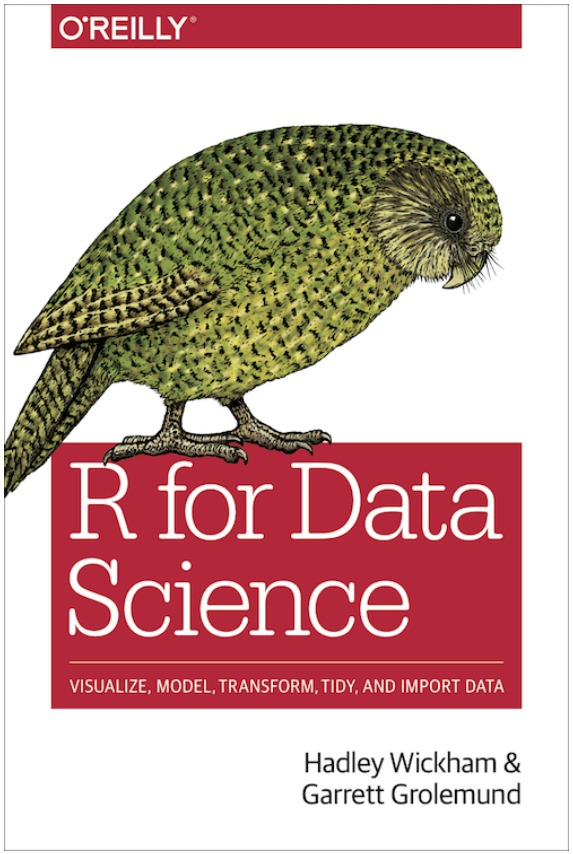
Socio-Informatics 348

Data Importing

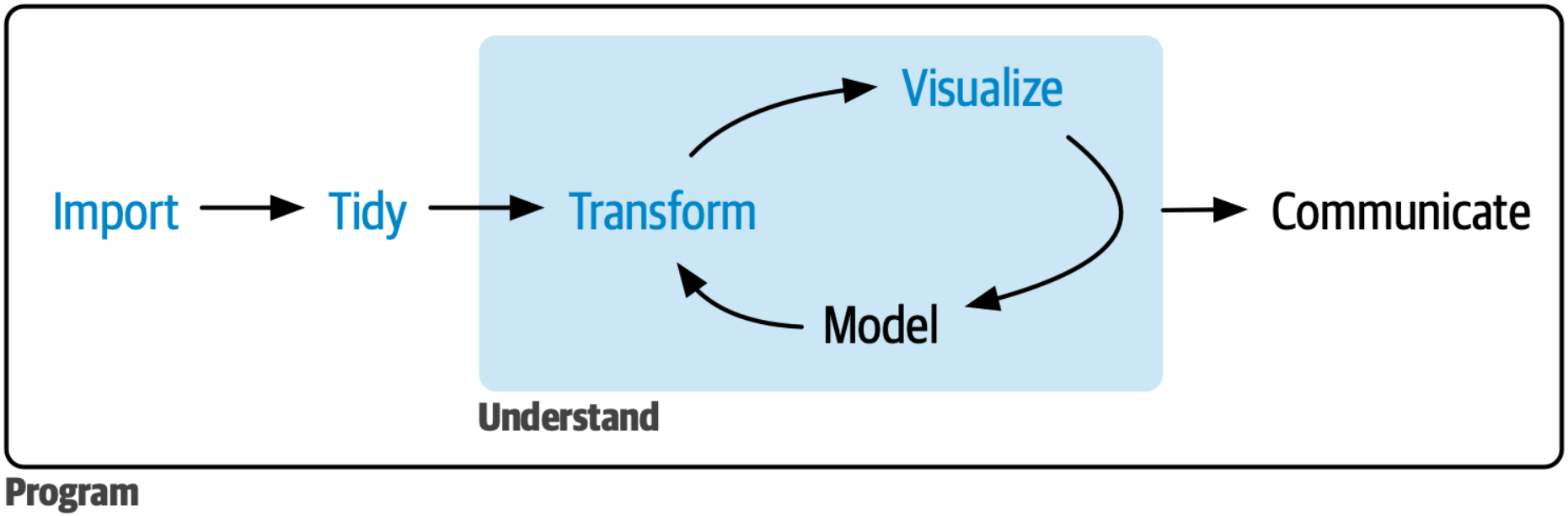
Dr Lisa Martin

Department of Information Science
Stellenbosch University

Today's Reading



R for Data Science, Wholegame, Importing


Import
Tidy
Transform
Visualize
Model
Communicate
Understand
Program

Example table:

Table 7.1: Data from the students.csv file as a table.

Student ID	Full Name	favourite.food	mealPlan	AGE
1	Sunil Huffmann	Strawberry yoghurt	Lunch only	4
2	Barclay Lynn	French fries	Lunch only	5
3	Jayendra Lyne	N/A	Breakfast and lunch	7
4	Leon Rossini	Anchovies	Lunch only	NA
5	Chidiegwu Dunkel	Pizza	Breakfast and lunch	five
6	Güvenç Attila	Ice cream	Lunch only	6

Comma Separated Values (csv)

Most common rectangular data file type:

```
Student ID,Full Name,favourite.food,mealPlan,AGE
1,Sunil Huffmann,Strawberry yoghurt,Lunch only,4
2,Barclay Lynn,French fries,Lunch only,5
3,Jayendra Lyne,N/A,Breakfast and lunch,7
4,Leon Rossini,Anchovies,Lunch only,
5,Chidiegwu Dunkel,Pizza,Breakfast and lunch,five
6,Güvenç Attila,Ice cream,Lunch only,6
```

read_csv() function:

```
students <- read_csv("data/students.csv")
#> Rows: 6 Columns: 5
#> ── Column specification ────────────────────────────────────────────────────
#> Delimiter: ","
#> chr (4): Full Name, favourite.food, mealPlan, AGE
#> dbl (1): Student ID
#>
#> ℹ Use `spec()` to retrieve the full column specification for this data.
#> ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

- From the `readr` package in the `tidyverse`
- Note the location of the file
- Use getwd() to find the current working directory
- Make sure the data folder and file are in the same directory

