

# AnnotationDbi: Introduction To Bioconductor Annotation Packages

*Marc Carlson*

October 14, 2025



**Figure 1:** Annotation Packages: the big picture

*Bioconductor* provides extensive annotation resources. These can be *gene centric*, or *genome centric*. Annotations can be provided in packages curated by *Bioconductor*, or obtained from web-based resources. This vignette is primarily concerned with describing the annotation resources that are available as packages. More advanced users who wish to learn about how to make new annotation packages should see the vignette titled "Creating select Interfaces for custom Annotation resources" from the *AnnotationForge* package.

Gene centric *AnnotationDbi* packages include:

- Organism level: e.g. *org.Mm.eg.db*.
- Platform level: e.g. *hgu133plus2.db*, *hgu133plus2.probes*, *hgu133plus2.cdf*.
- System-biology level: *GO.db*

Genome centric *GenomicFeatures* packages include

- Transcriptome level: e.g. *TxDb.Hsapiens.UCSC.hg19.knownGene*, *EnsDb.Hsapiens.v75*.
- Generic genome features: Can generate via *GenomicFeatures*

One web-based resource accesses [biomart](#), via the *biomaRt* package:

- Query web-based 'biomart' resource for genes, sequence, SNPs, and etc.

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

The most popular annotation packages have been modified so that they can make use of a new set of methods to more easily access their contents. These four methods are named: `columns`, `keytypes`, `keys` and `select`. And they are described in this vignette. They can currently be used with all chip, organism, and *TxDb* packages along with the popular *GO.db* package.

For the older less popular packages, there are still convenient ways to retrieve the data. The *How to use bimap from the ".db" annotation packages* vignette in the *AnnotationDbi* package is a key reference for learnign about how to use bimap objects.

Finally, all of the '.db' (and most other *Bioconductor* annotation packages) are updated every 6 months corresponding to each release of *Bioconductor*. Exceptions are made for packages where the actual resources that the packages are based on have not themselves been updated.

### 1 AnnotationDb objects and the select method

---

As previously mentioned, a new set of methods have been added that allow a simpler way of extracting identifier based annotations. All the annotation packages that support these new methods expose an object named exactly the same as the package itself. These objects are collectively called *AnntoationDb* objects for the class that they all inherit from. The more specific classes (the ones that you will actually see in the wild) have names like *OrgDb*, *ChipDb* or *TxDb* objects. These names correspond to the kind of package (and underlying schema) being represented. The methods that can be applied to all of these objects are `columns`, `keys`, `keytypes` and `select`.

In addition, another accessor has recently been added which allows extraction of one column at at time. the `mapIds` method allows users to extract data into either a named character vector, a list or even a *SimpleCharacterList*. This method should work with all the different kinds of *AnntoationDb* objects described below.

### 2 ChipDb objects and the select method

---

An extremely common kind of Annotation package is the so called platform based or chip based package type. This package is intended to make the manufacturer labels for a series of probes or probesets to a wide range of gene-based features. A package of this kind will load an *ChipDb* object. Below is a set of examples to show how you might use the standard 4 methods to interact with an object of this type.

First we need to load the package:

```
suppressPackageStartupMessages({  
  library(hgu95av2.db)  
})
```

If we list the contents of this package, we can see that one of the many things loaded is an object named after the package "hgu95av2.db":

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
ls("package:hgu95av2.db")

## [1] "hgu95av2" "hgu95av2.db" "hgu95av2ACCNUM"
## [4] "hgu95av2ALIAS2PROBE" "hgu95av2CHR" "hgu95av2CHRLNGTHS"
## [7] "hgu95av2CHRLLOC" "hgu95av2CHRLLOCEND" "hgu95av2ENSEMBL"
## [10] "hgu95av2ENSEMBL2PROBE" "hgu95av2ENTREZID" "hgu95av2ENZYME"
## [13] "hgu95av2ENZYME2PROBE" "hgu95av2GENENAME" "hgu95av2G0"
## [16] "hgu95av2G02ALLPROBES" "hgu95av2G02PROBE" "hgu95av2MAP"
## [19] "hgu95av2MAPCOUNTS" "hgu95av20MIM" "hgu95av20ORGANISM"
## [22] "hgu95av20RGPKG" "hgu95av2PATH" "hgu95av2PATH2PROBE"
## [25] "hgu95av2PFAM" "hgu95av2PMID" "hgu95av2PMID2PROBE"
## [28] "hgu95av2PROSITE" "hgu95av2REFSEQ" "hgu95av2SYMBOL"
## [31] "hgu95av2UNIPROT" "hgu95av2_dbInfo" "hgu95av2_dbconn"
## [34] "hgu95av2_dbfile" "hgu95av2_dbschema"
```

We can look at this object to learn more about it:

```
hgu95av2.db

## ChipDb object:
## | DBSCHEMAVERSION: 2.1
## | Db type: ChipDb
## | Supporting package: AnnotationDbi
## | DBSCHEMA: HUMANCHIP_DB
## | ORGANISM: Homo sapiens
## | SPECIES: Human
## | MANUFACTURER: Affymetrix
## | CHIPNAME: Affymetrix HG_U95Av2 Array
## | MANUFACTURERURL: http://www.affymetrix.com
## | EGSOURCEDATE: 2021-Apr14
## | EGSOURCENAME: Entrez Gene
## | EGSOURCEURL: ftp://ftp.ncbi.nlm.nih.gov/gene/DATA
## | CENTRALID: ENTREZID
## | TAXID: 9606
## | GOSOURCENAME: Gene Ontology
## | GOSOURCEURL: http://current.geneontology.org/ontology/go-basic.obo
## | GOSOURCEDATE: 2021-02-01
## | GOEGSOURCEDATE: 2021-Apr14
## | GOEGSOURCENAME: Entrez Gene
## | GOEGSOURCEURL: ftp://ftp.ncbi.nlm.nih.gov/gene/DATA
## | KEGGSOURCENAME: KEGG GENOME
## | KEGGSOURCEURL: ftp://ftp.genome.jp/pub/kegg/genomes
## | KEGGSOURCEDATE: 2011-Mar15
## | GPSOURCENAME: UCSC Genome Bioinformatics (Homo sapiens)
## | GPSOURCEURL:
## | GPSOURCEDATE: 2021-Feb16
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## | ENSOURCEDATE: 2021-Feb16
## | ENSOURCENAME: Ensembl
## | ENSOURCEURL: ftp://ftp.ensembl.org/pub/current_fasta
## | UPSOURCENAME: Uniprot
## | UPSOURCEURL: http://www.UniProt.org/
## | UPSOURCEDATE: Mon Apr 26 21:53:12 2021

##
## Please see: help('select') for usage information
```

If we want to know what kinds of data are retrieveable via `select`, then we should use the `columns` method like this:

```
columns(hgu95av2.db)

## [1] "ACCNUM"      "ALIAS"       "ENSEMBL"     "ENSEMBLPROT"
## [5] "ENSEMBLTRANS" "ENTREZID"    "ENZYME"      "EVIDENCE"
## [9] "EVIDENCEALL" "GENENAME"    "GENETYPE"    "GO"
## [13] "GOALL"       "IPI"         "MAP"         "OMIM"
## [17] "ONTOLOGY"    "ONTOLOGYALL" "PATH"        "PFAM"
## [21] "PMID"        "PROBEID"     "PROSITE"     "REFSEQ"
## [25] "SYMBOL"      "UCSCKG"      "UNIPROT"
```

If we are further curious to know more about those values for columns, we can consult the help pages. Asking about any of these values will pull up a manual page describing the different fields and what they mean.

```
help("SYMBOL")
```

If we are curious about what kinds of fields we could potentially use as keys to query the database, we can use the `keytypes` method. In a perfect world, this method will return values very similar to what was returned by `columns`, but in reality, some kinds of values make poor keys and so this list is often shorter.

```
keytypes(hgu95av2.db)

## [1] "ACCNUM"      "ALIAS"       "ENSEMBL"     "ENSEMBLPROT"
## [5] "ENSEMBLTRANS" "ENTREZID"    "ENZYME"      "EVIDENCE"
## [9] "EVIDENCEALL" "GENENAME"    "GENETYPE"    "GO"
## [13] "GOALL"       "IPI"         "MAP"         "OMIM"
## [17] "ONTOLOGY"    "ONTOLOGYALL" "PATH"        "PFAM"
## [21] "PMID"        "PROBEID"     "PROSITE"     "REFSEQ"
## [25] "SYMBOL"      "UCSCKG"      "UNIPROT"
```

If we want to extract some sample keys of a particular type, we can use the `keys` method.

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
head(keys(hgu95av2.db, keytype="SYMBOL"))  
## [1] "A1BG"      "A2M"      "NAT1"     "NAT2"     "NATP"     "SERPINA3"
```

