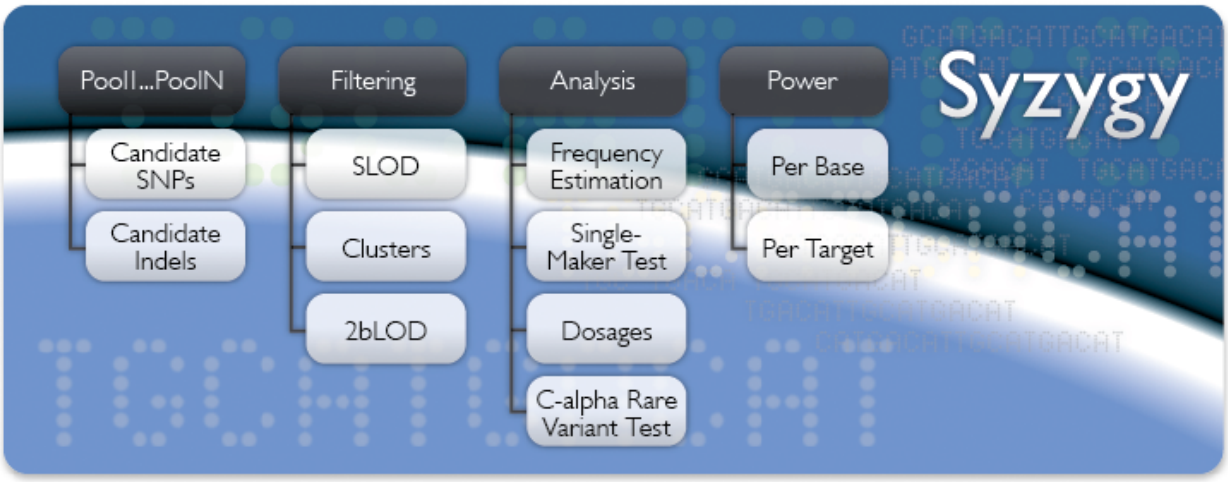


# Syzygy Documentation

***Release 1.1.0***

**Manuel A. Rivas and Mark J. Daly**

October 09, 2011

# CONTENTS

CHAPTER

ONE

# INTRODUCTION

Syzygy is a targeted sequencing post processing analysis tool with short read data that allows:

- Rare variant SNP detection
- Allele frequency estimation
- Single-marker association testing
- Group-wise marker test association
- Pooled experiment statistical summary (% dbSNP , TS/Tv, NS/S)
- VCF output
- Power to detect variant evaluation
- Annotation of rare variants discovered in the pooled sequencing experiment from primary sequencing data in BAM/SAM format.

The implementation of Syzygy could only be made available with the great work by the folks at the 1000 Genomes Project , SAMTools specification, and guidance from Mark Daly and David Altshuler. Latest optimizations and upgraded features made available by Peter Humburg, Andrew Rimmer, and Manuel Rivas.

CHAPTER

TWO

# INSTALLATION

## 2.1 Build Requirements

Python-2.6 or greater and R-2.8 or greater. Other dependencies for Syzygy are downloaded from PyPi.

## 2.2 Installing from Source

Currently Syzygy is available for download from https://sourceforge.net/projects/syzygy/files/syzygy/ .

Installation Instructions: C-shell instructions.

In Unix or Mac:

1. Download Syzygy from Sourceforge site.

   **https://sourceforge.net/projects/syzygy/files/syzygy/**
2. Decompress zip file.

   **tar xvfz syzygy-1.1.0.tar.gz cd syzygy-1.1.0**
3. Make sure PYTHONPATH is write-able and make sure R_LIBS is set with a write-able install path. If not:

   **mkdir $HOME/lib/python2.6/site-packages/**

   **mdkir $HOME/lib/R**
4. Make sure you have a compatible R and Python version running.

   **R –version**

   **python -V**
5. Set Environment Variables.

   **setenv PYTHONPATH=$PYTHONPATH:$HOME/lib/python2.6/site-packages**

   **setenv R_LIBS $HOME/lib/R**
6. Print back R_LIBS and PYTHONPATH

   **echo $PYTHONPATH**

   **echo $R_LIBS**

7. Install Syzygy

   **./configure –prefix=$HOME RLIB=$R_LIBS**

   **make**

   **make install**

8. Run Syzygy

   **./syzygy -h**

   Program should run and show the user parameters

**WARNING** If Numpy and other dependencies raise error upon download re-install numpy using. easy_install –*prefix=$HOME* numpy.

CHAPTER

THREE

# COMMAND LINE OPTIONS

Command Line Options for Syzygy:

**Required Arguments**

**-pif** <pool info file>

See Pool Info File section for input format.

**-tgf** <Target Info file>

See Target Info File section for input format.

**-hg** <integer>

Human Genome Build (e.g. 17 , 18, 19, etc).

**-ref** <fasta>

Reference FASTA file for genome.

**-dbsnp** <dbsnp file>

dbSNP file with all variants already discovered (chr":"position, if not available make pseudo dbsnp file).

**-outputdir** <Output Directory>

Working Directory.

