Anotación

Guillermo Ayala Gallego

Anotación

Guillermo Ayala Gallego

5/16/23

Tipos de paquetes

- **ChipDb**: Relativos a una plataforma, a un chip concreto de microarray.
- **OrgDb**: Se refieren a organismos.
- **TxDb**: Relativos a transcriptomas de un organismo.
- **BSgenome**: Genomas.

AnnotationDbi

- Las bases de datos de tipo **ChipDb**, **OrgDb** y **TxDb** heredan todos los métodos de la clase **AnnotationDbi::AnnotationDb**.
- Podemos aplicar los métodos: **columns**, **keytypes**,**keys** y **select**.
- Cargamos **AnnotationDbi** y un paquete de anotación, por ejemplo, **hgu133a.db**.

```
pacman::p_load("AnnotationDbi","hgu133a.db")
```

- La información contenida la podemos ver con

```
ls("package:hgu133a.db")
```

```
 [1] "hgu133a"              "hgu133a_dbconn"       "hgu133a_dbfile"
 [4] "hgu133a_dbInfo"       "hgu133a_dbschema"     "hgu133a.db"
 [7] "hgu133aACCNUM"        "hgu133aALIAS2PROBE"   "hgu133aCHR"
[10] "hgu133aCHRLENGTHS"    "hgu133aCHRLOC"        "hgu133aCHRLOCEND"
[13] "hgu133aENSEMBL"       "hgu133aENSEMBL2PROBE" "hgu133aENTREZID"
[16] "hgu133aENZYME"        "hgu133aENZYME2PROBE"  "hgu133aGENENAME"
[19] "hgu133aGO"            "hgu133aGO2ALLPROBES"  "hgu133aGO2PROBE"
[22] "hgu133aMAP"           "hgu133aMAPCOUNTS"     "hgu133aOMIM"
[25] "hgu133aORGANISM"      "hgu133aORGPKG"        "hgu133aPATH"
[28] "hgu133aPATH2PROBE"    "hgu133aPFAM"          "hgu133aPMID"
```

```
[31] "hgu133aPMID2PROBE"    "hgu133aPROSITE"       "hgu133aREFSEQ"
[34] "hgu133aSYMBOL"        "hgu133aUNIPROT"
```

Con el nombre del paquete también tenemos información.

```
hgu133a.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-U133A 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
| 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
```

- En los paquetes de anotación tenemos **columns**. Algunas de estas columnas pueden ser **keys**. Podemos realizar consultas en la base de datos utilizando una **key** y pedir que nos devuelva una o más de una **columns**.
- ¿Qué información podemos recuperar utilizando **select**?

```
columns(hgu133a.db)
```

```
 [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"         "PROBEID"      "PROSITE"      "REFSEQ"       "SYMBOL"
[26] "UCSCKG"       "UNIPROT"
```

- Pero: ¿qué información es?

```
help("ENTREZID")
```

- No todas las variables que hemos obtenido con **columns** son utilizables para realizar consultas.
- Aquellas utilizables para las consultas las podemos conocer con **keytypes**.
- A estas variables las llamamos llaves **keys**.

```
keytypes(hgu133a.db)
```

```
 [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"         "PROBEID"      "PROSITE"      "REFSEQ"       "SYMBOL"
[26] "UCSCKG"       "UNIPROT"
```

- ¿Cómo conseguir todas los valores de una llave determinada?

```
head(keys(hgu133a.db,keytype="ENTREZID"))
```

```
[1] "10"        "100"       "1000"      "10000"     "100008586" "10001"
```

```
head(keys(hgu133a.db,keytype="ENSEMBL"))
```

```
[1] "ENSG00000121410" "ENSG00000175899" "ENSG00000291190" "ENSG00000171428"
[5] "ENSG00000156006" "ENSG00000196136"
```

- Supongamos que tenemos algunos identificadores Affymetrix (**PROBEID**) y pretendemos conocer sus identificadores **ENTREZID**.

```
(ids = sample(keys(hgu133a.db,keytype="PROBEID"),5))
```

```
[1] "217573_at"   "209582_s_at" "202848_s_at" "200885_at"   "217645_at"
```

```
AnnotationDbi::select(hgu133a.db,keys=ids,columns=c("ENTREZID","ENSEMBL",
                                                    "SYMBOL"),keytype="PROBEID")
```

```
      PROBEID  ENTREZID         ENSEMBL        SYMBOL
1   217573_at      2905 ENSG00000161509        GRIN2C
2 209582_s_at      4345 ENSG00000091972         CD200
3 202848_s_at      2870 ENSG00000198055          GRK6
4   200885_at    333926 ENSG00000155367         PPM1J
5   200885_at       389 ENSG00000155366          RHOC
```

```
6   217645_at     51241 ENSG00000133983         COX16
7   217645_at 100529257 ENSG00000258644 SYNJ2BP-COX16
```

