

# HPO.db

January 28, 2026

---

HPOALIAS

*Map from HPO alias to HPO terms*

---

## Description

HPOALIAS is an R object that provides mapping from HPO alias to HPO terms

## Details

Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 20230405

## Value

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

## References

<https://hpo.jax.org/app/data/ontology>

## Examples

```
# Convert the object to a list
xx <- as.list(HPOALIAS)
```

HPOANCESTOR

*Annotation of HPO Identifiers to their Ancestors***Description**

This data set describes associations between HPO terms and their ancestor terms, based on the directed acyclic graph (DAG) defined by the Human Phenotype Ontology Consortium. The format is an R object mapping the HPO terms to all ancestor terms, where an ancestor term is a more general HPO term that precedes the given HPO term in the DAG (in other words, the parents, and all their parents, etc.).

**Details**

Each HPO term is mapped to a vector of ancestor HPO terms. Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 20230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOANCESTOR)
```

HPO.db

*Bioconductor annotation data package***Description**

Welcome to the HPO.db annotation Package. The purpose of this package is to provide detailed information about the latest version of the Human Phenotype Ontology. This package is updated biannually. You can learn what objects this package supports with the following command: `ls("package:HPO.db")` Each of these objects has their own manual page detailing where relevant data was obtained along with some examples of how to use it.

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**Examples**

```
# Convert the object to a list
ls("package:HPO.db")
```

---

HPOCHILDREN

*Annotation of HPO Identifiers to their Children*

---

**Description**

This data set describes associations between HPO terms and their direct children terms, based on the directed acyclic graph (DAG) defined by the Human Phenotype Ontology Consortium. The format is an R object mapping the HPO terms to all direct children terms, where a direct child term is a more specific HPO term that is immediately preceded by the given HPO term in the DAG.

**Details**

Each HPO term is mapped to a vector of children HPO terms. Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 20230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOCHILDREN)
```

HPODO

*Annotation of HPO Identifiers to DO ID***Description**

This data set describes associations between HPO ids and do ids, based on the Human Phenotype Ontology. The format is an R object mapping the HPO ids to do ids.

**Details**

Each HPO id is mapped to a vector of do ids. Mappings were based on data provided by: The Human Phenotype Ontology

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

[https://github.com/mapping-commons/mh\\_mapping\\_initiative](https://github.com/mapping-commons/mh_mapping_initiative)

**Examples**

```
# Convert the object to a list
xx <- as.list(HPODO)
```

HPODb-objects

*AnnotationDb objects and their progeny, methods etc.***Description**

AnnotationDb is the virtual base class for all annotation packages. It contain a database connection and is meant to be the parent for a set of classes in the Bioconductor annotation packages. These classes will provide a means of dispatch for a widely available set of select methods and thus allow the easy extraction of data from the annotation packages.

select, columns and keys are used together to extract data from an AnnotationDb object (or any object derived from the parent class). Examples of classes derived from the AnnotationDb object include (but are not limited to): ChipDb, OrgDb HPODb, InparanoidDb and ReactomeDb.

columns shows which kinds of data can be returned for the AnnotationDb object.

keytypes allows the user to discover which keytypes can be passed in to select or keys and the keytype argument.

keys returns keys for the database contained in the AnnotationDb object . This method is already documented in the keys manual page but is mentioned again here because it's usage with select is

so intimate. By default it will return the primary keys for the database, but if used with the keytype argument, it will return the keys from that keytype.

`select` will retrieve the data as a `data.frame` based on parameters for selected keys columns and keytype arguments. Users should be warned that if you call `select` and request columns that have multiple matches for your keys, `select` will return a `data.frame` with one row for each possible match. This has the effect that if you request multiple columns and some of them have a many to one relationship to the keys, things will continue to multiply accordingly.

So it's not a good idea to request a large number of columns unless you know that what you are asking for should have a one to one relationship with the initial set of keys. In general, if you need to retrieve a column (like GO) that has a many to one relationship to the original keys, it is most useful to extract that separately.

## Usage

```
columns(x)
keytypes(x)
keys(x, keytype, ...)
select(x, keys, columns, keytype, ...)
```

## Arguments

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>       | the AnnotationDb object. But in practice this will mean an object derived from an AnnotationDb object such as a OrgDb or ChipDb object.                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>keys</code>    | the keys to select records for from the database. All possible keys are returned by using the <code>keys</code> method.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>columns</code> | the columns or kinds of things that can be retrieved from the database. As with <code>keys</code> , all possible columns are returned by using the <code>columns</code> method.                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>keytype</code> | the keytype that matches the keys used. For the <code>select</code> methods, this is used to indicate the kind of ID being used with the <code>keys</code> argument. For the <code>keys</code> method this is used to indicate which kind of keys are desired from <code>keys</code>                                                                                                                                                                                                                                                                                                         |
| <code>...</code>     | other arguments. These include: <ul style="list-style-type: none"> <li><b>pattern:</b> the pattern to match (used by <code>keys</code>)</li> <li><b>column:</b> the column to search on. This is used by <code>keys</code> and is for when the thing you want to pattern match is different from the keytype, or when you want to simply want to get keys that have a value for the thing specified by the <code>column</code> argument.</li> <li><b>fuzzy:</b> TRUE or FALSE value. Use fuzzy matching? (this is used with <code>pattern</code> by the <code>keys</code> method)</li> </ul> |

## Value

`keys`, `columns` and `keytypes` each return a character vector or possible values. `select` returns a `data.frame`.

