# Compound Heterozygous Variant Pipeline

**Dustin Miller**

**11/26/2019**

# Compound Heterozygous Variant Pipeline Example

This pipeline example uses data from the Gabriella Miller Kids First Data Resource Center (KFDRC). The files in the KFDRC are controlled access so no example files can be provided. However, this guide will show you the steps we used to obtain phased data that has been queried for compound heterozygous variants.

All Dockerfile's and python scripts are available at github.com/dmiller903/ch-pipeline. Clone this repository on the system where you be processing your data. Place this file where it can be accessed by your docker container.

The steps of this pipeline assume that you have already installed Docker on the system where you will be processing your data.

## Consolidate manifest, biospecimen, and clinical files

Each of these files contain various information about the sample files. Using pieces from each of these files is necessary to determine family relationships, gender, and disease status. The consolidation of these files results in a tsv file with 5 columns: file_name, family_id, sample_id, proband (Yes, no), and sex (1 = male, 2 = female). This file is used repeatedly throughout the pipeline as an easy way to keep track of file names and family relationships, and is eventually used to create .fam files which are necessary for certain programs used in the pipeline.

First, it is necessary to pull a python image from Docker. This is only step where an image will be pulled instead of built.

```
docker pull python:3.8-rc-slim-buster
```

Next, execute the “kids_first_meta.py” script in a docker container using the python image that was pulled. You must attach a volume to the docker container. This volume is where the container will look for files and be able to save files to. In this example, our directory, “/Data/KidsFirst” is were the ch-pipeline repository was cloned to and within this directory is where the gVCF files from the KFDRC were downloaded. Within the container the “/Data/KidsFirst” directory is known as “/proj” as specified by “-v” and is set as the working directory with “-w”. “-t” allows the container to use a terminal and “python:3.8-rc-slim-buster” is the image that was previously pulled. The python script used for this step needs 3 input files: the manifest file, biospecimen file, and the clinical file. These files were downloaded from the KFDRC for the Idiopathic Scoliosis trio data and were placed in the path shown below. The name of the output file also needs to be specified.

```
docker run -v /Data/KidsFirst:/proj -w /proj -t python:3.8-rc-slim-buster \ #attached a volume called "proj" in the container and set as working directory
  python3 ch-pipeline/scripts/kids_first_meta.py \
  idiopathic_scoliosis/gVCF/kidsfirst-participant-family-manifest_2019-09-23.tsv \ #manifest file
  idiopathic_scoliosis/gVCF/participants_biospecimen_20190923.tsv \ #biospecimen file
  idiopathic_scoliosis/gVCF/participant_clinical_20190923.tsv \ #clincal file
  idiopathic_scoliosis/gVCF/consolidated.tsv #output file name
```

# Parse and rename files

gVCF files are different from VCF files in that they contain information for every position, including non-variant positions. Therefore, these files are large and would take a long time to process if the non-variant positions were included throughout the whole compound heterozygous pipeline.

This step uses the “consolidated.tsv” file to create a folder for each family ID and then within each of these family ID folders it creates a folder for each sample ID. It then takes each proband file and removes all non-variant sites. It outputs the parsed file to its corresponding sample ID folder. The script then filters each parent file for sites that only occur in the proband of that family.

To build an image, the “docker build” command requires that you be in the folder where the Dockerfile is located that is necessary to build the image. To build the image, use the “docker build” command below. “-t” is used to name the image, in this case, the image is being named “parse-pipeline”. The “.” tells Docker to use the Dockerfile that is in the directory.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/parse-docker
docker build -t parse-pipeline .
```

The “parse.py” script is needed for this step and takes 3 arguments: the “consolidated.tsv” file, the path where the original files downloaded from the KFDRC were saved, and the max number of cores to use. The difference with the “docker run” command used for this and subsequent steps is that we use the “-d” option. This option allows the container to run in the background and will save a log where you direct it to with “>” as seen below.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t parse-pipeline \
  python3 ch-pipeline/scripts/parse.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \ #consolidated tsv file
  idiopathic_scoliosis/gVCF \ #path were KFDRC files were downloaded
  42 \ #max number of cores to use
  > idiopathic_scoliosis/gVCF/parse.out #save docker log
```

# Combine each trio into a single file

During this step, the parsed gVCF file of each individual will be used to create a combined trio file for each family. In addition, a .fam file will be created for each trio.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/combine-trios-docker
docker build -t combine-trios .
```

The “combine_trios.py” script is needed for this step and takes 3 arguments: the “consolidated.tsv” file, the path where the original files downloaded from the KFDRC were saved and the max number of cores to use.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t combine-trios \
  python3 ch-pipeline/scripts/combine_trios.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  42 \
  > idiopathic_scoliosis/gVCF/combine_trios.out
```

# Liftover trio files and individual files from GRCh38 to GRCh37

The files used in the KFDRC are aligned to GRCh38. This is an issue as phasing programs and GEMINI need the files to be aligned to GRCh37. Therefore, this step takes all the combined trio files and the individual-parsed files and converts their positions to GRCh37 positions. At this point and up until phasing, the scripts will always use the trio files and individual-parsed files that were used to make the trio files. The individual files are still being used and processed so that the user can choose to phase an individual if wanted or potentially combine all individuals into one VCF prior to phasing. However, phasing individual files or combining all files into one VCF to be phased is not implemented currently.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/liftover-docker
docker build -t liftover-pipeline .
```

The “liftover.py” script is needed for this step and takes 2 arguments: the “consolidated.tsv” file and the path where the original files downloaded from the KFDRC were saved.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t liftover-pipeline \
  python3 ch-pipeline/scripts/liftover.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  > idiopathic_scoliosis/gVCF/liftover.out
```

# Remove unplaced, and multiallelic sites and duplicate sites from lifted files

During liftover, some randomly placed sites are included in the VCF file. These randomly placed sites are those that are in GRCh38 but the exact position in GRCh37 isn’t known. Therefore, for subsequent analysis, these sites are removed. Only sites with known positions, on a known chromosome are kept. In addition, positions that are multiallelic or are duplicates are removed because programs such as PLINK (used next) and SHAPEIT2 (used later) can not handle these types of sites. For trios, sites that contain any missing genotype information (i.e. “./.”) are removed to improve phasing accuracy.

This step does not require an additional image if all previous steps above have been followed. Use the parse-pipeline image explained above.

The “remove_unplaced_multiallelic.py” script is needed for this step and takes 2 arguments: the “consolidated.tsv” file and the path where the original files downloaded from the KFDRC were saved.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t parse-pipeline \
  python3 ch-pipeline/scripts/remove_unplaced_multiallelic.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  > idiopathic_scoliosis/gVCF/remove_unplaced_multiallelic.out
```

# Separate each trio and individual file into chromosome files, then generate plink files for trio's only

SHAPEIT2 requires that chromosomes be phased separately. PLINK files are also needed by SHAPEIT2 in order to phase. Therefore, this script separates each trio and individual file into chromosome files. This step also generates the necessary PLINK files needed for phasing.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/sep-chr-create-plink-docker
docker build -t sep-chr-plink .
```

The "separate_chr_generate_plink.py" script is needed for this step and takes 3 arguments: the "consolidated.tsv" file, the path where the original files downloaded from the KFDRC were saved, and the max number of cores to use.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t sep-chr-plink \
  python3 ch-pipeline/scripts/separate_chr_generate_plink.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  44 \
  > idiopathic_scoliosis/gVCF/separate_chr_generate_plink.out
```

# Phase each of the trios with a haplotype reference panel

During this step, each chromosome PLINK file of each trio is phased using SHAPEIT2. The parameters for phasing are set so that SHAPEIT2 uses family relationship genotype information and also uses a haplotype reference panel.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/phase-docker
docker build -t phase-pipeline .
```

The "phase_with_shapeit_individual_trio.py" script is needed for this step and takes 2 arguments: the "consolidated.tsv" file and the path where the original files downloaded from the KFDRC were saved.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t phase-pipeline \
  python3 ch-pipeline/scripts/phase_with_shapeit_individual_trio.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  > idiopathic_scoliosis/gVCF/phase_with_shapeit_individual_trio.out
```

# Revert REF/ALT to be congruent with reference panel

The phased results can have the REF and ALT alleles switched as compared to the reference genome. We are unsure exactly why this occurs. It may be that since each trio file only has 3 samples, the ALT allele is more common in the trio and becomes the REF. This step ensures that the REF/ALT alleles of the phased VCF files are congruent with the REF/ALT of the reference genome. In addition, sites with Mendel errors are removed.

This step uses the Docker Image from the previous step, "phase-pipeline".

The "alt_ref_revert.py" script is needed for this step and takes 2 arguments: the "consolidated.tsv" file and the path where the original files downloaded from the KFDRC were saved.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t phase-pipeline \
  python3 ch-pipeline/scripts/alt_ref_revert.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  > idiopathic_scoliosis/gVCF/alt_ref_revert.out
```

