Validate Documentation

Release 1.2

UNCW Cyverse

May 08, 2018

Contents

Validation is an evolving process and the documentation should reflect the experiences of users as they discover new and useful bits of information. updating is easy, and you can share comments and questions at this forum.

Validation is the positive control for your experimental data analysis. You have control over how long data analysis takes. Completing your positive controls first allows you to be ready for your data when you get it. Why validate a particular data analysis?

- Trust but verify: check that a widely used method will work for your data.
- Have analysis pipeline optimized; the moment you've collected all your data you can start the analysis (saving weeks or months to complete afterwards)
- Fewer requests from reviewers for changes to your data analysis methods
- Be confident you are doing the best possible data analysis

The CyVerse Validate workflow is versatile enough to handle you importing your GWAS tools, functions, or simulation methods to use in conjunction with the existing software. Speaking of imports, Validate is capable of handling a range of file formats, and each piece of software is very customizable. You have multiple options for adjusting your output each step of the way.

Finally, since the analysis are run on the NSF-supported national XSEDE computer systems, scalability of simulations is a non-issue; the computational demand for your simulations will scale up as your needs increase. All of your necessary files may be uploaded and accessed from the CyVerse Data Store.

CHAPTER 1

Version 1.2 Features

- Added python scripts to streamline common tasks and open new options for scaling and automation.

CHAPTER 2

Version 1.0 Features

Version 1.0 features include:

- The ability to test prediction
- Incorporate the following prediction applications: GenSel, BayesR, BLR/BATools, ridge regression
- Updated Demonstrate to handle prediction graphics
- Managing GWAS and prediction applications in parallel
- Incorporate D. Hand's H-measure to the current performance metrics
- Additional documentation to onboard new developers and statisticians

CHAPTER 3

Documentation

3.1 Useful Terms

Depending on your specialty coming into the lab, you may not need all of the terms on this page. You may not need any! Either way feel free to skip to the relevant section for whatever information you may be lacking: Biology Terms, Statistics Terms, Computer Science Terms, next page.

3.1.1 Organizations

- Stampede: One of the most powerful supercomputers in the world, dedicated primarily for scientific computation.
- Cyverse: National life science consortium (previously known as iPlant). This organiztion has many functions and purposes but primarily seems to allow for ease of collaboration.
- TACC: Texas Advanced Computer Center (TACC). "The center's mission is to enable discoveries that advance science and society through the application of advanced computing technologies."
- Xsede: "Xsede is a single virtual system that scientists can use to interactively share computing resources, data and expertise."
- Agave: "Agave is an open source, platform-as-a-service solution for hybrid cloud computing. It provides a full suite of services covering everything from standards-based authentication and authorization to computational, data, and collaborative services."

3.1.2 Biology Terms

- Genotype: The set of genes an organism possesses.
- Phenotype: The physical characteristic of an organism. Though normally biologists refer to a collection of physical characteristics in defining phenotype, our typical studies limit the phenotype definition strictly to quantitative traits such as height or weight. Most tools only use one of these quantitative characteristics for analysis

for the sake of consistency and for statistical accuracy, though other options for phenotype are entirely possible (e.g. binary options like disease/no disease).

- GWAS: Stands for _G_enome _W_ide _A_ssociation _S_tudies. Genome wide association studies examine variations in gene structure (genotypes) to see if these variations can be associated with certain physical traits or phenotypes. Because these studies involve entire genetic sequences, large sample sizes with a high number of individuals are required for success. The Stapleton Lab is focused largely on this area of research.
- SNP: Stands for _S_ingle _N_ucleotide _P_olymorphism.
- QTL: Stands for _Q_uantitative _T_rait _L_oci analysis.
- GxE: Stands for _G_enotype _E_nvironment interaction.

3.1.3 Statistics Terms

- Variable: As the name implies, a variable varies from object to object. To be specific, a variable is a symbol representing potentially any characteristic or object that can be either measured or counted. Some basic examples of variables (when dealing with people) might include eye color, height, or the number of pets one owns.
- Quantitative vs. Qualitative: A quantitative variable is a value that can be quantified, or measured and represented numerically. Height or weight are good examples of quantitative traits. Quantitative variables can be either discrete-only taking on a set number of values-or continuous-able to take on any value within a certain numerical range.
- Distribution: The description of the underlying observed or theoretical frequency of certain values occurring for a given variable. These frequencies, like the variables they describe, may be either discrete or continuous. Some well known discrete distributions include binomial and geometric. Continuous distributions include the normal distribution (AKA the “bell curve”) and the gamma distribution.
- Hypothesis Testing: A statistical method for testing the probability of a given hypothesis being true. A hypothesis test deals with both a null hypothesis (the hypothesis being tested, generally says that findings are produced by chance) and an alternative hypothesis(a possible trend or hypothesis explaining the data formation, typically says that findings are the results of some difference or trend in data). The five steps of a hypothesis test are: 1) State hypotheses 2) Determine significance level to compare p-value against (usually 0.05) 3) Calculate test statistic (standardized value of your sample’s attribute assuming the null hypothesis is true) 4) Calculate p-value 5) State the conclusion
- P-value: The value one refers to when making a final conclusion on certain statistical tests, particularly hypothesis tests. To be specific, a p-value is the probability of obtaining a certain sample (i.e. the sample chosen for the test) under the assumption that the null hypothesis is true. Since a p-value is a probability, it will always lie between 0 and 1. If a p-value is small, the probability of your findings being produced under the null hypothesis is small, meaning that the null hypothesis probably isn’t true. Thus, if a p-value is below a specified significance level, the null hypothesis of your test may be rejected.
- Statistical Significance: Typically used when describing a hypothesis test for differences in samples,
- Correlation: The measurement of the strength and direction of a linear relationship between two quantitative variables. Correlation is typically indicated by the correlation coefficient statistic r, and this statistic can take any value from -1 to 1. A positive correlation (r value between 0 and 1) indicates that as one variable increases, the other tends to increase also. In contrast, a negative correlation (between -1 and 0) indicates that as one variable increases, the other tends to decrease. Note the keyword “tends” in the previous explanations. Correlations guarantee nothing with respect to increase or decrease of variable values. Furthermore, and this is an important point, correlation does not equal causation! Meaning that just because two variables are related does not mean that one necessarily causes another to increase or decrease.