The data in RStudio:

```
students
#> # A tibble: 6 × 5
#>   `Student ID` `Full Name`      favourite.food     mealPlan             AGE
#>          <dbl> <chr>            <chr>              <chr>                <chr>
#> 1            1 Sunil Huffmann   Strawberry yoghurt Lunch only           4
#> 2            2 Barclay Lynn     French fries       Lunch only           5
#> 3            3 Jayendra Lyne    N/A                Breakfast and lunch  7
#> 4            4 Leon Rossini     Anchovies          Lunch only           <NA>
#> 5            5 Chidiegwu Dunkel Pizza              Breakfast and lunch  five
#> 6            6 Güvenç Attila    Ice cream          Lunch only           6
```

- Compare the NAs in `favourite.food` and `AGE` columns
- Should be NA, but was written as "N/A" in the csv file

The data in RStudio:

NA Values:

```
students <- read_csv("data/students.csv", na = c("N/A", ""))

students
#> # A tibble: 6 × 5
#>   `Student ID` `Full Name`      favourite.food     mealPlan             AGE
#>          <dbl> <chr>            <chr>              <chr>                <chr>
#> 1            1 Sunil Huffmann   Strawberry yoghurt Lunch only           4
#> 2            2 Barclay Lynn     French fries       Lunch only           5
#> 3            3 Jayendra Lyne    <NA>               Breakfast and lunch  7
#> 4            4 Leon Rossini     Anchovies          Lunch only           <NA>
#> 5            5 Chidiegwu Dunkel Pizza              Breakfast and lunch  five
#> 6            6 Güvenç Attila    Ice cream          Lunch only           6
```

- Use na = "N/A" to specify how to treat NAs that are default

The data in RStudio:

Variable names

