplot(data = titanicData, aes(x = mySurvived, fill = sex)) +  
  geom_bar(position = "fill")
```



# ggplot2 faceting

How to create same plot for each `myEmbarked` value? Use **faceting!**

# ggplot2 faceting

How to create same plot for each `myEmbarked` value? Use **faceting!**

`facet_wrap(~ var)` - creates a plot for each setting of `var`

- Can specify `nrow` and `ncol` or let R figure it out

# ggplot2 faceting

How to create same plot for each `myEmbarked` value? Use **faceting!**

`facet_wrap(~ var)` - creates a plot for each setting of `var`

- Can specify `nrow` and `ncol` or let R figure it out

`facet_grid(var1 ~ var2)` - creates a plot for each combination of `var1` and `var2`

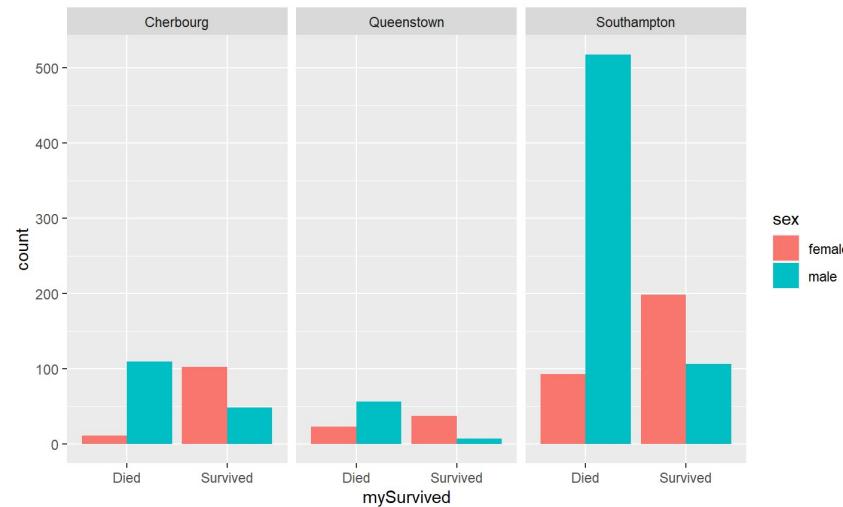
- `var1` values across rows
- `var2` values across columns
- Use `. ~ var2` or `var1 ~ .` to have only one row or column

# ggplot2 faceting

How to create same plot for each `myEmbarked` value? Use **faceting!**

- `facet_wrap(~ var)` - creates a plot for each setting of `var`

```
ggplot(data = titanicData, aes(x = mySurvived)) +  
  geom_bar(aes(fill = sex), position = "dodge") +  
  facet_wrap(~ myEmbarked)
```



# ggplot2 Plotting Recap

General `ggplot` things:

- Can set local or global `aes`
- Modify titles/labels by adding more layers
- Faceting (multiple plots) via `facet_grid` or `facet_wrap`
- Only need `aes` if setting a mapping value that is dependent on the data (or you want to create a custom legend!)

# Quick Examples

# ggplot2 Plotting: Numeric Variables

Numeric variables - generally, describe distribution via a histogram or boxplot!

Same process:

- Create base object
- Add geoms
- Use aes to specify aspects of the plot

# ggplot2 smoothed histogram

- **Kernel Smoother** - Smoothed version of a histogram
- Common `aes` values (from cheat sheet):

