

DATA SOCIETY®

Advanced visualization with R - Part 1

*One should look for what is and not what he thinks should be.
-Albert Einstein.*

Welcome



Who we are

- Data Society's mission is to **integrate Big Data and machine learning best practices across entire teams** and empower professionals to identify new insights
- We provide:
 - High-quality data science training programs
 - Customized executive workshops
 - Custom software solutions and consulting services
- Since 2014, we've worked with thousands of professionals to make their data work for them



Best practices for virtual classes

1. Find a quiet place, free of as many distractions as possible. Headphones are recommended.
2. Use Q and A tab to ask questions
3. Remove or silence alerts from cell phones, e-mail pop-ups, etc.
4. Participate in activities and ask questions. This will remain interactive!
5. Give your honest feedback so we can troubleshoot problems and improve the course.



Module completion checklist

Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	
Describe and build univariate plots to illustrate patterns in data	
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	
Create a multivariate plot (scatterplot matrix) and evaluate best uses	
Formulate the process of using ggplot2 to build plots	
Build a histogram with ggplot2	

Directory settings

- In order to maximize the efficiency of your workflow, you may want to encode your directory structure into variables
- Let the `main_dir` be the variable corresponding to your `skillsoft` folder

```
# Set `main_dir` to the location of your `skillsoft` folder (for Mac/Linux).
main_dir = "~/Desktop/skillsoft"
# Set `main_dir` to the location of your `skillsoft` folder (for Windows).
main_dir = "C:/Users/[username]/Desktop/skillsoft"

# Make `data_dir` from the `main_dir` and remainder of the path to data directory.
data_dir = paste0(main_dir, "/data")
# Make `plots_dir` from the `main_dir` and remainder of the path to plots directory.
plot_dir = paste0(main_dir, "/plots")

# Set directory to data_dir.
setwd(data_dir)
```

Class overview

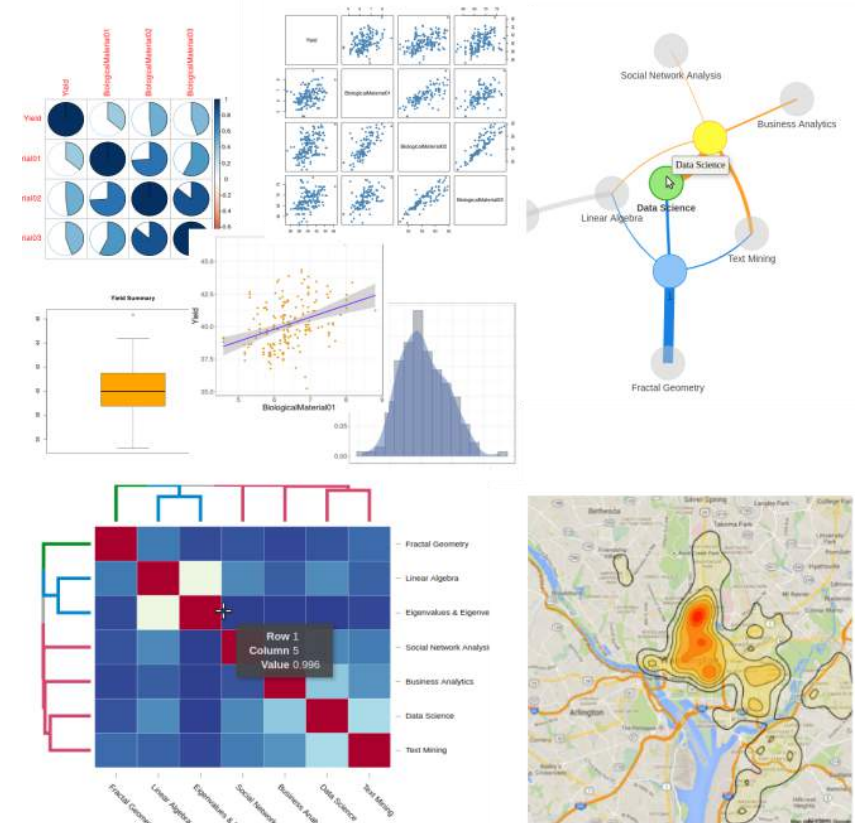
- In this class, we'll learn how to visualize data, create bi-variate and multi-variate plots using ggplot2 package
- But before delving into them, let's do a quick refresher on few visualization topics
 - basics of visualizing data in R,
 - exploratory data analysis and
 - creating univariate plots

Why visualize?

Why do we need to build visualizations?

- To provide valuable insights that are interpretable and relevant
- To give a visual or graphical representation of data / concepts
- To communicate ideas
- To provide an accessible way to see and understand trends, outliers, and patterns in data
- To confirm a hypothesis about the data

The images on the right show some common types of charts and graphs used in data science

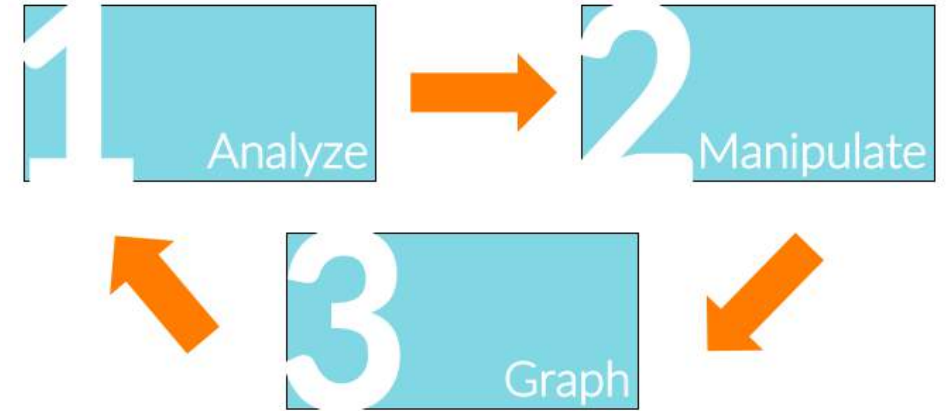


Polling questions

1. What visualization tool(s) do you use at your job?
2. What types of data visualizations do you produce or review?

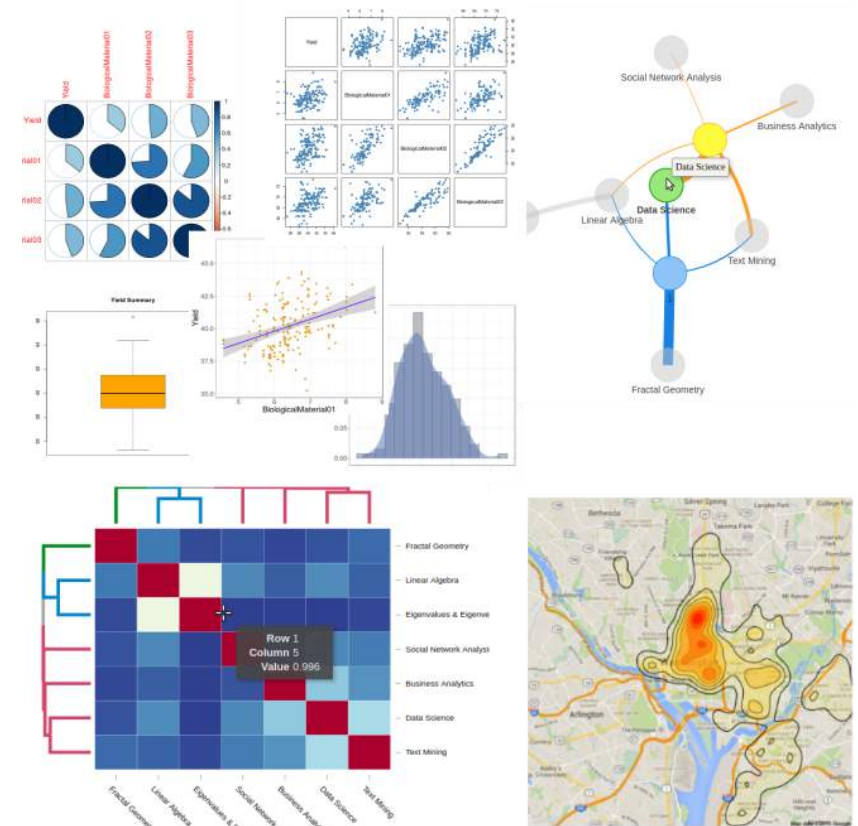
Exploratory data analysis (EDA)

- R is a powerful tool for EDA because the graphics tie in with the functions used to analyze data. You can create graphs **without** breaking your train of thought as you explore your data
- As the image on the right indicates, visualization is an **iterative process** that consists of the following steps:
 - Analyze data
 - Manipulate data
 - Graph the results
 - Repeat



Visualizing data in R

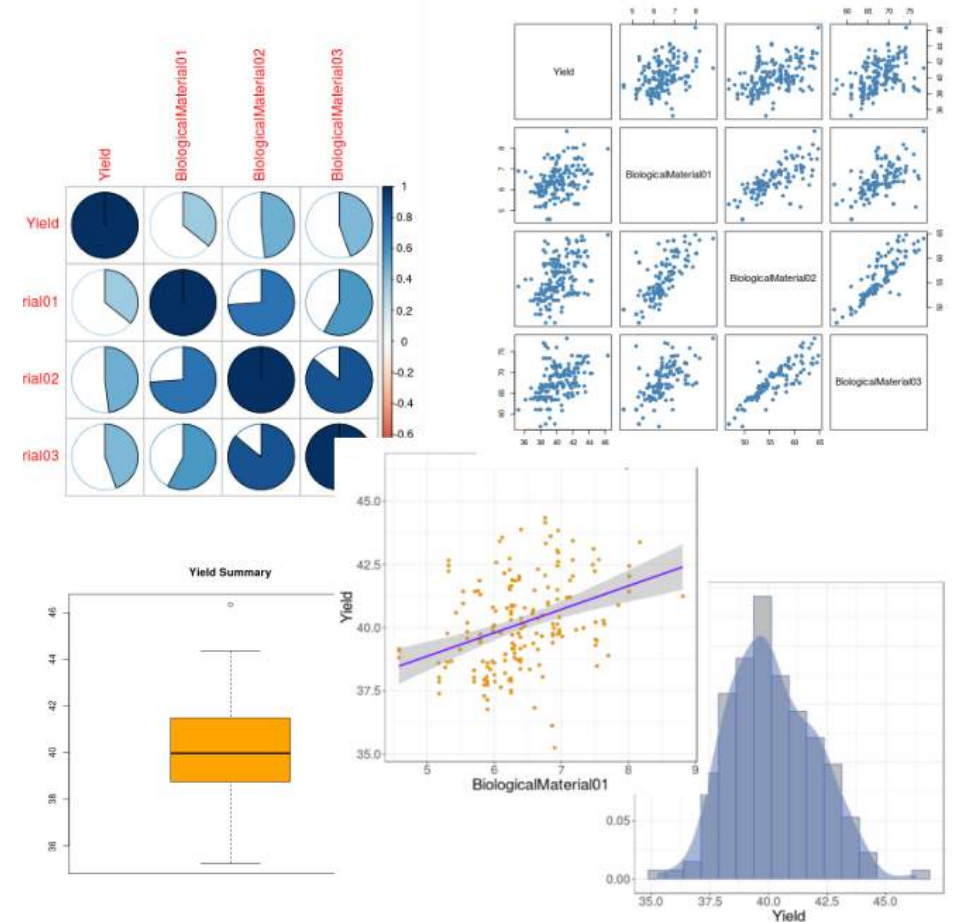
- R comes with multiple types of visualizations:
 - Basic plots & composite graphs
 - Maps
 - Dynamic visualizations
 - Interactive charts & dashboards
 - 3D graphics
- You can also save and share your work in a variety of file formats, and with adjustable image quality (SVG, PNG, JPEG, PDF, etc.)
- Use [this link](#) to see a list of help pages, vignettes, and code demos.