3.1.4 Computer Science Terms

- Quality Control: The process of making sure software is working correctly. Quality Control typically includes a series of tests to measure software performance and/or ease of use. Developers and tester may have a certain set of standards that the tester may want the software to adhere to, or a list of criteria that the program must satisfy.
- Documentation: The necessary documents explaining how to use the software. Typically, documentation includes instructions for using each of the functions included and perhaps a tutorial for going through some of the main features of a given program.
- Permissions: The authorization to read, write, or execute a program. Certain programs or files may not require access by everyone, so developers may set up permissions on their code such that only a certain group of people may use it freely. While certain permission barriers may be bypassed (e.g. through the "sudo" command on Linux or "Run as administrator" on Windows), permissions are usually set up for a reason, so it's best to just contact the developer if you need access to something blocked off.
- Github: A version-control website frequently used for collaboration on software design. Github allows for both public and private access to project repositories and keeps a history of different updates that users make to their programs. Anyone can make contributions to a public repository on Github. It also allows for you to set up webpages based on a project or organization and allows transfer of software permissions to other users or organizations.
- CLI: _c_ommand _l_ine _i_nterface (CLI). This is a way for a user to interact with a computer/application/etc. by way of the command line. (ex. old dos commands)
- GUI: _g_raphic _u_ser _i_nterface (GUI). This is a way for a user to interact with a computer/application/etc. by way of a graphic (ex. using microsoft word)
- API: _a_pplication _p_rogramming _i_nterface (API). A set of tools, resources, or routines which make developing applications on a given system easier. Following the logic from the two precedding definitions, API could be exaplined as the way in which interfacing happens when programming applications. For example, iPlant's Agave API allows for easier development of scientific apps on the Stampede supercomputer.
- Terminal: A portal for interacting with and recieving data from a computer.
- Auth-tokens: Authentication tokens are used as a kind of digital key which allow other computers to determine and verify user's identities.
- Node: (Within the context of the Stampede super computer) These are the speficic compute nodes which jobs are sent to.
- Allocation: (Within the context of this workflow) This a set amount of space that is given to you to run jobs on the stampede supercomputer
- iRODS: "The Integrated Rule-Oriented Data System (iRODS) is open source data management software for storing, searching, organizing, and sharing files and datasets that are large, important, and complex." Generally this is used so users can interact with the cyverse data store from their home computers.
- iCommands: A collection of commands to be used within IRODS for submitting or retrieving data from the data store.

3.2 Getting Started with Validate

It's helpful to think of Validate not as a single app (Winnow) but as a multi-stage workflow. A typical session will involve either generating a sample data set or importing your own from the CyVerse Data Store for analysis with your preferred GWAS or prediction tool. The corresponding outputs are then analyzed with Winnow, and rendered with Demonstrate.

These quick start tutorials assume basic command-line knowledge (start with the Software Carpentry tutorial if you're new, or refer to this cheat sheet for a quick reference). While the guides assume you have access to a Cyverse account and Stampede Allocation, these instructions should be easily adapted to run Validate on your own hardware.

3.2.1 Account Setup Guide

While it is possible to Validate using your own hardware, you'll benefit from greatly enhanced speed and reliability using high-speed compute resources freely-available from the open-source science community. You'll need to set up accounts with Cyverse, the Texas Advanced Computing Center (TACC), and Extreme Science & Engineering Discovery Environment (XSEDE) to gain the most benefit from these tutorials. If you're working in an academic laboratory, you should have little trouble registering for accounts and allocations available to users with a .edu email address. Bear in mind that large allocations may require approval for a specific research need.

Cyverse

Register at https://user.cyverse.org/. This portal will be your central access point for Atmosphere and the Discovery Environment, and provides some forums for community and support.

- Your default Data Store allocation is 100GB, which can be upgraded upon request to up to 1TB. For more information see http://www.cyverse.org/data-store.
- To gain access to Atmosphere images, which can be a convenient way to quickly get started in a linux environment pre-configured for GWAS analysis, you'll need to request access through your cyverse dashboard.

TACC

These guides refer to running the Validate workflow on the Stampede supercomputer at the TACC. You'll need to make an account at https://portal.tacc.utexas.edu/ to get started. Unless you are a professor or in charge of a collaborative project at your company/institution, simply list the PI option near the end of the application as *Ineligible*. After you've created your account, you'll need to configure two-factor authentication back at the user portal before you can log on to TACC machines.

XSEDE

Extreme Science & Engineering Discover Environment represents a collection of advanced digital services for scientific collaboration. Your account with them grants you access to the Agave API for job submission and data management, and allows you to run jobs on Stampede using "iPlant-collabs" project allocation.

- Register for an account following the steps at https://www.xsede.org/using-xsede
- To use Agave, request an allocation following the instructions at https://portal.xsede.org/allocation-request-steps

If you are a staff member at the iPlant Collaborative, send a message to support@iplantcollaborative.org to request access to the "iPlant-Collabs" allocation. Getting access may take a day or two, but you will know for sure upon trying out Stampede for yourself. **Note that you will not immediately receive a notification upon getting access. The only way to know for sure is to try it out!**

3.2.2 Accessing Stampede

To access Stampede, you may use a number of different SSH clients, depending on your operating system:

```
localhost$ ssh taccuserid@stampede.tacc.utexas.edu
```

For Windows:

SSH support is built into PowerShell for Windows 10.

For the technically curious or advanced, Cygwin is an open-source project to bring a Unix-like environment to windows. While the initial installation options may be overwhelming, all you need are the ssh tools to use the shell to access Stampede. You may also quickly get an Ubuntu image up and running through Atmosphere once you've set up your Cyverse account.

