BioBin User Guide

Current version: BioBin 2.3

Last modified: April 2017

Ritchie Lab

Geisinger Health System

URL: http://www.ritchielab.com/software/biobin-download

Email: software@ritchielab.psu.edu

Table of Contents

Overview

BioBin is a comprehensive bioinformatics tool for biologically-driven binning and association analysis of rare variants (RVs) in sequence data. It automates multi-level binning or collapsing of RVs into biological features such as genes, pathways, regulatory regions, evolutionary conserved regions (ECRs), and many others. BioBin improves on other variant binning algorithms through the use of prior biological information, which can highlight the potential cumulative effects of biologically aggregated RVs. BioBin requires the Library of Knowledge Integration (LOKI), which contains diverse biological knowledge from multiple collections of publically available databases. For applications of BioBin please visit the following cited publications [1–4].

BioBin

BioBin is a stand-alone command line application written in C++ that uses a prebuilt LOKI database. Source distributions are available for Mac and Linux operating systems and require minimal prerequisites to compile. Included in the distribution are tools that allow the user to create and update the LOKI database by downloading information directly from source websites. The computational requirements for BioBin are quite modest. For example, during testing, a whole-genome analysis of 185 people took just over two hours using a single core on a cluster (Intel Xeon X5675 3.06 GHz processor). However, because the vast amount of data included in the analysis must be stored in memory, the requirements for memory usage can be high. The aforementioned whole-genome analysis required approximately 13 GB of memory to complete. Even with large datasets, BioBin can be run quickly without access to expensive and specialized computer hardware or a computing cluster. The number of variants is the primary driver of memory usage.

Library of Knowledge Integration (LOKI) Database

Harnessing prior biological knowledge is a powerful way to inform collapsing feature boundaries. BioBin relies on the LOKI for database integration and boundary definitions. LOKI contains resources such as: the National Center for Biotechnology (NCBI) dbSNP and gene Entrez database information, Kyoto Encyclopedia of Genes and Genomes (KEGG) [5], Reactome [6], Gene Ontology (GO) [7], Protein families database (Pfam)[8], NetPath signal transduction pathways [9], Molecular INTeraction database (MINT) [10], Biological General Repository for Interaction Datasets (BioGrid) [11] , Pharmacogenomics Knowledge Base (PharmGKB), [12] Open Regulatory Annotation Database (ORegAnno) [13], as well as information from UCSC Genome Browser [14] about evolutionary conserved regions.

LOKI is used to provide a standardized interface and terminology to disparate sources each containing individual means of representing data. The three main concepts used in LOKI are *positions*, *regions,* and *groups*. The term *position* refers to single nucleotide polymorphisms (SNPs), single nucleotide variants (SNVs), or RVs. The definition of *region* can be applied to a broader scope of biology. Any segment with a start and stop position can be defined as a region, including genes, copy number variants (CNVs), insertions and deletions, and evolutionary conserved regions (ECRs). *Sources* are external databases, such as those listed above. A *group* is a set of organized data that has interconnected information. Groups include protein families, ontological groups, or any other groupings of genes, proteins or genomic regions provided by the external database sources.

LOKI is implemented in SQLite, a relational database management system, which does not require a

dedicated database server. Initial installation of LOKI requires approximately 100GB of disk storage necessary to accommodate the LOKI database file, source data, as well as python installer scripts. An updater script will automatically process and combine information from the various sources into a single database file (some of the temporary files are removed during this process). Once the build is complete, the LOKI database file required by BioBin will be under 25GB. The script to build LOKI is open source, publicly available on the Ritchie lab website (http://www.ritchielab.psu.edu/software/biobin-download), and is included with the BioBin software. Users with knowledge of relational databases can customize their LOKI database by including or excluding sources, providing additional sources, and updating source information as frequently as needed. LOKI runs locally and must be managed locally; however, research groups can also customize LOKI for their own purpose.