**Optional Arguments**

**-sndb** <true or false>

For second best base filtering if BAM files have annotations available. (default : false)

**-rarethr** <float freq>

Frequency for considering variants as rare. (default : .05)

**-bqthr** <integer>

Base Quality threshold. (default : 22)

**-mqthr** <integer>

Mapping Quality threshold. (default : 1, change this to 20 for STAMPY mapping)

**-ncpu** <integer>

Number of CPUs to use for Parallel execution. Recommended to be equal to number of Pools. (default : 1)

**-mode** <string>

Sequencing Mode. (default :pool, alternate: exome (for individual level sequencing experiment))

**-cluster** <string>

Type of cluster to use for parallel processing. Supports SOCK, MPI, PVM, NWS.

**-hosts** <string>

Lists of hosts to use for socket cluster.

**-power** <boolean>

Indicates whether powe calculations should be carried out.

**-powerIter** <integer>

Number of iterations to use for power calculations.

**-interval** <integer>

Approximate size of regions to split TGF file into (default: 60000 to be used with –module SplitTGF and Submit-Split.py in clusters)

**-bds** <integer>

Bounds for Calpha test to pad target regions. (default : 5)

Syzygy also allows for specific modules to be called instead of the entire pipeline via the –module command.

**-module** <module>

Name of a specific module. Use this to run on single stage of the pipeline.

CHAPTER

FOUR

# ALGORITHM

In this section we describe the algorithm used in Syzygy to call variants in pooled targeted sequencing data. The pooled targeted sequencing experimental design is different in that the reads do not have the identity of the individual it was intended to come from. The pooled targeted sequencing experiment is intended to minimize cost and maximize the utility of the billons of DNA sequence data that is generated by one lane of sequencing run.

## 4.1 Filtering Base Calls

The pooled study design requires optimal sensitivity to detect rare singletion variants in case or control individuals. Currently, the standard error rate of any sequencing run is approximately 1-5%. These error rates make it difficult to detect single copy alleles in a pool when you reach the number of 40-100 chromosomes per pool. The optimal design is to limit the variance and base specific error rates of certain quality scores. It is known that in all current novel sequencing technology quality score models are not extremely accurate and even a point-wise representation of quality score can be 5-10 units away from its true underlying quality score or log10(error probability), which could influence your likelihood computation. Therefore, we filter base calls based on the quality distribution either pre or post calibration and the coverage that can be attained at each of those thresholds.

## 4.2 Likelihood Computation

The Likelihood computation uses Bayes' Rule. For any given number of chromosomes the hypothesis being tested for each pool is:

$$\begin{aligned} H_0 &: \text{reference = number of chromosomes, non-reference = 0} \\ H_a &: \text{reference } \neq \text{ number of chromosomes, non-reference } > 0 \end{aligned}$$

We use Waterson's theta, specifically $\theta = \sum_{n=1}^{\#chrom-1}(1/n)$ , to generate the prior probability for the two hypotheses being tested, (e.g. pool with 100 chromosomes $P(\text{Ref} = 100, \text{non-Ref} = 0) = .99, P(\text{Ref} = 99, \text{non-Ref} = 1) = 0.05$, $P(\text{All others}) = 0.05$.)

The best test between $H_0$ and $H_a$ is based on the ratio of the two densities, the likelihood ratio giving rise to the pooled LOD score.

## 4.3 Error Model

### 4.3.1 Generating Error Estimates

Estimation of error rates is crucial in determining whether or not the non-reference observations are variants or errors. We use error co-variates post thresholding and recalibration that could possibly explain the base-to-base variation observed in empirical miscall rate including:

1. Strand - error rates and specific bases with high error rates are usually uncorrelated between + and - strand reads.
2. Sequence Context - dinucleotides and trinucleotides generally tend to have different error estimates based on its identity.
3. NQS2 (Neighborhood Quality Score 2) - defined as the relative depth of sequence coverage at a query base to the depth of coverage at the ten bases to the left and right. A drop in coverage, arising from bases in otherwise quality reads being dropped because of low confidence, strongly predicts a reduced accuracy at those bases which did get called at that position and strand.

We then proceed with error modeling steps as follows :

1. Remove dbSNP sites from consideration
2. Evaluate Empirical Miscall (mismatch to reference ) rate of positions across the 20,000+ observations for each strand separately.
3. For each position and strand calculate covariates to explore

### 4.3.2 Strand Consistency Filtering

We apply two approaches to testing whether or not the observations in both strand are consistent with each other. Generally, a good proxy for false positives when they are not. As in Sanger sequencing coverage was required on both strands to declare high confidence in polymorphisms, we introduce two tests of strand consistency in order to declare polymorphisms as high quality or low quality. Overall, we require a variant that exceed a combined LOD of 3 in one or more pools to be defined as a SNP, we compute in addition Fisher's exact test comparing the proportion of reference and non-reference alleles between + strand and - strand reads , a significant result indicates the two strands are discordant, whenever one of the strands suggests support for the null hypothesis $LOD_{fwd} < -1.5$ or $LOD_{rev} < -1.5$.

