BROOKE ANDERSON

R FOR ENVIRONMENTAL HEALTH RESEARCH

Contents

1

Prerequisites

I have based this workshop on examples for you to try yourself, because you won't be able to learn how to program unless you try it out. I've picked example data that I hope will be interesting to Navy and Marine Corp public health researchers and practitioners. You can download the slides from the workshop by clicking here.

I am assuming that you already have R and RStudio installed on your computer. You may want to check that you have a recent version of both, and if not, update you version before the workshop. Some of the packages and RStudio tools we'll be using will require newer versions of R and RStudio to work. You can run `sessionInfo()` in R to find out the version of R you have installed. Compare this version to the latest R release version listed at the **Comprehensive R Archive Network (CRAN)**[1]

[1] **Comprehensive R Archive Network (CRAN).** ...

To try out the examples, you will also need a bit more set-up:

1. Download git
2. Get a GitHub account
3. Install some R packages
4. Download example R Project

This section will walk you through each step.

1. Download git

In the workshop, you will learn how to use **git**.[^**git.** Open-source version control software ...] To try the examples, you will need to install git to your computer and make sure that your installation of RStudio can find this software, so you can use git for version control for R Projects. ...

2. Get a GitHub account

You'll also learn how to share and collaborate on an R Project using **GitHub**.[2] You will need to get a GitHub account to be able to post repositories on GitHub. ...

[2] **GitHub.** An online platform for directories tracked with the version control software `git`. This platform has become very popular for sharing code projects, as well as collaborating across a team on developing code and software. Other online git platforms exist and are used by some researchers, including **GitLab**. Once you've mastered using GitHub, you should be able to easy transfer those skills to other platforms like GitLab.

3. Install some R packages

This booklet uses a number of R packages beyond base R. To install all the packages that you'll need, run the following code in your version of R:

```
install.packages(c("readr", "ggplot2", "forcats",
    "magrittr", "dplyr", "lubridate", "sf", "tigris",
    "DT", "plotly", "leaflet", "flexdashboard",
    "tidyr", "stringr"))
```

