Psychological Statistics Lab 3

PSYC 2020-A01 / PSYC 6022-A01 | 2025-09-04 | Descriptive Statistics II

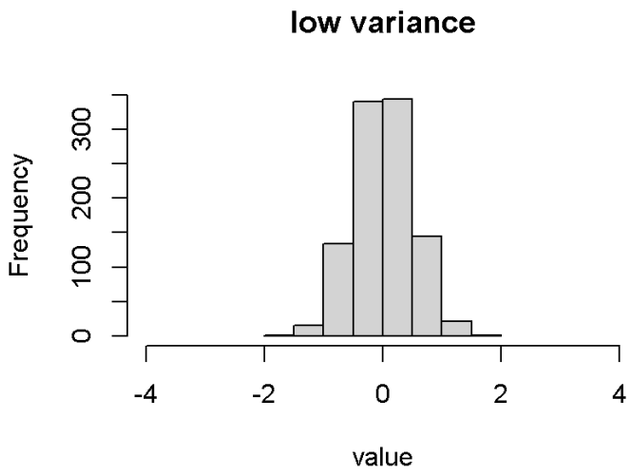
Jessica Helmer

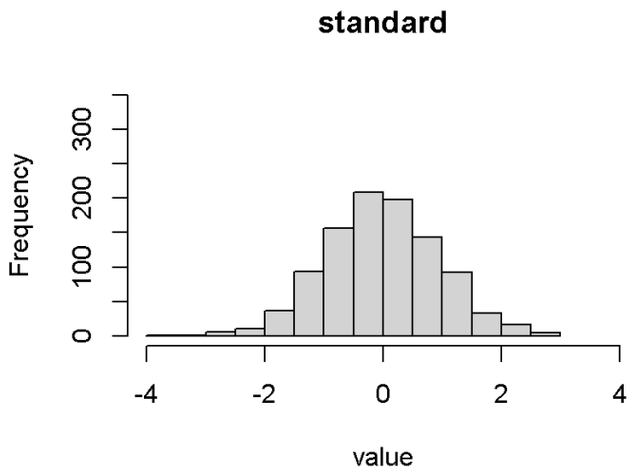
Outline

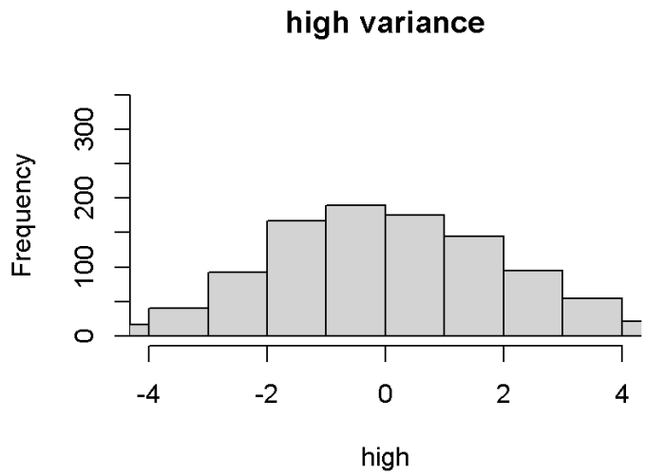
- Assignment 2 Review
- More on types of data
- Labeling data
- Aggregating data
- Descriptives by group
- Weighted mean

Measures of Variability Review

Describe the “spread” or “dispersion” of the data







Measures of Variability Functions

```
1 testscores <- c(88, 93, 92, 99, 96)
```

Variance: `var()` function

Standard Deviation: `sd()` function

Interquartile Range: `IQR()` function

- Difference between the 3rd and 1st quantile (so 50% of the data within this range)
- 25% of the data lower than the 1st quantile
- 75% of the data lower than the 3rd quantile
- IQR = 3rd - 1st

R Code | Start Over | Run Code

```
1 # let's look at some variances!
```

Quantiles: `quantile()` function

Object Types: Vector

We've already seen and used vectors

```
1 c(2, 3, 4, 2)
```

```
[1] 2 3 4 2
```

```
1 c("Cat", "Dog", "Bird")
```

```
[1] "Cat"  "Dog"  "Bird"
```

We know they can only hold one type of data

```
1 c("Cat", "Dog", 4)
```

```
[1] "Cat" "Dog" "4"
```

Shorthand for sequenced numbers:

```
1 1:10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Object Types: Matrix

Like a vector, but multidimensional

Can only hold one type of data

- Worse than a dataframe?
- Okay! But pop up in function output often
- So we need to know how to work with them