## Author(s)

Marc Carlson

**Examples**

```
## display the columns
## use hpo id keys
dokeys <- head(keys(HPO.db))
res <- select(x = HPO.db, keys = dokeys, keytype = "hpo id",
             columns = c("offspring", "term", "parent"))

## use term keys
dokeys <- head(keys(HPO.db, keytype = "term"))
res <- select(x = HPO.db, keys = dokeys, keytype = "term",
             columns = c("offspring", "hpo id", "parent"))
```

HPOGENE

*Annotation of HPO Identifiers to gene ID***Description**

This data set describes associations between HPO ids and gene ids, based on the Human Phenotype Ontology. The format is an R object mapping the HPO ids to gene ids.

**Details**

Each HPO id is mapped to a vector of gene ids. Mappings were based on data provided by: The Human Phenotype Ontology

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOGENE)
```

---

HPOMAPCOUNTS*Number of mapped keys for the maps in package HPO.db*

---

### Description

HPOMAPCOUNTS provides the "map count" (i.e. the count of mapped keys) for each map in package HPO.db.

### Details

This "map count" information is precalculated and stored in the package annotation DB. This allows some quality control and is used by the [checkMAPCOUNTS](#) function defined in AnnotationDbi to compare and validate different methods (like `count.mappedkeys(x)` or `sum(!is.na(as.list(x)))`) for getting the "map count" of a given map.

### Value

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

### See Also

[mappedkeys](#), `count.mappedkeys`, [checkMAPCOUNTS](#)

### Examples

```
# Convert the object to a list
xx <- as.list(HPOCHILDREN)
```

---

HPOMPO*Annotation of HPO Identifiers to MPO ID*

---

### Description

This data set describes associations between HPO ids and MPO ids, based on the Human Phenotype Ontology. The format is an R object mapping the HPO ids to MPO ids.

### Details

Each HPO id is mapped to a vector of MPO ids. Mappings were based on data provided by: The Human Phenotype Ontology

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

[https://github.com/mapping-commons/mh\\_mapping\\_initiative](https://github.com/mapping-commons/mh_mapping_initiative)

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOMPO)
```

---

HPOOBSOLETE

*Annotation of HPO identifiers by terms defined by Human Phenotype  
Ontology Consortium and their status are obsolete*

---

**Description**

This is an R object mapping HPO identifiers to the specific terms in defined by Human Phenotype Ontology Consortium and their definition are obsolete

**Details**

All the obsolete HPO terms that are collected in this index will no longer exist in other mapping objects.

Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 220230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>



HPOOFFSPRING

*Annotation of HPO Identifiers to their Offspring***Description**

This data set describes associations between HPO terms and their offspring terms, based on the directed acyclic graph (DAG) defined by the Human Phenotype Ontology Consortium. The format is an R object mapping the HPO terms to all offspring terms, where an ancestor term is a more specific HPO term that is preceded by the given HPO term in the DAG (in other words, the children and all their children, etc.).

**Details**

Each HPO term is mapped to a vector of offspring HPO terms. Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 20230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOOFFSPRING)
```

HPOPARENTS

*Annotation of HPO Identifiers to their Parents***Description**

This data set describes associations between HPO terms and their direct parent terms, based on the directed acyclic graph (DAG) defined by the Human Phenotype Ontology Consortium. The format is an R object mapping the HPO terms to all direct parent terms, where a direct parent term is a more general HPO term that immediately precedes the given HPO term in the DAG.

**Details**

Each HPO term is mapped to a named vector of HPO terms. The name associated with the parent term will be either *isa*, *partof*, where *isa* indicates that the child term is a more specific version of the parent, and *partof* indicate that the child term is a part of the parent.

Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 220230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
xx <- as.list(HPOPARENTS)
```

---

|            |                                           |
|------------|-------------------------------------------|
| HPOSYNONYM | <i>Map from HPO synonyms to HPO terms</i> |
|------------|-------------------------------------------|

---

**Description**

HPOSYNONYM is an R object that provides mapping from HPO synonyms to HPO terms

**Details**

Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 20230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOSYNONYM)
```

---

HPOTERM*Annotation of HPO Identifiers to HPO Terms*

---

**Description**

This data set gives mappings between HPO identifiers and their respective terms.

**Details**

Each HPO identifier is mapped to a HPOTerms object that has 2 slots: do\_id: HPO Identifier; Term: The term for that HPO id

Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 220230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a list
xx <- as.list(HPOTERM)
```

---

HPO\_dbconn*Collect information about the package annotation DB*

---

**Description**

Some convenience functions for getting a connection object to (or collecting information about) the package annotation DB.

