Vignette for the DOAGDC package

Fábio Malta de Sá Patroni

2019-05-02

Abstract

A large collection of clinical information and multi-omics data of different cancer types has been created through the project The Cancer Genome Atlas (TCGA). The efficient use of this information has the potential to unveil new observations and mechanisms that can impact on cancer treatment. Although the amount of data rises every year, data mining tools are not following at the same pace. We developed a platform for the analysis of TCGA data using the R programming language. The package called GDCtools allows analyses based on the separation of tumor groups according to continuous characteristics (e. g. genetic expression) or discrete (clinical characteristics), followed by differential gene expression analysis and pathway enrichment analysis. In addition, it is possible to use co-expression analysis. We searched for differentially expressed isoforms of HIF3A gene between normal and tumor related tissues, followed by the evaluation of patient prognosis. Our studies pointed for the first time the association of HIF-3a2 in colon adenocarcinoma and HIF-3a3 in prostatic adenocarcinoma with relapse-free survival and tumor progression. The approach utilized here can be applied to other TCGA analysis and help the user to achieve new insights in targets of clinical interest.

Package Description

The **DOAGDC** package was developed to download, organize and analyze Genomic Data Commons (GDC) data. It uses several well-known R packages, as *DESeq2*[1], *edgeR*[2], *mclust*[3], and *WGCNA*[4]. This is a free open-source software that implements academic research by the authors and co-workers. If you use it, please support the project by citing the appropriate journal article listed running the following command `citation("DOAGDC")`:

```
#>
#> To cite package 'DOAGDC' in publications use:
#>
#>   Fábio Malta de Sá Patroni (2019). DOAGDC: A Package to Download,
#>   Organize and Analyze Genomic Data Commons (GDC) Data. R package
#>   version 0.99.0. https://github.com/Facottons/DOAGDC
#>
#> A BibTeX entry for LaTeX users is
#>
#>   @Manual{,
#>     title = {DOAGDC: A Package to Download, Organize and Analyze Genomic Data Commons (GDC) Data},
#>     author = {Fábio Malta de Sá Patroni},
#>     year = {2019},
```

[1]Anders, Simon, and Wolfgang Huber. 2010. "Differential Expression Analysis for Sequence Count Data." Genome Biology 11 (10): R106. https://doi.org/10.1186/gb-2010-11-10-r106

[2]Robinson, M. D., D. J. McCarthy, and G. K. Smyth. 2010. "EdgeR: A Bioconductor Package for Differential Expression Analysis of Digital Gene Expression Data." Bioinformatics 26 (1): 139–40. https://doi.org/10.1093/bioinformatics/btp616

[3]Scrucca, Luca, Michael Fop, T Brendan Murphy, and Adrian E Raftery. 2016. "Mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models." https://journal.r-project.org/archive/2016/RJ-2016-021/RJ-2016-021.pdf

[4]Langfelder, Peter, and Steve Horvath. 2008. "WGCNA: An R Package for Weighted Correlation Network Analysis." BMC Bioinformatics 9 (1): 559. https://doi.org/10.1186/1471-2105-9-559

```
#>     note = {R package version 0.99.0},
#>     url = {https://github.com/Facottons/DOAGDC},
#>   }
```

Starting workflow - Functions explanation

The following image summarize the DOAGDC package workflow:

Downloading GDC Data Portal or GDC Legacy Archive data

GDC Data Portal and GDC Legacy Archive are composed by different types of data (e.g. gene and isoform expression data). It is possible to use the `dataType` argument to retrieve specific data. For instance, one could use `dataType = "methylation"` to download Methylation data or `dataType = "protein"` to download Protein Expression data. See `help("download_gdc")` for more information about the applicable parameters.

> **Important Note** - In this package all tumor names in `tumor` argument must be given by their abbreviation. For example, 'BRCA' must be used as `tumor` argument for 'Breast invasive carcinoma'. Check this page for more details.