- PuTTY: a no-frills Unix-esque SSH client
- WinSCP: A more user-friendly interface similar to Windows Explorer. Allows integration with PuTTY and drag-and-drop transfer of files from local drive to remote computer.

For Mac:

- Terminal: ssh is built into Mac OS. For extensive use many prefer iTerm2.
- Cyberduck

For Unix/Linux:

- Terminal: The standard Unix terminal may be used to access Stampede through the *ssh* command.

3.2.3 Running Validate on Syngenta data set

Install Cyverse CLI tools

Accessing Validate through the Agave API hugely streamlines the task of managing your data and accessing applications through the Cyverse Data Store with a variety of command-line tools. For a comprehensive overview please see the full documentation.

Download Agave API:

```
git clone https://github.com/iPlantCollaborativeOpenSource/cyverse-sdk.git
```

Change directory into the downloaded repository and unpack the cli tools:

```
cd cyverse-sdk
tar xf cyverse-cli.tgz
```

Add the cli tools to your home directory and bash profile:

```
mv cyverse-cli $HOME
echo "PATH=\$PATH:\$HOME/cyverse-cli/bin" >> ~/.bashrc
source ~/.bashrc
```

(note that on a Macintosh system your default profile may be .bash_profile)

Initialise the CLI tools:

```
tenants-init -t iplantc.org
```

Create an OAAuth client application with API keys:

```
clients-create -S -v -N my_client -D "Client used for app development"
```

Where -S saves the keys for future use, -D provides a brief description, and -N is your application name.

You will be asked to enter your CyVerse account information:

```
auth-tokens-create -S
```

The above code is the authorization of a token. Tokens are a form of short-lived (4 hours), temporary authenticiation and authorization used in place of your username and password. To continue interacting with Agave and other Cyverse APIs after the token has expired, you will simply type in the command 'auth-tokens-create' to refresh your token.

At this point:

```
apps-list
```

should return a list of all publicly-available apps.

Configure Job Submission

A template .JSON file is available to get started. The basic parameters define a few basic settings for your job– this example, "FLMMDongWangFull," requests two hours of runtime to process the publically-available Dong Wang simulation data, which is referenced in the parameter file using an agave url. While the parameters should be fairly self-explainatory, full details can be found in the documentation for the Agave API.

download fastLMM job skeleton:

```
wget https://github.com/CyVerse-Validate/Stampede-Files/raw/master/json/fastlmm-job.
↪json
```

(you may need to install wget if using a mac)

You can open fastlmm-job.json in your text editor of choice to go over your options & make any needed changes (depending on your needs and TACC system status, you may want to request more or less time for your job):

```
{
"jobName": "FLMMDongWangFull",
"softwareName": "FaST-LMM-hpc-2.07u1",
"requestedTime": "02:00:00",
"archive": true,
"inputs":{
            "inputPED": "agave://http://datacommons.cyverse.org/browse/iplant/home/
↪shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwang.ped",
            "inputMAP": "agave://http://datacommons.cyverse.org/browse/iplant/home/
↪shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwang.map",
            "inputPHENO": "agave://http://datacommons.cyverse.org/browse/iplant/home/
↪shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwangpheno.txt"
         },
"parameters":{
    "output": "full_results"
                }
}
```

Inputs must be specified based on the app you're running– ensure the file format is correct! Refer to individual app's documentation for more information. In this tutorial we are accessing PED, MAP and phenotype files from the public syngenta_sim dataset.

To access your own data from the Data Store, change the url to replace "/shared/" with the path to your user account; ie:

```
"agave://data.iplantcollaborative.org/yourusername/results/yourdata.ped",
```

Submit Job

from the working directory where the .json file is located, type:

```
jobs-submit -F fastlmm-job.json
```

If successful you should soon see:

```
Job [job_id] successfully submitted
```

Copy down the job_id. You'll use it to check the status of your job with:

```
jobs-status *jobid*
```

Once your job is complete, you can view the outputs with:

```
jobs-output *jobid*
```

And download selected outputs with:

```
jobs-output --download --path *filename* *jobid*
```

Run more GWAS / Prediction Apps for comparision

Validate helps you find the tools best-suited for working with your data sets. Depending on your experience & the nature of your project, you may wish to process the syngenta data set with a variety of GWAS / Prediction tools, including:

- Ridge Regression
- Plink
- Gemma
- GenSel

The job submission process for any app on Stampede is fairly similar– configure a JSON wrapper containing your input / output parameters, submit the task with jobs-submit, and move the output files into your data store.

Send Output Files to Winnow

For full inputs & outputs see here

The required files for winnow are the Known Truth file and the output from a GWAS tool (FastLMM in our case)

Once you download the fastlmm output, upload it to a new location in your DE:

```
files-upload -S data.iplantcollaborative.org -F *fastlmm output
which should now be local* yourusername/yourdatafolder
```

Download the winnow example skeleton:

```
wget https://github.com/CyVerse-Validate/Stampede-Files/raw/master/json/winnow-job.
↪json
```

You can edit and submit this file using the same process described above.

Visualize Results with Demonstrate

Currently it is easier to work with R on your own system than on Stampede.

The final step of Validation will be comparing the results using a visualization method of your choice.

Refer to *Uploading data to the Data Store and viewing in the Discovery Environment* to download results from Stampede to your own disk or the data store.

Demonstrate is the final step in the Validate known-truth pipeline. Using output from Winnow, it produces a set of graphics showing differences in a GWAS/QTL applications performance under varying heritability and population structure. Demonstrate also functions without the need for heritability or population structure, but different graphics will be produced in that case.