ChipDb

- Si trabajamos con microarrays nuestras características serán las sondas.
- Hemos de poder hacer corresponder este identificador con el gen al que corresponde.

```
pacman::p_load(hgu133a.db)
```

¿Qué sondas tienen correspondencia en Entrez?

```
mappedProbes = mappedkeys(hgu133aENTREZID)
```

- Lo guardamos en forma de lista.

```
mappedProbesList = as.list(hgu133aENTREZID[mappedProbes])
```

Por ejemplo, la primera posición de la lista nos da el identificador de Affymetrix y su identificador Entrez.

```
mappedProbesList[1]
```

```
$`1007_s_at`
[1] "780"
```

- Podemos ver todas las correspondencias que nos ofrece el paquete.

```
ls("package:hgu133a.db")
```

```
 [1] "hgu133a"              "hgu133a_dbconn"       "hgu133a_dbfile"
 [4] "hgu133a_dbInfo"       "hgu133a_dbschema"     "hgu133a.db"
 [7] "hgu133aACCNUM"        "hgu133aALIAS2PROBE"   "hgu133aCHR"
[10] "hgu133aCHRLENGTHS"    "hgu133aCHRLOC"        "hgu133aCHRLOCEND"
[13] "hgu133aENSEMBL"       "hgu133aENSEMBL2PROBE" "hgu133aENTREZID"
[16] "hgu133aENZYME"        "hgu133aENZYME2PROBE"  "hgu133aGENENAME"
[19] "hgu133aGO"            "hgu133aGO2ALLPROBES"  "hgu133aGO2PROBE"
[22] "hgu133aMAP"           "hgu133aMAPCOUNTS"     "hgu133aOMIM"
[25] "hgu133aORGANISM"      "hgu133aORGPKG"        "hgu133aPATH"
[28] "hgu133aPATH2PROBE"    "hgu133aPFAM"          "hgu133aPMID"
[31] "hgu133aPMID2PROBE"    "hgu133aPROSITE"       "hgu133aREFSEQ"
[34] "hgu133aSYMBOL"        "hgu133aUNIPROT"
```

- Otro ejemplo de determinar la correspondencia entre PROBEID e identificadores.

```
ids = c("39730_at", "1635_at", "1674_at", "40504_at", "40202_at")
pacman::p_load("hgu95av2.db")
columns(hgu95av2.db)
```

```
 [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"         "PROBEID"      "PROSITE"      "REFSEQ"       "SYMBOL"
[26] "UCSCKG"       "UNIPROT"
```

```
keytypes(hgu95av2.db)
```

```
 [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"         "PROBEID"      "PROSITE"      "REFSEQ"       "SYMBOL"
[26] "UCSCKG"       "UNIPROT"
```

```
columns = c("PFAM","SYMBOL")
AnnotationDbi::select(hgu95av2.db, keys=ids, columns, keytype="PROBEID")
```

```
    PROBEID    PFAM SYMBOL
1  39730_at PF00017   ABL1
2  39730_at PF08919   ABL1
3  39730_at PF07714   ABL1
4  39730_at PF00018   ABL1
5   1635_at PF00017   ABL1
6   1635_at PF08919   ABL1
7   1635_at PF07714   ABL1
8   1635_at PF00018   ABL1
9   1674_at PF00017   YES1
10  1674_at PF00018   YES1
11  1674_at PF07714   YES1
12 40504_at PF01731   PON2
13 40202_at PF00096   KLF9
```

OrgDb

```
pacman::p_load(org.Hs.eg.db)
```

- ¿Qué tipo de cosas o qué claves podemos manejar?

