



R Programming

Part of the "Getting Started with Positron" workshop

Posit Software, PBC



Multiple versions of R

R interpreter sessions

- Positron readily discovers and offers multiple versions of R
- Positron can have multiple, concurrent interpreter sessions, that can be
 - a mix of different R versions
 - a mix of R and Python sessions
 - multiple instances of a single R version

New ▾Open ▾Save

EXPLORER

OPEN EDITORS

USETHIS

R

OUTLINE

TIMELINE

create.R

create.R

addin.R

air.R

author.R

badge.R

block.R

browse.R

ci.R

citation.R

code-of-conduct.R

course.R

coverage.R

cpp11.R

cran.R

create.R

data-table.R

data.R

description.R

directory.R

1

2

3

4

5

6

7

8

9

10

11

12

13

R > create.R

#' Create a package or project

#'

#' @description

#' These functions create an R project:

#' * `create\_package()` creates an R package

#' * `create\_project()` creates a non-package project, i.e. a data analysis

#' project

#'

#' Both functions can be called on an existing project; you will be asked before

#' any existing files are changed.

#'

#' @inheritParams use_description

#' @param fields A named list of fields to add to `DESCRIPTION`, potentially

CONSOLE

TERMINAL

PROBLEMS

OUTPUT

PORTS

DEBUG CONSOLE

~/rrr/usethis

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> devtools::load_all()
i Loading usethis
>

main*

usethis

Quarto: 1.7.31

R 4.4.2

usethis ▾

SESSION

HELP

...

×

←

→

🏠

🔍

Ln 1, Col 1

Spaces: 2

UTF-8

LF

R

🗨

🔔

Select Interpreter Session

R 4.4.2 Currently Selected

/Library/Frameworks/R.framework/Versions/4.4-arm64/Resource...

Active Interpreter Sessions

New Interpreter Session...

New

Open

EXPLORER

OPEN EDITORS

USETHIS

OUTLINE

TIMELINE

create.R R

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

Python 3.11.11 (Pyenv)

~/pyenv/versions/3.11.11/bin/python

Python 3.10.16 (Pyenv)

~/pyenv/versions/3.10.16/bin/python

Python 3.13.3 (Global)

/opt/homebrew/bin/python3

Python 3.9.6 (Global)

/usr/bin/python3

R 4.4.2

/Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/bin/R

R 4.5.0

/Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/bin/R

R 4.3.3

/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/bin/R

Pyenv

Global

System

#' overriding default values. See [use_description()] for how you can set

#' personalized defaults using package options.

#' @param path A path. If it exists, it is used. If it does not exist, it is

#' created, provided that the parent path exists.

#' @param roxygen Do you plan to use roxygen2 to document your package?

#' @param rstudio If `TRUE`, calls [use_rstudio()] to make the new package or

#' project into an [RStudio

#' Project](https://r-pkgs.org/workflow101.html#sec-workflow101-rstudio-projects).

#' If `FALSE` and a non-package project, a sentinel `.here` file is placed so

#' that the directory can be recognized as a project by the

#' [here](https://here.r-lib.org) or

CONSOLE

TERMINAL

PROBLEMS

TEST RESULTS

OUTPUT

PORTS

~/rrr/usethis

>

main

0

1

0

0

usethis

Quarto: 1.7.31

R 4.4.2

usethis

SESSION

VARIABLES

PLOTS

No variables have been created.

Ln 1, Col 1

Spaces: 2

UTF-8

LF

R

create.R — usethis

0%

New

Open

<

>

Search

R 4.4.2

usethis

EXPLORER

OPEN EDITORS

create.R R

USETHIS

.github

.Rproj.user

.vscode

docs

inst

man

pkgdown

R

addin.R

air.R

author.R

badge.R

block.R

browse.R

ci.R

citation.R

code-of-conduct.R

course.R

coverage.R

cpp11.R

cran.R

create.R

data-table.R

data.R

description.R

directory.R

OUTLINE

TIMELINE

create.R

R > create.R > ...

1 #' Create a package or project

2 #'

3 #' @description

4 #' These functions create an R project:

5 #' * `create\_package()` creates an R package

6 #' * `create\_project()` creates a non-package project, i.e. a data analysis

7 #' project

8 #'

9 #' Both functions can be called on an existing project; you will be asked before

10 #' any existing files are changed.

11 #'

12 #' @inheritParams use_description

13 #' @param fields A named list of fields to add to `DESCRIPTION`, potentially

14 #' overriding default values. See [use_description()] for how you can set

15 #' personalized defaults using package options.

16 #' @param path A path. If it exists, it is used. If it does not exist, it is

17 #' created, provided that the parent path exists.

18 #' @param roxygen Do you plan to use roxygen2 to document your package?

19 #' @param rstudio If `TRUE`, calls [use_rstudio()] to make the new package or

20 #' project into an [RStudio

21 #' Project](https://r-pkgs.org/workflow101.html#sec-workflow101-rstudio-projects).

