
LIRICAL Documentation

Release 0.9

Peter Robinson

May 14, 2021

Contents:

1	LIRICAL	1
1.1	Setting up LIRICAL	1
1.2	Running LIRICAL	3
1.3	LIRICAL Output	13
1.4	LIRICAL: How does it work?	16
1.5	LIRICAL Options	19
1.6	LIRICAL Advanced Options	20
1.7	Tutorial	21

This application performs phenotype-driven prioritization of candidate diseases and genes in the setting of genomic diagnostics (exome or genome) in which the phenotypic abnormalities are described as [Human Phenotype Ontology \(HPO\)](#) terms.

1.1 Setting up LIRICAL

LIRICAL is a desktop Java application that requires several external files to run. This document details how to download these files and prepare to run LIRICAL. LIRICAL requires Exomiser to be installed as a library before it can be compiled and built.

1.1.1 Prerequisites

LIRICAL was written with Java version 8 but will compile under Java 11. If you want to build LIRICAL from source, then the build process described below requires [Git](#) and [maven](#).

1.1.2 Installation of Exomiser as a Java library

LIRICAL relies on some classes from the Exomiser. To build LIRICAL, we need to install the Exomiser code base locally. Note that the version of Exomiser must match the version indicated in LIRICAL's pom file (currently 12.1.0). To check this, search for the following line in the `pom.xml` file:

```
<exomiser.version>12.1.0</exomiser.version>
```

To do so, we clone the code and change into the Exomiser directory.

```
$ git clone https://github.com/exomiser/Exomiser.git
$ cd Exomiser
```

Now, we ensure that we are using the correct branch of Exomiser (release-12.0.0).

```
$ git checkout release-12.1.0
Switched to branch 'release-12.1.0'
Your branch is up to date with 'origin/release-12.1.0'.
$ git branch
  development
  master
* release-12.1.0
```

Finally, we use the maven system to install the Exomiser library locally so that it can be used by LIRICAL.

```
$ mvn install
```

This command will install the library in the `.m2` directory located in your home directory. If you like, explore `.m2/repository/org/monarchinitiative/exomiser/` to see how maven structures the repository. Occasionally, we have seen that an error occurs in the installation under some flavors of linux, which appears to be due to concurrency issues engendered by the unit tests. If you observe this error, try to install Exomiser without tests.

```
$ mvn install -DskipTests=true
```

1.1.3 Installation

Go the GitHub page of [LIRICAL](#), and clone or download the project. Build the executable from source with maven, and then test the build.

```
$ git clone https://github.com/TheJacksonLaboratory/LIRICAL.git
$ cd LIRICAL
$ mvn package
$ java -jar target/LIRICAL.jar
$ Usage: <main class> [options] [command] [command options]
Options:
  -h, --help
      display this help message
  (...)

```

LIRICAL requires [maven](#) version 3.5.3.

Prebuilt LIRICAL executable

Alternatively, go to the [Releases section](#) on the LIRICAL GitHub page and download the latest precompiled version of LIRICAL.

1.1.4 Exomiser database files

LIRICAL uses data files from the Exomiser. We recommend that always the latest version of these files be used. The data files are stored at the [Exomiser download site](#). You may need to scroll (right hand side) to see the subdirectory `latest`, which includes the current version of these files. Download either `1909_hg19.zip` (for the hg19/GRCh37 genome assembly) or “`1909_hg38.zip`” for the hg38/GRCh38 assembly). Of course, the datafile you use should match the assembly used to align and call the exome/genome data you want to analyze with LIRICAL. Unpack the file, e.g.,

```
$ unzip 1909_hg19.zip
```

Remember the path, since it will be needed to run LIRICAL with exome/genome data. We will use the argument:

```
-e /some/path/1909_hg19
```

where 1909_hg19 is the directory that is created by unpacking the archive file. The directory should contain 10 files including:

- 1909_hg19_genome.h2.db
- 1909_hg19_transcripts_ensembl.ser
- 1909_hg19_transcripts_refseq.ser
- 1909_hg19_transcripts_ucsc.ser
- 1909_hg19_variants.mv.db

These files are used by LIRICAL to annotate the VCF file and support variant interpretation.

1.1.5 The download command

LIRICAL requires four additional files to run.

1. `hp.obo`. The main Human Phenotype Ontology file
2. `phenotype.hpoa` The main annotation file with all HPO disease models
3. `Homo_sapiens_gene_info.gz` A file from NCBI Entrez Gene with information about human genes
4. `mim2gene_medgen` A file from the NCBI medgen project with OMIM-derived links between genes and diseases

LIRICAL offers a convenience function to download all four files to a local directory. By default, LIRICAL will download all four files into a newly created subdirectory called `data` in the current working directory. You can change this default with the `-d` or `--data` options (If you change this, then you will need to pass the location of your directory to all other LIRICAL commands using the `-d` flag). Download the files automatically as follows.

```
$ java -jar LIRICAL.jar download
```

LIRICAL will not download the files if they are already present unless the `--overwrite` argument is passed. For instance, the following command would download the four files to a directory called `datafiles` and would overwrite any previously downloaded files.

```
$ java -jar LIRICAL.jar download -d datafiles --overwrite
```

If desired, you can download these files on your own but you need to place them all in the same directory to run LIRICAL.

1.2 Running LIRICAL

LIRICAL is a command-line Java tool that runs with Java version 8 or higher. LIRICAL can be run both with and without genomic data in form of a VCF file from genome, exome, or NGS gene-panel sequencing.

To get help, run LIRICAL with a command or with the option “-h”:

```
$ java -jar target/LIRICAL.jar -h
Usage: java -jar LIRICAL [-hV] [COMMAND]
Likelihood Ratio Interpretation of Clinical Abnormalities
-h, --help          Show this help message and exit.
```

(continues on next page)

(continued from previous page)

```

-V, --version    Print version information and exit.
Commands:
  download, D      Download files for LIRICAL
  phenopacket, P   Run LIRICAL from a Phenopacket
  yaml, Y          Run LIRICAL from YAML file

```

Run LIRICAL with a specific command with the “-h” option to get information about the command, e.g.,

```

$ java -jar target/LIRICAL.jar download -h
Usage: java -jar LIRICAL download [-hVw] [-d=<datadir>]
Download files for LIRICAL
  -d, --data=<datadir>    directory to download data (default: data)
  -h, --help              Show this help message and exit.
  -V, --version           Print version information and exit.
  -w, --overwrite         overwrite previously downloaded files (default: false)

```

LIRICAL has three main commands, `download`, `phenopacket`, and `yaml`. The `download` command needs to be run before anything else and downloads some files required for LIRICAL analysis. LIRICAL can then be run using a Phenopacket or a YAML-formatted file as input.

1.2.1 Running LIRICAL with a Phenopacket

LIRICAL can be run with clinical data (HPO terms) only or with clinical data and a VCF file representing the results of gene panel, exome, or genome sequencing. The preferred input format is [Phenopackets](#), an open standard for sharing disease and phenotype information. This is a new standard of the [Global Alliance for Genomics and Health](#) that links detailed phenotype descriptions with disease, patient, and genetic information.

