R Workflow

Summer Semester 2023

Daniela Palleschi

Invalid Date

Table of contents

1 Preface

This book contains my tips-for-future-me regarding how I like to build and maintain my workflow with R. It began as a series of Quarto docs I kept in an RProject for me to refer back to whenever I forgot how I had done something or which git command I used to achieve something. It is informal and likely flawed. I make an effort to provide links to source of important info where relevant.

These documents were originally intended to be privately used. I'll slowly brush them up to improve readibly/quality, but this is definitely a work-in-progress.

2 Troubleshooting

```
knitr::opts_chunk$set(eval = F, # change this to 'eval = T' to reproduce the analyses; mak
                      echo = T, # 'print code chunk?'
                      message = F, # 'print messages (e.g., warnings)?'
                      error = F,
                      warning = F)
```

```
library(knitr)
hook_output = knit_hooks$get('output')
knit_hooks$set(output = function(x, options) {
  # this hook is used only when the linewidth option is not NULL
  if (!is.null(n <- options$linewidth)) {
    x = xfun::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n)) x = strwrap(x, width = n)
    x = paste(x, collapse = '\n')
  }
  hook_output(x, options)
})
```

In this document I record errors that I encounter and how I resolved them. Solutions will typically be from Googling, relevant links will be provided.

3 Rmarkdown knitting PDF

Error message:

```
tlmgr update --all --self
fmtutil [ERROR]: format directory not writable: /usr/local/texlive/2019/texmf-var/web2c
tlmgr: package repository https://mirror.clientvps.com/CTAN/systems/texlive/tlnet (not ver
tlmgr install: package already present: l3kernel
! LaTeX Error: Mismatched LaTeX support files detected.
(LaTeX)        Loading 'expl3.sty' aborted!
(LaTeX)
(LaTeX)        The L3 programming layer in the LaTeX format
(LaTeX)        is dated 2023-02-01, but in your TeX tree the files require
(LaTeX)        at least 2023-02-02.

Error: LaTeX failed to compile 03-exp123.tex. See https://yihui.org/tinytex/r/#debugging f
In addition: Warning messages:
1: Removed 156 rows containing non-finite values (stat_ydensity).
2: Removed 156 rows containing non-finite values (stat_boxplot).
3: Removed 156 rows containing non-finite values (stat_summary).
4: Removed 156 rows containing non-finite values (stat_summary).
5: Removed 156 rows containing non-finite values (stat_ydensity).
6: Removed 156 rows containing non-finite values (stat_boxplot).
7: Removed 156 rows containing non-finite values (stat_summary).
8: Removed 156 rows containing non-finite values (stat_summary).
Execution halted
```

In the Terminal, I type

```
kpsewhich --all --engine=pdftex pdflatex.fmt
```

And get:

```
(base) administrators-MacBook-3:Lifetime Project danielapalleschi$ kpsewhich --all --engin
/usr/local/texlive/2019/texmf-var/web2c/pdftex/pdflatex.fmt
```

So my pdftex version is from 2019 or soemthing.

In the Terminal, I typed:

```
sudo fmtutil-sys --all
```

And entered my laptop's administrator Password when prompted (it won't show that you're typing, but just type the password and hit Enter). Then Terminal was running for a couple of minutes, and ended with the output:

```
Transcript written on mex.log.
fmtutil [INFO]: /usr/local/texlive/2019/texmf-var/web2c/pdftex/mex.fmt installed.
fmtutil [INFO]: Disabled formats: 6
fmtutil [INFO]: Successfully rebuilt formats: 45
fmtutil [INFO]: Total formats: 51
fmtutil [INFO]: exiting with status 0
```

When I went to knit a PDF I still got the same errors.

Then I ran

```
fmtutil-user --all
```

No password required, it ran again for a few minutes. At the end, I got

```
Transcript written on mf.log.
fmtutil [INFO]: /Users/danielapalleschi/Library/texlive/2019/texmf-var/web2c/metafont/mf.b
fmtutil [INFO]: Disabled formats: 6
fmtutil [INFO]: Successfully rebuilt formats: 45
fmtutil [INFO]: Total formats: 51
fmtutil [INFO]: exiting with status 0
```

3.1 Solution

In the end I went into my Time Machine (external harddrive) to a point right before this problem started happening, and restored the 'Macintosh HD/Library' folder, because presumably something happened to the TexLive file or something in there. That solved the problem.

