The Biological Repository (BioR) and BioRTools User Guide v2.0

By Daniel Quest, Patrick Duffy, Raymond Moore, The BioR Team, and the BioR users.

Table of Contents:

1. Installation: 3

Installing inside Mayo with access to the Research Computing Facility (RCF) 3

Overview 3

Steps 3

Installing the Biological Repository Catalogs 3

Installing on a Stand-Alone Server or Workstation 4

Installing BioR Tools from Source 4

Java Heap Size 4

2. Quick Tour 4

Introduction 4

Data Modeling 5

BioR Catalog Shortcut 5

Finding out what is in a Catalog 5

Showing the Commands in BioRToolkit 6

Pretty Print 7

Get all Variants in a Gene 8

3. BioR Catalogs 11

The BioR Catalog Format 11

Catalog Creation Details 11

Catalogs Availible In BioR 12

Examples Matching Genomic Features 13

Positional Matches Using Tabix 13

Annotating Variants with Genes that Overlap 13

Compressing output to enforce 1-1 semantics 15

Expanded Genes (Xrefs) 16

Indexing Catalogs 17

Looking Up Information about a Gene 18

Example of Walking Cross References 20

Generating an OMIM Disorder Report for a Set of rsIDs 21

Putting it all Together – Making a Genomic Feature Annotation Program 29

Examples Matching Alleles (bior\_same\_variant) 30

Putting it All Together Building an AF Pipeline 44

Extracting Data with JSONPaths (bior\_drill) 44

Command Line Tools 44

Mixing In Scripts and Languages 45

To find all overlapping genes that are not the same gene: 45

Common Problems 45

Handling VCF Files with VERY large headers 45

Large Memory Requirements 45

BioR exits with some error I don’t understand 45

Creating Catalogs 46

Indexing your Samples 46

Creating Custom Catalogs 47

The Publication Process 47

Parsing and Converting the Data 48

Indexing the Data for Coordinate Based Search 48

Hints on Creating Indexes on Custom Catalogs 49

Use BioR to map SNP on rsID and find overlapping genes. 49

Case Study: Creating a Report that Maps rsIDs to Genes. 54

The Biological Repository (BioR) and BioRTools User Guide v 2.0

BioR is an annotation engine. Inside Mayo, it’s primary use is to annotate human variation, but it is not limited to that – it is a general purpose genomic data integration tool that enables coordinate based searches and joins based on strings. BioR is like programming using lego blocks, each block may not be exactly what you want, but you can put the blocks together to create programs extremely rapidly. The component ‘blocks’ include all existing UNIX commands, stand alone tools (e.g. bedtools), and the bior\_toolkit. This user guide will help get you up to speed in how to use BioR in one document.

# 1. Installation:

## Installing inside Mayo with access to the Research Computing Facility (RCF)

If you have access to the RCF, you are in luck! We have already installed BioRTools for you, all you need to do is put it in your path. Here are the steps to do that:

### Overview

The CLI is available through the **mayobiotools** utility.  No software needs to be downloaded as it's already pre-installed.  Make sure you select version 2.0 or greater.

### Steps

1. login to an RCF submission node server (example: "ssh crick6.mayo.edu")
2. execute "mayobiotools"
3. scan the list of packages for "java"
4. type corresponding package number and press enter
5. select a version that is 1.6 or higher
6. scan the list of packages for "bior\_scripts"
7. type corresponding package number and press enter
8. select "2.0.0" version
9. quit mayobiotools and save changes
10. logout and log back into the RCF submission node server
11. BioR Command Line Client commands are now available
12. Try this from the command line: "bior\_vcf\_to\_tjson -h"  if BioR is working you should see a help message.
13. To expore the bior scripts available on the command line type bior followed by a tab.

## Installing the Biological Repository Catalogs

On the RCF, no installation is needed. Catalogs can be found at $BIOR\_CATALOG ($bior in this documentation) If you are doing a stand alone server, download the catalog flat files and place them locally on your server in a similar directory structure. BioR Tools does not make any assumptions about the location of catalogs relative to each other, but it does assume that tabix indexes are in the same directory as the compressed catalog and that ID indices are in a folder called index in the same directory as the catalog.

## Installing on a Stand-Alone Server or Workstation

BioR is written in Java, so in principle it will work on any machine, but it depends on some command line tools (e.g. SNPEFF, VEP) that are not so friendly. The development team has BioR working on both Macintosh and Linux. To install, first make sure first that Java 1.6+ is installed and on your path (Java 1.7 is preferred). Then download the BioR executable and place it in your path.

## Installing BioR Tools from Source

Source installation requires that you have both Java 1.7 and Maven installed and on your path. It also requires that you have access to the Mayo NEXUS servers or you place several libraries in your ~/.m2 directory.

If you have troubles installing BioR or compiling it, please contact the BioR Team (dlrstitbiorall@mayo.edu) so we can update the documentation and make the process easier.

## Java Heap Size

On some machines, the default JVM size is 2GB. This is very large for BioR. By default the BioR toolkit is capped at 128M. To change this setting, change the BioR properties file for each command (e.g. bior\_pipeline/cli.properties - jvm.opts=-Xmx128m).

# 2. Quick Tour

## Introduction

BioR uses a [Pipe-And-Filter](http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html) architecture. Data to be annotated by BioR is streamed through a pipeline, a sequence of one or more pipes. Pipes is based on Flow Based Programming by J.P. Morrison. [DataFlow-Article](http://www.drdobbs.com/database/dataflow-programming-handling-huge-data/231400148?pgno=2), [Flow-Based-Programing](http://www.amazon.com/Flow-Based-Programming-2nd-Application-Development/dp/1451542321/).



Figure 1: BioRTools works by adding annotation to the right on the original file.

BioR leverages UNIX pipes to flow data from program to program. As BioR programs work on the data, they place annotation to the right (the red, blue and green colums in Figure 1).

## Data Modeling

BioR has adopted a lightweight approach to modeling annotation data. Only **core** annotation fields are modeled to enable supported search capabilities (e.g. coordinate search, accession ID search). Anything not classified as **core** is modeled into a "schema-free" data structure.



## BioR Catalog Shortcut

BioR commands commonly use long paths to files. One of the first things you will want to do when using BioR is to make an alias to the location of the BioR catalogs. For example if the BioR catalogs are located in $bior

Then, on bash, do the following command at the command line:

$ export bior=/data/path/

You may want to put this command in your .bashrc or .bash\_profile so that the $bior environment variable shows up next time you log in.

## Finding out what is in a Catalog

Each data source is 'published' into a BioR catalog file for use by the BioR scripts.  A Catalog is a collection of files (both data and indexes) that is understood by the BioR Pipes infrastructure. BioR's reference data consists of the raw files downloaded/updated and made available to BioR users. These files ARE NOT catalogs. Catalogs are transformed into the BioR standard catalog structure so that pipes can work on the content. BioR catalogs are bgziped files[[1]](#footnote-1) that contain 4 columns (\_landmark, \_minBP, \_maxBP, and JSON). A more comprehensive description of the BioR catalog format is in Chapter 3.

To see what is in a catalog, use the zcat command (gzcat on a mac) followed by the catalog filename, followed by less:

$ zcat $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | less

1 10954 11507 {"\_type":"gene","\_landmark":"1","\_strand":"+","\_minBP":10954,"\_maxBP":11507,"gene":"LOC100506145","note":"Derived by automated computational analysis using gene prediction method: GNOMON. Supporting evidence includes similarity to: 1 Protein","pseudo":"","GeneID":"100506145"}

...

Unix less is a good-low-memory command to look at data. Type q <enter> to quit less. A man less at the command line will tell you how to use the less command. I generally use up and down arrows to scroll through the data.

## Showing the Commands in BioRToolkit

All BioR commands start with bior\_ so once the BioRTools is installed and on your path you can type bior\_ followed by the tab key (twice) and it will show you all of the current commands in the toolkit:

$ bior\_

bior\_annotate bior\_drill bior\_lookup bior\_pretty\_print bior\_snpeff bior\_vcf\_to\_tjson

bior\_compress bior\_index bior\_overlap bior\_same\_variant bior\_vep

...

$ bior\_

Table 1 has a more complete description of these commands.

Commands in the toolkit operate on tab delimited data with a VCF style header (starting with “#”). Commands in the toolkit insert additional annotation to the right. Raw annotation is obtained by comparing JSON objects in columns to JSON objects in catalogs. Table 1.0 shows the format of columns <in,out> of each BioR function. For example bior\_vcf\_to\_tjson takes as an input VCF columns (and the header) and outputs VCF + JSON in the last column.

|  |  |
| --- | --- |
| Tool | Function |
| bior\_overlap<TJSON, TJSON> | Extract from catalog all the attributes positions based on genomic coordinates |
| bior\_same\_variant<TJSON,TJSON> | Matches variants based on position, REF/ALT |
| bior\_lookup<TJSON,TJSON> | Matches based on a string/identifier |
| bior\_index<TJSON,INDEX> | Creates and index on a string/identifier |
| bior\_vcf\_to\_tjson<VCF,TJSON> | Converts VCF format to a JSON objects |
| bior\_bed\_to\_json<BED,TJSON> | Converts BED format to JSON objects |
| bior\_drill<TJSON,Tab-Delim> | Extracts key-value relationships from JSON |
| bior\_pretty\_print<TJSON,JSON> | Prints the JSON to the screen in a more readable way |
| bior\_snpeff<VCF,TJSON> | Wraps the SNPEFF tool |
| bior\_vep<VCF,TJSON> | Wraps the VEP tool |
| bior\_annotate<VCF,XLS> | Extract subset of the most common variant annotations [TREAT] |

Table 1: Commands in the current BioR release.

Most every one of these commands supports the –h (help) flag to get information about how to use the command. To get help on bior\_vcf\_to\_tjson type:

$ bior\_vcf\_to\_tjson -h

NAME

bior\_vcf\_to\_tjson -- converts VCF data into JSON as an additional column

SYNOPSIS

bior\_vcf\_to\_tjson [--log] [--help]

...

\_

Several of the above functions use ‘Golden Identifiers’ to match records across catalogs. Table 2 shows the current golden identifiers used in the codebase and what function(s) use them.

|  |  |  |
| --- | --- | --- |
| ‘Golden Identifier’ | Functions | Definition |
| \_landmark | Bior\_overlap, bior\_same\_variant | Chromosome, or sequence ID that the interval is located on |
| \_minBP | Bior\_overlap, bior\_same\_variant | Minimum 1-based position (e.g. NCBI coordinates) on the landmark sequence |
| \_maxBP | Bior\_overlap, bior\_same\_variant | Maximum 1-based position on the landmark sequence |
| \_refAllele | bior\_same\_variant | REF as in VCF standard |
| \_altAlleles | bior\_same\_variant | ALT as in VCF standard |

## Pretty Print

Data in the 4th column of a catalog is stored as JSON. JSON can be deeply nested and hard to read if it is all smashed into one line. BioR has a command bior\_pretty\_print that can make reading JSON text easier. Take the earlier example and replace less with bior\_pretty\_print:

$ zcat $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 UNKNOWN\_1 1

2 #UNKNOWN\_2 10954

3 #UNKNOWN\_3 11507

4 #UNKNOWN\_4 {

"\_type": "gene",

"\_landmark": "1",

"\_strand": "+",

"\_minBP": 10954,

"\_maxBP": 11507,

"gene": "LOC100506145",

"note": "Derived by automated computational analysis using gene prediction method: GNOMON. Supporting evidence includes similarity to: 1 Protein",

"pseudo": "",

"GeneID": "100506145"

}

$

I commonly use –r to specify the row I want to pretty print, this is very useful when handling sparse data. In JSON if there is no value for a given key (instead of reporting NULL), then the key does not show up, so you may need to hunt around in the dataset a bit to find keys of interest.

