Using refGenome package

Wolfgang Kaisers, CBiBs HHU Dusseldorf

May 8, 2013

1 Object types inside refGenome package

The central classes inside this package are `refGenome` derived (S4) classes. There is one class for Ensembl genomes `ensemblGenome` and one class for UCSC genomes `ucscGenome`. The objects basically contain annotation data in tables and the address of a folder (called "basedir").

The `ensemblExons` class centers on exon-intron-exon boundaries. The class also contains tabled annotation and a folder address ("basedir").

1.1 Creation of empty refGenome objects

Empty objects of `refGenome` derived classes can be created with `ensembleGenome()` or `ucscGenome()`. After creation of an empty object the first step usually is to set the basedir address:

```
> library(refGenome)
> ens<-ensemblGenome()
> basedir(ens)<-system.file("extdata",package="refGenome")
```

The "basedir" folder is intended to contain all data which is associated with the current annotation set, e.g. downloaded gtf files, saved object data, saved SQLite versions of the data and potenially sequence information. In order to fill an empty object, annotation data has to be imported from external files.

1.2 Importing annotation data

The basic importing mechanism for `refGenome` objects is to import a "gtf" file. Therefore, the "gtf" files have to be downloaded. The download source and mechanism is explained for `ensemblGenome` and `ucscGenome` separately. There are specialized mechanisms in order to provide additional information either from within the gtf file (ensembl) or via other external files (ucsc).

1.3 Saving and loading data

The data content of `refGenome` objects can be saved and re-loaded in several ways. One way is the `saveGenome` method where the content is written into a compressed ".RData" file. One alternative is to write the content into a SQLite database via `writeDB`.

2 Ensembl Genomes

The `ensemblGenome` class is specialized for managing annotation data for ensemble Genomes.

2.1 Download and import data

For ensemblGenome objects, gtf files can be downloaded from Ensemble servers. Therefore, go to

`http://www.ensembl.org/info/data/ftp/index.html`

and choose a file from the "Gene sets" column. They are labeled "GTF". For example Version 62 of human genomic annotation can be downloaded from

`ftp://ftp.ensembl.org/pub/release-62/gtf/homo_sapiens/Homo_sapiens.GRCh37.62.gtf.gz`

A copy of the obtained file should then be placed in the the "basedir" directory. With the appropriate setting of basedir, annotation data can be imported with:

```
> ens_gtf<-"hs.ensembl.62.small.gtf"
> read.gtf(ens,ens_gtf)

[read.gtf.refGenome] Reading file 'hs.ensembl.62.small.gtf'.
[read.gtf.refGenome] Parsing attributes.
[read.gtf.refGenome] Finished reading 135 gtf lines and 424 gtfattributes lines.

> ens

Object of class 'ensemblGenome' with 135 rows and 11 columns.
   seqid         source feature start   end score strand
25     1     pseudogene    exon 11869 12227     .      +
34     1 protein_coding    exon 11872 12227     .      +
41     1 protein_coding    exon 11874 12227     .      +
28     1     pseudogene    exon 12010 12057     .      +
29     1     pseudogene    exon 12179 12227     .      +
35     1 protein_coding     CDS 12190 12227     .      +
   frame id         gene_id   transcript_id
25     . 25 ENSG00000223972 ENST00000456328
34     . 34 ENSG00000249291 ENST00000515242
41     . 41 ENSG00000253101 ENST00000518655
28     . 28 ENSG00000223972 ENST00000450305
29     . 29 ENSG00000223972 ENST00000450305
35     0 35 ENSG00000249291 ENST00000515242
```

The top lines of the contained table are shown when the object is printed.

