MIP_api Documentation

Release 4.0.0

Måns Magnusson, Robin Andeer

December 19, 2016

Contents

MIP

Mutation identification pipeline

Release 4.0.

MIP is a pipeline for clinical analysis of whole exome and whole genome sequence data.

Contents:

MIP enables identification of potential disease causing variants from sequence data.

CHAPTER 1

Overview

MIP performs whole genome or target region analysis of sequenced single-end and/or paired-end reads from the Illumina platform in fastq(.gz) format to generate annotated ranked potential disease causing variants. MIP performs QC, alignment, coverage analysis, variant discovery and annotation, sample checks as well as ranking the found variants according to disease potential with a minimum of manual intervention. MIP is compatible with Scout and Puzzle for visualization of identified variants. MIP has been in use in the clinical production at the Clinical Genomics facility at Science for Life Laboratory since 2014.

Features

- **Installation**
  - Simple automated install of all programs using conda/SHELL via supplied install script
- **Autonomous**
  - Checks that all dependencies are fulfilled before launching
  - Builds/Prepares/downloads references and/or files missing before launching
  - Decompose and normalise reference(s) and variant vcf(s)
  - Splits and merges files/contigs for samples and families when relevant
- **Automatic**
  - A minimal amount of hands-on time
  - Tracks and executes all module without manual intervention
  - Creates internal queues at nodes to optimize processing
  - Minimal IO between nodes and login node
- **Flexible:**
  - Design your own workflow by turning on/off relevant modules
  - Restart an analysis from anywhere in your workflow
  - Process one, or multiple samples using the module(s) of your choice
  - Supply parameters on the command line, in a pedigree file or via config files
  - Simulate your analysis before performing it
  - Redirect each modules analysis process to a temporary directory (@nodes or @login)
  - Limit a run to a specific set of genomic intervals
  - Use multiple variant callers and annotation programs
  - Optionally split data into clinical variants and research variants
- **Fast**
  - Analyses an exome trio in approximately 4 h
  - Analyses a X-ten sequenced genome in approximately 21 h
  - Rapid mode analyzes a WGS sample in approximately 4 h using a data reduction and parallelization scheme

- **Traceability**
  - Track the status of each modules through dynamically updated status logs
  - Recreate your analysis from the MIP log or generated config files
  - Logs sample meta-data and sequence meta-data
  - Logs version numbers of softwares and databases
  - Checks sample integrity (sex and relationship)
  - Test data output existens and integrity using automated tests
- **Annotation**
  - **Gene annotation**
    - Summarise over all transcript and output on gene level
  - **Transcript level annotation**
    - Separate pathogenic transcripts for correct downstream annotation
  - **Annotate all alleles for a position**
    - Split multi-allelic records into single records to ease annotation
    - Left align and trim variants to normalise them prior to annotation
  - Annotate coverage across genetic regions
  - Extracts QC-metrics and stores them in YAML format
- **Standardized**
  - Use standard formats whenever possible
- **Visualization**
  - Ranks variants according to pathogenic potential
  - Output is directly compatibel with Scout and Puzzle

CHAPTER 3

Example Usage

```
perl mip.pl -pMosaikBuild 0 -configFile 1_config.yaml
```

CHAPTER 4

Getting Started

4.1 Installation

MIP is written in Perl and therfore requires that Perl is installed on your OS (See Installation).

Change log (See Change Log)

4.2 Prerequisites

MIP will only require prerequisites when processing a modules that has dependencies (See Setup). However, some frequently used sequence manipulation tools e.g. samtools, PicardTools, Bedtools are probably good to have in your path.

4.2.1 Meta-Data

Meta data regarding the pedigree, gender and phenotype should be supplied for the analysis.

- Pedigree file (PLINK-format; See Pedigree File & MIP´s github repository).
- Configuration file (YAML-format; See Dynamic Configuration File & MIP´s github repository).

4.3 Usage

MIP is called from the command line and takes input from the command line (precedence), a config file (yaml-format) or falls back on defaults where applicable.

Lists are supplied as comma separated input, repeated flag entries on the command line or in the config using the yaml format for arrays.

Note: List or repeated entries need to be submitted with the same order for each element across all supplied lists.

Only flags that will actually be used needs to be specified and MIP will check that all required parameters and dependencies (for these flags only) are set before submitting to SLURM.

Program parameters always begins with “p” followed by a capital letter. Program parameters can be set to “0” (=off), “1” (=on) and “2” (=dry run mode). Any program can be set to dry run mode and MIP will create sbatch scripts, but not submit them to SLURM for these modules. MIP can be restarted from any module, but you need to supply previous dependent programs in dry run mode to ensure proper file handling.

MIP will overwrite data files when reanalyzing, but keeps all "versioned" sbatch scripts for traceability.

MIP allows individual target file calculations if supplied with a pedigree file or config file containing the supported capture kits for each sample.

You can always supply `perl mip.pl -h` to list all available parameters and defaults.

Example usage:

```
$ perl mip.pl -f 3 -sampleid 3-1-1A,3-2-1U -sampleid 3-2-2U -pFQC 0 -pMosaikBuild 2 -pMosaikAlign 2 -
```

This will analyze *family 3* using *three individuals* from that family and begin the analysis with programs *after MosaikAlign* and use all parameter values as specified in the *config file*, except those supplied on the command line, which has precedence.

Input

It is recommended to use MIP's naming convention for input Fastq files to accurately and automatically handel individual runs and lanes (See Setup).

Fastq files (gziped/uncompressed) should be place within the `-inFilesDirs` as `-inFilesDirs {PathToInFileDir}=sampleID`.

Note: MIP will automatically compress any non gzipped files if `-pGZip` is enabled. All files ending with .fastq or .fast.gz will be included in the run.

MIP scripts locations are specified by `-inScriptDir`.

All references and template files should be placed directly in the reference directory specified by `-referencesDir`, except for ANNOVAR db files, which should be located in *annovar/humandb*.

Output

Analyses done per individual is found under respective sampleID subdirectory and analyses done including all samples can be found under the family directory.

Sbatch Scripts

MIP will create sbatch scripts (.sh) and submit them in proper order with attached dependencies to SLURM. These sbatch script are placed in the output script directory specified by `-outScriptDir`. The sbatch scripts are versioned and will not be overwritten if you begin a new analysis. Versioned "xargs" scripts will also be created where possible to maximize the use of the cores processing power.

Data

MIP will place any generated datafiles in the output data directory specified by `-outDataDir`. All datatfiles are regenerated for each analysis. *STDOUT* and *STDERR* for each program is written in the *<program>/info* directory prior to alignment and in the *<aligner>/<program>info* directory post alignment.

Analysis Types

Currently, MIP handles WES `-at sampleID=wes`, WGS `-at sampleID=wgs` or Rapid analysis `-at sampleID=rapid` for acute patient(s).

The rapid analysis requires `BWA_MEM` and selects the data that overlaps with the regions supplied with the `-bwamemrdb` flag. MIP will automatically detect if the sequencing run is single-end or paired-end/interleaved paired-end and the length of the sequences and automatically adjust accordingly.

Note: In rapid mode; Sort and index is done for each batch of reads in the `BWA_Mem` call, since the link to infile is broken by the read batch processing. However `pPicardToolsSortSam` should be enabled to ensure correct fileending and merge the flow to ordinary modules.

Project ID

The `-projectID` flag sets the account to which core hours will be allocated in SLURM.

Aligner

Currently MIP officially supports two aligners Mosaik and BWA, but technically supports any aligner that outputs BAM files. Follow the instructions in Adding a new program to add your own favorite aligner.

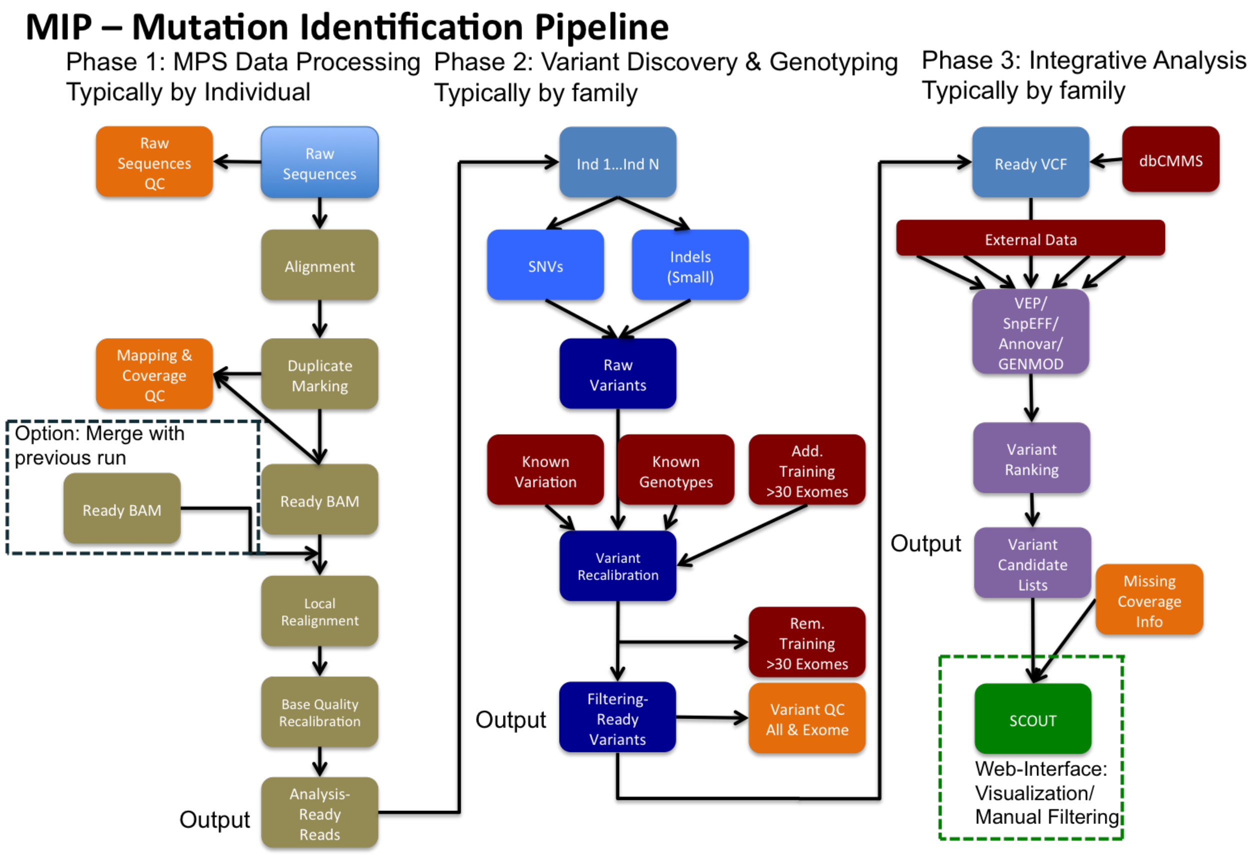
Log

MIP will write the active analysis parameters and *STDOUT* to a log file located in: `{OUTDIRECTORY}{FAMILYID}/{MIP_LOG}/{SCRIPTNAME_TIMESTAMP}`

Information, such as infile, programs, outdatafiles etc, for each analysis run is dynamically recorded in the a yaml file determined by the `-sampleInfoFile` flag. Information in the sampleInfo file will be updated in each analysis run if identical records are present and novel entries are added. The sampleInfo file is used in QCCollect to extract relevant qc metrics from the MPS analysis.

Pipeline WorkFlow

This is an example of a workflow that MIP can perform (used @CMMS).



CHAPTER 5

Change Log

MIP v3.0 –> v4.0

mip.pl

- Fixed chrY for female in SVRanking
- Fixed bug in SambambaDepth causing logging to be turned off
- Fixed bug causing MostCompleteBAM being incorrectly added when launching single module
- Fixed bug causing info for MarkDuplicates to not be logged to qc_sampleInfo
- Added flag for creating index file with GatherBAMFile. Default to TRUE
- Temporary fix allowing samtools to create index for GATKBaseRecalibration BAM instead of Picard to accomodate pileup.js
- Modified javaUseLargePages and reduceIO conditional statement to be implicitly boolean
- Added creation of directory for analysis config file unless directory already exists
- Validate all parameters
- Changed percentag_mapped_reads to percentage_mapped_reads
- Added Vcftools and Plink2 versions to qcmetrics
- Changed default for GATKReAligner to "0"
- **Added SLURM QOS to sbatch header**
  - Allowed values: [low, normal, high]. Defaults to "normal".
- Add program processing Markduplicates metric to qc_sampleInfo as "ProcessedBy" for log purpose
- Remade array parameter inFilesDirs to hash parameter inFilesDir
- Added CLNREVSTAT to config
- **Changed analysisType from scalar to hash (-at sampleID=analysisType)**
  - Default genomes
  - Can be supplied from pedigree using "Sequence_type"
  - Enables performing analysis under correct fastq location and with correct parameters for sample specific analyses
  - Changed default output structure
  - FamilyID is now root