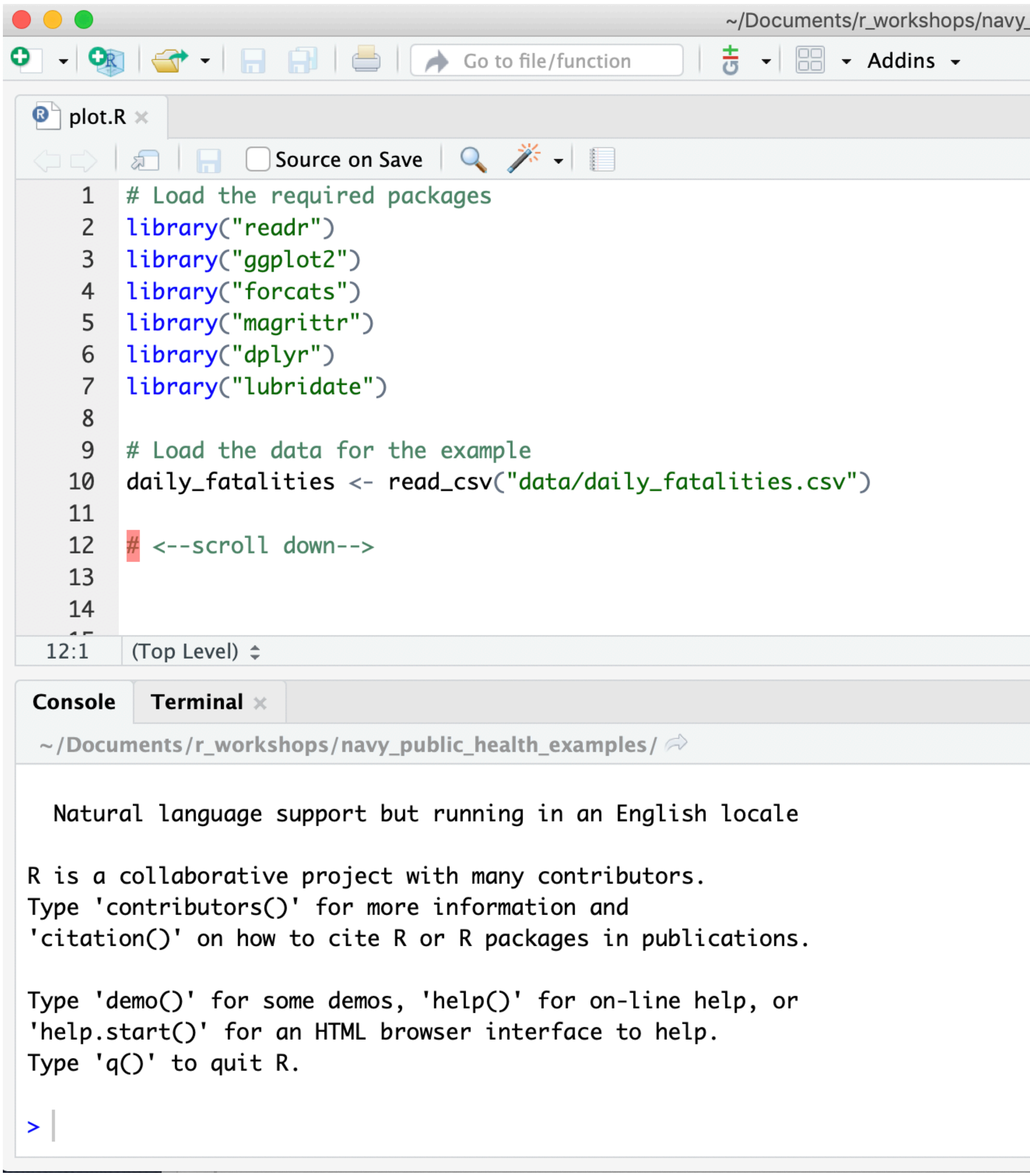
4. Download example R Project

I've created a repository on GitHub. You can find this example repository by clicking here. On the page takes you to, click on the "Clone or download" button and then select "Download ZIP".

This will download a single zipped file to your computer. When you unzip the file, it will be a special type of directory, an R Project directory. To open the R Project and start on the examples, open RStudio, then go to "File" -> "Open Project". A pop-up window will open to let you navigate through your files and find an R Project to open. Navigate to the directory you downloaded, which should be called "columbia_env_health_examples" and doubleclick on the file in this directory called "columbia_env_health_examples.Rproj".

This will open the project. In the "Files" pane of RStudio, you should see some subdirectories for "R" and "data". These have the example R code and data, respectively, for you to try the examples in this booklet. The code in each of the R files should run independently, including the code to load all required packages. Figure 1.1 shows what this package should look like once you've downloaded and opened it, as well as opened the "plot.R" file in the project's "R" subdirectory.

Click on the **Next** button (or navigate using the links at the top of the page) to continue.


~/Documents/r_workshops/navy_
Go to file/function
Addins
plot.R
Source on Save
12:1
(Top Level)
Console
Terminal
~/Documents/r_workshops/navy_public_health_examples/

```r
# Load the required packages
library("readr")
library("ggplot2")
library("forcats")
library("magrittr")
library("dplyr")
library("lubridate")

# Load the data for the example
daily_fatalities <- read_csv("data/daily_fatalities.csv")

# <--scroll down-->
```

```
  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.

>
```

Figure 1.1: What the example R Project for this booklet should look like once you've downloaded and opened it, as well as opened the 'plot.R' file in the project's 'R' subdirectory.

2 *Organize*

If you are using R to write for larger projects, including research for academic papers and theses, you should start putting some thought into how you organize your research files, including raw data, cleaned data, coding scripts for analysis, and output like paper drafts and figures.

2.1 R Projects

RStudio allows you to create "Projects", which help with this organization.

2.2 Directory organization

…

2.2.1 Keeping things tidy

My desk at work is very messy, with lots of paper printouts piled up. My car and my closet aren't terribly tidy, either. But I do keep my project directories very tidy, and I strongly recommend the practice.

