BioBin User Guide

Current version: BioBin 2.1

Last modified: August 2014

Ritchie Lab, Center for Systems Genomics

Pennsylvania State University, University Park, PA 16802

URL: https://ritchielab.psu.edu/ritchielab/software/

Email: software@ritchielab.psu.edu

Table of Contents

Overview

BioBin meets a critical need for an improved binning algorithm through the advantage of prior biological knowledge and potential cumulative effects of biologically aggregated RVs. BioBin requires the Library of Knowledge Integration (LOKI), which contains diverse prior knowledge from multiple collections of biological data. BioBin can be used to apply multiple levels of burden collapsing/testing, including: regulatory regions, evolutionary conserved regions, genes, and/or pathways without a need for an external feature file. BioBin aids rare variant analysis by binning variants according to prior biological knowledge.

BioBin

BioBin is a standalone 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 including 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 rare variants is the primary driver of memory usage.

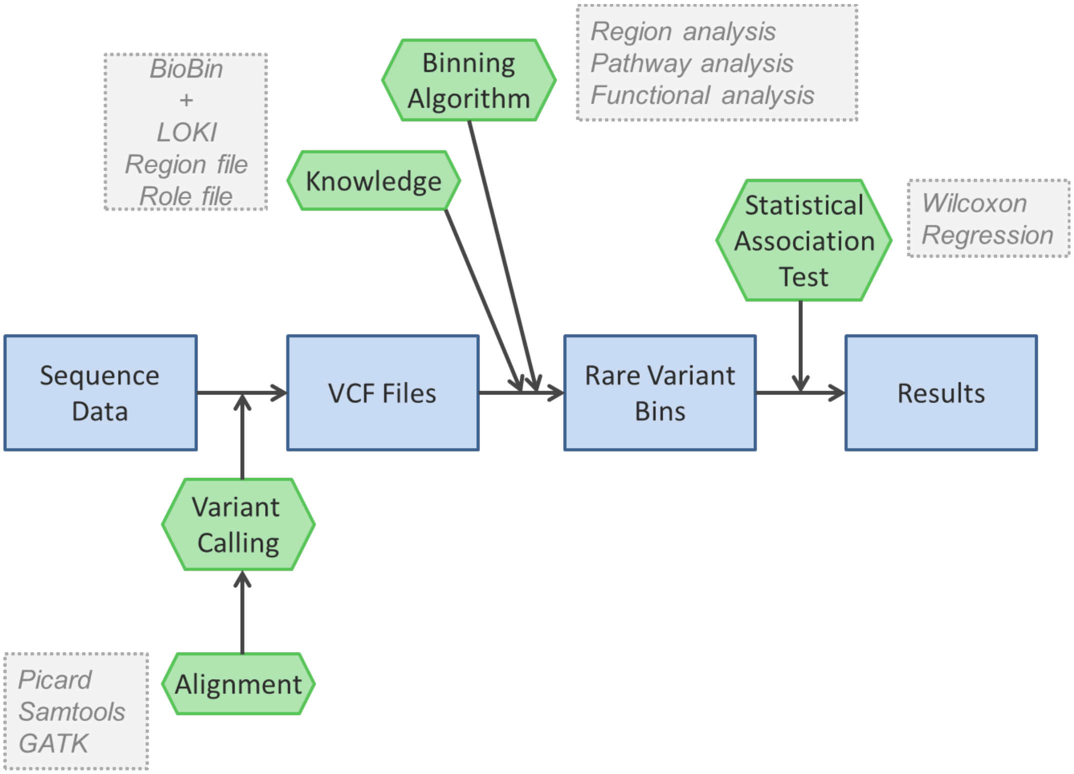
Library of Knowledge Integration (LOKI) Database

