ScaleHD Documentation

Release 0.324.2

Alastair Maxwell

Feb 03, 2022

Noteworthy Features

ScaleHD is a bioinformatics pipeline for use in broad-scope automated genotyping of parallel sequencing data of the *HTT* CAG/CCG repeat which is associated with Huntington Disease (HD) data. Existing software which aim to profile disease-causing nucleotide repeat loci (such as the HD-causing *HTT* CAG repeat) typically function on a generalised level, not focusing on any one particular disease-causing locus. In the case of the *HTT* CAG repeat, this can result in inaccuracies occurring in any genotypes produced, due to a plethora of biological phenomena which are present, complicating matters when it comes to genotyping the *HTT* CAG repeat. This software takes a direct approach to resolving these problems, and doing so in an unsupervised, automated manner.

ScaleHD takes a configuration XML document as input, which contains all required information for the instance of ScaleHD to run to completion. The details of this XML input are specified in *Specifying User Input(s)*. Raw data should be in FastQ format; both forward (_R1) and reverse (_R2) sequence reads are required for ScaleHD to function. ScaleHD will perform quality control, sequence alignment and genotyping on all FastQ file pairs presented by the user as input. If a sample fails to produce a genotype at any given stage, for whatever reason, a debug log is created so the user can (hopefully) understand why.

Currently, we are on Version 1.0. While the base algorithm has been implemented, much improvement still remains. As such, there will be continual development of ScaleHD for the foreseeable future. For more information on version changes, please check *Developer Documentation*.

As of April 2020, I am leaving the university for a new challenge. ScaleHD has been updated to Python 3.7 for future support; feel free to fork and maintain the code as you desire.

The documentation for this software is organised into the relevant sections.

CHAPTER 1

Info

- Contact information can be found at https://helloabunai.github.io
- The codebase for ScaleHD is open source, and is available on Github at https://www.github.com/helloabunai/ScaleHD/.
- The project page on the Python package index is at https://pypi.python.org/pypi/ScaleHD.
- If you use ScaleHD in a study, please cite ((unpublished at the moment – writing paper ha ha ha)).

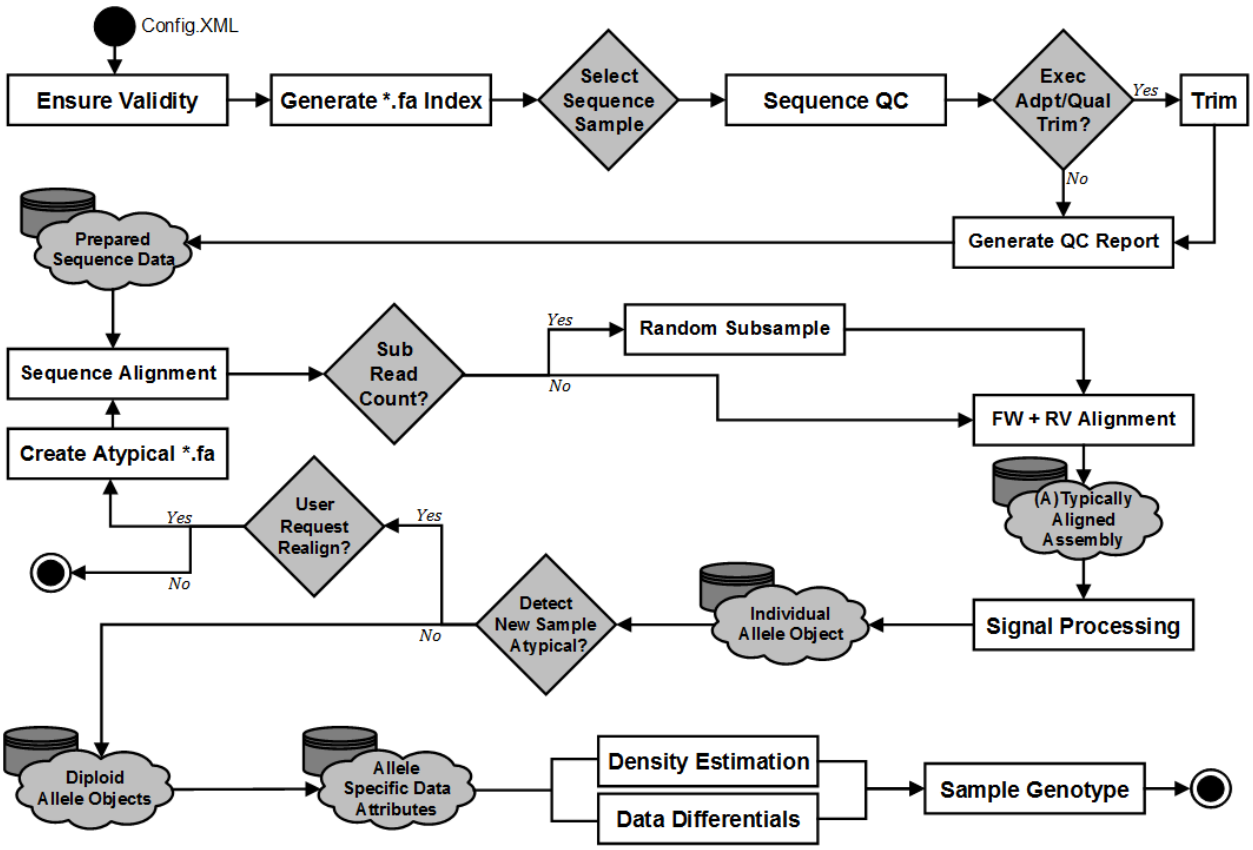
Development is undertaken at the University of Glasgow, Scotland, and is funded by the CHDI Foundation: http://www.chdifoundation.org/

1.1 Overview

As mentioned in the Index of this documentation, ScaleHD is a bioinformatics pipeline for use in broad-scope automated genotyping of Huntington Disease (HD) data. The pipeline is made up of four main stages: Sequence Quality Control (SeqQC), Sequence Alignment (SeqALN) and Automated Genotyping (GType). Once a stage has completed, as per user instructions, required information is automatically passed to the next stage (..a pipeline..).

It is designed with UNIX systems in mind, so there is unfortunately no Windows support. Why would you do research on Windows, anyway? For more details about hardware requirements, please see *Hardware requirements/recommendations*.

A broad overview of the pipeline stages can be viewed here:



1.2 ScaleHD-ALSPAC

There is a fork of ScaleHD, entitled ScaleHD-ALSPAC. The main functionality of this version is identical to "vanilla" ScaleHD, but differs in one key aspect. For the Avon Longitudinal Study of Parents and Children (http://www.bristol.ac.uk/alspac/), analysis of HTT data requires us to not know whether an individual has a CAG repeat size of over 31. As a genotyping platform, ScaleHD will expose this information multiple times throughout processing. Thus, this branch of ScaleHD will mask allele sizes from the end user, throughout intermediate files and processing stages, and the end genotype results.

If you don't have a need for this functionality, don't use this version of the application.

Due to the requirements of masking alleles, certain third party software output has to be culled in order to maintain this behaviour:

1.2.1 Quality Control

In the SeqQC stage of ScaleHD-ALSPAC, we are unable to produce a copy of cutadapt trimming reports as output for each sample. We continue to check the output of cutadapt, for warnings or error messages that would need to be passed on to the end user. However, this is done entirely in-memory.

FastQC would also compromise allele masking, and there is no realistic way to alter the output of this software. So, byebye FastQC (utilisation culled).

1.2.2 Sequence Alignment

Sequence alignment files are not written to disk and kept within memory objects and/or temporary files so that the end user is unable to manually inspect alignment assemblies. Instead of producing the sequence alignment as output of this stage, instead the required information (i.e. read count distributions) are passed straight to the next stage.

Incase the end user unexpectedly quits the program during any given stage, a purging function is ran on the output folder in order to delete any compromising data before it can be viewed.

1.2.3 Genotyping & SNP Calling

Genotype results are masked, so as to not compromise integrity. The reference labels saved to output summary CSV are edited as "31+", if applicable. For graphs rendered of read count distributions, any reads aligned to CAG31 or above are summed, and placed into CAG31. This obfuscates the true genotype of any disease-associated allele present.

1.3 Sequence Quality Control (SeqQC)

Assuming that all dependencies are functioning and input data is valid, the first stage of ScaleHD is Sequence Quality Control (SeqQC). This stage is utilised to attempt to provide further stages with the best possible quality data for aligning and genotyping.

As of version 0.312, we have included a pre-processing stage for demultiplexing sequence reads based on sequencing adapters. This is done via a wrapper for Cutadapt, called batchadapt – it is a lazy way for my colleagues to easily run cutadapt for sequence trimming on a folder of input files without having to think about bash scripts. As such, batchadapt has been added to the requirements for this package, and is automatically installed from pip along with other dependencies.

Initially, the pipeline will execute FastQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/) on input data. This provides a visual report on raw sequence quality, which may inform users of basic characteristics of their sequencing data.

The main focus of this stage utilises Cutadapt (http://cutadapt.readthedocs.io/en/stable/). In a ScaleHD configuration file, the user can specify three types of trimming to be carried out: quality, adapter or both. Quality trimming, if chosen, is recommended to be carried out before adapter trimming – if the user has chosen to use both types of trimming via ScaleHD, this is the order in which commands will be executed. Quality trimming will remove low-quality reads from an input data, providing any further analysis stages with the highest possible quality data available. See http://cutadapt.readthedocs.io/en/stable/guide.html#quality-trimming for more information.

Adapter trimming takes a literal sequence specified by the user, and trims all target reads of this sequence. This stage can have quite a profound impact on the downstream quality of genotyping, as sequence alignment is affected by proper adapter trimming by quite a noteworthy margin. If you are going to trim adapters, please make sure you are using the correct sequence for your data, and the correct positioning!

