Odyssey Tutorial

Written by Ryan Eller – Updated January 2019

# 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 located in the Tutorial\_Data folder. Average completion times range from 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 run independently of the main workflow and thus some rudimentary coding may be needed. These sub-modules are provided as an optional first step (as cleaned data is already provided for you within the tutorial folder). **You may skip these optional sub-modules, proceed to ‘Step 1: Running Quality Control Prior to Phasing/Imputation,’ and use the already processed HGDP tutorial data found in the tutorial data folder**. In the folder, 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 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 scripts are fully annotated so every line of code should have an explanation of what it does.

# 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 for some of the more advanced Odyssey operations (e.g. running GNU-Parallel). 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.

# Odyssey “From Raw to Results” Tutorial:

## Setting Up Odyssey’s Settings.conf File:

Odyssey allows for a large amount user control, most of which can be specified from a file called Settings.conf. 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. Other options allow you to fine tune the analysis by setting user cutoff criteria instead of sticking with the defaults. Settings.conf is divided into sections that range from configuring Odyssey to allowing HPC integration to controlling particular steps in the workflow. Two sections of Settings.conf must be setup prior to running any steps of Odyssey.

1. The Configuration Variables Section
   1. Specify either “One” or “Two” depending on how you configured Odyssey (See Odyssey Installation Instructions)
2. The Core Variables Section
   1. Specify the full directory in which you placed Odyssey. For example, if the Odyssey folder is placed in my home directory I would write:
      1. WorkingDir=”/home/Odyssey-2.0.0/”;
      2. **Note**: Remember to put the trailing ‘/’ on the directory path so that Odyssey know it is a directory and not a file
   2. If you are using Odyssey on a High Performance Cluster/System then set HPS\_Submit to “T”
      1. If your HPS supports sending emails on job starts/endings you can also specify your email under the ‘Email’ variables
      2. Lustre Stripping is important when handling large files on distributed filesystems. If your HPS supports Lustre stripping, it is recommended you leave the LustreStrip variable to “T”

**NOTE:** Every script in Odyssey (aside from the submodules) must be run from the Odyssey working directory. In other words, set your working directory to ./Odyssey-[#.#.#]/ and then begin executing Odyssey’s main scripts.

**NOTE:** If you do not need to clean your data (although it is recommended to prevent imputation errors downstream), you may skip to ‘Step 1: Running Quality Control Prior to Phasing/Imputation’

## Pre-Step A: Getting the Tutorial Data and Converting/Cleaning It [currently being revamped for improved performance/bugs]

### 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:

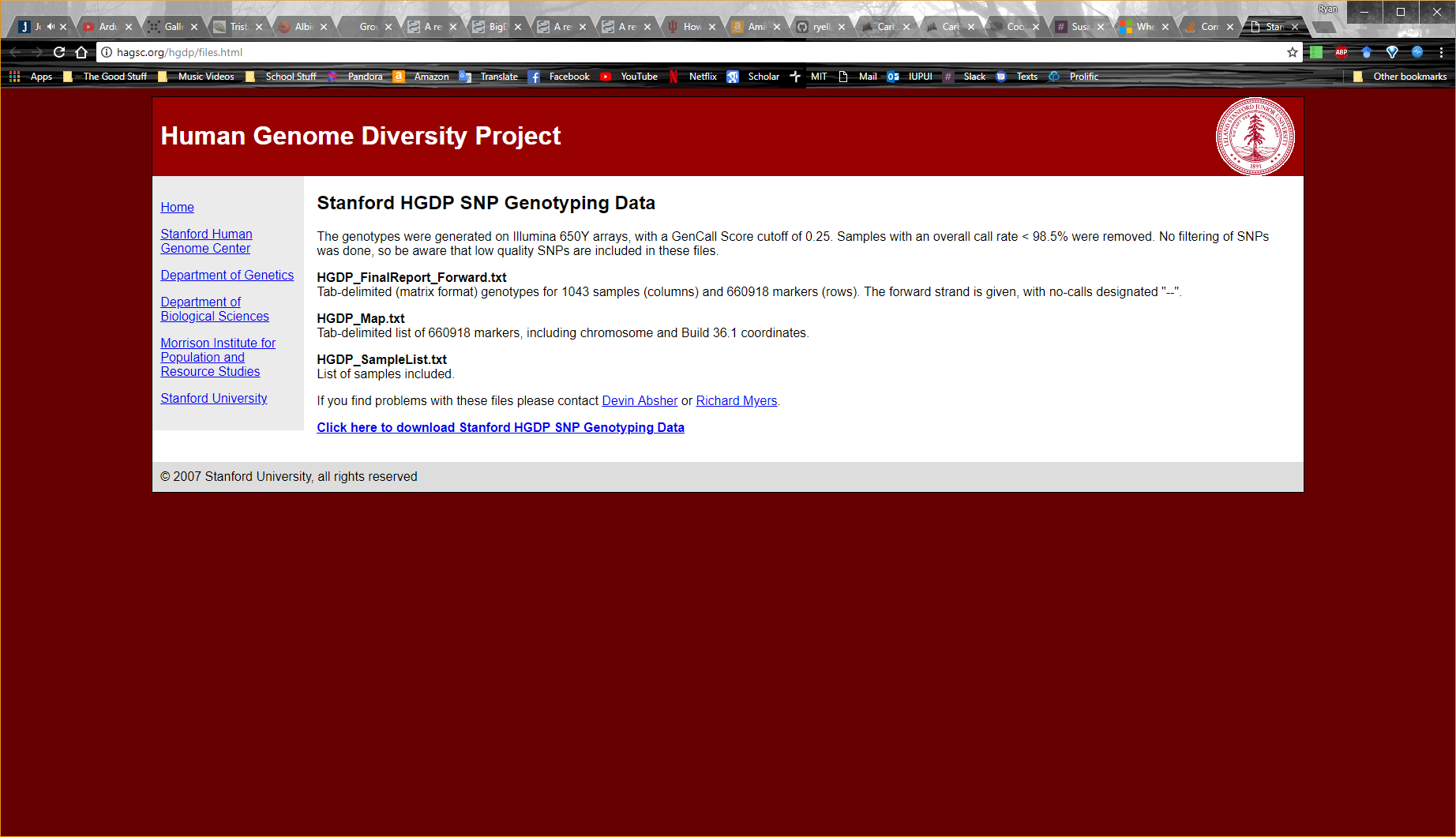
In this tutorial we will be using 100 European (Italian and French) individuals from the HGDP dataset. The dataset contains approx. 661,000 genome-wide markers (mapped to NCBI36 or hg18). A list of HGDP IDs used in this tutorial are defined (see the HGDP\_Euros tab within the HGDP\_StarterPack.xlsm) 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. However, Odyssey is flexible and can be run on any dataset provided it corresponds to its corresponding reference dataset and configured properly. 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.

