

# Summary of work on Organism.dplyr and MultiExperimentDb

*Yubo Cheng*

13 January 2017

## Contents

1	Abstract . . . . .	2
2	Introduction . . . . .	2
3	Representing and accessing SQL-based bioinformatics re-sources . . . . .	2
3.1	Background . . . . .	2
3.2	Methods . . . . .	2
3.3	Results. . . . .	3
3.4	Discussion . . . . .	11
4	Coordinated on-disk representation of multiple bioinformatic experiments . . . . .	11
4.1	Background . . . . .	11
4.2	Methods . . . . .	11
4.3	Results. . . . .	12
4.4	Discussion . . . . .	21
5	Summary and Conclusions . . . . .	21
6	References . . . . .	21
7	Appendix. . . . .	22

# 1 Abstract

---

Bioinformatics has become an important part of many areas of biology, and the methodology of extracting useful results from large amounts of raw data effectively and efficiently plays an essential role in analyzing and understanding bioinformatics data. Bioconductor contains plenty of resources of annotation data and experiment data, but it is not always easy and fast to query them.

Two R packages: `Organism.dplyr` and `MultiExperimentDb` are developed for easier use of genome wide annotation packages and comparison between multiple experiments. `Organism.dplyr` provides an integrated presentation of mapping between organism level information and genomic coordinates information, while `MultiExperimentDb` provides functionality for storing and comparing multiple bioinformatics experiments which contains large matrix data.

# 2 Introduction

---

# 3 Representing and accessing SQL-based bioinformatics resources

---

## 3.1 Background

Annotation packages in *Bioconductor* which hold organism level and genomic coordinates information are being used more and more by users who analyze of high-throughput genomic data.

`OrgDb` packages are used for mapping between a central gene identifier and other kinds of identifiers, while `TxDb` packages for connecting a set of genomic coordinates to various transcript oriented features. However, since organism level information and genomic coordinates information are stored in different packages/databases, mapping between gene identifiers and genomic ranges is not easy. For example, finding out transcript ranges according to accession number.

`Organism.dplyr` provides an alternative interface of gene identifier mapping functionality of the `OrgDb` packages (e.g., `org.Hs.eg.db`) and the genome coordinate functionality of the `TxDb` packages (e.g., `TxDb.Hsapiens.UCSC.hg38.knownGene`).

## 3.2 Methods

### 3.2.1 Approach

#### 3.2.1.1 Data representation

Package `Organism.dplyr` stores data using on disk sqlite database file combining data from a given '`TxDb`' package and the corresponding '`org`' package, thus one database contains information of gene identifiers and genomic coordinates. Once the database is created and stored at given directory, it can be accessed another time.

## Summary of work on Organism.dplyr and MultiExperimentDb

Data of OrgDb and TxDb are reconstructed and new tables are created in sqlite file. SQL codes for creating the database, including both schema and data, are stored in sql files, one file for each organism, separating from the rest of R codes. This strategy makes the SQL codes clean and easy to maintain.

A *src\_organism* object is created by this package to point to the database. R Package `dplyr` (a grammar of data manipulation) could be applied to object created by this package.

### 3.2.1.2 Software development

Best practices are applied to the software development:

- Unit tests are created to make sure each module of the codes works properly.
- R Package `roxygen2` is used for documentation of namespace and help pages.
- Object oriented programming is used for developing this package, and repeated codes are extracted to separate functions for reuse.

## 3.2.2 Features

This package provides functionalities of gene identifiers (entrez id, gene symbol, ensembl id, accession number, ipi number, go id, etc.) mapping, as well as mapping between identifiers and range coordinates (including different levels: gene, transcription, exon and cds).

Filters can be applied to genomic coordinates extractor functions, and all columns in the database can be possible filters. The filter functions give users flexibility of extracting genomic coordinates using combination of different conditions.

With all the data stored in sqlite file on disk, users need to create a *src\_organism* object to point to the database file, and only necessary data is brought into *R* instead of the whole database for each *R* command.

## 3.3 Results

### 3.3.1 Constructing a *src\_organism*

#### 3.3.1.1 Make sqlite database from 'TxDb' package

The `src_organism()` constructor creates an on disk sqlite database file with data from a given 'TxDb' package and corresponding 'org' package. When dbpath is given, file is created at the given path, otherwise temporary file is created.

Running `src_organism()` without a given path will save the sqlite file to a tempdir():

```
src <- src_organism("TxDb.Hsapiens.UCSC.hg38.knownGene")
```

Alternatively you can provide explicit path to where the sqlite file should be saved.

```
src <- src_organism("TxDb.Hsapiens.UCSC.hg38.knownGene", "path/to/save/sqlite")
```

`supportedOrganisms()` provides a list of organisms with corresponding 'org' and 'TxDb' packages being supported.

## Summary of work on Organism.dplyr and MultiExperimentDb

```
supportedOrganisms()
## # A tibble: 21 × 3
##       organism      OrgDb
##       <chr>      <chr>
## 1      Bos taurus org.Bt.eg.db
## 2 Caenorhabditis elegans org.Ce.eg.db
## 3 Caenorhabditis elegans org.Ce.eg.db
## 4      Canis familiaris org.Cf.eg.db
## 5 Drosophila melanogaster org.Dm.eg.db
## 6 Drosophila melanogaster org.Dm.eg.db
## 7      Danio rerio org.Dr.eg.db
## 8      Gallus gallus org.Gg.eg.db
## 9      Homo sapiens org.Hs.eg.db
## 10     Homo sapiens org.Hs.eg.db
## # ... with 11 more rows, and 1 more variables: TxDb <chr>
```

### 3.3.1.2 Make sqlite database from organism name

Organism name, genome and id could be specified to create sqlite database. Organism name (either Organism or common name) must be provided to create the database, if genome and/or id are not provided, most recent 'TxDb' package is used.

```
src <- src_ucsc("human", "path/to/save/sqlite")
```

### 3.3.1.3 Access existing sqlite file

Existing on disk sqlite file can be accessed without recreating the database.

```
path <- system.file("extdata", package = "Organism.dplyr")
src <- src_organism(dbpath = paste0(path, "/example.sqlite"))
src
## src:  sqlite 3.11.1 [C:\Program Files\R\R-devel\library\Organism.dplyr\extdata\example.sqlite]
## tbls: id, id_accession, id_go, id_go_all, id_omim_pm, id_protein,
##       id_transcript, ranges_cds, ranges_exon, ranges_gene, ranges_tx
```

## 3.3.2 Common operations

### 3.3.2.1 Basic operations

All methods from package [dplyr][1] can be used for a *src\_organism* object.