Static visualizations in R

- **Static plots**

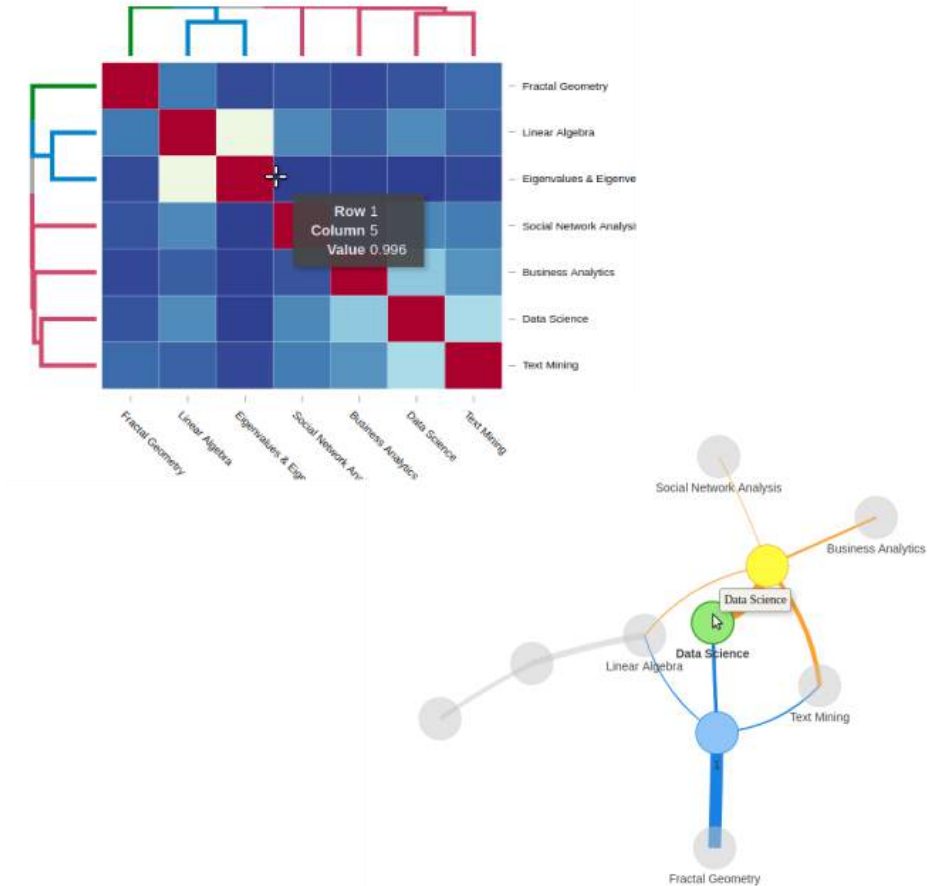
- Only **display** data, **without letting the user interact** with the visualization
- Are available through base R and multitudes of packages (e.g., `ggplot2`, `corrplot`)
- Are high quality (R visualizations are made in scalable vector graphics format, or SVG) and can be saved as SVG, PNG, JPEG, BMP, or PDF
- Are the best way to display patterns in data for **printed publications**




Interactive visualizations in R

- **Interactive plots**

- **Display** data and also **let the user interact** with the visualization by clicking, dragging, zooming, etc.
- Are available through many packages (e.g., `highcharter`, `plotly`, `htmlwidgets`)
- Are high quality (graphical elements are in SVG format), render as HTML pages, and can be saved as HTML documents and opened through browser
- Are the best way to display patterns in data for **web publications**, on **websites**, in **web applications**



Module completion checklist

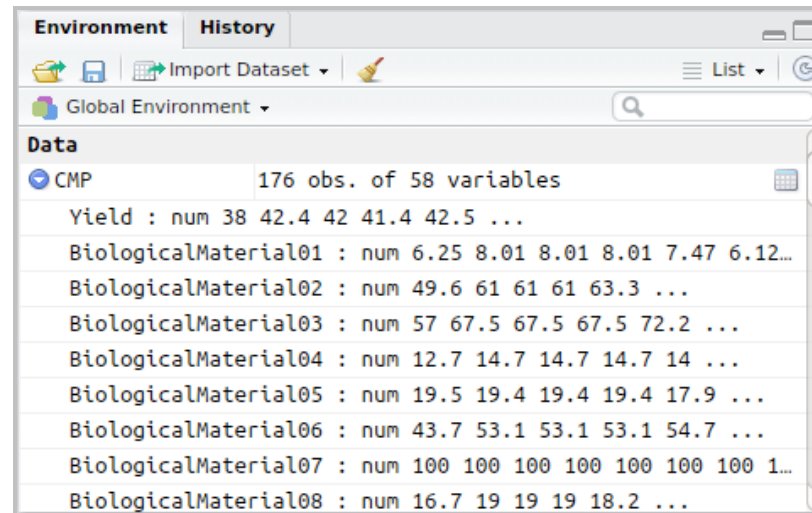
Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	
Describe and build univariate plots to illustrate patterns in data	
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	
Create a multivariate plot (scatterplot matrix) and evaluate best uses	
Formulate the process of using ggplot2 to build plots	
Build a histogram with ggplot2	

Loading the CMP dataset for EDA

- We will be using the **Chemical Manufacturing Process** (CMP) dataset to create various static visualizations
- Let's load the dataset from our `data_dir` into R's environment

```
# Set working directory to where we store data.
setwd(data_dir)

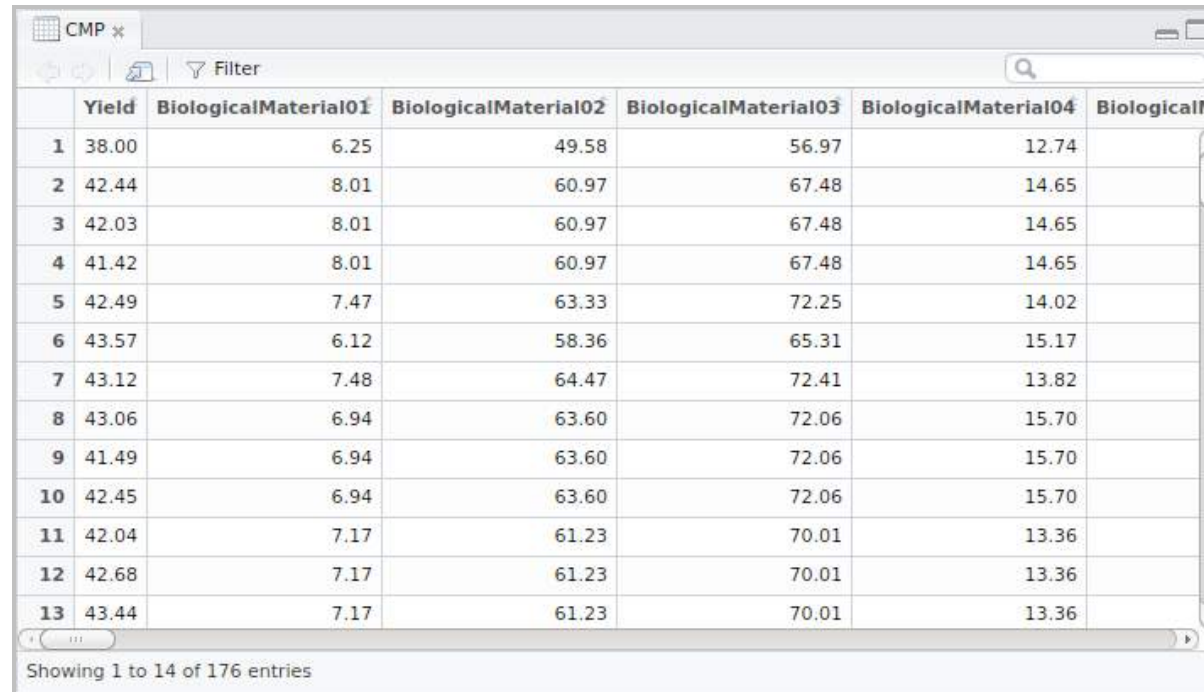
# Read CSV file called "ChemicalManufacturingProcess.csv"
CMP = read.csv("ChemicalManufacturingProcess.csv",
               header = TRUE,
               stringsAsFactors = FALSE)
```



About the CMP dataset

- The dataset consists of 176 observations and 58 variables

```
# View CMP dataset in the tabular data explorer.  
View(CMP)
```

