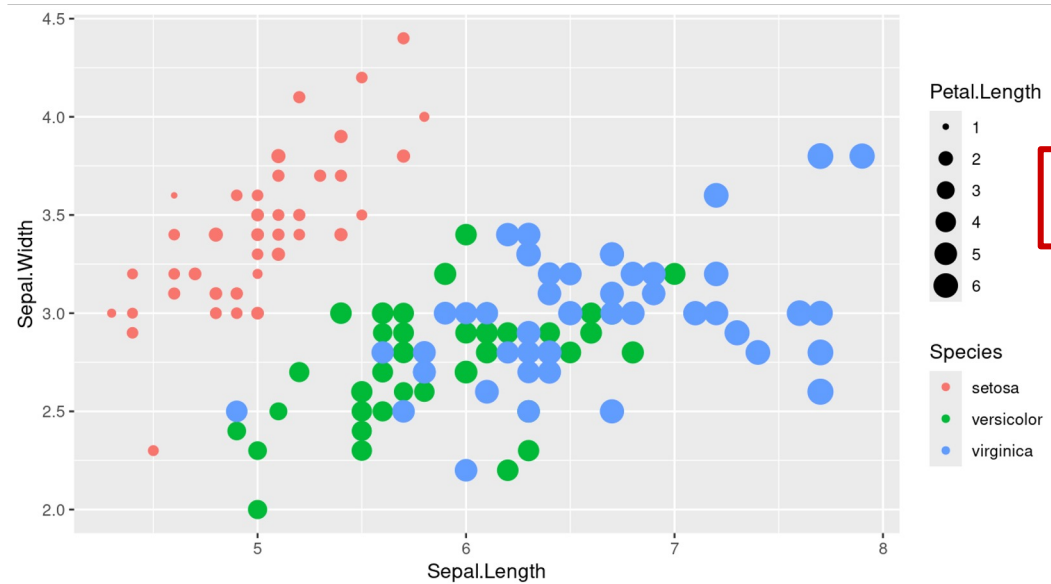


Introduction to plotting and statistics

Variables are individually represented in the dimensions of a plot



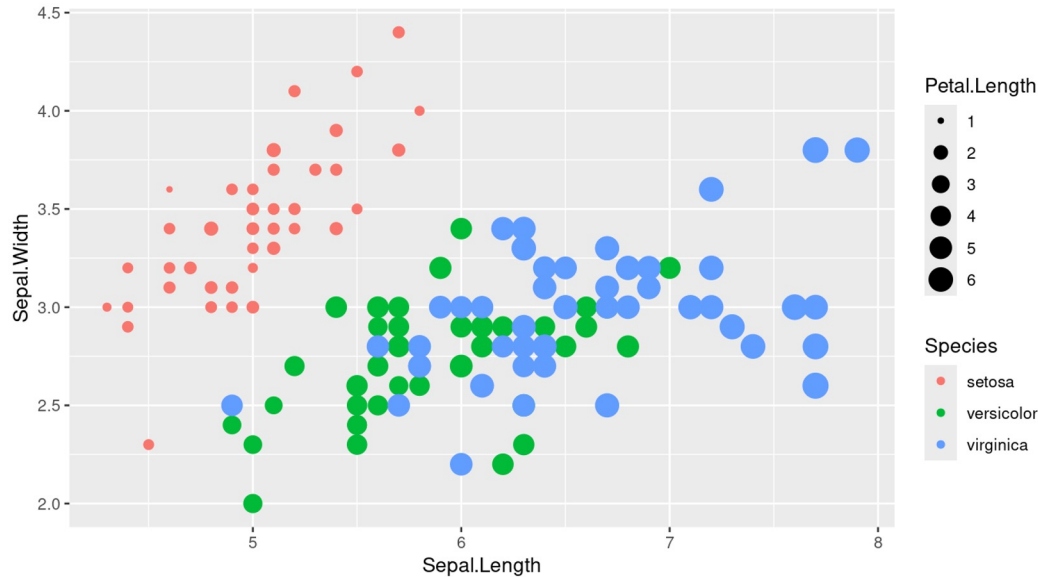
Y axis

Point size

Point color

X axis

Plot dimensions motivate data interpretation and analysis



“The pink points are smaller and are higher/to the left of the bigger blue/green points.”

“The **setosa** species has shorter petal lengths, and greater sepal width and sepal length, compared to the **versicolor** and **virginica** species.

To generate the plot, the data (oftentimes as a data frame) needs to include each variable as a column, with the data encoding each point within the row.

