SimPrily Documentation

Release 1.0.1

Ariella Gladstein

Jun 25, 2018

Contents:

CHAPTER 1

Introduction

SimPrily runs genome simulations with user defined parameters or parameters randomly generated by priors and computes genomic statistics on the simulation output.

1. Run genome simulation with model defined by prior distributions of parameters and demographic model structure.
2. Take into account SNP array ascertainment bias by creating pseudo array based on priors of number of samples of discovery populations and allele frequency cut-off.
3. Calculate genomic summary statistics on simulated genomes and pseudo arrays.

This is ideal for use with Approximate Bayesian Computation on whole genome or SNP array data.

Uses c++ programs macs and GERMLINE. For more information on these programs, see: https://github.com/gchen98/macs https://github.com/sgusev/GERMLINE

CHAPTER 2

Install and Environment Set up

- Python 2.7.6, 2.7.11, or 2.7.13 is required to run the code, with the requirements installed from requirements.txt. *Environments for Python 3 will soon be available*.
- We highly recommend running SimPrily with the provided Docker, Singularity, or virtual environment.

2.1 Container

2.1.1 Docker

A Docker Image built with Python 2.7.13, the requirements, and the SimPrily code can be found on Docker Hub https://hub.docker.com/r/agladstein/simprily/

cd to the directory you want to work in and then pull the Docker image. To pull the Docker container:

```
docker pull agladstein/simprily
```

2.1.2 Singularity

The Docker image can be pulled as a Singularity container.

To pull the Singularity container:

```
singularity pull docker://agladstein/simprily
```

Open Science Grid Connect

A prebuilt Singularity Image from the Docker Image is used for the Open Science Grid workflow. The Singularity Image on OSG Connect is available from `/cvmfs/singularity.opensciencegrid.org/agladstein/simprily\:latest`.

2.2 Virtual environment

2.2.1 Linux OS

cd to the directory you want to work in and then download the repository,

```
git clone https://github.com/agladstein/SimPrily.git
```

Install the virtual environment and install the requirements.

```
./setup/setup_env_2.7.sh
```

If you get an error during `pip-sync` try rebooting the system.

2.2.2 Virtual Machine for non-Linux

If you are running on a non-Linux OS, we recommend using the virtual machine, Vagrant (can be used on Mac or PC). In order to run Vagrant, you will also need VirtualBox.

Download Vagrant from https://www.vagrantup.com/downloads.html

Download VirtualBox from https://www.virtualbox.org/

cd to the directory you want to work in and then download the repository,

```
git clone https://github.com/agladstein/SimPrily.git
```

Start Vagrant, ssh into Vagrant, cd to SimPrily directory.

```
vagrant up
vagrant ssh
cd /vagrant
```

Install the virtual environment and install the requirements.

```
./setup/setup_env_vbox_2.7.sh
```

2.3 Local installation

We do not recommend this method

cd to the directory you want to work in and then download the repository,

```
git clone https://github.com/agladstein/SimPrily.git
```

If the above options do not work, the correct version of Python can also be installed locally:

```
cd mkdir python_prebuild
wget https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
mkdir python
tar -zxvf Python-2.7.6.tgz
cd Python-2.7.6
./configure --prefix=$(pwd)/../python
make
```

(continues on next page)

(continued from previous page)

```
make install
cd ..
export PATH=$(pwd)/python/bin:$PATH
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
pip install -r requirements.txt
python simprily.py --help
```

CHAPTER 3

Usage

`simprily.py` takes 4 required arguments and 2 optional arguments, and help, verbose, and profile options.

```
python simprily.py [-h] -p PARAM -m MODEL -i ID -o OUT [-g MAP] [-a ARRAY] [-v] [--
↪profile]
```

For quick help:

```
python simprily.py --help
```

e.g. One simulation (with pseudo array and genetic map):

```
python simprily.py -p examples/eg1/param_file_eg1_asc.txt -m examples/eg1/model_file_
↪eg1_asc.csv -g genetic_map_b37/genetic_map_GRCh37_chr1.txt.macshs -a array_template/
↪ill_650_test.bed -i 1 -o output_dir -v
```

e.g. One simulation (genetic map, no pseudo array):

```
python simprily.py -p examples/eg1/param_file_eg1.txt -m examples/eg1/model_file_eg1.
↪csv -g genetic_map_b37/genetic_map_GRCh37_chr1.txt.macshs -i 1 -o output_dir -v
```