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

**Note: The .tfam document (which complements the tped file created in the provided script) is already provided for convenience. However, the steps to create this are outlined in 00\_**HGDP\_StarterPack\_Technical\_Info.xlsm.

#### 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 and thus is optimized for this tutorial):
   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/0\_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. Set the working directory to the following:

$ cd ./Odyssey-[#.#.#]/

1. Once the 3 Stanford data files are placed within the 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 a terminal, navigate to the script (via “cd” commands), and enter the following:

$ sh ./0\_DataPrepModule/HGDP\_Starter-SubModule/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 one method is not better than the others, I found 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 literature (<https://www.ncbi.nlm.nih.gov/genome/tools/remap/docs/api>) explaining how to set it up.

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

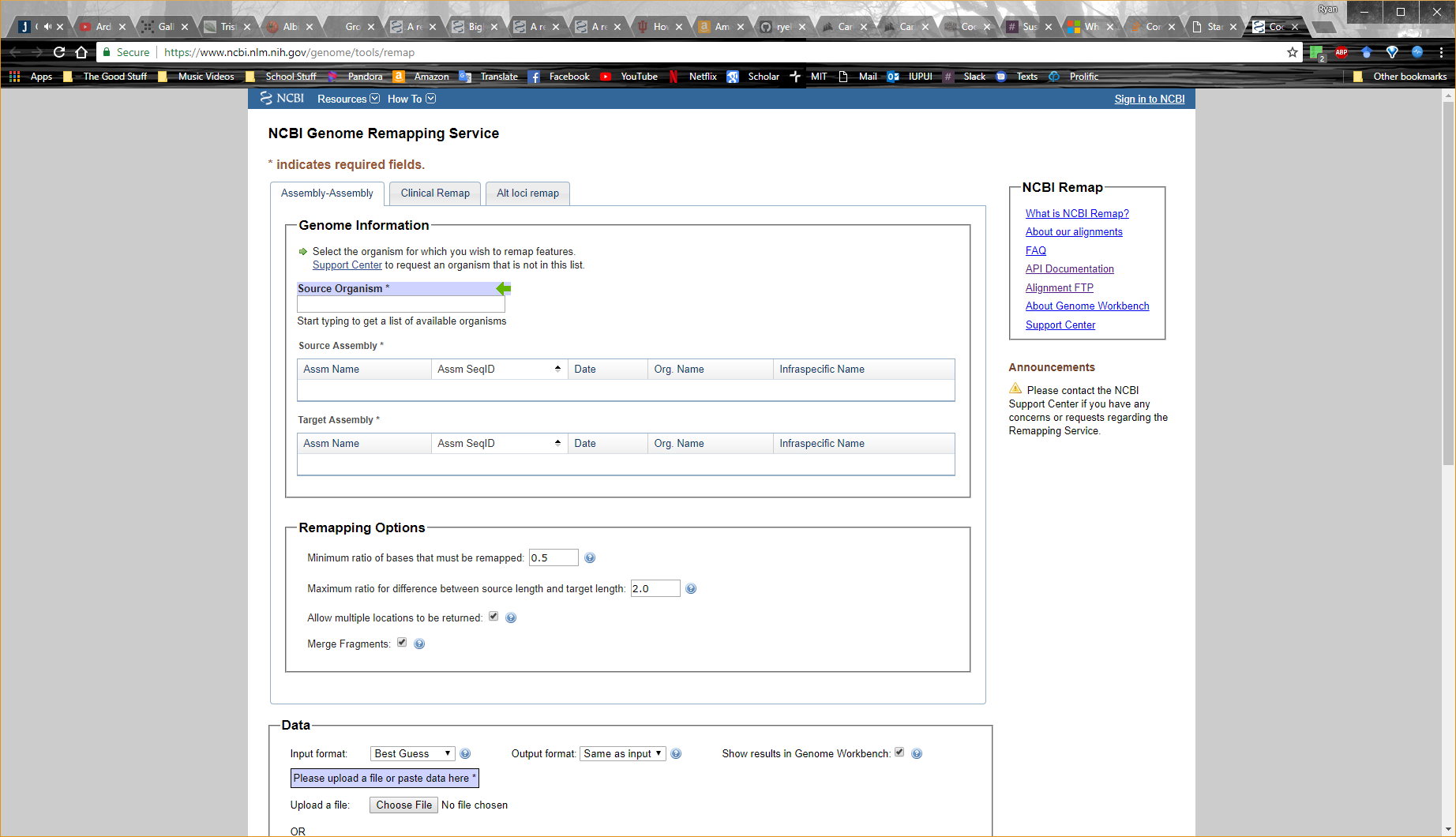
1. As this is a continuation of the HGDP\_Starter-SubModule, you should already have the “HGDP\_Euro” bed/bim/fam files (in addition to .hh and .log files), but know that the following method should work with ANY properly formatted Plink bed/bim/fam files. Move the bed/bim/fam Plink files into the Remapping\_Made\_Easy-SubModule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