## Get all Variants in a Gene

Lets do something useful, say we wanted all genetic variants in VCF format that overlap the BRCA1 gene from dbSNP. This section will illustrate how to use BioR to rapidly build a program that does just that. BioR is executed at the Linux/UNIX command line, so any command that is available at the command line can be used with BioR (grep, cut, sed, awk, perl, …). Lets start with the echo command to find BRCA1 in the gene catalog.

$ echo "BRCA1" | bior\_lookup -p gene -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 UNKNOWN\_1 BRCA1

2 LookupPipe {

"\_type": "gene",

"\_landmark": "17",

"\_strand": "-",

"\_minBP": 41196312,

"\_maxBP": 41277500,

"gene": "BRCA1",

"gene\_synonym": "BRCAI; BRCC1; BROVCA1; IRIS; PNCA4; PPP1R53; PSCP; RNF53",

"note": "breast cancer 1, early onset; Derived by automated computational analysis using gene prediction method: BestRefseq.",

"GeneID": "672",

"HGNC": "1100",

"HPRD": "00218",

"MIM": "113705"

}

$

The UNIX pipe (‘|’) allows you to stream the output of one command to the next. In this example, echo prints BRCA1 to the screen. bior\_lookup uses this ID to find the entry in the gene catalog with the key gene and value ‘BRCA1’. Now we have the genomic coordinates for BRCA1. Lets use these positions to find all catalog entries in dbSNP that are between 41196312 and 41277500 on chromosome 17.

$ echo "BRCA1" | bior\_lookup -p gene -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_overlap -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 UNKNOWN\_1 BRCA1

2 LookupPipe {

"\_type": "gene",

"\_landmark": "17",

"\_strand": "-",

"\_minBP": 41196312,

"\_maxBP": 41277500,

"gene": "BRCA1",

"gene\_synonym": "BRCAI; BRCC1; BROVCA1; IRIS; PNCA4; PPP1R53; PSCP; RNF53",

"note": "breast cancer 1, early onset; Derived by automated computational analysis using gene prediction method: BestRefseq.",

"GeneID": "672",

"HGNC": "1100",

"HPRD": "00218",

"MIM": "113705"

}

3 OverlapPipe {

"CHROM": "17",

"POS": "41196363",

"ID": "rs8176320",

"REF": "C",

"ALT": "T",

"QUAL": ".",

"FILTER": ".",

"INFO": {

"RSPOS": 41196363,

"RV": true,

"GMAF": 0.0050,

"dbSNPBuildID": 117,

"SSR": 0,

"SAO": 0,

"VP": "050000800201040517000100",

"GENEINFO": "BRCA1:672",

"WGT": 1,

"VC": "SNV",

"REF": true,

"U3": true,

"VLD": true,

"HD": true,

"GNO": true,

"KGPhase1": true,

"KGPROD": true,

"OTHERKG": true,

"PH3": true

},

"\_id": "rs8176320",

"\_type": "variant",

"\_landmark": "17",

"\_refAllele": "C",

"\_altAlleles": [

"T"

],

"\_minBP": 41196363,

"\_maxBP": 41196363

}

$

This command shows the first match in dbSNP that overlaps the BRCA1 gene according to the NCBI annotation. The version of dbSNP used to publish the catalog was a VCF file so many fields from the VCF standard are represented in the JSON. A combination of the UNIX cut command and bior\_drill can quickly extract a VCF file. When trying this example, decompose the commands and use them one at a time to understand what each command is doing.

$ echo "BRCA1" | bior\_lookup -p gene -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_overlap -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_drill -p CHROM -p POS -p ID -p REF -p ALT -p QUAL -p FILTER -p NONE | cut -f 3-10

CHROM POS ID REF ALT QUAL FILTER NONE

17 41196363 rs8176320 C T . . .

17 41196368 rs184237074 C T . . .

17 41196372 rs189382442 T C . . .

17 41196403 rs182218567 A G . . .

17 41196408 rs12516 G A . . .

17 41196534 rs34214126 A AG . . .

17 41196582 rs111791349 C T . . .

17 41196625 rs185966495 G C . . .

17 41196795 rs1060921 T A . . .

...

$

A simple VCF file constructed for all variants in the BRCA1 gene! There are a few small fixes that will need to be made to make it up to the VCF standard, and this quickstart glosses over many features (e.g. what is going on with “NONE” vs INFO? In the header). These issues and many others will be covered in more detail in the following sections.

# 3. BioR Catalogs

## The BioR Catalog Format

BioR enables users to rapidly transform tabular, hierarchical (e.g. XML) relational, and flat files into catalogs that can be indexed and searched. Catalogs are read-only snapshots of annotation data. In production, we snapshot datasets from outside groups and run an automated ‘publishing’ process that keeps all of the BioR catalogs up to date with reference data sources. Data in catalogs is organized as a BED-JSON hybrid (a subset of TJSON, or tab-delimited JSON, described in the paper). Columns 1-3 are identical to the required fields in BED files[[2]](#footnote-2)[[3]](#footnote-3) and thus allow many existing tools to work directly on BioR catalogs. Column 4 is a JSON string encoded object representing the entire contents of the original file. BioRTools depends on *golden identifiers* (identifiers that start with an underscore) to enable search. *Golden identifiers* are semantically consistent tightly controlled fields that are used by the toolkit uses to enable filtering and search (e.g. \_minBP/\_maxBP corresponds to one-baseed fully-closed genomic min/max).

## Catalog Creation Details

As an illustration, we will take a single gene BRCA1 and show it in the original annotation file and in BioR Catalog structure.

ORIGINAL

The gene BRCA1 is shown below from the original Genbank formatted file

hs\_ref\_GRCh37.p10\_chr17.gbs.gz:

gene complement(41196312..41277500)

/gene="BRCA1"

/gene\_synonym="BRCAI; BRCC1; BROVCA1; IRIS; PNCA4;

PPP1R53; PSCP; RNF53"

/note="breast cancer 1, early onset; Derived by automated

computational analysis using gene prediction method:

BestRefseq."

/db\_xref="GeneID:672"

/db\_xref="HGNC:1100"

/db\_xref="HPRD:00218"

/db\_xref="MIM:113705"

CATALOG

Below is the corresponding Catalog structure for the final column of gene BRCA1.

{

"gene": "BRCA1",

"gene\_synonym": "BRCAI; BRCC1; BROVCA1; IRIS; PNCA4; PPP1R53; PSCP; RNF53",

"note": "breast cancer 1, early onset; Derived by automated computational analysis using gene prediction method: BestRefseq.",

"GeneID": "672",

"HGNC": "1100",

"HPRD": "00218",

"MIM": "113705",

"\_type": "gene",

"\_landmark": "17",

"\_strand": "-",

"\_minBP": 41196312,

"\_maxBP": 41277500

}

The catalog format is simple, easy to read, and can be readily processed by third party JSON libraries. The format is also incredibly flexible, and has allowed us to ingest deeply nested XML structures and complex relational schemas into BioR. Construction of catalogs can be done with whatever programming language the user is familiar. Once the raw data is formatted bgzip and tabix can be used to compress and then index the catalog for genomic coordinate-based queries.

## Catalogs Availible In BioR

The BioR team has created more than 8,000 catalogs relevant to variant annotation from the following sources.

**Data sources currently available in BioR**

|  |  |  |
| --- | --- | --- |
| **Datasource** | **URL** | **Version** |
| 1000Genomes | http://www.1000genomes.org/category/ftp | 20110521 |
| BGI | http://soap.genomics.org.cn/soapsnp.html | hg19 |
| COSMIC | http://cancer.sanger.ac.uk/cancergenome/projects/cosmic/ | V63 |
| dbSNP | http://www.ncbi.nlm.nih.gov/snp/ | 137 |
| ESP6500 | https://esp.gs.washington.edu/drupal/ | build37 |
| HapMap | http://hapmap.ncbi.nlm.nih.gov/ | 2010-08\_phaseII+III |
| HGNC | http://www.genenames.org/ | 2012\_08\_12 |
| miRBase | http://www.mirbase.org/ | 8\_12\_12 |
| NCBIGene | http://www.ncbi.nlm.nih.gov/gene | GRCh37\_p10 |
| OMIM | http://www.omim.org/ | 2013\_02\_27 |
| UCSC | http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/ | hg19 |

# Examples Matching Genomic Features

## Positional Matches Using Tabix

BioR uses the same technology for compression (BGZIP) and coordinate based indexing as Tabix[[4]](#footnote-4). This means that coordinate based queries can use the traditional Tabix Commands. For example to show all the genes in a BioR catalog on Chromosome 17 in the range 41196312-41277500:

$ which tabix

/usr/bin/tabix

$ which bgzip

/usr/bin/bgzip

$ tabix $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz 17:41196312-41277500

17 41196312 41277500 {"\_type":"gene","\_landmark":"17","\_strand":"-","\_minBP":41196312,"\_maxBP":41277500,"gene":"BRCA1","gene\_synonym":"BRCAI; BRCC1; BROVCA1; IRIS; PNCA4; PPP1R53; PSCP; RNF53","note":"breast cancer 1, early onset; Derived by automated computational analysis using gene prediction method: BestRefseq.","GeneID":"672","HGNC":"1100","HPRD":"00218","MIM":"113705"}

17 41231278 41231833 {"\_type":"gene","\_landmark":"17","\_strand":"+","\_minBP":41231278,"\_maxBP":41231833,"gene":"RPL21P4","gene\_synonym":"RPL21\_58\_1548","note":"ribosomal protein L21 pseudogene 4; Derived by automated computational analysis using gene prediction method: Curated Genomic.","pseudo":"","GeneID":"140660","HGNC":"17959"}

bior@biordev:~$

On the RCF, tabix is located at: /projects/bsi/bictools/apps/alignment/tabix/0.2.5/tabix

## Annotating Variants with Genes that Overlap

A common and simple use of BioR is to ask what genes overlap variants of interest. NCBI Generates an annotation of genes that they store here: [ftp.ncbi.nih.gov/genomes/Homo\_sapiens](ftp://ftp.ncbi.nih.gov/genomes/Homo_sapiens)

This set of files is one of the authoritative sources for storing both the IDs for genes and the genomic coordinates. Unfortunately the gbs file is hard to use without the use of libraries. BioR allows you to do many quick and dirty analyses based on the position of genes. The following example assumes a VCF-like file with only 8 columns e.g.:

$ $ head example.vcf

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO

1 215848808 rs116645811 G A . . .

21 26965148 rs1135638 G A . . .

21 26965172 rs010576 T C . . .

