Quantitative Research Modeling Library

Project Plan v3

Team
sdMay19-06

Team Members
Josiah Anderson -- Meeting Facilitator
Doh Yun Kim -- Scribe
Gabriel Klein -- Report Manager
Drake Mossman -- Communication Manager
Jacob Richards -- Quality Assurance Manager
Nathan Schaffer -- Overseer

Client
Joseph Byrum
(Principal Financial Group)

Advisor
Srikanta Tirthapura

Contact
sdmay19-06@iastate.edu
https://sdmay19-06.sd.ece.iastate.edu

Last Updated
2 December 2018

Table of Contents

List of Figures

List of Tables

1 Introductory Material

1.1 Acknowledgement

Team 06's Faculty advisor: Srikanta Tirthapura
Team 06's client: Principal Financial, primarily Benjamin Harlander and Vishnu Vemuru
Interviewed Data Scientists: Josh Zimmerman, Q Mabasa, Krisoye Smith,
Yaoliang He, Markus Sauter
We would like to thank Srikanta Tirthapura for being our advisor and aiding us in this project.

1.2 Problem Statement

Principal Financial has a relatively new team of data scientists and interns who work within their Global Investments Department. The goal of this team is to analyze equity data to develop new quantitative strategies the company can use to make informed investment decisions. This team does not have much consistency or standardization in many of their common tasks, which slows down productivity and leaves excessive room for error by requiring duplication of efforts by individuals working on different yet overlapping tasks. Through interviewing members of the data science team at Principal Financial, we found three main areas where the process (or processes) being used were not optimal.

- The first area was in the aggregation of stock level data to the portfolio level. This process can be very time consuming, especially for less experienced team members. Nearly every project involves data aggregation and it is often done many many times within the life of one project.
- The second area was in the handling of predictive models. A lot of similar models are used on multiple projects, and currently they are being rewritten for each new project. This causes some duplicity that could be reduced.
- The third area is in the visualization of the data. Currently there is not a lot of visibility into the individual steps of the process used to create prediction models for stock data.

The inefficiencies in these three areas existed across many of the projects handled by the equities team at Principal, but for the scope of this project we decided to focus our efforts into solving these problems for one process specifically. Multiple student teams across the United States are working on the Dynamic Risk Premium 2.0, which from now on will be referred to as DRP 2.0. The goal of the DRP 2.0 is to create predictive models for stock data using machine learning models. Our task is to create a uniform software pipeline that can be used to connect each of the various components in this multi-step process. This solution needs to be easy to integrate into the current workflow for minimal disruption and save the company time by

introducing standards and automation. Our hope is to create a flexible, user-friendly pipeline that can be used for projects across Principal's Global Investments team.

1.3 Operating Environment

Since our project is entirely software-defined, the operating environment is defined by the systems it may run on and the data which it will be manipulating.

During our current development, our working environment is an AWS server we have been given access to, which contains decades of weekly and monthly stock data up to August 2018. The data is static, since it hasn't been updated for months. While it is mostly complete data, there are a few anomalies with NaN values that must be taken into consideration.

We anticipate the final environment of our library to be work computers within Principal's data science teams and servers running automatically scheduled tasks. When the operating environment is individual computers, the data may come from a local csv containing some view into a more expansive database, or it could come from a more direct database connection within the same script. Once DRP 2.0 moves past the prototype stage within Principal, parts of our library will likely be utilized in automatic tasks to predict the performance of current stocks given weekly-updated data. We expect the list of factors, companies involved, and completeness of the data for both of these cases to be fundamentally similar to those of the database we are working with now.

1.4 User Interface Description

DRP 2.0 will take the form of both a Python library and an R package wrapping a range of functions for data manipulation and aggregation within the predictive modeling domain. This library will be used by data scientists within our client's team at Principal Financial as well as student groups and interns which this team commonly works with. Users will have a range of abilities in programming, data science, and quant research. For this reason we will strive to write function interfaces which are intuitive and don't require excessive work to set up arguments. We also want to strive for flexibility, anticipating ways the pipeline may be adapted in the future, without obscuring the interface for the most common use cases. Users will also be interacting with our API and other documentation we create to help guide them in how to utilize our library for various purposes.

1.5 Assumptions and Limitations

Assumptions:

1. All data will be reachable through a SQL database consisting of stock level data taken from Factset and Bloomberg.
2. Users will have basic knowledge of either Python or R.
3. Users will have access to a personal computer to run scripts on.

Limitations:

1. Some data scientists using the pipeline may not be expert programmers.
2. Some data scientists may know either Python or R well but not both.
3. There exist already completed components with standardized inputs and outputs.

1.6 Expected End Product and Deliverables

Our final expected end product will be a unified pipeline for the DRP 2.0, standardizing the process of how data is passed between each stage.