BioBin Workflow
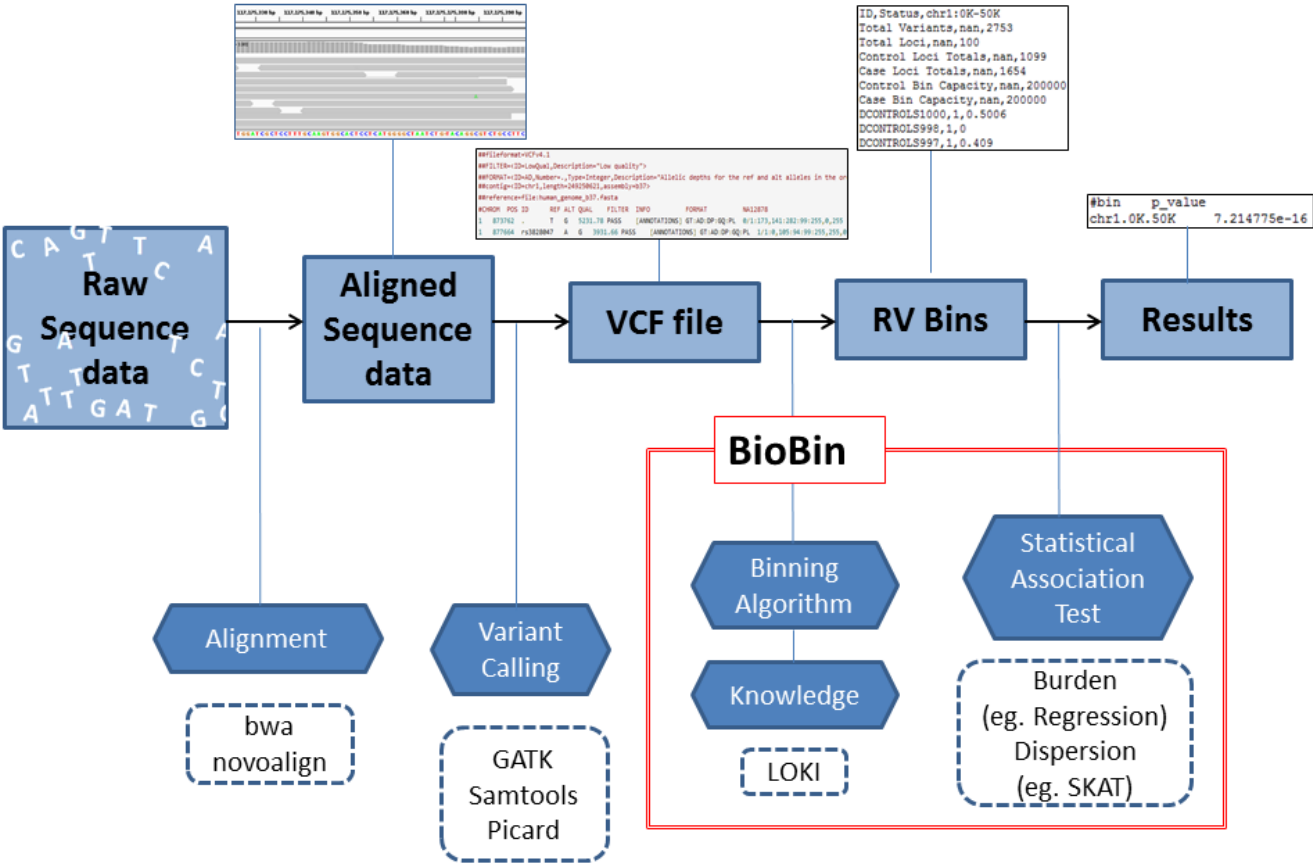

Figure1: This image illustrates a typical rare variant analysis pipeline and depicts the role of BioBin in this workflow. BioBin accepts VCF files and uses a binning algorithm that is driven by biological knowledge within the LOKI biorepository. Bins can simultaneously be statistically tested using designated burden (regression, Wilcoxon rank sum test) and dispersion (SKAT) approaches.
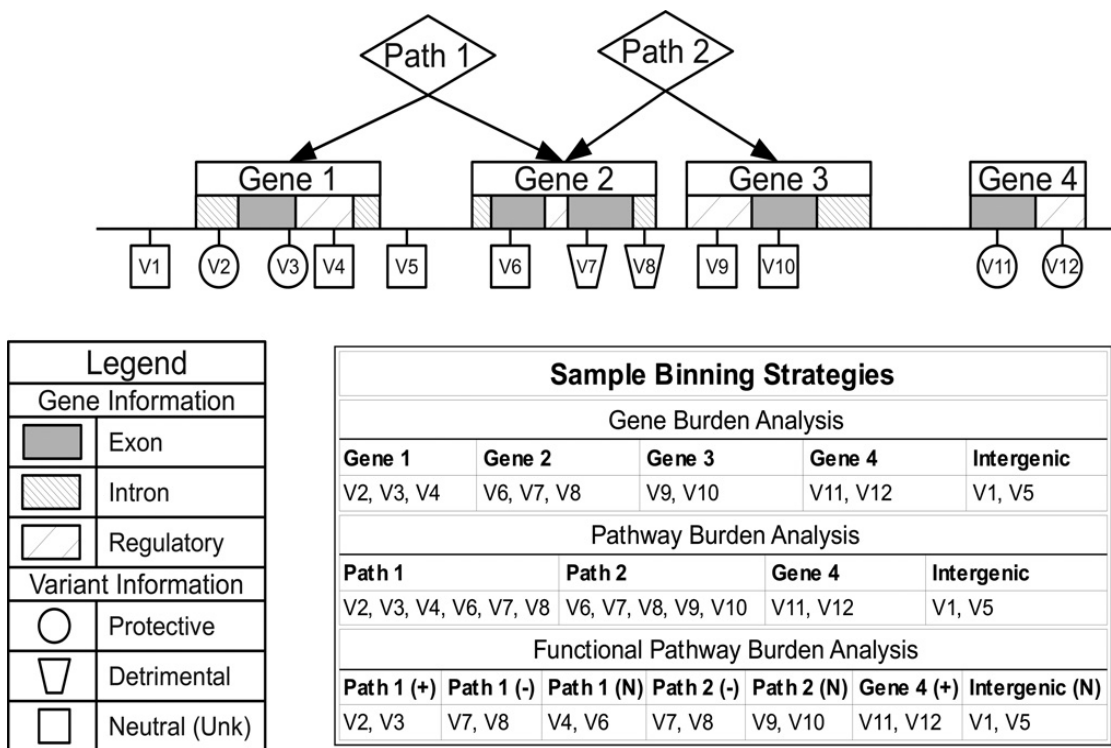
Figure 2: Variant binning within BioBin is performed using an allele frequency threshold approach defined by the non-major allele frequency (NMAF). NMAF is defined as 1 minus the frequency of the most common allele. For biallelic markers, NMAF and minor allele frequency (MAF) are interchangeable. BioBin will bin all variants with an allele frequency below the user-specified NMAF in the case and/or the control population (note: for quantitative traits, all samples are treated as controls). Such an approach allows BioBin to aggregate potential risk and protective variants. The scheme above illustrates three common binning strategies possible in BioBin: gene, pathway, and functional pathway binning.

BioBin Quick Reference

General Options:
    -h [--help]   Display help message
    -v [--version]   Display version
    -S [--sample-config]   Print a sample configuration to the screen
    --no-gsl-bt   Abort on GSL Errors; do not try to print backtrace

Command Line Options:
    --print-populations   Print populations available in LOKI
    --print-sources   Print the sources available in LOKI

BioBin Options:
    -D [--settings-db] (=knowledge.bio)   Location of the LOKI database
    -V [--vcf-file]   File containing VCF information
    -t [--threads] (=1)   Number of threads to use in PheWAS
    --add-group   List of filenames containing a group definition
    -G [--genomic-build] (=37)   Genomic build of the input files
    --ignore-build-difference (=N)   Ignore genome build differences in vcf
    --include-samples   File of samples to include (1 per line)
    --exclude-samples   File of samples to exclude (1 per line)

Bin Generation Options:
    -F [--maf-cutoff] (=0.05)   Maximum MAF of loci in a bin
    --maf-threshold (=0)   Minimum MAF of loci in a bin
    --keep-monomorphic (=N)   Keep monomorphic loci in memory and print them to the locus report
    -m [--bin-minimum-size] (=5)   Minimum number of variants in a bin
    -e [--bin-expand-size] (=50)   Number of bins above which bins are expanded into child bins
    -x [--bin-expand-roles] (=N)   Flag indicating to expand bins into roles
    --filter-bin-role (=N)   When expanding bins into roles, drop any bins with unknown role
    --keep-unknown-role (=N)   When filtering the bin roles, keep only bins with unknown role
    --bin-pathways (=N)   Include pathway bins in the analysis
    --bin-regions (=Y)   Include region (gene) bins in the analysis
    --bin-interregion (=Y)   Bin loci not in any region into generic base-pair bounded bins
    -i [--interregion-bin-length] (=50)   Number of kilobases in the generic base-pair bounded intergenic bins
    --interregion-bin-step   Step size of the intergenic bins; if not given, assumed to be bin length
    --set-star-referent (=Y)   Set star alleles as referent if Y, else treat star alleles as missing
    --bin-constant-loci   Bin loci that have no variation in any samples with non-missing-phenotype

Phenotype Options:
    -p [--phenotpye-filename]   Filename of the phenotype file
    --min-control-frac (=0.125)   Minimum fraction of control samples needed. If less, sets all controls
    --rare-case-control (=Y)   Considers MAF in either case or control population for rarity
    --phenotype-control-value (=0)   Numeric value of controls in the phenotype file
--drop-missing-phenotype-samples (=Y)   Flag indicates whether to drop samples not in case/control for any phenotype
--force-all-control (=N)   Flag indicates whether all the phenotypes should be considered as control

Report Generation Options:
    --report-prefix   A prefix to give to all of the generated reports
    --report-loci (=Y)   Flag indicating desire to write locus report
    --report-bins (=Y)   Flag indicating desire to write Bin report

--transpose-bins (=N)	Transpose the Bin report (bins on rows)
--no-summary (=N)	Suppress the summary information in a Bin report
-d [--output-delimiter] (=,)	The delimiter to use when outputting text files

Testing Options:

--covariates	File of covariates for use in statistical test
--weight-loci (=N)	Adds Madsen & Browning weights to each locus
--weight-model (=minimum)	Method of determining the appropriate calculated weight
--disease-model (=additive)	Disease model to use for determining individual contribution in a bin (additive, dominant, or recessive)
--test	A comma-separated list of statistical tests to perform on each bin (SKAT-linear, SKAT-logistic, linear, logistic, wilcoxon)

LOKI Options:

--include-group-names	Comma-separated list of group names to include in analysis
--include-group-file	Filename of list of group names to include in analysis
--include-region-names	Comma-separated list of region names to include in analysis
--include-region-file	Filename of list of region names to include in analysis
--max-group-size (=0)	Maximum number of regions in any group
-P [--population]	Population on which to base gene boundary extensions
-B [--region-boundary-extension] (=0)	Fixed amount of gene boundary extension
--region-file	File containing custom regions
--include-sources	Sources to include in analysis
--exclude-sources (=dbsnp,oreganno,ucsc_ecr)	Sources to exclude from analysis
--role-file	File containing custom roles
--weight-file	File containing custom locus weights
--ambiguity (=resolvable)	Strategy to use when faced with ambiguity in LOKI. Ambiguity mode (strict, resolvable, remissive)

Installation

BioBin is packaged with the GNU autotools, so installation occurs in four steps: unpacking, configuration, compilation, and installation. Each of these steps is described below. The user must first ensure that the prerequisites for running BioBin and for generating the supporting biological database, LOKI, are met.

