CMS Documentation

Release 92c46fb

Broad Institute

July 13, 2016

CHAPTER 1

Contents

1.1 About CMS 2.0

Composite of Multiple Signals (CMS) refers to a family of tests applied to population genetic datasets in order to (i) identify genomic regions that may have been subject to strong recent positive selection (a 'sweep') and (ii) to narrow signals of selection within such regions, in order to identify tractable lists of candidate variants for experimental scrutiny. In both of these cases, CMS requires (a) phased variation data for several populations, along with (b) the identity of the ancestral allele for a majority of sites listed. It was developed with humans in mind (e.g., the 1000 Genomes Project) but could in principle be applied to any diploid species with data in VCF or TPED format.

In its current instantiation (**'CMS 2.0'**), it includes scripts to (i) calculate a variety of selection metrics for each population, (ii) model the demographic history of the dataset using an exploratory approach, (iii) generate probability distributions for each selection metric from data simulated from demographic models, (iv) generate composite scores and (v) visualize signals of selection in the UCSC Genome Browser.

1.1.1 Background

The method used in CMS is described in greater detail in the following papers:

A Composite of Multiple Signals distinguishes causal variants in regions of positive selection Sharon R. Grossman, Ilya Shylakhter, Elinor K. Karlsson, Elizabeth H. Byrne, Shannon Morales, Gabriel Frieden, Elizabeth Hostetter, Elaine Angelino, Manuel Garber, Or Zuk, Eric S. Lander, Stephen F. Schaffner, and Pardis C. Sabeti *Science* 12 February 2010: **327** (5967), 883-886.Published online 7 January 2010 [DOI:10.1126/science.1183863]

Identifying recent adaptations in large-scale genomic data Grossman SR, Andersen KG, Shlyakhter I, Tabrizi S, Winnicki S, Yen A, Park DJ, Griesemer D, Karlsson EK, Wong SH, Cabili M, Adegbola RA, Bamezai RN, Hill AV, Vannberg FO, Rinn JL; 1000 Genomes Project, Lander ES, Schaffner SF, Sabeti PC. *Cell* 14 February 2013: **152** (4), 883-886.Published online 7 January 2010 [DOI:10.1016/j.cell.2013.01.035]

1.1.2 Coalescent simulations

CMS uses simulated population genetic data for a variety of purposes. For the purpose of flexibility, this pipeline is optimized for use with cosi 2, but it would theoretically be straightforward to substitute e.g. Hudson's ms.

Cosi2: an efficient simulator of exact and approximate coalescent with selection. Shlyakhter I, Sabeti PC, Schaffner SF. *Bioinformatics* 1 December 2014: **30** (23), 3427-9.Published online 22 August 2014 [DOI:10.1093/bioinformatics/btu562]

1.2 Installation

1.2.1 System dependencies

To be described in greater detail...

1.2.2 Manual Installation

Step 1: Install Conda

To use conda, you need to install the Conda package manager which is most easily obtained via the Miniconda Python distribution. Miniconda can be installed to your home directory without admin priviledges. On Broad Institute systems, you can make use of the ".anaconda3-4.0.0" dotkit.

Step 2: Configure Conda

Software used by the cms project is distributed through the bioconda channel for the conda package manager. It is necessary to add this channel to the conda config:

```
conda config --add channels bioconda
```

Step 3: Make a conda environment and install cms

It is recommended to install cms into its own conda directory. This ensures its dependencies do not interfere with other conda packages installed on your system. A new conda environment can be created with the following command, which will also install relevant cms dependencies. It is reccommended to use the Python3 version of the environmnent file:

```
conda env create -f=conda-environment_py3.yml -n cms-env
```

Step 4: Activate the cms environment

In order to use cms, you will need to activate its conda environment:

```
source activate cms-env
```

1.3 Command line tools

1.3.1 scans.py

This script contains command-line utilities for calculating EHH-based scans for positive selection in genomes, including EHH, iHS, and XP-EHH.

```
usage: scans.py subcommand
```