cd ./Odyssey-[#.#.#]/

1. Run the following script:

$ sh ./0\_DataPrepModule/Remapping\_Made\_Easy-SubModule/0a\_ChangeChrNameScheme-Convert2BED.sh

* 1. 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.
  2. **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).

1. 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 [currently being revamped for improved performance/bugs]

### 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:

* 0\_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>). We will also need to download reference data from the 1000 Genome Project and NCBI.

**Note:** This step is optional but recommended since it helps to clean up datasets for imputation thereby reducing the chances of downstream imputation errors caused by ‘dirty’ data.

### Data Prep Module Steps:

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

**./0\_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 **Settings.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. **PerformFixref:** Gives you the option of aligning the data to a reference genome in addition to removing positional duplicates. Removing positional duplicates will always be performed. This is recommended to avoid errors although BCFTools +fixref is a little conservative with excluding genotypes. If you are willing to take the risk of imputing with slightly dirty data (data that doesn’t match to a reference build) you may put this option to “F”
   2. **DownloadRef:** Downloads reference annotations to the RefAnnotationData folder, and extracts the 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>
      3. **NOTE:** You only need to perform this step once (if performing multiple data cleanups)
   3. **DataPrepStep1:** 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)
   4. **DataPrepStep2:** Aligns your BCF file to the reference annotation and fixes coordinates errors and allele mismatches with BCFtools
   5. **DataPrepStep3:** Sorts the reference aligned BCF output and convert it back into Plink format for Phasing
2. **DataVisualization:** This option allows you to visualize your data up to this point. It performs a salvo of quality control diagnostics including missingness, heterozygosity, and sample relatedness (via IBD). Setting this option to ‘T’ will start an interactive R session in which you can set QC cutoffs, visualize your data, save graphical output, and save lists of individuals who should be removed from imputation or downstream analysis based on the quality control cutoffs you specify.
3. **X11:** The R script is designed to be interactive and output graphs in real time to an X11 display (see Enabling Visualizations on a HPC.docx in the Readme folder). However, if you don’t have X11 installed (or don’t want to use it), you may turn it off by setting this variable to ‘F’
4. Refer to the “**Imputation Project Variables**” within **Settings.conf** and fill out the following bolded variables.
   1. **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 without the files jumbling together.
      1. Here I will name my project “HGDP\_EuroF” since I want all future folders labeled as such
5. Run **0\_DataPrepCleanup.sh** from
6. In my runthrough of the HGDP dataset I set **PerformFixref** = ‘T’, thus the dataset’s indels and non-biallelic SNP’s were dropped as Plink 1.9 is unable to code VCF files of multiallelic variants. This, will be rectified in Plink 2.0 although, imputation will likely make up for the variants that are lost in this step
7. Cleaned Plink data will be prefixed by “**DataFixStep3**”
8. After executing the following code:

[username ./Odyssey]$sh 0\_DataPrepCleanup.sh

I received DataFixStep3\_HGDP\_EuroF-RefFixSorted.bed/bim/fam in the 1\_Target 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:

* 1\_Target Folder

### Overview:

Cleaned data will now be further cleaned (there’s a lot of data cleaning) by filtering for minor allele frequency, Hardy-Weinberg equilibrium, and genotype and individual missingness. If you want to skip the previous cleanup steps you can find cleaned tutorial data in the Tutorial\_Data folder.

### Prerequisites:

* 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)

### WORKFLOW WARNINGS:

* If the user plans on phasing/imputing the X chromosome all samples must have a sex assigned on the input Plink .fam file. Individuals who are missing sex data will prevent the rest of the dataset (even if those people have a sex assigned) from being X-chromosome phased and X-chromosome imputed.
* Sample names must NOT contain whitespaces nor hard spaces (e.g. “\_”). For maximum compatibility it is recommended that the FID and IID both be numbers, however a continuous string of alpha-numerical characters (i.e. no special characters) is also supported

### 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:

**./1\_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.
  2. **NOTE:** In order to phase/impute the X chromosome, ALL SAMPLES MUST CONTAIN A VALID PLINK SEX CODE IN THE .FAM FILE. FAILURE TO INCLUDE A SEX CODE FOR EVERY SAMPLE WILL RESULT IN AN ERROR FOR X IMPUTATION

1. Refer to the “**Imputation Project Variables**” and “**1\_ImputeProjectSetup-QC-Split Script Variables**” sections within **Settings.conf** and fill out the following bolded variables.
   1. **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 without the files jumbling together.
      1. Here I will name my project “HGDP\_EuroF” since I want all future folders labeled as such
   2. **DownloadDefaultRefPanel:** This option allows you to download the default hg19 Reference Panel from the Impute2 Website to the Reference folder
      1. **NOTE:** You only need to perform this step once per Odyssey setup (unless you want to switch reference panels
   3. **GenoQC:** Excludesgenotypes missing in more than X% of samples (plink –geno command)
   4. **MindQC:** Excludes individuals who are missing more than X% of their genotypes (plink –mind command)
   5. **HweQC:** Sets the cutoff criteria for hardy-weinberg equilibrium (plink –hwe command)
   6. **MafQC:** Sets the cutoff criteria for minor allele frequency (plink -maf command)
   7. After making sure I only had 1 Plink dataset (a .bed, .bim, and .fam) in the **PLACE\_NEW\_PROJECT\_TARGET\_DATA\_HERE** folder I executed the script via the following command:

[username ./Odyssey]$sh 1\_ImputeProjectSetup-QC-Split.sh

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 2\_Phase 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).

### Special Note:

This script is the first time the HPC Configuration section is seen in Settings.conf. 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. Here resources requests can be configured for the phasing script. If you are not running on a HPC you can ignore this section.

### Prerequisites:

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

### Phasing Steps:

1. You will need to download a reference panel for Phasing/Imputation if you haven’t already. I used the 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 and downloaded via the DownloadDefaultRefPanel=’T’ option in the previous step)
   3. 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 Settings.conf or in the naming of the reference files themselves. Odyssey automatically looks for keywords to identify the genetic map, legend, and haplotype files. Settings.conf explains more about how to adjust these variables so I will defer further explanation to it.
2. 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 Settings.conf:
   1. **PhaseThreads:** lets us tell Shapeit2 how many computing threads we intend to use
   2. **Phase\_Memory**: Allows us to specify how much memory we want to request of the HPC per phasing job. This option may be ignored if not working with an HPC.
   3. **Phase\_Walltime**: Allows us to specify how much time we want to request of the HPC per phasing job. This option may be ignored if not working with an HPC.
   4. **XChromIdentifier**: Keyword Odyssey uses to identify the X chromosome in the Reference panel. Normally this should be set to either ‘Chr23’ or ‘X’
   5. **PhaseChrStart:** Specifies the AUTOSOMAL chromosome you want to start phasing
   6. **PhaseChrEnd:** Specifies the AUTOSOMAL chromosome you want to stop phasing
   7. **PhaseX** is a toggle that allows you to specifically control whether the X chromosome is phased
   8. **ExecutePhasingScripts** is a toggle for script submission (HPC or otherwise). **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 run all your phasing scripts, specify “T”. Or you can submit them individually from the script folder in which they are stored. If set to “T”, this command executes a “qsub” command for HPC users and a “sh” command for desktop users. All created scripts are housed in the “**Scripts2Shapeit**” folder within the Phase Project Directory.
3. After executing this script for all my phasing runs using the following code:

[username ./Odyssey]$sh 2\_PhasingScriptMaker

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).

### Prerequisites:

* A Reference Panel (same as in the Phase section)
* Phased data from Step 2

### Imputation Steps:

1. Impute4 works with the phased data created in the previous section. Here we must set the following variables in the “**3a\_ImputeScriptMaker Variables**” section of Settings.conf:
   1. **PhasingErrorAnalysis** should be run first all by itself (without any of the other options set to TRUE (“T”). This is a troubleshooter that will let you know whether any of the phasing steps encountered an error. More specifically, *Odyssey* looks for error messages in the phasing output files. Currently Odyssey flags those segments that contain either the word error or the word segmentation. These flagged files either need to be re-run manually or modified depending on the error. Please refer to the Shapeit manual for further troubleshooting on phasing jobs using Shapeit.
   2. **Impute\_Memory**: Allows us to specify how much memory we want to request of the HPC per imputation job. This option may be ignored if not working with an HPC.
   3. **Impute\_Walltime**: Allows us to specify how much time we want to request of the HPC per imputation job. This option may be ignored if not working with an HPC.
   4. **ImputeAutosomes:** A toggle that allows you to specifically control whether the autosomal chromosomes are imputed
   5. **ImputeX**: A toggle that allows you to specifically control whether the X chromosome is imputed
   6. **ImputeChrStart:** Used to specify the start of the range of AUTOSOMAL chromosomes to be imputed. This may be important if you don’t want to impute all the chromosomes at the same time (e.g. 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 human’s, chromosome 1 has 49 chunks alone).
   7. **ImputeChrEnd:** Used to specify the end of the range of AUTOSOMAL chromosomes to be imputed. For example, setting **ImputeChrStart** to “1” and **ImputeChrEnd** to “10” will impute half the dataset. Then once those are completed you can set the start and end to “11” and “23” respectively.
   8. **ExecuteImputationScripts** is a toggle for script submission (HPC or otherwise). **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 run all your imputation scripts, specify “T”. Or you can submit them individually from the script folder in which they are stored. If set to “T”, this command executes a “qsub” command for HPC users and a “sh” command for desktop users. All created scripts are housed in the “**Scripts2Shapeit**” folder within the Impute Project Directory.
2. After executing this script by running the following code:

[username ./Odyssey]$sh 3a\_ImputeScriptMaker

Using 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-Impute4 pipeline will need to be converted from a .gen to a .vcf. Furthermore, imputed data needs to be analyzed via SNPTEST in order to filter out poorly imputed variants using the imputation INFO score QC metric. Plink 2.0 is used to process the .gen and convert it into a dosage VCF, which BCFTools will merge to a final dosage .VCF.

**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.

**A Note on GNU-Parallel:** Concatenation and running SNPTEST 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 the host 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:

* Imputed data from Step 3

### Concatenation and Conversion Steps:

1. All imputed data manipulation commands work with the imputed data created in the previous section. Here we must configure the following variables in the “**3b\_ConcatConvert Variables**” section of Settings.conf:
   1. **ImputationErrorAnalysis** should be run first all by itself (without any of the other options set to TRUE (“T”). This is a troubleshooter that will let you know whether any of the imputation steps encountered an error. More specifically, *Odyssey* looks for error messages in the imputation output files. Currently Odyssey flags those segments that contain either the word error (which is sometimes caused by no variants being imputed in the segment – so not really an error) or the word segmentation (which is normally an error). These flagged files either need to be re-run or manually manipulated to fix the error. I will defer to the Impute program’s website for troubleshooting imputation errors.
   2. **ConcatImpute:** A toggle that allows you to switch on and off whether the Impute4 created .GEN files are concatenated into single chromosomal .GEN files
      1. **ConcatStart** & **ConcatEnd:** Allows for manual control of which chromosomes are concatenated (e.g. ConcatStart = 1 and ConcatEnd=2 will concatenate chromosomes 1 through 2).
   3. **AnalyzeINFO:** A toggle that allows you to create a SNP Report using SNPTEST, which allows you to populate the Imputation QC INFO metric needed in a later QC step.
      1. **INFOThresh**: Sets the INFO score (i.e. Imputation QC) cutoff. For example, setting INFOThresh= 0.3 will remove imputed variants < 0.3. You must specify a value here even if it “0” (zero) which will result in no filtering of the data.
   4. **FilterINFO**: A togglethat is a continuation of AnalyzeINFO and populates a SNP list based on the criteria you set in INFOThrsh. This list will be used in a future QC step.
      1. **INFOStart** & **INFOEnd**: Allows for manual control of the chromosomes whose info scores are being evaluated
   5. **Convert2VCF:** A togglethat allows you to convert the concatenated chromosomal .GEN file to a dosage VCF through Plink 2.0. During this step Plink filters the VCF by the INFO score filtered SNP list created in the previous **FilterINFO** command
      1. **VCFConvertStart** & **VCFConvertEnd**: Allows for manual control of the chromosomes to be converted into dosage VCF’s
   6. **MergeVCF:** A toggle that allows you to concatenate the dosage VCF files created previously into a single dosage VCF via BCFTools.
      1. **ConcatThreads**: Allows BCFTools to utilize more computing threads when merging the dosage VCFS. More threads will make the final dosage VCF file faster
   7. **Keep Temp**: An option that allows you to cleanup temporary files upon script termination. This option is helpful for troubleshooting if something erred out.
   8. **ConcatParallel**, **AnalyzeINFOParallel, FilterINFOParallel,** and **ConvertParallel** are variables that invoke GNU-Parallel to execute the concatenation, post-imputation QC, conversion, and merging steps listed above. If set to “T” this will invoke GNU-Parallel’s “parallel” after running the LOAD\_PARALLEL variable option in the ‘Parallel Processing Setup Option’ section located in the ./Configuration/Setup/ Programs\*\*.conf file. A more in-depth explanation can be found in the Programs\*\*.conf file as this tutorial will focus on simplicity and leave the Boolean parallel variables to their default of “F.”
2. This script may either be run interactively on the terminal (e.g. for desktop users)

[username ./Odyssey]$ sh 3b\_ConcatConvert.sh

Or it may be run non-interactively, for HPC users, by specifying a qsub command using the familiar PBS flags. However, for the script to be set to the proper working directory the -d flag must be set to the Odyssey working directory (e.g. ./Odyssey/). Since, the working directory should already be ./Odyssey/ setting -d to “$PWD” should work in all cases (shown bolded below)

