Working with TCGAbiolinks package

Antonio Colaprico, Tiago Chedraoui Silva, Luciano Garofano, Catharina Olsen, Davide Garolini, Claudia Cava, Isabella Castiglioni, Houtan Noushmehr, Gianluca Bontempi, Michele Ceccarelli

2015-07-14

Contents

1 Introduction

Motivation: The Cancer Genome Atlas (TCGA) provides us with an enormous collection of data sets, not only spanning a large number of cancers but also a large number of experimental platforms. Even though the data can be accessed and downloaded from the database, the possibility to analyse these downloaded data directly in one single R package has not yet been available.

TCGAbiolinks consists of three parts or levels. Firstly, we provide different options to query and download from TCGA relevant data from all currently platforms and their subsequent pre-processing for commonly used bio-informatics (tools) packages in Bioconductor or CRAN. Secondly, the package allows to integrate different data types and it can be used for different types of analyses dealing with all platforms such as diff.expression, network inference or survival analysis, etc, and then it allows to visualize the obtained results. Thirdly we added a social level where a researcher can found a similar intereset in a bioinformatic community, and allows both to find a validation of results in literature in pubmed and also to retrieve questions and answers from site such as support.bioconductor.org, biostars.org, stackoverflow,etc.

This document describes how to search, download and analyze TCGA data using the `TCGAbiolinks` package.

2 `TCGAQuery`: Searching TCGA open-access data

You can easily search TCGA samples using the `TCGAQuery` function. Using a summary of filters as used in the TCGA portal, the function works with the following parameters:

- **tumor** Tumor or list of tumors. The list of tumor is shown in the examples.
- **platform** Platform or list of tumors. The list of platforms is shown in the examples.
- **samples** List of TCGA barcodes
- **added.since** (format: mm/dd/YYYY)
- **added.up.to** (format: mm/dd/YYYY)
- **level** Options: 1,2,3,“mage-tab”
- **center**

2.0.1 Searching by tumor

You can filter the search by tumor using the tumor parameter.

```
query <- TCGAQuery(tumor = "gbm")
```

If you don’t remember the tumor name, or if you have incorrectly typed it. It will provide you with all the tumor names in TCGA. Also the names can be seen in the help pages `?TCGAQuery`

```
query <- TCGAQuery(tumor = "")
```

```
##
##
## Table: TCGA tumors
##
## -----  -----  -----  -----  -----  -----  -----  -----
## ACC    CNTL   GBM    LAML   LUSC   PCPG   STAD   UCS
## BLCA   COAD   HNSC   LCML   MESO   PRAD   TGCT   UVM
## BRCA   DLBC   KICH   LGG    MISC   READ   THCA   ACC
## CESC   ESCA   KIRC   LIHC   OV     SARC   THYM   BLCA
## CHOL   FPPP   KIRP   LUAD   PAAD   SKCM   UCEC   BRCA
## -----  -----  -----  -----  -----  -----  -----  -----
## ==========================================================
## ERROR: Disease not found. Select from the table above.
## ==========================================================
```

2.0.2 Searching by level

You can filter the search by level “1”, “2”, “3” or “mage-tab”

```
query <- TCGAQuery(tumor = "gbm", level = 3)
query <- TCGAQuery(tumor = "gbm", level = 2)
query <- TCGAQuery(tumor = "gbm", level = 1)
query <- TCGAQuery(tumor = "gbm", level = "mage-tab")
```

2.0.3 Searching by date

You can filter the search by date using the added.since and added.up.to parameters. For the moment, the format of date accepted is mm/dd/YYYY.

```
# Get all gbm data produced in 2013
query <- TCGAQuery(tumor = "gbm", added.since = "01/01/2013", added.up.to = "12/31/2013")
```

2.0.4 Searching by platform

You can filter the search by platform using the platform parameter.

```
query <- TCGAQuery(tumor = "gbm", platform = "IlluminaHiSeq_RNASeqV2")
```

