CfDRA
Center for Data Research & Analytics

Data Science Journey Using R

Jaynal Abedin

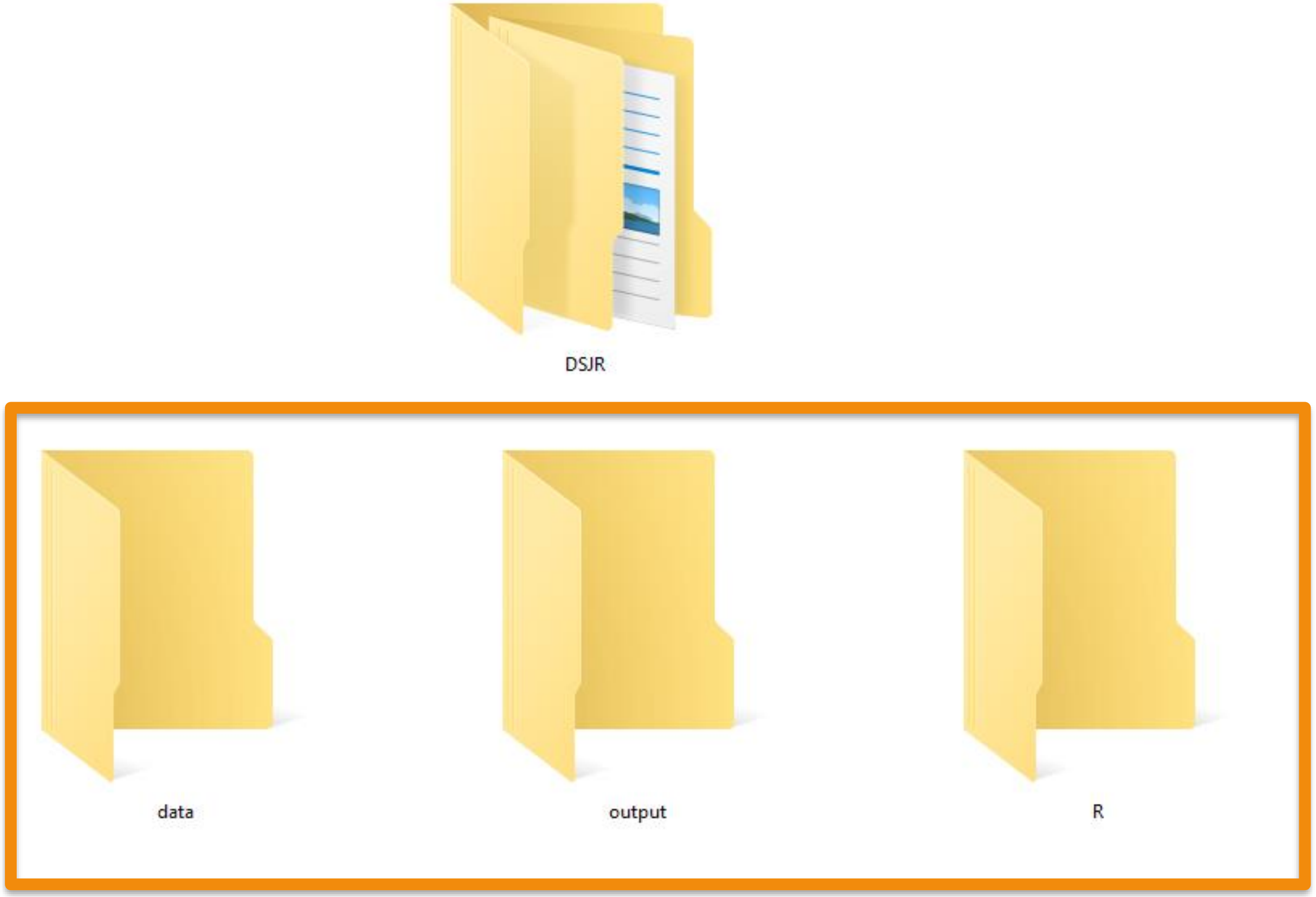
PhD in Data Science

15-Mar-2025

Best Practices for
Cleaning, Organizing, and Preparing Data for Analysis

RStudio Project

Use **RStudio Project** to organize your files

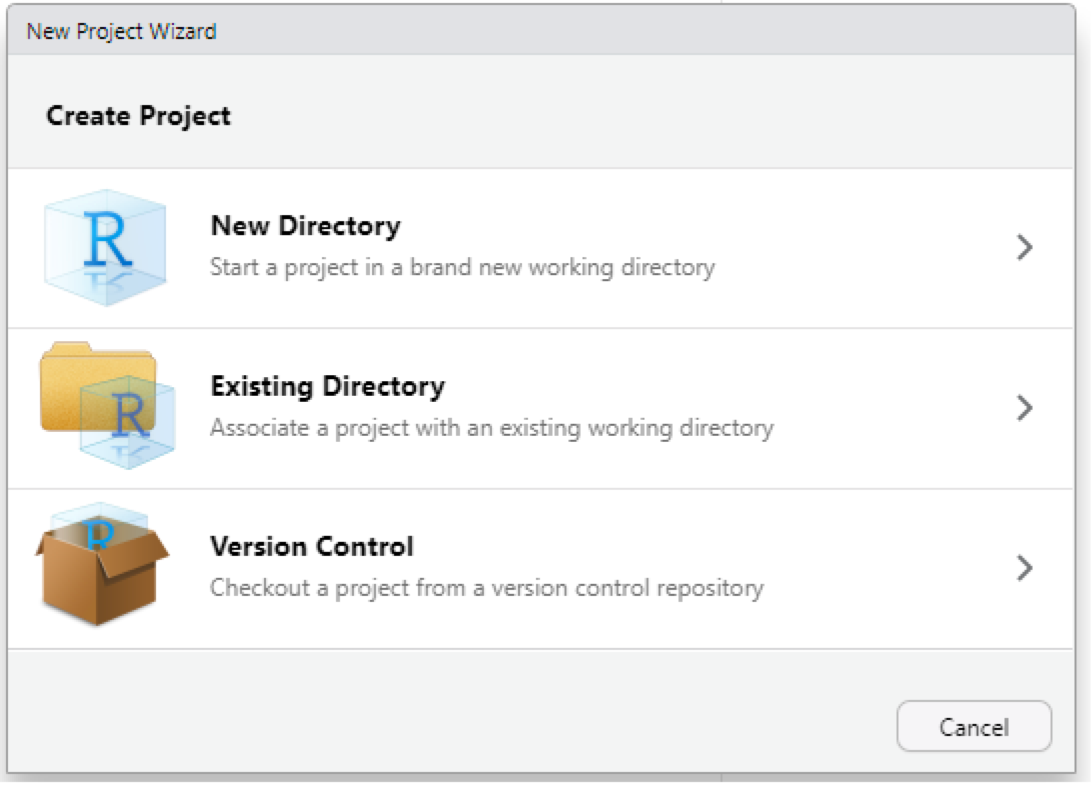


Project Directory →

Sub-Directories →

RStudio Project

Creating RStudio Project: **File→ New Project…**



New Project Wizard

Create Project

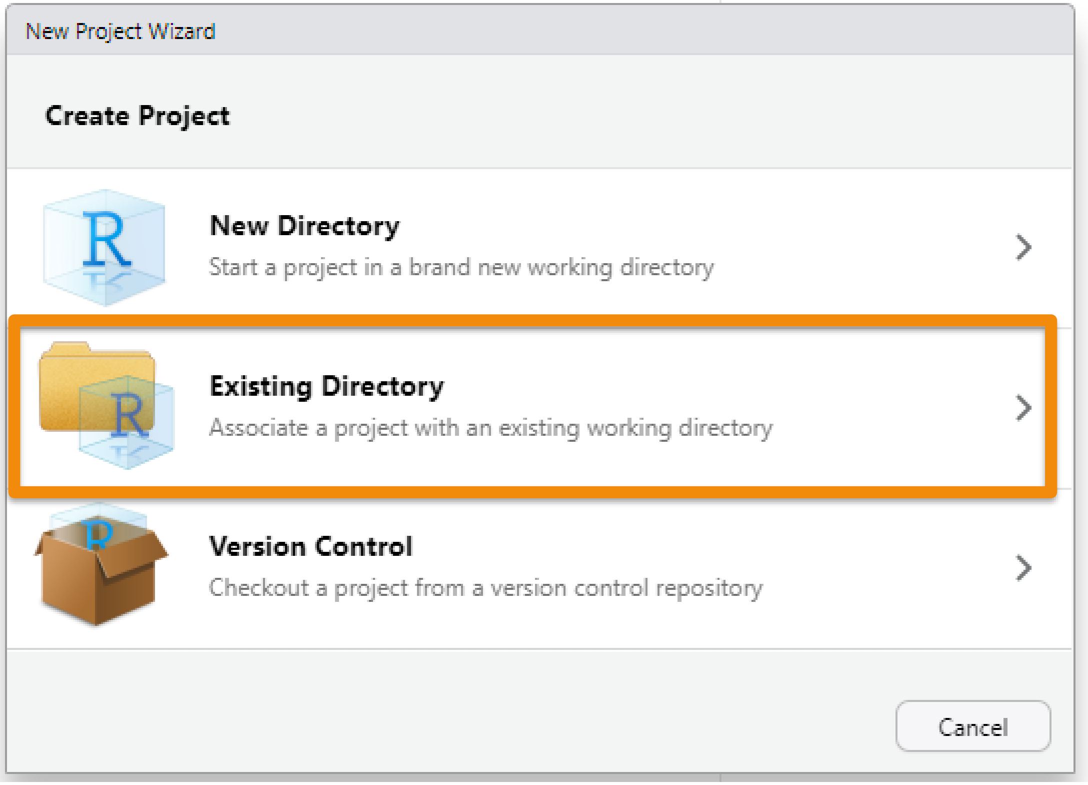
New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

RStudio Project

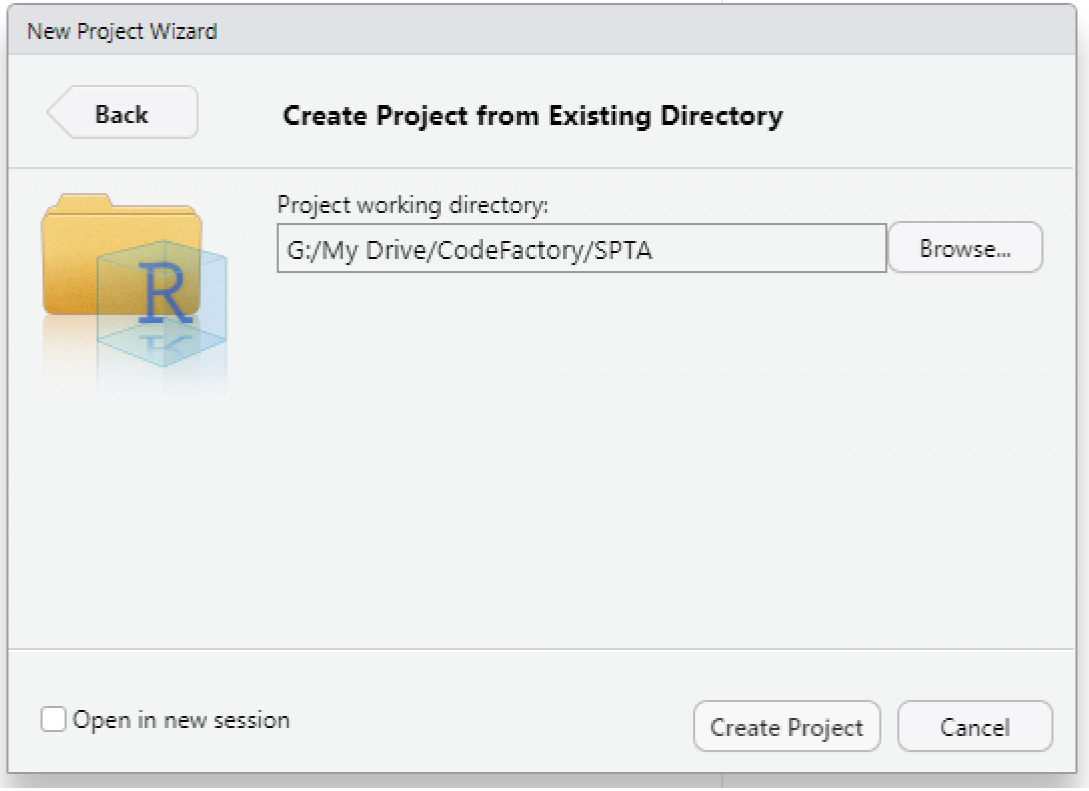
Creating RStudio Project: **File→ New Project…**