```
c + geom_density(kernel = "gaussian")
```

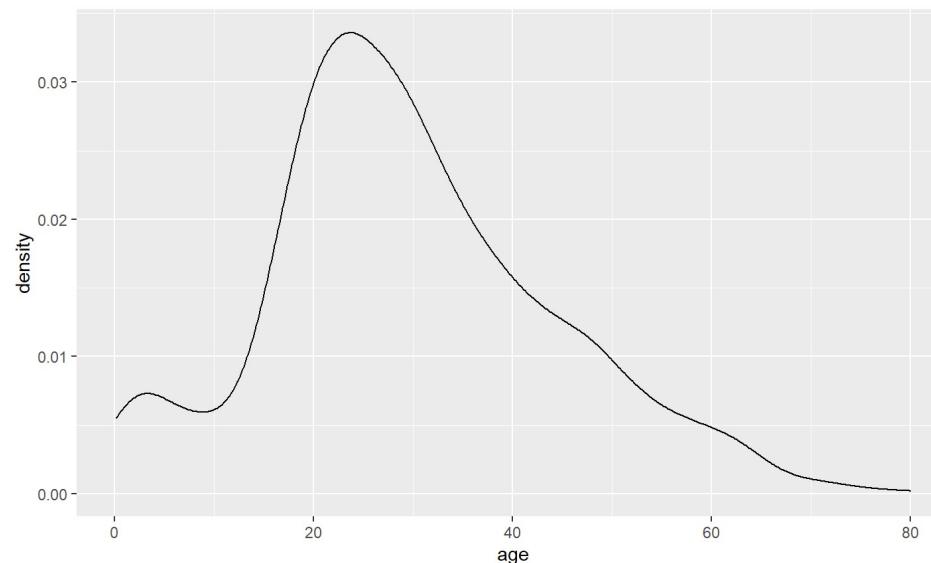
```
x, y, alpha, color, fill, group, linetype, size, weight
```

- Only `x` = is really needed

# ggplot2 smoothed histogram

- Kernel Smoother - Smoothed version of a histogram

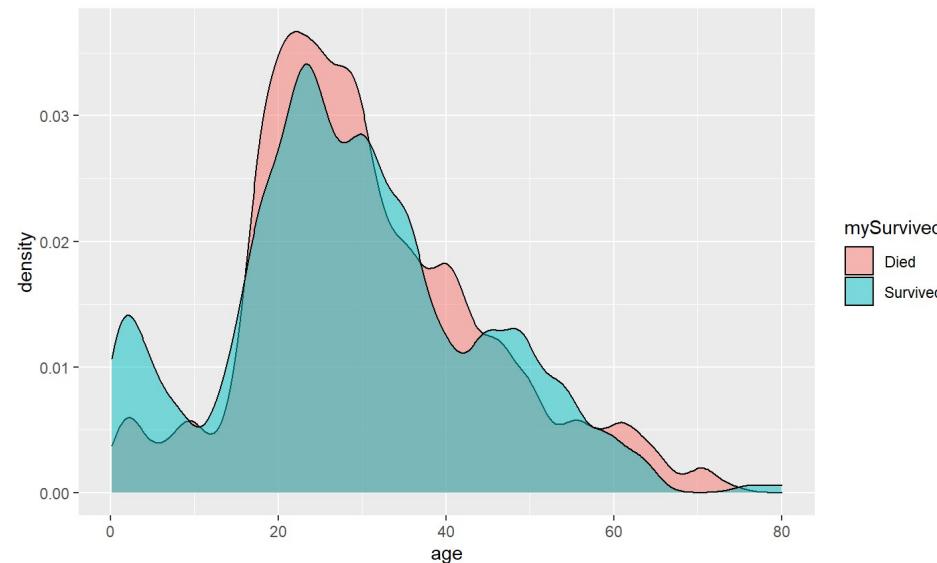
```
g <- ggplot(titanicData, aes(x = age))  
g + geom_density()
```



# ggplot2 smoothed histogram

- Kernel Smoother - Smoothed version of a histogram
- fill a useful aesthetic!

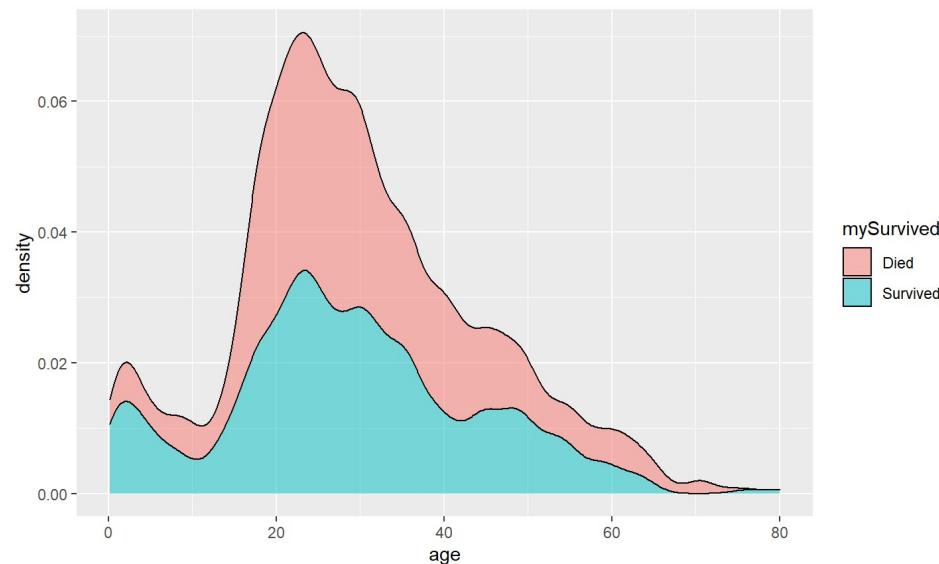
```
g + geom_density(adjust = 0.5, alpha = 0.5, aes(fill = mySurvived))
```



# ggplot2 smoothed histogram

- Kernel Smoother - Smoothed version of a histogram
- recall position choices of dodge, jitter, fill, and stack

