Bioconductor / AnnotationDbi

Code Issues 2 Pull requests Actions Projects Wiki Security Insights

New issue

Jump to bottom

multiVals for select? #2

Open LTLA opened this issue on 13 Dec 2019 · 11 comments

Assignees

LTLA commented on 13 Dec 2019

In my analysis code, I have not-uncommon occurrences of:

```
library(AnnotationHub)
ens.mm.v97 <- AnnotationHub()[["AH73905"]]
anno <- select(ens.mm.v97, keys=rownames(se),
    keytype="GENEID", columns=c("SYMBOL", "SEQNAME"))
rowData(se) <- anno[match(rownames(se), anno$GENEID),]
```

It would be nice to do something like:

```
anno <- select(ens.mm.v97, keys=rownames(se), multiVals="first",
    keytype="GENEID", columns=c("SYMBOL", "SEQNAME"))
```

... and save myself an extra line of code (and improve robustness to changes to the annotation object). Sort of like how I get an integer vector if I ask for `findOverlaps(..., select="first")`.

jmacdon commented on 28 Jan

The object in your case is an EnsDb, for which there already is a multiVals argument, and for which 'first' is the default.

```
ens.mm.v97 <- hub[["AH73905"]]
loading from cache
z <- select(ens.mm.v97, keys(ens.mm.v97), c("SYMBOL","SEQNAME"), "GENEID")
any(duplicated(z[,1]))
[1] FALSE
```

Johannes Rainier has a significantly different codebase in ensembldb than what is in AnnotationDbi, and doesn't seem to do the LEFT JOINs in AnnotationDbi that will end up blowing out the rows of the returned data.frame. For packages that dispatch on select from AnnotationDbi, I find it safer to use mapIds repeatedly instead

anno <- as.data.frame(c("SYMBOL", "GENENAME","WHATEVER"), function(x) mapIds(OrgDb, KEYS, x, COLUMN))

Which is one line, respects the order of the incoming KEYS, and doesn't suffer from row blowout.

LTLA commented on 28 Jan

> The object in your case is an EnsDb, for which there already is a multiVals argument, and for which 'first' is the default.

I assume I'm missing something here because there isn't a `multiVals=` in the `select()` documentation or `EnsDb` method. It doesn't get used if you pass it from `...`, either, based on reading the code.

> For packages that dispatch on select from AnnotationDbi, I find it safer to use mapIds repeatedly instead

Yeah, well, fine, but you and I are super-pros. Also I guess you're missing a `lapply` there.

I didn't mention this in my original post, but the code snippets above are used throughout the https://osca.bioconductor.org/ book, and it's a lot easier to teach people if there's a function that just does the job instead of expecting them to put together a one-liner like the above.

`select()` is pretty close to what I'd like, but even a wrapper around your code snippet would be fine. Just something that we can document and point people to for what I imagine to be a fairly common use case - well, for me at least.

mtmorgan commented on 28 Jan

It would help to have reproducible example (`se` is not defined). FWIW I did

```
> set.seed(123); egid = sample(keys(ens.mm.v97))
> anno = select(ens.mm.v97, egid, c("SYMBOL", "SEQNAME"), "GENEID")
> identical(egid, anno$GENEID)
[1] TRUE
```

and don't (yet) understand the problem?

Is this an issue with ensembldb, which from `packageDesecription("ensembldb")` has https://github.com/jorainer/ensembldb ?

jmacdon commented on 28 Jan

A better example of what I believe Aaron is getting at (although not a good use-case for what he wants, admittedly) is

```
> z <- select(ens.mm.v97, keys(ens.mm.v97), c("SYMBOL","TXNAME"), "GENEID")
> any(duplicated(z[,1]))
[1] TRUE
> head(z)
              GENEID SYMBOL             TXNAME
1 ENSMUSG00000000001  Gnai3 ENSMUST00000000001
2 ENSMUSG00000000003   Pbsn ENSMUST00000000003
3 ENSMUSG00000000003   Pbsn ENSMUST00000114041
4 ENSMUSG00000000028  Cdc45 ENSMUST00000000028
5 ENSMUSG00000000028  Cdc45 ENSMUST00000096990
6 ENSMUSG00000000028  Cdc45 ENSMUST00000115585
```