    - Data is stored under root in analysisType dir
    - pedigree is stored under root
    - scripts, mip_log, config and qc_sampleInfo are stored under FamilyID/analysis
- Changed "exomes" and "genomes" to "wes" and "wgs" respectively
- Added new baitset shortcut in pedigree: Agilent_SureSelectFocusedExome.V1.GRCh37_targets.bed
- Modified UpdateToAbsolutePath to get info from definitions directly
- **Move all rio BAMProcessing to same in and out directory to enable skipping of programs**
    - Modules that variant callers and qc modules output under their own dir, but gathered data is processed under alignment directory
- Call sub in RankVariants for select file and adjust MT|M
- Changed bcf generation to vcf.gz generation + tabix index
- Changed rankedBCFFile to rankedBinaryFile
- Changed svRankedBCFFile to svRankedBinaryFile
- Changed tag in qc_SampleInfo from BCFFile to VCFBinaryFile and VCFSVBCFFile to SVBinaryFile
- Change to tabix from bcf since it seems more forgiving and produce similar compression level
- Added "—rank_results" in genmod score for producing rank view in Scout
- Remove rank score sorting of clinical/ research
- Add bwa log and HLA log to qc_sampleInfo and copying back to hds
- Added decomposing using bcftools for variant callers and normalization for GATKVariantRecalibration
- **GRCH38 Error: Error: Field name 'phyloP46way_primate' not found dbSnpEff**
    - phyloP46way_primate -> phyloP20way_mammalian
    - phyloP100way_vertebrate -> phyloP7way_vertebrate
    - phastCons46way_primate -> phastCons20way_mammalian
    - phastCons100way_vertebrate -> phastCons7way_vertebrate
- Added novel calculation of F-score using plink2
- Add CLNVAR curation status (CLNREVSTAT) into rank model and bumped rank_model to version 1.17
- Added —disable_auto_index_creation_and_locking_when_reading_rods to GATKReAlign and GATKBaserecal
- Added SWEREF into rank model and to default (NOTE: only car 22 pending actual release)
- Added 'variant integrity' to sampleCheck
- Modifed InbreedingFactor regExp for plink2 .het output
- Added module to split fastq files, move original files to sub dir and then exit
- Exchanged 'vars' for our
- Made Path and outdataDirection and outDataFile be collected by recursive strategy
- Let MIP detect if no affected is in pedigree and warn and turn of genmod models, score and compound
- Updated VTref switch to avoid modifying same reference twice
- Updated rank model to locate SIFT and PolyPhen from CSQ-field directly

- Added LoFtool gene intolerance prediction to rank model
- Updated rank_model to version 1.18
- Added additional sorting of SV variants after vcfanno annotation
- Added extra sort of SV variants after ranking
- Process SV exome|rapid as one file instead of splitted per contig to ensure that no contig will lack variants
- Added insert qc metrics to qc_sampleInfo and qcmetrics
- **Added support for interleaved fastq files and relaxed file convention criteria**
    - Interleaved info is gathered from fastq header for read direction 1
    - BWAMem alignment is automatically adjusted accordingly
    - MIP now supports mixing of SE, PE and PE-interleaved files within the same fastq directory
    - Relaxed file convention criteria by collecting mandatory info from fastq header - just require sampleID in filename
    - Add fake date since this information is not recorded in fastq header for standardised file spec
- Added support for metadata in yaml format (pedigree and other meta data)
- MIP will look at the filending to detect file format
- Set mandatory keys in Plink pedigree format to be lower case throughout in MIP output
- Added additional reformating of yaml value for "sex" and "phenotype" under new keys to adhere to Plink format when required
- Modified and added pedigree templates
- Removed instanceTag and researchEthicalApproved
- Added temp_dir to genmod annotate and filter in sub VT
- Modfied pedigree file to genmod calls to use famFile and changed default to 'ped'
- Modified pedigree file default ending to ".yaml"
- **Modified qcCollect to new yaml structure**
    - Added sampleID level for evaluate
    - Cleaned up code
- Added cut-offs for evaluation of mendel and father
- Added collection of expected_coverage from ped.yaml and relay to qcCollect for evaluation
- **Removed extra feature annotations and some VEp field parsing**

    svVcfParserRangeFeatureAnnotationColumns:
    - 3 = Ensembl_gene_id - REMOVED
    - 4 = HGNC_symbol
    - 5 = Phenotypic_disease_model - REMOVED
    - 6 = OMIM_morbid - REMOVED
    - 7 = Ensembl_transcript_to_refseq_transcript - REMOVED
    - 8 = Gene_description - REMOVED

    svVcfParserSelectFeatureAnnotationColumns:

    - 3 = HGNC_symbol
    - 10 = Phenotypic_disease_model - REMOVED
    - 11 = OMIM_morbid - REMOVED
    - 14 = Ensembl_gene_id - REMOVED
    - 16 = Reduced_penetrance - REMOVED
    - 17 = Clinical_db_gene_annotation - REMOVED
    - 18 = Disease_associated_transcript - REMOVED
    - 19 = Ensembl_transcript_to_refseq_transcript - REMOVED
    - 20 = Gene_description - REMOVED
    - 21 = Genetic_disease_model - REMOVED
  - Removed additional VEP parsing: - GeneticRegionAnnotation - HGVScp - INTRON - EXON - STRAND - HGVSc - HGVSp
- Added GBCF file creation and key-path to qc_sampleInfo
- Added pedigree_minimal (.fam file) and config_file_analysis to qc_sample_info
- Add test of SV files in analysisrunstatus
- Expect select file to have full path and not be located in MIP reference directory
- Moved sacct module to case level

Install.pl

- Added boolean flag condaUpDate and changed flag perlInstall to boolean
- Renamed preferBioConda to preferShell and made it boolean
- Renamed flag update to noUpdate and made it boolean
- Activated CNVnator installation
- Added install script to conda env for printing software versions connected to MIP version
- Added Validate parameter checks, named arguments and sub description
- Updated genmod to version 3.5.6
- Added ability to set python version when creating conda env
- Updated chanjo to v4.0.0

vcfParser.pl

- Removed Sift and Polyhen parsing from CSQ field
- Change SYMBOL to HGNC_ID in vcfparser
- Added per_gene option

qcCollect.pl

- Changed percentag_mapped_reads to percentage_mapped_reads
- Added raw total sequences and reads mapped to qcCollect
- Added Vcftools and Plink2 versions to qcmetrics
- Updated regExp file to version 12

MIP v2.6 –> v3.0

- Added Net/SSLeay.pm to install.pl
- Added option to skip perl install
- Added Manta, Delly, FT and CNVnator as structural variant callers
- Added modules CombineStructuralVariants, SVVariantEffectPredictor, SVVCFParser, SVRanking
- Added merging of samples in "other" chains to family chain for parallel modules
- Added CNVnator version. Had to be done at start up since CNVnator does not add its version to the output.
- Added Delly version on sample level
- Added Manta version on sample level
- Added SVVEP version and cache version
- Fixed bug causing VEP version to be lost for snvs/indels
- Added SVVCFParser version
- Added SVGenmod/rankModel version
- Added test for GATKCombineVariantsPrioritizeCaller to not include turned of variant callers
- Added snpEff download of reference genome to avoid race conditions
- Fixed python virtuelenvironment to not check programs if uve = 0
- Added VEP/SVVEP assembly, gencode, gene build, HGMDPublic, polyphen, regbuild, Sift, version to qcmetrics
- made NIST ID settable
- Removed PicardMergeSwitch, now all files are merged or renamed (single files) for more consequent naming and easier processing
- Renamed 'fileEnding' to 'fileTag' and 'removefileEnding' to 'fileEnding'
- Change name of BAMCalibrationAndGTBlock to only BAMCalibrationBlock
- List::Util is in core module perl 5.18 replaces List::MoreUtils
- Use say instead of print where relevant
- Use internal Perl system commands instead of UNIX (copy, make_path)
- Removed mip log file if present in config to avoid appending to old log file. Supply on log file on cmd if you want to append to log file.
- Added plink2 installation via bioconda in install script
- Changed binary i MIP from plink to plink2
- Added MultiQC in install script and as MIP module
- Changed samtools stats module to include complete report for MultiQC processing
- Made pPicardToolsMergeSamFiles mandatory: Always run even for single samples to rename them correctly for standardised downstream processing. Will also split alignment per contig and copy to temporary directory for '-rio 1' block to enable selective removal of block submodules.
- Added LOFTEE VEP plugin: https://github.com/konradjk/loftee
- Added LofTool VEP plugin
- Added Modern::Perl '2014'

- Added PERL_UNICODE=SAD to install script, and hence bash_profile - stdin, stdout, and stderr to UTF8 as well as @ARGV and data handlers
- Use UTF-8 for all source script
- Added encoding UTF-8 pragma for open to default expect unicode when opening and writing
- Enforce perl 5.18 version
- Added autodie for generalised error and exception handling
- Removed dateTime and use less cumbersome core module Time::Piece
- Removed DV and added AD for samtools mpileup
- Added joint calling of SV using Manta
- Added SV analysis of exomes using Manta
- Modified CombineSVVariants to use Delly and CNVnator on sample-level and Manta on family level
- Added bcf generation of ranked vcf both select file and research
- Fixed bug in temp directory
- Bumped install version to 3.5.1
- Added Genmod temp dir flag
- Added sacct commands to trap for each sbatch to relay progress to MIP log file.status
- Made Sacct dependency into afterany
- Added pPrepareForVariantAnnotationBlock
- Removed pythonVirtualEnv and commands as conda is prefered
- Added sourceEnvironmentCommand
- Added '-pp' and '-ppm'
- Add bcf conversion of select and research variants to MIP
- Added check of programs mode to allowed values, more strict parsing for flaggs expecting numbers
- **Select variants prior to Plink processing using GATK Select variants**
  - Move processing to node, but keept final output printing to hds
- **Added SV annotation using 1000G SV and vcfanno -ends**
  - Added vcfanno, lua, config
  - Annotate from 1000G SV
  - Modified svrank_mdodel to take 1000G frequency in account
- Add vcfanno version in SVCombineVariantCallSets
- Updated fastqc to version 0.11.5
- Updated bwa to version 0.7.13
- **Updated sambamba to version 0.6.1**
  - Added "—fix-mate-overlaps" to avoid counting overlapping reads twice
  - Removed Sambamba version from MIP flagg
- Updated picardtools to version 2.3.0

- Updated Chanjo to version 3.4.1
- Updated Manta to version 0.29.6
- Updated Genmod to version 3.5.2
- Updated MultiQC to version 0.6
- Updated Vip to version 84
- Added Picardtools Markduplicates as a option and default
- Added more SambambaMarkDup options
- Make sambamba flagstats into subroutine to be used for all markdup
- Remade capture kit options into 1 hash flag, which will build all associated files if 1 is lacking
- **Make Covariates to be used in the recalibration in GATKBASERCAL to be flag and array option**
  - annotations, -Know and -knownSites
- Remade VEP install assembly flag to be array and used rerun install for each assembly version
- Remade SnpEff install genomeVersion flag to be array and used rerun download for each genome version
- Added assembly flag to VEP script and alias it to use GRCh prefix and number
- Fixed chr prefix for chanjo sex check
- Updated to GATK version 3.6
- Created contig splitted target files on the fly for non genome analysis to reduce the running time of GATK Realign, BaseRecal and Haplotype
- Added sub ReplaceIUPAC and used it on freebayes and samtools mpileup vcfs
- Changed analysisType default from exomes to genomes

MIP v2.4 –> v2.6

- Updated GATK to 3.5
- Added static binning capability for base recalibration (BQSR)
- Added option –disable_indel_quals to BSQR
- Added limit for exomes to only use target bases in recalibration
- Added MTAF to SnpEff and vcfParser for MT frequency annotations
- Added ‘trio’ detection to parameters instead of scriptParameters to avoid writing key to config
- Fixed bug when supplying -sambambaDepthCutOffs on cmd
- MIP now handles updating to absolute path for comma separated parameters correctly
- Removed write to cmd string in mip log for some internal parameters
- **Updated install script**
  - Added PIP to the condo env upon creation
  - Add check that condo is executable in system before launching rest of installation
  - Install script can now detect existing condo env and change cmd to accommodate
  - Added sambamba (0.5.9), vt (2015.11.10), bedtools (2.25.0), htslib (1.2.1) to bioconda install
  - Added option to prefer Bioconda install over shell for overlapping modules

  - Added soft link creation sub routine
  - Use soft link sub for sambamba (both bioconda and Shell)
  - Add soft link to snpEff och SnpSift for bioconda install
- Update FASTQC to 11.4 via bioconda
- Updated SnpEff to v4_2 via Shell
- Updated Plink to v1.90b3.26 64-bit (26 Nov 2015) via shell
- Updated vcfTools to 0.1.14 via SHELL
- Updated Chanjo to 3.1.1 via PIP
- Updated Genmod to 3.4 via PIP
- Updated Picardtools to 1.141 via bioconda
- Updated Samtools to 1.3
- Updated bcfTools to 1.3
- Updated htslib to 1.3
- Added picardTools installation via SHELL
- **Updated VEP to 83 via SHELL**
  - Trouble with distribution - htslib and sereal (only issues with testing and not with actual running the script)
  - Added installation of VEP plugin UpDownDistance
- Added use of VEP plugin UpDownDistance for MT contig only to avoid over annotation of the compact MT genome
- Added padding to 10 nucleotides for MT in Vcfparser
- Added test for undetermined in fastq file name and adjust qc-test to skip entirely for these reads
- Added samtools mpileup
- Added GATKCombineVariants to combine variants calls from multiple variant callers
- Added generalisation for supporting multiple variant callers in MIP dependencies and GATKCombinaVariants
- Added no-fail to sample check
- Modified installation of picardTools and SnpEff
- Add filtering to variant calls from samtools mpileup
- Add samtools/bcfTools versions
- Add removal of samtools pileup files
- Added test::Harness for TAP summary results and future inclusion of additional test scripts
- Add option to determine priority in variant callers as comma sep string
- Add check of variant callers active compared to prioritise flag
- Add sanity check of prioritisation flag
- Add option to turn on or off installation of programs in install.pl
- Added bcf file compression and indexing as sub
- Added vcfTobcf sub to GATKCombineVariants

- Switched vcf ready file from GATKVariantRecalibration to GATKCombineVariants
- **Added Freebayes variant caller**
  - Added to removeRedundantFiles
  - Added Freebayes version to qcCollect
- RemoveRedundant files info is now recorded in definition.yaml
- Added GATKCombineVariants to removeRedundant files
- Add bcftools norm to samtools pileup and freebayes output
- Add lastlogFilePath to qc_sampleInfo
- Made lanes and readDirections info more nested
- Add 1000G Phase 3 and Exac to Genmod annotation
- Changed regEx in test.t to include all until "," for INFO fields in Header
- Modified bioconda softlinks sub call to only execute if programs are installed
- Added MT.codon table sub for snpEff config to install script
- Remake GENMOD CADD file option to array
- Added padded target intervals to exome analysis again for GATKRealign and GATKHaplotypeCaller
- Reactivate GATKPaddedTarget parameter
- Made associatedPrograms arg into an array instead of a comma sep string
- Fixed check for when a capture kits is lacking from input and fallback to using "latest"
- Remade CheckParameterFiles to work with DataType
- Add evaluation with NIST as a module in MIP
- Fix the . mip.sh to bash mip.sh
- Added reference to define/definitions
- CheckParameterFiles now works with parameterExistsCheck directly instead of "d" and "f" enabling merge of directory and file sections
- Changed if for intervalListFile to be if($IntervalList) instead of analysisTypeExome|rapid
- Add programType=Aligner to define/definitions
- Remade sanity check of aligner to count if more than 1 aligner has been switched on (MosaikAligner, BWASampe, BWAMEM)
- Dynamic setting of 'aligner' depending in aligner supplied by outDirectoryName
- Renamed aligner to alignerOutDir
- Added genmod max_af
- Added canonical to VEP features