```
g + geom_density(adjust = 0.5, alpha = 0.5, position = "stack", aes(fill = mySurvived))
```



# ggplot2 boxplots

- **Boxplot** - Provides the five number summary in a graph
- Common `aes` values (from cheat sheet):

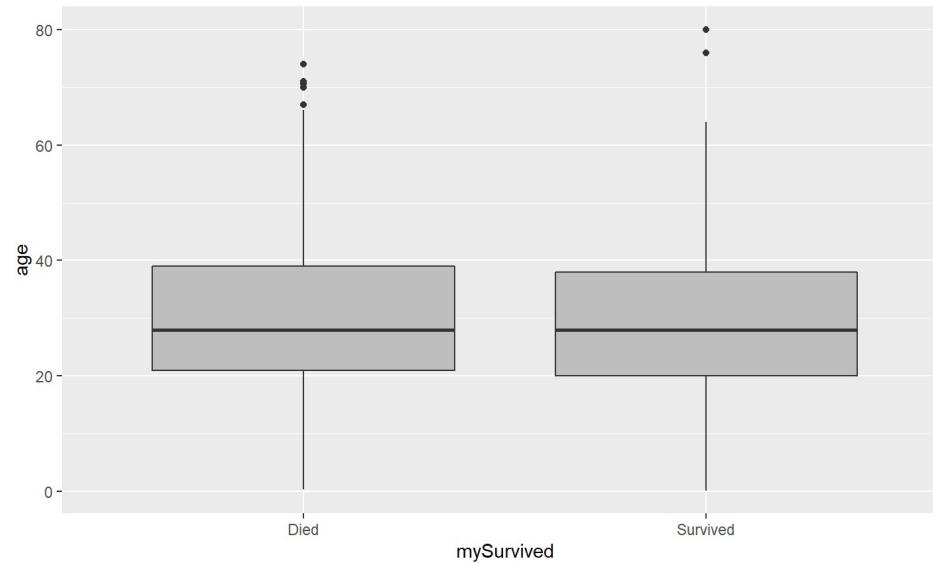
```
f + geom_boxplot()
```

```
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill,  
group, linetype, shape, size, weight
```

- Only `x =`, `y =` are really needed

# ggplot2 boxplots

