Genome into R

Ugo Borello, Eleonora Sara Bacci

2025-09-14

How do we access genomic data with R/Bioconductor?

Same R/Bioconductor packages store genomic data. Genomic information is stored as DNA sequence and genomic annotation. There are different public repositories from which these data is accessible using R/Bioconductor packages. Each repository has its own way to retrieve the data they store; we need to be very careful!

For this tutorial we will focus on the *Homo sapiens* genome; let's dive in!

First of all, we need to load some libraries.

```
library(org.Hs.eg.db)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(dbplyr)
library(Gviz)
library(plyranges)
library(ensembldb)
library(AnnotationDbi)
```

NCBI repository and the org.*.db package

One way to access the human genome annotation is through the `org.Hs.eg.db` package. This is a special R object that contains the annotation of the human genome that came from the NCBI Entrez database. It is an organism specific and gene-centric package! The name of the package give us some basic and useful information about its content: 'org' stand for organism, 'Hs' for Homo sapiens, 'eg' means that the gene identifier is the Entrez ID, and 'db' means that it is structured as a database.

```
org_hs <- org.Hs.eg.db
org_hs
```

```
## OrgDb object:
## | DBSCHEMAVERSION: 2.1
## | Db type: OrgDb
## | Supporting package: AnnotationDbi
## | DBSCHEMA: HUMAN_DB
## | ORGANISM: Homo sapiens
## | SPECIES: Human
## | EGSOURCEDATE: 2023-Sep11
## | EGSOURCENAME: Entrez Gene
## | EGSOURCEURL: ftp://ftp.ncbi.nlm.nih.gov/gene/DATA
## | CENTRALID: EG
## | TAXID: 9606
## | GOSOURCENAME: Gene Ontology
## | GOSOURCEURL: http://current.geneontology.org/ontology/go-basic.obo
## | GOSOURCEDATE: 2023-07-27
```

```
## | GOEGSOURCEDATE: 2023-Sep11
## | 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: 2023-Aug20
## | ENSOURCEDATE: 2023-May10
## | ENSOURCENAME: Ensembl
## | ENSOURCEURL: ftp://ftp.ensembl.org/pub/current_fasta
## | UPSOURCENAME: Uniprot
## | UPSOURCEURL: http://www.UniProt.org/
## | UPSOURCEDATE: Mon Sep 18 16:12:39 2023
```

We just created an object that contains the org.Hs.eg.db data.

`columns`, `keys`, `keytypes` and `select` are methods which we can use to access the data stored in the package. To know what kinds of data are retrievable *via* `select`, we shall use the `columns` method. To identify the fields we could potentially use as keys to query the database, we shall use the `keytypes` method. This method not always return values similar to what is shown by columns because some values make poor keys.

```
keytypes(org_hs)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
## [16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [26] "UNIPROT"
```

```
columns(org_hs)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
## [16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [26] "UNIPROT"
```

Let's test with a single gene

RELN is the acronym which identifies a specific gene coding for REELIN protein. Let's use this information to mine into the 'org.Hs.eg.db' object.

```
acr <- "RELN"

reln_info <- select(org_hs, keys=acr,
                    columns= c("ENSEMBL", "ALIAS", "GENENAME", "ENTREZID"),
                    keytype="SYMBOL")
```

```
reln_info
```

```
##   SYMBOL         ENSEMBL   ALIAS GENENAME ENTREZID
## 1   RELN ENSG00000189056    ETL7   reelin     5649
## 2   RELN ENSG00000189056    LIS2   reelin     5649
## 3   RELN ENSG00000189056 PRO1598   reelin     5649
## 4   RELN ENSG00000189056      RL   reelin     5649
## 5   RELN ENSG00000189056    RELN   reelin     5649
```

In the NCBI database every gene is identified by a unique ID, the ENTREZID: for *RELN* gene this ID is 5649. (https://www.ncbi.nlm.nih.gov/datasets/gene/5649/)

The ENTREZID allows us to mine into different genomic repository.