New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

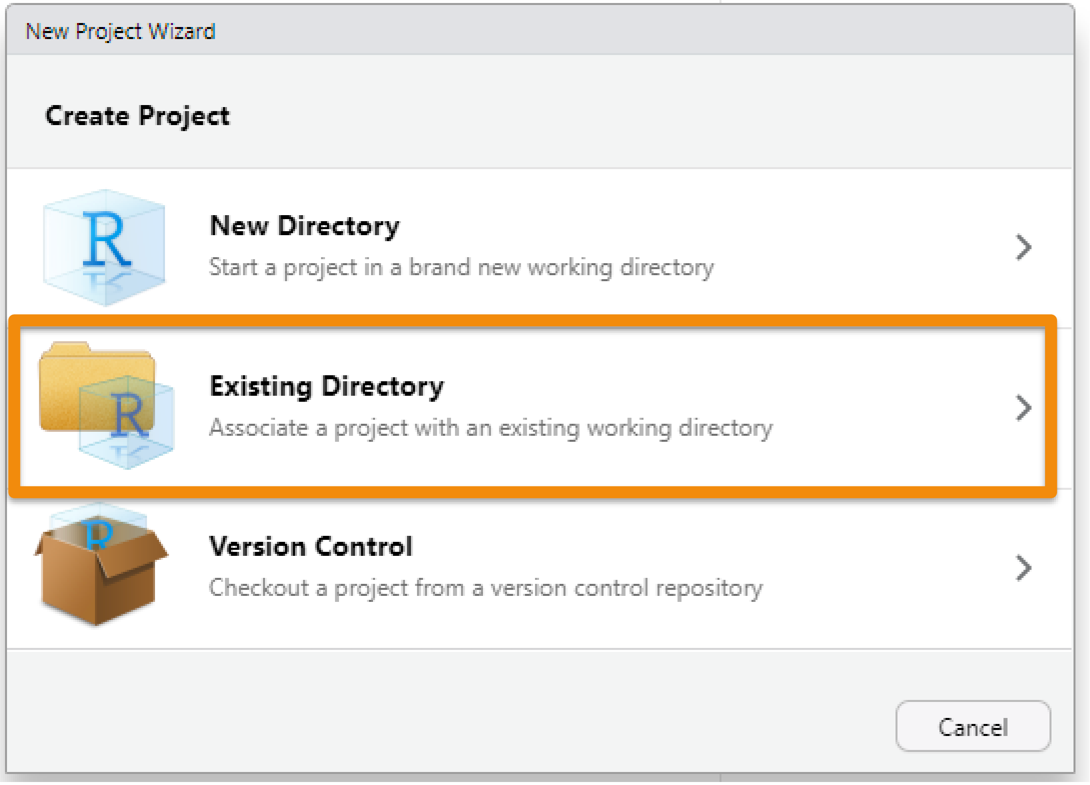


Project working directory: G:/My Drive/CodeFactory/SPTA

□ Open in new session

RStudio Project

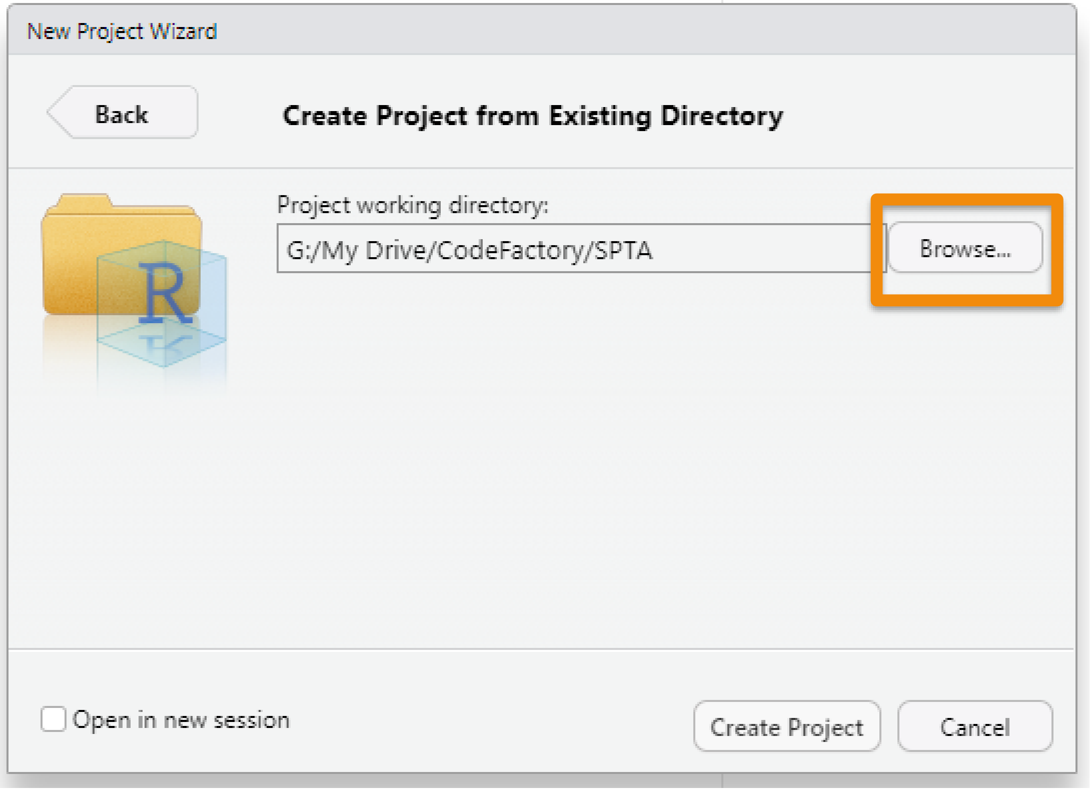
Creating RStudio Project: **File→ New Project...**



New Project Wizard

Create Project

New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel



New Project Wizard

Create Project from Existing Directory

Project working directory: G:/My Drive/CodeFactory/SPTA

Choose your desired folder for the project's directory

RStudio Project

- All your scripts should go into →

R

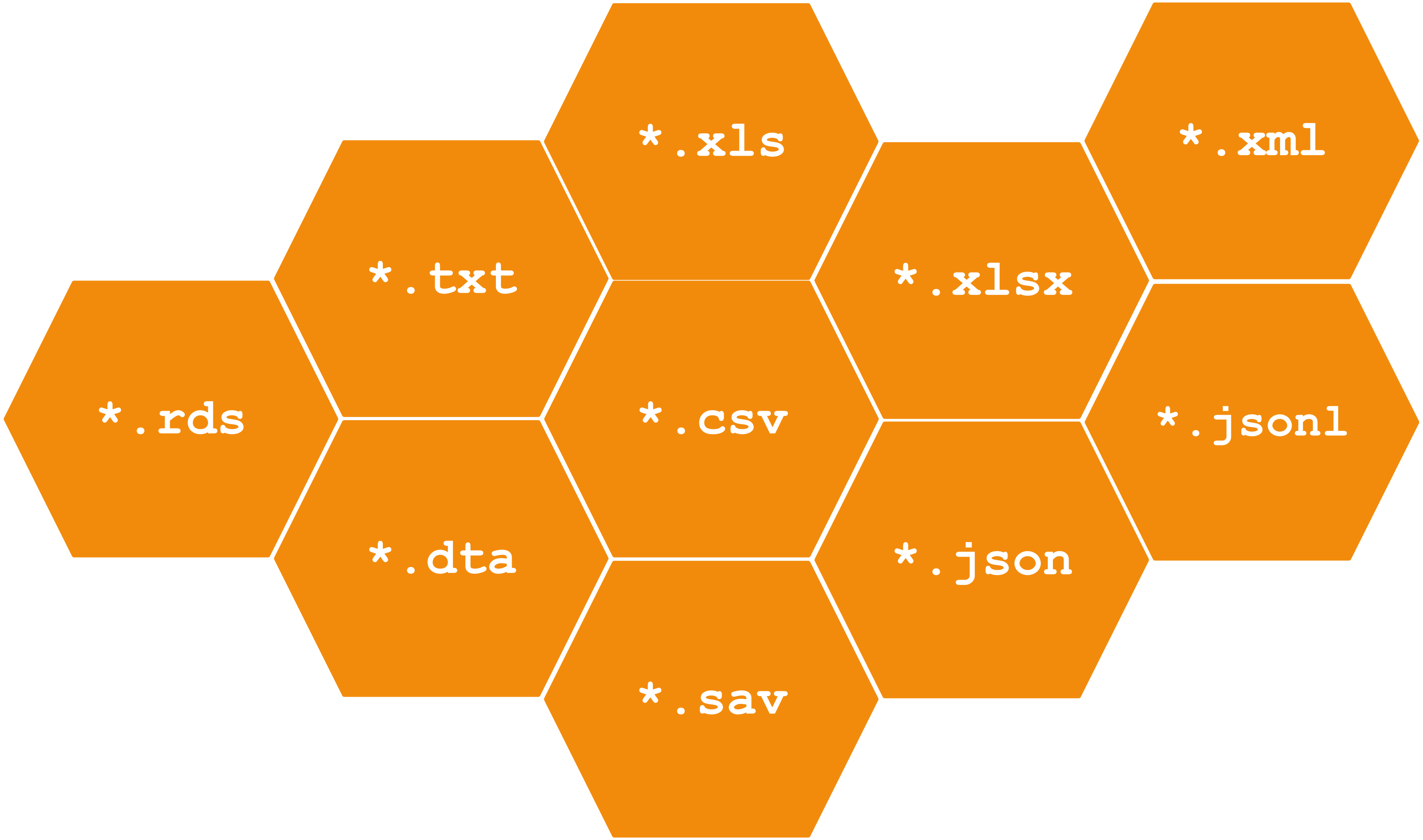
- Store all data into “**data**” folder →