If you don't remember the platform, or if you have incorrectly typed it. It will provide you with all the platforms names in TCGA. Also the names can be seen in the help pages ?TCGAQuery

```
query <- TCGAQuery(tumor = "gbm", platform = "")
```

```
##
##
## Table: TCGA Platforms
##
## --------------------  ------------------------------------  --------------------------------------
## 454                   HumanMethylation27                    IlluminaHiSeq_WGBS
## ABI                   HumanMethylation450                   Mapping250K_Nsp
## AgilentG4502A_07      IlluminaDNAMethylation_OMA002_CPI     Mapping250K_Sty
## AgilentG4502A_07_1    IlluminaDNAMethylation_OMA003_CPI     MDA_RPPA_Core
## AgilentG4502A_07_2    IlluminaGA_DNASeq                     microsat_i
## AgilentG4502A_07_3    IlluminaGA_DNASeq_automated           minbio
## bio                   IlluminaGA_DNASeq_Cont                minbiotab
## biotab                IlluminaGA_DNASeq_Cont_automated      Mixed_DNASeq
## CGH-1x1M_G4447A       IlluminaGA_DNASeq_Cont_curated        Mixed_DNASeq_automated
## diagnostic_images     IlluminaGA_DNASeq_curated             Mixed_DNASeq_Cont
## fh_analyses           IlluminaGA_miRNASeq                   Mixed_DNASeq_Cont_automated
## fh_reports            IlluminaGA_mRNA_DGE                   Mixed_DNASeq_Cont_curated
## fh_stddata            IlluminaGA_RNASeq                     Mixed_DNASeq_curated
## Genome_Wide_SNP_6     IlluminaGA_RNASeqV2                   Multicenter_mutation_calling_MC3
## GenomeWideSNP_5       IlluminaGG                            Multicenter_mutation_calling_MC3_Cont
## H-miRNA_8x15K         IlluminaHiSeq_DNASeq                  pathology_reports
## H-miRNA_8x15Kv2       IlluminaHiSeq_DNASeq_automated        SOLiD_DNASeq
## H-miRNA_EarlyAccess   IlluminaHiSeq_DNASeq_Cont             SOLiD_DNASeq_automated
## H-miRNA_G4470A        IlluminaHiSeq_DNASeq_Cont_automated   SOLiD_DNASeq_Cont
## HG-CGH-244A           IlluminaHiSeq_DNASeq_Cont_curated     SOLiD_DNASeq_Cont_automated
## HG-CGH-415K_G4124A    IlluminaHiSeq_DNASeq_curated          SOLiD_DNASeq_Cont_curated
## HG-U133_Plus_2        IlluminaHiSeq_DNASeqC                 SOLiD_DNASeq_curated
## HG-U133A_2            IlluminaHiSeq_miRNASeq                supplemental_clinical
## HT_HG-U133A           IlluminaHiSeq_mRNA_DGE                tissue_images
## HuEx-1_0-st-v2        IlluminaHiSeq_RNASeq                  WHG-1x44K_G4112A
## Human1MDuo            IlluminaHiSeq_RNASeqV2                WHG-4x44K_G4112F
## HumanHap550           IlluminaHiSeq_TotalRNASeqV2           WHG-CGH_4x44B
## --------------------  ------------------------------------  ------------------------------------
## ========================================================
## ERROR: Platform not found. Select from the table above.
## ========================================================
```

2.0.5 Searching by center

You can filter the search by center using the center parameter.

```
query <- TCGAQuery(tumor = "gbm", center = "mskcc.org")
```

