Functions and Errors

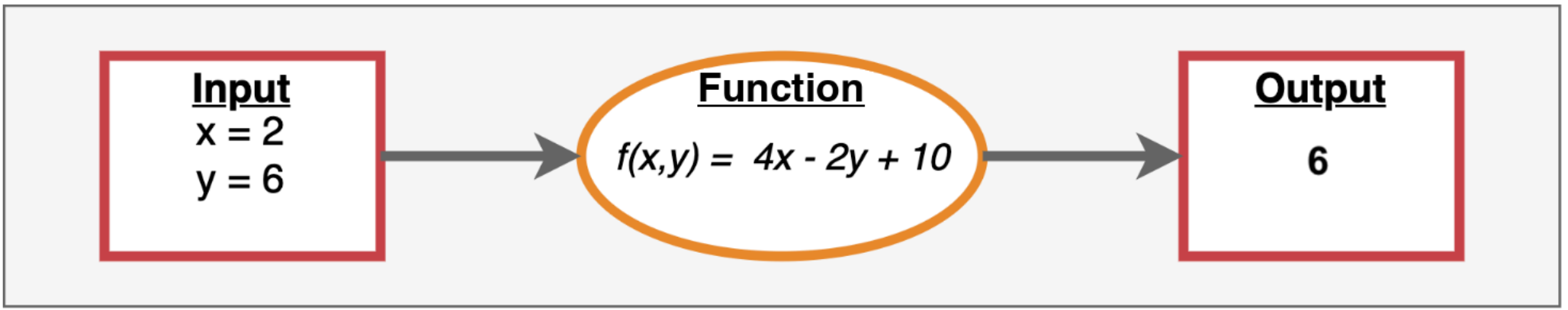
Functions

Functions are a relatively complex and advanced topic. We will cover them later. Since we use them right from the start (e.g. the `mean` function to compute the mean), it is important to be clear about a few aspects:

- Functions as objects
- Mandatory and optional arguments, and defaults
- Order of arguments
- Documentation

Functions

We can think of functions in R in the same way as classic mathematical functions. Given some **input** values, functions perform specific calculations and return the result as **output**.



Functions as objects

We have already seen that everything in R is an object. Functions too, although very different from other elements, are created and treated in R as objects. In this example, we create a function that takes **x** as input and **adds** the value **3** to **x**.

The output will depend on the value of x.

```
1 myfun <- function(x) {
2   return(x + 3)
3 }
4
5 myfun(2)
```

```
[1] 5
```

```
1 myfun(7)
```

```
[1] 10
```

Functions as objects

We can create, remove or overwrite them like any other object. We will see later how to create them, but keep in mind that all the functions we use are created as objects and stored in the environment.

Arguments

Function arguments are what we, as *users*, need to know and set correctly so that the function does what it was designed to do. In the previous example the only argument was x. Let's look at the `help` of the `mean()` function instead.

Arguments

There are 2 rules for setting these arguments:

- the order does not matter **IF I SPECIFY THE ARGUMENT NAME**, e.g. `x = vector`, `na.rm = TRUE`, etc.
- the **order matters** IF I DO **NOT SPECIFY THE ARGUMENT NAME.** I can omit `argument = value`, but I must respect the order in which the function was written.

Arguments

Let's try to use the `mean()` function:

```r
# create a vector by sampling 1000 elements
# from a normal distribution with mean 1 and sd 1
myvec <- rnorm(n = 1000, mean = 1, sd = 1)

mean(x = myvec) # x defined, trim and na.rm not defined, so they are
```

```
[1] 1.012073
```

```r
?mean
```

```r
mean(x = myvec, na.rm = TRUE) # x defined, na.rm defined
```

```
[1] 1.012073
```

```r
mean(myvec, TRUE) # what happens?
```