```
matrix(c(1, 2, 3, 4), nrow = 2)
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Default is to fill in columns first

```
matrix(c(1, 2, 3, 4), nrow = 2, byrow = T)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Object Types: Matrix

Can give names

`colnames()` Function

```
1 mymatrix <- matrix(c(0, 1, 2, 1, 2, 3, 2, 3,
2                      nrow = 3, byrow = T)
3 colnames(mymatrix) <- c("Var1", "Var2", "Var
4 mymatrix
```

```
     Var1 Var2 Var3
[1,]    0    1    2
[2,]    1    2    3
[3,]    2    3    4
```

Retrieves object column names, then assigns them something else

`dimnames` Argument

```
1 mymatrix <- matrix(c(0, 1, 2, 1, 2, 3, 2, 3,
2                      nrow = 3, byrow = T,
3                      dimnames = list(NULL, c("
4 mymatrix
```

```
     Var1 Var2 Var3
[1,]    0    1    2
[2,]    1    2    3
[3,]    2    3    4
```

Sets names while creating the matrix Takes a `list(row_names, colum_names)`

Object Types: List

Limitations of vectors:

- Can only hold one type of data

Enter the `list`

Can create by calling the `list()` function and giving it some objects

```
1  list(1, 2, "cat")
```

```
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] "cat"
```

```
1  list(1:4, c("a", "b"))
```

```
[[1]]
[1] 1 2 3 4

[[2]]
[1] "a" "b"
```

Object Types: Dataframes

More review

Like matrices, but can hold different types of data

```
1 matrix(c("cat", 2, "dog", 2),
2        nrow = 2, byrow = T,
3        dimnames = list(NULL, c("Pet", "Count"
```

```
     Pet   Count
[1,] "cat" "2"
[2,] "dog" "2"
```

```
1 mydataframe <- data.frame(
2   pet = c("cat", "dog"),
3   count = c(2, 2))
4 mydataframe
```

```
  pet count
1 cat     2
2 dog     2
```

Indexing

To access some subset of an object (vector, list, dataframe, etc.)

We already know one! The `$` operator indexes by selecting a column

```
1 mydataframe$pet
```

```
[1] "cat" "dog"
```

Indexing

To access some subset of an object (vector, list, dataframe, etc.)

Can also use brackets [ ]

```
1 myvector <- c(2, 4, 6, 8)
2
3 myvector[1]
```

```
[1] 2
```

Can select more than one position at a time

```
1 myvector[c(1, 2, 3)]
```

```
[1] 2 4 6
```

```
1 myvector[1:3]
```

```
[1] 2 4 6
```

Indexing

Need to account for multiple dimensions

Matrix: `[element position]` or `[row, column]`

```
1 mymatrix
```

```
     Var1 Var2 Var3
[1,]    0    1    2
[2,]    1    2    3
[3,]    2    3    4
```

```
1 mymatrix[1]
```

```
[1] 0
```

```
1 mymatrix[4]
```

```
[1] 1
```

```
1 mymatrix[2, ]
```

```
Var1 Var2 Var3
   1    2    3
```

```
1 mymatrix[3, 2]
```

```
Var2
```

Indexing

Dataframe: `[column]` or `[row, column]`

```
1 mydataframe[1]
```

```
  pet
1 cat
2 dog
```

```
1 mydataframe["pet"]
```

```
  pet
1 cat
2 dog
```

```
1 mydataframe$pet
```

```
[1] "cat" "dog"
```

```
1 mydataframe[2, "pet"]
```

```
[1] "dog"
```

Can also do more than one element at a time:

```
1 mydataframe[1:2, "count"]
```

```
[1] 2 2
```

Labeling Data

Adding meaningful labels or categories to your dataset.

Useful for organizing and interpreting your data.

- E.g., Adding labels like “Pass” or “Fail” based on test scores.

	id	order	rating
1	1	1	3
2	2	2	3
3	3	3	4
4	4	1	3
5	5	1	2

	id	order	rating
1	1	Pizza	Fair
2	2	Pasta	Fair
3	3	Salad	Good
4	4	Pizza	Fair
5	5	Pizza	Poor

Key Functions for Labeling Data

```r
scores <- c(85, 60, 45, 70, 50)
```

```r
gender <- c(1, 2, 1, 2, 2)
```

ifelse(test, yes, no)

For simple conditional labeling