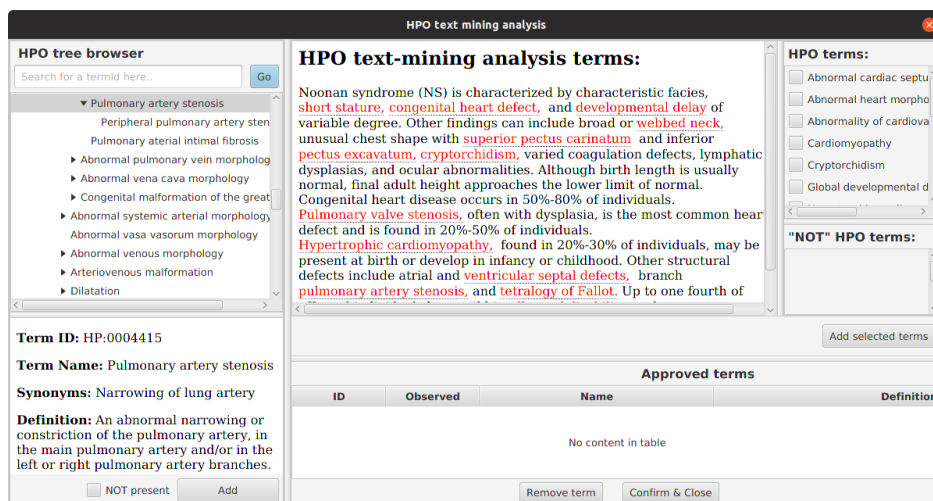


Fig. 1: For convenience, we provide a tool called [PhenopacketGenerator](#) that can be used to create a Phenopacket with a list of HPO terms and the path to a VCF file with which LIRICAL can be run.

Running LIRICAL with a Phenopacket file

LIRICAL can be run with clinical data (HPO terms) only or with clinical data and a VCF file representing the results of gene panel, exome, or genome sequencing. The preferred input format is [Phenopackets](#), an open standard for sharing disease and phenotype information. This is a new standard of the [Global Alliance for Genomics and Health](#) that

links detailed phenotype descriptions with disease, patient, and genetic information (The other allowed input format is YAML. See *Running LIRICAL with a YAML file (HPO and VCF data)*).

Preparing Phenopacket-formatted data

See the [Phenopackets](#) website for details on the format. LIRICAL expects the Phenopacket to be in JSON format. The following example shows a phenopacket representing an individual with [Pfeiffer syndrome](#).

```
{
  "subject": {
    "id": "example-1"
  },
  "phenotypicFeatures": [{
    "type": {
      "id": "HP:0000244",
      "label": "Turribrachycephaly"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }, {
    "type": {
      "id": "HP:0001363",
      "label": "Craniosynostosis"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }, {
    "type": {
      "id": "HP:0000453",
      "label": "Choanal atresia"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }, {
    "type": {
      "id": "HP:0000327",
      "label": "Hypoplasia of the maxilla"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }, {
    "type": {
      "id": "HP:0000238",
      "label": "Hydrocephalus"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }
}
```

(continues on next page)

(continued from previous page)

```
}
}},
"metaData": {
  "createdBy": "Peter R.",
  "resources": [{
    "id": "hp",
    "name": "human phenotype ontology",
    "namespacePrefix": "HP",
    "url": "http://purl.obolibrary.org/obo/hp.owl",
    "version": "2018-03-08",
    "iriPrefix": "http://purl.obolibrary.org/obo/HP_"
  }]
}
```

Running LIRICAL with clinical data

LIRICAL will perform phenotype-only analysis if the Phenopacket does not contain a `htsFiles` element. In this case, the only required argument is the phenopacket.

```
$ java -jar LIRICAL.jar phenopacket -p /path/to/example.json
```

LIRICAL Options for clinical/genomic analysis

The following options can be use to alter the default behavior of LIRICAL

```
-d, --data <directory>
```

By default, LIRICAL downloads several data files to a directory called `data` that it creates in the current working directory. If you download these files to some other directory, then you will need to indicate that path with this option.

```
-m, --mindiff <int>
```

By default, LIRICAL shows all differential diagnoses with a posterior probability of at least 1%, and at least 10 entries regardless of the posterior probability. If you want LIRICAL to show details about more differentials, set this option to the desired number.

```
-t, --threshold
```

This option controls the minimum post-test probability to show a differential diagnosis in HTML output. By default, LIRICAL shows all differentials with a posterior probability of 1% or greater.

```
-x, --prefix
```

The output file will be either `prefix.html` or `prefix.tsv`, whereby `prefix` can be set with this option (e.g., `-x example` would cause LIRICAL to output `example.html`). By default, the prefix is set to “lirical”.

```
-o, --output-directory
```

Directory into which to write output file(s).

```
--tsv
```

Use TSV instead of HTML output (Default: false).

```
--orpha
```

Use annotation data from [Orphanet](#).

Output

See *LIRICAL Output* for details on the HTML and TSV output files.

Running LIRICAL with a Phenopacket file (HPO and VCF data)

Preparing Phenopacket-formated data

The following example shows a phenopacket representing an individual with [Pfeiffer syndrome](#). The file is adapted from the phenopacket on *Running LIRICAL with a Phenopacket file*. We have removed several of the phenotypic features, and added an **HtsFiles** element that contains the path of the VCF file

(in our example, the path is `/example/path/Pfeiffer.vcf`, but obviously you need to adjust the path to a file located on your system).

```
{
  "subject": {
    "id": "example-1"
  },
  "phenotypicFeatures": [{
    "type": {
      "id": "HP:0000244",
      "label": "Turribrachycephaly"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }, {
    "type": {
      "id": "HP:0000238",
      "label": "Hydrocephalus"
    },
    "classOfOnset": {
      "id": "HP:0003577",
      "label": "Congenital onset"
    }
  }
],
  "htsFiles": [
    {
      "uri": "file://example/path/example.vcf",
      "description": "test",
      "htsFormat": "VCF",
      "genomeAssembly": "GRCh19",
      "individualToSampleIdentifiers": {
        "patient1": "NA12345"
      }
    }
  ],
  "metaData": {
```

(continues on next page)

(continued from previous page)

```

    "createdBy": "Peter R.",
    "resources": [{
      "id": "hp",
      "name": "human phenotype ontology",
      "namespacePrefix": "HP",
      "url": "http://purl.obolibrary.org/obo/hp.owl",
      "version": "2018-03-08",
      "iriPrefix": "http://purl.obolibrary.org/obo/HP_"
    }]
  }
}

```

Running LIRICAL with clinical and genomic data

LIRICAL will perform combined phenotype and variant analysis if the Phenopacket contains an `htsFiles` element. In this case, you need to indicate the path to the VCF file on your system as shown above (`/example/path/Pfeiffer.vcf`).

