## Project Rubric ST441/541

| | Does not meet expectations | Meets Expectations | Exceeds Expectations | Total points available |
|---|---|---|---|---|
| **Content** | | | | |
| Consistency of effort | No evidence of planning, work is done last minute, issues are opened but not resolved. Commits are infrequent. | Progress is made each week as evidenced by the opening of issues, comitting of work, and closing of issues. | | 15 |
| Response to feedback | Feedback is ignored or disregarded without justifcation. | Feedback is responded to, or if deviated from, justified. | | 5 |
| | | | | |
| **Organization** | | | | |
| Project Organization | Files and directories do not follow any conventional structure | The submitted repository follows the organizational best practices as outlined in "Good enough practices for scientific computing" | | 4 |
| Overview | README exists but doesn't help orient a viewer | README gives summary of project and points a viewer to the important parts of the repo including the project report and presentation slides. | | 2 |
| Reproducibility | Project is not reproducible due to: absolute file paths, missing dependencies, missing files or code that does not run. | Project is self contained and reproducible | Efforts to formalize or check the reproducability of the project, may include: make, the liftr package, experimentation on other systems. | 4 |
| | | | | |
| **Code** | | | | |
| Correctness | Code has errors that lead in incorrect results. | Code seems to be correct, based on some informal checks in code. | Key functions that are created have some formal testing for correctness. | 2 |
| Documentation | Code is hard to follow because there is no structure or narrative, there is no indication of the relationship between files. | Code is interleaved with narrative to provide structure and documentation. Relationships between files are documented. | | 2 |
| Efficiency | Code includes common sources of inefficiency: unnecessary iteration, not preallocating objects in loops, etc | Code doesn't contain obvious sources of inefficiency | Some effort has been made to identify and improve the efficiency of functions. May include: timing experiments, implementations in C++, reimplementations of existing functions | 2 |
| Style | Code is copy-paste-edited to perform repetitive tasks | Functions are combined with iteration to reduce repetition in code. | | 2 |
| Functions: Ease of use | Functions created as part of the project do not follow basic best practices: undescriptive names, no thought to argument order or defaults, no guidelines on how functions should be used. | Functions created as part of the project follow basic best practices: descriptive names, data arguments come first, good defaults for detail arguments, comments used to give basic documentation. | Effort has been made to improve the usability of functions created in the project. May include: formal documentation using roxygen, using S3 classes to refine the display and behaviour of output. | 2 |
| | | | | |
| **Communication** | | | | |
| Report: clarity | Report is hard to follow because there is no structure or due to grammatical or spelling errors. | Report clearly describes research objectives and progress made towards addressing them. | | 5 |
| Report: conciseness | Report exceeds page limit. | Report is within page limit. | | 2 |
| Presentation | Presentation is too long, or doesn't include a slide with title, name and github repo, or fails to give motivation or discuss results. | Presentation is within time limit and contains overview slide. Presentation covers motivation and results. | | 3 |
| | | | | |
| | | | **Total Points Available** | 50 |