Introduction to Data Science (11372 & G 11516)

Semester 2 2023

UNIVERSITY OF CANBERRA

INTRODUCTION TO DATA SCIENCE

Lecture 5

Dr. Ibrahim Radwan

DISTINCTIVE BY DESIGN

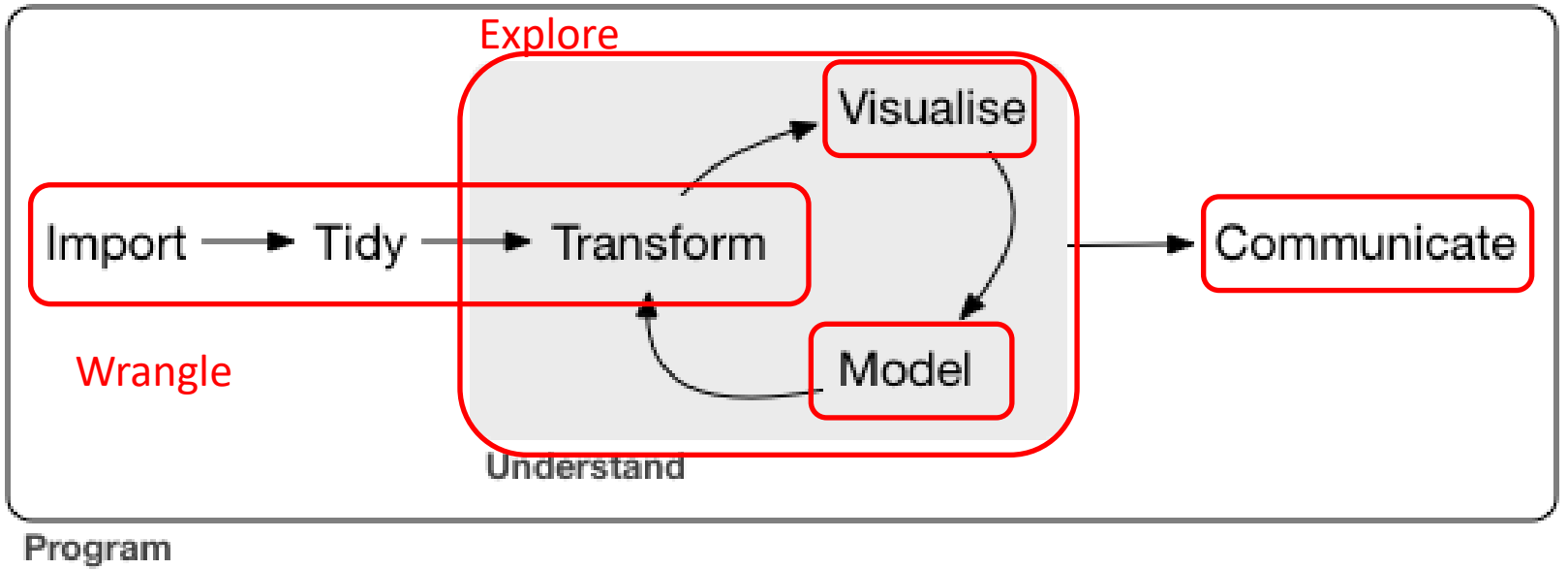
OUTLINE

UNIVERSITY OF CANBERRA

- Data Science, a Practical View
- Data Formats
- Tibbles vs. Data Frames
- Data Import Packages
- Variable Specifications and Parsing

DATA SCIENCE – PRACTICAL PERSPECTIVE

UNIVERSITY OF CANBERRA



Program Steps

1- Reading Data

2- Data Wrangling

3- Data Exploratory

4- Modelling

5- Result Communication

R for Data Science, by *Garrett Grolemund and Hadley Wickham*

DATA FORMATS

UNIVERSITY OF CANBERRA

- Most data scientists deal with the following data formats:
  1. Text files, separated by some delimiters (*e.g.*, spaces, commas (*i.e.*, csv), etc.)
  2. Spreadsheets (*.xls, .xlsx,* etc.)
  3. Databases (MySQL, SQL, Oracle, etc.)
  4. Other software-specific formats
- In the IDS unit, we will mostly deal with the plain and **rectangular** data files such as .txt, .csv, etc.
- As mentioned before, to manipulate these data, we need to read them in **data frames**

