Package: BiocBook (via r-universe)

November 25, 2026

Title Write, containerize, publish and version Quarto books with Bioconductor

Description A BiocBook can be created by authors (e.g. R developers, but also scientists, teachers, communicators, ...) who wish to 1) write (compile a body of biological and/or bioinformatics knowledge), 2) containerize (provide Docker images to reproduce the examples illustrated in the compendium), 3) publish (deploy an online book to disseminate the compendium), and 4) version (automatically generate specific online book versions and Docker images for specific Bioconductor releases).

Version 1.12.0

Date 2023-08-03

URL https://bioconductor.org/packages/BiocBook

BugReports https://github.com/js2264/BiocBook/issues

Depends R (>= 4.3)

Imports BiocGenerics, desc, digest, pak, cli, glue, gert, gh, gitcreds, httr, usethis, dplyr, purrr, tibble, methods, rprojroot, reticulate, stringr, yaml, tools, utils, rlang, quarto, renv

Suggests BiocManager, BiocStyle, knitr, testthat (>= 3.0.0), rmarkdown

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

biocViews Infrastructure, ReportWriting, Software

VignetteBuilder knitr

License MIT + file LICENSE

Collate 'doc.R' 'imports.R' 'init.R' 'BiocBook.R' 'BiocBook-methods.R' 'bookdown.R' 'check.R' 'editing.R' 'globals.R' 'llms.R' 'micromamba.R' 'pages.R' 'python.R' 'utils.R'

Config/testthat/edition 3

Config/pak/sysreqs
cmake git make libgit2-dev libicu-dev libpng-dev libuv1-dev libssl-dev python3 libx11-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:51:22 UTC

RemoteUrl https://github.com/bioc/BiocBook

RemoteRef RELEASE_3_24

RemoteSha aac96d2c0a09d2ff3bfaccd7ea7674a7a7c3102b

Contents

BiocBook 2
BiocBook-bookdown 5
BiocBook-editing 8
BiocBook-llms 10
micromamba 12

Index **16**

`BiocBook` *Handling BiocBook directories*

Description

`BiocBooks` are local R packages containing an extra `pages` folder to write up online book chapters.

1. A new `BiocBook` should be created using `init(new_package = "...")` (or alternatively, locally with `quick_init(new_package = "...", user = "...")`).
2. A newly created `BiocBook` can be accessed to in R using `biocbook <- BiocBook(path = "...")`.
3. To edit an existing `BiocBook` object, several helper functions exist:

- `add_preamble(biocbook)`
- `add_chapter(biocbook, title = "...")`
- `edit_page(biocbook, page = "...")`

Read the different sections down below for more details.

Usage

```
init(new_package, push = NA, skip_availability = FALSE, .local = FALSE)

quick_init(new_package, user)

BiocBook(path)
```

```
releases(object)

chapters(object)

## S4 method for signature 'BiocBook'
path(object)

## S4 method for signature 'BiocBook'
releases(object)

## S4 method for signature 'BiocBook'
chapters(object)

## S4 method for signature 'BiocBook'
show(object)
```

Arguments

new_package	Name to use when initiating a new BiocBook. This name should be compatible with package naming conventions from R and Bioconductor (i.e. no _ or -, no name starting with a number).
push	Optional. Logical, whether to automatically push commits to remote Github origin. If NA, a prompt will ask whether to push commits or not (default: NA).
skip_availability	Optional. Whether to skip package name availability (default: FALSE).
.local	Should only be used for examples/tests. Whether to create a matching Github repository or stay local (default: FALSE).
user	Ideally, the Github ID of the main author/organization.
path	Path of an existing BiocBook.
object	A BiocBook object, created by BiocBook or init().

Value

- init("newBook") creates a local directory, synchronizes it with the registered GitHub user, and invisibly returns a BiocBook object.
- BiocBook("newBook") returns a BiocBook object.
- path(bb), releases(bb) and chapters(bb) return the corresponding information related to the bb BiocBook.

The BiocBook class

A BiocBook object acts as a pointer to a local package directory, with book chapters contained in a pages/ folder as .qmd files.

This package directory requires a specific architecture, which is best set up using the init() function.

