Odyssey Tutorial

Written by Ryan Eller – Updated July 2018

# Brief Overview:

The Odyssey Tutorial is meant to get you quickly up to speed on how to use Odyssey and its main workflow, which includes using the DataPrep, Impute, Phase, and GWAS Modules. However, the tutorial will also briefly cover some useful smaller modules, including the sub-modules (HGDP Starter and Remapping\_Made\_Easy sub modules). The estimated time to complete the Tutorial (including imputing the subset of HGDP samples) depends on user skill, hardware and whether the user decides to use the pre-made HGDP cleaned dataset. The minimum time to complete everything is approximately 3-6 hours. There is very little code you will need to run outright as all of Odyssey is automated. However, the sub-modules responsible for data prep lie outside of the main Odyssey workflow. These sub-modules are covered under this tutorial as an optional first step (as cleaned data is already provided for you within the tutorial folder). If you want to skip these optional steps then you may skip to Step 1: Running Quality Control Prior to Phasing/Imputation of this manual. If you feel inclined to run the sub-modules then it will be important to note that any code you will need to execute starts with $ and is highlighted in grey (e.g. $ cd ..). Also note that all of the scripts are fully annotated – every line of code has a commented line explaining what it does and sometimes how the code works.

# What You Need to Know to Run Odyssey:

A basic understanding of Linux commands and filesystems (e.g. what is SCP, what’s a terminal). It would also be helpful to have a basic understanding and some hands-on experience with installing programs onto Linux platforms. Lastly, knowing how to run jobs on High Performance Systems (i.e. HPS or HPC) would be extremely beneficial as imputation is a computationally intensive task that is best suited for cluster computing and parallelization in general.

## Pre-Step A: Getting the Tutorial Data and Converting/Cleaning It

### Estimated Time to Completion:

* 0-60 minutes depending on whether you use the provided data or create it yourself from scratch

### *Folder Location:*

* Refer to the HGDP\_StarterSubModule within the DataPrepModule

### Overview:

In this tutorial we will be using 100 European (Italian and French) individuals from the HGDP dataset. The dataset contains approx. 661,000 genome-wide markers (mapped to NCBI36 or hg18). A list of HGDP IDs used in this tutorial are defined (see the HGDP\_Euros tab within the HGDP\_StarterPack.xlsm) as a reference. You are more than welcome to download this data for yourself and follow along with the tutorial, however, the data requires some extensive cleaning since it is mapped to an old Homo\_Sapien build, hg18, and Odyssey is setup to run on hg19 (or GRCh37) due to the reference genome provided by the Shapeit2 software. That being said Odyssey is flexible and can be run on any dataset provided it corresponds to its corresponding reference dataset and configured properly. I have provided a “cleaned” and remapped (to hg19/GRCh37) subset of the HGDP dataset for you to use. It is located in a zipped tarball in ./Odyssey/Tutorial\_Data. If you are adventurous or just excited to do more bioinformatic work, I’ve created several sub-modules to get the tutorial HGDP file I provided. The first sub-module, HGDP\_Starter-SubModule helps you download, convert, and filter the Stanford created HGDP flat data files into Plink format. The second sub-module, Remapping\_Made\_Easy-SubModule, helps you remap the HGDP Plink file from hg18 to hg19. Actually, the remapping sub-module could be used to remap any Plink .bed/.bim/.fam trio to any genome build supported by NCBI. So that submodule may be useful in and of itself.

### Sub-Module Prerequisites:

As these submodules are not official part of the Odyssey workflow, their scripts do not call into the Odyssey Config.conf and Programs.conf file which set variables and program paths. Therefore, as both sub-modules require plink, you will need to install it on the system and add it to your $PATH. If you can call “plink” from your terminal and it executes the sub-modules will work. If not, you will need to manually install the plink program and add it to $PATH. If you need help doing this try and execute the following command to temporarily add the Plink exec file to your PATH.