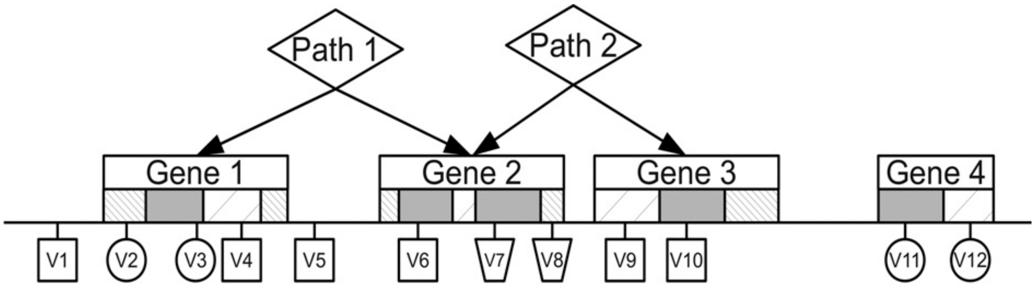
Harnessing prior biological knowledge is a powerful way to inform collapsing feature boundaries. BioBin relies on the Library of Knowledge Integration (LOKI) for database integration and boundary definitions.   LOKI contains resources such as: the National Center for Biotechnology (NCBI) dbSNP and gene Entrez database information [*Sayers et al. 2010*], Kyoto Encyclopedia of Genes and Genomes (KEGG) [*Kanehisa et al. 2011*], Reactome [*Croft et al. 2010*], Gene Ontology (GO) [*Dimmer et al 2011*], Protein families database (Pfam) [*Punta et al. 2011*], NetPath - signal transduction pathways [*Kandasamy et al. 2010*], Molecular INTeraction database (MINT) [*Licata et al. 2012*], Biological General Repository for Interaction Datasets (BioGrid) [*Stark et al. 2011*], Pharmacogenomics Knowledge Base (PharmGKB) [*McDonagh et al. 2011*], Open Regulatory Annotation Database (ORegAnno) [*Griffith et al. 2007*], and information from UCSC Genome Browser about evolutionary conserved regions [*Fujita et al. 2011*].

LOKI is used as a means 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 databases (such as those listed above) that contain *groups* of interconnected information, thus organizing the data in some way.

LOKI is implemented in SQLite, a relational database management system, which does not require a dedicated database server. The user must download and run installer scripts (python) and allow for 10-12 GB of data from the various sources. The updater script will automatically process and combine this information into a single database file (~ 6.7 GB range). A system running LOKI should have at least 50 GB of disk storage available. LOKI runs locally and must be managed locally; however, research groups can also customize LOKI for their own purpose.

BioBin Workflow





Legend	
Gene Information	
■	Exon
▨	Intron
▧	Regulatory
Variant Information	
○	Protective
▽	Detrimental
□	Neutral (Unk)

Sample Binning Strategies

Gene Burden Analysis

Gene 1	Gene 2	Gene 3	Gene 4	Intergenic
V2, V3, V4	V6, V7, V8	V9, V10	V11, V12	V1, V5

Pathway Burden Analysis

Path 1	Path 2	Gene 4	Intergenic
V2, V3, V4, V6, V7, V8	V6, V7, V8, V9, V10	V11, V12	V1, V5

Functional Pathway Burden Analysis

Path 1 (+)	Path 1 (-)	Path 1 (N)	Path 2 (-)	Path 2 (N)	Gene 4 (+)	Intergenic (N)
V2, V3	V7, V8	V4, V6	V7, V8	V9, V10	V11, V12	V1, V5

Biobin Quick Reference

General Options:

-h [--help]	Display help message
-v [--version]	Display version
-S [--sample-config]	Print a sample configuration to the screen

Command Line Options:

--print-populations	Print populations available in LOKI
--print-sources	Print the sources available in LOKI

BioBin Options:

-D [--settings-db] arg (=knowledge.bio)	Location of the database
-V [--vcf-file] arg	File containing VCF information
-C [--compressed-vcf] arg (=N)	Flag indicating VCF file is compressed
-F [--maf-cutoff] arg (=0.05)	Maximum minor allele frequency to consider eligible for bin inclusion
-k [--keep-common-loci] arg (=Y)	Flag indicating to keep data pertaining to common variants
--add-group arg	List of filenames containing a group collection definition
-d [--output-delimiter] arg (=,)	Delimiter to use when outputting text files
-p [--phenotype-filename] arg	Filename containing phenotype information
-m [--bin-minimum-size] arg (=5)	Minimum size of any bin
-e [--bin-expand-size] arg (=50)	Size above which bins are expanded into child bins
-x [--bin-expand-roles] arg (=N)	Flag indicating to expand bins into exons/intron regions
--filter-bin-role arg (=N)	Flag indicating desire to filter by unknown role
--keep-unknown-role arg (=N)	If true, keep only unknown role bins
--bin-pathways arg (=Y)	Flag indicating not to include pathways in the analysis
--bin-regions arg (=Y)	Flag indicating not to include genes in the analysis
--bin-interregion arg (=Y)	Flag indicating not to include intergenic bins in analysis
-i [--interregion-bin-length] arg (=50)	Number of kilobases intergenic bins can hold
--report-prefix arg	Prefix to give to all of the reports
--report-loci arg (=Y)	Flag indicating desire to write locus report
--report-bins arg (=Y)	Flag indicating desire to write bin report
--report-genotypes arg (=N)	Flag indicating desire to write genotype report
--report-locus-freq arg (=N)	Write Case v. Control Minor Allele Freq. report
--report-bin-freq arg (=N)	Write Bin Case v. Control Frequency report
--transpose-bins arg (=N)	Transpose the Bin report (bins on rows)
-G [--genomic-build] arg (=37)	Genomic build of input data
--phenotype-control-value arg	Phenotype control value
--min-control-frac arg (=0.125)	Minimum fraction of population needed for control cases
--rare-case-control arg (=Y)	Rarity of variants by both case and control populations
--overall-major-allele arg (=Y)	Determine the major allele by the overall population instead of control
--weight-loci arg (=N)	Add weights to the locus
--weight-model arg (=minimum)	Method for weighting loci (maximum, minimum, control, overall)
--disease-model arg (=additive)	Disease model (additive, dominant, or recessive)