When created, 3 slots are defined:

- `title`: The title contained in `/inst/assets/_book.yml`
- `local_path`: The absolute path to the book package directory
- `remote_repository`: If the book is synced with Github, this will indicate the remote

Creating a `BiocBook`

A new `BiocBook` should be created using the `init(new_package = "...")` function. This function performs the following operations:

1. It checks that the provided package name is available;
2. It logs in the GitHub user accounts;
3. It creates a new **local** repository using the `BiocBook` template from `js2264/BiocBook`;
4. It pushes the local repository to a **remote** Github repository;
5. It creates an empty `gh-pages` and sets it up to serve rendered books;
6. It edits several placeholders from the template and commits the changes.

The `init(new_package = "...")` function returns a `BiocBook` object.

Quickly create a local `BiocBook`

Alternatively, a **local** `BiocBook` can be quickly created using the `quick_init(new_package = "...", user = "...")` function.

This function only creates a new **local** repository, using the `BiocBook` template from `js2264/BiocBook`.

It does **NOT**:

- Check that the provided package name is available;
- Set up/push the local repository to a **remote** Github repository;
- Set up a `gh-pages` to serve rendered books;

This implies that functions committing/pushing (`publish()`) or checking remote status (`status()`) do not work properly with a `BiocBook` initiated with `quick_init()`, unless a remote is manually set up.

To enable Github support for a local `BiocBook`, one has to manually initiate a `git` repository and add a remote as follows:

```
git init
git symbolic-ref HEAD refs/heads/devel
git add .
git commit -m 'first commit'
git remote add origin git@github.com:<user>/<biocbook>.git
git push --set-upstream origin devel
```

Editing an existing `BiocBook`

`BiocBook` objects can be modified using the following helper functions:

- `add_preamble(biocbook)` to start writing a preamble;
- `add_chapter(biocbook, title = "...")` to start writing a new chapter;
- `edit_page(biocbook, page = "...")` to edit an existing chapter.

Publishing an existing BiocBook

Important: remember to add any dependency used in your chapters to the DESCRIPTION before publishing your book. Dependencies across chapters can be found with:

```
check_deps(biocbook)
```

Note that this will not always work 100%, always use good coding practices and add your dependencies to DESCRIPTION while writing new chapters.

To locally preview the book, one can use the following command:

```
preview(biocbook)
```

To publish changes, as long as the local BiocBook has been initiated with `init()`, the writer simply has to commit changes and push them to the `origin` remote. In R, this can be done as follows:

```
publish(biocbook)
```

The different available versions published in the `origin` `gh-pages` branch can be listed with

```
status(biocbook)
```

Examples

```
## In practice, you should not use `.local` argument.
unique_id <- as.numeric(Sys.time())
bookname <- paste(Sys.info()[['sysname']], unique_id, sep = '.')
init(bookname, .local = TRUE)
bb <- BiocBook(bookname)
chapters(bb)
releases(bb)
unlink(bookname, recursive = TRUE)
```

BiocBook-bookdown *Converting a* bookdown *book to a BiocBook*

Description

`from_bookdown()` converts a bookdown project into a BiocBook in place: in its own folder and, when that folder is a git repository, on its current branch, one commit per step, numbered `[BiocBook 1/10]`, `[BiocBook 2/10]`, and so on. Run it on a branch of its own: the conversion can then be reviewed step by step, and squashed or rebased onto the main branch like any other change.

1. **Add the BiocBook package files**: `DESCRIPTION`, the `Dockerfile`, the GitHub workflows, the `quarto` configuration (`inst/_quarto.yml`, `inst/assets/`) and `vignettes/Makefile`, filled in as `init()` does. The book keeps its `README.md`, which gets the BiocBook badges, and its `.gitignore` and `.Rbuildignore`, which get the entries of the template.
2. **Move the pages to `inst/`, as BiocBook expects**: `index.Rmd` becomes `inst/index.qmd`, and every chapter a page of `inst/pages/` that keeps its file name (e.g. `10-raw.Rmd` becomes `inst/pages/10-raw.qmd`). Asset folders (e.g. `img/`) move to `inst/pages/` too, so that the relative paths of the chapters keep working. Nothing else changes, so that git follows every file.