WATCH OUT! Not in all R annotation packages the ENTREZID is called ENTREZID!!!

UCSC repository and TxDb. package

The 'TxDb.Hsapiens.UCSC.hg19.knownGene' package is an annotation databases generated from the UCSC repository. A TxDb object stores the genomic positions of the 5' and 3' untranslated regions (UTRs), protein coding sequences (CDSs), and exons for a set of mRNA transcripts. The genomic positions are stored and reported with respect to a given genome assembly. (from Bioconductor https://bioconductor.org/packages/devel/bioc/vignettes/GenomicFeatures/inst/doc/GenomicFeatures.html)

In the UCSC repository the ENTREZID are called GENEID!

So we can see if our *RELN* gene is in this database or not.

First we need to create the object.

```
hg19_txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
hg19_txdb
```

```
## TxDb object:
## # Db type: TxDb
## # Supporting package: GenomicFeatures
## # Data source: UCSC
## # Genome: hg19
## # Organism: Homo sapiens
## # Taxonomy ID: 9606
## # UCSC Table: knownGene
## # Resource URL: http://genome.ucsc.edu/
## # Type of Gene ID: Entrez Gene ID
## # Full dataset: yes
## # miRBase build ID: GRCh37
## # transcript_nrow: 82960
## # exon_nrow: 289969
## # cds_nrow: 237533
## # Db created by: GenomicFeatures package from Bioconductor
## # Creation time: 2015-10-07 18:11:28 +0000 (Wed, 07 Oct 2015)
## # GenomicFeatures version at creation time: 1.21.30
## # RSQLite version at creation time: 1.0.0
## # DBSCHEMAVERSION: 1.1
```

We can use the same function as before and see if columns and keytype do completely match again.

```
keytypes(hg19_txdb)
```

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

```
columns(hg19_txdb)
```

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

In this case, there are fewer keys than columns!

What about the *RELN* gene here? We know already that its ENTREZID is 5649 and we can use this information to mine into this annotation package.

```
eid <- "5649"
txid <- select(hg19_txdb, eid, "TXNAME", "GENEID")

txid
```

```
##   GENEID     TXNAME
## 1   5649 uc010liz.3
## 2   5649 uc022ajq.1
## 3   5649 uc022ajr.1
```

Sanity check: ENTREZID and GENEID are the same!

But which information is stored in the TXNAME? Those are the names for the transcript of the *RELN* gene.

CDS and exons

Every gene in the genome is identified by a DNA sequence containig regulatory regions and coding region. The coding region is the sum of the gene exons, coding for a gene product. `CDS` stands for Coding DNA Sequence and it is the gene sequence translated into a product, generally a protein.

cdsBy() is used to extract gene coding sequences (CDS) from a genomic annotation object (such as our TxDb object), grouped by a specific feature such as gene or transcript.

```
cds <- cdsBy(hg19_txdb, by="tx", use.names=TRUE) #grouped by transcripts
head(cds)
```

```
## GRangesList object of length 6:
## $uc010nxq.1
## GRanges object with 3 ranges and 3 metadata columns:
##       seqnames      ranges strand |    cds_id    cds_name exon_rank
##          <Rle>   <IRanges>  <Rle> | <integer> <character> <integer>
##   [1]     chr1 12190-12227      + |         1        <NA>         1
##   [2]     chr1 12595-12721      + |         2        <NA>         2
##   [3]     chr1 13403-13639      + |         3        <NA>         3
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
##
## $uc001aal.1
## GRanges object with 1 range and 3 metadata columns:
##       seqnames      ranges strand |    cds_id    cds_name exon_rank
##          <Rle>   <IRanges>  <Rle> | <integer> <character> <integer>
##   [1]     chr1 69091-70008      + |         4        <NA>         1
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
##
## $uc009vjk.2
## GRanges object with 2 ranges and 3 metadata columns:
##       seqnames        ranges strand |    cds_id    cds_name exon_rank
##          <Rle>     <IRanges>  <Rle> | <integer> <character> <integer>
##   [1]     chr1 324343-324345      + |         5        <NA>         2
##   [2]     chr1 324439-325605      + |         6        <NA>         3
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
##
## ...
## <3 more elements>
```