The `-p` option is used to indicate the Phenopacket, and the `-e` option is used to indicate the location of the *Exomiser database files*. The minimal command (using all default settings) is as follows.

```
$ java -jar LIRICAL.java phenopacket -p /path/to/example.json -e /path/to/exomiser-
↳data/
```

LIRICAL Options for clinical/genomic analysis

All of the options for the phenotype-only phenopacket analysis (*Running LIRICAL with a Phenopacket file*) can be used for the clinical/genomic analysis. Additionally, the following options are available.

```
-b, --background
```

LIRICAL uses a background frequency file that records the frequency of predicted pathogenic variants in protein-coding genes (as estimated from gnomAD data). By default, LIRICAL will use pre-fabricated files for this (that are included in the `src/main/resources/background` directory). This is recommended for most users. If you create your own background file, then you can use it with the `-b` option, that should then indicate the path to a non-default background frequency file.

```
-e, --exomiser
```

Path to the Exomiser data directory (required for VCF-based analysis).

```
--transcriptdb
```

LIRICAL can use transcript data from UCSC, Ensembl, or RefSeq. The default is *RefSeq*, but transcript definitions from *UCSC* and *Ensembl* can also be used (e.g., `--transcriptdb UCSC` or `--transcriptdb ensembl`).

```
--global
```

By default, LIRICAL's default mode, which only ranks candidate genes for which at least one pathogenic allele is present in the VCF file. LIRICAL can also be run in a `--global` mode in which diseases are ranked irrespective of whether a disease gene is known for a disease or whether the gene is found to have a pathogenic allele or not.

1.2.2 Running LIRICAL with a YAML file

The other allowed input format is YAML. The format is designed to be as close as possible to that of the Exomiser YAML format, but some fields, such as negated HPO terms, as LIRICAL-specific.

Running LIRICAL with a YAML file (HPO data)

The recommended input format for running LIRICAL is the [Phenopacket](#), but LIRICAL also supports [YAML](#), which is a simple, human readable format that is commonly used for configuration files.

YAML

Before running LIRICAL, download and built it and set it up according to the instructions on the [Setting up LIRICAL](#) page. LIRICAL uses default values for many configuration options (see below). An example of the simplest possible YAML configuration file is shown:

```
---
analysis:
  # hg19 or hg38 is supported
  genomeAssembly: hg19
hpoIds: ['HP:0001156', 'HP:0001363', 'HP:0011304', 'HP:0010055']
```

This file can be found at `src/test/resources/yaml/simple.yaml`.

An example YAML file that uses several parameters is shown below:

```
## LIRICAL Analysis Template.
# These are all the possible options for running LIRICAL. Use this as a template for
# your own set-up.
---
analysis:
  # hg19 or hg38 is supported
  mindiff: 50
  threshold: 0.05
  tsv: false
  datadir: data
  orphanet: false
hpoIds: ['HP:0001156', 'HP:0001363', 'HP:0011304', 'HP:0010055']
negatedHpoIds: ['HP:0001328']
prefix: example
outdir: /path/to/output/directory
```

This file can be found at `src/test/resources/yaml/multiple_params.yaml`.

In YAML, lines that begin with # are comments, and the three dashes indicate the start of the contents of the file. The `analysis` element is used to hold a dictionary with options for running the program. The items in `analysis` refer to the genome assembly and to the paths of files required to run LIRICAL. Users must provide values for `genomeAssembly`, `vcf`, and `exomiser`. Default values will be use for the other three entries if the user does not provide values.

1. `mindiff` By default, LIRICAL shows all differential diagnoses with a posterior probability of at least 1%, and at least 10 entries regardless of the posterior probability. If you want LIRICAL to show details about more differentials, set this option to the desired number.

2. `threshold` This option controls the minimum post-test probability to show a differential diagnosis in HTML output. By default, LIRICAL shows all differnetials with a posterior probability of 1% or greater.

3. `tsv` Use TSV instead of HTML output (Default: `false`).
4. `datadir` The path with LIRICAL data that should be downloaded before running LIRICAL (see [Setting up LIRICAL](#) for details). This option should not be used if the default data location (`data`) is used.
5. `orphanet` If true, use annotation data from [Orphanet](#).

Additionally, `hpoIds` is a list of HPO term representing the clinical manifestations observed in the individual being analyzed. In contrast, `negatedHpoIds` represents phenotypic abnormalities (HPO terms) that were explicitly excluded in the proband.

Finally, `prefix` is the prefix of the output file (optional, default: `lirical`). For instance, if the prefix is `example1`, then the HTML output file will be `example1.html`.

Running LIRICAL

A typical command that runs LIRICAL using settings shown in the YAML file with the default data directory would be simply

```
$ java -jar LIRICAL.jar yml -y example.yml
```

Running LIRICAL with a YAML file (HPO and VCF data)

YAML

Before running LIRICAL, download and built it and set it up according to the instructions on the [Setting up LIRICAL](#) page. LIRICAL uses default values for many configuration options (see below), and a simple YAML configuration file would include the following information.

```
## LIRICAL Analysis Template.
# These are all the possible options for running LIRICAL. Use this as a template for
# your own set-up.
---
analysis:
  # hg19 or hg38 is supported
  genomeAssembly: hg19
  vcf: /path/to/example.vcf
  exomiser: /path/to/1811_hg19/
hpoIds: ['HP:0001156', 'HP:0001363', 'HP:0011304', 'HP:0010055']
prefix: example
```

This file can be found at `src/test/resources/yaml/hpo_and_vcf.yml`.

In YAML, lines that begin with `#` are comments, and the three dashes indicate the start of the contents of the file. The `analysis` element is used to hold a dictionary with options for running the program. The items in `analysis` refer to the genome assembly and to the paths of files required to run LIRICAL. Users must provide values for `genomeAssembly`, `vcf`, and `exomiser`. Default values will be use for the other three entries if the user does not provide values.

1. `vcf` is the path to the file we want to analyze (required).
2. `exomiser` is the path to the Exomiser data directory (see [Exomiser database files](#) for details) (required)
3. `genomeAssembly` This should be either `hg19` (or `hg37`, which is synonymous) or `hg38` (required)
4. `datadir` The path with LIRICAL data that should be downloaded before running LIRICAL (see [Setting up LIRICAL](#) for details). This option should not be used if the default data location (`data`) is used.

5. `background_freq` Most users will want to use the precomputed background files provided by LIRICAL. In this case, the correct background file (for hg19 or hg38) is determined automatically on the basis of the `genomeAssembly`. This option should be used to have LIRICAL ingest a custom background file
6. `transcriptdb`. This determines the set of transcripts used to call variants. Valid values are UCSC or RefSeq, and the default is UCSC (optional)
7. `global`. If the YAML file contains the line `global: true` then it will not discard candidate diseases with no known disease gene or candidates for which no predicted pathogenic variant was found in the VCF.

Any of the options described in [Running LIRICAL with a YAML file \(HPO data\)](#) can also be used here.