data

- Save/Export analytical outputs →

output

Getting Data into R Environment



*.rds

*.txt

*.dta

*.xls

*.csv

*.sav

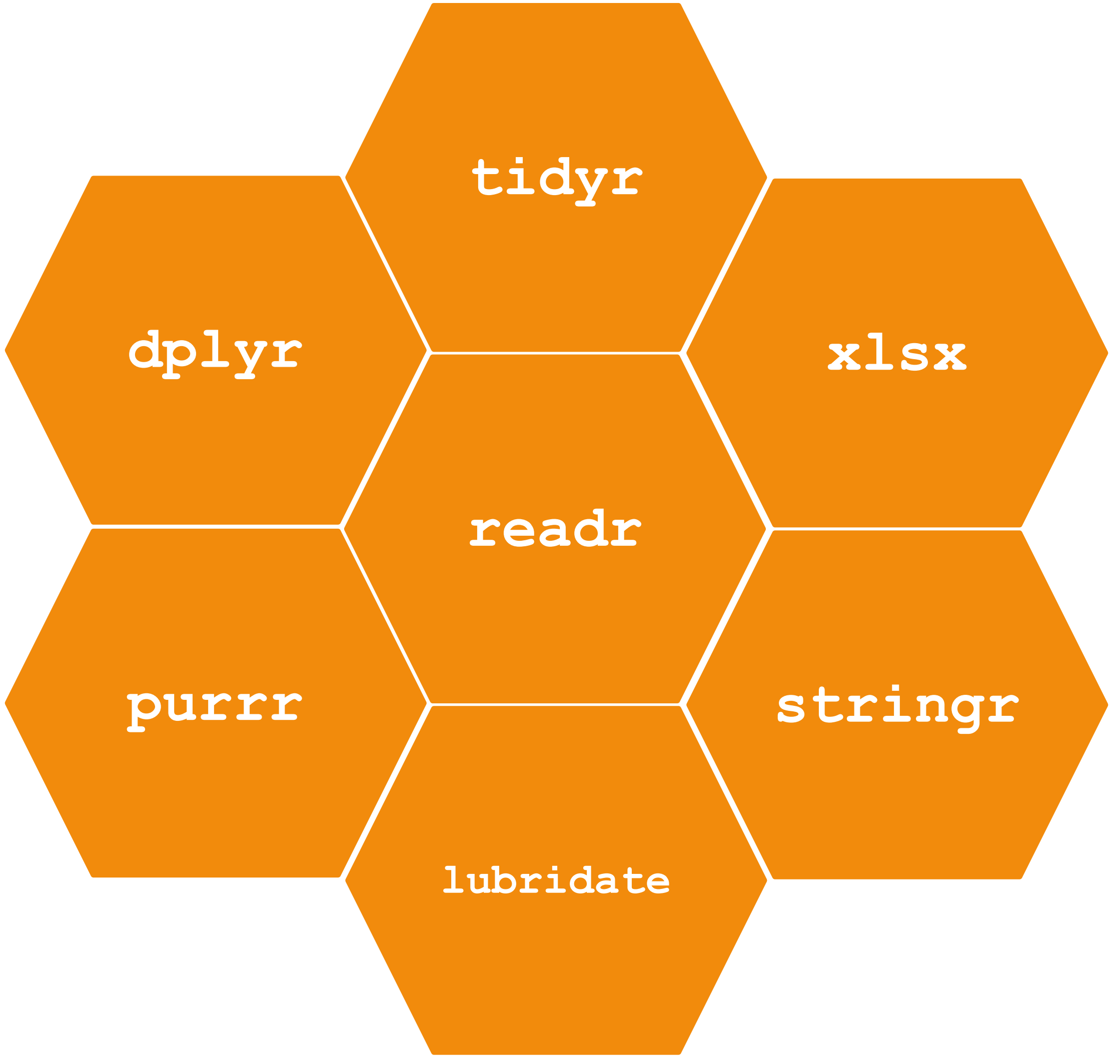
*.xlsx

*.json

*.xml

*.jsonl

R Libraries



tidyr

dplyr

xlsx

readr

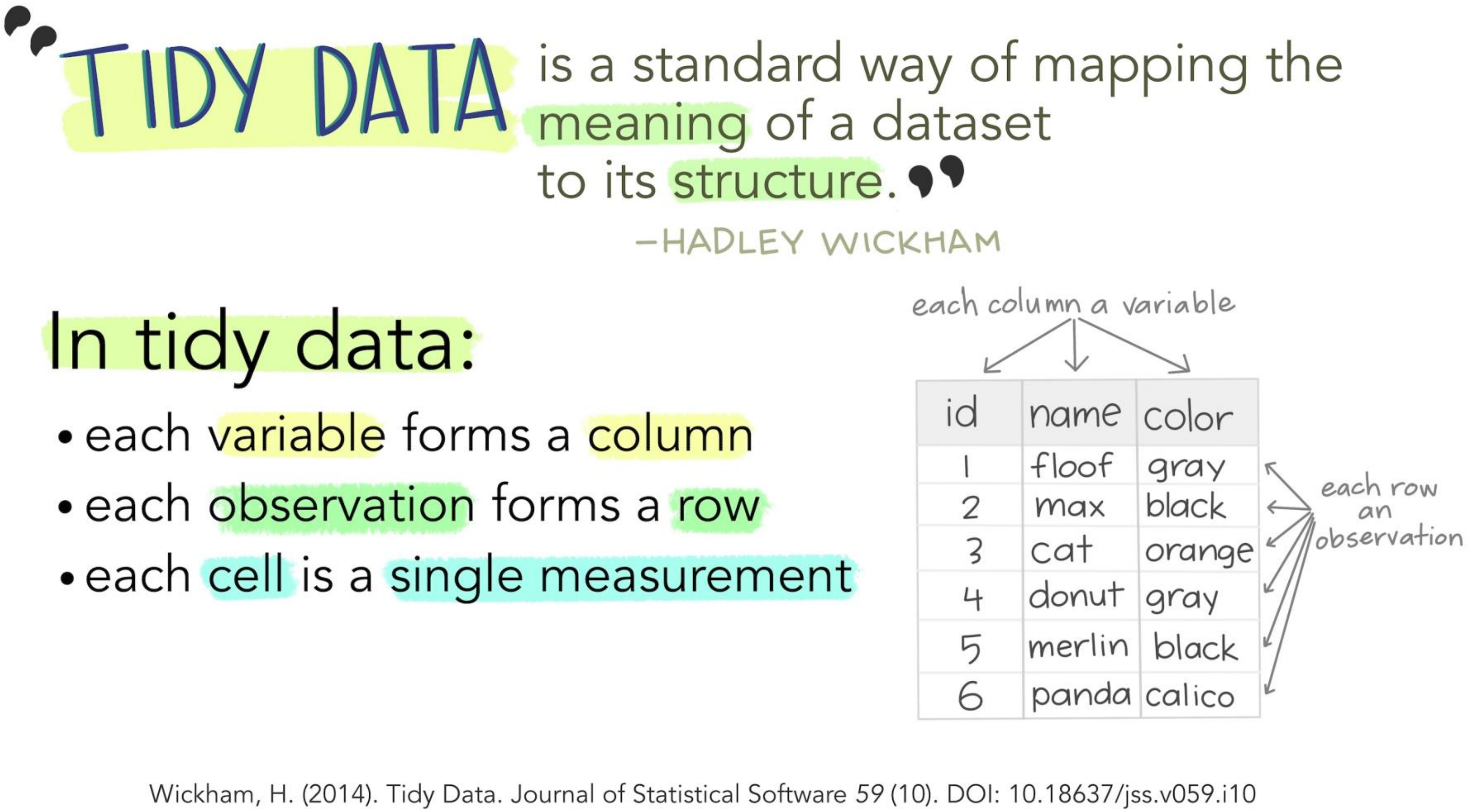
purrr

stringr

lubridate

These are the most **popular libraries** for data cleaning/processing

Tidy Data



> "TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."
>
> —HADLEY WICKHAM

In tidy data:

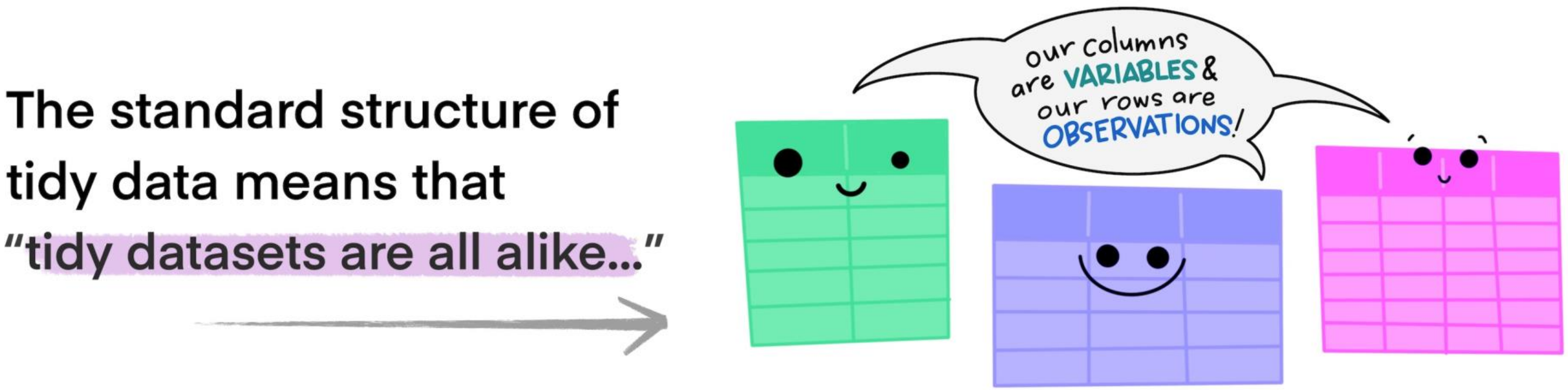
- each variable forms a column
- each observation forms a row
- each cell is a single measurement

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

Art: @allison_horst

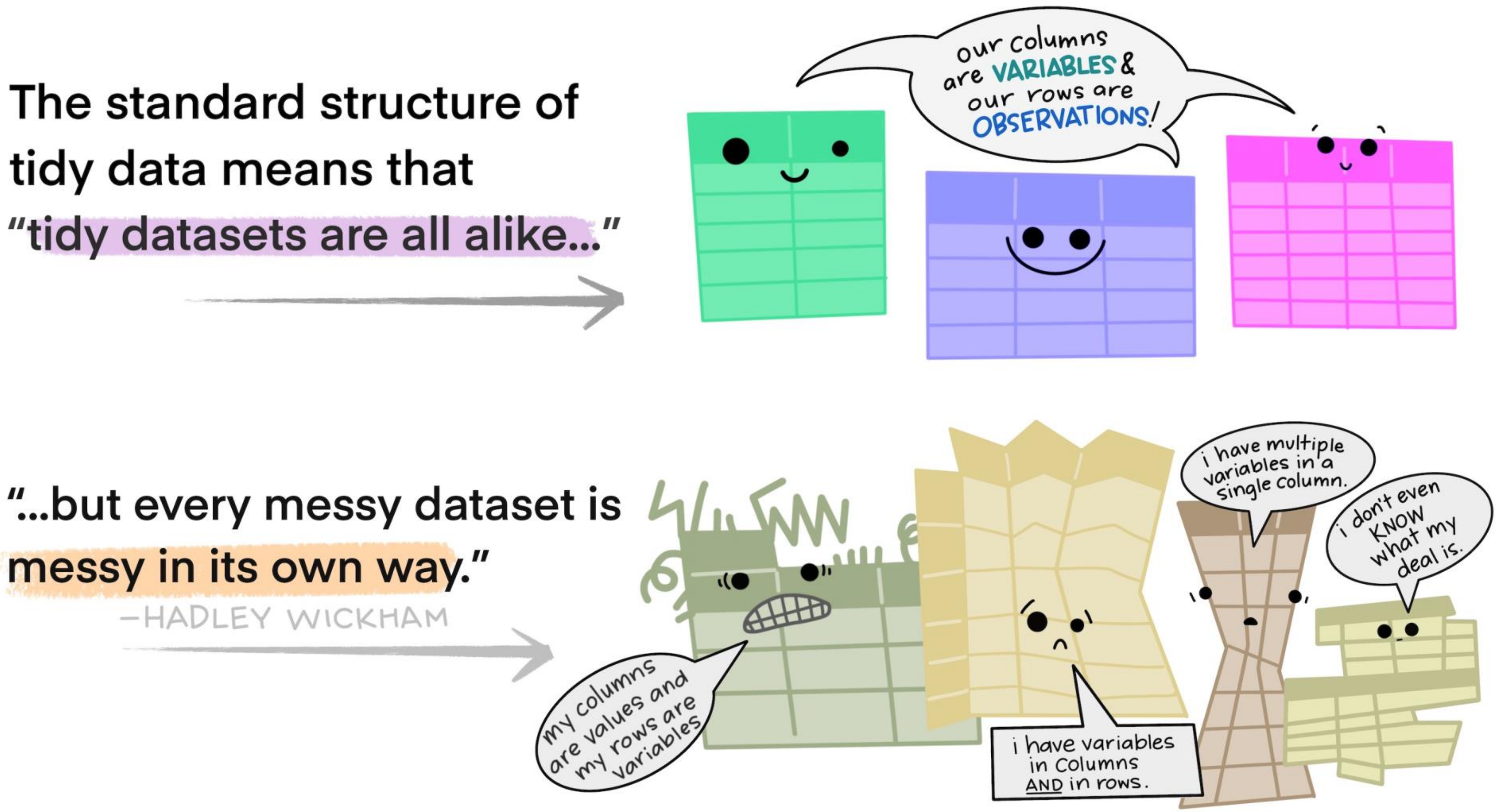
Tidy Data - Structure



The standard structure of tidy data means that "tidy datasets are all alike..."