If you don't remember the center or if you have incorrectly typed it. It will provide you with all the center names in TCGA.

```
query <- TCGAQuery(tumor = "gbm", center = "")
```

```
##
##
## Table: TCGA Centers
##
## -------------------  ------------------------  --------------------
## bcgsc.ca             intgen.org                rubicongenomics.com
## broad.mit.edu        jhu-usc.edu               sanger.ac.uk
## broadinstitute.org   jhu.edu                   systemsbiology.org
## combined GSCs        lbl.gov                   ucsc.edu
## genome.wustl.edu     mdanderson.org            unc.edu
## hgsc.bcm.edu         mskcc.org                 usc.edu
## hms.harvard.edu      nationwidechildrens.org   vanderbilt.edu
## hudsonalpha.org      pnl.gov                   bcgsc.ca
## -------------------  ------------------------  --------------------
## ========================================================
## ERROR: Center not found. Select from the table above.
## ========================================================
```

2.0.6 Searching by samples

You can filter the search by samples using the samples parameter. You can give a list of barcodes or only one barcode. These barcode can be partial barcodes.

```
# You can define a list of samples to query and download providing relative TCGA barcodes.
listSamples <- c("TCGA-E9-A1NG-11A-52R-A14M-07","TCGA-BH-A1FC-11A-32R-A13Q-07",
                 "TCGA-A7-A13G-11A-51R-A13Q-07","TCGA-BH-A0DK-11A-13R-A089-07",
                 "TCGA-E9-A1RH-11A-34R-A169-07","TCGA-BH-A0AU-01A-11R-A12P-07",
                 "TCGA-C8-A1HJ-01A-11R-A13Q-07","TCGA-A7-A13D-01A-13R-A12P-07",
                 "TCGA-A2-A0CV-01A-31R-A115-07","TCGA-AQ-A0Y5-01A-11R-A14M-07")

# Query all available platforms with a list of barcode
query <- TCGAQuery(samples = listSamples)

# Query with a partial barcode
query <- TCGAQuery(samples = "TCGA-61-1743-01A")
```

2.0.7 Examples

Some search examples are shown below:

```
query <- TCGAQuery(tumor = "gbm", added.since = "01/01/2013", added.up.to = "12/31/2013")

query <- TCGAQuery(tumor = c("gbm","lgg"), platform = c("HumanMethylation450","HumanMethylation27"))

query <- TCGAQuery(tumor = "gbm", platform = "HumanMethylation450", level = "3")

query <- TCGAQuery(samples = "TCGA-61-1743-01A-01D")

query <- TCGAQuery(samples = "TCGA-61-1743-01A-01D-0649-04", level = 3)

query <- TCGAQuery(samples = "TCGA-61-1743-01A-01D-0649-04", tumor = "OV", platform = "CGH-1x1M_G4447A")
```

3 `TCGAintegrate`: Summary of the common numbers of patient samples in different platforms

Some times researches would like to use samples from different platforms from the same patient. In order to help the user to have an overview of the number of samples in commun we created the function `TCGAintegrate` that will receive the data frame returned from `TCGAQuery` and produce a matrix n platforms x n platforms with the values of samples in commum.

Some search examples are shown below

```
query <- TCGAQuery(tumor = "brca",level = 3)
matSamples <- TCGAintegrate(query)
```

```
## [1] 604   1
## [1] 2228    1
## [1] 349   1
## [1] 913   1
## [1] 343   1
## [1] 38  1
## [1] 869   1
## [1] 884   1
## [1] 1218    1
## [1] 24  1
## [1] 6 1
## [1] 410   1
```

The result of the 3 platforms of `TCGAintegrate` result is shown below:

Table 1: Table common samples among platforms from TCGAQuery

	AgilentG4502A_07_3	HumanMethylation450	IlluminaHiSeq_RNASeqV2
AgilentG4502A_07_3	604	224	530
HumanMethylation450	224	913	775
IlluminaHiSeq_RNASeqV2	530	775	1218

4 `TCGAVersion`: Summary versions of the data in TCGA

Query version for a specific platform for example IlluminaHiSeq_RNASeqV2

```r
library(TCGAbiolinks)

BRCA_RNASeqV2_version <- TCGAVersion(tumor = "brca",
                                     platform = "illuminahiseq_rnaseqv2")
```

The result is shown below:

Table 2: Table with version, number of samples and size (Mbyte) of BRCA IlluminaHiSeq_RNASeqV2 Level 3

Version	Date	Samples	SizeMbyte
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.11.0/	2015-01-28 03:16	1218	1740.6
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.10.0/	2014-10-15 18:09	1215	1736.4
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.9.0/	2014-07-14 18:13	1182	1689.6
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.8.0/	2014-05-05 23:14	1172	1675.2
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.7.0/	2014-02-13 20:47	1160	1657.9
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.6.0/	2014-01-13 03:53	1140	1629.1
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.5.0/	2013-08-22 18:05	1106	1580.8
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.4.0/	2013-04-25 16:36	1032	1476.5
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.3.0/	2013-04-12 15:28	958	1369.3
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.2.0/	2012-12-17 18:23	956	1366.5
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.1.0/	2012-07-27 17:52	919	1312.9
unc.edu_BRCA.IlluminaHiSeq_RNASeqV2.Level_3.1.0.0/	2012-05-18 12:21	858	1226.1

5 `TCGADownload`: Downloading open-access data

You can easily download data using the `TCGADownload` function.

The arguments are:

- **data** The `TCGAQuery` output
- **path** location to save the files. Default: "."
- **type** Filter the files to download by type
- **samples** List of samples to download
- **quiet** Supress output messages? Default: FALSE
- **force** Download again if file already exists? Default: FALSE

5.0.8 Example of use

```r
# get all samples from the query and save them in the TCGA folder
# samples from IlluminaHiSeq_RNASeqV2 with type rsem.genes.results
# samples to normalize later

TCGADownload(query, path = "data", type = "rsem.genes.results")

TCGADownload(query, path = "data", type = "rsem.isoforms.normalized_results")


TCGADownload(query, path = "dataBrca", type = "rsem.genes.results",
             samples = c("TCGA-E9-A1NG-11A-52R-A14M-07",
                         "TCGA-BH-A1FC-11A-32R-A13Q-07")
             )
```

Comment: The function will structure the folders to save the data as: *Path given by the user/Experiment folder*

5.0.9 Table of types available for downloading

Platform	
Illuminahiseq_totalrnaseqv2 IlluminaHiSeq_RNASeqV2IlluminaGA_RNASeqV2 IlluminaHiSeq_RNASeqilluminaga_rnaseq genome_wide_snp_6	junction_quantificationrsem.genes.resultsrsem.isofo exon.q hg

6 TCGAPrepare: Preparing the data

You can easily read the downloaded data using the TCGAPrepare function. This function will prepare the data into a SummarizedExperiment object for downstream analysis. For the moment this function is working only with data level 3.

The arguments are:

- **query** Data frame as the one returned from TCGAQuery
- **dir** Directory with the files
- **type** File to prepare.
- **save** Save a rda object with the prepared object? Default: FALSE
- **filename** Name of the rda object that will be saved if save is TRUE
- **toPackage** Name of the package to prepare the data specific to that package.
- **summarizedExperiment** Should the output be a SummarizedExperiment object? Default: TRUE

6.0.10 Example of use

```
# get all samples from the query and save them in the TCGA folder
# samples from IlluminaHiSeq_RNASeqV2 with type rsem.genes.results
# samples to normalize later
data <- TCGAPrepare(query, dir = "data", save = TRUE, filename = "myfile.rda")
```