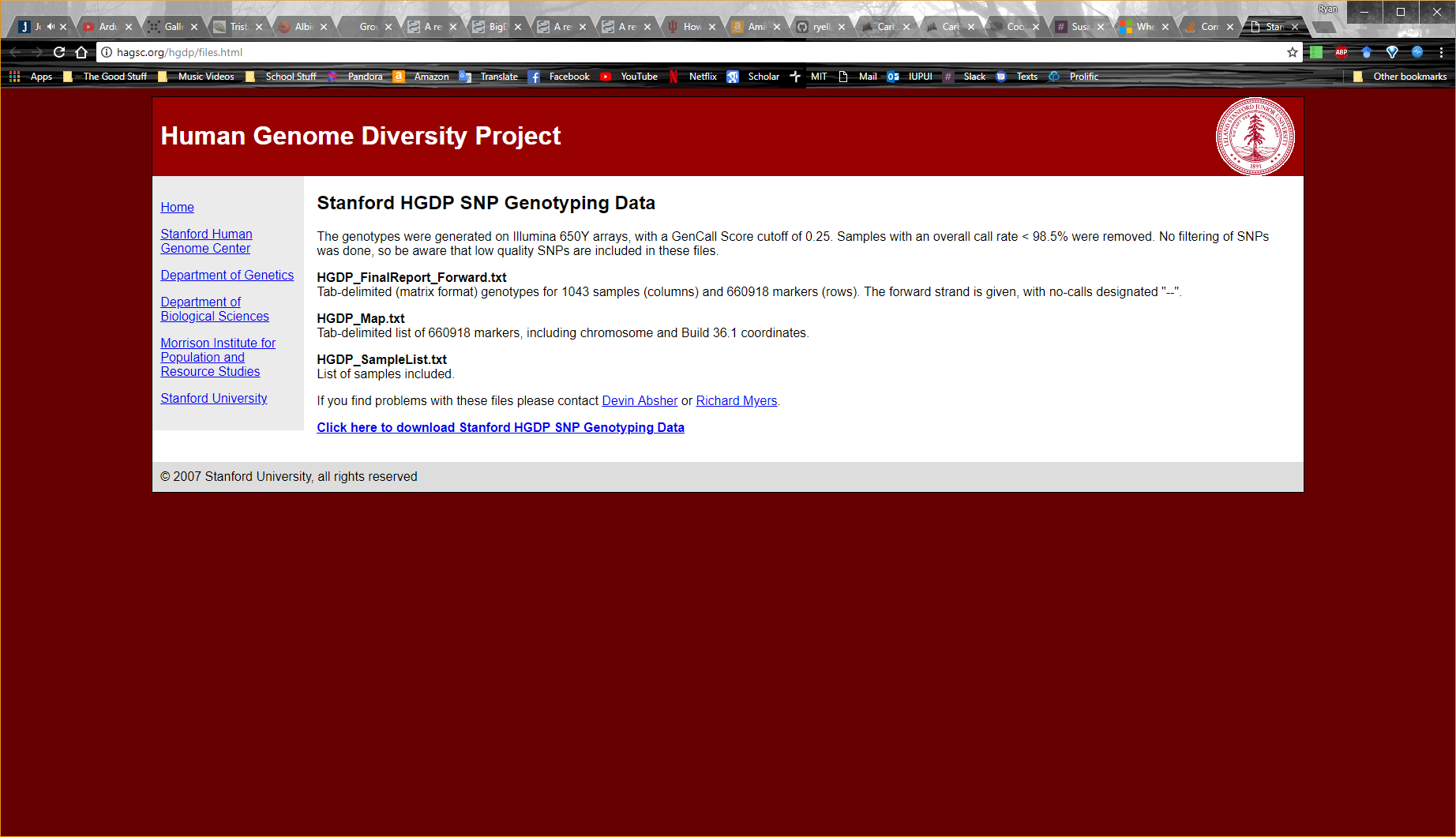
$ SET PATH=%PATH%; /path/to/Plink/exec

### Walkthrough of HGDP\_Starter Sub-Module:

**Note:** Some of the text files created within Excel used to create and filter the final HGDP files have been provided for you. If you want to learn more about how these were created, then refer to the HGDP\_StarterPack\_Technical\_Info.xlsm which provided as annotated explanation of every line in the “FixHGDP.sh” script provided within the HGDP\_Starter-SubModule folder.

#### HGDP Starter Sub-Module Steps:

Downloading the HGDP data isn’t automated as HTML links change. Therefore, before running the script make sure that the links exist.

1. We will need to download and unzip the HGDP raw files (3 in total) from Stanford University (there's other places to download HGDP data, but this is the one I used as of 2018):
   1. <http://hagsc.org/hgdp/files.html>
   2. It should look like this, and give you the option to download 3 files
2. Place the 3 files "**HGDP\_FinalReport\_Forward.txt**", "**HGDP\_Map.txt**", and "**HGDP\_SampleList.txt**" in the following directory: **./Odyssey/DataPrepModule/HGDP\_Starter-SubModule**
   1. Notice that the “./” refers to your current directory that will change based on your system setup
   2. The other files you may do whatever you want with as these won’t be utilized
3. While we are here in “**./Odyssey/DataPrepModule/HGDP\_Starter-SubModule**”, make this directory your current working directory (this will allow 01\_Convert-FilterHGDP.sh to orient itself and find all the files it needs to in order to convert/filter the HGDP files)
   1. Remember to change the “./” to reflect your current system setup

$ cd ./Odyssey/DataPrepModule/HGDP\_Starter-SubModule

1. Once the 3 Stanford data files are placed within the HGDP\_Starter-SubModule and the current working directory is set to ./HGDP\_Starter-SubModule, you may run 01\_Convert-FilterHGDP.sh which will convert the HGDP data into a Plink format and filter it down to 952 individuals.
   1. To run the script open terminal, navigate to the script via “cd” commands, and enter:

$ sh 01\_Convert-FilterHGDP.sh

1. Run 02\_FilterEuroHGDP.sh which will filter the 952 HGDP individuals down to those who contain French and Italian ancestry.

### Walkthrough of Remapping\_Made\_Easy-SubModule:

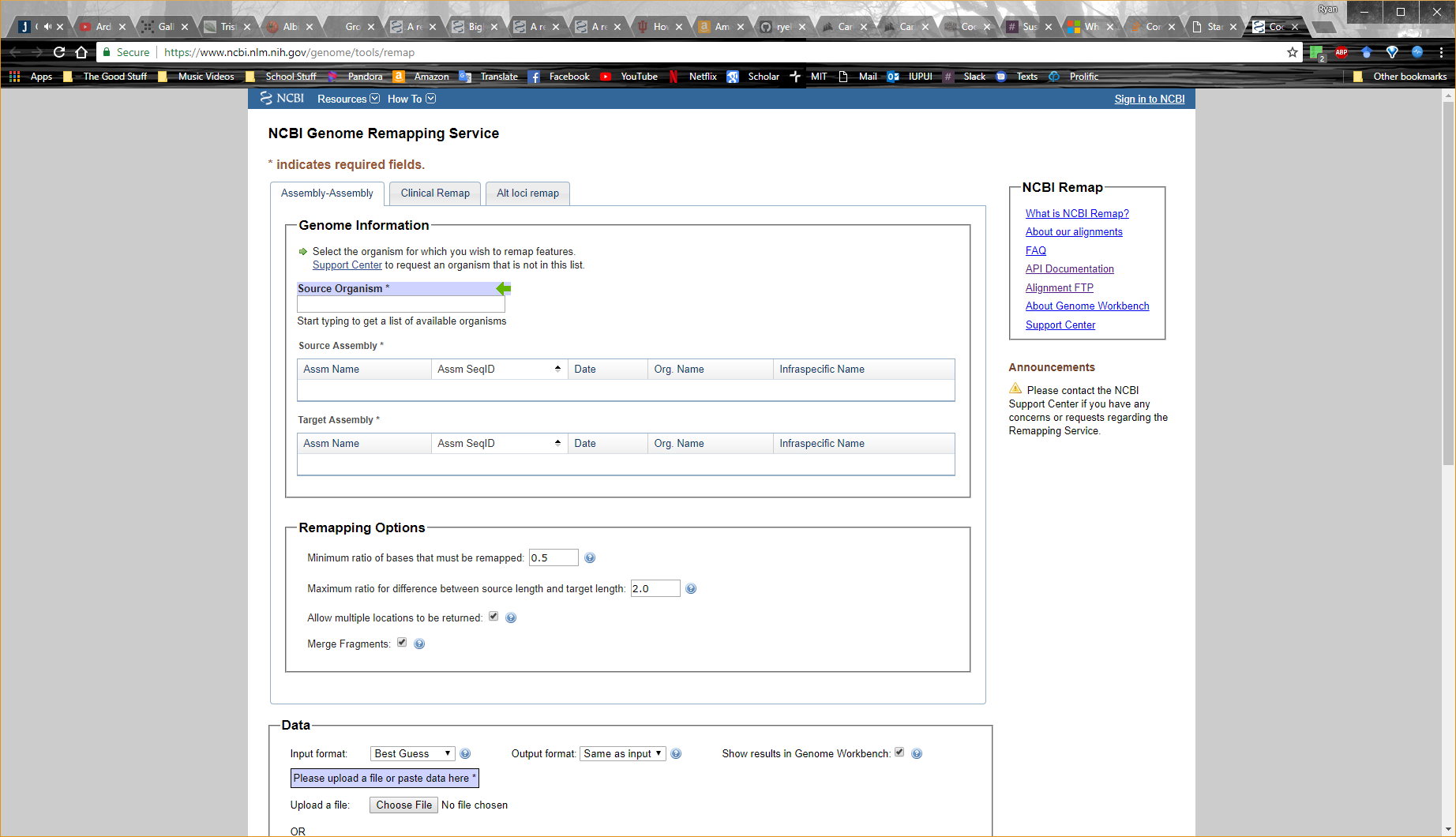
**Note:** Remapping a dataset from one genomic build to another can be done many ways (e.g. UCSC’s LiftOver, NCBI’s Remapping Service, etc.). While I don’t endorse one method over another, I found the following method using NCBI’s service to be the easiest for my particular workflow.

**Another Note:** NCBI offers a web version of their Remapping Service in addition to a Perl-based downloadable API. I will provide instructions on how to use the web-service, but I will also list the Perl command you can use to achieve the same remapping service with their API. If you do plan on using their API, I would highly suggest reading its documentation (<https://www.ncbi.nlm.nih.gov/genome/tools/remap/docs/api>) explaining how to set it up.

#### Remapping Made Easy Sub-Module Steps:

1. As this is a continuation of the HGDP\_Starter-SubModule, you should already have the “HGDP\_Euro” bed/bim/fam files (in addition to .hh and .log files), but know that the following method should work with ANY properly formatted Plink bed/bim/fam files. Move the bed/bim/fam Plink files into the Remapping\_Made\_Easy-SubModule folder directory.
2. Set the Remapping\_Made\_Easy-SubModule directory as your current working directory

$ cd ./Odyssey/Remapping\_Made\_Easy-SubModule

1. Run the **0a\_ChangeChrNameScheme-Convert2BED.sh** script from the Remapping\_Made\_Easy-SubModule folder directory. This script renames the .bim file since NCBI likes a particular naming convention with the chromosome names listed as chr1 and chrX instead of the Plink default of names like “1” and “23”. It also converts the renamed .bim file into a BED file. I’ve found remapping to be the easiest when you give NCBI a BED file. This is NOT a Plink .bed file (don’t you just love identical file extension names?). Despite the slightly confusing name, this can be easily created from a .bim Plink file and is done automatically with this script.
   1. **Note:** As long as you ONLY have A SINGLE .bed/bim/fam Plink document in this folder to begin with, the script will find the necessary files and carry your naming scheme throughout the process (adding numerical prefixes to keep track of the step number).
2. To perform the actual remapping on your data you will need an annotation file provided by NCBI. You may choose to either use NCBI’s online service or download their Perl-based API. The following instructions will focus on using the online service
   1. Go to: <https://www.ncbi.nlm.nih.gov/genome/tools/remap>
      1. 
   2. Select the “Assembly-Assembly” tab
   3. Input the following **Genome Information**
      1. Source Organism: Homo sapiens
      2. Source Assembly: NCBI36 (hg18) [GCF\_000001405.12](https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.12)
      3. Target Assembly: GRCh37 (hg19) [GCF\_000001405.13](https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.13)
   4. Input the following **Remapping Options**
      1. Default “Minimum ratio of bases…”: 0.5
      2. Default “Maximum ratio for difference…”: 2.0
      3. “Allow multiple locations to be returned”: False (no check within box)
      4. “Merge Fragments”: True (check within box)
   5. Input the following **Data**
      1. Input format: BED
      2. Output format: BED
      3. “Show results in Genome Workbench”: False (no check within box…unless you want to use it)
      4. Upload File: [Your newly created .BED2 file – NOT the Plink .bed file]
         1. **Note:** I made the script output a .BED2 file so that Plink won’t get confused with the .BED and .bed files)
   6. Submit (it takes a while for large files)
   7. Observe the mapping report to see how well the .BED was remapped. You should see that a majority of the variants from Chr1-22 + ChrX +ChrY have been successfully remapped (i.e. The number of source features and intervals should be roughly the same as the remapped features and intervals). Unfortunately, mitochondrial variants aren’t remapped. I believe this is due to an issue with the assembly versions – regardless there shouldn’t be too many and mitochondrial variants are not imputed anyway so we can drop them in the next step.
      1. Here’s the Perl-based equivalent for NCBI’s Remapper-API:

$ perl ./remap\_api.pl --mode asm-asm --from GCF\_000001405.12 --dest GCF\_000001405.13 --annotation [Path/to/BED\_FILE].BED --annot\_out remapped\_[BED\_FILE].BED --report\_out Report\_remapped[BED\_FILE].txt --allowdupes off --in\_format bed --out\_format bed

1. Download the annotation file from NCBI. It’s naming scheme should be “**remapped\_ [Name of your file].BED2**”
2. Place the remapped BED file into the Remapping\_Made\_Easy-SubModule
3. Execute the **0b\_RemNonMapped-UpdateBim.sh** script which will extract the variants that were remapped from the Remapped.BED2, remove the variants that weren’t remapped from the Plink file, update the genomic coordinates and chromosome of the variants that were remapped, change back the chromosomal naming scheme (from chrX to 23), and exit.
4. **Troubleshooting Note:** You may encounter an error due to non-permittable chromosome codes (e.g. chr4\_ctg9\_hap1). Automating troubleshooting is tough, so if this occurs you will need to modify or remove the offending NCBI entries in the remapped annotation file and re-run the script. While every troubleshooting situation is different you may find running 0c\_Modify\_BED\_File.sh will fix your issue as it is designed to remove variants mapped to contigs by removing BED entries containing underscores.
   1. Additional troubleshooting help may be found in the comment section within 0b\_RemNonMapped-UpdateBim.sh and 0c\_Modify\_BED\_File.sh

## Pre-Step B: Fixing the Mapping and Alleles to a Reference Annotation

### Estimated Time to Completion:

* 15-45 minutes depending on how much troubleshooting you’ll need to perform to get your data aligned to the reference genome

### Folder Location:

* DataPrepModule Folder

### Overview:

Now that the HGDP data has been converted, filtered and remapped we need to fix it to a reference genome like GRCh37 to fix any coordinate errors, flipped alleles, etc. To do this we will use the same data prep that the Sanger Imputation Server (<https://imputation.sanger.ac.uk/>) recommends. BCFtools and its plugin +fixref will do most of the heavy lifting (refer to <https://samtools.github.io/bcftools/howtos/plugin.fixref.html>). Unfortunately, this program only works with .bcf files so we will need to do additional conversions. We will also need to download reference data from 1000 Genome and NCBI.

### Module Prerequisites:

* BCFtools v1.8 or later and the BCFtools plugin +fixref
  + <https://samtools.github.io/bcftools/bcftools.html>
* htslib v1.8 or later (comes with BCFtools)
  + Specify the BCFtools executable file in the Programs.conf file
* Reference data (.fasta reference and a .vcf annotation)

**Note:** Since you are transitioning from Odyssey Sub-Modules to Modules (the main part of the pipeline) you will be able to execute all scripts from the main Odyssey Folder (so you should never have to worry about working directories again) and specify the run options from the Config.conf configuration file. You will also need to specify all Odyssey dependency programs in the Programs.conf configuration file.

### Setting Up Program Dependencies:

While we will refer you to the individual installation guides of Odyssey’s prerequisites, you will need to specify the Odyssey prerequisite program executable file path in Programs.conf (this is to avoid having to go into a discussion about modifying $PATH – unless you had to do this in the sub-modules in which case you are ahead of the curve).

For example, if I have the executable plink installed in /home/programs/plink1.09/, then to setup Plink within Programs.conf I would fill out the following:

Plink\_Exec="/home/programs/plink1.09/plink";

### Specifying Odyssey Options:

Odyssey allows for a large amount user control, most of which can be specified from the Config.conf file. Some of these options allow you to modify the execution of a script, skip analysis you don’t want, and provide important input for the programs to function. Even more control can be taken if you delve into the fully annotated scripts themselves. With the exception of the sub-modules all scripts can be viewed from the main Odyssey folder and are numbered to reflect the order they should be run.

### Data Prep Module Steps:

1. To start cleaning your data put your Plink .bed/bim/fam files here:

**./Odyssey/DataPrepModule/PLACE\_DATA\_2B\_FIXED\_HERE**

* 1. **Note:** You do not have specify the name of the file to Odyssey. Odyssey takes the name you have given the file trio (although they all must have the same “base name”; e.g. Dataset.bed, Dataset.bim, Dataset.fam) and runs with your naming scheme.
  2. If you are following along from earlier in the tutorial I would recommend renaming the files to something shorter (for example I renamed my files to “HGDP\_EuroF”).

1. From **Config.conf** look at the section labelled “**DataPrep Module Variables**.” You can control the execution of each step by telling the step to either run (T) or not run (F). You may try running all the steps at the same time to save time, but if an error occurs you can run the steps individually to identify the problem more easily. In general, the steps are as follows:
   1. **DownloadRef:** Downloads reference annotations to the RefAnnotationData folder, and extracts data. The default is to download reference data for GRCh37 from:
      1. <ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz>
      2. <ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b150_GRCh37p13/VCF/All_20170710.vcf.gz>
   2. **Step 1:** Converts you input Plink files into a .BCF while renaming the chromosome codes
      1. **NOTE:** BCFtools +fixref can only “align/fix” your data to the reference annotation you provide (e.g. human\_g1k\_v37.fasta) if the chromosome codes from your bcf match the chromosome codes in the fasta annotation. If they don’t match, then the variants will be dropped.
      2. To check the chromosome codes of your fasta you may refer to the assembly report that normally accompanies the fasta download (e.g. <https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.25#/st_info-by-region>)
      3. To check the chromosome codes for your data you may run a head command on your VCF file and check the contig ID’s (e.g. $ head -30 DataFixStep1\_HGDP\_Euro.vcf)
   3. **Step 2:** Aligns your BCF file to the reference annotation and fixes coordinates errors and allele mismatches with BCFtools
   4. **Step 3:** Sorts the reference aligned BCF output and convert it back into Plink format for Phasing
2. Run **0\_DataPrepCleanup.sh**
3. Any indels or non-biallelic SNP’s will be dropped in this step as Plink 1.9 is unable to code VCF files of multiallelic variants. This however, will change with the advent of Plink 2.0. In addition, imputation will likely make up for the variants that are lost in this step
4. Cleaned Plink data will be prefixed by “**DataFixStep3**”
5. After running **0\_DataPrepCleanup.sh** DataFixStep3\_HGDP\_EuroF-RefFixSorted.bed, bim, fam in the DataPrepModule folder which contained 501457 markers and 100 people.

## Step 1: Running Quality Control Prior to Phasing/Imputation:

### Estimated Time to Completion:

* 15 minutes

### Folder Location:

* Target Project Folder

### Overview:

Cleaned data will now be further cleaned (there’s lots of data cleaning) by filtering for minor allele frequency, Hardy-Weinberg equilibrium, and genotype and individual missingness filters of 95% (ie. Individuals missing more than 5% of genotypic data and genotypes missing in more than 5% of individuals will be dropped). If you want to skip the previous steps you can find the cleaned data that has been generated up to this step in the Tutorial\_Data folder within Odyssey.

### Prerequisites:

* Plink 1.9
* A Plink .bed, .bim, and .fam file to be QC’ed prior to phasing (if using the tutorial data unzip Tutorial\_Data.tar.gz and use it)

### QC Steps:

1. Place a SINGLE Plink .bed/bim/fam dataset (preferably a dataset that has been correctly mapped to a reference annotation – see the previous step) into the following folder:

**./Odyssey/Target/PLACE\_NEW\_PROJECT\_TARGET\_DATA\_HERE**

* 1. **Note:** For our HGDP data, DataFixStep3\_HGDP\_EuroF-RefFixSorted, I renamed the files to HGDP\_EuroImputedSex since I’m not a big fan of really long names and over the course of the pipeline the names tend to grow by around 10-15 characters. As can be inferred by the name I also used Plink to impute the sex of individuals based on the 12742 X chromosome variants in the dataset out of convenience.

1. Refer to the “**Variables Used for Most of the Scripts**,” “**Imputation Project Variables**,” and “**1\_ImputeProjectSetup-QC-Split Script Variables**” sections within the **Config.conf** file in the main Odyssey directory and fill out the following bolded variables.
   1. **WorkingDir** allows you to set the path of the Odyssey folder so that it will serve as a permanent working directory for Phasing, Imputation, and GWAS analyses
      1. **Tip:** To make your life easier add a “/” to the end of your path letting Linux know that Odyssey is a folder not a file
   2. **Email** allows you to get notifications from the HPC (High Performance Cluster) to alert you when your job starts, begins, ends, and aborts. This isn’t required but is useful.
   3. **BaseName** allows you to set the name of the Imputation Project. Projects are a method of allowing you to keep organized as all the files needed to Phase and Impute your data will be organized within directories that have the BaseName. It also helps modularize Odyssey so that you can quickly export directories full of phased or imputed data. This allows you to run multiple Imputation Projects at once without files jumbling together.
      1. Here I will name my project “HGDP\_EuroF” since I want all future folders and phased and imputed data labeled as such
   4. After making sure I only put 1 Plink dataset (a .bed, .bim, and .fam) into the **PLACE\_NEW\_PROJECT\_TARGET\_DATA\_HERE** folder I executed **1\_ImputeProjectSetup-QC-Split.sh**.
   5. As an aside, the QC the script performs can be seen in the following Plink commands executed from within the script:
      1. --geno 0.05
      2. --hwe 0.000001
      3. --maf 0.025
      4. --mind 0.05
   6. The result is 26 quality-controlled Plink bed/bim/fam files that contain 476560 variants (combined) from 100 people ready to be loaded into Shapeit2 for phasing. Chromosomes 24, 25, 26 (XY, Y, and MT) are empty in the HGDP data that I processed.

## Step 2: Phasing

### Estimated Time to Completion:

* 30 minutes – roughly 2 minutes per chromosome

### Folder Location:

* Data Deposited into Phase Project Folder

### Overview:

Chromosomal separated data that has been QC’ed will be phased by Shapeit2 based on 1000 Genome Haplotypes from Phase 3 (GRCh37 – hg19). Reference data can be downloaded from <https://mathgen.stats.ox.ac.uk/impute/1000GP_Phase3.html>, but know that any reference dataset can be used in Odyssey if setup correctly. From this step onward, all data used in the pipeline must be generated from the prior step. In other words, you cannot (easily) run the phasing step without first running the quality control step that proceeds it due to its file handling and naming conventions. Imputing the data in the next step without first phasing it in this step is also ill advised.

### Special Note:

This script is the first time that setting up a HPC (High Performance Cluster) Torque header is possible. Due to the resource intensive nature of phasing and imputing, it’s recommended that you use some sort of HPC, but if you have a high-end Linux computer (or perhaps you’re not pressed for time and can afford to run a couple of computing cores at a time) you can still use Odyssey for phasing with a minor modification. Altering the computing resources requested through Torque resource manager is as simple as altering the #PBS commands. Unfortunately, this isn’t something you can modify from Config.conf (yet). Instead you will need to open **2\_PhasingScriptMaker.sh** and look for the annotated tag:

**## ????????????????????????????????????**

**## Alter Script for HPC if necessary**

**## ????????????????????????????????????**

Underneath it will be the header for the Torque resource monitor. Adjust it however you’d like, and if you don’t know what a HPC or a Torque resource monitor is, you can ignore this entire note as your computer will simply treat the #PBS tags as commented out notation.

