Universität zu Köln

SFB 1252 PROMINENCE IN LANGUAGE

Practical Skills for Working with Linguistic Data

Short name: Practical Linguistic Skills (PLS)

Fahime (Fafa) Same
f.same@uni-koeln.de

WiSe23/24

Session 5: 15.11.2023
Review & File Operations in R

Quick Review

Minitask 3: R Markdown and Tidyverse

Objective

To practice R Markdown and a few basic functions in tidyverse.

Code Chunk Options

You can give different customizations to your chunks, e.g., whether or not to print the code in the output.

1. **include = FALSE** prevents code and results from appearing in the finished file.
2. **echo = TRUE** prevents code, but not the results from appearing in the finished file.
3. **message = FALSE** prevents messages that are generated by code from appearing in the finished file.
4. **warning = FALSE** prevents warnings that are generated by code from appearing in the finished.
5. **fig.cap = "..."** adds a caption to graphical results.

```
{r setup, include=FALSE}
```

Important Functions for Now

- **select()** Choose specific columns.
- **filter()** Subset rows based on conditions.
- **rename()** Renaming columns.
- **mutate()** Create new variables/columns.
- **arrange()** Sort the data based on one or more variables.

General syntax of a tidyverse command:
function_name(dataframe, arguments)
Arguments are the specific parameters or conditions that guide the function. [1]

[1] Script from the last session can be found at Ilias/Scripts/2023-11-08_markdown

Pipe Operator (%>%)

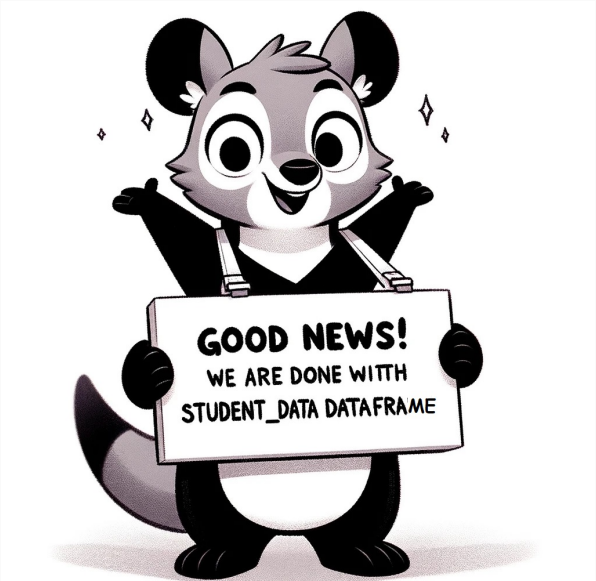
- Shortcut: shift+ctrl+M (Windows) , command+shit+M (Mac)
- The "pipe" operator (%>%), allows you to chain together multiple operations so that the output of one operation becomes the input for the next.

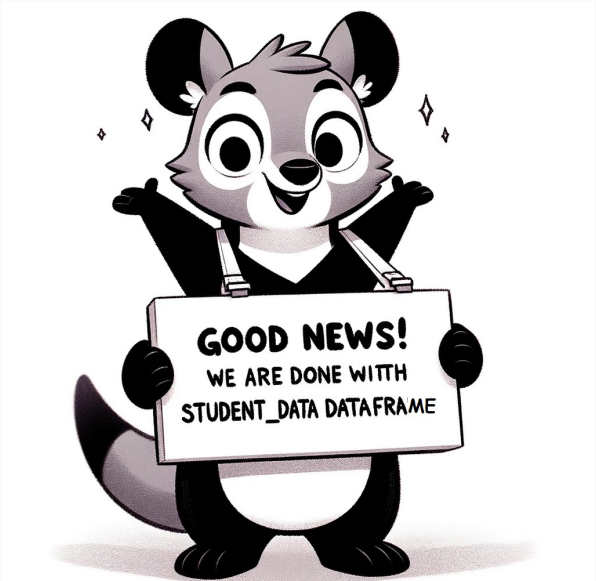
https://info5940.infosci.cornell.edu/slides/pipes-and-functions-in-r/#9

```
student_data_pipe <- student_data %>% #start a pipeline
  select(Name, Score) %>% #select only Name and Score columns
  rename(StudName = Name) %>% #rename Name column to StudName
  mutate(ScoreCorrected = Score + 1) #Increase the score by 1
```

File Operations in R

Importing Data into R





Let's work with real data!

Importing Data into R

When importing data into R, there are three key aspects to consider:

- file path
- file formats
- options

Introduction to File Paths

Path A file path specifies the location of a file or folder in a computer's file system.

Path is essential for reading and writing files in R.

Introduction to File Paths

Path A file path specifies the location of a file or folder in a computer's file system. **Path** is essential for reading and writing files in R.