```r
# Labeling as Pass/Fail
labels <- ifelse(scores >= 60,
                 "Pass", "Fail")

# Combining into a data frame
data <- data.frame(Scores = scores,
                   Label = labels)

data
```

```
  Scores Label
1     85  Pass
2     60  Pass
3     45  Fail
4     70  Pass
5     50  Fail
```

cut()

For creating bins or ranges

```r
# Creating bins
categories <- cut(scores,
                  breaks = c(0, 50, 70, 100),
                  labels = c("Low", "Medium",
                             "High"))

# Combining into a data frame
data <- data.frame(Scores = scores,
                   Category = categories)

data
```

```
  Scores Category
1     85     High
2     60   Medium
3     45      Low
4     70   Medium
5     50      Low
```

factor()

Convert numeric or character data into labeled categories

```r
# Converting to factor with labels
gender_labeled <- factor(gender,
                         levels = c(1, 2),
                         labels = c("Male",
                                    "Female"))

# Combine into a data frame
data <- data.frame(gender,
                   gender_labeled)

data
```

```
  gender gender_labeled
1      1           Male
2      2         Female
3      1           Male
4      2         Female
5      2         Female
```

Descriptives by Group

Aggregation involves summarizing or grouping data based on certain variables.

Commonly used to calculate summary statistics like mean, variance, or sum for each group.

Benefits:

- Data summary
- Data simplification
- Statistical analysis
- Trend identification
- Data privacy
- Data-driven decisions