LOKI Options:

--include-group-names arg	List of group names to include
--include-group-file arg	File containing a group definition
-P [--population] arg	Population to base the gene boundaries on
-B [--region-boundary-extension] arg	Amount to expand the genes by (when using NO-LD)
--region-file arg	File containing custom regions
--include-sources arg	List of source names to include
--exclude-sources arg (=dbsnp,oreganno,ucsc_ecr)	List of source names to exclude
--role-file arg	File containing custom roles
--weight-file arg	File containing custom locus or region weights
--ambiguity arg (=resolvable)	Ambiguity mode (strict, resolvable, permissive)

Installation

BioBin is packaged with the GNU autotools, so installation occurs in four steps: unpacking, configuration, compilation and installation. Each of those steps will be described below, but first the user must ensure that the prerequisites for running BioBin are met, as well as the prerequisites for generating the supporting biological database, which we have called the Library of Knowledge Integration (LOKI).

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 1.42 or later
- SQLite, version 3.5.4 or later
- suds for Python, version 0.4 or later*
- apsw for Python*
- liftOver binaries (for building populations)

Unpacking

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

```
$ tar -xvzf biobin-2.1.0.tar.gz
```

This will unpack the source code into a directory called `biobin-2.1.0`. For all of the following commands, we assume that you are in this directory.

```
$ cd ./biobin-2.1.0
```

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, and this is the time for the user to specify 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 you want to leave the database unchanged.

- **--enable-debug**

For the 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. 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 that if the settings were incorrect and you received an error during the compilation of the LOKI database, you must follow the steps given in Rebuilding the Database below.

Compiling BioBin

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

```
$ 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, 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. 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 and all of the options available to the user can be found in the Options Optionssection 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 on how to run and the arguments re. 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 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. By using the vcf-merge tool from vcftools, a user can combine data from disparate studies or groups into a single VCF file.

NOTE: Using the standard **vcf-merge** tool distributed with the vcftools package has the potential to insert missing data when the referent alleles were dropped from the individual files. We have provided a modified **vcf-merge** in the **scripts** directory of the distribution, which treats all alleles that are missing in one of the files as referent, instead of missing.

Phenotype Files

In order to perform case/control analysis with BioBin, a user should provide a file that contains the phenotypes of the individuals contained in the above VCF file. The phenotype file must be a plain text file that contains two columns separated by whitespace. The first column contains the exact ID string that is found in the VCF file and the second column contains a floating-point value that represents the phenotype of the individual. 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. Below is an example of a sample phenotype file:

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

Region File

BioBin gives the option of using a file to define biological knowledge that is not in the LOKI database. This file defines 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, and 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, i.e. 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 (insert citation) 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 weight 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 we see below that negative weights are allowed, but 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 or below; 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 that BioBin can generate. 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 line describing the columns, and subsequent lines are described for each file below.

Bins Report

This report provides detail on bins generated by BioBin. The first line is the header line. Lines 2-7 give summary information on the bins, and each line after 7th row corresponds to an individual in the study.

After the header and summary rows (rows 1-7), the columns correspond to the contribution of variants of each individual to the bin. Column 1 contains the individual ID; this is the unique identifier of each individual contained in the VCF dataset. Column 2 gives the phenotypic status of each individual, summary rows have a phenotype value of -1. Each column after the phenotype column corresponds to individual bins.

The summary rows summarize the variants and loci in each bin. Row 2 gives the total number of variants contained within a 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. As a check, this line should ALWAYS equal the sum of the rows 7+ if no weighting is used.

Row 3 gives the total number of loci that are contained within the 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 give this same number, but they exclude those loci for which data is entirely missing from either the case or control populations.

Rows 6 and 7 give the total bin capacity for either the cases or controls. The capacity is defined to be the absolute maximum number of variants that could be contributed to a given bin. An example bin file is shown below (spaces added for clarity):