The deliverables will be:

1. The DRP 2.0 pipeline framework as Python/R packages
2. Documentation on using the pipeline's functionality for each version of the library

DRP 2.0 Pipeline

The pipeline will allow a user to perform data science functions by taking raw stock level data from a database and transforming it into portfolio simulations and predictions using user input. The user will be able to specify the stock universe, portfolio strategies, prediction models, and factor policies among other variables to run different simulations and see different predictions. It will be implemented in Python and be accessible as both a Python package as written and as an R package through a wrapper. There will also be a number of places where diagnostic or similar interfaces can examine the data as it's being processed for visualization and examination. This deliverable will be delivered on May 10th, 2019 as the Spring semester ends.

Documentation

The documentation for the pipeline will consist of guides for using either the Python or R packages. Each will contain essentially the same material, with potentially different function names or argument types as the languages differ in requirements. They will cover all relevant functions and data types necessary for a user to operate the pipeline from start to finish. Both will be delivered as pdf documents for users to browse as necessary while using the packages. This deliverable will also be delivered on May 10th, 2019 as the Spring semester ends.

1.7 Related Work / Market Survey / Literature Review

The desire for creation of the DRP 2.0, for which we are designing a pipeline, is directly caused by shortcomings of the DRP, a previous product of Principal Financial. The DRP's purpose, similar to its successor, was to utilize a multiplicity of descriptive factors and predictive models to accurately and consistently predict future performance of stocks, ultimately aiding portfolio management decisions. The primary features of this product which the company hopes to improve in DRP 2.0 include flexibility and scalability. Our research into the functionality of the current DRP consisted of reading through internal Principal Financial documents describing it.

Although DRP 2.0 directly builds on the functionality of DRP, we will not be reusing any code, but are rather starting our work on the DRP 2.0 pipeline as a greenfield project. Some of the dashboards and applications that will interact with the pipeline may or may not be built upon previous work in the DRP, but that doesn't affect our project since we are only concerned with defining the interfaces with these applications.

Beyond research into the previous version of the DRP, our team investigated competitor products with similar purposes to the DRP 2.0. We looked into products for both Python and R which previously existed with purposes similar to ours. Quantopian and Quantiacs are Python libraries for quantitative data analysis. Another library, caret (Classification And REgression Training), exists for creating predictive models of data in R.

After looking through tutorials and the API for Quantopian [1], we discovered a surprising amount of similarities between its functionality and the the requirements for DRP 2.0. Specifically, Quantopian presents itself as a Python toolkit with functions for testing predictive models. It offers its users a select set of historical data, including minute intervals of trading summaries and some corporate fundamental data for free, with additional data available with subscription. Functions within the library are individually documented in addition to overarching guidance for how to use their library, IDE, and data. While there is significant feature overlap between Quantopian and DRP 2.0, DRP 2.0 will provide benefit to Principal by allowing more specific customization of function parameters, the ability to work in either R or Python, and a disassociation with external datasets or code which isn't open-source.

Caret is an open-source package for R that streamlines and standardizes common tasks in the creation of predictive models [2]. Unlike DRP 2.0 and Quantopian, caret is not specific to the financial sector. Rather, it has functions which can be applied to a plethora of predictive modeling contexts. For this reason, its documentation, though thorough, is also more broad than that of Quantopian. Principal already uses this library in some of their research and development. Because of the open source nature of the package, we may be able to also utilize some of its interfaces and structure as guidance when creating our pipeline.

2 Specifications and Analysis

2.1 Proposed Approach

2.1.1 Functional Requirements

1. The pipeline will consume raw stock level data from an AWS Postgres database
2. The pipeline will be able to aggregate data to factor portfolios

3. The pipeline will be able to build models on factor portfolios, generate predictions, and calculate model performance
4. The pipeline will be able to score stocks using a factor policy
5. The pipeline will be able to simulate portfolio returns
6. Users will be able to customize the stock universe, factor portfolio strategies, model algorithms, and factor policy of the DRP 2.0 pipeline
7. Each function of the pipeline shall be able to be called and run independently
8. The pipeline will be able to handle missing or invalid stock level data
9. It will be able to interface with the DRP 2.0 dashboards
10. It will be able to run in both Python and R

2.1.2 Non-Functional Requirements

1. (*Performance*) The pipeline will be able to handle up to several GB of data
2. (*Performance*) The pipeline should be able to handle millions of observations and hundreds of factors
3. (*Maintainability*) The pipeline will have documented and standardized inputs and outputs for each of its functions and interfaces
4. (*Usability*) The pipeline interfaces will be intuitive to a novice data scientist
5. (*Accessibility*) The pipeline will be useable by novice programmers
6. (*Security*) The pipeline will not leak confidential data to outside sources
7. (*Compatibility*) The pipeline will be able to perform the same functionality as provided existing scripts

2.1.3 Constraints Considerations

The programming languages primarily used by data scientists within our client's team are R and Python. The interface of our program/library will be need to be either R and Python because that is what Principal's data scientist team is most familiar with.