### Prerequisites:

* Shapeit2
* An Impute2 Reference Panel
* Quality Controlled Plink data (bed,bim,fam) from Step 1

### Phasing Steps:

1. You will first need to download a reference panel for Phasing and Imputation. I used this reference panel from the Impute2 site as Impute2 tends to do well with using admixed reference panels (and also when you have admixed target data).
   1. <https://mathgen.stats.ox.ac.uk/impute/impute_v2.html#reference> (List of all Impute2 reference panels
   2. <https://mathgen.stats.ox.ac.uk/impute/1000GP_Phase3.html> (The reference panel I used)
   3. Perform the following commands to wget the files and unzip their contents into the reference folder

$ wget --directory-prefix=./Odyssey/Reference/ <https://mathgen.stats.ox.ac.uk/impute/1000GP_Phase3.tgz>

$ tar xzvf 1000GP\_Phase3.tgz

$ wget --directory-prefix=./Odyssey/Reference/ <https://mathgen.stats.ox.ac.uk/impute/1000GP_Phase3_chrX.tgz>

$ tar xzvf 1000GP\_Phase3\_chrX.tgz

* 1. Using your own dataset is perfectly fine as well. You may need to adjust some of the variables listed in the “**Reference dataset section**” within Config.conf though as Odyssey looks for distinctive keywords to identify the genetic map, legend, and haplotype files. Config.conf explains more about how to adjust these variables so I will defer further explanation to it.

1. Shapeit2 works with the QC’ed Plink files created in the previous section. All we need to do is set the following variables in the “**2\_PhasingScriptMaker Variables**” section of Config.conf:
   1. **PhasingThreads** lets us tell the HPC as well as Shapeit2 how many computing cores we intend to use
      1. **Note:** If you are not using a HPC, setting **PhasingThreads** will simply tell Shapeit2 how many threads to run on your desktop
   2. **PhaseX** is a toggle that allows you to specifically control whether the X chromosome is phased
   3. **ExecutePhasingScripts** is again a toggle for HPC users. **2\_PhasingScriptMaker.sh** is a script which makes a bunch of smaller scripts that control the phasing of individual chromosomes. Having the option to create a script and not run it (i.e. for testing purposes) can be valuable. When you are ready to submit all your phasing scripts to the HPC, specify “T”. Or you can submit them individually from the script folder in which they are stored. This command executes a qsub command. For Linux desktop users you can set this variable to “F” and then run all the created scripts in the “**Scripts2Shapeit**” folder within the Phase Project Directory.
2. After executing this script for all my phasing runs I’m able to view the results, output logs, and the original scripts themselves from within the Phase Project Directory folder.

## Step 3: Imputation

### Estimated Time to Completion:

* 30 minutes

### Folder Location:

* Data Deposited into Impute Project Folder

### Overview:

Shapeit2 phased data will be imputed by Impute4 (for chromosomes 1-22) and Impute2 (for X chromosome due to Impute4 not being able to impute the missing genotypes of the non-PAR regions) based on 1000 Genome Reference Data (mentioned earlier in Step 2).

1. **Special Note:** Similar to the Phasing script, the Imputation script also has an option to modify the HPC Torque header. Again, it’s recommended that you use some sort of HPC, but imputing from a high-end Linux computer is possible. Altering or deleting the #PBS commands is as simple as as looking for the following tag within the **3a\_ImputeScriptMaker.sh script**:

**## ????????????????????????????????????**

**## Alter Script for HPC if necessary**

**## ????????????????????????????????????**

Underneath it will be the header for the Torque resource monitor. Adjust it however you’d like, and if you don’t know what a HPC or a Torque resource monitor is, you can ignore this entire note as your computer will simply treat the #PBS tags as commented out notation.

### Prerequisites:

* Impute4 (Odyssey uses Impute4 do its speed over Impute2 and its refined output files in comparison to its predecessor). At the time of this writing you will need to contact the developer to download Impute4 (<https://jmarchini.org/impute-4/>)
* Impute2 (for imputing the X chromosome)
* An Impute2 Reference Panel (same as in the Phase section)
* Phased data from Step 2

