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 4: 08.11.2023
R markdown & Tidyverse

Introduction to R Markdown

What is Markdown?

Markdown

1. lightweight markup language
2. for creating formatted text
3. using a plain-text editor.
4. can be converted to HTML and many other formats
5. file extension: .md

What is Markdown?

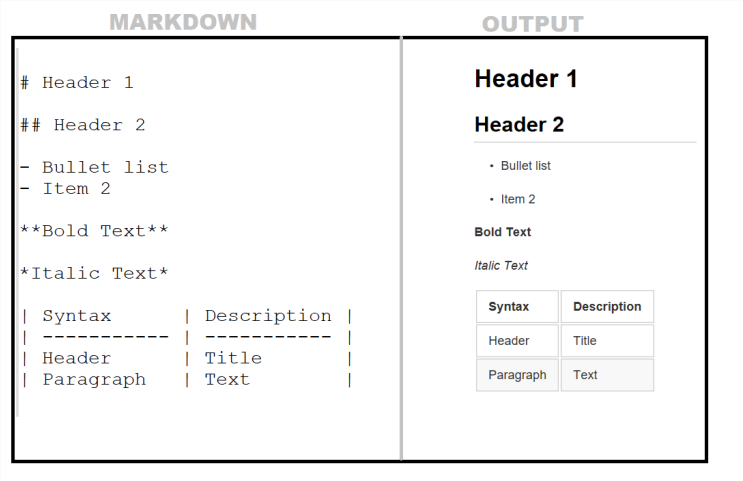
Markdown

1. lightweight markup language
2. for creating formatted text
3. using a plain-text editor.
4. can be converted to HTML and many other formats
5. file extension: .md

Slides: Ilias/Slides/2023_11_08_Session4

Script from the last session: Ilias/Scripts/2023-10-25_Tidyverse

Markdown Example



Markdown cheatsheet: `https://www.markdownguide.org/cheat-sheet/`

Try out this online markdown editor: `https://stackedit.io/app#`

What is R Markdown?

From now on, we will write our code in R Markdown (.rmd) instead of R scripts (.r).

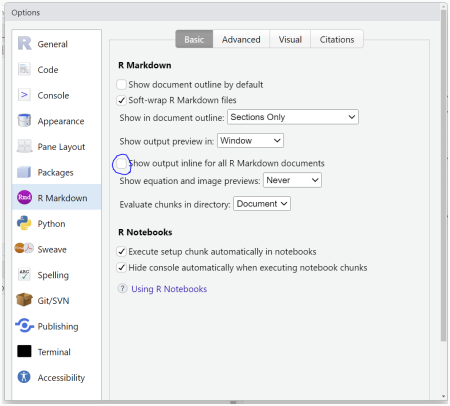
- A markdown file in RStudio
- You can use a single R Markdown file to both
  1. save and execute code
  2. generate high quality reports that can be shared with an audience

First step:

```
install.packages("rmarkdown")
install.packages("knitr")
```

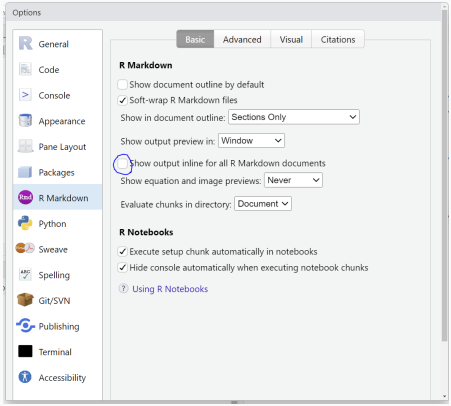
Preparatory Steps

1. In your RStudio, go to Tools → Global Options → (left panel) R Markdown → **uncheck** "Show output inline for all R Markdown document"



Preparatory Steps

1. In your RStudio, go to Tools → Global Options → (left panel) R Markdown → **uncheck** "Show output inline for all R Markdown document"



2. File → New File → R Markdown

❶ A YAML header surrounded by − − − HEADER − − −

❷ R code chunks surrounded by ``` CODE ```

❸ text mixed with markdown

3. You can use the "Knit" button to render the file and preview the output with a single click or keyboard shortcut: Windows (ctrl + shift + K), Mac (Command + Shift + K).

How to Type Code Chunks

1. **EASIEST:** Keyboard shortcut **Ctrl + Alt + I** on Windows or **Cmd + Option + I** on Mac.
2. The green "Add Chunk" command  in the editor toolbar → select R.
3. Typing the chunk delimiters ```` ```{r} ```` and ```` ``` ````.

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 = FALSE` 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}
```

Useful Links

1. Cheatsheet: `https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf`
2. Markdown basics: `https://rmarkdown.rstudio.com/lesson-1.html`

Tidyverse

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/2023-10-25_Tidyverse

Our Student Dataframe

```
# Create the data frame
student_data <- data.frame(
  ID = c(1, 2, 3, 4, 5, 6, 7, 8),
  Name = c("Alice", "Bob", "Carol", "Dave", "Rachel", "Richard",
       "John", "Monica"),
  Score = c(90, 85, 92, 88, 95, 80, 83, 86)
)
```

Function: select()

This function is used for subsetting columns and removing unwanted columns.

Components

- **Function Name:** `select`
- **Data:** The data frame you are manipulating.
- **Arguments:** The columns you want to **keep or remove**.

```
select(data, column_names)
onlyNames <- select(student_data, Name) # Keeping only the Name
    column
```

Function: filter()

This function **filters** rows based on specified conditions.

```
filter(student_data, Score > 90) # filter rows where scores is
    great than 90
below90 <- filter(student_data, Score <= 90) # Filter rows
    where scores are less or equal to 90. Assign the new
    filtered dataframe to the variable below90
filter(student_data, Name == "Alice") # filter rows that the
    name of the student is Alice
```

NOTE And (&) and or (|) operators allow you to combine two multiple conditions:

- mother tongue = German & gender = female (both conditions should met)
- mother tongue = German | age ¡ 30 (one condition suffices)

Function: rename()

As the name suggests, this function is used for renaming columns.

```
rename(data, new_name = old_name)

rename(student_data, "Number" = "ID")
```

Function: mutate()

This function is used to create new columns or modifying the existing columns.

```
mutate(data, new_column = something)
mutate(student_data, occupation = "student") # Adding an
    occupation column with the value "student"
mutate(student_data, Score = Score +1) # Increasing the scores
    by 1
mutate(student_data, NameScore = str_c(Name,"-", Score)) #
    Concatenating Name and Score columns
mutate(student_data, Uppercase = str_to_upper(Name)) # Turn
    names into uppercase
```

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`

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
```

Best Practices for Coding in R

1. **Punctuation**
   - Assignment Operator ($<-$): Use for variable assignment; avoid $=$.
   - Quotes (" " or ' '): Both work, but be consistent.
   - Commas: Separate function arguments and vector/matrix elements.
   - Parentheses: Necessary for functions (func()) and control structures (if()). No space between the function name and parentheses.
2. **Indentation** Use indentation to make your code more readable.
3. **Naming Conventions** Stick to lowercase, camelCase, or separate words with underscores whenever possible (my_dataframe). Give short but meaningful names.
4. **Code Structure**
   - Comments: Use # liberally to comment on your code.
   - Whitespace: Use blank lines to separate logical blocks of code.
5. **Error Messages** Pay close attention; they usually point to the issue.

QUESTIONS?