# Debugging and troubleshooting in R

## Debugging

(verb) de-buhg-ing

> Being the detective in a crime movie where you are also the murderer

Jorrit Mesman
Uppsala University
2023



# Workshop Outline

- Presentation (30 min.)
  - A five-step strategy to deal with problems in your R code
- Troubleshooting walkthrough + questions & discussion (45 min.)

## Introduction

- Dealing with unexpected problems and errors happens often
- Usually not included in introductory courses
- Everyone develops their own techniques

- Here: a five-step approach to deal with errors
- Little focus on specific errors, but rather a general strategy that can be applied to most code-related issues

# Troubleshooting strategy

- (0. Double check your code)
- 1. Locate problem/error
- 2. Understand error message
- 3. Read documentation
- 4. Search help on the internet
- 5. Follow-up steps (reach out, try examples, or change approach)

# Step 0 - Double check your code

- Implied first step
- Probably most mistakes are caused by small oversights
  - Forgotten brackets, commas, pipes, + signs, etc.
  - Forgetting to import a library or load a function
  - Referring to a wrong file or directory
- Often creates hard-to-understand error messages

- Most important step often immediately leads to solution
- You need to know what the problem is to look for a solution
- In general:
  - Run code step by step
    - Optionally using the Rstudio debugger mode
  - debug() and traceback() functions, if your problem relates to a function
- The part where the error is created is not always the part where the real problem occurred. Try to trace back!

```
> if(Sys.Date() < as.Date("2022-10-01")){
   values = c(1, 2, 3)
}
> mean(values)
Error in mean(values) : object 'values' not found
```

"Finding your bug is a process of confirming the many things that you believe are true — until you find one which is not true."

- Norm Matloff

- Running code line by line, tracking your environment
- You can implement some <a href="mailto:print">print()</a> statements as well to find more quickly where the problem occurs
- debug(function\_name), browser(), and undebug(function\_name)

- In case your problem is not related to an error:
  - Identify where the problem causing the faulty behaviour is located
    - Convert warnings to errors options(warn = 2)
    - You can use stop() in combination to an if-statement to generate an error when a certain condition is triggered (e.g. an empty data.frame is created)
- Functions; work from inside to outside (or from top to bottom if it involves pipes)
  - Or click "Show traceback" in Rstudio (or traceback() outside Rstudio)
- Loops; see at what step the problem occurs

- What if your error only occurs sometimes
  - Empty your environment before you run your code, or even re-start R (to get rid of loaded packages)
  - If you interact with servers or websites, the problem may lay in an interrupted connection or a server that is down.
- R Session Aborted, R encountered a fatal error (bomb message)
  - Often memory-related. Try to make your code use less memory.
- Package installation errors
  - Not really considered here, but similarly, find the package that is causing the real problem (it may not be the one you're trying to install)

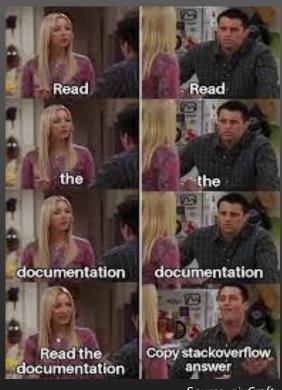


# Step 2 - Understand the error message

- Ideally, an error message is informative, pointing you to a solution
- Examples
  - read.csv("data\_lake\_temp.csv") -> cannot open file 'data\_lake\_temp.csv': No such file or directory
  - var / 2 -> non-numeric argument to binary operator
  - $\frac{df[,z:=x+y]}{}$  Check that is.data.table(DT) == TRUE. Otherwise, := and `:=`(...) are defined for use in j, once only and in particular ways. See help(":=")
  - ggplot(df) + geom\_line() -> `geom\_line()` requires the following missing aesthetics: x and y
- You may start to recognise common error messages over time
- Works best in packages/functions that aim to "fail fast" (see recommendations at the end)

## Step 3 - Read the documentation

- ?function\_name
- Often skipped, but advisable before looking on the Internet
- Functions may require different inputs than you expect.
- Look at the types/classes of the arguments
- Documentations often contain an example; compare it to your own function call



Source: nixCraft

## Step 4 - Search help on the internet

- Important that you identified the cause of your problem (Step 1)
- If an error message is generated, you can use this in your search
- Suggestions for search query:
  - r \*package\_name\* \*error-message\* \*description\*
- StackOverflow often provides good suggestions, but there are other sources
- ChatGPT (or alternatives)
- Specific packages may have a dedicated vignettes/FAQ/Cheat Sheet/Github page

# Step 4 - Search help on the internet

## String manipulation with stringr:: CHEAT SHEET

The stringr package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.