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

```
 [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"
```

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

```
 [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"
```

- Los primeros identificadores de genes utilizando el código ENTREZID.

```
head(keys(org.Hs.eg.db, keytype="ENTREZID"))
```

```
[1] "1"  "2"  "3"  "9"  "10" "11"
```

- Los mismos pero en ENSEMBL.

```
head(keys(org.Hs.eg.db, keytype="ENSEMBL"))
```

```
[1] "ENSG00000121410" "ENSG00000175899" "ENSG00000291190" "ENSG00000171428"
[5] "ENSG00000156006" "ENSG00000196136"
```

- Y en Gene Ontology.

```
head(keys(org.Hs.eg.db, keytype="GO"))
```

```
[1] "GO:0003674" "GO:0005576" "GO:0005615" "GO:0005886" "GO:0008150"
[6] "GO:0031093"
```

- Supongamos que elegimos un sistema de identificación, por ejemplo, ENTREZID.

```
(ids = keys(org.Hs.eg.db, keytype="ENTREZID")[1:5])
```

```
[1] "1"  "2"  "3"  "9"  "10"
```

- A partir de estos identicadores podemos obtener el resto.

```
AnnotationDbi::select(org.Hs.eg.db,keys=ids,column="SYMBOL",keytype='ENTREZID')
```

```
  ENTREZID SYMBOL
1        1   A1BG
2        2    A2M
3        3  A2MP1
4        9   NAT1
5       10   NAT2
```

- Supongamos que nos fijamos en el gen con código **ENSG00000000003** en Ensembl.

```
id = "ENSG00000000003"
```

- Buscamos su correspondencia en Gene Ontology.

```
(res =  AnnotationDbi::select(org.Hs.eg.db, keys=id, columns="GO", keytype="ENSEMBL"))
```

```
          ENSEMBL         GO EVIDENCE ONTOLOGY
1 ENSG00000000003 GO:0005515      IPI       MF
2 ENSG00000000003 GO:0005886      IBA       CC
```

```
3 ENSG00000000003 GO:0039532      IMP       BP
4 ENSG00000000003 GO:0043123      HMP       BP
5 ENSG00000000003 GO:0070062      HDA       CC
6 ENSG00000000003 GO:1901223      IDA       BP
```

- Como vemos no tenemos una correspondencia 1-1.
- Al mismo gen le corresponden distintos términos GO.
- Si solamente tenemos interés en ellos podemos hacer

```
res[,"GO"]
```

```
[1] "GO:0005515" "GO:0005886" "GO:0039532" "GO:0043123" "GO:0070062"
[6] "GO:1901223"
```

- Puesto que tenemos identificadores **GO** podemos utilizar el paquete **GO.db** para obtener los términos **GO** correspondientes.

```
pacman::p_load("GO.db")
```

Y los términos GO serían

```
AnnotationDbi::select(GO.db, keys=res[,"GO"], columns="TERM", keytype="GOID")
```

```
        GOID
1 GO:0005515
2 GO:0005886
3 GO:0039532
4 GO:0043123
5 GO:0070062
6 GO:1901223

1                                                                                protein bi
2                                                                                plasma mem
3 negative regulation of viral-induced cytoplasmic pattern recognition receptor signaling pa
4                                    positive regulation of I-kappaB kinase/NF-kappaB sign
5                                                                    extracellular ex
6                                                negative regulation of NIK/NF-kappaB sign
```

TxDb

- Los paquetes TxDb están centrados en el genoma.

```
pacman::p_load("TxDb.Dmelanogaster.UCSC.dm3.ensGene")
txdb = TxDb.Dmelanogaster.UCSC.dm3.ensGene
```

- ¿De qué clase es este objeto?

```
class(txdb)
```

```
[1] "TxDb"
attr(,"package")
[1] "GenomicFeatures"
```

- Las opciones que tenemos

```
columns(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"
```

```
keytypes(txdb)
```

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

- Para el manejo de este tipo de bases de datos es útil **GenomicFeatures**.

```
pacman::p_load(GenomicFeatures)
```

- Por ejemplo: ¿Qué cromosomas tenemos?

```
seqlevels(txdb)
```

```
 [1] "chr2L"     "chr2R"     "chr3L"     "chr3R"     "chr4"      "chrX"
 [7] "chrU"      "chrM"      "chr2LHet"  "chr2RHet"  "chr3LHet"  "chr3RHet"
[13] "chrXHet"   "chrYHet"   "chrUextra"
```

- Cuando se carga la base de datos todos los cromosomas están activos y lo que hagamos nos dará información sobre todos ellos.