And finally, if we have some keys, we can use `select` to extract them. By simply using appropriate argument values with `select` we can specify what keys we want to look up values for (keys), what we want returned back (columns) and the type of keys that we are passing in (keytype)

```
#1st get some example keys  
k <- head(keys(hgu95av2.db, keytype="PROBEID"))  
# then call select  
select(hgu95av2.db, keys=k, columns=c("SYMBOL", "GENENAME"), keytype="PROBEID")  
  
## 'select()' returned 1:1 mapping between keys and columns  
  
##      PROBEID  SYMBOL  
## 1   1000_at   MAPK3  
## 2   1001_at   TIE1  
## 3  1002_f_at  CYP2C19  
## 4  1003_s_at  CXCR5  
## 5   1004_at  CXCR5  
## 6   1005_at  DUSP1  
##  
##                                     GENENAME  
## 1                                     mitogen-activated protein kinase 3  
## 2 tyrosine kinase with immunoglobulin like and EGF like domains 1  
## 3                                     cytochrome P450 family 2 subfamily C member 19  
## 4                                     C-X-C motif chemokine receptor 5  
## 5                                     C-X-C motif chemokine receptor 5  
## 6                                     dual specificity phosphatase 1
```

And as you can see, when you call the code above, `select` will try to return a data.frame with all the things you asked for matched up to each other.

Finally if you wanted to extract only one column of data you could instead use the `mapIds` method like this:

```
#1st get some example keys  
k <- head(keys(hgu95av2.db, keytype="PROBEID"))  
# then call mapIds  
mapIds(hgu95av2.db, keys=k, column=c("GENENAME"), keytype="PROBEID")  
  
## 'select()' returned 1:1 mapping between keys and columns  
  
##                                     1000_at  
##                                     "mitogen-activated protein kinase 3"  
##                                     1001_at  
## "tyrosine kinase with immunoglobulin like and EGF like domains 1"
```

```
##                                1002_f_at
##      "cytochrome P450 family 2 subfamily C member 19"
##                                1003_s_at
##      "C-X-C motif chemokine receptor 5"
##                                1004_at
##      "C-X-C motif chemokine receptor 5"
##                                1005_at
##      "dual specificity phosphatase 1"
```

## 3 OrgDb objects and the select method

An organism level package (an 'org' package) uses a central gene identifier (e.g. Entrez Gene id) and contains mappings between this identifier and other kinds of identifiers (e.g. GenBank or Uniprot accession number, RefSeq id, etc.). The name of an org package is always of the form *org.<Ab>.<id>.db* (e.g. *org.Sc.sgd.db*) where *<Ab>* is a 2-letter abbreviation of the organism (e.g. *Sc* for *Saccharomyces cerevisiae*) and *<id>* is an abbreviation (in lower-case) describing the type of central identifier (e.g. *sgd* for gene identifiers assigned by the Saccharomyces Genome Database, or *eg* for Entrez Gene ids).

Just as the chip packages load a *ChipDb* object, the org packages will load a *OrgDb* object. The following exercise should acquaint you with the use of these methods in the context of an organism package.

### Exercise 1

Display the *OrgDb* object for the *org.Hs.eg.db* package.

Use the *columns* method to discover which sorts of annotations can be extracted from it. Is this the same as the result from the *keytypes* method? Use the *keytypes* method to find out.

Finally, use the *keys* method to extract UNIPROT identifiers and then pass those keys in to the *select* method in such a way that you extract the gene symbol and KEGG pathway information for each. Use the help system as needed to learn which values to pass in to *columns* in order to achieve this.