21 26965205 rs1057885 T C . . .

21 26976144 rs116331755 A G . . .

21 26976222 rs7278168 C T . . .

21 26976237 rs7278284 C T . . .

21 26978790 rs75377686 T C . . .

$

Now, lets annotate these variants based on the genes they overlap:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene | cut -f 9 --complement > example.vcf.genes

$ head example.vcf.genes

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO GeneID gene

1 215848808 rs116645811 G A . . . 7399 USH2A

21 26965148 rs1135638 G A . . . 54148 MRPL39

21 26965172 rs010576 T C . . . 54148 MRPL39

21 26965205 rs1057885 T C . . . 54148 MRPL39

21 26976144 rs116331755 A G . . . 54148 MRPL39

21 26976222 rs7278168 C T . . . 54148 MRPL39

21 26976237 rs7278284 C T . . . 54148 MRPL39

21 26978790 rs75377686 T C . . . 54148 MRPL39

$

Feel free to use bior\_pretty\_print instead of bior\_drill to explore the data. Try drilling out other columns. In-fact, if anything is unclear, break the command apart and run parts of the command to get a better understanding of what steps are doing (e.g. run cat, then cat | bior\_vcf\_to\_tjson | bior\_pretty\_print, then cat | bior\_vcf\_to\_tjson | bior\_overlap | bior\_pretty\_print, and so on to understand the transformations done in the pipeline).

This is a simple script based on the above technique to show the genes that contain variants in your VCF file:

$ $ head example.vcf

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO

1 215848808 rs116645811 G A . . .

21 26965148 rs1135638 G A . . .

21 26965172 rs010576 T C . . .

21 26965205 rs1057885 T C . . .

21 26976144 rs116331755 A G . . .

21 26976222 rs7278168 C T . . .

21 26976237 rs7278284 C T . . .

21 26978790 rs75377686 T C . . .

$

In many examples, more than one gene may overlap a variant. By default, BioR will ‘fan-out’ the rows replicating each input row for each result in the result set.

Here is an example of a quick script to look for rsIDs in an entire exome sequencing run (followed by variant calling formatted as VCF) where we annotate the rsID-gene relationships:

$ cat /data2/bsi/staff\_analysis/m088341/BioR/exome\_test/s\_P68.variants.final.vcf | cut -f 3 | grep -v "\." | bior\_lookup -p ID -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | grep -v "##" | grep -v "^ID"| bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p gene | cut -f 2 --complement | head

#UNKNOWN\_1 gene

rs146405013 LINC00115

rs3115849 LINC00115

rs61768173 LINC00115

rs4970461 LOC100130417

rs4372192 SAMD11

rs6605066 SAMD11

rs6672356 SAMD11

rs6605067 SAMD11

rs6605067 NOC2L

This is one way to get the variants that overlap more than one gene:

$ cat /data2/bsi/staff\_analysis/m088341/BioR/exome\_test/s\_P68.variants.final.vcf | cut -f 3 | grep -v "\." | bior\_lookup -p ID -d $BIOR\_CATALOG/dbSNP/137/00-All\_GRCh37.tsv.bgz | grep -v "##" | grep -v "^ID"| bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p gene | cut -f 2 --complement | grep -v "#UNKNOWN" | grep -v "\." | cut -f 1 | uniq -c | grep -v "1 rs"

2 rs6605067

2 rs2839

2 rs262688

2 rs1043703

2 rs17692

2 rs2294532

2 rs1043683

2 rs1043681

2 rs10523

2 rs649639

...

In this case, the variants are sorted, so uniq can be used directly, but in other cases, consider the unix sort command (right before uniq). How many variants overlap at least two genes in this exome sample?

$ $ wc -l moreThan1.rsID

3778 moreThan1.rsID

## Compressing output to enforce 1-1 semantics

Lets say we want to enforce 1-in/1-out semantics (no duplicated variants), BioR has a utility (bior\_compress) that can help with that. Here we will start directly with the rare variants. A simple sed command replaces the counts and gets us back to rsIDs.

$ sed 's/ .\* //' < moreThan1.rsID

rs6605067

rs2839

rs262688

rs1043703

rs17692

rs2294532

rs1043683

rs1043681

rs10523

rs649639

...

Now we can annotate them in much the same way as before: (or we could modify the above pipeline – probably want to do that when we want to keep all the input data, but this gives us example variants that overlap two genes quickly). Run this example without bior\_compress to see the default behavior when there is more than one result for a row.

$ sed 's/ .\* //' < moreThan1.rsID | bior\_lookup -p ID -d $BIOR\_CATALOG/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p gene | cut -f 1,3 | bior\_compress 2 | head

#UNKNOWN\_1 gene

rs6605067 SAMD11|NOC2L

rs2839 SAMD11|NOC2L

rs262688 PRKCZ|LOC100506504

rs1043703 THAP3|DNAJC11

rs17692 THAP3|DNAJC11

rs2294532 THAP3|DNAJC11

rs1043683 THAP3|DNAJC11

rs1043681 THAP3|DNAJC11

rs10523 THAP3|DNAJC11

$

# Expanded Genes (Xrefs)

The HUGO/HGNC table has database cross-references for gene ids and names. The bior\_lookup command allows us to ‘walk’ these cross references. Here is an example:

$ bior\_vcf\_to\_tjson < example.vcf | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene | cut -f 9 --complement | bior\_lookup -d $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz -p Approved\_Symbol | bior\_drill -p Approved\_Symbol -p Entrez\_Gene\_ID -p Ensembl\_Gene\_ID -p UniProt\_ID

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO GeneID gene Approved\_Symbol Entrez\_Gene\_ID Ensembl\_Gene\_ID UniProt\_ID

1 215848808 rs116645811 G A . . . 7399 USH2A USH2A 7399 ENSG00000042781 O75445

21 26965148 rs1135638 G A . . . 54148 MRPL39 MRPL39 54148 ENSG00000154719 Q9NYK5

21 26965172 rs010576 T C . . . 54148 MRPL39 MRPL39 54148 ENSG00000154719 Q9NYK5

21 26965205 rs1057885 T C . . . 54148 MRPL39 MRPL39 54148 ENSG00000154719 Q9NYK5

21 26976144 rs116331755 A G . . . 54148 MRPL39 MRPL39 54148 ENSG00000154719 Q9NYK5

...

Lookup requires that the referenced column (last by default change it with the –c flag) is an ID that has been indexed in the source catalog. ID based indexes are stored in a directory called ‘index’ at the same level in the filesystem as the catalog. For example, here are all of the indexes for the HGNC catalog:

## Indexing Catalogs

$ ls $bior/hgnc/2012\_08\_12/index/

hgnc\_GRCh37.Approved\_Symbol.idx.h2.db hgnc\_GRCh37.Entrez\_Gene\_ID.idx.h2.db hgnc\_GRCh37.UniProt\_ID.idx.h2.db

hgnc\_GRCh37.Ensembl\_Gene\_ID.idx.h2.db hgnc\_GRCh37.HGNC\_ID.idx.h2.db

On the RCF, the administrators are very restrictive about space, so additional indexes must be placed in user/project space. Stand-alone installs can easily place all indexes in the index directory directly under the directory the catalog is in. BioR allows users to make additional indexes through the bior\_index command. The help documentation contains:

1) bior\_index -d $BIOR\_CATALOG/NCBIGene/GRCh37\_p10/genes.tsv.bgz -p HGNC

OR

2) bior\_index -d $BIOR\_CATALOG/NCBIGene/GRCh37\_p10/genes.tsv.bgz -p HGNC -i

/data/myindexes/genes.HGNC.idx.h2.db

Option 1, used by the BioR team to create indexes, will create the index file in the index folder in the same directory as the catalog (as shown in the example for hgnc above). Option 2, most often used by BioR end users, creates the index in any directory. When using an index created via the second method, you need to adjust the lookup command appropriately. This will be covered more comprehensively in the section on creating custom catalogs.

To make an index, use bior\_pretty\_print to show the contents of the catalog, and then run the index command.

## Looking Up Information about a Gene

Say we wanted to find "Approved\_Symbol", "Entrez\_Gene\_ID", "Ensembl\_Gene\_ID", "UniProt\_ID", and other common alternative symbols for every gene we have in a list. We can use the BioR lookup command:

First, we don't know the catalog Structure of HGNC, here is a way to look at the structure of a catalog:

$ zcat $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 UNKNOWN\_1 .

2 #UNKNOWN\_2 0

3 #UNKNOWN\_3 0

4 #UNKNOWN\_4 {

"HGNC\_ID": "HGNC:5",

"Approved\_Symbol": "A1BG",

"Approved\_Name": "alpha-1-B glycoprotein",

"Status": "Approved",

"Locus\_Type": "gene with protein product",

"Locus\_Group": "protein-coding gene",

"Previous\_Symbols": [],

"Previous\_Names": [],

"Synonyms": [],

"Name\_Synonyms": [],

"Chromosome": "19q",

"Date\_Approved": "1989-06-30",

"Date\_Modified": "2010-07-08",

"Accession\_Numbers": [],

"Enzyme\_IDs": [],

"Entrez\_Gene\_ID": "1",

"Ensembl\_Gene\_ID": "ENSG00000121410",

…

"Pubmed\_IDs": [

"2591067"

],

"RefSeq\_IDs": [

"NM\_130786"

],

"Record\_Type": "Standard",

"Primary\_IDs": [],

"Secondary\_IDs": [],

"CCDS\_IDs": [

"CCDS12976.1"

],

"VEGA\_IDs": [],

"mapped\_GDB\_ID": "GDB:119638",

"mapped\_Entrez\_Gene\_ID": "1",

"mapped\_OMIM\_ID": "138670",

"mapped\_RefSeq": "NM\_130786",

"UniProt\_ID": "P04217",

"mapped\_Ensembl\_ID": "ENSG00000121410",

"UCSC\_ID": "uc002qsd.4",

"mapped\_Mouse\_Genome\_Database\_ID": "MGI:2152878",

"mapped\_Rat\_Genome\_Database\_ID": "RGD:69417"

}

$

To join the information in this catalog, to the information that we have collected in the gene table, we need to tell bior what field in the HGNC table matches the LAST column in our sample data + annotation. In this case, we will join on approved symbol (note: if you ever get an error with doing a lookup, you may need an index file - look into the bior\_index command documentation, using –h for help, or contact the bior team for help – running bior commands ).