Sub-commands:

selscan_file_conversion

Process a bgzipped-VCF (such as those included in the Phase 3 1000 Genomes release) into a gzip-compressed tped file of the sort expected by selscan.

```
usage: scans.py selscan_file_conversion [-h] [--startBp STARTBP]
                                        [--endBp ENDBP] [--ploidy PLOIDY]
                                        [--considerMultiAllelic]
                                        [--rescaleGeneticDistance]
                                        [--includeLowQualAncestral]
```

```
[--codingFunctionClassFile CODINGFUNCTIONCL
[--sampleMembershipFile SAMPLEMEMBERSHIPFIL
[--filterPops FILTERPOPS [FILTERPOPS ...]]
[--filterSuperPops FILTERSUPERPOPS [FILTERS
[--loglevel {DEBUG,INFO,WARNING,ERROR,CRITI
[--version] [--tmpDir TMPDIR]
[--tmpDirKeep]
inputVCF genMap outPrefix outLocation
chromosomeNum
```

Positional arguments:

inputVCF	Input VCF file
genMap	Genetic recombination map tsv file with four columns: (Chromosome, Position(bp), Rate(cM/Mb), Map(cM))
outPrefix	Output file prefix
outLocation	Output location
chromosomeNum	Chromosome number.

Options:

--startBp=0	Coordinate in bp of start position. (default: %(default)s).
--endBp	Coordinate in bp of end position.
--ploidy=2	Number of chromosomes expected for each genotype. (default: %(default)s).
--considerMultiAllelic=False	Include multi-allelic variants in the output as separate records
--rescaleGeneticDistance=False	Genetic distance is rescaled to be out of 100.0 cM
--includeLowQualAncestral=False	Include variants where the ancestral information is low-quality (as indicated by lower-case x for AA=x in the VCF info column) (default: %(default)s).
--codingFunctionClassFile	A python class file containing a function used to code each genotype as '1' and '0'. coding_function(current_value, reference_allele, alternate_allele, ancestral_allele)
--sampleMembershipFile	The call sample file containing four columns: sample, pop, super_pop, gender
--filterPops	Populations to include in the calculation (ex. "FIN")
--filterSuperPops	Super populations to include in the calculation (ex. "EUR")
--loglevel=DEBUG	Verboseness of output. [default: %(default)s]
	Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_ehh

Perform selscan's calculation of EHH.

```
usage: scans.py selscan_ehh [-h] [--gapScale GAPSCALE] [--maf MAF]
                            [--threads THREADS] [--window WINDOW]
                            [--cutoff CUTOFF] [--maxExtend MAXEXTEND]
                            [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTIO
                            [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                            inputTped outFile locusID
```

Positional arguments:

inputTped	Input tped file
outFile	Output filepath
locusID	The locus ID

Options:

--gapScale=20000	Gap scale parameter in bp. If a gap is encountered between two snps > GAP_SCALE and < MAX_GAP, then the genetic distance is scaled by GAP_SCALE/GA (default: %(default)s).
--maf=0.05	Minor allele frequency. If a site has a MAF below this value, the program will not use it as a core snp. (default: %(default)s).
--threads=1	The number of threads to spawn during the calculation. Partitions loci across threads. (default: %(default)s).
--window=100000	When calculating EHH, this is the length of the window in bp in each direction from the query locus (default: %(default)s).
--cutoff=0.05	The EHH decay cutoff (default: %(default)s).
--maxExtend=1000000	The maximum distance an EHH decay curve is allowed to extend from the core. Set <= 0 for no restriction. (default: %(default)s).
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_ihs

Perform selscan's calculation of iHS.

```
usage: scans.py selscan_ihs [-h] [--gapScale GAPSCALE] [--maf MAF]
                            [--threads THREADS] [--skipLowFreq]
                            [--dontWriteLeftRightiHH] [--truncOk]
                            [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTIO
                            [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                            inputTped outFile
```

Positional arguments:

inputTped Input tped file

outFile Output filepath

Options:

--gapScale=20000 Gap scale parameter in bp. If a gap is encountered between two snps > GAP_SCALE and < MAX_GAP, then the genetic distance is scaled by GAP_SCALE/GA (default: %(default)s).

--maf=0.05 Minor allele frequency. If a site has a MAF below this value, the program will not use it as a core snp. (default: %(default)s).

--threads=1 The number of threads to spawn during the calculation. Partitions loci across threads. (default: %(default)s).

--skipLowFreq=False Do not include low frequency variants in the construction of haplotypes (default: %(default)s).

--dontWriteLeftRightiHH=False When writing out iHS, do not write out the constituent left and right ancestral and derived iHH scores for each locus.(default: %(default)s).

--truncOk=False If an EHH decay reaches the end of a sequence before reaching the cutoff, integrate the curve anyway. Normal function is to disregard the score for that core. (default: %(default)s).