As an example, for the platform IlluminaHiSeq_RNASeqV2 we prepared two samples (TCGA-DY-A1DE-01A-11R-A155-07 and TCGA-DY-A0XA-01A-11R-A155-07) for the rsem.genes.normalized_results type. In order to create the object mapped the gene_id to the hg19. The genes_id not found are then removed from the final matrix. The default output is a SummarizedExperiment is shown below.

```
    data
```

```
## class: RangedSummarizedExperiment
## dim: 19947 2
## metadata(0):
## assays(1): raw_counts
## rownames(19947): A1BG|1 A2M|2 ... TMED7-TICAM2|100302736
##   LOC100303728|100303728
## rowRanges metadata column names(2): gene_id entrezgene
## colnames(2): TCGA-DY-A1DE-01A-11R-A155-07
##   TCGA-DY-A0XA-01A-11R-A155-07
## colData names(1): sample
```

```
    head(assay(data,"raw_counts"))
```

```
##              TCGA-DY-A1DE-01A-11R-A155-07 TCGA-DY-A0XA-01A-11R-A155-07
## A1BG|1                            13.6732                      13.0232
## A2M|2                           5030.4792                    1461.9358
## NAT1|9                            70.5969                      61.1727
## NAT2|10                           13.7272                      44.4892
## SERPINA3|12                       62.7528                      40.4447
## AADAC|13                           1.4708                       1.0111
```

In order to create the SummarizedExperiment object we mapped the rows of the experiments into GRanges. In order to map miRNA we used the miRNA from the anotation database TxDb.Hsapiens.UCSC.hg19.knownGene, this will exclude the miRNA from viruses and bacteria. In order to map genes, genes alias, we used the biomart hg19 database (hsapiens_gene_ensembl from grch37.ensembl.org).

In case you prefere to have the raw data. You can get a data frame without any modification setting the `summarizedExperiment` to false.

```
    class(data)
```

```
## [1] "data.frame"
```

```
    dim(data)
```

```
## [1] 20531     2
```

```
    head(data)
```

```
##              TCGA-DY-A1DE-01A-11R-A155-07 TCGA-DY-A0XA-01A-11R-A155-07
## ?|100130426                        0.0000                       0.0000
## ?|100133144                       11.5308                      32.9877
## ?|100134869                        4.1574                      12.5126
## ?|10357                          222.1498                     102.8308
## ?|10431                         1258.9778                     774.5168
## ?|136542                           0.0000                       0.0000
```

6.0.11 Table of `types` available for the `TCGAPrepare`

Platform	
Illuminahiseq_totalrnaseqv2 IlluminaHiSeq_RNASeqV2IlluminaGA_RNASeqV2 IlluminaHiSeq_RNASeqilluminaga_rnaseq genome_wide_snp_6	junction_quantificationrsem.genes.resultsrsem.isofo exon.q

6.0.12 Preparing the data with TCGAPrepare - toPackage

This section will show how to integrate `TCGAbiolinks` with other packages. Our intention is to provide as many integrations as possible.

The example below shows how to use `TCGAbiolinks` with `ELMER` package (expression/methylation analysis). The TCGAPrepare for the methylation data will Removing probes with NA values in more than 0.80% samples and remove the anottation data, fot the expression data it will take the log2(expression + 1) of the expression matrix in order to To linearize the relation between methylation and expressionm also it will prepare the rownames as the specified by the package.

```
############## Get tumor samples with TCGAbiolinks
library(TCGAbiolinks)
```

```
query <- TCGAQuery(tumor = "GBM",level = 3, platform = "HumanMethylation450")
TCGADownload(query,path = "TCGA/450k")
met <- TCGAPrepare(query,dir = "TCGA/450k",
                   save = TRUE,
                   filename = "met.rda",
                   toPackage = "ELMER")

query.rna <- TCGAQuery(tumor="GBM",level=3, platform="IlluminaHiSeq_RNASeqV2")
TCGADownload(query.rna,path="TCGA/rna",type = "rsem.genes.normalized_results")
exp <- TCGAPrepare(query.rna, dir="TCGA/rna", save = TRUE, filename = "exp.rda",toPackage = "ELMER")

############# To EMLER
library(ELMER)

########## genne annotation
geneAnnot <- txs()
geneAnnot$GENEID <- paste0("ID",geneAnnot$GENEID)
geneInfo <- promoters(geneAnnot,upstream = 0, downstream = 0)
############ probe
probe <- get.feature.probe()

mee.gbm.glial.with.exp <- fetch.mee(meth = gbm.glial.m,
                                    exp = exp,
                                    probeInfo = probe,
                                    TCGA = TRUE,
                                    geneInfo = geneInfo)
```