3.1 How to run with a Container

3.1.1 Docker

```
docker run -t -i --mount type=bind,src="$(pwd)",dst=/app agladstein/simprily python /
↪app/simprily.py [-h] -p PARAM -m MODEL -i ID -o OUT [-g MAP] [-a ARRAY] [-v] [--
↪profile]
```

3.1.2 Singularity

```
singularity exec simprily.simg python /app/simprily.py [-h] -p PARAM -m MODEL -i ID -
↪o OUT [-g MAP] [-a ARRAY] [-v] [--profile]
```

3.2 Required Input

-p PARAM	The location of the parameter file
-m MODEL	The location of the model file
-i ID	The unique identifier of the job
-o OUT	The location of the output directory

or

--param PARAM	The location of the parameter file
--model MODEL	The location of the model file
--id ID	The unique identifier of the job
--out OUT	The location of the output directory

3.3 Optional Input

-h	Shows a help message and exists
-v	Increase output verbosity. This includes 3 levels, `-v`, `-vv`, and `-vvv`
--profile	Print a log file containing the time in seconds and memory use in Mb for main functions
-g MAP	The location of the genetic map file
-a ARRAY	The location of the array template file, in bed format. The third column is used as the physical positions of the SNP for the pseudo array.

or

--help	Shows a help message and exists
-v	Increase output verbosity. This includes 3 levels, `-v`, `-vv`, and `-vvv`
--profile	Print a log file containing the time in seconds and memory use in Mb for main functions
--map MAP	The location of the genetic map file
--array ARRAY	The location of the array template file, in bed format. The third column is used as the physical positions of the SNP for the pseudo array.

3.4 Additional information on input arguments

3.4.1 ID

This is a unique identifier for the job. It is used in the names of the output files. For example, the output file with parameter values and summary statistics is named `results_{IDid}.txt`.

3.4.2 output_dir

This is where all the output goes. Within the output_dir the directory `results` will always be created. The `results` directory contains the results file `results_{jobid}.txt` with the parameter values and summary statistics. Additionally, the directories `germline_out` and `sim_data` are also created, but will be empty if the `germline` or `pedmap` arguments in the model file are not included.

Be careful when running large numbers of jobs (>2000). It is bad practice to run large numbers of jobs and direct all the output to the same directory, because listing the contents of the directory becomes very slow. Instead, we recommend creating directory "buckets". See section Recommendations for other HTC workflows.

3.4.3 param_file.txt

Examples of param_file.txt can be found in examples. The param_file.txt must define the parameters of the demographic model and the minimum derived allele frequency to be used to create the pseudo array, if a pseudo array is to be created.

All time parameters must end in `_t`.

All parameter values should be given in pre-coalescent scaled units. That is, Ne should be given in units of chromosomes, and time should be given in units of generations. The code will scale to the appropriate coalescent units for the simulation.

The definition can be hard-coded parameter values, such as:

```
A = 1000
B = 1000
T1_t = 100
```

The definition can be a prior, such as:

```
A = (1e3.0:1e4.0)
B = (1e3.0:1e4.0)
T1_t = (10:500)
```

Log base 10 can be used for the parameter definitions by using `1eX` or `1Ex`. This is recommended when using a prior with a very large range (See ABCtoolbox manual).

If pseudo arrays are to be created, the derived allele frequency must be defined. For example,

```
A = (1e3.0:1e4.0)
B = (1e3.0:1e4.0)
T1_t = (10:500)
daf = (0.01:0.1)
```

currently only a range of values is supported for daf. Therefore if you want to hard code a value, use the same value as the min and max of the prior.

3.4.4 model_file.csv

Examples of model_file.csv can be found in examples.

The demographic model, SNP ascertainment model, and additional options are defined in the model_file.csv. The demographic model defines events in populations' history, including population divergence, instantanious effective population size changes, exponential growth, gene flow and admixture. We use a coalescent simulation, so models must be defined backwards in time, starting from the present, with each event going back in the past. The SNP ascertainment model defines how to create a pseudo SNP array using a template SNP array, a set of discovery populations and a minor allele frequency cutoff. The SNP ascertainment model should be used when comparing to real SNP array data.