2.1.4 Technology Considerations

We have few technology considerations for this project. It will need to be able to run on our client's work computers primarily, which means it can't be immensely computationally heavy. However, the client's current scripts do not put significant strain on the machines, and we don't expect our project to either. As mentioned in the constraints section, our client's employees are most familiar with Python and R, so we will focus on these languages if they need to be exposed directly to the code.

2.1.5 Security Considerations

There are no significant security considerations for our project. Our chosen approach doesn't require any communication with the internet or other computers, remaining entirely local.

Additionally, our approach doesn't persist any sensitive information on the computer after running. Thus any and all information involved is discarded after runtime. For these reasons we do not have any further considerations for the security of the project.

2.1.6 Safety Considerations

There are also no significant safety considerations for our project. Our approach doesn't contain any sort of physical component, meaning there is no danger of physical harm to a user. Additionally our project is meant to handle financial information and not any sort of safety-critical data, so there is no danger of the results bringing about harm either. The worst that could happen is a loss of money due to a poorly created model which we will of course be taking precautions against using our testing plan in section 3. In any case, we have no further considerations for injury in the use of our project.

2.1.7 Standards

As our project is solely software based and also exclusively for use by our client internally, there are no standards we are explicitly required to follow. That said there are a few standards we have encountered that would still be useful for us to at least take guidance from.

IEEE 12207-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering--Software life cycle processes
This standard defines a framework of processes for developing a software system among other activities. It's intended to be customized for the user's specific purpose, and doesn't include specific details for many of the processes it mentions. Instead it gives the purpose, outcomes, and tasks associated with each process in general and leaves it to the user to define the way it will be applied to their project [3].

IEEE 29119-4-2015 - ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing--Part 4: Test techniques
This standard is about the various test techniques and test coverages that are used during the development of a software. It intends to help educate testers, test managers, and developers on different test techniques and the appropriate situations to use them. It is divided into the test design techniques one can use and the test coverages one should use for each test [4].

IEEE 1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation
This standard defines processes for verifying and validating (V&V) software projects such as ours. Additionally it allows for four levels of desired integrity, and each only requires certains parts of the whole set of processes. There are also guidelines for documentation for accountability. This standard is intended to help make sure the project is acceptable for the client and meets all desire requirements [5].

While each of these standards contains a lot of useful detail applicable to our project, they are also all very complex and likely very difficult for our small team to implement. It would take our team several weeks of work simply to understand and begin using the entirety of each document as they each are dozens if not hundreds of pages long. They also require defining many of the processes as they are only frameworks. These standards seem more fit to be used by a much larger team with a dedicated managerial staff. That said, we may reference them and use bits and pieces without conforming to the standards explicitly.

2.1.8 Possible Risks and Risk Management

Risk Map	Consequence	Likelihood	Risk Description	Project Impact	Risk Area	Symptoms	Risk Response	Response Strategy
H	5	D	Significant conceptual misunderstanding of pipeline processes	Most, if not all of the work done up to this point will be scrapped, significant setback in schedule	Scope, Schedule	We present our work to Principal and they tell us the functions aren't what they wanted	Mitigation	Continuous feedback from Principal at every stage of development will significantly decrease the likelihood
S	4	D	One of the PowerBI dashboards doesn't correspond well to any of the interfaces or functions we've defined	Extend scope of project to rework or add to function library	Schedule	Code for dashboard is not easily reworked to use DRP 2.0 library	Mitigation	Early interactions with the dashboards will allow us to identify incompatibilities when we still have time to correct for them
S	4	C	Don't get to the end of the development by our deadline	Have to present unfinished project	Scope	Semester finished, project not complete	Monitor and Prepare	Follow schedule; Include sufficient buffer
S	4	D	The first prototype is not received well	Considerable time from the risk buffer must be taken out to make changes to the prototype	Organizational, Schedule	During the presentation of the first prototype Principal does not like it	Mitigation	Make rapid changes to the prototype with all team members thriving to finish the redesign as quickly as possible
S	3	A	A team member gets busy with other course work	A team member who is busy with other work might not be able to finish their tasks on time or hand in substandard work	Organizational, Schedule	A team member is unavailable at work at designated times or are missing deadlines	Mitigation	Redistribute the work the team member's work to the other people
S	4	C	Procedural Risk: Improper process implementation, conflicting priorities, or lack of clarity in responsibilities	Team members code being incompatible with each other, lack of understanding of goals	Implementation	Incompatible code	Mitigation	As a group discuss the problems and identify the solutions
M	3	C	Code written in Python can't be easily ported to R package	Reduce the scope of the project so that our library is only in Python	Scope, Schedule	When attempting to port to R, we find large parts of the code that would need significant rework to function in R	Mitigation	Investigate what specific aspects of Python code might be difficult to port and try to work around them
M	2	B	A team member gets stuck on a part of their task and can't get work progressed	The person who is stuck cannot get work progressed, potentially risking delays	Schedule, Organizational	A team member is not getting any sufficient work done a task even after working on it for a (set amount of time)	Monitor and Prepare	Consistent communication amongst team about blockers; Assistance from team members early when necessary
L	2	D	Changes submitted for one task breaks code that was previously finished	The code will not function right and will need to be fixed before moving on, taking additional time and effort	Schedule	Previously tested code not working as expected following push to git repository	Mitigation	Ensure problem is caught early if it occurs through automated regression tests in a continuous integration framework