Data generally comes in “wide” format

Example of wide format:

Student	Math	Literature	PE
A	99	45	56
B	73	78	55
C	12	96	57

- Can be easier to understand
- More convenient for certain analyses or visualization
- No information is repeated in each line

Long data format is the “tidy” format

Example of long format:

Student	Subject	Score
A	Math	99
A	Literature	45
A	PE	56
B	Math	73
B	Literature	78
B	PE	55
C	Math	12

- Multiple rows correspond to a single observation and measurement
- One column for each variable
- Additional columns store metadata or grouping variables

Manipulating data frames: Long to wide data

```
library(tidyr)

# Example long format data
long_data <- data.frame(
  Subject = c("A", "A", "B", "B"),
  Time = c(1, 2, 1, 2),
  Measurement = c(10, 15, 12, 18)
)

# Convert long format data to wide format
wide_data <- spread(long_data, key = Time, value = Measurement)

# View the wide format data
print(wide_data)
```

```
> head(long_data)
```

	Subject	Time	Measurement
1	A	1	10
2	A	2	15
3	B	1	12
4	B	2	18

```
> wide_data
```

	Subject	1	2
1	A	10	15
2	B	12	18

Now `1` and `2` can be directly compared

Manipulating data frames: Wide to long data

```
library(tidyr)

# Example wide format data
wide_data <- data.frame(
  Subject = c("A", "B"),
  Time1 = c(10, 12),
  Time2 = c(15, 18)
)

# Convert wide format data to long format
long_data <- gather(wide_data, key = Time, value = Measurement, -Subject)

# View the long format data
print(long_data)
```

> wide_data

	Subject	Time1	Time2
1	A	10	15
2	B	12	18

>

>

> long_data

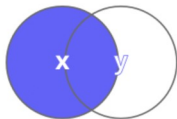
	Subject	Time	Measurement
1	A	1	10
2	A	2	15
3	B	1	12
4	B	2	18

Now `Time` and `Measurement`
can be directly compared

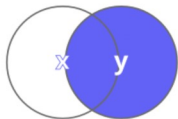
Merging data: What if you want to compare data from two different data frames?

dplyr joins

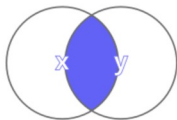
left_join(x, y)



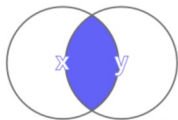
right_join(x, y)



inner_join(x, y)

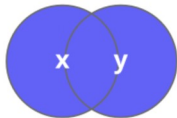


semi_join(x, y)

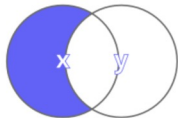


(never duplicate rows of x)

full_join(x, y)



anti_join(x, y)



```
library(dplyr)

# Example datasets
df1 <- data.frame(ID = c(1, 2, 3), Name = c("Alice", "Bob", "Charlie"))
df2 <- data.frame(ID = c(2, 3, 4), Score = c(85, 90, 95))

# Inner join
inner_merged <- inner_join(df1, df2, by = "ID")

# Left join
left_merged <- left_join(df1, df2, by = "ID")

# Right join
right_merged <- right_join(df1, df2, by = "ID")

# Full join
full_merged <- full_join(df1, df2, by = "ID")

# Semi-join
semi_merged <- semi_join(df1, df2, by = "ID")

# Anti-join
anti_merged <- anti_join(df1, df2, by = "ID")
```


* `_join()` functions combine the information across two data frames

```
> full_merged
```

	ID	Name	Score
1	1	Alice	NA
2	2	Bob	85
3	3	Charlie	90
4	4	<NA>	95

```
library(dplyr)

# Example datasets
df1 <- data.frame(ID = c(1, 2, 3), Name = c("Alice", "Bob", "Charlie"))
df2 <- data.frame(ID = c(2, 3, 4), Score = c(85, 90, 95))

# Inner join
inner_merged <- inner_join(df1, df2, by = "ID")

# Left join
left_merged <- left_join(df1, df2, by = "ID")

# Right join
right_merged <- right_join(df1, df2, by = "ID")

# Full join
full_merged <- full_join(df1, df2, by = "ID")

# Semi-join
semi_merged <- semi_join(df1, df2, by = "ID")

# Anti-join
anti_merged <- anti_join(df1, df2, by = "ID")
```

