Data Wrangling in R - Data Wrangling - 1

One should look for what is and not what he thinks should be. (Albert Einstein)

Data Wrangling: Topic introduction

In this part of the course, we will cover the following concepts:

- Data wrangling basics
- Use dplyr for data wrangling

Warm-up

- Have a look at this subset of employment data, consisting of employees' first names, last names, dates of birth, and tenure with the company

- What are some of the **problems** you imagine you will encounter when attempting to work with this dataset?

- What **solutions** might you be interested in trying out?

first_name	last_name	birthday	tenure
Alison	Murphy	03/19/1984	NA
Anton	Lepage	11/08/1973	5 years 6 months
Arvind	Agrawal	24/02/1992	14 days
Ayers	Sheila	July 3, 1965	1.5 years

- Share your thoughts in the virtual chat

Module completion checklist

Objective	Complete
Load and evaluate the dataset	
Address missing values in data	

Data wrangling

- Data scientists spend the vast majority of their time preparing data for analysis using an array of data wrangling techniques
- You might also hear these processes referred to as:
  - data cleaning
  - data remediation
  - data munging
- Data wrangling refers to a variety of methods to **transform data** from its current state into a format suitable for analysis and processing
- Both manual and automatic processes contribute to data wrangling

Examples of data wrangling

- Subsetting relevant data and removing irrelevant information
- Locating missing data, and filling or removing incomplete observations
- Arranging and organizing data to make it more structured and readable
- Merging or augmenting a dataset with other data sources

Introducing CMP data set

- We are going to explore a new data set called `ChemicalManufacturingProcess` from AppliedPredictiveModeling package in R
- This dataset contains information about a chemical manufacturing process
- The goal is to understand the relationship between the process and the resulting yield

Introducing CMP data set contd

- Raw material in this process is put through a sequence of 27 steps to generate the final pharmaceutical product
- Of the 57 characteristics, there are:
  - 12 measurements of the biological starting material, and
  - 45 measurements of the manufacturing process

Directory settings

- In order to maximize the efficiency of your workflow, you may want to use the `box` package and encode your directory structure into `variables`

```
install.packages("box")
```

- Let the `main_dir` be the variable corresponding to your materials folder

```
# Set `main_dir` to the location of your materials folder.

path = box::file()
main_dir = dirname(dirname(path))
```

Directory settings (cont'd)

1. We will store all datasets in the `data` directory inside of the materials folder in your environment, so we'll save its path to a `data_dir` variable
2. We will save all of the plots in the `plots` directory corresponding to `plot_dir` variable

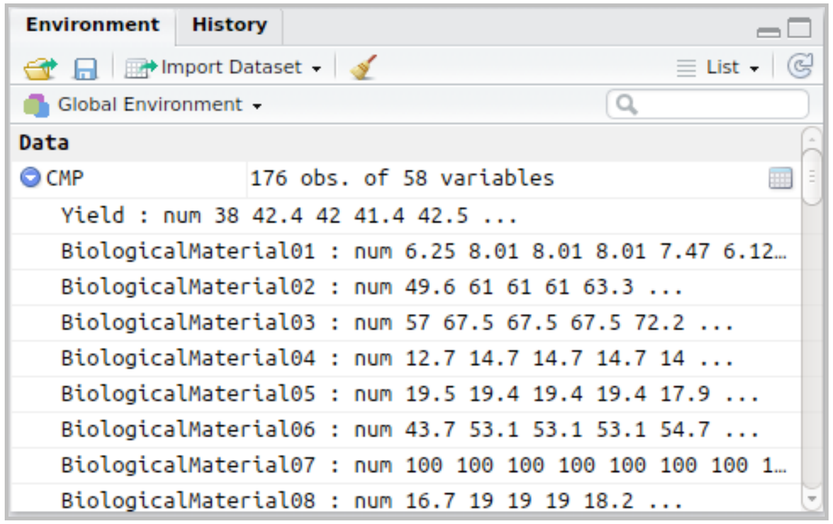
- To append a string to another string, use `paste0` command and pass the strings you would like to paste together

```
# 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")
```