Table 1: Risk Description Chart

Level	Descriptor
1	Insignificant
2	Minor
3	Moderate
4	Major
5	Catastrophic

Table 2: Risk Consequences Mapping

Level	Descriptor	Description
A	Almost certain	Almost certain Expected to occur in most circumstances
B	Likely	Will probably occur in most circumstances
C	Moderate	Should occur at some time
D	Unlikely	Could occur at some time
E	Rare	may occur only in exceptional circumstances

Table 3: Risk Likelihood Mapping

	Consequence				
Likelihood	1	2	3	4	5
A	M	S	S	H	H
B	L	M	S	H	H
C	L	M	M	S	H
D	L	L	M	S	H
E	L	L	L	M	S

Table 4: Risk Mapping from Likelihood and Consequence to Severity

Key	Risk Map
H	High Risk
S	Significant risk
M	Moderate risk
L	Low risk

Table 5: Risk Severity Mapping

2.1.9 Feasibility Assessment

Technical Feasibility
Our current scope of the project is creating a unified pipeline for Principal for their DRP 2.0. We are working with the data science team of Principal and working with them on this project. Our team will take existing processes that Principal has created and create the unified pipeline based off those processes and standardizing the inputs and outputs of each process.

With respect to the current state of the project, our project looks technically feasible. We are not interacting or changing the different processes itself, and focused on what we can do which is create a more organized pipeline for Principal.

Economic Feasibility
Please also look at 2.1.9 Cost Considerations. The only costs our project is incurring are travel expenses, and the travel expenses look perfectly reasonable at this stage. Unless some major change of scope happens during the project lifetime, currently our project is economically feasible.

Legal Feasibility
Looking at the current technologies we are using, we are focused on using open source tools and the tools Principal are providing to our team. We expect to see no legal problems during our project.

Operational Feasibility
The project scope is a the scope our team has chosen after carefully interviewing different employees at Principal and thinking of plausible projects our team could do. We ended up on deciding to narrow down the focus of the project on one of the many projects Principal has and focus on creating a unified pipeline for that project.

Our project itself is also a prototype of what Principal could do or expand in the future, so it will not impact or cause any problems for the current operations that Principal is using. Principal is currently looking for a prototype not a proof of concept or an actual full scale product, so our project satisfies the operational feasibility.

Scheduling Feasibility
We currently have two semesters to complete our project. As long as our team follows the current schedule we have set and not encounter any major problems during phase 2 and 3 of schedule, we foresee we will be able to finish our project on time.

2.1.10 Cost Considerations

Our project does not involve buying any software, hardware, physical parts, and etc.
The costs we do incur are all travel costs between traveling between Iowa State University and Principal, which is located in Des Moines.
In total, our team believes we will make around 10 trips to Principal. The distance between Iowa State and Principal is 38 miles. Our team needs two cars to travel to Principal. Parking at Principal costs 3 dollars everytime we go there. Using the Standard mileage rates of 2018 which is 0.545 USD per mile, one trip will cost 88.84 USD. Since we are making 10 trips, our costs estimation will be 888.40 USD.

2.1.11 Proposed Milestones and Evaluation Criteria

- Investigate
    - Interview different data scientists that work at Principal to gather information about their coding practices.
    - Research example code and find commonalities between them.
- Prototype
    - Taking information we receive from the interview process and create a prototype.
- Test
    - Run prototyped code with real simulation code.
    - Test for bugs and errors in code.
    - Fix and adjust any problems with code.
- Final Project
    - Present final product to Principal executives.

2.1.12 Project Tracking Procedures

The project management systems we chose to utilize are Gitlab issues and GroupMe. Using Gitlab issues allows us to track progress of specific tasks by allowing us to assign tasks and establish deadlines to each individual. Gitlab Issues allow us to also review tasks at our bi-weekly meetings and help keep all team members on the same page. We also chose to use GroupMe for instant messaging which allows for quicker communication between group members.

2.2 Statement of Work

2.2.1 Task Objective

Our objective for this project is to take data manipulation and prediction processes currently used by our client and more easily connect them with an automated pipeline. These include the splitting of training and testing data from a chronological set, normalization, non-linear feature engineering, cross validation, model creation, and of course prediction. The training data must be kept safe from contamination due to the temporal nature of the desired predictions. Our project will allow data scientists to quickly and easily create and test models using a variety of parameters and without needing to write boilerplate code or worry about errors in the data manipulation processes.

