

## Importing Data Your way

Select a file through file explorer

```
read.table(file.choose())
```

file.choose() prompts R to open file explorer

Provide the full path and extension

Directories must be divided by / or \

```
read.table("C:/Users/User/Documents/file.txt")
```

Working Directory

The working directory sets a default location to import and export files

**getwd()** View the current working directory

```
setwd("C:/Users/User/Documents/")
```

Set a new working directory

Once the working directory is set, only the file name and extension needs to be specified

```
read.table("file.txt")
```

## Importing and Exporting Functions

File type

Import

Export

Text files

```
read.table("file.txt",
            header = FALSE, sep = "\t")
```

```
write.table(df, file = "file.txt", sep = "\t",
            row.names = TRUE, col.names = TRUE)
```

Comma separated values

```
read.csv("file.csv", header = TRUE)
```

```
write.csv(df, "file.csv", row.names = TRUE,
          col.names = TRUE)
```

Excel worksheet  
(from openxlsx package)

```
read.xlsx("file.xlsx", sheet = "Sheet1")
read.xlsx("file.xlsx", sheet = 1)
```

Single worksheet: 

```
write.xlsx(df, file = "file.xlsx")
```

  
Multiple worksheets: 

```
write.xlsx(list(S1 = df1, S2 = df2),
            file = "file.xlsx")
```

R data

```
load("file.rda")
```

```
save(list = ls(), file = "file.rda")
```

### Import/Export Arguments

**header** file to be imported contains column names as the first line (TRUE) or not (FALSE)

**col.names** include column names as the first row (TRUE) or not (FALSE) in exported file

**append** if the data should be written as a new worksheet in an existing workbook

**list=ls()** save entire R environment

**sep** how columns are (import) or will (export) be delimited

**row.names** include row names as the first column (TRUE) or not (FALSE) in exported file

**sheet** the sheet number (first is sheet 1) or the name of the worksheet to import/export

**file** the location, file name, and extension of the file to be imported/exported

Deliminators	sep =	Example
Tab	"\t"	A    B
Comma	","	A,B
Whitespace	" "	A B
Semi-colon	";"	A;B
Vertical Bar	" "	A B

## Data frame manipulation with the dplyr package

## Combine data frames

```
bind_rows(df1, df2)
```

Append rows of two or more data frames

The diagram shows two input tables being joined. The first table has columns A and B with rows (X1, a) and (X2, z). The second table has columns A and B with rows (X3, d) and (X4, t). The result table has columns A and B with rows (X1, a), (X2, z), (X3, d), and (X4, t).

```
bind_cols(df1, df2)
```

Append columns of two or more data frames

A	B				
X1	a	+			
X2	z				

```
left_join(df1, df2, by = "A")
```

Append matching rows in df1 and df2

A	B		A	C		A	B	C
x1	a	+	x2	1	→	x1	a	2
x2	z		x1	2		x2	z	1

## Sort rows by value(s)

**arrange** (df, A, B)  
Order rows by values of a one or more columns in sequence (low to high, A to Z)

```
arrange(df, desc(A), desc(B))
```

Order rows by values of a one or more columns in sequence (high to low, Z to A)

## Create new columns

Diagram illustrating a column shift operation:

A	B	C

→

A	B	C	D

```
mutate(df, D = A + B)
```

Compute and append one or more new columns

```
mutate(df, A = ifelse(X, Y, A))
```

If condition X true, then replace with Y, otherwise  
keep original value

## Piping

**%>%** Passes object on left hand side of the first argument of a function on righthand side

Piping with %>% makes code more readable

```
iris %>%
  select(Species, Sepal.Length)%>%
  arrange(Species)
```

## Subset columns

The diagram shows a transformation of a data grid. On the left, a 3x6 grid has columns labeled A, B, C, D, E, and F. Columns A and B are light gray, C and D are dark blue, and E and F are light blue. An arrow points to the right, where a 3x3 grid is shown with columns C, D, and F. These columns are dark blue, light blue, and light blue respectively, matching the colors from the original grid.

```
select(df, C, D, F)
```

Select columns A, B, and C from data frame df

```
select(df, -A, -B)
```

Remove columns A and B  
from data frame df

## Selection helpers

```
select(df, contains("."))
```

Select columns whose name contains a character string

```
select(df, starts_with("Pre"))
```

Select columns whose name starts with a character string

```
select(df, ends_with("ent"))
```

Select columns whose name ends with a character string

```
select(df, everything())
```

Select all columns in sequence unless otherwise specified

```
select(df, lastcol(offset))
```

Select the last column with optional offset

## Subset rows

```
filter(df, A > 7)
```

Subset rows that meet logical criteria

## Logical criteria

- ==** Equals
- !=** Does not equal
- >** Greater than
- >=** Greater than or equal to
- <** Less than
- <=** Less than or equal to
- is.na()** Is NA
- !is.na()** Is not NA
- A %in% c()** Any value in c()
- between(A, left, right)**  
Numeric value between left and right values

---

From the **stringr** package

```
filter(df, str_detect())
```

<code>str_detect(A, "a")</code>	String contains
<code>str_detect(A, "^A")</code>	String begins with
<code>str_detect(A, "s\$")</code>	String ends with
<code>str_detect(A, "\\d")</code>	String contains digits

## Combine logical criteria

AND (&) : All logical criteria met

```
> filter(df, A > 7 & B == "c")
```

OR(|): At least one logical criteria met

```
> filter(df, A > 7 | B == "c")
```

## Aggregate data

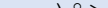
**group\_by** (group)  
Group data into rows  
of the same value

```
group_by(group) %>%  
  mutate(...)
```

Compute new variables by group

**group\_by**(group) %>%  
**summarise**(...)

Compute separate summary row for each group



**ungroup\_by** (group)  
Remove grouping from  
data frame

Diagram illustrating the process of feature group selection. It shows two rows of 3x3 grids. The top row shows a grid with 3 groups (grey, blue, green) being transformed into a grid with 6 groups (grey, blue, green, blue, green, green). The bottom row shows a grid with 3 groups (grey, blue, green) being transformed into a grid with 3 groups (blue, blue, green).

**Contact:** [madgichelp@trentu.ca](mailto:madgichelp@trentu.ca)