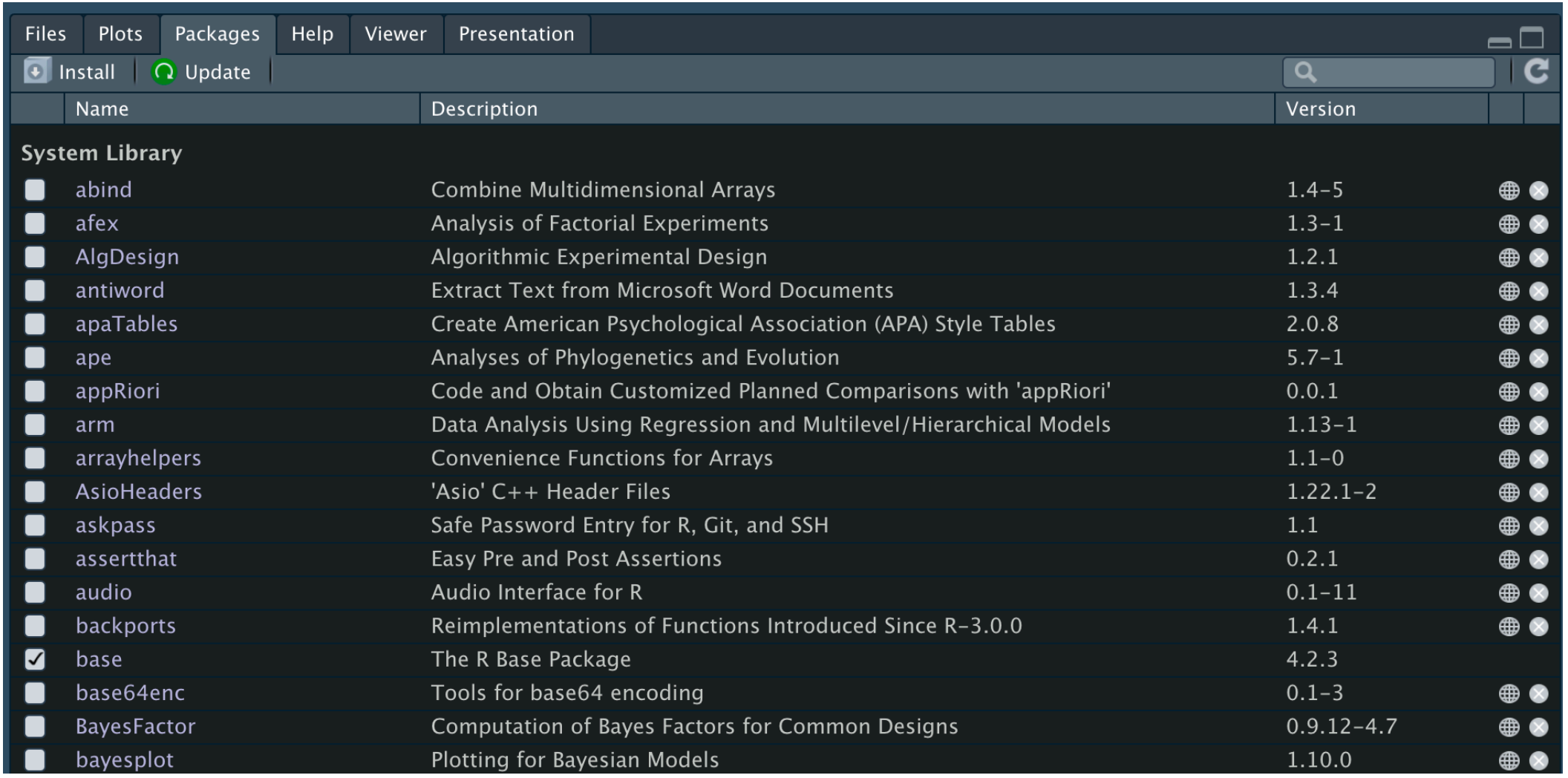
```
Error in `mean.default()`:
! 'trim' must be numeric of length one
```

Packages

In R we can install and load additional packages, which simply make available libraries of functions created by other users. To use a package:

- Install the package with `install.packages("packagename")`
- Load the package with `library(packagename)`
- Access a function without loading the package with `packagename::functionname()` (useful if you only need one function or if there are conflicts)

Packages


Files
Plots
Packages
Help
Viewer
Presentation
Install
Update
Name
Description
Version
System Library
abind Combine Multidimensional Arrays 1.4-5
afex Analysis of Factorial Experiments 1.3-1
AlgDesign Algorithmic Experimental Design 1.2.1
antiword Extract Text from Microsoft Word Documents 1.3.4
apaTables Create American Psychological Association (APA) Style Tables 2.0.8
ape Analyses of Phylogenetics and Evolution 5.7-1
appRiori Code and Obtain Customized Planned Comparisons with 'appRiori' 0.0.1
arm Data Analysis Using Regression and Multilevel/Hierarchical Models 1.13-1
arrayhelpers Convenience Functions for Arrays 1.1-0
AsioHeaders 'Asio' C++ Header Files 1.22.1-2
askpass Safe Password Entry for R, Git, and SSH 1.1
assertthat Easy Pre and Post Assertions 0.2.1
audio Audio Interface for R 0.1-11
backports Reimplementations of Functions Introduced Since R-3.0.0 1.4.1
base The R Base Package 4.2.3
base64enc Tools for base64 encoding 0.1-3
BayesFactor Computation of Bayes Factors for Common Designs 0.9.12-4.7
bayesplot Plotting for Bayesian Models 1.10.0

How to solve problems in R

In R, errors are:

- inevitable
- part of the code itself
- educational

R and errors

There are different **alert** levels when we write code:

- `messages`: the function returns something that is useful to know, but everything is fine
- `warnings`: the function informs us about something *potentially* problematic, but (more or less) everything is fine
- `errors`: the function not only informs us about an **error**, but the requested operations were not executed

How to solve them?