2.2 Attribute data in Ensembl Genome gtf files

In Ensembl gtf files there is additional data contained in the last column ("attributes"). Contained attribute types can be listet with "tableAttributeTypes". Specific attributes can be shifted into the main (gtf-) table by "moveAttributes":

```
> tableAttributeTypes(ens)

[tableAttributeTypes.refGenome] Row number in gtf-table: 135.

    exon_number       gene_name      protein_id
            135             135              19
transcript_name
            135

> moveAttributes(ens,c("gene_name","transcript_name","exon_number"))

[moveAttributes.ensemblGenome] Moving   135 attributes of type 'gene_name' from 'gtfattrib
[moveAttributes.ensemblGenome] Moving   135 attributes of type 'transcript_name' from 'gtf
[moveAttributes.ensemblGenome] Moving   135 attributes of type 'exon_number' from 'gtfattr
[moveAttributes.ensemblGenome] Finished. Reduced attributes table size from 424 to 19 rows
```

3 UCSC Genomes

Downloading of annotation data for UCSC genomes is a bit more complicated than for Ensemble Genomes because additional data must be downloaded in separate files. The Homepage for UCSC browser can be found under:

`http://genome.ucsc.edu/`

In order to import UCSC annotation data into `refGenome` objects files containing the data have to be downloaded from the USCS Table Browser which can be found under:

`http://genome.ucsc.edu/cgi-bin/hgTables`

or by following the "Tablel Browser" link in the left panel on the homepage. On the Table Browser:

- Select genome, assembly and track (UCSC genes)
- Choose table (knownGene)
- Choose output format (GTF -gene transfer format for knownGene table)
- Insert a name for the output file
- Download the file (get output)

The basic table to be imported is "knownGene". The knownGene table has to be downloaded in GTF format (otherwise the read.gtf function will complain

about "wrong number of columns").

In order to extend the available information additionally the tables "kgXref", "knownToEnsembl" and "knownIsoforms" can be downloaded and imported. These tables come in plain "csv" format. Select "all fields from selected table" as output format.

Do not use "add custom tracks" or modify the tables elsewhere tracks because the importing functions will check for apropriate number of columns.

After downloading, all tables should be placed into a separate folder which we from now on call "basedir".`ucscGenome` objects keep a `basedir` as standard location for all writing and reading procedures.

```
> uc<-ucscGenome()
> basedir(uc)<-"/my/ucsc/basedir"
> read.gtf(uc,"ucsc_knownGene.gtf")
> addXref(uc,"kgXref.csv")
> addEnsembl(uc,"knownToEnsembl.csv")
> addIsoforms(uc,"ucsc_knownisoforms.csv")
```

3.1 Load stored data

Once, annotation data is imported and stored, `ucscGenome` objects can be restored with the `load.ucsc` function which is shown below on example data:

```
> ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
> uc<-load.ucsc(ucfile)
```

4 Extracting data subsets

There are specialized functions for extracting data for multiple purposes.

4.1 Extracting data for sets of `seqid`'s

For preparation of `seqid` based extraction, the contained `seqid`'s can be tabled:

```
> tableSeqids(ens)
```

```
         1 GL000213.1
       111         24
```

Extraction of subsets based on `seqid` can be done with `extractSeqids`. The sequence id's for extraction are specified as regular expression:

```
> en1<-extractSeqids(ens,"^1$")
> en1
```

```
Object of class 'ensemblGenome' with 111 rows and 14 columns.
   seqid         source feature start   end score strand
25     1     pseudogene    exon 11869 12227     .      +
34     1 protein_coding    exon 11872 12227     .      +
41     1 protein_coding    exon 11874 12227     .      +
28     1     pseudogene    exon 12010 12057     .      +
29     1     pseudogene    exon 12179 12227     .      +
35     1 protein_coding     CDS 12190 12227     .      +
   frame id         gene_id   transcript_id  gene_name
25     . 25 ENSG00000223972 ENST00000456328    DDX11L1
34     . 34 ENSG00000249291 ENST00000515242 AL627309.2
41     . 41 ENSG00000253101 ENST00000518655   DDX11L11
28     . 28 ENSG00000223972 ENST00000450305    DDX11L1
29     . 29 ENSG00000223972 ENST00000450305    DDX11L1
35     0 35 ENSG00000249291 ENST00000515242 AL627309.2
   transcript_name exon_number
25     DDX11L1-002           1
34  AL627309.2-201           1
41    DDX11L11-201           1
28     DDX11L1-001           1
29     DDX11L1-001           2
35  AL627309.2-201           1
```

It looks cumbersome for single chromosomes but allows extraction of complex patterns.