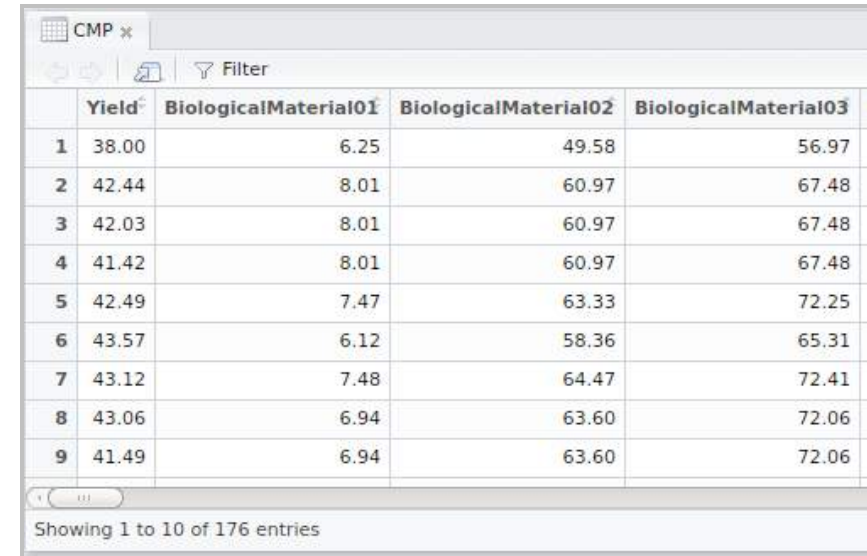


	Yield	BiologicalMaterial01	BiologicalMaterial02	BiologicalMaterial03	BiologicalMaterial04	BiologicalMaterial05
1	38.00	6.25	49.58	56.97	12.74	
2	42.44	8.01	60.97	67.48	14.65	
3	42.03	8.01	60.97	67.48	14.65	
4	41.42	8.01	60.97	67.48	14.65	
5	42.49	7.47	63.33	72.25	14.02	
6	43.57	6.12	58.36	65.31	15.17	
7	43.12	7.48	64.47	72.41	13.82	
8	43.06	6.94	63.60	72.06	15.70	
9	41.49	6.94	63.60	72.06	15.70	
10	42.45	6.94	63.60	72.06	15.70	
11	42.04	7.17	61.23	70.01	13.36	
12	42.68	7.17	61.23	70.01	13.36	
13	43.44	7.17	61.23	70.01	13.36	

Showing 1 to 14 of 176 entries

Subsetting data

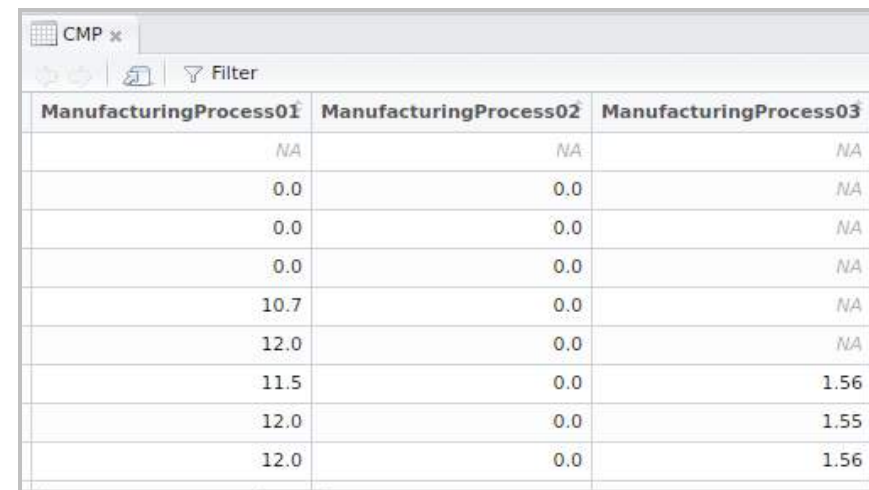
- In this module, we will explore only a subset of this dataset, which includes the following variables:
 - yield
 - 3 material variables
 - 3 process variables



	Yield	BiologicalMaterial01	BiologicalMaterial02	BiologicalMaterial03
1	38.00	6.25	49.58	56.97
2	42.44	8.01	60.97	67.48
3	42.03	8.01	60.97	67.48
4	41.42	8.01	60.97	67.48
5	42.49	7.47	63.33	72.25
6	43.57	6.12	58.36	65.31
7	43.12	7.48	64.47	72.41
8	43.06	6.94	63.60	72.06
9	41.49	6.94	63.60	72.06

Showing 1 to 10 of 176 entries

...



ManufacturingProcess01	ManufacturingProcess02	ManufacturingProcess03
NA	NA	NA
0.0	0.0	NA
0.0	0.0	NA
0.0	0.0	NA
0.0	0.0	NA
10.7	0.0	NA
12.0	0.0	NA
11.5	0.0	1.56
12.0	0.0	1.55
12.0	0.0	1.56

Subsetting data (cont'd)

```
# Let's make a vector of column indices we would like to save.
column_ids = c(1:4, #<- concatenate a range of ids
               14:16)#<- with another a range of ids
column_ids
```

```
[1]  1  2  3  4 14 15 16
```

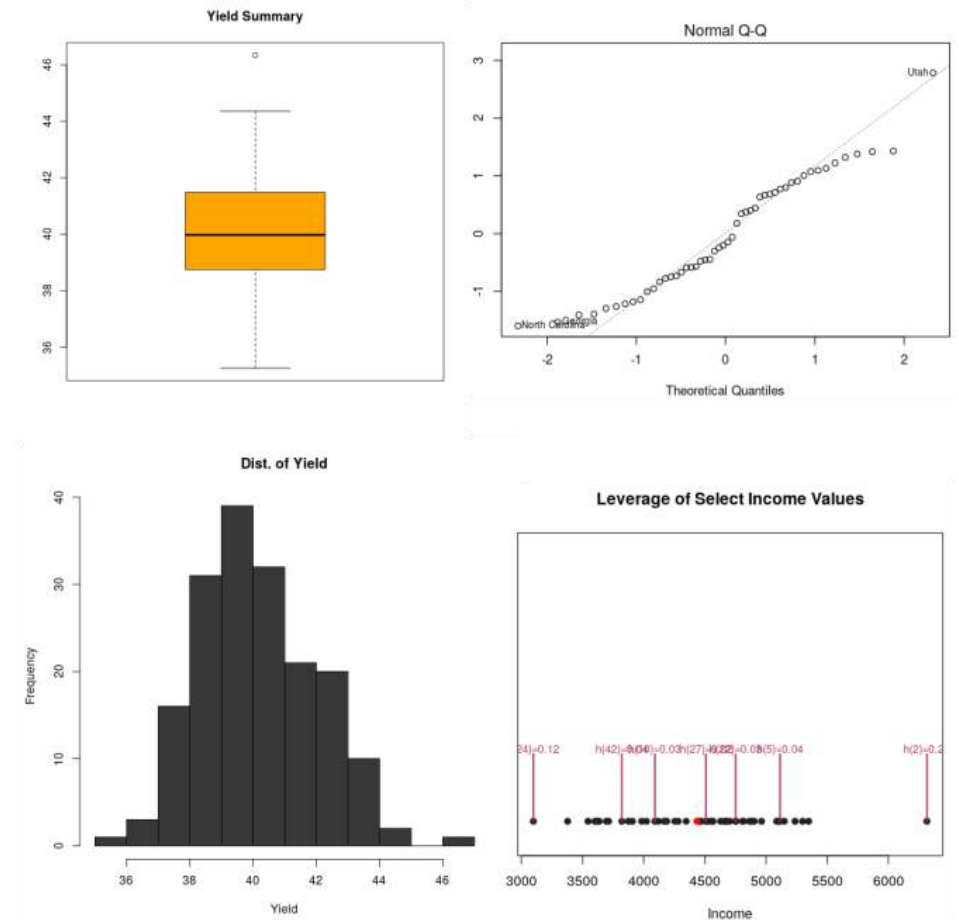
```
# Let's save the subset into a new variable.
CMP_subset = CMP[ , column_ids]
str(CMP_subset)
```

```
'data.frame':  176 obs. of  7 variables:
 $ Yield      : num  38 42.4 42 41.4 42.5 ...
 $ BiologicalMaterial01 : num  6.25 8.01 8.01 8.01 7.47 6.12 7.48 6.94 6.94 6.94 ...
 $ BiologicalMaterial02 : num  49.6 61 61 61 63.3 ...
 $ BiologicalMaterial03 : num  57 67.5 67.5 67.5 72.2 ...
 $ ManufacturingProcess01: num  NA 0 0 0 10.7 12 11.5 12 12 12 ...
 $ ManufacturingProcess02: num  NA 0 0 0 0 0 0 0 0 0 ...
 $ ManufacturingProcess03: num  NA NA NA NA NA NA 1.56 1.55 1.56 1.55 ...
```

Univariate plots

- Univariate plots are used to visualize distribution of a **single variable**
- They are used primarily in the initial stages of EDA when we would like to learn more about individual variables in our data
- They are also used in combination with other univariate plots to compare data distributions of different variables
- Univariate plots include the following popular graphs: boxplot, histogram, density curve, dot plot, QQ plot, and bar plot

Different univariate plots



Univariate plots: boxplots

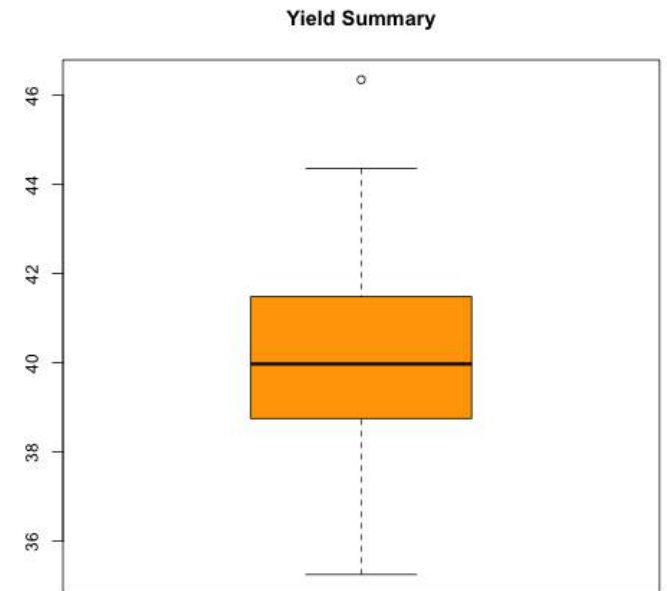
- A box-and-whisker plot (a.k.a. boxplot) is a graph that depicts the distribution of a continuous variable through its five-point summary:

```
summary(CMP_subset$Yield)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
35.25	38.75	39.97	40.18	41.48	46.34

- **Min value** (35.25): bottom whisker of the boxplot
- **Max value** (46.34): circle (it's an outlier!)
- **1st quartile value** (38.75): bottom of the box
- **Median (2nd quartile) value** (39.97): line in the box
- **3rd quartile value** (41.48): top of the box

