WRIC Data Processing Workshop

Nina Ziegenbein

Welcome!

This document serves as a guided tutorial for our workshop on **06.03.2025** at Steno Diabetes Center Aarhus. It is designed to help you get started using the WRIC_preprocessing package, but you can use it afterwards to follow it on your own.

Before we dive into the code-along session, please ensure you have the following installed:

- **R**: The programming language we'll be using for data analysis and visualization.
- **A Programming Environment (RStudio)**: I will demonstrate using RStudio, but feel free to use a different IDE.
- **Quarto**: To render this tutorial, if you choose to go through it by yourself.

Click here for installation links and instructions

- **Install R**
- **Install RStudio**
- **Install Quarto**

FAQ and "Programming Terms"

What is the difference between R and RStudio?

R is a programming language, while **RStudio** is a, so called, integrated development environment (IDE) designed specifically for R, offering a user-friendly interface for coding, plotting, and managing projects. You can think of R like a language, like english, where RStudio is Word - a program you can write english inside. But you could also use other programs for example libre office or latex.

qmd vs R vs Rmd

There are different types of files where you can write R code. A file ending in **.R** is a standard R script for writing and running R code, while in **.qmd (Quarto Document)** or **.Rmd (R Markdown)** you can combine code and text for dynamic reports, or for example this tutorial.

Getting Started

All functions are contained in the `R/WRIC_preprocessing.R` file. But to also have access to example data, doc_strings, tutorials and HowTo's please download the entire repository for this workshop.

1. Download the entire repository from GitHub

- Go to the repository on GitHub and click the **Code** button.
- Click **Download ZIP** to download the entire repository, then extract it. If you have Git installed on your computer or GitHub Desktop, feel free to clone the repository instead of downloading it. `git clone https://github.com/hulmanlab/WRIC_processing.git`

2. Set your working directory

- Set your working directory in R to the downloaded WRIC_processing directory. That means your path should end in `".../WRIC_processing"`. You can check the full path, by [add info for both mac and windows]
- **Via RStudio menu:** Session -> Set Working Directory -> Choose Directory
- **Via Terminal:**

```
setwd("path/to/this/folder/WRIC_processing")
getwd() # To check the current directory
```

Why do we change the working directory

The working directory is where RStudio is looking for files. Meaning we can easily type source('R/WRIC_preprocessing.R') instead of the entire file path. There is nothing wrong with specifiying the entire file path and not changing the working directory. But for this workshop it makes it a lot easier, when we can all use the same code and this ensure that all code runs on your computer without having to change anything.

3. Importing the functions from the "package"

- `source()` runs an R script and loads its functions into your environment

```
source('R/WRIC_preprocessing.R')
```

```
RedCap API configuration file 'config.R' not found. Proceeding without.


Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Preprocess WRIC data

Now let's preprocess the txt files, that is created by the WRIC. The function `preproces_WRIC_file()` disentangle the meta-data at the top of the file (ID, comment etc) and creates DataFrames and csv-files with the actual data, seperated between both rooms and summarized between the two measurements for each room.

```
result <- preprocess_WRIC_file("example_data/data.txt")
```

```
Rows: 2601 Columns: 67
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr   (4): X1, X18, X35, X52
dbl  (56): X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X2...
lgl   (3): X17, X34, X51
time  (4): X2, X19, X36, X53

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
R1_metadata <- result$R1_metadata
R2_metadata <- result$R2_metadata
df_room1 <- result$df_room1
df_room2 <- result$df_room2
```

The function returns a list with “R1_metadata”, “R2_metadata”, “df_room1” and “df_room2”. Each item of the list is a DataFrame of either the metadata or the preprocessed actual data for either room 1 or 2. If ´save_csv‘ is True, then the DataFrames will be saved as csv files with “id_comment_WRIC_data.csv” or “id_comment_WRIC_metadata.csv”.

Let’s look at the output really quick for room 1:

```
View(R1_metadata)
View(df_room1)
```

But the function can do a lot more and has a lot of extra parameters you can specify. The following is the exact same function call, but mentioning all optional parameters you can call with their default values. Default means, that if you do not specify this parameter, this is the value that the parameter has by default.