4.2 Extracting primary assembly data

Usually the interesting part of the annotation data is the the primary assembly (where alternative haplotypes are excluded). Therefore functions which return the proper terms are supplied:

```
> ensPrimAssembly()

[1] "^([0-9]{1,2})$|^[XY]|MT$"

> ucPrimAssembly()

[1] "^chr[0-9XYM]{1,2}$"
```

Extraction of primary assembly `seqid`'s i is done by:

```
> enpa<-extractSeqids(ens,ensPrimAssembly())
> tableSeqids(enpa)

  1
111

> ucpa<-extractSeqids(uc,ucPrimAssembly())
> tableSeqids(ucpa)

chr1
   6
```

4.3 Extract features

Subsets defined by `features` can allso be tabled and extracted:

```
> tableFeatures(enpa)

        CDS        exon start_codon  stop_codon
          8          98           3           2

> enpf<-extractFeature(enpa,"exon")
> enpf

Object of class 'ensemblGenome' with 98 rows and 14 columns.
   seqid         source feature start   end score strand
25     1     pseudogene    exon 11869 12227     .      +
34     1 protein_coding    exon 11872 12227     .      +
41     1 protein_coding    exon 11874 12227     .      +
28     1     pseudogene    exon 12010 12057     .      +
29     1     pseudogene    exon 12179 12227     .      +
42     1 protein_coding    exon 12595 12721     .      +
   frame id         gene_id   transcript_id  gene_name
25     . 25 ENSG00000223972 ENST00000456328    DDX11L1
34     . 34 ENSG00000249291 ENST00000515242 AL627309.2
41     . 41 ENSG00000253101 ENST00000518655   DDX11L11
28     . 28 ENSG00000223972 ENST00000450305    DDX11L1
29     . 29 ENSG00000223972 ENST00000450305    DDX11L1
42     . 42 ENSG00000253101 ENST00000518655   DDX11L11
   transcript_name exon_number
25     DDX11L1-002           1
34  AL627309.2-201           1
41    DDX11L11-201           1
28     DDX11L1-001           1
29     DDX11L1-001           2
42    DDX11L11-201           2
```

4.4 Extract data for single genes and transcripts

There are some functions which extract objects that contain data for single genes (or transcripts). These functions provide a closer insight into specific regeions.

Objects which contain data for single genes can be extracted with

```
> dxe<-extractByGeneName(enpa,"DDX11L1")
> dxu<-extractByGeneName(ucpa,"DDX11L1")
```

From these extracts we can view the contained transcripts with the `tableTranscript.id` function:

```
> tableTranscript.id(dxe)

ENST00000450305 ENST00000456328
              6               3
```

```
> tableTranscript.id(dxu)

uc001aaa.3 uc010nxr.1
         3          3
```