[m102417@crick4 ~]$ cat mygenes.txt

MRPL39

PANX2

BRCA1

[m102417@crick4 ~]$ cat mygenes.txt | bior\_lookup -d $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz -p Approved\_Symbol

#UNKNOWN\_1 LookupPipe

MRPL39 {"HGNC\_ID":"HGNC:14027","Approved\_Symbol":"MRPL39","Approved\_Name":"mitochondrial ribosomal protein L39","Status":"Approved","Locus\_Type":"gene with protein product","Locus\_Group":"protein-coding gene","Previous\_Symbols":[],"Previous\_Names":[],"Synonyms":["RPML5","MRP-L5","MGC104174","PRED66","PRED22","C21orf92","L39mt","MSTP003","MGC3400","FLJ20451"],"Name\_Synonyms":[],"Chromosome":"21q11.2-q21","Date\_Approved":"2001-02-28","Date\_Modified":"2012-09-13","Accession\_Numbers":["AB051346"],"Enzyme\_IDs":[],"Entrez\_Gene\_ID":"54148","Ensembl\_Gene\_ID":"ENSG00000154719","Mouse\_Genome\_Database\_ID":"MGI:1351620","Specialist\_Database\_Links":"<!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <a href=\"http://www.sanger.ac.uk/perl/genetics/CGP/cosmic?action=gene&amp;ln=MRPL39\">COSMIC</a><!--,--> <!--,--> <!--,--> <!--,--> <!--,--> <!--,--> ","Specialist\_Database\_IDs":["","","","","","","","","","","MRPL39","","","","","",""],"Pubmed\_IDs":["11543634"],"RefSeq\_IDs":["NM\_017446"],"Gene\_Family\_Tag":"MRPL","Gene\_family\_description":"\"Mitochondrial ribosomal proteins / large subunits\"","Record\_Type":"Standard","Primary\_IDs":[],"Secondary\_IDs":[],"CCDS\_IDs":["CCDS13573.1","CCDS33522.1"],"VEGA\_IDs":["OTTHUMG00000078371"],"mapped\_GDB\_ID":"GDB:11503068","mapped\_Entrez\_Gene\_ID":"54148","mapped\_OMIM\_ID":"611845","mapped\_RefSeq":"NM\_017446","UniProt\_ID":"Q9NYK5","mapped\_Ensembl\_ID":"ENSG00000154719","UCSC\_ID":"uc002yln.3","mapped\_Mouse\_Genome\_Database\_ID":"MGI:1351620"}

PANX2 ...

...

$

Now lets extract Entrez\_Gene\_ID, Ensembl\_Gene\_ID, and UniProt\_ID from the catalog:

[m102417@crick4 ~]$ cat mygenes.txt | bior\_lookup -d /data5/bsi/catalogs/bior/v1/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz -p Approved\_Symbol | bior\_drill -p Entrez\_Gene\_ID -p Ensembl\_Gene\_ID -p UniProt\_ID

#UNKNOWN\_1 Entrez\_Gene\_ID Ensembl\_Gene\_ID UniProt\_ID

MRPL39 54148 ENSG00000154719 Q9NYK5

PANX2 56666 ENSG00000073150 Q96RD6

BRCA1 672 ENSG00000012048 P38398

[m102417@crick4 ~]$

## Example of Walking Cross References

The HGNC table does not contain information about the disease/condition, only the ID in OMIM. Lets say you would like to also find this information for a select set of genes. In this case, we can use two catalogs, (1) the HGNC catalog and (2) the genemap directly from OMIM. The figure below shows the contents of the genemap catalog currently in BioR:

$ zcat $bior/omim/2013\_02\_27/genemap\_GRCh37.tsv.bgz | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 UNKNOWN\_1 .

2 #UNKNOWN\_2 .

3 #UNKNOWN\_3 .

4 #UNKNOWN\_4 {

"Chromosome.Map\_Entry\_Number": 1.1,

"MonthEntered": 9,

"Day": 11,

"Year": 95,

"Cytogenetic\_location": "1pter-p36.13",

"GeneSymbols": "CCV",

"Gene\_Status": "P",

"Title": "Cataract, congenital, Volkmann type",

"Title\_cont": "",

"MIM\_Number": 115665,

"Method": "Fd",

"Comments": "",

"Disorders": "Cataract, congenital, Volkmann type (2)",

"Disorders\_cont": " "

}

$

In this catalog, "MIM\_Number" represents the OMIM id for the “Disorder” free text field describing the disease. Given a list of genes, if we want the value of the “Disorder” field in OMIM we can cross-walk from the gene list through the HGNC catalog to find the MIM number and then again to genemap catalog to produce a Gene-OMIM\_ID-Disorder file:

$ cat mygenes.txt

MRPL39

PANX2

BRCA1

$ cat mygenes.txt | bior\_lookup -d $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz -p Approved\_Symbol | bior\_drill -p mapped\_OMIM\_ID | bior\_lookup -d $bior/omim/2013\_02\_27/genemap\_GRCh37.tsv.bgz -p MIM\_Number | bior\_drill -p Disorders

#UNKNOWN\_1 mapped\_OMIM\_ID Disorders

MRPL39 611845

PANX2 608421 .

BRCA1 113705 {Breast-ovarian cancer, familial, 1}, 604370 (3); {Pancreatic cancer,

$

Note: period '.' always means the value was not in the dataset. So in this case, some genes are not associated with disorders in OMIM.

## Generating an OMIM Disorder Report for a Set of rsIDs

Want OMIM

cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -­‐d $catalogs/NCBIGene/GRCh37\_p10/ genes.tsv.bgz | bior\_drill -­‐p GeneID -­‐p gene -­‐p MIM | cut -­‐f9 -­‐ -­‐complement | bior\_lookup -­‐d $catalogs/omim/2013\_02\_27/ genemap\_GRCh37.tsv.bgz -­‐p MIM\_Number | bior\_drill -­‐p Disorders > example.w\_omim.

Use lookup to also find any disease/condition information in OMIM. First, the gene catalog just happens to have the OMIM id ("MIM"), so alter the command to drill that out:

Want OMIM

cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -­‐d $catalogs/NCBIGene/GRCh37\_p10/ genes.tsv.bgz | bior\_drill -­‐p GeneID -­‐p gene -­‐p MIM | cut -­‐f9 -­‐ -­‐complement | bior\_lookup -­‐d $catalogs/omim/2013\_02\_27/ genemap\_GRCh37.tsv.bgz -­‐p MIM\_Number | bior\_drill -­‐p Disorders > example.w\_omim.

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene | cut -f 9 --complement | bior\_lookup -d $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz -p Approved\_Symbol

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene -p MIM | cut -f 9 --complement

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO GeneID gene MIM

1 215848808 rs116645811 G A . . . 7399 USH2A 608400

1 215848808 rs116645811 G T . . . 7399 USH2A 608400

…

$

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene -p MIM | cut -f 9 --complement | bior\_lookup -d $bior/omim/2013\_02\_27/genemap\_GRCh37.tsv.bgz -p MIM\_Number | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 1

2 POS 215848808

3 ID rs116645811

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 GeneID 7399

10 gene USH2A

11 MIM 608400

12 LookupPipe {

"Chromosome.Map\_Entry\_Number": 1.1272,

"MonthEntered": 1,

"Day": 27,

"Year": 4,

"Cytogenetic\_location": "1q41",

"GeneSymbols": "USH2A, RP39",

"Gene\_Status": "C",

"Title": "Usherin",

"Title\_cont": "",

"MIM\_Number": 608400,

"Method": "Fd",

"Comments": "",

"Disorders": "Usher syndrome, type 2A, 276901 (3); Retinitis pigmentosa 39, 613809",

"Disorders\_cont": " ",

"Mouse\_correlate": "1(Ush2a)"

}

$

Looks like we want the column "Disorders":

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene -p MIM | cut -f 9 --complement | bior\_lookup -d $bior/omim/2013\_02\_27/genemap\_GRCh37.tsv.bgz -p MIM\_Number | bior\_drill -p Disorders

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO GeneID gene MIM Disorders

1 215848808 rs116645811 G A . . . 7399 USH2A 608400 Usher syndrome, type 2A, 276901 (3); Retinitis pigmentosa 39, 613809

...

22 50616806 rs5771206 A G . . . 56666 PANX2 608421 .

$

OK, lets go and get some information from some variant catalogs that are not Allele frequencies:

First, dbSNP has all kinds of useful information including

"INFO.dbSNPBuildID":

"INFO.SSR": SSR 1 Integer 247,783 0.49% SNP Suspect Reason Code SNP Suspect Reason Code, 0 - unspecified, 1 - Paralog, 2 - byEST, 3 - Para\_EST, 4 - oldAlign, 5 - other. Count in column D is non-zero

Sequence Annotation Flags

"INFO.SCS": Integer 12,533 0.02% SNP Clinical Significance SNP Suspect Reason Code, 0 - unspecified, 1 - Paralog, 2 - byEST, 3 - Para\_EST, 4 - oldAlign, 5 - other. Count in column D is non-zero

"INFO.CLN": CLN 0 Flag 31,524 0.06% SNP is Clinical Includes LSDB,OMIM,TPA,Diagnostic

"INFO.SAO": SAO 1 Integer 14,908 0.03% SNP Allele Origin SNP Allele Origin: 0 - unspecified, 1 - Germline, 2 - Somatic, 3 - Both. Count in column D is non-zero

"\_id": The rs\_id, a (near)universal identifier for the Variant.

(to see a compiled list of what is in this, go to the bsi documentation: http://bsiweb.mayo.edu/dbsnp)

This text file is a good guide (downloaded from dbSNP: ftp://ftp.ncbi.nih.gov/snp/organisms/human\_9606/VCF/00-snp\_info\_tags.txt)

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

...

}

10 SameVariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

"RSPOS": 26965148,

"RV": true,

"GMAF": 0.2395,

"dbSNPBuildID": 86,

"SSR": 0,

"SAO": 0,

"VP": "05030000030507051f000100",

"GENEINFO": "MRPL39:54148",

"WGT": 1,

"VC": "SNV",

"S3D": true,

"SLO": true,

"REF": true,

"SYN": true,

"ASP": true,

"VLD": true,

"G5A": true,

"G5": true,

"HD": true,

"GNO": true,

"KGPhase1": true,

"KGPilot123": true,

"KGPROD": true,

"OTHERKG": true,

"PH3": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

$

To match variants, use same\_variant:

Now build a table with: rs\_id, dbSNPBuildID, SSR, SCS, CLN, SAO, and CLN, do this:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_drill -p \_id -p dbSNPBuildID -p INFO.SSR -p INFO.SCS -p INFO.CLN -p INFO.SAO -p INFO.CLN | cut -f 9 --complement

unfortunately, the variants in this example file, did not have any results, as these annotations are rather sparse. Finding variants with these properties can be a trick. Here is a trick that I use to cat all variants from a specific gene:

$ zcat $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | grep "\"gene\":\"BRCA1\""

17 41196312 41277500 {"\_type":"gene","\_landmark":"17","\_strand":"-","\_minBP":41196312,"\_maxBP":41277500,"gene":"BRCA1","gene\_synonym":"BRCAI; BRCC1; BROVCA1; IRIS; PNCA4; PPP1R53; PSCP; RNF53","note":"breast cancer 1, early onset; Derived by automated computational analysis using gene prediction method: BestRefseq.","GeneID":"672","HGNC":"1100","HPRD":"00218","MIM":"113705"}

$

Then to find a variant in dbSNP with an SAO annotation:

$ zcat $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | grep "\"gene\":\"BRCA1\"" | bior\_overlap -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | grep SAO | bior\_pretty\_print

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 UNKNOWN\_1 17

2 #UNKNOWN\_2 41196312

3 #UNKNOWN\_3 41277500

4 #UNKNOWN\_4 {

"\_type": "gene",

"\_landmark": "17",

"\_strand": "-",

"\_minBP": 41196312,

"\_maxBP": 41277500,

"gene": "BRCA1",

"gene\_synonym": "BRCAI; BRCC1; BROVCA1; IRIS; PNCA4; PPP1R53; PSCP; RNF53",

"note": "breast cancer 1, early onset; Derived by automated computational analysis using gene prediction method: BestRefseq.",

"GeneID": "672",

"HGNC": "1100",

"HPRD": "00218",

"MIM": "113705"

}

5 #UNKNOWN\_5 {

"CHROM": "17",

"POS": "41196363",

"ID": "rs8176320",

"REF": "C",

"ALT": "T",

"QUAL": ".",

"FILTER": ".",

"INFO": {

"RSPOS": 41196363,

"RV": true,

"GMAF": 0.0050,

"dbSNPBuildID": 117,

"SSR": 0,

"SAO": 0,

"VP": "050000800201040517000100",

"GENEINFO": "BRCA1:672",

"WGT": 1,

"VC": "SNV",

"REF": true,

"U3": true,

"VLD": true,

"HD": true,

"GNO": true,

"KGPhase1": true,

"KGPROD": true,

"OTHERKG": true,

"PH3": true

},

"\_id": "rs8176320",

"\_type": "variant",

"\_landmark": "17",

"\_refAllele": "C",

"\_altAlleles": [

"T"

],

"\_minBP": 41196363,

"\_maxBP": 41196363

}

$

COSMIC:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/cosmic/v63/CosmicCompleteExport\_GRCh37.tsv.bgz | bior\_pretty\_print -r 40

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 40190405

3 ID rs115908228

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

...