Look at all available tables.

```
src_tbls(src)
## [1] "id"          "id_accession" "id_go"        "id_go_all"
## [5] "id_omim_pm"  "id_protein"   "id_transcript" "ranges_cds"
## [9] "ranges_exon" "ranges_gene"  "ranges_tx"
```

Look at data from one specific table.

## Summary of work on Organism.dplyr and MultiExperimentDb

```
tbl(src, "id")
## Source:   query [?? x 6]
## Database: sqlite 3.11.1 [C:\Program Files\R\R-devel\library\Organism.dplyr\extdata\example.sqlite]
##
##      entrez      map      ensembl  symbol
##      <chr>    <chr>      <chr>    <chr>
## 1  100506674    5p12      <NA>    BRCAT54
## 2  102723839    <NA>      <NA>    BRCAT107
## 3    394269    17q21      <NA>    BRCA1P1
## 4    394269    17q21      <NA>    BRCA1P1
## 5    394269    17q21      <NA>    BRCA1P1
## 6    394269    17q21      <NA>    BRCA1P1
## 7      5728 10q23.3 ENSG00000171862 PTEN
## 8      5728 10q23.3 ENSG00000171862 PTEN
## 9      5728 10q23.3 ENSG00000171862 PTEN
## 10     5728 10q23.3 ENSG00000171862 PTEN
## # ... with more rows, and 2 more variables: genename <chr>, alias <chr>
```

Look at fields of one table.

```
colnames(tbl(src, "id"))
## [1] "entrez" "map"      "ensembl" "symbol"  "genename" "alias"
```

Below are some examples of querying tables using dplyr.

### 1. Gene symbol start with "BRCA"

```
tbl(src, "id") %>%
  filter(symbol %like% "BRCA%") %>%
  dplyr::select(entrez, map, ensembl, symbol) %>%
  distinct() %>% arrange(symbol) %>% collect()
## # A tibble: 7 x 4
##      entrez      map      ensembl  symbol
##      <chr>    <chr>      <chr>    <chr>
## 1      672    17q21 ENSG0000012048 BRCA1
## 2    394269    17q21    <NA>    BRCA1P1
## 3      675 13q12.3 ENSG00000139618 BRCA2
## 4     60500    13q21    <NA>    BRCA3
## 5 102723839    <NA>      <NA>    BRCAT107
## 6 100506674    5p12      <NA>    BRCAT54
## 7      8068   11q23    <NA>    BRCATA
```

### 2. Go info for gene symbol "PTEN"

```
inner_join(tbl(src, "id"), tbl(src, "id_go")) %>%
  filter(symbol == "PTEN") %>%
  dplyr::select(entrez, ensembl, symbol, go, evidence, ontology)
## Joining, by = "entrez"
## Source:   query [?? x 6]
## Database: sqlite 3.11.1 [C:\Program Files\R\R-devel\library\Organism.dplyr\extdata\example.sqlite]
##
##      entrez      ensembl symbol      go evidence ontology
```

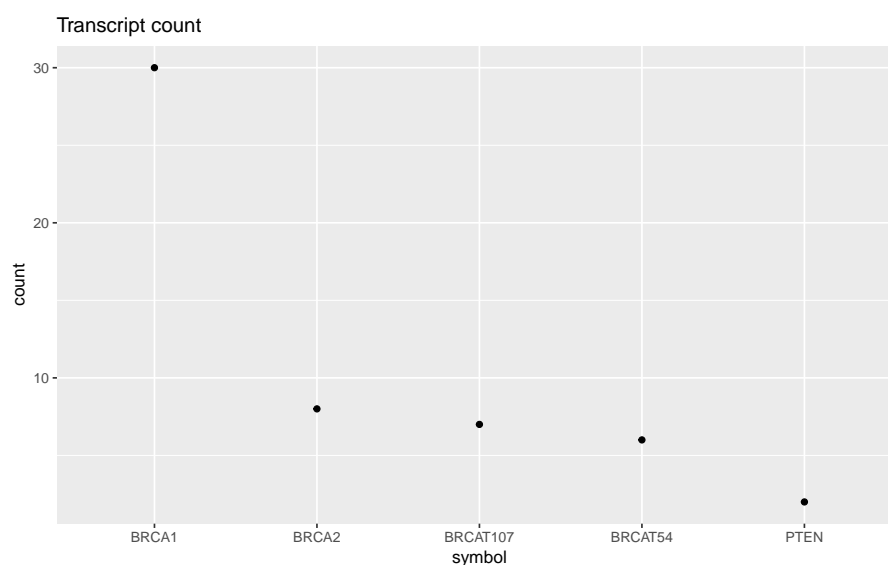
## Summary of work on Organism.dplyr and MultiExperimentDb

```
##      <chr>      <chr> <chr>      <chr>      <chr>      <chr>
## 1    5728 ENSG00000171862 PTEN G0:0000079      TAS      BP
## 2    5728 ENSG00000171862 PTEN G0:0000287      IEA      MF
## 3    5728 ENSG00000171862 PTEN G0:0001525      IEA      BP
## 4    5728 ENSG00000171862 PTEN G0:0001933      IDA      BP
## 5    5728 ENSG00000171862 PTEN G0:0001933      ISS      BP
## 6    5728 ENSG00000171862 PTEN G0:0002902      IEA      BP
## 7    5728 ENSG00000171862 PTEN G0:0004438      IDA      MF
## 8    5728 ENSG00000171862 PTEN G0:0004721      IDA      MF
## 9    5728 ENSG00000171862 PTEN G0:0004722      IDA      MF
## 10   5728 ENSG00000171862 PTEN G0:0004725      IDA      MF
## # ... with more rows
```

### 3. Genes transcripts count

```
txcount <- inner_join(tbl(src, "id"), tbl(src, "ranges_tx")) %>%
  dplyr::select(symbol, tx_id) %>%
  group_by(symbol) %>%
  summarise(count = count(symbol)) %>%
  dplyr::select(symbol, count) %>%
  arrange(desc(count)) %>%
  collect(n=Inf)
## Joining, by = "entrez"

txcount
## # A tibble: 5 × 2
##   symbol count
##   <chr> <int>
## 1 BRCA1    30
## 2 BRCAT54    8
## 3 BRCA2     7
## 4 PTEN      6
## 5 BRCAT107    2
```



## Summary of work on Organism.dplyr and MultiExperimentDb

### 4. Gene coordinates of symbol "PTEN" and "BRCA1" as *GRanges*

```
inner_join(tbl(src, "id"), tbl(src, "ranges_gene")) %>%
  filter(symbol %in% c("PTEN", "BRCA1")) %>%
  dplyr::select(gene_chrom, gene_start, gene_end, gene_strand,
               symbol, map) %>%
  collect() %>% GenomicRanges::GRanges()
## Joining, by = "entrez"
## GRanges object with 2 ranges and 2 metadata columns:
##      seqnames      ranges strand |      symbol      map
##      <Rle>         <IRanges> <Rle> | <character> <character>
## [1]   chr10 [87863113, 87971930]   + |      PTEN    10q23.3
## [2]   chr17 [43044295, 43170245]   - |     BRCA1    17q21
## -----
## seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

### 3.3.2.2 "select" interface

Methods `select()`, `keytypes()`, `keys()`, `columns()` and `mapIds` from *AnnotationDbi* are implemented for *src\_organism* objects.

#### 1. `keytypes()`

Use `keytypes()` to discover which keytypes can be passed to `keytype` argument of methods `select()` or `keys()`.