Additionally, `hpoIds` is a list of HPO term representing the clinical manifestations observed in the individual being analyzed. Finally, `prefix` is the prefix of the output file (optional, default: `lirical`) For instance, if the prefix is `example1`, then the HTML output file will be `example1.html`. `prefix` is *not* used to represent the path to the outfile.

The following YAML file contains values for all of the options.

```
## LIRICAL Analysis Template.
# These are all the possible options for running LIRICAL. Use this as a template for
# your own set-up.
---
analysis:
# hg19 or hg38 is supported
  genomeAssembly: hg19
  vcf: /Users/peterrobinson/Documents/data/Pfeiffer.vcf
  exomiser: /Users/peterrobinson/Documents/data/exomiser/1802_hg19/
  datadir: /path/to/custom_location1/
  background: /path/to/custom_location2/background-hg38.txt
  transcriptdb: refseq
hpoIds: [ 'HP:0001363', 'HP:0011304', 'HP:0010055' ]
negatedHpoIds: [ 'HP:0001328' ]
prefix: example2
outdir: /path/to/output/directory
```

This file can be found at `src/test/resources/yaml/hpo_and_vcf_mult.yml`. This YAML file additionally has a list of HPO terms that represent abnormalities that were **excluded** in the proband (`negatedHpoIds`).

You can use either example file as a starting point for your own configuration file.

Running LIRICAL

A typical command that runs LIRICAL using settings shown in the YAML file with the default data directory would be simply

```
$ java -jar LIRICAL.jar yaml -y example.yml
```

1.2.3 Choosing between YAML and Phenopacket input formats

How should users choose between YAML and Phenopackets as an input format?

YAML or Phenopacket as input?

How should users choose between YAML and Phenopackets as an input format? In general, we recommend that users choose [Phenopackets](#) as the input format. YAML is a simple format that can easily be edited by hand in a text editor

and is suitable for testing and demonstration, but is not as flexible or robust as Phenopackets. We have provided a simple tool that creates Phenopackets for use by LIRICAL and other similar software ([PhenopacketGenerator](#)). As a convenience, we present the same simple case in first YAML and then Phenopacket format.

YAML version

The data represents an individual with some characteristic manifestations of [neurofibromatosis type 2](#), in whom [Tibial pseudoarthrosis](#) (HP:0009736), a characteristic feature of neurofibromatosis type 1, has been ruled out.

```
analysis:
  mindiff: 50
  threshold: 0.05
  tsv: true
  orphanet: true
hpoIds: ['HP:0002321', 'HP:0000365', 'HP:0000360', 'HP:0009589', 'HP:0002858']
negatedHpoIds: ['HP:0009736']
prefix: NF2-example
```

Save this file as `example.yml` and then run LIRICAL as

```
$ java -jar LIRICAL.jar yml -y example.yml
```

Phenopackets version

The identical data can be represented in Phenopacket format (in which only required fields are used) as follows.

```
{
  "id": "proposita",
  "subject": {
    "id": "proposita",
  },
  "phenotypicFeatures": [ {
    "type": {
      "id": "HP:0000360",
      "label": "Tinnitus"
    }
  }, {
    "type": {
      "id": "HP:0002321",
      "label": "Vertigo"
    }
  }, {
    "type": {
      "id": "HP:0000365",
      "label": "Hearing impairment"
    }
  }, {
    "type": {
      "id": "HP:0009589",
      "label": "Bilateral vestibular Schwannoma"
    }
  }, {
    "type": {
      "id": "HP:0002858",
```

(continues on next page)

(continued from previous page)

```

        "label": "Meningioma"
      }
    }, {
      "type": {
        "id": "HP:0009736",
        "label": "Tibial pseudoarthrosis"
      },
      "negated" : "true"
    }
  ],
  "metaData": {
    "createdBy": "Hpo Case Annotator : 1.0.13",
    "submittedBy": "HP:probinson",
    "resources": [{
      "id": "hp",
      "name": "human phenotype ontology",
      "url": "http://purl.obolibrary.org/obo/hp.owl",
      "version": "2018-03-08",
      "namespacePrefix": "HP",
      "iriPrefix": "http://purl.obolibrary.org/obo/HP_"
    }]
  }
}

```

Save this file as `example.json` and then run LIRICAL as

```
$ java -jar LIRICAL.jar phenopacket -p example.json
```

Identical results should be obtained for both cases. See [Running LIRICAL with a Phenopacket file](#) and [Running LIRICAL with a YAML file \(HPO data\)](#) for more information about parameters and running LIRICAL with genomic data from VCF files.

1.3 LIRICAL Output

LIRICAL accepts phenopackets or YAML files as input (see [Running LIRICAL](#)). In either case, LIRICAL can output either an HTML file with a detailed summary of its analysis results, or a tab-separated value (TSV) file that can be used by bioinformatic pipelines. On typical computers, LIRICAL will run from about 15 to 60 seconds, or longer if a whole-genome file is used as input.

1.3.1 LIRICAL HTML Output

Sample information and list of differentials

The HTML output page begins with a summary of the sample name and a list of the [HPO](#) terms used to run the program. By default, LIRICAL shows a detailed output only for the top 10 differential diagnoses (or more if more diagnoses have a posterior probability above the default threshold of 1%). The minimum number of differential diagnoses to show can be changed with the `-m` option, and the probability threshold can be changed with the `-t` option.

Disease evaluations

LIRICAL evaluates each of the diseases in the HPO database and estimates the probability that a disease explains the observed phenotypic abnormalities (and if applicable, the observed variants).

LIRICAL: Likelihood Ratio Interpretation of Clinical Abnormalities

[Sample](#) [Differential diagnosis](#) [Remaining genes](#) [Settings](#) [About](#)

Sample name: n/a

Observed Phenotypic Features

- Vertigo (HP:0001211)
- Hearing impairment (HP:0000265)
- Tinnitus (HP:0000369)
- Bilateral vestibular Schwannoma (HP:0009589)
- Meningioma (HP:0002858)

Excluded phenotypic features:

- Excluded: Tibial pseudoarthrosis (HP:0009736)

LIRICAL analysis performed on 2019/10/11.