Note: Cutadapt adapter trimming requires additional reading to fully understand what action(s) is(are) executed. The application uses additional arguments to indicate what position on the trimming targets that any given adapter should be trimmed from (5 prime, 3 prime, anchored). Please see Cutadapt documentation on removing adapters for more information: http://cutadapt.readthedocs.io/en/stable/guide.html#removing-adapters.

Once complete, the trimmed data is then stored temporarily within the current sample's output folder. The input data is not moved or altered in any way; all further processing is carried out on the trimmed data, which is deleted upon sample completion. This avoids multiple copies of data being produced by each stage of the pipeline, and ensures that input data is not modified.

1.4 Sequence Alignment (SeqAlign)

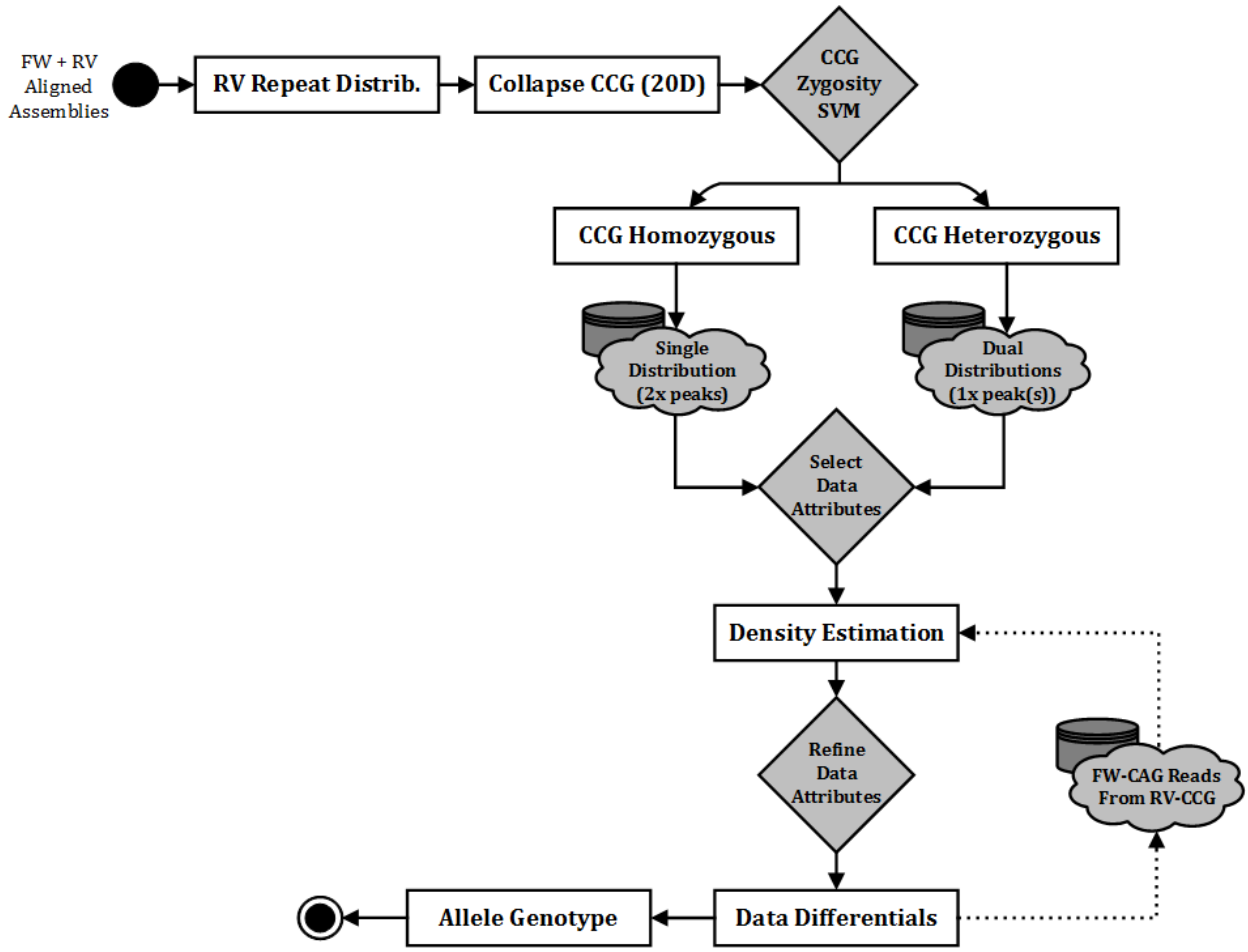
Next, is the sequence alignment stage. We utilise the Burrows-Wheeler Aligner (http://bio-bwa.sourceforge.net/) to carry out alignment of our sequences to a reference library. Our reference library structure of 4000 varying CAG/CCG references can be read about in *Data Assumptions*. Sample reference libraries are also located within the Github repository, under /src/ScaleHD/config.

BWA-MEM accepts a large array of arguments as input, to fine-tune the alignment behaviour. For our use-case, the default parameters perform sufficiently, with a few exceptions. Due to the highly repetitive nature of the *HTT* CAG/CCG repeat, we needed to increase Insertion/Deletion penalties raised to 6/6 and Gap Extension penalties to 4/4.

ScaleHD will first align forward reads to our library of 4000 varying *HTT* references, which takes the majority of the processing time for this stage in the pipeline. While this number of references may seem obtuse, it provides a robust platform on which to build genotyping approaches and the trade-off of accuracy vs processing time is considered worth it. The second stage takes the reverse reads for a sample, and aligns to our library of 20 references, with 100 static CAG and varying CCG values.

The purpose of this, is to reduce the complexity of genotyping a nucleotide-repeat associated disease with more than one relevant repeat tract. Consider two states of data: CCG-heterozygous (e.g. a sample with CCG7 on one allele, CCG10 on another) and CCG-homozygous (e.g. both alleles being CCG7). If we are able to determine the CCG-zygosity of a sample with absolute precision, then we have the opportunity to reduce any read count distributions we must search through for an allele's CAG value.

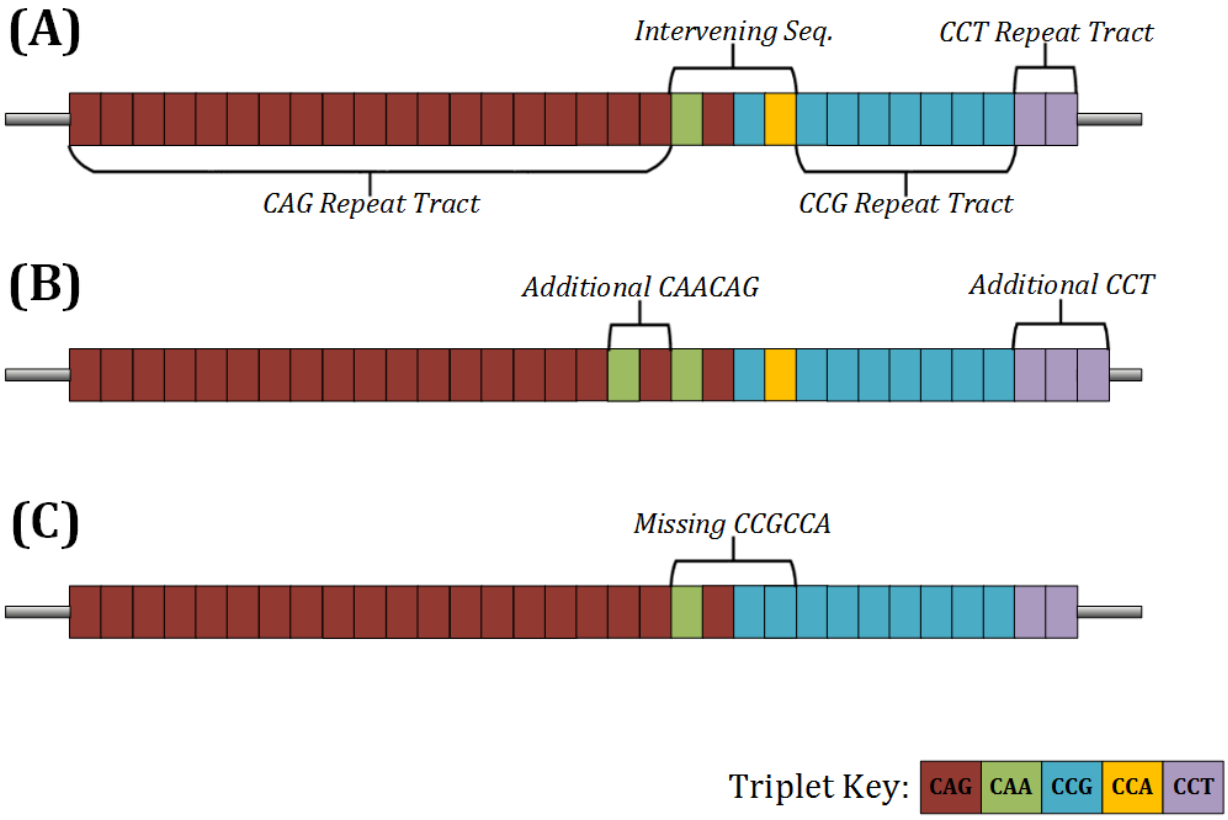
For example, we have an unknown sample and need to genotype from scratch. From our reverse alignment, we receive data which is of much better quality for the 3' end of the *HTT* data- the CCG repeat region. From here, we can better derive a precise distribution of CCG reads, allowing us to confidently determine CCG values. In this case, we have determined 7 and 10 from the reverse alignment, a CCG-heterozygous sample. Now that we know the sample is heterozygous, this simplifies the work required to determine CAG values for each allele; we know to expect only one 'CAG peak' within each CCG distribution. This is better represented in the visual workflow, which represents how the aligned data from this stage of ScaleHD is used to deliver accurate genotypes:



Once alignment is complete, we move onto the genotyping stage.