This goes beyond "well-organized", which we just covered (putting all project files in one directory, using subdirectories to divide up files in a project, using consistent names for project file directories, etc.). Keeping a project directory "tidy" means having **only one** version of each file. Often, as you develop a project, especially when collaborators are involved, you can end up with many versions of a file. For example, you may have the draft of a journal article's text saved in some versions with the file name reflecting the date of the draft ("draft_may_12.docx"), some versions that include the initials of people who reviewed it ("paper_draft_ba_rp_mb2.docx"), and so on.

This type of organization—having multiple versions of project files, with the file names meant to help you keep track of them—results in very cluttered and hard-to-manage project directories. Instead, at any given moment, each of your project directories should have only one version of each project file. You certainly won't want to lose information from edits and changes to the files along the way, so it's smart to use some type of **version control system**[**version control system.** ...] on each project directory. This will allow you to track the changes you've made to each file and to go back and revisit the file at any moment in the project's history. A later section of this booklet will describe how you can use `git` for version control for R projects.

2.2.2 *Avoiding repetition*

One key to efficient organization is to **avoid repetition**.[1] In practice, it often happens that you're using the same code across several projects. For example, say you have some code that calculates the apparent temperature from air temperature and some measure of dewpoint temperature. If you have organized your project files to have one directory per paper, with all the associated code for a paper within the project's directory, then you may find you're often copying and pasting the code to calculate apparent temperature into different directories.

[1] In programming, you'll often hear this advice as "Do Not Repeat Yourself".

This situation makes for a tricky balance—you want to organize your files and have a separate directory for each project, but cutting and pasting code can be a recide for disaster. Each time you move the code, there's a chance for an error to slip in. Also, what if you want to make a change in the code? Say you hear about a better algorithm for calculating apparent temperature? You will either need to go through all of your projects that use that code and change the code everywhere, or you will have to settle for different projects using different algorithms.

This means it's time to start thinking about writing your own **R package**.[*R package** ...] You do not have to publish every R package you write—it's fine to just use it yourself and not share it more widely. Regardless, a package is the right place to store related code that you use often, as well as documentation (and possibly tests) to go with that code. A later section of this booklet will go over a bit about how to write your own R packages, as well as references for learning more about package development.

2.3 *Learn more*

[R Reproc. Research] is an excellent book with advice on improving the reproducibility of projects using R, including academic research projects.

3 Track

Years ago, I tried to learn to use git version control software with R, and it was a total fail. Maybe it's just me, but I found it really hard to wrap my head around the text-dominated, command-line interface classically used for git. However, RStudio now includes tools that provide a **GUI**-style[1] interface to most of the functionality you'll need from git for R-based projects. I highly recommend trying git by using it through RStudio first, and then once you develop a mental map of what's going on, it's much easier to transfer partially or completely to running git from a shell.

[1] **Graphical User Interface (GUI).** …

In this part, I'll discuss both git (the software that allows you to track changes to your R projects), as well as GitHub (an online platform for sharing and collaborating on version-controlled projects). I'll also discuss, near the end, how to use either a Bash Shell or the `git2r` package to do some one-off git tasks that can't be done directly through the RStudio GUI-style git interface.

3.1 git

3.2 GitHub

3.3 Terminal

I find that, for 90% of what I want to do in git, I can do it through the GUI-style interface RStudio provides for git. The few exceptions include:

- Setting a remote and doing the initial push to that remote
- Reverting a commit
- Creating a new branch
- Merging two branches

For these tasks, I usually open up a **Bash Shell**[2] and run a git command from there. However, there's also a package called `git2r` that lets you run any of these git commands from the R command line, so you could also do all these (fairly rare) tasks from the R console without ever opening a Bash Shell, if you master the `git2r` package.

[2] **Bash Shell.** A … . "Bash" stands for "Bourne-Again Shell", Stephen Bourne, a Unix developer.