#### **Detect Matches**



str\_detect(string, pattern) Detect the
presence of a pattern match in a string.
str detect(fruit, "a")

str\_which(string, pattern) Find the indexes of strings that contain a pattern match. str\_which(fruit, "a")

str\_count(string, pattern) Count the number
of matches in a string.
str\_count(fruit, "a")

**str\_locate**(string, **pattern**) Locate the positions of pattern matches in a string. Also **str\_locate\_all**. *str\_locate*(*fruit*, "a")

#### **Subset Strings**



str\_sub(string, start = 1L, end = -1L) Extract
substrings from a character vector.
str\_sub(fruit, 1, 3); str\_sub(fruit, -2)

str\_subset(string, pattern) Return only the strings that contain a pattern match. str\_subset(fruit, "b")

str\_extract(string, pattern) Return the first pattern match found in each string, as a vector. Also str\_extract\_all to return every pattern match. str\_extract(fruit, "[aeiou]")

str\_match(string, pattern) Return the first
pattern match found in each string, as a
matrix with a column for each () group in
pattern. Also str\_match\_all.
str\_match(sentences, "(althe) ([^ \]+)")

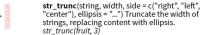
## Manage Lengths



**str\_length**(string) The width of strings (i.e. number of code points, which generally equals the number of characters). *str\_length(fruit)* 



 $\begin{array}{l} \textbf{str\_pad}(\text{string, width, side} = \text{c}(\text{"left", "right", "both"}), pad = \text{" "}) \ \text{Pad strings to constant} \\ \text{width.} \ str\_pad(\textit{fruit, 17}) \end{array}$ 





str\_trim(string, side = c("both", "left", "right"))
Trim whitespace from the start and/or end of a
string. str\_trim(fruit)

### **Mutate Strings**



a string

str\_sub() <- value. Replace substrings by
identifying the substrings with str\_sub() and
assigning into the results.
str\_sub(fruit, 1, 3) <- "str"</pre>

**str\_replace**(string, **pattern**, replacement) Replace the first matched pattern in each string. str\_replace(fruit, "a", "-")

str\_replace\_all(string, pattern, replacement) Replace all matched patterns in each string. str\_replace\_all(fruit, "a", "-")

str\_to\_lower(string, locale = "en")¹ Convert
strings to lower case.
str\_to\_lower(sentences)

str\_to\_upper(string, locale = "en")¹ Convert
strings to upper case.
str\_to\_upper(sentences)

str\_to\_title(string, locale = "en")¹ Convert
strings to title case. str\_to\_title(sentences)

## Join and Split



{xx} {yy}

str\_c(letters, LETTERS)

str\_c(..., sep = "", collapse = NULL) Collapse
a vector of strings into a single string.

str\_c(..., sep = "", collapse = NULL) Join
multiple strings into a single string.

str\_c(letters, collapse = "")

str\_dup(string, times) Repeat strings times
times. str\_dup(fruit, times = 2)

str\_split\_fixed(string, pattern, n) Split a
vector of strings into a matrix of substrings
(splitting at occurrences of a pattern match).
Also str\_split to return a list of substrings.
str\_split\_fixed(fruit, " ", n=2)

str\_glue(..., .sep = "", .envir = parent.frame())
Create a string from strings and {expressions}
to evaluate. str\_glue("Pi is {pi}")

str\_glue\_data(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA") Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate. str\_glue\_data(mtcars, "frownames(mtcars)) has hab hb")

#### **Order Strings**



str order(x, decreasing = FALSE, na\_last =
TRŪE, locale = "en", numeric = FALSE, ...]¹ Return
the vector of indexes that sorts a character
vector. x[str\_order(x)]



str\_sort(x, decreasing = FALSE, na\_last = TRUE, locale = "en", numeric = FALSE, ...)<sup>I</sup> Sort a character vector.

### Helpers



apple banana

pear

**str\_conv**(string, encoding) Override the encoding of a string. str\_conv(fruit,"ISO-8859-1")

str\_view(string, pattern, match = NA) View
HTML rendering of first regex match in each
string. str\_view(fruit, "[aeiou]")

str\_view\_all(string, pattern, match = NA) View
HTML rendering of all regex matches.
str\_view\_all(fruit, "[aeiou]")

str\_wrap(string, width = 80, indent = 0, exdent = 0) Wrap strings into nicely formatted paragraphs. str\_wrap(sentences, 20)



<sup>1</sup> See bit.ly/ISO639-1 for a complete list of locales.

# Step 5 - Follow-up steps

- Contact colleagues
- For complex workflows; go through example setups
- Ask on the Internet
  - Use the right channel
  - If possible include a reproducible example (and perhaps sessionInfo())
    - https://stackoverflow.com/help/minimal-reproducibleexample
- Change approach
  - If your code doesn't work, no one on the internet seems to have done something similar, and no one can help you, you might be using a wrong approach



Source: thecoderpedia.com

# Troubleshooting strategy

- (0. Double check your code)
- 1. Locate problem/error
- 2. Understand error message
- 3. Read documentation
- 4. Search help on the internet
- 5. Follow-up steps (reach out, try examples, or change approach)

## Avoiding errors

- Don't do too much in one script split up tasks
- Comment your code
- Try to copy as little code as possible within a script/project, instead use loops/apply functions, if needed
- Build in exception handling or code that throws informative errors
- Design to "fail fast", e.g. check if the arguments into your function are correct
- For packages: automated testing (e.g. testthat package)
- Workflow management software (targets package)

# Thank you for your attention!

## References

- https://adv-r.hadley.nz/debugging.html
- https://cosimameyer.com/post/mastering-debugging-in-r/
- https://bookdown.org/yih\_huynh/Guide-to-R-Book/trouble.html

Lake Erken data: <a href="https://data.fieldsites.se/portal/">https://data.fieldsites.se/portal/</a>