```
students |>
  rename(
    student_id = `Student ID`,
    full_name = `Full Name`
  )
#> # A tibble: 6 × 5
#>   student_id full_name        favourite.food     mealPlan             AGE
#>        <dbl> <chr>            <chr>              <chr>                <chr>
#> 1          1 Sunil Huffmann   Strawberry yoghurt Lunch only           4
#> 2          2 Barclay Lynn     French fries       Lunch only           5
#> 3          3 Jayendra Lyne    <NA>               Breakfast and lunch  7
#> 4          4 Leon Rossini     Anchovies          Lunch only           <NA>
#> 5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch  five
#> 6          6 Güvenç Attila    Ice cream          Lunch only           6
```

- `rename()` function to rename variables
- From the `dplyr` package in the `tidyverse`

The data in RStudio:

Variable names

```
students |> janitor::clean_names()
#> # A tibble: 6 × 5
#>   student_id full_name        favourite_food     meal_plan           age
#>        <dbl> <chr>            <chr>              <chr>               <chr>
#> 1          1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
#> 2          2 Barclay Lynn     French fries       Lunch only          5
#> 3          3 Jayendra Lyne    <NA>               Breakfast and lunch 7
#> 4          4 Leon Rossini     Anchovies          Lunch only          <NA>
#> 5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
#> 6          6 Güvenç Attila    Ice cream          Lunch only          6
```

- Use `library(janitor)`

The data in RStudio:

Variable types

```
students <- students |>
  janitor::clean_names() |>
  mutate(
    meal_plan = factor(meal_plan),
    age = parse_number(if_else(age == "five", "5", age))
  )

students
#> # A tibble: 6 × 5
#>   student_id full_name        favourite_food     meal_plan             age
#>        <dbl> <chr>            <chr>              <fct>                 <dbl>
#> 1          1 Sunil Huffmann   Strawberry yoghurt Lunch only              4
#> 2          2 Barclay Lynn     French fries       Lunch only              5
#> 3          3 Jayendra Lyne    <NA>               Breakfast and lunch     7
#> 4          4 Leon Rossini     Anchovies          Lunch only             NA
#> 5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch     5
#> 6          6 Güvenç Attila    Ice cream          Lunch only              6
```

Other arguments

Notes/Comments in your csv

```
[some notes written here]
Student ID,Full Name,favourite.food,mealPlan,AGE
1,Sunil Huffmann,Strawberry yoghurt,Lunch only,4
2,Barclay Lynn,French fries,Lunch only,5
3,Jayendra Lyne,N/A,Breakfast and lunch,7
4,Leon Rossini,Anchovies,Lunch only,
5,Chidiegwu Dunkel,Pizza,Breakfast and lunch,five
6,Güvenç Attila,Ice cream,Lunch only,6
```

```
students <- read_csv("data/students.csv", na = c("N/A", ""), skip = 1)
```

Other arguments

Notes/Comments in your csv

```
# Some comments written here
Student ID,Full Name,favourite.food,mealPlan,AGE
1,Sunil Huffmann,Strawberry yoghurt,Lunch only,4
2,Barclay Lynn,French fries,Lunch only,5
3,Jayendra Lyne,N/A,Breakfast and lunch,7
4,Leon Rossini,Anchovies,Lunch only,
5,Chidiegwu Dunkel,Pizza,Breakfast and lunch,five
6,Güvenç Attila,Ice cream,Lunch only,6
```

```
students <- read_csv("data/students.csv", na = c("N/A", ""), comment = "#")
```

Other arguments

col_names

```
1,Sunil Huffmann,Strawberry yoghurt,Lunch only,4
2,Barclay Lynn,French fries,Lunch only,5
3,Jayendra Lyne,N/A,Breakfast and lunch,7
4,Leon Rossini,Anchovies,Lunch only,
5,Chidiegwu Dunkel,Pizza,Breakfast and lunch,five
6,Güvenç Attila,Ice cream,Lunch only,6
```