Examples

- In order to download specific data from Uterine Carcinosarcoma (UCS):
  - **Isoform expression data from GDC Legacy Archive:**

```r
download_gdc(dataType = "isoform",
            dataBase = "legacy",
            tumor = "UCS",
            workDir = "/path/to/home/dir")
```

  - **Methylation data from GDC Legacy Archive:**

```r
download_gdc(dataType = "methylation",
            dataBase = "legacy",
            tumor = "UCS",
            workDir = "/path/to/home/dir",
            Platform = "Illumina Human Methylation 450")
```

  - **Mutation data from GDC Data Portal:**

```r
download_gdc(dataType = "mutation",
            dataBase = "GDC",
            tumor = "UCS",
            workDir = "/path/to/home/dir",
            Platform = "Illumina HiSeq")
```

  - **Gene expression data from GDC Data Portal:**

```r
download_gdc(dataType = "gene",
            dataBase = "GDC",
            tumor = "UCS",
            HTSeq = "FPKM",
            workDir = "/path/to/home/dir")
```

Important Note - Some data are specific to a database:

- Exclusive in in GDC Legacy Archive: “protein”, “Exon quantification”, “miRNA gene quantification”, “miRNA isoform quantification”, “isoform”, “image”, and “clinical”.
- Exclusive in in GDC Data Portal: code{“miRNA Expression Quantification”, “Isoform Expression Quantification”(miRNA).

Concatenate patients in one table

The downloaded files are individualized by patients. All next analysis In the example below:d by **DOAGDC** require the values organized in a single table (object). This requirement can be achieved by running the `concatenate_files()` function. See `help("concatenate_files")` for more information.

Concatenate data from Uterine carcinosarcoma (UCS) examples:

- **Isoform expression normalized data from GDC Legacy Archive:**

```r
concatenate_files(dataType = "isoform",
                normalization = TRUE,
                tumorData = TRUE,
                dataBase = "legacy",
                workDir = "/path/to/home/dir",
                tumor = "UCS")
```

- **Mutation data from GDC Legacy Archive:**

```r
concatenate_files(dataType = "mutation",
                workflowType = "varscan",
                Platform = "Illumina GA",
                tumorData = TRUE,
                dataBase = "legacy",
                workDir = "/path/to/home/dir",
                tumor = "UCS")
```

- **miRNA expression data from GDC Data Portal and save concatenate table to disk:**

```r
concatenate_files(dataType = "miRNA",
                use.hg19.mirbase20 = TRUE,
                tumorData = TRUE,
                dataBase = "GDC",
                workDir = "/path/to/home/dir",
                tumor = "UCS",
                saveData = TRUE)
```

- **Gene expression not normalized data from GDC Data Portal:**

```
concatenate_files(dataType = "gene",
                HTSeq = "Counts",
                tumorData = TRUE,
                normalization = FALSE,
                dataBase = "GDC",
                workDir = "/path/to/home/dir",
                tumor = "UCS")
```

- **Methylation data from GDC Legacy Archive:**

```
# extracting all "HIF3A" probes
concatenate_files(dataType = "gene",
                    Name = "HIF3A",
                    tumorData = TRUE,
                    dataBase = "legacy",
                    Platform = "Illumina Human Methylation 450",
                    workDir = "/path/to/home/dir",
                    tumor = "UCS")

# selecting the "cg16672562" and converting its beta to m values
beta2mValues(probeName = "cg16672562",
            env = UCS_LEGACY_methylation_tumor_data)
```

> **Important Note** - If your data is already concatenate into a single matrix, please use the `table2DOAGDC()` function to import your data into the DOAGDC environment (`table2DOAGDC(data_table, dataType = "gene", dataBase = "legacy", tumor = "UCS")`).

Groups formation

The Differential Expression Analysis requires group identification of samples/patients. The **DOAGDC** package offers three possible ways of group formation: *mclust* package, *cox HR*[5] and clinical data.

The n groups formed are named: `{'G1', 'G2', ..., 'Gn'}`. It is advised the formation of up to five groups. This avoid a huge complexity, loss of statistical power (few samples by groups), and a long run time in the next step.

Using mclust package

The `groups_identification_mclust()` function is powered by the *mclust* package. Information about the package can be found here. Some parameters like `modelName` and `group.number` must be pass to the function, if the auto setup performed by *mclust* is neither desired, nor needed. The sample classification in groups generate some values of uncertainty. Optionally, the `uncertaintyCutoff` can be used to select a maximum uncertainty value, the suggested value is 0.05. After the first run of the `groups_identification_mclust()` function, it generates images to help you define the desired cut value. If that is the case, remember to run the `groups_identification_mclust()` function using the `reRunPlots` argument as `TRUE`. See `help("groups_identification_mclust")` for more information about the applicable parameters.