### Solution:

```
library(org.Hs.eg.db)
columns(org.Hs.eg.db)

## [1] "ACCNUM"      "ALIAS"      "ENSEMBL"    "ENSEMBLPROT"
## [5] "ENSEMBLTRANS" "ENTREZID"   "ENZYME"     "EVIDENCE"
## [9] "EVIDENCEALL" "GENENAME"   "GENETYPE"   "GO"
## [13] "GOALL"      "IPI"        "MAP"        "OMIM"
## [17] "ONTOLOGY"   "ONTOLOGYALL" "PATH"       "PFAM"
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## [21] "PMID"          "PROSITE"        "REFSEQ"         "SYMBOL"
## [25] "UCSCKG"        "UNIPROT"
```

```
help("SYMBOL") ## for explanation of these columns and keytypes values
```

```
keytypes(org.Hs.eg.db)
```

```
## [1] "ACCNUM"        "ALIAS"          "ENSEMBL"        "ENSEMBLPROT"
## [5] "ENSEMBLTRANS" "ENTREZID"       "ENZYME"         "EVIDENCE"
## [9] "EVIDENCEALL"   "GENENAME"       "GENETYPE"       "GO"
## [13] "GOALL"         "IPI"           "MAP"           "OMIM"
## [17] "ONTOLOGY"      "ONTOLOGYALL"   "PATH"          "PFAM"
## [21] "PMID"          "PROSITE"        "REFSEQ"         "SYMBOL"
## [25] "UCSCKG"        "UNIPROT"
```

```
uniKeys <- head(keys(org.Hs.eg.db, keytype="UNIPROT"))
```

```
cols <- c("SYMBOL", "PATH")
```

```
select(org.Hs.eg.db, keys=uniKeys, columns=cols, keytype="UNIPROT")
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
##   UNIPROT SYMBOL PATH
## 1  A8K052   A1BG <NA>
## 2  P04217   A1BG <NA>
## 3  Q68CK0   A1BG <NA>
## 4  Q8IYJ6   A1BG <NA>
## 5  Q96P39   A1BG <NA>
## 6  V9HWD8   A1BG <NA>
```

So how could you use select to annotate your results? This next exercise should help you to understand how that should generally work.

### Exercise 2

Please run the following code snippet (which will load a fake data result that I have provided for the purposes of illustration):

```
load(system.file("extdata", "resultTable.Rda", package="AnnotationDbi"))
head(resultTable)
```

```
##           logConc    logFC LR.statistic      PValue      FDR
## 100418920 -9.639471 -4.679498    378.0732 3.269307e-84 2.613484e-80
## 100419779 -10.638865 -4.264830    291.1028 2.859424e-65 1.142912e-61
## 100271867 -11.448981 -4.009603    222.3653 2.757135e-50 7.346846e-47
## 100287169 -11.026699 -3.486593    206.7771 6.934967e-47 1.385953e-43
## 100287735 -11.036862  3.064980    204.1235 2.630432e-46 4.205535e-43
## 100421986 -12.276297 -4.695736    190.5368 2.427556e-43 3.234314e-40
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

*The rownames of this table happen to provide entrez gene identifiers for each row (for human). Find the gene symbol and gene name for each of the rows in resultTable and then use the merge method to attach those annotations to it.*

### Solution:

```
annots <- select(org.Hs.eg.db, keys=rownames(resultTable),
                 columns=c("SYMBOL", "GENENAME"), keytype="ENTREZID")

## 'select()' returned 1:1 mapping between keys and columns

resultTable <- merge(resultTable, annots, by.x=0, by.y="ENTREZID")
head(resultTable)
```

##	Row.names	logConc	logFC	LR.statistic	PValue	FDR
## 1	100127888	-10.57050	2.758937	182.8937	1.131473e-41	1.130624e-38
## 2	100131223	-12.37808	-4.654318	179.2331	7.126423e-41	6.329847e-38
## 3	100271381	-12.06340	3.511937	188.4824	6.817155e-43	7.785191e-40
## 4	100271867	-11.44898	-4.009603	222.3653	2.757135e-50	7.346846e-47
## 5	100287169	-11.02670	-3.486593	206.7771	6.934967e-47	1.385953e-43
## 6	100287735	-11.03686	3.064980	204.1235	2.630432e-46	4.205535e-43
##	SYMBOL				GENENAME	
## 1	SLC04A1-AS1			SLC04A1 antisense RNA 1		
## 2	ARL8BP2			ARL8B pseudogene 2		
## 3	RPS28P8			ribosomal protein S28 pseudogene 8		
## 4	MPVQTL1			Mean platelet volume QTL1		
## 5	<NA>			<NA>		
## 6	TTY13B			testis-specific transcript, Y-linked 13B		

## 4 Using select with GO.db

When you load the GO.db package, a *GODb* object is also loaded. This allows you to use the `columns`, `keys`, `keytypes` and `select` methods on the contents of the GO ontology. So if for example, you had a few GO IDs and wanted to know more about it, you could do it like this:

```
library(GO.db)
GOIDs <- c("GO:0042254", "GO:0044183")
select(GO.db, keys=GOIDs, columns="DEFINITION", keytype="GOID")