22 #' If `FALSE` and a non-package project, a sentinel `.here` file is placed so

23 #' that the directory can be recognized as a project by the

24 #' [here](https://here.r-lib.org) or

CONSOLE

TERMINAL

PROBLEMS

TEST RESULTS

OUTPUT

PORTS

~/rrr/usethis

>

SESSION

VARIABLES

R 4.4.2

filter

No variables have been created.

PLOTS

main

0↓ 1↑

0 0

usethis

Quarto: 1.7.31

Ln 1, Col 1

Spaces: 2

UTF-8

LF

R

R 4.4.2

R 4.5.0

R 4.5.0

R 4.3.3

Why is it useful to have multiple R versions?

- If you are responsible for R code that must run "elsewhere" or that was developed in the past
 - You maintain a package and other users might have different R versions
 - You maintain a data product that gets deployed to a server
 - You develop code that's meant to run in some other, high performance environment
 - You need to revisit an analysis that was crafted 1-2 years ago
- Having multiple R versions locally helps you replicate and solve issues arising from interactions between your code and another R version

Why is it useful to have multiple R sessions?

- Put a long-running task in its own session, while you continue interactive work in another session
- Live comparison between, e.g., 2 different R versions or 2 different versions of an R package
- Develop a self-contained document, e.g., a vignette, in one session, while you continue casual interactive work in another session
- Do you have more use cases?

How do you end up with multiple R versions?

- Highly recommended tool:
 - [rig: The R Installation Manager](#)
- rig is not part of Positron, they just work well together
- rig is especially important for macOS users
 - It is almost impossible to have multiple, functional R versions on macOS if you just install R from CRAN

Other joys of rig

- Out-of-the-box set-up of default CRAN mirror
- Creates and configures user-level, R-version-specific package libraries
- Installs pak, a nifty R package for package installation
- Installs appropriate Rtools versions on Windows
- Additionally:
 - Updates macOS R installation so you can use lldb to debug C / C++ code
 - Tidies up R-related cruft in the Windows registry

5_m 00_s

Your turn

Multiple R sessions (option 1 of 2)

- Create 2 (or more!) concurrent R sessions of the same or different versions
- Create different objects in each session and explore how that plays out in the Variables pane
- Restart 1 session (but not the other(s)). What do you see in the Variables pane now?
- Click the ⓘ in the Console action bar to see session metadata. Can you find the R executable path?
- Can you figure out how to rename an R session?

Your turn

Multiple R versions (option 2 of 2)

- Install rig: <https://github.com/r-lib/rig>. Note the docs in repo's README.
- Use rig to list your existing R versions. Which one is the system default?
- Ask rig to list all available R versions.
- Install another R version. Make it the new default (or not).
- Launch Positron and see that the new R version is now also available.
- Consider going back to your previous default R version.

Air

Air formatter for R



posit-dev.github.io/air

- When you install Positron, you get Air "for free" as a bootstrapped extension.
- Air formats your R code based on some opinionated, but also widely-used and agreed-upon, conventions — details at posit-dev.github.io/air/formatter.html.

Air + Positron: practical suggestions

- Air extension ships with Positron and includes the Air binary so Air should just work in Positron.
- You can configure Air to be used at the
 - user level to say “I use Air to format the R code in all of my projects”: by adjusting your user setting (more on this in the next module) or
 - workspace level to say “We use Air to format the R code in this project”: by adjusting your workspace settings (more on this in the next module too) or with `usethis::use_air()`
- You can also use Air “one-off” from the *Command palette*:
 - *Air: Initialize Workspace Folder*
 - *Air: Format Workspace Folder*
 - *Format Document*
 - *Format Selection*
- The Git diff is a great way to see what has changed, so inspect it before committing and pushing!
- There are various ways to disable Air formatting of a specific file, line (or lines in a pipeline), etc.

Your turn

Format some ugly R code

- The example project we downloaded earlier has a file with poorly formatted R code: `air-practice.R`
- Open it in Positron. Alternatively, open a personal R file with questionable formatting.
- Remove the `# fmt: skip file` line, so that Air will format the file.
- Experiment with the *Format Selection* and/or *Format Document* commands to see how Air would reformat it.
- Inspect the Git diff to see what's changed.

If you ❤️ Air (and you will ❤️ Air)

- `usethis::use_air()` turns on Format on Save for R and Quarto documents by putting this in `settings.json`*:

```
{
  "[r]": {
    "editor.formatOnSave": true,
    "editor.defaultFormatter": "Posit.air-vscode"
  },
  "[quarto]": {
    "editor.formatOnSave": true,
    "editor.defaultFormatter": "quarto.quarto"
  }
}
```

* More on settings in the next module!