[5] Cox, D. R. 1972. "Regression Models and Life-Tables." Journal of the Royal Statistical Society. Series B (Methodological). WileyRoyal Statistical Society. https://doi.org/10.2307/2985181

Examples

- Generate two groups based on “HIF3A” gene expression (hosted in “legacy” data base) in UCS tumor:

```
groups_identification_mclust(dataType = "gene",
                             group.number = 2,
                             Name = "HIF3A",
                             modelName = "E",
                             env = "env name without quotes",
                             tumor = "UCS",
                             dataBase = "legacy",
                             workDir = "/path/to/home/dir")
```

- Optionally it is possible to rerun the function changing some parameters in a quick way (i.e using the `reRunPlots` argument):

```
groups_identification_mclust(dataType = "gene",
                             group.number = 3,
                             Name = "HIF3A",
                             reRunPlots = TRUE,
                             env = "env name without quotes",
                             tumor = "UCS",
                             uncertaintyCutoff = 0.05,
                             dataBase = "legacy",
                             workDir = "/path/to/home/dir")
```

Using Clinical Data

Clinical terms vary among the 33 available tumors. When the desired clinical term is unknown, use the following code, for instance, to discovery all terms related with the word ‘weight’ use the `term_keyword` argument:

```
clinical_terms(workDir = "/path/to/home/dir",
               tumor = "UCS",
               dataBase = "legacy",
               term_keyword = "weight")
```

It will return all clinical terms available that containing the word ‘weight’ (e.g. ‘weight_kg_at_diagnosis’). A possible usage example of group identification with the `term` ‘weight_kg_at_diagnosis’ and the ‘BRCA2’ gene expression can be as follow:

```
clinical_terms(workDir = "/path/to/home/dir",
               Name = "BRCA2",
               tumor = "UCS",
               dataBase = "legacy",
               dataType = "gene",
               env = "env name without quotes",
               term = "weight_kg_at_diagnosis")
```

A box plot and text files with all statistics run are generated. See `help("clinical_terms")` for more information about the applicable parameters.

Another usage of this function is an exploratory analysis of all possible terms shared by 'n' tumors containing the `term_keyword` desired. The next example shows the mentioned usage by finding among three tumors ("UCS", "OV", and "BRCA") all terms containing the word 'weight', common to the three tumors at same time, using the `tumor_with_term` argument equal to 3.

```
clinical_terms(workDir = "/path/to/home/dir",
                tumor = c("UCS", "OV", "BRCA"),
                dataBase = "gdc",
                term_keyword = "weight",
                tumor_with_term = 3)
```

Using Cox HR

Cox HR is another option for group identification. This is the most time consuming method compared to the other two. As before, expression values are need as well as the environment name to be assign to `env` argument. Therefore, the `concatenate_files()` function should be run before this step. The same is true for the last two "Groups formation" options.

A Kaplan-Meier[6] analysis is also performed within this function and it is divided by Distant Metastasis-Free Survival , Relapse-Free Survival, and Overall Survival. See `help("groups_identification_coxHR")` for more information about the applicable parameters.

Examples

- Generate groups based on cox HR and "HIF3A" gene expression (hosted in "legacy" data base) in UCS tumor:

```
groups_identification_coxHR(Name = "HIF3A",
                            dataType = "gene",
                            env = "env name without quotes",
                            tumor = "UCS",
                            dataBase = "legacy")
```

Differential Expression Analysis (DEA)

The **DOAGDC** is powered by three different packages for this step of the analysis: *EBSeq*[7], *DESeq2*, and *edgeR*.

EBSeq DEA example

- Using the "BRCA2" gene expression (inside the environment object, i.e. `UCS_LEGACY_gene_tumor_data`) to compare the "G2_over_G1", two of the groups identified by the `groups_identification_mclust()` function. In the example below: seven rounds, and the "LRT" test provided by the *EBSeq*, FDR values maximum of .05, and Fold Change cutoff of ± 2. See `help("dea_EBSeq")` for more information about the applicable parameters.