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_nsl

Perform selscan's calculation of nSL.

```
usage: scans.py selscan_nsl [-h] [--gapScale GAPSCALE] [--maf MAF]
                            [--threads THREADS] [--truncOk]
                            [--maxExtendNsl MAXEXTENDNSL]
                            [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTIO
                            [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                            inputTped outFile
```

Positional arguments:

inputTped Input tped file

outFile Output filepath

Options:

--gapScale=20000 Gap scale parameter in bp. If a gap is encountered between two snps > GAP_SCALE and < MAX_GAP, then the genetic distance is scaled by GAP_SCALE/GA (default: %(default)s).

--maf=0.05 Minor allele frequency. If a site has a MAF below this value, the program will not use it as a core snp. (default: %(default)s).

--threads=1 The number of threads to spawn during the calculation. Partitions loci across threads. (default: %(default)s).

--truncOk=False If an EHH decay reaches the end of a sequence before reaching the cutoff, integrate the curve anyway. Normal function is to disregard the score for that core. (default: %(default)s).

--maxExtendNsl=100 The maximum distance an nSL haplotype is allowed to extend from the core. Set <= 0 for no restriction. (default: %(default)s).

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_xpehh

Perform selscan's calculation of XPEHH.

```
usage: scans.py selscan_xpehh [-h] [--gapScale GAPSCALE] [--maf MAF]
                              [--threads THREADS] [--truncOk]
                              [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPT
                              [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                              inputTped outFile inputRefTped
```

Positional arguments:

inputTped Input tped file

outFile Output filepath

inputRefTped Input tped for the reference population to which the first is compared

Options:

--gapScale=20000 Gap scale parameter in bp. If a gap is encountered between two snps > GAP_SCALE and < MAX_GAP, then the genetic distance is scaled by GAP_SCALE/GA (default: %(default)s).

--maf=0.05 Minor allele frequency. If a site has a MAF below this value, the program will not use it as a core snp. (default: %(default)s).

--threads=1 The number of threads to spawn during the calculation. Partitions loci across threads. (default: %(default)s).

--truncOk=False If an EHH decay reaches the end of a sequence before reaching the cutoff, integrate the curve anyway. Normal function is to disregard the score for that core. (default: %(default)s).

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_norm_nsl Undocumented

Normalize Selscan's nSL output

```
usage: scans.py selscan_norm_nsl [-h] [--bins BINS]
                                 [--critPercent CRITPERCENT]
                                 [--critValue CRITVALUE] [--minSNPs MINSNPS]
                                 [--qbins QBINS] [--winSize WINSIZE] [--bpWin]
                                 [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXC
                                 [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                 inputFiles [inputFiles ...]
```

Positional arguments:

inputFiles A list of files delimited by whitespace for joint normalization. Expected format for iHS/nSL files (no header):

<locus name> <physical pos> <freq> <ihh1/sL1> <ihh2/sL0> <ihs/nsl> Expected format for XP-EHH files (one line header): <locus name> <physical pos> <genetic pos> <freq1> <ihh1> <freq2> <ihh2> <xpehh>

Options:

--bins=100 The number of frequency bins in [0,1] for score normalization (default: %(default)s)

--critPercent=-1.0 Set the critical value such that a SNP with iHS in the most extreme CRIT_PERCENT tails (two-tailed) is marked as an extreme SNP. Not used by default (default: %(default)s)

--critValue=2.0 Set the critical value such that a SNP with |iHS| > CRIT_VAL is marked as an extreme SNP. Default as in Voight et al. (default: %(default)s)

--minSNPs=10 Only consider a bp window if it has at least this many SNPs (default: %(default)s)

--qbins=20 Outlying windows are binned by number of sites within each window. This is the number of quantile bins to use. (default: %(default)s)

--winSize=100000 GThe non-overlapping window size for calculating the percentage of extreme SNPs (default: %(default)s)

--bpWin=False If set, will use windows of a constant bp size with varying number of SNPs

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_norm_ihs Undocumented

Normalize Selscan's iHS output

```
usage: scans.py selscan_norm_ihs [-h] [--bins BINS]
                                 [--critPercent CRITPERCENT]
                                 [--critValue CRITVALUE] [--minSNPs MINSNPS]
                                 [--qbins QBINS] [--winSize WINSIZE] [--bpWin]
                                 [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXC
                                 [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                 inputFiles [inputFiles ...]
```