We previously selected only the *RELN* gene transcripts. We can use them to obtain the *RELN* cds.

```
reln_tx<- cds[names(cds) %in% txid$TXNAME]

reln_tx
```

```
## GRangesList object of length 3:
## $uc010liz.3
## GRanges object with 64 ranges and 3 metadata columns:
##        seqnames              ranges strand |    cds_id    cds_name exon_rank
##           <Rle>           <IRanges>  <Rle> | <integer> <character> <integer>
##    [1]     chr7 103629578-103629803      - |     93139        <NA>         1
##    [2]     chr7 103557522-103557632      - |     93138        <NA>         2
##    [3]     chr7 103473984-103474119      - |     93137        <NA>         3
##    [4]     chr7 103417004-103417074      - |     93136        <NA>         4
##    [5]     chr7 103393629-103393661      - |     93135        <NA>         5
##    ...      ...                 ...    ... .       ...         ...       ...
##   [60]     chr7 103130189-103130346      - |     93080        <NA>        60
##   [61]     chr7 103126644-103126863      - |     93079        <NA>        61
##   [62]     chr7 103124100-103124297      - |     93078        <NA>        62
##   [63]     chr7 103123320-103123418      - |     93077        <NA>        63
##   [64]     chr7 103113259-103113355      - |     93073        <NA>        64
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
##
## $uc022ajq.1
## GRanges object with 65 ranges and 3 metadata columns:
##        seqnames              ranges strand |    cds_id    cds_name exon_rank
##           <Rle>           <IRanges>  <Rle> | <integer> <character> <integer>
##    [1]     chr7 103629578-103629803      - |     93139        <NA>         1
##    [2]     chr7 103557522-103557632      - |     93138        <NA>         2
##    [3]     chr7 103473984-103474119      - |     93137        <NA>         3
##    [4]     chr7 103417004-103417074      - |     93136        <NA>         4
##    [5]     chr7 103393629-103393661      - |     93135        <NA>         5
##    ...      ...                 ...    ... .       ...         ...       ...
##   [61]     chr7 103126644-103126863      - |     93079        <NA>        61
##   [62]     chr7 103124100-103124297      - |     93078        <NA>        62
##   [63]     chr7 103123320-103123418      - |     93077        <NA>        63
##   [64]     chr7 103113449-103113453      - |     93075        <NA>        64
##   [65]     chr7 103113333-103113355      - |     93074        <NA>        65
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
##
## $uc022ajr.1
## GRanges object with 65 ranges and 3 metadata columns:
##        seqnames              ranges strand |    cds_id    cds_name exon_rank
##           <Rle>           <IRanges>  <Rle> | <integer> <character> <integer>
##    [1]     chr7 103629578-103629803      - |     93139        <NA>         1
##    [2]     chr7 103557522-103557632      - |     93138        <NA>         2
##    [3]     chr7 103473984-103474119      - |     93137        <NA>         3
##    [4]     chr7 103417004-103417074      - |     93136        <NA>         4
##    [5]     chr7 103393629-103393661      - |     93135        <NA>         5
##    ...      ...                 ...    ... .       ...         ...       ...
##   [61]     chr7 103126644-103126863      - |     93079        <NA>        61
##   [62]     chr7 103124100-103124297      - |     93078        <NA>        62
##   [63]     chr7 103123320-103123418      - |     93077        <NA>        63
##   [64]     chr7 103118836-103118841      - |     93076        <NA>        64
##   [65]     chr7 103113259-103113355      - |     93073        <NA>        65
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

There are 3 *RELN* transcripts..