- Queremos, por ejemplo, trabajar solamente con **chr2L**.

```
seqlevels(txdb) = "chr2L"
```

- Supongamos que queremos conocer los **GENEID** de los primeros genes.

```
(keysGENEID = head(keys(txdb, keytype="GENEID"),n=3))
```

```
[1] "FBgn0000003" "FBgn0000008" "FBgn0000014"
```

```
columns = c("TXNAME", "TXSTART","TXEND","TXSTRAND")
AnnotationDbi::select(txdb, keysGENEID, columns, keytype="GENEID")
```

```
       GENEID      TXNAME TXSTRAND  TXSTART    TXEND
1 FBgn0000003 FBtr0081624        +  2648220  2648518
2 FBgn0000008 FBtr0100521        + 18024494 18060339
3 FBgn0000008 FBtr0071763        + 18024496 18060346
4 FBgn0000008 FBtr0071764        + 18024938 18060346
5 FBgn0000014 FBtr0306337        - 12632936 12655767
6 FBgn0000014 FBtr0083388        - 12633349 12653845
7 FBgn0000014 FBtr0083387        - 12633349 12655300
8 FBgn0000014 FBtr0300485        - 12633349 12655474
```

- Los objetos **TxDb** nos permiten obtener las anotaciones como **GRanges**.

```
(txdb.tr = transcripts(txdb))
```

```
GRanges object with 5384 ranges and 2 metadata columns:
         seqnames            ranges strand |     tx_id     tx_name
            <Rle>         <IRanges>  <Rle> | <integer> <character>
     [1]    chr2L         7529-9484      + |         1 FBtr0300689
     [2]    chr2L         7529-9484      + |         2 FBtr0300690
     [3]    chr2L         7529-9484      + |         3 FBtr0330654
     [4]    chr2L       21952-24237      + |         4 FBtr0309810
     [5]    chr2L       66584-71390      + |         5 FBtr0306539
     ...      ...               ...    ... .       ...         ...
  [5380]    chr2L 22892306-22918560      - |      5380 FBtr0331166
  [5381]    chr2L 22892306-22918647      - |      5381 FBtr0111127
  [5382]    chr2L 22959606-22960915      - |      5382 FBtr0111241
  [5383]    chr2L 22959606-22961179      - |      5383 FBtr0111239
  [5384]    chr2L 22959606-22961179      - |      5384 FBtr0111240
  -------
  seqinfo: 1 sequence from dm3 genome
```

- También podemos obtener información sobre los exones.

```
1 (txdb.ex = exons(txdb))
```

```
GRanges object with 13850 ranges and 1 metadata column:
          seqnames            ranges strand |   exon_id
             <Rle>         <IRanges>  <Rle> | <integer>
      [1]    chr2L         7529-8116      + |         1
      [2]    chr2L         8193-8589      + |         2
      [3]    chr2L         8193-9484      + |         3
      [4]    chr2L         8229-9484      + |         4
      [5]    chr2L         8668-9484      + |         5
      ...      ...               ...    ... .       ...
  [13846]    chr2L 22959606-22959815      - |     13846
  [13847]    chr2L 22959877-22960833      - |     13847
  [13848]    chr2L 22959877-22960876      - |     13848
  [13849]    chr2L 22959877-22960915      - |     13849
  [13850]    chr2L 22960932-22961179      - |     13850
  -------
  seqinfo: 1 sequence from dm3 genome
```

- Podemos incluir metadatos adicionales como puede ser el identificador del gen.

```
1 transcripts(txdb, columns = c("tx_id","tx_name","gene_id"))
```

```
GRanges object with 5384 ranges and 3 metadata columns:
         seqnames            ranges strand |     tx_id     tx_name
            <Rle>         <IRanges>  <Rle> | <integer> <character>
     [1]    chr2L         7529-9484      + |         1 FBtr0300689
     [2]    chr2L         7529-9484      + |         2 FBtr0300690
     [3]    chr2L         7529-9484      + |         3 FBtr0330654
```

```
   [4]    chr2L       21952-24237      + |         4 FBtr0309810
   [5]    chr2L       66584-71390      + |         5 FBtr0306539
   ...      ...               ...    ... .       ...         ...
[5380]    chr2L 22892306-22918560      - |      5380 FBtr0331166
[5381]    chr2L 22892306-22918647      - |      5381 FBtr0111127
[5382]    chr2L 22959606-22960915      - |      5382 FBtr0111241
[5383]    chr2L 22959606-22961179      - |      5383 FBtr0111239
[5384]    chr2L 22959606-22961179      - |      5384 FBtr0111240
              gene_id
      <CharacterList>
   [1]     FBgn0031208
   [2]     FBgn0031208
   [3]     FBgn0031208
   [4]     FBgn0263584
   [5]     FBgn0067779
   ...             ...
[5380]     FBgn0250907
[5381]     FBgn0250907
[5382]     FBgn0086683
[5383]     FBgn0086683
[5384]     FBgn0086683
-------
seqinfo: 1 sequence from dm3 genome
```