4 Rmarkdown and kableExtra with R version 4.3

5 Stargazer

5.1 Solution

6 31.08.2022: updated to R version 4.2.1 (2022-06-23), now stargazer won't compile tables where model names are too long

7 Solution: discussed here https://www.reddit.com/r/rstats/comments/ucmtdn

8 code below from linked solution https://gist.github.com/alexeyknorre/b0780836f4cec

8.1 Quick fix for stargazer <= 5.2.3 is.na() issue with long model names in R >= 4.2

9 Unload stargazer if loaded

detach(“package:stargazer”,unload=T) # Delete it remove.packages(“stargazer”) # Download the source download.file(“https://cran.r-project.org/src/contrib/stargazer_5.2.3.tar.gz”, destfile = “stargazer_5.2.3.tar.gz”) # Unpack untar(“stargazer_5.2.3.tar.gz”) # Read the sourcefile with .inside.bracket fun stargazer_src <- readLines(“stargazer/R/stargazer-internal.R”) # Move the length check 5 lines up so it precedes is.na(.) stargazer_src[1990] <- stargazer_src[1995] stargazer_src[1995] <- “ ” # Save back writeLines(stargazer_src, con=“stargazer/R/stargazer-internal.R”) # Compile and install the patched package install.packages(“stargazer”, repos = NULL, type=“source”) ### FROM NOW ON: shorter model names for stargazer

‘

10 Git set-up

For a reproducible workflow

11 How to follow

All code should be added into the ***Terminal*** tab (in the Console pane in RStudio). Alternatively, you don't need to use the Terminal at all, as RStudio adds a Git tab to the Environment pane. For a much more in-depth (and better) guide, follow the amazing Happy git with R book by the even more amazing Jenny Bryan.

I use git right in the Terminal window in RStudio. To get to the Terminal window, look where you usually see the output from your code (probably the bottom left window, 'Console'). You should see a tab 'Terminal'. This is where you want to go. Now you can start setting up git using the following commands.

For the purposes of this guideline, I'll be using my 'dissertation' folder as an example, which contains all the files and folders necessary to knit my dissertation with oxforddown.

All code chunks below contain copy-and-pasted commands and output from my Terminal, so you can see exactly how it should look.

12 Summary of commands

Table 12.1: Summary of useful git commands to get started

command	function
git init	initiate git
git add folder/	stage/add a folder and its contents
git add filename.ext	stage/add a file
git add -u	stage all modified but unstaged files
git commit -m ”message”	commit the changes to local git
git push	push current state of local git to remote repo
git remote -v	print remote fetch/pull URLs
git reset	unstage all
git checkout	remove unstaged changes
git log –oneline	List the unique hash for each commit in a repo
git revert 2f5451f –no-edit	undo a commit with the hash 2f5451f (replace this with the relevant commit has
git reset –soft HEAD~	undo last commit (that was not pushed)
git reset –soft HEAD~2	undo last 2 commits (that were not pushed)

13 Local repositories

Local repositories are those that are stored on your machine (computer). They're great for tracking changes you make along the way, but if your machine is lost/stolen/damaged, your local repository goes along with it. It's a great idea to get familiar with git in local contexts, but remote repositories are the real life savers (next section).

13.1 Setting up git

13.1.1 Install git

You'll first need to already have git installed locally. To see how to do that, go here: https://git-scm.com/download/. If you don't know whether you already have git installed, use the code in the section below (check git version).

13.1.2 Check git version

`git version` = check your version of git

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git version # this is where
git version 2.32.1 (Apple Git-133) # and this was the output
```

`git config --list` = check your configurations

`git config --global user.name "your-name"` = globally set your name `git config --global user.email "your-email"` = globally set your email `pwd` = print working directory `cd` = change directory

```
pwd
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation
```

```
cd dissertation
# no output
```

```
pwd
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation/d
```

```
## What the Terminal should look like after you've run all 3 commands
(base) administrators-MacBook-3:Dissertation danielapalleschi$ pwd # command 1
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation
(base) administrators-MacBook-3:Dissertation danielapalleschi$ cd dissertation # command 2
(base) administrators-MacBook-3:dissertation danielapalleschi$ pwd # command 3
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation/d
(base) administrators-MacBook-3:dissertation danielapalleschi$
```

13.2 Adding git repo

`git status` = check status of git repo `git init` = put a git repo in to track changes `git add` = add files/folders to track

`git status`

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git status # command
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../../../.DS_Store
        ../../../.gitignore
        ../../
        ../../../Participant Money/
        ../../../R template scripts/
        ../../../Stats/
        ../../../Talks/
        ../../../presupposition.docx

nothing added to commit but untracked files present (use "git add" to track)
(base) administrators-MacBook-3:dissertation danielapalleschi$
```

