Data Scientifique

GradProSkills

Data Management in R

Session I: Importing data and recoding variables

Instructor: Yara Abu Awad

Data Scientifique

HOME

Data Scientifique fosters data science initiatives among researchers

Our mission is to help researchers develop and strengthen their analytical, computational, & programming techniques to facilitate health knowledge mobilization.

Data Scientifique offers opportunities for researchers to leverage advances in data management/integration, quantitative statistical methods, and data visualization through Educational Workshops, Enrichment Awards, and Data Clinic. Data Scientifique is a professional development catalyst for faculty, post-docs, and graduate students. Resources include tools to optimize research collaboration, archival data repositories, and open-science initiatives.

In September 2015, Data Scientifique was founded by Jennifer J McGrath, PhD MPH (PERFORM Chair, Childhood Preventive Health & Data Science) and Muhammed Idris, PhD (former PERFORM Postdoctoral Scholar, current TED Resident: www.ted.com/speakers/muhammed_idris). Housed within the PERFORM Centre, in February 2018 Data Scientifique expanded with a satellite office in the Vanier Library to make data services and workshops accessible to the wider community. Since January 2019, Data Scientifique is under the direction of Jennifer J McGrath, PhD MPH and Yara Abu Awad, ScD MS MBA (current PERFORM & Horizon Postdoctoral Scholar).

www.datascientifique.ca

Workshop Schedule

- **In this workshop, the following topics will be covered:**
  - Session I: Importing data and recoding variables
  - Session II: Merging and reshaping data

Learning Objectives

- **In this workshop participants will:**
  1. Import data from different formats including the SPSS .sav, SAS .sas7bdat and .csv
  2. Change variable format
  3. Recode variables into new categories
  4. Estimate the number of missing observations in data
  5. Merge datasets
  6. Concatenate datasets
  7. Save datasets
  8. Selecting and deleting columns
  9. Selecting and deleting rows
  10. Subsetting data

Quick Poll

How many people here have coded in R before?

Plan for today

- I will talk for about an hour
- You will get a chance to practice afterwards with the exercises provided. I recommend that you work in pairs.

Importing data (rows & columns)

Source	File extension	Package	Function
Comma delimited Excel, simple text editor (e.g. notepad)	.csv	Base R	read.csv()
Simple text editor	.txt	Base R	read.table()
SPSS	.sav	foreign haven	read.spss() read_spss()
SAS	.sas7bdat	haven	read_sas()
R	.rds	Base R	readRDS()
PostgreSQL, ArcGIS	.dbf	foreign	read.dbf()
Excel	.xlsx	readxl xlsx	read_excel() read.xlsx()
R	.rds .RData	Base R	readRDS() load() or import with the RStudio menu

Imported data is different!

etc..

Step 1: Some words of advice

- Check your data to make sure that missing values, character variables and numeric variables were imported correctly
  - You can do this by: visually inspecting your data, the str() and summary() functions
- If you have leading zeros, import that variable as a character or R will assume it is numeric and delete those zeros
- I recommend converting your ID variable to character if it is numeric to avoid loss of important characters
- Trial and error!
- Google!
- Let’s look at read.csv

Step 1: Import .csv

Read.csv is the function you are using to import your data

```
df = read.csv('C:\\User\\Yara\\mydata.csv')
```

df is the name you are assigning to your dataset and it will be an object in R.
If you skip df = then R will import your data and print it in the console.

The path to your file and the full name of your file (note the quotation marks). You don't need the full path if you have already set your working directory to the folder where this file is stored.
Rstudio Menu: Session -> Set Working Directory -> Choose Directory

If you have set the working directory to the folder where the file is stored then this becomes

```
df = read.csv('mydata.csv')
```

I will now demonstrate in R

How to handle missing data?

- MCAR (Missing Completely at Random): You can drop those observations (but you will lose power)
- MAR (Missing at Random): You can explain the missingness
  - Multiple imputation to fill in missing values
    - Amelia II is a nice package to do this in R followed by the package zelig for pooled data analysis
  - Inverse Probability Weighting to adjust for differential loss to follow-up
- MNAR: You can do nothing. Sad times.

How does R handle missing data?

- Special value of NA is assigned to missing data
- NA stands for ‘Not Available’
- The function is.na: asks if an object is missing and returns TRUE/FALSE
- You can set an object named x to missing as follows:
  - x <- NA or x = NA

Missing Data

- How to determine how much missing data there is in your dataset for an individual column:
  - Continuous variables: summary(df$variable)
  - Categorical variables: table(df$variable, useNA = ‘always’)
- In the entire dataset:
  - sum(is.na(df)) – returns the total number of cells with missing data
  - colSums(is.na(df)) – returns the total number of missing cells per column

Recoding variables as missing

- What does *recode* mean?
  - It means to transform a variable
- Correctly code missing as NA in R
  - Example 1: if all cells containing ‘777’ are actually missing in your data, you can set them to NA as follows (only do this if ‘777’ is not a valid value in *any* cell of your data)

```
df[df == ‘777’] <- NA
```

  - Example 2: if you want to code all “I don’t know” answers to a question as missing:

```
df$questionr = ifelse(df$question == “I don’t know”, NA, df$question)
```

Changing format of data: factor to character

- Character variable accidentally imported as factor?

