# Statistics for Public Health Research

Center for Data Research and Analytics

Session 5:

Data Preparation for Statistical Analysis Using R

1

## Content

- Recap on measurement scales
- Link between measurement scales and R data types
- Hands on demonstration



2

*"If a thing exists, it exists in some amount; and if it exists in some amount, it can be measured."*

- E. L. Thorndike (1914)

3

## Measurement Scales

| Math/ Logical | Nominal | Ordinal | Interval | Ratio |
|---|---|---|---|---|
| X ÷ | × | × | × | √ |
| + - | × | × | √ | √ |
| < > | × | √ | √ | √ |
| = ≠ | √ | √ | √ | √ |

Appropriate mathematical and logical operations for different measurement scales



4

## Measurement Scales Summary

- **Nominal:** It can only represent distinct categories and we can perform equality check
- **Ordinal:** It can represent distinct categories and we can arrange in ordered sequence but the difference between consecutive position is not meaningful
- **Interval:** Difference is meaningful, but multiplication is not meaningful, and there is an absence of absolute zero
- **Ratio:** Difference and multiplication is meaningful and there is an absolute zero



5

## Example Data-1

- We have a data into a spreadsheet `Example_data_session_4.csv` →
- How do we know which column is in which measurement scale?
- How do R programming will understand measurement scale of a column?

| 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)



6

## Data Dictionary of Example Data-1

- The definition of each variable is as follows:

| 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<br>2 = Mild Pain<br>3 = Severe Pain |
| painYn | Knee pain status | 0, 1 | |
| marital | Marital status | 1, 2, 3 or 4 | 1 = Married<br>2 = Divorced<br>3 = Widow(er)<br>4 = Never Married |



7

## Installing & Loading R Libraries

- **Installation**
  - Open R console
  - Connect the PC with internet
  - Type:
    `install.packages("pkgname",lib="location",dependencies=TRUE)`
  - `install.packages("readr", dependencies = TRUE)`
- **Loading**
  - `library(readr)`

**We may require to load certain libraries to do specific type of tasks**



8

## Importing Data into R Environment

- There are several ways to import a csv file into R
- We will use `read_csv()` from `readr` library
- Before writing any code into R, firstly set a working directory by pointing R to the folder where you have the data files

```
setwd("C:/Users/jayna/Documents/GitHub/Materials")
```

- R code to import `Example_data_session_4.csv` file

```
library(readr)
library(dplyr)
dfSession4a <- read_csv(
  file = "Example_data_session_4.csv"
)
```



9

## Checking Variable Properties

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

```
> glimpse(dfSession4a)
Rows: 305
Columns: 7
$ clusterID   <chr> "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, ~
$ age         <dbl> 30, 36, 25, 19, 10, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, ~
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, ~
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ~
>
```

Center for Data Research and Analytics 23 January 2022 10

10

## Checking Variable Properties

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

```
> glimpse(dfSession4a)
Rows: 305
Columns: 7
$ clusterID   <chr> "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, ~
$ age         <dbl> 30, 36, 25, 19, 10, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, ~
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, ~
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ~
>
```

`<dbl> = Numeric`

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

Center for Data Research and Analytics 23 January 2022 11

11

## Measurement Scales and R Data Types

- 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

Center for Data Research and Analytics 23 January 2022 12

12

## Nominal (Factor) Variable

- The following code will make sure `sex` variable is in nominal scale

```
dfSession4a <- dfSession4a %>%
  mutate(
    sex = 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



13

## Checking Variable Properties

- Now look at the properties of `sex` variable

```
Rows: 305
Columns: 7
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", ~
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056~
$ sex         <fct> Female, Female, Female, Male, Female, Female, Male, Male, Fe~
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, ~
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, ~
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ~
```



14

## Checking Variable Properties

- Now look at the properties of `sex` variable

```
Rows: 305
Columns: 7
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", ~
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056~
$ sex         <fct> Female, Female, Female, Male, Female, Female, Male, Male, Fe~
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, ~
$ pain        <dbl> 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 3, ~
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ~
```

`<fct> = Factor`



15

## Ordinal (Ordered Factor) Variable

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