`git init`

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
```

Hmm, GitHub recently changed 'master' to 'main'. Let's do the same, first globally but also locally:

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git config --global init.de
(base) administrators-MacBook-3:dissertation danielapalleschi$ git branch -m main
```

Now let's check the status of our repo:

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git status
On branch main # So changing 'master' to 'main' worked :)

No commits yet

Untracked files: # list of all the files to add...
  (use "git add <file>..." to include in what will be committed)
        .github/
        .gitignore
        00-introduction.Rmd
        00-introduction.tex
        01-how-to-use.tex
        01-processing.Rmd
        01-processing.pdf
        01-processing.tex
        02-tense-lifetime.Rmd
        02-tense-lifetime.pdf
        02-tense-lifetime.tex
        03-exp123.Rmd
        03-exp123.log
        03-exp123.maf
```

```
        03-exp123.mtc
        03-exp123.mtc0
        03-exp123.pdf
        03-exp123.tex
        03-rmd-basics-cites-and-refs.tex
        04-exp345.pdf
        04-exp345.tex
        04a-exp4-context.Rmd
        04b-exp5-world.Rmd
        04c-exp6-ps-replication.Rmd
        05-dfg.Rmd
        05-dfg.pdf
        05-dfg.tex
        06-conclusion.Rmd
        LICENSE
        README.md
        _bookdown.yml
        bibliography/
        data/
        docs/
        figures/
        front-and-back-matter/
        index.Rmd
        scripts_and_filters/
        templates/

nothing added to commit but untracked files present (use "git add" to track)
```

For now, I want to commit the chapters I'm working on. So I'll add them.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 03-exp123.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 04a-exp4-context.Rm
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 04b-exp5-world.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 04c-exp6-ps-replica
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 05-dfg.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 01-processing.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 02-tense-lifetime.R
base) administrators-MacBook-3:dissertation danielapalleschi$ git status # now check the s
On branch main

No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   01-processing.Rmd
        new file:   02-tense-lifetime.Rmd
        new file:   03-exp123.Rmd
        new file:   04a-exp4-context.Rmd
        new file:   04b-exp5-world.Rmd
        new file:   04c-exp6-ps-replication.Rmd
        new file:   05-dfg.Rmd

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .github/
        .gitignore
        00-introduction.Rmd
        00-introduction.tex
        01-how-to-use.tex
        01-processing.pdf
        01-processing.tex
        02-tense-lifetime.pdf
        02-tense-lifetime.tex
        03-exp123.log
        03-exp123.maf
        03-exp123.mtc
        03-exp123.mtc0
        03-exp123.pdf
        03-exp123.tex
        03-rmd-basics-cites-and-refs.tex
        04-exp345.pdf
        04-exp345.tex
        05-dfg.pdf
        05-dfg.tex
        06-conclusion.Rmd
        LICENSE
        README.md
        _bookdown.yml
        bibliography/
        data/
        docs/
        figures/
        front-and-back-matter/
        index.Rmd
```

```
        scripts_and_filters/
        templates/
```

13.3 Tracking changes

`git diff` = Show unstaged changes between your index and working directory `git commit -m "<message>"` = commit the changes, with the following message `git log` = Display the entire commit history using the default format

First, I made some changes: Rename 'Interpretation' to 'Discussion' and remove the 'save.image…' code chunk from the end of `04a-exp4-context.Rmd`.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git diff # command
diff --git a/04a-exp4-context.Rmd b/04a-exp4-context.Rmd
index ab6a2bb..a883ab4 100644
--- a/04a-exp4-context.Rmd
+++ b/04a-exp4-context.Rmd
```

Commit this change

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git commit -m "Update Conte
[main (root-commit) dabe29e] Update Context Exp headers
 7 files changed, 10363 insertions(+)
 create mode 100644 01-processing.Rmd
 create mode 100644 02-tense-lifetime.Rmd
 create mode 100644 03-exp123.Rmd
 create mode 100644 04a-exp4-context.Rmd
 create mode 100644 04b-exp5-world.Rmd
 create mode 100644 04c-exp6-ps-replication.Rmd
 create mode 100644 05-dfg.Rmd
```