3. **Move the book settings to the BiocBook config**: the title, authors and list of chapters, with their parts and appendices, go to `_book.yml`; the bibliographies, the CSS and the output options `quarto` has an equivalent for go to `_format.yml`. Bibliography files and CSS move to `inst/assets/`.
4. **Remove the build files BiocBook replaces**: `_bookdown.yml`, `_output.yml`, the rendered book (`output_dir`), the intermediate files of `bookdown`, and the `Makefile`, shell scripts or GitHub workflows that render the book with `bookdown`.
5. **Use the BiocBook landing page**: the preamble of `index.Rmd` becomes the welcome part of the landing page of the template.
6. **Use BiocBook cross-references** (see "Rewriting rules").
7. **Turn questions and solutions into BiocBook callouts**, in `msmbstyle` books.
8. **Use BiocBook figure layout options**.
9. **Set up each BiocBook page on its own**: `bookdown` runs every chapter in a single R session (unless `new_session: yes`), while `quarto` renders each chapter in its own. The `library()`, `require()` and `options()` calls of `index.Rmd` (and the `before_chapter_script`) are therefore repeated in a hidden chunk at the top of each chapter.
10. **Describe the BiocBook in** `DESCRIPTION`: title, description and authors from the `index.Rmd` header, the packages the pages use in `Imports`, and the licence of the book: a Creative Commons licence its pages link to, its `LICENSE` file, or else the MIT licence of the template.

A step with nothing to change makes no commit, and the commits are numbered once they are all made. Rewrites leave code chunks, inline code and the blank lines of the book alone.

`bookdown`'s `\@ref()` renders a number only, while `quarto`'s `@` references render their own "Figure", "Table" or "Chapter": a type word right before a reference is therefore dropped, and other references render their number only, as in `bookdown`.

What still needs a human is listed in `MIGRATION.md`, at the root of the book, which is not committed: downloads during the build, cached chunks, `knitr::write_bib()` calls, unresolved cross-references, dependencies that are not on CRAN or Bioconductor, output options that were not translated, and files that are not part of a BiocBook. It also records how often each rewriting rule was applied. Delete it once done.

Usage

```
from_bookdown(
  path = ".",
  package = NULL,
  user = NULL,
  style = c("auto", "bookdown", "msmbstyle"),
  appendices = NULL,
  commit = TRUE,
  skip_availability = FALSE
)
```

Arguments

`path`	Path to the `bookdown` project, i.e. the folder holding `index.Rmd` (and optionally `_bookdown.yml`). When it is in a git repository, it must be the root of the repository.

`package`	Name of the book package. Defaults to the `book_filename` of `_bookdown.yml`, else to the name of the folder.
`user`	GitHub user or organization of the book, used in the links of the template and in the name of its `Docker` image. Defaults to the owner of the `origin` remote of the repository, when it is on GitHub, else to the user of the GitHub token.
`style`	Syntax to convert: `"bookdown"`, `"msmbstyle"` (bookdown with the `msmbstyle` question and solution blocks), or `"auto"` to detect it from the output format and the pages.
`appendices`	Optional. Chapters to list as appendices of the new book (file names, with or without extension), in addition to those following a `# (APPENDIX)` header.
`commit`	Whether to commit each step, when `path` is a git repository. Its working tree must then have no uncommitted changes.
`skip_availability`	Whether to skip checking that the dependencies of the book are on CRAN or Bioconductor, which requires network access.

Value

A `BiocBook` object, invisibly.