```
aggregate(x ~ group, FUN)
```

Descriptives by Group Example

R Code | Start Over | Run Code

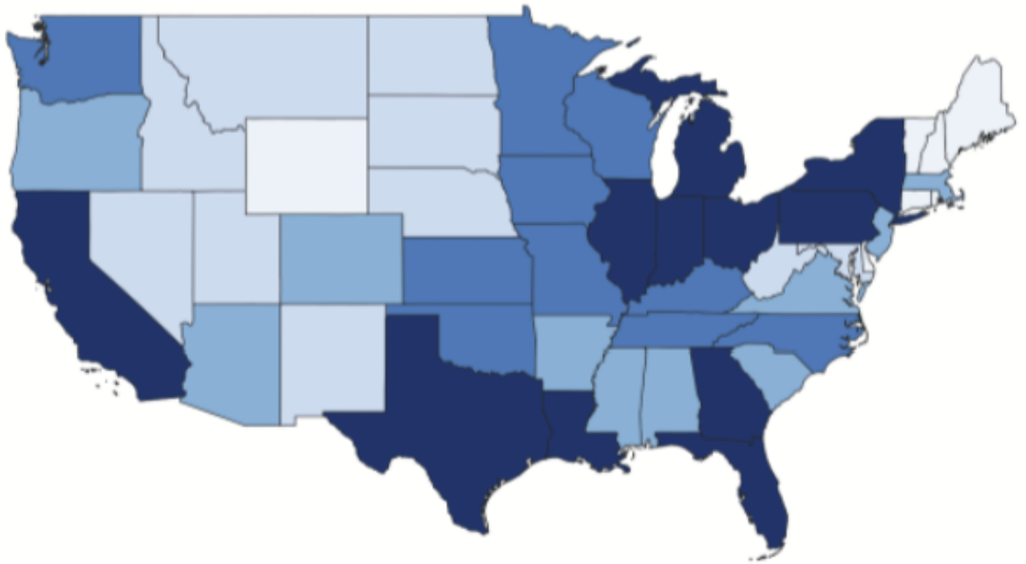
```
1  # average Sepal Length by iris species?
```

Descriptives by Group Visualization

Why visualize aggregated data?

- Makes group-level insights easier to interpret.
- Highlights patterns, trends, and outliers in grouped data
- Helps communicate findings effectively

(a) Point map of US hospitals



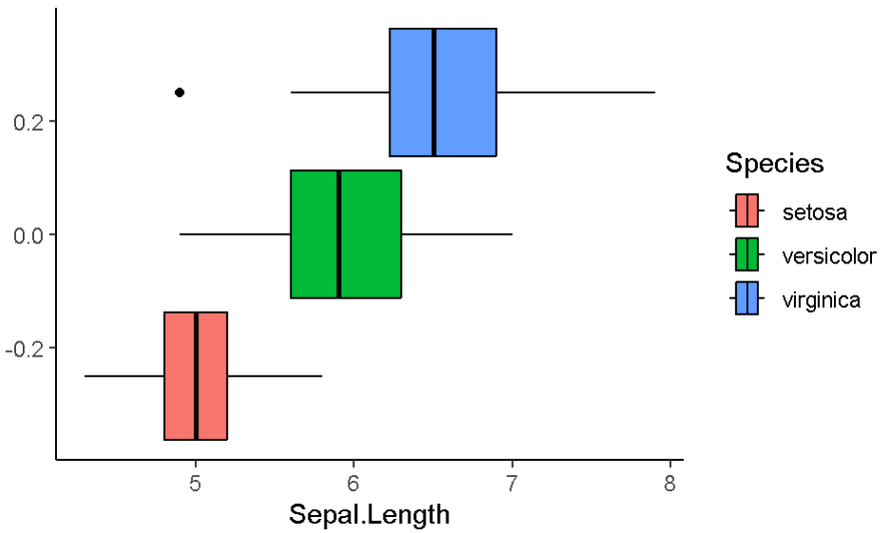
(b) Same data, but aggregated by state into a polygon (choropleth) map

Visualizations: Boxplots

