Authors - JT Kashuba
Group - Keyboard Warriors
Last Modified - 2/10/21

A document used to keep track of things the group needs to finish before final submission. *Note: this document is organized so that the most recent items for the agenda can be found at the top of the document and go backwards in chronological order*

FINAL Agenda (week of Feb 8)

FINAL GOAL: COMPLETE SUBMISSION AND SUBMIT PROJECT TONIGHT (2-9-21)

Revised Goal: Finish before 5:30pm 2-10-21, just need Programmer_Documentation, User_Documentation, and finalizing other docs (SRS/SDS, etc)

BEFORE SUBMISSION: Go over project submission requirements and make sure nothing is being overlooked

---

Things JT wants to check-off on before they are "final versions" to be uploaded:

- Installation instructions
- SRS/SDS
- User_Documentation
- README.md
- Double-check table of contents page numbers

Make sure documentation references that csv files (to be used in our system) MUST start with 1 header line which describes what each column in the file represents

---

Finalize formatting in files to prepare for converting to PDF for upload to repo:

- *Directory named Documentation contains:*
  - SRS
  - SDS
  - Project Plan - Directory containing:
    - roles/responsibilities
    - Individual progress
    - Documentation from group meetings

Agendas from group meetings
benchmarks/goals
Documentation specifications
Function checks
Questions for client
Programmer_Documentation
User_Documentation
Presentation Slides for client

User_Documentation.pdf

---

install instructions (1 thing: rename to install_instructions.txt)

Add up total hours in individual progress reports and list total at the top of the doc

---

something to look out for (which is tedious but necessary) when doing the editing for finalizing files before conversion to PDF: MAKE SURE THE FONT IS UNIFORM

I just noticed several parts of the SDS that were still in Arial and had to be changed to TNR as well as adjusting the font size. little things like this matter

Times New Roman size 12 is the standard for everything other than section headers which can/should be larger font size

Another thing we need to standardize: indentation of paragraphs vs no indentation of paragraphs

there are a ton of inconsistencies in this regard so we should discuss which we want to use and STICK TO A STANDARD

One more note: I'm guilty of this myself - the term "User/Users" vs "user/users". We need to decide if there are cases this should be a Proper Noun vs an improper noun, or if they should all be uniform.

Programmer_Documentation.pdf or .txt
Everyone move your individual function docstrings/explanations/example function calls into “Programmer Documentation” and we’ll go from there

Project_Plan.pdf (from Task_and_Assignment_Breakdown.xlsx

Cameron: Finish commenting his code

JT: Finish the README, edit Noah's SDS section 3, diagram captions for SDS, finish the references to SECTION 666

River is donezo

Noah: Finish SRS 3.1, 3.2, section 3 SDS

Nick: Test cases for read_from_file, split_data, denoise. Check empty pickle files for read_from_file

---

cameron/river/nick assigned sections in SDS section 3 AND section 4

Commenting code (River, Nick, Noah, Cameron)

River testing preprocessing functions in tree (can meet w/ Noah or JT to help speed this up)

Cameron fix modeling and meet w/ JT + SDS sections, meeting w/ Noah

Nick - test cases, SDS sections, meeting w/ Noah

(When creating a tree, root node is always set as read_from_file)

Make sure the things at the top of files we have all the proper info: short description, author(s), group (Keyboard Warriors), last edited date

Update page numbers at beginning of SRS/SDS docs for table of contents

References/Acknowledgments

Cameron and JT need to finish section 3 of SRS, Noah will help with 3.2 Functions for validity checks

---

Tree use case diagrams

Integration tests

Finish SDS and SRS

For JT: in SRS make sure there are 2 use cases for add_node and replace_node, although similar, also do execute_path and execute_tree

---

For/From 2-3-21

Functions that still need to be tested:

NICK
logarithm
cube_root
split_data
read_from_file - edge cases
denoise

plotting functions got messed up. Was working with multiple types of input files, single/multiple column csv files - but now it's not working on multiple column csv files. Also can no longer plot multiple time series'

RIVER
assign_time
design_matrix
Clip
Only remaining issue with these 3 is handling multiple columns
If there are 3 columns, impute_outliers (for example) needs to affect and return all 3 columns

CAMERON

difference
ts2db & db2ts

NOAH
execute pipeline
execute tree

After individual function testing, we'll start testing the system.

End of meeting:

Could use help determining what other documentation still needs to happen, trying to formalize a list

Discuss presentation, plan how to present

discuss when to meet next / what still needs to be done
Finish testing, prepare presentation, complete documentation
If there's time: more functionality for reading_in_files, more functionality with plotting

Game plan for Progress Meeting 2
==========

For/From 2-1-21

Functions that haven't been started:
RIVER
Design_matrix x2
River will post on slack to ask about the 2 different design_matrix

CAMERON
Ts2db
Will write the db2ts function

NICK

split_data
mse(y_test: np.array(), y_forecast: np.array())
mape
smape

Tuple containing a time series AND a 2nd Tuple of 2 Numpy arrays (orange and green) which each contain multiple numpy arrays
= (time_series, ([[1,2,3], [2,3,4], [3,4,5], … []], [[4,5], [5,6], [6,7], … []]))

==========

Agenda for 1-27-21

Schedule time for a 2nd group meeting this week. LOTS to cover.
Schedule group meeting w/ Flores.

TimeSeries object:
Preprocessing:

- Which libraries are we using?
  - Pandas? Numpy? scikit-learn?
- Cannot find documentation for premade denoise() function
- clip() documentation? What does it do?
- Will the ts parameter in assign_time() be a list or a dict?
  - If func is given a time series with no time stamps, then how does that work with a dict (since you cannot have empty keys)

Misc.:

- Requirements.txt? Should probably start creating a list of dependencies

Trees:

- Could we have every time series function take all possible parameters even if it doesn't need them. That way when executing a node in the tree I can just pass all of the parameters into the function call without needing conditional checks on the function type.

Nosetests for testing?

<u>The SRS/SDS documentation actually needs to be considerably more detailed than previously assumed:</u>

Emailed Prof Flores about SRS 2.6 Use Cases, asked if we need to document every single function. If so, we'll need everyone to do a write-up for each of the functions they're assigned to. Once we have all 26 functions "use cases" documented, JT can edit the wording/grammar/style to make the SRS document cohesive. Same with SRS 3.1 External Interfaces, 3.2 Functions. (3.2 example for 2. Sequence of operations in processing inputs: will need to be explained

conversationally, not “specifically”... meaning explaining the sequence of operations in layman’s terms rather than the nuts and bolts laid out in computer programming language.)

Looking at the SDS again, we might need to assign 4. Software Modules & 5. Dynamic Models of Operational Use to the people working on each function too.

==========

Nick is saying we need to address some low level technical aspects of the functions to finish the working parts of the application.

We need to figure out the best format for the time series data in its raw form. What’s the best way to store the node information?

Might be good to keep all the data in a Pandas dataframes

==========

People are accessing the time series object differently:

River is calling the time series as if it were a dictionary in assign_time

Cameron is accessing it as if it were a list

In the time series implementation, Nick has the time series stored as a dictionary. When he goes to use functions, he converts it into a list and calls np.mean (time_series.py line 43) on that list.

The time series object has 2 values, the time and the data at that time. Let’s agree on a standard for how to use/access these objects.

Cameron is suggesting we have 2 numpy arrays, one is a times array and the other is a values array. (decided to use Pandas dataframe)