Using the ggplot2 package: Structuring a plot command

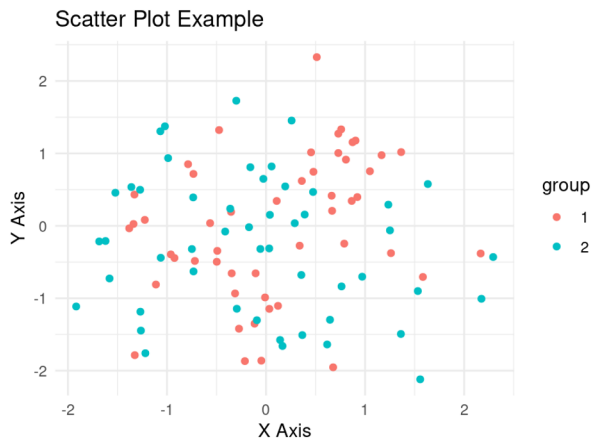
- Data
- Aesthetics (aes)
- Geometries (geom)
- Scales
- Coordinate systems
- Facets
- Themes

```
library(ggplot2)

# Sample data
df <- data.frame(
  x = rnorm(100),
  y = rnorm(100),
  group = factor(rep(1:2, each = 50))
)

# Creating a scatter plot
ggplot(df, aes(x = x, y = y, color = group)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Scatter Plot Example", x = "X Axis", y = "Y Axis")
```

Using the ggplot2 package: Structuring a plot command



```
library(ggplot2)

# Sample data
df <- data.frame(
  x = rnorm(100),
  y = rnorm(100),
  group = factor(rep(1:2, each = 50))
)

# Creating a scatter plot
ggplot(df, aes(x = x, y = y, color = group)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Scatter Plot Example", x = "X Axis", y = "Y Axis")
```

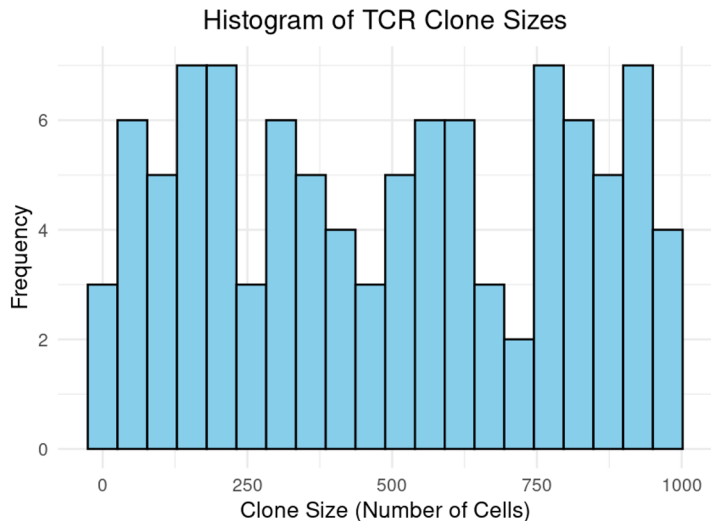
Using the ggplot2 package: Structuring a plot command

```
library(ggplot2)
library(dplyr)

# Step 1: Simulate data
set.seed(123) # Set seed for reproducibility
num_clonotypes <- 100 # Specify the number of different clonotypes

# Create a data frame with random cell counts for each clonotype
tcr_data <- tibble(
  clonotype = paste("TCR", seq_len(num_clonotypes), sep=""),
  cell_count = sample(1:1000, num_clonotypes, replace=TRUE) # Random cell
)

# Step 2: Create a histogram using ggplot2
ggplot(tcr_data, aes(x = cell_count)) +
  geom_histogram(bins = 20, fill = "skyblue", color = "black") +
  theme_minimal() +
  labs(
    title = "Histogram of TCR Clone Sizes",
    x = "Clone Size (Number of Cells)",
    y = "Frequency"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5) # Center the plot title
  )
```



Performing statistics

```
# Randomly generated sample data: Immune marker levels in two patient groups
group1 <- rnorm(30, mean = 5, sd = 1.5) # Patients with a mutation
group2 <- rnorm(30, mean = 4.5, sd = 1.2) # Patients without the mutation

# Perform a t-test
test <- t.test(group1, group2)

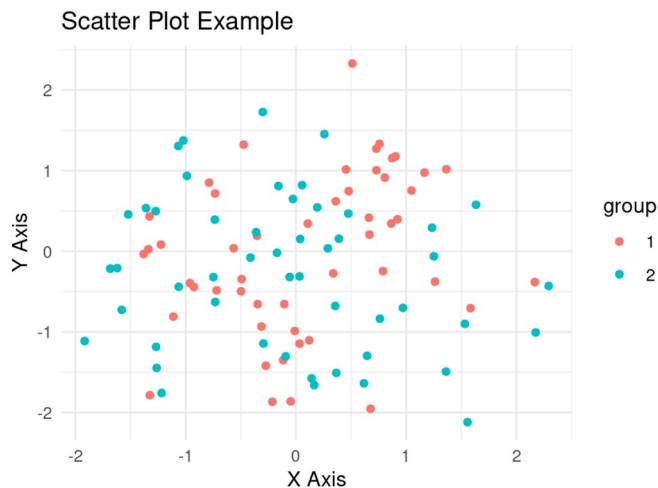
# Print the result
print(test)
```

```
> print(test)
```