```
keytypes(src)
## [1] "accnum"      "alias"       "cds_chrom"   "cds_end"
## [5] "cds_id"      "cds_name"    "cds_start"   "cds_strand"
## [9] "ensembl"     "ensemblprot" "ensembltrans" "entrez"
## [13] "enzyme"      "evidence"    "evidenceall" "exon_chrom"
## [17] "exon_end"    "exon_id"     "exon_name"   "exon_rank"
## [21] "exon_start"  "exon_strand" "gene_chrom"   "gene_end"
## [25] "gene_start"  "gene_strand" "genename"     "go"
## [29] "goall"       "ipi"         "map"         "omim"
## [33] "ontology"    "ontologyall" "pfam"        "pmid"
## [37] "prosite"     "refseq"      "symbol"      "tx_chrom"
## [41] "tx_end"      "tx_id"       "tx_name"     "tx_start"
## [45] "tx_strand"   "tx_type"     "unigene"     "uniprot"
```

#### 2. `columns()`

Use `columns()` to discover which kinds of data can be returned for the *src\_organism* object.

```
columns(src)
## [1] "accnum"      "alias"       "cds_chrom"   "cds_end"
## [5] "cds_id"      "cds_name"    "cds_start"   "cds_strand"
## [9] "ensembl"     "ensemblprot" "ensembltrans" "entrez"
## [13] "enzyme"      "evidence"    "evidenceall" "exon_chrom"
## [17] "exon_end"    "exon_id"     "exon_name"   "exon_rank"
## [21] "exon_start"  "exon_strand" "gene_chrom"   "gene_end"
## [25] "gene_start"  "gene_strand" "genename"     "go"
## [29] "goall"       "ipi"         "map"         "omim"
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
## [33] "ontology"      "ontologyall"  "pfam"         "pmid"
## [37] "prosite"       "refseq"       "symbol"        "tx_chrom"
## [41] "tx_end"        "tx_id"        "tx_name"       "tx_start"
## [45] "tx_strand"     "tx_type"      "unigene"       "uniprot"
```

### 3. keys()

`keys()` returns keys for the *src\_organism* object. By default it returns the primary keys for the database, and returns the keys from that keytype when the keytype argument is used.

Keys of entrez

```
head(keys(src))
## [1] "100506674" "102723839" "394269"      "5728"        "60500"       "672"
```

Keys of symbol

```
head(keys(src, "symbol"))
## [1] "BRCA1"      "BRCA1P1"    "BRCA2"      "BRCA3"      "BRCAT107"   "BRCAT54"
```

### 4. select()

`select()` retrieves the data as a *tibble* based on parameters for selected keys columns and keytype arguments. If requested columns that have multiple matches for the keys, `select_tbl()` will return a *tibble* with one row for each possible match, and `select()` will return a data frame.

```
keytype <- "symbol"
keys <- c("PTEN", "BRCA1")
columns <- c("entrez", "tx_id", "tx_name", "exon_id")
select_tbl(src, keys, columns, keytype)
## Joining, by = "entrez"
## Source:   query [?? x 5]
## Database: sqlite 3.11.1 [C:\Program Files\R\R-devel\library\Organism.dplyr\extdata\example.sqlite]
##
##   symbol entrez  tx_id   tx_name exon_id
##   <chr>  <chr>  <int>    <chr>   <int>
## 1 BRCA1    672 147976 uc002icq.4 439447
## 2 BRCA1    672 147976 uc002icq.4 439452
## 3 BRCA1    672 147976 uc002icq.4 439453
## 4 BRCA1    672 147976 uc002icq.4 439455
## 5 BRCA1    672 147976 uc002icq.4 439456
## 6 BRCA1    672 147976 uc002icq.4 439457
## 7 BRCA1    672 147976 uc002icq.4 439460
## 8 BRCA1    672 147976 uc002icq.4 439462
## 9 BRCA1    672 147976 uc002icq.4 439464
## 10 BRCA1   672 147976 uc002icq.4 439465
## # ... with more rows
```

### 5. mapIds()

`mapIds()` gets the mapped ids (column) for a set of keys that are of a particular keytype. Usually returned as a named character vector.



## Summary of work on Organism.dplyr and MultiExperimentDb

```
mapIds(src, keys, column = "tx_name", keytype)
## Joining, by = "entrez"
##          PTEN          BRCA1
## "uc001kfb.4" "uc002icq.4"
```

### 3.3.2.3 Genomic coordinates extractors

Eleven genomic coordinates extractor methods are available in this package: `transcripts()`, `exons()`, `cds()`, `genes()`, `promoters()`, `transcriptsBy()`, `exonsBy()`, `cdsBy()`, `intronsByTranscript()`, `fiveUTRsByTranscript()`, `threeUTRsByTranscript()`. Two versions of results are provided: *tibble* (`transcripts_tbl()`) and *GRanges* or *GRangesList* (`transcripts()`).

Filter can be applied to all extractor functions. A named list of vectors can be used to restrict the output, valid filters can be retrieved by `possibleFilters()`.