7 Examples TCGAQuery, TCGADownload, TCGAPrepare

7.1 Gene Expression IlluminaHiSeq_RNASeq

You can easily search TCGA samples, download and prepare a matrix of gene expression.

```
# Query platform IlluminaHiSeq_RNASeq withot a list of barcode
query <- TCGAQuery(tumor = "brca", platform = "IlluminaHiSeq_RNASeq", level = "3")

# You can define a list of samples to query and download providing relative TCGA barcodes.
listSamples <- samplesfilter(query)

# Download only first 5 samples for test.

TCGADownload(query, path = "dataBrca", type = "gene.quantification",
             samples = listSamples$IlluminaHiSeq_RNASeq[1:5])

# Prepare expression matrix with gene id in rows and samples (barcode) in columns
# rsem.genes.results as values
BRCAMatrix <- TCGAPrepare(query,"dataBrca",type = "gene.quantification")
```

7.2 Gene Expression IlluminaHiSeq_RNASeqV2

You can easily search TCGA samples, download and prepare a matrix of gene expression.

```r
# You can define a list of samples to query and download providing relative TCGA barcodes.

listSamples <- c("TCGA-E9-A1NG-11A-52R-A14M-07","TCGA-BH-A1FC-11A-32R-A13Q-07",
                 "TCGA-A7-A13G-11A-51R-A13Q-07","TCGA-BH-A0DK-11A-13R-A089-07",
                 "TCGA-E9-A1RH-11A-34R-A169-07","TCGA-BH-A0AU-01A-11R-A12P-07",
                 "TCGA-C8-A1HJ-01A-11R-A13Q-07","TCGA-A7-A13D-01A-13R-A12P-07",
                 "TCGA-A2-A0CV-01A-31R-A115-07","TCGA-AQ-A0Y5-01A-11R-A14M-07")

# Query all available platforms with a list of barcode
query <- TCGAQuery(samples = listSamples)

# Query platform IlluminaHiSeq_RNASeqV2 with a list of barcode
query <- TCGAQuery(tumor = "brca", samples = listSamples,
  platform = "IlluminaHiSeq_RNASeqV2", level = "3")

# dont run
#TCGADownload(query, path = "dataBrca", type = "gene.quantification",samples = listSamples)

# Download a list of barcodes with platform IlluminaHiSeq_RNASeqV2
TCGADownload(query, path = "dataBrca", type = "rsem.genes.results",samples = listSamples)

# Prepare expression matrix with gene id in rows and samples (barcode) in columns
# rsem.genes.results as values
BRCAMatrix <- TCGAPrepare(query,"dataBrca",type = "rsem.genes.results")
```

The result is shown below:

Table 5: Example of
7 samples in column

	TCGA-E9-A1RH-11A-34R-A169-07	TCGA-E9-A1NG-11A-52R-A14M-07	TCGA-A7-A13G-11A-51R-
C14orf156|81892	1103	829	
IKBKG|8517	794	990	
SNORD116-18|100033430	0	0	
FRAT1|10023	259	143	
LNX1|84708	755	872	
YLPM1|56252	4749	4606	
MTNR1A|4543	0	0	
AAMP|14	4604	4091	
KIF3C|3797	748	712	
GPR161|23432	202	296	

7.3 CNV

You can easily search TCGA samples, download and prepare a matrix of gene expression.

```r
# Define a list of samples to query and download providing relative TCGA barcodes.
samplesList <- c("TCGA-02-0046-10A-01D-0182-01",
                 "TCGA-02-0052-01A-01D-0182-01",
                 "TCGA-02-0033-10A-01D-0182-01",
                 "TCGA-02-0034-01A-01D-0182-01",
                 "TCGA-02-0007-01A-01D-0182-01")
```

```r
# Query platform Genome_Wide_SNP_6 with a list of barcode
query <- TCGAQuery(tumor = "gbm", level = 3, platform = "Genome_Wide_SNP_6")

# Download a list of barcodes with platform Genome_Wide_SNP_6
TCGADownload(query, path = "samples")

# Prepare matrix
GBM_CNV <- TCGAPrepare(query, dir = "samples", type = ".hg19.seg.txt")
```