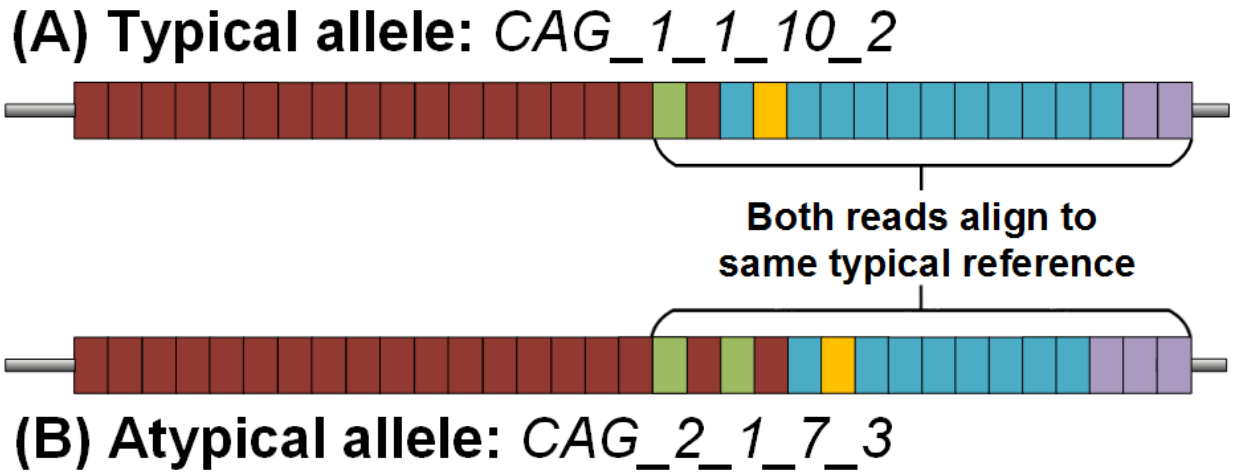
1.5 Automated Genotyping (Gtype)

The genotyping procedure of ScaleHD has three stages. The results from each stage are utilised in step with one another to act as a double-confirmation of the data. The first stage involves Digital Signal Processing (DSP). This stage's main function is to determine the structure of the *HTT* CAG/CCG repeat tracts. As seen in the figure below, a typical structure consists of a CAG tract, an intervening sequence, a CCG tract and a CCT tract (Pêcheux et al. 1995). In 95% of samples, the structure of the intervening sequence is one CAACAG hexamer, and one CCGCCA hexamer. This 'typical' structure is represented as: CAG_1_1_CCG_CCT. Example allele structures can be seen below.



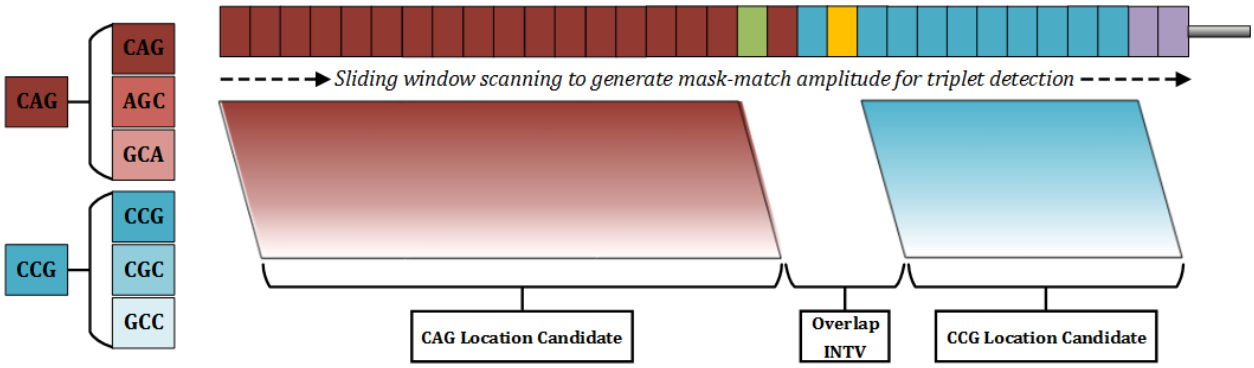
A) A standard HTT sample structure, nominally described as 'typical'. B) An example of a common atypical structure with an additional CAACAG in the intervening sequence. C) A futher example of a common atypical structure, with a missing CCGCCA.

However, in ~5% of samples, the structure differs (Yu et al. 2000). These 'atypical' samples, have an intervening sequence which is different from the normally observed one. These can vary in literal size, but an example of one atypical allele would be represented as: CAG_2_1_CCG_CCT. This is important as, due to the string-similarity of the hexamers in the intervening sequence, an atypical allele can be incorrectly assigned to a typical reference with a larger CCG. For example, a sample with the genotype CAG_1_1_9_CCT (A) would be the exact same length as CAG_2_1_7_CCT (B):



Thus, the DSP module will scan each read in an aligned assembly (aligned to a typically structured reference). This scanning is done to determine the literal structure of repeat units present, and if any alleles are atypical in nature. This is important as changes to the genotype may have relevant clinical implications. Existing generic microsatellite genotyping software does not take this specific *HTT* behaviour into account and will always provide an incorrect genotype, a problem that ScaleHD solves.

For each tract in *HTT* data (CAG, CCG, CCT), masks of every possible string rotation (e.g. CAG -> AGC -> GCA) are read across each read, in a sliding window. This is done, as not every read in an assembly will be the exact same length and may not start at the exact same position – this results occasionally in certain repeat tracts 'starting' on an offset base position, by one or two bases. In order to successfully identify *all* repeat tracts in *all* reads, these rotations are required. This mechanism produces a 'staggered assembly', which can be seen in the figure below.



Once we have identified where each repeat tract lies within a read, we can easily derive the literal string value of the intervening sequence – the information between the end of the CAG tract and the start of the CCG tract. However, with absurdly low rates, we have observed novel insertion mutations within the actual intervening sequence itself. As a result, the DSP module was extended to then scan the intervening sequence itself instead of naïvely extracting information in a position-based way.

With this information now curated, we create allele objects. Instances of this class contain relevant information *per allele*, which is useful to us in the case of samples where one allele is typical and one allele is atypical – despite being from the same sample, these alleles will have utilised different reference libraries, will have different assemblies, and so on. These attributes are assigned to each allele. We must then sort these alleles down to just two valid alleles, due to the diploid nature of the data.

We do this with the second genotyping stage, a heuristically driven filter model, which takes in characteristics of available data from allele objects to determine which two allele objects are valid choices. We sort objects by total number of aligned reads – the top allele is always taken as a 'primary' allele (quotmation marks to note that 'primary' here does not have any biological meaning; it is simply the allele we primarily chose from the assembly). Then we use weighted evidence from the remaining allele objects to determine which is valid; for example, relative read count in relation to CAG size, CAG size in comparison to the primary allele, CCG differences if CAG values are close, and so on.

Once filtered, we then pass to the final stage, genotype confirmation via Density Estimation and First Order Differentials. For each allele object, we observe the CAG distribution for each CCG value determined by DSP: that is, from the read counts against a forward-aligned 4000 reference library, we take the 200 values specific to the current CCG. In this sub-distribution, we then use density estimation to confirm if a peak is present at our suspect location. That is, from the CAG value derived from DSP, we observe if the number of aligned reads at that position is appropriately different from the rest of the distribution. If so, we then use first-order differentials (n-(n-1), across the whole distribution) to observe the largest change in aligned reads – a peak – an allele.

After this, graphs are rendered, results are written to output, and whatever else.

1.5.1 SNP Calling

In addition to our automated genotyping, we also run third party SNP Calling software to determine the presence of mutations in any given sample.

This function will execute with both Freebayes and GATK, but will only append output from the chosen algorithm, as stated in the user's configuration XML document.

Update: as of SHD 0.322, only freebayes is used. You no longer have a choice.

The output will exist with two columns per allele; Variant Call and Variant Score.

Variant Call will be structured as:

```
{X} -> {Y} @ SNP_POS
```

Where X is the base pair from the reference sequence, and Y is the mutated base pair present in the current sample. SNP_POS indicates the base pair position of the mutation within the sample reads.

Variant Score returns the PHRED score for the current mutation; only mutations from the current contig are considered and the mutation with the highest PHRED score is selected for output in the InstanceReport table.

1.5.2 References

Pêcheux C, Mouret J F, Dürr A, Agid Y, Feingold J, Brice A, Dodé C, and Kaplan J C. Sequence analysis of the CCG polymorphic region adjacent to the CAG triplet repeat of the HD gene in normal and HD chromosomes. *Med Genet*. May 1995; 32(5): 399–400.

Yu S, Fimmel A, Fung D, and Trent RJ. Polymorphisms in the CAG repeat: A source of error in Huntington disease DNA testing. *Clin Genet*. 2000; 58: 469–472.

1.6 Software Dependencies

ScaleHD functions entirely on open-source software, and itself is open-source. As ScaleHD is primarily a Python based package, certain Python libraries are required for it to work properly. These dependencies should be installed upon running the installer script, or when installing the package from PIP. However, if you are curious as to what packages are used (and which versions..) then you can find that information here.

1.6.1 Python Packages

All software packages have an accompanying version number listed. These are the versions of each package from which ScaleHD was developed. Newer versions may function fine, but are not tested. It is highly recommended to avoid older versions, and if possible, use the exact same version as during development. If you do not want to overwrite any packages of which you have a newer version, please see the sub-section in *Detailed Instructions* about utilising virtual environments. As of ScaleHD version 0.321, dependencies have been updated to the latest available stable versions (at the time of writing). Syntax interaction for a few dependencies had changed, so ScaleHD may not function properly with versions other than that listed here.