```
possibleFilters()
## [1] "AccnumFilter"      "AliasFilter"      "Cds_chromFilter"
## [4] "Cds_idFilter"      "Cds_nameFilter"   "Cds_strandFilter"
## [7] "EnsemblFilter"     "EnsemblprotFilter" "EnsembltransFilter"
## [10] "EntrezFilter"      "EnzymeFilter"     "EvidenceFilter"
## [13] "EvidenceallFilter" "Exon_chromFilter" "Exon_idFilter"
## [16] "Exon_nameFilter"   "Exon_rankFilter"  "Exon_strandFilter"
## [19] "FlybaseFilter"     "Flybase_cgFilter" "Flybase_protFilter"
## [22] "Gene_chromFilter"  "Gene_strandFilter" "GenenameFilter"
## [25] "GoFilter"          "GoallFilter"      "IpiFilter"
## [28] "MapFilter"         "MgiFilter"        "OmimFilter"
## [31] "OntologyFilter"    "OntologyallFilter" "PfamFilter"
## [34] "PmidFilter"        "PrositFilter"     "RefseqFilter"
## [37] "SymbolFilter"      "Tx_chromFilter"   "Tx_idFilter"
## [40] "Tx_nameFilter"     "Tx_strandFilter"  "Tx_typeFilter"
## [43] "UnigeneFilter"     "UniprotFilter"    "WormbaseFilter"
## [46] "ZfinFilter"        "Cds_startFilter"  "Cds_endFilter"
## [49] "Exon_startFilter"  "Exon_endFilter"   "Gene_startFilter"
## [52] "Gene_endFilter"    "Tx_startFilter"   "Tx_endFilter"
```

All filters take two parameters: value and condition, condition could be one of "=", "!=" , "startsWith", "endsWith", ">", "<", ">=" and "<=", default condition is "=".

```
EnsemblFilter("ENSG00000171862")
## class: EnsemblFilter
## condition: ==
## value: ENSG00000171862
SymbolFilter("BRCA", "startsWith")
## class: SymbolFilter
## condition: startsWith
## value: BRCA
```

Besides, `GRangesFilter()` could also be used as filter for the methods with result displaying as *GRanges* or *GRangesList*.

```
filters <- list(SymbolFilter(c("PTEN", "BRCA1")),
                EntrezFilter(5728),
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
GRangesFilter(as("chr10:87869000-87876000", "GRanges"))
transcripts_tbl(src, filter=filters)
## filter by 'granges' only supported by methods returning GRanges or GRangesList
## Joining, by = "entrez"
## Source:  query [?? x 8]
## Database: sqlite 3.11.1 [C:\Program Files\R\R-devel\library\Organism.dplyr\extdata\example.sqlite]
##
##   tx_chrom tx_start  tx_end tx_strand tx_id    tx_name symbol entrez
##   <chr>    <int>    <int>    <chr> <int>    <chr>  <chr>  <chr>
## 1   chr10  87863113 87971930      + 87010 uc001kfb.4 PTEN   5728
## 2   chr10  87863438 87942691      + 87011 uc057ush.1 PTEN   5728
## 3   chr10  87864449 87867049      + 87012 uc057usi.1 PTEN   5728
## 4   chr10  87864468 87894326      + 87013 uc057usj.1 PTEN   5728
## 5   chr10  87925523 87933487      + 87016 uc057usm.1 PTEN   5728
## 6   chr10  87952199 87961309      + 87017 uc057usn.1 PTEN   5728
transcripts(src, filter=filters)
## Joining, by = "entrez"
## GRanges object with 3 ranges and 4 metadata columns:
##      seqnames          ranges strand |      tx_id    tx_name
##      <Rle>            <IRanges> <Rle> | <integer> <character>
## [1]   chr10 [87863113, 87971930]      + |    87010 uc001kfb.4
## [2]   chr10 [87863438, 87942691]      + |    87011 uc057ush.1
## [3]   chr10 [87864468, 87894326]      + |    87013 uc057usj.1
##      symbol      entrez
##      <character> <character>
## [1]      PTEN      5728
## [2]      PTEN      5728
## [3]      PTEN      5728
## -----
## seqinfo: 455 sequences (1 circular) from hg38 genome
```

Transcript coordinates of gene symbol equal to “PTEN” or “BRCA1”, and transcript start position between 87863438 and 87933487.

```
transcripts_tbl(src, filter = list(
  SymbolFilter(c("PTEN", "BRCA1")),
  Tx_startFilter(87863438, ">="),
  Tx_startFilter(87933487, "<=")
))
## Joining, by = "entrez"
## Source:  query [?? x 7]
## Database: sqlite 3.11.1 [C:\Program Files\R\R-devel\library\Organism.dplyr\extdata\example.sqlite]
##
##   tx_chrom tx_start  tx_end tx_strand tx_id    tx_name symbol
##   <chr>    <int>    <int>    <chr> <int>    <chr>  <chr>
## 1   chr10  87863438 87942691      + 87011 uc057ush.1 PTEN
## 2   chr10  87864449 87867049      + 87012 uc057usi.1 PTEN
## 3   chr10  87864468 87894326      + 87013 uc057usj.1 PTEN
## 4   chr10  87925523 87933487      + 87016 uc057usm.1 PTEN
```

### 3.4 Discussion

#### 3.4.1 Assessment of Organism.dplyr

##### 3.4.1.1 Strengths

- Combine data of gene identifiers and genomic coordinates into one sqlite file
- Provide flexibility of filters for genomic coordinates extractor functions
- sqlite file can be stored on disk and it is easy to access multiple times

##### 3.4.1.2 Weakness

- It takes longer time to create sqlite file the first time
- The sqlite file could be big in size

#### 3.4.2 Future development

- Make filter functions more flexible by adding conditions (and, or) between filters
- Support more organisms

## 4 Coordinated on-disk representation of multiple bioinformatic experiments

---

### 4.1 Background

Package MultiExperimentDb provides functionality for storing and comparing multiple *SummarizedExperiment* objects, the data can be added to one object and stored on disk for reuse.

This package is designed for comparing data between different experiments, the experiments added to a *MultiExperimentDb* object should have some similarity such as common features or samples. This overlap allows features or samples to be extracted and viewed together across experiments by combining by row or column.

### 4.2 Methods

Package MultiExperimentDb creates a *MultiExperimentDb* object to store all data on disk, with assays data in matrix format stored in HDF5 file, and annotation data like rowData, colData, rowRanges stored in sqlite database.

This design reduces the overall memory footprint and can provide faster random access to subsets of data because we are indexing into data on disk vs making copies of in-memory objects. By storing large matrices on disk in HDF5 file and displaying in R using *DelayedMatrix* object, minimal data needs to be brought into R. This design fastens matrix data manipulation, including subset, binding, etc.

## Summary of work on Organism.dplyr and MultiExperimentDb

Data of *MultiExperimentDb* object with multiple SummarizedExperiments is stored in one sqlite file and one HDF5 file. Assay data in large matrix format is stored in HDF5 file, when one experiment is added, assay data of that experiment is added to the HDF5 file as one dataset with experiment name as dataset name. Rownames, colnames, rowdata, coldata, rowranges are stored in one sqlite database file and each experiment represented by one unique index and one unique experiment name.

### 4.3 Results

#### 4.3.1 Constructing a *MultiExperimentDb*

The `MultiExperimentDb()` constructor creates an empty *MultiExperimentDb* instance. When `hdf5path` and `sqlitepath` are given, files are created at the given path(s), otherwise temporary files are created. An empty *MultiExperimentDb* object needs to be created as first step before adding any experiments.

Creating a *MultiExperimentDb* instance without given paths will save the hdf5 and sqlite files in a `tempfile()`:

```
library(MultiExperimentDb)
medb <- MultiExperimentDb()
```

Alternatively you can provide explicit paths to where the files should be saved.