**Usage**

```
HPO_dbconn()
HPO_dbfile()
HPO_dbschema(file="", show.indices=FALSE)
HPO_dbInfo()
```

**Arguments**

|                           |                                                                                                                                                           |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>file</code>         | A connection, or a character string naming the file to print to (see the <code>file</code> argument of the <a href="#">cat</a> function for the details). |
| <code>show.indices</code> | The CREATE INDEX statements are not shown by default. Use <code>show.indices=TRUE</code> to get them.                                                     |

**Details**

`HPO_dbconn` returns a connection object to the package annotation DB. **IMPORTANT:** HPOn't call [dbDisconnect](#) on the connection object returned by `HPO_dbconn` or you will break all the [AnnDbObj](#) objects defined in this package!

`HPO_dbfile` returns the path (character string) to the package annotation DB (this is an SQLite file).

`HPO_dbschema` prints the schema definition of the package annotation DB.

`HPO_dbInfo` prints other information about the package annotation DB.

**Value**

`HPO_dbconn`: a `DBIConnection` object representing an open connection to the package annotation DB.

`HPO_dbfile`: a character string with the path to the package annotation DB.

`HPO_dbschema`: none (invisible NULL).

`HPO_dbInfo`: none (invisible NULL).

**See Also**

[dbGetQuery](#), [dbConnect](#), [dbconn](#), [dbfile](#), [dbschema](#), [dbInfo](#)

**Examples**

```
## Count the number of rows in the "hdo_term" table:
HPO_dbconn()
```

---

HPOmetadata

*Annotation of HPO Identifiers to HPO Terms*


---

**Description**

This data set gives mappings between HPO identifiers and their respective terms.

**Details**

Each HPO identifier is mapped to a `HPOTerms` object that has 2 slots: `name`: HPO name; `value`: The value

Mappings were based on data provided by: Human Phenotype Ontology With a date stamp from the source of: 20230405

**Value**

HPO\_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

HPO\_dbfile: a character string with the path to the package annotation DB.

HPO\_dbschema: none (invisible NULL).

HPO\_dbInfo: none (invisible NULL).

**References**

<https://hpo.jax.org/app/data/ontology>

**Examples**

```
# Convert the object to a data.frame
library(AnnotationDbi)
xx <- toTable(HPOmetadata)
```

# Index

- \* **classes**
  - HPOdb-objects, [4](#)
- \* **datasets**
  - HPO.db, [2](#)
  - HPO\_dbconn, [11](#)
  - HPOALIAS, [1](#)
  - HPOANCESTOR, [2](#)
  - HPOCHILDREN, [3](#)
  - HPODO, [4](#)
  - HPOGENE, [6](#)
  - HPOMAPCOUNTS, [7](#)
  - HPOmetadata, [12](#)
  - HPOMPO, [7](#)
  - HPOOBSOLETE, [8](#)
  - HPOOFFSPRING, [9](#)
  - HPOPARENTS, [9](#)
  - HPOSYNONYM, [10](#)
  - HPOTERM, [11](#)
- \* **methods**
  - HPOdb-objects, [4](#)
- \* **utilities**
  - HPO\_dbconn, [11](#)
- AnnDbObj, [12](#)
- cat, [12](#)
- checkMAPCOUNTS, [7](#)
- columns (HPOdb-objects), [4](#)
- columns, HPOdb-method (HPOdb-objects), [4](#)
- count.mappedkeys, [7](#)
- dbconn, [12](#)
- dbConnect, [12](#)
- dbDisconnect, [12](#)
- dbfile, [12](#)
- dbGetQuery, [12](#)
- dbInfo, [12](#)
- dbschema, [12](#)
- HPO (HPO.db), [2](#)
- HPO.db, [2](#)
- HPO\_dbconn, [11](#)
- HPO\_dbfile (HPO\_dbconn), [11](#)
- HPO\_dbInfo (HPO\_dbconn), [11](#)
- HPO\_dbschema (HPO\_dbconn), [11](#)
- HPOALIAS, [1](#)
- HPOANCESTOR, [2](#)
- HPOCHILDREN, [3](#)
- HPOdb-class (HPOdb-objects), [4](#)
- HPOdb-objects, [4](#)
- HPODO, [4](#)
- HPOGENE, [6](#)
- HPOMAPCOUNTS, [7](#)
- HPOmetadata, [12](#)
- HPOMPO, [7](#)
- HPOOBSOLETE, [8](#)
- HPOOFFSPRING, [9](#)
- HPOPARENTS, [9](#)
- HPOSYNONYM, [10](#)
- HPOTERM, [11](#)
- keys (HPOdb-objects), [4](#)
- keys, HPOdb-method (HPOdb-objects), [4](#)
- keytypes (HPOdb-objects), [4](#)
- keytypes, HPOdb-method (HPOdb-objects), [4](#)
- mappedkeys, [7](#)
- select (HPOdb-objects), [4](#)
- select, HPOdb-method (HPOdb-objects), [4](#)