Linting Names Using Regular Expressions

Problem: Naming inconsistencies led to duplication of clone entries in Benchling and subsequently clone files in AWS S3 buckets.

Goal: Create an automated linter to ensure all names match a prespecified format that will send results via Slack or email.

To begin with, I explored Benchling at Zach’s request, as he hoped to get me involved in some basic data curation on the platform. I downloaded Registry data from Benchling to mark all clone entries whose library names were three digits long, as library sizes had expanded into the four-digit range. This was becoming a problem for the scientists, as clones would no longer be displayed in order. (Library 1234 would come before Library 234.) Working in Excel, I padded the three-digit libraries with a leading zero and bulk-uploaded the corrected clone names and library categories to Benchling. A similar problem was found to exist in the well names, although this could not as easily be fixed with a bulk upload, as Benchling would automatically remove the leading zeros in the well category. When I was doing this, I noticed another issue: the naming inconsistencies led to duplication of clone entries in Benchling, e.g. a clone entity might be entered as both “ALB-L097-523-B6” and as “ALB-L0097-523-B6”, but with different data associated with them in the Warehouse. This subsequently caused a similar problem with the clone files in the AWS S3 bucket. My task was to create an automated linter to ensure all names were properly formatted that would send results via Slack or email.

 To do this, I needed to become familiar with version control, for which our team uses Gitea. I practiced forking and cloning repositories, making branches, committing, pushing, and pulling, all via the command line. I was added to the Lodo organization on Gitea and formed a new linting branch in the regex repo. This repo uses R and Python; I worked with R.

 I began by looking at the code already in the repo and trying to familiarize myself with regex. (I would later find a regex cheat sheet that probably would have been very helpful at the beginning. C’est la vie.) I created Perl regexes for each subcategory of the clone names and then wrote some code that identified and changed improperly named clones. Around this point I switched from working with Benchling data to working with AWS S3 file names, as part of the Benchling issue was outsourced. I used my adapted code to write functions to modify or create dataframes with naming issues flagged under a new column and a pipeline demonstrating function usage. This was originally hardcoded with a specific local file. I then learned how to pull files directly from the AWS buckets via the command line and utilized that to standardize my pipeline. zdk helped me formulate my code as a package. I created tests for my package using the testthat package. From here, I wrote an executable script so that running my linter could be automated. I then looked at the Slack API and the Slackr package in order to print the results directly to the #dev channel on Slack.

 My top takeaways from this project: Stack Overflow is a godsend. The command line gets slightly less scary the more you use it. There’s a lot of things I “wouldn’t know” so I should just ask zdk.

Things I Learned:

* Regexes



Here, six regular expressions are assigned to variables. These variables dictate patterns that should be followed for an expression to match. For instance, mol\_rx specifies a pattern of 3 capital letters. lib\_rx specifies a pattern of a capital ‘L’ followed by 4 digits. file\_rx currently allows anything, because there is not yet a set pattern for the file names.

* Slack API: Slackr



This code uses the Slackr package in R to print a dataframe directly to the #dev channel using slackr\_bot and to add a file to the #dev channel using slackrUpload. slackrSetup() reads from a file specifying api\_token, channel: #dev, username: lodo-dev-bot, and incoming\_webhook\_url.

* AWS S3 buckets



This is a terminal command that reads the data from the AWS S3 bucket specified into a text file called “s3file”. Amazon Web Services offers clients a variety of cloud computing services. S3 provides virtually limitless object storage space on the cloud, and this space is organized into buckets. Lodo stores its clone data in one such bucket.

* Version control via Gitea



This is the typical procedure for adding a new version via the command line. git status shows the current branch and any changes that have been made since the last commit. The changes I wish to track are added with git add and then committed with an explanatory message, in this case “total files outputted” because the change made here was to display the total number of files linted in the Slack output. Finally, the committed changes are pushed to the origin branch on Gitea, where a pull request can be made to integrate my edits.

* R
	+ Packages





I was working to expand Lodo’s Regex package. Admittedly, I did little of the structural work and concentrated on the package content. I wrote various functions for the package. These were grouped into three categories/file types: the linting methods, the tests for these methods, and an executable script for using these methods. Above are corresponding examples of the first two types.

* + Scripts



Here is an R script. The shebang at the top invokes Rscript to make the file executable. Of course, for this to be meaningful, there should be something being executed, hence to call to main() at the bottom. Within the script, commands to the terminal can be executed, as seen above.