[6] Kaplan EL, Meier P. Nonparametric estimation from incomplete observations. Journal of the American Statistical Association. 1958;53:457–81

[7] Leng, Ning, John a. Dawson, James a. Thomson, Victor Ruotti, Anna I. Rissman, Bart M G Smits, Jill D. Haag, Michael N. Gould, Ron M. Stewart, and Christina Kendziorski. 2013. "EBSeq: An Empirical Bayes Hierarchical Model for Inference in RNA-Seq Experiments." Bioinformatics 29 (8): 1035–43. https://doi.org/10.1093/bioinformatics/btt087

```r
dea_EBSeq(Name = "BRCA2",
            dataBase = "legacy",
            tumor = "UCS",
            rounds = 7,
            normType = "QuantileNorm",
            groupGen = "mclust",
            pairName = "G2_over_G1",
            FC.cutoff = 2,
            FDR.cutoff = 0.05,
            env = UCS_LEGACY_gene_tumor_data)
```

DESeq2 DEA example

- Using the "HIF3A" gene expression (inside the environment object, i.e. `UCS_LEGACY_gene_tumor_data`) to compare the "Stage_IV_over_Stage_I", two of the subgroups formed by the clinical term "clinical_stage". In the example below: two CPU cores (*DESeq2* offers parallelization), and the "LRT" test In the example below:d by the *DESeq2*, FDR values maximum of .01, and Fold Change cutoff of ±3. See `help("dea_DESeq2")` for more information about the applicable parameters.

```r
dea_DESeq2(Name = "HIF3A",
            dataBase = "legacy",
            tumor = "UCS",
            coreNumber = 2,
            test = "LRT",
            groupGen = "clinical",
            clinical_pair = "Stage_IV_over_Stage_I",
            FC.cutoff = 3,
            FDR.cutoff = 0.01,
            env = UCS_LEGACY_gene_tumor_data)
```

edgeR DEA example

- Using the "ENSG00000242268.2" gene expression (inside the environment object, i.e. `UCS_GDC_gene_tumor_data`) to compare the "G2_over_G1", two of the groups identified by the `groups_identification_coxHR()` function. In the example below: the "exacttest" method In the example below:d by the *edgeR*, FDR values maximum of .1, and Fold Change cutoff of ±3. See `help("dea_edgeR")` for more information about the applicable parameters.

```r
dea_edgeR(Name = "ENSG00000242268.2",
            dataBase = "gdc",
            tumor = "UCS",
            Method = "exacttest",
            groupGen = "CoxHR",
            pairName = "G2_over_G1",
            FC.cutoff = 3,
            FDR.cutoff = 0.1,
            env = UCS_GDC_gene_tumor_data)
```

Venn diagram of differential expression genes list

Alternatively, it is possible to use more than one DEA package. When the case, **DOAGDC** In the example below:s `CrossThem()`. This function executes a Venn diagram, returning only one consensus list of genes between the DEA packages. This consensus genes list can be used in Pathway Enrichment Analysis workflow step. See `help("CrossThem")` for more information about the applicable parameters.

Venn diagram example

- Crossing all three DEA packages and using statistical values from *DESeq2* package:

```r
CrossThem(FinalData = "DESeq2",
            n.pack = 3,
            packageNames = c("DESeq2", "edgeR", "EBSeq"),
            env = "env name without quotes")
```

- Crossing only two DEA packages (e.g. *DESeq2* and *edgeR* and using statistical values from *edgeR* package:

```r
CrossThem(FinalData = "edgeR",
            n.pack = 2,
            packageNames = c("DESeq2", "edgeR"),
            env = "env name without quotes")
```

Visualization

PCA and *Heat map* are two important visualization tools in **DOAGDC**. It is possible to use either the differential expressed genes from a single DEA package (i.e. *DESeq2*, *edgeR* or *EBSeq*), or the result list from `CrossThem()` function.