Loading data set

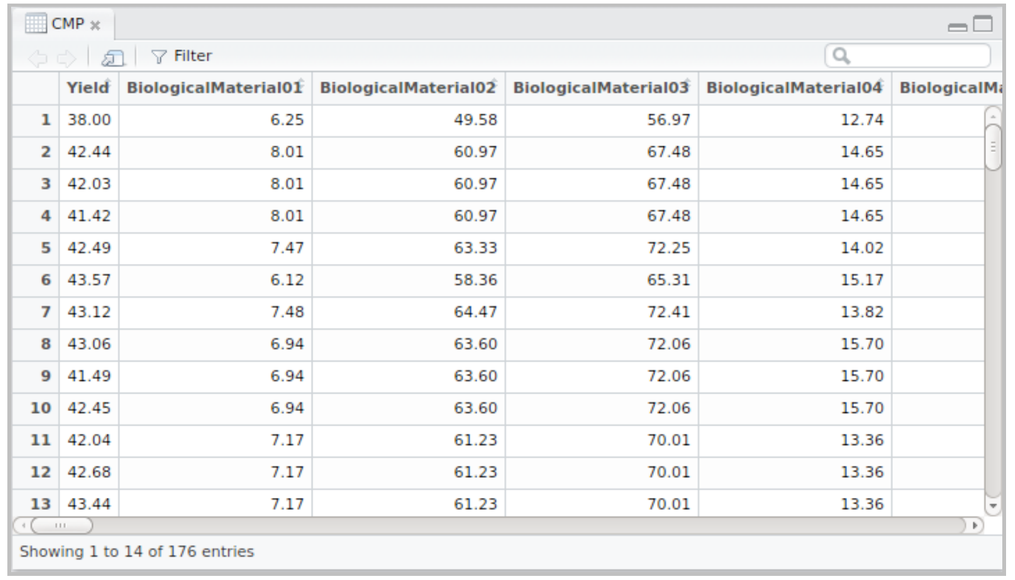
- Let's load the dataset from our `data_dir` into R's environment

```
# Read CSV file called
"ChemicalManufacturingProcess.csv"
CMP = read.csv(file =
file.path(data_dir,"/ChemicalManufacturingProcess.csv")
#<- provide file path
                header = TRUE,            #<- if file
has header set to TRUE
                stringsAsFactors = FALSE) #<- read
strings as characters, not as factors
```



- The dataset consists of `176` observations and `58` variables

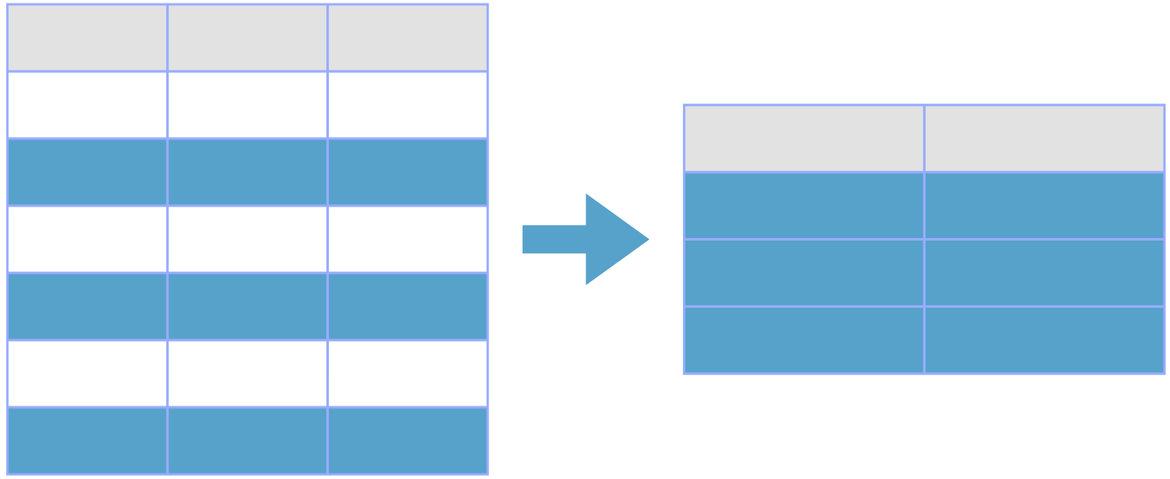
```
# View CMP dataset in tabular data explorer.
View(CMP)
```



	Yield	BiologicalMaterial01	BiologicalMaterial02	BiologicalMaterial03	BiologicalMaterial04	BiologicalMa
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	

Subsetting data

- Data subsetting is a great place to start, especially if you are working with a large high dimensional dataset.
- You may want to disregard particular features or specific observations.
- Subsetting also helps save bandwidth and memory for downstream computation.