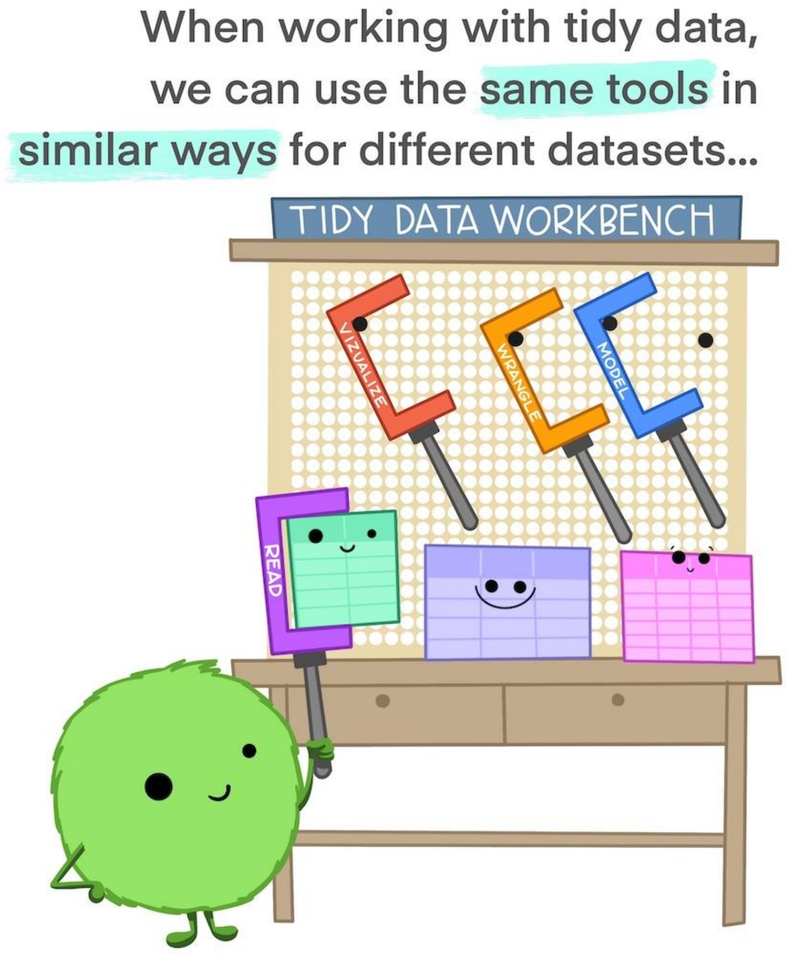
Art: @allison_horst

Tidy Data - Structure


The standard structure of tidy data means that "tidy datasets are all alike…"
our columns are VARIABLES & our rows are OBSERVATIONS!
"…but every messy dataset is messy in its own way."
—HADLEY WICKHAM
my columns are values and my rows are variables
i have variables in columns AND in rows.
i have multiple variables in a single column.
i don't even KNOW what my deal is.

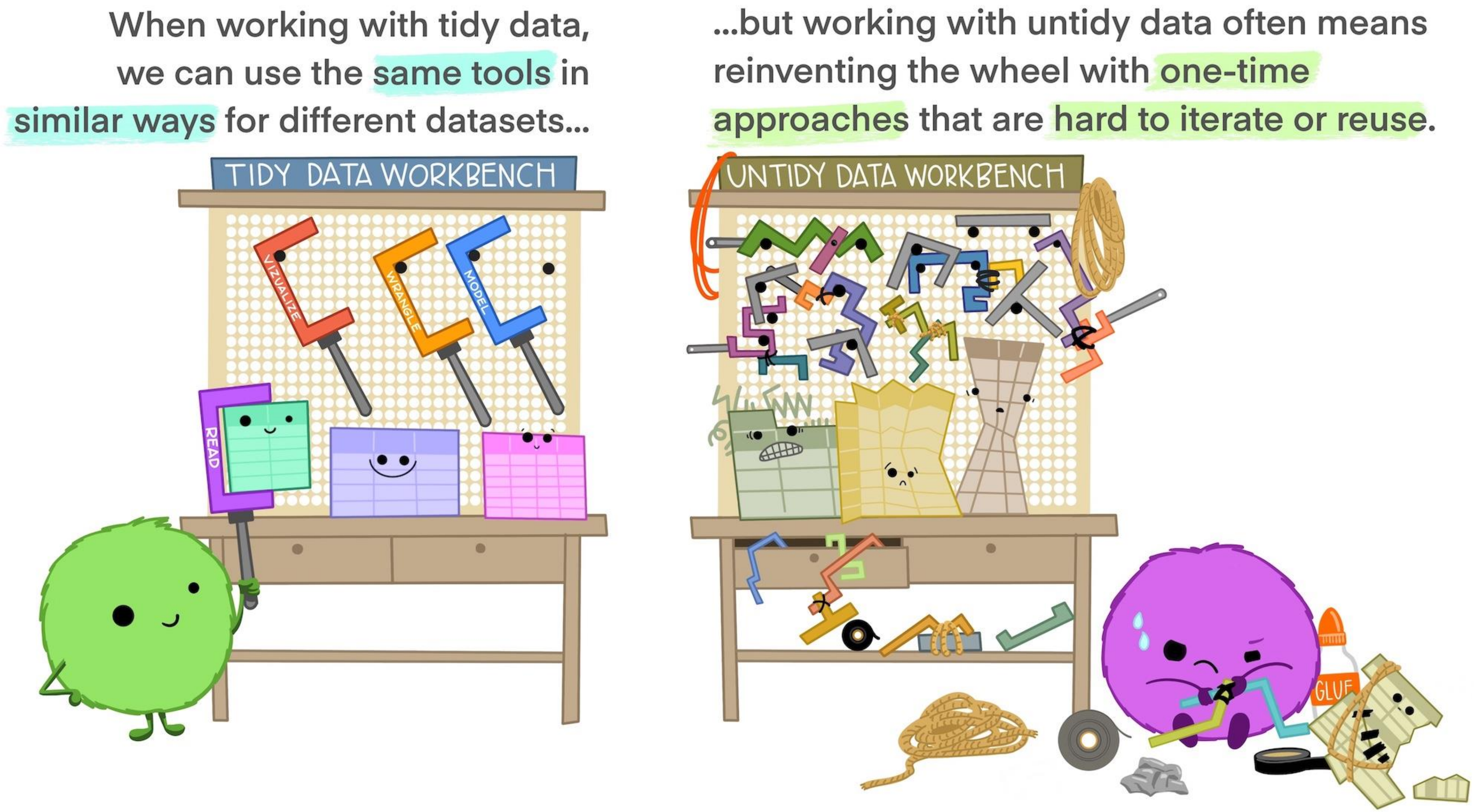
Art: @allison_horst

Working with Tidy Data


When working with tidy data, we can use the same tools in similar ways for different datasets...
TIDY DATA WORKBENCH
VIZUALIZE
WRANGLE
MODEL
READ

Art: @allison_horst

Working with Tidy Data

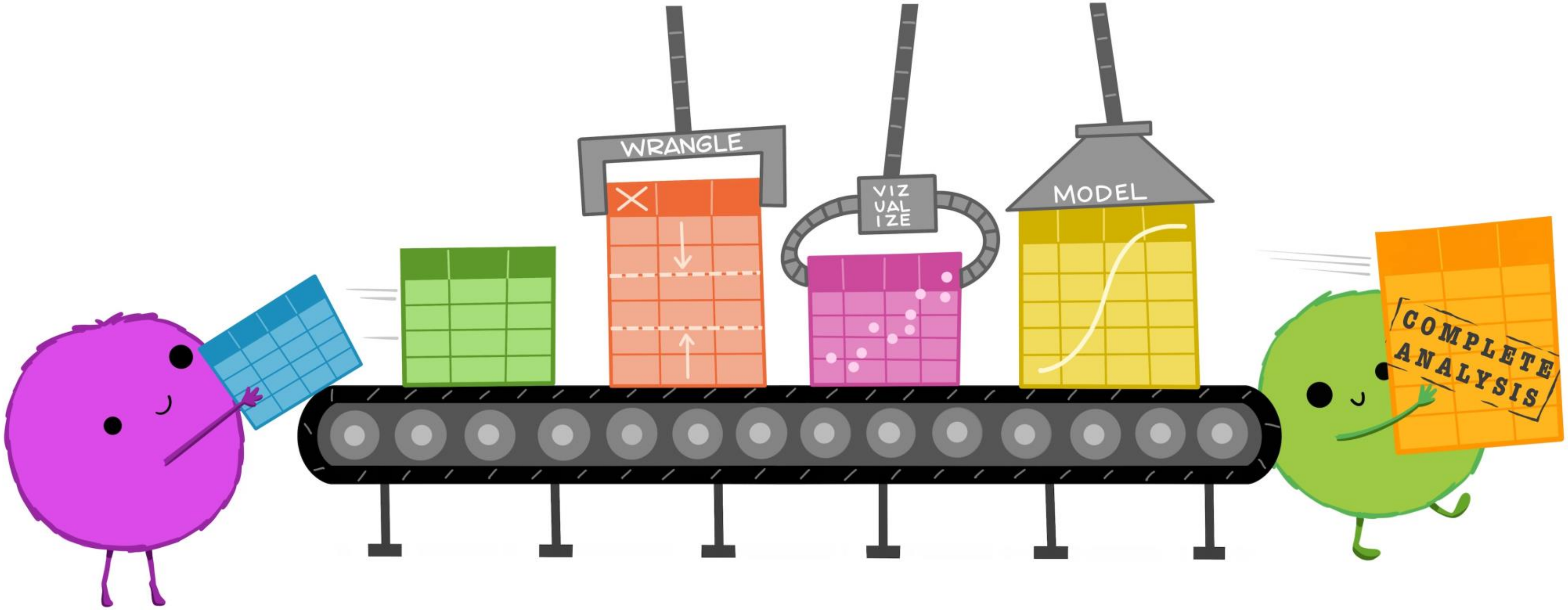


When working with tidy data, we can use the same tools in similar ways for different datasets...

...but working with untidy data often means reinventing the wheel with one-time approaches that are hard to iterate or reuse.