Positional arguments:

inputFiles	A list of files delimited by whitespace for joint normalization. Expected format for iHS/nSL files (no header): <locus name> <physical pos> <freq> <ihh1/sL1> <ihh2/sL0> <ihs/nsl> Expected format for XP-EHH files (one line header): <locus name> <physical pos> <genetic pos> <freq1> <ihh1> <freq2> <ihh2> <xpehh>

Options:

--bins=100	The number of frequency bins in [0,1] for score normalization (default: %(default)s)
--critPercent=-1.0	Set the critical value such that a SNP with iHS in the most extreme CRIT_PERCENT tails (two-tailed) is marked as an extreme SNP. Not used by default (default: %(default)s)
--critValue=2.0	Set the critical value such that a SNP with |iHS| > CRIT_VAL is marked as an extreme SNP. Default as in Voight et al. (default: %(default)s)
--minSNPs=10	Only consider a bp window if it has at least this many SNPs (default: %(default)s)
--qbins=20	Outlying windows are binned by number of sites within each window. This is the number of quantile bins to use. (default: %(default)s)
--winSize=100000	GThe non-overlapping window size for calculating the percentage of extreme SNPs (default: %(default)s)
--bpWin=False	If set, will use windows of a constant bp size with varying number of SNPs
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

selscan_norm_xpehh Undocumented

Normalize Selscan's XPEHH output

```
usage: scans.py selscan_norm_xpehh [-h] [--bins BINS]
                                   [--critPercent CRITPERCENT]
                                   [--critValue CRITVALUE] [--minSNPs MINSNPS]
                                   [--qbins QBINS] [--winSize WINSIZE]
                                   [--bpWin]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,E
                                   [--version] [--tmpDir TMPDIR]
                                   [--tmpDirKeep]
                                   inputFiles [inputFiles ...]
```

Positional arguments:

inputFiles A list of files delimited by whitespace for joint normalization. Expected format for iHS/nSL files (no header):

<locus name> <physical pos> <freq> <ihh1/sL1> <ihh2/sL0> <ihs/nsl> Expected format for XP-EHH files (one line header): <locus name> <physical pos> <genetic pos> <freq1> <ihh1> <freq2> <ihh2> <xpehh>

Options:

--bins=100 The number of frequency bins in [0,1] for score normalization (default: %(default)s)

--critPercent=-1.0 Set the critical value such that a SNP with iHS in the most extreme CRIT_PERCENT tails (two-tailed) is marked as an extreme SNP. Not used by default (default: %(default)s)

--critValue=2.0 Set the critical value such that a SNP with |iHS| > CRIT_VAL is marked as an extreme SNP. Default as in Voight et al. (default: %(default)s)

--minSNPs=10 Only consider a bp window if it has at least this many SNPs (default: %(default)s)

--qbins=20 Outlying windows are binned by number of sites within each window. This is the number of quantile bins to use. (default: %(default)s)

--winSize=100000 GThe non-overlapping window size for calculating the percentage of extreme SNPs (default: %(default)s)

--bpWin=False If set, will use windows of a constant bp size with varying number of SNPs

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

store_selscan_results_in_db

Aggregate results from selscan in to a SQLite database via helper JSON metadata file.

```
usage: scans.py store_selscan_results_in_db [-h]
                                            [--loglevel {DEBUG,INFO,WARNING,ERROR,C
                                            [--version] [--tmpDir TMPDIR]
                                            [--tmpDirKeep]
                                            inputFile outFile
```

Positional arguments:

inputFile	Input *.metadata.json file
outFile	Output SQLite filepath

Options:

--loglevel=INFO	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

1.3.2 cms_modeller.py

This script contains command-line utilities for exploratory fitting of demographic models to population genetic data.

```
usage: cms_modeller.py [-h] {target_stats,bootstrap,point,grid,optimize} ...
```

Sub-commands:

target_stats perform per-site(/per-site-pair) calculations of population summary statistics for model target values

```
usage: cms_modeller.py target_stats [-h] [--freqs] [--ld] [--fst]
                                    inputTpeds recomFile regions out
```

Positional arguments:

inputTpeds	comma-delimited list of input tped files (only one file per pop being modelled; must run chroms separately or concatenate)
recomFile	recombination map
regions	tab-separated file with putative neutral regions
out	outfile prefix

Options:

--freqs=False	calculate summary statistics from within-population allele frequencies
--ld=False	calculate summary statistics from within-population linkage disequilibrium
--fst=False	calculate summary statistics from population comparison using allele frequencies

bootstrap perform bootstrap estimates of population summary statistics in order to finalize model target values

```
usage: cms_modeller.py bootstrap [-h] [--in_freqs IN_FREQS] [--in_ld IN_LD]
                                 [--in_fst IN_FST]
                                 nBootstrapReps out
```

Positional arguments:

nBootstrapReps	number of bootstraps to perform in order to estimate standard error of the dataset (should converge for reasonably small n)
out	outfile prefix

Options:

--in_freqs	comma-delimited list of infiles with per-site calculations for population. One file per population – for bootstrap estimates of genome-wide values, should first concatenate per-chrom files
--in_ld	comma-delimited list of infiles with per-site-pair calculations for population. One file per population – for bootstrap estimates of genome-wide values, should first concatenate per-chrom files
--in_fst	comma-delimited list of infiles with per-site calculations for population pair. One file per population-pair – for bootstrap estimates of genome-wide values, should first concatenate per-chrom files

point run simulates of a point in parameter-space

```
usage: cms_modeller.py point [-h] [--cosiBuild COSIBUILD]
                             [--dropSings DROPSINGS] [--genmapRandomRegions]
                             [--stopAfterMinutes STOPAFTERMINUTES]
                             [--calcError CALCERROR]
                             [--targetvalsFile TARGETVALSFILE] [--plotStats]
                             inputParamFile nCoalescentReps outputDir
```

Positional arguments:

inputParamFile	file with model specifications for input
nCoalescentReps	num reps
outputDir	location to write cosi output

Options:

--cosiBuild=/Users/vitti/Desktop/COSI_DEBUG_TEST/cosi-2.0/coalescent	which version of cosi to run? (*automate installation)
--dropSings	randomly thin global singletons from output dataset (i.e., to model ascertainment bias)
--genmapRandomRegions=False	cosi option to sub-sample genetic map randomly from input
--stopAfterMinutes	cosi option to terminate simulations
--calcError	file specifying dimensions of error function to use. if unspecified, defaults to all. first line = stats, second line = pops
--targetvalsFile	targetvalsfile for model
--plotStats=False	visualize goodness-of-fit to model targets

grid run grid search

```
usage: cms_modeller.py grid [-h] [--cosiBuild COSIBUILD]
                            [--dropSings DROPSINGS] [--genmapRandomRegions]
                            [--stopAfterMinutes STOPAFTERMINUTES]
                            [--calcError CALCERROR]
                            inputParamFile nCoalescentReps outputDir
                            grid_inputdimensionsfile
```

Positional arguments:

inputParamFile file with model specifications for input

nCoalescentReps num reps

outputDir location to write cosi output

grid_inputdimensionsfile file with specifications of grid search. each parameter to vary is indicated: KEY INDEX [VALUES]

Options:

--cosiBuild=/Users/vitti/Desktop/COSI_DEBUG_TEST/cosi-2.0/coalescent which version of cosi to run? (*automate installation)

--dropSings randomly thin global singletons from output dataset (i.e., to model ascertainment bias)

--genmapRandomRegions=False cosi option to sub-sample genetic map randomly from input

--stopAfterMinutes cosi option to terminate simulations

--calcError file specifying dimensions of error function to use. if unspecified, defaults to all. first line = stats, second line = pops

optimize run optimization algorithm to fit model parameters

```
usage: cms_modeller.py optimize [-h] [--cosiBuild COSIBUILD]
                                [--dropSings DROPSINGS]
                                [--genmapRandomRegions]
                                [--stopAfterMinutes STOPAFTERMINUTES]
                                [--calcError CALCERROR] [--stepSize STEPSIZE]
                                [--method METHOD]
                                inputParamFile nCoalescentReps outputDir
                                optimize_inputdimensionsfile
```

Positional arguments:

inputParamFile file with model specifications for input

nCoalescentReps num reps

outputDir location to write cosi output

optimize_inputdimensionsfile file with specifications of optimization. each parameter to vary is indicated: KEY INDEX

Options:

--cosiBuild=/Users/vitti/Desktop/COSI_DEBUG_TEST/cosi-2.0/coalescent which version of cosi to run? (*automate installation)