3.3.1 Accessing a Bash Shell

3.3.2 `git2r` package

3.4 Learn more

Hadley Wickham's excellent book on R Packages includes a great chapter on Git and GitHub. While the chapter (and book) is focused on R packages specifically, the guidance in this chapter would apply to any R Project directory under version control, whether or not the R Project is for a package. This book is available free online or as a print version through O'Reilly (and carried at many Barnes & Nobles).

If you're using git a lot for R projects, it's helpful to have some resources available with more on using git through a Bash Shell. I like the book Pragmatic Guide to Git by Travis Swicegood for a quick, short reference and Git in Practice by Mike McQuaid if I'm trying to dig a bit deeper and figure out how git works. I have also heard good things about Pro Git by Scott Chacon and Ben Straub, which is available for free online.

StackOverflow[3] is also invaluable to quickly look up how to do something in git. There are many tasks in git where I never remember the command, but I do remember enough about what the functionality is called to be able to quickly use Google to find a StackOverflow thread that gives me the call. Reading through a book or tutorial on git, even if you don't remember the commands you learn, can help you learn some of the vocabulary[4], and knowing that vocabulary will help you search for answers when you need them.

[3] **StackOverflow**. …

[4] Some good git-related words to know to help you search for calls for rarer tasks: "commit", "branch", "merge", "revert", "push", "pull", "merge conflict", "remote", "origin", "master", "fork", "clone", "pull request".

Finally, if you can find a way to do it, I think the best and easiest way to learn to use git and GitHub with R is to collaborate with someone who's used these tools before. Most of the time, these tools are very easy to use, but the small percent of the time that they're not, it can be significant stumbling blocks (in terms of the time it takes to figure out the fix) the first few times you use the tools, while someone familiar with them and working on the project can diagnose and get you over those bumps as you learn the ropes.

4
Package

As with many other things in R, the threshold of complexity for writing your own package has recently lowered dramatically. If you are writing some of your own functions in R to use for your research, and you've become comfortable with writing functions, you should try putting them in an R package. You don't have to share this package through publishing it on CRAN or another repository—instead, it's fine to start by just writing packages for your own use, or for your research group's private use. However, once you start writing packages, you will see how straightforward it is, and you may want to take the extra steps to prepare your package for CRAN and submit it.

4.1 R packages

4.2 Publishing

4.2.1 Why publish your packages?

4.2.2 Where to publish packages

4.2.3 Extra steps for publishing

If you want to submit your package to CRAN, you'll need to take a few extra steps to get it ready. These steps aren't necessary for a package you plan to just use privately, but they wouldn't be a bad idea to try to do even in that case.

First, you'll need to make sure that the functions in your package are comprehensively documented. [What are requirements for this for CRAN?]

In addition to documenting specific functions, you should also consider writing overall tutorials that walk a user through how to use your package in a few example scenarios. **vignette**[1]

[1] **vignette.** ...

You will also need to make sure that your package can run on different operating systems. If your package only has functions written exclusively in R, and doesn't interact with [the computer system?] or other programs on your computer, this shouldn't be an issue. However, you should still check. [Travis

CI, winbuilder]
[CRAN checks]

4.3 Checking

4.4 Learn more

One wonderful resource for writing and publishing R packages is the book R Packages by Hadley Wickham. This book is available for a reasonable price in paperback (many Barnes & Nobles carry it in their computer section) and is also available for free online. The author of this book also created and maintains the `devtools` **package**[2]
[Coursera, Leanpub]

[2] `devtools` **package.** An R package with functions that facilitate R package development. These functions are well-documented in the package documentation as well as through the book R Packages. This package also includes a function for installing R packages from code posted on GitHub (`install_github`), which is very useful if you want to find and use packages that are not yet available on CRAN and other repositories or if you want to use the latest, "development" version of a package. This package is available on CRAN and so can be installed with `install.packages("devtools")`.

5
Collect

For Environmental Health researchers, I think one of the most exciting developments in R recently is how it is changing how we can collect data, both for exposures and outcomes. One direction for this development is how researchers can collect and measure original data from experiments, including through new measurement technologies (e.g., phone-based Apps) and through new or rapidly changing health-related measurements (e.g., metabolomics, flow cytometry) and associated open-source software.