Art: @allison_horst

Working with Tidy Data



Art: @allison_horst

Art: @allison_horst

Ensure Correct Scale of Measurement

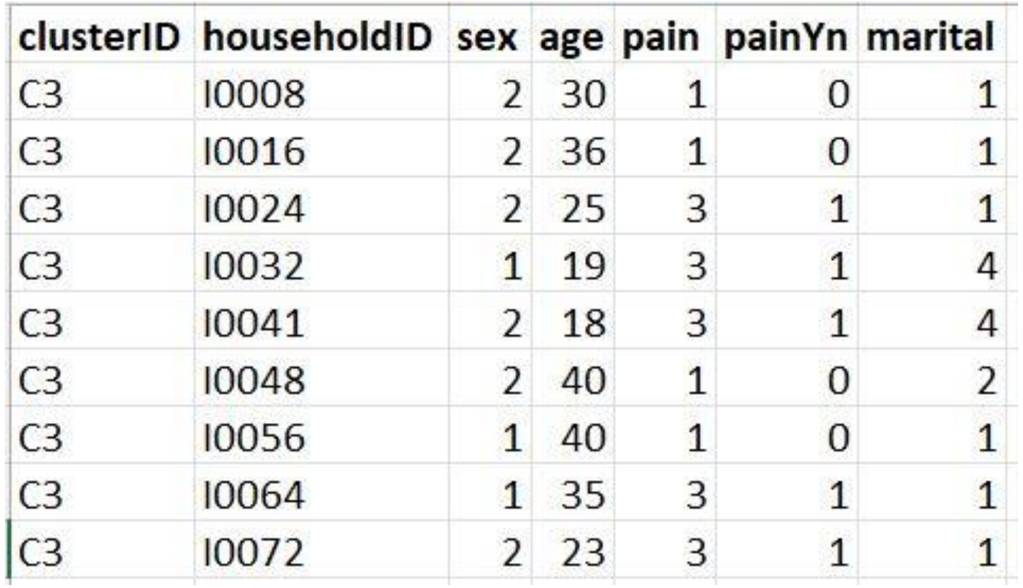
`*.csv`

`readr`

`dplyr`

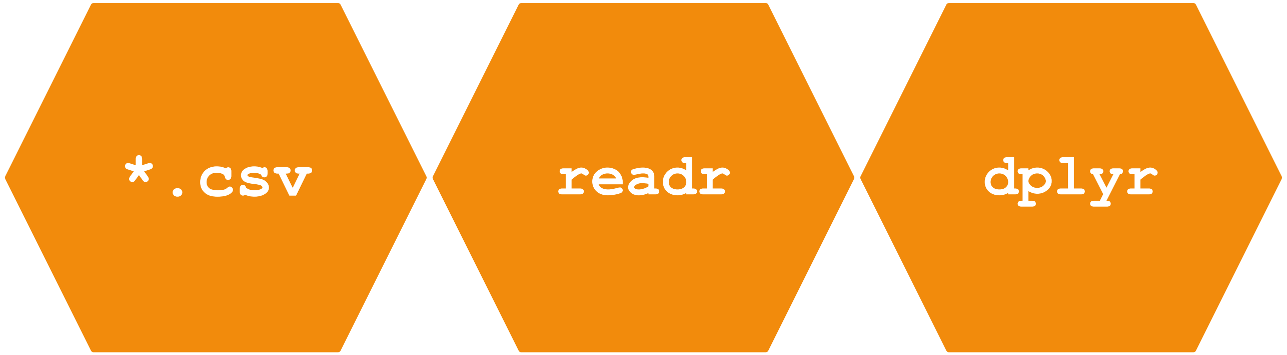
- We have a data into a spreadsheet

  `Data-1.csv`



clusterID	householdID	sex	age	pain	painYn	marital
C3	I0008	2	30	1	0	1
C3	I0016	2	36	1	0	1
C3	I0024	2	25	3	1	1
C3	I0032	1	19	3	1	4
C3	I0041	2	18	3	1	4
C3	I0048	2	40	1	0	2
C3	I0056	1	40	1	0	1
C3	I0064	1	35	3	1	1
C3	I0072	2	23	3	1	1

Ensure Correct Scale of Measurement

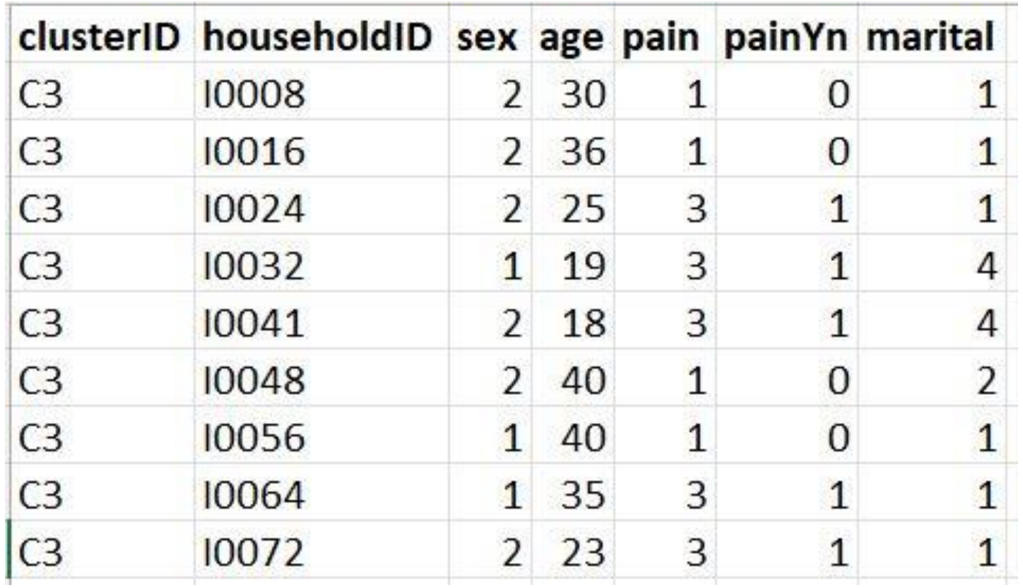


`*.csv`

`readr`

`dplyr`

- We have a data into a spreadsheet `Data-1.csv`
- How do we know **which column is in which measurement scale**?
- How do R programming will know the measurement scale of the columns?



clusterID	householdID	sex	age	pain	painYn	marital
C3	I0008	2	30	1	0	1
C3	I0016	2	36	1	0	1
C3	I0024	2	25	3	1	1
C3	I0032	1	19	3	1	4
C3	I0041	2	18	3	1	4
C3	I0048	2	40	1	0	2
C3	I0056	1	40	1	0	1
C3	I0064	1	35	3	1	1
C3	I0072	2	23	3	1	1

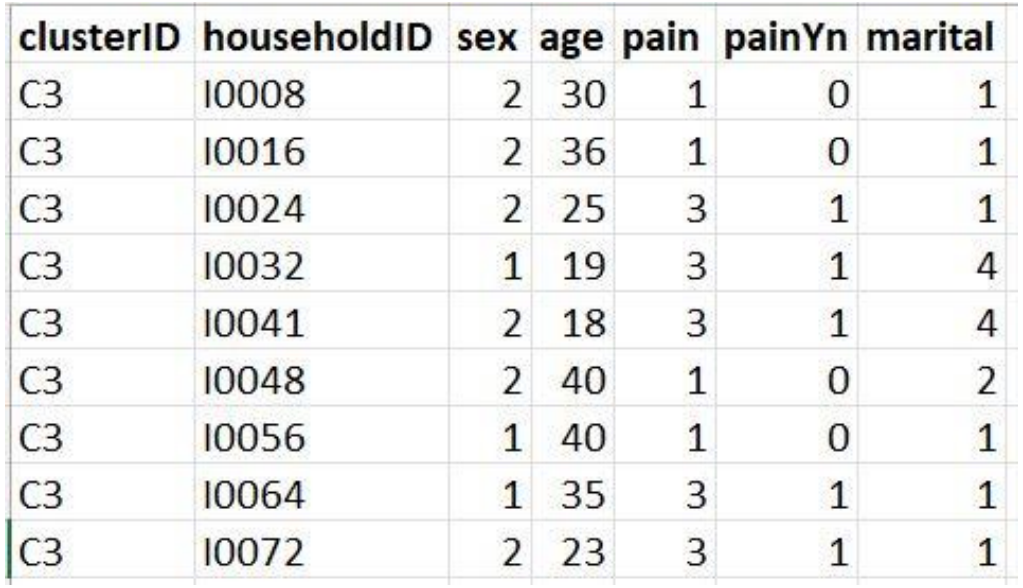
Ensure Correct Scale of Measurement

*.csv

readr

dplyr

- We have a data into a spreadsheet `Data-1.csv`
- How do we know **which column is in which measurement scale**?
- How do R programming will know the measurement scale of the columns?



clusterID	householdID	sex	age	pain	painYn	marital
C3	I0008	2	30	1	0	1
C3	I0016	2	36	1	0	1
C3	I0024	2	25	3	1	1
C3	I0032	1	19	3	1	4
C3	I0041	2	18	3	1	4
C3	I0048	2	40	1	0	2
C3	I0056	1	40	1	0	1
C3	I0064	1	35	3	1	1
C3	I0072	2	23	3	1	1

We need a codebook (data dictionary)

Ensure Correct Scale of Measurement

Name of Variable	Variable Label	Possible Values	Value Label (if any)
clusterID	Cluster ID	C1	
householdID	Household ID	I0001	
sex	Sex of respondent	1 or 2	1 = Male, 2 = Female
age	Age of respondent (in years)	25	
pain	Knee pain level	1, 2, or 3	1 = No Pain 2 = Mild Pain 3 = Severe Pain
painYn	Knee pain status	0, 1	
marital	Marital status	1, 2, 3 or 4	1 = Married 2 = Divorced 3 = Widow(er) 4 = Never Married

The definition of each variables

Key Variable

Name of Variable	Variable Label	Possible Values	Value Label (if any)
clusterID	Cluster ID	C1	
householdID	Household ID	I0001	
sex	Sex of respondent	1 or 2	1 = Male, 2 = Female
age	Age of respondent (in years)	25	
pain	Knee pain level	1, 2, or 3	1 = No Pain 2 = Mild Pain 3 = Severe Pain
painYn	Knee pain status	0, 1	
marital	Marital status	1, 2, 3 or 4	1 = Married 2 = Divorced 3 = Widow(er) 4 = Never Married

- The Key Variable (ID) variable(s) are the most important information we seek as an analyst / data scientist
- Each row should be **uniquely identifiable** based on one or more variable

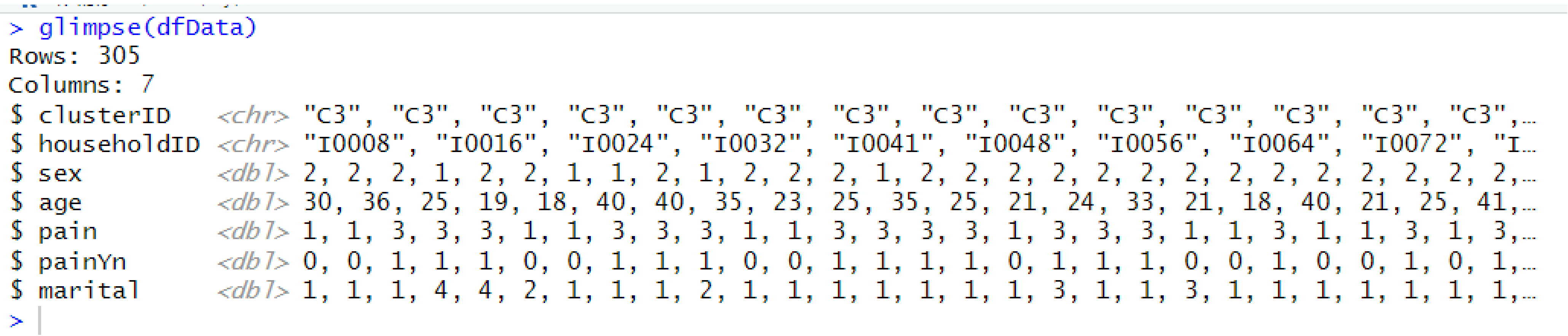
Ensure Correct Scale of Measurement

- We will use `read_csv()` from `readr` library
- R code to import `Data-1.csv` file

```
library(readr)
library(dplyr)
dfData1 <- read_csv(
  file = "Data-1.csv"
)
```

Ensure Correct Scale of Measurement

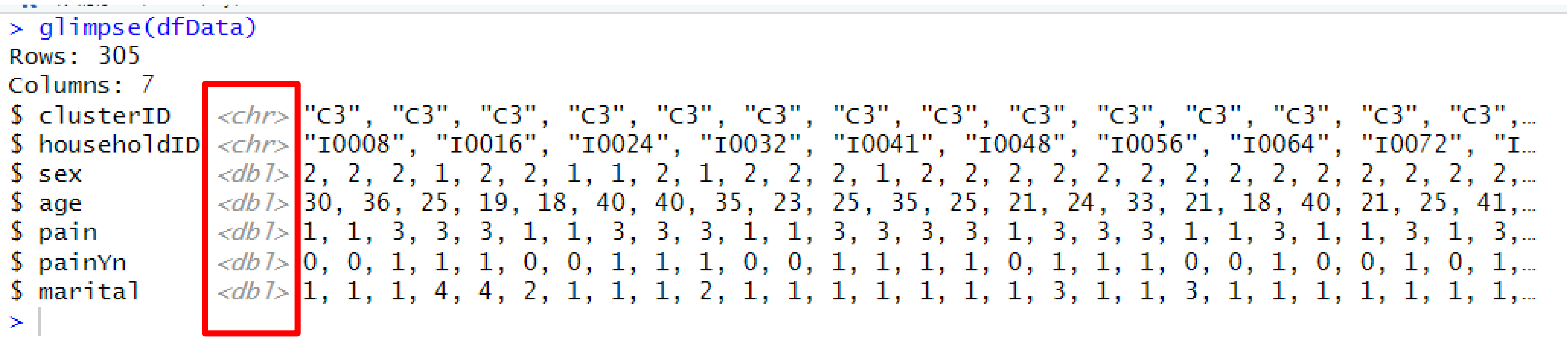
- After importing data, check the properties of variables using `glimpse()` function



```
> glimpse(dfData)
Rows: 305
Columns: 7
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3",…
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0064", "I0072", "I…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,…
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18, 40, 21, 25, 41,…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, 3, 1, 1, 3, 1, 3,…
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,…
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1,…
> 
```

Ensure Correct Scale of Measurement

- After importing data, check the properties of variables using `glimpse()` function



```
> glimpse(dfData)
Rows: 305
Columns: 7
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3",…
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0064", "I0072", "I…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,…
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18, 40, 21, 25, 41,…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, 3, 1, 1, 3, 1, 3,…
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,…
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1,…
> 
```

`<dbl> = Numeric`

How do we know which variable is in which measurement scale?