Rewriting rules

`bookdown` / `msmbstyle`	`quarto` / `BiocBook`
`Figure \@ref(fig:x)` / `\@ref(fig:x)`	`@fig-x` / `[-@fig-x]`, chunk `x` labelled `fig-x`
`Table \@ref(tab:x)` / `\@ref(tab:x)`	`@tbl-x` / `[-@tbl-x]`, chunk `x` labelled `tbl-x`
`(\#eq:x)` / `Equation \@ref(eq:x)` / `\@ref(eq:x)`	`{#eq-x}` / `@eq-x` / `[-@eq-x]`
`Chapter \@ref(x)` / `\@ref(x)`	`@sec-x` / `[-@sec-x]`, `{#x}` renamed `{#sec-x}`
`# (PART) X {-}`	`part: "X"` in `_book.yml`
`# (APPENDIX) X {-}`	`appendices:` in `_book.yml`
‘`r msmbstyle::question_begin()‘ ... `question_end()`	a `.callout-note` titled "Question", in a `.callout-question` di
‘`r msmbstyle::solution_begin()‘ ... `solution_end()`	a collapsed `.callout-note` titled "Solution", in a `.callout-an
chunk option `fig.margin=TRUE`	‘#
chunk option `fig.fullwidth=TRUE`	‘#
output option `margin_references: TRUE`	`citation-location: margin`

See Also

`BiocBook`

Examples

```
## A minimal bookdown project, in a git repository
bd <- file.path(tempdir(), "bookdown-example")
dir.create(bd)
```

```
writeLines(c(
    "---", "title: My bookdown book", "author: Jane Doe", "---", "",
    "# Preface {-}", "", "See chapter \\@ref(intro)."
), file.path(bd, "index.Rmd"))
writeLines(
    c("# Introduction {#intro}", "", "Some text."),
    file.path(bd, "01-intro.Rmd")
)
gert::git_init(bd)
gert::git_config_set("user.name", "Jane Doe", repo = bd)
gert::git_config_set("user.email", "jane@example.com", repo = bd)
gert::git_add(".", repo = bd)
gert::git_commit("A bookdown book", repo = bd)

## Converted in place, one commit per step
bb <- from_bookdown(bd, package = "myBook", user = "jane", skip_availability = TRUE)
chapters(bb)
gert::git_log(repo = bd)$message
unlink(bd, recursive = TRUE)
```

BiocBook-editing *Editing BiocBook accessory files*

Description

Editing functions for BiocBooks. See BiocBook help sections for extended description.

Usage

```
check_deps(book)

edit_yml(book, yml = c("_book", "_website", "_knitr", "_format"), open = TRUE)

edit_bib(book, open = TRUE)

edit_requirements_yml(book, open = TRUE)

edit_css(book, open = TRUE)

preview(book, browse = FALSE, watch = FALSE)

publish(book, message = "Publishing")

status(book)

add_preamble(book, open = TRUE)

add_chapter(book, title, file = NA, position = NULL, open = TRUE)
```

```
edit_page(book, file, open = TRUE)

add_python_chapter(book, title, file = NA, position = NULL, open = TRUE)
```

Arguments

book	A BiocBook object, opened with BiocBook or created by init().
yml	Which .yml should be opened?
open	Optional. Whether to open the file for interactive editing (default: TRUE)
browse	Optional. Passed to quarto_preview() (default: FALSE).
watch	Optional. Passed to quarto_preview() (default: FALSE).
message	Optional. Message used when committing with publish().
title	A character string for a title for the new chatper. If file is not explicitely provided, the title should only contain alphanumeric characters and spaces
file	Optional. A character string for the name of the .qmd file to write the new chapter. The extension .qmd has to be provided. If not provided, the file name is deduced from the title argument.
position	Optional. A position to insert the chapter. For example, if position = 2, the new chapter will be inserted after the first existing chapter (i.e. the Welcome page)

Value

- add_*, edit_*: A BiocBook object (invisible).
- publish: TRUE (invisible) if pushing to Github was successful;
- preview: Local URL to browse dynamically rendered book;
- status: A tibble of the existing versions found on the Github repository (branch gh-pages) and of the existing Dockerfiles.

add_* functions

add_chapter() and add_preamble are convenient functions to add pages to a BiocBook.

add_python_chapter() adds a page wired to execute python code while the book builds (see BiocBook-python). It also declares BiocBook and reticulate in the Suggests field of the book DESCRIPTION, so that the build machines install them.

edit_* functions

Several accessory files can be manually edited:

- edit_page(): manually edit any page listed in chapters(book)
- edit_bib(): manually edit /inst/assets/bibliography.bib
- edit_yml(): manually edit the different yml in /inst/assets/
- edit_requirements_yml(): manually edit /inst/requirements.yml

Maintenance functions

Extra functions are provided to faciliate the maintenance of BiocBooks.