$ tells R to look for a column inside the object df

Note: you are replacing an existing column! Do this with care

df is the name you assigned your dataset and it is an object in R

```
df$variablename = as.character(df$variablename)
```

'variablename' is the name of your column / variable in df

this function converts data to character format

The data that you want to convert. In this example, it is the column named 'variablename'

Note that in R data.frame objects: column names are in character format

Changing format of data: factor to numeric

- Numeric variable accidentally imported as factor?

this function converts
data to numeric format

```
df$variablename = as.numeric(as.character(df$variablename))
```

Rule of thumb: the innermost function is executed first. So the column is converted to character and then to numeric. I recommend this for factor variables so there is no loss of information.

Why do we go to the trouble of converting data from one type to another?

Categorize Continuous variable

- Example: this is done often in Nutritional epi for frequency of consumption

```
df$fruitsperday_cat = cut(df$fruitsperday, 4)
```

OR

```
df$fruitsperday_cat = cut(df$fruitsperday, breaks = c(0,0.5,2,5,9), labels =
c('none', 'Up to 2', 'three to five', 'six to nine' ), include.lowest = T)
```

Change category names in a variable:

- Example, turn string answers on a Likert-like scale (‘strongly agree’, ‘agree’, ‘disagree’, ‘strongly disagree’) to numbers (1,2,3,4)

```
df$question_numeric = ifelse(df$original_question == ‘strongly agree’, 1, NA)
```

How would you complete this transformation?

Arithmetically transform a variable

- Fahrenheit to Celsius

```
df$celsius = (df$fahrenheit – 32) * 5/9
```

- Log transform

```
df$lvariable = log(df$variable)
```

- Exponentiate

```
df$expvariable = exp(df$variable)
```

etc…

The tidyverse

- So far we have been looking at Base R
- Increasingly popular alternative is the package dplyr which is part of the tidyverse
- It's a little more intuitive
- More info here:

https://www.tidyverse.org/

The tidyverse

Tidyverse

Packages Blog Learn Help Contribute

R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

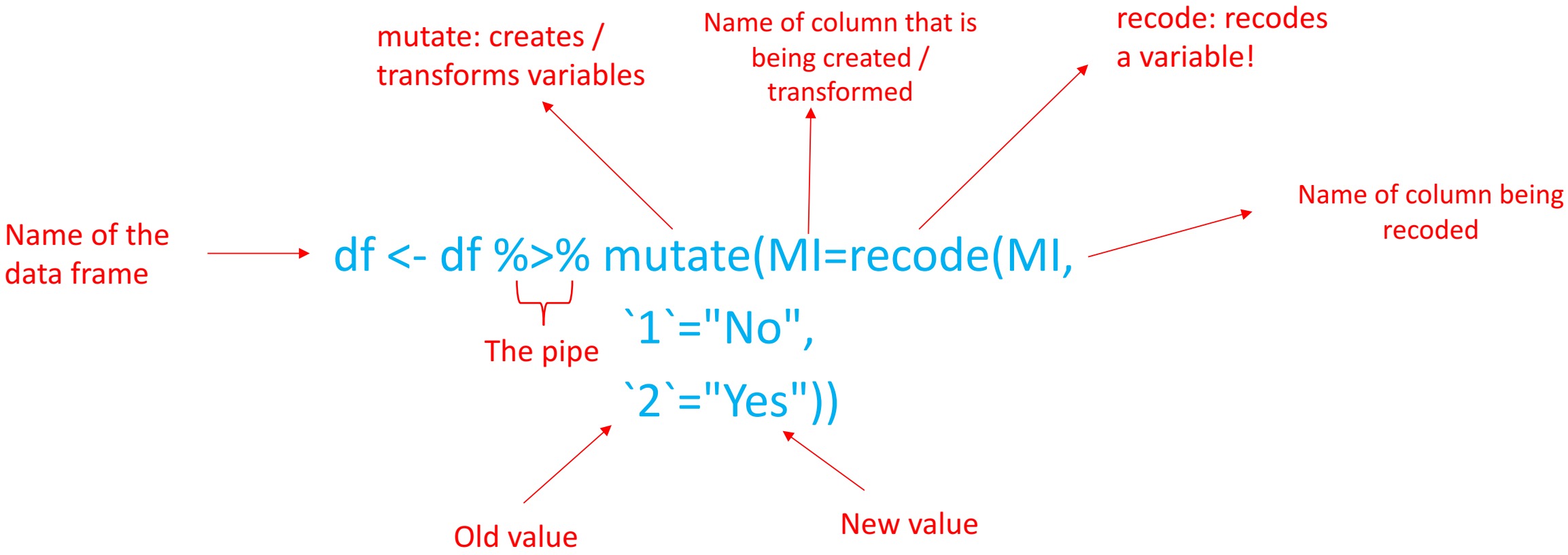
Install the complete tidyverse with:

```
install.packages("tidyverse")
```