Subsetting data

- In this module we will explore a subset of this data set, which includes the following variables
  - yield,
  - 3 material variables, and
  - 3 process variables

CMP

Filter

	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

...

CMP

Filter

ManufacturingProcess01	ManufacturingProcess02	ManufacturingProcess03
NA	NA	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

- First, let's construct a vector of the column indices we would like to save
- Then we'll use it to create a dataframe with only the columns we want

```
# 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           #<- verify that we have the correct set of columns
```

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

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

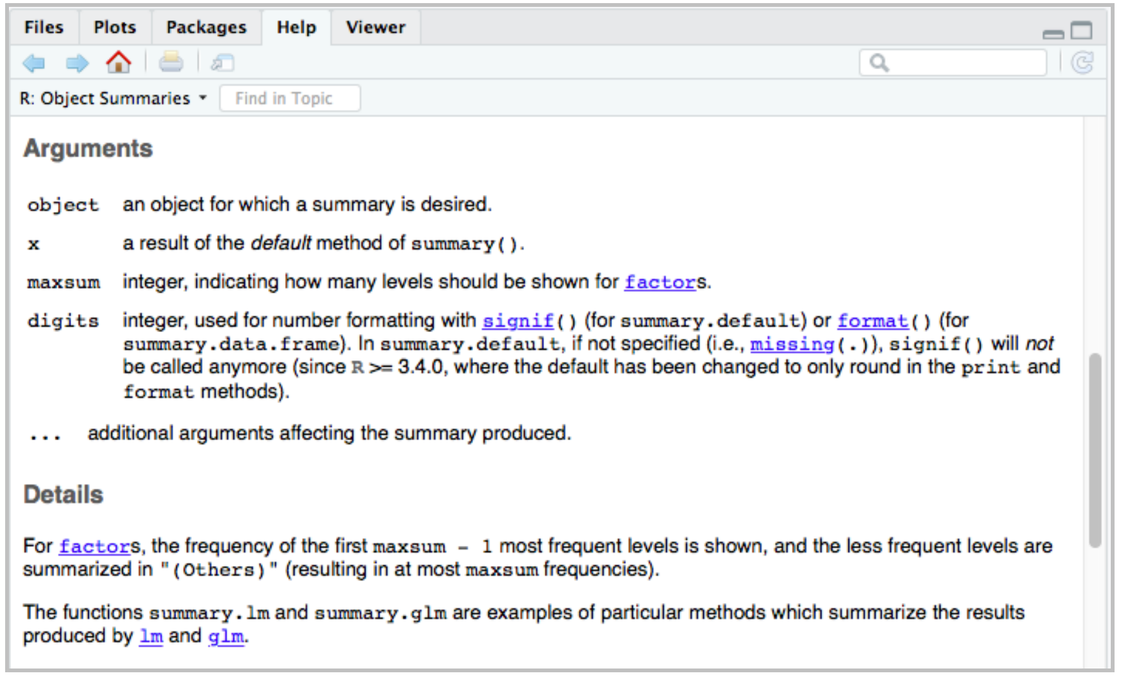
```
'data.frame':   176 obs. of  7 variables:
 $ Yield                  : num  38 42.4 42 41.4 42.5 ...
 $ BiologicalMaterial01   : num  6.25 8.01 8.01 8.01 7.47 6.12 7.48 6.94 6.94 6.94 ...
 $ BiologicalMaterial02   : num  49.6 61 61 61 63.3 ...
 $ BiologicalMaterial03   : num  57 67.5 67.5 67.5 72.2 ...
 $ 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 ...
```

Calculating summary statistics

- Summary statistics provide valuable insights that can inform the remaining steps of the data science cycle
- It can assist with data wrangling by identifying NAs, nulls, and outliers
- It can help you identify the distribution and skewness of your data which impact model choice

Summary statistics

- To get quick summary statistics for a vector, a data frame, or a subset of your data frame, use `summary()`
- For numeric data types, `summary()` will output:
  - Min: The minimum value
  - 1st Qu: The value of the 25th percentile
  - Mean: The mean value
  - Median: The median value
  - 3rd Qu: The value of the 75th percentile
  - Max: The maximum value
  - Number of NAs present