--dropSings randomly thin global singletons from output dataset (i.e., to model ascertainment bias)

--genmapRandomRegions=False cosi option to sub-sample genetic map randomly from input

--stopAfterMinutes cosi option to terminate simulations

--calcError file specifying dimensions of error function to use. if unspecified, defaults to all. first line = stats, second line = pops

--stepSize scaled step size (i.e. whole range = 1)

--method=SLSQP algorithm to pass to scipy.optimize

1.3.3 likes_from_model.py

This script contains command-line utilities for generating probability distributions for component scores from pre-specified demographic model(s).

```
usage: likes_from_model.py [-h]
                           {run_neut_sims,get_sel_trajs,run_sel_sims,scores_from_sims,likes
                           ...
```

Sub-commands:

run_neut_sims run neutral simulations

```
usage: likes_from_model.py run_neut_sims [-h] [--cosiBuild COSIBUILD]
                                         [--dropSings DROPSINGS]
                                         [--genmapRandomRegions]
                                         n inputParamFile outputDir
```

Positional arguments:

n num replicates to run

inputParamFile file with model specifications for input

outputDir location to write cosi output

Options:

--cosiBuild=/Users/vitti/Desktop/COSI_DEBUG_TEST/cosi-2.0/coalescent which version of cosi to run? (*automate installation)

--dropSings randomly thin global singletons from output dataset to model ascertainment bias

--genmapRandomRegions=False cosi option to sub-sample genetic map randomly from input

get_sel_trajs run forward simulations of selection trajectories and perform rejection sampling to populate selscenarios by final allele frequency before running coalescent simulations for entire sample

```
usage: likes_from_model.py get_sel_trajs [-h] [--cosiBuild COSIBUILD]
                                         [--dropSings DROPSINGS]
                                         [--genmapRandomRegions]
                                         [--freqRange FREQRANGE]
                                         [--nBins NBINS]
                                         nSimsPerBin maxSteps inputParamFile
                                         outputDir
```

Positional arguments:

nSimsPerBin number of selection trajectories to generate per allele frequency bin

maxSteps number of attempts to generate a selection trajectory before re-sampling selection coefficient and start time.

inputParamFile file with model specifications for input

outputDir location to write cosi output

Options:

--cosiBuild=/Users/vitti/Desktop/COSI_DEBUG_TEST/cosi-2.0/coalescent which version of cosi to run? (*automate installation)

--dropSings randomly thin global singletons from output dataset to model ascertainment bias

--genmapRandomRegions=False cosi option to sub-sample genetic map randomly from input

--freqRange=.05-.95 range of final selected allele frequencies to simulate, e.g. .05-.95

--nBins=9 number of frequency bins

run_sel_sims run sel. simulations

```
usage: likes_from_model.py run_sel_sims [-h] [--cosiBuild COSIBUILD]
                                        [--dropSings DROPSINGS]
                                        [--genmapRandomRegions]
                                        [--freqRange FREQRANGE]
                                        [--nBins NBINS]
                                       n trajDir inputParamFile outputDir
```

Positional arguments:

n num replicates to run per sel scenario

trajDir location of simulated trajectories (i.e. outputDir from get_sel_trajs)

inputParamFile file with model specifications for input

outputDir location to write cosi output

Options:

--cosiBuild=/Users/vitti/Desktop/COSI_DEBUG_TEST/cosi-2.0/coalescent which version of cosi to run? (*automate installation)

--dropSings randomly thin global singletons from output dataset to model ascertainment bias

--genmapRandomRegions=False cosi option to sub-sample genetic map randomly from input

--freqRange=.05-.95 range of final selected allele frequencies to simulate, e.g. .05-.95

--nBins=9 number of frequency bins

scores_from_sims get scores from simulations

```
usage: likes_from_model.py scores_from_sims [-h] [--inputTped INPUTTPED]
                                            [--inputIhs INPUTIHS]
                                            [--inputdelIhh INPUTDELIHH]
                                            [--inputXpehh INPUTXPEHH] [--ihs]
```

```
                                                [--delIhh] [--xpehh XPEHH]
                                                [--fst_deldaf FST_DELDAF]
                                                [--normalizeIhs NORMALIZEIHS]
                                                [--normalizeDelIhh NORMALIZEDELIHH]
                                                [--normalizeXpehh NORMALIZEXPEHH]
                                                outputFilename
```

Positional arguments:

outputFilename	where to write scorefile

Options:

--inputTped	tped from which to calculate score
--inputIhs	iHS from which to calculate delihh
--inputdelIhh	delIhh from which to calculate norm
--inputXpehh	Xp-ehh from which to calculate norm
--ihs=False	Undocumented
--delIhh=False	Undocumented
--xpehh	inputTped for altpop
--fst_deldaf	inputTped for altpop
--normalizeIhs	filename for parameters to normalize to; if not given then will by default normalize file to its own global dist
--normalizeDelIhh	filename for parameters to normalize to; if not given then will by default normalize file to its own global dist
--normalizeXpehh	filename for parameters to normalize to; if not given then will by default normalize file to its own global dist

likes_from_scores get component score probability distributions from scores

```
usage: likes_from_model.py likes_from_scores [-h] [--thinToSize] [--ihs]
                                             [--delihh] [--xp] [--deldaf]
                                             [--fst] [--freqRange FREQRANGE]
                                             [--nBins NBINS]
                                             neutFile selFile selPos
                                             nLikesBins outPrefix
```

Positional arguments:

neutFile	file with scores for neutral scenarios (normalized if necessary)
selFile	file with scores for selected scenarios (normalized if necessary)
selPos	position of causal variant
nLikesBins	number of bins to use for histogram to approximate probability density function
outPrefix	save file as

Options:

--thinToSize=False	subsample from simulated SNPs (since nSel << nLinked < nNeut)
--ihs=False	Undocumented

--delihh=False Undocumented

--xp=False Undocumented

--deldaf=False Undocumented

--fst=False Undocumented

--freqRange=.05-.95 range of final selected allele frequencies to simulate, e.g. .05-.95

--nBins=9 number of frequency bins

1.3.4 composite.py

This script contains command-line utilities for combining component statistics – i.e., the final step of the CMS 2.0 pipeline.

```
usage: composite.py [-h]
                    {poppair,outgroups,bayesian_gw,bayesian_region,ml_region}
                    ...
```

Sub-commands:

poppair collate all component statistics for a given population pair (as a prerequisite to more sophisticated group comparisons

```
usage: composite.py poppair [-h] [--xp_reverse_pops] [--deldaf_reverse_pops]
                            in_ihs_file in_delihh_file in_xp_file
                            in_fst_deldaf_file outfile
```

Positional arguments:

in_ihs_file file with normalized iHS values for putative selpop

in_delihh_file file with normalized delIhh values for putative selpop

in_xp_file file with normalized XP-EHH values

in_fst_deldaf_file file with Fst, delDaf values for poppair

outfile file to write with collated scores

Options:

--xp_reverse_pops=False include if the putative selpop for outcome is the altpop in XPEHH (and vice versa)

--deldaf_reverse_pops=False finclude if the putative selpop for outcome is the altpop in delDAF (and vice versa)

outgroups combine scores from comparisons of a putative selected pop to 2+ outgroups.

```
usage: composite.py outgroups [-h] infiles likesfile outfile
```

Positional arguments:

infiles comma-delimited set of pop-pair comparisons

likesfile text file where probability distributions are specified for component scores

outfile file to write with finalized scores

bayesian_gw default algorithm and weighting, genome-wide

```
usage: composite.py bayesian_gw [-h] inputparamfile
```

Positional arguments:

inputparamfile file with specifications for input

bayesian_region default algorithm and weighting, within-region

```
usage: composite.py bayesian_region [-h]
                                    chrom startBp endBp selPop altPops
                                    demModel
```

Positional arguments:

chrom chromosome containing region

startBp start location of region in basepairs

endBp end location of region in basepairs

selPop Undocumented

altPops comma-delimited

demModel Undocumented

ml_region machine learning algorithm (within-region)

```
usage: composite.py ml_region [-h] chrom startBp endBp selPop altPops demModel
```

Positional arguments:

chrom chromosome containing region

startBp start location of region in basepairs

endBp end location of region in basepairs

selPop Undocumented

altPops comma-delimited

demModel Undocumented

1.4 Sample workflow

CMS provides a computational framework to explore signals of natural selection in diploid organisms. This section describes how to do so at an abstract level.

1.4.1 Preliminary considerations

CMS is a computational and statistical framework for exploring the evolution of populations within a species at a genomic level. To that end, the user must first provide a dataset containing genotype calls for individuals in at least one putative selected population and at least one putative 'outgroup.' CMS 2.0 is designed to be flexible with respect to the number and configuration of input populations – that is, given input of however many populations, the user can easily calculate CMS scores for any configuration of these populations. Nonetheless, CMS still relies on the user to define these populations appropriately.

