Final Project

J. Lucas McKay

2024-04-19

Due 5 PM Wednesday, 1 May 2024

The BMI-510 final project is designed to get you in the habit of creating documented, reusable code for your projects and colleagues. The project is due May first to accommodate grade entry deadlines May 3. **No extensions are possible.**

Objective

- You will create a github repository containing a small R package with functions described below.
- You may create as many support functions as you like, but the ones listed below should be `@export`ed and documented appropriately with Roxygen2 comments so that they can be queried with `?functionName`.
- You will turn in a link to your github repo.
- Online sources are allowable. No discussion is allowed between students.
- **ChatGPT NOTE** ChatGPT is allowed, but be aware that it can produce identical code if two students use very similar prompts, which can trigger anti-plagiarism software. It is probably not a good idea to use verbatim ChatGPT output in your functions. (Verbatim output for Roxygen2 comments is expected and fine.)

Functions

- This project has a total of **20 points**.
    - The *code* for each function below is worth **1 point**, with the exceptions of more complex `standardizeNames` (**2 points**) and `downloadRedcapReport` (**3 points**).
    - The *documentation* for each function below is worth **1 point**.
    - The `NAMESPACE`, `README.md`, and `.gitignore` files in your package are each worth **1 point**.

1. `logLikBernoulli = function(data)` Write a function that takes a vector like `data = c(1,0,0,0,1,1,1,)` and calculates the parameter `p` that maximizes the log-likelihood $\log(P(p|data))$. Use a grid-based search with `p` in steps of 0.001.

2. `survCurv = function(status,time)` Write a function that takes a numerical vector `status` and a numerical vector `time`, and calculates and plots a survival curve `S(t)`. Test your function on the dataset here (https://jlucasmckay.bmi.emory.edu/global/bmi510/Labs-Materials/survival.csv).

3. `unscale = function(x)`: Write a function that takes a vector that has been put through `scale` and reverses the centering/scaling, if any.

4. `pcApprox = function(x, npc)`: Write a function that returns an approximation to the data `x` based on `npc` PCs. (Note that the approximation should be rescaled and centered to match the original data).

5. `standardizeNames = function(data)`: Write a wrapper around `dplyr::rename_with` and `janitor::make_clean_names` that converts the variables in a `tibble` `data` to `"small_camel"` case (or another case if you like). The idea here is to have a reliable function that standardizes the variable

names in data you're dealing with. This function should import elements of the `janitor` and `snakecase` packages; Roxygen2 will handle these dependencies for you.

6. `minimumN = function(x1,x2)` : Write a wrapper around `pwr::pwr.t2n.test` that takes either one (`x1`) or two (`x2`) samples of preliminary data and returns the minimum sample size needed for a t-test of the null hypotheses that either $\mu_{X1} == 0$ or $\mu_{X1} == \mu_{X2}$ with 80% power at alpha=0.05.

7. `downloadRedcapReport = function(redcapTokenName,redcapUrl,redcapReportId)` : Using the block of RedCap template code below, write a function that:

    - uses `Sys.getenv()` to read an API token called `redcapTokenName` from the users' `.REnviron` file.
    - queries `redcapUrl` to return the Redcap Report `redcapReportId` . (Notice these are the data from our simulated stroke study, now nicely and securely hosted on RedCap.)
    - returns the contents as a `tibble` .

```
# This is an example R script that REDCap generates for you to modify for your own purpo
ses. Be sure to create an .REnviron file in your directory to securely hold your API tok
en. Be sure NOT to sync your .REnviron file to github!

#!/usr/bin/env Rscript
token <- "6189879441F5C29A25245880677488BF"
url <- "https://redcap.emory.edu/api/"
formData <- list("token"=token,
    content='report',
    format='csv',
    report_id='46524',
    csvDelimiter='',
    rawOrLabel='raw',
    rawOrLabelHeaders='raw',
    exportCheckboxLabel='false',
    returnFormat='csv'
)
response <- httr::POST(url, body = formData, encode = "form")
result <- httr::content(response)
print(result)
## # A tibble: 91 × 7
##    record_id  code visit_number muscle    mv arm   condition
##        <dbl> <dbl>        <dbl> <chr>  <dbl> <chr> <chr>
##  1       101     1            2 FDI     97.7 Sham  Post Sham
##  2       102     1            1 TA      93.4 Sham  Baseline
##  3       103     1            2 TA     102.  Sham  Post Sham
##  4       104     2            1 FDI     91.3 Stim  Baseline
##  5       105     2            2 FDI    107.  Stim  Post Stim
##  6       106     2            1 TA     107.  Stim  Baseline
##  7       107     2            2 TA     152.  Stim  Post Stim
##  8       108     3            1 FDI     83.9 Sham  Baseline
##  9       109     3            2 FDI     94.1 Sham  Post Sham
## 10       110     3            1 TA     113.  Sham  Baseline
## # ℹ 81 more rows
```