}

10 SameVariantPipe {

"Gene\_name": "ETS2",

"Accession\_Number": "ENST00000360214",

"HGNC\_ID": "3489",

"Sample\_name": "107702",

"ID\_sample": "1520464",

"ID\_tumour": "1442839",

"Primary\_site": "breast",

"Site\_subtype": "NS",

"Primary\_histology": "carcinoma",

"Histology\_subtype": "HER-positive\_carcinoma",

"Genome-wide\_screen": "n",

"Mutation\_ID": "94254",

"Mutation\_CDS": "c.646G\u003eA",

"Mutation\_AA": "p.G216S",

"Mutation\_Description": "Substitution - Missense",

"Mutation\_GRCh37\_genome\_position": "21:40190405-40190405",

"Mutation\_GRCh37\_strand": "+",

"Mutation\_somatic\_status": "Confirmed somatic variant",

"Pubmed\_PMID": "20668451",

"Sample\_source": "NS",

"Tumour\_origin": "primary",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 40190405,

"\_maxBP": 40190405,

"\_id": "."

}

$

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/cosmic/v63/CosmicCompleteExport\_GRCh37.tsv.bgz | bior\_drill -p Mutation\_ID -p Mutation\_CDS -p Mutation\_AA -p Mutation\_GRCh37\_strand | cut -f 9 --complement

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO Mutation\_ID Mutation\_CDS Mutation\_AA Mutation\_GRCh37\_strand

1 215848808 rs116645811 G A . . . . . . .

1 215848808 rs116645811 G T . . . . . . .

...

21 40190405 rs115908228 G A . . . 94254 c.646G>A p.G216S +

...

22 30857373 rs2240345 A C . . . 330401 c.1005T>G p.D335E -

...

22 39621797 rs35978693 G T . . . 39683 c.657C>A p.P219P -

$

Want UCSC Tracks (blacklisted)cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -­‐d $catalogs/ucsc/hg19/ wgEncodeDacMapabilityConsensusExcludable\_GR

Ch37.tsv.bgz | bior\_drill -­‐p score | complement > example.w\_ucsc.vcf

UCSC:

The UCSC catalogs related to TREAT are the following:

export ucsc=$bior/ucsc/ ;

export blacklistedFile=$ucsc/hg19/wgEncodeDacMapabilityConsensusExcludable\_GRCh37.tsv.bgz ;

export repeatFile=$ucsc/hg19/rmsk\_GRCh37.tsv.bgz ;

export regulationFile=$ucsc/hg19/oreganno\_GRCh37.tsv.bgz ;

export uniqueFile=$ucsc/hg19/wgEncodeDukeMapabilityRegionsExcludable\_GRCh37.tsv.bgz ;

export tssFile=$ucsc/hg19/switchDbTss\_GRCh37.tsv.bgz ;

export tfbsFile=$ucsc/hg19/tfbsConsSites\_GRCh37.tsv.bgz ;

export enhancerFile=$ucsc/hg19/vistaEnhancers\_GRCh37.tsv.bgz ;

export conservationFile=$ucsc/hg19/phastConsElements46wayPrimates\_GRCh37.tsv.bgz ;

To annotate with any of these files, do something like this:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $blacklistedFile | bior\_drill -p score | cut -f 9 --complement

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO score

1 215848808 rs116645811 G A . . . .

1 215848808 rs116645811 G T . . . .

1 215848808 rs116645811 G G . . . .

1 215848808 rs116645811 G C . . . .

...

unfortunately, our example file does not overlap many of these rare features. Another way to think about this is "what genes of interest overlap some UCSC genomic feature".

$ zcat $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_overlap -d $blacklistedFile | grep -v "{}" | bior\_drill -c -2 -p gene | cut -f 5

gene

MTND1P23

MTND2P28

TTC34

RNU1-1

RSPO1

HFM1

AMY2A

NOTCH2NL

NBPF17P

PMF1

PMF1-BGLAP

PCNXL2

RYR2

MTND2P27

This list of genes could then be used in a lookup query later, or you could cut the JSON instead of the gene name and use that to overlap the data in your VCF file in a filtering process.

A similar technique can be use to pair down the variants based on those variants that you do NOT want because overlapping some genomic feature would indicate it is unlikely to be significant.

## Putting it all Together – Making a Genomic Feature Annotation Program

Below is a simple example of an annotation program using the simple scripts.

$ cat treatGF.bior

bior\_vcf\_to\_tjson < /dev/stdin \

| bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz \

| bior\_drill -p gene -p GeneID -p MIM \

| bior\_lookup -d $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz -p Approved\_Symbol -c -3 \

| bior\_drill -p Approved\_Symbol -p Entrez\_Gene\_ID -p Ensembl\_Gene\_ID -p UniProt\_ID \

| bior\_lookup -d $bior/omim/2013\_02\_27/genemap\_GRCh37.tsv.bgz -p MIM\_Number -c -5 \

| bior\_drill -p Disorders \

| bior\_overlap -d $bior/mirbase/release19/hsa\_GRCh37.p5.tsv.bgz -c -9 \

| bior\_drill -p ID \

| bior\_overlap -d $bior/ucsc/hg19/wgEncodeDacMapabilityConsensusExcludable\_GRCh37.tsv.bgz -c -10 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/phastConsElements46way\_GRCh37.tsv.bgz -c -11 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/oreganno\_GRCh37.tsv.bgz -c -12 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/tfbsConsSites\_GRCh37.tsv.bgz -c -13 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/switchDbTss\_GRCh37.tsv.bgz -c -14 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/vistaEnhancers\_GRCh37.tsv.bgz -c -15 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/wgEncodeDukeMapabilityRegionsExcludable\_GRCh37.tsv.bgz -c -16 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/rmsk\_GRCh37.tsv.bgz -c -17 \

| bior\_drill -p score \

| bior\_overlap -d $bior/ucsc/hg19/wgEncodeDukeMapabilityRegionsExcludable\_GRCh37.tsv.bgz -c -18 \

| bior\_drill -p score \

| ./removeJSON.pl

$

# Examples Matching Alleles (bior\_same\_variant)

Allele Frequencies:

on the RCF:

BGI:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/BGI/hg19/LuCAMP\_200exomeFinal.maf\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"chromosome\_id": "chr21",

"genomic\_position": 25887019,

"index\_of\_major\_allele": 0,

"major\_allele": "A",

"index\_of\_minor\_allele": 2,

"minor\_allele": "G",

"number\_A": 710,

"number\_C": 1,

"number\_G": 428,

"number\_T": 2,

"estimated\_minor\_allele\_freq": 0.278705,

"estimated\_major\_allele\_freq": 0.721295,

"is\_in\_dbSNP": 1,

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148,

"\_type": "variant",

"\_id": "."

}

$

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/BGI/hg19/LuCAMP\_200exomeFinal.maf\_GRCh37.tsv.bgz | bior\_drill -p estimated\_major\_allele\_freq -p estimated\_minor\_allele\_freq | cut --complement -f 9

…

22 30823196 rs5753130 T C . . . 0.576518 0.423482

22 30856121 rs35764129 G A . . . 0.957359 0.042641

22 30857373 rs2240345 A C . . . 0.610933 0.389067

22 30857448 rs5749104 A G . . . 0.587232 0.412768

22 30857645 rs114917409 C G . . . . .

22 30858149 rs115111929 A C . . . . .

22 30860830 rs2269961 C T . . . 0.808176 0.191824

…

dbSNP:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

"RSPOS": 26965148,

"RV": true,

"GMAF": 0.2395,

"dbSNPBuildID": 86,

"SSR": 0,

"SAO": 0,

"VP": "05030000030507051f000100",

"GENEINFO": "MRPL39:54148",

"WGT": 1,

"VC": "SNV",

"S3D": true,

"SLO": true,

"REF": true,

"SYN": true,

"ASP": true,

"VLD": true,

"G5A": true,

"G5": true,

"HD": true,

"GNO": true,

"KGPhase1": true,

"KGPilot123": true,

"KGPROD": true,

"OTHERKG": true,

"PH3": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

$

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

"RSPOS": 26965148,

"RV": true,

"GMAF": 0.2395,

"dbSNPBuildID": 86,

"SSR": 0,

"SAO": 0,

"VP": "05030000030507051f000100",

"GENEINFO": "MRPL39:54148",

"WGT": 1,

"VC": "SNV",

"S3D": true,

"SLO": true,

"REF": true,

"SYN": true,

"ASP": true,

"VLD": true,

"G5A": true,

"G5": true,

"HD": true,

"GNO": true,

"KGPhase1": true,

"KGPilot123": true,

"KGPROD": true,

"OTHERKG": true,

"PH3": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

$

dbSNP:

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO INFO.dbSNPBuildID INFO.SSR INFO.SCS INFO.CLN INFO.SAO \_id

1 215848808 rs116645811 G A . . . . . . . . .

1 215848808 rs116645811 G T . . . . . . . . .

1 215848808 rs116645811 G G . . . . . . . . .

1 215848808 rs116645811 G C . . . . . . . . .

1 215848808 rs116645811 C A . . . . . . . . .

...

$

ESP:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/ESP/build37/ESP6500SI\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": "PASS",

"INFO": {

"DBSNP": [

"dbSNP\_86"

],

"EA\_AC": [

"7111",

"1489"

],

"AA\_AC": [

"3307",

"1099"

],

"TAC": [

"10418",

"2588"

],

"MAF": [

"17.314",

"24.9433",

"19.8985"

],

"GTS": [

"AA",

"AG",

"GG"

],

"EA\_GTC": [

"2954",

"1203",

"143"

],

"AA\_GTC": [

"1229",

"849",

"125"

],

"GTC": [

"4183",

"2052",

"268"

],

"DP": 75,

"GL": [

"MRPL39"

],

"CP": 1.0,

"CG": 3.0,

"AA": "A",

"CA": [

"."

