Stat 431 Final Portfolio

Table of contents

Independent Learning (IL):

Independent Learning (IL):

These objectives show your ability to seek out new information and adapt to new tools to solve data analysis problems.

[IL-1] Adding new skills:

- I can find and adopt new packages to accomplish tasks.
- I can adapt to different syntax styles (tidy, base, formula style, data.table).
- I can use tutorials, etc. to enhance my understanding of new concepts

Level: 3

Justification

Aside from the new packages that I needed to learn to complete this quarter's assignments (which were often taught as a part of that weeks course material) I frequently relied on other packages to help streamline difficult portions of assignments. A notable example was using lubridate to help in the formatting of the dates/times of when the ISS would pass over the U.S.'s state capitals in Lab 3.

Lab 3 Example

Coming into Spring Quarter, I was already accustomed to tidy seeing as that was Dr. Robinson's preferred R dialect. Adjusting to the occasional implementation of base R was at first a difficult transition, but with continued practice I now rely heavily on the base R's ability to, within a relatively little amount of code, be able to manipulate data frames, matrices, etc. (at the cost of readability, of course).

Hierarchical Clustering Example - Lines 65,66 (Example from worse, full-of-for-loops version of the code)

Beyond the class provided tutorials, I often used other videos/resources to solidify my understanding of certain concepts. The examples provided are YouTube videos, the first of which gives me the mathematical/statistical understanding of average-linkage in hierarchical clustering while the other provided me with a deep introduction of working with reading in APIs (bonus points for the speaker being a Seattlite).

Average-Linkage Hierarchical Clustering

APIs Introduction

[IL-2] Online resources:

- I can use online resources (Google, ChatGPT, StackOverflow) to solve problems, debug, or find new tools.
- I can find source code for similar projects to use as starting points for my own
- I can read the documentation of an API to figure out how to access data.

Level: 4

Justification

As I feel will be the standard from now on, Chat GPT was integral in getting over some of those coding roadblocks this year. I found myself using chat the most when wanting to confirm whether something I thought to be true (e.g. Is “row.names” a built in specification to look for how the rows are organized in the function merge()?). But at times, I’d copy in my code, tell chat where the error occurred, and what kind of error I received. Typically, the code revision chat would return to me me wouldn’t work, but it would point me in the right direction.

ChatGPT Example 1

ChatGPT Example 2

Also finding inspiration on the internet from similar projects and using those as my own starting points occurred often this quarter. My most notable example had to have been during lab 8 when my aerospace buddies decided I needed to attempt to make art using Lorenz attractors. I made sure to reference the skeleton of one of the suggested articles on this topic.

10 Million Points With ggplot clifford attractors

Lab 8 Code

Although I completed the APIs practice assignment and lab to a degree that I feel would have demonstrated that “I have mostly mastered” the skill, I decided to take on the API’s challenge assignment and focus on just being able to read Teleport’s API formatting and reading in the scores for the 264 cities referenced in their urban_areas.json document. While it was initially confusing seeing only more API references being returned by fromJSON(rawToChar(res$content)), I eventually was able to pull out each city’s life quality scores by finding the correct reference “/{city}/scores/”. From there, I just used some simple string manipulation to return all 264 city life quality scores.

Challenge 3 API Code

Video of me debugging gradient descent code using browser():

https://www.youtube.com/watch?v=MpEPSdLZ4kk

Reproducible Workflow (RW):

These objectives show your ability to produce artifacts and deliverables that are organized, documented, version tracked, and responsibly designed. ______________________

[RW-1] File, code, and data management:

- I can use Git and GitHub to track my progress and collaborate (creating repos, cloning, forking, pull requesting).
- I always use R Projects and the {here} package to organize my scripts, notebooks, data, and applications.

Level: 4

Justification

Whenever working with GitHub, I always made sure to work out of Branches/Forks. My preference for which depended on whether or not I needed any of the code that my group mates were actively working on in their branches (however, the size of these projects/assignments rarely justified needing their code after we delegate what portion of the assignment would be designated to)

Branch Example

Fork Example

I've made it a priority of mine to make sure that my folders, scripts, and packages are well organized this year. This can be seen in the organization of my folders and packages in the video below:

https://www.youtube.com/watch?v=8DAWA_1KcSw

Although I only needed to locally load packages in Lab 2 this year, I'm referencing code from another class that demonstrates that I use that package to make my code reproducible

Exam 2 Questions

When working on Lab 4, my group decided that we wanted our R package to have our data files be present and accessible within the package. This lead to me needing to make my code reproducible for all of those who (theoretically) would want access to it. To make sure that any potential user would be able to access the data files they downloaded upon installation of the package, I used the system.file() function that automatically finds where the package was located in local directories, regardless of where the user has their files stored (find.package is called under the hood of system.file() after checking that the file exists using file.exists).

Lab 4 Example

[RW-2] Notebooks:

- I can use Quarto and/or R Markdown to produce a reproducible notebook and polished rendered document.
- I can use appropriate chunk options (echo, error, cache, etc.) to render my qmd/Rmd quickly and cleanly.

Level: 3

Justification

Only a few instances this quarter have I found myself rendering qmd files, but my recent and most notable example was for the Generative Art Lab. When running an art piece using Lorenz Attractors, there were originally 10 million data points leading to slow run times. Using cache = TRUE on that specific chunk of code stopped me from having to experience that 5+ min run time.

Lab 8 Example

I also often take notes on Quarto in bullet format with interwoven code chunks from lecture material

- Reference: Any of my weekly class notes

This quarter, because I didn't often work with Quarto or Markdown files, I didn't often need to work with echo, error, cache, etc., but last quarter in STAT 331 I often used them.

331-lab7 Example

331-Challenge7 Example

[RW-3] Code style

- My code is clear, readable, well-organized, and well-commented.
- I can use a package-based workflow to organize my analyses

Level: 3

Justification

Project 1 Example - Code Split between Brandon and I

Project 3 Example- Lines 123 and On

Technical Communication (TC):

These objectives show your ability to communicate the processes you have implemented in your code, as well as the data conclusions and results.

[TC-1] Project summaries:

- I can clearly and succinctly summarize the contributions of my project.
- I accurately interpret statistical or modeling results.
- I consider the appropriate scope and impact of my project results.

Level: 3

Justification

Project 1 Shiny App Summary and Capabilities

Project 3 Model Summary and Changeable Inputs Description

[TC-2] Documentation:

- I provide ample documentation and tutorials for my custom functions.
- I provide user-friendly guides for my tools and software

Level: 3

Justification

Elbow Graph Project 3 Documentation- Example of good commenting/documentation of the functions, their parameters, and their return values

Dr. Bodwin provided some criticisms on the UX of our application, most notably that should a user open up our app blind, they wouldn't know what is going on within the app or where the data came from. I made small edits in our shiny app to add static text that summarizes the origins of the data and the purpose/functionality of the app.

Project 1 App Description

Data Manipulation (DM):

These objectives relate to the collection, cleaning, processing, and preparing of datasets for analysis.

[DM-1] Data preparation:

- I can read in datasets to R, including untidy ones.
- I can clean datasets to deal with missing data, typos, poor formatting, etc.

Level: 4

Justification

When working with national energy output data set in the Shiny app project, I was not only able to convert our data set into a tidy one, but I believe I was able to do it in a way that recognized a pattern/consistency in the formatting of the data that then allowed for a more streamlined conversion to a tidy format (as seen when comparing to my group mates resolution). I utilized the map_df() function to quickly read through every sheet (which represented one years worth of energy data) after having worked out some of the formatting issues in the sheets.

Project 1 Example

In lab 2 when prompted to choose a messy/hard to understand graphic and create a better, cleaner version of the same data, I needed to significantly alter the formatting of the information provided by the respondents. Once the data was mutated into a more coder-friendly format, I was then able to apply the final alterations to prepare it to be input into a Leaflet map.

Lab 2 Example

[DM-2] Data wrangling

- I can cleverly use pivoting, grouping, and joining to wrangle data.
- I can use mapping ({purrr}), applying (tapply, lapply, …), and/or iteration (for loops) to perform repeated tasks.

Level: 3

Justification

Similar justification to that which was found in DM-1

[DM-3] Data collection

- I can use API urls to access JSON data and convert it to a data frame
- I can webscrape simple tables and information

Level: 3

Justification

As mentioned above, to reinforce my understanding of APIs and collecting data from them, I took on reading off the city score data information from Teleport's API.

Challenge 3 API Code

Although I did not find myself using webscraping on online tables and information beyond the practice assignments, I decided to webscrape the wikipedia page on webscraping.

Webscraping Webscraping

Professional Visualization (PV):

[PV-1] ggplot: grammar of graphics

- I can use less common geometries, including those from ggplot extension packages.
- I can use the correct aesthetics to map variables
- I understand how geometries inherit aesthetics I can add annotations to my plot

Level: 3

Justification

Project 2 Example

Video of me working with ggplot: https://www.youtube.com/watch?v=5BzdgbUCypw

Lab 8 Example- Working with different geometrics in ggplot

[PV-2] ggplot: theme

- I can edit the titles, subtitles, captions, axis labels, etc. to create a clearly labelled plot
- I can choose colors ("scales") and themes to make a visually pleasing and accessible plot

Level: 3

Justification

Reference Justification Above

[PV-3] Dynamic visualizations

- I can use a package like {gganimate} to create self-contained gifs
- I can use a package like {plotly}, {ggplotly}, {leaflet}, {ggirafe}, etc. to make interactable html widgets

Level: 3

Justification

Although I hadn't used gganimate much (aside from the practice activities during week 2), last quarter my group mates and I decided it would be good to pursue a challenge in animating some of the graphs that we had created for our final project.

331 Final Project gganimate

Project 1 Example - Leaflet Code Produced in Shiny App

Project 2 Example - Goal Scored Leaflet Map

Project 3 Plotly Example- Created a simple plotly graph to allow the user to see what the total WCSS was for each cluster combination. In Create Elbow Graph code.

[PV-4] Shiny

- I can create a functional Shiny app.
- I understand the principle of reactivity, and how to use it.

Level: 4

Justification

I felt that the multi-step interactivity in our shiny app on the U.S.'s yearly energy output more than demonstrated that I have a strong knowledge of reactivity. Instead of having a series of user input prompts that displayed the specifically requested output, the output was dynamic and automatically displayed the changed inputs from the user.

Project 1 Example

Software Development (SD):

These objectives relate to your ability to develop correct, usable, well-designed, and sophisticated software in the R language.

[SD-1] R programming language details

- I understand non-standard evaluation (aka "Tidy Eval" or "unquoted objects"), and I can use tunneling in my functions.
- I understand functional programming, and I can use functions as objects in my code design
- I understand object-oriented programming, and I can define my own classes and methods.

Level: 3

Justification

Particularly for the latter half of the class, I needed non-standard evaluation for my functions to run properly when a user specified that the needed conclusions drawn between two particular variables, e.g. explanatory and response variables.

Lab 5 Predict from Coefs Example

Lab 5 Multiple Linear Regression Example

In both project 2 and project, functional programming was necessary to clarify my workflow and strengthen the readability of my code.

Project 2 Example - Creating the leaflet map of each countries scores required me to break up the workflow with two separate functions: getCountryLatLong() which simply had the stored lat long values for each country, and getCountryGoals() which worked on splitting up, organizing, and rejoining one of the data sets in our project to accurately display the information such that it could be ported into Leaflet.

Project 3 Example - Lines 123 and On

[SD-2] Package creation:

- I can create a folder that is installable as an R package, possibly using {usethis} helper functions
- I can document my functions using {roxygen2} style commenting
- I can write and run unit tests using {testthat}
- I can design a package that is user-friendly and has well-designed functions.

Level: 4

Justification

Package creation and documentation all demonstrated above in all projects, as well as in labs 4, 5, and 7.

Lab 5 Linear Regression Unit Tests

Project 2 Unit Tests

Reference the struggle that was lab 4. Show the initial failure of trying to use ReadRDS() to pull from the ext. int data folder which eventually only lead to reading in the names of the files as opposed to the three datasets associated with the assignment. Then manually adding and removing folders from the initial project's folder options to match the required workflow for our functions to be able to reference the datasets.

When initially working with packages in lab 4, I spent an inordinate amount of time learning how folders are organized in package creation. This rabbit hole came from me wanting to understand how a user would be able to install a package and then have access to the data that came with the package. I initially thought that creating my own "data" folder and storing the .csvs within that folder would allow the data to be user accessible (seen **here**). However, I quickly learned through **CRAN's Writing R Extensions Manual**that the data subdirectory (either manually or automatically created) "should not be used for other data files needed by" a package. It then recommended I follow more typical convention by creating the inst/extdata directory for such purposes which allow for the contents within this subdirectory to be directly copied into the installation directory (functionality specific to inst).

Matrix Operations (MO):

These objectives show your ability to manipulate data-related information in the form of vectors and matrices, rather than in high-level data structures.

[MO-1] Theory:

- I understand the difference between ordinary multiplication and matrix multiplication, and how to implement each in R
- I can implement and briefly explain the matrix equations for multiple linear regression and ridge regression

Level: 3

Justification

Although some portions of the code were pushed by my group mate Ben Laufer, we both worked extensively in each others' delegated portion of the assignments and felt that we both had a strong understanding of each others code's functionality.

Lab 5 Slr/Mlr Implementation of Matrix Multiplication

Lab 5 Slr/Mlr Gradient Descent

Lab 5 Ridge Regression

[MO-2] Object structures:

- I can convert data objects into the necessary matrix structures to perform operations on them.
- I can convert results of matrix operations to convenient data analysis formats

Level: 3

Justification

Same justification as in MO-1

Algorithms and Iteration (AI):

These objectives ask you to design code-based approaches to statistical computing problems, usually involving iteration to a stopping condition.

[AI-1] Iteration to approximate value:

- I can write a loop that updates values until a certain approximate stopping condition.
- I can explain and implement gradient descent in simple cases.
- I can choose reasonable starting conditions and step sizes.

Level: 3

Justification

Same Justification as in MO-1/2

Clever Way to Set Starting Slope/Intercept

[AI-2] Iteration to exact convergence:

- I can write a loop that updates values until perfect convergence is reached.
- I can explain and implement k-means and hierarchical clustering.
- I can identify moments of randomness or user choice in the starting conditions.

Level: 3

Justification

Hierarchical Clustering Example - Before Pre-Vectorized

Hierarchical Clustering Example- Vectorized (not fully functional)

Although this Gradient Descent implementation sets a maximum on the number of iterations allowed, a simple adjustment to the loop condition would allow a user to achieve perfect convergence. By using a while loop as opposed to a for loop, and setting the the while condition to check if the change in the slope was greater than a minimum condition (e.g. .0001).

Lab 5 Slr/Mlr Gradient Descent

In project 3, I allowed users to specify how many clusters they wanted to include when creating the elbow graph. Another easy way of implementing randomness or user input in the gradient descent would be to specify the learning rate or error boundary.

Project 3 Example - Lines 123 and on

[AI-3] Generative art:

- I can apply a variety of generative art functions to make a visually pleasing piece.
- I can explain why particular changes to the code result in particular differences in the visualization.

Level: 4

Justification

Generative Art Lab

My Most Recent Order From SloDoCo (Trajectory Curves Doughnut):

- Uses Rcpp package which allows for faster run times due to under the hood operations (or lack thereof)

- By applying translations or alternative trigonometric functions to lines 25 and 26, you are able to alter the length, width, and shape of the “doughnut”. Multiplying positive coefficients to the sin() functions leads to a “taller” doughnut, whereas multiplying positive coefficients to the cos() functions leads to a “wider” doughnut. And altering the 4 coefficients (a, b, c, and d) will either increase or decrease the magnitude in which the doughnut becomes “taller” or “wider”

The Last Braincell Graphing Judo Economic Barriers of Entry

- Entirely generated from the Jasmines package.
    - entity_circle() creates a radial display of data from which the unfold and style commands take reference
    - unfold_meander() creates the random pathing visible in the strands of the art piece
    - style_ribbon() adds the strand link texture to the path the points follow as produced by unfold_meander()

Horribly Designed Tulip Field

- Produced entirely from ggplot2
    - geom_point() produces the vibrantly colored points visible around the outskirts of the circle
    - geom_line(), after adding an alpha level to change the line’s opacity, creates that trail as if the points moved from one to the other.
    - theme_void() allowed me to remove any axis or labels in one function call
    - coord_polar() put the points on a polar (radial) axis, creating the circular effect

---

[AI-4] Creating an algorithm:

- I can invent and implement my own iterative algorithm.

Level: 3

Justification

Project 3 Example - Lines 123 and on

Hierarchical Clustering Example

Code Design (CD):

These objectives relate to making wise or clever choices in how you implement a procedure in code; including creating functions and objects, or thinking about the clarity and efficiency of processes.

[CD-1] Speed and Efficiency:

- I can recognize moments of possible slowdown in my code, and use built-in functions or parallelizing to speed them up.
- I always use and design vectorized functions whenever possible.

Level: 3

Justification

Before and After Vectorizing the Hierarchical Clustering Method

Hierarchical Clustering Example - Before Pre-Vectorized

Hierarchical Clustering Example- Vectorized (not fully functional)

As mentioned above, using the C++ port in R to more effectively compute 10,000,000 points on a trajectory curves graph.

Vectorized functions in the createElbowGraph() function as well.

Project 3 Example - Lines 123 and on

[CD-2] Object handling:

- I can make reasonable choices in my code design about when to save intermediate objects.
- I can convert objects between types and structures as needed.

Level: 3

Justification

As seen in Project 3, I use a variety of intermediate objects to temporarily store information to be passed onto helper functions or to aide the final calculation of, for example, a cluster combinations WCSS. Also within Project 3, one can see the through the progression of the elbow_dat variable (starting line 149) that I go through a variety of conversions to produce the WBSS for each cluster value:

```
#matrix
elbow_dat <- dat |> select({{explanatory}}, {{response}}) |> as.matrix()
...
#data frame
clusters <- data.frame(clusters = kmeanspp(elbow_dat, i))
...
#list of dfs
split_clust <- split_clust <- split(clusters, clusters$clusters)
...
#list of dfs
og_values <- map(split_clust, ~ getOriginalValues(.x$obs_num,
                                                  data.frame(elbow_dat),
                                                  {{explanatory}},
                                                  {{response}}))
...
#Numerical Vector
cluster_ss <- map_int(og_values, ~getWCSS(.x))
...
#Integer
wcss <- cbind(seq(from = 2, to = length(wcss) + 1), wcss)
```

Project 3 Example - Lines 123 and on

[CD-3] Supporting functions:

- I write helper/shortcut functions to streamline repeated tasks and make my code easier to read.
- I use intermediate functions to streamline repeated or looping processes.

Level: 3

Justification

When creating the elbow graph to help the user manually decide the number of clusters to choose from, I found it beneficial to break the workflow up into 3 parts: Main createElbowGraph() function that will be representative of the structure of the algorithm, getOriginalValues() which takes the list of clusters and returns the original values of the data but in the same order (to represent which points belong to which cluster), and getWCSS() which takes the within cluster sum of squares and totals them to spit out the total WCSS for that # of clusters.

Both getOriginalValues() and getWCSS() use mapping functions to streamline the looping process of returning original values from data in a cluster and calculating the WCSS. getOriginalValues() uses the standard map() function to return a list of lists so that the ordering of the clusters is preserved, whereas getWCSS() uses map_int() to return an integer vector with each value representing a single cluster's squared euclidean distances.

Project 3 Example - Lines 123 and on

Helper functions were also key to a good workflow in Lab 4 and Project 2

- Helper function that pulls the csvs from the inst/extdata directory
- Helper function that returns a country's total home/away goals scored in all world cup competitions
- Helper Function that returns a country's latitude and longitude

Lab4/Project 2 Example

[CD-4] Algorithmic process:

- My loops are clean and efficient
- Proper values are calculated to update objects and/or determine stopping conditions
- I have built in checks for possible problems or extreme cases in the algorithm

Level: 3

Justification

My work in Project 3 was a strong example of having clean and efficient loops while also having built in checks for possible user-error problems. I have a simple for-loop that iterates through the maximum number of clusters the user wants to analyze which then calculate the within cluster sum of squares for each cluster combination. In project 3, I also implemented checks (if then stop() statements) on what value of kmax the user chose, setting a lower limit of 0 and an upper limit on the # of rows present in the data provided.

Project 3 Example - Lines 123 and On

In lab 7, I produce the hierarchical clustering method. And while the draft of my code that I submitted had various redundancies in loops that could have been vectorized, it still demonstrated my ability to update stopping conditions (within a while loop) by manipulating the length of a list of lists that held the indices of each cluster.

Hierarchical Clustering Example

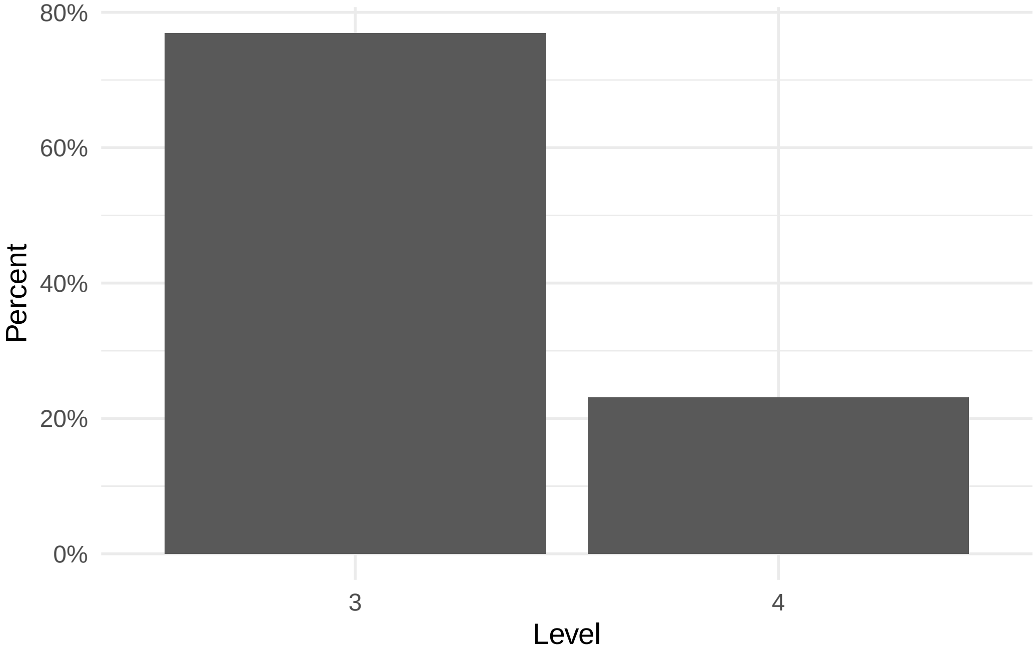
In Lab 4, we implemented a variety of user input checks to ensure that any potential users wouldn't break the code by inputting any erroneous values. In the getMatchInfo(id), I simply insure that the user is inputting a numeric value, seeing as the id values were numerical. The winningTeam check edit was one that my group mates originally wrote, but that I edited to correctly catch when the user was inputting any invalid world cup years (every four years starting 1930).

Lab 4 getMatchInfo(id) Check

Lab 4 winningTeam Check

Summary

```
Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(count)` instead.
```


80%
60%
40%
20%
0%
Percent
3
4
Level