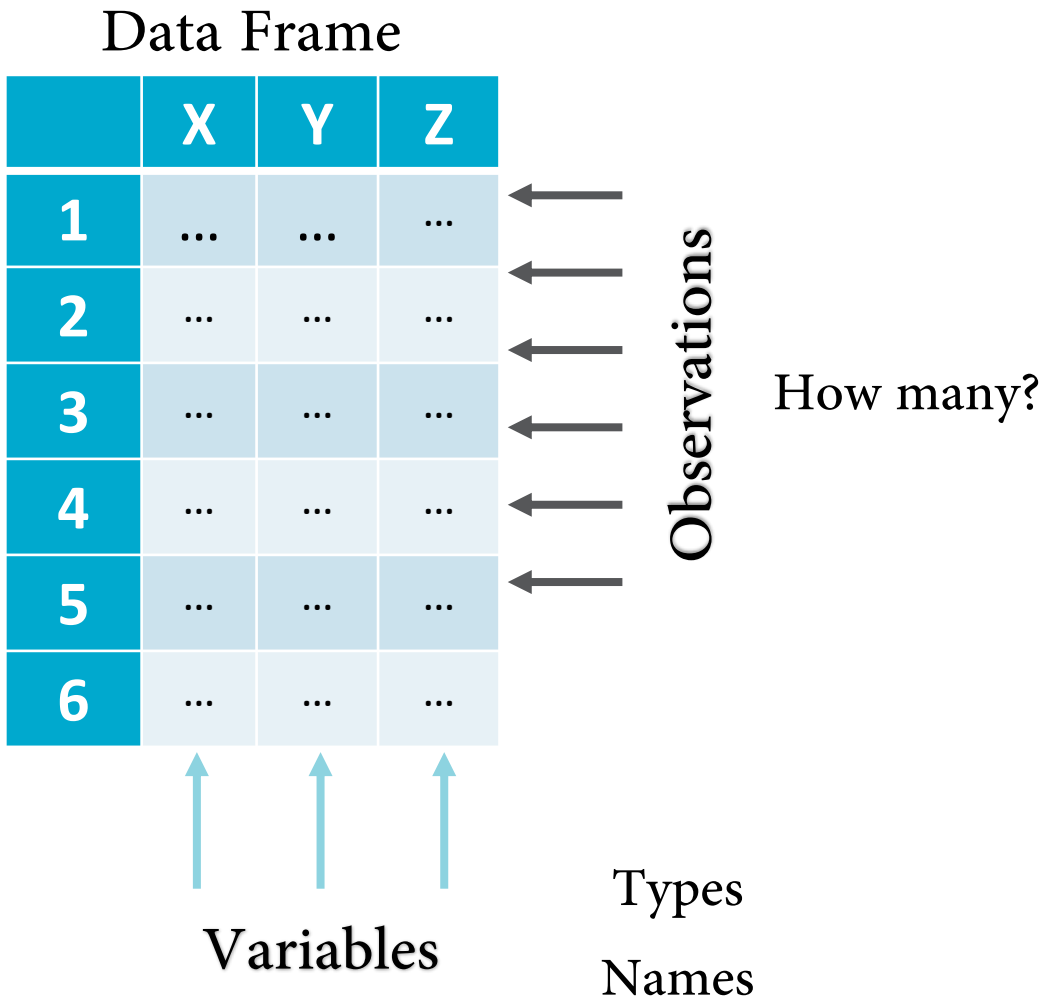
DATA FRAMES – RECAP

UNIVERSITY OF CANBERRA

```
ID, Name, Age
23424, Ana, 45
11234, Charles, 23
77654, Susanne, 76
```