```
dfSession4a <- dfSession4a %>%
  mutate(
    pain = 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



16

## Checking Variable Properties

- Let's check `pain` variable now using `glimpse()`

```
> glimpse(dfSession4a)
Rows: 305
Columns: 7
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", ~
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056~
$ sex         <fct> Female, Female, Female, Male, Female, Female, Male, Male, Fe~
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, ~
$ pain        <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No ~
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 1, ~
```



17

## Checking Variable Properties

- Let's check `pain` variable now using `glimpse()`

```
> glimpse(dfSession4a)
Rows: 305
Columns: 7
$ clusterID   <chr> "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", "C3", ~
$ householdID <chr> "I0008", "I0016", "I0024", "I0032", "I0041", "I0048", "I0056~
$ sex         <fct> Female, Female, Female, Male, Female, Female, Male, Male, Fe~
$ age         <dbl> 30, 36, 25, 19, 18, 40, 40, 35, 23, 25, 35, 25, 21, 24, 33, ~
$ pain        <ord> No Pain, No Pain, Severe Pain, Severe Pain, Severe Pain, No ~
$ painYn      <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ marital     <dbl> 1, 1, 1, 4, 4, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 1, ~
```

`<ord> = Ordinal`



18

## Binary Variable (Logical R Data)

- 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:

```
dfSession4a <- dfSession4a %>%
  mutate(
    painYn = as.logical(x=painYn)
  )
```

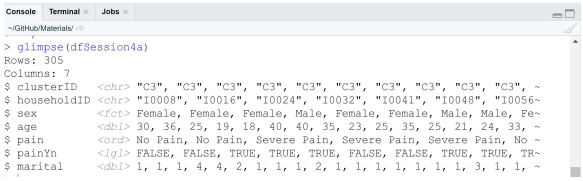
`as.logical()` is to create a binary variable which is not a nominal variable



19

## Checking Variable Properties

- Variable Properties for `painYn`

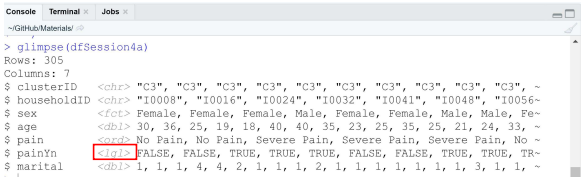





20

## Checking Variable Properties

- Variable Properties for `painYn`




`<lgl>` = Logical (Binary variable)



21

## Data Output

- Once the processing is done, we sometimes save the processed data
- `write_csv()` function is one of the function to save the processed data into a csv file

```
write_csv(
  x = dfSession4a,
  file = "df_session4_processed_data.csv"
)
```



22

## Comment inside R code

- 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)
- To make the code easy to understand by collaborator, we need to write sufficient description of the code
- To write description inside the code, we must use a # symbol followed by the descriptive text, for example:

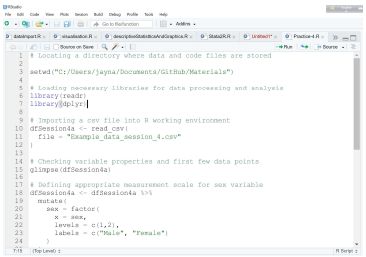
```
# to make sure sex is a nominal variable
dfSession4a <- dfSession4a %>%
  mutate(
    sex = factor(
      x = sex,
      levels = c(1,2),
      labels = c("Male", "Female")
    )
  )
```



23

## Saving R code (R Script)

- The entire code file is then saved using a `*.R` extension, for example, `Practice-4.R`






24

## Task

- Make sure marital variable is a nominal variable in R. (Hints: `mutate()`, `factor()`)



25

## Summary

- Measurement scales are linked with data types in R programming
- Appropriately define variables in R so that it is a correct representation of underlying measurement scales
- To import a csv file into R use `read_csv()` from readr library
- To create new variable use, `mutate()` from `dplyr` library
- To create nominal and ordinal variable use `factor()`
- To check properties of each variable from a data use `glimpse()` from `dplyr` library
- To export processed data into a csv file, `write_csv()`



26