```
g <- ggplot(titanicData, aes(x = mySurvived, y = age))  
g + geom_boxplot(fill = "grey")
```

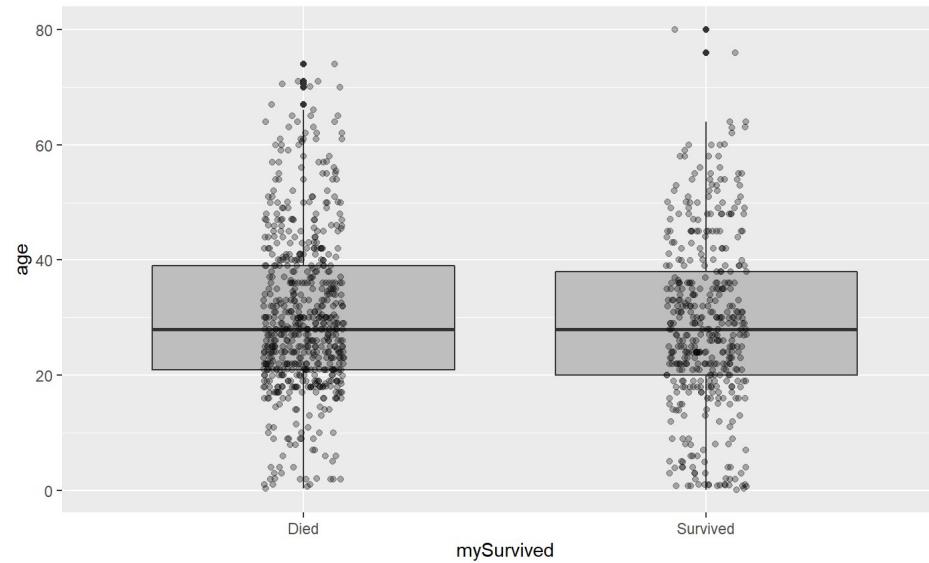


# ggplot2 boxplots with points

- Can add data points (jittered) to see shape of data better (or use violin plot)

```
g +
```

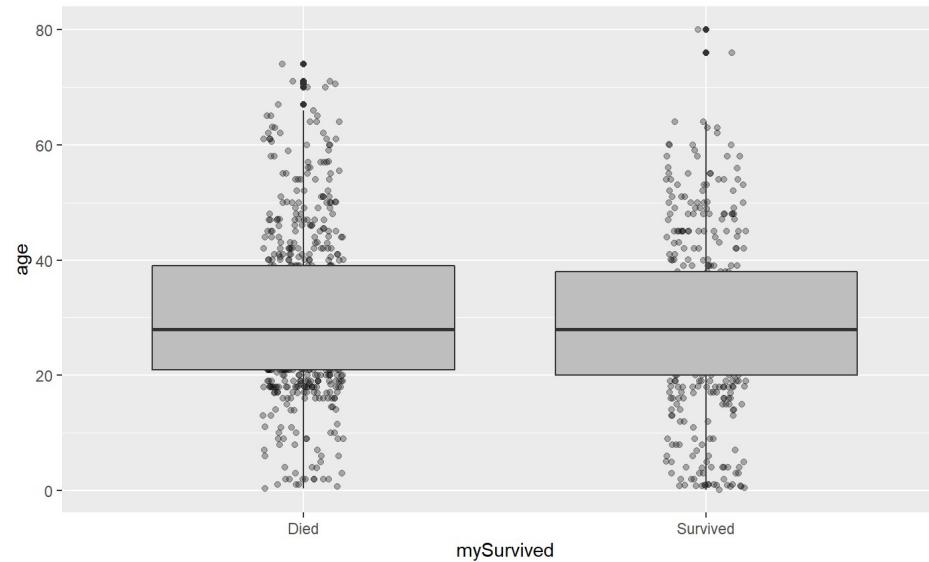
```
geom_boxplot(fill = "grey") +  
geom_jitter(width = 0.1, alpha = 0.3)
```



# ggplot2 boxplots with points

- Order of layers important!

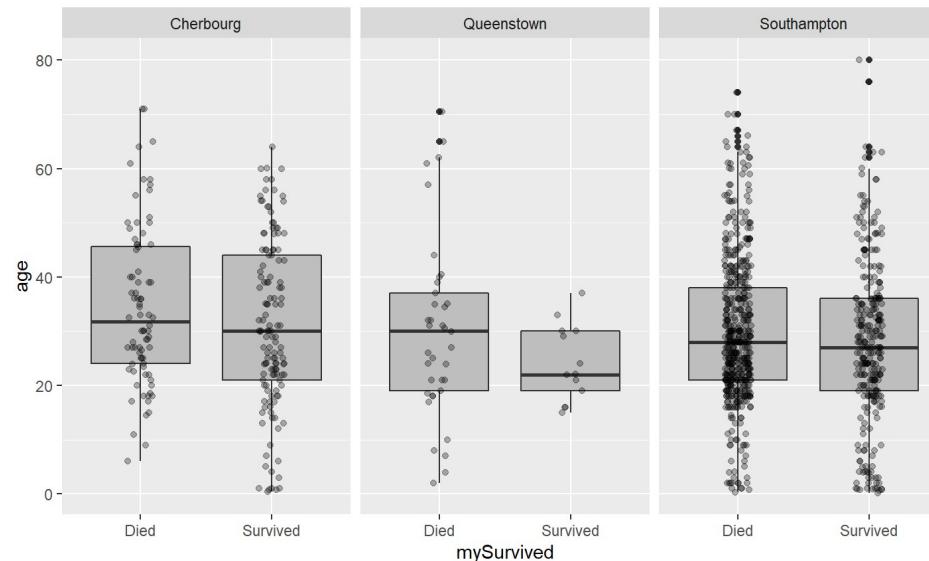
```
g +  
  geom_jitter(width = 0.1, alpha = 0.3) +  
  geom_boxplot(fill = "grey")
```



# ggplot2 faceting

- Can facet easily!