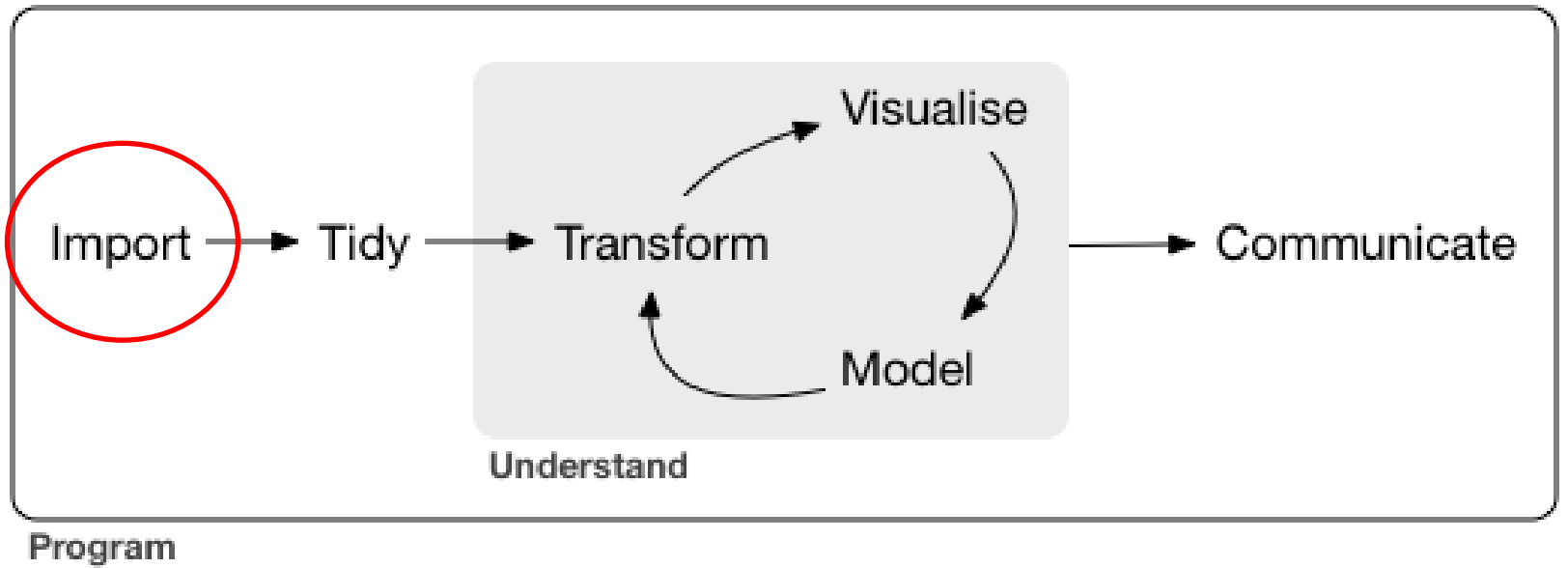
data.csv

How to read this using the base functions in R?



IMPORTING DATA

UNIVERSITY OF CANBERRA



Program Steps

1- Reading Data

2- Data Wrangling

3- Data Exploratory

4- Modelling

5- Result Communication

R for Data Science, by *Garrett Grolemund and Hadley Wickham*

IMPORTING DATA

UNIVERSITY OF CANBERRA

Data.csv

```
ID, Name, Age
23424, Ana, 45
11234, Charles, 23
77654, Susanne, 76
```

How to put the data into DF?

Data Frame

	X	Y	Z
1	…	…	…
2	…	…	…
3	…	…	…
4	…	…	…
5	…	…	…
6	…	…	…

Core Libraries in R

```
read.csv(…)
```

External Libraries

```
read_csv(…)
```

IMPORTING DATA

UNIVERSITY OF CANBERRA

Data.csv

```
ID, Name, Age
23424, Ana, 45
11234, Charles, 23
77654, Susanne, 76
```

Let's try reading using *read.csv()*

IMPORTING DATA

UNIVERSITY OF CANBERRA

ID, Name, Age

23424, Ana, 45

11234, Charles, 23

77654, Susanne, 76

Let's try reading using *read.csv()*

```
### Data Importing via Core Libraries in R
```{r}
# file path
csv_file_path <- "data/data_sample.csv"