PCA example

- Using the gene list generated by *DESeq2*, after compare "G2_over_G1" pair groups formed by the `groups_identification_coxHR()` function using "hsa-mir-21" miRNA expression data, in the "UCS" tumor type (all inside the environment object, i.e. `UCS_GDC_miRNA_Expression_Quantification_tumor_data`). See `help("PCA_Analysis")` for more information about the applicable parameters.

```r
PCA_Analysis(Tool = "DESeq2",
            dataType = "miRNA",
            Name = "hsa-mir-21",
            env = UCS_GDC_miRNA_Expression_Quantification_tumor_data,
            pairName = "G2_over_G1")
```

Heat map example

- Using the gene list generated from `CrossThem()` function, after compare "G2_over_G1" pair groups formed by the `groups_identification_mclust()` function using "Caspase-8-M-E" protein expression data, in the "UCS" tumor type (all inside the environment object, i.e. `UCS_LEGACY_protein_tumor_data`). In the example below: the "euclidean" method, "row" as method of scale, and custom image Width/Height of 6 inches (default unit). See `help("draw_heatmap")` for more information about the applicable parameters.

```r
draw_heatmap(Tool = "CrossThem.edgeR",
             dataType = "protein",
             Name = "Caspase-8-M-E",
             Method = "euclidean",
             ScaleMethod = "row",
             Width = 6,
             Height = 6,
             env = UCS_LEGACY_protein_tumor_data,
             pairName = "G2_over_G1")
```

Pathway Enrichment Analysis

The **DOAGDC** offers five different PEA: Gene Ontology, Gene Set Enrichment Analysis, Disease Ontology, Reactome Enrichment, and KEEG.

Gene Ontology example

In order to perform the GO analysis, the `GOnto()` function should be used. This function is powered by *goseq* and `enrichGO()` from (*clusterProfiler* package). There are three possible condition arguments to be used in the GO analysis: `"Upregulated"`, `"Downregulated"`, or `"ALL"` (without the Upregulated and Downregulated separation).

- Using the gene list generated by *DESeq2*, after comparison of "G2_over_G1" pair groups formed by the `groups_identification_coxHR()` function using "uc002peg.3" isoform expression data, in the "UCS" tumor type (all inside the `UCS_LEGACY_isoform_tumor_data` environment object). In the example below: "GeneID". See `help("GOnto")` for more information about the applicable parameters.

```r
GOnto(condition = "Upregulated",
        Tool = "edgeR",
        ID = "GeneID",
        pairName = "G2_over_G1",
        env = UCS_LEGACY_isoform_tumor_data)
```

Gene Set Enrichment Analysis example

- Using the gene list generated by *edgeR*, after compare "G2_over_G1" pair groups formed by the `groups_identification_coxHR()` function using "uc002peg.3" isoform expression data, in the "UCS" tumor type (all inside the `UCS_LEGACY_isoform_tumor_data` environment object). In the example below: "GeneID", and FDR values maximum of .05. See `help("GSEA")` for more information about the applicable parameters.

```r
GSEA(Tool = "edgeR",
    ID = "GeneID",
    pairName = "G2_over_G1",
    FDR.cutoff = 0.05,
    env = UCS_LEGACY_isoform_tumor_data)
```

Disease Ontology and Reactome Enrichment example

- Using the gene list generated by *edgeR*, after compare “G2_over_G1” pair groups formed by the `groups_identification_coxHR()` function using an isoform expression data, in the “UCS” tumor type (all inside the `UCS_LEGACY_isoform_tumor_data` environment object). In the example, default values were passed. See `help("DO.REAC.ENRICH")` for more information about the applicable parameters.

```
DO.REAC.ENRICH(Tool = "edgeR",
                pairName = "G2_over_G1",
                env = UCS_LEGACY_isoform_tumor_data)
```

KEEG example

- Using the gene list generated by *EBSeq*, after compare “G2_over_G1” pair groups formed by the `groups_identification_mclust()` function using methylation probe “cg16672562” data, in the “UCS” tumor type (all inside the `UCS_LEGACY_methylation_tumor_data` environment object). In the example, default values were passed. See `help("KEEG_ENRICH")` for more information about the applicable parameters.

```
KEEG_ENRICH(Tool = "EBSeq",
            pairName = "G2_over_G1",
            env = UCS_LEGACY_methylation_tumor_data)
```

Co-expression Analysis

This analysis is very powerful, and it requires a powerful machine with at least 64gb of ram to run the `co_expression()` function without problems and crashes. A server/cluster would be the perfect machine to run this function.