MIP v2.0 –> v2.4

- Bugfixes
- **Updated most program version (see docs) and databases**
  - Logs versions and databases
- Added -pVT

- Added -allSites option to GenoTypeVCFs
- Added version tag to definitions.yaml
- Cleaned some old parameter names
- Added test for parameter compatibility between defineParameters.yaml and config
- Added new parameter snpSiftAnnotationOutInfoKey
- Changed **SnpSift_** for 1000G and EXACAF to facilitate downstream processing since both work on KEY=AF
- Remade dbsnpAF parsing to accommodate multiple entries for the same env
- Added vt decompose and normalise subroutine for both reference and variant vcf
- Removed vcf_parser —split
- MIP now works only on config tags from select file meta data header for select genes
- Added genmod version and removed RankVariants version
- Add test for VEP cache and directory version linked
- Added option OverclippedReadFilter to GATKBaseRecalibration/PrintReads
- Exchange grep for any in array check and use eq instead of // for stringency
- Added vt decompose and normalise subroutine call for relevant downloadable references ("indels", "mills", "dbsnp", "hapmap", "dbsnpex", "1000g_snps")
- Add check for ingoing references that VT has been used if VT is on
- Fixed bug in AnalysisRunStatus modules caused when first processing -rio 1 and then -rio 0
- Fixed bug when adding samples to pedigree to already processed samples
- Removed Radial:sw and LR_score from dbNSFP annotation as these have become obsolete
- Remade RemoveRedundant files
- Added bcf compression alternativ
- Added perl oneliner to VT that removes '*' alt.allele after decomposing as it does not add any new info

MIP v1.0 –> v2.0

- Major code refactoring
- Bugfixes
- **Updated most program version (see docs) and databases**
    - Logs versions and databases
- **Removed modules -pMerge_anvar, -pAdd_dp**
    - MIP no longer creates master templates, instead this is taken care of dynamically
- Added -pVeP, -pSnE, -pVcP -pChanjoSexCheck
- Module PicardSortSam is now integrated in alignment modules
- **Use VCF format where appropriate**
    - Created standardised VCF list levels (",", ":", "|")
- **Clinical transcripts are selected after VEP annotation using VCFParser**
    - Removes ethical issue with overlapping genes

- **Full resolution in annotation**
  - Gene
  - Transcripts
  - Multiple alleles
  - Split multi allelic calls into single records
  - Use SO terms
  - Calculate Sift an PolyPhen per transcript and allele
  - Remade transcript and cDNA and protein info from VEP CSQ field
  - Switched from MAF to AF
- Use Log4Perl for logging
- All processes create temp directory on (default @nodes)
- Creates automatic migration to and from nodes
- Deploy more aggressive scatter/gather technique. Processing per contig whenever possible.
- Analyse order in contig size not number
- Use piping in SnpSift annotation and where possible
- **Reduce IO between nodes using -rio flag. Will run modules sequentially where appropriate.**
  - Created automatic removal of files when appropiate at tempDir
- **Flag changes**
  - -huref/–humanGenomeReference –> -hgr/–humanGenomeReference
  - -rea/–researchEthicalApproval Tag for displaying research candidates in Scout (defaults to “notApproved”)
  - -tmd/–tempDirectory Set the temporary directory for all programs (defaults to “/scratch/SLURM_JOB_ID”;supply whole path)
  - -nrm/–nodeRamMemory The RAM memory size of the node(s) in GigaBytes (Defaults to 24)
  - -ges/–genomicSet Selection of relevant regions post alignment (Format=sorted BED; defaults to “”)
  - -rio/–reduceIO Run consecutive models at nodes (defaults to “0”)
  - -l/–logFile Mip log file (defaults to “{outDataDir}/{familyID}/mip_log/{timestamp}/{scriptname}_{timestamp}.log”)
  - -pGZ/–pGZip –> -pGZ/–pGZipFastq
  - -pFQC/–pFastQC –> -pFqC/–pFastQC
  - -moaannpe/–mosaikAlignNeuralNetworkPeFile –> -moaape/–mosaikAlignNeuralNetworkPeFile
  - -moaannse/–mosaikAlignNeuralNetworkSeFile –> -moaase/–mosaikAlignNeuralNetworkSeFile
  - -pBWA_mem/–pBwaMem –> -pMem/–pBwaMem
  - -bwamemrdb/–bwaMemRapidDb –> -memrdb/–bwaMemRapidDb
  - -pBWA_aln/–pBwaAln –> -pAln/–pBwaAln
  - -pBWA_sampe/–pBwaSamp –> -pSap/–pBwaSampe
  - -picardpath/–picardToolsPath –> -ptp/–picardToolsPath
  - -picttmpd/–PicardToolsTempDirectory –> removed

- -pPicT_sort/–pPicardToolsSortSam –> removed
- -pPicT_merge/–pPicardToolsMergeSamFiles –> -pPtM/–pPicardToolsMergeSamFiles
- -pPicT_mergerr/–pPicardToolsMergeRapidReads -> -pPtMR/–pPicardToolsMergeRapidReads
- -picT_mergeprev/–picardToolsMergeSamFilesPrevious –> -ptmp/–picardToolsMergeSamFilesPrevious
- -pPicT_markdup/–pPicardToolsMarkduplicates –> -pPtMD/–pPicardToolsMarkduplicatesWithMateCigar
- -pCh_B/–pChanjoBuild –> -pChB/–pChanjoBuild
- -pChS/–pChanjoSexCheck
- -pCh_C/–pChanjoCalculate –> -pChA/–pChanjoAnnotate
- -chccut/–chanjoCalculateCutoff –> -chacut/–chanjoAnnotateCutoff
- -pCh_I/–pChanjoImport –> -pChI/–pChanjoImport
- -pCC_bedgc/–pGenomeCoverageBED –> -pGcB/–pGenomeCoverageBED
- -xcov/–xCoverage –> -gcbcov/–GenomeCoverageBEDMaxCoverage
- -pCC_picmm/–pPicardToolsCollectMultipleMetrics –> -pPtCMM/–pPicardToolsCollectMultipleMetrics
- -pCCE_pichs/–pPicardToolsCalculateHSMetrics –> -pPtCHS/–pPicardToolsCalculateHSMetrics
- -extbl/–exomeTargetBedInfileLists –> -ptchsetl/–exomeTargetBedInfileLists
- -extpbl/–exomeTargetPaddedBedInfileLists –> -ptchsetpl/–exomeTargetPaddedBedInfileLists
- -pRCP/–pRCovPlots –> -pRcP/–pRCovPlots
- -gatkpath/–genomeAnalysisToolKitPath –> -gtp/–genomeAnalysisToolKitPath
- -gatkbdv/–GATKBundleDownLoadVersion –> -gbdv/–GATKBundleDownLoadVersion
- -gatktmpd/–GATKTempDirectory –> removed
- -gatktpbl/–GATKTargetPaddedBedIntervalLists –> -gtpl/–GATKTargetPaddedBedIntervalLists
- -gatkdcov/–GATKDownSampleToCoverage –> -gdco/–GATKDownSampleToCoverage
- -pGATK_real/–pGATKRealigner –> -pGrA/–pGATKRealigner
- -gatkrealknset1/–GATKReAlignerINDELKnownSet1 –> -graks1/–GATKReAlignerINDELKnownSet1
- -gatkrealknset2/–GATKReAlignerINDELKnownSet2 –> -graks2/–GATKReAlignerINDELKnownSet2
- -pGATK_baserecal/–pGATKBaseRecalibration –> -pGbR/–pGATKBaseRecalibration
- -gatkbaserecalknset/–GATKBaseReCalibrationSNPKnownSet –> -gbrkse/–GATKBaseReCalibrationSNPKnownSet
- -pGATK_hapcall/–pGATKHaploTypeCaller –> -pGhC/–pGATKHaploTypeCaller
- -gatkhapcallsnpknset/–GATKHaploTypeCallerSNPKnownSet –> -ghckse/–GATKHaploTypeCallerSNPKnownSet
- -pGATK_genotype/–pGATKGenoTypeGVCFs –> -pGgT/–pGATKGenoTypeGVCFs
- -gatkgenotyperefgvcfinfile/–GATKGenoTypeGVCFsRefGVCFInfile –> -ggtgrl/–GATKGenoTypeGVCFsRefGVCF

- -pGATK_varrecal/–pGATKVariantRecalibration –> -pGvR/–pGATKVariantRecalibration
- -gatkexrefsnp/–GATKExomeReferenceSNPs –> -gvrtss/–GATKVariantReCalibrationTrainingSetDbSNP
- -gatkvarrecaltrhapmap/–GATKVariantReCalibrationTrainingSetHapMap –> -gvrtsh/–GATKVariantReCalibrationTrainingSetHapMap
- -gatkvarrecaltrd1000Gsnp/–GATKVariantReCalibrationTrainingSet1000GSNP –> -gvrtsg/–GATKVariantReCalibrationTrainingSet1000GSNP
- -gatkvarrecaltromni/–GATKVariantReCalibrationTrainingSet1000GOmni –> -gvrtso/–GATKVariantReCalibrationTrainingSet1000GOmni
- -gatkvarrecaltrdbmills/–GATKVariantReCalibrationTrainingSetMills –> -gvrtsm/–GATKVariantReCalibrationTrainingSetMills
- -gatkvarrecaltsfilterlevel/–GATKVariantReCalibrationTSFilterLevel –> -gvrtsf/–GATKVariantReCalibrationTSFilterLevel
- -gvrevf/–GATKVariantReCalibrationexcludeNonVariantsFile
- -gvrsmr/–GATKVariantReCalibrationSpliMultiRecord
- -pGATK_phaseTr/–pGATKPhaseByTransmission –> -pGpT/–pGATKPhaseByTransmission
- -pGATK_readPh/–pGATKReadBackedPhasing –> -pGrP/–pGATKReadBackedPhasing
- -gatkreadphphaseqthr/–GATKReadBackedPhasingPhaseQualityThresh –> -grpqth/–GATKReadBackedPhasingPhaseQualityThreshold
- -pGATK_varevalall/–pGATKVariantEvalAll –> -pGvEA/–pGATKVariantEvalAll
- -pGATK_varevalexome/–pGATKVariantEvalExome –> -pGvEE/–pGATKVariantEvalExome
- -gatkvarevaldbsnp/–GATKVariantEvalDbSNP –> -gveedbs/–GATKVariantEvalDbSNP
- -gatkvarevaldbgold/–GATKVariantEvalGold –> -gveedbg/–GATKVariantEvalGold
- -pANVAR/–pAnnovar –> -pAnV/–pAnnovar
- -anvarpath/–annovarPath –> -anvp/–annovarPath
- -anvargbv/–annovarGenomeBuildVersion –> -anvgbv/–annovarGenomeBuildVersion
- -anvartn/–annovarTableNames –> -anvtn/–annovarTableNames
- -anvarstn/–annovarSupportedTableNames –> -anvstn/–annovarSupportedTableNames
- -anvarmafth/–annovarMAFThreshold –> -anvarmafth/–annovarMAFThreshold
- -pVeP/–pVariantEffectPredictor Annotate variants using VEP (defaults to "1" (=yes))
- -vepp/–vepDirectoryPath Path to VEP script directory (defaults to ""; supply whole path)
- -vepc/vepDirectoryCache Specify the cache directory to use (supply whole path, defaults to "")
- -vepf/–vepFeatures VEP features (defaults to ("refseq","hgvs","symbol","numbers","sift","polyphen","humdiv"); comma sep)
- -pVcP/–pVCFParser Parse variants using vcfParser.pl (defaults to "1" (=yes))
- -vcpvt/–vcfParserVepTranscripts Parse VEP transcript specific entries (defaults to "0" (=no))
- -vcprff/–vcfParserRangeFeatureFile Range annotations file (defaults to ""; tab-sep)
- -vcprfa/–vcfParserRangeFeatureAnnotationColumns Range annotations feature columns (defaults to ""; comma sep)

- -vcpsf/–vcfParserSelectFile File containging list of genes to analyse seperately (defaults to “”;tab-sep file and HGNC Symbol required)
- -vcpsfm/–vcfParserSelectFileMatchingColumn Position of HGNC Symbol column in SelectFile (defaults to “”)
- -vcpsfa/–vcfParserSelectFeatureAnnotationColumns Feature columns to use in annotation (defaults to “”; comma sep)
- -pSnE/–pSnpEff Variant annotation using snpEFF (defaults to “1” (=yes))
- -snep/–snpEffPath Path to snpEff. Mandatory for use of snpEff (defaults to “”)
- -snesaf/–snpSiftAnnotationFiles Annotation files to use with snpSift (comma sep)
- -snesdbnsfp/–snpSiftDbNSFPFile DbNSFP File (defaults to “dbNSFP2.6.txt.gz”)
- -snesdbnsfpa/–snpSiftDbNSFPAnnotations DbNSFP annotations to use with snpSift (defaults to (“SIFT_pred”,”Polyphen2_HDIV_pred”,”Polyphen2_HVAR_pred”,”LRT_pred”,”MutationTaster_pred”,”GERP++_NR comma sep)
- -pRankVar/–pRankVariants –> -pRaV/–pRankVariants
- -rs/–rankScore –> removed
- -gf/–geneFile –> -ravgf/–geneFile
- -imdbfile/–ImportantDbFile Important Db file (Defaults to “”) –> removed
- -imdbte/–ImportantDbTemplate Important Db template file used to create the specific family ‘-im_dbmf’ master file (Defaults to “”) –> removed
- -imdbmf/–ImportantDbMasterFile Important Db master file to be used when selecting variants (defaults to “{outDataDir}/{familyID}/{familyID}.intersectCollect_selectVariants_db_master.txt”;Supply whole path) –> removed
- -imdbfof/–ImportantDbFileOutFiles The file(s) to write to when selecting variants with intersectCollect.pl. Comma sep (defaults to “{outDataDir}/{familyID}/{aligner}/GATK/candidates/ranking/{familyID}_orphan.selectVariants, {outDataDir}/{familyID}/{aligner}/GATK/candidates/ranking/clinical/{familyID}.selectVariants”; Supply whole path/file) –> removed
- -ravcs/–caddWGSSNVs Annotate whole genome sequencing CADD score (defaults to “0” (=no))
- -ravcsf/–caddWGSSNVsFile Whole genome sequencing CADD score file (defaults to “whole_genome_SNVs.tsv.gz”)
- -ravc1kg/–cadd1000Genomes 1000 Genome cadd score file (defaults to “0” (=no))
- -ravc1kgf/–cadd1000GenomesFile 1000 Genome cadd score file (defaults to “1000G.tsv.gz”)
- -ravwg/–wholeGene Allow compound pairs in intronic regions (defaults to “1” (=yes))
- -ravrm/–rankModelFile Rank model config file (defaults to “”)
- -pSCheck/–pSampleCheck –> -pScK/–pSampleCheck
- -pQCC/–pQCCollect –> -pQcC/–pQCCollect
- -QCCsampleinfo/–QCCollectSampleInfoFile –> -qccsi/–QCCollectSampleInfoFile
- -QCCregexp/–QCCollectRegExpFile –> -qccref/–QCCollectRegExpFile
- -pREM/–pRemovalRedundantFiles –> -pReM/–pRemoveRedundantFiles
- -pAR/–pAnalysisRunStatus –> -pArS/–pAnalysisRunStatus

CHAPTER 6

Installation

6.1 Automated Installation

This installation procedure assumes that you have a working perl version and a Miniconda installation.

1. "Install" MIP

```
clone the official git repository
$ git clone https://github.com/henrikstranneheim/MIP.git
$ cd MIP
```

After this you can decide whether to make MIP an "executable" by either adding the install directory to the `$PATH` in e.g. "`~/.bash_profile`" or move all the files from this directory to somewhere already in your path like "`~/usr/bin`". Remember to make the file(s) executable by `chmod +x file`.

2. Create the install instructions for MIP

```
$ perl install.pl
This will generate a batch script "mip.sh" for the install in your working directory.
```

3. Run the bash script

```
$ bash mip.sh
This will install all the dependencies of MIP and other modules included in MIP into a conda env
However a fresh version of perl and cpanm is installed outside of the conda environment, but are
```

Note: This will add the following lines to bashrc and bash_profile if the install perl version is not found in your path:

```
'export PATH=$HOME/perl-PERLVERSION/:$PATH' >> ~/.bashrc
'eval `perl -I ~/perl-PERLVERSION/lib/perl5/ -Mlocal::lib=~/perl-PERLVERSION/`' >> ~/.bash_profile
'export PERL_UNICODE=SAD' >> ~/.bash_profile
```

3. Run MIP

```
$ source activate mip
$ mip.pl -h
```

6.2 Manual Installation

1. Install a fresh copy of Perl

On UNIX, Perl5 can be installed by following these instructions. It uses Perlbrew.

2. To switch to the new Perl installation, you might need to run:

```
$ INSTALLER_PERL_VERSION=5.16.0
$ perlbrew switch perl-$INSTALLER_PERL_VERSION
```

3. "Install" MIP

```
clone the official git repository
$ git clone https://github.com/henrikstranneheim/MIP.git
$ cd MIP
$ perl mip.pl -h
```

   After this you can decide whether to make MIP an "executable" by either adding the install directory to the `$PATH` in e.g. "`~/.bash_profile`" or move all the files from this directory to somewhere already in your path like "`~/usr/bin`". Remember to make the file(s) executable by `chmod +x file`.

4. Dependencies

   You need to make sure all depedencies are installed and loaded (See Setup). However, MIP should tell you if something is missing.

5. To install the dependencies - use cpanm:

```
cpanm <dependency>
$ cpanm YAML
```

CHAPTER 7

Setup

7.1 Filename convention

The permanent filename should follow the following format:

```
{LANE}_{DATE}_{FLOW CELL}_{IDN}_{BARCODE SEQ}_{DIRECTION 1/2}.fastq.qz
```

Note: The *familyID* and *sampleID(s)* needs to be unique and the sampleID supplied should be equal to the {IDN} in the filename.

However, MIP will except filenames in other formats as long as the filename contains the sampleID and the mandatory information can be collected from the fastq header.

7.2 Dependencies

Make sure you have loaded/installed all dependencies and that they are in your `$PATH`. You only need to load the dependencies that are required for the modules that you want to run. If you fail to install dependencies for a module, MIP will tell you what dependencies you need to install (or add to your `$PATH`) and exit. MIP comes with an install script `install.pl`, which will install all necessary programs to execute models in MIP via bioconda and/or $SHELL. Version after the software name are tested for compatibility with MIP.

Program/Modules

- Perl modules: YAML.pm, Log4perl.pm, List::MoreUtils, DateTime, DateTime::Format::ISO8601, DateTime::Format::HTTP, DateTime::Format::Mail, Set::IntervalTree from CPAN, since these are not included in the perl standard distribution
- Simple Linux Utility for Resource Management (SLURM)
- FastQC (version: 0.11.5)
- Mosaik (version: 2.2.24)
- BWA (version: 0.7.15)
- BWAKit (version: 0.7.12)
- Sambamba (version: 0.6.3)
- SAMTools (version: 1.3.1)
- BedTools (version: 2.26.0)

- PicardTools (version: 2.5.0)
- Chanjo (version: 3.4.1)
- Manta (version: 1.0.0)
- GATK (version: 3.6)
- freebayes (version: 1.0.2)
- VT (version: 20151110)
- VEP (version: 84) with plugin "UpDownDistance, LoFtool, LoF"
- vcfParser.pl (Supplied with MIP; see vcfParser)
- SnpEff (4.2)
- ANNOVAR (version: 2013-08-23)
- GENMOD (version: 3.5.6)
- variant_integrity (version: 0.0.4)
- VcfTools (version: 0.1.0)
- BcfTools (version: 1.3.1)
- Htslib (version: 1.3.1)
- PLINK2 (version: 1.90b3x35)
- MultiQC (version: 0.8dev0)

Depending on what programs you include in the MIP analysis you also need to add these programs to your `$PATH`:

- FastQC
- Mosaik
- BWA
- SAMTools
- Tabix
- BedTools
- VcfTools
- PLINK

and these to your python `virtualenvironment`:

- Chanjo
- GENMOD
- Cosmid (version: 0.4.9.1) for automatic download

To make sure that you use the same commands to work on the virtualenvironment, you need to install a virtual environment wrapper. We recommend pyenv and pyenv-virtualenvwrapper. To enable the virualenvwrapper add: `pyenv virtualenvwrapper` to your `~/.bash_profile`.

7.2.1 Databases/References

Please checkout Cosmid to download references and/or databases on your own or via MIP.

MIP can build/download many program prerequisites automatically:

Note: Download is only enabled when using the default parameters of MIP and requires a Cosmid installation in your python virtualenvironment.

Automatic Download:

1. Human Decoy Genome Reference (1000G)
2. The Consensus Coding Sequence project database (CCDS)
3. Relevant references from the 1000G FTP Bundle (mills, omni, dbsnp etc)

Automatic Build:

Human Genome Reference Meta Files:

1. The sequence dictionnary (".dict")
2. The ".fasta.fai" file

Mosaik:

1. The Mosaik align format of the human genome {mosaikAlignReference}.
2. The Mosaik align jump database {mosaikJumpDbStub}.
3. The Mosaik align network files {mosaikAlignNeuralNetworkPeFile} and {mosaikAlignNeuralNetwork-SeFile}. These will be copied from your MOSAIK installation to the MIP reference directory.

BWA:

1. The BWA index of the human genome.

Note: If you do not supply these parameters (Mosaik/BWA) MIP will create these from scratch using the supplied human reference genom as template.

Capture target files:

1. The "infile_list" and .pad100.infile_list files used in {pPicardToolsCalculateHSMetrics}
2. The ".pad100.interval_list" file used by some GATK modules.

Note: If you do not supply these parameters MIP will create these from scratch using the supplied latest supported capture kit ".bed" file and the supplied human reference genome as template.

ANNOVAR: The choosen Annovar databases are downloaded before use if lacking in the annovar/humandb directory using Annovars built-in download function.

Note: This applies only to the supported annovar databases. Supply flag "–annovarSupportedTableNames" to list the MIP supported databases.

CHAPTER 8

MIP Analysis

You can modify all parameters to MIP in order of precedence using:

1. Command line
2. Pedigree file
3. Config file
4. Definitions file (typically not done by user)

8.1 Start standard analysis

```
    $ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
```

`-rio 1` means that two blocks will be performed at the nodes without transfer of files between HDS and SLURM nodes within the block. These two blocks are:

[BAMCalibrationBlock]

- [PicardTool MergeSamFiles]
- [Sambamba Markduplicates]
- [GATK ReAlignerTargetCreator/IndelRealigner]
- [GATK BaseRecalibrator/PrintReads]

[VariantAnnotationBlock]

- [PrepareForVariantAnnotationBlock]
- [VT]
- [VariantEffectPredictor]
- [VCFParser]
- [SnpEff]
- [RankVariants]

`-rio 0` means that MIP will copy in and out files from HDS and SLURM nodes between each module. Thus increasing the network traffic.

8.2 Excluding a program from the analysis

```
$ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
```

8.3 Skipping a already processed module i.e expect that the ouput has already been generated

```
$ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
```

8.4 Simulate standard analysis

```
$ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
```

`-dra` means that MIP will run in dru run mode i.e simulation mode. MIP will execute everything except the final sbatch submission to SLURM and updates to qc_sampleInfo.yaml.

`-dra 2` will include simulation of downloading/building of references.

`-dra 1` will simulate analysis without downloading/building of references.

`-dra 0` no simulation

One can use `-dra 1` or `-dra 2` to generate sbatch scripts which then can be submitted manually by the user individually or sequentially using `sbatch --dependency`. Note that this will not update qc_sampleInfo.yaml as this is done at MIP runtime.

8.5 Rerun analysis using exactly the same parameters as last analysis run

```
$ perl develop/modules/MIP/mip.pl -c /mnt/hds/proj/cust003/develop/exomes/0/0_config.yaml
```

8.6 Rerun analysis using exactly the same parameters as last analysis run, but in simulation mode

```
$ perl develop/modules/MIP/mip.pl -c /mnt/hds/proj/cust003/develop/exomes/0/0_config.yaml -dra 2
```

8.7 Generate all supported standard programs

```
$ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
```

This will print a string with programs in mode 2 (expect ouput) in chronological order (as far as possible, some things are processed in parallel):

```
$ --pGZipFastq 2 --pFastQC 2 --pBwaMem 2 --pPicardToolsMergeSamFiles 2 --pSambambaMarkduplicates
```

Thus you will always have the actual program names that are supported facilitating starting from any step in the analysis for instance updating qc_sampleInfo.yaml and rerunning module in [BAMCalibrationBlock] skipping [Sambamba Markduplicates]:

```
    $ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
```

You can of course start or skip any number of modules as long as it is sane to do so (MIP will not check this but just execute)

8.8 You can also modulate the mode of '-pp' using -ppm:

```
    $ perl develop/modules/MIP/mip.pl -f 0 -c develop/references/CMMS_Rasta_Config.v1.4.yaml -rio 1
    $ --pGZipFastq 1 --pFastQC 1 --pBwaMem 1 --pPicardToolsMergeSamFiles 1 --pSambambaMarkduplicates
```

CHAPTER 9

Adding a new program

You need to perform a series of tasks to properly add a program to MIP. An overview of the steps can be found here:

1. *Call DefineParameters*
2. *Command line arguments in GetOptions*
3. *if-block run checker in MAIN*

1. Print program name to `MIPLOGG` and `STDOUT`
2. Call your custom subroutine (ses below) with relevant parameters

4. *Custom subroutine*

1. Writes SBATCH headers
2. Figure out i/o files
3. Builds out the body of the SBATCH script
4. Calls *FIDsubmitJob*

More details follow below. Chanjo, a program which is part of the coverage analysis, will be used as an example.

9.1 Call DefineParameters

This subroutine takes a number of input parameters. There are basically three parameter types: "program", "file", and "attribute". Try to group your parameter definitions with related programs.

```
DefineParameters("pChanjoBuild", "program", 1, "MIP", 0, "nofileEnding", "CoverageReport");

DefineParameters("chanjoBuildDb", "path", "CCDS.current.txt", "pChanjoBuild", "file");

DefineParameters("pChanjoCalculate", "program", 0, "MIP", 0, "nofileEnding", "MAIN");

DefineParameters("chanjoCalculateCutoff", "program", 10, "pChanjoCalculate", 0)
```

Table 9.1: DefineParameters - parameters

Parameter	Example	Description
Name	pChanjoBuild	Program names start with 'p' by convention, otherwise it's up to you.
Type	program	Can be either program or path.
Default	1	**Program**: 1/0 as on/off, **file**: <path to file> or 'nodefault', **attribute**: e.g 10 or 'nodefault'
Associated program	MIP	Typically the program that calls this program. **program**: usually MIP, **file/attribute**: <Name>.
Exists check	0	Perform a check that a file is in the reference directory. Either: 0, 'file', 'directory'.
File ending	nofileEnding	File ending when module is finished. MIP uses this to determine input files downstream in the Chain. **file/attribute**: skip.
Chain	MAIN	The chain to which the program belongs to. **file/attribute**: skip.
Check install	chanjo	The program handle to check whether it is in the `$PATH`. **file/attribute**: skip.