```
boxplot(CMP_subset$Yield,  
        col = "orange",  
        main = "Yield Summary")  
#<- add title
```

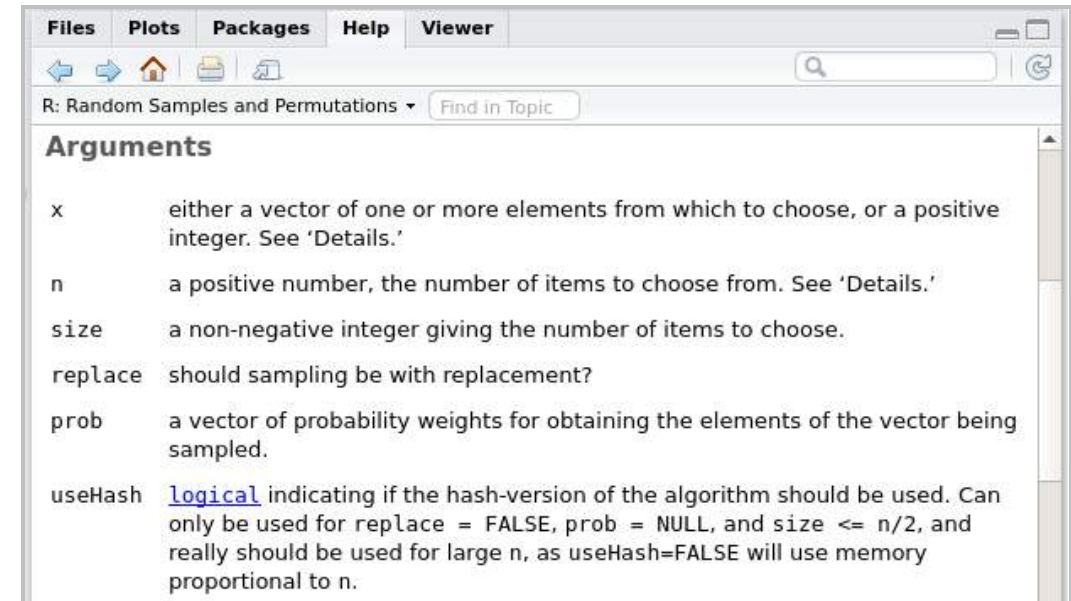


Sample colors for visualization

- The `sample` function in R, as the name suggests, lets us take a sample of a specific number of elements from a vector, with or without replacement

?sample

- It has the following arguments:
 - `x` is a vector from which we take elements to form our sample
 - `n` is a number to indicate the number of elements constituting our sample



Sample colors for visualization (cont'd)

- Let's sample the colors to pick a different color for every variable in our dataset

```
# Set random seed to get the same sample every time!
set.seed(1)

# We have as many variables as columns in our data.
# Save number of columns to a variable using `ncol` function.
n_cols = ncol(CMP_subset)

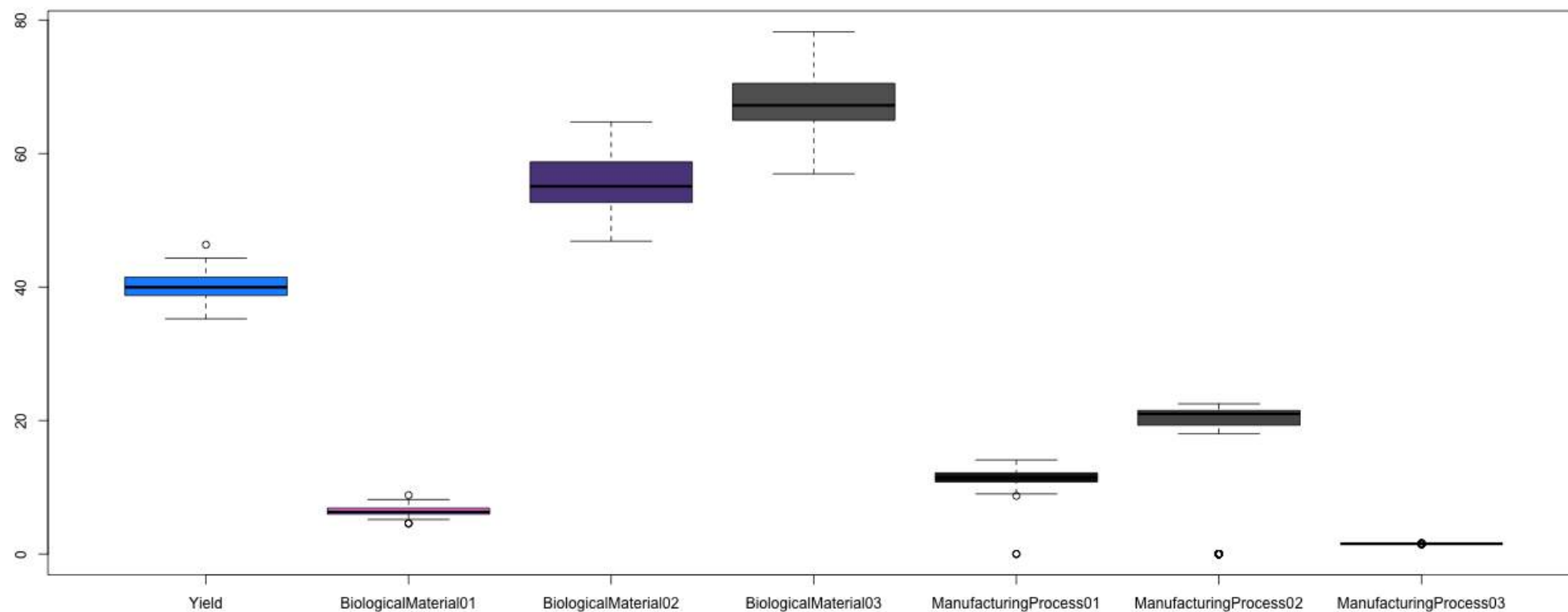
# Pick a sample of colors for each
# variable in our data set
col_sample = sample(colors(), #<- vector of colors
                    n_cols)  #<- n elements to sample
col_sample
```

```
[1] "dodgerblue1"  "orchid1"      "mediumpurple4" "grey38"
[5] "grey9"        "gray34"       "grey46"
```

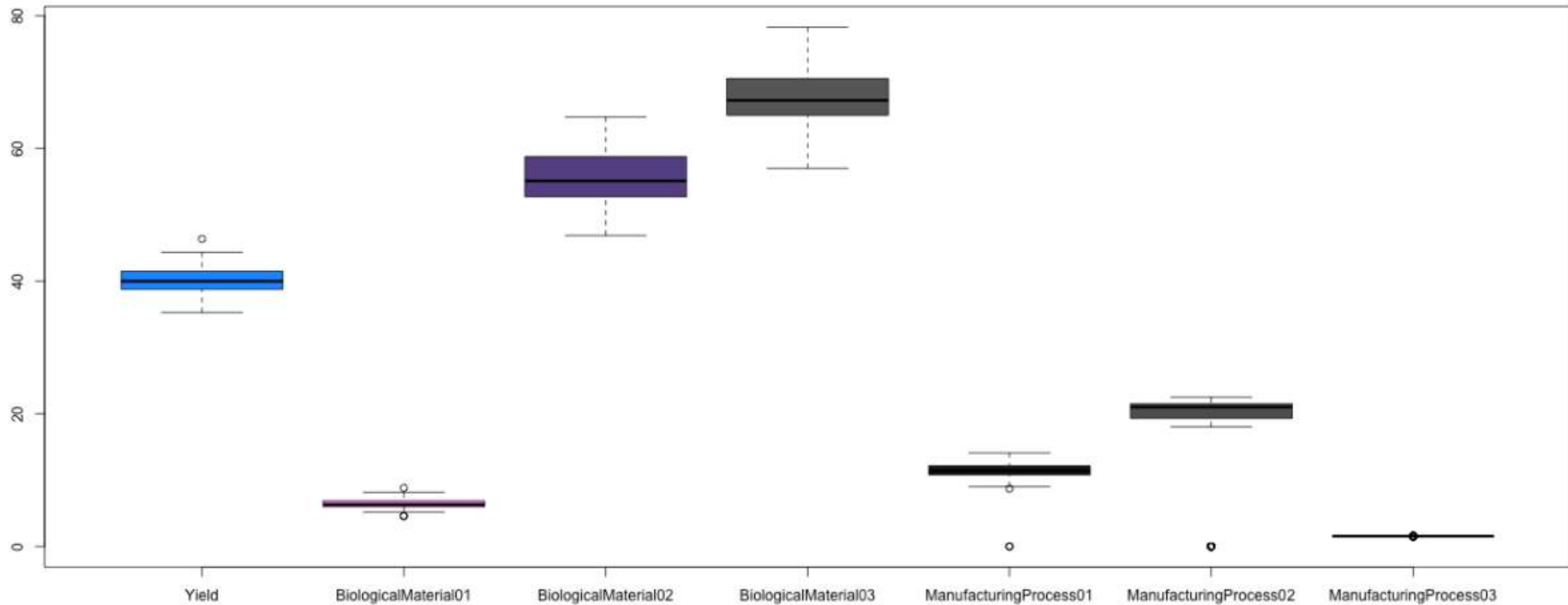

Compare variables: boxplots combined

- We can specify color in a plotting function by passing the color to the argument `col`:

```
# Make a box plot of all variables and give a vector of colors for each of them.  
boxplot(CMP_subset, col = col_sample)
```



Polling question



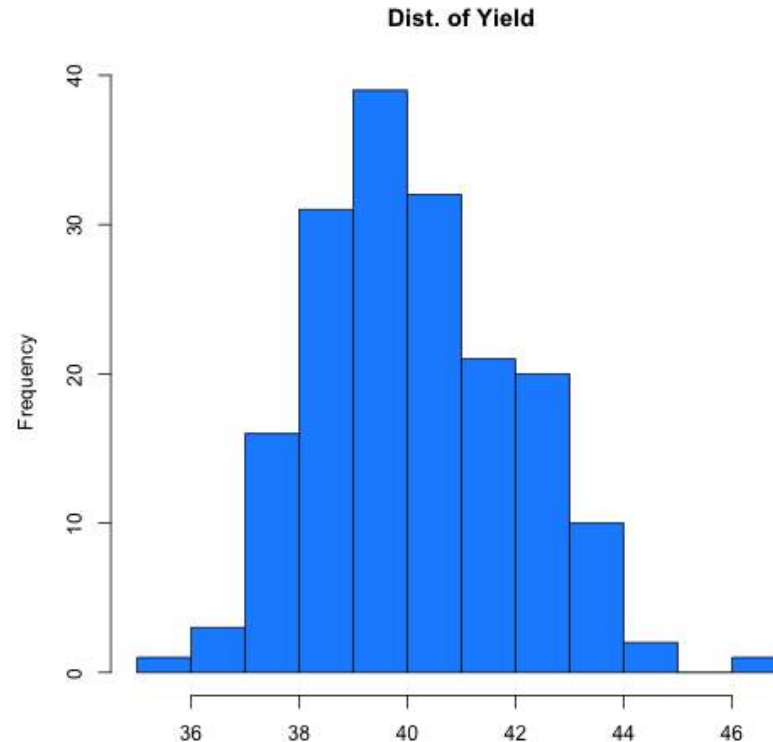
Which variable has the broadest distribution?

Univariate plots: histogram

- A histogram represents the **distribution of numerical data**
 - The height of each bar has been calculated as the number of observations in that range
 - `hist()` produces a basic histogram of any *numeric* variable
- Let's plot a histogram of `Yield`

Graphical summary