## 'select()' returned 1:1 mapping between keys and columns

##      GOID
## 1 GO:0042254
## 2 GO:0044183
##
```



```
## 1 A cellular process that results in the biosynthesis of constituent macromolecules, assembly, a
## 2
```

## 5 Using select with TxDb packages

A *TxDb* package (a 'TxDb' package) connects a set of genomic coordinates to various transcript oriented features. The package can also contain Identifiers to features such as genes and transcripts, and the internal schema describes the relationships between these different elements. All TxDb containing packages follow a specific naming scheme that tells where the data came from as well as which build of the genome it comes from.

### Exercise 3

Display the TxDb object for the [TxDb.Hsapiens.UCSC.hg19.knownGene](#) package.

As before, use the `columns` and `keytypes` methods to discover which sorts of annotations can be extracted from it.

Use the `keys` method to extract just a few gene identifiers and then pass those keys in to the `select` method in such a way that you extract the transcript ids and transcript starts for each.

### Solution:

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)

## Loading required package: GenomicFeatures
## Loading required package: Seqinfo
## Loading required package: GenomicRanges

txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
txdb

## TxDb object:
## # Db type: TxDb
## # Supporting package: GenomicFeatures
## # Data source: UCSC
## # Genome: hg19
## # Organism: Homo sapiens
## # Taxonomy ID: 9606
## # UCSC Table: knownGene
## # UCSC Track: GENCODE V48lift37
## # Resource URL: https://genome.ucsc.edu/
## # Type of Gene ID: Entrez Gene ID
## # Full dataset: yes
## # Nb of transcripts: 381987
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## # Db created by: txdbmaker package from Bioconductor
## # Creation time: 2025-10-13 16:17:01 -0700 (Mon, 13 Oct 2025)
## # txdbmaker version at creation time: 1.5.6
## # RSQLite version at creation time: 2.4.3
## # DBSCHEMAVERSION: 1.2

columns(txdb)

## [1] "CDSCHROM" "CSEND" "CDSID" "CDSNAME" "CDSPHASE"
## [6] "CDSSTART" "CDSSTRAND" "EXONCHROM" "EXONEND" "EXONID"
## [11] "EXONNAME" "EXONRANK" "EXONSTART" "EXONSTRAND" "GENEID"
## [16] "TXCHROM" "TXEND" "TXID" "TXNAME" "TXSTART"
## [21] "TXSTRAND" "TXTYPE"

keytypes(txdb)

## [1] "CDSID" "CDSNAME" "EXONID" "EXONNAME" "GENEID" "TXID"
## [7] "TXNAME"

keys <- head(keys(txdb, keytype="GENEID"))
cols <- c("TXID", "TXSTART")
select(txdb, keys=keys, columns=cols, keytype="GENEID")

## 'select()' returned 1:many mapping between keys and columns

##      GENEID  TXID  TXSTART
## 1         1 346592 58856544
## 2         1 346593 58856549
## 3         1 346594 58856549
## 4         1 346595 58856549
## 5         1 346596 58858220
## 6         1 346597 58858224
## 7         1 346598 58858226
## 8         1 346599 58859832
## 9         1 346600 58861960
## 10        1 346601 58864387
## 11        10 164980 18248792
## 12        10 164981 18248797
## 13       100 354325 43213537
## 14       100 354326 43247246
## 15       100 354327 43247259
## 16       100 354328 43247266
## 17       100 354329 43248162
## 18       100 354330 43248163
## 19       100 354331 43248163
## 20       100 354332 43248163
## 21       100 354333 43248163
## 22       100 354334 43248163
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## 23      100 354335 43248163
## 24      100 354336 43248163
## 25      100 354337 43248163
## 26      100 354338 43248163
## 27      100 354339 43248163
## 28      100 354340 43248163
## 29      100 354341 43248163
## 30      100 354342 43248163
## 31      100 354343 43248163
## 32      100 354344 43248163
## 33      100 354345 43248163
## 34      100 354346 43248163
## 35      100 354347 43248163
## 36      100 354348 43248163
## 37      100 354349 43248163
## 38      100 354350 43248163
## 39      100 354351 43248163
## 40      100 354352 43248163
## 41      100 354353 43248163
## 42      100 354354 43248163
## 43      100 354355 43248163
## 44      100 354356 43248163
## 45      100 354357 43248163
## 46      100 354358 43248163
## 47      100 354359 43248164
## 48      100 354360 43248165
## 49      100 354361 43248166
## 50      100 354362 43248167
## 51      100 354363 43248167
## 52      100 354364 43248170
## 53      100 354365 43248179
## 54      100 354366 43248179
## 55      100 354367 43248181
## 56      100 354368 43248193
## 57      100 354369 43248194
## 58      100 354370 43248194
## 59      100 354371 43248196
## 60      100 354372 43248201
## 61      100 354373 43248204
## 62      100 354374 43248204
## 63      100 354375 43248237
## 64      100 354376 43248247
## 65      100 354377 43248297
## 66      100 354378 43248451
## 67      100 354379 43248478
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## 68      100 354380 43251648
## 69      100 354381 43252393
## 70      100 354382 43252613
## 71      100 354383 43254964
## 72      100 354384 43256188
## 73      100 354385 43264712
## 74     1000 326517 25512843
## 75     1000 326518 25513798
## 76     1000 326519 25530930
## 77     1000 326520 25531045
## 78     1000 326521 25531176
## 79     1000 326522 25531619
## 80     1000 326523 25531728
## 81     1000 326524 25531885
## 82     1000 326525 25542619
## 83     1000 326526 25562927
## 84     1000 326527 25585934
## 85    10000  32798 243651535
## 86    10000  32799 243663021
## 87    10000  32800 243663026
## 88    10000  32801 243663052
## 89    10000  32802 243663065
## 90    10000  32803 243663075
## 91    10000  32804 243663536
## 92    10000  32805 243665084
## 93    10000  32806 243665098
## 94    10000  32807 243668551
## 95    10000  32808 243668551
## 96    10000  32809 243668551
## 97    10000  32810 243773808
## 98    10000  32812 243809181
## 99    10000  32813 243820838
## 100    10000  32814 243820854
## 101    10000  32815 243821815
## 102    10000  32816 243828074
## 103    10000  32817 243850739
## 104    10000  32818 243850931
## 105    10000  32819 243858893
## 106 100008586 370503 49315881
## 107 100008586 370505 49325445
```