Differential diagnosis: posterior probability above 5.0%

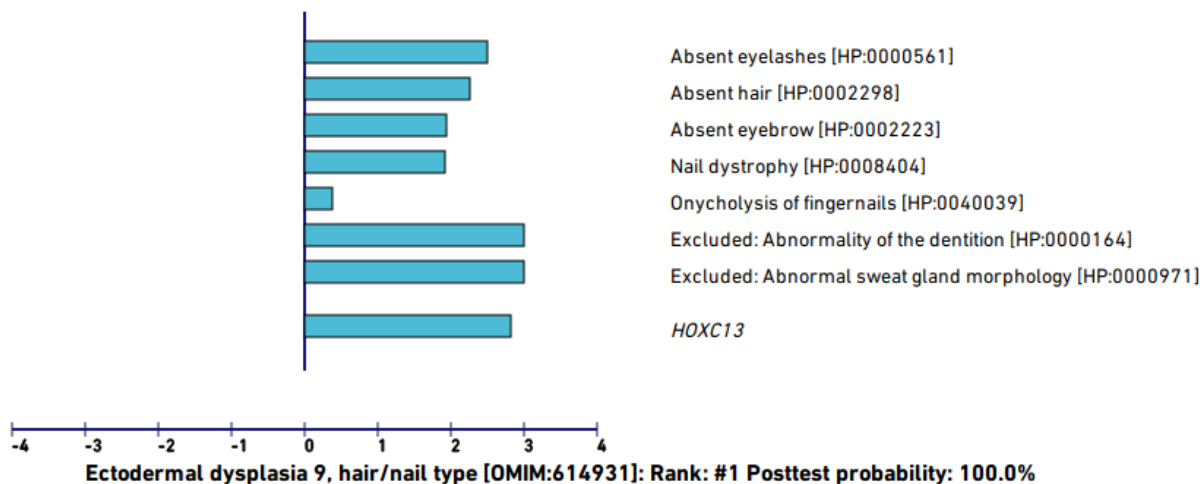
Top differential diagnoses:

1. NEUROFIBROMATOSIS, TYPE II
2. VON HIPPEL-LINDAU SYNDROME; VHL/VON HIPPEL-LINDAU SYNDROME, MODIFIERS OF INCLUDED
3. DEAFNESS, AUTOSOMAL DOMINANT 9; DENA2
4. HYPEROSTOSIS CRANIALIS INTERNA; HCIN
5. MENIERE DISEASE
6. CHARI MALFORMATION TYPE I

For example, the following figure shows the evaluation of a simulated case based on a published case report of an individual with pure hair and nail ectodermal dysplasia (ECTD9) related to a pathogenic variant in the *HOXC13* gene (Khan et al., 2017).

LIRICAL has estimate the composite likelihood ratio score at 8.951 (note that this is expressed on a \log_{10} scale, so that the likelihood ratio is actually $10^{8.951}$). The posttest probability is close to 100%.

The contribution of each of the HPO terms entered for the proband is shown. The contribution of each term is indicated by the length of the blue bar (which shows the decadic logarithm of the likelihood ratio for the term. For instance, if the bar is 2 units long, then the likelihood ratio is $10^2=100$).

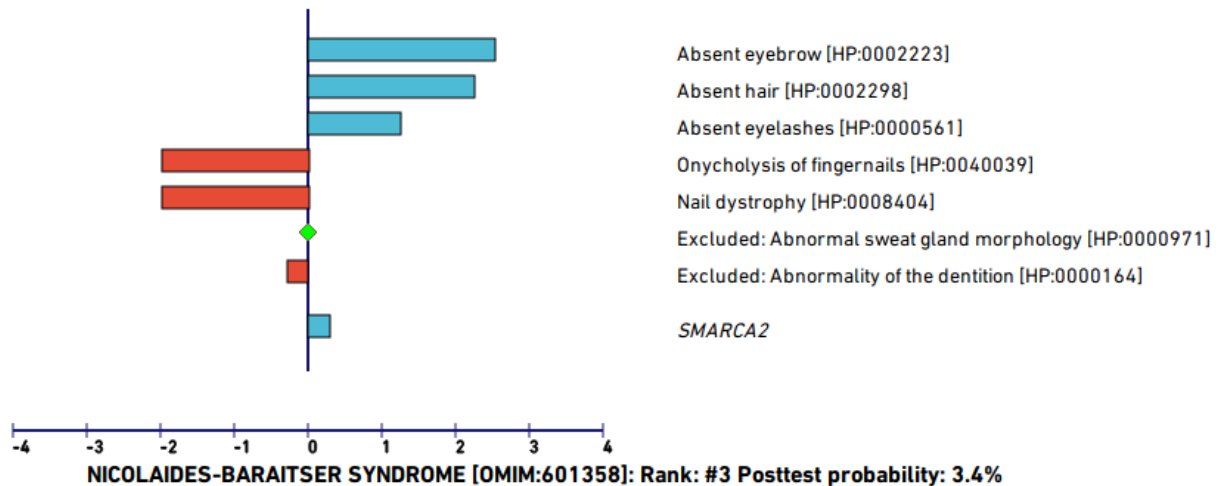


The user should inspect the top differentials. In this case, the posterior probability of the remaining differentials drops off quickly. For instance, the third best hit, Nicolaides Baraitser syndrome, has a posttest probability of only 3.6% and several of the observed phenotypes are not characteristic of this syndrome and thus reduce the match score (indicated as red bars).

Other information

LIRICAL shows a list of candidate diseases and genes with a low post-test probability in the section [Genes/ Diseases with low posttest probability](#) (Click on the Table to show details).

In some cases, we have observed that some variants are linked to gene entities that do not have an NCBI Gene ID. This effectively means that LIRICAL will not further analyze these variants. If any such variants are found, they will be shown in a section called [Gene symbols that could not be annotated](#). If the section is not present,



then all variants were annotated. In practice, the gene symbols that cannot be linked to an NCBI ID represent accession numbers that are not confirmed genes, e.g., CR627135, AX746851, AK096159, We have found that using refseq as the transcript option means that all genes also have a Gene ID, but UCSC includes transcripts that do not have a Gene ID, and so users can experiment with both options. This information is provided for completeness' sake, but it is not diagnostically important.

1.3.2 LIRICAL TSV Output

If LIRICAL is run with the `--tsv` option, it will output a tab-separated values (TSV) file with the results for each of the diagnoses. For example, the following command will run LIRICAL on a Phenopacket and output a TSV file with the results.

```
$ java -jar LIRICAL.jar phenopacket \
  --global \
  -e /path(..)/1811_hg19 \
  -p /path(..)/example-phenopacket.json \
  --tsv
```

By default, LIRICAL outputs the data to a file called `lirical.tsv`. This can be altered with the `-x <prefix>` option.

Table 1: LIRICAL's TSV format

Item	Explanation
rank	placement of the candidate diagnosis by LIRICAL
diseaseName	Name of the candidate disease
diseaseCurie	disease ID, e.g., OMIM:154700
pretestprob	Pretest probability of the candidate disease
posttestprob	Posttest probability of the candidate disease
compositeLR	Combined likelihood ratio of the candidate disease (logarithm of the product of all individual LR's)
entrezGeneId	Identifier of the candidate disease gene (if available)
variants	variant evaluation (if available)

The file begins with comment lines (that start with an exclamation mark) that provide information about the HPO terms used to run the analysis.

1.4 LIRICAL: How does it work?

In medical genetics and related fields, [Human Phenotype Ontology \(HPO\)](#) analysis has become one of the standard methods for diagnostics. Current algorithms such as [Exomiser](#) and [Phenomizer](#) use a variety of semantic and statistical approaches to prioritize the typically long lists of genes with candidate pathogenic variants, but do not provide robust estimates of the strength of the predictions beyond the placement in a ranked list, nor do they provide measures of how much any individual phenotypic observation has contributed to the prioritization result. LIRICAL exploits the clinical likelihood ratio framework to provide an estimate of the posttest probability of candidate diagnoses, the likelihood ratio for each observed HPO phenotype, and the predicted pathogenicity of observed variants.