],

"EXOME\_CHIP": [

"no"

],

"GWAS\_PUBMED": [

"."

],

"GM": [

"NM\_017446.3",

"NM\_080794.3"

],

"FG": [

"coding-synonymous",

"coding-synonymous"

],

"AAC": [

".",

"."

],

"PP": [

"299/339",

"299/354"

],

"CDP": [

"897",

"897"

],

"GS": [

".",

"."

],

"PH": [

".",

"."

]

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

$

HapMap:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/hapmap/2010-08\_phaseII+III/allele\_freqs\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"rsNumber": "rs1135638",

"chrom": "chr21",

"pos": 25887019,

"strand": "+",

"build": "ncbi\_b36",

"refallele": "G",

"otherallele": "A",

"\_type": "variant",

"\_landmark": "21",

"\_minBP": 26965148,

"\_maxBP": 26965148,

"\_strand": "+",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_id": "rs1135638",

"CEU": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:CEPH-60-trios:4",

"QC\_code": "QC+",

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/hapmap/2010-08\_phaseII+III/allele\_freqs\_GRCh37.tsv.bgz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"rsNumber": "rs1135638",

"chrom": "chr21",

"pos": 25887019,

"strand": "+",

"build": "ncbi\_b36",

"refallele": "G",

"otherallele": "A",

"\_type": "variant",

"\_landmark": "21",

"\_minBP": 26965148,

"\_maxBP": 26965148,

"\_strand": "+",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_id": "rs1135638",

"CEU": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:CEPH-60-trios:4",

"QC\_code": "QC+",

"refallele\_freq": 0.177,

"refallele\_count": 40,

"otherallele\_freq": 0.823,

"otherallele\_count": 186,

"totalcount": 226

},

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:US\_African-30-trios:4",

"QC\_code": "QC+",

"refallele\_freq": 0.277,

"refallele\_count": 31,

"otherallele\_freq": 0.723,

"otherallele\_count": 81,

"totalcount": 112

},

"CHD": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:US\_Chinese:4",

"QC\_code": "QC+",

"refallele\_freq": 0.289,

"refallele\_count": 63,

"otherallele\_freq": 0.711,

"otherallele\_count": 155,

"totalcount": 218

},

"GIH": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:US\_Gujarati:4",

"QC\_code": "QC+",

"refallele\_freq": 0.49,

"refallele\_count": 97,

"otherallele\_freq": 0.51,

"otherallele\_count": 101,

"totalcount": 198

},

"MEX": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:US\_Mexican-30-trios:4",

"QC\_code": "QC+",

"refallele\_freq": 0.237,

"refallele\_count": 27,

"otherallele\_freq": 0.763,

"otherallele\_count": 87,

"totalcount": 114

},

"YRI": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:Yoruba-60-trios:4",

"QC\_code": "QC+",

"refallele\_freq": 0.269,

"refallele\_count": 79,

"otherallele\_freq": 0.731,

"otherallele\_count": 215,

"totalcount": 294

}

}

$

"CHB": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:Han\_Chinese:4",

"QC\_code": "QC+",

"refallele\_freq": 0.278,

"refallele\_count": 74,

"otherallele\_freq": 0.722,

"otherallele\_count": 192,

"totalcount": 266

},

"TSI": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:Italian:4",

"QC\_code": "QC+",

"refallele\_freq": 0.201,

"refallele\_count": 41,

"otherallele\_freq": 0.799,

"otherallele\_count": 163,

"totalcount": 204

},

"JPT": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:Japanese:4",

"QC\_code": "QC+",

"refallele\_freq": 0.339,

"refallele\_count": 76,

"otherallele\_freq": 0.661,

"otherallele\_count": 148,

"totalcount": 224

},

"LWK": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:Luhya\_Kenyan:4",

"QC\_code": "QC+",

"refallele\_freq": 0.323,

"refallele\_count": 71,

"otherallele\_freq": 0.677,

"otherallele\_count": 149,

"totalcount": 220

},

"MKK": {

"center": "sanger",

"protLSID": "urn:LSID:illumina.hapmap.org:Protocol:Human\_1M\_BeadChip:3",

"assayLSID": "urn:LSID:sanger.hapmap.org:Assay:H1Mrs1135638:3",

"panelLSID": "urn:lsid:dcc.hapmap.org:Panel:Maasai\_Kenyan-60-trios:4",

"QC\_code": "QC+",

"refallele\_freq": 0.163,

"refallele\_count": 51,

"otherallele\_freq": 0.837,

"otherallele\_count": 261,

"totalcount": 312

},

"ASW": {

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/hapmap/2010-08\_phaseII+III/allele\_freqs\_GRCh37.tsv.bgz | bior\_drill -p CEU.refallele\_freq -p CEU.otherallele\_freq -p YRI.refallele\_freq -p YRI.otherallele\_freq -p JPT.refallele\_count -p JPT.otherallele\_count -p JPT.totalcount -p CHB.refallele\_count -p CHB.otherallele\_count -p CHB.totalcount | cut --complement -f 9

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO CEU.refallele\_freq CEU.otherallele\_freq YRI.refallele\_freq YRI.otherallele\_freq JPT.refallele\_count JPT.otherallele\_count JPT.totalcount CHB.refallele\_count CHB.otherallele\_count CHB.totalcount

1 215848808 rs116645811 G A . . . . . . . . . . . . .

1 215848808 rs116645811 G T . . . . . . . . . . . . .

1 215848808 rs116645811 G G . . . . . . . . . . . . .

1 215848808 rs116645811 G C . . . . . . . . . . . . .

1 215848808 rs116645811 C A . . . . . . . . . . . . .

1 215848808 rs116645811 C T . . . . . . . . . . . . .

1 215848808 rs116645811 C G . . . . . . . . . . . . .

1 215848808 rs116645811 C C . . . . . . . . . . . . .

1 215848808 rs116645811 A A . . . . . . . . . . . . .

1 215848808 rs116645811 A T . . . . . . . . . . . . .

1 215848808 rs116645811 A G . . . . . . . . . . . . .

1 215848808 rs116645811 A C . . . . . . . . . . . . .

1 215848808 rs116645811 T A . . . . . . . . . . . . .

1 215848808 rs116645811 T T . . . . . . . . . . . . .

1 215848808 rs116645811 T G . . . . . . . . . . . . .

1 215848808 rs116645811 T C . . . . . . . . . . . . .

21 26965148 rs1135638 G A . . . 0.177 0.823 0.269 0.731 76 148 224 74 192 266

21 26965172 rs010576 T C . . . . . . . . . . . . .

21 26965205 rs1057885 T C . . . 0.154 0.846 0.238 0.762 30 56 86 26 58 84

21 26976144 rs116331755 A G . . . . . . . . . . . . .

21 26976222 rs7278168 C T . . . 1.0 0 0.739 0.261 76 10 86 79 11 90

21 26976237 rs7278284 C T . . . . . . . . . . . . .

21 26978790 rs75377686 T C . . . . . . . . . . . . .

21 26978950 rs3989369 A G . . . 0.035 0.965 0.265 0.735 2 224 226 10 264 274

...

1000 Genomes:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/1000\_genomes/20110521/ALL.wgs.phase1\_release\_v3.20101123.snps\_indels\_sv.sites\_GRCh37.tsv.gz | bior\_pretty\_print -r 17

# COLUMN NAME COLUMN VALUE

- ----------- ------------

1 CHROM 21

2 POS 26965148

3 ID rs1135638

4 REF G

5 ALT A

6 QUAL .

7 FILTER .

8 INFO .

9 VCF2VariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": ".",

"FILTER": ".",

"INFO": {

".": true

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

10 SameVariantPipe {

"CHROM": "21",

"POS": "26965148",

"ID": "rs1135638",

"REF": "G",

"ALT": "A",

"QUAL": "100",

"FILTER": "PASS",

"INFO": {

"AVGPOST": 1.0,

"RSQ": 0.9999,

"SNPSOURCE": [

"LOWCOV",

"EXOME"

],

"AN": 2184,

"LDAF": 0.7609,

"VT": "SNP",

"AA": "A",

"AC": [

1661

],

"ERATE": 2.0E-4,

"THETA": 3.0E-4,

"AF": 0.76,

"ASN\_AF": 0.71,

"AMR\_AF": 0.8,

"AFR\_AF": 0.72,

"EUR\_AF": 0.8

},

"\_id": "rs1135638",

"\_type": "variant",

"\_landmark": "21",

"\_refAllele": "G",

"\_altAlleles": [

"A"

],

"\_minBP": 26965148,

"\_maxBP": 26965148

}

$

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_same\_variant -d $bior/1000\_genomes/20110521/ALL.wgs.phase1\_release\_v3.20101123.snps\_indels\_sv.sites\_GRCh37.tsv.gz | bior\_drill -p INFO.AF -p INFO.EUR\_AF -p INFO.ASN\_AF -p INFO.AFR\_AF -p INFO.AMR\_AF | cut -f 9 --complement

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO INFO.AF INFO.EUR\_AF INFO.ASN\_AF INFO.AFR\_AF INFO.AMR\_AF

1 215848808 rs116645811 G A . . . . . . . .

...

1 215848808 rs116645811 T C . . . . . . . .

21 26965148 rs1135638 G A . . . 0.76 0.8 0.71 0.72 0.8

21 26965172 rs010576 T C . . . 0.01 . . 0.04 0.01

21 26965205 rs1057885 T C . . . 0.76 0.8 0.71 0.72 0.8

21 26976144 rs116331755 A G . . . 9.0E-4 . . 0.0041 .

21 26976222 rs7278168 C T . . . 0.11 0.0026 0.14 0.24 0.14

21 26976237 rs7278284 C T . . . 0.12 0.0026 0.14 0.27 0.14

21 26978790 rs75377686 T C . . . 0.01 . . 0.04 0.01

21 26978950 rs3989369 A G . . . 0.91 0.96 0.97 0.75 0.94

...

$

## Putting it All Together Building an AF Pipeline

TREAT]$ cat treatAF.bior

export bior=$bior/

cat /dev/stdin | bior\_vcf\_to\_tjson \

| bior\_same\_variant -d $bior/dbSNP/137/00-All\_GRCh37.tsv.bgz \

| bior\_drill -p \_id -p INFO.dbSNPBuildID -p INFO.SSR -p INFO.SCS -p INFO.CLN -p INFO.SAO \

| bior\_same\_variant -c -7 -d $bior/cosmic/v63/CosmicCompleteExport\_GRCh37.tsv.bgz \

| bior\_drill -p Mutation\_ID -p Mutation\_CDS -p Mutation\_AA -p Mutation\_GRCh37\_strand \

| bior\_same\_variant -c -11 -d $bior/1000\_genomes/20110521/ALL.wgs.phase1\_release\_v3.20101123.snps\_indels\_sv.sites\_GRCh37.tsv.gz \

| bior\_drill -p INFO.ASN\_AF -p INFO.AMR\_AF -p INFO.AFR\_AF -p INFO.EUR\_AF \

| bior\_same\_variant -c -15 -d $bior/BGI/hg19/LuCAMP\_200exomeFinal.maf\_GRCh37.tsv.bgz \

| bior\_drill -p estimated\_minor\_allele\_freq \

| bior\_same\_variant -c -16 -d $bior/ESP/build37/ESP6500SI\_GRCh37.tsv.bgz \

| bior\_drill -p INFO.MAF[0] -p INFO.MAF[1] -p INFO.MAF[2] \

| bior\_same\_variant -c -19 -d $bior/hapmap/2010-08\_phaseII+III/allele\_freqs\_GRCh37.tsv.bgz \

| bior\_drill -p CEU.refallele\_freq -p CEU.otherallele\_freq \

| ./removeJSON.pl

TREAT]$

# Extracting Data with JSONPaths (bior\_drill)

To extract data that is embedded in a JSON document as an array you can use drill.path[1] to get the first element in the array, drill.path[1].field to get a field in a json array or drill.path[\*] to get all elements in the array.