An `EnsDb` is hard to use for an example, as the 1:many mappings are usually things you wouldn't want to undo. Although one might argue that returning a `DataFrame` with a `CharacterList` in the third column might be a reasonable thing to do. A better example would be something like

```
> z <- select(org.Hs.eg.db, keys(org.Hs.eg.db), c("ENTREZID","ALIAS","ACCNUM"))
'select()' returned 1:many mapping between keys and columns
>dim(z)
> [1] 3182457       3
> length(keys(org.Hs.eg.db))
[1] 61217
```

Where multiple 1:many mappings absolutely blow out the number of rows due to two LEFT JOINs between tables. And if you just want one row per gene (and naively want to use just the first one, because what other choice is materially better), you either have to remove duplicates in the first column as Aaron does in his example, or use `mapIds` to keep to one result regardless.

Not sure, but it might be relatively simple to do pretty much what Aaron suggests, adding in a `multiVals` argument that will then cause `select` to simply loop over the input columns using `mapIds` instead of doing what it already does, in which case `select` could return either a `data.frame` with one row per input ID, or a `DataFrame` with a `CharacterList` for any 1:many mappings.

LTLA commented on 29 Jan

Thanks **@jmacdon**, the `org.Hs.eg.db` example is exactly what I want to protect against. The wider context is that I am trying to populate the `rowData()` of a SE so I need a 1:1 output from `select()`.

jmacdon commented on 29 Jan

This is something I have had in my affycoretools package for a while, intended to do what you want in the context of an `ExpressionSet`

```
setMethod("annotateEset", c("ExpressionSet","ChipDb"),
          function(object, x, columns = c("PROBEID","ENTREZID","SYMBOL","GENENAME"), multivals
= "first"){
    ## Doing mapIds(chipDb, featureNames(object), "PROBEID","PROBEID") fails for many
packages, and wastes compute cycles
    ## so we just add that by hand.
    addcol <- FALSE
    if(any(columns == "PROBEID")) {
        addcol <- TRUE
        columns <- columns[-grep("PROBEID", columns)]
        collst <- list(PROBEID = featureNames(object))
    }
    multivals <- switch(multivals,
                        first = "first",
                        list = "CharacterList",
                        CharacterList = "CharacterList",
                        stop("The multivals argument should be 'first', 'list' or
'CharacterList'", call. = FALSE))
    annolst <- lapply(columns, function(y) mapIds(x, featureNames(object), y, "PROBEID",
multiVals = multivals))
    if(addcol) annolst <- c(collst, annolst)
    anno <- switch(multivals,
                   first = as.data.frame(annolst),
                   DataFrame(annolst))
    names(anno) <- if(addcol) c("PROBEID", columns) else columns
    if(!isTRUE(all.equal(row.names(anno), featureNames(object))))
        stop(paste("There appears to be a mismatch between the ExpressionSet and",
```

```
                    "the annotation data.\nPlease ensure that the summarization level",
                    "for the ExpressionSet and the annotation package are the same.\n"),
              call. = FALSE)
    andf <- dfToAnnoDF(anno)
    featureData(object) <- andf
    stopifnot(validObject(object))
    object
}))
```

Since this function is intended to put an `AnnotatedDataFrame` into an `ExpressionSet`, there is extra cruft, but the general idea is there.

jmacdon commented on 29 Jan

@LTLA Given that there are multiple versions of `select` that dispatch on lots of different objects, and what you want is a method to easily add annotation data to either a `SummarizedExperiment` or maybe a `SingleCellExperiment`, does it make more sense to add a function similar to `annotateEset` in (probably) `SummarizedExperiment` that can return a 1:1 mapping, agnostic to the underlying object from which the data are coming?