All instances of any argument must start with a dash followed by the corresponding argument parameters, and value(s). Each new argument must be a new line. All variables and values must be separated by commas (white space will be ignored, so it is okay to include spaces). The model arguments can appear in any order.

All parameters must be called with a name corresponding to its definition in the param file. This is how parameter values are assigned to the simulation model. For example,

```
-macs,./bin/macs,
-length,5000000,
-s,1231414,
-t,2.5e-8,
-r,1e-8,
-h,1e5,
# define a sample size of 50 haploid individuals for populations 1 and 2
-I, 2, 50, 50
# define the effective population size at present for population 1
-n, 1, A
# define the effective population size at present for population 2
-n, 2, B
# define a divergence event (join backwards in time) between populations 1 and 2
-ej, T1, 1, 2
```

Setup simulation arguments

One of the following two flags must be included:

`-macs` use the original simulator MaCS. This option will stream the MaCS simulation output directly to be read into a python bitarray.

`-macs_file` read in static output from MaCS. This should only be used for rigorous testing.

Following the `-macs` and `-macs_file` flags there should be a path to either the executable or static file in relation to the working directory. For example:

If you are using a virtual environment the path to macs should be

```
-macs, ./bin/macs
```

If you are using Docker or Singularity the path to macs should be

```
-macs, /app/macs
```

or if you want to use a static file,

```
-macs_file, tests/test_data/sites1000000.txt
```

-length The number base pairs you want to simulate. Must be included.

-s random seed. Must be an integer. If no input is given, no seed will be used, and everything will be random. If a seed is provided, reproducible parameters will be picked from the priors. Using a seed will also cause reproducible simulations with macs.

Demographic simulation arguments

All argument flags are based on macs arguments (see macs and ms manual for more detail).

`-t`: mutation rate per site per 4N generations

`-d`: enable debugging messages. No entry will default to allowing debugging messages. This will not work when using macsswig

`-h`: history. Refers to the number of previous base pairs to retain

`-r [r]`: recombination rate per site per 4N generations

`-c [f lambda]`: f = ratio of gene conversion rate to crossover rate. track len(lambda) is mean length of tract in base pairs. *This has not been tested.*

`-T`: Print each local tree in Newick format to standard out. *This has not been tested.*

`-G [alpha]`: Assign growth rate alpha across populations where alpha=-log(Np/Nr).

`-I [n n_n]`: Assign all elements of the migration matrix for n populations. Values in matrix set to mig_rate/(n-1). The length of n_n should be equal to n

`-m [i,j m]`: i, j is associated with a location in the migration matrix m is assigned to the value at (i, j)

`-ma [m_nn]`: Assign values to all elements of migration matrix for n populations

`-n [i size]`: Population i set to size

`-g [i alpha]`: assigns alpha value as explained in -G to population i

`-eG [t alpha]`: t is a time value. alpha behaves the same as in -G

`-eg [t i alpha]`: t is a time value. alpha behaves the same as in -G. i is a population that alpha is assigned to at time t.

`-eM [t m]`: t is a time value. Assign migration rate m to all elements in migration matrix at time t

`-em [t i,j m_ij]`: t is a time value. i and j make up point in a population matrix. assigns migration rate m_ij to the population at i, j at time t

`-ema [t n m_nn]`: t is a time value. Assign migration rates within the migration matrix for n populations at time t.

`-eN [t size]`: t is a time value. Assigns size to all populations at time t

`-en [t i size_i]`: t is a time value. assigns size_i to population i at time t

`-es [t i p]`: t is a time value. splits population i by p at time t

`-ej [t i j]` t is a time value. joins population i with population j at time t

SNP array ascertainment arguments

If the user would like to create a pseudo array from the simulation, the array template must be included in the command line argument with the flag `-a`, and four additional arguments must be included in the model_file:

`-discovery`, followed by the populations (defined by their numbers from `-n`) that should be used to discover the SNP (e.g. the HapMap populations). These are the populations that will be used to create the pseudo array. When calculating summary statistics, summary statistics based on whole genome simulation and pseudo array will be calculated for these populations.