# Command Line Tools

Want SNPeff

cat example.vcf | bior\_snpeff | bior\_drill –p Effect –p Effect\_impact –p Functional\_class –p Amino\_acid\_change | cut -­‐f 9 -­‐ -­‐complement > example.w\_genes.vcf

Want SIFT & PolyPhen

cat example.vcf | bior\_vep | bior\_drill –p Consequence –p SIFT –p PolyPhen –p SIFT\_Score –p PolyPhen\_Score | cut -­‐f 9 -­‐ -­‐complement > example.w\_genes.vcf

TREAT]$ cat treatTOOLS.bior

bior\_vep < /dev/stdin \

| bior\_drill -p Allele -p Gene -p Feature -p Feature\_type -p Consequence -p cDNA\_position -p CDS\_position -p Protein\_position -p Amino\_acids -p Codons -p HGNC -p SIFT\_TERM -p SIFT\_Score -p PolyPhen\_TERM -p PolyPhen\_Score \

| bior\_snpeff \

| bior\_drill -p Effect -p Effect\_impact -p Functional\_class -p Codon\_change -p Amino\_acid\_change -p Gene\_name -p Gene\_bioType -p Coding -p Transcript -p Exon

TREAT]$

# Mixing In Scripts and Languages

## To find all overlapping genes that are not the same gene:

zcat $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_overlap -d $bior/v1/NCBIGene/GRCh37\_p10/genes.tsv.bgz | perl -e 'while (<>) {chomp; @a=split(/\t/,$\_); if($a[3] ne $a[4]){print $a[3]."\t".$a[4]."\n";} }' | bior\_drill -c -2 -p gene | bior\_drill -c -2 -p gene |  less

# Common Problems

## Handling VCF Files with VERY large headers

All BioR commands store the header in memory. This is done because commands like bior\_vcf\_to\_tjson use the header to understand the structure of the data lines and parse the lines into JSON more intelligently (e.g. identify numbers instead of strings, identify arrays, ect.). In production, we have noticed that some headers are extreamly large (multiple megabytes). When a user runs BioR, the header is expanded into objects in memory for each BioR command. This can lead to BioR slowing to a crawl when the ram on the machine is exceeded.  Internally what happens is that the header is chopped off and stored in memory, then each row streams through the system as an array of strings.  The data rows are not that large, but the metadata in the header may get copied many times in memory as transformations are done on the data.  The best workaround for this problem is to use grep to cut off all excess header lines (e.g. lines that are not descriptive) then push the BioR output on to the file. Recombine the header if needed.

 e.g.

zcat example.vcf.gz | head -n 10000 | grep -v "##" > mylongheader.vcf

zcat example.vcf.gz | bior\_vcf\_to\_tjson | bior\_mycommands >> mylongheader.vcf

## Large Memory Requirements

Sometimes users complain about large memory requrirements from BioR – especially SNPEff. SNPEff, when run in production requires 4Gb of Ram. BioR will align large insertions and deletions prior to sending them to SNPEff using the same exact method used in SNPEff. When processing these large variants, both BioR and SNPEff can crash. The current work-around for dealing with large variants is to pre-screen them and filter them out to another file prior to annotating with SNPEff. Hopefully the BioR team will be able to collect better statistics and not align large variants in the future.

## BioR exits with some error I don’t understand

Rerun the same exact command with logging enabled (-l) and submit both the input file, and the results of the log to the BioR team. We will try to help you ASAP.

# Creating Catalogs

## Indexing your Samples

Lets say you want to get variants in your sample that overlap a gene.  One way to do this is to stream the variants e.g:

> cat example.vcf | head

##fileformat=VCFv4.0

#CHROM POS ID REF ALT QUAL FILTER INFO

21 26960070 rs116645811 G A . . .

21 26965148 rs1135638 G A . . .

21 26965172 rs010576 T C . . .

21 26965205 rs1057885 T C . . .

21 26976144 rs116331755 A G . . .

21 26976222 rs7278168 C T . . .

21 26976237 rs7278284 C T . . .

21 26978790 rs75377686 T C . . .

>cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | grep "\"gene\":\"PANX2\""

22 50616005 rs35195493 C G . . . {"CHROM":"22","POS":"50616005","ID":"rs35195493","REF":"C","ALT":"G","QUAL":".","FILTER":".","INFO":{".":true},"\_id":"rs35195493","\_type":"variant","\_landmark":"22","\_refAllele":"C","\_altAlleles":["G"],"\_minBP":50616005,"\_maxBP":50616005} {"\_type":"gene","\_landmark":"22","\_strand":"+","\_minBP":50609160,"\_maxBP":50618724,"gene":"PANX2","gene\_synonym":"hPANX2; PX2","note":"pannexin 2; Derived by automated computational analysis using gene prediction method: BestRefseq.","GeneID":"56666","HGNC":"8600","HPRD":"09760","MIM":"608421"}

22 50616806 rs5771206 A G . . . {"CHROM":"22","POS":"50616806","ID":"rs5771206","REF":"A","ALT":"G","QUAL":".","FILTER":".","INFO":{".":true},"\_id":"rs5771206","\_type":"variant","\_landmark":"22","\_refAllele":"A","\_altAlleles":["G"],"\_minBP":50616806,"\_maxBP":50616806} {"\_type":"gene","\_landmark":"22","\_strand":"+","\_minBP":50609160,"\_maxBP":50618724,"gene":"PANX2","gene\_synonym":"hPANX2; PX2","note":"pannexin 2; Derived by automated computational analysis using gene prediction method: BestRefseq.","GeneID":"56666","HGNC":"8600","HPRD":"09760","MIM":"608421"}

$

If you just want variants that overlap any gene, you can always do something like:

>zcat $bior/NCBIGene/

GRCh37\_p10/genes.tsv.bgz | bior\_overlap -d ./example.tsv.gz |

grep -v "{}" | less

That works fine for a single gene, but what if you are starting with a list of genes?  e.g.

>cat mygenes.txt

MRPL39

PANX2

BRCA1

...

In this case you may want to use an index on your data.  To create the index, do something like:

>cat example.vcf | bior\_vcf\_to\_tjson | grep "^#" | cut -f 1,2,9 |

bior\_drill -k -p \_maxBP > example.tsv

>sort -k1,1 -k2,2n example.tsv

>bgzip example.tsv

>tabix example.tsv.gz

>tabix -s 1 -b 2 -e 3 example.tsv.gz

Now use lookup to get the gene locations, and overlap to overlap those locations with your data:

>cat mygenes.txt | bior\_lookup -p gene

-d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz |

bior\_overlap -d ./example.tsv.gz | bior\_pretty\_print

You can now use bior\_same\_variant to annotate variants that overlap your genes.

## Creating Custom Catalogs

One of the most powerful things about BioR is that users can publish their own catalogs and integrate new data into the system. They can also share these catalogs with others making the system extensible and much more powerful than a system where the catalogs must all be maintained by a single annotation team.

### The Publication Process

Publishing a catalog requires (1) a parser that understands arbitrarily formatted file formats, and (2) indexing tools.  Parsers convert arbitrary data representations into JSON with a set of 'golden identifiers' the BioR system understands.  Example 'golden identifiers’ include \_landmark, \_minBP, and \_maxBP.  'Golden identifiers' are always prefixed with an underscore ('\_') and must be absolutely consistent at both in terms of syntax and semantics.  For example, \_minBP uses the standard 1-based coordnate system (e.g. NCBI/Blast) not interbase coordinates (<http://gmod.org/wiki/Introduction_to_Chado#Interbase_Coordinates>), and \_strand is represented as '+', '-', or '.' and NOT 'complement' as in the gbs files from NCBI.  One of the functions of a parser, is to convert from arbitrary file formats into JSON, the other is to extract the 'golden identifiers' and place them in the JSON.  'Golden identifiers'  are created so that BioR programs (e.g. bior\_overlap.sh) can work on the information regardless of the source file format (e.g. VCF, GFF, GBS, XML, RelationalDB, Tab-Delimited, ...).

As they become availible, parsers, will be exposed to users as command line tools.  For example, bior\_vcf\_to\_variants.sh is a parser that converts vcf to BioR JSON.

In summary, to make a custom catalog, you need:

1.       Columns 1-3 bed-like (chr            start       stop) [1-based]

2.       The 4th column is a series of key-value pairs enclosed by quotes and brackets

3.       The 4 column contains “Golden identifiers” [ \_landmark, \_minBP, and \_maxBP ]

Once this is created, use bgzip & tabix to compress and index it for genomic search. For those samples that do NOT have a genomic position, use the following values (bior\_create\_catalog will do this for you).

|  |  |
| --- | --- |
| **Golden Identifier** | **Default Value** |
| \_landmark | UNKNOWN ( a period ‘.’ is also ok) |
| \_minBP | 0 |
| \_maxBP | 0 |

Zero is important because it has to be an integer and must be greater than zero.  The JSON does not have to have the golden attribute if you won't search on it.

### Parsing and Converting the Data

If a parser for the file format is available (e.g. bior\_vcf\_to\_tjson, bior\_bed\_to\_tjson, ect.) publishing a custom catalog is extremely easy.  Using the standard BioR tools, a publication pipeline can be constructed rapidly.  For example:

zcat 00-All.vcf.gz | bior\_vcf\_to\_tjson.sh | cut -f 9 | bior\_drill.sh -k -p \_landmark -p \_minBP -p \_maxBP > dbSNP.tsv

This pipeline streams the original VCF file past the parser (bior\_vcf\_to\_tjson), removes the content of the original VCF (cut -f 9) - this is ok, as all of this information is duplicated in the JSON format, drill out the key attributes (bior\_drill.sh) so that they can be indexed, and then output to a raw data file (dbSNP.tsv).  The raw output file should look like this:

$ head dbSNP.tsv

1    10144    10145    {"CHROM":"1","POS":"10144","ID":"rs144773400","REF":"TA","ALT":"T","QUAL":".","FILTER":".","INFO":{"RSPOS":10145,"dbSNPBuildID":134,"SSR":0,"SAO":0,"VP":"050000000005000002000200","WGT":1,"VC":"DIV","ASP":true,"OTHERKG":true},"\_id":"rs144773400","\_type":"variant","\_landmark":"1","\_refAllele":"TA","\_altAlleles":["T"],"\_minBP":10144,"\_maxBP":10145}

1    10177    10177    {"CHROM":"1","POS":"10177","ID":"rs201752861","REF":"A","ALT":"C","QUAL":".","FILTER":".","INFO":{"RSPOS":10177,"dbSNPBuildID":137,"SSR":0,"SAO":0,"VP":"050000000005000002000100","WGT":1,"VC":"SNV","ASP":true,"OTHERKG":true},"\_id":"rs201752861","\_type":"variant","\_landmark":"1","\_refAllele":"A","\_altAlleles":["C"],"\_minBP":10177,"\_maxBP":10177}

...