```
medb <- MultiExperimentDb(hdf5path = "path/to/save/hdf5/",
                          sqlitepath = "path/to/save/sqlite")
```

The class supports common operations such as `length()`, `dim()`, `dimnames()` etc.

`loadMultiExperimentDb(hdf5path, sqlitepath)` can be used to create a *MultiExperimentDb* object from existing hdf5 and sqlite files stored on disk. It can be used when experiments are added to *MultiExperimentDb* object and saved on disk, then need to be accessed another time.

```
path <- system.file("extdata", package = "MultiExperimentDb")
medb <- loadMultiExperimentDb(paste0(path, "/medb.h5"),
                             paste0(path, "/medb.sqlite"))

medb
## class: MultiExperimentDb
## hdf5path: C:/Program Files/R/R-devel/library/MultiExperimentDb/extdata/medb.h5
## sqlitepath: C:/Program Files/R/R-devel/library/MultiExperimentDb/extdata/medb.sqlite
## dim: 1300 8
## experiments:
##   geuFPKM1 (1000 x 6)
##     rownames: ENSG00000152931.6, ENSG00000183696.9,
##               ENSG00000139269.2, ..., ENSG00000161016.10, ENSG00000150787.3
##     colnames: HG00096, HG00097, HG00099, HG00100, HG00101, HG00102
##   geuFPKM2 (1001 x 6)
##     rownames: ENSG00000115211.10, ENSG00000231419.2,
##               ENSG00000196233.6, ..., ENSG00000167460.9, ENSG00000171208.5
##     colnames: HG00099, HG00100, HG00101, HG00102, HG00103, HG00104
```

## Summary of work on Organism.dplyr and MultiExperimentDb

Get sqlite database path and HDF5 file path of a *MultiExperimentDb* object with the `hdf5path()` and `sqlitepath()` accessors.

```
hdf5path(medb)
## [1] "C:/Program Files/R/R-devel/library/MultiExperimentDb/extdata/medb.h5"
sqlitepath(medb)
## [1] "C:/Program Files/R/R-devel/library/MultiExperimentDb/extdata/medb.sqlite"
```

### 4.3.2 Common operations on a *MultiExperimentDb*

#### 4.3.2.1 Adding data

An experiment (i.e., *SummarizedExperiment* object) can be added to a *MultiExperimentDb* instance with `addExperiment()`. Experiment names must be unique.

Add data to the *MultiExperimentDb*.

```
medb <- MultiExperimentDb()
library(geuvPack)
data(geuFPKM)
medb <- addExperiment(medb, geuFPKM[1:1000, 1:6], "geuFPKM1")
medb <- addExperiment(medb, geuFPKM[300:1300, 3:8], "geuFPKM2")
experimentNames(medb)
## [1] "geuFPKM1" "geuFPKM2"
medb
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944719d928.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29443d1f77d3.sqlite
## dim: 1300 8
## experiments:
##   geuFPKM1 (1000 x 6)
##     rownames: ENSG00000152931.6, ENSG00000183696.9,
##       ENSG00000139269.2, ..., ENSG00000161016.10, ENSG00000150787.3
##     colnames: HG000096, HG000097, HG000099, HG00100, HG00101, HG00102
##   geuFPKM2 (1001 x 6)
##     rownames: ENSG00000115211.10, ENSG00000231419.2,
##       ENSG00000196233.6, ..., ENSG00000167460.9, ENSG00000171208.5
##     colnames: HG000099, HG00100, HG00101, HG00102, HG00103, HG00104
```

#### 4.3.2.2 Extract experiment and assay data

The `experiment()` function extracts a single experiment from a *MultiExperimentDb* as a *SummarizedExperiment* object, all methods of *SummarizedExperiment* can be applied, such as `colData()`, `rowRanges()`, etc.

```
se <- experiment(medb, "geuFPKM1")
colData(se)[, 1:3]
## DataFrame with 6 rows and 3 columns
##       Source.Name Comment.ENA_SAMPLE. Characteristics.Organism.
##       <character>          <factor>          <factor>
## HG000096      HG000096      ERS185276      Homo sapiens
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
## HG00097      HG00097      ERS185206      Homo sapiens
## HG00099      HG00099      ERS185128      Homo sapiens
## HG00100      HG00100      ERS185086      Homo sapiens
## HG00101      HG00101      ERS185085      Homo sapiens
## HG00102      HG00102      ERS185453      Homo sapiens
rowRanges(se)[, 1:3]
## GRanges object with 1000 ranges and 3 metadata columns:
##           seqnames      ranges strand | source
##           <Rle>         <IRanges> <Rle> | <factor>
## ENSG00000152931.6 chr5 [ 59783540, 59843484] + | HAVANA
## ENSG00000183696.9 chr7 [ 48128225, 48148330] + | HAVANA
## ENSG00000139269.2 chr12 [ 57846106, 57853063] + | HAVANA
## ENSG00000169129.8 chr10 [116054583, 116164515] - | HAVANA
## ENSG00000134602.11 chrX [131157293, 131209971] + | HAVANA
##           ...      ...      ...      ...      ...
## ENSG00000169231.8 chr1 [155165379, 155178842] - | HAVANA
## ENSG00000250937.2 chr12 [ 8404007, 8450140] + | HAVANA
## ENSG00000123595.5 chrX [ 13707244, 13728625] + | HAVANA
## ENSG00000161016.10 chr8 [146015150, 146017972] - | HAVANA
## ENSG00000150787.3 chr11 [112097088, 112140678] + | HAVANA
##           type      score
##           <factor> <numeric>
## ENSG00000152931.6 gene      <NA>
## ENSG00000183696.9 gene      <NA>
## ENSG00000139269.2 gene      <NA>
## ENSG00000169129.8 gene      <NA>
## ENSG00000134602.11 gene      <NA>
##           ...      ...      ...
## ENSG00000169231.8 gene      <NA>
## ENSG00000250937.2 gene      <NA>
## ENSG00000123595.5 gene      <NA>
## ENSG00000161016.10 gene      <NA>
## ENSG00000150787.3 gene      <NA>
## -----
## seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