Prerequisites

The following are prerequisites for building and running BioBin. The packages that are needed only for building the LOKI database are indicated with an asterisk.

- A modern C++ compiler

- Boost Libraries for C++, version 1.50 or later

- SQLite, version 3.9.0 or later

- suds for Python, version 0.4 or later*

- apsw for Python*

- liftOver binaries (for building populations)

Unpacking

BioBin is distributed as a zipped tarball, and the command for unpacking the distribution is:

$ tar -xvzf biobin-2.3.0.tar.gz

This will unpack the source code into a directory called biobin-2.3.0. For all of the following commands, we assume that you are in this directory.

$ cd ./biobin-2.3.0

Configuration

In order to compile BioBin, the user must first configure the software. This script will attempt to detect all of the prerequisites on the user's system. This is the time for the user to indicate system-specific options, such as the location of the installed program. The command is:

$ ./configure

The configure script can also take a number of helpful options, some of which are detailed below:

- **--help**

This option will list all of the available options that can be passed to the configure script.

- **--prefix=[path]**

This option tells BioBin to install itself into the given path, which is useful if you do not have administrative access to the computer. By default, the program will be in [path]/bin, and the LOKI database will be in [path]/share. Note: when using this option, the path given must be an absolute path and cannot use any shell expansions, such as the "~" notation.

- **--disable-loki**

This option disables the compilation of the LOKI biological database. Since the compilation of the database will take a few hours with a high speed internet connection, this option is helpful if you are installing a new version of BioBin, but want to leave the database unchanged.

- **--enable-debug**

For advanced users, this option will turn off all optimization and turn on debugging symbols, which can be helpful in diagnosing a problem with the BioBin software.

Compilation

LOKI Setup

Due to the size of the LOKI database, it is not distributed along with the BioBin code. Instead, we provide the means for a user to build the LOKI database by downloading the data directly from the sources. The LOKI database must be compiled before installation described further below.

Note: If the settings were incorrect and you received an error during the compilation of the LOKI database, you must follow the steps below.

Compiling BioBin

After the LOKI setup steps, BioBin should be compiled. Simply run the following command and the program will build and generate the LOKI database.

$ make

Installation

At the time of installation, the program and database are moved into their final locations, as defined during the configure step. Typically, the user will need administrative rights to complete the installation step. To install both BioBin and LOKI, type:

make install

If you only want to install BioBin, you can type:

make install-exec

And if you only want to install the LOKI database, you can type:

make install-data

During the installation of LOKI, the database is copied to the destination directory, and it is named "yyyy.mm.dd.knowledge.bio", where "yyyy.mm.dd" is the date of creation of the knowledge database. However, for ease of use, the installer will also create a shortcut called simply "knowledge.bio" that will point to this installed file. This is designed so that a user may have multiple concurrent LOKI databases that each correspond to a different snapshot in time.

Rebuilding the Database

BioBin uses a static LOKI database and alone it will not capture updates made to the sources as the sources are updated. Thus, from time to time, it is necessary for the user to rebuild the database with the most recent information. Assuming that the user configured BioBin to build the LOKI database in the first step, the command to discard the current LOKI database from the build directory is:

$ make clean

From this point on, the user can re-run the compilation and installation steps to regenerate and reinstall the LOKI database. Note that the old LOKI database will **NOT** be deleted from the installation directory, but the shortcut will be updated to the most recent database.

Population Creation

After the LOKI database has been created and installed, the user may create population-specific genetic boundaries. This process is described in the

section at the end of the document. Note that if the database is rebuilt, the populations must also be rebuilt.

Input

Configuration File

This file is a plain text file containing all options for BioBin that can be used instead of specifying all the options on the command line. Each line in this file must either be blank, a comment beginning with a pound (#) character, or a configuration value pair. A configuration value pair is given in the form:

> config-name = value

The configuration value is dependent on the specific configuration option that is being used. All of the options available to the user can be found in the "Options" section of this manual.

LOKI Database

The database given to BioBin must be a LOKI SQLite database conforming to version 2.0 table specification. This database is built when compiling BioBin, but it can be built or updated by using the included updater scripts that are distributed with BioBin. The updater script can be found in the BioBin distribution at **updater/loki-build.py**. Please see the help documentation of this script for details. Note that the user will need to provide the absolute path to the database file when using this script, as it will not search the data path in the same way that BioBin does.

VCF File

BioBin must also be given a single VCF file (either compressed or uncompressed) conforming to the 4.0 version of the specification (http://www.1000genomes.org/node/101). The VCF file must contain all individuals included in the study. If a VCF file contains samples that are not included in the phenotype file (see below), these samples will be dropped from the analysis. Additionally, BioBin provides options to drop specified samples from the analysis irrespective of their phenotype status (see the "--include-samples" and "--exclude-samples" options).

If needed, a user can combine data from disparate studies or groups into a single VCF file by using the vcf-merge tool from vcftools. Please note, however, that merging VCF files should be done with care, as the default action is to insert missing data when the referent alleles were dropped from individual files. You may want to assume homozygous referent data when merging, but this may not be reflective of the true sequencing. We recommend using a joint-called VCF whenever possible.

Deletions spanning a SNP are represented as a star ("*") in VCF files generated using GATK v3.4-46 and above. To avoid double counting of samples that have a deletion spanning a SNP present in other samples, BioBin sets star alleles to referent (default) or treats samples with star alleles as missing when binning SNPs that overlap with a deletion (see the "--set-star-referent" option).

Phenotype Files

In order to perform association analysis with BioBin, a user should provide a file containing phenotypes of the individuals in the VCF file. The phenotype file must be a plain text file that contains two or more columns separated by whitespace. Each column after the first is considered a new phenotype. The first column contains the exact sample ID string that is found in the VCF file. The second column contains a floating-point value that represents the phenotype of the individual. Samples present in the VCF file, but not in the phenotype file are by default dropped from the analysis ("--drop-missing-phenotype-samples (=Y)). Similarly, samples included in the phenotype file but with "nan" or "." in place of a phenotype value are considered as missing and are excluded from the analysis. The first line should begin with a pound (#) character and contain the names of the phenotypes; if the header line is not included, the phenotypes will be named sequentially as "Pheno_1", "Pheno_2", and so forth. Further lines beginning with the pound (#) character are treated as comments and are ignored, as are blank lines.

NOTE: by default, BioBin considers a value of "0" to be a control sample. To change this default behavior, the user can use the "--phenotype-control-value" argument.