```
g + geom_boxplot(fill = "grey") +
  geom_jitter(width = 0.1, alpha = 0.3) +
  facet_wrap(~ myEmbarked)
```



# ggplot2 scatter plots

Two numerical variables

- **Scatter Plot** - graphs points corresponding to each observation
- Common `aes` values (from cheat sheet):

`e + geom_point()`

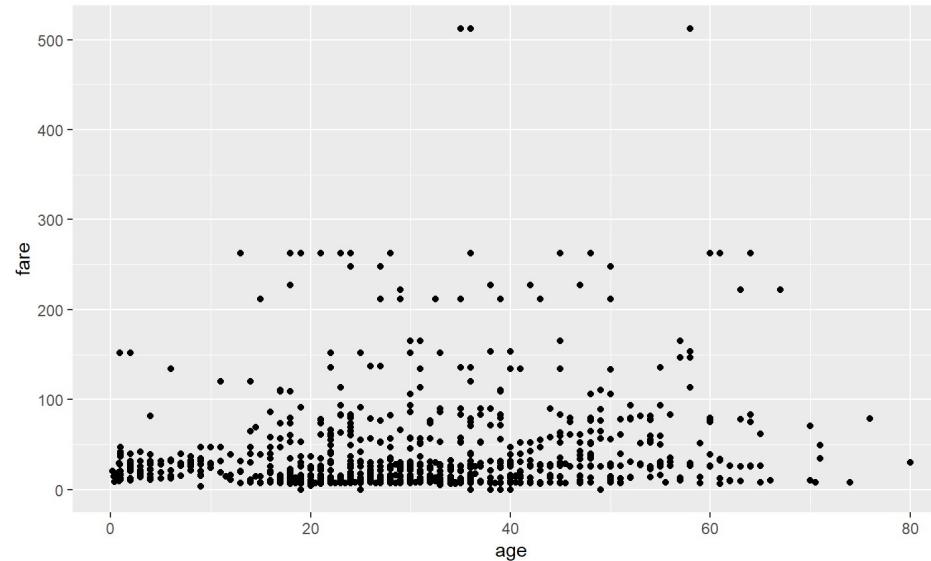
`x, y, alpha, color, fill, shape, size, stroke`

- Only `x =`, `y =` are really needed

# ggplot2 scatter plots

- **Scatter Plot** - graphs points corresponding to each observation

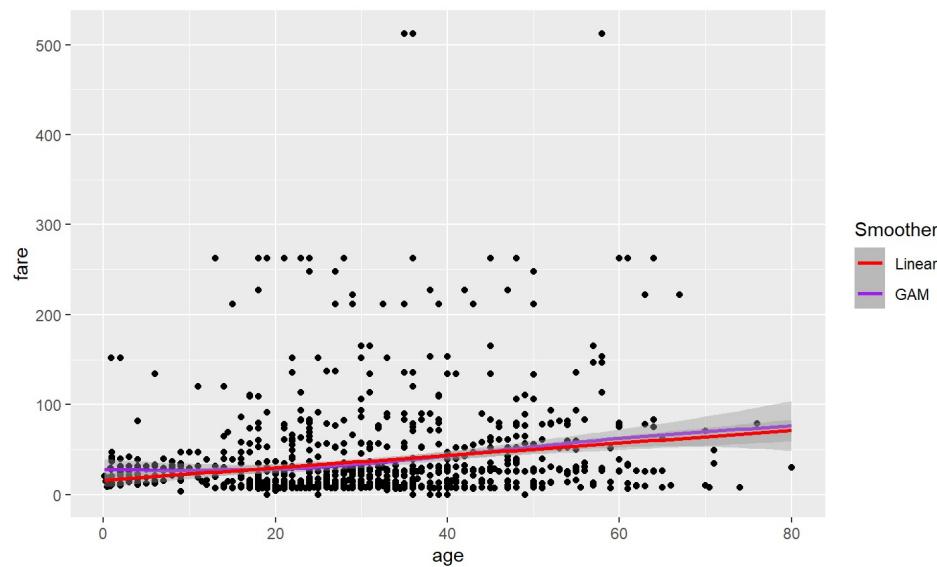
```
g <- ggplot(titanicData, aes(x = age, y = fare))  
g + geom_point()
```



# ggplot2 scatter plots with trend line

- Add trend lines easily (linear and loess - a smoother)