The `co_expression()` is powered by the *WGCNA* package, more information about it can be found here. See `help("co_expression")` for more information about the `co_expression()` function.

```
co_expression(Data = NULL,
                workDir,
                tumor,
                normalization = TRUE,
                tumorData = TRUE,
                networkType = "unsigned",
                minModuleSize = 15,
                nthreads = 8,
                MEDissThres = 0.25,
                env = ,
                saveCheckpoints = TRUE loadCheckpoint)
```

Quick start

This is an entire example that shows how to run a DEA analysis, with groups generated by ‘Caspase-8-M-E’ protein expression:

```r
library(DOAGDC)
```

```r
# set the work dir path
DIR <- "/any/full/path/name"

# set the data type to use in groups identification step.
Data <- "protein"

# set the desired data base to be used ("gdc" or "legacy")
# for using GDC Legacy Archive
DB <- "legacy"

# set the tumor type
# for using 'Uterine Carcinosarcoma'
Tumor <- "UCS"


## step 1: Download
download_gdc(dataType = Data, tumor = Tumor, dataBase = DB,
  workDir = DIR)


## step 2: concatenate into a single table
# first normalized data to groups identification
concatenate_files(dataType = Data,
  Name = "Caspase-8-M-E",
  normalization = TRUE,
  tumorData = TRUE,
  dataBase = DB,
  workDir = DIR,
  tumor = Tumor)
```

It was created an object called `UCS_LEGACY_protein_tumor_data`, its name is based on your initial setups (i.e. **tumor__data-base__data-type__tumor__data**). This object prevents data of being messed up, keeping tumors, databases, and data types in the correct environment. All downstream objects are going to be store inside this environment. From now on, you must use the name of this recent created object as value to the `env` function arguments.

```r
# now, uses not normalized data, as required by the DEA packages
# remember to use the object mentioned later
concatenate_files(dataType = "gene",
  normalization = FALSE,
  tumorData = TRUE,
  dataBase = "legacy",
  workDir = DIR,
  tumor = Tumor,
  env = UCS_LEGACY_protein_tumor_data)

## step 3: groups identification
# forcing three groups here ("G1", "G2", "G3")
groups_identification_mclust(dataType = Data,
  Name = "Caspase-8-M-E",
  tumor = Tumor,
  group.number = 3,
  dataBase = DB,
  workDir = DIR,
  env = UCS_LEGACY_protein_tumor_data)
```

```
## step 4: DEA
# using edgeR 'exactTest' method
dea_edgeR(dataType = Data,
  Name = "Caspase-8-M-E",
  Method = "exactTest",
  workDir = DIR,
  env = UCS_LEGACY_protein_tumor_data)
```

```
## step 5: visualization
# PCA of 'G2_over_G1'
PCA_Analysis(Tool = "edgeR",
  dataType = Data,
  Name = "Caspase-8-M-E",
  pairName = "G2_over_G1",
  env = UCS_LEGACY_protein_tumor_data)
```

```
# Heat Map of 'G2_over_G1'
draw_heatmap(Tool = "edgeR",
  dataType = Data,
  Name = "Caspase-8-M-E",
  Method = "euclidean",
  pairName = "G2_over_G1",
  env = UCS_LEGACY_protein_tumor_data)
```

```
## step 6: Pathway enrichment analysis
# using only the 'Upregulated' genes
# powered by enrichGO (image below) and goseq
GOnto(condition = "Upregulated",
  Tool = "edgeR",
  env = UCS_LEGACY_protein_tumor_data)
```

Session info

```
sessionInfo()
#> R version 3.4.4 (2018-03-15)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Linux Mint 18.3
#>
#> Matrix products: default
#> BLAS: /usr/lib/libblas/libblas.so.3.6.0
#> LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=pt_BR.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=pt_BR.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=pt_BR.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] DOAGDC_0.99.0
#>
#> loaded via a namespace (and not attached):
#>  [1] locfit_1.5-9.1   Rcpp_1.0.1       edgeR_3.16.5     codetools_0.2-16
#>  [5] lattice_0.20-38  digest_0.6.18    grid_3.4.4       magrittr_1.5
#>  [9] evaluate_0.13    stringi_1.4.3    limma_3.30.13    rmarkdown_1.12
#> [13] tools_3.4.4      stringr_1.4.0    xfun_0.6         yaml_2.2.0
#> [17] compiler_3.4.4   htmltools_0.3.6  knitr_1.22
```