- Obtenemos las regiones CDS con

```
(txdb.cds = cds(txdb))
```

```
GRanges object with 11003 ranges and 1 metadata column:
          seqnames            ranges strand |    cds_id
             <Rle>         <IRanges>  <Rle> | <integer>
      [1]    chr2L         7680-8116      + |         1
      [2]    chr2L         8193-8589      + |         2
      [3]    chr2L         8193-8610      + |         3
      [4]    chr2L         8229-8610      + |         4
      [5]    chr2L         8668-9276      + |         5
      ...      ...               ...    ... .       ...
[10999]    chr2L 22959877-22960833      - |     10999
[11000]    chr2L 22959877-22960873      - |     11000
[11001]    chr2L 22959877-22960876      - |     11001
[11002]    chr2L 22960932-22960995      - |     11002
[11003]    chr2L 22960932-22961048      - |     11003
-------
seqinfo: 1 sequence from dm3 genome
```

- En los tres casos hemos obtenido un objeto **GRanges**.
- A este tipo de objetos podemos aplicar otros métodos que nos dan infor-

mación adicional.

```
1 length(txdb.cds)
```

```
[1] 11003
```

```
1 strand(txdb.cds)
```

```
factor-Rle of length 11003 with 2 runs
  Lengths: 5489 5514
  Values :    +    -
Levels(3): + - *
```

- A partir de un objeto de clase **TxDb** podemos obtener un **GRangesList** en la cual separamos por alguna característica.
- Por ejemplo, los objetos **GRanges** para los distintos genes.

```
1 transcriptsBy(txdb, by="gene")
```

```
GRangesList object of length 2986:
$FBgn0000018
GRanges object with 1 range and 2 metadata columns:
      seqnames            ranges strand |     tx_id     tx_name
         <Rle>         <IRanges>  <Rle> | <integer> <character>
  [1]    chr2L 10973443-10975273      - |      4279 FBtr0080168
  -------
  seqinfo: 1 sequence from dm3 genome

$FBgn0000052
GRanges object with 3 ranges and 2 metadata columns:
      seqnames          ranges strand |     tx_id     tx_name
         <Rle>       <IRanges>  <Rle> | <integer> <character>
  [1]    chr2L 6041178-6045593      - |      3537 FBtr0079221
  [2]    chr2L 6041178-6045970      - |      3538 FBtr0079219
  [3]    chr2L 6041178-6045970      - |      3539 FBtr0079220
  -------
  seqinfo: 1 sequence from dm3 genome

$FBgn0000053
GRanges object with 2 ranges and 2 metadata columns:
      seqnames          ranges strand |     tx_id     tx_name
         <Rle>       <IRanges>  <Rle> | <integer> <character>
  [1]    chr2L 7014861-7023940      - |      3704 FBtr0079431
  [2]    chr2L 7018085-7023940      - |      3705 FBtr0100353
  -------
  seqinfo: 1 sequence from dm3 genome

...
<2983 more elements>
```