```
g + geom_point() +
  geom_smooth(aes(col = "loess")) +
  geom_smooth(method = lm, aes(col = "linear")) +
  scale_colour_manual(name = 'Smoother', values =c('linear'='red', 'loess'='purple'),
                      labels = c('Linear', 'GAM'), guide = 'legend')
```



# ggplot2 scatter plots with text

- May want to add value of correlation to plot
- `paste()` or `paste0()` handy

```
paste("Hi", "What", "Is", "Going", "On", "?", sep = " ")
```

```
## [1] "Hi What Is Going On ?"
```

```
paste("Hi", "What", "Is", "Going", "On", "?", sep = ".")
```

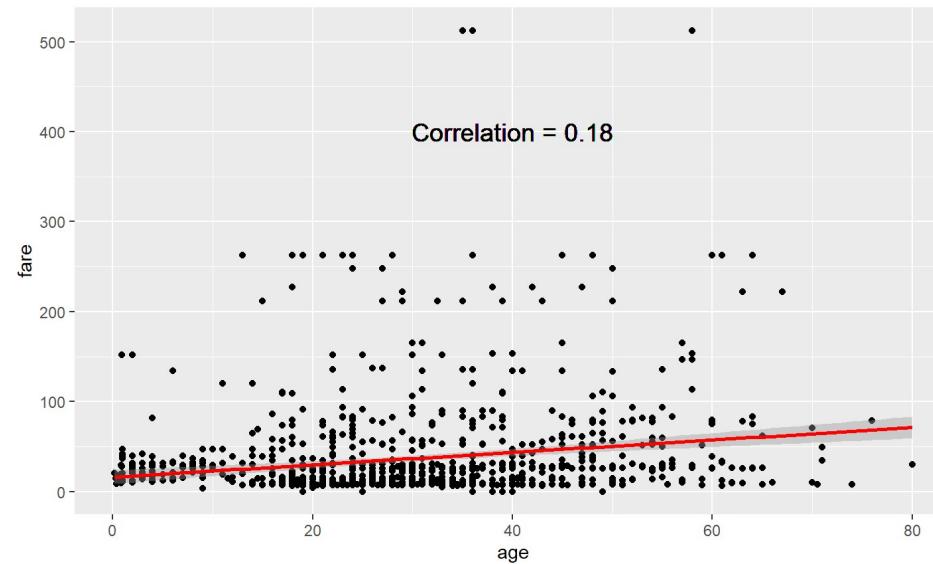
```
## [1] "Hi.What.Is.Going.On.?"
```