8 `clinic` & `clinicFilt`: Working with clinical data

You can retrive clinical data using the `clinic` function. The parameters of this function are:

- cancer ("OV","BRCA","GBM", etc)
- clinical_data_type ("clinical_patient","clinical_drug", etc)

A full list of cancer and clinical data type can be found in the help of the function.

```r
# Get clinical data
clinical_brca_data <- clinic("brca","clinical_patient")
clinical_uvm_data_bio <- clinic("uvm","biospecimen_normal_control")
clinical_brca_data_bio <- clinic("brca","biospecimen_normal_control")
clinical_brca_data <- clinic("brca","clinical_patient")
```

Also, some functions to work with clinical data are provided. For example the function `clinicFilt` will filter your data, returning the list of barcodes that matches all the filter.

The parameters of `clinicFilt` are:

- **barcode** List of barcodes
- **clinical_patient_data** clinical patient data obtained with clinic function Ex: clinical_patient_data <- clinic("LGG","clinical_patient")
- **HER** her2 neu immunohistochemistry receptor status: "Positive" or "Negative"
- **gender** "MALE" or "FEMALE"
- **PR** Progesterone receptor status: "Positive" or "Negative"
- **stage** Pathologic Stage: "stage_IX", "stage_I", "stage_IA", "stage_IB", "stage_IIX", "stage_IIA", "stage_IIB", "stage_IIIX","stage_IIIA", "stage_IIIB", "stage_IIIC", "stage_IV" -
- **ER** Estrogen receptor status: "Positive" or "Negative"

```r
bar <- c("TCGA-G9-6378-02A-11R-1789-07", "TCGA-CH-5767-04A-11R-1789-07",
         "TCGA-G9-6332-60A-11R-1789-07", "TCGA-G9-6336-01A-11R-1789-07",
         "TCGA-G9-6336-11A-11R-1789-07", "TCGA-G9-7336-11A-11R-1789-07",
         "TCGA-G9-7336-04A-11R-1789-07", "TCGA-G9-7336-14A-11R-1789-07",
         "TCGA-G9-7036-04A-11R-1789-07", "TCGA-G9-7036-02A-11R-1789-07",
         "TCGA-G9-7036-11A-11R-1789-07", "TCGA-G9-7036-03A-11R-1789-07",
         "TCGA-G9-7036-10A-11R-1789-07", "TCGA-BH-A1ES-10A-11R-1789-07",
         "TCGA-BH-A1F0-10A-11R-1789-07", "TCGA-BH-A0BZ-02A-11R-1789-07",
         "TCGA-B6-A0WY-04A-11R-1789-07", "TCGA-BH-A1FG-04A-11R-1789-08",
         "TCGA-D8-A1JS-04A-11R-2089-08", "TCGA-AN-A0FN-11A-11R-8789-08",
         "TCGA-AR-A2LQ-12A-11R-8799-08", "TCGA-AR-A2LH-03A-11R-1789-07",
         "TCGA-BH-A1F8-04A-11R-5789-07", "TCGA-AR-A24T-04A-55R-1789-07",
         "TCGA-AO-A0J5-05A-11R-1789-07", "TCGA-BH-A0B4-11A-12R-1789-07",
         "TCGA-B6-A1KN-60A-13R-1789-07", "TCGA-AO-A0J5-01A-11R-1789-07",
         "TCGA-AO-A0J5-01A-11R-1789-07", "TCGA-G9-6336-11A-11R-1789-07",
         "TCGA-G9-6380-11A-11R-1789-07", "TCGA-G9-6380-01A-11R-1789-07",
         "TCGA-G9-6340-01A-11R-1789-07","TCGA-G9-6340-11A-11R-1789-07")

S <- SampleTypes(bar,"TP")
S2 <- SampleTypes(bar,"NB")

# Retrieve multiple tissue types  NOT FROM THE SAME PATIENTS
SS <- MultiSampleTypes(bar,c("TP","NB"))

# Retrieve multiple tissue types  FROM THE SAME PATIENTS
SSS <- MatchedCoupledSampleTypes(bar,c("NT","TP"))

# Get clinical data
clinical_brca_data <- clinic("brca","clinical_patient")
female_erpos_herpos <- clinicFilt(bar,clin, HER="Positive", gender="FEMALE", ER="Positive")
```