- cutadapt (1.18)
- generatr (0.252)
- batchadapt (0.22)
- lxml (4.3.2)

- matplotlib (2.2.4)
- numpy (1.16.2)
- pandas (0.24.1)
- peakutils (1.3.2)
- seaborn (0.9.0)
- PyPDF2 (1.26.0)
- pysam (0.15.2)
- regex (2018.11.22)
- reportlab (3.5.12)
- scipy (1.2.1)
- sklearn (0.20.2)

1.6.2 Third Party Binaries

ScaleHD also uses third party binaries to carry out certain functions. These binaries are expected to exist on your system $PATH variable, so they can be called by a UNIX subprocess without invoking a shell (as this is a huge security risk and generally bad design, and anyone who uses shells within a subprocess should feel bad about themselves). For more information on how to put binaries on your $PATH, please see the section *Installation*. As the user can select which stages of ScaleHD to run, only the binaries utilised in each selected stage will be required. *However, for peace of mind it is recommended to just provide all required third party binaries on your $PATH anyway*. As with python dependencies, a software version is listed. Newer versions of dependencies may work with ScaleHD; using older versions is not recommended.

- Java (1.8.0_20)
- FastQC (0.11.7)
- SeqTK (1.3-r106)
- BWA-MEM (0.7.17-r1188)
- Samtools (1.9)
- Picard (2.18.23)
- FreeBayes (v1.2.0-2-g29c4002)

1.7 Installation

As ScaleHD is a pipeline with a potentially varied userbase (in terms of background), certain users may need more instructions on how to get the software up and running. As such, the documentation for installation is split into two sub-sections. For users who are comfortable with UNIX systems, command line terminals and general package/dependency management, please read the section *Quick Instructions*. For other users, who may never have used UNIX before, or are uncomfortable with command line interfaces, please see the *Detailed Instructions*.

As I have developed the software on OS X, and most of the end-users in our lab will be running the pipeline on this operating system, instructions are tailored to it as a result. **However**, if you are using a GNU-UNIX type operating system (Ubuntu, for example), there will be notes on how to install ScaleHD for your OS, too. ScaleHD has been developed on OS X 10.11.6 and tested on OS 10.10-10.11.6, as well as Ubuntu 14.04 LTS.

1.7.1 Quick Instructions

If you know your way around a terminal, these instructions are for you.

ScaleHD uses Python 2.7.14, so ensure your target for package installation is the correct version of Python on your system.

```
pip install scalehd
```

This will install all python dependencies for you. Depending on your user privilege level, you may want to use sudo. For each third party binary, please compile the source to your systems specification, or use a pre-compiled binary provided by the developer. Add an entry to your $PATH for each of 'java', 'fastqc', 'seqtk', 'bwa', 'samtools', 'r' and aliases for 'picard' and 'gatk'. **These binaries must be as listed here, in lower case.** This pipeline has only been tested on bash and zsh, so other shells may exhibit unexpected behaviour(s). Once you have augmented your $PATH as above, you are good to go.

1.7.2 Detailed Instructions

If you're new to UNIX, bioinformatics, or command line interfaces in general, these instructions are for you. We will install ScaleHD from a completely clean install, step-by-step. This means, we assume your UNIX operating system has no dependencies installed at all, and requires everything installed from scratch. We assume absolutely zero knowledge of any techniques required to install and use ScaleHD, so it will be very in-depth. Let's begin!

Before we install anything related to ScaleHD, it is worth mentioning that if you are also working on OS X, you will be required to install command line tools to make the remaining installation procedure possible. To do this, open a terminal. Press +Space to open spotlight search, and type "Terminal". In this prompt, type the command:

```
xcode-select --install
```

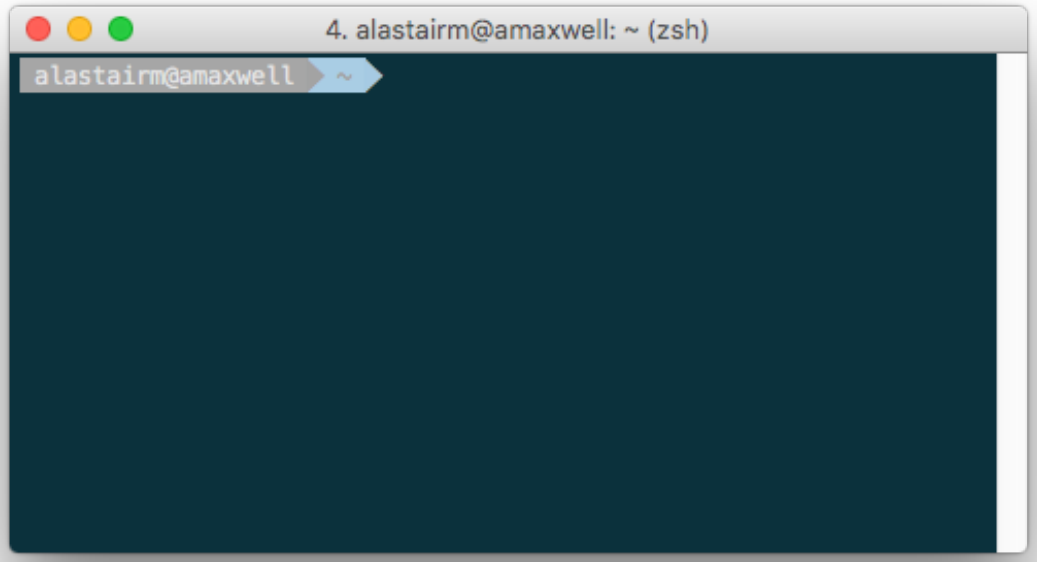
A GUI prompt will appear; press 'Install'. This will download a ~130MB package and install it for you. You need not do this on Ubuntu as Canonical have the sense to include a C++ compiler and package manager with their operating system, whereas Apple do not.

The next step is to install PIP. PIP is a Python package manager, which allows you to install software from the Python package index, over the internet, without being required to get dependencies manually. Your computer system may already have PIP installed, but incase it doesn't, we will install it here. You need to download the Python script get-pip.py from https://bootstrap.pypa.io/get-pip.py. Once you have this downloaded, we need to run it to install PIP.

The next step is to run the script. Go back to your terminal which you opened previously, and prepare to enter a new command.

Warning: PIP requires Python to function. If you are working on OS X or any modern version of Ubuntu, you already have Python 2.7 installed, and do not need to do anything else. If you lack access to Python 2.7, contact your system administrator to get it installed, or see https://www.python.org/downloads/release/python-2713/.

Note: Throughout these instructions, we will refer to a "builds" directory, where we will be installing third-party binaries. In our examples, we have used the folder path '/Users/alastairm/Documents/Builds', wherein will be sub-folders for each binary. How you allocate your binaries is up to you, but this tutorial will follow this builds directory format.



Now we run our downloaded script to install PIP. Assuming that your script is in your user Downloads folder (/Users/user-name-here/Downloads/), we will execute the following command:

```
python2 ~/Downloads/get-pip.py
```

The specific command you need to run will depend on the location that you chose to download get-pip.py. Regardless, This will install PIP onto your system, allowing you to install ScaleHD and any Python-based dependencies with no effort at all. In order to do that, in the same terminal, issue a new command:

```
pip install scalehd
```

This will install all python dependencies for you. Depending on your user privilege level, you may want to prefix the above command with 'sudo', which will run the command at an administrator level. If you are unsure, talk to your system administrator. Now that ScaleHD and its dependencies have been installed, we need to install the required third-party binaries which are not available for installation from PIP.

Warning: For all the following third-party binaries, ScaleHD attempts to detect binaries with a lower case filename. If your binary files are called something else, ScaleHD will be unable to locate them on your $PATH. Please ensure that your binaries are all respectively named 'fastqc', 'seqtk', 'bwa', 'samtools' and 'r'. ScaleHD also requires aliases for 'picard' and 'gatk', which we will cover in their respective sub-sections.

1.7.3 Java (1.8.0_20)

Java is required for certain packages to run, but is normally distributed as part of any UNIX based operating system and should already be installed on your system. To check, open a terminal and type the command:

```
java -version
```

This will return a string, telling you which version of Java is installed (if any). ScaleHD has been developed with 1.8.0_20 installed, but any 1.8 version of Java should function the same. If you do not have Java installed, please install the Java Development Kit **and** the Java Runtime Environment for your system. See http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html and http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html for more information.