Back to the `iris` dataset, distribution of Sepal Length by species

Anatomy of a boxplot:

- “Minimum”
- 25th Quantile (Q1)
- Median
- 75th Quantile (Q3)
- “Maximum”
- Points representing outliers



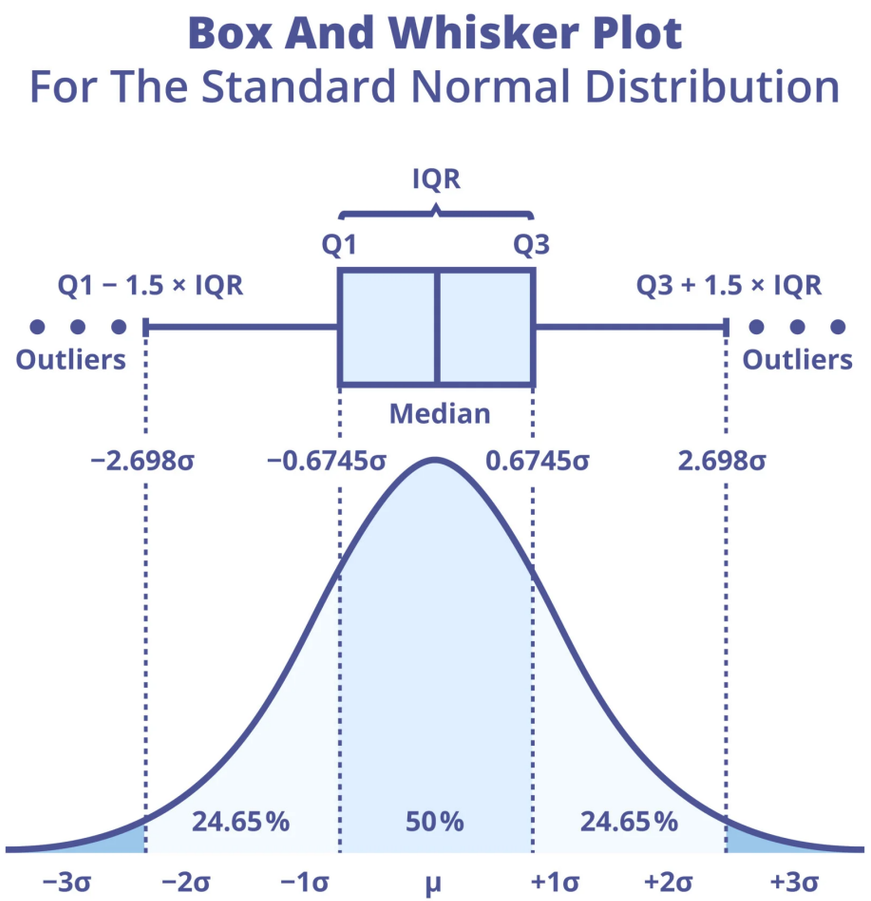
“Minimum” and “maximum” are not the *true* min and max

- Minimum: $Q1 - 1.5 * IQR$
- Maximum: $Q3 + 1.5 * IQR$

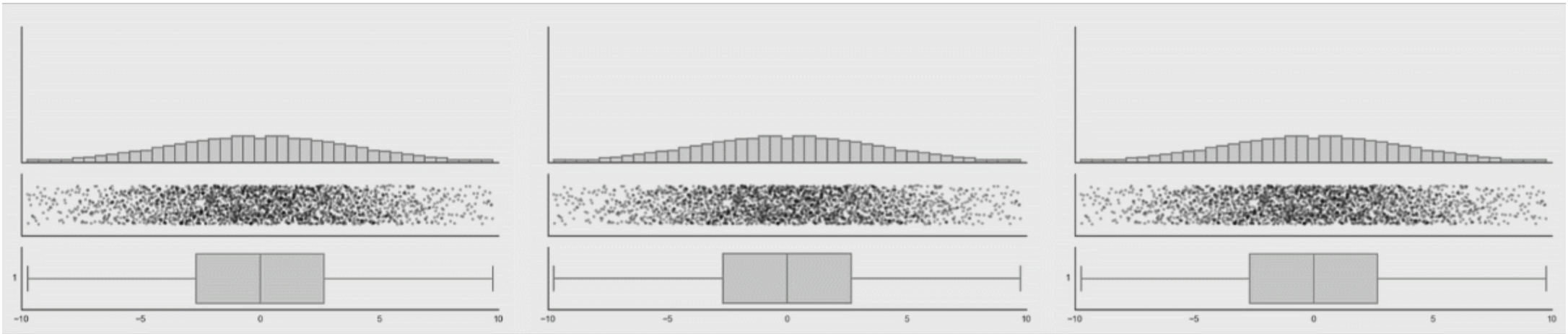
Means that the whiskers contain ~99% of the data, rest are outliers

Visualizations: Boxplots

One more resource for boxplot anatomy



Visualizations: Boxplots…



Datasaurus Dozen Boxplots

Let’s Do Some Visualization

Base R Graphics: Boxplot