```
# Univariate plot: histogram.  
hist(CMP_subset$Yield,  
      col = col_sample[1],  
      xlab = "Yield",      #<- set x-axis label  
      main = "Dist. of Yield") #<- set title
```



Univariate plots: histogram (cont'd)

We can also look at just the numerical summary of histogram by setting `plot = FALSE`

```
# Histogram data without plot.  
hist(CMP_subset$Yield, plot = FALSE)
```

Numerical summary

```
$breaks  
[1] 35 36 37 38 39 40 41 42 43 44 45 46 47  
  
$counts  
[1] 1 3 16 31 39 32 21 20 10 2 0 1  
  
$density  
[1] 0.005681818 0.017045455 0.090909091  
0.176136364 0.221590909 0.181818182  
[7] 0.119318182 0.113636364 0.056818182  
0.011363636 0.000000000 0.005681818  
  
$mids  
[1] 35.5 36.5 37.5 38.5 39.5 40.5 41.5 42.5  
43.5 44.5 45.5 46.5  
  
$xname  
[1] "CMP_subset$Yield"  
  
$equidist  
[1] TRUE  
  
attr(,"class")  
[1] "histogram"
```

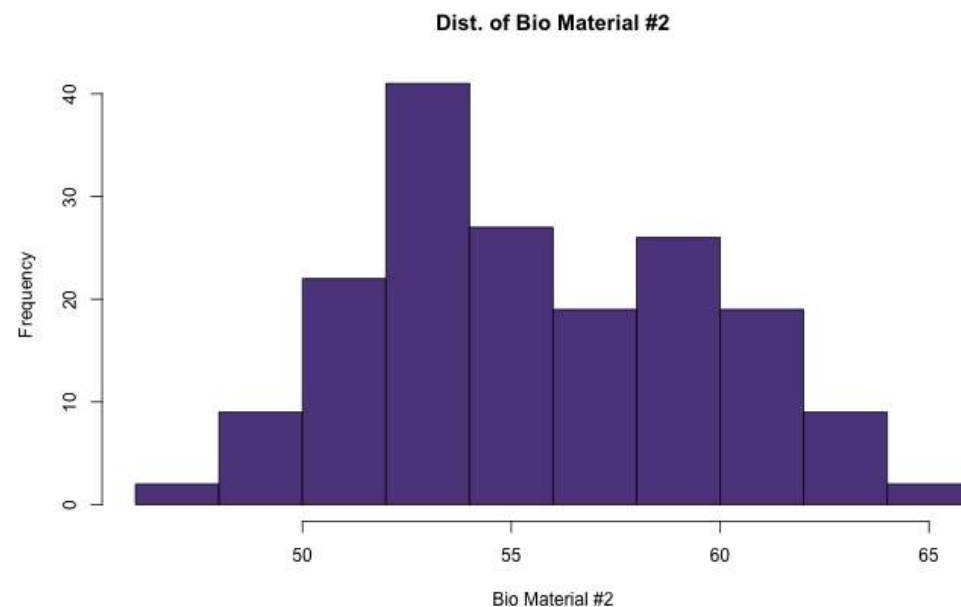
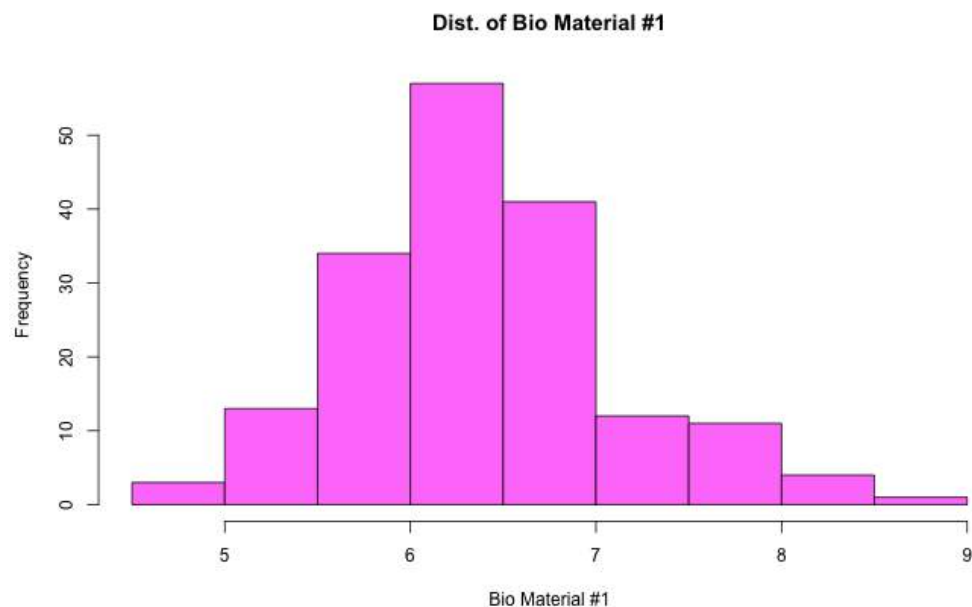
Compare variables: histograms combined

We can plot two or more charts on the same page by laying them in **subplots**

- The `par()` function lets us set graphical parameters
- We can use the `mfrow` argument of `par()` to create subplots by dividing the plotting area into a grid with specified number of rows and columns

```
# Make a combined histogram plot.
par(                                #<- set plot area parameters with `par`
  mfrow = c(1, 2))                 #<- split area into 1 row 2 columns with `mfrow`
hist(CMP_subset$BiologicalMaterial01, #<- add 1st histogram
     col = col_sample[2],
     xlab = "Bio Material #1",
     main = "Dist. of Bio Material #1")
hist(CMP_subset$BiologicalMaterial02, #<- add 2nd histogram
     col = col_sample[3],
     xlab = "Bio Material #2",
     main = "Dist. of Bio Material #2")
```

Compare variables: histograms combined (cont'd)



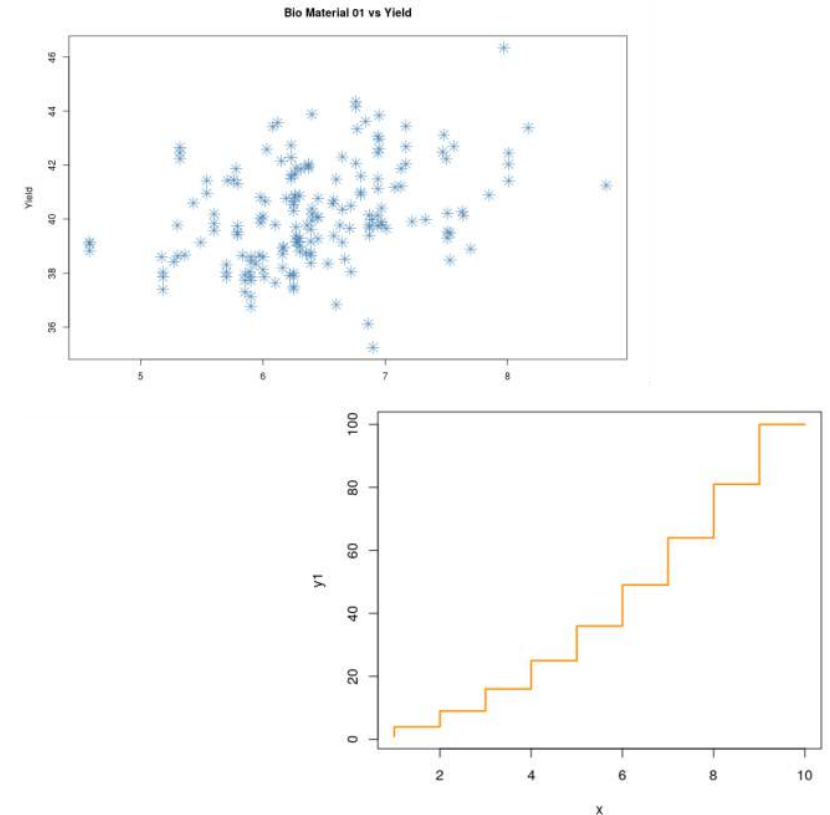
- Notice the range of values for each of these variables!

Module completion checklist

Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	✓
Describe and build univariate plots to illustrate patterns in data	✓
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	
Create a multivariate plot (scatterplot matrix) and evaluate best uses	
Formulate the process of using ggplot2 to build plots	
Build a histogram with ggplot2	

Bivariate plots

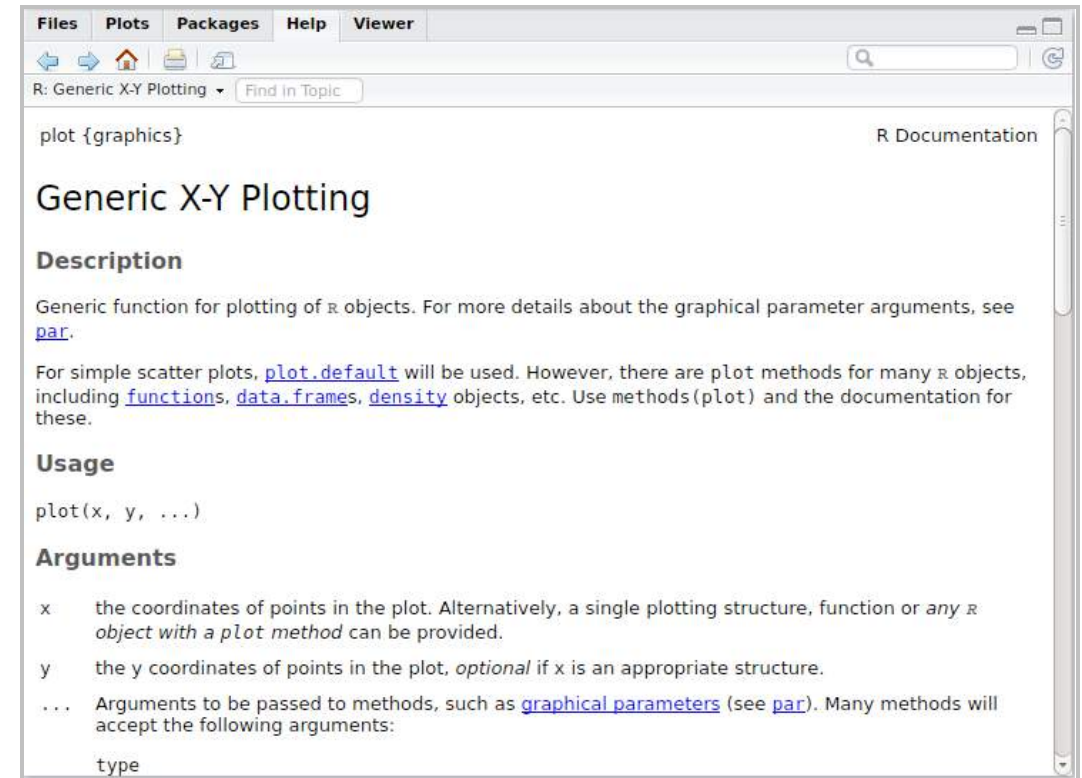
- Bivariate plots are used to visualize data distribution and relationships between **two variables**
- They are used heavily throughout different stages of EDA **to learn more about how one variable is related to another**
- They are also used **in combination with other bivariate plots to compare relationships between different pairs of variables**
- Bivariate plots include scatterplots and line graphs



Bivariate plots: R `plot` function

?plot

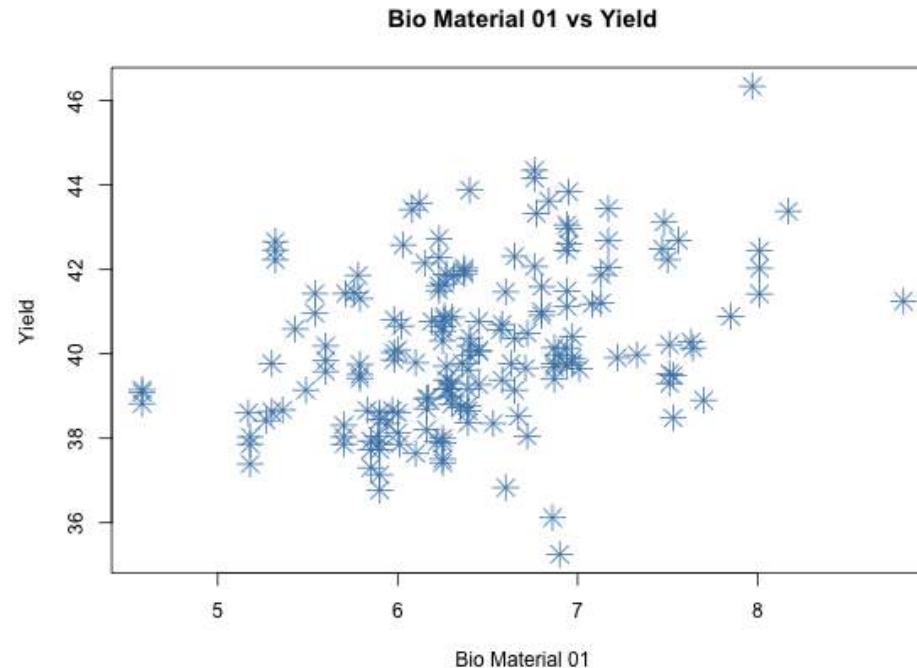
- The `plot` function is the most popular, robust, and useful visualization for numeric data in R
- It takes 2 main arguments:
 - Variable for `x-axis`, and
 - Variable for `y-axis`
- Produces a basic **scatterplot**
- Can also produce a **lineplot** with certain combination of parameters



Bivariate plots: scatterplot

Let's create a scatterplot between the first (Yield) and second (BioMaterial01) variables in our dataset