Ensure Correct Scale of Measurement

- The following table shows the **links between measurement scales and R data types**

Measurement Scale	R Data Types	R Function
`Nominal`	Character or Factor	`factor() or as.factor()`
`Ordinal`	Ordered Factor	`factor(ordered = TRUE) or as.factor(ordered = TRUE)`
`Interval`	Numeric	`as.numeric()`
`Ratio`	Numeric	`as.numeric()`
	Logical (TRUE or FALSE)	**`as.logical()`**

Now let’s define appropriate measurement scale for our data within R

Nominal (Factor) Variable

- The following code will create a new variable `sexNominal` that will represent correct scale (nominal)

```
dfData1 <- dfData1 %>%
  mutate(
    sexNominal = factor(
      x = sex,
      levels = c(1,2),
      labels = c("Male", "Female")
    )
  )
```

Nominal (Factor) Variable

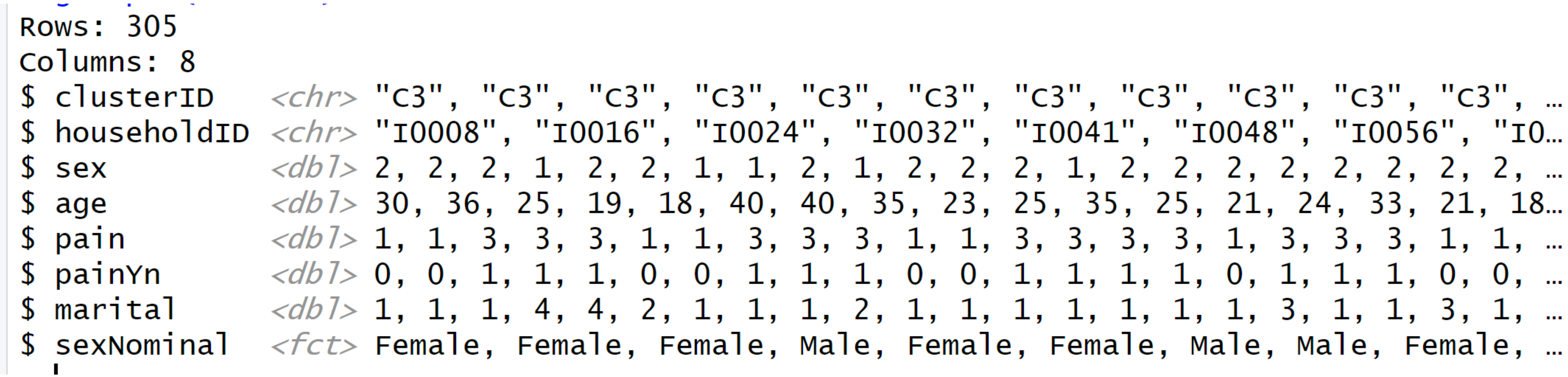
- The following code will create a new variable `sexNominal` that will represent correct scale (nominal)

```
dfData1 <- dfData1 %>%
  mutate(
    sexNominal = factor(
      x = sex,
      levels = c(1,2),
      labels = c("Male", "Female")
    )
  )
```

- `mutate()` is a function under `dplyr` library that is being used to create new variable
- `factor()` is a function to convert a variable into either nominal or ordinal scale

Nominal (Factor) Variable

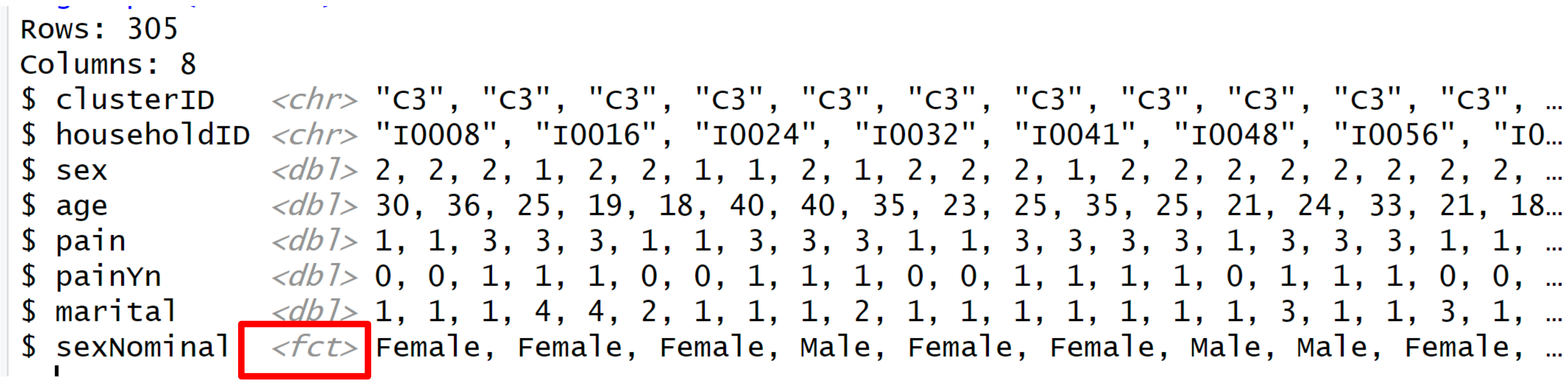
- Now look at the properties of `sexNominal` variable



```
Rows: 305
Columns: 8
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", …
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, …
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, …
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, …
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, …
$ sexNominal  <fct> Female, Female, Female, Male, Female, Female, Male, Male, Female, …
```

Nominal (Factor) Variable

- Now look at the properties of `sexNominal` variable



```
Rows: 305
Columns: 8
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", …
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, …
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, …
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, …
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, …
$ sexNominal  <fct> Female, Female, Female, Male, Female, Female, Male, Male, Female, …
```

`<fct> = Factor`

Ordinal (Ordered Factor) Variable

- Let’s create an ordinal variable from our original `pain` variable

```
dfData1 <- dfData1 %>%
  mutate(
    painOrdinal = factor(
      x = pain,
      levels = c(1,2, 3),
      labels = c("No Pain", "Mild Pain", "Severe Pain"),
      ordered = TRUE
    )
  )
```

Ordinal (Ordered Factor) Variable

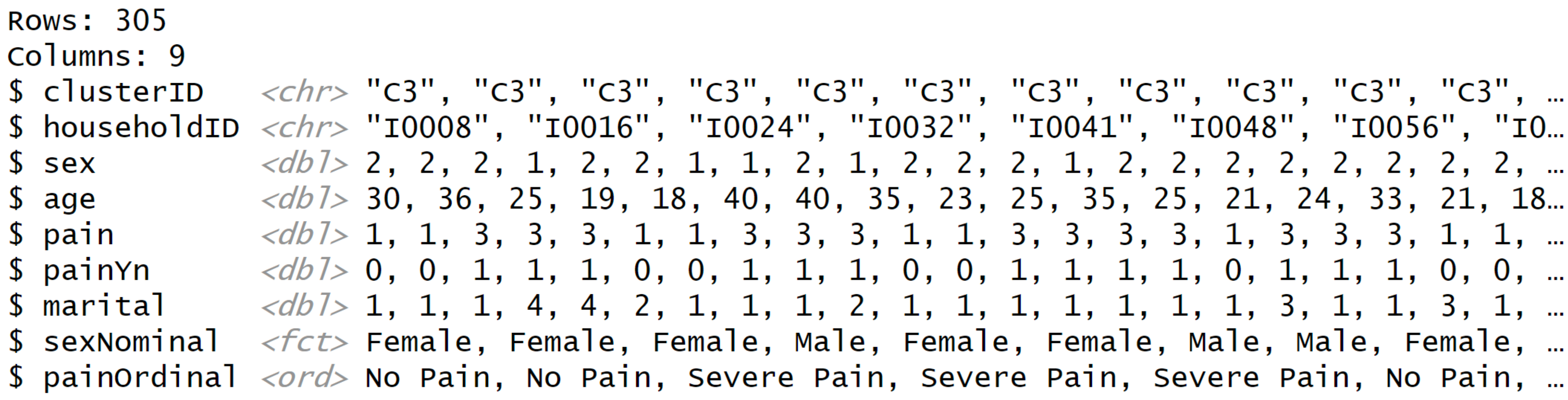
- Let’s create an ordinal variable from our original `pain` variable

```
dfData1 <- dfData1 %>%
 mutate(
  painOrdinal = factor(
   x = pain,
   levels = c(1,2, 3),
   labels = c("No Pain", "Mild Pain", "Severe Pain"),
   ordered = TRUE
  )
 )
```

`Ordered = TRUE` to make sure it is an ordinal variable

Ordinal (Ordered Factor) Variable

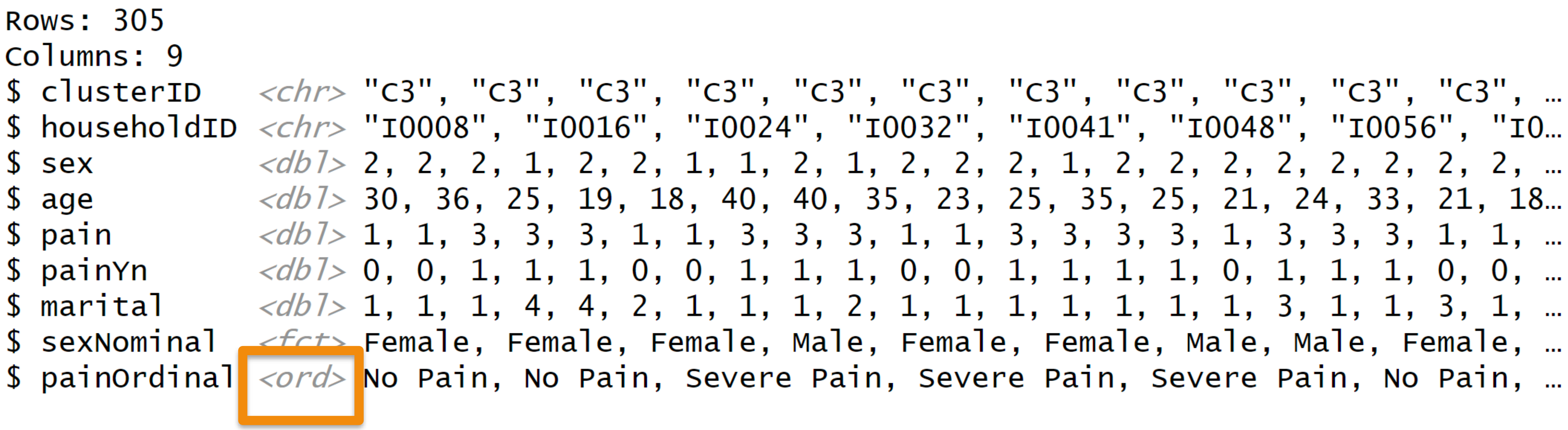
- Now look at the properties of `painOrdinal` variable



```
Rows: 305
Columns: 9
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", …
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, …
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, …
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, …
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, …
$ sexNominal  <fct> Female, Female, Female, Male, Female, Female, Male, Male, Female, …
$ painOrdinal <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No Pain, …
```