9.2 Command line arguments in *GetOptions*

This is the method that parses the command line input and stores the options. To add your own defined parameters you need to add lines like this:

```
'<short_option>|<long_option>:<s(tring)/n(umber)>' => \$parameter{'<long_option>'}{'value'},
```

You should replace anything that looks like `<placeholder>`:

```
'pCh|pChanjoBuild:n' => \$parameter{'pChanjoBuild'}{'value'},  # ChanjoBuild coverage analysis
'chbdb|chanjoBuildDb:s' => \$parameter{'chanjoBuildDb'}{'value'},  # Central SQLite database path
'pCh_C|pChanjoCalculate:n' => \$parameter{'pChanjoCalculate'}{'value'}, # Chanjo coverage analysis
'chccut|chanjoCalculateCutoff:n' => \$parameter{'chanjoCalculateCutoff'}{'value'}, # Cutoff used for
```

Again, program options begin with a leading "p" by convention. Make sure you don't cause any naming conflicts.

Lists can also be specified with a special syntax. Basically you need to assign the option to an array instead of `$scriptParameters`.

```
'ifd|inFilesDirs:s'  => \@inFilesDirs, #Comma separated list
```

Later in your code when you would like to access those values you would join on ",".

```
@inFilesDirs = join(',', @inFilesDirs);
```

Note: MIP doesn't use True/False flags, all options take at least one argument. For program options it's possible to turn on (1), off (0) and run programs in dry mode (2). All program options should specify "n(umber)" as argument type.

9.3 if-block run checker in MAIN

The if-block checks whether the program is set to run but it also has a number of additional responsibilities.

Perhaps the most important is to define dependencies. This is done by placing your if-statement after the closest upsteam process to yours. ChanjoBuild, for example, needs to wait until *PicardToolsMarkDuplicates* has finished processing the BAM-files before running.

```
# Closest upsteam dependency for Chanjo
if ($scriptParameter{'pPicardToolsMarkduplicates'} > 0) {
  # Body...
}

# This is where Chanjo fits!
if ($scriptParameter{'pChanjoBuild'} > 0) {
  # Body...
}
```

Next (inside the if-block) it should print an announcement to two file handles:

```
for my $fh (STDOUT, MIPLOGG) { print $fh "\nChanjoBuild\n"; }
```

Lastly it should call a *Custom subroutine*, e.g. for each individual sample or per family, which will write a SBATCH script(s), submit them to SLURM, which executes the module.

Note: `$sampleInfo` is a hash table storing sample information, for example filename endings from different stages of the pipeline. It's used to determine input filenames for your program.

9.4 Custom subroutine

First up, let's choose a relevant (and conflict free) name for our subroutine.

```
sub ChanjoBuild {
  # Body...
}
```

If we pass ALL nessesary variables into the subroutine and assign them as scoped variables it's easy to overview variables used inside.

```
my $sampleID = $_[0];
my $familyID = $_[1];
my $aligner = $_[2];
# etc ...
```

9.4.1 a) SBATCH headers

SBATCH headers are written by the *ProgramPreRequisites* subroutine. It takes a number of input arguments.

```
ProgramPreRequisites($sampleID, "ChanjoBuild", "$aligner/coverageReport", 0, *CHANJOBUI, 1, $runtimeH
```

Table 9.2: ProgramPreRequisites - paramaters

Parameter	Example	Description
Directory	11-1-1A	Either a sample ID (e.g. IDN) or family ID depending on where output is stored.
Program	chanjo	Used in SBATCH script filename.
Program directory	`$aligner/coverageReport`	Defines output directory under *Directory*. Path should include current aligner by convention.
Call type	0	Options: *SNV*, *INDEL* or *BOTH*. Can be set to: 0 ???
File handle	`*CHANJO`	The program specific file handle which will be written to when generating the SBATCH script. Always prepend: '*'.
Cores	1	The number of cores to allocate.
Process time	1.5	An estimate of the runtime for the particular sample in hours.

9.4.2 b) Figure out i/o files

It's up to you to figure out where your program should store output files. Basically you need to ask yourself whether putting them in the family/sample foler makes the most sense.

It's a good idea to first specify both in- and output directories.

```
my $baseDir = "$outDataDir/$sampleID/$aligner";
my $inDir = $baseDir;
my $outDir = "$baseDir/coverageReport";
```

If you depend on earlier scripts to generate infile(s) for the new program it's up to you to figure out the closest program upstream. After that you can ask for the file ending.

```
my $infileEnding = $sampleInfo{ $familyID }{ $sampleID }{'pPicardToolsMarkduplicates'}{'fileEnding'};
```

`$sampleInfo` is a hash table in global scope.

MIP supports multiple infiles and therefore MIP needs to check if the file(s) have been merge or not.This is done with the *CheckIfMergedFiles* subroutine, which returns either a 1 (files was merged) or 0 (no merge of files)

```
my ($infile, $mergeSwitch) = CheckIfMergedFiles($sampleID);
```

Note: `$infilesLaneNoEnding` is a global hash table containing information about the filename-bases (compare filename-endings).

9.4.3 c) Build SBATCH body

This is where you fit relevant parameters into your command line tool interface. Print everything to the file handle you defined above.

```
print CHANJOBUI "
# ------------------------------------------------------------
#  Create a temp JSON file with exon coverage annotations
# ------------------------------------------------------------\n";
print CHANJOBUI "chanjo annotate $storePath using $bamFile";
print CHANJOBUI "--cutoff $cutoff";
print CHANJOBUI "--sample $sampleID";
print CHANJOBUI "--group $familyID";
print CHANJOBUI "--json $jsonPath";

# I'm done printing; let's drop the file handle
close(CHANJOBUI);
```

Note: A `wait` command should be added after submitting multiple processes in the same SBATCH script with the `&` command. This will ensure SLURM waits for all processes to finish before quitting on the job.

9.4.4 d) Call *FIDSubmitJob*

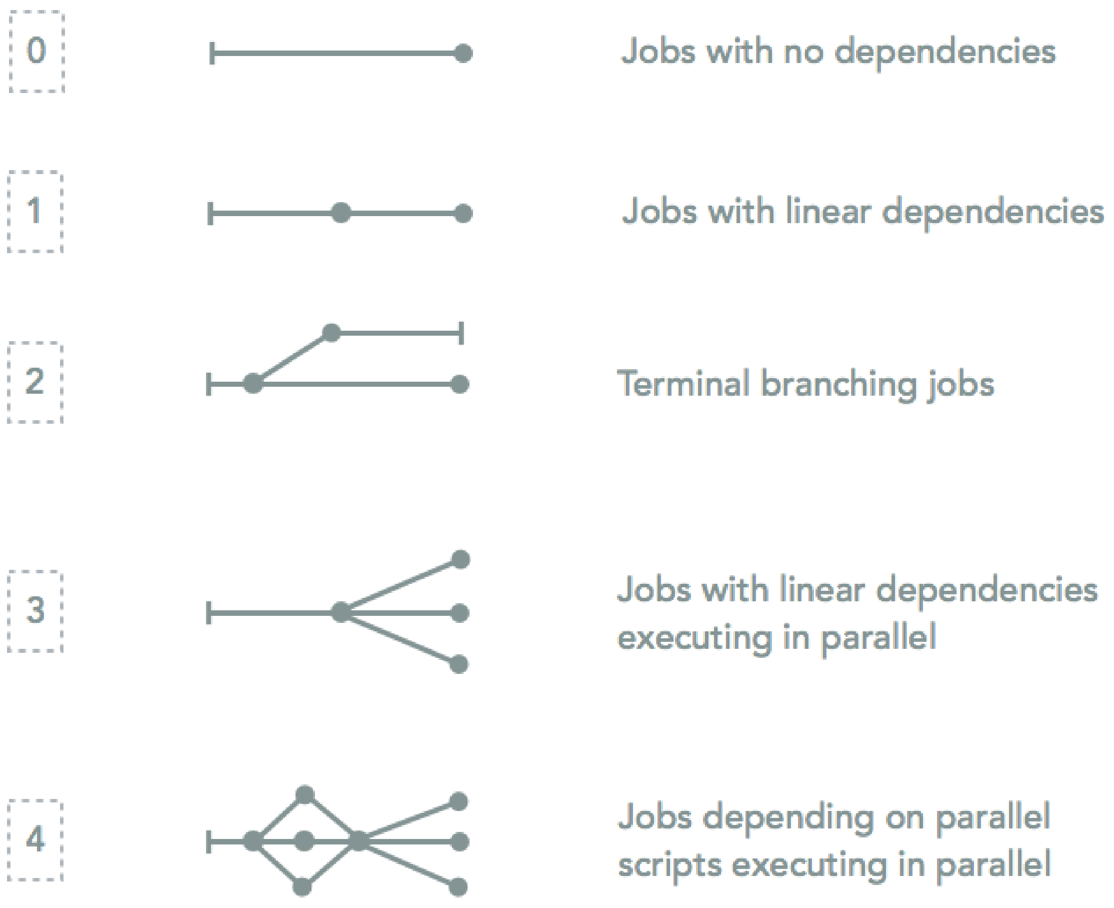
This subroutine is responsible for actually submitting the SBATCH script and handling dependencies. You should only call this if the program is supposed to run for real (not dry run).

```
if ( ($runMode == 1) && ($dryRunAll == 0) ) {
  # ChanjoBuild is a terminally branching job: linear dependencies/no follow up
  FIDSubmitJob($sampleID, $familyID, 2, $parameter{'pChanjoBuild'}{'chain'}, $filename, 0);
}
```

Table 9.3: FIDSubmitJob - paramaters

Parameter	Example	Description
Sample ID	11-1-1A	The sample ID/person IDN
Family ID	11	The family ID
Dependency type	2	Choose between type 0-4 (see below)
Chain key	$parameter{'pChanjo'}{'chain'}	The chain defined in *DefineParameters*
SBATCH filename	`$filename`	Always use this variable. It automagically points to your SBATCH script file.
Script tracker	0	Huh? Something about parallel processes...

To figure out which option (integer) to supply as the third argument to *FIDSubmitJob* you can take a look at this illustration.


0
Jobs with no dependencies
1
Jobs with linear dependencies
2
Terminal branching jobs
3
Jobs with linear dependencies executing in parallel
4
Jobs depending on parallel scripts executing in parallel

Note: `$filename` is a variable that is created in *ProgramPreRequisites*. It points to your freshly composed SBATCH script file and should be supplied to *FIDSubmitJob* by all custom subroutines.

Note: `$parameter{'pChanjoBuild'}{'chain'}` is just the chain that you set in *DefineParameters*. In this case we could've replaced it with "MAIN".

9.5 Further information

For your convinience a template program module can be found in the project folder hosted on GitHub. [ADD LINK TO TEMPLATE]

CHAPTER 10

Structure

10.1 mip.pl

Central hub and likely the only script most users will ever interact directly with.

```
$ echo "Running MIP on Uppmax, analyzing all samples in family 10"
$ mip.pl -c CMMS_Uppmax_config.yaml -f 10
```

10.2 Sequence QC

Raw sequence quality control: FastQC

10.3 Alignment

Currently MIP supports these aligners:

1. Mosaik (WES, WGS)
2. BWA (WES, WGS, Rapid WGS)

10.4 BAM file manipulation

- Sorting and indexing: PicardTools (SortSam)
- Duplicate marking: PicardTools (MarkDuplicates & MarkDuplicatesWithMateCigar)
- Realignment and base recalibration: GATK (Realigner & BaseRecalibration)

10.5 Coverage QC

- Coverage Report and QC metrics: Chanjo & BedTools
- QC metrics: PicardTools (MultipleMetrics & HSmetrics)

10.6 Variant calling

- Variant discovery and recalibration: GATK (HaploTypeCaller, GenoTypeGVCFs & VariantRecalibration)

10.7 Variant QC

- All variants: GATK (VariantEval)
- Exonic variants: GATK (VariantEval)

10.8 Variant Selection

Select transcripts that overlap a gene list: vcfParser

10.9 Variant annotation

Collect transcript and amino acid information and information from external databases as well as annotation of inheritance models: VEP, vcfParser, SnpEff, ANNOVAR, GENMOD

10.10 Variant evaluation

Score and rank each variant using weighted sums according to disease causing potential: GENMOD score (see `genmod_score`)

10.11 qcCollect.pl

Collects QC data from the MPS analysis in YAML format. (see QCCollect).

10.12 covplots_exome.R / covplots_genome.R

Plots coverage across chromosomes.

CHAPTER 11

vcfParser

Parses vcf files to reformat/add INFO fields and metaData headers and/or select entries belonging to a subgroup e.g. a list of genes. Input can be piped or supplied as an infile.

11.1 Usage

`vcfParser.pl infile.vcf > outfile.vcf`

`vcfParser.pl infile.vcf --parseVEP 1 -rf External_Db.txt -rf_ac 3 -sf genes.v1.0.txt -sf_mc 3 -sf_ac 3,4,11,15,17,20 -sof selected_genes.vcf > outfile.vcf`

11.2 Installation

vcfParser is written in Perl, so naturally you need to have Perl installed. The perl module Set::IntervalTree is required and are used to add "ranged" annotations.

11.2.1 VEP

Parses the output from VEP to include RefSeq transcripts. The transcript and protein annotations, moste severe consequence and gene annotations are also included in the output . Transcript protein predictions (Sift and Polyphen) can also be included.

11.2.2 Select Mode

A list of genes and their corresponding HGNC Symbol can be used to fork the analysis into "selected" genes and "orphan" genes.

GuideLines on format for database of genes

- The database file should contain a header line starting with "#".
- The number of headers should match the number of field elements for each entry.
- Do not use whitespace in headers.
- Do not use ";" in file.

- Separate elements in fields with ",". Do not use ", ".
- No whitespace in the beginning or end within fields.
- No entries should be duplicated within database.
- Length of gene coordinates should be greater than 0
- Only digits in gene coordinate entries

11.2.3 Range Annotations

vcfParser can also add range annotations to the vcf by using the Set::IntervalTree perl cpan module and a file with chromosomal coordinates and features to be annotated.

CHAPTER 12

QCCollect

Collects information on MPS analysis from each analysis run. Uses YAML files for input and output. QCCollect uses a yaml file for matching the outdata produced in each run to another yaml file with regular expression used to actually collect the data from the output files. The collected data is then written to disc in yaml format.