# read csv file using the core functions in R
df <- read.csv(csv_file_path)

# check data frame contents
print(df)

# operations on data frame
dim(df)

# check structure of the data frame
str(df)
```
```

ID <int>	Name <chr>	Age <int>
23424	Ana	45
11234	Charles	23
77654	Susanne	76

3 rows

TIBBLES – A CUSTOMIZED DATA FRAMES

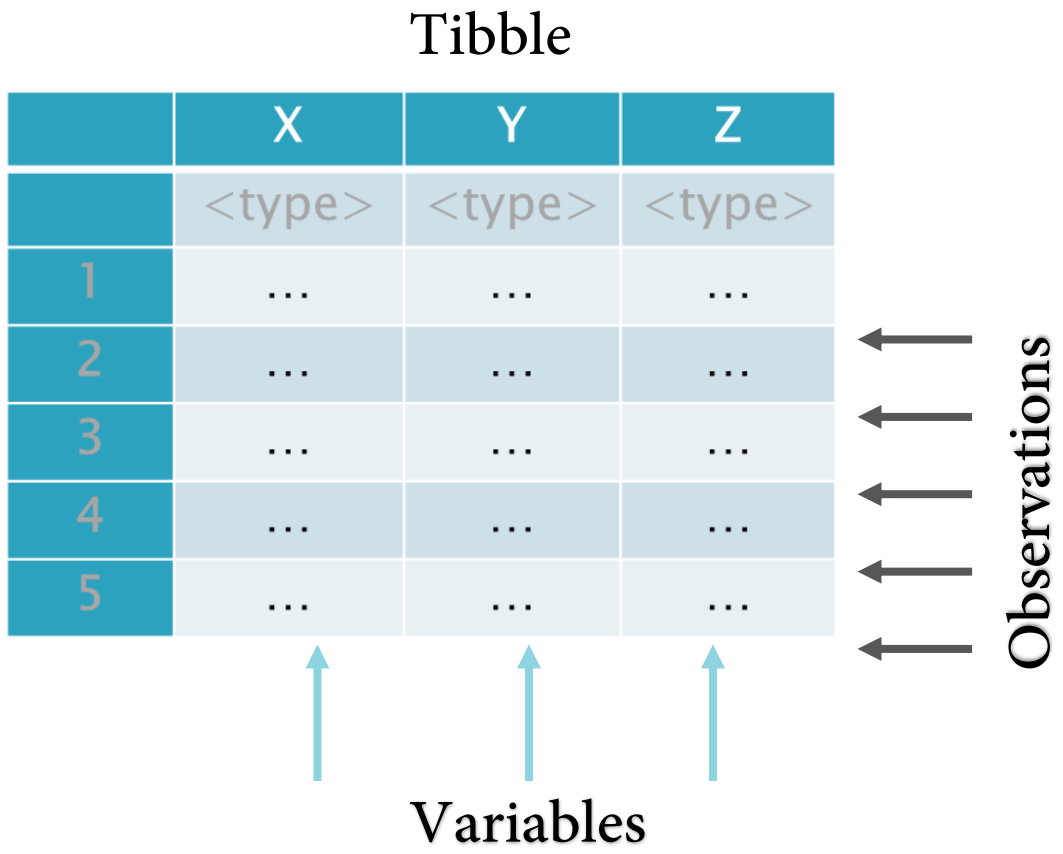
UNIVERSITY OF CANBERRA

- Tibbles are enhanced data frame objects where some of the features have been added and some others were dropped to make the data preparation step a bit easier when dealing with big datasets

int, dbl, chr, factor, lgl, **Col specifications**

10 rows and the columns that fits on the screen **Nicely printed**

Reading data into Tibbles is 10 times faster than the using the base R functions



TIBBLES

UNIVERSITY OF CANBERRA

- Tibbles are part of the **`tidyverse`** library that has been developed by *Hadley Wickham.*
- The **`tidyverse`** library provides an effective way to read, visualise and clean the data
- Tidyverse has sub-packages such as:
  - '*readr*' to import data
  - '*tidyr*' and `dplyr` to wrangle the data
  - '*ggplot2*' to visualize the data

TIBBLES VS. DATA FRAMES

UNIVERSITY OF CANBERRA

Using Tibbles instead of Data Frames

To use tibble, we first need to install and call the tidyverse library package

```{r}
# Step 1 - Install the library, if it is not already installed
if (!("tidyverse" %in% rownames(installed.packages()))){
  install.packages("tidyverse")
  }

# Step 2 - Call the library
library("tidyverse")

# Step 3 - Start using the functions and objects inside this library

# creating a tibble from existing data frame
my_data <- iris # this is an existing data frame
my_data
# how is it printed?

# Can you convert it to tibble?
my_data2 <- as_tibble(my_data)

# is there any difference.
my_data2
```

TIBBLES VS. DATA FRAMES (2)