```
ID,                     Status, TTLL10, WRAP73
Total Variants,         -1,     32,     63
Total Loci,             -1,     5,      5
Control Loci Totals,    -1,     5,      5
Case Loci Totals,       -1,     5,      5
Control Bin Capacity,   -1,     134,    172
Case Bin Capacity,      -1,     130,    130
NA06984,                0,      0,      1
NA06985,                0,      0,      0
                            (truncated)
NA20504,                1,      0,      1
NA20506,                1,      0,      0
                            (truncated)
```

Locus Report

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

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 ID which could have been given in the VCF file or generated by BioBin.

Column 4 gives the alleles and their frequencies, as calculated from the control population. A pipe (|) character separates individual alleles, and the allele and frequency are separated by a colon (:). The alleles are ordered from most frequent to least frequent, and the minor allele frequency is defined to be the frequency of the second most common allele.

Column 5 gives the non-major allele frequency in the case population. The non-major allele frequency is defined to be the frequency of all alleles other than the most common allele in the control population.

Column 6 gives the status of the locus. If the minor allele frequency is below the threshold for binning, this column will be 1, otherwise it will be 0.

Column 7 gives all genes that the particular locus is contained within or associated with, separated by a pipe (|). Column 8 gives the names of all of the bins that contain the locus, again separated by a pipe (|). Below is a sample of the output (spaces added and decimals truncated for clarity):

```
Chromosome, Location, ID,          Alleles,        Case AF, Rare, Gene(s), Bin(s)
1,          1115503,  rs111751804, T:0.97|C:0.03, 0.02,    1,    TTLL10,  TTLL10
1,          1115548,  rs114390380, G:0.99|A:0.01, 0.02,    1,    TTLL10,  TTLL10
1,          1118275,  rs61733845,  C:0.96|T:0.04, 0.05,    1,    TTLL10,  TTLL10
1,          1120377,  rs116321663, T:0.99|A:0.01, 0.01,    1,    TTLL10,  TTLL10
1,          1120431,  rs1320571,   G:0.96|A:0.04, 0.04,    1,    TTLL10,  TTLL10
1,          3548136,  rs2760321,   C:0.85|T:0.15, 0.19,    0,    WRAP73,  WRAP73
1,          3548832,  rs2760320,   G:0.94|C:0.06, 0.04,    1,    WRAP73,  WRAP73
1,          3548855,  rs114376964, T:0.99|C:0.01, 0.01,    1,    WRAP73,  WRAP73
1,          3551737,  rs116230480, C:0.99|T:0.01, 0,       1,    WRAP73,  WRAP73
```

Genotype Report

The genotype report is a listing of the genotypes for each person and for each locus. In essence, this is a repetition of information contained within the phenotype file and the VCF file inputs.

Column 1 gives the ID of each person, and the second column gives the phenotypic status of each person, as given in the phenotype report. Columns 3+ give the encoded genotype of each person for each locus. The encoded genotype is defined as (# of alleles)*(allele index on strand 1) + (allele index on strand 2), with an encoded genotype of -1 indicating missing data. Here, the allele index is not necessarily the same as the index given in the VCF file, but rather an allele index of **n** represents the allele that has precisely **n** more common alleles in the control population. Below is a sample of the output (spaces added and lines truncated for clarity):

```
ID,      Status, rs111751804, rs114390380, ...
NA06984, 0,      0/0,         0/0,         ...
NA06985, 0,      0/0,         0/0,         ...
                          (truncated)
NA20515, 1,      0/0,         1/0,         ...
NA20516, 1,      0/0,         0/0,         ...
                          (truncated)
```

Allele Frequency Report

The allele frequency report is a listing of the non-major allele frequencies by locus. This information is provided for convenience, but it repeats much of the information contained within the locus report. This file can be used to see if any loci might be statistically significant because of great differences between the control and case frequencies.

Column 1 gives the identification string of the locus (column 3 in locus report). Columns 2 and 3 give the non-major allele frequency for both control and case populations, respectively. If the number of cases or controls with data at this locus is 0, then the frequency will be reported as -1. Column 4 is 1 if the locus is considered a rare variant, and 0 otherwise (column 6 of the locus report), and column 5 is a pipe-separated list of bins containing the locus (column 8 of the locus report). Below is a sample output (spaces added for clarity):