LIRICAL makes use of the clinical likelihood ratio (LR) framework to perform phenotype-driven genomic diagnostics that addresses these shortcomings. The LR is defined as the probability of a given test result in an individual with the target disorder divided by the probability of that same result in an individual without the target disorder. LIRICAL can be run in a phenotype-only mode or can be run to analyze both phenotype and genotype findings (e.g., from Exome or Genome sequencing). The following pages explain the algorithmic details.

1.4.1 LIRICAL's Phenotype Score

LIRICAL calculates a likelihood ratio score for phenotypic observations for each differential diagnosis. The phenotype likelihood ratio score can be combined with LIRICAL's genotype likelihood ratio score for a combined analysis of phenotypes and genetic data (such as exome or genome sequencing) or can be used as a tool to assess phenotype data alone.

This page explains how to interpret LIRICAL's phenotype score.

Each disease shows a detailed explanation of the matching score. For instance, the match with [Ectodermal Dysplasia 9, Hair/nail Type](#) shown on [LIRICAL HTML Output](#) shows the following:

```
E:Specific learning disability[HP:0001328] [152.894]
E:Obesity[HP:0001513] [62.561]
E:Rod-cone dystrophy[HP:0000510] [45.396]
Q<D:Macular degeneration[HP:0000608]<Retinal degeneration[HP:0000546] [32.605]
E:Strabismus[HP:0000486] [16.648]
E:Global developmental delay[HP:0001263] [6.800]
Q~D:Attenuation of retinal blood vessels[HP:0007843]~Abnormal retinal_
↪morphology[HP:0000479] [1.267]
```

Each match shows a code for the category of the match, followed by details of the matching term (only one term is shown for exact matches), and the matching score.

The algorithm is implemented in the function `getLikelihoodRatio` in the class `PhenotypeLikelihoodRatio` (see the Java code for details). The algorithm checks each query term for the best match to disease terms and uses a sequence of rules to try to find the match. For the following explanation, we will refer to the query HPO term as h_i and the disease as \mathcal{D} .

The likelihood ratio is calculated as $LR(h_i) = \frac{P(h_i|\mathcal{D})}{P(h_i|\neg\mathcal{D})}$. The following sections describe how the numerator is calculated, i.e., the probability that an individual with the disease has the phenotypic feature. The denominator is always calculated as the probability that an individual with an arbitrary Mendelian disease has the feature in question (which is calculated based on the entire HPO database for Mendelian diseases). See the manuscript for more details.

1. h_i is identical to one of the terms to which \mathcal{D} is annotated.

In this case, $P(h_i|\mathcal{D})$ is equal to the frequency of h_i among all individuals with disease \mathcal{D} (by default, this is taken to be 100%, but in many cases more precise frequencies are available in the HPO database).

In the output file of LIRICAL, such matches are shown with the code **E**. The likelihood ratio for this match is 84.767.

Table 2: Exact phenotypic feature match (E)

Example
E:Nail dystrophy[HP:0008404][84.767]

2. h_i is an ancestor (superclass) of one or more of the terms to which \mathcal{D} is annotated.

Because of the annotation propagation rule of subclass hierarchies in ontologies, $P(h_i|\mathcal{D})$ is implicitly annotated to all of the ancestors of the set of annotating terms. For instance, if the computational disease model of some disease \mathcal{D} includes the HPO term *Polar cataract* (HP:0010696) then the disease is implicitly annotated to the parent term *Cataract* (HP:0000518) (to see this consider that any person with a polar cataract can also be said to have a cataract).

In this case, the probability of h_i in disease \mathcal{D} is equal to the maximum frequency of any of the ancestors of h_i in \mathcal{D} .

Table 3: Query term is parent of a disease term term (D<Q)

Example
D<Q:Short middle phalanx of the 5th finger[HP:0004220]<Brachydactyly[HP:0001156][29.847]

3. h_i is a child term (subclass) of one or more of the terms to which \mathcal{D} is annotated.

In this case, h_i is a descendant (i.e., specific subclass of) some term h_j of \mathcal{D} . For instance, disease \mathcal{D} might be annotated to *Syncope* (HP:0001279), and the query term h_i is *Orthostatic syncope* (HP:0012670), which is a child term of *Syncope*. In addition, *Syncope* has two other child terms, *Carotid sinus syncope* (HP:0012669) and *Vasovagal syncope* (HP:0012668). According to our model, we will adjust the frequency of *Syncope* in disease \mathcal{D} (say, 0.72) by dividing it by the total number of child terms of h_j (so in our example, we would use the frequency $0.72 \times 1/3 = 0.24$).

Table 4: Query term is child of disease term (Q<D)

Example
Q<D:Macular degeneration[HP:0000608]<Retinal degeneration[HP:0000546][32.605]

In this example, the likelihood ratio is 2.082. *Macular degeneration* (HP:0000608) is a subclass of *Retinal degeneration* (HP:0000546).

4. h_i and some term to which \mathcal{D} is annotated have a non-root common ancestor.

This option pertains if options (ii) and (iii) do not, i.e., h_i is not a child term of any disease term h_j and no disease term h_j is a child of h_i .

If this is the case, then we find the closest common-ancestor, and determine the likelihood ratio according to the formula $LR(h_i) = \frac{P(h_i|\mathcal{D})}{P(h_i|\neg\mathcal{D})}$. Because the common ancestor is higher up in the HPO hierarchy, the likelihood ratio tends to be lower and sometimes substantially lower. In order to limit the amount of negative influence of any one query term, the likelihood ratio is defined to be at least 1/100.

Table 5: Non-root distant match (Q~D)

Example
Q~D:Macular degeneration[HP:0000608]~Abnormal retinal morphology[HP:0000479][0.127]

In this example, *Macular degeneration* (HP:0000608) is not a direct child of *Abnormal retinal morphology* (HP:0000479) – it is a “grandchild”, i.e., *Macular degeneration* is a direct child of *Abnormal macular morphology* (HP:0001103) which in turn is a direct child of *Abnormal retinal morphology*. Therefore, it is considered to be a non-root distant match. It is assigned a likelihood ratio of 0.127.

5. h_i does not have any non-root common ancestor with any term to which \mathcal{D} is annotated.

In this case, a heuristic value of 1/100 is assigned for the likelihood ratio.

Table 6: No match (NM)

Example
NM:Specific learning disability[HP:0001328][0.010]

6. phenotypic abnormality h_i is explicitly excluded from disease \mathcal{D} .