UNIVERSITY OF CANBERRA

```
34  # tibbles vs. dataframes
35  # 1- prinintg (first 10 rows, all cols that fits with the screen, type of column)
36  tbl_3 <- tibble(
37    a = lubridate::now() + runif(1e3) * 86400,
38    b = lubridate::today() + runif(1e3) * 30,
39    c = 1:1e3,
40    d = runif(1e3),
41    e = sample(letters, 1e3, replace = TRUE)
42  )
43  tbl_3
44  df_3 <- data.frame(
45    a = lubridate::now() + runif(1e3) * 86400,
46    b = lubridate::today() + runif(1e3) * 30,
47    c = 1:1e3,
48    d = runif(1e3),
49    e = sample(letters, 1e3, replace = TRUE)
50  )
51  df_3
52  # if you want to print more rows, n = ?
53  print(tbl_3, n=20, width=Inf)
54  # also you can use the view
55  view(tbl_3)
56  # 2- subsetting
57  df4 <- data.frame(x = 1:3, y = 3:1)
58  class(df4[, 1:2])
59  class(df4[, 1])
60  tbl4 <- tibble(x = 1:3, y = 3:1)
61  class(tbl4[, 1:2])
62  class(tbl4[,1])
63  class(tbl4[[1]])
64  class(tbl4$x)
65  # Tibbles are also stricter with $. Tibbles never do partial matching.
66  df5 <- data.frame(abc = 1)
67  df5$a
68  tbl5 <- tibble(abc = 1)
69  tbl5$a
```

IMPORTING DATA – SCENARIO

UNIVERSITY OF CANBERRA

- Suppose we have the following CSV file:
  - We can call the the `***readr***' library to read the data
  - For example, we can use the read_csv("filename.csv") function to read the contents
  - The returned object is a tibble

ID, Name, Age

23424, Ana, 45

11234, Charles, 23

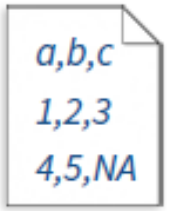
77654, Susanne, 76

```
> read_csv("data_sample.csv")
Parsed with column specification:
cols(
  ID = col_double(),
  Name = col_character(),
  Age = col_double()
)
# A tibble: 3 x 3
     ID Name      Age
  <dbl> <chr>   <dbl>
1 23424 Ana        45
2 11234 Charles    23
3 77654 Susanne    76
>
```

IMPORTING DATA - TIDYVERSE

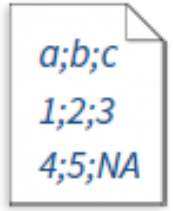
UNIVERSITY OF CANBERRA

```
read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"),
  quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000,
  n_max), progress = interactive())
```



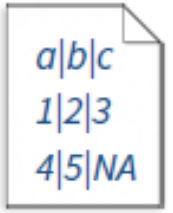
A	B	C
1	2	3
4	5	NA

Comma Delimited Files
read_csv("file.csv"**)**



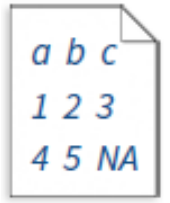
A	B	C
1	2	3
4	5	NA

Semi-colon Delimited Files
read_csv2("file2.csv"**)**



A	B	C
1	2	3
4	5	NA

Files with Any Delimiter
read_delim("file.txt", delim = "|"**)**



A	B	C
1	2	3
4	5	NA

Tab Delimited Files
read_tsv("file.tsv"**)**
Also **read_table()**

To save data into csv or txt file

Comma delimited file
write_csv(x, path, na = "NA", append = FALSE, col_names = !append**)**
File with arbitrary delimiter
write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = !append**)**

IMPORTING DATA – AGAIN!

UNIVERSITY OF CANBERRA

```
### Using Tibbles instead of Data Frames

```{r}
# file path
csv_file_path <- "data/data_sample.csv"