### Indexing the Data for Coordinate Based Search

For positional search, BioR supports indexing using Tabix.  Tabix/bgzip should be installed in the RCF environment.  First, compress the raw input.  Assuming it is sorted:

$ bgzip dbSNP.tsv

Then run the tabix command:

$ tabix -s 1 -b 2 -e 3 dbSNP.tsv.gz &

That's it! you can now use your custom catalog as a database in BioR commands (e.g. bior\_overlap.sh -d /path/to/your/database.tsv.gz).

### Hints on Creating Indexes on Custom Catalogs

In addition to coordinate based search, users may also want to search a custom catalog based on IDs. The process is exactly the same as in indexing a catalog described earlier in this document, but there are some gotcha’s that users need to be aware of.

1. The catalog structure will not automatically join data. This can be frustrating as the data provider may not give the data to you in a desirable form (e.g. you may want to know everything the data provider knows about a gene, but they may have their data organized by variant or drug) so you will have to ‘flip’ the data around so that all information about a gene can be provided to users of your catalog. The BioR team has done this many times, and for Java programmers, there is a robust library (BioR-Catalog) and examples to help in the publication of new-complex catalogs.
2. The BioR indexer command currently does not tolerate duplicate keys, so while duplicate keys can be in the data itself, you can’t index on those keys. Running bior\_index with logging enabled will help to ensure the keys you would like to index on are valid. To index multiple ways simultaneously, multiple catalogs need to be created
3. Regardless of what tools are used to construct the JSON column, it must validate as proper JSON. Use jslint to validate: <http://jsonlint.com/>
4. JSON should not contain fields that are empty. While adding period “.” As the value for a given key will work, it wastes space and consumes additional CPU resources so is not recommended.

## Use BioR to map SNP on rsID and find overlapping genes.

Say we obtained a simple tab-delimited file that is not in VCF format, but we still want to obtain an annotation. The following file’s header for this is:  rsid without the “rs”, chrom, position, and 0/1 representing presence or absence in our study.  There are over 5 million in this file. The goal is to show how the first 100 or 1000 of these map to various genes

$ zcat b132\_SNPChrPosOnRef\_37\_1.bcp.gz | more

3       13      32446841        0

4       13      32447221        0

5       7       91839109        1

6       7       91747130        1

7       7       91779556        1

8       7       92408328        0

9       7       92373453        0

10      7       92383887        0

11      7       11364200        0

12      7       11337163        0

13      7       11387690        0

14      7       11380841        0

15      7       11602931        1

16      7       11602898        1

17      7       11583798        1

18      7       11597474        1

19      7       11597155        1

20      7       11597104        1

21      7       11596933        1

22      7       11596501        1

…

Try playing around with something like this to get started: (it may not be exactly what you want but we can work on that)

NCBIGene:

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_pretty\_print

#   COLUMN NAME      COLUMN VALUE

-   -----------      ------------

1   CHROM            1

2   POS              215848808

3   ID               rs116645811

4   REF              G

5   ALT              A

6   QUAL             .

7   FILTER           .

8   INFO             .

9   VCF2VariantPipe  {

                       "CHROM": "1",

                       "POS": "215848808",

                       "ID": "rs116645811",

                       "REF": "G",

                       "ALT": "A",

                       "QUAL": ".",

                       "FILTER": ".",

                       "INFO": {

                         ".": true

                       },

                       "\_id": "rs116645811",

                       "\_type": "variant",

                       "\_landmark": "1",

                       "\_refAllele": "G",

                       "\_altAlleles": [

                         "A"

                       ],

                       "\_minBP": 215848808,

                       "\_maxBP": 215848808

                     }

10  OverlapPipe      {

                       "\_type": "gene",

                       "\_landmark": "1",

                       "\_strand": "-",

                       "\_minBP": 215796236,

                       "\_maxBP": 216596738,

                       "gene": "USH2A",

                       "gene\_synonym": "dJ1111A8.1; RP39; US2; USH2",

                       "note": "Usher syndrome 2A (autosomal recessive, mild); Derived by automated computational analysis using gene prediction method: BestRefseq.",

                       "GeneID": "7399",

                       "HGNC": "12601",

                       "HPRD": "02042",

                       "MIM": "608400"

                     }

$

$ cat example.vcf | bior\_vcf\_to\_tjson | bior\_overlap -d $bior/NCBIGene/GRCh37\_p10/genes.tsv.bgz | bior\_drill -p GeneID -p gene | cut -f 9 --complement

##fileformat=VCFv4.0

#CHROM              POS         ID            REF         ALT         QUAL     FILTER   INFO       gene       GeneID

1              215848808           rs116645811       G             A             .               .               .               USH2A   7399

...

21           26965148             rs1135638            G             A             .               .               .               MRPL39                54148

21           26965172             rs010576              T              C             .               .               .               MRPL39                54148

21           26965205             rs1057885            T              C             .               .               .               MRPL39                54148

21           26976144             rs116331755       A             G             .               .               .               MRPL39                54148

21           26976222             rs7278168            C             T              .               .               .               MRPL39                54148

21           26976237             rs7278284            C             T              .               .               .               MRPL39                54148

21           26978790             rs75377686          T              C             .               .               .               MRPL39                54148

21           26978950             rs3989369            A             G             .               .               .               MRPL39                54148

21           26979752             rs61735760          C             T              .               .               .               MRPL39                54148

21           34022588             rs115683257       C             A             .               .               .               SYNJ1     8867

21           34029195             rs114053718       A             G             .               .               .               SYNJ1     8867

21           34058146             rs114942253       C             T              .               .               .               SYNJ1     8867

21           34059352             rs2254562            T              C             .               .               .               SYNJ1     8867

...

$

Now, we want to find "Approved\_Symbol", "Entrez\_Gene\_ID", "Ensembl\_Gene\_ID", "UniProt\_ID", ...

We can use the BioR lookup command:

First, we don't know the catalog Structure of HGNC, here is a way to look at the structure of a catalog:

## Case Study: Creating a Report that Maps rsIDs to Genes.

$ zcat $bior/hgnc/2012\_08\_12/hgnc\_GRCh37.tsv.bgz | bior\_pretty\_print

#  COLUMN NAME  COLUMN VALUE

-  -----------  ------------

1  UNKNOWN\_1    .

2  #UNKNOWN\_2   0

3  #UNKNOWN\_3   0

4  #UNKNOWN\_4   {

                  "HGNC\_ID": "HGNC:5",

                  "Approved\_Symbol": "A1BG",

                  "Approved\_Name": "alpha-1-B glycoprotein",

                  "Status": "Approved",

                  "Locus\_Type": "gene with protein product",

                  "Locus\_Group": "protein-coding gene",

                  "Previous\_Symbols": [],

                  "Previous\_Names": [],

                  "Synonyms": [],

                  "Name\_Synonyms": [],

                  "Chromosome": "19q",

                  "Date\_Approved": "1989-06-30",

                  "Date\_Modified": "2010-07-08",

                  "Accession\_Numbers": [],

                  "Enzyme\_IDs": [],

                  "Entrez\_Gene\_ID": "1",

                  "Ensembl\_Gene\_ID": "ENSG00000121410",

                  "Specialist\_Database\_Links": "\u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003ca href\u003d\"[http://merops.sanger.ac.uk/cgi-bin/merops.cgi?id\u003dI43.950\](http://merops.sanger.ac.uk/cgi-bin/merops.cgi?id%5Cu003dI43.950%5C)"\u003eMEROPS\u003c/a\u003e\u003c!--,--\u003e \u003ca href\u003d\"[http://www.sanger.ac.uk/perl/genetics/CGP/cosmic?action\u003dgene\u0026amp;ln\u003dA1BG\](http://www.sanger.ac.uk/perl/genetics/CGP/cosmic?action%5Cu003dgene%5Cu0026amp;ln%5Cu003dA1BG%5C)"\u003eCOSMIC\u003c/a\u003e\u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e \u003c!--,--\u003e ",

                  "Specialist\_Database\_IDs": [

                    "",

                    "",

                    "",

                    "",

                    "",

                    "",

                    "",

                    "",

                    "",

                    "I43.950",

                    "A1BG",

                    "",

                    "",

                    "",

                    "",

                    "",

                    ""

],

                  "Pubmed\_IDs": [

                    "2591067"

                  ],

                  "RefSeq\_IDs": [

                    "NM\_130786"

                  ],

                  "Record\_Type": "Standard",

                  "Primary\_IDs": [],

                  "Secondary\_IDs": [],

                  "CCDS\_IDs": [

                    "CCDS12976.1"

                  ],

                  "VEGA\_IDs": [],

                  "mapped\_GDB\_ID": "GDB:119638",

                  "mapped\_Entrez\_Gene\_ID": "1",

                  "mapped\_OMIM\_ID": "138670",

                  "mapped\_RefSeq": "NM\_130786",

                  "UniProt\_ID": "P04217",

                  "mapped\_Ensembl\_ID": "ENSG00000121410",

                  "UCSC\_ID": "uc002qsd.4",

                  "mapped\_Mouse\_Genome\_Database\_ID": "MGI:2152878",

                  "mapped\_Rat\_Genome\_Database\_ID": "RGD:69417"

                }

$

To join the information in this catalog, to the information that we have collected in the gene table, we need to tell bior what field in the HGNC table matches the LAST column in our sample data + annotation.  In this case, we will join on approved symbol (note: if you ever get an error with doing a lookup, you may need an index file - look into the bior\_index command or contact the bior team for help).

grep "^22.\*rs3721" gene\_snp.db132.gene.coding.dat | more

22 7332 UBE2L3 rs372150 29047

22 150223 YDJC rs372150 23030

22 164592 CCDC116 rs372150 15754

22 23753 SDF2L1 rs372150 8782

22 23753 SDF2L1 rs372108 45008

22 23759 PPIL2 rs372150 -12903

22 23759 PPIL2 rs372108 0

22 29799 YPEL1 rs372150 -44455

22 29799 YPEL1 rs372108 -8229

22 83746 L3MBTL2 rs3721 0

22 150356 CHADL rs3721 0

22 5905 RANGAP1 rs3721 -14542

1. <http://samtools.sourceforge.net/tabix.shtml> [↑](#footnote-ref-1)
2. http://genome.cshlp.org/content/12/6/996.full [↑](#footnote-ref-2)
3. Quinlan AR and Hall IM, 2010. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 26, 6, pp. 841–842. [↑](#footnote-ref-3)
4. <http://bioinformatics.oxfordjournals.org/content/27/5/718.abstract> [↑](#footnote-ref-4)