R programming basics: coding etiquette

Ecological Systems Modeling

Jan 20-24, 2025

Active participation (optional)

- Open RStudio in Jupyter Hub
- In the Files/Plots/etc. pane, navigate to: \$HOME/Labs/Intro_R_part3/
- Click on Intro_to_R_part3_codingEtiquette.rmd
- File should open in the Source pane
- Run the code chunks, add to chunks, or type code in Console

Learning objectives

- Describe several types of best coding practices
- Apply this knowledge in your code for this course
 - Lab assignments
 - Class project

Overview: best coding practices

- 1. Abide by the DRY (Don't Repeat Yourself) principle
- 2. Follow some easy-to-remember naming convention
- 3. Keep the code as straightforward as possible
- 4. Limit the length of a line of code
- 5. Use comments frequently
- 6. Use consistent indentation
- 7. Whenever and wherever possible, avoid deep nesting

1. DRY: Don't Repeat Yourself

- Don't write the same code repeatedly
- Example shows duplicate C# code
- Instead, write and re-use functions (more on this later)

```
Duplicate Code
         ---
Clusters
                               File: CSharpBracketSearcher.cs in project: ILSpy.csproj

    1 2 duplicate blocks

                               249
                                                                if (!verbatim) inString = false;
                               250

    1 2 duplicate blocks

                               251
                                                           case '/':
▶ 📫 3 duplicate blocks
                               252
                                                                if (blockComment) {
▶ ↑ 2 duplicate blocks
                               253
                                                                    Debug.Assert(offset > 0);
                                                                    if (document.GetCharAt(offset - 1) == '*') {
                               254

    1 3 duplicate blocks

                               255
                                                                         blockComment = false:

    1 2 duplicate blocks

                               256

    1 2 duplicate blocks

                               257
                                                                if (!inString && !inChar && offset + 1 < document.TextLength) {
                               258

    3 duplicate blocks

                                                                    if (!blockComment && document.GetCharAt(offset + 1) == '/') {
                               259

    1 2 duplicate blocks

                               260
                                                                         lineComment = true:

    16 duplicate blocks

                               261
                                                                    if (!lineComment && document.GetCharAt(offset + 1) == '*') {
                               262

    3 duplicate blocks

                               263
                                                                         blockComment = true;

    ▶ 1 6 duplicate blocks

                               264

    1 9 duplicate blocks

                               265
                               266
                                                               break;

    1 9 duplicate blocks

                                                           case '"':
                               267

    6 duplicate blocks

                               268
                                                               if (!(inChar | lineComment | blockComment)) 

    4 duplicate blocks

                               File: CSharpBracketSearcher.cs in project: ILSpy.csproj

    1 2 duplicate blocks

                               155
                                                               if (!verbatim) inString = false;

    1 2 duplicate blocks

                               156
                                                               break:
                               157
                                                           case '/':

    2 duplicate blocks

                               158
                                                                if (blockComment) {

    2 duplicate blocks

                               159
                                                                    Debug.Assert(i > 0);

    2 duplicate blocks

                               160
                                                                    if (document.GetCharAt(i - 1) == '*') {
                               161
                                                                        blockComment = false;

    2 duplicate blocks

                               162

    1 2 duplicate blocks

                               163
▶ 

    2 duplicate blocks
                               164
                                                                if (!inString && !inChar && i + 1 < document.TextLength) {
                                                                    if (!blockComment && document.GetCharAt(i + 1) == '/') {
                               165
▶ 1 2 duplicate blocks
                               166
                                                                         lineComment = true;

    1 2 duplicate blocks

                               167

    3 duplicate blocks

                               168
                                                                    if (!lineComment && document.GetCharAt(i + 1) == '*') {
                               169
                                                                         blockComment = true;

    2 duplicate blocks

                               170

    3 duplicate blocks

                               171

    1 2 duplicate blocks

                               172
                                                                break:
                               173

    1 2 duplicate blocks

                                                               if (!(inChar || lineComment || blockComment))
                               174

    3 duplicate blocks

    1 2 duplicate blocks

170 clusters
```