```
length(ranges(reln_tx))
```

```
## [1] 3
```

```
#sanity check
names(reln_tx)
```

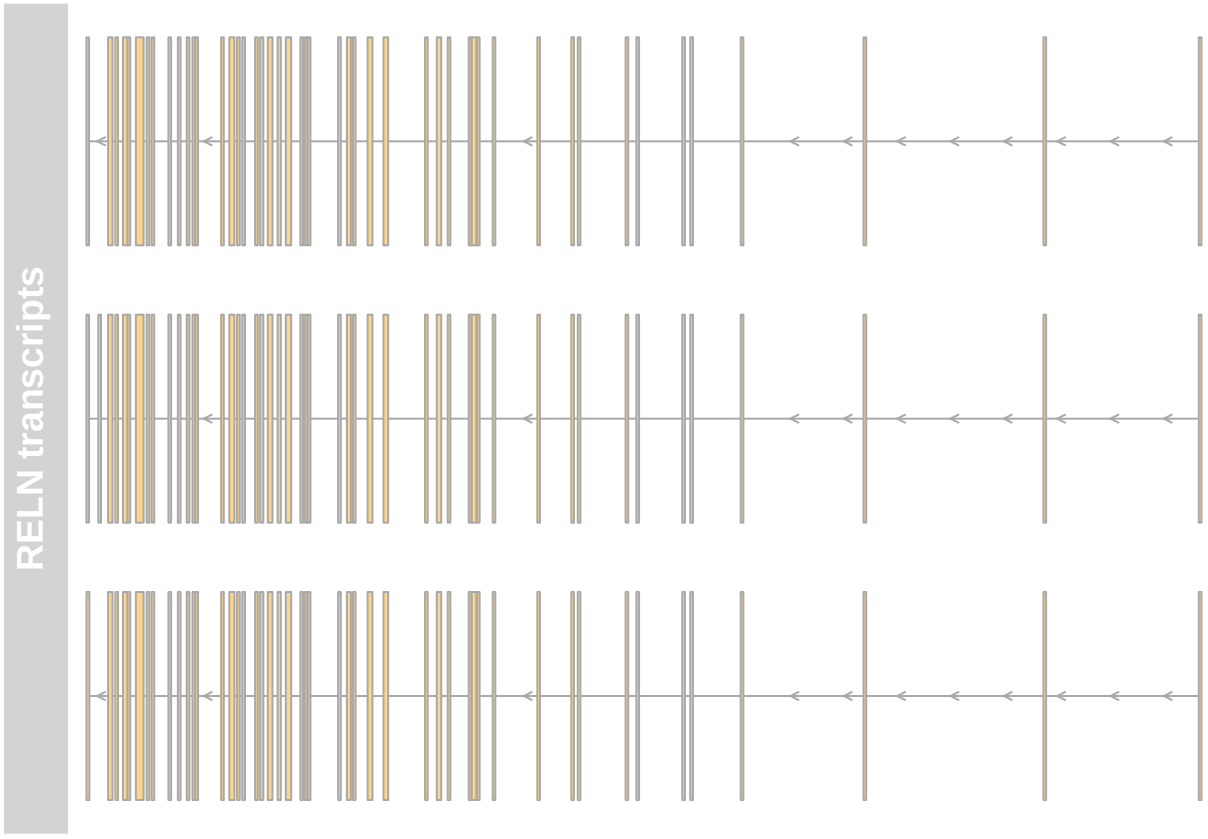
```
## [1] "uc010liz.3" "uc022ajq.1" "uc022ajr.1"
```

```
txid$TXNAME == names(reln_tx)
```

```
## [1] TRUE TRUE TRUE
```

Let's plot the different *RELN* trasnscripts (mRNA)