2.2.2 Task Approach

After discussing several alternatives, the approach we've decided on is to create a pipeline framework to meet our objective. The pipeline will provide hooks along with a standardized system of inputs and outputs that allow users to plug in the scripts they'd like to use for testing with little hassle. We'll develop the pipeline using Python, a common data science language.

Strengths:

1. Python is a common general purpose language
2. Python is already used by some of the client's employees
3. Easy to plug in and swap various components
4. Allows client to focus on developing modules instead of the whole process at once
5. Saves a lot of boilerplate code for users

Weaknesses:

1. High level of parameterization needed could require verbose function calls or object initialization
2. Requires users to learn a new set of functions to interface with
3. May require current script inputs and outputs to be changed to conform to the standard

Overall we feel that the strengths of this approach outweigh the weaknesses, and that other approaches aren't significant improvements (See Section 2.2.3). While users will have to learn the set of functions we provide, this is true for really any solution we come up with for automating processes. Similarly, there may be significant explicit configuration required resulting in verbose code, but this is again true for any approach that wants to provide that level of customization.

On the other hand, Python is a powerful general purpose language with significant popularity in data science. Additionally the library will be easily integratable into the client's current workflow, as it should simply replace large blocks of code with a single lines that have the pipeline do the rest of the work. The only barrier here is for current script practices to conform to the new standard. To ease the transition we'll make sure to spend ample time developing an intuitive standard that can integrate with current scripts without too much modification.

The pipeline will automate much of the testing process shown below in Figure 1. The processes and connections between them will be run using function calls, while using user-defined scripts at the chosen interfacing points. There are also points at which data can be accessed and visualized if desired.

The internals of the library will make heavy use of the other libraries our client is already making use of including numpy, pandas, sklearn, and matplotlib. Each of these already implements a lot of very useful functionality, so our library will mainly be focused on combining them to perform larger tasks.

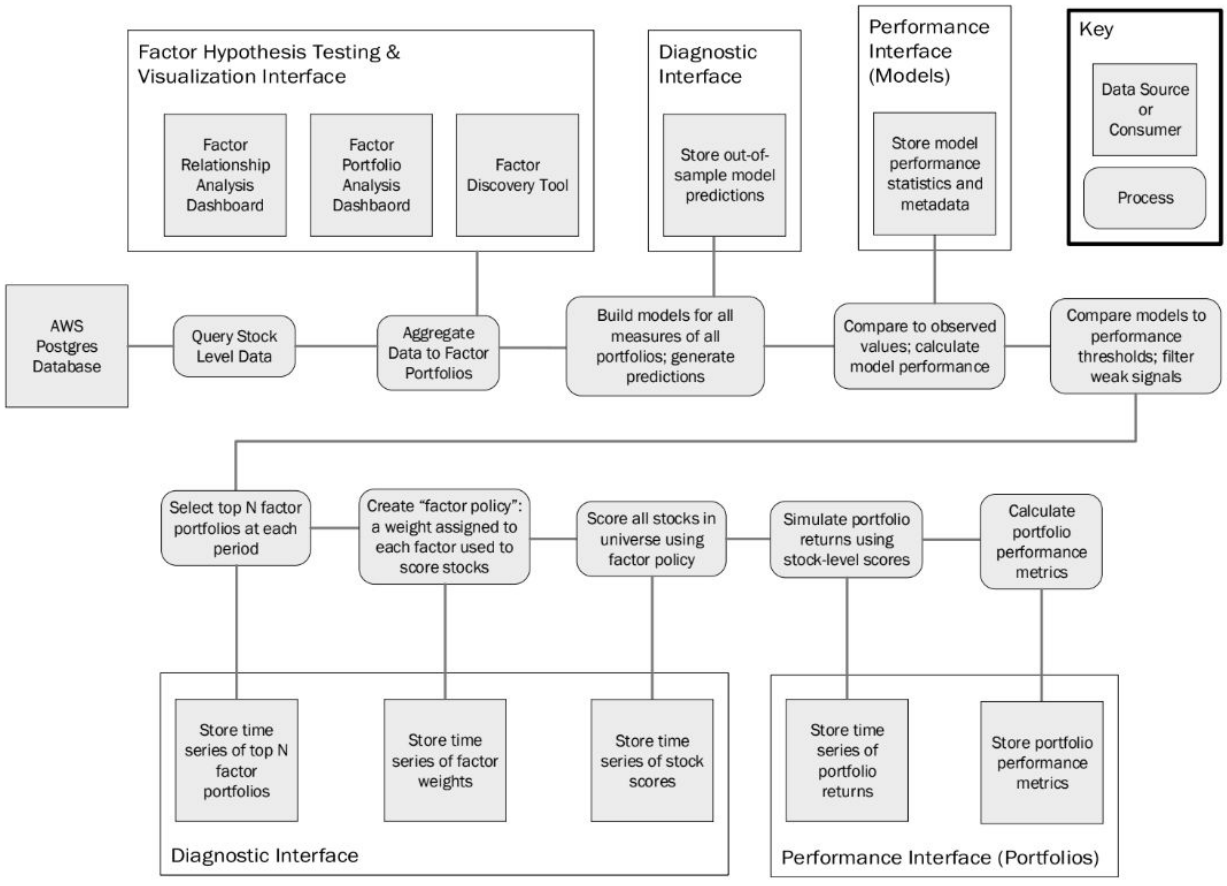

Figure 1: Pipeline Process Diagram with Interface Hooks

2.2.3 Alternative Approaches

The following consist of several approaches we considered but eventually rejected in favor of our current plan.