Get a single assay as *DelayedMatrix* by calling the `assay()` function with the *MultiExperimentDb* and experiment name. It returns the *HDF5Array* assay data corresponding to the current *MultiExperimentDb* row and column selections, example is given in subsetting section.

```
assay(medb, "geuFPKM1")[, 1:3]
## DelayedMatrix object of 1000 x 3 doubles:
##           HG00096      HG00097      HG00099
## ENSG00000152931.6 0.10185777 0.07810952 0.04898067
## ENSG00000183696.9 8.18380495 5.68691051 2.43465333
## ENSG00000139269.2 1.19991029 1.57357170 0.52161578
## ENSG00000169129.8 0.83193983 0.06977775 0.93108575
## ENSG00000134602.11 27.64642237 24.39557150 16.44537352
##           ...      .      .
## ENSG00000169231.8 1.415487e+00 1.250576e+00 1.244898e+00
## ENSG00000250937.2 -3.594809e-03 -1.205084e-02 1.081309e-02
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
## ENSG00000123595.5 2.934674e+01 2.854369e+01 2.315665e+01
## ENSG00000161016.10 1.457868e+03 1.359101e+03 8.191642e+02
## ENSG00000150787.3 4.776680e+00 4.479469e+00 1.990667e+00
```

### 4.3.2.3 Subsetting

A *MultiExperimentDb* object can be subset by overlapping rows or columns using standard numeric indices or feature/column names.

Subset all experiments by common rownames.

```
medb[c("ENSG00000171603.11", "ENSG00000230216.1"),,]
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944719d928.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29443d1f77d3.sqlite
## dim: 2 8
## experiments:
##   geuFPKM1 (2 x 6)
##     rownames: ENSG00000171603.11, ENSG00000230216.1
##     colnames: HG000096, HG000097, HG000099, HG00100, HG00101, HG00102
##   geuFPKM2 (2 x 6)
##     rownames: ENSG00000171603.11, ENSG00000230216.1
##     colnames: HG000099, HG00100, HG00101, HG00102, HG00103, HG00104
```

Return a subset of the “geuFPKM1” experiment with specific rows and column names.

```
medb[1:6,c("HG000099", "HG00101"), "geuFPKM1"]
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944719d928.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29443d1f77d3.sqlite
## dim: 6 2
## experiments:
##   geuFPKM1 (6 x 2)
##     rownames: ENSG00000152931.6, ENSG00000183696.9,
##               ENSG00000139269.2, ENSG00000169129.8, ENSG00000134602.11,
##               ENSG00000136237.12
##     colnames: HG000099, HG00101
```

A *MultiExperimentDb* object can also be subset by a *GRanges* object. The `grangesFromIdentifiers()` helper creates a *GRanges* from an *OrgDb*, *TxDb* and specified ‘keys’ and ‘keytypes’. This function uses `select()` with a specified *OrgDb* and *TxDb* package to convert given gene symbols or names to genomic positions. See `?grangesFromIdentifiers` man page for details.

Convert gene symbols PTEN and BRCA1 to genomic position:

```
granges <- grangesFromIdentifiers(org = "org.Hs.eg.db",
  keys = c("BRCA1", "CLSTN1", "WDR45"), keytype = "SYMBOL",
  txdb = "TxDb.Hsapiens.UCSC.hg38.knownGene")
## 'select()' returned 1:1 mapping between keys and columns
granges
## GRanges object with 3 ranges and 1 metadata column:
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
##          seqnames          ranges strand |      gene_id
##          <Rle>          <IRanges> <Rle> | <character>
##    11152      chrX [49074429, 49101170]   - |      11152
##    22883      chr1 [ 9729026, 9824526]   - |      22883
##      672     chr17 [43044295, 43170245]   - |        672
##    -----
##    seqinfo: 455 sequences (1 circular) from hg38 genome
```

Search all experiments by gene symbol BRCA1, CLSTN1 and WDR45:

```
medb <- medb[granges,,]
medb
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944719d928.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29443d1f77d3.sqlite
## dim: 3 8
## experiments:
##   geuFPKM1 (2 x 6)
##     rownames: ENSG00000171603.11, ENSG00000230216.1
##     colnames: HG000096, HG000097, HG000099, HG00100, HG00101, HG00102
##   geuFPKM2 (3 x 6)
##     rownames: ENSG00000171603.11, ENSG00000230216.1,
##               ENSG00000172992.6
##     colnames: HG000099, HG00100, HG00101, HG00102, HG00103, HG00104
assay(medb, "geuFPKM1")
## DelayedMatrix object of 2 x 6 doubles:
##               HG000096 HG000097 HG000099 HG00100 HG00101
## ENSG00000171603.11 21.474466 23.357203 13.218601 21.748905 22.694890
## ENSG00000230216.1  3.136087  5.162155  3.223215  5.606075  5.466393
##               HG00102
## ENSG00000171603.11 27.397227
## ENSG00000230216.1  5.279783
```

A *MultiExperimentDb* can be subset on all common rows across experiments:

```
intersectRownames(medb, rownames=NULL)
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944719d928.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29443d1f77d3.sqlite
## dim: 2 8
## experiments:
##   geuFPKM1 (2 x 6)
##     rownames: ENSG00000171603.11, ENSG00000230216.1
##     colnames: HG000096, HG000097, HG000099, HG00100, HG00101, HG00102
##   geuFPKM2 (2 x 6)
##     rownames: ENSG00000171603.11, ENSG00000230216.1
##     colnames: HG000099, HG00100, HG00101, HG00102, HG00103, HG00104
```

or all common columns across experiments:

```
intersectColnames(medb, colnames=NULL)
## class: MultiExperimentDb
```



## Summary of work on Organism.dplyr and MultiExperimentDb

```
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944719d928.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29443d1f77d3.sqlite
## dim: 3 4
## experiments:
##   geuFPKM1 (2 x 4)
##     rownames: ENSG00000171603.11, ENSG00000230216.1
##     colnames: HG000099, HG00100, HG00101, HG00102
##   geuFPKM2 (3 x 4)
##     rownames: ENSG00000171603.11, ENSG00000230216.1,
##               ENSG00000172992.6
##     colnames: HG000099, HG00100, HG00101, HG00102
```