- Podemos agrupar los exones por gen.

```
exonsBy(txdb, by="gene")
```

```
GRangesList object of length 2986:
$FBgn0000018
GRanges object with 2 ranges and 2 metadata columns:
      seqnames            ranges strand |   exon_id   exon_name
         <Rle>         <IRanges>  <Rle> | <integer> <character>
  [1]    chr2L 10973443-10974210      - |     11021        <NA>
  [2]    chr2L 10974268-10975273      - |     11022        <NA>
  -------
  seqinfo: 1 sequence from dm3 genome

$FBgn0000052
GRanges object with 6 ranges and 2 metadata columns:
      seqnames          ranges strand |   exon_id   exon_name
         <Rle>       <IRanges>  <Rle> | <integer> <character>
  [1]    chr2L 6041178-6042011      - |      8949        <NA>
  [2]    chr2L 6042075-6044351      - |      8950        <NA>
  [3]    chr2L 6044428-6045526      - |      8951        <NA>
  [4]    chr2L 6044428-6045593      - |      8952        <NA>
  [5]    chr2L 6045894-6045970      - |      8955        <NA>
  [6]    chr2L 6045921-6045970      - |      8956        <NA>
  -------
  seqinfo: 1 sequence from dm3 genome

$FBgn0000053
GRanges object with 8 ranges and 2 metadata columns:
      seqnames          ranges strand |   exon_id   exon_name
         <Rle>       <IRanges>  <Rle> | <integer> <character>
  [1]    chr2L 7014861-7016374      - |      9389        <NA>
  [2]    chr2L 7016437-7017486      - |      9390        <NA>
  [3]    chr2L 7017545-7017966      - |      9391        <NA>
  [4]    chr2L 7018085-7018374      - |      9392        <NA>
  [5]    chr2L 7018149-7018374      - |      9393        <NA>
  [6]    chr2L 7018431-7019080      - |      9394        <NA>
  [7]    chr2L 7019135-7019382      - |      9395        <NA>
  [8]    chr2L 7023529-7023940      - |      9396        <NA>
  -------
  seqinfo: 1 sequence from dm3 genome

...
<2983 more elements>
```

- Podemos tener los CDS agrupados por transcrito.

```
cdsBy(txdb, by="tx")
```

```
GRangesList object of length 4951:
$`1`
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]    chr2L 7680-8116      + |         1        <NA>         1
  [2]    chr2L 8193-8610      + |         3        <NA>         2
  -------
  seqinfo: 1 sequence from dm3 genome

$`2`
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]    chr2L 7680-8116      + |         1        <NA>         1
  [2]    chr2L 8193-8589      + |         2        <NA>         2
  [3]    chr2L 8668-9276      + |         5        <NA>         3
  -------
  seqinfo: 1 sequence from dm3 genome

$`3`
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]    chr2L 7680-8116      + |         1        <NA>         1
  [2]    chr2L 8229-8610      + |         4        <NA>         2
  -------
  seqinfo: 1 sequence from dm3 genome

...
<4948 more elements>
```

- Los intrones agrupados por transcrito.

```
1 intronsByTranscript(txdb)
```

```
GRangesList object of length 5384:
$`1`
GRanges object with 1 range and 0 metadata columns:
      seqnames    ranges strand
         <Rle> <IRanges>  <Rle>
  [1]    chr2L 8117-8192      +
  -------
  seqinfo: 1 sequence from dm3 genome

$`2`
GRanges object with 2 ranges and 0 metadata columns:
```

```
      seqnames    ranges strand
         <Rle> <IRanges>  <Rle>
  [1]    chr2L 8117-8192      +
  [2]    chr2L 8590-8667      +
  -------
  seqinfo: 1 sequence from dm3 genome

$`3`
GRanges object with 1 range and 0 metadata columns:
      seqnames    ranges strand
         <Rle> <IRanges>  <Rle>
  [1]    chr2L 8117-8228      +
  -------
  seqinfo: 1 sequence from dm3 genome

...
<5381 more elements>
```

Las regiones UTR 5' y 3' agrupadas por transcrito.

```
1 fiveUTRsByTranscript(txdb)
```

```
GRangesList object of length 4733:
$`1`
GRanges object with 1 range and 3 metadata columns:
      seqnames    ranges strand |   exon_id   exon_name exon_rank
         <Rle> <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L 7529-7679      + |         1        <NA>         1
  -------
  seqinfo: 1 sequence from dm3 genome

$`2`
GRanges object with 1 range and 3 metadata columns:
      seqnames    ranges strand |   exon_id   exon_name exon_rank
         <Rle> <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L 7529-7679      + |         1        <NA>         1
  -------
  seqinfo: 1 sequence from dm3 genome

$`3`
GRanges object with 1 range and 3 metadata columns:
      seqnames    ranges strand |   exon_id   exon_name exon_rank
         <Rle> <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L 7529-7679      + |         1        <NA>         1
  -------
  seqinfo: 1 sequence from dm3 genome
```

```
...
<4730 more elements>
```

```
threeUTRsByTranscript(txdb)
```

```
GRangesList object of length 4721:
$`1`
GRanges object with 1 range and 3 metadata columns:
      seqnames    ranges strand |   exon_id   exon_name exon_rank
         <Rle> <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L 8611-9484      + |         3        <NA>         2
  -------
  seqinfo: 1 sequence from dm3 genome

$`2`
GRanges object with 1 range and 3 metadata columns:
      seqnames    ranges strand |   exon_id   exon_name exon_rank
         <Rle> <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L 9277-9484      + |         5        <NA>         3
  -------
  seqinfo: 1 sequence from dm3 genome

$`3`
GRanges object with 1 range and 3 metadata columns:
      seqnames    ranges strand |   exon_id   exon_name exon_rank
         <Rle> <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L 8611-9484      + |         4        <NA>         2
  -------
  seqinfo: 1 sequence from dm3 genome

...
<4718 more elements>
```