When running analysis on quantitative traits, columns 2+ should include numeric phenotype/s and the option "--force-all-control" should be set to Y, so that all samples are considered as controls.

Below is an example of a sample phenotype file containing two phenotypes, P1 and P2, and a case control status for each sample and each phenotype:

```
#ID    P1  P2
# This is a sample phenotype file
Person1 0   1
Person2 1   1
Person3 0   0
Person4 0   1
Person5 1   0
```

Region File

BioBin gives the option of using a file to define biological knowledge that is not in the LOKI database. This file defines custom regions based on the location in the genome. Custom regions are defined outside of information in LOKI, therefore, given regions will not be contained in any LOKI groups or pathways.

The first column in the file is the chromosome, the second is the ID of the region, and the third and fourth columns give the boundaries of the region. Note that if an ID variable is repeated in a file, BioBin will bin all variants together. This prevents multiple bins with the same names. Below is an example of a sample region file:

```
# This is a sample region file
1       REG1            12345  22380
1       REG2            347220          356700
4       GENEA           57390 63457
```

Role File

Similar to the region file, this file allows the user to define the roles of certain variants or regions in the genome. Roles are intended as a secondary binning mechanism following gene binning. BioBin currently restricts the number of unique roles to no more than 60. This file can be used to define the

roles of variants according to regions in the genome, such as exons or introns. It can also be used to assign functional roles to variants, e.g. damaging.

The format of this file is very similar to the region file above. The first column is the chromosome; the second column lists the ID of the role, and the third and fourth columns list the start and end positions of the roles. Optionally, a user can give a fifth column, which is the associated gene ID that the role is associated with. Because the roles are a secondary binning mechanism, if no gene ID is given, the role is assumed to apply to any variant within the given boundaries. If a gene ID is given, the role only applies for the given gene ID. Below is an example of a sample role file:

```
# This is a sample role file
1       role1   14578  15890
1       role2   12456  14800  REG1
2       role1   45783  48000
```

Weight File

Similar to the role file mentioned above, a weight file is a means for assigning weights to loci contained within a certain region. BioBin will calculate the inverse allele frequency weighting of Madsen and Browning [15] by using the "**--weight-loci**" option. The weight file could be used to provide an alternate weighting calculation or to increase or decrease the weights of certain regions in the genome. Additionally, the user may provide a gene ID which restricts the weighting of the locus to within the given gene and region.

The format is identical to the role file, with the roles replaced with real-valued weights that should be applied to loci in the regions given. There are no restrictions on the values of the weights, and as shown in the example below, negative weights are allowed. However, the user should have some knowledge and reason for using negative weights, as some of the standard statistical tests may not be applicable in that case. Note that if a gene ID is given, the weight will only be applied when binning to the level of regions; the weight will NOT be applied when binning at the pathway level. An example of the input is given below:

```
# This is a sample weight file
1       1.34            14578  15890
1       0.21            12456  14800  REG1
2       -1.5            45783  48000
```

Output

This section lists all of the possible outputs generated by BioBin. Generation of any of these reports can be turned on or off at the user's request through the given options on the command line or in the configuration file. All files listed in this section are delimited text files, where the delimiter is chosen in the configuration file (default is a comma). In all of these files, the first line is a header row describing the columns. Subsequent lines are described for each file below.

Bins Report

This report provides details on bins generated by BioBin. If multiple phenotypes are provided to BioBin, there will be one bins file per phenotype as the specific distribution of the phenotype values may impact the binning strategy.

An example bins file is shown below (spaces added for clarity):

```
ID,              T2D, TTLL10, WRAP73
Total Variants,    nan,  32,   63
Total Loci,        nan,  5,    4
Control Variants,  nan,  18,   12
Case Variants,     nan,  14,   51
Control Bin Capacity, nan,  134,   172
Case Bin Capacity,   nan,  130,   130
Gene(s),           nan, ,      ,
linear p-value,    nan,  0.452, 0.0032
NA06984,           0,    0,    1
NA06985,           0,    0,    0
                              (truncated)
NA20504,           1,    0,    1
NA20506,           1,    0,    0
                              (truncated)
```

The first row is the header line. It contains sample IDs (e.g. NA06984), the phenotype being assessed (e.g. T2D), and individual bin identifiers (e.g. gene TTLL10). Rows 2-8 provide summary information on the variants and loci in each bin. If statistical tests are applied, the p-value for each test is given, one per line, starting with row 9. Each subsequent row lists individuals in the study along with information on their phenotypic status and contribution to each bin (more details below).

Rows:

Row 2 (column 3+) gives the total number of variants contained within each bin. This is also referred to as the "size" of the bin. It is defined to be the sum of every individual's contribution to the given bin, or the sum of each individual's genotype (i.e. total number of alternate alleles across all samples and all loci in a bin, *e.g.* genotype 0/0 counts as 0, 0/1 counts as 1, and 1/1 counts as 2). As a check, this line should ALWAYS equal the sum of values in rows with sample IDs (i.e. in rows starting at row 8 + the number of statistical tests, since a p-value for each test is reported on a separate line in the bins report starting with row 9), if no weighting is used.

Row 3 (column 3+) gives the total number of loci that are contained within each bin. In comparison to values in row 2, a locus corresponds to the physical location of a variation. The number of variants at that locus would be the sum of each individual's genotype. The number of loci on the other hand is independent of the individuals in the dataset (for a fixed allele frequency).

Rows 4 and 5 (column 3+) give the total number of variants in the control and case populations for each bin, respectively. The sum of rows 4 and 5 should equal the total number of variants in row 2.

Rows 6 and 7 (column 3+) provide the total bin capacity for the control and case populations each bin and phenotype. Bin capacity is defined as the total number of alleles (both reference and alternate alleles) in called genotypes across all loci that contribute to a given bin (maximum bin capacity for cases or controls is equal to the number of cases (or controls) x 2 x total number of loci in a bin, but the observed capacity if often lower than that due to either missing genotype calls at some loci or missing phenotype information).

Row 8 (column 3+) provides the gene names of the genes that have binned variants for the given pathway bin. This row will only have gene names in a pathway binning analysis. It will be blank value for gene and inter-region binning analyses.

When testing for statistical significance, each test is printed in a separate line (starting at row 9), and the p-value for that test is printed for each bin. The order of the statistical tests is arbitrary and may not correspond to the order in which it was provided on the command line.

<u>Columns:</u>

After the header, summary and any test rows (rows 1-8 + number of statistical tests), the columns correspond to individual phenotypic status information, as well as the contribution of variants of each individual to the bin, as described in more detail below.

Column 1 (rows 8 + number of statistical tests) contains the individual ID. This is the unique identifier of each individual contained in the VCF dataset.

Column 2 (rows 8 + number of statistical tests) provides the phenotypic status of each individual (e.g. "0" and "1" to denote controls and cases, respectively, or numeric values for quantitative traits).

Columns 3+ (rows 8 + number of statistical tests) correspond to the contribution of variants of each individual to a given bin. For example, "0" in row 10, column 3 means that individual NA06984 did not contribute any alternate alleles to the *TTLL10* bin, whereas "1" in row 10, column 4 means that this individual contributed 1 alternate allele to the *WRAP73* bin (assuming no loci weighting was used). If "--weight-loci" = Y, values in these fields will represent weighted contributions of each individual to each bin.