```
paste0("Hi", "What", "Is", "Going", "On", "?")
```

```
## [1] "HiWhatIsGoingOn?"
```

# ggplot2 scatter plots with text

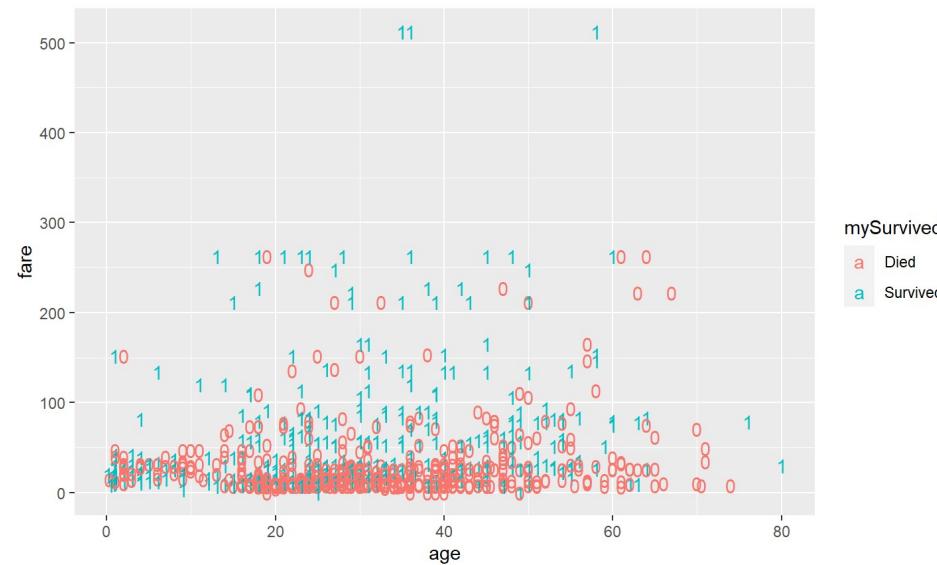
```
correlation <- cor(titanicData$fare, titanicData$age, use = "complete.obs")
g + geom_point() +
  geom_smooth(method = lm, col = "Red") +
  geom_text(x = 40, y = 400, size = 5,
            label = paste0("Correlation = ", round(correlation, 2)))
```



# ggplot2 scatter plots with text points

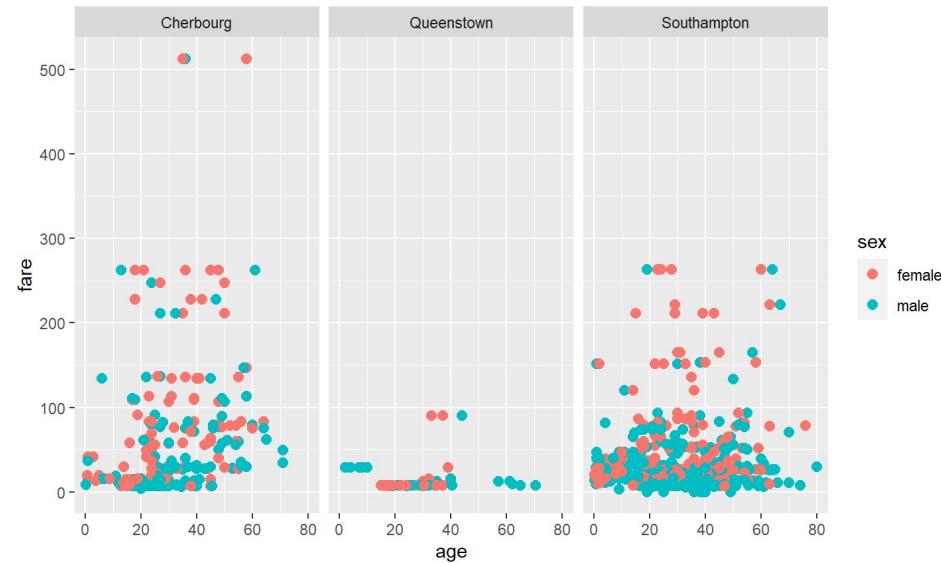
- Text for points with `geom_text`