The function you will want to use depends on what type of data you have:

Data with Heritability and Population Structure Specified

If you want to visualize differences in your data based on heritability or population structure, you'll want to use the original function Demonstrate. To run Demonstrate, type R on your terminal or command line to open the R console. From there use:

```
library(Demonstrate)
```

If nothing happens, then you did it correctly! Now the Demonstrate package is loaded. Here are the options to run the function:

```
Demonstrate(dir, make.AUC.plot=TRUE, AUC.plot.title="Mean AUC By
Population Structure and Heritability", make.MAE.plot=TRUE,
MAE.plot.title="Mean MAE By Population Structure and
Heritability",herit.strings=list("_03_","_04_","_06_")
,herit.values=list(0.3,0.4,0.6),struct.strings=list("PheHasStruct","PheNPStruct"),
↪struct.values=list(TRUE,FALSE))
```

In this function, dir represents the directory where all Winnow output is stored. These default values are based on the sample data found within this repository. Once run, the function will create two graphs on the mean absolute error (MAE) and area under the receiver operator curve (AUC) across varying levels of heritability and/or population structure. The graphs are in pdf format.

Other data from Winnow

For other types of data, or if you're more interested in comparing GWAS tools than comparing data, you will want to use the Demonstrate2 function. Before running it though, you will need to include the function in your global environment:

```
source("<path to>/Demonstrate2.R")
```

Then run the function:

```
Demonstrate2(dir, make.pos.plot=TRUE, pos.plot.title="True Positives by False␣
↪Positives", make.error.plot=TRUE, error.plot.title="Plot of AUC by MAE", extra.
↪plots=TRUE, AUC.axis.min=0, AUC.axis.max=1.0, MAE.axis.min=0, MAE.axis.max=2.0)
```

Assuming all outputs are kept, Demonstrate2 will output five files in total. First, two frequency histograms illustrating the distribution of both true and false positives (if multiple Winnow files were in the original directory, the pdf files will have multiple pages). Second, a .csv file detailing the average sensitivity, specificity, and precision of each file.

Finally, two plots based on true vs. false positives and mean absolute error vs. area under the curve will be produced. Demonstrate2 will color the points based on the file they came from, so you can compare multiple GWAS analysis results on the same plot.

3.2.4 Generating Simulation Data with Alphasim

Running Alphasim through Agave

** Estimated Time ~30min **

Prerequisites: Agave CLI, general knowledge of executing using Agave, Access to Cyverse data store, and Stampede allocation (if you want).

Helpful Documentation

- Alphasim: http://www.alphagenes.roslin.ed.ac.uk/wp-content/uploads/AlphaSimManual/AlphaSim.html
- Alphasim parameter file: https://github.com/CyVerse-Validate/Stampede-Files/blob/master/AlphaSim-1.04/AlphaSimInputInformation.txt

Locate / Open parameter file

- You can download a sample parameter files from our repository. The parameter file allows you to pass commands into the AlphaSim software.
- Once you have access the parameter file, upload it to your Data Store. The default parameter file has specifications for running multiple generations for corn genetics.

Upload with command (where USERNAME is your Cyverse username and FOLDERNAME is where you would like it to be located on the data store): files-upload -S data.iplantcollaborative.org -F AlphaSimSpec.txt USERNAME/FOLDERNAME

- Save the following as a JSON file (This can be saved locally if you have the Agave CLI installed) with username replaced with your CyVerse username and the path to the file to the parameter file within your Data Store.

```
{
  "jobName": "testAlphaSim",
  "softwareName": "AlphaSim-1.04u1",
  "requestedTime": "05:00:00",
  "archive": true,
  "inputs":{
    "specificationFile": "agave://data.iplantcollaborative.org/USERNAME/ALPHASIM_SPEC_
↪PATH"
          }
    }
```

- With the following command you can run AlphaSim (publicly available on Agave). This is assuming that you don't want to change any of the parameters:: jobs-submit -F [path to json file]
- After submitting your job there will be a returned message which will list your job-id - This is helpful for looking up the job status and downloading output later.

- The command :: job-status [yourjobid]

will allow you to see when your job will be done

- Once this finishes it will save onto your Data Store, if Archive is set to true, under Archive/Jobs in your personal DE folder.
- You can download your outputs from jobs with the general command

```
jobs-output --download --path [path to the desired file relative to the job␣
↪output folder on the data store] [JOBID]
```

- You will download your outputs to your local computer from the Data Store with the following commands

```
jobs-output --download --path AlphaSim/Selection/SelectionFolder1/SnpSolutions.
↪txt [yourjobid]
jobs-output --download --path AlphaSim/SimulatedData/Gender.txt [yourjobid]
jobs-output --download --path AlphaSim/SimulatedData/PedigreeTbvTdvTpv.txt␣
↪[yourjobid]
jobs-output --download --path AlphaSim/SimulatedData/AllIndividualSnpChips/
↪Chip1Genotype.txt [yourjobid]
jobs-output --download --path AlphaSim/Chromosomes/Chromosome1/
↪Chip1SnpInformation.txt [yourjobid]
```

- Alternatively, feel free to modify this bash script to your needs to automate pulling in all of the output files: https://github.com/CyVerse-Validate/Stampede-Files/blob/master/AlphaSim-1.04/getFilesExample.sh
- If you are okay with these outputs they are now useable. You also have the option to convert to pedmap (which is a more standard format) using our merger application.

Convert to ped/map

Download the merger.py from the github repository here (or if you have already installed Validate, it is included): https://github.com/CyVerse-Validate/Validate/tree/master/CurrentReleaseStable/Util_1/Merger

Run the merger with the AlphaSim outputs with the following commands from the directory where your AlphaSim output is loc

:: python PATHTOMERGE/merge.py –output ALPHASIMPREFIX alphasim –snp SimulatedData/Chip1SnpInformation.txt –pedigree SimulatedData/PedigreeTbvTdvTpv.txt –gender SimulatedData/Gender.txt –geno SimulatedData/AllIndividualsSnpChips/Chip1Genotype.txt –sol Selection/SelectionFolder1/SnpSolutions.txt

This will yield ALPHASIMPREFIX.ped and ALPHASIMPREFIX.map in the directory specified in your above output flag.