Locus Report

The locus report provides information about every locus contained within the VCF file and how it was binned using BioBin. This file is helpful when looking at group-independent characteristics of the data.

An example locus file is shown below (spaces added for clarity):

```
Chromosome, Location, ID,        Gene(s), P1_Bin(s), P2_Bin(s)
1,      1115503, rs111751804, TTLL10, TTLL10,   TTLL10
1,      1115548, rs114390380, TTLL10, TTLL10,
1,      1118275, rs61733845, TTLL10, TTLL10,   TTLL10
1,      1120377, rs116321663, TTLL10, ,        TTLL10
1,      1120431, rs1320571,  TTLL10, TTLL10,   TTLL10
1,      3548136, rs2760321,  WRAP73, WRAP73,   WRAP73
1,      3548832, rs2760320,  WRAP73, ,
1,      3548855, rs114376964, WRAP73, WRAP73,
1,      3551737, rs116230480, WRAP73, WRAP73,   WRAP73
```

Columns 1, 2, and 3 identify each locus or variant. The first column is the chromosome, the second is the base pair position on the chromosome, and the third column is a unique rsID from the VCF file or an ID generated by BioBin, if rsID is not available.

Column 4 lists all genes that the particular locus is contained within or associated with, separated by a pipe (|).

Columns 5+ provide the names of all of the bins that contain the locus, again separated by a pipe (|). In the case of multiple phenotypes, each phenotype will be an additional column.

<u>Options</u>

This section describes all of the options available to a user either in the configuration file or on the command line. Unless it is explicitly stated that an option can be given multiple times, options given on the command line override any options given in the configuration file. Command line options must be preceded by two dashes (--). Some options have a shortened version available, the flag will be given

in parentheses; the shortened version only requires a single dash before the option. Finally, when an argument is required, the type will be given in brackets ([]) with the default value given in parentheses within the brackets, if it exists.

NOTE: any option that is marked with an asterisk (*) has not been well tested.

Command Line Only Options

help (-h)

Prints a help message describing all options available to the user on the command line and exits.

sample-config (-S)

Prints a sample configuration file to the screen and exits. This is helpful when getting started to save a properly formatted configuration file. The output can be redirected to a file of the user's choosing.

version (-v)

Prints a short copyright message along with the version of BioBin and exits.

no-gsl-bt

Causes BioBin to crash if it runs out of memory while running statistical tests (running out of memory may occur, e.g., when running SKAT tests on very large bins). If this option is not included and BioBin runs out of memory, the default behavior is to set the p-value for the problematic bin to >=1 (depending on location of the memory error, which will be seen in the error output) and to proceed to running statistical tests on the next bin without crashing.

print-populations

Prints a tab-delimited list of populations available in LOKI along with a description of the population.

print-sources

Prints a list of sources available in the LOKI database. Results from this option are intended to be used with the "--include-sources" and "--exclude-sources" options.

Configuration and Command Line Options

BioBin Options

settings-db (-D) [string (knowledge.bio)]

This argument is the location of the LOKI database file. It may be an absolute or relative path to the SQLite database. If the user provides a relative path and BioBin cannot find the file, BioBin will also search the data directory provided during installation.

vcf-file (-V) [string]

This is the VCF file to be used by BioBin in determining the variants and calculating allele frequencies. This can be either a relative or absolute path. Note that as of BioBin version 2.1, compression of the VCF file is detected automatically based on the file extension. Additionally, because this file is read only once, and in order, the user may stream an input VCF using input redirection. This can alleviate

the need to store large amounts of duplicate data on the system, lessening the storage requirements.

threads (-t) [integer (1)]

Gives the number of parallel threads that BioBin should use when generating bins for multiple phenotypes. Note that the reading of the VCF file is performed first in a single thread, but all phenotypes are binned in parallel with this argument. If not performing a PheWAS with BioBin, this argument will provide no benefit.

add-group [string]*

A filename containing a custom group. This could be a disease-dependent list of genes or pathways that already exist in LOKI. For details on the formatting of the group file, see Creating Custom Groups. This option may be repeated multiple times.

genomic-build (-G) [string]

This argument gives the genomic build on which the VCF file is based (default is read from the VCF). BioBin will liftover any data in the VCF file and any input files into a consistent genomic build to allow for accurate comparisons with the LOKI database. Note that the current LOKI database (as of July, 2015) is build 38.

ignore-build-difference

By default, BioBin checks to ensure that the genomic build of the input VCF file provided by a user using the "-G" option matches the build in the VCF file (if it can be determined based on the information in the header). Alternatively, a user can include this option to force BioBin to ignore a potential difference in genomic builds (not recommended unless a user knows what he/she is doing).

include-samples

This option lets a user provide a file with a list of sample IDs (one per line; sample IDs should match the IDs from the input VCF file) to be included in the analysis. Sample IDs that are in the VCF file, but are not included in this list will be excluded from the analysis.

exclude-samples

This option lets a user provide a file with a list of sample IDs (one per line; sample IDs should match the IDs from the input VCF file) to be excluded from the analysis. Sample IDs that are in the VCF file and are not listed here will be included in the analysis.

Bin Generation Options

maf-cutoff (-F) [float (0.05)]

This is the cutoff for the minor allele frequency. Loci with minor allele frequencies less than or equal to this value in cases and/or controls (note: for quantitative phenotypes all samples should be set to controls by setting "--force-all-controls" to Y) will be considered rare variants and will be grouped into bins by BioBin.

maf-threshold [float (0)]

This is the minimum allele frequency to consider for bin inclusion. Only loci with allele frequencies greater than the threshold in cases and/or controls (note: for quantitative phenotypes all samples should be set to controls by setting "--force-all-controls" to Y) will be binned. This can be useful for

excluding singletons from analysis, which can skew results, especially when using a weighting strategy.

keep-monomorphic [Y/N (N)]

If this option is turned on, BioBin will keep monomorphic loci in memory, even though they could never contribute to any bin. Monomorphic markers may appear in an all-sites VCF, or by selecting an alternate encoding strategy using the "--disease-model" argument.

bin-minimum-size (-m) [integer (5)]

This is the minimum size of a bin (total variants) in order to keep the bin for analysis. Any bins that contain fewer variants than this threshold are deleted before writing output. **Note**: please take caution when interpreting results with low bin sizes. Statistical tests may not be appropriate and trustworthy with lower variant values

bin-expand-size (-e) [integer (50)]

This is the bin size at which BioBin attempts to expand the bin by subdividing if possible. If there are no further subdivisions possible, these bins will be kept (provided that their size is above the bin-minimum-size).

bin-expand-roles (-x) [Y/N (N)]

A boolean flag indicating the desire to subdivide bins according to custom role information provided in the *role-file* input. Roles can be exons, introns, regulatory regions or annotations such as functional prediction.

filter-bin-role [(Y/N) (N)]

This option allows the user to filter the bins by the roles provided in the role file(s). By enabling this option, the "--keep-unknown-role" option determines whether the unknown roles are filtered or kept.

keep-unknown-role [(Y/N) (N)]

When filtering by role by enabling the "--filter-bin-role" option, this option dictates how the bins will be filtered. When enabled, bins with unknown role are kept, and when disabled all bins with unknown role are discarded.

bin-pathways [Y/N (N)]

A boolean flag indicating the desire to create bins according to the pathways (or groups) contained in LOKI. If "--bin-genes" is turned off, this flag will still use the genetic information contained within LOKI, but the bins will not expand into genes.