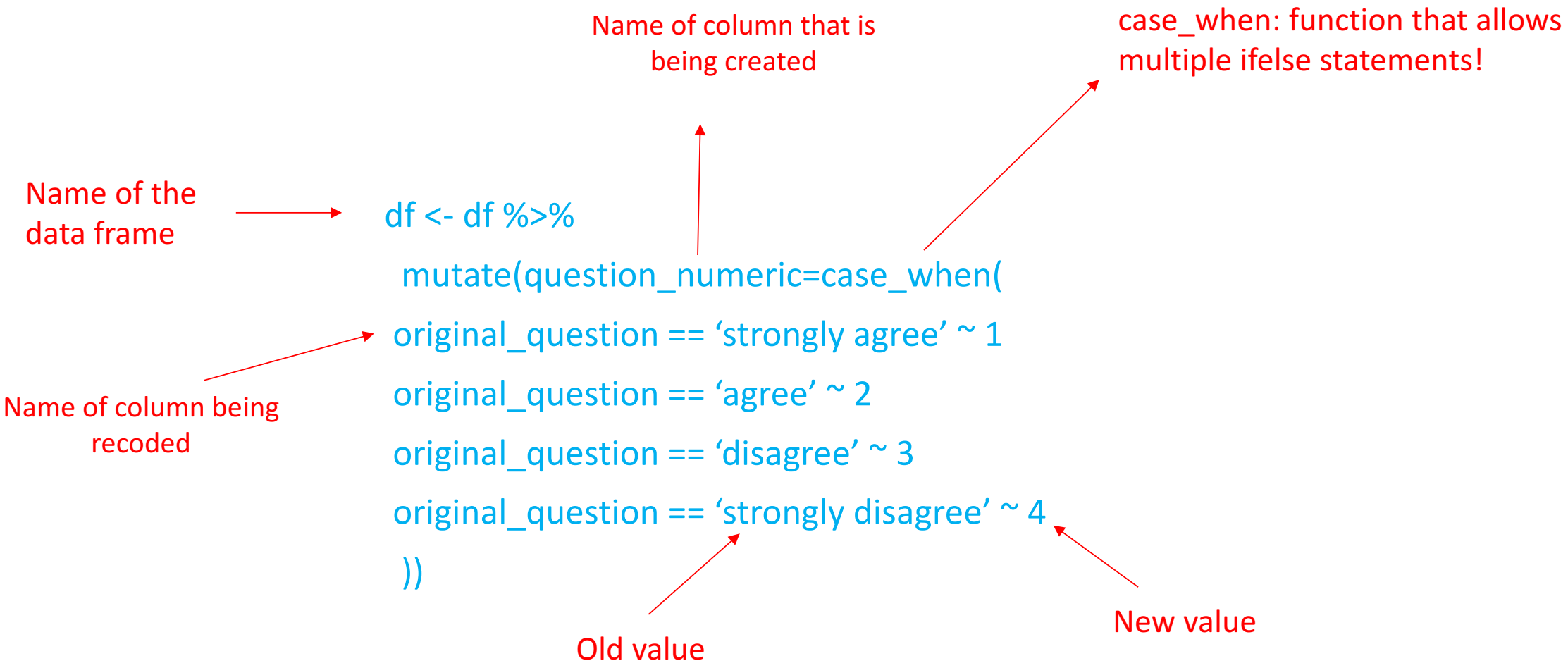
The pipe %>%

- Takes the object on the left and funnels it into the function on the right
- Makes code cleaner and easier to read

Using dplyr to recode a column



Alternative to nested ifelse statements



Stats by groups using group_by

```
df  %>%
  group_by(sex) %>%
  summarise(median_weight = median(weight, na.rm = TRUE))
```

Using dplyr with missing data

- Counting missing values in one column:

```
df %>%
summarise(count = sum(is.na(column_name)))
```

- Replace a pr-eset missing value with NA:

```
df <- df %>%
mutate(variable1 = replace(variable1, variable1 == “777", NA))
%>%
mutate(variable2 = replace(variable2, variable2 == "N/A", NA))
```

replace_na in dplyr

When your numeric columns that are 0 were accidentally set to missing:

```
df_na_replaced <- df %>%
                  mutate_if(is.numeric, replace_na,0)
```

On to the exercises!