To convert your ped/map files to bed/bim/fam format, required by many applications such as fastlmm, follow these steps:

Convert to bed/bim/fam

Download plink from the github repository here (or if you have already installed Validate, it is included): https://github.com/CyVerse-Validate/Validate/blob/master/CurrentReleaseStable/GWAS_1/plink

Run Plink with the following command from the directory where your ped/map files are located: :: PATHTOPLINK/plink –file ALPHASIMPREFIX –out ALPHASIMPREFIX –make-bed

- At this point, you should have all the original output available to you as well as ped/map files and bed/bim/fam files and can use the appropriate files for your job.

3.2.5 Submitting jobs on Stampede

You have two alternatives for running your tools on Stampede. The Agave API allows you to *Configure Job Submission* as outlined in our quickstart guide.

For further reference see:

- http://agaveapi.co/what-is-a-job2/
- http://agaveapi.co/documentation/tutorials/job-management-tutorial/

You can also use Stampede's parametric launcher to distribute your work across multiple nodes. For large jobs, this has tangible speed benefits & is in general a more secure & efficient computing practice. The TACC guide for the parametric launcher can be found at the link below, or scroll down for our quick-start tutorial.

- https://www.tacc.utexas.edu/research-development/tacc-software/the-launcher

Parametric Launcher

Running multiple jobs through the launcher on Stampede

Prerequisities: Agave CLI, general knowledge of executing using Agave, Access to the Cyverse data store, and Stampede allocation

Job Format:

When you start a job using the parametric launcher, it will always look for a parameter file in your current working directory titled "paramlist". The first step is to put together this file and store it where you want to run your job from. Below is an example of a paramlist for the prediction app bayesR. You can see that it is simply a list of commands to run the bayesR application with the specified inputs and outputs.

```
/home1/04109/ksierrac/applications/bayesR-master/bin/bayesR -bfile /work/04109/
↪ksierrac/bayes_speedtest/data/AlphaSim_1 -out /work/04109/ksierrac/bayes_speedtest/
↪alpha_output/AlphaSim_1_out;
/home1/04109/ksierrac/applications/bayesR-master/bin/bayesR -bfile /work/04109/
↪ksierrac/bayes_speedtest/data/AlphaSim_2 -out /work/04109/ksierrac/bayes_speedtest/
↪alpha_output/AlphaSim_2_out;
/home1/04109/ksierrac/applications/bayesR-master/bin/bayesR -bfile /work/04109/
↪ksierrac/bayes_speedtest/data/AlphaSim_3 -out /work/04109/ksierrac/bayes_speedtest/
↪alpha_output/AlphaSim_3_out;
/home1/04109/ksierrac/applications/bayesR-master/bin/bayesR -bfile /work/04109/
↪ksierrac/bayes_speedtest/data/AlphaSim_4 -out /work/04109/ksierrac/bayes_speedtest/
↪alpha_output/AlphaSim_4_out;
```

The advantage of using the launcher in this situation is that all you have to do is to build the paramlist file. Once you submit, the launcher will automatically submit these multiple jobs for you.

To run these jobs through parametric launcher

- **To run the above example, you would use this command:** :: sbatch -N 64 -n 128 -t 48:00:00 $TACC_LAUNCHER_DIR/launcher.slurm
- The submit command is "sbatch", the same as submitting a single job to Stampede. Call the launcher by adding "$TACC_LAUNCHER_DIR/launcher.slurm" at the end of your command.

(Note: If you have not loaded the launcher module, first run the command "module load launcher" followed by "module save")

- -N specifies number of nodes, -n specifies number of processes running at once, and -t specifies the allowed time for the jobs to run. For a full list of possible flags, see this link: https://portal.tacc.utexas.edu/user-guides/stampede#tablesbatchoptions

3.2.6 Uploading data to the Data Store and viewing in the Discovery Environment

For a comprehensive guide please see here

For this tutorial we will use the publicly-available BayesR example data, which we can upload using the "importing from URL" method.

1. In the Discovery Environment, click the Data button on the left-hand side. Here we'll create a new folder to import the data to.
2. With the Data window open, click on your user name and then file -> new folder. Any name works– here we'll use "BayesRData"
3. With "BayesRData" open, click upload -> import from URL.
4. In another tab, browse to https://github.com/syntheke/bayesR/tree/master/example/. BayesR requires .BED, .BIM, and . FAM files to run, so you'll need to upload simdata.bed, simdata.fam, and simdata.bim to your DE folder.
5. To upload the data, open the file on github and then right-click on "Raw" to copy the link address. You can then paste this into the "upload from URL option" in the Discovery Environment.

3.2.7 GWAS Tools

FastLMM

The main genome wide association studies tool that we have used, FaST-LMM stands for Factored Spectrally Transformed Linear Mixed Models. It is a tool from Microsoft Research designed for analyses of very large data sets, and has been tested on data sets with over 120,000 individuals.

Running the Program

The easiest way to run fastlmm is through Agave, as it will interpret your json job file and compile the command for you. However, if you are interested in running fastlmm directly through SLURM, we provide documentation for that as well.

Run through Agave

To run fastlmm, you will need a phenotype file as well as a set of EITHER ped/map or bed/bim/fam data files. The example json below includes ped/map files from the syngenta data, and can be run as-is if you would like to test it before using your own data.