```
plot(CMP_subset[, 2], CMP_subset[, 1],  
     xlab = "Bio Material 01", ylab = "Yield", #<- add axis labels  
     main = "Bio Material 01 vs Yield",       #<- add title  
     pch = 8,                                #<- type of symbol to use  
     cex = 2,                                #<- scale of the point  
     col = "steelblue")
```

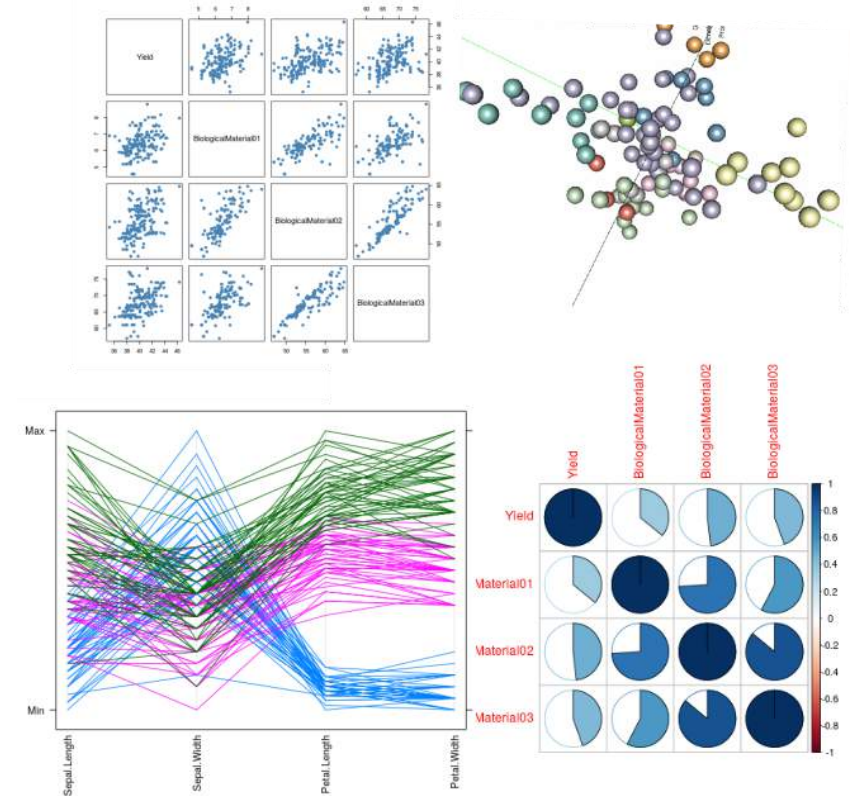


Module completion checklist

Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	✓
Describe and build univariate plots to illustrate patterns in data	✓
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	✓
Create a multivariate plot (scatterplot matrix) and evaluate best uses	
Formulate the process of using ggplot2 to build plots	
Build a histogram with ggplot2	

Multivariate plots

- Multivariate plots are used to visualize data distribution and relationships between **3 or more variables** (3D plots are essentially a special case of multivariate plots)
- They are used extensively during different stages of EDA and in combination with machine learning methods **to learn more about how variables are related to each other**
- Multivariate plots include the following popular graphs: scatterplot matrix, correlation plot, and parallel coordinates plot

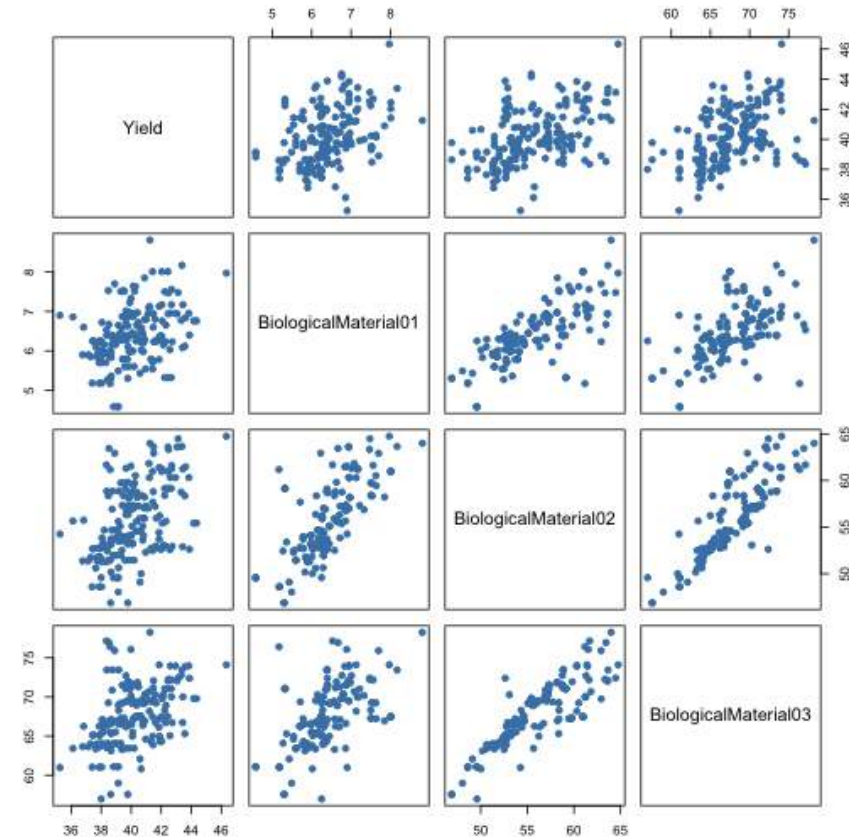


Multivariate plots: scatterplot matrix

- If we would like to quickly have a glance at relationships between multiple variables, **scatterplot matrix** is our friend
- Using the `pairs` function, we can pass many variables to the plotting function and voila!

Scatterplot matrix is symmetric around the diagonal—each variable occupies the same row and column. To see the scatterplot for variable 1 and 2, pick the tile in row 1 and column 2!