Check the log

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git log
commit dabe29e9d594c578971e07e09e1b57c89572b582 (HEAD -> main)
Author: Daniela Palleschi <palleschi.daniela@gmail.com>
Date:   Fri Jul 1 14:16:26 2022 +0200

    Update Context Exp headers
```

But this updated 'main'. Let's try to only update the relevant file.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git commit 04a-exp4-context
[main 85468f5] Updated Context Exp headers
 1 file changed, 1 insertion(+), 4 deletions(-)
```

Good. Now check log.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git log
commit 85468f585608b5c8d94d4b9af40898c5a961110c (HEAD -> main)
Author: Daniela Palleschi <palleschi.daniela@gmail.com>
Date:   Fri Jul 1 14:18:08 2022 +0200

    Updated Context Exp headers

commit dabe29e9d594c578971e07e09e1b57c89572b582
Author: Daniela Palleschi <palleschi.daniela@gmail.com>
Date:   Fri Jul 1 14:16:26 2022 +0200

    Update Context Exp headers
```

Stage all modified files.

```
git add -u
```

14 Remote repositories with RStudio

14.1 Taken for granted

From this point on, I take for granted:

- git is already installed on my machine
- I have a GitHub account
- already got RStudio set up with an SSH key (Tools > Global Options > Git).
  - with an SSH key, I should always make sure to copy the SSH url and not the HTTPS (which would require a Personal Access Token (PAT))!

If these steps haven't already been taken, check out the first chapter from https://happygitwithr.com/index.html

If you've done all that, follow along from here (following https://happygitwithr.com/rstudio-git-github.html#rstudio-git-github)

14.2 Setting up

Steps:

1. Create a repo in GitHub

- make sure to click 'create README.md'

2. Create a new RProject with Version Control

- add the URL from your GitHub repo, press the green 'Clone <>' button
- copy the url, but make sure you choose the relevant format! SSH if you've got an SSH key set up, or HTTPS if you're using a personal access token (PAT)

3. stage (add), commit, some arbitray changes

- using either the Git tab, or preferably using the Terminal (see below for examples)

```
# add a line to your README
echo "This is a line from RStudio" >> README.md
```

```
# check the status
git status
# stage the change
git add README.md
# commit the change and add a message
git commit -m "first commit from RStudio"
# push the change to GitHub
git push
# now, go refresh your GH repo and see the change
```

14.3 Other commands

- check working directory

```
# print working directory; should be to your project folder
pwd
```

- change working directory

```
# change directory, in case not your project folder
cd
```

- list all files in working directory

```
# list all files in WD
ls
```

- list all files in working directory, including ignored ones

```
# list all files including ignored ones
ls -a
```

- check git status (what's been staged, what not)

```
# check status
status
```

- add files to .gitignore (so they won't be monitored)

```
# add file/folder to gitignore (so git will ignore it)
echo Slides/ >> .gitignore
```

- check remote repo URL

```
# check what your remote repo URL is
git remote -v
```

14.4 Keeping up changes

If you've already started working on the project locally, you can just drag-and-drop your files/folders into the Terminal

```
# stage change (need to do it every time)
add filename.Rmd
# commit these changes to git
commit -m "message about commit"
# check the status
status
# push these changes to remote repo
push
```

Alternatively, you don't need to use the Terminal at all (but it's good practice to know how to in order to understand the logic/workflow). In the 'Environment' pane, you should see a 'Git' tab (if git is indeed set up). You can tick the 'Staged' button for all files whose changes you want to commit. Then click 'commit'. A message window will come up, add your message and click 'Commit'. Then, Click 'Push' to push to GH.

15 Git for collaboration

If you've got a git repo set up and want to add a member:

In a GitLab repo: + Project Information > Members > Invite members

16 New laptop or expired SSH key

If you have everything backed up, and are running your same projects etc. on a new machine (or if your SSH key simply expired), then you just need to generate a new SSH key in RStudio. Then, go to your GitHub or GitLab Settings, go to the tab for SSH keys, and add a the new SSH key. In other words, follow these steps:

1. **Create SSH key** (in RStudio) and save to clipboard (e.g., CTRL+C)

- RStudio > Tools > Global Options > Git/SVN > SSH key > Create SSH Key… > OK > View public key; copy key

2. **Add SSH key** to you GitHub and/or GitLab account

- GitHub > Settings > SSH and GPG keys > New SSH key; paste key
- GitLab > Preferences > SSH keys > Key / title etc.; paste key

17 Quarto

18 Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

18.1 Quarto versus Rmarkdown

Important differences I've noticed:

- `YAML`:
  - Rmd `output` = `format` in Quarto
  - Rmd `html_document` = `html` in Quarto

19 R Packages for a Reproducibile Workflow

20 `here` package

- uses file path relative to the current project