Here are all of the possible inputs that may be used in your job submission:

- inputPHENO : (required) phenotype file corresponding to PLINK fileset
- inputCOVAR : (optional) covariate file

Choose one group of data inputs:

Transposed PLINK:

- inputTPED : TPED file for transposed PLINK set
- inputTFAM : TFAM file for transposed PLINK set

Main PLINK:

- inputPED : PED file for main PLINK set
- inputMAP : MAP file for main PLINK set

Binary PLINK:

- inputBED : BED file for PLINK binary set
- inputBIM : BIM file for PLINK binary set
- inputFAM : FAM file for PLINK binary set

Parameters:

- output : (required) The name of the output file (without extension)
- verboseOutput : (optional) Trigger for verbose mode, 1 if true, 0 if false

Optional advanced parameters:

- C : Trigger for whether covariate file is included (1 if true, 0 if false)
- B : Set the binary PLINK fileset as the primary input (1 if true, 0 if false)
- SimFileset : Specify the PLINK grouop used to calculate the genetic similarity matrix (PEDMAP, BEDBIMFAM, or TPEDTFAM)
- mpheno : Proper column number for phenotype file (if multiple phenotypes are used)
- T : Transposed PLINK fileset is the primary input (1 if true, 0 if false)

You can save this example json and modify it for your needs:

```
{
      "jobName": "FLMMDongWangFull",
      "softwareName": "FaST-LMM-hpc-2.07u1",
      "requestedTime": "02:00:00",
      "archive": true,
              "inputs":{
                      "inputPED": "agave://http://datacommons.cyverse.org/browse/
↪iplant/home/shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwang.ped",
                      "inputMAP": "agave://http://datacommons.cyverse.org/browse/
↪iplant/home/shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwang.map",
                      "inputPHENO": "agave://http://datacommons.cyverse.org/browse/
↪iplant/home/shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwangpheno.txt"
              },
              "parameters":{
                      "output": "full_results"
              }
}
```

Run directly through SLURM

The program requires a minimum of three files to function: 1) A PEDMAP set of files, 2) a phenotype file corresponding to the PEDMAP set, and 3) a set of PLink formatted files to compute the genetic similarity matrix decomposition (does not need to be different from number 1).

The input flags you would use are as follows:

- -file : Denotes the file name for the PLINK .ped/.map files
- -bfile : Denotes the name for PLINK .bed/.bim/.fam files
- -tfile : Denotes the name for PLINK .tped/.tfam files
- -pheno : Denotes the name of the phenotype file (including extension)

- -covar : Denotes the name of the covariate file (including file extension)
- -out : The name of your final output file, which is placed into the same directory as your program and data unless otherwise specified

These are the bare minimum options needed to run FaST-LMM; however, some other options for considerations are:

- -verboseOutput : use this flag to show more complex and detailed output; does not require a file to be named
- -fileSim : The name of the PLINK set used for computing the genetic similarity matrix and its decomposition
- -pValuePrintThreshold : Restricts the output file to only include SNPs with a p-value less than or equal to the specified threshold

An example of executing the FaST-LMM program might look like:

```
fastlmmc -verboseOutput -bfile toydata -pheno toydata.phe.txt -covar toydata.covar.
↪txt -out MyResults.csv
```

Puma

Puma (Penalized Unified Multiple-locus Association) “comprises a family of statistical methods designed to identify weak associations in genome-wide association studies that are not detectable by conventional analytical methods.” Puma was developed by Jason Mezey at Cornell University, and we have installed it on Agave so that it is available for use.

Puma download (if you would like to install the software yourself rather than run through Agave) is available here: http://mezeylab.cb.bscb.cornell.edu/Software.aspx

Example data can be found here and uploaded to your data store for testing use with Agave: https://github.com/CyVerse-Validate/Stampede-Files/tree/master/Puma/data

Here is an example JSON job file which you can save and modify for your own use:

```
{
  "jobName": "puma-test-1",
  "softwareName": "Puma-1.0u1",
  "processorsPerNode": 16,
  "requestedTime": "01:00:00",
  "memoryPerNode": 32,
  "nodeCount": 1,
  "batchQueue": "serial",
  "archive": true,
  "archivePath": "",
  "inputs": {
      "tped": "agave://data.iplantcollaborative.org/PATHTODATA/DATA.tped",
      "tfam": "agave://data.iplantcollaborative.org/PATHTODATA/DATA.tfam"
  },
  "parameters":{
      "regression": "LINEAR",
      "penalty":"LASSO",
      "name":"try1"
  }
}
```

These are all of the possible inputs you can specify for your job:

Inputs:

- tped (required) [genotype data in plink TPED format]

- tfam (required) [phenotype (and sex) data in plink TFAM format]
- sex [if tfam is used, this includes sex as a covariate]
- covariates [file storing matrix with each column being a covariate]
- regression (required) [specify regression model as either LINEAR or LOGISTIC]
- sma [if set, performs only standard single marker analysis]
- **penalty (required) [space delimited list of methods to run, select from:** LASSO ALASSO LOG NEG MCP VBAY]
- name (required) [name to be appended to results files]

Advanced inputs:

- **screen_p_value [marginal p-values below which markers are passed to method]** (default = 0.01)
- **pML_restarts [number of posterior modes explored]** (default = 100)
- results [specify folder where results are saved. Defaults to local folder]
- **nthreads [number of threads used to run in parallel]** (default = machine default)
- restrictedPathSearch [1 dimensional path search for non-convex penalties]

When you run the job, it will return a file of pvalues as well as an R results file. The best way to read this data is to use an R extraction program which will summarize the results for you:

- Download extract_puma_results.R and place a copy in the directory with your results file: https://github.com/CyVerse-Validate/Stampede-Files/blob/master/Puma/extract_puma_results.R)
- Modify the file with your results file name (example: "results_testjob_LASSO.R") in line 9: :: result = dget("FILENAME.R")
- Save extract_puma_results.R
- Start R and run these lines (where OUTPUTFILENAME is what you want your summarized results file to be called):

```
sink('OUTPUTFILENAME.txt')
print(source('extract_puma_results.R'))
sink()
```

This will place the summarized results file in your working directory.

3.2.8 Prediction Tools

BayesR

Speed Test Results

Our BayesR speed test on a third of the syngenta data set on the Cyverse Data store completed successfully in 22 hours and 25 minutes, running 333 jobs on 64 nodes, two processes per node.

