A wrapper to query DGIdb using R

Thomas Thurnherr*, Franziska Singer, Daniel J. Stekhoven, Niko Beerenwinkel

October 17, 2016

Abstract

Annotation and interpretation of DNA aberrations identified through next-generation sequencing is becoming an increasingly important task, especially in the context of data analysis pipelines for medical applications, where aberrations are associated with phenotypic and clinical features. A possible approach for annotation is to identify drugs as potential targets for aberrated genes or pathways. DGIdb accumulates data from 15 different gene-target interaction resources and allows querying these through their web interface as well as public API. rDGIdb is a wrapper to query DGIdb using R/Bioconductor. The package provides its output in a similar format as the web interface, and thereby allows integration of DGIdb queries into bioinformatic pipelines.

Contents

1 Standard workflow

To query DGIdb [1], we first load the package.

```
library(rDGIdb)
```

Next, we prepare a list of genes for which we want to query drug targets. If you already have a list of genes with variants, these can either be loaded from a file or typed manually. Genes have to be provided as a character vector.

Here, we purposely use a non-existant gene name `XYZA` for illustration.

```
genes <- c("TNF", "AP1", "AP2", "XYZA")
```

With a vector of genes, we can query DGIdb using the `queryDGIdb()` function. The argument `genes` is a required argument, all other arguments are optional. These optional arguments are used as filters. If they are not provided, the query returns all results for a specific gene. See further below for more details on optional arguments.

*thomas.thurnherr@bsse.ethz.ch

```
result <- queryDGIdb(genes)
```

1.1 Accessing query results

After querying, we access the `rDGIdbResult` object that was returned by `queryDGIdb`. The S4 class `rDGIdbResult` contains several tables (`data.frame`), which roughly reflect each result tab on the DGIdb web interface at http://dgidb.genome.wustl.edu.

The results are available in the following four formats:

Result summary Drug-gene interactions summarized by the source(s) that reported them.

Detailed results Search terms matching exactly one gene that has one or more drug interactions.

By gene Drug interaction count and druggable categories associated with each gene.

Search term summary Summary of the attempt to map gene names supplied by the user to gene records in DGIdb.

The results can be accessed through helper functions.

```
## Result summary
resultSummary(result)

## Detailed results
detailedResults(result)

## By gene
byGene(result)

## Search term summary
searchTermSummary(result)
```

1.2 Using optional query arguments

There are three optional arguments to `queryDGIdb`.

```
queryDGIdb(genes = genes,
    sourceDatabases = NULL,
    geneCategories = NULL,
    interactionTypes = NULL)
```

The package provides helper functions to list possible values for these optional arguments.

```
## Available source databases
sourceDatabases()
```

```
##  [1] "CIViC"                           "CancerCommons"
##  [3] "ChEMBL"                          "ClearityFoundationBiomarkers"
##  [5] "ClearityFoundationClinicalTrial" "DoCM"
##  [7] "DrugBank"                        "GuideToPharmacologyInteractions"
##  [9] "MyCancerGenome"                  "MyCancerGenomeClinicalTrial"
## [11] "PharmGKB"                        "TALC"
## [13] "TEND"                            "TTD"
```

```
## [15] "TdgClinicalTrial"
```

```
## Available gene categories
geneCategories()
```

```
##  [1] "abc transporter"                          "b30_2 spry domain"
##  [3] "cell surface"                             "clinically actionable"
##  [5] "cytochrome p450"                          "dna directed rna polymerase"
##  [7] "dna repair"                               "drug metabolism"
##  [9] "drug resistance"                          "druggable genome"
## [11] "exchanger"                                "external side of plasma membrane"
## [13] "fibrinogen"                               "g protein coupled receptor"
## [15] "growth factor"                            "histone modification"
## [17] "hormone activity"                         "ion channel"
## [19] "kinase"                                   "lipase"
## [21] "lipid kinase"                             "methyl transferase"
## [23] "myotubularin related protein phosphatase" "neutral zinc metallopeptidase"
## [25] "nuclear hormone receptor"                 "phosphatidylinositol 3 kinase"
## [27] "phospholipase"                            "protease"
## [29] "protease inhibitor"                       "protein phosphatase"
## [31] "pten family"                              "rna directed dna polymerase"
## [33] "serine threonine kinase"                  "short chain dehydrogenase reductase"
## [35] "thioredoxin"                              "transcription factor binding"
## [37] "transcription factor complex"             "transporter"
## [39] "tumor suppressor"                         "tyrosine kinase"
## [41] "unknown"
```

```
## Available interaction types
interactionTypes()
```

```
##  [1] "activator"                    "adduct"
##  [3] "agonist"                      "allosteric modulator"
##  [5] "antagonist"                   "antibody"
##  [7] "antisense"                    "antisense oligonucleotide"
##  [9] "binder"                       "blocker"
## [11] "chaperone"                    "cleavage"
## [13] "cofactor"                     "competitive"
## [15] "immunotherapy"                "inducer"
## [17] "inhibitor"                    "inhibitory allosteric modulator"
## [19] "inverse agonist"              "ligand"
## [21] "modulator"                    "multitarget"
## [23] "n/a"                          "negative modulator"
## [25] "other/unknown"                "partial agonist"
## [27] "partial antagonist"           "positive allosteric modulator"
## [29] "potentiator"                  "product of"
## [31] "stimulator"                   "suppressor"
## [33] "vaccine"
```

Perhaps we are only intersted in a subset of available source databases, gene categories, or interaction types. Using the list above, we can make the query more specific by adding filters.

For example, with a given set of genes, we only want interactions from `DrugBank` and `MyCancerGenome`. In addition, genes have to have the attribute: `clinically actionable`; and interactions have to show at least one of these labels: `suppressor`, `activator`, or `blocker`.