1.7.4 FastQC (0.9.2)

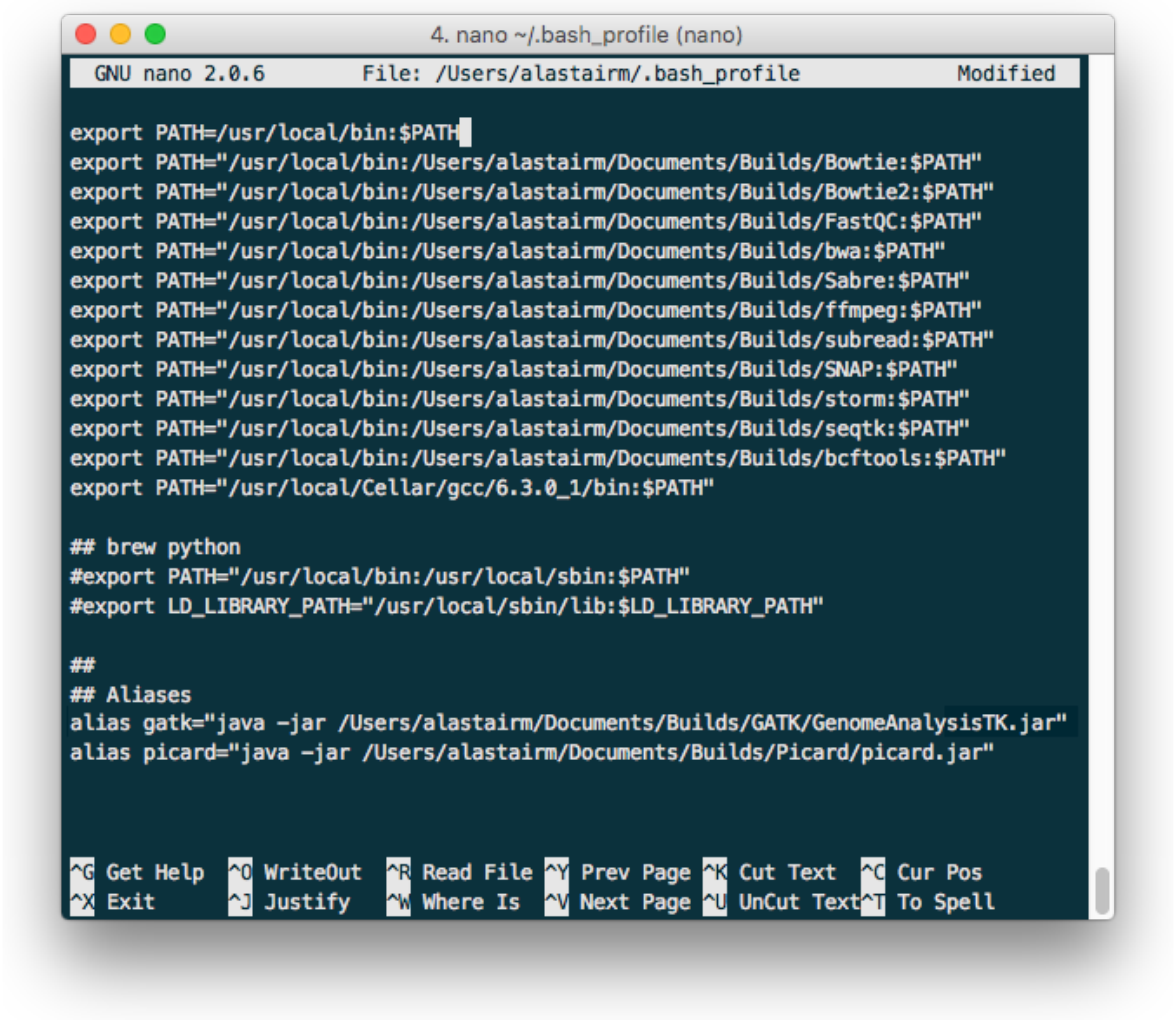
FastQC is a bioinformatic tool which generates visual reports on the quality of a particular input sequence data from NGS platforms. It is available as a GUI (graphical user interface) based program, but we need to acquire a version which runs as a command line tool so that ScaleHD can run it in a pipeline. To do this, please download FastQC from https://www.bioinformatics.babraham.ac.uk/projects/download.html#fastqc. Regardless if you are using OS X or Ubuntu, please download the Win/Linux zip file. Extract this zip, then locate the binary titled "fastqc". Copy this binary file to your Builds folder, then open a terminal.

Note: We will now add FastQC to our $PATH. By default, OS X and Ubuntu will be using the BASH environment, so these instructions are for BASH. If you are using an alternative shell, you are advanced enough to know how to add things to your $PATH already and shouldn't really be reading this.

We are going to edit a text file in our terminal. Here, the program 'nano' is used, but you can use whichever editor you prefer.

```
nano ~/.bash_profile
```

This will open a screen, similar to this (but your file will be blank):



Adding things to your $PATH is as simple as typing the following:

```
export PATH="/usr/local/bin:/path/to/your/binary/file:$PATH"
```

for each binary. In our example case of the Builds directory, and for FastQC, we would thus type:

```
export PATH="/usr/local/bin:/Users/alastairm/Documents/Builds/FastQC:$PATH"
```

And now, FastQC is on our system's $PATH, and can be called/executed from any location in the shell. However, before we do that, you need to 'apply' the changes. This is done by sourcing your bash_profile file:

```
source ~/.bash_profile
```

Alternatively, you can restart your terminal application, or log out and log in.

1.7.5 SeqTK (1.2-r101-dirty)

As the methodology for adding binaries to your path has been covered in install-fastqc, we will not cover it again for every single binary. SeqTK is available at https://github.com/lh3/seqtk, where you will also find instructions on how to

compile the application for your system. Once compiled, move the binary to your Builds folder, and add that directory to your $PATH.

1.7.6 BWA-MEM (0.7.15-r1140)

Burrows-Wheeler Aligner is available from http://bio-bwa.sourceforge.net/. Extract the downloaded tarball, then move into that directory:

```
tar -zxvf ~/Downloads/bwa-whatever.tar.bz2
cd ~/Downloads/bwa-whatever/
```

Inside this directory, we will compile the source code into the binary executable:

```
./configure
make
make install
```

This is the standard trio of commands to configure a source for your system, make the binary, and install the binary. A file, 'bwa', will appear in the current directory after successful compilation. Move this binary to your Builds folder, and add that directory to your $PATH.

1.7.7 Samtools (1.3.1)

Samtools is available from http://samtools.sourceforge.net/. Installation is identical to that of BWA-MEM. Extract the downloaded tarball, and move into the newly extracted directory. Configure, make and make install. Copy the new samtools binary to your Builds directory, and add it to your $PATH.

1.7.8 Picard (2.18.3)

Picard can be downloaded from https://broadinstitute.github.io/picard/.

Warning: Depending on which shell environment your operating system uses as default (or whichever shell you have chose to use), aliases may not be correctly read from your user profile by the bourne shell, which is the environment utilised by python's subprocess module. In order to account for this, modifications to how ScaleHD interacts with Picard were made as of version 0.31.

Previously, ScaleHD interacted with Picard via a user-generated bash alias. However, throughout more robust testing of different environments, we encountered certain combinations of operating systems and shell environments being unable to successfully get the required information for aliases to function. As such, we have changed (as of ScaleHD v0.31) how we interact with this program.

The user must create a unix script, which handles input arguments and launches the Picard JAR. An example script will look like the following:

```
java -jar /Users/alastairm/Documents/Builds/Picard/picard.jar␣
↪CreateSequenceDictionary REFERENCE=$1 OUTPUT=$2
```

As usual, replace the literal directory with your own Builds path. Save this as a file (with no extension) called 'picard'. Include this in the same folder as the Picard JAR, so that your ~/Builds/Picard folder looks like:

Builds

```
├── Picard
│   ├── picard ##the binary script we just made
│   ├── picard.jar ##the download jar archive
```

Then, make our script executable:

```
chmod +x /Users/alastairm/Documents/Builds/Picard/picard
```

Once made executable, add the Picard folder to your $PATH. Picard is now set-up for ScaleHD.

1.7.9 FreeBayes (v1.1.0-60-gc15b070)

FreeBayes has also been included within the SNP calling module of ScaleHD. Throughout development and testing, we observed a stronger performance of amplicon flanking sequence SNP detection with freebayes, and as such, the output of this binary is treated with more prominance in ScaleHD. Freebayes is available on github at https://github.com/ekg/freebayes. The readme for that repository contains installation instructions, which consists of a standard make/make install. It is also available for download from Homebrew, for easier installation on OS X.

Once installed (assuming via Homebrew, or by installing to /usr/local/bin with 'sudo make install'), the binary will be on your $PATH and ready for use by ScaleHD.

1.7.10 Virtual Environments

Virtual Environments allow a Python user to create a separate terminal environment, which is separate from the 'main' environment of the operating system, but acts in an identical manner. This allows you to create an environment for a specific purpose, e.g. installing specific versions of packages that you did not wish to overwrite in your 'main' environment. This is useful if you have certain Python packages installed for other projects, which require a different version than that of ScaleHD.

To read up on Virtual Environments, we recommend reading this tutorial: http://docs.python-guide.org/en/latest/dev/virtualenvs/.

1.7.11 Common Issues

When colleagues were testing the software, these were the most common issues encountered when installing and/or running ScaleHD. If you're having trouble installing ScaleHD, hopefully an answer to your issue will be here.

- LibXML headers missing

For this issue, you are missing the libxml2-dev and libxslt-dev libraries from your system. These packages should be installed as part of lxml, which is included in the setup script for ScaleHD, and should have been installed automatically. However, you can check if you are missing this package by opening a terminal, launching the Python interpreter (run the command 'python'), and then trying the command 'import lxml'. If this fails, then you know the package did not install properly, for whatever reason.

To remedy this, you can install lxml 'manually' from pip:

```
STATIC_DEPS=true sudo pip install lxml
```

This command will require an internet connection, as it will download the sources for each developer library and build them for you, hopefully resolving any issues you have with lxml.

- LibXML parsing error stack

As of the time of writing, there is no functionality within ScaleHD to check the structural integrity of your XML configuration files used, outside of the validity of provided attribute flags. If you have malformed XML, such as misplaced tags, ScaleHD will not launch and you will be greeted with a debug stack from lxml failing to parse invalid XML. Please check your XML and try running ScaleHD again.

- SciPy stack errors

Sometimes the SciPy stack installs incorrectly from PIP, or fails quietly (i.e. the install failed, but claimed it was successful). If this is the case for you, we recommend installing the SciPy stack at a user level.

```
pip install --user numpy scipy cython matplotlib
```

If this still refuses to work, you can look into installing these dependencies via Homebrew (see: https://brew.sh/).

```
brew tap homebrew/science && brew install python numpy scipy matplotlib
```

1.7.12 ScaleHD on Windows

Natively, ScaleHD has no support for Windows operating systems. However, with Windows 10, Microsoft has provided a way by which to run a unix environment on your Windows computer. This allows ScaleHD to run on Windows machines, through this Linux Subsystem Layer. The Linux Subsystem is only available for Windows 10, versions 1709 and later. Other versions of Windows are not supported.

To install the Linux Subsystem:

Open PowerShell (search for the application in the start menu), and type the following command:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

This will require a reboot of your system to complete.