The software computes a strand specific LOD score ("SLOD") that requires an EM iteration of the frequency estimates based on the partition of the pools in the study and tests the null hypothesis of reads coming from + direction and - direction coming from the population of the maximum likelihood estimate of the frequency versus the alternative hypotheses of reads from the + direction coming from the population of the MLE estimate of the frequency and - reads from the population of zero non-reference observations , and of reads in the - direction coming from the population of the MLE estimate of the frequency and + reads coming from the population of zero non-reference observations , taking the maximum of the two likelihoods and comparing to the null hypothesis gives the strand specific LOD (SLOD) which is then used to single out low quality SNP calls with very high frequent non-reference observations in both forward and reverse strands but significantly discordant.

## 4.4 Frequency Estimation and Association Test

Syzygy also allows for estimating frequency of called variants in the classified states and the samples being studied. This is an EM algorithm that iterates over all the pools to search for the MLE estimate of frequency that explains the data. In a similar fashion we have devised a weighted association likelihood ratio test that can be used to prioritize variants for downstream follow up and genotyping in large cohorts. The weighted association likelihood ratio test

takes into account uncertainty in the number of counts of non-reference chromosomes available on a per pool basis. Empirical data demonstrated that overall behavior of sampling of events with one, two, three, and four copies of the non-reference alleles was successful in distinguishing each of those categories and therefore methods developed to reflect such a sampling distribution would give a relatively close approximation to its true underlying statistic. Dosages or posterior probabilities are generated for each of the possible non-reference counts in a per pool basis and experimental-wide. These are the snps.dosage files and snps.dosage.pool files generated by Syzygy.

## 4.5 Rare Variant Testing

The C-alpha manuscript (Neale, B.M., Rivas, M.A., Voight, B.F., Altshuler, D. et al). gives more detail. http://dx.doi.org/doi:10.1371/journal.pgen.1001322

## 4.6 Summary Statistics

Syzygy software also provides summary statistics for the design of the experiment on pools. Generally, the experimenter is interested in whether or not there is ascertainment bias in the set of case and control pools that can severely impact the interpretation of rare-variants in resequencing studies. The software outputs statistics of Power to detect a singleton for each position and pool in the targeted experiment, and this can be easily processed for downstream analysis for the overall differences in power to detect variants at different frequencies for any two classifications (case/control or extreme phenotype). The power estimates are largely impacted by the final coverage and error estimates at each position which are assumed to be fixed after the post processing and error modeling steps.

CHAPTER

FIVE

# INPUT/OUTPUT FILES

Compressing short-read sequence data and information pertaining to their base (base quality scores, 4 base probabilities, 4 base intensities, and other error encodings) and their corresponding alignment properties to some genome reference (mapping quality assignment, duplication properties) is becoming the primary concern in sequencing with next-generation technology. The 1000 Genomes Project has implemented the first version of a strategy to encode most of this information in one standard file format while trying to maintain as small a mark as possible on disk space while retaining most of the core information needed for downstream processing of the data for analysis driven by the design of the project.

Pooled resequencing studies are mainly experiments designed for follow up of regions/exonic targets that have been implicated by genome-wide association studies.

Syzygy integrates the design and processes learned from 1000 Genomes Data Processing coordination to afford a robust tool with standard file formats to analyze such datasets and help follow up of rare-variant detection and downstream analysis.

## 5.1 Input Files

Syzygy currently accepts sequencing input in the BAM/pileup file format. SAM (Sequence Alignment/Map) format is a generic format for storing large nucleotide sequence alignments. BAM is the binary version of the SAM format. BAM/Pileup files can be generated with SAMTools and we expect all users of Syzygy to be comfortable with SAMTools and have it accessible for processing of BAM files. Required files for Syzygy are the target info file (.tgf) and the pool info file (.pif) . The formats are explained in depth below.

### 5.1.1 Target Info File

Experiment design information such as the targets, exons, and genes that were designed to be sequenced should be annotated in the target info file with column headers: (Feature_Name, chr, Start, Stop, Length, genome_build). A Sample lines from such a file might look like this:

| FEATURE_NAME | CHR | START_POSITION | END_POSITION | LENGTH | GENOME_BUILD |
|---|---|---|---|---|---|
| LRRC32_e03 | 11 | 76048296 | 76050200 | 1905 | 18 |
| LRRC32_e02 | 11 | 76054563 | 76054646 | 84 | 18 |
| PTGER4_e03 | 5 | 40727638 | 40728237 | 600 | 18 |

### 5.1.2 Pool Info File

Pooling design information should be annotated in the pif files. The .pif file should have four columns (PoolBAM, Phenotype, Inds, Chroms). The PoolBAM column should specify the location of the BAM file (post pre-processing

, i.e. calibration and merging), the second column, Phenotype, should have a binary label of case/control (1/0) status or phenotypic extreme status; high/low (1/0) , the third column should specify the number of individuals in the corresponding pool, and the fourth column should specify the number of chromosomes in the corresponding pool.

| PoolBAM | Phenotype | Inds | Chroms |
|---|---|---|---|
| 5_NJCDCASES.merged.bam | 1 | 50 | 100 |
| 7_NJCDCASES.merged.bam | 1 | 50 | 100 |
| 18_PRCDCASES.merged.bam | 1 | 50 | 100 |