```
# install here package
install.packages("here")

# read in a csv file
readr::read_csv(here::here("folder", "subfolder", "subsubfolder", "file.csv"))
# or
readr::read_csv(here::here("folder/subfolder/subsubfolder/file.csv"))
```

21 `pacman` package

- `p_load()` function checks if listed packages are already installed
  - if yes, loads them (as with `library()`)
  - if no, installs and then lodas them (as long as they're CRAN packages!)

```
# install pacman
install.packages("pacman")

# install and/or load packages
pacman::p_load(here,
               renv,
               kableExtra)
```

22 `renv` package

The `renv` packages stores your project-local environment. That is, it creates a time capsule of the R and package versions that you use *within an R Project*.

The benefits:

- your output is not dependent on whichever project version you currently have stored on your machine
  - e.g., if you come back to an analysis after a long time, the output should still be the same even if a certain package version has deprecated a function your analysis uses.
- makes the project *reproducible* for collaborators as well, who will likely have some different package versions (or will have missing packages) on their machine

The hard-to-get-your-head-around (imo):

- with each new R Project with a renv.lock file, you need to install all the necessary packages again (even if they're on your machine)
  - this is because each `renv` doesn't look beyond your project folder!
- remembering to update the `renv.lock` file frequently!

> ⚠ Running `renv` functions
>
> Remember to always run `renv` functions ***in the console***! You do not want e.g., `renv` initialising a new `renv.lock` file every time you render your documents.

22.1 Activate or init

- to initialise a new *project-local environment*:

```
renv::init()
```

22.2 Snapshot

- update `renv.lock` file

```
renv::snapshot()
```

After updating the `renv.lock` file, remember to commit/push these changes to git (if you're using it)!

> Version control
>
> After updating the `renv.lock` file (i.e., running `renv::snapshot()`), remember to commit/push these changes to git (if you're using git)! Otherwise, your `renv.lock` file will be outdated compared to your output.

22.3 Restore

- will restore your project to the most recent `renv.lock` file versions
  - this step follows `snapshot()`, which updates the `renv.lock` file
- very useful after updating R!

```
renv::restore()
```

22.4 Upgrade

- to update the `renv` package:

```
# upgrade renv version
renv::upgrade()
```

22.5 Update and Hydrate

```
# updates all packages (stored in renv.lock) in the renv cache
renv::hydrate(update = "all")

# update should have no effect now, but doesn't hurt to check
renv::update()

# now take a snapshot with the updated packages
renv::snapshot()
```

Version control (repeated)

After updating the `renv.lock` file (i.e., running `renv::snapshot()`), remember to commit/push these changes to git (if you're using git)! Otherwise, your `renv.lock` file will be outdated compared to your output.

23 UpdateR and (re-)Install Packages

This script is to be run whenever you want to update R and/or install your required packages. Whenever you install a new package, try to remember to add it to this file. This way, whenver you update, you can just run this script and will then have all your required packages ready to go. Newer versions of RStudio automatically check for missing packages when opening a script, so forgetting to install CRAN packages will be less detrimental to your workflow.

24 Check for R update: `updateR`

Do you need to update R? If using a Mac, you can do this by using the `updateR` package (some steps found here):

```
# install devtools if you don't have it already
install.packages('devtools')
```

```
# install updateR
devtools::install_github('andreacirilloac/updateR')
```

```
# run; you will be prompted for admin password
updateR::updateR()
```

`updateR` macOS compatibility

As of mid-2023, `updateR()` spits out an error after entering your password:

```
Updated ~/.Rprofile
Password:Error in if (!compactible) stop(sprintf(e, status$latest, macOS), call. = FALSE
  argument is of length zero
```

This seems to have something to do with the newest Mac OS (see: discussion on the package GitHub). For now, manual update only directly from CRAN.

25 Check for R update on Windows: `installR`

The `installR` package also has an `updateR()` function which only works on Windows. You will need to run the function in directly in R, *not* RStudio!