```
g + geom_text(aes(label = survived, color = mySurvived))
```



# ggplot2 faceting

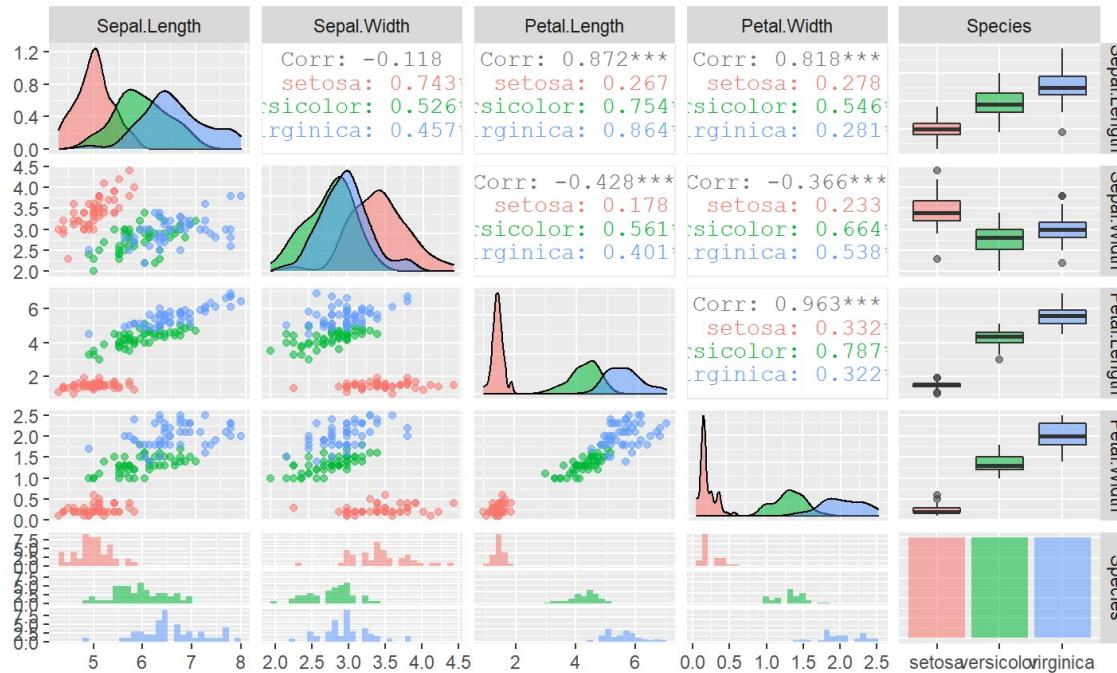
```
g + geom_point(aes(color = sex), size = 2.5) +  
  facet_wrap(~ myEmbarked)
```



# ggpairs

Many extension packages that do nice things!

```
library(GGally) #install GGally if needed  
ggpairs(iris, aes(colour = Species, alpha = 0.4))
```



# ggplot2 Plotting: Numeric variables

Numeric variable - entries are a numerical value where math can be performed

Most common plots:

- Histogram (`geom_hist`), Density (`geom_density`)
- Boxplot (`geom_boxplot`), Violin plot (`geom_violin`)
- Scatter plot (`geom_point`), Smoothers (`geom_smooth`)
- Jittered points (`geom_jitter`)
- Text on plot (`geom_text`)

# ggplot2 Plotting Recap

General `ggplot` things:

- Can set local or global `aes`
- Modify titles/labels by adding more layers
- Use either `stat` or `geom` layer
- Faceting (multiple plots) via `facet_grid` or `facet_wrap`
- Only need `aes` if setting a mapping value that is dependent on the data (or you want to create a custom legend!)
- `esquisse` is a [great package for exploring ggplot2!](#)

# Quick Examples