As is widely known, in addition to providing access via the `select` method, *TxDb* objects also provide access via the more familiar `transcripts`, `exons`, `cds`, `transcriptsBy`, `exonsBy` and `cdsBy` methods. For those who do not yet know about these other methods, more can be learned by seeing the vignette called: *Making and Utilizing TxDb Objects* in the *GenomicFeatures* package.

## 6 Using select with EnsDb packages

Similar to the *TxDb* objects/packages discussed in the previous section, *EnsDb* objects/packages provide genomic coordinates of gene models along with additional annotations (e.g. gene names, biotypes etc) but are tailored to annotations provided by Ensembl. The central methods `columns`, `keys`, `keytypes` and `select` are all implemented for *EnsDb* objects. In addition, these methods allow also the use of the *EnsDb* specific filtering framework to retrieve only selected information from the database (see vignette of the *ensemldb* package for more information).

In the example below we first evaluate which columns, keys and keytypes are available for *EnsDb* objects and fetch then the transcript ids, genomic start coordinate and transcript biotype for some genes.

```
library(EnsDb.Hsapiens.v75)

## Loading required package: ensemblDb
## Loading required package: AnnotationFilter
##
## Attaching package: 'ensemldb'
##
## The following object is masked from 'package:stats':
##
##   filter

edb <- EnsDb.Hsapiens.v75
edb

## EnsDb for Ensembl:
## |Backend: SQLite
## |Db type: EnsDb
## |Type of Gene ID: Ensembl Gene ID
## |Supporting package: ensemblDb
## |Db created by: ensemblDb package from Bioconductor
## |script_version: 0.3.0
## |Creation time: Thu May 18 09:15:45 2017
## |ensembl_version: 75
## |ensembl_host: localhost
## |Organism: homo_sapiens
## |taxonomy_id: 9606
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## |genome_build: GRCh37
## |DBSCHEMAVERSION: 2.0
## | No. of genes: 64102.
## | No. of transcripts: 215647.
## |Protein data available.

## List all columns
columns(edb)

## [1] "ENTREZID"          "EXONID"          "EXONIDX"
## [4] "EXONSEQEND"        "EXONSEQSTART"    "GENEBIOTYPE"
## [7] "GENEID"            "GENENAME"        "GENESEQEND"
## [10] "GENESEQSTART"      "INTERPROACCESSION" "ISCIRCULAR"
## [13] "PROTDOMEND"        "PROTDOMSTART"    "PROTEINDOMAINID"
## [16] "PROTEINDOMAINSOURCE" "PROTEINID"        "PROTEINSEQUENCE"
## [19] "SEQCOORDSYSTEM"    "SEQLength"       "SEQNAME"
## [22] "SEQSTRAND"         "SYMBOL"          "TXBIOTYPE"
## [25] "TXCDSSEQEND"       "TXCDSSEQSTART"   "TXID"
## [28] "TXNAME"            "TXSEQEND"        "TXSEQSTART"
## [31] "UNIPROTDB"         "UNIPROTID"       "UNIPROTMAPPINGTYPE"

## List all keytypes
keytypes(edb)

