Azul-rmarkdown-tutorial

2022-09-29

to use rmatkdow we first need to install it

```
install.packages("rmarkdown")
tinytex::install_tinytex()
```

R Markdown

#hey This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Best practices for scientific computing Summary

2.1 Homework Assignment 09-29

2.1

Assertion- a statement that something holds true at a particular point in a program. These can be used to ensure that inputs are valid, and outputs are consistent.

Automated tests- test that check to make sure that a single unit of code is returning the correct results.These also check that pieces of code work correctly when combined and that their behavior is not changed by modifications.

Symbolic debugger- interactive program inspector that allows the user to see exactly how the code is executing. This allowing the user to see what is going wrong directly rather than the user having to anticipate the error or asume the error with indirect evidence. > Steps from 5-8 have less answers because i havent experimented in those aspects

2.2

1.**write programs for people not computers** A - A program should not require its readers to hold more than a handful of facts in memory at once. We've done this by keeping track of our commands and annotating on our code B - In the past we've named different aspects of our code in order to have a distinctive and meaningful name that leads us to understand what data the code output is providing. C - We've also made sure to stay consistent with naming. By this I'm reffering to us keeping the same name whith the exact same formatting and capitalization so that we are able to read and understand our commands.

2. **Let the computer do the work** A - we've made the computer repeat tasks B - Saved recent commands in a file to reuse - Setting a working director and naming files by keeping different folders with specific files.

3. **Make Incremental Changes** A - I've Worked in small steps with frequent feedback and course correction - this was more for my statistics class when completing a semester project first we needed to collect all the data and then we followed deparate steps at separate time. B - Use a version control system - we started using Git to keep track of our document and any changes we make. C - Put everything that has been created manually in version control- we annotate manually a description of changes when we are committing our document to git.

4. **Don't repeat yourself (or others).** A - Every piece of data must have a single authoritative representation in the system

    - ive done this by keeping a name on all the data so that it can be found when needed. B - Modularize code rather than copying and pasting - although for some code commands i do copy and paste I now have been Modularizing code because of the way my files are named compared to others whose code I am using as a rubric. C- re- use code instead of rewriting it - when in the past doing different assignments using the same data file it is easier to keep the same subset names to stay organized.

5.**Plan for Mistakes** B - use an off-the-shelf unit testing library - i believe we've also done this by using the help console in r and in the terminal. as for a & c i dont think ive done much of.

6. **Optimize Software Only after It Works Correctly** B - write code in the highest-level language possible - We've done this by using shorter commands that give us the same output of larger commands.

7. **Document Design and Purpose, Not Mechanics** -A - document interfaces and reasons, not implementations- This I've done by annotating (#) when coding -B - refactor code in preference to explaining how it works- For this I have also just annotated under or avove the comand to in the same order as the input itself but insted of a real input i describe what the input that goes in certain place should be.

8. **Collaborate.** For this best practice I havent done any of the recomendations since ive never worke in a group while coding in one specific file.