R is also facilitating and leveraging rapid developments in how researchers can access and query secondary data, including from [large admin databases?] and [data repositories encouraged or required for some NIH-funded projects]. … This section will provide an introduction to some of the ideas and techniques behind these developments for collecting secondary or public-use data for environmental health research, as well as give you somee directions on where to go to find R packages that facilitate collecting open data from R.

5.1 Open data

5.1.1 [Admin databases?]

Data collected by government groups like EPA, NOAA, USGS.

5.1.2 [Research repositories?]

Here, especially health-related data. NIH-supported (e.g., Metabolomics Workbench). NHANES?

5.2 Web services [?]

5.3 ROpenSci

One of the best places to explore R packages for accessing open data for science is **ROpenSci**.[1] Many of its packages facilitate access to databases of open

[1] **ROpenSci**. …

data relevant to scientific research that have API access [?]. You can browse through its packages on its **Packages** page

Examples of some packages relevant to collecting data for environmental health research include:

- bomrang: Australian Government Bureau of Meteorology ('BOM') Data Client
- clifro: Easily Download and Visualise Climate Data from CliFlo
- dbhydroR: 'DBHYDRO' Hydrologic and Water Quality Data
- essurvey: Download Data from the European Social Survey on the Fly
- FedData: Functions to Automate Downloading Geospatial Data Available from Several Federated Data Sources
- camsRad: R Client for CAMS Radiation Service
- ccafs: CCAFS GCM Data R Client
- dbhydroR: R interface to the South Florida Water Management District's DBHYDRO Database
- biomartr: Genomic Data Retrieval
- clifro: Easily Download and Visualise Climate Data from CliFlo
- DataSpaceR: An R Interface to 'the CAVD DataSpace'
- essurvey: Download Data from the European Social Survey on the Fly
- fingertipsR: Fingertips Data for Public Health
- getCRUCLdata: Use and Explore 'CRU' 'CL' v. 2.0 Climatology Elements
- getlandsat: Get Landsat 8 Data from Amazon Public Data Sets
- googleLanguageR: Call Google's 'Natural Language' API, 'Cloud Translation' API, 'Cloud Speech' API and 'Cloud Text-to-Speech' API
- GSODR: Global Surface Summary of the Day ('GSOD') Weather Data Client
- hydroscoper: Interface to the Greek National Data Bank for Hydrometeorological Information
- MODIStsp: A Tool for Automating Download and Preprocessing of MODIS Land Products Data
- nasapower: NASA POWER API Client
- opencage: Interface to the OpenCage API
- osmdata: Import 'OpenStreetMap' Data as Simple Features or Spatial Objects
- weathercan: Download Weather Data from the Environment and Climate Change Canada Website
- wateRinfo: Download Time Series Data from Waterinfo.be
- USAboundariesData: Datasets for the 'USAboundaries' package
- USAboundaries: Historical and Contemporary Boundaries of the United States of America
- tidyhydat: Extract and Tidy Canadian 'Hydrometric' Data
- stats19: Work with Open Road Traffic Casualty Data from Great Britain
- smapr: Acquisition and Processing of NASA Soil Moisture Active-Passive (SMAP) Data

- rWBclimate: A package for accessing World Bank climate data
- rusda: Interface to USDA Databases
- rsnps: Get 'SNP' ('Single-Nucleotide' 'Polymorphism') Data on the Web
- rrricanesdata: Data for Atlantic and east Pacific tropical cyclones since 1998
- rrricanes: Web scraper for Atlantic and east Pacific hurricanes and tropical storms
- ropenaq: Accesses Air Quality Data from the Open Data Platform OpenAQ
- rnoaa: 'NOAA' Weather Data from R
- rnaturalearth: World Map Data from Natural Earth
- riem: Accesses Weather Data from the Iowa Environment Mesonet
- rgpdd: R Interface to the Global Population Dynamics Database
- rdhs: API Client and Dataset Management for the Demographic and Health Survey (DHS) Data
- rdefra: Interact with the UK AIR Pollution Database from DEFRA
- prism: Access Data from the Oregon State Prism Climate Project