```
# install installr
install.packages("installr")
```

```
# run; you will be prompted for admin password
installr::updateR()
```

26 Global options

My preferred global options (RStudio > Tools > Global options):

- General > Basic
  - **R Sessions**
    - uncheck 'Restore previously open source documents at startup'
  - **Workspace** (for reproducibile workflow!!!)
    - uncheck 'Restore .RData into workshapce at startup'
    - Save workspace to .RData on exit: ***Never***
- Code > Display
  - General
    - Show whitespace characters
    - Allow scroll past end of document
    - highlight selected line
- Appearance
  - Editor theme: Cobalt

27 Install packages

Packages can be available on CRAN or through developer versions.

27.1 CRAN packages

First make a list of your packages available on CRAN.

```
# CRAN packages ####

# run THIS chunk after updating R/RStudio
# from 'install_packages_if_missing.R'; roughly alphabetical
required_packages <- c("binom", "bookdown", "broman", "citr", "dplyr", "doBy",
                       "beepr", # to play sounds
                       "emmeans", "EMAtools", "grid",
                       "ggplot2", "ggdark", "ggstatplot",
                       "ggpp", # grammar extension of ggplot2
                       "ggpubr",
                       "ggrain",
                       "formatR", "janitor",
                       "here", "knitr", "kableExtra",  "lme4", "lmerTest",
                       "png", "pryr", "papaja", "performance",
                       "tidyverse", "Rmpfr", "rmarkdown", "rotations",
                       "quarto",
"starpolishr", "stargazer", "simr",
                       "xtable")
```

Then, run this list through a for-loop that checks whether you've got these packages installed already. If not, the package will be installed.

```
for (package in required_packages) {
  print(paste0("checking for install of ", package))
  if (!requireNamespace(package)) install.packages(package, repos = "http://cran.rstudio.c
}
```

27.2 Install dev packages: `devtools` and `remotes`

Now install the packages that aren't on CRAN. First, install `devtools` and `remotes` if you don't already have them installed.

```
# Install remotes package if necessary
if(!requireNamespace("devtools", quietly = TRUE)) install.packages("devtools")
```

```
# Install remotes package if necessary
if(!requireNamespace("remotes", quietly = TRUE)) install.packages("remotes")
```

27.2.1 `ggrain` package

For raincloud plots geom

```
if (!require(remotes)) {
    install.packages("remotes")
}
remotes::install_github('jorvlan/raincloudplots')

library(raincloudplots)
```

27.2.2 `papaja` package

- APA formatted templates

```
# download developer papaja (code https://github.com/crsh/papaja)

# Install the stable development version from GitHub
remotes::install_github("crsh/papaja")

# Install the latest development snapshot from GitHub
remotes::install_github("crsh/papaja@devel")
```

27.2.3 `rbbt` package

- integrates Zotero with RStudio

```
# Install rbbt Addin from GitHub to use Zotero
remotes::install_github("paleolimbot/rbbt")
```

To set-up a citation entry shortcut: RStudio > Tools > Modify Keyboard Shortcuts > enter 'Zotero' in the search > choose 'Insert Zotero Citations' > add a shortcut (I like Ctrl+K)

27.2.4 `brms` package

- for running Bayesian models

```
# brms packages ####

# From https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
# run the next line if you already have rstan installed
# remove.packages(c("StanHeaders", "rstan"))
install.packages("rstan", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
```

27.2.5 `cmdstanr` package

- for running Bayesian models

```
# and cmdstanr (https://mc-stan.org/cmdstanr/)
remotes::install_github("stan-dev/cmdstanr")
# or
# install cmdstanr
cmdstanr::install_cmdstan(cores = parallel::detectCores(), overwrite = TRUE)
```

27.2.6 `bcogsci` package

- companion package for the textbook An Introduction to Bayesian Data Analysis for Cognitive Science

```
# run the nextline if you don't have 'devtools' already installed
# install.packages("devtools")
devtools::install_github("bnicenboim/bcogsci")
```

27.3 Optional developer packages

- these only need to be installed if specifically wanted

27.3.1 `stargazer` package

- these steps were taken when updatin to R v.4.2.1 as a work around to a problem that cropped up; only run this chunk if you start having problems with stargazer
- as of March 14, 2023 it's still needed

