Stat 360 Project Rubric

Brad McNeney

Project overview

The project is to consist of R code in your group's project folder on GitHub and documentation submitted as a PDF file to a Crowdmark assessment. The code and documentation are due at 11:59pm on Tuesday April 20.

Code on GitHub (40 marks)

1. (2 marks) A README.md file should include your group members' names and a list of the files in your project folder, with a short description of the contents of each file.
2. (13 marks) The main `mars.R` file should include the `mars()` function and any others, such as `fwd_stepwise()`, that are called by `mars()`. Arrange your functions in a "top-down" manner, with higher-level functions appearing first, followed by successive levels of lower-level functions. You are graded on the following criteria:
   - Data structures (2). The input data structures should be a formula, data and `mars.control` object. The output data structure is an S3 object of class `mars` that inherits from class `lm`.
   - Correctness (5). There should be no errors in the code and it should correctly implement MARS (see *Test suite* below).
   - Readability (4): The steps and logic of your implementation should be clearly layed out. It should be easy for someone else in the class to read your code and understand what is going on.
   - Efficiency (2): Take steps to avoid computational inefficiencies, such as excessive copying of large R objects.
3. (15 marks) User interface (methods). Include one file for each method you implement for MARS objects. The criteria for the user interface are:
   - Correct (4 marks). All methods work correctly.
   - Comprehensive (5 marks). You should implement `anova()`, `plot()`, `predict()`, `print()` and `summary()` methods for `mars` objects. (You can use the implementations of `residuals()` and `fitted()` that you inherit from the `lm` class.)
   - Familiar (4 marks). The user interface should look familiar to someone who has used `lm()` and `glm()`.
   - Use your creativity (2 marks). This is the part of the project where you can combine your skills as an R programmer and data analyst. What features would help users gain insight into their data and the fitted model? Think of something we have not discussed as a class and implement it. (**Note** I haven't thought of anything myself and may remove this part of the rubric.)
4. (10 marks) Test suite. A file `test.R` that includes at least three worked examples with non-trivial data. The examples should show a user how to call `mars()` and illustrate the methods that you have written for MARS objects (see *User interface* above). Computations for each example should take no more than one minute. I will provide one example dataset that the project marker will use to verify that your code works correctly. You should provide at least two other examples. The criteria for the test suite are:
   - Correct (4 marks). All examples run without errors.

- Comprehensive (4 marks). Taken together, the examples should illustrate **all** of your functions/methods.
- Interesting (2 marks). Use the most interesting data you can find. Trivial examples will get no marks.

Documentation (20 marks)

The documentation is for your `mars()` function and should include the following sections. The sections are those of a typical R documentation file. See the help file for the `lm()` function for an example and Section 2.1.1 of the Writing R Extensions Manual for further details. (The manual is really too much detail for your project, but I provide the link in case you are interested.) You can write the documentation using a word processor or RMarkdown, or have the documentation generated from roxygen2 comments in your source file. If you go the roxygen2 route, you may need to copy the help file to HTML for printing; e.g., from an R session with your documentation directory as working directory, issue the following R commands:

```
sink("mars.html"); tools::Rd2HTML("mars.Rd"); sink()
```

1. (2 marks) Description (brief) – a one- or two-line description of what the function does
2. (1 mark) Usage – how to call the function
3. (2 marks) Arguments – a list of arguments and their meaning
4. (5 marks) Details – a precise and detailed description of what the function does
5. (3 marks) Value – a description of the function's return value
6. (1 mark) Author(s) – your name(s)
7. (1 mark) References – a reference to the Friedman paper and any other sources you think are necessary
8. (2 marks) See Also – a brief description of the methods written for MARS objects
9. (3 marks) Examples – Three complete examples of how to use your function and the methods written for its output. You can use examples from your test suite (see *Test suite* above).