Tips for good scientific coding practices

by Candace Savonen – adapted from AlexsLemonade

Style guides help people read your code

Just how incorrect punctuation and grammar can distract a reader from the message in your writing, code that doesn't follow a style can be difficult for others to understand and use. Your code is not as useful if it isn't easily readable, which is why naming conventions, code style, and consistency are important.

We suggest following a style guide like one of these:
- Hadley Wickham's R Style Guide
- Google's R Style Guide.

`set.seed()` helps people reproduce your results

Setting the seed is needed when performing any kind of analyses that use random sampling or something that may vary each time you re-run it.

How to set the seed:

Step 1) Put any number as your argument for the function and run `set.seed` like below. The number *itself* that you use doesn't matter.

```
set.seed(54321)
```

Step 2) Run your analyses like normal. Here we are sampling five numbers using the function, `sample()` and sampling `5` numbers in the set of numbers, `1:10` (one through ten).

```
set.seed(54321)
sample(1:10, 5)
```

Output: `[1]  4  7  2 10  6`

Test this out If you run your analyses again, and still have the seed set to the same number, you will get exactly the same results. But if you change the number, the results will also change.

```
set.seed(54321)
sample(1:10, 5)
```

Output: `[1]  4  7  2 10  6`

```
# Change the seed to something else
set.seed(2020)
# These 'results' will be different now
sample(1:10, 5)
```

Output: `[1]  7  6  8  1 10`

```
# Go back to the original seed number we used
set.seed(54321)
# These 'results' go back to being what they were before
sample(1:10, 5)
```

Output: `[1]  4  7  2 10  6`

Setting the seed makes so your results are *exactly* reproducible. For more explanation on `set.seed`, here's a StackOverflow post and a mini tutorial on seed setting in R.

Note that sometimes functions have built-in arguments for setting the seed or do not respect setting the seed in the global environment like we showed above. We suggest running a step, script, or notebook multiple times in different sessions and comparing the output to ensure that you get the same result.

R Notebooks are helpful for documenting your science

R Markdown documents are helpful for scientific code by allowing you to keep detailed notes, code, and output in one place.

They also have the added benefit of having HTML file output that is styled and easy to read. Saving your R Markdown will create an HTML file containing the code and output to be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file). The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

`sessionInfo` tells people what packages you used

The `sessionInfo` function prints out what packages and versions you used in your analysis. This way, when others try to reproduce your research, they know what packages you have loaded and how things were set for the code you ran.

More reading on good scientific code:

- 'Ten simple rules' for reproducible computational research- PLoS
- 'Ten simple rules' for documenting scientific software
- Guide to reproducible code

* This guide was adapted from the AlexsLemonade.