Welch Two Sample t-test

```
data: group1 and group2
t = 1.2838, df = 57.999, p-value = 0.2043
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2731337  1.2498937
sample estimates:
mean of x mean of y
 5.23333  4.74495
```

Adding statistics to your plot using ggpubr()

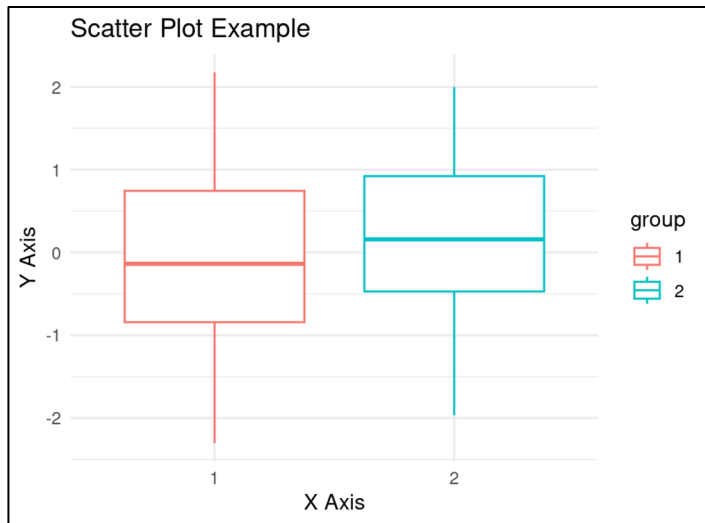


```
library(ggplot2)

# Sample data
df <- data.frame(
  x = rnorm(100),
  y = rnorm(100),
  group = factor(rep(1:2, each = 50))
)

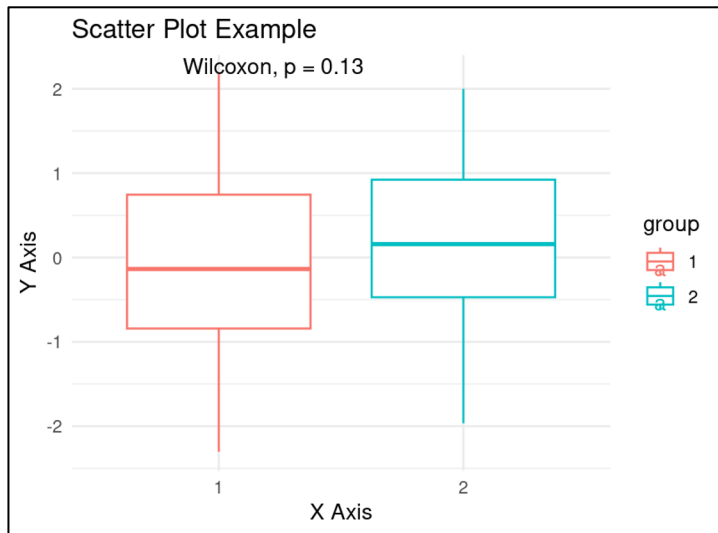
# Creating a scatter plot
ggplot(df, aes(x = x, y = y, color = group)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Scatter Plot Example", x = "X Axis", y = "Y Axis")
```

What if we made it a boxplot?



```
# Creating a box plot
ggplot(df, aes(x = group, y = y, color = group)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Scatter Plot Example", x = "X Axis", y = "Y Axis")
```

Adding a Wilcoxon test summary to the plot



```
library(ggpubr)
# Creating a box plot
ggplot(df, aes(x = group, y = y, color = group)) +
  geom_boxplot() +
  stat_compare_means(method = 'wilcox') +
  theme_minimal() +
  labs(title = "Scatter Plot Example", x = "X Axis", y = "Y Axis")
```


Hands-on: Reading, writing, and interpreting data structures

- These slides are available on the course website under **Resources: During the course: Lecture slides**
 - Start a new Rscript in Posit
- Work through the blocks of code under the **Course: Basic plotting and statistics** page
 - Review the details on how the code works in the Lecture slides for assistance
 - Put a post-it on your laptop if you get stuck, indicating for a TA to come up to you
 - Work through the blocks of code on this page, practicing in both your Rscript and the terminal
- Taking the next step
 - There are a list of **Additional exercises** at the bottom of the page for you to try on your own

Goal: Start thinking about your own datasets.