## [1] "ENTREZID"          "EXONID"          "GENEBIOTYPE"
## [4] "GENEID"            "GENENAME"        "PROTDOMID"
## [7] "PROTEINDOMAINID"   "PROTEINDOMAINSOURCE" "PROTEINID"
## [10] "SEQNAME"           "SEQSTRAND"       "SYMBOL"
## [13] "TXBIOTYPE"         "TXID"            "TXNAME"
## [16] "UNIPROTID"

## Get the first
keys <- head(keys(edb, keytype="GENEID"))

## Get the data
select(edb, keys=keys, columns=c("TXID", "TXSEQSTART", "TXBIOTYPE"),
       keytype="GENEID")

##           GENEID          TXID TXSEQSTART          TXBIOTYPE
## 1 ENSG000000000003 ENST00000373020  99883667    protein_coding
## 2 ENSG000000000003 ENST00000496771  99887538 processed_transcript
## 3 ENSG000000000003 ENST00000494424  99888439 processed_transcript
## 4 ENSG000000000005 ENST00000373031  99839799    protein_coding
## 5 ENSG000000000005 ENST00000485971  99848621 processed_transcript
## 6 ENSG000000000419 ENST00000371588  49551404    protein_coding
## 7 ENSG000000000419 ENST00000466152  49551404 processed_transcript
## 8 ENSG000000000419 ENST00000371582  49551404    protein_coding
## 9 ENSG000000000419 ENST00000494752  49551433 processed_transcript
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## 10 ENSG00000000419 ENST00000371584 49551482 protein_coding
## 11 ENSG00000000419 ENST00000413082 49552685 protein_coding
## 12 ENSG00000000419 ENST00000371583 49551490 protein_coding
## 13 ENSG00000000457 ENST00000367771 169818772 protein_coding
## 14 ENSG00000000457 ENST00000367770 169822215 protein_coding
## 15 ENSG00000000457 ENST00000423670 169823652 protein_coding
## 16 ENSG00000000457 ENST00000470238 169828260 processed_transcript
## 17 ENSG00000000457 ENST00000367772 169821804 protein_coding
## 18 ENSG00000000460 ENST00000498289 169631245 processed_transcript
## 19 ENSG00000000460 ENST00000472795 169763871 processed_transcript
## 20 ENSG00000000460 ENST00000496973 169764184 processed_transcript
## 21 ENSG00000000460 ENST00000481744 169764184 processed_transcript
## 22 ENSG00000000460 ENST00000359326 169764190 protein_coding
## 23 ENSG00000000460 ENST00000466580 169764220 processed_transcript
## 24 ENSG00000000460 ENST00000459772 169764220 processed_transcript
## 25 ENSG00000000460 ENST00000286031 169764550 protein_coding
## 26 ENSG00000000460 ENST00000456684 169764220 protein_coding
## 27 ENSG00000000460 ENST00000413811 169764181 protein_coding
## 28 ENSG00000000938 ENST00000374005 27938575 protein_coding
## 29 ENSG00000000938 ENST00000374004 27938811 protein_coding
## 30 ENSG00000000938 ENST00000374003 27939180 protein_coding
## 31 ENSG00000000938 ENST00000457296 27942025 protein_coding
## 32 ENSG00000000938 ENST00000475472 27949895 processed_transcript
## 33 ENSG00000000938 ENST00000468038 27949918 processed_transcript
## 34 ENSG00000000938 ENST00000545953 27938803 protein_coding
## 35 ENSG00000000938 ENST00000399173 27938803 protein_coding
```

