Miscellaneous topics

Abhijit Dasgupta

BIOF 339

Search strategies

Google/Bing/DuckDuckGo

- A problem we have here is that "R" is just a letter of the alphabet, so we get too many results
- The same term (say, `filter` or `print`) is used in various contexts and in various computer languages

Google/Bing/DuckDuckGo

- **Strategy 1:** Use "CRAN" instead of "R" to mean R. If there is a package that meets your needs, this will pick it up
- **Strategy 2:** You can use "-" to qualify what you don't want to search for. So you could do "signal R -python" to look for sites which are not talking about Python
- **Strategy 3:** Restrict yourself to StackOverflow or Cross-Validated, which are dedicated to computer issues
    - On StackOverflow and CrossValidated, R issues have the tag "r"
    - Have thick skin, since things can get heated sometimes if you are thought to have asked a "stupid" question

rseek.org, a better choice

Twitter

The R community is organized on Twitter with the hashtag "#rstats"

- This is a very active community
- Welcoming, diverse, patient, quick, fun
- Lots of top developers contribute daily (Wickham, Averick, Kuhn, lots of RStudio folk, package developers)
- Can virtually follow all the major and minor R conferences, since someone is certainly live-tweeting. Just need to find the hashtag or conference Twitter handle
- Almost never bashed for asking a "stupid" question

Blog aggregators

R-bloggers (link) & R weekly (link)

- Find blogs on almost any R topic under the sun (since 2005)
- Announcements of new packages
- Hundreds of contributing blogs
- Some curated tutorials

Useful blogs

- RStudio has several blogs which are quite useful and informative
  - R Views (link)
  - includes a "Top 40" monthly new packages update
  - Tidyverse blog (link)
- STHDA
  - Really useful site
- Shirin's Playground
- ouR data generation

Multimedia sources

YouTube and video

- R Consortium channel: This channel contains videos from several useR and other conferences, including the excellent R/Medicine conference
- RStudio webinars This site also contains links to all the `rstudio::conf` conference videos
- The New York and DC R conferences. Yours truly was a speaker at the DC 2018 and 2019 conferences

Other websites of interest

- Awesome-R: A curated list of R packages and tools
- Flowing Data: One of the top visualization blogs out there, based in R, by Nathan Yau
- crazyhottommy/getting-started-wtih-genomics-tools-and-resources
- The Carpentries (Data Carpentry and [Software Carpentry](
    - Intro to R and RStudio for Genomics (link)
    - R for Reproducible Scientific Analysis (link)

Stealing code

GitHub

GitHub is a website where developers come to play. It hosts *repositories* of code where people can submit issues, contribute code and co-develop software products.

Most R developers put their developing code on GitHub. There are over 108,000 repositories on GitHub using R.

To see what's there, click here

Developers to follow:

- RStudio
- ROpenSci
- tidyverse

Changing some default behaviors

.Rprofile

You can create a `.Rprofile` file either in each project or globally (place the file in your HOME folder)

Every time R starts, it will look at this file and load things if you so specify

Some examples you could put in there to be available every time

```
## ht == headtail
ht = function(d, n=6) rbind(head(d, n), tail(d, n))

local({
  r = getOption("repos")
  r["CRAN"] = "https://cran.rstudio.com/"
  options(repos = r)
})
```

> Don't put anything in there that might make your R non-portable, for example `options(stringsAsFactors=F)`.

See this chapter of "Efficient R Programming" by Gillespie and Lovelace.

Changing default operations for a R class

R uses what is called the *S3* system for object oriented programming. It is a simplistic system where you create a default function and then specify functions for different classes. For example:

```r
format_output <- function(x,...){
  # Make a S3 class
  UseMethod('format_output',x)
}

format_output.lm <- function(x, refs=NULL, labs=NULL, pretty=T){
  tmp <- summary(x)$coef
  if(is.null(refs)){
    term <- attr(x$terms, 'term.labels')
  } else {
    term <- names(refs)
  }
  out <- data.frame(tmp[,c(1,2,4)])
  names(out) <- c('LOR','SE','pvalue')

  ## Truncated for space, see https://github.com/webbedfeet/abhiR.git
```

So class-specific functions just need the name of the class after the dot.

Changing default operations for a R class

Sometimes, there already is a default that you want to change. Then you don't need to create the generic first since it already exists

```r
print.lm <- function(x){
  suppressPackageStartupMessages(require(tidyverse))
  require(broom)
  out <- tidy(x) %>%
    select(term, estimate, p.value)
  print(out)
}
```

So now:

```r
m <- lm(mpg ~ wt, data = mtcars)
print(m)
```

```
# A tibble: 2 × 3
  term        estimate  p.value
  <chr>          <dbl>    <dbl>
1 (Intercept)    37.3  8.24e-19
2 wt             -5.34 1.29e-10
```

Creating your own function repository

You should create functions that you use all the time and make your own repository

Create each function in a separate file, and then load them using the `source` function.

Why is this a good idea?

- Functions are meant to be re-usable recipes for particular purpose
- If we hide functions in general script files, we'll have a hard time finding them
- Separating functions out into separate files allows easier
  - editing
  - documenting
  - loading

Creating packages

Creating packages sounds intimidating, but really isn't

The `devtools` package makes it very easy. So does RStudio.

R packages

R packages

Once you have some functions written, it may be worthwhile creating a R package for your own purposes

R packages have a particular structure that can be created with the `package.skeleton` function.

A few nice tutorials for writing R packages are:

1. by Hilary Parker
2. by Daniel Sjoberg
3. by Sharon Machlis
4. by Neeraj Dhanraj

Version control

Version control systems (VCS) are systems that allow you to keep track of changes in files in a way that allows "rewinding".

Examples are git, mercurial and subversion

git tends to be the most popular VCS.

There are several online collaborative environments that utilize git. These include GitHub, GitLab and BitBucket.

Version control

The basic idea is that

- you make small edits in your code files
- you then "save" the changes as a *commit* in git
- if you want, you can then "push" your changes to an online repository (repo), so others can use and share the code

If you want to learn git, which is a really useful skill, here are some resources:

- Happy git with R
- Learn the basics of git in under 10 minutes
- Git immersion

Version control systems like git will save your bacon more times than not!!