Once complete, you need to choose a linux distribution to install to your newly installed unix subsystem. Unfortunately you must do this via the Windows Store, because yay for unnecessary homogenisation. Open the Windows Store, and search for your distribution. We have tested ScaleHD on Windows 10 only in ubuntu 18.04.

Once installed, you can open a command prompt (cmd.exe) to begin. Launch your installed distribution's shell. For our example:

```
ubuntu1804
```

This changes your prompt from a Windows shell to the Ubuntu Bash environment. From here, you can follow ScaleHD installation instructions as above.

1.8 Hardware requirements/recommendations

ScaleHD uses technologies such as Sequence Alignment to automatically genotype HD samples. Such processes can be quite intensive on your computer, but hardware requirements are not extraordinary compared with similar software in the same field.

The pipeline has been developed on a Late 2013 Mac Pro, which has the following hardware:

- 3.5 GHz 6 Core, 12 Thread Intel Xeon E5
- 64 GB 1866 MHz DDR3 RAM
- 1TB PCI-E Solid State Drive
- AMD FirePro D500

However, you do not necessarily require the same specification of computer to successfully run ScaleHD. The amount of CPU threads and system RAM you have installed will improve the performance of sequence alignment, but the rest of the pipeline is generally not demanding, in terms of performance. Developed features, such as Digital Signal Processing, require a non-insignificant amount of processing cycles; this is due to the single-threaded nature of Python more than any hardware bottlenecks. There is a planned re-write of the Digital Signal Processing module in a compiled language (instead of an interpreted one), in the future.

1.8.1 Minimum specifications

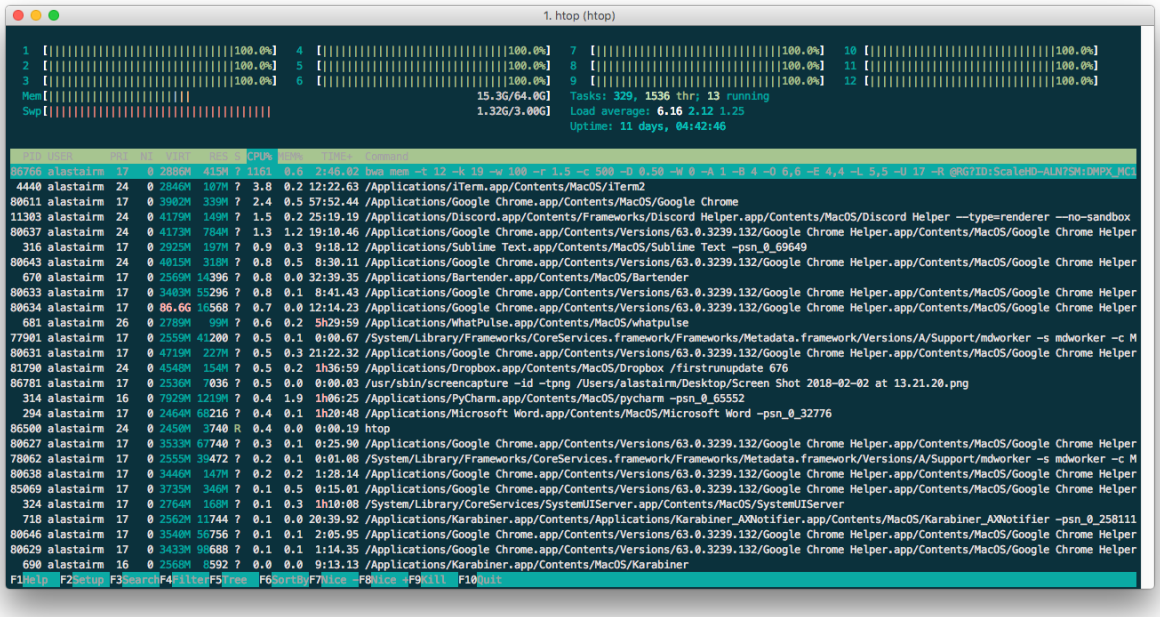
Some 'minimum specifications' for a satisfactory level of performance should be understood as (roughly) follows:

- A modern Intel (2013 onwards) or AMD Ryzen (2017 onwards) chipset, with a minimum of 4 threads (the more, the better).
- An absolute minimum of 16GB DDR3 RAM, but ideally more.
- A large amount of data storage (>500GB free).

For an example of ScaleHD's genotyping performance on my development machine, here is a description of ScaleHD running on a cohort of 408 illumina miseq samples:

- 816 input files, totalling 15GB
- Utilising all 12 threads
- Maximum RAM usage at ~17GB
- A total processing time of 27 hours
- An average sample processing time of 7 minutes
- The longest sample processing time being 13 minutes
- An output folder totalling 14.8GB

And a snapshot of the resources utilised by the most intensive module of ScaleHD, the sequence alignment wrapper:



Hopefully this information gives you enough of an idea to plan your sequence run with ScaleHD.

1.9 Data Assumptions

ScaleHD makes a few assumptions about the input data which you provide it. Details about those assumptions can be found in this section, to aid the user in preparing their input into an appropriate format for ScaleHD to process it successfully.

1.9.1 File Input

ScaleHD utilises both forward and reverse reads from amplicon sequencing in order to genotype a sample effectively. In order to determine whether an individual file is a sample's forward or reverse data, ScaleHD looks for the substring "_R1" (forward) and "_R2" (reverse) at the end of each file name. Thus, user input should follow this behaviour within the specified input folder.

In addition to the file name, ScaleHD expects any input folder to contain an even number of files, and only sequence input data (either *.fastq.gz, or unzipped) within said input folder. To be verbose, valid input file extensions are as follows:

- sample_name.fastq.gz
- sample_name.fq.gz
- sample_name.fastq
- sample_name.fq

No other files will be considered valid. If there is a non-even number of input files present (i.e. not every sample has two files, R1 and R2), ScaleHD will not run.

1.9.2 Multiplexed Data

If your sequence data originates from a samplelibrary combined with other non-*HTT* data, then your data will need to be de-multiplexed before it can be used within ScaleHD. In order to demultiplex data, we typically use Cutadapt (http://cutadapt.readthedocs.io/en/stable/) to select *HTT* reads only on the basis of the presence of the PCR primer binding site in the reads. Even for non-multiplexed data (*HTT* only data) it is useful to use Cutadapt to collect reads that start with the PCR primer binding site. There are two benefits in doing this:

- Removing the reads that, for some reason, would not start with a PCR primer binding site.
- Trimming all the reads at the same position within the PCR primer binding site. This allows processing of reads which start at the same position, and also allows for trimming of the hetereogeneity spacer we often use for sequencing library preparation.

Due to colleague unfamiliarity with Bash scripting, it was easier to write a simple python wrapper to run Cutadapt in batch mode, for particular barcodes which our lab uses. See BatchAdapt at https://github.com/helloabunai/Batchadapt. Documentation for this small wrapper is in progress at the time of writing, so a simple instruction set is as follows:

- Clone the Github repository
- Run the setup script in a terminal: python setup.py install
- Run "Batchadapt –help" to see all input options

More detailed explanation of the demultiplexing process is a work in progress and will be updated here when complete.

1.9.3 Reference Libraries

Due to the multiple reference design approach to determining legitimate alleles that ScaleHD uses, the pipeline expects an input reference library to conform to a certain standard.

For the *forward* reference library, there must be a total dimensional value of 4000 references within the library; CAG1-200 with CCG1-20. Each individual reference sequence label must also conform to a naming standard. For example, a reference sequence with CAG 17 and CCG 6 (i.e. CAG17_CCACAG1_CCGCCA1_CCG6_CCT2) must be labelled as:

```
>17_1_1_6_2
GCGACCCTGGAAAAGCTGATGAAGGCCTTCGAGTCCCTCAAGTCCTTCCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCA
```

For the *reverse* reference library, we utilise a static 100CAG with varying CCG1-20. This is as we only utilise the reverse reads to determine CCG values and resultant zygosity. An example of an individual reverse reference is:

```
>100_1_1_15_2
CAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCA
```

Basically, the only thing that is able to be altered about a user's input reference library is the forward and reverse flanks of the sequence. This is an intentional design decision, and ScaleHD will not function properly with alternatively styled reference libraries. Please see my other software, RefGeneratr, at https://github.com/helloabunai/RefGeneratr. This package allows for generating custom reference libraries with ease, and is actually installed as a dependency for ScaleHD.

An example of valid reference libraries can be seen on ScaleHD's github page, under /src/ScaleHD/config/4k-HD-INTER.fas and /src/ScaleHD/config/20TypicalReverse.fasta. Valid file extensions are:

- reference_name.fasta
- reference_name.fas
- reference_name.fa

1.10 Specifying User Input(s)

ScaleHD is a Command Line Interface (CLI) based application, and users will interact with it via a terminal environment. Once installed, the program can be launched by simply issuing the command 'scalehd' in a terminal. This, by default, will not do anything, but will list all the available input options to the user. Here we describe what these options are, how the user can modify them and what those changes will do to the functionality of ScaleHD.

Short argument	Long argument	Function
-h	–help	Prints a help message
-v	–verbose	Enables terminal feedback for the user (def: blank)
-c {string}	–config	Specify a directory to your configuration file.
-t {integer}	–threads	Number of CPU threads to use (def: system max).
-p	–purge	Remove all output files except the HTML5-based report.
-s	–simple	Enable simple 95% confidence interval genotyping output.
-e	–enshrine	Do not remove non-uniquely mapped read from SAM files.
-b	–broadscope	Do not subsample fastq data with high read counts.
-g	–groupsam	Output all sorted SAM files into one job-wide directory.
-j {string}	–jobname	Customises the output folder name for this job.
-o {string}	–output	Specify a directory you wish output to be directed at

1.10.1 Configuration File

ScaleHD's main arguments, rather than filling out cumbersome details on a command line (where editing can be frustrating), instead uses an XML document. This allows for an easy mechanism by which to reproduce results of a processing job – if you use the same configuration file, you can guarantee that data produced at the end will be identical.