bin-regions [Y/N (Y)]

A boolean flag indicating the desire to create or subdivide bins by gene or region information contained within LOKI. This option should also be on if the user chooses to input a custom region file. If both this option and "--bin-pathways" are turned off, the only bins retrieved will be interregion.

bin-interregion [Y/N (Y)]

A boolean flag indicating a desire to group any loci not in a feature according to their position on the chromosome. This option will create bins according to the size given by the "--interregion-bin-length" option.

interregion-bin-length (-i) [integer (50)]

This number is the length of the intergenic bins, in kilobases. The default is to make bins of 50 kilobases, so the first bin on any chromosome would go from position 1 to position 50,000 (NOT 50).

interregion-bin-step [integer]

This number gives the size, in kilobases, of the step size of the intergenic bins. By setting this value, the user can obtain sliding windows of intergenic bins. As an example, if the interregion step size is 10, with the default interregion bin length of 10, the first intergenic bin on a chromosome will go from position 1 to position 50,000, and the second intergenic bin will be from position 10,001 to 60,000. If not provided, the step size will be equal to the interregion bin length, providing mutually exclusive but comprehensive intergenic bins.

set-star-referent [Y/N (Y)]

This option sets star ("*") alleles (see description of star alleles in the Input section under "VCF file") to referent alleles (default) to avoid double counting of samples that have a deletion spanning a SNP that is present in other samples. Alternatively one may switch it off (*i.e.* set "--set-star-referent" to N) and this will result in setting samples with star alleles to missing. Irrespective of whether this option is turned on or off, samples with star alleles will still be contributing to a bin because of a deletion that they have, but will not be counted when considering the SNP that this deletion is spanning (this is to prevent double counting of such samples).

bin-constant-loci [Y/N (N)]

By default, BioBin bins only those loci that exhibit variation in either cases or controls. With this option turned on, BioBin will also bin loci that have no variation in any samples with non-missing phenotype.

Phenotype Options

phenotype-filename (-p) [string]

This is the filename of the phenotype file described above. The filename can be given as either a relative or absolute path.

min-control-frac [float (0.125)]

This gives the minimum fraction of the population in the control group for BioBin to consider the data valid. Because the allele frequencies are calculated from the control population, if the control population is too small (or nonexistent), the allele frequencies calculated will not have sufficient refinement to discern rare and common variants. Additionally, if the case population fraction is below this threshold, BioBin will report a warning.

rare-case-control [(Y/N) Y]

Flag indicating whether BioBin should determine rarity of variants by both case and control populations. Enabling this option reduces a bias that was seen where large bins were more likely to be significant in a case/control analysis.

phenotype-control-value [float (0)]

This option gives the value of the control population in the phenotype file. All other values will be considered cases.

drop-missing-phenotype-samples [Y/N (Y)]

This option causes samples that are not included in the phenotype file to be dropped from the analysis.

force-all-control [Y/N (N)]

This option forces all samples to be considered as controls. It should be set to Y when running an analysis on quantitative traits. When this is set to N (default), samples with a phenotype value equal to "--phenotype-control-value" (0 by default) will be treated as controls and all the remaining samples will be treated as cases (with the exception of samples with phenotype value equal to "nan" or ".", which will be set to missing and excluded from the analysis).

Report Generation Options

report-prefix [string]

This option gives the user the opportunity to set the prefix of any reports that are to be output. The filename of reports will be "prefix-<report>.csv". This option may be used to place reports in a given directory, but care must be taken to ensure that the directory exists prior to running BioBin.

report-loci [Y/N (Y)]

A boolean flag indicating the desire to output the locus report.

report-bins [Y/N (Y)]

A boolean flag indicating the desire to output the bins report.

transpose-bins [Y/N (N)]

A boolean flag indicating a desire to transpose the bins report (all columns become rows and vice versa). Using this option will facilitate the use of tools such as grep to filter bins by name.

no-summary [Y/N (N)]

A boolean flag indicating the desire to suppress the summary information (rows 2-8) in the Bins report.

output-delimiter (-d) [string (,)]

The output text file field delimiter.

Testing Options

covariates [string]

Gives the name of a file containing covariates to be loaded for the various statistical test(s) to be performed. If given, this file has the same format as the phenotype file described above, and all covariates in this file will be used. Note that not all tests may use covariates, and some may require linear independence of the covariates, which will be checked and enforced by dropping any colinear covariates. Additionally, any categorical covariates (such as site or project) MUST be dummy-encoded; BioBin assumes that covariates entered are numeric.

weight-loci [(Y/N) N]

Add weights to the loci in the output according to Madsen and Browning's weighted sum statistic [15]. The weight for a given locus is defined to be:

$$\frac{2n + 2}{\sqrt{n(m + 1)(2n - m + 1)}}$$

Where n is the total (non-missing) population, and m is the number of variants at a given locus. Note that in BioBin, the population for a given locus may not be equal to the total study population, as the weighting algorithm excludes missing persons at a locus.

weight-model [maximum/minimum/control/overall (minimum)]

Flag indicating the weighting method to use in calculating the locus weights. As previously noted [16] , basing the allele rarity decisions on the control population increases type I error dramatically. Similarly, we have noticed that calculating the weights based solely on the control population also increases this error, for the exact same reason. This flag allows the user a way to use weights while controlling the error. The following methods are available:

- **maximum**: The weight is the maximum of those calculated for case and control populations.

- **minimum**: The weight is the minimum of those calculated for case and control populations. In preliminary tests, this method yielded both the highest power and the lowest false positive rate.

- **control**: The weight is calculated based on the control population only. This method is not recommended, and is provided as a means to verify historical results.

- **overall**: The weight is calculated based on the overall population, without regard to case or control status.

Note that in the case of either no cases or completely missing case or control population for a given locus, all of the methods are equivalent.

disease-model [additive/dominant/recessive (additive)]

This option gives the disease model used by BioBin in calculating an individual's contribution to a bin. The models are defined by the following table:

Genotype	additive	dominant	recessive
Major / Major	0	0	0
Major / Minor	1	1	0
Minor / Minor	2	1	1

test [linear/logistic/SKAT-linear/SKAT-logistic/wilcoxon]

Provides a comma-separated list of statistical tests to run on every bin generated according to the user's binning preferences. For any test requiring case/control status (such as logistic, SKAT-logistic and wilcoxon), the phenotype being tested is the observed case/control status, as determined by the phenotype control value option provided to BioBin. In the case of quantitative trait analysis (as in linear and SKAT-linear), the dependent variable is the value provided in the phenotype file. Note that only the p-value of the test is provided, and in the case of logistic regression, no tests for convergence are performed. This option may be provided more than once.

NOTE: Within BioBin, SKAT p-values are calculated using Davies method [17] with $\sim 10^{-8}$ accuracy level (dependent on the machine on which BioBin is running) and the reported SKAT p-values range from -1 to >1. Here is a description of all possible SKAT p-values/p-value ranges as reported by

BioBin:

SKAT p-value = –1: regression error, model did not converge.

SKAT p-value = (-1,0]: the result of the p-value calculation is smaller than the margin of error. The true p-value may be smaller than the absolute p-value reported.

SKAT p-value = (0,1]: reported p-value is equal to the observed p-value +/- accuracy.

SKAT p-value > 1: error during SKAT p-value calculation.