### 4.3.2.4 Combine by columns or rows

To look at assay data of all experiments together, combine all rows of assays in a *MultiExperimentDb* object with matching columns, numbers of columns of each experiments in the *MultiExperimentDb* object don't need to be the same. When the argument `all.columns` is TRUE, the output is a matrix with columns across all assays where missing values are represented with NA. Default is FALSE, only columns that exist in all assays display.

```
rbindme(medb)
## DelayedMatrix object of 5 x 4 doubles:
##           HG000099  HG00100  HG00101  HG00102
## ENSG00000171603.11 13.218601 21.748905 22.694890 27.397227
## ENSG00000230216.1  3.223215  5.606075  5.466393  5.279783
## ENSG00000171603.11 13.218601 21.748905 22.694890 27.397227
## ENSG00000230216.1  3.223215  5.606075  5.466393  5.279783
## ENSG00000172992.6 10.875091 13.999134 16.376246 14.116252
rbindme(medb, all.columns=TRUE)
## DelayedMatrix object of 5 x 8 doubles:
##           HG000096  HG000097  HG000099      .  HG00103  HG00104
## ENSG00000171603.11 21.474466 23.357203 13.218601      .      NA      NA
## ENSG00000230216.1  3.136087  5.162155  3.223215      .      NA      NA
## ENSG00000171603.11      NA      NA 13.218601      . 26.124471 22.235168
## ENSG00000230216.1      NA      NA  3.223215      .  7.324214  6.869039
## ENSG00000172992.6      NA      NA 10.875091      . 17.132905 16.163483
```

Combine all columns of assays in a *MultiExperimentDb* object with matching rows, numbers of rows of each experiments in the *MultiExperimentDb* object don't need to be the same. When the argument `all.rows` is TRUE, the output is a matrix with rows across all assays where missing values are represented with NA. Default is FALSE, only rows that exist in all assays display.

```
cbindme(medb)
## DelayedMatrix object of 2 x 12 doubles:
##           HG000096  HG000097  HG000099      .  HG00103  HG00104
## ENSG00000171603.11 21.474466 23.357203 13.218601      . 26.124471 22.235168
## ENSG00000230216.1  3.136087  5.162155  3.223215      .  7.324214  6.869039
cbindme(medb, all.rows=TRUE)
## DelayedMatrix object of 3 x 12 doubles:
##           HG000096  HG000097  HG000099      .  HG00103  HG00104
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
## ENSG00000171603.11 21.474466 23.357203 13.218601 . 26.124471 22.235168
## ENSG00000230216.1 3.136087 5.162155 3.223215 . 7.324214 6.869039
## ENSG00000172992.6 NA NA NA . 17.132905 16.163483
```

### 4.3.3 Comparing treated vs untreated 'airway' data

This package is designed for comparing data between different experiments with similarity (overlapping features or samples across experiments). The construction of displaying assay data with *DelayedMatrix* object reduces memory usage and optimizes performance. Below is an example of analyzing data from multiple experiments using *MultiExperimentDb*.

In the example below, a *RangedSummarizedExperiment* object of read counts in genes for an RNA-Seq experiment on human airway smooth muscle cell lines is used. Two experiments are generated from airway: *airway\_untrt* with four untreated cell lines and *airway\_trt* with four treated cell lines.

Create an empty *MultiExperimentDb* object.

```
medb0 <- MultiExperimentDb()
```

Add data for treated and untreated experiments to the 'medb0' object.

```
library(airway)
data(airway)
medb0 <- addExperiment(medb0,
  airway[,colData(airway)$dex == "untrt"], "airway_untrt")
medb0 <- addExperiment(medb0,
  airway[,colData(airway)$dex == "trt"], "airway_trt")

experimentNames(medb0)
## [1] "airway_untrt" "airway_trt"
length(medb0)
## [1] 2
dim(medb0)
## [1] 64102      8
dimnames(medb0)
## CharacterList of length 2
## [[1]] ENSG00000000003 ENSG00000000005 ENSG00000000419 ... LRG_98 LRG_99
## [[2]] SRR1039508 SRR1039512 SRR1039516 ... SRR1039513 SRR1039517 SRR1039521
```

Comparing object sizes of the airway data in the original *RangedSummarizedExperiment* format to the *MultiExperimentDb* format we see the *MultiExperimentDb* object is much smaller.

```
format(object.size(airway), units = "Mb")
## [1] "58.9 Mb"

format(object.size(medb0), units = "Mb")
## [1] "8.3 Mb"
```

Subset the data by the BRCA1 and BRCA2 gene symbols.

## Summary of work on Organism.dplyr and MultiExperimentDb

```
granges <- grangesFromIdentifiers(org = "org.Hs.eg.db",
  keys = c("BRCA1", "BRCA2"), keytype = "SYMBOL",
  txdb = "TxDb.Hsapiens.UCSC.hg19.knownGene")
## 'select()' returned 1:1 mapping between keys and columns
medb0 <- medb0[granges,,]
medb0
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29447ba230f7.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944284c1d89.sqlite
## dim: 6 8
## experiments:
##   airway_untrt (6 x 4)
##     rownames: ENSG00000012048, ENSG000000139618, ENSG000000215515,
##       ENSG000000267002, ENSG000000267340, ENSG000000267595
##     colnames: SRR1039508, SRR1039512, SRR1039516, SRR1039520
##   airway_trt (6 x 4)
##     rownames: ENSG00000012048, ENSG000000139618, ENSG000000215515,
##       ENSG000000267002, ENSG000000267340, ENSG000000267595
##     colnames: SRR1039509, SRR1039513, SRR1039517, SRR1039521
```

Look at data of experiment "airway\_trt" after subset.

```
airway_trt <- experiment(medb0, "airway_trt")
colData(airway_trt)
## DataFrame with 4 rows and 9 columns
##       SampleName      cell      dex      albut      Run avgLength
##       <factor> <factor> <factor> <factor> <factor> <integer>
## SRR1039509 GSM1275863 N61311      trt      untrt SRR1039509      126
## SRR1039513 GSM1275867 N052611      trt      untrt SRR1039513      87
## SRR1039517 GSM1275871 N080611      trt      untrt SRR1039517     126
## SRR1039521 GSM1275875 N061011      trt      untrt SRR1039521      98
##       Experiment      Sample      BioSample
##       <factor> <factor> <factor>
## SRR1039509 SRX384346 SRS508567 SAMN02422675
## SRR1039513 SRX384350 SRS508572 SAMN02422670
## SRR1039517 SRX384354 SRS508576 SAMN02422673
## SRR1039521 SRX384358 SRS508580 SAMN02422677
rowRanges(airway_trt)
## GRangesList object of length 6:
## $ENSG00000012048
## GRanges object with 99 ranges and 0 metadata columns:
##       seqnames      ranges strand
##       <Rle> <IRanges> <Rle>
## [1] 17 [41196312, 41197819] -
## [2] 17 [41196313, 41197819] -
## [3] 17 [41196822, 41197819] -
## [4] 17 [41197580, 41197819] -
## [5] 17 [41197646, 41197819] -
## ... ... ...
## [95] 17 [41277294, 41277376] -
## [96] 17 [41277294, 41277387] -
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
## [97]      17 [41277294, 41277419] -
## [98]      17 [41277294, 41277467] -
## [99]      17 [41277294, 41277468] -
##
## ...
## <5 more elements>
## -----
## seqinfo: 2 sequences from an unspecified genome; no seqlengths
assay(medb0, "airway_trt")
## DelayedMatrix object of 6 x 4 integers:
##
##      SRR1039509 SRR1039513 SRR1039517 SRR1039521
## ENSG00000012048      98      95      168      155
## ENSG000000139618     54      30      55      40
## ENSG000000215515      0       0       0       0
## ENSG000000267002     35      24      40      45
## ENSG000000267340      1       3       9       1
## ENSG000000267595      3       5       6       6
```