Here is an example configuration file:

```
<config data_dir="~/path/" forward_reference="~/path/" reverse_reference="~/path/">
  <instance_flags demultiplex="False" quality_control="True" sequence_alignment="True
↪" atypical_realignment="True" genotype_prediction="True" snp_calling="True"/>
  <demultiplex_flags forward_adapter="GCGACCCTGG" forward_position="5P" reverse_
↪adapter="GCAGCGGCTG" reverse_position="5P" error_rate="0" min_overlap="10" min_
↪length="" max_length=""/>
  <trim_flags trim_type="Adapter" quality_threshold="5" adapter_flag="-a" forward_
↪adapter="GATCGGAAGAGCACACGTCTGAACTCCAGTCAC" reverse_adapter=
↪"AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" error_tolerance="0.39"/>
  <alignment_flags min_seed_length="19" band_width="100" seed_length_extension="1.5"␣
↪skip_seed_with_occurrence="500" chain_drop="0.50" seeded_chain_drop="0" seq_match_
↪score="1" mismatch_penalty="4" indel_penalty="6,6" gap_extend_penalty="6,6" prime_
↪clipping_penalty="5,5" unpaired_pairing_penalty="17"/>
  <prediction_flags snp_observation_threshold="2" quality_cutoff="0"/>
</config>
```

Within the **<config>** branch, there are three attributes to which the user must assign a value. *data_dir* must point to your input folder, consisting of an even number of input data files (see: *Data Assumptions*). *forward_reference* points to a .fasta reference file, for which alignment is carried out on forward reads. *reverse_reference* points to a .fasta reference file, for reverse alignment.

<instance_flags> determines which stage(s) of ScaleHD that the user wishes to run. These are all simple True/False boolean options, where 'False' means a stage will not be processed. Other stages all function as expected.

<demultiplex_flags> provides input arguments for demultiplexing, if chosen. These are arguments for Batchadapt, which is a simple wrapper I wrote for a colleague to run Cutadapt (http://cutadapt.readthedocs.io/en/stable/) on a folder

of input sample pairs. x_position flags must be '5P', '3P' or 'AP'. Adapter strings must only contain A,T,G,C. Other arguments are positive integers, only.

<trim_flags> is for specifying options that get passed to cutadapt for sequence adapter/quality trimming. For an explanation of these flags, you can find the cutadapt documentation at: http://cutadapt.readthedocs.io/en/stable/.

Note: *forward_adapter* and *reverse_adapter* are quite important, and you should carefully select how much adapter you wish to trim from your sequences. Incorrect trimming can have a notable downstream effect on the quality of resultant alignments, and thus, genotypes.

<alignment_flags> is for specifying options that get passed to bwa-mem for sequence alignment. A detailed explanation of all bwa-mem flags can be found in the page's documentation at: http://bio-bwa.sourceforge.net/bwa.shtml. However, you may be confused about which command line flags correspond to which XML fields in ScaleHD's input configuration. An explanation:

ScaleHD XML	BWA-MEM argument
min_seed_length	-k <INT>
band_width	-w <INT>
seed_length_extension	-r <FLOAT
skip_seed_with_occurrence	-c <INT>
chain_drop	-D <FLOAT>
seeded_chain_drop	-W <INT>
seq_match_score	-A <INT>
mismatch_penalty	-B <INT>
indel_penalty	-O [INT, INT]
gap_extend_penalty	-E [INT, INT]
prime_clipping_penalty	-L [INT, INT]
unpaired_pairing_penalty	-U <INT>

<prediction_flags> now has a function! wow!

Since SNP calling has been implemented, the user has the option of choosing the cutoff for filtering what we consider to be a valid SNP, which is done with the snp_observation_threshold flag. Values accepted are range(1,5). 1 being the most lenient value in determining a SNP as valid, 5 being the most harsh. I typically use 2. Do what you want.

Quality cutoff indicates a positive integer which is utilised as a further filter for variant validation.

The VCF "QUAL" is described in the latest specification as:

```
    QUAL - quality: Phred-scaled quality score for the assertion made in ALT. i.e.␣
↪10log10 prob(call in ALT is
wrong). If ALT is '.' (no variant) then this is 10log10 prob(variant), and if ALT is␣
↪not '.' this is 10log10
prob(no variant). If unknown, the missing value should be specified. (Numeric)
```

The user can control the Phred-scaled score lower limit for a valid SNP to be included with ScaleHD output via this option. This is entirely up to the user, and how sensitive you want SNP calling to be. If ScaleHD misses a mutation because specified quality cutoffs were too strict, that's not really ScaleHD's fault.

1.11 Running ScaleHD

An example of running the program in a "default" state (max threads, remove non-unique reads, subsample high read count) would look like the following:

```
ScaleHD -v -c ~/Documents/Work/ScaleHDTests/ArgumentConfig.xml -j "hello_documentation
↪" -o ~/Documents/Work/ScaleHDTests/Output
```

Once this command is entered into a terminal, ScaleHD will read the input from ArgumentConfig.xml, process the input data specified there, and genotype all samples where possible. This is all you need to do to run ScaleHD; when the input is set up correctly, no input is required from the end user.

1.12 Monitoring Performance

If you're running ScaleHD in verbose mode (-v/–verbose), the pipeline will print to the terminal any updates on stages for each sample that is being processed. This involves a colour code, for success (green), warning/information (yellow) and failure (red). However, if you're more interested in progress for each individual sample as it is being processed, you will need to look at additional software. Personally, I find it easiest to use the program *htop*, which you can read about here: https://hisham.hm/htop/. Running htop will allow you to see what third-party binaries that ScaleHD has launched (i.e. what stage of the sample processing it is on), as well as resource usage.

Normally, we just leave a job running overnight and return to it being finished (or almost finished) the next morning.

1.13 Output Hierarchy

ScaleHD's output is fairly straightforward. For each job that you run, there will be one master output folder containing all data produced by the pipeline. If the user specified a '-j' argument in the command line interface, the output folder will be named as such; if not, the folder will be named ScaleHD-DateTime.

Within the master folder, there will be a few instance-wide report files:

- AlignedDistributions.csv
- InstanceGraphs.pdf
- InstanceReport.csv
- UtilisedConfiguration.xml

AlignedDistributions contains a matrix of read count distributions for each allele, in each sample processed by the pipeline. Each individual allele's distribution will be aligned to the same index within this matrix; the peak, or n-value, of each allele is at the same index. This allows the user an easy method by which to cross-compare levels of somatic mosaicism.

InstanceGraphs is a series of graphs rendered for each sample. These graphs plot the CCG distribution, and respective CAG distributions for each allele in a sample. This allows the user to easily search for samples that may require manual inspection and view read count distributions quickly.

InstanceReport is the 'main' report output from ScaleHD, and displays allele genotypes and relevant statistics, data characteristics and errors (where applicable). A detailed explanation of what each of the attributes/flags in this file explains, please see *Miscellanious Definitions*.

UtilisedConfiguration is a literal copy of the ScaleHD configuration file specified as input by the user when the job was launched.

In addition to this, each stage of the ScaleHD pipeline will have a respective output folder with relevant data within:

- SeqQC – containing FastQC reports and any trimming reports
- Align – containing aligned reads (in *.SAM format), an alignment report and alignment statistics. These files are present for both FW and RV data.

- Predict – containing a genotype graph PDF, reports on each allele's genotype (and associated flags), as well as a penalties log for each allele (stating what was deducted from each allele during confidence calculation). Variant calls are also present in this folder, with a VCF file from Freebayes available for further analysis. However, if variants were detected in contigs which are not either of the alleles in a sample, these 'irrelevant' variants are reported on in their own report file.

1.14 HTML based output

As of version 0.321, ScaleHD renders HTML output that provides an easy-to-share summary of all samples processed within a run of ScaleHD. We utilise Gridism, JQuery, Chart.js (and extensions), and MSAViewer to render all aspects of ScaleHD output within the browser, minimising file size. Proper write-up documentation of this will come for version 0.322, which will have additional features added to the webpage output.

1.15 Miscellanious Definitions

Within the instance-wide output table produced by ScaleHD, there are many flags or data entities which require explanation. Throughout the development of ScaleHD, we ended up determining a range of characteristics that indicate the believability of data produced from amplicon sequencing when attempting genotyping via our multiple reference library-based method. These characteristics provide us with a heuristic method to determine if an attempt at automated genotyping was successful, or not. Here, we define these characteristics and what they mean in literal terms. They range in importance, but all are useful in creating a representation of data quality. Some are self-explanatory, but are explained anyway.

A further note on SNP calling: Either Freebayes or GATK may detect variants in allele contigs which are not relevant to the literal alleles of any given sample. I.E., a sample with alleles 17_1_1_7_2 and 23_1_1_10_2 may have variants reported in a highly irrelevant contig, 40_1_1_5_2. SNPs are only reported within InstanceResults.csv if they are found within the appropriate contigs for that sample – other 'irrelevant' variant reports are written to IrrelevantVariants.txt in the sample's specific output folder. An individual SNP will be reported in the format "{originalbase }->{mutated base}: @{base pair position in read}". E.G. "C->T: @36".

Update: as of SHD 0.322, only freebayes is used.

The significance levels are as follows:

- N/A – This means the flag contains discrete information and does not need to be interpreted in regards to genotyping quality.
- Dependent – This flag may be significant, depending on other flags. For example, a high level of somatic mosaicism may be an indicator of poor genotyping quality when the CAG repeat tract size is within the non HD-causing allele size range.
- Minor – This entity is of minor significance and in the vast majority of samples will not be a deterministic factor for genotyping quality.
- Moderate – This entity is of moderate significance. It is unlikely to render a sample's genotype invalid on its own but may contribute to inaccurate genotyping.
- Major – This entity is of major significance and is strongly associated with genotyping quality. If any major informative flags are raised, it is recommended to manually inspect the alignment/mapping outputs for that sample.