# read csv file using the core functions in R
tbl <- read_csv(csv_file_path)

# check data frame contents
print(tbl)

# operations on data frame
dim(tbl)

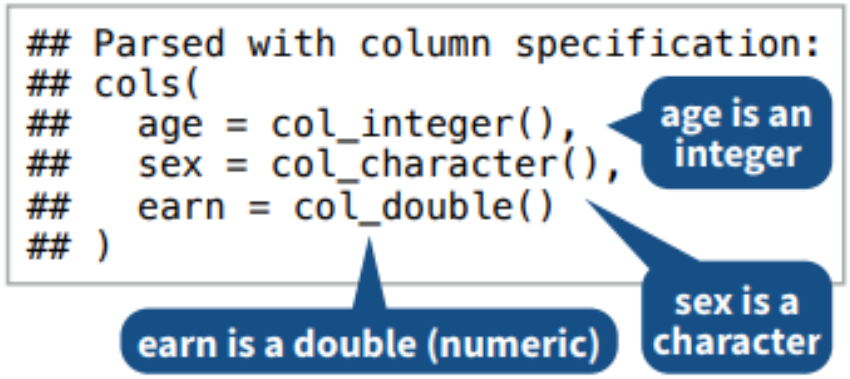
# check structure of the data frame
str(tbl)
```
```

```
spec_tbl_df [3 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ID  : num [1:3] 23424 11234 77654
 $ Name: chr [1:3] "Ana" "Charles" "Susanne"
 $ Age : num [1:3] 45 23 76
 - attr(*, "spec")=
  .. cols(
  ..   ID = col_double(),
  ..   Name = col_character(),
  ..   Age = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

DATA IMPORT – COL SPECIFICATIONS

UNIVERSITY OF CANBERRA

- The functions of *'readr'* sub-package **guess** and assign the types of each column
- A message shows the type of each column will appear after the "read_" function



For each col_* function, there is a parse_* function

col_guess function uses the first 1000 rows from each column to guess the column type.

What about if:

- The first 1000 rows are just special case and the rest are different?
- The first 1000 rows are `na` ?

DATA IMPORT – COL SPECIFICATIONS (2)

UNIVERSITY OF CANBERRA

```r
# col specifications
# logicals
lgl <- parse_logical(c("TRUE", "FALSE", "NA"))
str(lgl)
# integers
intgr <- parse_integer(c("1", "2", "3"))
str(intgr)
# date
dt <- parse_date(c("2010-01-01", "1979-10-14"))
str(dt)
# which strings should be specified as missing
intgr_missing <- parse_integer(c("1", "231", ".", "456"), na = ".")
str(intgr_missing)
# sometimes the parsing fails, NAs will be used when failure is occuring
x <- parse_integer(c("123", "345", "abc", "123.45", "221"))
x
# here you can use problems
problems(x)

# parse numbers with . or , or $ or %
dbl <- parse_double("1.24")
str(dbl)
num <- parse_number("$123,456,789")
num
num <- parse_number("$123,456,789")
num
num <- parse_number("$123.456.789", locale=locale(grouping_mark = '.'))
num
num <- parse_number("123'456'789", locale = locale(grouping_mark = "'"))
num
```

DATA IMPORT – COL SPECIFICATIONS (3)

UNIVERSITY OF CANBERRA

- Using the *problems()* function with the read_* functions from the '*readr*' sub-package to check the problems occurred while parsing and guessing the column types

```
problems(tibble_object)
```

DATA IMPORT – COL SPECIFICATIONS (4)

UNIVERSITY OF CANBERRA

```r
# base function to read data.frame
df <- read.csv(readr_example("challenge.csv"))
df
view(df)
df[1001,]
str(df)
problems(df) # no problems as everything are chars
#####################################
# readr functions to read csv
tbl <- read_csv(readr_example("challenge.csv"))
tbl
view(tbl)
tbl[1001,]
str(tbl)
#####################################
# use problems to check the problems in the tbl parsing
problems(tbl)

# how to fix the problems
# first let us fix the problem by hard-specifying the column types
tbl2 <- read_csv(
  readr_example("challenge.csv"),
  col_types = cols(
    x = col_double(),
    y = col_character()
  )
)
tbl2
tail(tbl2) # there are dates stored with this column
# change the character to date
tbl3 <- read_csv(
  readr_example("challenge.csv"),
  col_types = cols(
    x = col_double(),
    y = col_date()
  )
)
```

KEY TAKEAWAYS

UNIVERSITY OF CANBERRA

- As a data scientist, you will deal with different data formats and data sources
- Reading the rectangular data into Tibbles are much efficient than using the Data.Frame
- Tibble is a customized and enhanced version of the basic data frame, where the features that resist the test of the time have been kept and the frustrating ones have been dropped
- Using the functions inside *'readr'* sub-package to read and to write flat data are much faster and efficient than using the basic functions in R

RECOMMENDED READING

UNIVERSITY OF CANBERRA

- You are recommended to read chapters 10 & 11 from the *"R for Data Science"* book:
  - https://r4ds.had.co.nz/tibbles.html
  - https://r4ds.had.co.nz/data-import.html