`-sample`, followed by the populations (defined by their numbers from `-n`) that are the samples of interest for demographic interest.

`-daf`, followed by the parameter name for daf.

`-random_discovery`, followed by `True` or `False`. True will add a random number of individuals to the discovery populations to use as the "panel" to create the pseudo array. When this option is False, the total number of simulated discovery populations is equal to the number "genotyped" and in the "panel".

For example:

```
-macs,./bin/macs,
-length,5000000,
-s,1231414,
-t,2.5e-8,
-r,1e-8,
-h,1e5,
-I, 2, 50, 50
-n, 1, A
-n, 2, B
-ej, T1, 1, 2
-discovery, 1
-sample, 2
-daf, daf
-random_discovery, True
```

An example of an array template is:

```
chr22       0       15929526
chr22       0       15991515
chr22       0       16288162
chr22       0       16926611
chr22       0       16990146
chr22       0       17498992
chr22       0       17540297
chr22       0       17728199
chr22       0       17760714
chr22       0       18180154
chr22       0       18217275
chr22       0       18220413
```

Ordering of time-specific events

When using priors, if some demographic events must happen in a certain order, the order can be specified by adding the order number to the argument. For example say there are two demographic events, a population split and instantaneous growth, but the instantaneous growth must happen before the population split, we can indicate that in the model file:

```
-en_1, Tgrowth, 1, A2
-ej_2, Tsplit, 2, 1
```

Additionally, the same format can be used to indicate that multiple events should happen at the same time. If there are multiple events that should happen at the same time, the word `inst` should be used instead of a time parameter after

the first definition of the time. *(this will actually cause the times to be just different enough that macs is happy.)* For example, say we wanted growth to occur at the same time as the population split:

```
-en_1, Tgrowth, 1, A2
-ej_1, inst, 2, 1
```

In this case, the population split will technically be simulated slightly after the growth.

germline

currently has a bug

The option `-germline` can be included in the model_file to use GERMLINE to find shared IBD segments between all simulated individuals from pseudo array. Does not use the genetic map to run GERMLINE. Runs GERMLINE as:

```
bash ./bin/phasing_pipeline/gline.sh ./bin/germline-1-5-1/germline  ped_name map_name␣
↪out_name "-bits 10 -min_m min_m"
```

If GERMLINE does not run, try rebuilding it on the machine you are trying to run on:

```
cd ./bin/germline-1-5-1
make clean
make
```

pedmap

The option `-pedmap` can be included in the model_file to print a ped and map file of the pseudo array data.

CHAPTER 4

High Throughput Computing

4.1 Open Science Grid

1. Create an OSG Connect account. https://osgconnect.net/signup
2. Join the project SimPrily
3. Create an ssh key pair

Log onto Open Science Grid Connect

```
ssh user-name@login01.osgconnect.net
```

Clone the entire repository. *We only need the pegasus_workflow directory*

```
git clone https://github.com/agladstein/SimPrily.git
```

4.1.1 Test with interactive Singularity container

Start the Singularity container and run a small test.

```
[agladstein@login02 ~]$ singularity shell --home $PWD:/srv --pwd /srv /cvmfs/
↪singularity.opensciencegrid.org/agladstein/simprily\:latest
Singularity: Invoking an interactive shell within container...

$ bash
agladstein@login02:~$ export PATH=/usr/local/bin:/usr/bin:/bin
agladstein@login02:~$ python /app/simprily.py examples/eg2/Param_file_eg2.txt␣
↪examples/eg2/model_file_eg2.csv 2 out_dir
```

4.1.2 Submit a Pegasus workflow

All components of the Pegasus workflow are located in the directory `pegasus_workflow`.

Start the workfow by running `submit` on the command line from the `pegasus_workflow` directory. There are 3 required arguments and 2 optional arguments

```
./submit -p PARAM -m MODEL -j NUM [-g MAP] [-a ARRAY]
```

Required

-p PARAM	The location of the parameter file
-m MODEL	The location of the model file
-j NUM	The number of jobs to run. The `ID` will go from 1 to `NUM`.

Optional

-g MAP	The location of the genetic map file
-a ARRAY	The location of the array template file, in bed form

We recommend that all testing be done before submiting workflows to OSG. Therefore we do not include the verbose options. Pegasus provides run information, so we do not include the profile option with the OSG workflow.