Data for interesting transcripts can be extracted by `extractTranscript`:

```
> extractTranscript(dxe,"ENST00000456328")

Object of class 'ensemblGenome' with 3 rows and 14 columns.
    transcript_id gene_name seqid     source feature start
1 ENST00000456328   DDX11L1     1 pseudogene    exon 11869
2 ENST00000456328   DDX11L1     1 pseudogene    exon 12613
3 ENST00000456328   DDX11L1     1 pseudogene    exon 13221
    end score strand frame id         gene_id
1 12227     .      +     . 25 ENSG00000223972
2 12721     .      +     . 26 ENSG00000223972
3 14409     .      +     . 27 ENSG00000223972
  transcript_name exon_number
1     DDX11L1-002           1
2     DDX11L1-002           2
3     DDX11L1-002           3

> extractTranscript(dxu,"uc010nxr.1")

Object of class 'ucscGenome' with 3 rows and 14 columns.
  transcript_id gene_name seqid         source feature
1    uc010nxr.1   DDX11L1  chr1 hg19_knownGene    exon
2    uc010nxr.1   DDX11L1  chr1 hg19_knownGene    exon
3    uc010nxr.1   DDX11L1  chr1 hg19_knownGene    exon
  start   end score strand frame id    gene_id
1 11874 12227     0      +     .  4 uc010nxr.1
2 12646 12697     0      +     .  5 uc010nxr.1
3 13221 14409     0      +     .  6 uc010nxr.1
         ensembl clusterId
1 ENST00000456328         1
2 ENST00000456328         1
3 ENST00000456328         1
```

5 Accumulate data for whole genes

The function `getGenePositions` accumulates position data for whole genes. Genes are grouped by `gene_name` . So this attribute type should be moved into the `gtf` table via `moveAttributes`. Due to inclusion of `exon_number` data into the table, this attribute type should also be moved into the `gtf` table.

```
> gpe<-getGenePositions(enpa)
> gpe

  seqid start   end  gene_name         gene_id exon_number
2     1 11869 14409    DDX11L1 ENSG00000223972           6
```

```
6     1 11872 14412 AL627309.2 ENSG00000249291           3
7     1 11874 14409   DDX11L11 ENSG00000253101           4
3     1 14363 29806     WASH7P ENSG00000227232           9
5     1 29554 31109 MIR1302-10 ENSG00000243485           3
1     1 30366 30503 MIR1302-10 ENSG00000221311           1
4     1 34554 36081    FAM138A ENSG00000237613           3
```

```
> gpu<-getGenePositions(ucpa)
> gpu
```

```
  seqid start   end strand    gene_id gene_name
1  chr1 11874 14409      + uc001aaa.3   DDX11L1
2  chr1 11874 14409      + uc010nxr.1   DDX11L1
         source
1 hg19_knownGene
2 hg19_knownGene
```

6 Overlapping

The overlap function is used to supply annotation for genomic ranges. The function takes two `data.frame`'s which contain query (qry) and reference (ref) ranges respectively. Each dataset will be identified by it's id.

The routine assumes that query and reference tables are ascending sorted by column 'start'. Otherwise the result will be incorrect (i.e. missing hits). The function assumes that there is no overlap between reference ranges. It will otherwise return only one, possibly arbitrary, hit per query range.

The function returns a `data.frame`. For each query range, there will be one row.

```
> qry<-data.frame(
+                   id=1:6,
+                   start=c(10,18,61,78,82,110),
+                   end=c(15,22,63,87,90,120))
> ref<-data.frame(
+                   id=1:5,
+                   start=c(20,40,60,80,100),
+                   end=c(25,45,65,85,105))
> overlap(qry,ref)
```

```
  overlap leftDiff rightDiff queryid refid
0      no        0         5       1     0
1       l        2         3       2     1
2       n        1         2       3     3
3       b        2         2       4     4
4       r        2         5       5     4
5      no        5         0       6     0
```

The query and reference record are identified by "queryid" and "refid". The type of overlap is encoded in the "overlap" column. The overlap encodings are explained as follows:

- **no**. The query range does not overlap with any reference ranges.
- **l** The query range overhangs the matching reference range on the left (lower coordinate) side.
- **n** The query range is completely contained within a reference range. There is no overhang.
- **b** The query range overhangs the matching reference range on both sides.
- **r** The query range overhangs the matching reference range on the right (higher coordinate) side.

The added "leftDiff" and "rightDiff" columns contain the distance between the query and reference range boundaries: leftDiff is the difference between the left (lower coordinate) margins and rightDiff is the difference between the right (higher coordinate) margins.