MIP produces a sampleInfo yaml file, containing all sample and family information used in each analysis run.

12.1 Usage

```
perl qcCollect.pl -si [SampleInfoFilePath] -r [regularExpressionFilePath] -o
[Outfile]
```

12.2 Installation

qcCollect is written in Perl, so naturally you need to have Perl installed.

12.3 SetUp

1. The regular expression file needs to be created. The regExp file used at CMMS can be printed from qcCollect using the `-preg & -prego` flags

Table 12.1: qcCollect Parameters

Short/Long	Default Value	Type	Summary
-si/–sampleInfoFile	Na	`String`	The sample info file (Yaml;supply whole path)
-r/–regExpFile	Na	`String`	The regular expresion file (Yaml;supply whole path)
-o/–outfile	"qcmetrics.yaml"	`String`	The output file
-preg/–printRegExp	0	`Integer`	Print RegExp YAML file used at CMMS switch
-prego/–printRegExpOutFile	"qc_regExp.yaml"	`String`	The RegExp YAML outfile
-h/–help	Na	Na	Display help message
-v/–version	Na	Na	Display version

CHAPTER 13

rank_modelv1.18

Genmod score uses the weighted sum model (WSM) approach to rank the most likely pathogenic variant.

Generally, the higher value the more likely pathogenic variant.

Genmod_score uses config files to define the rank model, which enables customized set-up and versioning of rank models.

The WSM uses the following alternatives and weights in the rank model:

Rank score range: -33 <= rs <= 44

13.1 Consequence

Each alleles variant effect on individual transcripts are evaluated using a rule-based approach defined by SO-terms. The SO-terms themselves are ranked in order of severity and this ranking is used to defined the weight of the consequence alternative. The performance score is based on the most severe consequence within each gene.

Performance value for the SO-terms:

- transcript_ablation = 10
- initiator_codon_variant = 9
- frameshift_variant = 8
- stop_gained = 8
- start_lost = 8
- stop_lost = 8
- splice_acceptor_variant = 8
- splice_donor_variant = 8
- inframe_deletion = 5
- transcript_amplification = 5
- splice_region_variant = 5
- missense_variant = 5
- protein_altering_variant = 5
- inframe_insertion = 5
- incomplete_terminal_codon_variant = 5

- non_coding_transcript_exon_variant = 3
- synonymous_variant = 2
- mature_miRNA_variant = 1
- non_coding_transcript_variant = 1
- regulatory_region_variant = 1
- upstream_gene_variant = 1
- regulatory_region_amplification = 1
- TFBS_amplification = 1
- 5_prime_UTR_variant = 1
- intron_variant = 1
- 3_prime_UTR_variant = 1
- feature_truncation = 1
- TF_binding_site_variant = 1
- stop_retained_variant = 1
- feature_elongation = 1
- regulatory_region_ablation = 1
- TFBS_ablation = 1
- coding_sequence_variant = 1
- downstream_gene_variant = 1
- NMD_transcript_variant = 1
- intergenic_variant = 0
- not_reported = 0

13.2 Frequency

The alternative allele frequency (AF) in public databases (1000G, ExAC, MTAF, SWEREF). The highest reported alternative frequency and observation count (locusDB) reported from the databases is used to calculate the performance value.

Definitions:

- Not reported: AF Na
- Very Rare: AF < 0.0005
- Rare: 0.0005 <= AF < 0.005
- Intermediate: 0.005 <= AF < 0.02
- Common: AF >= 0.02

Performance value for maximum AF:

- Not reported = 4
- Very rare = 3

- Rare = 2
- Intermediate = 1
- Common = -12

Observation Count

The observation count (Obs) from the local variant database locusDB.

Definitions:

- Not reported: Obs Na
- Very Rare: Obs < 5
- Rare: 5 <= Obs < 10
- Intermediate: 10 <= Obs < 20
- Common: Obs >= 20

Performance value for maximum Obs:

- Not reported = 4
- Very Rare = 3
- Rare = 2
- Intermediate = 1
- Common = -12

13.3 Inheritance Model(s)

The segregation pattern for the variant within the family. These models are currently annotated using genmod models. A variant that is annotated as autosomal compound with no compound partner with a rank score greater than 10 will receive a penalty of -6 to the variants rank score. For single samples this rule will be enforced for variants with inheritance model autosomal dominant, autosomal dominant denovo in addition to the autosomal compound annotation.

Definitions:

- Autosomal Recessive, denoted 'AR_hom'
- Autosomal Recessive denovo, denoted 'AR_hom_dn'
- Autosomal Dominant, 'AD'
- Autosomal Dominant denovo, 'AD_dn'
- Autosomal Compound Heterozygote, 'AR_comp'
- X-linked dominant, 'XD'
- X-linked dominant de novo, 'XD_dn'
- X-linked Recessive, 'XR'
- X-linked Recessive de novo, 'XR_dn'

Performance value for inheritance models:

- Valid model = 1
- No model = -12

- AR_comp penalty = -6

13.4 Protein Functional Prediction

The predicted functional effect on the protein. Currently 2 protein effect predictors are used (Sift, PolyPhen2). Each predictors can contribute 1 point each to the overall protein predictor performance score.

SIFT predicts whether an amino acid substitution is likely to affect protein function based on sequence homology and the physico-chemical similarity between the alternate amino acids [1].

PolyPhen-2 predicts the effect of an amino acid substitution on the structure and function of a protein using sequence homology, Pfam annotations, 3D structures from PDB where available, and a number of other databases and tools (including DSSP, ncoils etc [2].

Definitions:

- Sift Terms:
  - "D" Deleterious (score<=0.05)
  - "T" Tolerated (score>0.05)
- PolyPhen2HumVar Terms:
  - "D": Probably damaging (>=0.909)
  - "P": Possibly damaging (0.447<=pp2_hvar<=0.909)
  - "B": Benign (pp2_hvar<=0.446)

Performance value for protein predictors:

- Sift:
  - D = 1
- PolyPhen2HumVar:
  - D or P = 1

13.5 Gene Intolerance Score

EXAC gene intolerance score - calculated by VEP's LoFtool plugin.

Definitions:

- Not reported: LoFtool Na
- Low: LoFtool < 0.0001
- Medium: 0.0001 <= LoFtool < 0.01
- High LoFtool < 0.01

Performance value for gene intolerance score:

- Not reported = 0
- Low = 2
- Medium = 1
- High = 0

13.6 Variant Quality Filter

Each variant call has a filter tranche attached to it indicating the quality of the actual variant call.

Definitions:

- PASS
- Other (Tranches e.g. For GATK [3]: "VQSRTrancheBOTH99.90to100.00"

We also evaluate the combined GQ score called a Model score for reducing the impact of poor quality genotypes across a case.

Definitions:

- Low quality (GQ => 20)
- High quality (GQ > 20)

Performance value for variant quality filter:

- Filter tranche:
  - PASS = 3
  - Other = 0
- Model score:
  - Low quality = -5
  - High quality = 0

13.7 Conservation

The level of conservation for a sequence element (PhastCons [4]), nucleotides or classes of nucleotides PhyloP [5] both from the Phast [6] package as well as genomic constraint score GERP [7] is used. The Phast datasets used in the conservation calculation were generated by the UCSC/Penn State Bioinformatics comparative genomics alignment pipeline. A description of this analysis can be found at UCSC. Each type of conservation can contribute 1 point each to the overall conservation performance score.

Definitions:

- Conserved
  - PhastCons: 0.8 >= Score <= 1
  - GERPRS: Score >= 2
  - PhyloP: Score > 2,5

Performance value for conservation:

- Conserved:
  - PhastCons100way_vertebrate = 1
  - PhyloP100way_vertebrate = 1
  - GERP++RS = 1

13.8 Combined Annotation Dependent Depletion (CADD)

CADD is a tool for scoring the deleteriousness of single nucleotide variants as well as insertion/deletions variants in the human genome. C-scores strongly correlate with allelic diversity, pathogenicity of both coding and non-coding variants, and experimentally measured regulatory effects, and also highly rank causal variants within individual genome sequences. The CADD-score is a pre-calculated for all SNVs and for indel from 1000G-project [8].

Definitions:

- Strongly deleterious (CADD > 40)
- deleterious (40 >= CADD > 30)
- Mildly deleterious (30 >= CADD > 20)
- Probably deleterious (20 >= CADD > 10)
- Benign (10 >= CADD >= 0)

Performance value for CADD:

- Strongly deleterious = 5
- Deleterious = 4
- Mildly deleterious = 3
- Probably deleterious = 2
- Benign = 0

13.9 ClinVar

ClinVar [9] is a freely accessible, public archive of reports of the relationships among human variations and phenotypes, with supporting evidence. Each variant in clinvar has a record of clinical significance (CLNSIG):

Definitions:

- Uncertain significance = 0
- Not provided = 1
- Benign = 2
- Likely benign = 3
- Likely pathogenic = 4
- Pathogenic = 5
- Drug response = 6
- Histocompatibility = 7
- Other = 255

Performance value for ClinVar:

- Uncertain significance = 0
- Not provided = 0
- Benign = -1
- Likely benign = 0

- Likely pathogenic = 2
- Pathogenic = 5
- Drug response = 0
- Histocompatibility = 0
- Other = 0

Clinical review status

Clinical review status (CLNREVSTAT) is a measure on the certainty of the supporting evidence for the variant's clinical significance.

Definitions:

- not_reported
- no_assertion
- no_criteria
- single
- mult
- conf
- exp
- guideline

Performance value for CLNREVSTAT:

- not_reported = 0
- no_assertion = 0
- no_criteria = 0
- single = 1
- conf = 1
- mult = 2
- exp = 3
- guideline = 4

13.10 Spidex

Spidex is a database for snvs that have been predicted to affect splicing.

Definitions:

- Not reported = 0
- Low (-1 < dpsi < 1)
- Medium (-1 <= dpsi > 1 & -2 > dpsi < 2)
- High (-2 <= dpsi >= 2)

Performance value for Spidex:

- Low = 0

- Medium = 3
- High = 5

CHAPTER 14

rank_modelv1.11

Genmod score uses the weighted sum model (WSM) approach to rank the most likely pathogenic variant.

Generally, the higher value the more likely pathogenic variant.

Genmod_score uses config files to define the rank model, which enables customized set-up and versioning of rank models.

The WSM uses the following alternatives and weights in the rank model:

Rank score range: -25 <= rs <= 27

14.1 Consequence

Each alleles variant effect on individual transcripts are evaluated using a rule-based approach defined by SO-terms. The SO-terms themselves are ranked in order of severity and this ranking is used to defined the weight of the consequence alternative. The performance score is based on the most severe consequence within each gene.

Performance value for the SO-terms:

- transcript_ablation = 10
- initiator_codon_variant = 9
- frameshift_variant = 8
- stop_gained = 8
- start_lost = 8
- stop_lost = 8
- splice_acceptor_variant = 8
- splice_donor_variant = 8
- inframe_deletion = 5
- transcript_amplification = 5
- splice_region_variant = 5
- missense_variant = 5
- protein_altering_variant = 5
- inframe_insertion = 5
- incomplete_terminal_codon_variant = 5

- synonymous_variant = 2
- non_coding_transcript_exon_variant = 1
- mature_miRNA_variant = 1
- non_coding_transcript_variant = 1
- regulatory_region_variant = 1
- upstream_gene_variant = 1
- regulatory_region_amplification = 1
- TFBS_amplification = 1
- 5_prime_UTR_variant = 1
- intron_variant = 1
- 3_prime_UTR_variant = 1
- feature_truncation = 1
- TF_binding_site_variant = 1
- stop_retained_variant = 1
- feature_elongation = 1
- regulatory_region_ablation = 1
- TFBS_ablation = 1
- coding_sequence_variant = 1
- downstream_gene_variant = 1
- NMD_transcript_variant = 1
- intergenic_variant = 0
- not_reported = 0

14.2 Frequency

The alternative allele frequency (AF) in public databases (1000G, ExAC). The highest reported alternative frequency reported from the databases is used to calculate the performance value.

Definitions:

- Not reported: AF Na
- Rare: AF <= 0.005
- Intermediate: 0.005 <= AF <= 0.02
- Common: AF > 0.02

Performance value for maximum AF:

- Not reported = 3
- Rare = 2
- Intermediate = 1
- Common = -12

14.3 Inheritance Model(s)

The segregation pattern for the variant within the family. These models are currently annotated using genmod models. A variant that is annotated as autosomal compound with no compound partner with a rank score greater than 10 will receive a penalty of -6 to the variants rank score. For single samples this rule will be enforced for variants with inheritance model autosomal dominant, autosomal dominant denovo in addition to the autosomal compound annotation.

Definitions:

- Autosomal Recessive, denoted 'AR_hom'
- Autosomal Recessive denovo, denoted 'AR_hom_dn'
- Autosomal Dominant, 'AD'
- Autosomal Dominant denovo, 'AD_dn'
- Autosomal Compound Heterozygote, 'AR_comp'
- X-linked dominant, 'XD'
- X-linked dominant de novo, 'XD_dn'
- X-linked Recessive, 'XR'
- X-linked Recessive de novo, 'XR_dn'

Performance value for inheritance models:

- Valid model = 1
- No model = -12
- AR_comp penalty = -6

14.4 Protein Functional Prediction

The predicted functional effect on the protein. Currently 2 protein effect predictors are used (Sift, PolyPhen2). Each predictors can contribute 1 point each to the overall protein predictor performance score.

SIFT predicts whether an amino acid substitution is likely to affect protein function based on sequence homology and the physico-chemical similarity between the alternate amino acids [1].

PolyPhen-2 predicts the effect of an amino acid substitution on the structure and function of a protein using sequence homology, Pfam annotations, 3D structures from PDB where available, and a number of other databases and tools (including DSSP, ncoils etc [2].