[username ./Odyssey]$ qsub -l nodes=1:ppn=5,vmem=16gb,walltime=05:00:00 -M [email address] -m a -j oe -o [output/directory/Log.out] -N [name of job] -d "$PWD" ./3b\_ConcatConvert.sh

1. After executing the script on its default settings, with KeepTemp=‘T’, I’m left with several files per chromosome in the ConcatImputation Folder:
   1. .GEN (concatenated chromosomal .gen file)
   2. .log + .pgen + .psam + .pvar (Plink dosage file)
   3. .snpstat (SNPTEST SNP report that contains several metrics on the imputed chromosome including the INFO score)
   4. .snpstatOut (is a log file for SNPTEST which contains the run results from SNPTEST **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**
2. 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 downstream analysis.

## Step 5: Imputation Analysis and Visualization (GWAS)

### Estimated Time to Completion:

* 20 minutes

### Folder Location:

* Data Deposited into GWAS Project Folder

### Overview:

Imputation (GWAS) analysis is separated from the rest of the pipeline in case the user wants to perform a GWAS independently of imputing a dataset. Therefore, while Odyssey provides the option of using an Imputation Project on which to run an analysis, there is also the option of specifying a dosage .VCF (the file must be a dosage VCF accepted by Plink 2.0) and a Plink formatted phenotype file to run an analysis. In addition, the user must specify a GWAS Project name (similar to the Imputation Project’s BaseName), which will create the project directory to house the analysis files. This is designed so that a user may perform multiple analyses on an imputation project’s dosage .VCF (or a specified dosage .VCF) without the files being overwritten.

**Special Note:** The GWAS Project requires a phenotype file, which for this tutorial, is supplied as a dummy phenotype file that randomly assigns a zero through one phenotype to the HGDP individuals. It can be found in ./GWAS/Phenotype/DummyPheno.txt (refer to Plink’s specification on phenotype/covariate files -- <https://www.cog-genomics.org/plink/1.9/input#pheno> )

### Prerequisites:

* Dosage .vcf data from Step 4
* A Phenotype file (a dummy phenotype file is provided in . /GWAS/Phenotype/)

### Imputation Analysis and Visualization Steps:

1. We must configure the variables in the “**GWAS Project Variables**” section of Settings.conf
   1. **GWASRunName:** A variable that names the GWAS Project folder in which all the analysis results will be saved. The name should be indicative of the type of analysis that will be run and must also be a continuous string of characters (i.e. no spaces or special characters, although hardspaces, “\_” are supported).
   2. **VCF2PGEN:** This step only needs to be performed once on a dataset. It will convert the input dosage VCF to a Plink dosage .pgen file. This will allow Plink to run faster if multiple analyses are performed on a single dataset (which is most likely the case)
   3. **Pheno\_File:** A variable that specifies the name of the Plink formatted Phenotype file (see Plink formatting help here -- -- <https://www.cog-genomics.org/plink/1.9/input#pheno>) that should be placed in the ./Odyssey/GWAS/Phenotype/ directory. Again, this filename should not contain spaces or special characters and it also must contain the extension (e.g. “.txt”, “.pheno”)
   4. **GWASPhenoName:** The name of the column in the specified Phenotype file that the user wishes to analyze. Note that the name listed for this variable must match exactly the column name in the phenotype file. Column names within the pheno file must not contain whitespaces
   5. Dosage VCF Input: There are several methods of loading a dosage vcf into the GWAS project. Either specify the imputation project that you want to analyze (in this way the GWAS Project is a continuation of the Imputation Project) or manually specify a dosage VCF (thus the GWAS Project would be performed separately from Imputation Projects)
      1. To specify an Imputation project to analyze fill out the **ImputationProject2Analyze** variable so that it matches the name of the imputation project that created the dosage vcf you wish to analyze (e.g. **ImputationProject2Analyze** should be the same as the **BaseName** variable specified earlier in the Imputation Project Variables Section. Odyssey will then look into the Impute folder under the specified Imputation Project and search for the “1DONE….vcf.gz” file created in the imputation step. This option will automatically lookup the corresponding sample sex information file (i.e. the .fam file) found within the Imputation project’s Target file.
      2. To specify a dosage .vcf not created from an Imputation Project you will need to set **GWASOverride** to “T” (the default is “F”), specify the complete path to the dosage .vcf.gz file you wish to analyze via the **ManualVCFInput** variable, and specify the full path to the Plink formatted sex information .fam file via the **ManualSexInput** variable.
   6. **PLINK\_OPTIONS** is a variable that allows you to specify additional Plink commands to customize the GWAS to your needs. For more information on the types of options you can specify to run various analysis types refer to the Plink user manual (<https://www.cog-genomics.org/plink2>).
   7. **GWAS\_Memory**: Allows us to specify how much memory we want to request of the HPC per GWAS job. This option may be ignored if not working with an HPC.
   8. **GWAS\_Walltime**: Allows us to specify how much time we want to request of the HPC per GWAS job. This option may be ignored if not working with an HPC.
   9. **GWAS\_Threads** is a variable that specifies the maximum number of threads at your disposal that Plink can utilize to run the analyses. In general, the more threads you can afford the faster the analysis.
   10. **ExecuteGWASScripts** is a toggle that allows you to control whether the scripts that are created to run the GWAS and then visualize the output are submitted either to a HPS (if the **HPS\_Submit** variable is set to ‘T’) via a qsub command or to the command line via a sh command (if the **HPS\_Submit** variable is set to ‘F’).
2. This script may either be run by executing the following command:

[username ./Odyssey]$ sh 4\_AutomatePlink.sh

1. Following the termination of the script you will be left with the following files within a GWAS Project folder
   1. [GWAS Analysis Name].[PhenoName].[plink analysis].gz = The raw zipped Plink analysis file
   2. AbbrevResults\_[variants analyzed]\_Top10000Results.csv/txt = An abbreviated list of GWAS results showing the top hits organized by unadjusted p-value (although multiple testing Benjamini-Hochberg and Bonferroni adjustments are also present with the abbreviated table)
   3. \*\*Analyze-Visualize.sh = The script used to perform the Plink analysis and subsequent R visualization
   4. [Phenotype]\_[GWAS Analysis Name]\_Manhattan-Plot.html + 1\_PlotlyDependencies = The interactive Manhattan plot, which plots the top 10,000 results from the AbbrevResults…csv file. The .html must be viewed in the same location as its dependency folder
   5. [Phenotype]\_[GWAS Analysis Name]\_QQ\_Plot.png = A QQ plot constructed via the entire results file (unlike the Manhattan-Plot which is based on the abbreviated results)

# Population Stratification Add-In Tutorial:

## Population Stratification Add-In:

### Estimated Time to Completion:

* 25 minutes

### Folder Location:

* Data Deposited into the PopStratModule Folder

### Overview/Conceptual Implementation:

The Population Stratification add-in helps to analyze and visualize admixed populations in an admixed dataset and allows the user to subset a specified ancestral population that is of interest for analysis. Essentially, 2 datasets are merged, a reference dataset such as the admixed 1000 Genomes that contains individuals of known ancestries, and a target dataset that contains individuals of unknown/unverified ancestries. A PCA is performed on the combined dataset in Plink to reduce the number of dimensions to 20 principal components (PC). The PopStrat add-in also requires a Plink accepted (tab delimited) list of FID’s and IID’s that correspond to individuals in the reference dataset that are of the ancestry or ancestries the user wants to retain for analysis. R is then used to determine the number of PC’s to use for outlier detection. As each PC contributes to the total amount of variance seen in the dataset, the user has the option of specifying which PC’s are used for outlier removal, by specifying the percent of total variance a PC must contribute to be included. For example, the default is to keep PC’s that contribute >1% of total variance (this roughly corresponds to 5-8 PC’s which is around the same amount of PC’s that Eigensoft, a popular population genetics program that also utilizes automatic outlier detection/removal, uses to remove ancestral outliers (with its default being 10 PC’s). Given a variance cutoff, Odyssey automatically analyzes the eigenvectors of the selected PC’s from the individuals listed by the user that contain an ancestry of interest and creates an N-dimensional (depending on the number of PC’s kept) centroid that is deemed the ancestral center of the ancestry of interest. Odyssey also calculates the standard deviation from the eigenvectors of the ancestry of interest individuals. Given a user-specified number of standard deviations, Odyssey creates an N-dimensional area surrounding the ancestry of interest centroid, which forms the criteria of the automatic ancestral outlier removal. Individuals which are deemed outliers and the complete PCA table are output as .txt documents and the unitless centroid position and areas are reported.

**Special Note:** The Population Stratification add-in contains an Excel document, “00\_PopStrat\_Helper,” that may help in organizing and creating the Plink formatted list of FID’s and IID’s which is used to calculate the centroid and corresponding cutoff criteria for the automatic outlier removal. The document already contains the ancestry information for the 1000 Genomes Phase 3 and HGDP datasets, which can be used as reference datasets. Feel free to use this document to help organize the reference and target datasets or you can safely ignore it and populate the list manually.

### Prerequisites:

* A reference dataset (ancestries are known) (1000 Genomes Phase 3 recommended)
* A target dataset (ancestries are unknown/unverified)
* A Plink formatted list of FID’s and IID’s specifying individuals from the reference dataset that are of the ancestry the user wants to keep