```
# Plot scatterplots for many variables.  
pairs(CMP_subset[, 1:4], #<- select variables  
      pch = 19,          #<- set symbol  
      col = "steelblue") #<- set color
```



Knowledge check 1



Exercise 1



Module completion checklist

Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	✓
Describe and build univariate plots to illustrate patterns in data	✓
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	✓
Create a multivariate plot (scatterplot matrix) and evaluate best uses	✓
Formulate the process of using ggplot2 to build plots	
Build a histogram with ggplot2	

Visualizing data with `ggplot2`

R's `ggplot2` package lets us:

- **Explore** data efficiently
- **Communicate** a visual story flexibly and efficiently
- **Layer** raw, summarized and contextual data
 - Demonstrate relationships
- **Reproduce** and extend work easily



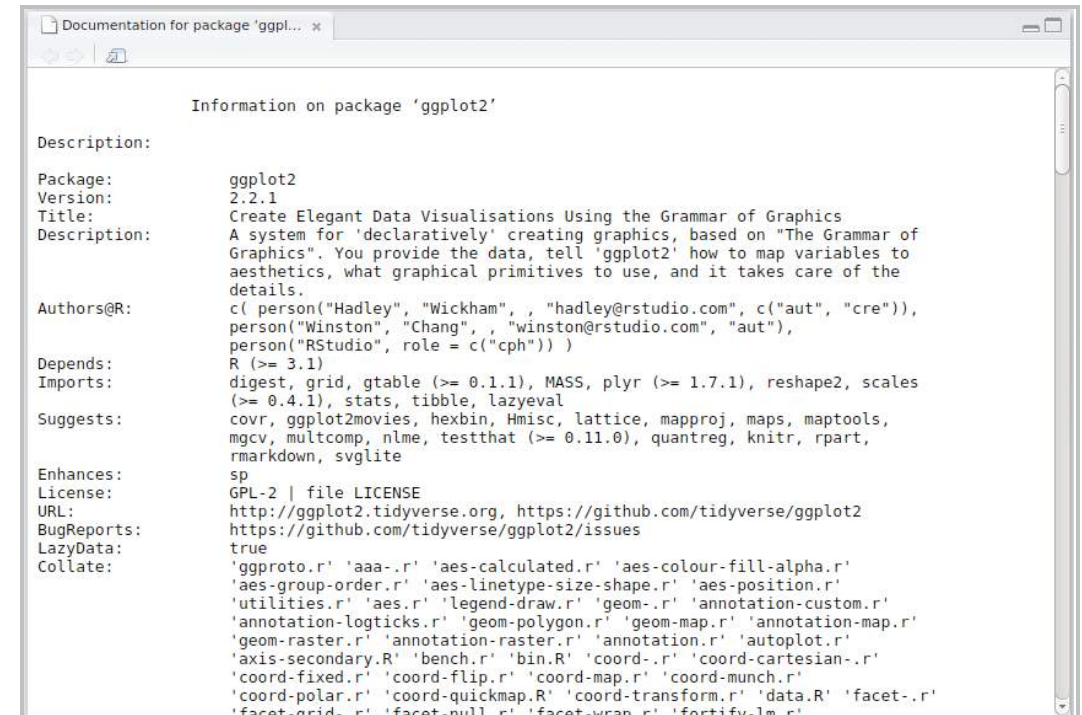
Install `ggplot2`

Since `ggplot2` is an external package, we need to install it first

```
install.packages("ggplot2")

# Then we need to load it to our environment.
library(ggplot2)

# Take a look at the documentation.
library(help = "ggplot2")
```



Working with `ggplot2`

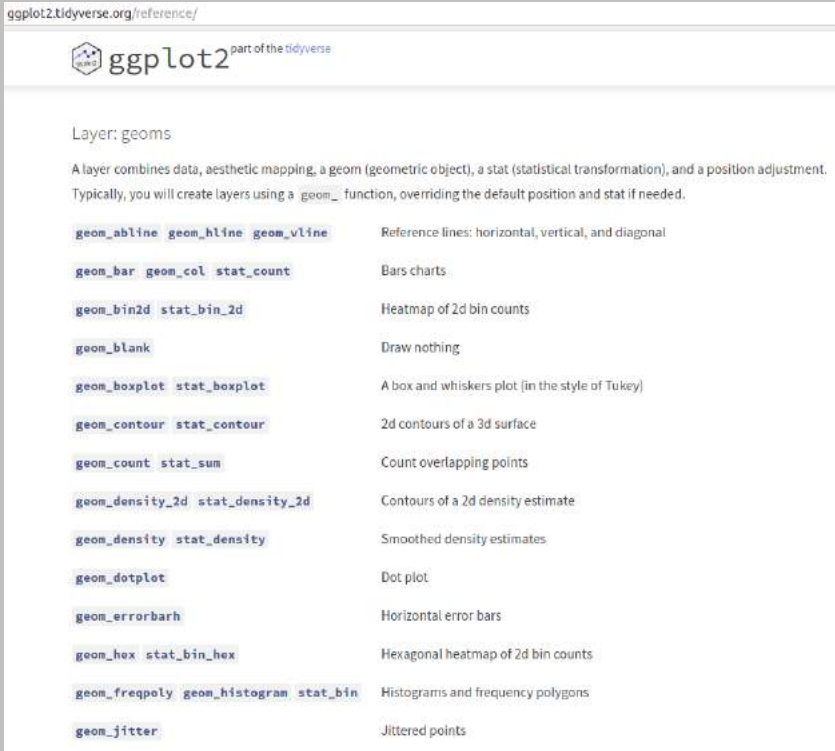
?ggplot

- The main function in the package is `ggplot`
 - It creates the initial plot object (i.e., an empty canvas)
- The two main arguments in the function are:
 - **data**: lets `ggplot` know which data is to be plotted
 - **mapping**: a named list of `aes [thetics]` that lets `ggplot` know which variables are to be mapped to which axes



Working with `ggplot2` (cont'd)

- Each chart in `ggplot2` package is like a **layered cake**
 - After the initial plot object has been created as a base, other layers are put on top of it
- Each chart type corresponds to a geom (i.e., layer)
- We will demonstrate some of the most used geoms in the R community: `geom_histogram`, `geom_point`, `geom_smooth` and `geom_density`
- You can view the entire list of geoms and documentation [here](#)



The screenshot shows the 'ggplot2' reference page from tidyverse.org. It features a header with the 'ggplot2' logo and the text 'part of the tidyverse'. Below the header, there is a section titled 'Layer: geoms' with a descriptive paragraph. The main content is a table listing various geom and stat functions with their corresponding descriptions.

Layer: geoms	
A layer combines data, aesthetic mapping, a geom (geometric object), a stat (statistical transformation), and a position adjustment. Typically, you will create layers using a <code>geom_</code> function, overriding the default position and stat if needed.	
<code>geom_abline</code> <code>geom_hline</code> <code>geom_vline</code>	Reference lines: horizontal, vertical, and diagonal
<code>geom_bar</code> <code>geom_col</code> <code>stat_count</code>	Bars charts
<code>geom_bin2d</code> <code>stat_bin_2d</code>	Heatmap of 2d bin counts
<code>geom_blank</code>	Draw nothing
<code>geom_boxplot</code> <code>stat_boxplot</code>	A box and whiskers plot (in the style of Tukey)
<code>geom_contour</code> <code>stat_contour</code>	2d contours of a 3d surface
<code>geom_count</code> <code>stat_sum</code>	Count overlapping points
<code>geom_density_2d</code> <code>stat_density_2d</code>	Contours of a 2d density estimate
<code>geom_density</code> <code>stat_density</code>	Smoothed density estimates
<code>geom_dotplot</code>	Dot plot
<code>geom_errorbarh</code>	Horizontal error bars
<code>geom_hex</code> <code>stat_bin_hex</code>	Hexagonal heatmap of 2d bin counts
<code>geom_freqpoly</code> <code>geom_histogram</code> <code>stat_bin</code>	Histograms and frequency polygons
<code>geom_jitter</code>	Jittered points

Working with `ggplot2`: stages

Set up

1. Specify data
2. Link data to visuals
3. Assign shapes

Adjust

1. Visual effects
2. Axes
3. Legend

Polish

1. Customize theme
2. Layer statistics
3. Text



Module completion checklist

Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	✓
Describe and build univariate plots to illustrate patterns in data	✓
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	✓
Create a multivariate plot (scatterplot matrix) and evaluate best uses	✓
Formulate the process of using ggplot2 to build plots	✓
Build a histogram with ggplot2	

`ggplot2` stage 1: set up

In order to make a base plot, we need to:

1. Specify data
2. Link data to visuals
3. Assign shapes

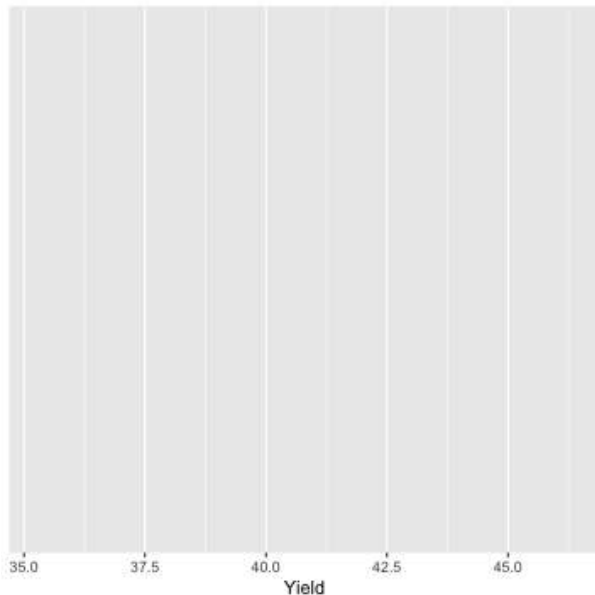
Translating into `ggplot2` syntax will require us to call the `ggplot()` function, and

1. Set its first argument to the data we want to use (e.g., `CMP_subset`)
2. Set its second argument to the mapping of our choice (e.g., `Yield` to be plotted on `x-axis`)
3. Add a geom layer to produce a histogram (e.g., add `geom_histogram` to the base plot)

Building a plot with `geom_histogram`

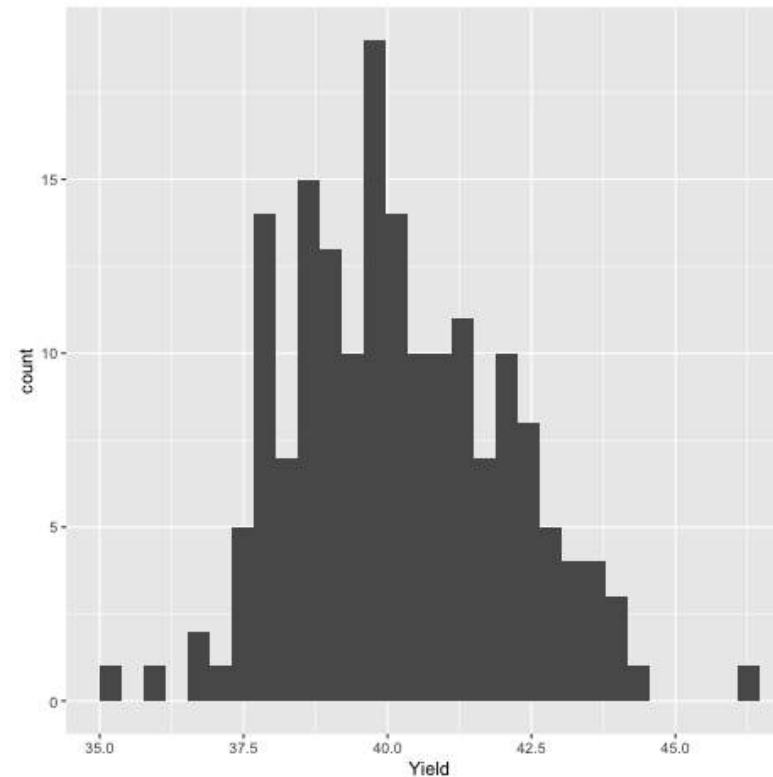
- Let's create a histogram with `ggplot2`
- We start with creating a base plot using the function `ggplot()` and passing data to it