For maximum genotyping accuracy we recommend manual inspection for all samples for which any major flag was raised and for alleles with >47 CAGs.

```
*n denotes the number of reads for the modal allele; **very low reads is
defined as an n value containing <=200 reads
```

1.15.1 Confidence Calculation

For each allele, ScaleHD calculates the confidence level in the provided genotyping result. This information is taken from a variety of sources, and attempts to paint an evidence-based picture of the data quality, and resultant genotype confidence. Each allele starts with 100% confidence, and penalties are applied when certain data characteristics were discovered throughout the genotyping process. Follows is a list of evidence used to best determine each allele's confidence level:

- If the First Order Differential peak confirmation stage required to re-run itself, with a lower threshold. More re-calls results in a higher penalty.
- Rare characteristics, such as homozygous haplotypes, or neighbouring/diminished peaks, incur a penalty.
- Atypical alleles are treated with more caution, and scores are weighted slightly more severely than typical alleles.
- Simple data aspects such as total read count within a sample/distribution/peak are used.
- Mapping percentages are taken into account, albeit as a minor factor within this algorithm.
- "Fatal" errors, such as Differential Confusion, incur a significant penalty.

Any confidence score is capped at 100%. If the quality of data in a particular sample is high enough for alleles to be awarded a confidence score higher than 100%, they are reported as 100%, regardless. Generally, a 'good' score is anything over 80%, and we have found that samples returning a score of over 60% are considered believable. Anything less than this may justify manual inspection.

1.16 Version ChangeLog

As this software is in continual development, changes will be listed here. Typically a new release will consist of a collection of minor bugfixes, but occasionally we have to release major overhauls to modules, include new features, or release hotfixes because I was hungry and mistakenly pushed an update without thorough enough testing. Don't write code at lunchtime.

1.16.1 Version 1.0

- Python 3.7 port
- Goodbye!

1.16.2 Version 0.324.2

- Hotfix for demultiplexing I/O bug that was introduced somehow? (update BatchAdapt if you use demultiplexing)
- Hotfix for genotyping zygosity bug where predictions were not correctly overruled by heuristics
- Minor bug fix with HTML templates rendering incorrect version strings / missing templates
- Hotfix for my dumbass deployment script for updates breaking version strings

1.16.3 Version 0.324

- Futher minor bugfixes regarding deployment of build scripts incorrectly providing sources in 0.323.

1.16.4 Version 0.323

- Minor bugfix for SNP calling where data was in an unexpected vector shape
- Minor bugfix for HTML report generation where certain exceptions were preventing incorrect data scraping

1.16.5 Version 0.322

- Removed novel atypical flag indicator from sequences with no intervening sequence at all
- Added alignment statistics to default ScaleHD output for samples which aligned, but could not be processed further
- Swapped standard HTML summary in genHTML for javascript element (filterable etc)
- Wrote brief help section on genHTML output
- Fixed some minor genotyping bugs with rare, atypical structures
- Fixed un-prompted read count subsampling in samples with atypical allele structures
- Fixed MSAViewer alignment showing certain reads off-position by 1 base pair
- Improved genHTML handling of failed samples (more information as to why, within detailed view)
- Removed choice between SNP calling algorithms; freebayes used exclusively

1.16.6 Version 0.321

- Fixed some syntax errors with array handling due to a dependency update changing interactions
- Added –simple flag for command line interface, providing a more literally-interpretable genotyping outputs
- Fixed minor demultiplexing error (path finding)
- Added entire HTML5 based output, extracting information from ScaleHD instance objects

1.16.7 Version 0.320

- Updated dependencies to latest versions (see _sect_reqpack)
- Minor tweaks to (syntax) interaction with updated versions of dependencies
- Fixed Matplotlib font missing warning spam on certain systems
- Fixed SKLearn ConvergenceWarnings spam
- Fixed Samtools memory block merging spam

1.16.8 Version 0.318

- Minor distribution scraping errors for homozygous haplotypes
- Logging bugfix with file going missing because i'm bad at my job
- SNP Calling masking for ScaleHD-ALSPAC
- Framework for simplified 95%C.I. output (feature not implemented in this version; undergoing testing)

1.16.9 Version 0.317

- Minor genotype graph render bugfixes
- Added file I/O of u.x. stdout log for easier troubleshooting
- Fixed minor bugs to do with SNP calling I/O paths and me being a bad programmer when hungry
- Added sanitisation stage to check for a user attempting to demultiplex files which have already been demultiplexed
- Minor tweaks for Windows 10 Linux Subsystem support
- Refactoring config backend interpreter to make it less dumpster-fire-awful

1.16.10 Version 0.316

- Added some minor documentation for SNP Calling (_sect_genotyping)
- Heuristic allele filtering engine has been completely rewritten to not be absolute garbage.
- Parallelised the DSP module within ScaleHD to execute on multiple contigs of data at once, if enabled.
- Parallelisation introduced issue with allele structure incrementing objects would behave improperly – this is now fixed.
- Disabled subsampling of aligned assemblies (due to multi-threading speedup; no longer required).
- Implemented broad error catching around SNP calling libraries, instead of just exiting upon failure.
- Fixed bug with PDF rendering of result distributions utilising an incorrect value for aligned read counts.
- Fixed bug where atypical alleles which changed from CCG-homozygous to CCG-heterozygous were not identified.
- Fixed error where the heuristic filtering engine suspects an expanded allele, but ended up calling a homozygous haplotype.
- Casting issue where two alleles returned different dimension-shaped arrays for FOD genotype calling, was resolved.

1.16.11 Version 0.314/5

- Fixed homozygous haplotype casting error
- Fixed diminished alleles being skipped (or not flagged) in particular cases of read drop-off in homozygous expansions

1.16.12 Version 0.313

- Fixed a rare error wherein graphs would not be rendered where an atypical allele rewrote the CCG-zygosity from heterozygous to homozygous.
- Added a flag for when the two core genotyping algorithms cannot agree on the status of one allele; this manifests as an expanded allele being missed due to significantly low read count.
- Allele sorting algorithm has been tweaked to correct some mistakes in my garbage code.
- Fixed rare error where FastQC would be executed on incorrect data.
- Fixed certain genotyping flags being applied on a sample wide basis as opposed to an individual allele basis.

1.16.13 Version 0.312

- Added an additional (optional) pre-processing stage, including sequence demultiplexing via Batchadapt.
- CCG First order differential bugfix in situations where peak-calling returned multiple variables when unexpected.
- Added Batchadapt to the required python package list for ScaleHD. Installed automatically from PIP where possible.

1.16.14 Version 0.311

- Moron hotfix for dumb reverse aggregate distribution bug I introduced with v0.310

1.16.15 Version 0.310

This is a minor update to ScaleHD. SNP calling implementation is now in alpha.

- Fixed a bug where genotyping would complete, but raise an exception at the end of the genotyping module, due to particular arrays not being flattened.
- Implemented Picard/GATK/Freebayes into the SNP calling module of ScaleHD.
- Added PyVCF as a Python library requirement for scraping data from variant calls.
- Modified the requirements for Picard/GATK to be integrated with ScaleHD on the user's system $PATH.
- Added Freebayes to the list of required binaries in __backend; addition user $PATH check
- Added new XML flag for user to specify a strictness value, for determining legitimate SNP calls.
- Minor codebase re-arranging in preparation for Digital Signal Processing to be replaced by a c++ binary, for performance.

1.16.16 Version 0.300

We now consider version 0.300 a "release-candidate alpha", if such a thing exists. I.E. The functionality performs as desired, 99% of the time (figure not accurate and i am not legally liable for any repercussions of assuming ScaleHD is 99% accurate haHAa). From this point onwards, new releases will contain new features, or a large collection of bug fixes. Minor iterations are (hopefully) over.

- Removed Rpy2 and R-interface codebase in preparation for switching bayesian confirmation model to a native python library.
- Added additional flag for ScaleHD output, describing how many reads that mapped to multiple references were removed (if enabled by the user).
- Switched output rendering pipeline from Prettyplotlib to Seaborn (PPL is no longer supported).
- Minor backend modifications in relation to the above.
- SKLearn deprecation on label encoder fixes
- Minor genotyping fixes (thresholds)

1.16.17 Version 0.252

- Modified the N-Aligned distribution logic to utilise pre-smoothing data distribution as opposed to post-smoothing.
- Bugfix with label in (a)typical allele being assigned an estimated CAG attribute which was not an integer.
- FastQ subsampling workflow modified to remove possibility of incorrect percentages applying to genotyping confidence.
- Fixed the algorithm which calculates Somatic Mosaicism for each allele (i.e. no longer reading from incorrect attributes).
- Some other stuff that I forgot.

1.16.18 Version 0.251

- Removed the redundant workflow codebase for Assembly processing (i.e. using BAM as input; feature not required/desired anymore).
- Refactored the input method that the user can specify to subsample input reads, or not.
- Scope fix for instances that do not use SeqQC.
- Alternative shell pathing check for requisite binaries fix (e.g. using zsh instead of bash)

1.16.19 Version 0.250

- CCG distribution cleanup threshold tweaks
- Added handler for atypical-typical 50:50 read ratio assembly contigs.
- Added a threshold context manager for Neighbouring Allele Peak algorithm.
- Added differential confusion flag for samples which ScaleHD cannot sort via heuristics.
- Begun to implement Polymorphism detection..

1.17 Planned Features

Here, future plans for ScaleHD are listed. These include new features to the pipeline, as well as improvements to existing features.

- Re-implement DSP module in a compiled language for speed improvements.
- Development of a Electron + Node.js GUI frontend.
- Development of a novel data output format for rapid in-depth data sharing.
- Tidy up the spaghetti code, oh my godddddd.