Summary statistics: CMP

```
?summary

summary(data) #<- Either the data frame or single column
```

```
summary(CMP_subset) #<- getting summary statistics of CMP_subset
```

```
     Yield        BiologicalMaterial01 BiologicalMaterial02 BiologicalMaterial03
 Min.   :35.25   Min.   :4.580        Min.   :46.87        Min.   :56.97
 1st Qu.:38.75   1st Qu.:5.978        1st Qu.:52.68        1st Qu.:64.98
 Median :39.97   Median :6.305        Median :55.09        Median :67.22
 Mean   :40.18   Mean   :6.411        Mean   :55.69        Mean   :67.70
 3rd Qu.:41.48   3rd Qu.:6.870        3rd Qu.:58.74        3rd Qu.:70.43
 Max.   :46.34   Max.   :8.810        Max.   :64.75        Max.   :78.25

 ManufacturingProcess01 ManufacturingProcess02 ManufacturingProcess03
 Min.   : 0.00          Min.   : 0.00          Min.   :1.47
 1st Qu.:10.80          1st Qu.:19.30          1st Qu.:1.53
 Median :11.40          Median :21.00          Median :1.54
 Mean   :11.21          Mean   :16.68          Mean   :1.54
 3rd Qu.:12.15          3rd Qu.:21.50          3rd Qu.:1.55
 Max.   :14.10          Max.   :22.50          Max.   :1.60
 NA's   :1              NA's   :3              NA's   :15
```

Module completion checklist

Objective	Complete
Load and evaluate the dataset	✔
Address missing values in data	

Working with missing data: max values

- Missing values pose a significant issue to data analysis even when using simple functions like `max()`

```
# Let's try and compute the maximum value of the 1st manufacturing process.
max_process01 = max(CMP_subset$ManufacturingProcess01)
max_process01
```

```
[1] NA
```

```
max_process02 = max(CMP_subset$ManufacturingProcess01, na.rm = TRUE)
max_process02
```

```
[1] 14.1
```

- To get an actual number, use `na.rm = TRUE` to ignore NA values

Working with missing data: imputing

- What if the function being used does not have the method `na.rm`? Or what if removing `NA`s skews the results?
- Data imputation with one of the following values will help to overcome this:
  - `0`
  - `mean`
  - `median`
  - any other special value appropriate for a given dataset and data type (e.g., handling of categorical variables with missing data should be handled differently from imputing numeric variables)
- Replacing `NA`s with **`mean`** may not work well if the data contains outliers

Working with missing data

- Function `is.na` will provide a vector of `TRUE` or `FALSE` for each element of a given vector
- It is hard to track elements that are indeed `NA` for datasets containing even a moderate number of data points

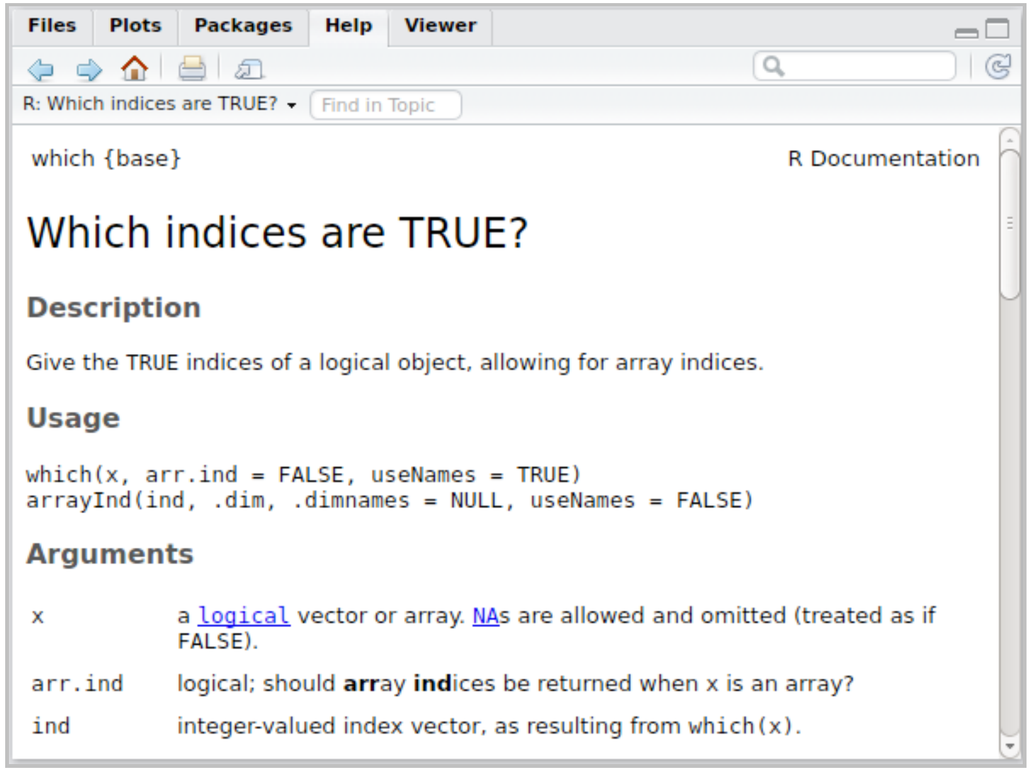
```
# Let's take a look at `ManufacturingProcess01`
# and see if any of the values in it are `NA`.
is.na(CMP_subset$ManufacturingProcess01)
```

```
  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Working with missing data (cont'd)