Example workflow submissions

e.g. (No pseudo array and no recombination map)

```
./submit -p ../examples/eg2/param_file_eg2.txt -m ../examples/eg2/model_file_eg2.csv -
↪j 10
```

e.g. (include pseudo array, but no recombination map)

```
./submit -p ../examples/eg2/param_file_eg2_asc.txt -m ../examples/eg2/model_file_eg2_
↪asc.csv -j 10 -a ../array_template/ill_650_test.bed
```

e.g. (recombination map, but no pseudo array)

```
./submit -p ../examples/eg2/param_file_eg2.txt -m ../examples/eg2/model_file_eg2.csv -
↪j 10 -g ../genetic_map_b37/genetic_map_GRCh37_chr1.txt.macshs
```

e.g. (include pseudo array, and recombination map)

```
./submit -p ../examples/eg2/param_file_eg2_asc.txt -m ../examples/eg2/model_file_eg2_
↪asc.csv -j 10 -a ../array_template/ill_650_test.bed -g ../genetic_map_b37/genetic_
↪map_GRCh37_chr1.txt.macshs
```

4.1.3 Monitoring and Debugging

To find the run times of the executable:

```
pegasus-statistics -s all
```

Then, look at `Transformation statistics`.

4.1.4 How the Pegasus workflow works

`submit` -> `tools/dax-generator` -> `wrappers/run-sim.sh`

`submit` will run `tools/dax-generator`, which constructs the workflow. The `dax-generator` is the main Pegasus file. The `dax-generator` creates the HTCondor dag file. It also tells Pegasus where the local files are and transfers files (from submit host to compute node) so they are available for the job. It also defines how to handle output files.

`wrappers/run-sim.sh` is the wrapper that runs in the container. It modifies the environment, and runs SimPrily.

4.2 Recommendations for other HTC workflows

coming soon

CHAPTER 5

Calculating summary statistics on real data

5.1 Data format

Real data must be in PLINK .tped file with 0's and 1's. Sites in rows, individuals in columns (first 4 columns chr, rsnumber, site_begin, site_end). The populations must be in the same order as specified in the model file for the simulations.

Put the individuals in the correct order https://www.cog-genomics.org/plink2/data#indiv_sort

```
plink --bfile bfile --indiv-sort f sample_order.txt --make-bed --out bfile_ordered
```

To get in the .tped format from .bed .bim .fam with 0's and 1's refer to https://www.cog-genomics.org/plink2/formats#tped

```
plink --bfile bfile --recode transpose 01 --output-missing-genotype N --out tfile01
```

5.2 Usage

`real_data_ss.py` takes 5 arguments:

1. `model_file`
2. `param_file`
3. `output_dir`
4. `genome_file`
5. `array_file`

e.g.

```
python real_data_ss.py examples/eg1/model_file_eg1.csv examples/eg1/param_file_eg1.
↪txt out_dir ~/data/HapMap_example/test_10_YRI_CEU_CHB.tped ~/data/HapMap_example/
↪test_10_YRI_CEU_CHB_KHV_hg18_ill_650.tped
```

(continues on next page)

(continued from previous page)

CHAPTER 6

Tutorial with Docker

1. What do you want to simulate? How many simulations?
2. Create your model.csv and param.txt input files.
3. Perform a small test simulation.

6.1 1. Define your simulation

1. Draw your model.
2. Estimate the required resources.
3. Decide where to run your simulations.

6.2 2. Create input files

6.3 3. Perform test simulation

6.3.1 a. Pull Docker image

Pull the latest SimPrily Docker image:

```
docker pull agladstein/simprily
```

Once you have successfully pulled the image you will see something like this:

```
Using default tag: latest
latest: Pulling from agladstein/simprily
f49cf87b52c1: Pull complete
7b491c575b06: Pull complete
```

(continues on next page)

(continued from previous page)

```
b313b08bab3b: Pull complete
51d6678c3f0e: Pull complete
09f35bd58db2: Pull complete
f7e0c30e74c6: Pull complete
c308c099d654: Pull complete
339478b61728: Pull complete
d16221c2883e: Pull complete
df211aed0ee8: Pull complete
94afb574a896: Pull complete
b253919783b5: Pull complete
45cb233ca3a5: Pull complete
Digest: sha256:1de7a99a23264caa22143db2a63794fa34541ccaf9155b9fb50488b5949a9d7d
Status: Downloaded newer image for agladstein/simprily:latest
```