Definitions:

- Sift Terms:
  - "D" Deleterious (score<=0.05)
  - "T" Tolerated (score>0.05)
- PolyPhen2HumVar Terms:
  - "D": Probably damaging (>=0.909)
  - "P": Possibly damaging (0.447<=pp2_hvar<=0.909)
  - "B": Benign (pp2_hvar<=0.446)

Performance value for protein predictors:

- Sift:
  - D = 1
- PolyPhen2HumVar:
  - D or P = 1

14.5 Variant Quality Filter

Each variant call has a filter tranche attached to it indicating the quality of the actual variant call.

Definitions:

- PASS
- Other (Tranches e.g. For GATK [3]: "VQSRTrancheBOTH99.90to100.00"

Performance value for variant quality filter:

- PASS = 3
- Other = 0

14.6 Conservation

The level of conservation for a sequence element (PhastCons [4]), nucleotides or classes of nucleotides PhyloP [5] both from the Phast [6] package as well as genomic constraint score GERP [7] is used. The Phast datasets used in the conservation calculation were generated by the UCSC/Penn State Bioinformatics comparative genomics alignment pipeline. A description of this analysis can be found at UCSC. Each type of conservation can contribute 1 point each to the overall conservation performance score.

Definitions:

- Conserved
  - PhastCons: 0.8 >= Score <= 1
  - GERPRS: Score >= 2
  - PhyloP: Score > 2,5

Performance value for conservation:

- Conserved:
  - PhastCons = 1
  - PhyloP = 1
  - GERP = 1

14.7 Combined Annotation Dependent Depletion (CADD)

CADD is a tool for scoring the deleteriousness of single nucleotide variants as well as insertion/deletions variants in the human genome. C-scores strongly correlate with allelic diversity, pathogenicity of both coding and non-coding variants, and experimentally measured regulatory effects, and also highly rank causal variants within individual genome sequences. The CADD-score is a pre-calculated for all SNVs and for indel from 1000G-project [8].

Definitions:

- Strongly deleterious (CADD > 40)
- deleterious (CADD > 30)
- Mildly deleterious (CADD > 20)
- Probably deleterious (CADD > 10)

Performance value for CADD:

- Strongly deleterious = 4
- Deleterious = 3
- Mildly deleterious = 2
- Probably deleterious = 1

14.8 ClinVar

ClinVar [9] is a freely accessible, public archive of reports of the relationships among human variations and phenotypes, with supporting evidence.

Definitions:

- Uncertain significance = 0
- Not provided = 1
- Benign = 2
- Likely benign = 3
- Likely pathogenic = 4
- Pathogenic = 5
- Drug response = 6
- Histocompatibility = 7
- Other = 255

Performance value for ClinVar:

- Uncertain significance = 0
- Not provided = 0
- Benign = -1
- Likely benign = 0
- Likely pathogenic = 1
- Pathogenic = 2
- Drug response = 0
- Histocompatibility = 0
- Other = 0

CHAPTER 15

rank_modelv1.5

Genmod_score uses the weighted sum model (WSM) approach to rank the most likely pathogenic variant.

Generally, the higher value the more likely pathogenic variant.

Genmod_score uses config files to define the rank model, which enables customized set-up and versioning of rank models.

The WSM uses the following alternatives and weights in rank model "v1.5":

Rank score range: -25 <= rs <= 23

15.1 Consequence

Each alleles variant effect on individual transcripts are evaluated using a rule-based approach defined by SO-terms. The SO-terms themselves are ranked in order of severity and this ranking is used to defined the weight of the consequence alternative. The performance score is based on the most severe consequence within each gene.

Performance value for the SO-terms:

- transcript_ablation = 5
- splice_donor_variant = 4
- splice_acceptor_variant = 4
- stop_gained = 4
- frameshift_variant = 4
- stop_lost = 4
- initiator_codon_variant = 4
- inframe_insertion = 3
- inframe_deletion = 3
- missense_variant = 3
- transcript_amplification = 3
- splice_region_variant = 3
- incomplete_terminal_codon_variant = 3
- synonymous_variant = 1
- stop_retained_variant = 1

- coding_sequence_variant = 1
- mature_miRNA_variant = 1
- 5_prime_UTR_variant = 1
- 3_prime_UTR_variant = 1
- non_coding_transcript_exon_variant = 1
- non_coding_transcript_variant = 1
- intron_variant = 1
- NMD_transcript_variant = 1
- upstream_gene_variant = 1
- downstream_gene_variant = 1
- TFBS_ablation = 1
- TFBS_amplification = 1
- TF_binding_site_variant = 1
- regulatory_region_variant = 1
- regulatory_region_ablation = 1
- regulatory_region_amplification = 1
- feature_elongation = 1
- feature_truncation = 1
- intergenic_variant = 0

15.2 Frequency

The alternative allele frequency (AF) in public databases (1000G, ExAC). The highest reported alternative frequency reported from the databases is used to calculate the performance value.

Definitions:

- Not reported: AF Na
- Rare: AF <= 0.005
- Intermediate: 0.005 <= AF <= 0.02
- Common: AF > 0.02

Performance value for maximum AF:

- Not reported = 3
- Rare = 2
- Intermediate = 1
- Common = -12

15.3 Inheritance Model(s)

The segregation pattern for the variant within the family. These models are currently annotated using genmod.

Definitions:

- Autsomal Recessive, denoted 'AR_hom'
- Autsomal Recessive denovo, denoted 'AR_hom_dn'
- Autsomal Dominant, 'AD'
- Autsomal Dominant denovo, 'AD_dn'
- Autosomal Compound Heterozygote, 'AR_comp'
- X-linked dominant, 'XD'
- X-linked dominant de novo, 'XD_dn'
- X-linked Recessive, 'XR'
- X-linked Recessive de novo, 'XR_dn'

Performance value for inheritance models:

- Valid model = 1
- No model = -12

15.4 Protein Functional Prediction

The predicted functional effect on the protein. Currently 2 protein effect predictors are used (Sift, PolyPhen2). Each predictors can contribute 1 point each to the overall protein predictor performance score.

SIFT predicts whether an amino acid substitution is likely to affect protein function based on sequence homology and the physico-chemical similarity between the alternate amino acids [1].

PolyPhen-2 predicts the effect of an amino acid substitution on the structure and function of a protein using sequence homology, Pfam annotations, 3D structures from PDB where available, and a number of other databases and tools (including DSSP, ncoils etc [2].

Definitions:

- Sift Terms:
  - "D" Deleterious (score<=0.05)
  - "T" Tolerated (score>0.05)
- PolyPhen2HumVar Terms:
  - "D": Probably damaging (>=0.909)
  - "P": Possibly damaging (0.447<=pp2_hvar<=0.909)
  - "B": Benign (pp2_hvar<=0.446)

Performance value for protein predictors:

- Sift:
  - D = 1
- PolyPhen2HumVar:

  - D or P = 1

15.5 Variant Quality Filter

Each variant call has a filter tranche attached to it indicating the quality of the actual variant call.

Definitions:

- PASS
- Other (Tranches e.g. For GATK [3]: “VQSRTrancheBOTH99.90to100.00”

Performance value for variant quality filter:

- PASS = 3
- Other = 0

15.6 Conservation

The level of conservation for a sequence element (PhastCons [4]), nucleotides or classes of nucleotides PhyloP [5] both from the Phast [6] package as well as genomic constraint score GERP [7] is used. The Phast datasets used in the conservation calculation were generated by the UCSC/Penn State Bioinformatics comparative genomics alignment pipeline. A description of this analysis can be found at UCSC. Each type of conservation can contribute 1 point each to the overall conservation performance score.

Definitions:

- Conserved
  - PhastCons: 0.8 >= Score <= 1
  - GERPRS: Score >= 2
  - PhyloP: Score > 2,5

Performance value for conservation:

- Conserved:
  - PhastCons = 1
  - PhyloP = 1
  - GERP = 1

15.7 Combined Annotation Dependent Depletion (CADD)

CADD is a tool for scoring the deleteriousness of single nucleotide variants as well as insertion/deletions variants in the human genome. C-scores strongly correlate with allelic diversity, pathogenicity of both coding and non-coding variants, and experimentally measured regulatory effects, and also highly rank causal variants within individual genome sequences. The CADD-score is a pre-calculated for all SNVs and for indel from 1000G-project [8].

Definitions:

- Strongly deleterious (CADD > 40)
- deleterious (CADD > 30)

- Mildly deleterious (CADD > 20)
- Probably deleterious (CADD > 10)

Performance value for CADD:

- Strongly deleterious = 4
- Deleterious = 3
- Mildly deleterious = 2
- Probably deleterious = 1

15.8 ClinVar

ClinVar [9] is a freely accessible, public archive of reports of the relationships among human variations and phenotypes, with supporting evidence.

Definitions:

- Uncertain significance = 0
- Not provided = 1
- Benign = 2
- Likely benign = 3
- Likely pathogenic = 4
- Pathogenic = 5
- Drug response = 6
- Histocompatibility = 7
- Other = 255

Performance value for ClinVar:

- Uncertain significance = 0
- Not provided = 0
- Benign = -1
- Likely benign = 0
- Likely pathogenic = 1
- Pathogenic = 2
- Drug response = 0
- Histocompatibility = 0
- Other = 0

CHAPTER 16

svrank_modelv1.0

Genmod score uses the weighted sum model (WSM) approach to rank the most likely pathogenic variant.

Generally, the higher value the more likely pathogenic variant.

Genmod_score uses config files to define the rank model, which enables customized set-up and versioning of rank models.

The WSM uses the following alternatives and weights in the rank model:

Rank score range: -33 <= rs <= 31

16.1 Consequence

Each alleles variant effect on individual transcripts are evaluated using a rule-based approach defined by SO-terms. The SO-terms themselves are ranked in order of severity and this ranking is used to defined the weight of the consequence alternative. The performance score is based on the most severe consequence within each gene.

Performance value for the SO-terms:

- transcript_ablation = 10
- initiator_codon_variant = 9
- frameshift_variant = 8
- stop_gained = 8
- start_lost = 8
- stop_lost = 8
- splice_acceptor_variant = 8
- splice_donor_variant = 8
- inframe_deletion = 5
- transcript_amplification = 5
- splice_region_variant = 5
- missense_variant = 5
- protein_altering_variant = 5
- inframe_insertion = 5
- incomplete_terminal_codon_variant = 5

- non_coding_transcript_exon_variant = 3
- synonymous_variant = 2
- mature_miRNA_variant = 1
- non_coding_transcript_variant = 1
- regulatory_region_variant = 1
- upstream_gene_variant = 1
- regulatory_region_amplification = 1
- TFBS_amplification = 1
- 5_prime_UTR_variant = 1
- intron_variant = 1
- 3_prime_UTR_variant = 1
- feature_truncation = 1
- TF_binding_site_variant = 1
- stop_retained_variant = 1
- feature_elongation = 1
- regulatory_region_ablation = 1
- TFBS_ablation = 1
- coding_sequence_variant = 1
- downstream_gene_variant = 1
- NMD_transcript_variant = 1
- intergenic_variant = 0
- not_reported = 0

16.2 Frequency

The alternative allele frequency (AF) in public databases (1000G). The highest reported alternative frequency reported from the database is used to calculate the performance value.

Definitions:

- Not reported: AF Na
- Very Rare: AF < 0.0005
- Rare: 0.0005 <= AF < 0.005
- Intermediate: 0.005 <= AF < 0.02
- Common: AF >= 0.02

Performance value for maximum AF:

- Not reported = 4
- Very rare = 3
- Rare = 2

- Intermediate = 1
- Common = -12

For *imprecise* variants each confidence interval (SVR1000G and SVL1000G) around the breakpoints are scored:

Definitions:

- Not reported: AF Na
- Rare: 0 <= AF < 0.005
- Intermediate: 0.005 <= AF < 0.02
- Common: AF >= 0.02

Performance value for maximum AF:

- Not reported = 4
- Rare = 2
- Intermediate = 1
- Common = -12

16.3 SV Type

PRECISE variants are scored higher than IMPRECISE variants.

Definitions:

- Precise

Performance value for maximum AF:

- Precise = 3

16.4 SV Length

Length of the structural variant (SVLen). Shorter SVs are scored higher.

Definitions:

- Not reported
- Long: 1000001 <= SVLen <= 100000000
- Medium: 50001 <= SVLen <= 1000000
- Short: 1 <= SVLen <= 50000

Performance value for SVLen:

- Not reported = 0
- Long = -3
- Medium = 3
- Short = 8

16.5 Gene Intolerance Score

EXAC gene intolerance score - calculated by VEP's LoFtool plugin.

Definitions:

- Not reported: LoFtool Na
- Low: LoFtool < 0.0001
- Medium: 0.0001 <= LoFtool < 0.01
- High LoFtool < 0.01

Performance value for gene intolerance score:

- Not reported = 0
- Low = 2
- Medium = 1
- High = 0

16.6 Inheritance Model(s)

The segregation pattern for the variant within the family. These models are currently annotated using genmod models. A variant that is annotated as autosomal compound with no compound partner with a rank score greater than 10 will receive a penalty of -6 to the variants rank score. For single samples this rule will be enforced for variants with inheritance model autosomal dominant, autosomal dominant denovo in addition to the autosomal compound annotation.

Definitions:

- Autosomal Recessive, denoted 'AR_hom'
- Autosomal Recessive denovo, denoted 'AR_hom_dn'
- Autosomal Dominant, 'AD'
- Autosomal Dominant denovo, 'AD_dn'
- Autosomal Compound Heterozygote, 'AR_comp'
- X-linked dominant, 'XD'
- X-linked dominant de novo, 'XD_dn'
- X-linked Recessive, 'XR'
- X-linked Recessive de novo, 'XR_dn'

Performance value for inheritance models:

- Valid model = 1
- No model = -12
- AR_comp penalty = -6

16.7 Variant Quality Filter

Each variant call has a filter tranche attached to it indicating the quality of the actual variant call.

Definitions:

- PASS
- Other (Tranches e.g. For GATK [3]: "VQSRTrancheBOTH99.90to100.00"

We also evaluate the combined GQ score called a Model score for reducing the impact of poor quality genotypes across a case.