In the HPO annotation resource, each disease is represented by a list of HPO terms that characterize it together with metadata including provenance, and in some cases, frequency and onset information. Some diseases additionally have explicitly excluded terms (there are a total of 921 such annotations in the September 2019 release of the HPOA data). These annotations are used for phenotypic abnormalities that are important for the differential diagnosis. For instance, Marfan syndrome and Loeys-Dietz syndrome share many phenotypic abnormalities. The feature *Ectopia lentis* (HP:0001083) is characteristic of Marfan syndrome but is not found in Loeys-Dietz syndrome. The likelihood ratio for such query terms is assigned an arbitrary value of $\frac{1}{1000}$, i.e., the ratio for a candidate diagnosis is reduced by a factor of one thousand if an HPO term is present in the proband that is explicitly excluded from the disease.

Table 7: Excluded in query and present in disease (XP)

Example
XP:Ectopia lentis[HP:0001083][0.001]

If a term is excluded in the query, but not annotated one way or another in the disease, then the likelihood ratio is calculated without additional heuristics. These query terms generally result in a likelihood ratio near 1 and do not affect the differential diagnostic ranking much.

Table 8: Excluded in query and not annotated in disease (XA)

Example
XA:Abnormality of alkaline phosphatase activity[HP:0004379][1.008]

On the other hand, if the query includes a negated term that is explicitly excluded in the disease, then the opposite value is assigned, i.e., the ratio for a candidate diagnosis is increased by a factor of one thousand if an HPO term is present in the proband that is explicitly excluded from the disease.

Table 9: Excluded in both query and disease (XX)

Example
XX:Trident hand[HP:0004060][1000.000]

1.4.2 LIRICAL’s Genotype Score

We can estimate the pathogenicity of a variant on the basis of a computational pathogenicity score that ranges from 0 (predicted benign) to 1 (maximum pathogenicity prediction). LIRICAL uses the pathogenicity score prediction of *Exomiser*. Our model depends on the assumed mode of inheritance of the disease, and provides an estimate likelihood

ratio for the observed genotype. For example, we expect two pathogenic alleles in an autosomal recessive disease and one in an autosomal dominant disease. Our model takes into account the expected frequency of seeing predicted pathogenic variants in the population. Genes known to carry few common functional variants in healthy individuals may be judged more likely to cause certain kinds of disease than genes known to carry many such variants (Petrovski et al., 2013).

LIRICAL's model provides an integrated score for each gene that assesses the observed genotype, comparing its probability given that a disease associated with the gene is present in the proband vs. the probability that the genotype is unrelated to the clinical manifestations observed in the proband. See the manuscript for algorithmic details.

1.5 LIRICAL Options

This page summarizes the options explained in detail in the *Running LIRICAL* section.

1.5.1 Download

The `download` command downloads files required to run LIRICAL:

- `Homo_sapiens_gene_info.gz`
- `hp.obo`
- `phenotype.hpoa`
- `mim2gene_medgen`

By default, LIRICAL will create a directory called `data` and download the files there. LIRICAL will download to a non-default directory if the user passes the `-d` option.

Table 10: `download` command

short	long	Default	Explanation
<code>-d</code>	<code>--download</code>	<code>data</code>	directory to download data
<code>-w</code>	<code>--overwrite</code>	<code>false</code>	overwrite previously downloaded files, if any

1.5.2 Running LIRICAL with a phenopacket

The `phenopacket` command runs LIRICAL from a Phenopacket file.

Table 11: phenopacket command

short	long	Default	Explanation
-p	--phenopacket	n/a	path to Phenopacket
-d	--download	data	directory that contains the downloaded data
-g	--global	false	retain candidate diseases even if no candidate gene is known or no candidate variant is found in VCF file.
-m	--mindiff	10	minimal number of differential diagnoses to show in the HTML output file.
-o	--output-directory	n/a	directory into which to write output file(s).
-x	--prefix	lirical	prefix of outfile
-t	--threshold	0.01	minimum post-test probability to show a diagnosis in the HTML output. This option, together with --mindiff, controls the number of panels that show information about candidates in the HTML output.
none	--transcriptdb	ucsc	transcript database. Valid options are UCSC, Ensembl, and RefSeq
none	--tsv	false	Use TSV instead of HTML output

1.5.3 YAML

The `yaml` command runs LIRICAL from a YAML configuration file. Users should indicate all non-default arguments within the YAML file. The only valid argument for the `yaml` command is the path to the YAML file (`-y <path>`).

```
$ java -jar LIRICAL.java yaml -y example.yaml
```

Table 12: yaml command

short	long	Default	Explanation
-y	--yaml	n/a	path to yaml configuration file

1.6 LIRICAL Advanced Options

Most users will not need these commands, which are hidden from the normal user menu. The LIRICAL code base contains functionalities that we used to develop and validate the program, and we describe them here briefly.

1.6.1 Generating the background files

LIRICAL uses the files `src/main/resources/background/background-hg19.tsv` and `src/main/resources/background/background-hg38.tsv` to estimate the expected population frequencies of predicted pathogenic variants. The important classes are `BackgroundFrequencyCommand.java` and `GenicIntoleranceCalculator.java`. You do not need to generate the files yourself to run Exomiser (they are included in the resource files). The following command generates the files.


```
java -jar target/LIRICAL.jar background -e /path/to/exomiser/1811_hg19 -g hg19
```

1.7 Tutorial

This tutorial shows how to use LIRICAL to evaluate an exome.

1.7.1 Setup

Follow the instructions in [Setting up LIRICAL](#) to install the Exomiser database. Note the location of the Exomiser database (it will be needed to run LIRICAL, see below). Most users should download the pre-built version of LIRICAL available on the [Releases](#) page. Instructions are also offered for building LIRICAL from source if desired.

1.7.2 The data

We have simulated an exome VCF file by adding a disease associated variant to a VCF file derived from project.NIST.hc.snps.indels.NIST7035.vcf. A disease-associated mutation in the TGFBR2 gene (see Patient 4 in [Cao et al., 2018](#)) was spiked into the VCF file.

Download the VCF file (LDS2.vcf) from [Figshare](#).

1.7.3 Creating a phenopacket

Here is an excerpt of the text that described patient 4 in the above cited article:

Patient 4 is a 9-year-old girl. She was clinically diagnosed with suspected Marfan syndrome according to the first impression. She was 144 cm tall and weighed 24 kg. Her father was 176 cm tall and weighed 53 kg. The ↵
 ↪phenotypes of this patient include strabismus, refractive error, pectus carinatum, ↵
 ↪scoliosis, arachnodactyly, and camptodactyly. The patient's main cardiovascular abnormalities were Sinus of Valsalva aneurysm, aortic root dilation, aortic regurgitation, atrial septal defect, patent foramen ovale, pulmonary artery dilatation, and tricuspid valve prolapse with regurgitation. Craniofacial abnormalities of the patient include bifid uvula, malar hypoplasia, and micrognathia.

Use the [PhenopacketGenerator](#) to create a Phenopacket.

To set up PhenopacketGenerator, you will first need to set the location of the hp.obo file. Download hp.obo from the [Download page](#) of the HPO website. Enter your Biocurator id by selecting “Set biocurator id” from the edit menu, and enter an arbitrary Phenopacket ID and proband ID. Use the dropdown menus to enter “9 years” for Age and “Female” for sex.