## 5.2 Output Files

Syzygy generates several files for tracking and downstream description of pooled resequencing datasets. The files generated are:

1. .summary file
2. .SNPcalls
3. .AllPos
4. Dosage
5. Pool by Pool (pbp)
6. RareTest.Calpha
7. Power File

The File formats are described in depth below:

### 5.2.1 Summary File

The Pooled Summary File includes all the variants that were detected and then followed upon for more robust filtering. This file has 16 columns (chr:offset,alleles,gene, type, rs, base_change, annot, pop_nr_freq,f_nr_case, f_nr_con, ml_chisq, lrt_stat, f+ , f- , S_LOD, fisher_med )

| chr:position | alleles | gene | type | rs# | base_chg | annot | dist |
|---|---|---|---|---|---|---|---|
| chr1:1261358 | AG | DVLI | ns | rs1061335 | GGC597GAC | Gly597Asp | NA |

| pop_nr_freq | f_nr_case | f_nr_con | chisqstat_assoc | lrt_stat | f+ | f- | SLOD | p_val |
|---|---|---|---|---|---|---|---|---|
| .001 | .001 | 0 | 1.001 | 1.3 | .0008 | .001 | -20 | 1 |

### 5.2.2 SNPCalls File

The SNPCalls file includes detailed output of the observations made in the forward and reverse strand along with their corresponding LOD scores for positions in the experiment detected as a variant. An example output is provided below:

| Pool | chr:position | ref | Afwd | Cfwd | Gfwd | Tfwd | Dfwd | Ifwd | CovFwd |
|---|---|---|---|---|---|---|---|---|---|
| PoolCase1 | chr1:1261358 | G | A:5 | C:0 | G:630 | T:0 | D:0 | I:0 | 635 |

| Arev | Crev | Grev | Trev | Drev | Irev | Covrev | Ref | Alt | Covtot | LODf | LODR | LODt | flag |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A:4 | C:0 | G:491 | T:0 | D:0 | I:0 | 495 | G | A | 1100 | 2.8 | 1.5 | 5.2 | NA |

### 5.2.3 AllPositions File

The AllPositions file contains all the positions in the experiment for all of the pools combined in one file.

| Pool | chr:position | ref | Afwd | Cfwd | Gfwd | Tfwd | Dfwd | Ifwd | CovFwd |
|---|---|---|---|---|---|---|---|---|---|
| PoolCase1 | chr1:1261357 | G | A:0 | C:1 | G:630 | T:0 | D:0 | I:0 | 631 |
| PoolCase1 | chr1:1261358 | G | A:5 | C:0 | G:630 | T:0 | D:0 | I:0 | 635 |

| Arev | Crev | Grev | Trev | Drev | Irev | Covrev | Ref | Alt | Covtot | LODf | LODR | LODt | flag |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A:0 | C:0 | G:491 | T:0 | D:0 | I:0 | 491 | G | C | 1122 | 0.1 | -20 | -19.5 | NA |
| A:4 | C:0 | G:491 | T:0 | D:0 | I:0 | 495 | G | A | 1130 | 2.8 | 1.5 | 5.2 | NA |

### 5.2.4 Dosage File

The Dosage file contains posterior probabilities of non-reference observations per pool and experimental wide. Per pool (chr:position, pool, [non-ref 0:all chroms]). Experiment wide contains posterior probabilities of non-reference observations per phenotype label (chr:position, case/control, LRT stat, [non-ref 0:all nonref].

Example: 4 chromosomes , non-ref 0-4

| chr:position | Pool | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| chr2:27584444 | Pool1 | .02 | .95 | .03 | 0 | 0 |
| chr2:27584545 | Pool1 | .99 | .01 | 0 | 0 | 0 |
| chr2:27584551 | Pool1 | 0 | .005 | .015 | .10 | .88 |

### 5.2.5 Pool by Pool File

The pool by pool file (.pbp) contains summary statistics on each of the pools themselves. Currently, the statistics that are tracked are Counts of SNPs detected and separated into three bins (Strand LOD: S, S<= 0; 0 < S < 5; S>= 5), transition:transversion ratio, dbSNP %, NS/S ratio.

| Category | High | Poor | FLT |
|---|---|---|---|
| CountTot | 2460 | NA | NA |
| Counts | 2024 | 327 | 33 |
| dbSNP | 0.59 | .08 | .02 |
| NS/S | 1.09 | 1.6 | 0.5 |
| TS/Tv | 2.04 | .92 | .94 |

### 5.2.6 Power File

The Power file contains for each of the pools the power to detect a singleton under the error rate scenario given to each position.

| chr:position | exonid | pool1 | pool2 |
|---|---|---|---|
| chr11:76048291 | LRRC32_e03_cds | 97.0 | 94.0 |
| chr11:76048292 | LRRC32_e03_cds | 98.0 | 95.0 |
| chr11:76048293 | LRRC32_e03_cds | 100.0 | 96.0 |
| chr11:76048294 | LRRC32_e03_cds | 99.0 | 97.0 |
| chr11:76048295 | LRRC32_e03_cds | 99.0 | 95.0 |

CHAPTER

SIX

# DATASET AND SAMPLE COMMANDS

Here is a sample session of working with Syzygy and Crohns Targeted sequencing data with 10 pools : 5 case pools of 50 individuals per pool and 5 control pools of 50 individuals per pool.