- To determine or confirm appropriate population groupings, identify outliers, etc., we recommend that users first characterize their dataset using such methods as likeliness clustering (e.g. STRUCTURE), principal components analysis, or phylogenetic methods (see e.g. SNPRelate)
- Each population should be randomly thinned to the same number of individuals, none of whom should be related within the past few generations.

- Larger samples are generally preferable (e.g. 50-100+ diploid individuals per population). However, depending on such factors as the landscape of recombination in the species, the quality/density of genotype data, and the extent of neutral genetic divergence between represented populations, it may be possible to leverage smaller datasets. As CMS necessitates the generation of a demographic model for the given dataset, the user is advised to use their model to generate simulated data with which to perform power estimations.

1.4.2 Data formatting

CMS requires the user to provide population genetic (i.e., within-species) diversity data, including genotype phase and allele polarity.

- If your dataset is **unphased**, you can preprocess it using a program like Beagle or PLINK.
- The identity of the **ancestral allele** at each site is typically determined by comparison to outgroups at orthologous sites. Inferred ancestral sequence is available for a number of species through e.g. Ensembl via their ftp. You can use VCFtools to populate the "AA=" section of your VCF's INFO field.
- In most cases, the user will want to provide a **genetic recombination map**. If this is unavailable, CMS will assume uniform recombination rates when calculating haplotype scores. Human recombination maps are available from the HapMap Project.
- CMS works with TPED datafiles, and includes support to convert from VCF using the command line tool scans.py.

1.4.3 Demographic modeling

CMS combines several semi-independent component tests for selection in a Bayesian or Machine Learning framework. In the former case, a demographic model for the species in question is critical in order to furnish posterior distributions of scores for said component tests under alternate hypotheses of neutrality or selection. Put otherwise: a demographic model is a (conjectural) descriptive historical account of our dataset, including population sizes and migration rates across time, that can be used to generate simulated data that 'looks like' our original dataset. We then simulate many scenarios of selection in order and calculate the distributions of component scores for adaptive, linked, and neutral variants. These distributions form the basis of our Bayesian classifier. We can also circumvent the need to define posterior score distributions by using simulated data as training data for Machine Learning implementations of CMS.

Our modeling framework is designed to accomodate an arbitrary number of populations in a tree of arbitrary complexity; as such, it is designed to be exploratory, allowing users to iteratively perform optimizations while visualizing the effect on model goodness-of-fit. For rigorous demographic inference (i.e., in the case of a model with known topology and tractably few parameters), users may consider programs such as dadi or diCal.

Following Schaffner et al 2005 , our framework calculates a range of population summary statistics as target values, and defines error as the Root Mean Square discrepancy between target and simulated values. These summary statistics are calculated by bootstrap estimate from user-specified putative neutral regions. For human populations, the Neutral Regions Explorer is a useful resource.

The user must specify tree topology and ranges for parameter values. These can be added and removed as desired through the script params.py. After target values have been estimated and model topology defined, the user can iteratively search through subsets of parameter-space using cms_modeller.py with a masterfile specifying search input.

1.4.4 Calculating selection statistics

CMS packages a number of previously described population genetic tests for recent positive selection. Haplotype scores are calculated using selscan.

1.4.5 Combining scores

CMS 2.0 allows users to define CMS scores flexibly with respect to (i) number and identity of putative selected/neutral populations, (ii) assumed demographic model, (iii) input component scores, (iv) method of score combination. In each case the user should motivate their choices and consider how robust a putative signal of selection is to variation or arbitrariness in these factors.

1.4.6 Identifying regions

CMS is motivated by the need to resolve signals of selection – that is, to identify genetic variants that confer adaptive phenotypes. Because selective events can alter patterns of population genetic diversity across large genomic regions, we take a two-step approach to this goal: we first identify putative selected regions (using CMS, another framework, prior knowledge, etc.), and then examine each region with CMS to identify a tractable list of candidate variants for further scrutiny.

1.4.7 Localizing signals

Once regions are defined, we can reapply our composite framework in order to thin our list of candidate variants for further scrutiny and prioritize those sites that have the strongest evidence of selection (or other compelling evidence, e.g. overlap with known or predicted functional elements).