Ordinal (Ordered Factor) Variable

- Now look at the properties of `painOrdinal` variable



```
Rows: 305
Columns: 9
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", …
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, …
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, …
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, …
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, …
$ sexNominal  <fct> Female, Female, Female, Male, Female, Female, Male, Male, Female, …
$ painOrdinal <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No Pain, …
```

`<ord> = Ordinal`

Binary (Logical) Variable

- A variable that only contain 0 and 1, can be represented a logical data type in R
- `painYn` variable in the data can be represented as a logical variable as:

```
dfData1 <- dfData1 %>%
  mutate(
    painYn = as.logical(x=painYn)
  )
```

Binary (Logical) Variable

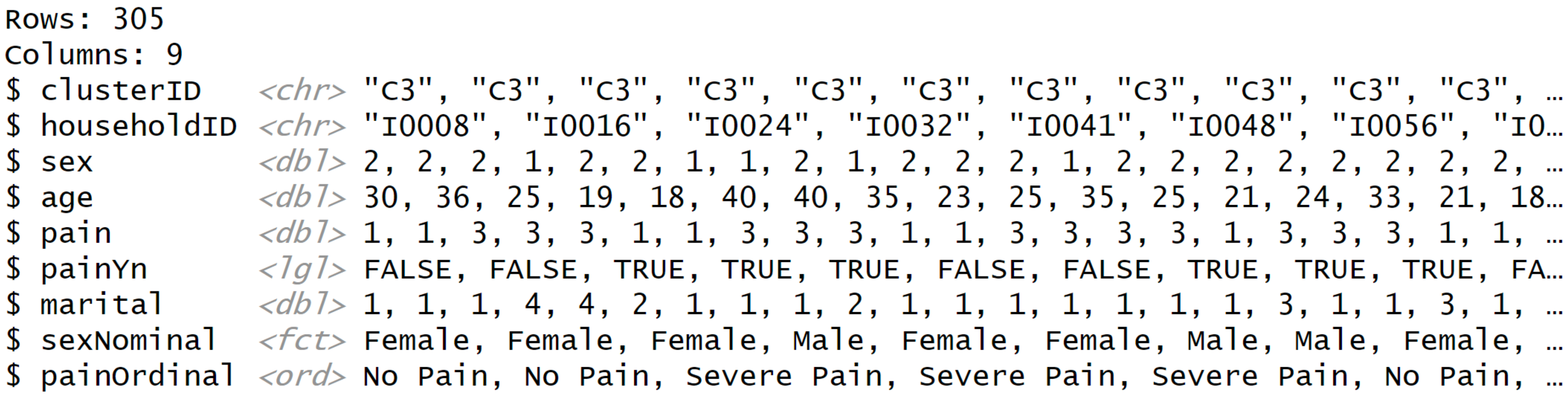
- A variable that only contain 0 and 1, can be represented a logical data type in R
- `painYn` variable in the data can be represented as a logical variable as:

```
dfData1 <- dfData1 %>%
  mutate(
    painYn = as.logical(x=painYn)
  )
```

`as.logical() is` to create a binary variable which is not a nominal variable

Binary (Logical) Variable

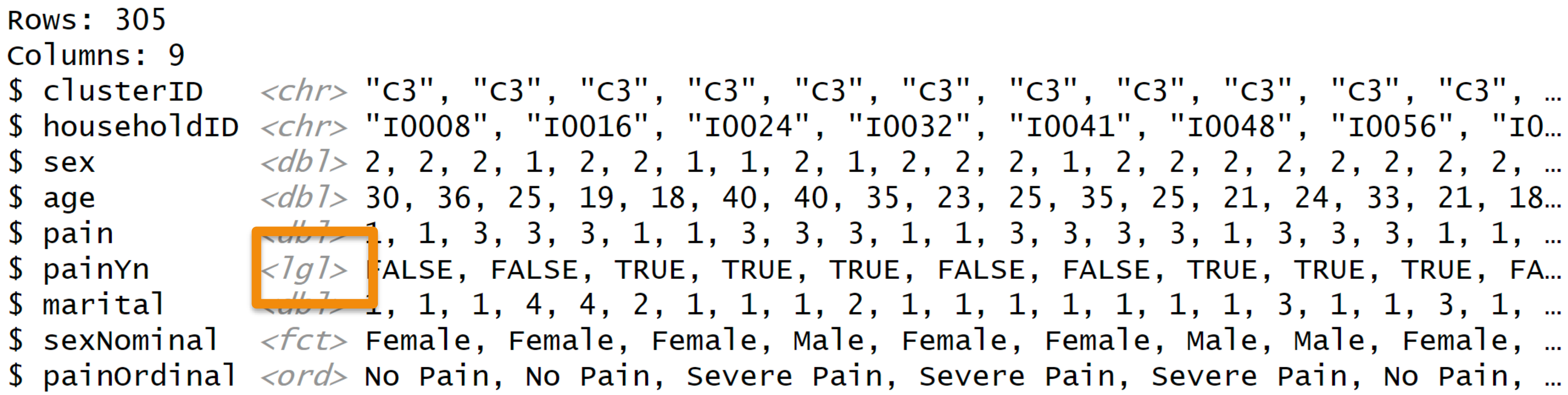
- Variable Properties for `painYn`



```
Rows: 305
Columns: 9
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", …
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0…
$ sex         <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, …
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18…
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, …
$ painYn      <lgl> FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, FA…
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, …
$ sexNominal  <fct> Female, Female, Female, Male, Female, Female, Male, Male, Female, …
$ painOrdinal <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No Pain, …
```

Binary (Logical) Variable

- Variable Properties for `painYn`

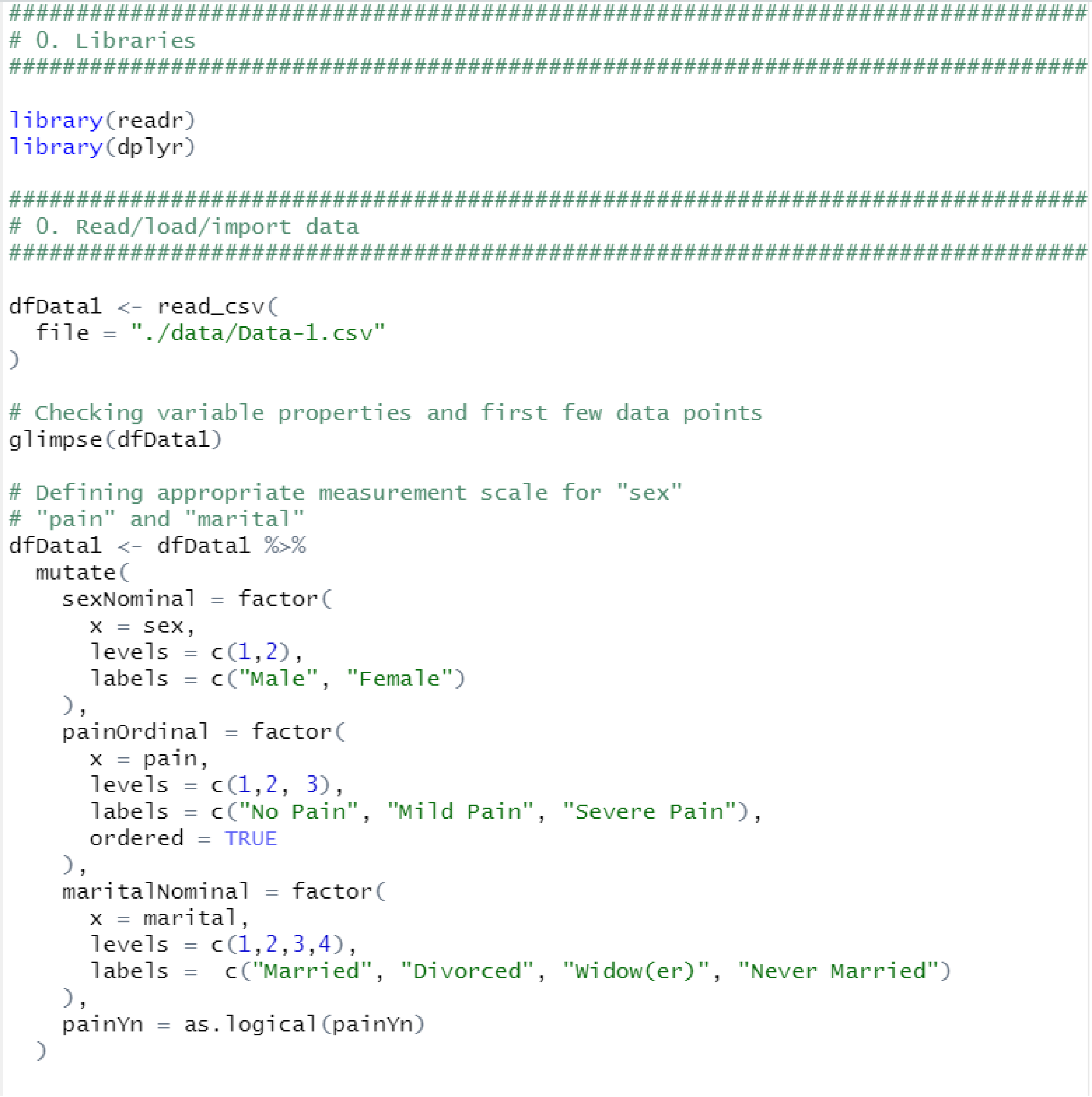


```
Rows: 305
Columns: 9
$ clusterID    <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", …
$ householdID  <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", "I0…
$ sex          <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, …
$ age          <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21, 18…
$ pain         <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, …
$ painYn       <lgl> FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, FA…
$ marital      <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, …
$ sexNominal   <fct> Female, Female, Female, Male, Female, Female, Male, Male, Female, …
$ painOrdinal  <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No Pain, …
```

`<lgl> = Logical` (Binary variable)

```
dfData1 <- dfData1 %>%
  mutate(
    maritalNominal = factor(
      x = marital,
      levels = c(1,2,3,4),
      labels = c("Married", "Divorced", "Widow(er)", "Never Married")
    )
  )
```