1. **mkdir /broad/crohns/**
2. **cd /broad/crohns/**
3. Make symlinks or copy BAM files to /broad/crohns/
   (a) **ln -s /seq/fc/lane/hwd.1.bam Pool1Case.bam**
4. Make TGF file (See TGF File Input Section).
5. Make PIF file (See PIF File Input Section).
6. Run Syzygy. Without LSF (Not Recommended - Memory Intensive Jobs).

   **./syzygy –tgf crohnstarget.tgf –pif crohnstarget.pif –samtoolspath /seq/dirseq/samtools/current/ –ref ~rivas/reference/hg18.fasta –dbsnp ~rivas/reference/dbsnp129.dbsnp –outputdir /broad/crohns/ –hg 18**
7. Run Syzygy. With a compute cluster like LSF (Highly Recommended). -n Number of pools . Syzygy allows parallel processing.

   **bsub -q long -r -R "rusage[mem=4000]" -n 14 ./syzygy –tgf crohnstarget.tgf –pif crohnstarget.pif –samtoolspath /seq/dirseq/samtools/current/ –ref ~rivas/reference/hg18.fasta –dbsnp ~rivas/reference/dbsnp129.dbsnp –outputdir /broad/crohns/ –hg 18 –ncpu 14**

CHAPTER

SEVEN

# MODULES

**mod** *ReadBAM* – SAM Pileup Parsing and Threshold

---

**module:: ReadBAM**

**platform** Unix

**synopsis** Selects Mapping and Base Quality Thresholding to call SNPs.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org><rivas@well.ox.ac.uk> , Peter Humburg <humburg@well.ox.ac.uk>, Andrew Rimmer <rimmer@well.ox.ac.uk>

**mod** *ErrorModel* – Error Model for Pooled Data

---

**module:: GenerateErrorModel**

**platform** Unix

**synopsis** Generates Error Model for Pooled Sequencing Data.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *runrscriptparallel* – Module to Call R Script

---

**module:: runrscriptparallel**

**platform** Unix

**synopsis** Runs R likelihood computation script.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *CleanUpSNPCalls* – Module select candidate SNP and indel calls.

---

**module:: runrscriptparallel**

**platform** Unix

**synopsis** Selects candidate SNP and indel calls for downstream filtering.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *AnnotateSNP* – SNP Annotation Server

---

**module:: AnnotateSNP**

**platform** Unix

**synopsis** Annotates SNPs detected with a prespecified human genome build.

moduleauthor:: Tim Fennell <tfenne@broadinstitute.org>

**mod** *EMFreqAssocStrand* – EM Algorithm to estimate frequencies and dosage

---

**module:: EMFreqAssocStrand**

**platform** Unix

**synopsis** Expectation Maximization Algorithm for Frequency Estimation, Dosage, Strand Test.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *FinalAnnotation* – Generates Final Summary Output of Candidate Variants with Filtering Flags

---

**module:: FinalAnnotation**

**platform** Unix

**synopsis** Generates Candidate SNP List with Filtering Flags along with Annotation and Frequency Estimates and Single Marker-Test Statistic.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *CalphaRareTest* – Apply C-alpha Test for Rare Variant Association Testing

---

**module:: CalphaRareTest**

**platform** Unix

**synopsis** C-alpha Test for Rare Variant Association Testing.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *SyzygyPower* – Bootstrap Evaluate of Power to detect a singleton

---

**module:: SyzygyPower**

**platform** Unix

**synopsis** Evaluates Power to detect a singleton with bootstrapping at every loci in targeted experiment. Run as –power

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

**mod** *VCFConvert* – Syzygy output to VCF file

---

**module:: VCFConvert**

**platform** Unix

**synopsis** Converts Syzygy output to VCF file.

moduleauthor:: Manuel A. Rivas <rivas@broadinstitute.org>

CHAPTER

EIGHT

# FUNCTION LIBRARIES

## 8.1 statslib

Syzygy statistical library functions for Rare-Variant Test statistic, EM algorithms, and others.

**cval** (*p*, *n*, *m*)
Returns denomintor C-alpha test statistic for simple senario.

**Parameters**

- **p** (*float*) – probability case/control , extremes selection.
- **n** (*integer*) – Number of times a variant occurs in the sample collection.
- **m** (*integer*) – Number of variants observed.

**calpha** (*m*, *n*, *y*)
Returns C-alpha test statistic. A Z one-sided test.

**Parameters**

- **y** (*list*) – Counts of Cases in each of the variants.
- **n** (*integer*) – Number of times a variant occurs in the sample collection.
- **m** (*integer*) – Number of variants observed.

**simm** (*m*, *n*, *p1*, *p2*, *w1*, *w2*, *niter*)
Returns a vector of test statistics.

**Parameters**

- **p1** (*float [0-1]*) – Binomial probability of drawing cases in risk variants.
- **p2** (*float [0-1]*) – Binomial probability of drawing cases in control variants.
- **w1** (*float*) – Weight of Risk variant component.
- **w2** (*float*) – Weight of Control Variant component.
- **n** (*integer*) – Number of times a variant occurs in the sample collection.
- **m** (*integer*) – Number of variants observed.