BSgenome

```
pacman::p_load(BSgenome.Dmelanogaster.UCSC.dm3)
tx2seqs = extractTranscriptSeqs(BSgenome.Dmelanogaster.UCSC.dm3,TxDb.Dmelanogaster.UCSC.dm3.
```

- La secuencia correspondiente al primer gen sería

```
tx2seqs[[1]]
```

```
1880-letter DNAString object
seq: CTACTCGCATGTAGAGATTTCCACTTATGTTTTCTC...CAGAGAATCTAGTTTTTCAATAAAATTTCCCCAAGT
```

- Si queremos conocer la secuencia entre las posiciones 1000 y 1020 entonces

```
tx2seqs[[1]][1000:1020]
```

```
21-letter DNAString object
```

```
seq: ATATTGATGTCTTTCGTACCC
```

- También podemos trasladar estas secuencias a proteínas con

```
suppressWarnings(translate(tx2seqs[[1]]))
```

```
626-letter AAString object
seq: LLACRDFHLCFLYFQQPRREPTFEQVSACGQQLSPL...LSYLIELSWPRDVLVQ*LVLISDCNRESSFSIKFPQ
```

- Para todos los transcritos lo hacemos con

```
suppressWarnings(translate(tx2seqs))
```

```
AAStringSet object of length 5384:
       width seq                                               names
   [1]   626 LLACRDFHLCFLYFQQPRREPT...Q*LVLISDCNRESSFSIKFPQ 1
   [2]   600 LLACRDFHLCFLYFQQPRREPT...Q*LVLISDCNRESSFSIKFPQ 2
   [3]   614 LLACRDFHLCFLYFQQPRREPT...Q*LVLISDCNRESSFSIKFPQ 3
   [4]   743 IAVVIHLVHKFV*TLQRFKSFP...H*FPRGRK**E*SEK*IKLYS 4
   [5]  1350 RQINT*YVCTYTLNTRSQEKTG...LIITVPKNNFK*CK*IMMLKI 5
   ...   ... ...
[5380]  2389 LDFL*KCDISS*PRPKGKCTFV...*FILYNHIFAN*TI*IKNKLN 5380
[5381]  2365 ITLVKGCVQIIGCYRRLQQLIK...LIYFI*PHICELNYLNKK*IK 5381
[5382]   416 PI*IIQLLIHPM*LMTPVVNLR...KKCLPHRTSLVNFY*KLLSIK 5382
[5383]   486 V*IGH*VEQLIVIKSIGP*K*I...KNVYRTELVL*ISIKNYFLSN 5383
[5384]   471 V*IGH*VEQLIVIKSIGP*K*I...*KMFTAQN*SCKFLLKTTFYQ 5384
```

- No todo lo que se transcribe se traslada. Para obtener obtener las que realmente se trasladan se puede hacer

```
cds2seqs = extractTranscriptSeqs(BSgenome.Dmelanogaster.UCSC.dm3,cdsBy(txdb, by="tx"))
translate(cds2seqs)
```

```
AAStringSet object of length 4951:
       width seq                                               names
   [1]   285 MGERDQPQSSERISIFNPPVYT...HDRFNEITQDDKSTVWQRIY* 1
   [2]   481 MGERDQPQSSERISIFNPPVYT...QSEMLYFRKKMALEIVDGEL* 2
   [3]   273 MGERDQPQSSERISIFNPPVYT...HDRFNEITQDDKSTVWQRIY* 3
   [4]  1008 MDAQFEHLCRICAANTKSKTNS...YKGPTSKSHHSMGTRSTRQR* 5
   [5]  1008 MDAQFEHLCRICAANTKSKTNS...YKGPTSKSHHSMGTRSTRQR* 6
   ...   ... ...
[4947]  2287 MHPYVPSVLVVVVLAISVKAHI...LRTLSQELLGIPGQKAKDCI* 5380
[4948]  2287 MHPYVPSVLVVVVLAISVKAHI...LRTLSQELLGIPGQKAKDCI* 5381
[4949]   364 MTPVVNLRSKRLADPEVTCFAP...GRQVRAGFYNYDKFKCFQLH* 5382
[4950]   404 MDLYDGIDTRARSSQIDGWSSG...GRQVRAGFYNYDKFKCFQLH* 5383
[4951]   372 MVFGNKNVTDPAGSKNGCQKTN...GRQVRAGFYNYDKFKCFQLH* 5384
```