Next, double check the images you've pulled:

```
docker image ls
```

You should see something like this:

```
REPOSITORY                    TAG                 IMAGE ID            CREATED
↪      SIZE
agladstein/simprily           latest              1d3fbe956b00        5 hours ago
↪      938MB
```

6.3.2 b. Run SimPrily

Run one small example with the Docker container

```
docker run -t -i --mount type=bind,src=/home/agladstein/src/SimPrily,dst=/app␣
↪agladstein/simprily python /app/simprily.py -p examples/eg1/param_file_eg1.txt -m␣
↪examples/eg1/model_file_eg1.csv -g genetic_map_b37/genetic_map_GRCh37_chr1.txt.
↪macshs -a array_template/ill_650_test.bed -i 1 -o output_dir -v
```

You should see something like this:

```
debug-1: Debug on: Level 1
JOB 1
debug-1: name   total   panel   genotyped
debug-1: A      140     0       140
debug-1: B      20      0       20
debug-1: total samples: 160
debug-1: Perform simulation and get sequences
debug-1: Number of sites in simulation: 10309
debug-1: Calculating summary statistics

#########################
### PROGRAM COMPLETED ###
#########################
```

Then, you should see a new directory created `/home/agladstein/src/SimPrily/output_dir`. In that directory, you should see the directories

```
sim_data
germline_out
results
```

and the directory `results` should have the file `results_1.txt`, which should look something like this:

```
A       AN_t    B       AB_t    AN      SegS_A_CGI      Sing_A_CGI      Dupl_A_CGI
↪  TajD_A_CGI      SegS_B_CGISing_B_CGI       Dupl_B_CGI      TajD_B_CGI      FST_AB_
↪CGI
29380.6397673   1615.50194862   42155.6351482   2546.95287896   10000.0 9795    3880
↪  1283    -1.30415802172  4360       1690    674     -0.488311472745 0.
↪00115531480069
```

6.4 4. Perform HTC simulations

CHAPTER 7

For Developers

7.1 Containers

7.1.1 Docker

Notes on installing Docker, creating a Docker image, and running a Docker container. *The following instructions for Docker require sudo privaliges. Check the Docker documentation for what to do if you do not have sudo.*

Installing Docker

Check that Docker is installed:

```
sudo docker run hello-world
```

Quick and easy install script provided by Docker:

```
curl -sSL https://get.docker.com/ | sh
```

OR

If not on Linux, you can use Vagrant.

```
vagrant up
vagrant ssh
```

Then, continue with Linux steps.

See https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-docker-ce

For Mac or Windows see Docker documentation.

Dockerize

1. Create Dockerfile
2. Build Docker image
3. Push Docker image to Docker Hub

1. Create Dockerfile In the directory with the necessary code and requirements.txt

Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:2.7

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install -r requirements.txt

# Create directory for OSG
RUN mkdir -p /cvmfs

# Make executable
RUN chmod +x /app/simprily.py

# Make port 80 available to the world outside this container
EXPOSE 80

# Define entry point
#ENTRYPOINT ["python", "/app/simprily.py"]
```

See https://docs.docker.com/engine/reference/builder/

2. Build Docker imiage

```
sudo docker build -t agladstein/simprily .
```

3. Push Docker image to Docker Hub

Must first login to Docker Hub

```
sudo docker login
```

```
sudo docker push agladstein/simprily
```

Run program with Docker container

Pull image:

```
sudo docker pull agladstein/simprily
```

Run program:

```
docker run -t -i --mount type=bind,src=/home/agladstein/docker_test/SimPrily,dst=/app␣
↪agladstein/simprily_autobuild:version1 python /app/simprily.py -p examples/eg1/
↪param_file_eg1.txt -m examples/eg1/model_file_eg1.csv -g genetic_map_b37/genetic_
↪map_GRCh37_chr1.txt.macshs -a array_template/ill_650_test.bed -i 1 -o output_dir -v
```

try running with port ‘‘-p‘‘

or Run Docker container interactively to poke around