2. Easy-to-remember naming convention

- Use descriptive and succinct names for all objects
- Good: Underscores ("_"), periods ("."), or combo of upper and lower case
- Avoid: Uninformative names
- Fail: Spaces and certain symbols such as "/"

Good name	Good alternative	Avoid or fail	
Max_temp_C	MaxTemp	Maximum Temp (°C)	
Precipitation_mm	Precipitation	precmm	
Mean_year_growth	MeanYearGrowth	Mean growth/year	
sex	sex	M/F	
weight	weight	w.	
cell_type	CellType	Cell type	
Observation_01	${\tt first_observation}$	1st Obs.	

3. Straightforward and succint code

- Nobody want to read messy code, including your future self
- Would you want to look at this?

```
1 # Not-so-good
 2 for (i in 1:nrow(df <- data.frame(x=c(1, 2, 3, 4)))) { print(df[i,])}
     lst = list(data.frame(y=c(6, 7, 8, 9, 10), z = c(2,2,2,2,2)),
                 m = matrix(5, 5, nrow=5))
 6 # Better
 7 df \leftarrow data.frame(x = c(1:4))
 8 for (i in 1:nrow(df)) {
    print(df[i,])
10 }
   df2 \leftarrow data.frame(y = c(6, 7, 8, 9, 10), z = c(rep(2, 5)))
12 mat <- matrix (5, 5, nrow = 5)
13 lst \leftarrow list(df2, mat)
```

3. Straightforward and succint code

- Use multiple lines to create new objects, etc.
- Reduce text using operators and functions (e.g., : and rep())

```
1 # Not-so-good
 2 for (i in 1:nrow(df \leftarrow data.frame(x=c(1,2,3,4)))) { print(df[i,])}
     lst = list(data.frame(y=c(6,7,8,9,10)), z = c(2,2,2,2,2)),
                 m = matrix(5, 5, nrow=5))
 6 # Better
 7 df \leftarrow data.frame(x = c(1:4))
 8 for (i in 1:nrow(df)) {
     print(df[i,])
10 }
   df2 < - data.frame(y = c(6:10), z = c(rep(2, 5)))
12 mat \leftarrow matrix(5, 5, nrow = 5)
13 lst <- list(df2, mat)
```

4. Limit the length of a line of code

Again, short lines of code are easier to read and understand

```
1 # Not-so-good
 2 for (i in 1:nrow(df \leftarrow data.frame(x=c(1,2,3,4)))) { print(df[i,])}
     lst = list(data.frame(y=c(6,7,8,9,10)), z = c(2,2,2,2,2)),
                 m = matrix(5, 5, nrow=5))
 6 # Better
7 df \leftarrow data.frame(x = c(1:4))
 8 for (i in 1:nrow(df)) {
    print(df[i,])
10 }
11 df2 <- data.frame(y = c(6:10), z = c(rep(2, 5)))
12 mat \leftarrow matrix (5, 5, nrow = 5)
13 lst <- list(df2, mat)
```

5. Use comments frequently

- Allows for explanations, instructions, etc. for analyses
- Your future self and others who review your code will thank you
- Especially important if code is in a script (rather than R Markdown)

```
1 # This is a script for degree-day modeling
2 # Author: Brittany Barker
3 # Last updated: Jan 13, 2024
4 library(here) # Useful for project-relevant paths
5
6 # Lower developmental threshold (base 50F)
7 LDT50 <- 50
8
9 # Create an empty data frame to store results
10 out_all <- data.frame(matrix(ncol = 4, nrow = 0))
11
12 # Etc, etc.</pre>
```

6. Use consistent indentation

- Use tabs, not spaces
- Helps keep track of nested code blocks
- RStudio will auto-indent!

```
1 # Notice the 4 levels of indentation
 2 years <- 2022:2023</pre>
   days <- 1:3
 4 # Loop through each year in `years`
 5 for (year in years) {
     if (year %% 2 == 0) {
    # Next loop through each day
    for (day in days) {
         msg <- paste("Day", day, "in", year)</pre>
        print(msg)
10
11
12
13 }
```

7. Avoid deep nesting