It seems to be a specific use case for a specific object, and is thus probably better housed with the code that generates the object. Obvious downside being that `SummarizedExperiment` would then need to suggest `ensembldb` and import all the various iterations of `mapIds`.

LTLA commented on 29 Jan

scater already hosts a function for doing this annotation from `biomaRt`; it would not be much of a stretch to do it for the various `*Db` objects onto SCEs. Also happy to see a more general SE approach somewhere but that should not live in **SummarizedExperiment** IMO.

Kayla-Morrell self-assigned this on 14 Mar

Kayla-Morrell commented on 23 Jun

@LTLA and **@jmacdon** - Martin and I have been working on this issue and think we finally have something that will do what you are looking for. If you want to take a look at the `openIssue` branch of `AnnotationDbi` this is where we have the code implemented. I did a basic test with some of the above code:

```
> z <- select(org.Hs.eg.db, keys(org.Hs.eg.db), c("ENTREZID", "ALIAS", "ACCNUM"))
'select()' returned 1:many mapping between keys and columns
> head(z)
  ENTREZID ALIAS   ACCNUM
1        1   A1B AA484435
2        1   A1B AAH35719
3        1   A1B AAL07469
4        1   A1B AB073611
5        1   A1B ACJ13639
6        1   A1B AF414429
> dim(z)
[1] 3182457       3
> length(keys(org.Hs.eg.db))
[1] 61217
> z2 <- select(org.Hs.eg.db, keys(org.Hs.eg.db), c("ENTREZID", "ALIAS", "ACCNUM"), multiVals =
"first")
> head(z2)
DataFrame with 6 rows and 3 columns
     ENTREZID       ALIAS      ACCNUM
  <character> <character> <character>
1           1         A1B    AA484435
2           2        A2MD    AAA51551
3           3        A2MP    DB301195
4           9        AAC1    AAB62398
5          10        AAC2    AAA59905
6          11        AACP          NA
> dim(z2)
[1] 61217     3
> z3 <- select(org.Hs.eg.db, keys(org.Hs.eg.db), c("ENTREZID", "ALIAS", "ACCNUM"), multiVals =
"list")
> head(z3)
DataFrame with 6 rows and 3 columns
     ENTREZID                 ALIAS                        ACCNUM
  <character>                <list>                        <list>
1           1        A1B,ABG,GAB,... AA484435,AAH35719,AAL07469,...
2           2 A2MD,CPAMD5,FWP007,... AAA51551,AAA51552,AAH26246,...
3           3             A2MP,A2MP1 DB301195,DV080209,DV080210,...
4           9    AAC1,MNAT,NAT-1,... AAB62398,AAB84384,AAB86878,...
5          10    AAC2,NAT-2,PNAT,... AAA59905,AAA64584,AAA64585,...
6          11        AACP,NATP1,NATP                            NA
> dim(z3)
[1] 61217     3
```

Feel free to test it out and let us know your thoughts. Thanks!

☺

LTLA commented on 23 Jun

From a quick glance, this is perfect. I would further suggest that:

- The `keys` are used in the output row names, so I can just `rowData(se) <- z2` without loss of SE row names. (Can't exactly remember what happens for the `rowData`, but I do remember that `colData<-` wipes the column names.)

- Haven't checked but I would like to confirm that I get all-`NA` rows if a key is not present. On a related note, errors like:

```
> select(org.Hs.eg.db, keys="a", keytype="ENTREZID", columns="SYMBOL")
Error in .testForValidKeys(x, keys, keytype, fks) :
  None of the keys entered are valid keys for 'ENTREZID'. Please use the keys method to
see a listing of valid arguments.
```

  would be better served as warnings in `multiVals="first"` as I would expect to get an all-NA `DataFrame` out.

jmacdon commented on 23 Jun

Looks good, although should it always return a `DataFrame`? I can see an argument for the return object being a `data.frame` if multiVals is missing or is 'first', a regular list if multiVals is 'list', and a `DataFrame` if multiVals is 'CharacterList'.

Assignees

Kayla-Morrell

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants