MSc Statistical Programming 2024

Assessed Practical Assignment

Preamble

- **Assignment due date:** The report is due Wednesday, 1st Week of Hilary (22nd January 2025) at 12 pm (noon) UK Time.
- **Submission instructions:** Submission is through the Inspera system. Please make sure your practical number (*e.g.*, P123) is on the first page of your submission.
- **Submitted assignment format:** You are required to submit both a knitr code `.rnw` file, as well as a single PDF, which you get from compiling your knitr `.rnw` file. Your `.rnw` file should be self-contained, and should compile when run in this way from an empty R session (for example file "`P123.rnw`")

  ```
  knitr::knit("P123.rnw"); tinytex::latexmk("P123.tex")
  ```

- To ensure an empty R session if you're using RStudio, try restarting RStudio entirely, or clearing the workspace and restarting R. For compilation, you may assume all files mentioned in this assessment are available to you in the same directory as your `.rnw` file, when it is being compiled. Your PDF should contain the answers to the questions below, in the order the questions are set. For each part of the question, you should submit a combination of R code, the output of the code, plots or tables, and any further text elaborating on the answer, as required. Please show code in the output file when directed to do so. All code used must be included in the `.rnw` file you submit. There is no word or page limit on the report length.
- **Example:** An example question PDF, along with a solution `.rnw` file and solution PDF, are included online. You are welcome to start from the example solution `.rnw` file as a template, and modify it to include your answers to the assignment PDF. The example solution PDF was compiled using code as above.
- **Assessment criteria:** You will primarily be assessed on how well you answer each question, and whether the `.rnw` file you submit compiles on a new system. You will secondarily be assessed for the following:
  - the quality of your report including figures and tables;
  - the quality of your R code including the appropriate use of functions as well as structure, variable and function naming; and
  - the clarity of your written answers.

If your submitted `.rnw` file does not compile due to missing CRAN R libraries, these will be installed and the code re-run.

- **Code requirements:** The code you provide code must be written in R, with the exception of the `Rcpp` question with requires writing a small amount of C++. You can use any R package used in the course lectures or practicals. You can also use R packages not covered in this course, provided they are reasonably related to material covered in the course — for example, additional `tidyverse` or `ggplot2` related libraries, or things related to `knitr` like `kableExtra`. Note that you might find it helpful to wrap library calls like `suppressPackageStartupMessages(library(testthat))` to avoid printing warning messages to the output file.

U.K. House Prices

Please note: throughout this question, in your answers, unless otherwise stated, please **DO NOT** include R code in your output report.

In this question, we are going to use `Average-prices-2024-07.csv`, which contains average house prices for houses in the U.K., available from [1]. These data are available and republished under the following attribution statement:

> Contains HM Land Registry data, © Crown copyright and database right 2020.
> This data is licensed under the Open Government Licence v3.0.

We are further going to use `series-201024.csv`, which contains information about inflation in the U.K., available from [2]. Specifically, this contains historical values of "CPIH INDEX 00": the Consumer Prices Index that includes owner occupiers' Housing costs for the U.K., with 2015 as the base year, defined as 100. This data is also released under the Open Government Licence v3.0.

1. Read in the U.K. house prices data, and make a plot of the average house price for the four nations: England, Scotland, Wales, and Northern Ireland., as a function of time.

2. Make a new plot that shows the average prices for England and also the average price of houses in the four regions of Cambridge, *i.e.*, those that have "Cambridge" in the region name. Restrict the date range to only dates where both England and Cambridge level regional information are available.

3. The prices of houses in Cambridge are elevated, but are they the most elevated regions? For each region in England, as defined by regions where the first character of the `Area_Code` variable is "E", compare the per-month average price for that region with the average for England, and then take a median of the ratio (*i.e.*, if x and y are vectors representing the prices for that region and England respectively, matched by date, then you could take median(x/y)). From this, find the 10 regions with the highest values of the median ratio that you just calculated. Make a table for those 10 regions that includes the name of the region, the median ratio just calculated, the initial and final house prices for the dates examined, and the percentage increase between the initial and final prices, for houses in that region for the studied time period. Are any of the Cambridge regions included in the top 10?

4. Read in the inflation data, and by restricting it to the per-month entries, merge it into the housing data. Over the range of dates for which both inflation and the England region house prices are defined, make a single plot that shows both the increase in English house prices, as well as inflation, over the studied period. Which has risen

faster over the studied period: house prices or inflation? You may find it useful to use multiple y-axes. *Note*: the CPIH defines 2015 to be 100%, so for your written answer, report the percentage changes since then.

5. We can use the U.K. house price data to analyze seasonal patterns in house price changes. Create a beeswarm plot showing the distribution of month-over-month percentage price changes for each month of the year. Color-code the points by decade (1960s, 1970s, *etc.*) using a color-blind-friendly palette. Calculate and highlight the largest monthly price drop in the dataset, annotating its month, year, and percentage decrease. Also show the mean of each monthly average as a black point. Use appropriate axis labels, titles, and legends. (*Note*: you may need to install `ggplot` and `ggbeeswarm` for the plots; and `viridis` for color-blind friendly palettes.) Which month and year experienced the greatest month-over-month percentage price drop?