Cleaning / exploring data:

- cleanEHR: The Critical Care Clinical Data Processing Tools
- colocr: Conduct Co-localization Analysis of Fluorescence Microscopy Images
- cRegulome: Obtain and Visualize Regulome-Gene Expression Correlations in Cancer
- EndoMineR: Functions to mine endoscopic and associated pathology datasets
- geoaxe: Split 'Geospatial' Objects into Pieces
- hddtools: Hydrological Data Discovery Tools
- isdparser: Parse 'NOAA' Integrated Surface Data Files
- visdat: Preliminary Visualisation of Data
- skimr: A frictionless, pipeable approach to dealing with summary statistics

5.4 Learn more

6
Process

For environmental health research, once you have collected your raw data, you will often need to do a bit of work processing the data before you can apply epidemiological models. As one example, you may pull **gridded data**[1] on an environmental exposure of interest, but need to link it with health data that is aggregated based on administrative boundaries (e.g., counties). As another example, you might have daily temperature data and want to identify the dates of heat waves in a community based on that data.

[1] **gridded data.** …

R has some wonderful tools for processing data that are relevant to environmental health research. Here, I'll focus on tools from a few packages I've developed, but in "Learn more", I'll also point you to more resources for finding R tools that might be relevant to your own environmental health research projects.

I strongly encourage you, as you work through this section, to start thinking about possibly creating your own R packages to solve data processing tasks you commonly face for your research. All the code for the packages I'll discuss is available on GitHub, so you can look at this code as examples as you think about writing your own packages.

6.1 Learn more

- CRAN taskviews
- Bioconductor [taskviews?]
- ROpenSci

[CRAN] One excellent example of CRAN-based packages [?] for processing environmental data is the suite of packages created and maintained by scientists at the **United States Geological Survey (USGS)**[2] This group has a collection of packages, listed at … . The packages include … For a very cool example of using some of these packages for a timely application for environmental health, check out these visualizations of flooding during Hurricanes …, as well as the code used to create them.

[2] **United States Geological Survey (USGS).** …

For biological data, **Bioconductor**[3] can be a great resource for finding packages to process and pull out relevant data. For example, Bioconductor has a large collection of packages for working with biological data collected through flow cytometry, mass spectrometry (e.g., metabolomics), RNA sequencing, [others]

[3] **Bioconductor.** …

7

Final Words

8
Bibliography

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2019). *rmarkdown: Dynamic Documents for R*. R package version 1.12.

Bache, S. M. and Wickham, H. (2014). *magrittr: A Forward-Pipe Operator for R*. R package version 1.5.

Cheng, J., Karambelkar, B., and Xie, Y. (2018). *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*. R package version 2.0.2.

Iannone, R., Allaire, J., and Borges, B. (2018). *flexdashboard: R Markdown Format for Flexible Dashboards*. R package version 0.5.1.1.

Pebesma, E. (2019). *sf: Simple Features for R*. R package version 0.7-3.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., and Despouy, P. (2018). *plotly: Create Interactive Web Graphics via 'plotly.js'*. R package version 4.8.0.

Vaidyanathan, R., Xie, Y., Allaire, J., Cheng, J., and Russell, K. (2018). *htmlwidgets: HTML Widgets for R*. R package version 1.3.

Wickham, H. (2019). *forcats: Tools for Working with Categorical Variables (Factors)*. R package version 0.4.0.

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., and Woo, K. (2018a). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.1.0.

Wickham, H., François, R., Henry, L., and Müller, K. (2019). *dplyr: A Grammar of Data Manipulation*. R package version 0.8.0.1.

Wickham, H., Hester, J., and Francois, R. (2018b). *readr: Read Rectangular Text Data*. R package version 1.3.1.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2018). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.9.

Xie, Y. (2019). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.22.

Xie, Y. and Allaire, J. (2018). *tufte: Tufte's Styles for R Markdown Documents*. R package version 0.4.

Xie, Y., Cheng, J., and Tan, X. (2018). *DT: A Wrapper of the JavaScript Library 'DataTables'*. R package version 0.5.