```
?which
```

- `which` function is an invaluable utility function in R's `base` package
- It takes either a vector/array of `logical` values OR a vector/array of any values with a comparison statement in one of the comparison operators (==, !=, >, <, >=, <=) and a value to which we are comparing to
- It returns the `indices` of all `TRUE` values of the `logical` vector, or the indices of all the values that meet the condition we specified



Working with missing data: Identifying NA values

- Here we use `which()` to identify the index of all values that are `TRUE`

```
# Let's save this vector of logical values to a variable.
is_na_MP01 = is.na(CMP_subset$ManufacturingProcess01)

# To determine WHICH elements in the vector are `TRUE`and are NA, we will use `which` function.

# Since we already have a vector of `TRUE` or `FALSE` logical values
# we only have to give it to `which` and it will return all of the
# indices of values that are `TRUE`.
which(is_na_MP01)
```

```
[1] 1
```

```
# This is also a correct way to set it up.
which(is_na_MP01 == TRUE)
```

```
[1] 1
```

Working with missing data: Locating NA values

- After identifying which entry in the `ManufacturingProcess01` is `NA`, it can be selected programmatically without having to type its index manually using bracket syntax

```
# Let's save the index to a variable.
na_id = which(is_na_MP01)
na_id
```

```
[1] 1
```

```
# Let's view the value at the `na_id` index.
CMP_subset$ManufacturingProcess01[na_id]
```

```
[1] NA
```

Working with missing data: Mean replacement

- The next step is to compute a value suitable for replacing the given `NA`
- For demonstration purposes, use the `mean` of the variable as a replacement

```
# Compute the mean of the `ManufacturingProcess01`.
mean_process01 = mean(CMP_subset$ManufacturingProcess01)
mean_process01
```

```
[1] NA
```

- Set `na.rm = TRUE` in order to compute the `mean` of the variable that contains `NA`s

```
# Compute the mean of the `ManufacturingProcess01` and set `na.rm` to `TRUE`.
mean_process01 = mean(CMP_subset$ManufacturingProcess01, na.rm = TRUE)
mean_process01
```

```
[1] 11.20743
```

Working with missing data

- Next, take the mean and assign it to the missing value within the vector using bracket syntax

```
# Assign the mean to the entry with the `NA`.
CMP_subset$ManufacturingProcess01[na_id] = mean_process01
CMP_subset$ManufacturingProcess01[na_id]
```

```
[1] 11.20743
```

```
max_process01 = max(CMP_subset$ManufacturingProcess01)
max_process01
```

```
[1] 14.1
```

- Did it work?

Working with missing data (cont'd)

- Next, repeat the process for the remaining manufacturing variables

```
# Impute missing values of `ManufacturingProcess02` with the mean
is_na = is.na(CMP_subset$ManufacturingProcess02)
na_id = which(is_na)
mean_process02 = mean(CMP_subset$ManufacturingProcess02, na.rm = TRUE)
CMP_subset$ManufacturingProcess02[na_id] = mean_process02


# Impute missing values of `ManufacturingProcess03` with the mean
is_na = is.na(CMP_subset$ManufacturingProcess03)
na_id = which(is_na)
mean_process03 = mean(CMP_subset$ManufacturingProcess03, na.rm = TRUE)
CMP_subset$ManufacturingProcess03[na_id] = mean_process03
```

Knowledge check

Module completion checklist

Objective	Complete
Load and evaluate the dataset	✔
Address missing values in data	✔

Congratulations on completing this module!

You are now ready to try Tasks 1-5 in the Exercise for this topic