### Imputation Steps:

1. Impute4 and Impute2 works with the phased data created in the previous section. All we need to do is set the following variables in the “**3a\_ImputeScriptMaker Variables**” section of Config.conf:
   1. **ImputeX** is a toggle that allows you to specifically control whether the X chromosome is imputed
   2. **ImputeChrStart** is a way to specify from what chromosome imputation begins. This may be important if you don’t want to impute all the chromosomes at the same time (i.e. when a job handler on a HPC limits the amount of jobs you can submit at any one time since Impute will “chunk” the chromosomes and impute them separately – for humans chromosome 1 has 49 chunks alone).
   3. **ImputeChrEnd** is a way to specify what chromosome imputation ends. Similar to ImputeChrStart. In this way you can specify specific ranges for imputation to occur. For example setting **ImputeChrStart** to “1” and **ImputeChrEnd** to “10” will impute half the dataset. Then once those are completed you can change them to “11” and “23” respectively.
   4. **ExecuteImputationScripts** is a toggle for HPC users. **3a\_ImputeScriptMaker Variables** is a script which makes a bunch of smaller scripts that control the imputation of individual chromosomes. Having the option to create a script and not run it (i.e. for testing purposes) can be valuable. When you are ready to submit all your imputation scripts to the HPC, specify “T”. Or you can submit them individually from the script folder in which they are stored. This command executes a qsub command. For Linux desktop users you can set this variable to “F” and then run all the created scripts in the “**Scripts2Impute**” folder within the Impute Project Directory.
2. After executing this script on its default settings for all my imputation runs (although I did impute chromosomes 1-10 and 11-23 separately due to our HPC restrictions) I’m able to view the results (**RawImputation** Folder), output logs (**Scripts2Impute** Folder), and the original scripts (**Scripts2Impute** Folder) themselves from within the Impute Project Directory folder.

## Step 4: Concatenation of Imputation Results

### Estimated Time to Completion:

* 1-2 hours

### Folder Location:

* Data Deposited into Impute Project Folder

### Overview:

Data that is imputed through the Shapeit2-Impute2/4 pipeline will need to be converted from the .gen to a .vcf. In addition, imputation QC data needs to be calculated in order to filter out the poorly imputed variants. Therefore, this section will require the newest Plink 2.0 in order to utilize its dosage processing capabilities and convert to a dosage VCF, BCFTools to concatenate the dosage .VCF’s, and QCTool2 to analyze the .GEN files and give us INFO scores (the Impute4 QC metric).

1. **Special Note:** As of this writing, Plink 2.0 is unable to merge or concatenate VCF files, which is why BCFTools is being utilized. In the future, this pipeline will be simplified to use only Plink 2.0 in order merge the VCF’s.
2. **A Note on GNU-Parallel:** Concatenation and running QCTool2 in serial (one after another) takes a considerable amount of time. Therefore, this script offers the capabilities of GNU-Parallel, a free-to use (but citation required) tool that decreases the time to completion of these tasks exponentially by running the commands in parallel. This option is more advanced as you are required to setup GNU-Parallel on your system yourself, but the benefits of doing so, in my opinion are well worth it. For simplicity, we will not run GNU-Parallel in this tutorial, but know that it is an option for those in need of speed.

### Prerequisites:

* Plink 2.0
* QCTool2
* BCFTools
* Imputed data from Step 3

### Concatenation and Conversion Steps:

1. These hodgepodge of Plink, BCFTools, and QCTool2 commands works with the imputed data created in the previous section. All we need to do is set the following variables in the “**3b\_ConcatConvert Variables**” section of Config.conf:
   1. **ConcatImpute** is a toggle that allows you to switch on and off whether the Impute2/4 created .GEN files are concatenated to a single chromosomal .GEN file
   2. **AnalyzeINFO** is a toggle that allows you to create a SNP Report using QCTool2 which allows you to populate the Imputation QC INFO metric. Note that you will need to run this command in order to filter the VCF file in a later step
   3. **FilterINFO** allows you to set an INFO score cutoff. Any imputed variant that is less than this cutoff will be filtered out of the dataset. You must specify a value here even if it “0” (zero) which will result in no filtering of the data.
   4. **ConcatParallel** and **AnalyzeINFOParallel** are variables that invoke GNU-Parallel to execute the concatenation (see **ConcatImpute**) and QCTool report SNP statistic (see **AnalyzeINFO**) commands. For the sake of this tutorial and simplicity, leave these Boolean variables to their default of “F.”
   5. Convert2VCF is a toggle that allows you to convert the concatenated chromosomal .GEN file to a dosage VCF through Plink 2.0. During this step Plink also filters the VCF by the INFO score cutoff you specified (see **FilterINFO**)
   6. MergeVCF is a toggle that allows you to concatenate the dosage VCF files into a single dosage VCF via BCFTools.
2. After executing this script on its default I’m left with several files per chromosome in the ConcatImputation Folder within the Impute Project Directory:
   1. .GEN (concatenated chromosomal .gen file)
   2. .log + .pgen + .psam + .pvar (Plink dosage file)
   3. .snpstat (QCTool2 snp report that contains several metrics on the imputed chromosome including the INFO score)
   4. .snpstatOut (is a log file for QCTool2 which contains the run results from QCTool AND a count of the total number of variants imputed for the particular chromosome and how many are left after INFO filtering)
   5. .list (a file that contains the variants within the chromosome specified that meet the INFO score requirements specified in **FilterINFO**
3. Most importantly I also have a 1DONE\_[BaseName].vcf.gz file which is the final product of the Odyssey pipeline. This dosage VCF file can be inputted into analysis programs such as Plink 2.0, SNPTEST, GenAble, etc. for all sorts of analyzes