Chemical Similarity

Please note: throughout this question, in your answers, unless otherwise stated, please DO include R code in your output report. Furthermore, you don't need to validate the input arguments of any function you write.

Chemical similarity is one of the most important concepts in cheminformatics. It is useful when predicting the properties of chemical compounds, especially in drug discovery, when we want to screen a very large databases of chemical structures to find potential drug leads. This approach exploits the molecular similarity principle: *similar compounds have similar properties*.

Similarity-based virtual screening uses a query compound to search for other compounds that have similar chemical structural components in the hope that they will also have similar molecular properties. Molecules are often represented in a computer by fixed-length fingerprints, where each element in the bit vector indicates the presence or absence of a chemical substructure (such as a hydroxyl group, -OH, or an amine group, -NH_2). For example, a water molecule, H_2O, would have the hydroxyl -OH bit turned on (1) but the amine -NH_2 bit turned off (0). By comparing the bits in a fingerprint, we can quickly identify how similar two molecules are. In R, we can use the element-wise logical AND operator, '&', to determine if both bits at the same position are turned on; and the element-wise logical OR operator, '|', to determine if either bit is turned on:

```
# Simple fingerprints example
fp1 <- c(1,1,0,0)
fp2 <- c(1,0,1,0)
fp1 & fp2

## [1]  TRUE FALSE FALSE FALSE

fp1 | fp2

## [1]  TRUE  TRUE  TRUE FALSE
```

One of the most widely used fingerprints in cheminformatics is the Morgan fingerprint, better known as circular fingerprints or Extended Connectivity Fingerprints (ECFP) [3]. The method uses a radius parameter, r to visit atoms that are bonded to the current atom and update the identifier of that atom to reflect its sub-structural context. The most commonly used radius value is $r = 2$, but by convention the diameter is used in ECFP notation, so we would refer to it as an ECFP4 fingerprint.

In cheminformatics, we typically compute the similarity between the fingerprints of two molecules using the Tanimoto coefficient [4]; but in fact, the same statistic was described by

Jaccard in 1912 [5]. The Jaccard index measures the similarity between two finite sample sets, A and B, and is defined as the ratio of the size of the *intersection* and the size of the *union* of the same sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

The Tanimoto similarity is calculated using bit vectors such as the ECFP4 fingerprint. If X and Y are the bit vectors of two molecules we would like to compare, X_i is the i^{th} bit of X; and $\wedge$ and $\vee$ are the bit-wise AND and bit-wise OR operators, respectively; then:

$$T_s(X,Y) = \frac{\sum_i (X_i \wedge Y_i)}{\sum_i (X_i \vee Y_i)}$$

A Tanimoto similarity, T_s, value of 1.0 indicates the two molecule's fingerprints are identical, while 0.0 indicates they are completely different. In this practical, we will implement the Tanimoto similarity calculation in a number of different ways; benchmark and test them; and use them to help in the design new peptidomimetic drugs.

Amino Acids and Peptidomimetics

There are 20 standard naturally occurring amino acids (AA) that constitute the building blocks of all proteins in living systems. Each of these amino acids is encoded by three consecutive nucleotides (A, T, G, or C) in the genetic code. The sequence of these amino acids influences the three-dimensional shape of the resulting protein. Amino acids have different chemical properties. Some are hydrophobic, *e.g.*, while others are hydrophilic: this causes proteins to fold so that the 'water-hating' amino acids become buried, while the 'water-loving' amino acids prefer to sit on the surface of the protein next to the aqueous environment they are found in. A *peptide* is a short chain of amino acids linked by amide bonds, and is much smaller than protein. A wide range of peptides are known, including plant, antibiotic, fungal, venom, anticancer, immune related and inflammatory peptides, and many more.

One strategy in drug discovery is to design new molecules that resemble peptides or the parts of proteins that are known to be involved in a therapeutically relevant pathway. We call these drugs *peptidomimetics*. A prominent class of peptidomimetic drugs include those used to treat viral infections like HIV (*e.g.*, saquinavir, indinavir, atazanavir) and COVID-19 (*e.g.*, nirmatrelvir): they are specifically designed to mimic viral peptides but bind more tightly to their viral protein target, thus inhibiting them and blocking viral maturation.

Medicinal chemists have synthesized thousands of "non-natural" amino acids (NNAA) and these are available for purchase from suppliers like MolPort. So if we can identify non-standard amino acids that are chemically similar to standard amino acids, we can use them

to design new compounds that mimic natural substrates or proteins, but might bind better than their natural counterparts: this is the start of peptidomimetic drug discovery.

1. Read in the CSV file `AA-NNAA-FP.csv`. It contains information about natural amino acids ("AA") and non-natural amino acids ("NNAA"). Each row represents one molecule. (i) What are the column names, and (ii) how many natural and non-natural amino acids are there? *Note*: there are multiple forms of the natural amino acids provided in the CSV file: the *zwitterionic* form has charged amino and carboxylic acid groups. (*Hint:* the `IsNatural` column is a boolean...)