Types of File Paths

1. **Absolute Path**: Specifies the exact location of a file or directory from the root directory. `"D:/sciebo_fafa/PLS/session5/data.csv"`

Introduction to File Paths

Path A file path specifies the location of a file or folder in a computer's file system. **Path** is essential for reading and writing files in R.

Types of File Paths

1. **Absolute Path**: Specifies the exact location of a file or directory from the root directory. `"D:/sciebo_fafa/PLS/session5/data.csv"`
2. **Relative Path**: Specifies the location relative to the current working directory. `"./session5/data.csv"`

Introduction to File Paths

Path A file path specifies the location of a file or folder in a computer's file system. **Path** is essential for reading and writing files in R.

Types of File Paths

1. **Absolute Path**: Specifies the exact location of a file or directory from the root directory. `"D:/sciebo_fafa/PLS/session5/data.csv"`
2. **Relative Path**: Specifies the location relative to the current working directory. `"./session5/data.csv"`
   - It is (kinda) self-contained.
   - Single dot (.) represents the current directory, e.g., PLS.
   - Double dot (..) represents the parent directory (one directory level up from the current directory), e.g., sciebo_fafa

Backslash: Windows' File Path Dilemma

1. Windows Paths use **backslash** \ to separate directories.
   `C:\Users\Fafa\Documents\data.csv`
2. Mac and Linux Paths Use forward slashes / to separate directories.
   `C:/Users/Fafa/Documents/data.csv`

Backslash: Windows' File Path Dilemma

1. Windows Paths use **backslash** \ to separate directories.
   `C:\Users\Fafa\Documents\data.csv`
2. Mac and Linux Paths Use forward slashes / to separate directories.
   `C:/Users/Fafa/Documents/data.csv`

TO WINDOWS USERS: In R, you need to use double backslashes \\, or forward slash / because a single backslash is an escape character.

So the path becomes

C:\\Users\\Fafa\\Documents\\data.csv

OR

C:/Users/Fafa/Documents/data.csv (much-preferred solution)

Working with Paths in R

1. `getwd()`: Function to get the current working directory.
2. `setwd()`: Function to set the current working directory.
3. `list.files(".", recursive = TRUE)`: list all files recursively in your working directory.
4. use **TAB** to see the files in your current working directory.

File Formats that can be Imported into R

1. Plain text files
   - CSV (Comma-Separated Values)
   - TSV (Tab-Separated Values)
   - FWF (Fixed Width Format)
   - Delimited Text Files (custom delimiters)
   - Flat non-delimited text files
2. Excel files
   - XLS and XLSX
3. XML and HTML files
   - XML (eXtensible Markup Language)
   - HTML (Hypertext Markup Language)
4. JSON (JavaScript Object Notation) Files
5. Binary files
   - RData/RDS
   - SPSS, Stata, SAS Files
6. Other
   - Statistical Software Formats
   - Image Files
   - SQL Databases
   - Word documents

Our Focus Today

1. Plain text files
   - CSV (Comma-Separated Values)
   - TSV (Tab-Separated Values)
   - FWF (Fixed Width Format)
   - Delimited Text Files (custom delimiters)
   - Flat non-delimited text files
2. Excel files
   - XLS and XLSX
3. XML and HTML files
   - XML (eXtensible Markup Language)
   - HTML (Hypertext Markup Language)
4. JSON (JavaScript Object Notation) Files
5. Binary files
   - RData/RDS
   - SPSS, Stata, SAS Files
6. Other
   - Statistical Software Formats
   - Image Files
   - SQL Databases
   - Word documents

Importing Plain and Excel Files

You can import data from various file formats using different functions:

1. **read_csv()**: Imports data from a CSV (comma-separated values) file.
   `data <- read_csv("path/data.csv")`
2. **read_tsv()**: Imports data from a TSV (tab-separated values) file.
3. **read_excel()**: Imports data from an Excel file (.xlsx or .xls).
4. **read_delim()**: Imports data from a delimited text file with user-specified delimiter (e.g., comma, tab, etc.). Default delimiter: tab.
   `data <- read_delim("path/data.txt", delimiter = "\t")`
5. **read_file()**: Imports the contents of a text file as one continuous piece of text.

Import Options

- file: The path to the file to be read.
- col_names:
  1. TRUE (to use the first line as column headers)
  2. FALSE (to generate default names)
- locale: Controls the encoding
- na: A character vector of what to interpret as NA values.
- skip: Number of lines to skip before reading data
- comment: A string used to identify comments.
- sheet: Specifies which sheet to read [only read_excel function].
- range: A cell range to read from [only read_excel function].

Writing Files for Export

Writing files is the opposite of reading files into R. It is for exporting data.

```
write_csv(your_dataframe, "your_filename.csv")
write_tsv()
write_delim()
```

QUESTIONS?