```
docker run --rm -it --entrypoint=/bin/bash agladstein/simprily_autobuild:version1
```

Cheat sheet

Some useful commands

```
docker build -t friendlyname .  # Create image using this directory's Dockerfile
docker run -p 4000:80 friendlyname  # Run "friendlyname" mapping port 4000 to 80
docker run -d -p 4000:80 friendlyname         # Same thing, but in detached mode
docker container ls                                # List all running containers
docker container ls -a             # List all containers, even those not running
docker container stop <hash>           # Gracefully stop the specified container
docker container kill <hash>         # Force shutdown of the specified container
docker container rm <hash>        # Remove specified container from this machine
docker container rm $(docker container ls -a -q)         # Remove all containers
docker image ls -a                             # List all images on this machine
docker image rm <image id>            # Remove specified image from this machine
docker image rm $(docker image ls -a -q)   # Remove all images from this machine
docker rmi $(docker images -q)  # Remove all containers from this machine
docker login             # Log in this CLI session using your Docker credentials
docker tag <image> username/repository:tag  # Tag <image> for upload to registry
docker push username/repository:tag            # Upload tagged image to registry
docker run username/repository:tag                   # Run image from a registry
```

Resources

https://docs.docker.com/get-started/ https://github.com/wsargent/docker-cheat-sheet https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-docker-ce https://docs.docker.com/engine/reference/builder/ https://docs.docker.com/engine/reference/commandline/run/#add-bind-mounts-or-volumes-using-the-mount-flag http://codeblog.dotsandbrackets.com/persistent-data-docker-volumes/

7.1.2 Singularity

These are preliminary notes, not specific to a SimPrily Singularity container.

Installing Singularity

To install Singularity:

```
git clone https://github.com/singularityware/singularity.git
cd singularity
sudo apt-get install libtool
sudo apt-get install autotools-dev
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install automake
./autogen.sh
./configure --prefix=/usr/local
make
sudo make install
```

Create empty image

To create an empty Singularity image:

```
create --size 2048 simprily-little.img
```

Make or pull a container

1. Make container by dumping docker layers into empty image:

```
import simprily-little.img docker://agladstein/simprily-little
```

or

2. Pull container

```
singularity pull docker://centos:latest
```

or

3. Bootstrap

Create Singularity specification file.

For example:

```
Bootstrap: docker
From: ubuntu:latest

%runscript

    echo "I can put here whatever I want to happen when the user runs my container!"
    exec echo "Hello Monsoir Meatball" "$@" #The $@ is where arguments go

%post

   echo "Here we are installing software and other dependencies for the container!"
   apt-get update
   apt-get install -y git
```

Then build image from Singularity file:

```
sudo singularity bootstrap analysis.img Singularity
```

Run container

1. from Singularity Hub

```
singularity run shub://vsoch/hello-world
```

or

2. from local container with input arguement

```
singularity run analysis.img Ariella
```

Shell into a container

```
singularity shell centos7.img
```

Resources

- http://singularity.lbl.gov/quickstart
- http://singularity.lbl.gov/singularity-tutorial
- https://singularity-hub.org/faq

7.2 Testing

The shell script `autoTesting.sh` is included for quick automated testing of included examples.

It is run as:

```
./autoTesting.sh PYTHON [EXAMPLE_INT]
```

Where,
`PYTHON` is the python to use
`EXAMPLE_INT` is the specific example number to test (optional). If it is not specified, it will test all of the examples.

7.3 Creating Documentation

- Install Sphinx:

```
pip install Sphinx
```

- To edit the Read The Docs, edit the Sphinx .rst files in `SimPrily/docs`.
- Build the html from restructured text:

```
~/simprily_env/bin/sphinx-build -b html source build
```

7.3.1 Resources

- http://www.sphinx-doc.org/en/stable/tutorial.html
- https://github.com/ralsina/rst-cheatsheet/blob/master/rst-cheatsheet.rst
- https://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html#headings
- http://rest-sphinx-memo.readthedocs.io/en/latest/ReST.html

7.4 Other Notes

- If you use import a new Python package make sure you add it to the requirements.txt file then create the requirements.in. This will insure that the package installed in the virtual environment and Docker image.

```
pip-compile --output-file requirements.txt requirements.in
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search