```
result <- preprocess_WRIC_file(
    "./example_data/data.txt",
    code="id",
    manual=NULL,
    save_csv=TRUE,
    path_to_save=NULL,
    combine=TRUE,
    method="mean",
    start=NULL,
    end=NULL,
    notefilepath= NULL,
    keywords_dict=NULL
)
```

```
Rows: 2601 Columns: 67
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr   (4): X1, X18, X35, X52
dbl  (56): X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X2...
lgl   (3): X17, X34, X51
time  (4): X2, X19, X36, X53

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Here are explanations and options to all parameters you can specify:

- **filepath:** [String, filepath] Directory path to the WRIC .txt file.
- **code** [String] Method for generating subject IDs. Default is "id", also possible to specify "id+comment", where both ID and comment values are combined or "manual", where you can specify your own.
- **manual** [String] Custom codes for subjects in Room 1 and Room 2 if `code` is "manual".
- **save_csv** [Logical], whether to save extracted metadata and data to CSV files or not. Default is True
- **path_to_save** [String] Directory path for saving CSV files, NULL uses the current directory, NULL is Deafult.
- **combine** [Logical], whether to combine S1 and S2 measurements. Default is True
- **method** [String] Method for combining measurements ("mean", "median", "s1", "s2", "min", "max").
- **start** [character or POSIXct or NULL], rows before this will be removed, if NULL takes first row e.g "2023-11-13 11:43:00"
- **end** [character or POSIXct or NULL], rows after this will be removed, if NULL takes last rows e.g "2023-11-13 11:43:00"
- **notefilepath:** If you specify a path to the corresponding notefile, the code will try to automatically extract the datetime and current protocol specification (sleeping, exercising, eating etc). If possible please read the How To Note File, before you start your study for consistent note taking. If there is a TimeStamp in the note e.g "Participants starts eating at 16:10", the time of the creation of the note will be overwritten with the time specified in the free-text of the note. The "protocol" is extracted by keyword search. You can check currently included keywords and extend them by checking the keywords_dict in the extract_note_info() function of the WRIC_preprocessing.R file.
- **keywords_dict:** [Nested List] A "dictionary" with keywords for extracting protocol information out of the notefile.

Your Turn

So now it is your turn. Using the `preprocess_WRIC_file()` method create a csv file using "data.txt" in folder example_data.

1) create a csv with the name "XXXX_WRIC_data.csv" combining S1 and S2 measurements by taking the mean between them.
2) create a csv, but cut-off the start to 10:45 on 13/11/2023 and the end to 11:58 on the same day. The csv should be saved as "testing_start_end_parameter_WRIC_data.csv".
3) *Optional:* Try out the notefilepath parameter and see what happens.

Automatic note file extraction - adaptation to your notes

One helpful feature of the `preprocess_WRIC_file()` method is to automatically extract the protocol from the note_file, that is filled in manually during the experiment. With "protocol" I mean coding wether the participant is currently sleeping, eating, exercising etc. This enables quick processing and easy access to extract and compare various e.g. eating periods. Let's try it:

```
result <- preprocess_WRIC_file("./example_data/data.txt", notefilepath="./example_data/note.
```

```
Rows: 2601 Columns: 67
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr   (4): X1, X18, X35, X52
dbl  (56): X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X2...
lgl   (3): X17, X34, X51
time  (4): X2, X19, X36, X53

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
[1] "Starting time for room 1 is 2023-11-13 21:14:22 and end 2023-11-15 06:35:48 and for roo
```

```
Drift: 1.35
```

```
View(result$df_room1)
```

When looking at `df_room1` now, we can see a new column called "protocol". We can see the file starts with 0 and at 22:41:21 changes to 1.

Your Turn

1) Look into the `note.txt` file and find out why there is a change at 22:41:21 and what 0 and 1 might represent. Are there more numbers? What do they represent?
2) When comparing with previous results, notice that the file now starts at a later time and stops at an earlier one. Why might that be? _OBS:_Since we keep reusing variable names (result, df_room1 etc) and use the same data.txt file to create csv_files, we overwrite those files and variables. That is completely fine for this tutorial, where we are focused on how to use it and not the results. But be careful in your own work!

A bit more information about extracting data from the notefile

- write about drift and explain that subtracted from times in notes
- explain how notefile parameter works with example (e.g. necessary to create example data for that)

Docstrings

- show docstring to see parameter documentation

Batch Processing

- looping over folder with multiple files
- super quick excursion: RedCap API -> verweis auf seperates tutorial

Working with a subset of the data (specific time)

- example from already extracted data, import it again as dataframe
- extract certain time frame
- maybe some examplary biostats example? -> ask chatgpt for something simple

Visualizing WRIC data

```r
2 + 2
```

```
[1] 4
```