```
# 31.08.2022: updated to R version 4.2.1 (2022-06-23), now stargazer won't compile tables
# Solution: discussed here https://www.reddit.com/r/rstats/comments/ucmtdn/issue_with_star
# code below from linked solution https://gist.github.com/alexeyknorre/b0780836f4cec04d41a
## Quick fix for stargazer <= 5.2.3 is.na() issue with long model names in R >= 4.2
# Unload stargazer if loaded
detach("package:stargazer",unload=T)
# Delete it
remove.packages("stargazer")
# Download the source
download.file("https://cran.r-project.org/src/contrib/stargazer_5.2.3.tar.gz", destfile =
# Unpack
untar("stargazer_5.2.3.tar.gz")
# Read the sourcefile with .inside.bracket fun
stargazer_src <- readLines("stargazer/R/stargazer-internal.R")
# Move the length check 5 lines up so it precedes is.na(.)
stargazer_src[1990] <- stargazer_src[1995]
stargazer_src[1995] <- ""
# Save back
writeLines(stargazer_src, con="stargazer/R/stargazer-internal.R")
# Compile and install the patched package
install.packages("stargazer", repos = NULL, type="source")
### FROM NOW ON: shorter model names for stargazer
```

27.3.2 PsyTeachR `Introdataviz`

- can produce violin plots and raincloud plots

```
# how to install the introdataviz package to get split and half violin plots
devtools::install_github("psyteachr/introdataviz", dependencies = TRUE)
```

27.3.3 `starpolishr` package

- availabile on github
- post-polishing of stargazer tables

```
# install.packages("devtools")
devtools::install_github("ChandlerLutz/starpolishr")
```

27.3.4 CogSci paper template

- used to write CogSci conference proceedings papers

```
# Install CogSci paper template
devtools::install_github("kemacdonald/cogsci2016")
```

27.3.5 `BRRR` package

- play sounds from different rappers

```
if(!require(devtools)) {install.packages("devtools")}
devtools::install_github("brooke-watson/BRRR")
library("BRRR")
skrrrahh("kendrick")
```

28 Loading packages

Moving forward, use the following code at the beginning of your scripts instead of the long list of `library(package)`. This will also check whether packages are installed, if they are it'll load them, and if they're not it'll install them and then load them.

```
# don't use scientific notation
options(scipen=999)

# copy this chunk at the beginning of new chapters; will automatically load packages and i

## First specify the packages of interest
packages <- c("here", "tidyverse", "dplyr", "formatR", "stringr",
              "janitor", "dplyr", "ggplot2", "lmerTest", "stargazer",
              "MASS", "afex", "knitr", "gridExtra", "grid",
              "paletteer",
              "remotes",
              "Rmisc")

## Now load or install&load all
package.check <- lapply(
  packages,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      install.packages(x, dependencies = TRUE)
      library(x, character.only = TRUE)
    }
  }
)
```

29 Manual installs

29.1 tinyTex

- to render documents with LaTeX under the hood, run the following in the *terminal*: `quarto install tinytex`

29.2 Citations with Zotero

To use the `rbbt` package installed above, we need to have Zotero installed (and Better BibTex).

29.2.1 install Zotero

https://www.zotero.org/download/

29.2.2 Better BibTex

Install from the [website]{https://retorque.re/zotero-better-bibtex/installation/}

- then in Zotero: Tools / Add-ins / Settings wheel / choose downloaded file / Restart Zotero
- set your citation keys: Zotero > Settings > Better BibTex > Citation key formula > (I like "zotero.clean")
  - if you need to update the BibTex key in your docs:
    - highlight all your files in the Zotero Library, right-click > Better BibTex > Refresh BibTex key

30 Zotero

For a reproducible workflow

31 Zotero group library

1. Go to Zotero website
2. Create a group library
3. On main Zotero webpage > Groups > Manage members > Send More Invitations

32 Update citation keys

33 Docker

For a reproducible workflow

34 Docker

Text from freecodecamp.org.

> Docker is a free software developed by Docker Inc. It was presented to the general public on March 13, 2013 and has become since that day a must in the world of IT development. It allows users to create independent and isolated environments to launch and deploy its applications. These environments are then called containers. This will let the developer run a container on any machine. As you can see, with Docker, there are no more dependency or compilation problems. All you have to do is launch your container and your application will launch immediately.

35 Install

https://www.docker.com/

35.1 Open Dock

Find Docker on your machine and open it. You can select the recommended settings.

35.2 Set up hub.docker account

You can do that here, or wait to be prompted when you open Docker.