Standalone Application

One way we considered meeting our objective was by creating a standalone application that could run various data manipulation processes through a UI. The user would feed data to the application and choose the processes to be run on it. The UI would also allow for a significant amount of custom configuration for users to adjust as needed.

Strengths:

1. Doesn't require any coding to process data
2. Able to keep track of user preferences and state between sessions
3. Intuitive to use
4. Can easily save and reload models and results
5. Doesn't require installation of dependencies
6. Doesn't require user to use a particular language for other data manipulation

Weaknesses:

1. Difficult to feed results back into code
2. Unfamiliar concept for client's employees
3. Additional functionality requires building more UI
4. Less platform independent
5. Doesn't update along with packages automatically

We didn't choose this approach mainly because our client wants the end product to be available to many users easily and without a high learning curve. This application would require users to learn an entirely new interface that doesn't even mesh well with their existing workflows. Normally data is passed almost exclusively through code, so adding an application that has to import and export the data into the mix is somewhat awkward.

Browser Application

Another approach we considered was a browser application that could run all of the data manipulation processes remotely. The user could upload data and choose processes to be run on it. The application would provide a UI that could configure the processes as necessary. These processes would then be run on a server and the results sent back to the user.

Strengths:

1. All processes can be run on a server with above average processing power and memory

2. Results can be saved remotely and shared with other users
3. Doesn't require the user's computer to be available while running
4. Doesn't require coding to process data
5. Able to keep track of user preferences and state between sessions
6. Intuitive to use
7. Doesn't require installation of dependencies
8. Doesn't require user to use a particular language for other data manipulation

Weaknesses:

1. User must be online initially and to get the results
2. Big data must be uploaded and downloaded often
3. Difficult to feed results back into code
4. Unfamiliar concept for client's employees
5. Additional functionality requires building more UI
6. Requires a server to be accessed from

Similar to the previous approach, the browser application was not chosen because of the learning curve and adjustment to workflow required. With the added complication of uploading and downloading the data, this approach could serious disrupt the flow of data without significant benefits.

2.2.4 Expected Results

Once the project is complete, we expect to have a functioning Python pipeline that allows users to test scripts modularly and to perform significant data manipulations such as engineering nonlinear features, cross validation, or backtesting. This will be possible through a variety of function calls made available to the user. By stringing a few of these calls together, we expect the user to be able to create complex models able to predict responses based on the features given.

The pipeline will be portable and easy to distribute. It will simply need to be added using a package manager and then imported into the code to be used. Users can expect to be able to use all the functionality immediately and with little hassle. The code will also be written consistently and with maintainability in mind, allowing for new functionality to be added as necessary in the future.

3 Testing and Implementation

There are primarily two parts to our plan to test our product: requirement testing and user testing. The first will test each of the functional and non-functional requirements we have defined for our deliverables, ensuring correctness of the algorithms, and adequate functionality within its use context. The second will test our design and documentation for our users' perceptions, ensuring that its use is intuitive and self-explanatory. Outcomes from either of these test types could lead to reworking our product and retesting.

3.1 Requirements Testing

The primary goal of requirements testing is to ensure that our automation of tasks remains correct (does not change the expected outputs for each simulation) and performs well within its context.

3.1.1 Functional Testing

1. Compare 10+ runs of rolling window calculations between our solution and the previous method, varying selectable parameters, including, but not limited to:
    a. Window train size
    b. Window test size
    c. Buffer size
    d. Predictive model / algorithm
    e. Each run should produce the same result from legacy code to new solution

2. The user can observe the output of the tests run in at least one manner.

3. Augment varying amounts of additional data to a partially-constructed model and reverify the model's functionality. The model must still functions with 5+ trials.

4. Run simulations with different models provided by Principal, verifying the correctness of their results.

5. Calculate each of the models individual performance to the performance threshold.

6. Calculate "factor policy" weight and assign to each factor.

7. Run 5+ simulations with data from each of FactSet and Bloomberg, verifying their results

3.1.2 Non-Functional Testing

1. Benchmark the runtimes of models in both old scripts and new scripts over tests for functional requirements and ensure that the runtime for the new version isn't longer (within a threshold)
2. Run a simulation with a data set of 2-5 Gb and verify it finishes.
3. Run a simulation over data consisting of > 100 features and > 1,000,000 observations, verifying that it completes.
4. Test with 6+ data scientists, getting feedback to ensure the product does save them time.

3.2 Usability Testing

Usability testing is paramount to our project because if the product we produce isn't intuitive and convenient to use, it will defeat one of its primary purposes of saving our users time.

3.2.1 Process

To test our library's usability, we will be putting it in the hands of data scientists, especially the members of Principal's team whom we interviewed at the onset of the project. We will ask them both to use it in their typical tasks and, if necessary, to try it in contexts and tasks that would more thoroughly test its functionality. After a fixed length of time testing our product, we will meet with them to elicit their feedback.

3.3 Results

No testing has been done in our project so far.

4 Estimated Resources and Timeline

4.1 Estimated Resources

4.1.1 Personnel Effort Requirements

Each team member is expected to put in around 7 hours a week for this project. Each hour of work needs to be filled with hard work and focus in order for this project to succeed.

4.1.2 Other Resource Requirements

Other resources needed for this project are: previous code from the team before us, examples of the different tests they run, and examples of the data format.

4.1.3 Financial Requirements

Since this project is a software only project, there is no cost needed for hardware or any special equipment. The data needed for the project is provided by Principal at no cost.

4.2 Project Timeline

Our timeline is divided up into 4 phases. The first phase being the planning phase, the second the first prototype phase, the third being the third prototype phase, and the final phase being testing/documentation.

Task Number	TASK TITLE	START DATE	DUE DATE	DURATION	10-01 to 10-05					10-08 to 10-12					10-15 to 10-19					10-22 to 10-26					10-29 to 11-02				
					M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
1	Interview	10/1/18	10/5/18	4																								
2	Plan out Designs	10/8/18	10/19/18	11																								
3	Research Most Used Functions	10/8/18	10/12/18	4																								
4	Create Generialized Functions	10/15/18	10/19/18	4																								
5	Consult Design Plan	10/22/2018	10/26/18	4																								
6	Refine Design Plan	10/29/18	11/2/18	3																								

Table 6: Gantt Chart Phase 1

Phase 1

The first phase of our project involved planning out the design and functionality of our project. One of the most important parts in this phase was the interviewing the employees of Principal. Due to the scope of the project being vague at the start, our team needed to decide the scope of the project before anything could start. After the interviews, the rest of phase one is focused on trying to create a concise design for our project.

As of the current time, we have successfully met all of the phase 1 goals. While planning the design took a little longer than expected, the other parts went well so we managed to finish phase 1 on time.

Task Number	TASK TITLE	START DATE	DUE DATE	DURATION	11-05 to 11-09					11-12 to 11-16					11-19 to 11-23					11-26 to 11-30					12-03 to 12-07					12-10 to 12-14					01-14 to 01-18				
					M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
7	Plan out Prototype #1	11/5/18	11/9/18	4																																	
8	Create Library Functions	11/12/18	11/30/18	18																																	
9	Show/Refine Prototype #1	12/3/18	12/7/18	4																																	
10	Continue Refining Prototype #1	1/14/19	1/18/19	4																																	

Table 7: Gantt Chart Phase 2

Phase 2

Phase 2 is where we start prototyping our design. Due to the nature of our project, the prototype at this stage is focused on creating the unified pipelines for the first parts of the DRP 2.0. Focusing on the data aggregation and data query parts, we will first create the prototype for these components, constantly getting feedback from Principal and refining the project.

We are currently in phase 2. We are in the progress of creating the first part of the prototype. So far we have created the first few modules in the DRP 2.0 diagram.

Task Number	TASK TITLE	START DATE	DUE DATE	DURATION	01-21 to 01-25					01-28 to 02-01					02-04 to 02-08					02-11 to 02-15					02-18 to 02-22					02-25 to 03-01					03-04 to 03-08				
					M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
11	Plan out Prototype #2 (based off #1)	1/21/19	1/25/19	4																																	
12	Create Prototype #2	1/28/19	2/8/19	10																																	
13	Refine Prototype #2	2/11/19	2/15/19	4																																	
14	Create Final Design	2/18/19	3/8/19	20																																	

Table 8: Gantt Chart Phase 3

Phase 3

Phase 3 is the second phase of the prototype designs, focusing on the later parts of the DRP 2.0 processes. We will continue to focus on the creation of the unified pipeline here for the later stages, and also work on the standardization of all the process in this phase too. Same as phase 2, we will be in constant contact with Principal to continuing refining our prototype. Towards the end of phase 3, we will work on bring all the different processes together in one unified pipeline.

Task Number	TASK TITLE	START DATE	DUE DATE	DURATION	03-11 to 03-15					03-18 to 03-22					03-25 to 03-29					04-01 to 04-05					04-08 to 04-12					04-15 to 04-19					04-22 to 04-26					04-29 to 05-03				
					M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
15	Testing	3/11/19	4/5/19	24																																							
16	Functional Testing	3/11/19	3/15/19	4																																							
17	Non-functional Testing	3/25/19	3/29/19	4																																							
18	Usability Testing	4/1/19	4/5/19	4																																							
19	Documentation	4/8/19	4/12/19	4																																							
	Risk Buffer			0																																							

Table 9: Gantt Chart Part 4

Phase 4

Phase 4 is the testing and documentation phase. For the first part of phase 4, we will focus on testing our unified pipeline to make sure it works as intended. After testing has finished, we will work on documentation of the pipeline and how it works. If we have time towards the end of phase 4, we will also focus on working together with Principal to teach our clients how to use our product.

4.3 Work Breakdown Structure

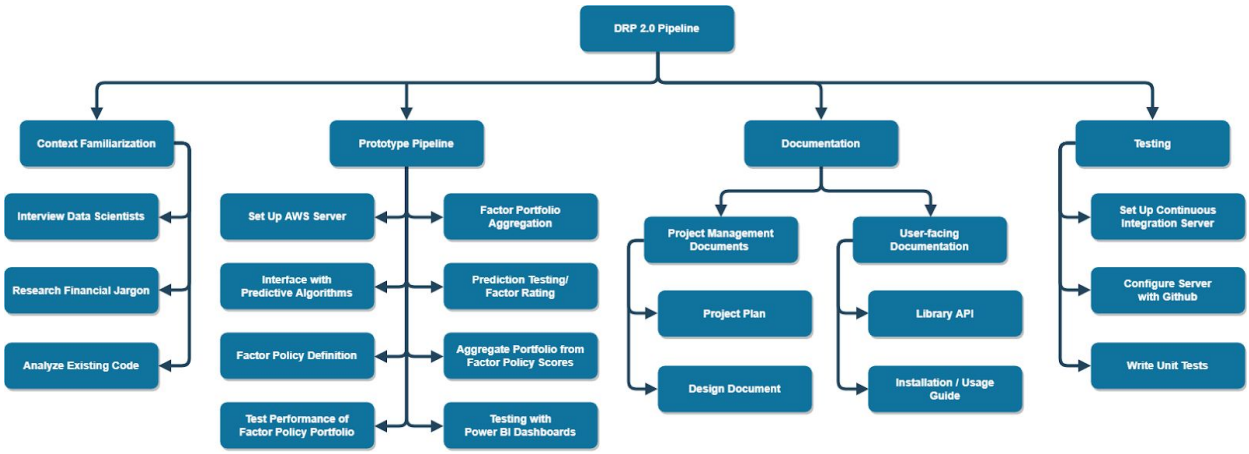

Figure 2: Work Breakdown Structure

5 Closing Material

5.1 Conclusion

This project aims to help the data science teams at Principal Financial work at a more efficient pace by providing a pipeline, creating a unified way of aggregating data and running models on that data.

Our team will create a pipeline framework that will automate some of the tasks Principal's Data Science team already does to allow them to save time on their redundant tasks, and to further streamline their work. This will allow for consistency and less mistakes in the process of aggregating data, constructing, running, and using the output from those models.

In the first couple months of the project we interviewed multiple Principal employees, discussed their preferred directions of the project, and spent time clearing up any confusion regarding the details of the project. During the discussions with our client we were given multiple options for our project. We decided to work on the DRP 2.0 pipeline to allow their different modules to communicate and connect in an efficient manner. After completing a multitude of interviews and meetings, we began working on the technical side of the project. Our team has currently developed a function to retrieve specified stock data from the database and organize it into portfolios defined by the user. This now allows the user to categorize and specify the stocks they want through the parameters of a function instead of rewriting the code multiple times. An R language wrapper was also created for our current functions so Data Scientists who are more familiar with R can use our code effortlessly. We are currently in the process of setting up Continuous Integration within gitlab to prevent us from breaking the project during our various merges and pushes to the repository.

By continuing to follow the steps outlined in this proposal, we can provide a more versatile and automated system of predictive stock analysis for the data science teams at Principal Financial.

5.2 References

[1] “Quantopian: The Place For Learning Quant Finance,” *Quantopian*. [Online]. Available: https://www.quantopian.com/. [Accessed: 26-Nov-2018].

[2] M. Kuhn, “The caret Package,” *Github Sites*, 26-May-2018. [Online]. Available: http://topepo.github.io/caret/index.html. [Accessed: 26-Nov-2018].

[3] “ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes," in *ISO/IEC/IEEE 12207:2017(E) First edition 2017-11*, pp.1-157, 15 Nov. 2017

[4] “ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing--Part 4: Test techniques," in *ISO/IEC/IEEE 29119-4:2015*, pp.1-149, 8 Dec. 2015

[5] “IEEE Standard for System, Software, and Hardware Verification and Validation," in *IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/Cor1-2017)*, pp.1-260, 29 Sept. 2017