Definitions:

- Low quality (GQ => 20)
- High quality (GQ > 20)

Performance value for variant quality filter:

- Filter tranche:
  - PASS = 3
  - Other = 0
- Model score:
  - Low quality = -5
  - High quality = 0

CHAPTER 17

Dynamic Configuration File

MIP uses dynamic configuration files in YAML format to load parameters for each analysis run. An example configuration file can be found here.

To facilitate using different clusters, projects and tailoring the MIP analysis to each run without having to create new configuration files each time you can supply a cluster/project specifc configuration file. Certain paths in the configuration file information will be updated to the current analysis when MIP executes.

This requires that these two entries are added to the configuration file:

1. 'clusterConstantPath: {value}', specifying the project path.
2. 'analysisConstantPath: {value}', specifying the analysis directory.

Entries in the configuration file containing the following "dynamic strings" will be updated in MIP:

- CLUSTERCONSTANTPATH! = 'clusterConstantPath: {value}'
- ANALYSISCONSTANTPATH! = 'analysisConstantPath: {value}'
- ANALYSISTYPE! = 'analysisType: {value}'
- FDN! = '-f familyID' (from command line)

For instance, the pedigree file entry in the configuration file can be supplied like this:

'pedigreeFile: CLUSTERCONSTANTPATH!/ANALYSISTYPE!/FDN!/FDN!_pedigree.txt'

and each path element will be replaced with the corresponding value as specified in the configuration file, command line (precedence) or pedigree file.

Note: Any entries not containing "dynamic strings" will not be modified by MIP.

Both updated and constant entries will be written to the analysis specific folder if specified by '-wc'.

Capture kits info supplied in the configuration file should be on sampleID level:

```
FDN:
  IDN:
    Flag: Entry
```

CHAPTER 18

Pedigree File

Family meta data file. Records important metrics for tracking samples and find biases in isolation of DNA or subsequent sequence analysis.

Among other things, the file enables:

1. Automatic coverage specification (correct target file(s))
2. Application of mendelian filtering models, e.g. *autosomal dominant*, based on pedigree, sex and disease status
3. Collection of analysis info for the sequence analysis pipeline

MIP supports 2 file formats for pedigree metadata PLINK and YAML:

YAML

An example pedigree file with additional metadata can be found at metadata.yaml.

PLINK

The pedigree file format defined by PLINK, although we currently only support tab-sep pedigree files.

The first row should start with a "#" (hash) and contain relevant headers separated by tabs describing each column. The first six columns are mandatory. The name and order of the headers should follow:

Table 18.1: Mandatory Columns

ColumnName	Type	Summary
familyID	`String`	Family identification number (mandatory)
sampleID	`String`	Sample identification number (mandatory)
father	`String`	Father identification number (mandatory)
mother	`String`	Mother identification number (mandatory)
sex	`String`	'1'=male '2'=female 'other'=unknown (mandatory)
phenotype	`String`	'-9'=missing '0'=missing '1'=unaffected '2'=affected (mandatory)

In addition to these mandatory columns we use the pedigree file to record meta data on each individual. Entries within each column should be separated with ";" (semi-colon) and entered in consecutive order. Each individual recorded in the pedigree file is written on one line and a tab should separate each entry. No individual should be recorded twice. The order of individuals below the header line does not matter.

If there is no information on the parents or the grandparents they should be encoded as "0".

An example pedigree file can be found here.

The pedigree file should named: `<FDN>_pedigree.txt`.

Table 18.2: Additional columns in the pedigree file

ColumnName	Default Value	Type	Summary
CMMSID	Na	`String`	The clinics identification number for the individual
Tissue_origin	Na	`String`	Tissue of Isolation (DNA/RNA)
Isolation_kit	Na	`String`	Kit used to isolate nucleic acids
Isolation_date	Na	`Integer`	Date of performing isolation of nucleic acids
Isolation_personnel	Na	`String`	Personnel performing isolation of nucleic acids
Medical_doctor	Na	`String`	Responsible clinician(s)
Inheritance_model	Na	`String`	Probable disease genetic model inheritance within pedigree
Phenotype_terms	Na	`String`	Phenotypic terms associated with the disorder
CMMS_seqID	Na	`String`	Batch identification
SciLifeID	Na	`String`	ScilifeLab identification
Capture_kit	Na	`String`	Capture kit used in library preparation
Capture_date	Na	`Integer`	Date of performing capture procedure
Capture_personnel	Na	`String`	Personnel performing capture procedure
Clustering_date	Na	`Integer`	Date of clustering
Sequencing_kit	Na	`String`	Sequencing kit
Clinical_db	dbCMMS	`String`	The clinical database
Clinical_db_gene_annotation	IEM	`String`	Genes associated with a disease group within the clinical database
Sequencing_type	Na	`String`	Type of sequencing performed

18.1 Pedigree capture kits aliases

- Agilent Sure Select
  - Agilent_SureSelect.V2 => Agilent_SureSelect.V2.GenomeReferenceSourceVersion_targets.bed
  - Agilent_SureSelect.V3 => Agilent_SureSelect.V3.GenomeReferenceSourceVersion_targets.bed
  - Agilent_SureSelect.V4 => Agilent_SureSelect.V4.GenomeReferenceSourceVersion_targets.bed
  - Agilent_SureSelect.V5 => Agilent_SureSelect.V5.GenomeReferenceSourceVersion_targets.bed
  - Agilent_SureSelectCRE.V1 => Agilent_SureSelectCRE.V1.GenomeReferenceSourceVersion_targets.bed
  - Agilent_SureSelectFocusedExome.V1 => Agilent_SureSelectFocusedExome.V1.GenomeReferenceSourceVersion_targets.b
  - Latest => Agilent_SureSelectCRE.V1.GenomeReferenceSourceVersion_targets.bed
- NimbleGen
  - Nimblegen_SeqCapEZExome.V2 => Nimblegen_SeqCapEZExome.V2.GenomeReferenceSourceVersion_targets.bed
  - Nimblegen_SeqCapEZExome.V3 => Nimblegen_SeqCapEZExome.V3.GenomeReferenceSourceVersion_targets.bed

Note: You can use other target region files with MIP but then you have to supply the complete filename with ".bed" ending.

18.1.1 Abbrevations

Abbreviation	Explation
FDN	Family ID
CMMSID	The CMMS sampleID
CMMS SeqID	BatchID e.g. WES8
SciLifeID	The id tag provided by Science for Life Laboratory
AR	Autosomal recessive
AD	Autosomal dominant

CHAPTER 19

Individual Identification Number (IDN)

Ensure that each individual are anonymized and unique. The IDN also facilitates tracking of operations and analyses performed upon the individual.

Note: Changes to the IDN format will be recorded in this document.

19.1 IDN Definition

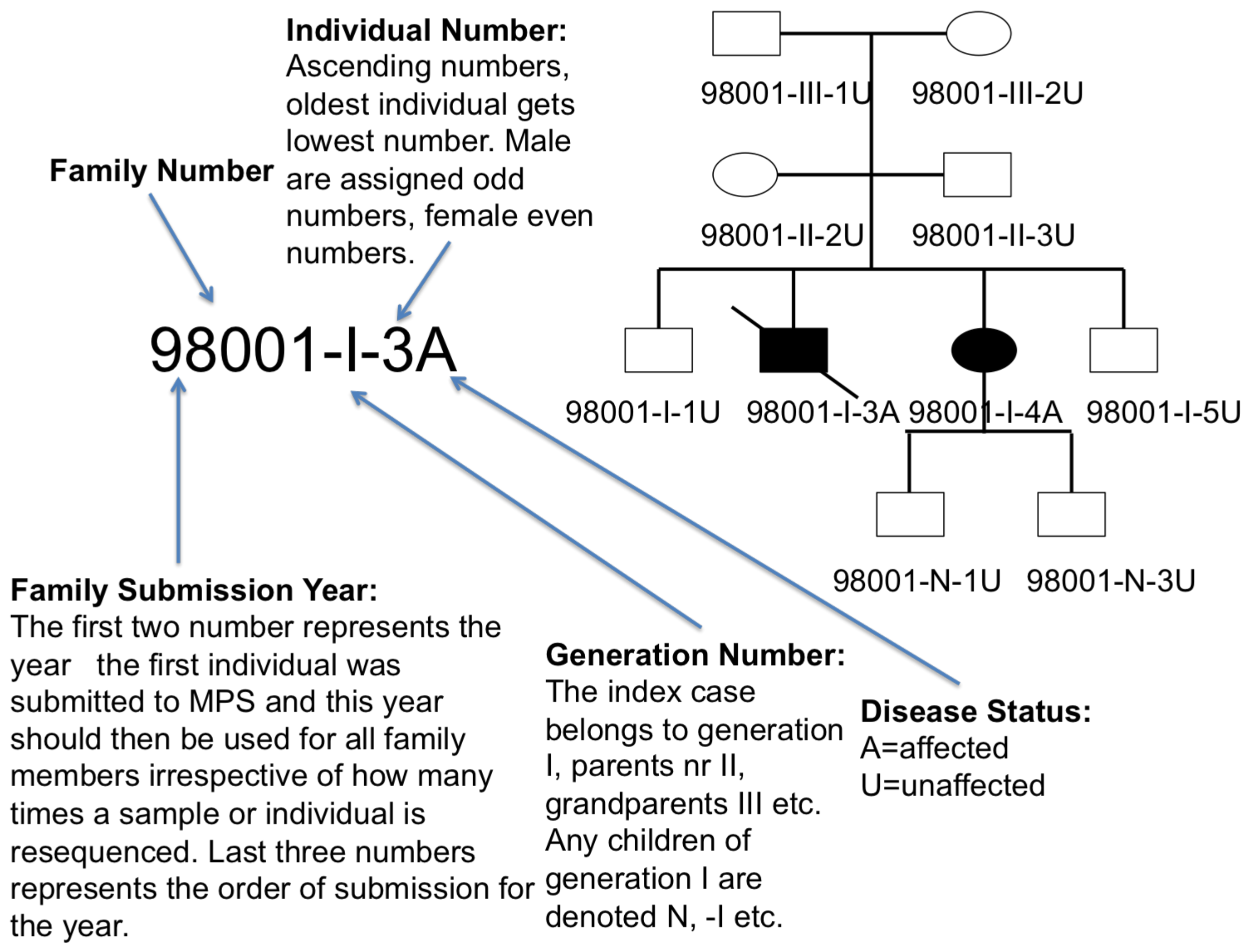
The IDN consists of a three digits connected by a dash and a disease status (DS) letter after the last digit. The disease status letter can be either an "A" denoting affected subjects, a "U" denoting unaffected subjects or "X" for unknown phenotype. Each subject can only have 1 IDN and once set it should never be changed.

- The first digit represents the family identification number (FamilyID/FDN). The two first numbers, in the first digit, represent the year that the first individual of the family was submitted to massively parallel sequencing, MPS. The attached year is determined by the first individual to be submitted to MPS and this year should then be used for all family members irrespective of how many times a sample or individual is resequenced. This rule is enforced to not create multiple IDNs for the same individual and to make sure that all family members are grouped and analyzed in the proper family. The following three numbers are the a continuous number for each family that has been submitted for the year.
- The second digit represents the generation identification number (GenerationID/GDN) within that family. The generation with the affected child is the defined as GDN = "I". Older generation are numbered in ascending order from GDN I, starting with II. Younger generations are numbered in descending order from GDN I starting with "N" (=0 in Roman numerals).
- The last digit represents the subject identification number (SubjectID/SDN) within the family and generation. Male subjects will have odd SDN numbers and female subjects even numbers. The lowest subject IDs will be given to oldest subject within each family and generation and then in ascending order (both even and odd numbers are counted). However, since there can be later additions in the pedigree this is not strictly enforced.
- The letter after the SDN is the disease status (DS) letter, which can be either of three possible letters. A = affected, U = unaffected and X = unknown.

19.1.1 Example

FamilyID.GenerationID.SubjectID(DS) or FDN.GDN.SDN(DS)

A child in the affected child generation being the second oldest male sibling in family 1 and the first to be submitted to sequencing within the family in 1998 would be written as: 98001-I-3A (Figure 1).


Individual Number:
Ascending numbers, oldest individual gets lowest number. Male are assigned odd numbers, female even numbers.
Family Number
98001-I-3A
98001-III-1U
98001-III-2U
98001-II-2U
98001-II-3U
98001-I-1U
98001-I-3A
98001-I-4A
98001-I-5U
98001-N-1U
98001-N-3U
Family Submission Year:
The first two number represents the year the first individual was submitted to MPS and this year should then be used for all family members irrespective of how many times a sample or individual is resequenced. Last three numbers represents the order of submission for the year.
Generation Number:
The index case belongs to generation I, parents nr II, grandparents III etc. Any children of generation I are denoted N, -I etc.
Disease Status:
A=affected
U=unaffected

CHAPTER 20

The Code

20.1 Subroutines

20.1.1 FIDSubmitJob

Handles all communication with SLURM. All jobIDs and SLURM dependencies for all programs are set and submitted here. Each program in MIP belongs to a "path" and together with the sampleID and/or familyID creates a chain of dependencies determining the execution order in SLURM.

Paths

The central flow in MIP is called the *MAIN* path. MIP supports branching from the MAIN path for both familyIDs and sampleIDs. It is also possible to create completely separate paths, which are not associated at all with the MAIN path. However, once branched of from the MAIN path there is currently no support to merge the branch to the trunk (i.e. MAIN path) again.

Each program is supplied with a dependency flag, which determines its dependencies in the path.

Dependency Flags:

```
-1 = Not dependent on earlier scripts, and are self cul-de-sâcs
0 = Not dependent on earlier scripts
1 = Dependent on earlier scripts (within sampleID_path or familyID_path)
2 = Dependent on earlier scripts (within sampleID_path or familyID_path), but are self cul-de-sâcs.
3 = Dependent on earlier scripts and executed in parallel within step
4 = Dependent on earlier scripts and parallel scripts and executed in parallel within step
5 = Dependent on earlier scripts both family and sample and adds to both familyID and sampleId jobs
```

Hash

All jobIDs are saved to the jobID hash using `$jobID{FAMILYID_PATH}{CHAINKEY}`. Only the last jobID(s) required to set the downstrem dependencies are saved.

CHAPTER 21

Indices and tables

- genindex
- modindex
- search