**return_fraction_rare_all** (*chr*, *start*, *stop*, *probweightsdictcases*, *probweightsdictcontrols*, *snplist*)
Attempt to return a weighted C-Statistic.

**return_phat** (*raresnplist*, *snpsall*, *caseall*)
Returns an estimate for p under null.

**Parameters**

- **raresnplist** – Dictionary of all variants discovered.
- **snpsall** – Dictionary of counts in samples for all variants.
- **caseall** – Dictionary of counts in cases for all variants.

**return_rare_snplist** (*snplistall*, *frequencydict*, *freqcutoff*)
Returns dictionary of variants labeled as rare.

**Parameters**

- **snplistall** – Dictionary of counts in samples for all variants.
- **frequencydist** – Dictionary of frequencies of variants.
- **freqcutoff** (*float*) – Cutoff to label a variant as rare.

**reparameterize** (*p*)
Returns the reparameterized version of p to theta

**Parameters** **p** (*float*) – Binomial probability of drawing cases in variants.

**return_rarelist** (*snplist*, *min*, *max*)
Returns dictionary of rare snps based on counts.

**Parameters**

- **snplist** – Dictionary of variants and their counts.
- **min** (*integer*) – Minimum cutoff of counts to include variant in C-alpha.
- **max** (*integer*) – Maximum cutoff of counts to include variant in C-alpha.

**return_rarelist_region** (*snprarelist*, *start*, *stop*, *chr*, *bounds*)
Returns dictionary of rare snps for a region.

**Parameters**

- **snprarelist** – Dictionary of Rare variants.
- **start** (*integer*) – Start Genomic Position of Target.
- **stop** (*integer*) – Stop Genomic Position of Target.
- **chr** (*integer or string*) – Chromosome
- **bounds** – Bounds to pad the targeted region.

**Calpha_GenTnewF** (*varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*)
Returns number of variants used in statistic, and numerator of statistic.

**Parameters**

- **varcnts** – Dictionary of counts in samples for all variants.
- **casecnts** – Dictionary of counts in cases for all variants.
- **chr** (*integer or string*) – Chromosome to apply C-alpha.
- **start** (*integer*) – Start of Region to apply C-alpha.
- **end** (*integer*) – End of Region to apply C-alpha.
- **raresnplist** – Dictionary of rare variants.
- **minvar** (*integer*) – Minimum number counts to consider in C-alpha.
- **maxvar** (*integer*) – Maximum number of counts to consider in C-alpha.
- **theta** (*float [0-1]*) – reparameterized phat.

- **phat** (*float [0-1]*) – Binomial probability of drawing cases in variants.
- **bounds** (*integer*) – Bounds to pad the targeted region.

**Calpha_GenCnew**(*varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*)
Returns number of variants used in statistic, and denominator of statistic.

**Parameters**

- **varcnts** – Dictionary of counts in samples for all variants.
- **casecnts** – Dictionary of counts in cases for all variants.
- **chr** (*integer or string*) – Chromosome to apply C-alpha.
- **start** (*integer*) – Start of Region to apply C-alpha.
- **end** (*integer*) – End of Region to apply C-alpha.
- **raresnplist** – Dictionary of rare variants.
- **minvar** (*integer*) – Minimum number counts to consider in C-alpha.
- **maxvar** (*integer*) – Maximum number of counts to consider in C-alpha.
- **theta** (*float [0-1]*) – reparameterized phat.
- **phat** (*float [0-1]*) – Binomial probability of drawing cases in variants.
- **bounds** (*integer*) – Bounds to pad the targeted region.

**EvalWeightSimm**(*CaseInds*, *ControlInds*, *CaseCounts*, *ControlCounts*)
Returns weight for C-alpha statistic that depends on frequency of variants in populations.

**Parameters**

- **CaseInds** (*integer*) – Number of Case Individuals
- **ControlInds** (*integer*) – Number of Control Individuals
- **CaseCounts** (*integer*) – Counts of case chromosomes with variant
- **ControlCounts** (*integer*) – Counts of control chromosomes with variant

**Calpha_GenTnewWeight**(*varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*, *weights*)
Returns numerator for weighted C-alpha test.

**Calpha_GenCnewWeight**(*varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*, *weights*)
Returns denominator for weighted C-alpha test.

**TnewRoeder**(*y*, *n*, *phat*)
Returns numerator of C-alpha

**Parameters**

- **y** (*int*) – Number of counts of variant in cases
- **n** (*int*) – Number of counts of variant in all samples
- **phat** (*float*) – Binomial probability of drawing cases in variants

**CnewRoeder**(*y*, *n*, *phat*)
Returns denominator of C-alpha

**Parameters**

- **y** (*int*) – Number of counts of variant in cases

- **n** (*int*) – Number of counts of variant in all samples
- **phat** (*float*) – Binomial probability of drawing cases in variants

**return_EM_paramsfixedneut** (*probarray*, *varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*)
Returns EM mixture weights, posterior probabilities, and binomial p estimates.