LOKI Options

The following options are applicable to the knowledge available to the user. These options either define new knowledge or limit the knowledge that is loaded from LOKI.

include-group-names [string]*

This option allows the user to specify the names of specific groups (or pathways) to include from LOKI when constructing bins. The names must match exactly those contained within LOKI. This option can be given more than once.

include-group-file [string]*

This option gives the name of a file containing a list of names of groups to be included from LOKI when constructing bins. This is a means of easily combining many calls to the "--include-group-names" option. This option may be provided more than once.

include-region-names [string]

This option is a comma-separated list of specific regions to include from LOKI when constructing bins. Only loci within the given regions from this list will be included when constructing bins, even in the case of pathway binning. This option may be provided more than once.

include-region-file [string]

This option gives the name of a file containing a list of names of regions to include from LOKI when constructing bins. This is a means to easily combine many calls to the "--include-region-names" option. This option may be provided more than once.

max-group-size [integer (0)]

This option sets the maximum number of regions to be included in any group from LOKI.

population (-P) [string]*

This option gives the population on which to base the boundaries of the genes in LOKI. The population must be one that exists in LOKI, and if the population is nonexistent, the gene boundaries will be determined by the canonical boundaries.

gene-boundary-extension (-B) [integer (0)]*

This option gives a base pair extension (both upstream and downstream) to a feature, which extends the boundaries and likely increases the number of loci binned. This option is only valid without a population.

region-file [string]

This option gives the name of a custom region file that defines the regions of interest to the user. In order to bin on custom regions, you must include a file of the correct format here and also turn on "--bin-regions." You may use this option more than once to include multiple region files.

include-sources [string]

A comma-separated list of source names to include in the binning decisions. If no arguments are given, BioBin will include all information from the LOKI database when binning. Note that the source names must match exactly those contained within the LOKI database. This option can be repeated multiple times. Note that it is an error to both include and exclude the same source.

exclude-sources [string (dbsnp, oreganno, ucsc_ecr)]

A comma-separated list of source names to exclude. Three sources are excluded by default, and this option may be given more than once. Note that it is an error to both include and exclude the same source.

role-file [string]

This option should be used to input custom role files, such as introns and exons boundaries. Note, to use this option one must include the appropriate custom role file AND turn on role binning "--bin-expand-roles". The user can also include filtering options with the role file, "--filter-bin-role" and "--keep-unknown-role".

weight-file [string]

This option is used to input custom weights on specific loci, as described in the weight file input. When using this file, any weights will be multiplicative with any calculated weights based on allele frequency, and if a locus is not present in this file or a given locus-region combination is not present, the weight will default to 1.

ambiguity [strict/resolvable/permissive (resolvable)]

This option defines the level of permissible ambiguity from the sources in LOKI. In many pathway sources such as GO or Biogrid, the gene IDs are given multiple times because many genes are known by multiple aliases. LOKI provides for means of resolving this ambiguity, which is covered in more detail in the Biofilter manual, available from http://ritchielab.com. In the strict ambiguity mode, only links that can be identified without any ambiguity are allowed. In permissive mode, any possible link that could be made is used. In the default resolvable mode, the links allowed are those that can be uniquely resolved using any of the heuristics defined in the Biofilter manual.

Creating Populations for LD

Instead of using arbitrary base-pair extensions for features, one might want to extend boundaries using known LD patterns. BioBin has the capability to use population-specific boundaries of genes through the use of the configuration option. By default, only a single default population is defined by the loader (LOKI build script), and it is incumbent on the user to define any auxiliary populations. A population is defined by both a HapMap population and a cutoff defined by either an R^2 value or a D' value. Note that it is possible to have multiple boundary populations based on a single HapMap population. An example could be a CEU population with a D' cutoff of either 0.9 or 0.8.

To create these populations within BioBin, we provide a script buildPopulations.py, which downloads the data from HapMap and loads the data into the LOKI database for use by BioBin. Note that if the database is rebuilt, the populations must also be rebuilt, and this is not an automated step.

Prerequisites

In order to use the buildPopulations.py script, the user must have available both the liftOver binaries, as well as the helper programs, "pop_loader," and "ldspline," which are distributed with Biofilter (software available from http://ritchielab.com). The liftOver binaries can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/. By default, when BioBin is made, the prerequisites are also made and installed.

Usage

In order to run buildPopulations.py, the user must be able to write to the LOKI database. Typically, this will mean that the user must have the same rights as described in the Installation section, which are usually administrative rights. By default, buildPopulations.py is installed alongside BioBin, so the command to use is:

> # buildPopulations.py [OPTIONS]

Where the options are described below.

Options

The options to the buildPopulations.py script allow the user to dictate the populations and cutoffs as well as the location of BioBin and any other necessary helper programs. All options are given on the command line, and the format is identical to the command line options given in the BioBin description.

populations (-p) [string]

This option gives a comma-separated list of HapMap populations from which to generate LD-based boundaries. The populations must be the 3-letter HapMap abbreviations, or their one-letter shortcuts (*e.g.*, "C" is synonymous with "CEU"). This option may be provided more than once on the command line, and all populations given will be used. Note that this is a mandatory option.

dprime (-d) [float]

This option gives a comma-separated list of floating point values to use as cutoffs for the D' value in generating the LD-based boundaries. This option may be provided more than once, and all given values will be used. Note that either this option or "--rsquared" is required.

rsquared (-r) [float]

This option gives a comma-separated list of floating point values to use as cutoffs for the R^2 value in generating LD-based boundaries. This option may be provided more than once, and all given values will be used. Note that either this option or "--dprime" is required.

liftover (-l) [string]

This option gives the location of the "liftOver" binary needed to convert the build 36 HapMap files to the build 38 Biofilter data. If this option is not given, the script assumes that the liftOver binary is in the path and can simply be called "liftOver". The liftOver binaries can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/.

poploader (-o) [string]

This option gives the location of the pop_loader binary file that was previously built. By default, the script assumes that the pop_loader program is located in the path, and can be called by executing the command "pop_loader".

ldspline (-s) [string]

This option gives the location of the LD-spline tool provided by the Ritchie Lab. By default, LD-spline is provided with BioBin and is installed alongside the "biobin" executable. The default value of this option is simply "ldspline".

db (-b) [string]

This option provides the location of the LOKI database that contains the gene information. This value is passed to the BioBin executable and will follow the same rules for finding the database as used in BioBin.

Example

The following example shows the creation of LD-specific boundaries for both the CEU population at R^2 values of 0.8 and 0.9 and D' values of 0.85 and 0.95. Note the different methods of specifying multiple cutoff values:

```
# buildPopulations.py -p CEU -r 0.8 -r 0.9 -d 0.85,0.95
Downloading hg18ToHg19.over.chain.gz
Downloading ld_chrX_CEU.txt.gz
```

<div align="center">(truncated)</div>

```
Downloading ld_chr10_CEU.txt.gz
Extracting ld_chrX_CEU.txt.gz
```

<div align="center">(truncated)</div>

```
Extracting ld_chr10_CEU.txt.gz
```

<div align="center">(to be continued...)</div>

Creating Custom Groups

If a user wishes to create a group of genes or regions that is not already contained within the LOKI knowledge database, (s)he can do so using a custom group file. In addition to defining which genes are in which group, a user can also define relationships between the groups themselves. These relationships are currently not used in BioBin, but they may be at some point in the future.