```
gr <- unlist(reln_tx)
elementMetadata(gr)$transcript <- names(gr)
track <- GeneRegionTrack(gr, name= "RELN transcripts")
plotTracks(track)
```



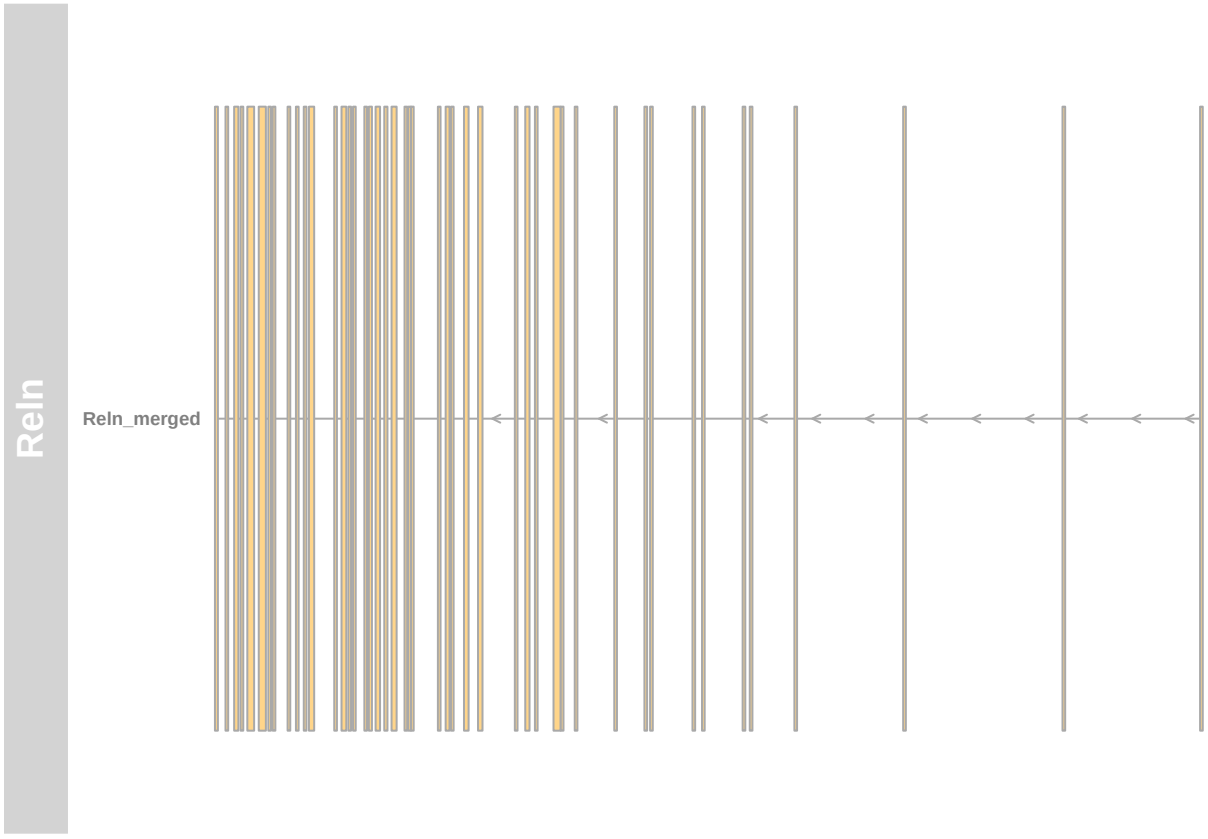
If we collapse all the *RELN* transcripts, eliminating all overlaps, we obtain the *RELN* CDS model

```
gene_gr <- reduce_ranges(gr)
name <- rep("reln_gene", each=66)
names(gene_gr) <- name
strand_neg <- rep("-", each=66)
strand(gene_gr) <- strand_neg

mcols(gene_gr)$feature <- "exon"
mcols(gene_gr)$transcript <- "Reln_merged"
mcols(gene_gr)$gene <- "Reln"

track <- Gviz::GeneRegionTrack(gene_gr,
                               genome = "hg19_txdb",
                               chromosome = as.character(seqnames(gene_gr)[1]),
                               name = "Reln",
                               transcriptAnnotation = "transcript")

plotTracks(track)
```


Reln
Reln_merged