```
# Create base plot.  
ggp1 = ggplot(CMP_subset,    #<-Set data  
              aes(x = Yield)) #<-Set aesthetics  
ggp1
```



Now use `geom_histogram()` to **add** a histogram layer to the base plot

```
# Add histogram layer.  
ggp1 + geom_histogram()
```



`ggplot2` stage 2: adjust

To make the visualization prominent, adjust:

1. Visual effects
2. Axes
3. Legend

Depending on the plot and the final goal, you may want to:

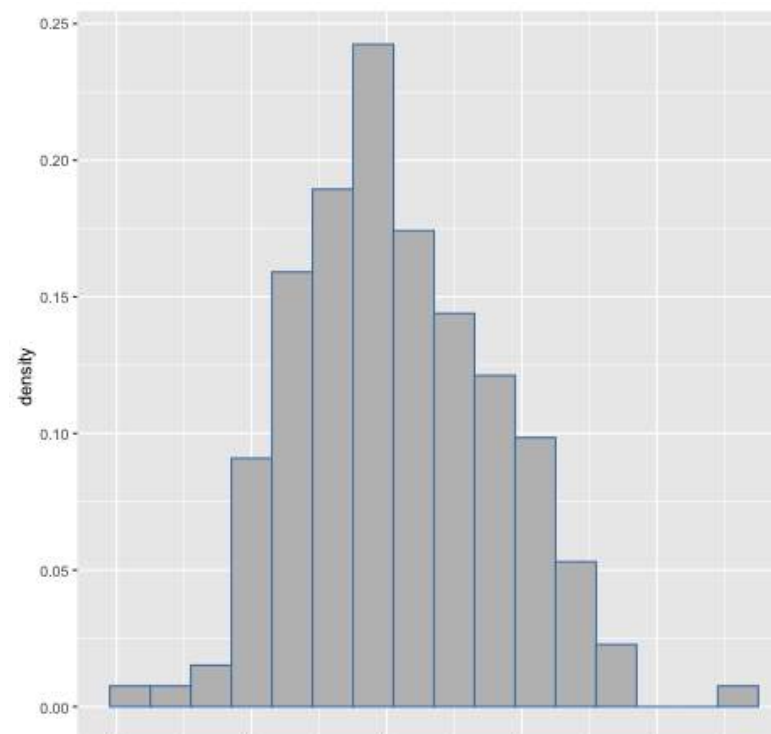
1. Change colors, add more layers with extra information (e.g., additional data sources, reference lines or points, etc.)
2. Add axes labels and adjust the breaks in data (e.g., change scale, bins in the histogram)
3. Add a legend if there are more than one variable or dataset combined on the plot

Plotting with `geom_histogram`: adjust

If instead of frequency counts we want probability density estimates, we can:

1. Change aesthetic mapping of `y-axis` to `type density`
2. Adjust `binwidth` for better smoothness
3. Customize `color` and `fill`

```
ggp1 = ggplot +  
  geom_histogram(aes(y = ..density..), #<- y-axis of type 'density'  
                 binwidth = 0.75, #<- adjust binwidth  
                 color = "steelblue",  
                 fill = "gray")  
ggp1
```



`ggplot2` stage 3: polish

In order to make visualization complete, we need to polish:

1. Layer statistics
2. Customize theme
3. Text

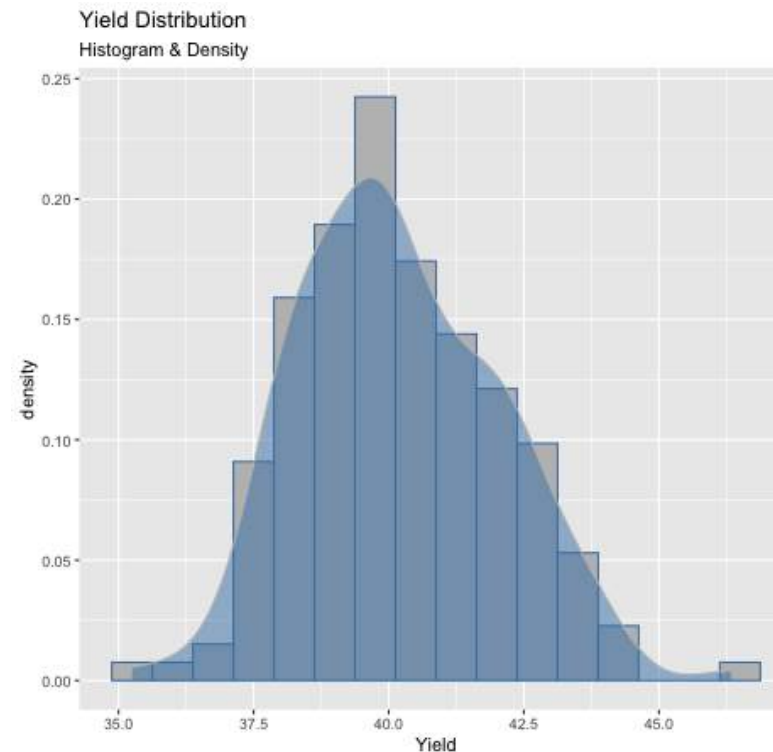
Depending on the plot and the final goal, we may want to:

1. Add statistical layers (e.g., density, mean markers, loss curves, etc.)
2. Customize plot theme
3. Add/change any text labels or add free-form text to the plot

`geom_histogram`: add density layer

- Let's polish our histogram by adding a density layer using `geom_density()`
 - Customize this layer to have with 50% opacity by setting the argument `alpha = 0.5`
 - Fill it with “steelblue” color and add a “gray” border
 - Add a title and a subtitle to our histogram

```
ggp1 = ggp1 +  
  geom_density(alpha = 0.5,      #<- add  
    density layer with 50% opacity  
    color = "gray",  
    fill = "steelblue")+  
  labs(title = "Yield Distribution",  
    subtitle = "Histogram & Density")  
ggp1
```

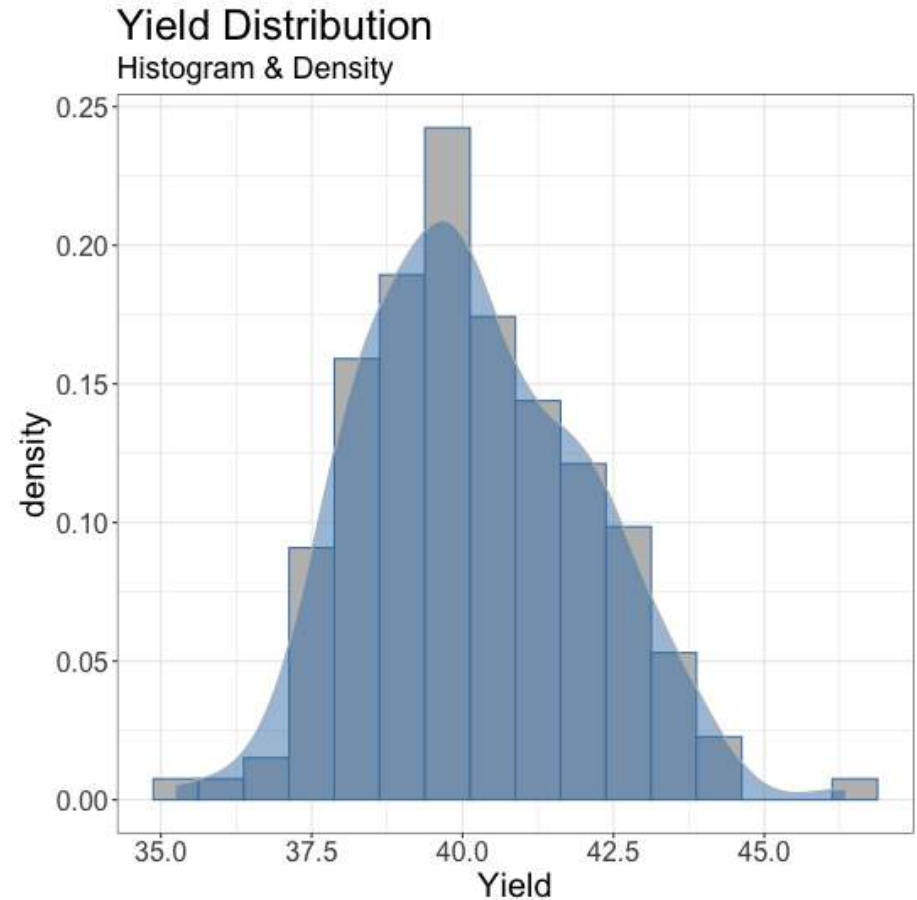


`geom_histogram`: polish theme

Now add a black and white **theme** to overwrite default and customize the theme's elements, such as its title, text, and subtitle

```
ggp1 = ggplot +  
  theme_bw() +      #<- add a black and white  
  theme  
  
  # Customize elements of the theme.  
  theme(axis.title = element_text(size = 20),  
        axis.text = element_text(size = 16),  
        plot.title = element_text(size = 25),  
        plot.subtitle = element_text(size = 18))
```

```
# Display polished plot.  
ggp1
```



Knowledge check 2



Exercise 2



Module completion checklist

Objective	Complete
Define the exploratory data analysis (EDA) cycle and the differences between static and interactive visualizations	✓
Describe and build univariate plots to illustrate patterns in data	✓
Choose when to use bivariate plots and construct examples using base r to illustrate patterns in data	✓
Create a multivariate plot (scatterplot matrix) and evaluate best uses	✓
Formulate the process of using ggplot2 to build plots	✓
Build a histogram with ggplot2	✓

Summary

- **Data visualization** is a vital tool that can help us unearth crucial insights from our dataset
- It enables people to identify analysis results visually and helps to grasp difficult concepts
- In today's module, we have walked through various static visualizations such as **univariate, bivariate, and multivariate** plots and used **ggplot2** to plot a histogram
- In the next module, we will use ggplot2 to create a scatterplot and learn how to transform data for more complex visualizations

This completes our module
Congratulations!