- check_deps(): is used to find dependencies from chapter pages that are not listed in DESCRIPTION
- preview(): is used to dynamically render the book locally
- publish(): is used to commit and push to remote Github branch
- status(): is used to list the book versions already deployed on the Github repository (branch gh-pages) and of the existing Dockerfiles

See Also

BiocBook

Examples

```
## In practice, you should not use `.local` argument.
unique_id <- as.numeric(Sys.time())
bookname <- paste(Sys.info()[['sysname']], unique_id, sep = '.')
bb <- init(bookname, .local = TRUE)
add_preamble(bb, open = FALSE)
add_chapter(bb, title = "Chapitre Un", open = FALSE)
unlink(bookname, recursive = TRUE)
```

BiocBook-llms *Making a BiocBook readable by AI assistants*

Description

Every BiocBook serves an /llms.txt file, following the llms.txt proposal: a Markdown index of the book that AI assistants can read instead of crawling its HTML pages.

- With quarto >= 1.11, quarto itself writes llms.txt and a Markdown copy of every page (<page>.llms.md, next to <page>.html), when the book's llms-txt option is on. Books created from template >= 1.1.0 switch it on through the llms quarto profile (inst/_quarto-llms.yml), which vignettes/Makefile activates only when the installed quarto supports it: older versions reject the option and would otherwise fail to render the book.
- enrich_llms_txt() then adds what only BiocBook knows: the book package and its version, how to install it, its Docker image, its python environment and where the other versions of the book live. Books created from template >= 1.1.0 run it after every render, from inst/assets/post-render.R. With an older quarto, which writes no llms.txt for books, it writes a minimal one listing the book chapters.

enrich_llms_txt() can be run any number of times: the BiocBook context is kept between <!-- biocbook:start --> and <!-- biocbook:end --> markers, and replaced rather than duplicated. It also tidies two quirks of quarto's output: chapter numbers rendered as raw pandoc spans in llms.txt links, and code blocks labelled downlit (by code-link: true) rather than r in the Markdown copies of the pages.

Usage

```
enrich_llms_txt(output_dir = "docs", project_dir = ".")
```

Arguments

output_dir	Folder holding the rendered book. In a post-render script, quarto provides it as the QUARTO_PROJECT_OUTPUT_DIR environment variable.
project_dir	The book's quarto project folder, i.e. the inst/ folder holding _quarto.yml. Post-render scripts run from there, hence the default.

Value

The path to llms.txt, invisibly.

Markdown copies of the pages

enrich_llms_txt() renames the Markdown copy of each page from <page>.llms.md to <page>.md, so that a page's Markdown lives at its URL with .md in place of .html (e.g. pages/intro.md for pages/intro.html). The links of llms.txt, and those between the copies, follow. It then links every page to its copy:

- in the page's <head>, with <link rel="alternate" type="text/markdown" href="intro.md">, which is how AI assistants find the Markdown version of a page;
- under the page's title, with a "View as Markdown" link, for readers. To hide it, add .biocbook-md-link { display: no to the book's inst/assets/book.scss.

Unless a site holds a .nojekyll file, GitHub Pages runs Jekyll on it, which turns every .md file into an HTML page of the same name: the rendering of <page>.md would then collide with the book's own <page>.html. The biocbook workflow of books created from template 1.1.3 or later writes .nojekyll at the root of the site. In older books, add touch llms-index/.nojekyll to the "Index the llms.txt of every deployed version" step of .github/workflows/biocbook.yml, right after mkdir -p llms-index.

Examples

```
## In practice, `enrich_llms_txt()` runs after every render of the book,
## from `inst/assets/post-render.R`
unique_id <- as.numeric(Sys.time())
bookname <- paste(Sys.info()[['sysname']], unique_id, sep = '.')
quick_init(bookname, user = "dummy")
docs <- file.path(bookname, "inst", "docs")
dir.create(docs)
enrich_llms_txt(docs, project_dir = file.path(bookname, "inst"))
cat(readLines(file.path(docs, "llms.txt")), sep = "\n")
unlink(bookname, recursive = TRUE)
```

micromamba *Executing* python *code in a BiocBook*

Description

BiocBook pages can execute python code as well as R code. The code is executed on every render, including with GHA or when the *Bioconductor Build System* (BBS) rebuilds the book: R CMD build runs vignettes/Makefile, which runs quarto render.