**Parameters** **probarray** – list of probs [0-1]

**return_gradient** (*probarray*, *varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*, *probcomps*, *piarray*)
Returns Gradient measure of statistic to choose optimal K (number of mixture components).

**return_gradient** (*probarray*, *varcnts*, *casecnts*, *chr*, *start*, *end*, *raresnplist*, *minvar*, *maxvar*, *theta*, *phat*, *bounds*, *probcomps*, *piarray*)
Returns Gradient measure of statistic to choose optimal K (number of mixture components).

**ChooseKGradDiag** (*pval*, *alpha*, *gradientarray1*, *gradientarray2a*, *gradientarray2b*, *gradientarray3*, *probarray*, *probcompvec*)
Returns optimal K based on Gradient Diagnostic that fits data.

**Parameters**

- **pval** (*float*) – pvalue of region statistic
- **alpha** (*float*) – alpha value to reject the null
- **gradientarray1** – gradient under 1 component
- **gradientarray2a** – gradient under 2 componeent (protective, neutral)
- **gradientarray2b** – gradient under 2 component (risk, neutral)
- **gradientarray3** – gradient under 3 components (risk, neutral, protective)
- **probarray** – Binomial probability under the component
- **probcompvec** – Probability of being in component

## 8.2 SAMPileuphelper

Syzygy utility functions written for SAM pileup formats and processing of syzygy outputs.

**pmisspower** (*powerlist*)
Returns probability to miss variant of type with power assignment

**Parameters** **probmiss** – List of Power in each of the pool to detect variant of type.

**return_LRT_assoc** (*casechroms*, *controlchroms*, *weightsarraycase*, *weightsarraycontrol*, *popfreq*, *casefreq*, *controlfreq*)
Returns weighted association LRT statistic for single markers

**Parameters**

- **casechroms** – Number of Chromosomes in Cases.
- **controlchroms** – Number of Chromosomes in Controls.
- **weightsarraycase** – posterior probability/dosage of nonref chromosome counts in case.
- **weightsarraycontrol** – posterior probability/dosage of nonref chromosome counts in controls.
- **popfreq** (*float*) – MLE of Population Frequency.

- **casefreq** (*float*) – MLE of case frequency.
- **controlfreq** (*float*) – MLE of control frequency.
- **casechroms** – int
- **controlchroms** – int

**checktransitiontransversion** (*alleles*)
Returns boolean (T/F) if Ts/Tv change.

**Parameters** **alleles** (*string*) – Alleles of variant

**return_prob_weights** (*snp*, *probvector*, *datadictionary*, *map*)
Returns posterior probabilities of non-reference chromosome counts for a variant discovered in all chromosomes (Cases/Control).

**Parameters** **probvector** – Dictionary of arrays with nonreference dosage for each pool.

**reverse_complement** (*seq*)
Returns reverse complement of string

**Parameters** **seq** (*string*) – Sequence 'ACGT'

**ascii_list** (*s*)
Returns integer value list of ascii values in pileup.

**Parameters** **s** – list of ascii characters

CHAPTER

NINE

# APPENDIX

## 9.1 Weighted Pooled Single-Marker Likelihood Ratio Test (WPLRT)

The standard association likelihood ratio test is given by:

$$LRT = \frac{b(k^*_{case}, n^*_{case}, p^*_{cases})b(k^*_{control}, n^*_{control}, p^*_{control})}{b(k^*_{case}, n^*_{case}, p^*_{pop})b(k^*_{control}, n^*_{control}, p^*_{pop})},$$

where $k^*_{case}, n^*_{case}, k^*_{control}, n^*_{control}$ corresponds to the case count of minor allele chromosomes, case chromosomes, control count of minor allele chromosomes, control chromosomes respectively and $p^*_{case}, p^*_{control}, p^*_{pop}$ corresponds to an allele frequency estimate generated after several iterations (procedure described elsewhere) and $b(k,n,p)$ corresponds to the binomial density function.

The asymptotic approximation is given by:

$$\begin{aligned}
\chi^2_1 &\approx -2 * \log\left[\frac{b(k^*_{case}, n^*_{case}, p^*_{cases})b(k^*_{control}, n^*_{control}, p^*_{control})}{b(k^*_{case}, n^*_{case}, p^*_{pop})b(k^*_{control}, n^*_{control}, p^*_{pop})}\right] \\
\chi^2_1 &\approx -2[(\log(b(k^*_{case}, n^*_{case}, p^*_{cases})) + \log(b(k^*_{control}, n^*_{control}, p^*_{control}))) \\
&- (\log(b(k^*_{case}, n^*_{case}, p^*_{pop})) + \log(b(k^*_{control}, n^*_{control}, p^*_{pop})))]
\end{aligned}$$

We extend this test to apply to pooled sequencing data by adding weights, where weights correspond to $P(data|\text{non-ref} = k)$ , where data is representative of all the data combined from all pools and non-ref = k represents the number of non-reference alleles that are in the samples. Hence,

$$\begin{aligned}
WPLRT &= -2 * [(\overbrace{\sum_{k=0}}^{\text{case chrom}} w_k \log(b(k, n, p^*_{cases})) + \overbrace{\sum_{k=0}}^{\text{control chrom}} w_k \log(b(k, n, p^*_{control}))) \\
&- (\overbrace{\sum_{k=0}}^{\text{case chrom}} w_k \log(b(k, n, p^*_{pop})) + \overbrace{\sum_{k=0}}^{\text{control chrom}} w_k \log(b(k, n, p^*_{pop})))],
\end{aligned}$$

where $w_k = P(\text{pooled experiment data}|\text{ non-ref = k})$.