2. Split the data frame you created from the CSV file into two new dataframes, `dfAA` and `dfNNAA`, one for the natural AAs and the other for the NNAAs, respectively. You will notice that the last column of these tables contains space-separated binary fingerprints — these are actually ECFP4 2048-bit fingerprints. Write tests using `testthat` to confirm that all fingerprints in the original data frame are 2048 bits long. (*Hint:* the `strsplit()` function might be useful.)

3. Implement the Tanimoto similarity calculation for two fingerprints, using a for loop in R to iterate over each pair of corresponding bits. It should return a numeric value. Write tests to check that the returned value is never negative or greater than 1.

4. Implement the Tanimoto similarity calculation again but this time using vectorized R instead of for-loops.

5. The `bitops` R package provides functions for bitwise operations on integer vectors. Use the `bitops::bitAnd()` and `bitops::bitOr()` functions to implement a third version of the Tanimoto similarity calculation.

6. Write a fourth version of the Tanimoto similarity calculation using `Rcpp`. Note: you will have to install a compiler if you haven't already.

7. Using the fingerprints of the first natural amino acid (`dfAA` and of the first non-natural amino acid (`dfNNAA`, write code to check that all of your implementations generate the same Tanimoto similarity result.

8. Now benchmark these four Tanimoto similarity calculation methods (R loop, vectorized R, `bitops`, and `Rcpp`):

   (a) Give a table of the timings in seconds for each implementation called a different number of times: 1000, 5000, 10000, 25000, 50000, and 75000, 100000;

(b) Generate a plot with the title, "Performance Comparison of Tanimoto Similarity Methods", an x-axis label of "Number of Repetitions"; a y-axis label of "Time (seconds)"; and include a legend indicating which color is which method.

(c) Using 'big-O' notation, describe the complexity of the four algorithms.

(d) Which is the fastest implementation, and why?

9. Using your fastest implementation of the Tanimoto similarity calculation, compare all of the natural amino acids in `dfAA` with all of the non-natural amino acids in `dfNNAA`. For each amino acid in dfAA, find the top three most similar NNAAs (*i.e.*, sort by descending Tanimoto similarity). Create three separate tables: one for the most similar NNAA, one for the second-most similar NNAA, and one for the third-most similar NNAA. They should all have three columns: "AAName", "NNAAName", and "Similarity". Sort each table by the name of each natural AA alphabetically, and use the captions:

   (a) Most Similar Non-natural AA

   (b) Second Most Similar Non-natural AA

   (c) Third Most Similar Non-natural AA

10. MolPort is a chemical compound supplier, and the names of the non-natural amino acids are actually purchasable compounds. Given your results, do you think MolPort sells natural amino acids? Explain.

11. There are two more possible, *parallel* implementations of the Tanimoto similarity function: (i) using forks with `mclapply()`; and (ii) using sockets with `makeCluster()`. As we saw in Lecture 6, the Microsoft Windows version of R only supports parallel programming with sockets; but Linux and macOS also offer the forks option. Write a fifth version of the Tanimoto similarity function using sockets; and if you are able to access a Linux or macOS system, *optionally* implement a sixth and final version using forks. Complete the benchmarking by comparing all your implementations. Plot a comparison of timings of each method, iterating over increasing numbers of the natural AAs each time incrementing in steps of 2, each time comparing each AA's fingerprint to the non-natural AAs in `dfNNAA`, incrementing in steps of 5. You are effectively creating a grid of (2, 4, 6, ... 20) AAs versus (5, 10, 15, ... 50) NNAA comparisons, or 20×50 to give a total of 1000 comparisons. Plot the timings in seconds for each experiment and each implementation against the total number of comparisons. Fit both a linear and a quadratic model to each set of timings, and report the coefficient of determination, R^2, for each implementation and fitted model type. Comment on the relative efficiency of each implementation versus the total number of comparisons.

References

1. UK House Price Index: data downloads July 2024; `https://www.gov.uk/government/statistical-data-sets/uk-house-price-index-data-downloads-july-2024#download-the-data`; Accessed: 2024-11-16.

2. Office of National Statistics, Consumer Price Inflation Team, CPIH INDEX 00: ALL ITEMS 2015=100, Series ID: L522; `https://www.ons.gov.uk/economy/inflationandpriceindices/timeseries/l522/mm23/previous`; Accessed: 2024-10-20.

3. Rogers, D., and Hahn, M. "Extended-Connectivity Fingerprints", *Journal of Chemical Information and Modeling*, **50**: 742-754, 2010.

4. Tanimoto, T. T. "An Elementary Mathematical theory of Classification and Prediction", *Internal IBM Technical Report*, **8**: 1957.

5. Jaccard, P. "The distribution of the flora in the alpine zone", *New Phytologist*, **11**: 37-50, 1912.