```
Locus,       Control NMAF, Case NMAF,  Rare, Bins
rs111751804, 0.0286885,    0.0230769,  1,    TTLL10
rs114390380, 0.0127119,    0.0153846,  1,    TTLL10
rs61733845,  0.036,        0.046875,   1,    TTLL10
rs116321663, 0.00645161,   0.00757576, 1,    TTLL10
rs1320571,   0.0387324,    0.0384615,  1,    TTLL10
rs2760321,   0.154412,     0.190476,   1,    WRAP73
rs2760320,   0.0580645,    0.0378788,  1,    WRAP73
rs114376964, 0.00320512,   0.00757576, 1,    WRAP73
rs116230480, 0.0032258,    0,          1,    WRAP73
rs115982402, 0.00320512,   0.00757576, 1,    WRAP73
```

Bin Frequency Report

The bin frequency report is similar to the allele frequency report in that it attempts to identify the most significant bins by looking at the difference in frequency of the case and control populations. A bin frequency is defined to be the total contributions of a population to a bin divided by the bin capacity for the population, as defined in the Bins Reportbins report section.

The first column of the file is the name of the bin. The second and third columns give the frequencies for the control and case populations, respectively. Note that if the bin capacity is 0, the frequency is technically missing and will be reported in this file as -1. Below is a sample output (spaces added for clarity):

```
Bin,    Control Freq., Case Freq.
TTLL10, 0.11194,       0.130769
WRAP73, 0.186047,      0.238462
```

Options

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.

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

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.

compressed-vcf (-C) [Y/N (N)]

This boolean flag tells BioBin if the given VCF file has been compressed using gzip, or one of its variants such as bgzip.

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

This is the cutoff for the minor allele frequency. Variants with minor allele frequencies below this value will be considered rare variants and will be grouped into bins by BioBin.

keep-common-loci (-k) [Y/N (N)]

This option provides an option for saving memory by ignoring any loci above the given minor allele threshold. By default, this option is turned off.

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.

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

The output text file field delimiter.

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. This option may be repeated multiple times, and if an individual is found in more than one file, the value found in the last file is considered definitive.

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.

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

This is the 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, only bins with unknown role are kept, and when disabled all bins with unknown role are discarded.

bin-pathways [Y/N (Y)]

A boolean flag indicating the desire to create bins according to the pathways (or groups) contained in LOKI. If **bin-genes** are 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 0 to position 50,000 (NOT 50).

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.

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.

report-genotypes [Y/N (N)]

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

report-locus-freq [Y/N (N)]

A boolean flag indicating the desire to output the allele frequency report.

report-bin-freq [Y/N (N)]

A boolean flag indicating the desire to output the bin frequency report.

genomic-build (-G) [string (37)]

This argument gives the genomic build on which the VCF file is based. BioBin will liftover any data in the VCF file into a consistent genomic build to allow for accurate comparisons with the LOKI database. NOTE: all other data such as custom regions must be in the same build as the LOKI database, which is currently build 37.

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.

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 determining 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.

overall-major-allele [(Y/N) Y]

Flag indicating desire to determine the major allele by the overall population instead of the control. By enabling this option, the case and control population are truly interchangeable in the statistical analysis.

weight-loci [(Y/N) N]

Add weights to the loci in the output according to Madsen and Browning's weighted sum statistic [*Madsen and Browning 2009*]. The weight for a given locus is defined to be:

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

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 method to use in determining the population to use in calculating the locus weights. As noted in (insert citation of increased type-I error guy), simply 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

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 filename 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.

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 filename 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-region." 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."

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.psu.edu. 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 Error: Reference source not found configuration option. By default, only a single default population is defined by the loader, 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`, that 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. 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 is 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 to generate LD-based boundaries from. 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 37 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

                        (truncated)

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

                        (truncated)

Extracting ld_chr10_CEU.txt.gz
```

(to be continued...)

Creating Custom Groups

If a user wished 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, which defines all of the groups that a user will need. 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 following format, which will be explained 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 line 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
   ```