## 9.2 Computing 2bLOD

With 2nd best base annotation.

After SNP Calling we setup the following hypotheses for downstream filtering with 2nd best base data :

$$\begin{aligned}
\mathcal{H}_0 &: \text{Site is a SNP} \\
\mathcal{H}_a &: \text{Site is Not a SNP}
\end{aligned}$$

$$P\left(\mathcal{H}|\text{ 2ndbest , Trans}\right) = P(\text{ 2ndbest ref }|\mathcal{H}\text{, best nonref,Trans})P(\text{ Trans }|\mathcal{H}),$$

and compute "2bLOD" as $\log_{10}(\mathcal{L}(\mathcal{H}_a)/\mathcal{L}(\mathcal{H}_0))$.

The exact computation of $P(\text{ 2ndbest ref }|\mathcal{H}\text{, best nonref, Trans})$ and $P(\text{ 2ndbest nonref }|\mathcal{H}\text{, best Ref, Trans})$ depends on the following parameter estimates;

$$\begin{aligned}
\mathcal{H}_0 : \hat{p} &= .33 \text{ (or learned transition probability)}, \\
\mathcal{H}_a : \hat{p} &\geq .67 \text{ (keep away from learning unless enough sites )}, \\
\mathcal{H}_0 : \tilde{p} &= .33 \text{ (or learned transition probability)}, \\
\mathcal{H}_a : \tilde{p} &\geq .67 \text{ (keep away from learning unless enough sites )},
\end{aligned}$$

where $\hat{p}$ corresponds to the former and $\tilde{p}$ corresponds to the latter.

## 9.3 Computing SLOD

$$\begin{aligned}
\mathcal{H}_0 &: f^+ = f^- = f^{MLE} \\
\mathcal{H}_a &: f_+ = f^{MLE}, f^- = 0 \\
\mathcal{H}_2 &: f^+ = 0, f^- = f^{MLE-}
\end{aligned}$$

We test the hypothesis against the null by computing $\max(\mathcal{H}_1, \mathcal{H}_2)$ versus $\mathcal{H}_0$. In pooled scenario, the computation follows:

For $n$ pools with $(j_1, j_2, \ldots, j_n)$ chromosomes. The Likelihood of $\mathcal{H}_0$ is calculated as:

$$\mathcal{L}\left(\mathcal{H}_0\right) = \prod_{i \in n} \left[ \sum_{j=0}^{j=j_i} P\left(X = j|\text{ data}\right) \right]$$

Applying Bayes' Theorem,

$$P(X = j|\text{data}) = P(\text{data}^+|X = j)P(X = j|f^+)P(\text{data}^-|X = j)P(X = j|f^-).$$

then,

$$\mathcal{L}\left(\mathcal{H}_0\right) = \prod_{i \in n} \left[ \sum_{j=0}^{j=j_i} P\left(X = j|\text{ data}^+\right) \sum_{j=0}^{j=j_i} P\left(X = j|\text{ data}^-\right) \right]$$

More specifically,

$$P(data|X = j) = \binom{\text{coverage}}{\text{nonrefdata}} (\hat{p})^{\text{nonrefdata}} (1 - \hat{p})^{\text{coverage - nonrefdata}}$$

where $\hat{p} =$ nonref chromosomes/ total chromosomes , and nonref chromosomes is $j$ in the general equation. Also,

$$P(X = j|f) = \binom{\text{chromosomes}}{\text{nonref chromosomes}} (\tilde{p})^{\text{nonref chromosomes}} (1 - \tilde{p})^{\text{chromosomes - nonref chromosomes}}$$

where $\tilde{p} =$ frequency ( + or -).

Next, since we are specifying the hypotheses beforehand, $\mathcal{H}_1$ and $\mathcal{H}_2$ reduces to:

$$\mathcal{L}\left(\mathcal{H}_1\right) = \prod_{i \in n} \left[ \sum_{j=0}^{j=j_i} P\left(X = j | \text{ data}^+\right) P\left(X = 0 | \text{ data}^-\right) \right],$$

and

$$\mathcal{L}\left(\mathcal{H}_2\right) = \prod_{i \in n} \left[ P\left(X = 0 | \text{ data}^+\right) \sum_{j=0}^{j=j_i} P\left(X = j | \text{ data}^-\right) \right].$$

The reported "SLOD" is simply the $\log$ of the ratio of likelihood densities computed for the best supporting alternative hypothesis versus that of the null hypothesis. Since the hypotheses should be supported by all pools then we take the joint likelihood under each hypothesis ($\rightarrow \prod$) .

CHAPTER

TEN

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

# INDEX

## Symbols



## A



## C



## D



## E

## L



## M



## P



## R



## S



## T



## W