Group File Format

Custom groups are defined using a plain text file with a specific format. Each file defines a set of related groups and forms a single source. The file must follow the format below:

```
[Source Name] [Source Description]
GROUP [Name] [Description]
```

([alias] [alias]*)|(CHILDREN [group] [group]+)|(GROUP ...)

[Source Name] [Source Description]

The first line of the file must contain the name of the collection of groups, along with the type of collection.

The **"Source Name"** must be a string with no spaces, and it must be unique from any other source already defined in the LOKI database. The number of sources is very limited, and if the name of this group is not the same as any database of biological knowledge, there should be no namespace conflicts.

The **"Source Description"** is an optional string designed to help the user keep track of the actual meaning behind the group. The description may contain any character except a newline ("\n").

GROUP [Name] [Description]

This line defines the beginning of a new group. This must be given on the second line, and it may occur on subsequent lines within the file.

The **"Name"** must be a string with no spaces, and it must be unique from any other group name defined within the current custom group file. This name can be used to identify the current group as a child of another group.

The **"Description"** is an optional string used to describe the group.

*[alias] [alias]**

This line is a whitespace-separated list of gene names that can be found in the LOKI database. Currently, if an alias maps to more than one region that has been found in the database, the group will be considered to contain all of the genes that have the given alias. Note that this is equivalent to the "permissive" ambiguity setting, but it only applies to the loading of groups from files.

CHILDREN [group] [group]+

This line defines associations between groups within the custom group collection. The first group given is considered the parent, and all subsequent groups are the children. Note that there must be at least two groups given in this line.

Examples

Because this is possibly the most complex input file available to BioBin, we have provided a couple examples below. The simple group definition should be sufficient for anyone attempting to use a list of genes that are associated with a given disease. The more complex example illustrates an interrelated pathology.

Simple Group Definition

This file is a single group containing a simple collection of genes that are associated with Alzheimer's

        ALZHEIMERS Alzheimer's Collection
        GROUP alz-assoc Genes associated with Alzheimer's
        AGT
        APH1A
        APOA1BP
        APOA2

CAMK1G
CFH
CHRNB2

Complex Group Collection

The following example shows a slightly more complex collection of interrelated groups, still using the Alzheimer's data above, but split into two groups, one with genes starting with the letter "A" and one with genes starting with the letter "C". Additionally, there is a parent super-group that contains both subgroups. Also, this file demonstrates the inclusion of more than one gene on a single line, as can be seen in the "alz-assoc-A" group.

ALZ-COMPLEX Alzheimer's Complicated
GROUP alz-assoc-A Genes assoc. w/ Alzheimer's (beg. w/ A)
AGT APH1A
APOA1BP APOA2

GROUP alz-assoc-C Genes assoc. w/ Alzheimer's (beg. w/ C)
CAMK1G
CFH
CHRNB2

GROUP alz-master Master group for Alzheimer's
CHILDREN alz-master alz-assoc-A alz-assoc-C

Example Usage

On the software download page, we have provided data to perform example analyses. Chromosome 22 from the CEU-TSI targeted exome project is available for download. We have also provided the relevant phenotype file needed for a BioBin analysis. Lastly, we have provided three sample configuration files and pertinent custom files to highlight three example analyses. All example analyses should be run from the directory of the files.

1. **Standard gene burden test.**

    In this analysis, BioBin creates region and inter-region bins based on Entrez gene information. Run the analysis with command:

        biobin gene_analysis.config

2. **Pathway burden test with an applied exon/intron filter.**

    This analysis requires the provided custom role file containing intron/exon boundaries for chromosome 22, based on UCSC gene browser information. Run this analysis with the command:

        biobin external_roles.config

3. **Alternative gene source burden analysis using a provided custom region file.**

    Finally, this analysis replicates the first gene burden analysis using gene boundary information from the UCSC gene browser. Run this analysis with command:

biobin external_regions.config

References:

1. Moore CCB, Basile AO, Wallace JR, Frase AT, Ritchie MD. A biologically informed method for detecting rare variant associations. BioData Min. 2016;9:27.

2. Moore CB, Wallace JR, Frase AT, Pendergrass SA, Ritchie MD. BioBin: a bioinformatics tool for automating the binning of rare variants using publicly available biological knowledge. BMC Med. Genomics. 2013;6:S6.

3. Basile AO, Wallace JR, Peissig P, McCarty CA, Brilliant M, Ritchie MD. KNOWLEDGE DRIVEN BINNING AND PHEWAS ANALYSIS IN MARSHFIELD PERSONALIZED MEDICINE RESEARCH PROJECT USING BIOBIN. Pac. Symp. Biocomput. Pac. Symp. Biocomput. 2016;21:249–60.

4. Moore CB, Wallace JR, Wolfe DJ, Frase AT, Pendergrass SA, Weiss KM, et al. Low Frequency Variants, Collapsed Based on Biological Knowledge, Uncover Complexity of Population Stratification in 1000 Genomes Project Data. PLoS Genet. 2013;9:e1003959.

5. Kanehisa M, Goto S, Sato Y, Furumichi M, Tanabe M. KEGG for integration and interpretation of large-scale molecular data sets. Nucleic Acids Res. 2012;40:D109-114.

6. Milacic M, Haw R, Rothfels K, Wu G, Croft D, Hermjakob H, et al. Annotating cancer variants and anti-cancer therapeutics in reactome. Cancers. 2012;4:1180–211.

7. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, et al. Gene Ontology: tool for the unification of biology. Nat. Genet. 2000;25:25–9.

8. Finn RD, Bateman A, Clements J, Coggill P, Eberhardt RY, Eddy SR, et al. Pfam: the protein families database. Nucleic Acids Res. 2014;42:D222–30.

9. Kandasamy K, Mohan S, Raju R, Keerthikumar S, Kumar GSS, Venugopal AK, et al. NetPath: a public resource of curated signal transduction pathways. Genome Biol. 2010;11:R3.

10. Zanzoni A, Montecchi-Palazzi L, Quondam M, Ausiello G, Helmer-Citterich M, Cesareni G. MINT: a Molecular INTeraction database. FEBS Lett. 2002;513:135–40.

11. Stark C, Breitkreutz B-J, Reguly T, Boucher L, Breitkreutz A, Tyers M. BioGRID: a general repository for interaction datasets. Nucleic Acids Res. 2006;34:D535–9.

12. McDonagh EM, Whirl-Carrillo M, Garten Y, Altman RB, Klein TE. From pharmacogenomic knowledge acquisition to clinical applications: the PharmGKB as a clinical pharmacogenomic biomarker resource. Biomark. Med. 2011;5:795–806.

13. Montgomery SB, Griffith OL, Sleumer MC, Bergman CM, Bilenky M, Pleasance ED, et al. ORegAnno: an open access database and curation system for literature-derived promoters, transcription factor binding sites and regulatory variation. Bioinformatics. 2006;22:637–40.

14. Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, et al. The Human Genome Browser at UCSC. Genome Res. 2002;12:996–1006.

15. Madsen BE, Browning SR. A Groupwise Association Test for Rare Mutations Using a Weighted Sum Statistic. PLoS Genet. 2009;5:e1000384.

16. Lemire M. Defining rare variants by their frequencies in controls may increase type I error. Nat. Genet. 2011;43:391–2.

17. Wu MC, Lee S, Cai T, Li Y, Boehnke M, Lin X. Rare-Variant Association Testing for Sequencing Data with the Sequence Kernel Association Test. Am. J. Hum. Genet. 2011;89:82–93.