We can modify the queries above to retrieve only genes encoded on chromosome Y. To this end we use filter objects available for *EnsDb* objects and its methods.

```
## Retrieve all gene IDs of all lincRNAs encoded on chromosome Y
linkY <- keys(edb,
  filter=list(GeneBiotypeFilter("lincRNA"), SeqNameFilter("Y")))
length(linkY)

## [1] 48

## We get now all transcripts for these genes.
txs <- select(edb, keys=linkY, columns=c("TXID", "TXSEQSTART", "TXBIOTYPE"),
  keytype="GENEID")
nrow(txs)

## [1] 66

## Alternatively, we could specify/pass the filters with the keys argument.
txs <- select(edb, keys=list(GeneBiotypeFilter("lincRNA"), SeqNameFilter("Y")),
  columns=c("TXID", "TXSEQSTART", "TXBIOTYPE"))
```

## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
## Note: ordering of the results might not match ordering of keys!  
nrow(txs)  
## [1] 66
```

The version number of R and packages loaded for generating the vignette were:

```
## R version 4.5.1 Patched (2025-09-10 r88807)  
## Platform: x86_64-apple-darwin20  
## Running under: macOS Monterey 12.7.6  
##  
## Matrix products: default  
## BLAS: /Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRblas.0.dylib  
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRlapack.dylib; LAP  
##  
## locale:  
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8  
##  
## time zone: America/New_York  
## tzcode source: internal  
##  
## attached base packages:  
## [1] stats4 stats graphics grDevices utils datasets methods  
## [8] base  
##  
## other attached packages:  
## [1] EnsDb.Hsapiens.v75_2.99.0  
## [2] ensemblDb_2.33.2  
## [3] AnnotationFilter_1.33.0  
## [4] TxDb.Hsapiens.UCSC.hg19.knownGene_3.22.1  
## [5] GenomicFeatures_1.61.6  
## [6] GenomicRanges_1.61.5  
## [7] Seqinfo_0.99.2  
## [8] G0.db_3.22.0  
## [9] hgu95av2.db_3.13.0  
## [10] AnnotationForge_1.51.0  
## [11] org.Hs.eg.db_3.22.0  
## [12] AnnotationDbi_1.71.2  
## [13] IRanges_2.43.5  
## [14] S4Vectors_0.47.4  
## [15] Biobase_2.69.1  
## [16] BiocGenerics_0.55.3  
## [17] generics_0.1.4  
## [18] DBI_1.2.3  
## [19] knitr_1.50
```



## AnnotationDbi: Introduction To Bioconductor Annotation Packages

```
##
## loaded via a namespace (and not attached):
## [1] KEGGREST_1.49.2          SummarizedExperiment_1.39.2
## [3] rjson_0.2.23             xfun_0.53
## [5] lattice_0.22-7          vctrs_0.6.5
## [7] tools_4.5.1             bitops_1.0-9
## [9] curl_7.0.0              parallel_4.5.1
## [11] RSQLite_2.4.3           highr_0.11
## [13] blob_1.2.4              pkgconfig_2.0.3
## [15] Matrix_1.7-4            compiler_4.5.1
## [17] Rsamtools_2.25.3        Biostings_2.77.2
## [19] BiocStyle_2.37.1        codetools_0.2-20
## [21] GenomeInfoDb_1.45.12    htmltools_0.5.8.1
## [23] lazyeval_0.2.2         RCurl_1.98-1.17
## [25] yaml_2.3.10            crayon_1.5.3
## [27] BiocParallel_1.43.4     cachem_1.1.0
## [29] DelayedArray_0.35.3     abind_1.4-8
## [31] digest_0.6.37          restfulr_0.0.16
## [33] fastmap_1.2.0          grid_4.5.1
## [35] cli_3.6.5              SparseArray_1.9.1
## [37] S4Arrays_1.9.1         XML_3.99-0.19
## [39] UCSC.utils_1.5.0       bit64_4.6.0-1
## [41] rmarkdown_2.30         XVector_0.49.1
## [43] httr_1.4.7            matrixStats_1.5.0
## [45] bit_4.6.0             png_0.1-8
## [47] memoise_2.0.1         evaluate_1.0.5
## [49] BiocIO_1.19.0         rtracklayer_1.69.1
## [51] rlang_1.1.6           BiocManager_1.30.26
## [53] jsonlite_2.0.0        R6_2.6.1
## [55] ProtGenerics_1.41.0    MatrixGenerics_1.21.0
## [57] GenomicAlignments_1.45.4
```