```
students <- read_csv("data/students.csv", na = c("N/A", ""), col_names = FALSE)
```

Other arguments

col_names

```
1,Sunil Huffmann,Strawberry yoghurt,Lunch only,4
2,Barclay Lynn,French fries,Lunch only,5
3,Jayendra Lyne,N/A,Breakfast and lunch,7
4,Leon Rossini,Anchovies,Lunch only,
5,Chidiegwu Dunkel,Pizza,Breakfast and lunch,five
6,Güvenç Attila,Ice cream,Lunch only,6
```

```
students <- read_csv("data/students.csv", na = c("N/A", ""),
                     col_names = c("student_id","full_name","favourite_food", "meal_plan", "age"))
```

Other formats

- `read_csv2()` reads semicolon-separated files.
  - These use ; instead of , to separate fields and are common in countries that use , as the decimal marker.
- `read_tsv()` reads tab-delimited files.
- `read_delim()` reads in files with any delimiter, attempting to automatically guess the delimiter if you don't specify it.
- `read_fwf()` reads fixed-width files.
- `read_table()` reads a common variation of fixed-width files where columns are separated by white space.
- `read_log()` reads Apache-style log files.

Column Types

- `readr` automatically guesses the type of each column
  - Logical: Only F, T, FALSE, or TRUE (ignoring case)
  - Numerical: Contains only numbers (e.g., 1, -4.5, 5e6, Inf)
  - Date or Date-Time: ISO8601 standard format (more on dates later)
  - Character (string): Contains any other text
- You can also specify the type of each column

Column Types

- You can also specify the type of each column
- Does not need to be specified for ALL columns

```
df <- read_csv("data.csv",
               col_types = cols(
                 column1 = col_character(),                  # Text / string
                 column2 = col_double(),                     # Numeric (decimal)
                 column3 = col_integer(),                    # Integer
                 column4 = col_date(format = "%Y-%m-%d"),    # Date
                 column5 = col_logical()                     # TRUE/FALSE
               ))
```

Reading from multiple files

```
sales_files <- c("data/01-sales.csv", "data/02-sales.csv", "data/03-sales.csv")
read_csv(sales_files, id = "file")
#> # A tibble: 19 × 6
#>   file                month    year brand  item     n
#>   <chr>               <chr>   <dbl> <dbl> <dbl> <dbl>
#> 1 data/01-sales.csv January  2019     1  1234     3
#> 2 data/01-sales.csv January  2019     1  8721     9
#> 3 data/01-sales.csv January  2019     1  1822     2
#> 4 data/01-sales.csv January  2019     2  3333     1
#> 5 data/01-sales.csv January  2019     2  2156     9
#> 6 data/01-sales.csv January  2019     2  3987     6
#> # i 13 more rows
```

- `id = "file"` tells `read_csv()` to create a new column called `id` that contains the name of the file

Reading from multiple files

- What if there are lots of similar files in a folder?
- Use the `list.files()` function to get a list of file names
- Note that there needs to be a common pattern in the file names

```
sales_files <- list.files("data", pattern = "sales\\.csv$", full.names = TRUE)
sales_files
#> [1] "data/01-sales.csv" "data/02-sales.csv" "data/03-sales.csv"
```

Writing to files

Writing to csv

```
write_csv(students, "students.csv")
```

- Writing as csv is the most common
- However, you lose some information when you do this
- For example, the `meal_plan` column is a factor
- When you write it to a csv, it is converted to a character vector

Writing to files

Alternative: Writing to rds

```
write_csv(students, "students.csv")
```

- Writing as csv is the most common
- However, you lose some information when you do this
- For example, the `favourite.food` column is a factor
- When you write it to a csv, it is converted to a character vector