### Population Stratification Add-In Steps:

1. The add-in requires several inputs in order to function:
   1. **A Target Dataset**. A plink (bed/bim/fam) file trio that contains individuals of unknown/unverified ancestries should be placed in the following directory: ./Odyssey/PopStratModule/PLACE\_Target-Ref\_Datasets\_HERE
   2. **A Reference Dataset**. A plink (bed/bim/fam) file trio that contains individuals of known ancestries should also be placed in the following directory: ./Odyssey/PopStratModule/PLACE\_Target-Ref\_Datasets\_HERE
      1. Reference sets that are already setup (and thus recommended) for Odyssey are:
         1. 1000 Genomes Phase 3 -- <ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/>
            1. You will need to convert and merge the VCF’s into Plink (bed/bim/fam) files prior to using it in the add-in
         2. HGDP (Stanford) -- <http://hagsc.org/hgdp/files.html>
            1. Using this dataset requires an extensive amount of work since you will need to convert it to a Plink format and remap it to GRCh38. However, following both segments of Pre-Step A in this tutorial will walk you through both of these tasks.
   3. **A Comma-Separated-Value (CSV) List of Reference Individual FID’s and IID’s.** These individuals will be used to set the criteria for the automatic outlier removal. For instance, if I want to exclude all individuals in my Target dataset who are NOT ancestrally European, I would populate my CSV list with individuals from the reference dataset who have known European ancestries. That list should be placed in the following directory: ./Odyssey/PopStratModule/PCA\_Analyses/[PCA\_Analysis\_Name]
      1. **NOTE**: The directory you need to put the CSV document in does not exist until after Step 1 due to the assumption that not all PCA analyses will exclude the same ancestral groups. Therefore, you will need to complete at least up through Step 1 prior to depositing the CSV.
      2. **ADDITIONAL NOTE**: The name of the CSV that is placed in the PCA Analysis Folder is irrelevant as long as it is the only .csv document in the folder and that it DOES NOT contain any spaces or special characters
2. The add-in works similar to the main Odyssey pipeline in that it utilizes a .conf file to control the variables and steps of the add-in. Therefore, you will need to fill out PopStratConfig.conf
   1. **WorkingDir** is the same variable that is used by Config.conf for Odyssey. Simply specify the complete path to the Odyssey Folder. Make sure to include a ‘/’ on the end of the path so that Odyssey knows this in a folder directory.
   2. **RefDataset** is the “Base” name of the Plink reference dataset (.bed/.bim/.fam) so make sure to not include the extension (e.g. Put RefDataset=“refdataset” and not RefDataset=“refdataset.bed”).
   3. **TargetDataset** is the “Base” name of the Plink target dataset (.bed/.bim/.fam), similar to the RefDataset.
   4. **PCA\_Analysis\_Name** is the name of the PCA analysis (and outlier removal) you are going to perform. This will also be the name of the folder you will need to put the CSV of Reference individuals that will be used for outlier detection in.
   5. **PC\_VariancePerc** specifies the cutoff criteria for the principal components that will be used to determine the centroid dimensions for outlier removal and detection. Principal components that contribute more than the percent (written as a decimal) listed in this variable will be included in the analysis. The default is 1% (0.01) and should be enough for most uses.
   6. **PC\_StandardDev** specifies the area around the centroid that is considered within the limits of the specified ancestry. A larger number of standard deviations makes the outlier removal less strict. For little to no outlier removal specify a very large number such as 10. The default is 3.
   7. **Max\_Memory** specifies the maximum allottable memory Plink can use for the analysis. This may be helpful when working on systems in which memory is strictly monitored (e.g. on HPS systems where using more memory than requested results in a killed job)
   8. **PlinkExec** specifies the path to the Plink exec similar to how Odyssey’s Programs.conf functions. Give the full path to the plink executable file
   9. **R\_Rscript** specifies the path to Rscript within the R “bin” directory. This essentially calls R from terminal. Give the full path to the Rscript executable file
   10. **VariantExtract** is a toggle that performs Step 1, which is the first step in merging the reference and target datasets (specifically extracting the variant names from the 2 .bim files to find commonalities)
   11. **AttemptMerger** is a toggle to perform Step 2, which attempts to merge the 2 datasets outright. However, the chances of this merge succeeding is slim for a bunch of reasons which is why…
   12. **FlipTarget** is provided, which troubleshoots merging the 2 datasets by flipping the alleles that failed to merge on the target dataset so that they (hopefully) now match the reference dataset. If the merge fails again, then the variants that failed to merge are removed (ideally there shouldn’t be many – but it’s still good to check the logs) and the merge is repeated.
   13. **PrepData** to perform a PCA not all variants are needed, so this variable is provided which toggles the step to remove/subset variants that are in high LD. This creates a pruned dataset of variants in low LD which is used for the PCA.
       1. The Plink command is specifically: --indep-pairwise 1500 150 0.4
   14. **Perform\_PCA** is a toggle that performs a Plink PCA on the pruned dataset of low LD variants.
   15. **Analyze\_PCA** is a toggle that analyzes the output of the PCA in R. IMPORTANTLY, make sure the .csv is added to the user specified PCA Analysis folder within “PCA\_Analyses” prior to running this step.
   16. **Cleanup** is a toggle that removes the temporary folder for the add-in saving space. Keeping the temporary files may assist is troubleshooting, but by default this option is set to ‘T.’
3. After performing all 6 steps the final output should be a PNG that illustrates individuals kept for analysis, the reference individuals used to determine the ancestral outliers, and the outliers that were removed on the top 2 principal components. In addition, two text documents will be output which contains the list of outliers (AncestryOutlier.txt) as well as the PCA table used to create the PNG (PCA\_CompleteTable.txt). Centroid statistics are printed to standard out on the screen.
4. The outlier text document can therefore, be used in subsequent analyses in the GWAS step of the main Odyssey pipeline to remove admixed individuals yielding an analysis dataset from a single population.
   1. This can be implemented by using the “--remove” tag in Plink followed by a path to AncestryOutliers.txt