The consequence is that the python packages a book uses must be installed at build time, wherever the book is built. setup_python() does that from within the book itself, building the conda environment declared in inst/requirements.yml (including its pip: section, if any), so the book carries its own python environment rather than relying on one being present.

- setup_python(): provision and activate the book's conda environment from inst/requirements.yml, and put its command-line tools on the PATH. Call it from any page that needs python.
- python_envs(): list the conda environments setup_python() has cached on this machine, and remove the ones that are no longer needed.
- micromamba(): path to the micromamba binary the book provisions with, downloading a pinned, checksummed copy on first use if none is present.

micromamba is the only provisioning dependency, and it is a single self-contained binary: no conda installation, no base environment, no python. micromamba() looks for one in RETICULATE_CONDA, then on the PATH, then in BiocBook's cache, and only then downloads it. On a build machine that has no conda at all (which includes the r-universe build image Bioconductor is migrating to), it is what makes the book buildable. The book's Docker image downloads it while the image is built, and keeps it, together with the book's environment.

Usage

```
micromamba(version = .micromamba_version, quiet = FALSE)

setup_python(
  requirements = NULL,
  envname = NULL,
  prefix = NULL,
  conda = micromamba()
)

python_envs(remove = NULL)
```

Arguments

version	Pinned micromamba release to use, as "<version>-<build>". Bump it here rather than tracking latest.
quiet	Whether to suppress progress messages.

requirements	Path to a conda environment file, which may hold a pip: section (see "Python libraries and the R session"). If NULL, inst/requirements.yml is located by walking up from the working directory to the folder holding _quarto.yml, which works both while quarto renders a page and from a normal R session.
envname	Name of the conda environment to create or re-use. If NULL, the name: declared in the environment file is used.
prefix	Full path of the conda environment. If NULL, a path under BiocBook's cache is derived from envname and from a hash of the requirements file (see "Cached environments"). Environments are always addressed by path rather than by name, since several conda root prefixes may each hold an environment of the same name.
conda	Path to the conda, mamba or micromamba binary to provision with. Defaults to micromamba(), which finds or fetches one.
remove	Names of cached environments to delete, as listed by python_envs(). A name without its hash suffix (e.g. "BiocBook") matches every environment created from a requirements.yml declaring that name:, and "all" deletes every cached environment.

Value

setup_python() invisibly returns the path of the environment it activated, or of the interpreter that was already configured. python_envs() returns a tibble with one row per cached environment (name, path, size_mb, modified), or, when remove is set, the rows of the environments it deleted, invisibly. add_python_chapter() invisibly returns book.

Cached environments

Environments live under BiocBook's cache (tools::R_user_dir("BiocBook", "cache")), in a folder named after the name: declared in requirements.yml followed by a short hash of the file itself, e.g. envs/BiocBook-3f2a9c1b7d4e. An environment is therefore re-used for as long as requirements.yml is unchanged: editing the file (adding, removing or re-pinning a package) gives a new environment on the next render, and two books only ever share an environment when they declare exactly the same thing.

Older environments are not deleted automatically, since another book on the same machine may still use them. python_envs() lists them with their size, and python_envs(remove = ...) deletes them.

Engines

quarto binds an execution engine per file. A page holding at least one R chunk uses knitr, and its python chunks then run through reticulate in a single session shared with R. A page with no R chunk at all uses jupyter instead.

Prefer knitr. It is the only way to share objects between R and python, it keeps the book's house style (collapse, comment, fig.align from inst/assets/_knitr.yml) and code-link, which apply to knitr pages only – and, decisively, the Bioconductor builders provide python3 but not jupyter, so a jupyter page cannot be rendered there at all.