We can query DGIdb using the function call below.

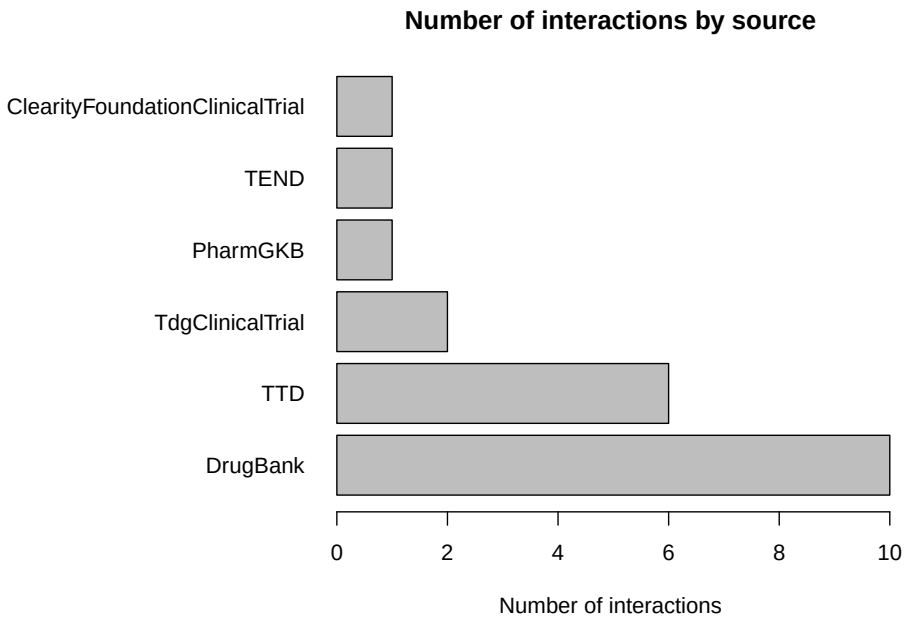
```
resultFilter <- queryDGIdb(genes,
                sourceDatabases = c("DrugBank","MyCancerGenome"),
                geneCategories = "clinically actionable",
                interactionTypes = c("suppressor","activator","blocker"))
```

In case no gene-drug interaction satisfies these conditions, the result is returned empty.

1.3 Basic visualization of results

The package also provides basic visualization functionality for query results. `plotInteractionsBySource` generates a bar plot that shows how many interactions were reported for each source. As input the function requires the query result object of class `rDGIdbResult`. Additional arguments are passed to the `barplot`.

```
plotInteractionsBySource(result, main = "Number of interactions by source")
```



1.4 Version numbers of DGIdb resources

DGIdb may no use the latest version of each resources it integrates. The current version numbers of all resources can be printed using `resourceVersions`.

1.5 Input in VCF file format

From a variant call format (VCF) file, variants can be annotated within R using the variant annotation workflow provided by the `VariantAnnotation` package from Bioconductor [2]. For more information on how to filter variants, please see the package documentation/vignette.

```
library("VariantAnnotation")
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
library("org.Hs.eg.db")
vcf <- readVcf("file.vcf.gz", "hg19")
```

```
seqlevels(vcf) <- paste("chr", seqlevels(vcf), sep = "")
rd <- rowRanges(vcf)
loc <- locateVariants(rd, TxDb.Hsapiens.UCSC.hg19.knownGene, CodingVariants())
symbols <- select(x = org.Hs.eg.db, keys = mcols(loc)$GENEID,
                  columns = "SYMBOL", keytype = "ENTREZID")
genes <- unique(symbols$SYMBOL)
```

2 How to get help

Please consult the package documentation first.

```
?queryDGIdb
?rDGIdbFilters
?rDGIdbResult
?plotInteractionsBySource
```

If this does not solve your problem, we are happy to help you. Questions regarding the rDGIdb package should be posted to the Bioconductor support site: https://support.bioconductor.org, which serves as a repository of questions and answers. This way, other people can benefit from questions and corresponding answers, which minimizes efforts by the developers.

3 Session info

- R version 3.3.1 (2016-06-21), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: rDGIdb 1.0.0
- Loaded via a namespace (and not attached): BiocStyle 2.2.0, R6 2.2.0, curl 2.1, evaluate 0.10, formatR 1.4, highr 0.6, httr 1.2.1, jsonlite 1.1, knitr 1.14, magrittr 1.5, stringi 1.1.2, stringr 1.1.0, tools 3.3.1

4 Citing this package

If you use this package for published research, please cite the package as well as DGIdb.

```
citation('rDGIdb')
```

References

[1] Alex H. Wagner, Adam C. Coffman, Benjamin J. Ainscough, Nicholas C. Spies, Zachary L. Skidmore, Katie M. Campbell, Kilannin Krysiak, Deng Pan, Joshua F. McMichael, James M. Eldred, Jason R. Walker, Richard K. Wilson, Elaine R. Mardis, Malachi Griffith, and Obi L. Griffith. DGIdb 2.0: mining clinically relevant drug–gene interactions. *Nucleic Acids Res*, 44(D1):D1036–D1044, nov 2015. URL: http://dx.doi.org/10.1093/nar/gkv1165, doi:10.1093/nar/gkv1165.

[2] Valerie Obenchain, Michael Lawrence, Vincent Carey, Stephanie Gogarten, Paul Shannon, and Martin Morgan. Variantannotation: a bioconductor package for exploration and annotation of genetic variants. *Bioinformatics*, 30(14):2076–2078, 2014. doi:10.1093/bioinformatics/btu168.