Running BayesR through Agave

From your local machine, create a folder for the output and cd into it:

```
mkdir outputfolder
cd outputfolder
```

Download the BayesR job skeleton:

```
wget https://github.com/CyVerse-Validate/Stampede-Files/raw/master/json/bayesR-job.
↪json
```

Open this with a text editor and edit the following parameters:

1. For jobName, anything will work. For this example, I used testBayesR
2. For software name, enter "bayesR-2.00u1"
3. For requested time, enter "02:00:00"

d. For inputBED enter "agave://data.iplantcollaborative.org/username/folder/simdata.bed", replacing username and folder with your own. For instance, mine would be: "agave://data.iplantcollaborative.org/swb5075/BayesRData/simdata.bed" and repeat this for the other two input files (BIM and FAM files)

e. For output, anything will work. I will use "BayesRTesting". Save and close this file.

Note: You can copy and paste this into a website like http://jsonlint.com/ to ensure that the JSON file is formatted correctly.

Using your local terminal, make sure you are in the same directory as the job.json file.

Enter:

```
jobs-submit -F bayesR-job.json.
```

You should get a response: "Successfully submitted job . The sample data takes about twenty minutes to run but you can check this by entering: jobs-status with the same job ID as above. Once the job is complete, the response to this command will be "FINISHED"

We now want to download the output data. You can enter jobs-output-list to see all of the included files. For instance I may enter:

```
jobs-output-list 3895995830152073701-ee4acae9fffff7a7-0001-007
```

BayesR has six output files (.frq, .gv, .hyp, .log, .model, .param files) which begin with the output parameter from the job json file. In my case, testresults.

In order to download, for example, the frequency file enter:

```
jobs-output --download -- path testresults.frq 3895995830152073701-ee4acae9fffff7a7-
↪0001-007
```

Here the –path parameter refers to which file you want to download and the long string at the end is your job ID.

GenSel

Running GenSel through Agave

Estimated Time ~30min

Prerequisities: Agave CLI, general knowledge of executing using Agave, Access to the Cyverse data store, and Stampede allocation (if you want)

Helpful Documentation

- Gensel (User Manual) http://www.biomedcentral.com/content/supplementary/1471-2105-12-186-s1.pdf
- Gensel (Cyverse page) https://pods.iplantcollaborative.org/wiki/display/DEapps/GenSel

Locating the parameter/input files and running Gensel:

To run the software you need one parameter file (.inp) and three input files (.gs, .newbin, .192).

- All of these files can be found in the Data Store following the path iplant/home/shared/iplantcollaborative/example_data/gensel/
- For more information on these inputs read the documentation listed above

1. Save the following as a JSON file (This can be save locally as long as you have the Agave CLI installed)

```
{
  "jobName": "testGenSel",
  "softwareName": "GenSel-2.14",
  "nodeCount": 1,
  "batchQueue": "serial",
  "requestedTime": "02:00:00",
  "processorsPerNode": 16,
  "archive": false,
  "archivePath": "",
  "inputs":{
  "phenotypeFileName":"agave://http://datacommons.cyverse.org/browse/iplant/home/
↪shared/iplantcollaborative/example_data/gensel/DMI.gs",
  "markerFileName":"agave://http://datacommons.cyverse.org/browse/iplant/home/shared/
↪iplantcollaborative/example_data/gensel/gpegeno.newbin",
  "includeFileName":"agave://http://datacommons.cyverse.org/browse/iplant/home/shared/
↪iplantcollaborative/example_data/gensel/DMIg.192"
  },
  "parameters":{
  "Parameter_File":"agave://http://datacommons.cyverse.org/browse/iplant/home/shared/
↪iplantcollaborative/example_data/gensel/run.inp"
  }
}
```

2. With the following command you can run Gensel (publicly available on Agave) This is assuming that you don't want to change any of the parameters/inputs listed in the sample json file.

```
jobs-submit -F (your path the JSON file you just saved)
```

RidgePredict

Running RidgePredict through Agave

Estimated Time ~15min

Prerequisities: Agave CLI, general knowledge of executing using Agave, Access to the Cyverse data store, and Stampede allocation (if you want)

Helpful Documentation

- The RidgePredict app uses Ridge Regression based on the SciKitLearn Ridge package: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Locating the parameter/input files and running RidgePredict:

To run the software you need one .ped parameter file, for example: agave://http://datacommons.cyverse.org/browse/iplant/home/shared/syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwang.ped You only need one parameter, which will be the name you wish to have for your output.

1. Save the following as a JSON file and modify to your needs:

```
{
  "jobName": "ridge-test-1",
  "softwareName": "RidgePredict-1.1",
  "processorsPerNode": 16,
  "requestedTime": "01:00:00",
  "memoryPerNode": 32,
  "nodeCount": 1,
  "batchQueue": "serial",
  "archive": false,
  "archivePath": "",
  "inputs": {
      "inputPed": "agave://http://datacommons.cyverse.org/browse/iplant/home/shared/
↪syngenta_sim/Dong_Wang_sim/Analysis_Files/dongwang.ped"
  },
  "parameters":{
      "outputPed": "ridge-test-output.ped"
  }
}
```

3.3 Links & Citations

3.3.1 Resources

-
-
-

3.3.2 Citations

-
-
-

3.3.3 User Manuals & Project Websites

-
-
-

3.4 Getting Help

- Since Validate analyzes the outputs of other developers' association and prediction tools, you may need to refer to the documentation for the individual apps you are using to ensure that you're using the proper input & output files
- Post on the github issues page for the Validate project for a personal response.
- Email stapletonlab@gmail.com

3.5 Running the pipeline

The Validate Pipeline represents our latest efforts to to simplify high-speed GWAS computing in the XSEDE cloud. While advanced users may wish to refer to the workflow overview for running custom applications, the pipeline includes scripts and wrappers for several common tools you can get started with right away.

A typical remote session on Stampede consists of several steps: -Set up Agave CLI tools on Stampede -create JSON files for job submission -submit jobs -retrieve results -repeat to analyze outputs in other GWAS tools, validation, or visualization

The Validate Pipeline handles job submission and data management for you, automatically submitting jobs, monitoring their progress, and moving the results into your datastore. Automation of these steps makes it feasible to run multiple iterations of your analysis through multi-step workflows with considerably less user input. Researchers will enjoy fewer hours spent watching job queues, and users with extensive data requirements may be able to run customized workflows that would be unfeasible for a single user to submit and manage "by hand."

What you'll need: -Accounts w/ Cyverse & TACC -SSH client application -Stampede allocation -Data to be tested uploaded to the Data Store -Validate Pipeline directory available on github

3.5.1 Setup Stampede2 and run PipelineAgavePy

This tutorial assumes that you have set up accounts with TACC and Cyverse and be assigned to a project allocation.

Install Validate 1.2

(initial step: SSH into Stampede2 with your client of choice)

- Your Stampede2 home directory supports up to 10GB of data, while your work directory allows for up to 1TB. Use the alias command "cdw" and "cdh" to switch been your work and home directories, respctively:

  ```
  >$ cdw
  ```

- Create a virtual environment to install the agavepy package with pip. See virtualenv for more information. **is this necessary or just good practice?**

  ```
  >$ virtualenv Validate1.2
  ```

- Change to new environment directory.

  ```
  >$ cd Validate1.2
  ```

- We need to use our new environment.

```
>$ source bin/activate
```

- Next we need to install the packages required to run the Pipeline.

```
>$ pip install agavepy numpy pandas statsmodels scipy
```

- Finally we need to clone the repo from Github that contains the Pipeline.

```
>$ git clone https://github.com/CyVerse-Validate/Validate.git
```

You now should be setup to run the pipeline.

- Move to PipelineDevel directory. Use the standard tools cd, pwd, and ls to navigate on Stampede from the command line. If you've just completed the previous steps, simply use:

```
>$ cd PipelineDevel
```

- the command to initialize the pipeline requires several parameters. In addition to your Agave account infomation, you must specify if you want to run a simulation, what type of GWAS or GenSel tools to use, whether you need Winnow for further analysis, the type of data being used (PED, BED, TFAM), and where the data is located on the Discovery Environment.
- A sample command line looks like this:

```
- >$ python PipelineAgavePy.py -u *username* --password *password* -i p -f /
↪username/My_Data -lmm True
```

  ##Password option is not required. If leftoff you will be prompted for password. t is not required. True is not required after app option.

This would instruct the script to run the TPED/TFAM files in the specified datastore directory through FaSt-LMM.

Full Meta-Parameters

Table 1: Full Meta-Parameters

Command	Verbose, "Description"	
-u	–username *username*	Username for Agave services.
	–password	Password used for Agave services.
-i	–InFormat *inputformat*	Defines data input format. p for PED/MAP, b for BIM/BED/FAM, t for TPED/TFAM.
-f	–Folder *working_directory*	Folder to be pipelined. This folder should contain all inputs as well as the known-truth file for the given dataset.
-pno	–pheno *filename*	Name (including extension) for the covariate file for the given dataset.
-lmm	–fastlmm	Run Fast-LMM
-rdg	–ridge	Run Ridge.
-bay	–bayes	Run BayesR.
-plk	–plink	Run Plink.
-qxp	–qxpak	Run QxPak.
-gma	–gemma	Run Gemma.
-pma	–puma	Run Puma.

Modifying JSON Parameters

While the default settings should be adequate for most tasks, you may need to edit the python script with custom JSON settings if your jobs require additional processing or are timing out. This is handled in Validate/JsonBuilder.py. By editing this python file, you can easily alter the requested time for each app in the workflow. Simply change the value for "requestedTime" to a value that meets your needs.

Data Management

3.6 Documentation Notes & Discussion

3.6.1 Some suggestions for the documentation:

- Some overview of the workflow is needed. E.g. where the jobs are run, where the output is put, and then why you need to download it, and then upload it again to run winnow.

With the pipeline this will no longer be a problem. Users will give their Cyverse login info at the beginning and the pipeline will automatically access the users Discovery Environment. Jobs are run on Stampede2 using Agave. The output from the job is stored on the users DE. There will be an option to run Winnow in the initial command. Winnow output will be created and placed in a different directory in the user's DE.

- Under "Configure Job Submission" it would be useful to have some general background, for example what the job is going to do. What is FastLMM and what kinds of inputs does it need? It isn't clear from the json how to direct the results from a job to a particular place or folder. Some details on where the output is going would be informative. In the past I've only written submission scripts with command line arguments - could you add an example of a command line argument and how it would look in the json?

*First part reference the analyses descriptions in the tutorial. Second reference answer to question preceding. Third part reference wrappers used for each job under cyverse-validate/stampede-files.

- Under "Submit Job" is there a way to access your current or previous job ids?

All of the jobs that are submitted are tied to the user's Agave login (i.e. Cyverse login) information. The default JSON's from Validation 1.0 were using the directory "archive" on the DE to store all the job and it was listed by Job ID. Michael is working on each job going into a directory based on the type of analysis submitted (i.e. all simulations in one directory, all GWAS in another directory, GenSel another, Winnow another, etc.) and then each of those jobs will have a dated name. So if a user needed to go looked at a job they could access the information through the DE. An alternative would be to use the Agave ToGo website as it also

- "Run more GWAS / Prediction Apps for comparision" (note "comparision" is spelled wrong). It would be good to have some json examples that work for each of these programs. It would also be good to have links to the publications and webpages to each of the programs.

*Find a good way to reference parameters available from published apps. Going forward there will be a json builder as part of the pipeline so examples of all the json's will not be necessary.

- "Send Output Files to Winnow" Some background is needed on Winnow and what it does. The link is also not very helpful. What format should the "known truth" file be in? What are the fit statistics that it calculates and how are they calculated? What does Results.txt look like? Some people, like me, probably won't use Winnow because we have some other stats and analysis that we'd like to run. It might be worth stating that this is optional.
- "Demonstrate" section - It seems like this package requires a very specific type of input directory. It would be good to have some more information/examples on the inputs, and some examples of the outputs. How are multiple programs visualized and compared?

[8:55] (Note that answers to some of my questions are found in other places in the docs, but links and more info just need to be added to the quickstart guide)