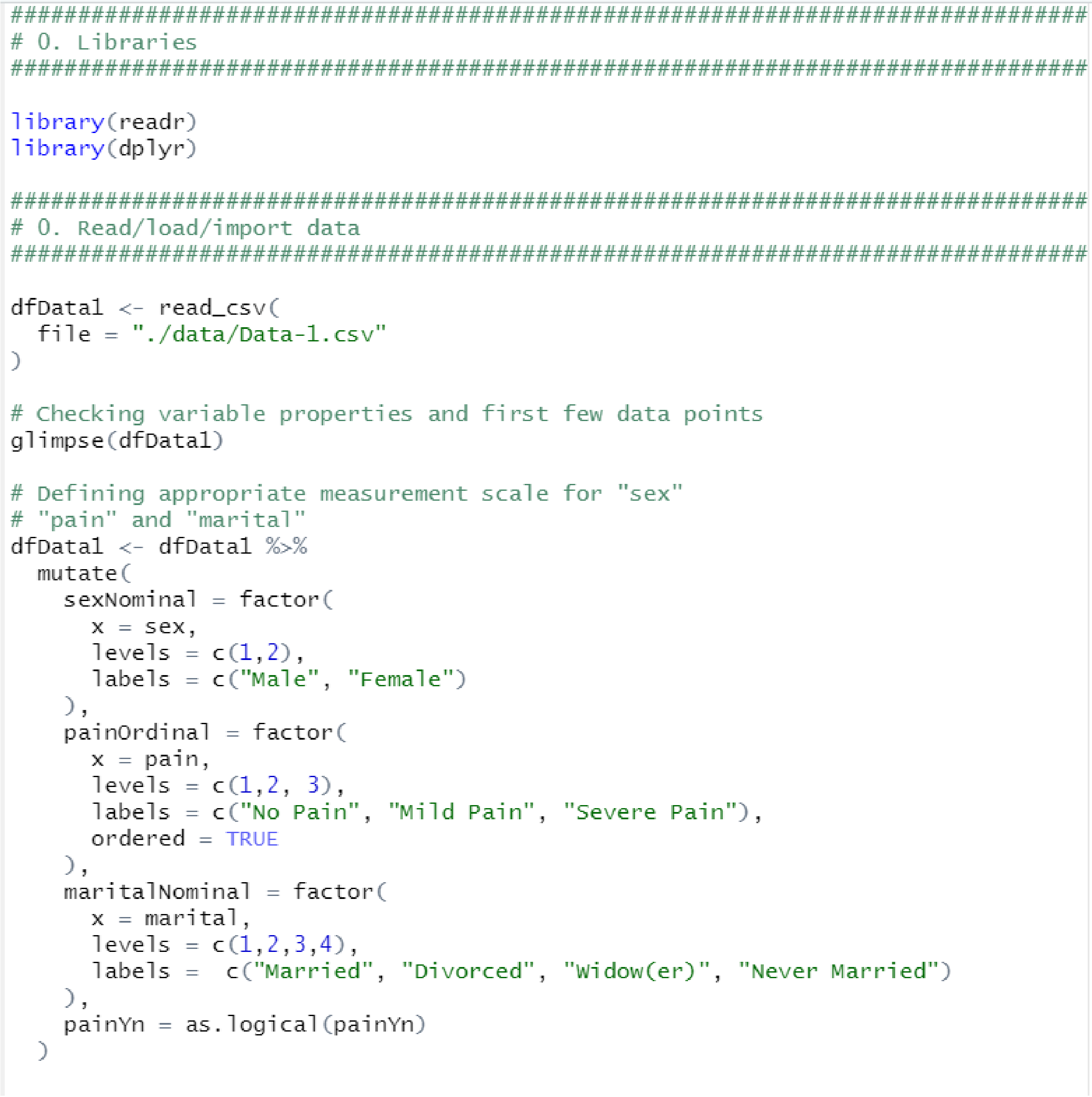
R Script (*.R)



```r
################################################################################
# 0. Libraries
################################################################################

library(readr)
library(dplyr)

################################################################################
# 0. Read/load/import data
################################################################################

dfData1 <- read_csv(
  file = "./data/Data-1.csv"
)

# Checking variable properties and first few data points
glimpse(dfData1)

# Defining appropriate measurement scale for "sex"
# "pain" and "marital"
dfData1 <- dfData1 %>%
  mutate(
    sexNominal = factor(
      x = sex,
      levels = c(1,2),
      labels = c("Male", "Female")
    ),
    painOrdinal = factor(
      x = pain,
      levels = c(1,2, 3),
      labels = c("No Pain", "Mild Pain", "Severe Pain"),
      ordered = TRUE
    ),
    maritalNominal = factor(
      x = marital,
      levels = c(1,2,3,4),
      labels =  c("Married", "Divorced", "Widow(er)", "Never Married")
    ),
    painYn = as.logical(painYn)
  )
```

R Script (`*.R`)

To reproduce the same results or share the analysis with collaborator, we need to save the R code into a file; we call it **script** file (or simply R script)



```
################################################################################
# 0. Libraries
################################################################################

library(readr)
library(dplyr)

################################################################################
# 0. Read/load/import data
################################################################################

dfData1 <- read_csv(
  file = "./data/Data-1.csv"
)

# Checking variable properties and first few data points
glimpse(dfData1)

# Defining appropriate measurement scale for "sex"
# "pain" and "marital"
dfData1 <- dfData1 %>%
  mutate(
    sexNominal = factor(
      x = sex,
      levels = c(1,2),
      labels = c("Male", "Female")
    ),
    painOrdinal = factor(
      x = pain,
      levels = c(1,2, 3),
      labels = c("No Pain", "Mild Pain", "Severe Pain"),
      ordered = TRUE
    ),
    maritalNominal = factor(
      x = marital,
      levels = c(1,2,3,4),
      labels =  c("Married", "Divorced", "Widow(er)", "Never Married")
    ),
    painYn = as.logical(painYn)
  )
```

- Library `dplyr`
- Verb `filter()`
- Conditional operator

```
NewOrExistingData <- ExistingData %>%
  filter(
    VariableName1==value,
    VariableName2=="value-text"
  )
```

Data Processing: Subset Rows

- Library `dplyr`
- Verb `select()`

```
NewData <- ExistingData %>%
  select(
    Variable1, Variable2, Variable2
  )
```

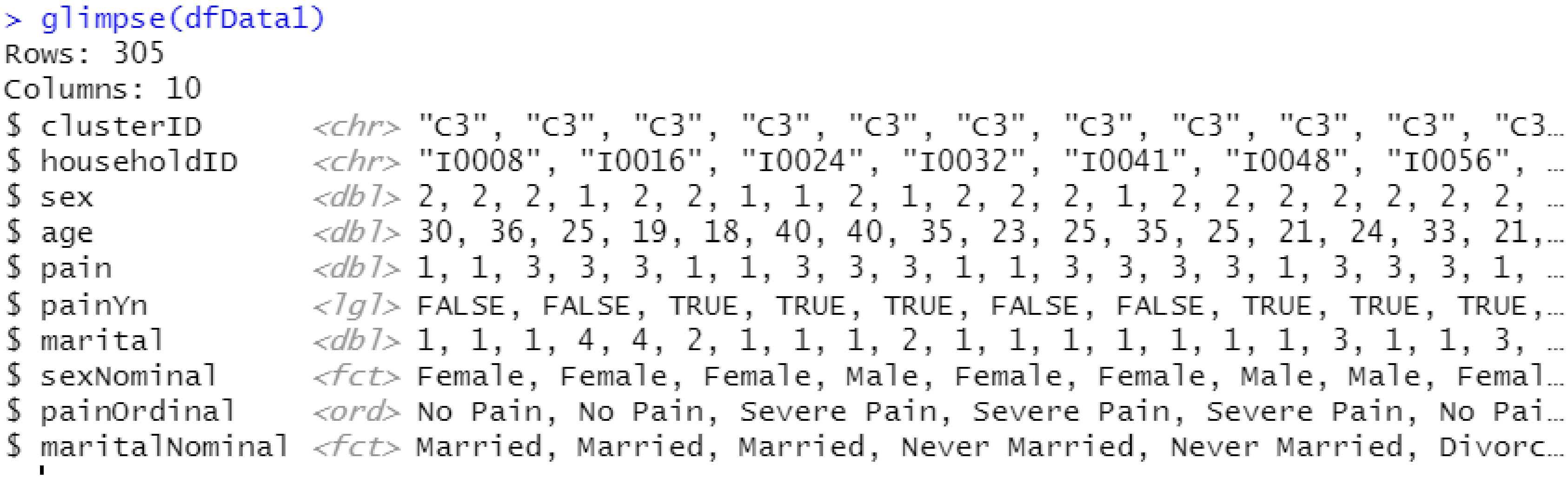
Data Processing: Subset columns

- Library `dplyr`
- Verb `mutate()`

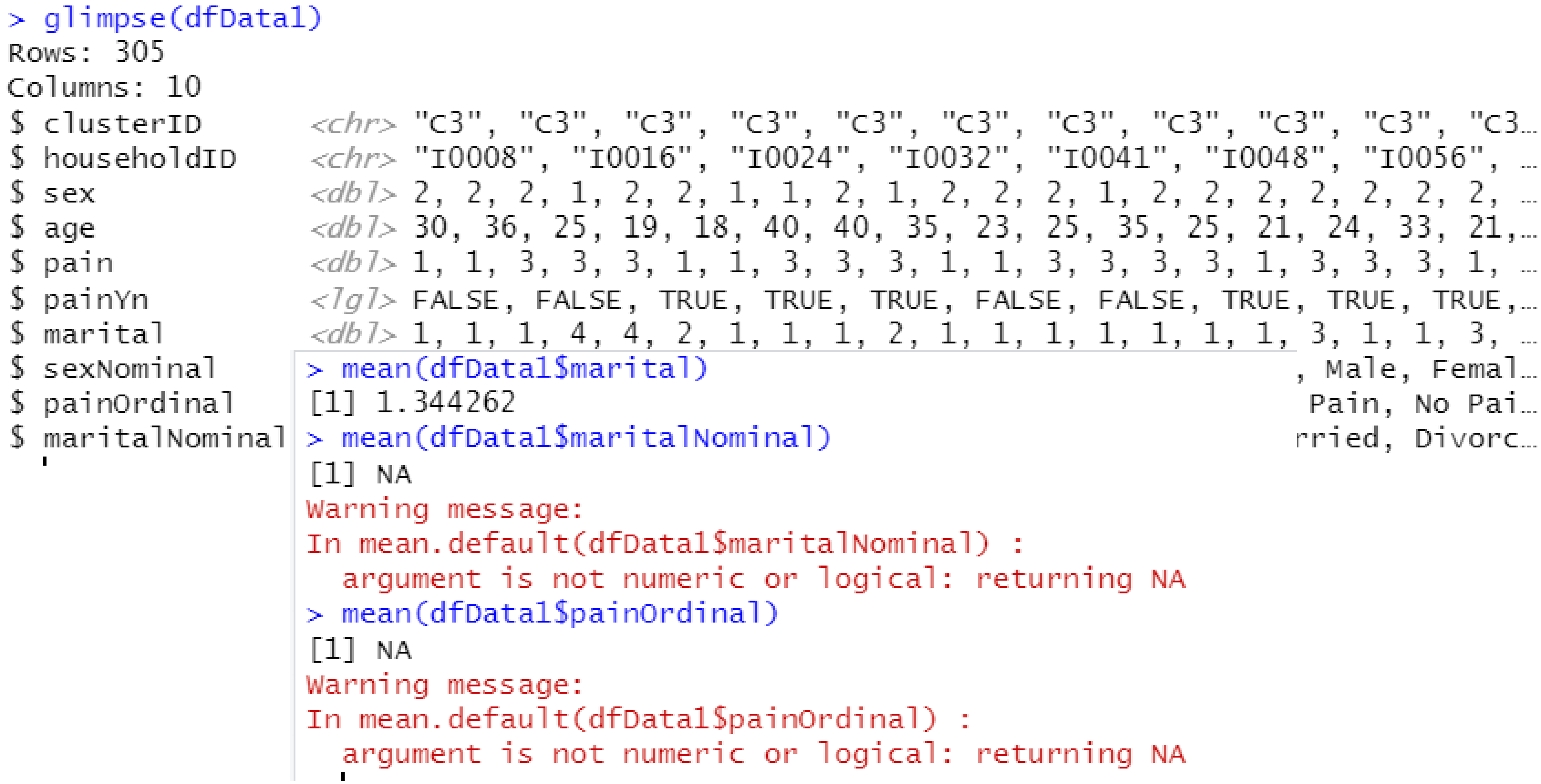
```
ExistingData <- ExistingData %>%
  mutate(
    NewVariable = OldVariable1 + OldVariable2
  )
```

Data Processing: Creating New Variable

What If we ignore scale of measurement



```
> glimpse(dfData1)
Rows: 305
Columns: 10
$ clusterID      <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3…
$ householdID    <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", …
$ sex            <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, …
$ age            <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21,…
$ pain           <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, …
$ painYn         <lgl> FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE,…
$ marital        <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, …
$ sexNominal     <fct> Female, Female, Female, Male, Female, Female, Male, Male, Femal…
$ painOrdinal    <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No Pai…
$ maritalNominal <fct> Married, Married, Married, Never Married, Never Married, Divorc…
```



```
> glimpse(dfData1)
Rows: 305
Columns: 10
$ clusterID      <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3…
$ householdID    <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056", …
$ sex            <dbl> 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, …
$ age            <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, 21,…
$ pain           <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, 1, …
$ painYn         <lgl> FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE,…
$ marital        <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, …
$ sexNominal                                                                  , Male, Femal…
$ painOrdinal                                                                Pain, No Pai…
$ maritalNominal                                                            rried, Divorc…
```

```
> mean(dfData1$marital)
[1] 1.344262
> mean(dfData1$maritalNominal)
[1] NA
Warning message:
In mean.default(dfData1$maritalNominal) :
  argument is not numeric or logical: returning NA
> mean(dfData1$painOrdinal)
[1] NA
Warning message:
In mean.default(dfData1$painOrdinal) :
  argument is not numeric or logical: returning NA
```