Snippets

Built-in snippet example: for loop

```
> for
```

 for	[keyword]	<pre>for (\${1:variable} in \${2:vector}) { \${0} }</pre>	✕
 for	A for loop		
 for cats ::			
 for oreign ::			

👉 helps you create code like 👈

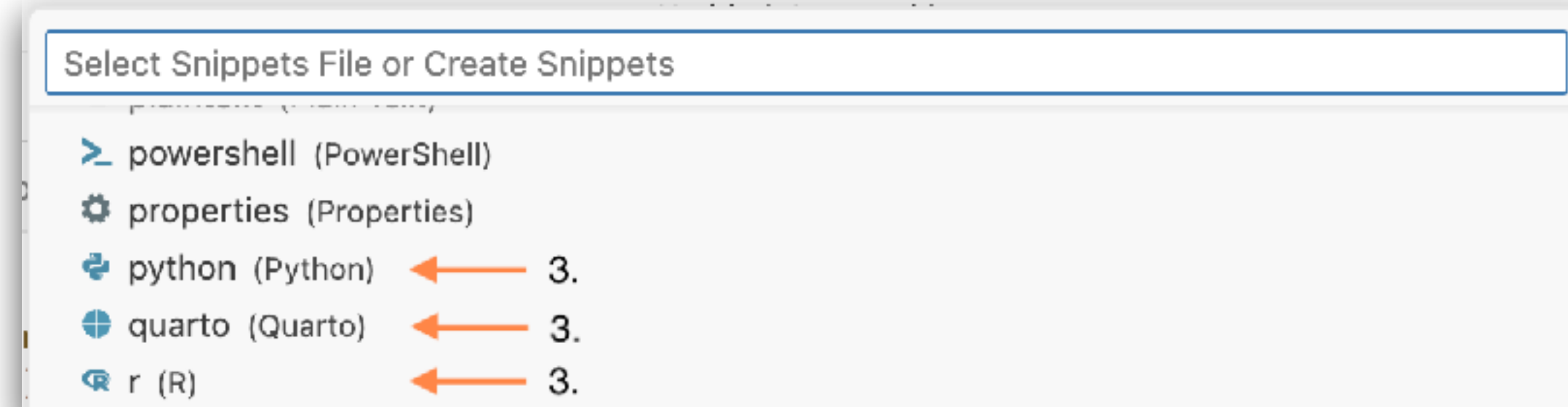
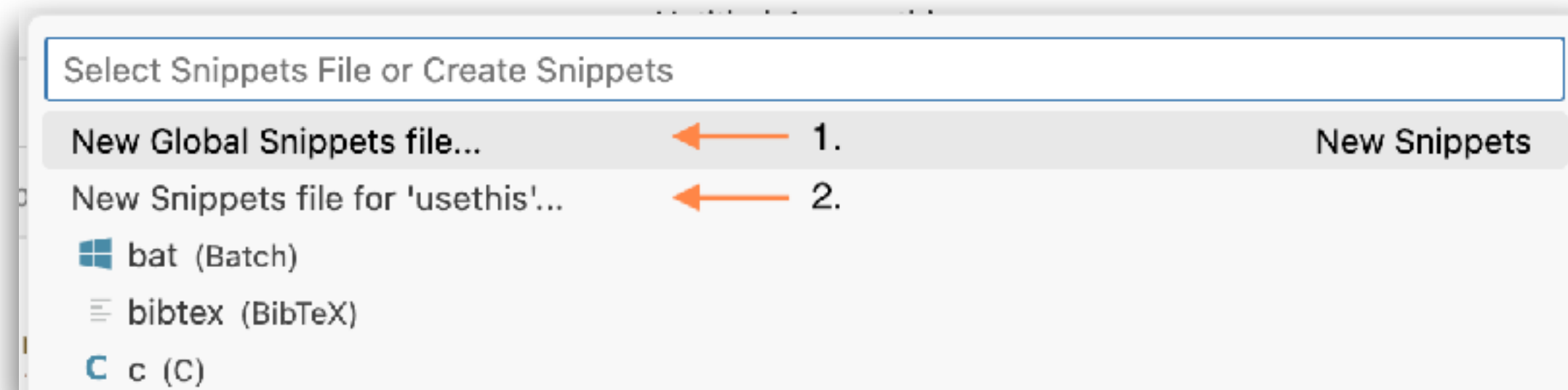
```
for (variable in vector) {  
  # code to repeat  
}
```

Snippets

- Positron's R support provides a few snippets related to R's reserved words.
- Positron provides fewer built-in snippets than RStudio.
- You can configure additional snippets at the user or workspace level.
- Positron uses TextMate syntax for snippets, inherited from VS Code. This is different from RStudio's snippet syntax.
- Snippets are typically inserted via the usual completions offered by IntelliSense. There's also a dedicated command: *Insert Snippet*.
- <https://positron.posit.co/r-snippets.html>

How to configure your own snippets

Command palette: *Snippets: Configure Snippets*



1. Global Snippets file: User-level. Potentially more than 1 language.
2. Workspace-specific file: Specific to 1 workspace. Potentially more than 1 language.
3. Language-specific file: User-level.

<https://positron.posit.co/r-snippets.html>

Debugging

Debugging

- Positron supports debugging R code via these base mechanisms
 - `debug()`
 - `debugonce()`
 - `browser()`
- Positron also supports breakpoints
 - <https://positron.posit.co/guide-r-debugging.html>