OrganismDb

- Un paquete de tipo OrganismDb nos permite combinar información (para el organismo con que estemos trabajando) de **GO.db** con el correspondiente **TxDb** y **OrgDb**.

```
pacman::p_load(Homo.sapiens)
```

```
(keys = head(keys(Homo.sapiens, keytype="ENTREZID"), n=2))
```

```
[1] "1" "2"
```

```
columns = c("SYMBOL","TXNAME")
AnnotationDbi::select(Homo.sapiens, keys, columns, keytype="ENTREZID")
```

```
  ENTREZID SYMBOL     TXNAME
1        1   A1BG uc002qsd.4
2        1   A1BG uc002qsf.2
3        2    A2M uc001qvk.1
4        2    A2M uc009zgk.1
```

- También podemos obtener .

```
transcripts(Homo.sapiens, columns=c("TXNAME","SYMBOL"))
```

```
GRanges object with 82960 ranges and 2 metadata columns:
                seqnames        ranges strand |          TXNAME          SYMBOL
                   <Rle>     <IRanges>  <Rle> | <CharacterList> <CharacterList>
      [1]           chr1   11874-14409      + |      uc001aaa.3         DDX11L1
      [2]           chr1   11874-14409      + |      uc010nxq.1         DDX11L1
      [3]           chr1   11874-14409      + |      uc010nxr.1         DDX11L1
      [4]           chr1   69091-70008      + |      uc001aal.1           OR4F5
      [5]           chr1 321084-321115      + |      uc001aaq.2           <NA>
      ...            ...           ...    ... .           ...            ...
  [82956] chrUn_gl000237        1-2686      - |      uc011mgu.1           <NA>
  [82957] chrUn_gl000241   20433-36875      - |      uc011mgv.2           <NA>
  [82958] chrUn_gl000243   11501-11530      + |      uc011mgw.1           <NA>
  [82959] chrUn_gl000243   13608-13637      + |      uc022brq.1           <NA>
  [82960] chrUn_gl000247     5787-5816      - |      uc022brr.1           <NA>
  -------
  seqinfo: 93 sequences (1 circular) from hg19 genome
```

biomaRt

- El paquete **biomaRt** es un interfaz para poder acceder a una serie de bases de datos que implementan BioMart

```
pacman::p_load(biomaRt)
```

- Podemos ver la lista de **mart**'s que tenemos

```
listMarts(host="www.ensembl.org")
```

- Elegimos utilizar **ensembl**.

```
(ensembl = useMart("ENSEMBL_MART_ENSEMBL",host="www.ensembl.org"))
```

- Ahora hemos de elegir el conjunto de datos a utilizar.
- Podemos ver los disponibles con

```
head(listDatasets(ensembl))
```

- Elegimos la correspondiente a ser humano.

```
(ensembl = useMart("ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl",
     host="www.ensembl.org"))
```

- Podemos ver los atributos con

```
head(listAttributes(ensembl))
```

- De hecho, son

```
nrow(listAttributes(ensembl))
```

- Podemos comprobar que hay muchos que identifican el gen con las sondas de Affymetrix, en concreto, con los AffyID.

```
affyids=c("202763_at","209310_s_at","207500_at")
```

```
getBM(attributes=c('affy_hg_u133_plus_2', 'entrezgene'),
                     filters = 'affy_hg_u133_plus_2',
                     values = affyids, mart = ensembl)
```

- Podemos obtener todos los identificadores.

```
head(getBM(attributes='affy_hg_u133_plus_2', mart = ensembl))
```