`boxplot()` function

Its arguments are:

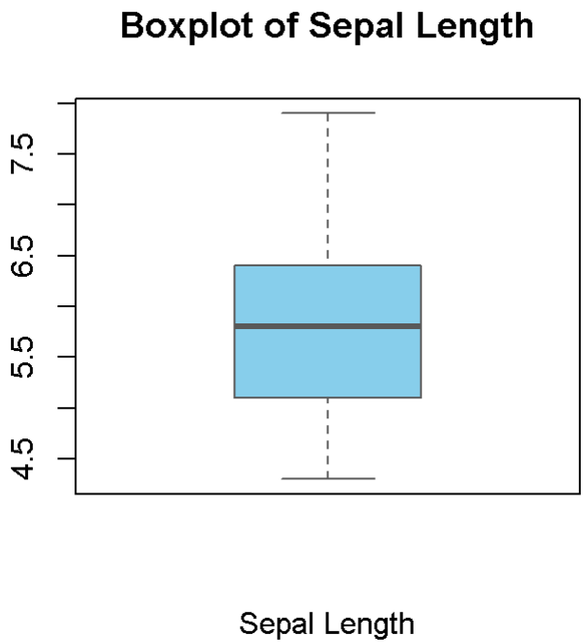
Required arguments:

- `x` = vector (variable) you want to plot

Optional arguments:

- `main`: title
- `xlab`: label for x-axis
- `ylab`: label for y-axis
- `border`: color for bar borders
- `col`: color for bars
- `horizontal`: `T`/`F` to switch

Plot Code



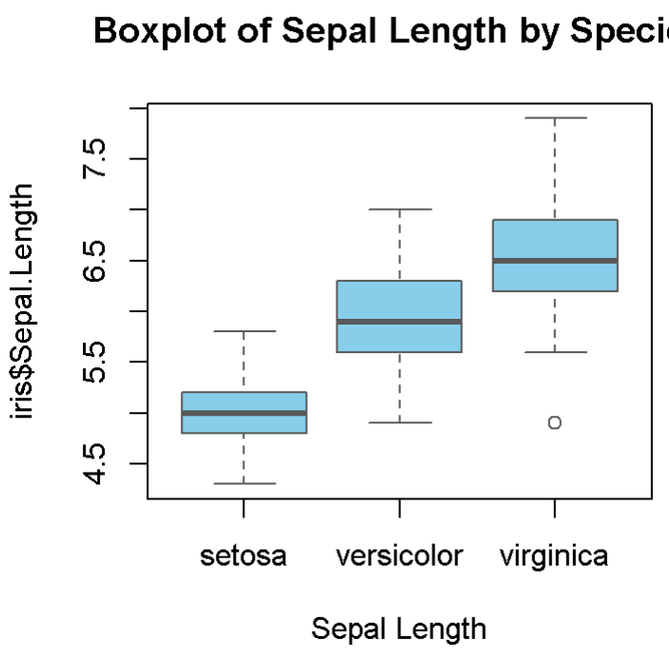
Let’s Do Some Visualization

Base R Graphics: Boxplot

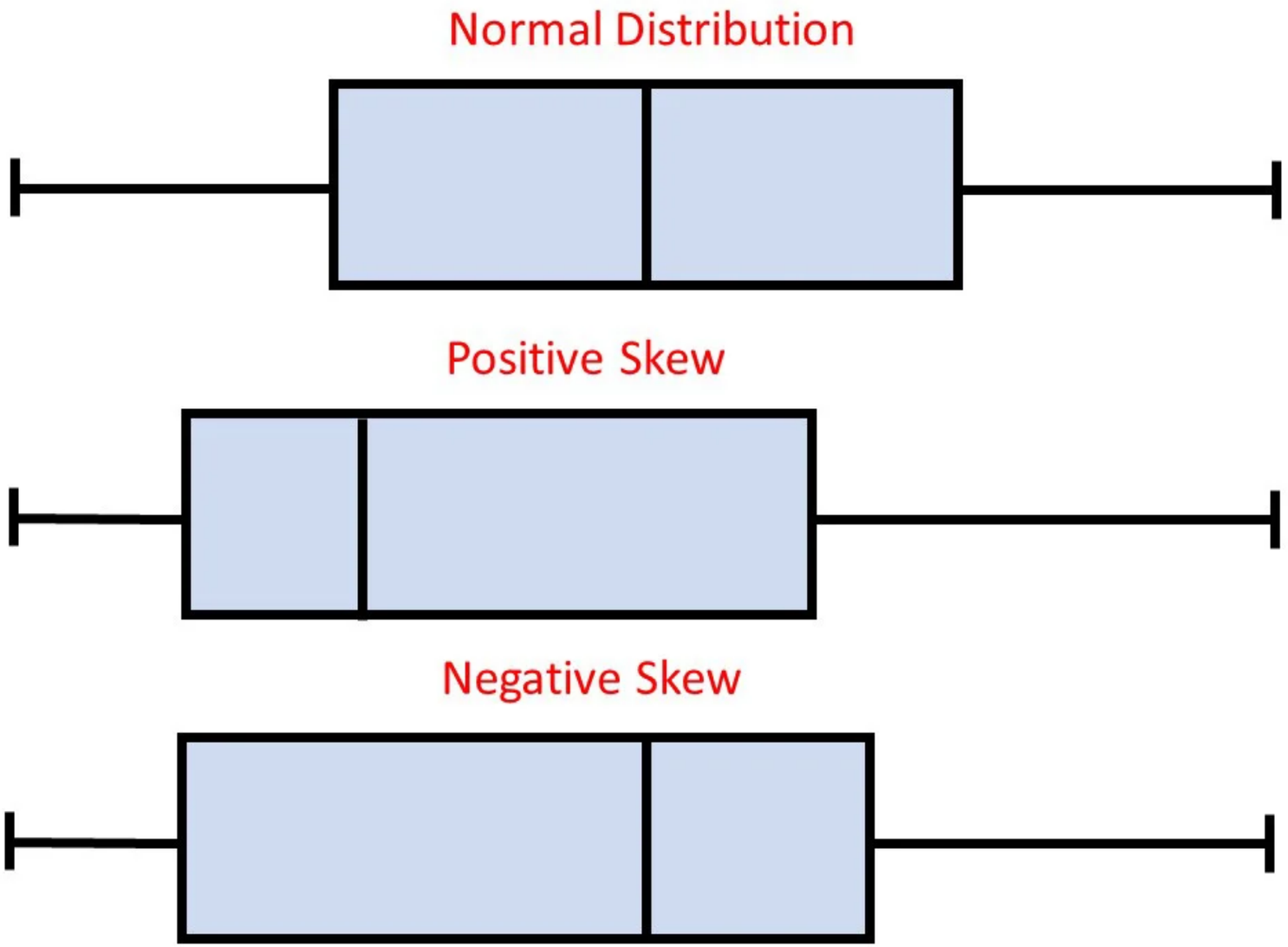
To group them, you can change the `x` to a “formula”

```
outcome_var ~ group_var
```

Plot | Code



Visualizations: Boxplot



No Assignment!