# Concat and merge phased trio chromosome files into one VCF file

To make subsequent analysis of the phased files easier, this step concatenates all phased chromosomes for each trio into a single trio file. Then each concatenated trio file is merged into a single file that contains all trios. In addition, a .fam file is created for this single, merged file.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/concat-merge-phased-vcf
docker build -t concat-pipeline .
```

The "concat_merge_phased_vcf.py" script is needed for this step and takes 3 arguments: the "consolidated.tsv" file, the path where the original files downloaded from the KFDRC were saved, and the name of the disease (this will be used in the output file name).

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t concat-pipeline \
  python3 ch-pipeline/scripts/concat_merge_phased_vcf.py \
  idiopathic_scoliosis/gVCF/consolidated.tsv \
  idiopathic_scoliosis/gVCF \
  idiopathic_scoliosis \
  > idiopathic_scoliosis/gVCF/concat_merge_phased_vcf.out
```

# Trim and normalize VCF file

Prior to annotation and loading a database into GEMINI, GEMINI recommends left trimming and normalizing VCF files. Files that are not left trimmed and normalized can be annotated incorrectly and may be incorrectly handled by GEMINI. This step uses VT tools to trim and normalize the phased VCF file.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/vt-docker
docker build -t vt .
```

The “vt_split_trim_left_align.sh” script is needed for this step. Trim and normalize your file using the example in the .sh script. You can use this file, but please change the name of the input file within this script to the name of your input file. Also, change the name of the output file to the name you want.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t vt \
  bash ch-pipeline/scripts/vt_split_trim_left_align.sh \
  > idiopathic_scoliosis/gVCF/vt_split_trim_left_align.out
```

# Annotate with snpEff and query for CH variants with GEMINI

snpEff is used to annotate. Annotation provides information on the effects of variants on known genes. GEMINI allows users to explore genetic variation based on the annotations of the genome.

Using the same logic as previous steps, a docker image needs to be created.

```
cd /Data/KidsFirst/ch-pipeline/docker-images/snpeff-gemini
docker build -t snpeff-gemini .
```

In order to annotate, please use the example provided in “annotate.sh”. If you want to annotate exactly how we did, you can use this file, but please change the name of the input file within this script to the name of your input file. Also, change the name of the output file to the name you want. snpEff has many user options and different ways to annotate so we did not want to define how the user should annotate, so please feel free to use whatever parameters you want for your data.

Once you have your .sh file set up, execute the docker container as shown here.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t snpeff-gemini \
  bash ch-pipeline/scripts/annotate.sh \
  > idiopathic_scoliosis/gVCF/annotate.out
```

In order to load a GEMINI database, please use the example provided in “gemini_load.sh”. The Docker Image provides CADD files for GEMINI to use when loading the database. If you want to load differently and take advantage of other GEMINI loading features, change the parameters in the .sh file. It is important to have a .fam file to load so GEMINI can use family relationships. This file should be available if you have been following every step of this pipeline (created during the concat/merge step).

Once you have your .sh file set up, execute the docker container as shown here.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t snpeff-gemini \
  bash ch-pipeline/scripts/gemini_load.sh \
  > idiopathic_scoliosis/gVCF/gemini_load.out
```

To perform a GEMINI query, see the example in “gemini_query.sh”. There are numerous ways one could query a gemini database. In our example, we query for compound heterozygous variants with (an impact_severity of ‘HIGH’ or is a loss of function variant) or (a variant with an impact_severity of ‘MED’ and has a minor allele frequency <= 0.005 and a CADD score >=20). This example also queries for de novo variants using the same parameters as CH variants. Feel free to use this exact same query but please change the name of the input file within this script to the name of your input file. Also, change the name of the output files to the names you want. See GEMINI’s documentation for additional query options.

Once you have your .sh file set up, execute the docker container as shown here.

```
docker run -d -v /Data/KidsFirst:/proj -w /proj -t snpeff-gemini \
  bash ch-pipeline/scripts/gemini_query.sh \
  > idiopathic_scoliosis/gVCF/gemini_query.out
```