From the edit menu, select “Set path to hp.obo file”, then select the location of the hp.obo on your computer. After a moment, the ontology will load and “Enter HPO terms” will be clickable. Load HPO terms for this case by clicking “Enter HPO term”. Paste the clinical description above into the text-mining window of PhenopacketGenerator, click “Analyze”, select HPO terms, click “Add selected terms”, then “Confirm and Close”.

Then, select the location of the VCF file that you saved in the previous step, and enter the Genome assembly (hg19).

You can now export the phenopacket. Use the filename LDS2.json (or choose another name and adjust the following command accordingly).

HPO tree browser

- ▶ Blood group
- ▶ Clinical course
- ▶ Clinical modifier
- ▶ Frequency
- ▶ Mode of inheritance
- ▶ Past medical history
- ▶ Phenotypic abnormality

Click on HPO term in the tree browser to display additional information

☐ NOT present

HPO text-mining analysis terms:

Patient 4 is a 9-year-old girl. She was clinically diagnosed with suspected Marfan syndrome according to the first impression. She was 144 cm tall and weighed 24 kg. Her father was 176 cm tall and weighed 53 kg. The phenotypes of this patient include strabismus, refractive error, pectus carinatum, scoliosis, arachnodactyly, and camptodactyly. The patient's main cardiovascular abnormalities were Sinus of Valsalva aneurysm, aortic root dilation, aortic regurgitation, atrial septal defect, patent foramen ovale, pulmonary artery dilatation, and tricuspid valve prolapse with regurgitation. Craniofacial abnormalities of the patient include bifid uvula, malar hypoplasia, and micrognathia.

HPO terms:

- ☐ Abnormal cardiac septu
- ☐ Abnormality of the card
- ☒ Aortic regurgitation
- ☒ Arachnodactyly
- ☒ Atrial septal defect
- ☒ Bifid uvula

"NOT" HPO terms:

Approved terms

ID	Observed	Name	Definition
No content in table			

File Edit Help

Phenopacket ID

Proband ID

Age

Months

Sex

13 observed terms, 0 excluded terms

/home/robinp/data/lirical/LDS2.vcf

Wrote to /home/robinp/data/lirical/lids2.json

Ontology loaded

1.7.4 Running LIRICAL

Run LIRICAL as follows.

```
$ java -jar LIRICAL.jar phenopacket -p LDS2.json -e /path/to/exomiser-data/ -x LDS2
```

1.7.5 Viewing the results

The above command will create a new file called `LDS2.html` (the `-x` option controls the prefix of the output file). Open this file in a web browser. The top of the page shows some information about the input files and a list of observed and excluded HPO terms. The next section shows summarized representations of the top candidates.

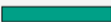







Rank	Post-test probability	Disease	ID	Profile match	LR (log)	Gene
1	100% 	LOEYS-DIETZ SYNDROME 2	OMIM:610168		20.281	TGFBR2
2	0%	ROBINOW SYNDROME, AUTOSOMAL DOMINANT 3	OMIM:616894		-1.379	DVL3
3	0%	ANTLEY-BIXLER SYNDROME WITHOUT GENITAL ANOMALIES OR DIS(1)	OMIM:207410		-4.291	FGFR2
4	0%	KABUKI SYNDROME 1	OMIM:147920		-4.655	KMT2D
5	0%	SAETHRE-CHOTZEN SYNDROME	OMIM:101400		-7.232	FGFR2
6	0%	APERT SYNDROME	OMIM:101200		-7.858	FGFR2
7	0%	ROBINOW SYNDROME, AUTOSOMAL DOMINANT	OMIM:180700		-9.983	DVL3
8	0%	SMITH-LEMLI-OPITZ SYNDROME	OMIM:270400		-10.49	DHCR7
9	0%	HAMAMY SYNDROME	OMIM:611174		-12.293	IRX5
10	0%	VAN MALDERGEM SYNDROME 1	OMIM:601390		-13.09	DCHS1

Fig. 2: Summary view of the top candidates.

Each row in the summary shows the rank, post-test probability, and name/ID of the disease. The row includes a sparkline representation of the phenotypic profiles of each candidate, with green bars indicating positive contributions and red bars negative contributions to the diagnosis. The last bar represents the genotype likelihood ratio if LIRICAL was run with a VCF file. Mousing over the individual bars will show the name of the HPO term or gene, and all sparklines show the terms in the same order.

LIRICAL then presents a detailed analysis of each of the top candidates. The summary shows information about identified variants and the phenotypic profile. Mousing over the graphic shows information about the likelihood ratio and the type of the match.

The remaining part of the HTML output page contains information about the other top candidates and a list of all diseases analyzed. The bottom of the page includes explanations and documents the settings used for the analysis.

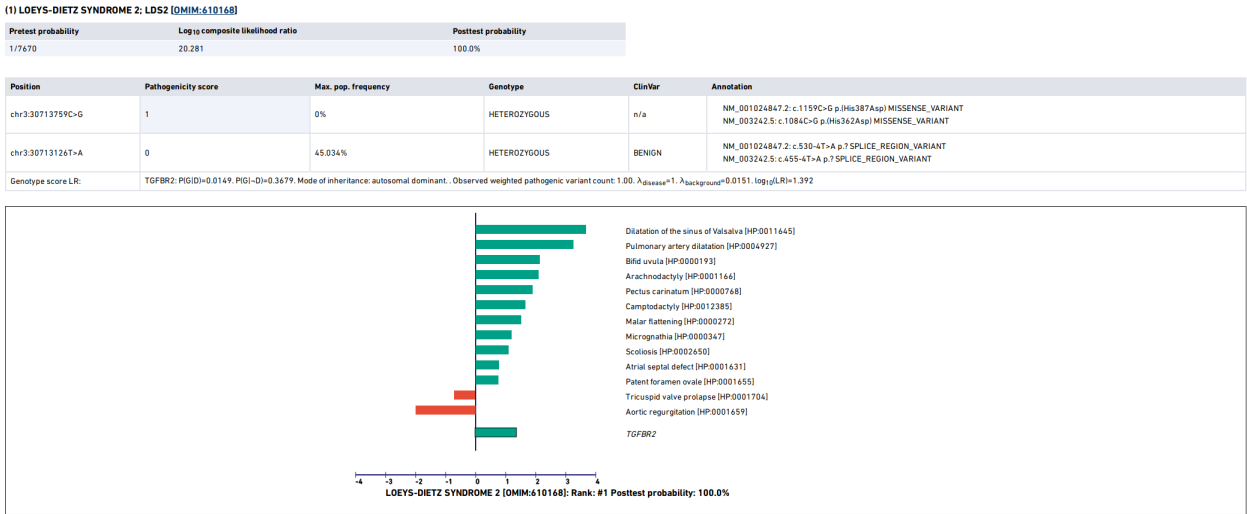


Fig. 3: Detailed view of the top candidate LoEys-Dietz syndrome type 2.