Combine all columns of assays in *medb* with matching rows to look at assay data of all experiments together. First four columns come from the treated data and the last four from untreated.

```
cbindme(medb0)
## DelayedMatrix object of 6 x 8 integers:
##
##      SRR1039508 SRR1039512 SRR1039516 SRR1039520 SRR1039509
## ENSG00000012048     322     418     326     265     98
## ENSG000000139618     67      79      66      96     54
## ENSG000000215515      0       0       0       0       0
## ENSG000000267002     77      84      67      60     35
## ENSG000000267340      5       7      10       8       1
## ENSG000000267595      3       2       4       5       3
##
##      SRR1039513 SRR1039517 SRR1039521
## ENSG00000012048     95     168     155
## ENSG000000139618     30      55      40
## ENSG000000215515      0       0       0
## ENSG000000267002     24      40      45
## ENSG000000267340      3       9       1
## ENSG000000267595      5       6       6
```

Restore data from disk and do another subset.

```
medb1 <- loadMultiExperimentDb(hdf5path = hdf5path(medb0),
                              sqlitepath = sqlitepath(medb0))

medb1
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29447ba230f7.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944284c1d89.sqlite
## dim: 64102 8
## experiments:
##   airway_untrt (64102 x 4)
##   rownames: ENSG000000000003, ENSG000000000005, ENSG000000000419,
##   ..., LRG_98, LRG_99
```

## Summary of work on Organism.dplyr and MultiExperimentDb

```
##      colnames: SRR1039508, SRR1039512, SRR1039516, SRR1039520
##      airway_trt (64102 x 4)
##      rownames: ENSG000000000003, ENSG000000000005, ENSG000000000419,
##      ..., LRG_98, LRG_99
##      colnames: SRR1039509, SRR1039513, SRR1039517, SRR1039521
```

Search across treated and untreated for given rownames.

```
medb1[c("ENSG00000213613", "ENSG00000267595"),,]
## class: MultiExperimentDb
## hdf5path: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file29447ba230f7.h5
## sqlitepath: C:\Users\YU19864\AppData\Local\Temp\RtmpicIbyY\file2944284c1d89.sqlite
## dim: 2 8
## experiments:
##      airway_untrt (2 x 4)
##      rownames: ENSG00000213613, ENSG00000267595
##      colnames: SRR1039508, SRR1039512, SRR1039516, SRR1039520
##      airway_trt (2 x 4)
##      rownames: ENSG00000213613, ENSG00000267595
##      colnames: SRR1039509, SRR1039513, SRR1039517, SRR1039521
```

## 4.4 Discussion

MultiExperimentDb package is created for storing and comparing different bioinformatic experiments in *SummarizedExperiment* format within one object, so the experiments added to a *MultiExperimentDb* object should have some similarity such as common features or samples. It can be used for analyzing data from multiple similar experiments. Also it works better with experiments which contains large data matrices (for example, microarray gene expression data, read counts in genes for RNA-Seq experiments, etc.) and small annotation data (rowData, colData and rowRanges).

The size of HDF5 file is relatively small, but sqlite file can be big if annotation data is big. However, even when sqlite file is big, it is stored on disk, and when a *MultiExperimentDb* object is created to point to the sqlite file and HDF5 file, the object size is smaller than the original *SummarizedExperiment* objects. Besides, by using *DelayedMatrix* to display assay data in R, subset and binding functions return results quickly.

## 5 Summary and Conclusions

---

## 6 References

---

Pages H, Carlson M, Falcon S and Li N (2016). AnnotationDbi: Annotation Database Interface. R package version 1.37.0.

## Summary of work on Organism.dplyr and MultiExperimentDb

Huber, W., Carey, J. V, Gentleman, R., Anders, S., Carlson, M., Carvalho, S. B, Bravo, C. H, Davis, S., Gatto, L., Girke, T., Gottardo, R., Hahne, F., Hansen, D. K, Irizarry, A. R, Lawrence, M., Love, I. M, MacDonald, J., Obenchain, V., Ole's, K. A, Pag'es, H., Reyes, A., Shannon, P., Smyth, K. G, Tenenbaum, D., Waldron, L., Morgan and M. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), pp. 115-121.

Oles A, Morgan M and Huber W (2016). *BiocStyle: Standard styles for vignettes and other Bioconductor documents*. R package version 2.3.28.

Hadley Wickham, Kirill Müller (2016). *DBI: R Database Interface*. R package version 0.5-1.

Pagès H (2016). *DelayedArray: Delayed operations on array-like objects*. R package version 0.1.3.

Hadley Wickham, Winston Chang, RStudio, R Core team (2016). *devtools: Tools to Make Developing R Packages Easier*. R package version 1.12.0.

Arora S, Morgan M, Carlson M and Pagès H (2016). *GenomeInfoDb: Utilities for manipulating chromosome and other 'seqname' identifiers*. R package version 1.11.6.

Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, Morgan M and Carey V (2013). "Software for Computing and Annotating Genomic Ranges." *PLoS Computational Biology*, 9.

Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, Morgan M and Carey V (2013). "Software for Computing and Annotating Genomic Ranges." *PLoS Computational Biology*, 9.

Pagès H (2016). *HDF5Array: HDF5 back end for DelayedArray objects*. R package version 1.3.4.

Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, Morgan M and Carey V (2013). "Software for Computing and Annotating Genomic Ranges." *PLoS Computational Biology*, 9.

Fischer B and Pau G (2016). *rhdf5: HDF5 interface to R*. R package version 2.19.0.

Kirill Müller, Hadley Wickham, David A. James, Seth Falcon, Liam Healy. *RSQLite: 'SQLite' Interface for R*. R package version 1.1-2.

Pagès H, Lawrence M and Aboyoun P (2016). *S4Vectors: S4 implementation of vectors and lists*. R package version 0.13.5.

Morgan M, Obenchain V, Hester J and Pagès H (2016). *SummarizedExperiment: SummarizedExperiment container*. R package version 1.5.3.

Wickham, Hadley. "testthat: Get started with testing." *The R Journal* 3.1 (2011): 5-10.

## 7 Appendix

---

1. Organism.dplyr repository
2. MultiExperimentDb repository