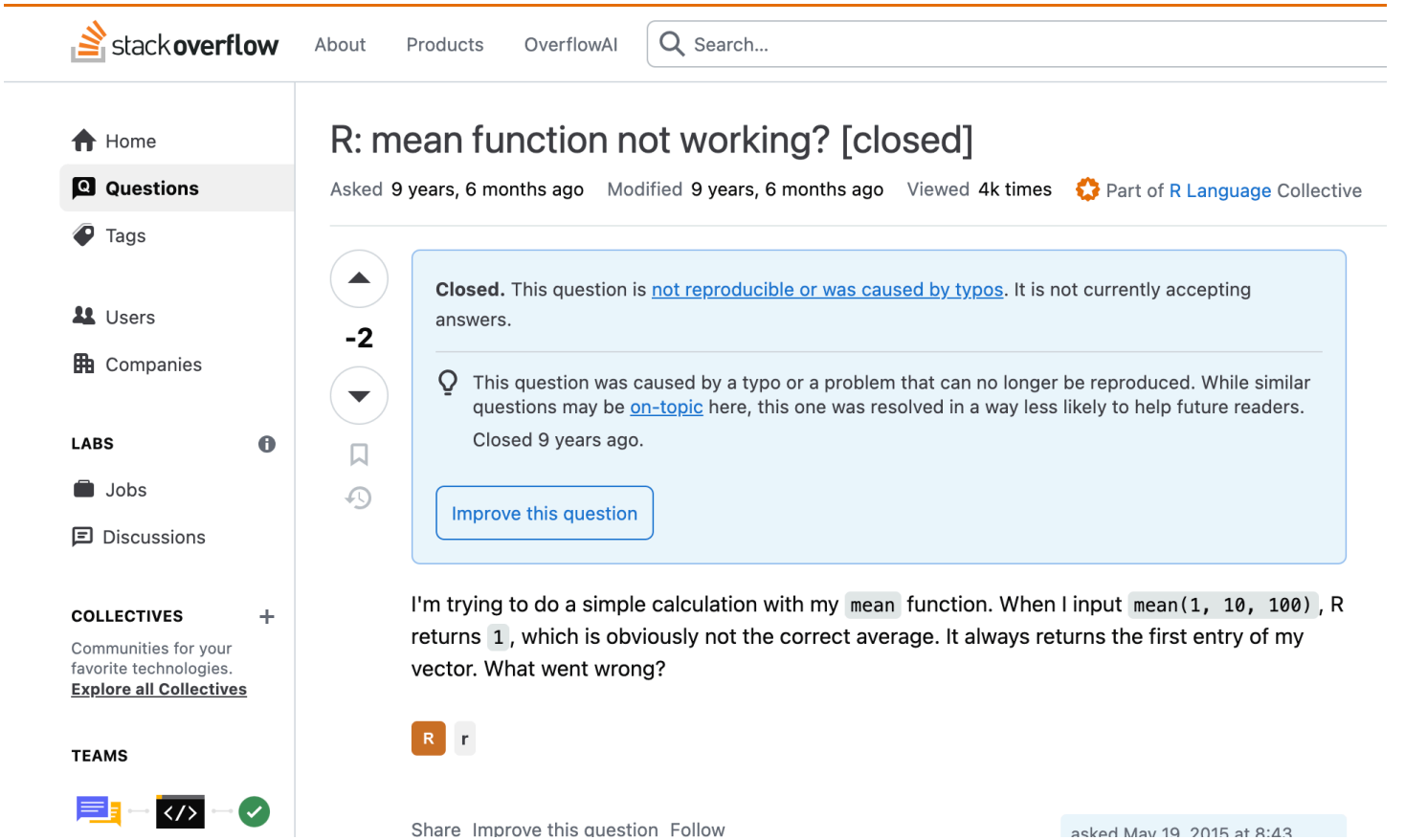
- Understand the message
- Read the function documentation
- Search for the message online
- Ask for help in dedicated forums

Function documentation

- Every function has a documentation page, accessible with `?functionname`, `??functionname` or `help(functionname)`
- We can also look up the package documentation
- We can search online for the function name or for the message we get

Stack overflow

Stack overflow is a discussion forum about anything involving code (statistics, programming, etc.). It is full of common errors, *How to do …* questions and extremely useful answers/solutions. In 90% of cases the problem you have is common and a solution already exists.



AI

Be careful

- Ask the question clearly
- Make sure you have understood the answer
- Double-check!