Python libraries and the R session

python chunks run through reticulate, inside the R process that renders the page. A process holds a single copy of each shared library name: once R has loaded one, e.g. the system's libssl.so.3 through an R package using OpenSSL, a python package needing a library of the same name gets that copy, whatever version it was built against. conda packages link such libraries under their usual names, and are usually built against more recent versions than a Linux distribution ships, hence errors like these on the Bioconductor builders (Ubuntu 24.04):

- ImportError: ... libssl.so.3: version 'OPENSSL_3.2.0' not found (required by .../libcurl.so.4), importing h5py (and cooler), whose HDF5 links libcurl;
- undefined symbol: hb_ft_font_get_ft_face, importing matplotlib >= 3.11, which links libraqm and therefore harfbuzz.

Which libraries R has loaded first depends on the machine and on the page, so a book can build in its Docker image and fail on the Bioconductor builders, and importing python packages before attaching R packages does not help.

Wheels from PyPI carry their compiled dependencies under names of their own, so they cannot pick up a library R has loaded. Declare the packages that python chunks import in a pip: section of inst/requirements.yml: setup_python() installs them with the environment's own pip, once the conda packages are in place.

```
dependencies:
    - python=3.12
    - samtools=1.24
    - pip:
        - cooler==0.10.4
        - matplotlib==3.11.2
```

- List under pip:, with pinned versions, every package that python chunks import. Their own compiled dependencies (numpy, h5py...) then come from PyPI too. pip does not replace a package that conda has already installed, so do not let a conda package bring them in.
- Keep the conda part for python itself and for command-line tools, e.g. samtools from bioconda. A tool runs in its own process, with its own libraries, so it cannot clash with the R session. setup_python() puts the environment's tools on the PATH, for system2(), bash chunks and python's subprocess.
- Some conda packages bring python libraries along: bioconda's deeptools pulls numpy, matplotlib, pysam and pyBigWig from conda. Install such a package under pip: instead (deeptools is on PyPI).
- A package without a wheel for the build machine (Linux x86_64 for the Bioconductor builders) is compiled from source, against the libraries of the environment, and can clash again.

Examples

```
## In practice, you should not use `.local` argument.
unique_id <- as.numeric(Sys.time())
bookname <- paste(Sys.info()[['sysname']], unique_id, sep = '.')
bb <- init(bookname, .local = TRUE)
add_preamble(bb, open = FALSE)
```

```
add_chapter(bb, title = "Chapitre Un", open = FALSE)
add_python_chapter(bb, title = "Chapitre Deux", open = FALSE)
unlink(bookname, recursive = TRUE)

## `conda` environments cached on this machine
python_envs()
```

Index

add_chapter *(*BiocBook-editing*)*, 8
add_preamble *(*BiocBook-editing*)*, 8
add_python_chapter *(*BiocBook-editing*)*, 8

BiocBook, 2, *7, 8*, *10*
BiocBook-bookdown, 5
BiocBook-class *(*BiocBook*)*, 2
BiocBook-editing, 8
BiocBook-llms, 10
BiocBook-python *(*micromamba*)*, 12

chapters *(*BiocBook*)*, 2
chapters,BiocBook-method *(*BiocBook*)*, 2
check_deps *(*BiocBook-editing*)*, 8

edit_bib *(*BiocBook-editing*)*, 8
edit_css *(*BiocBook-editing*)*, 8
edit_page *(*BiocBook-editing*)*, 8
edit_requirements_yml
        *(*BiocBook-editing*)*, 8
edit_yml *(*BiocBook-editing*)*, 8
enrich_llms_txt *(*BiocBook-llms*)*, 10

from_bookdown *(*BiocBook-bookdown*)*, 5

init *(*BiocBook*)*, 2

micromamba, 12

path,BiocBook-method *(*BiocBook*)*, 2
preview *(*BiocBook-editing*)*, 8
publish *(*BiocBook-editing*)*, 8
python_envs *(*micromamba*)*, 12

quick_init *(*BiocBook*)*, 2

releases *(*BiocBook*)*, 2
releases,BiocBook-method *(*BiocBook*)*, 2

setup_python *(*micromamba*)*, 12
show,BiocBook-method *(*BiocBook*)*, 2
status *(*BiocBook-editing*)*, 8