The result is shown below:

```
## ER Positive Samples:
##   TCGA-BH-A1ES
##   TCGA-BH-A0BZ
##   TCGA-B6-A0WY
##   TCGA-BH-A1FG
##   TCGA-D8-A1JS
##   TCGA-AN-A0FN
##   TCGA-AR-A2LQ
##   TCGA-BH-A1F8
##   TCGA-AR-A24T
##   TCGA-AO-A0J5
##   TCGA-BH-A0B4
##
## HER Positive Samples:
##   TCGA-AN-A0FN
##   TCGA-BH-A1F8
##
## GENDER FEMALE Samples:
##   TCGA-BH-A1ES
##   TCGA-BH-A1F0
##   TCGA-BH-A0BZ
##   TCGA-B6-A0WY
##   TCGA-BH-A1FG
##   TCGA-D8-A1JS
##   TCGA-AN-A0FN
##   TCGA-AR-A2LQ
##   TCGA-AR-A2LH
##   TCGA-BH-A1F8
##   TCGA-AR-A24T
##   TCGA-AO-A0J5
##   TCGA-B6-A1KN
```

```
## [1] "TCGA-AN-A0FN" "TCGA-BH-A1F8"
```

9 TCGA Downstream Analysis

After preparing the gene expression from TCGA data using the `TCGAPrepare` function, you can do a normalization of genes using the function `RnaSeqNormalization`, do a quantile filter of genes with the `RnaSeqFilt` function. Also, iin order to classify your samples (barcode) you can use the `MultiSampleTypes` function, the typeSample “NT” will return the “Solid Tissue Normal”” samples, while the typeSample “TP” will return “Primary Solid Tumor”” samples.

```
# Downstream analysis using gene expression data
# TCGA samples from IlluminaHiSeq_RNASeqV2 with type rsem.genes.results

library(TCGAbiolinks)

# dataBRCA in TCGAbiolinks package is a table from TCGA BRCA [10 samples] and comes from
# BRCAMatrix <- TCGAPrepare(query,"dataBrca") from above example
# dataBRCA <- BRCAMatrix

# normalization of genes
dataNorm <- TCGAbiolinks::RnaSeqNormalization(dataBRCA, geneInfo)

# quantile filter of genes
dataFilt <- RnaSeqFilt(dataNorm, 0.25)

# selection of normal samples "NT"
samplesNT <- MultiSampleTypes(colnames(dataFilt), typesample = c("NT"))

# selection of tumor samples "TP"
samplesTP <- MultiSampleTypes(colnames(dataFilt), typesample = c("TP"))
```

9.1 `DEArnaSEQ` & `CreateTabLevel` Differential expression analysis (DEA)

Perform DEA (Differential expression analysis) to identify differentially expressed genes (DEGs) using the `DEArnaSEQ` function. This function receives as parameters:

- **mat1** The matrix of the first group (in the example group 1 is the normal samples),
- **mat2** The matrix of the second group (in the example group 2 is tumor samples)
- **Cond1type** Label for group 1
- **Cond1type** Label for group 2

After, we filter the output of dataDEGs by abs(LogFC) >=1, and uses the `CreateTabLevel` function to create a table with DEGs (differentially expressed genes), log Fold Change (FC), false discovery rate (FDR), the gene expression level for samples in Cond1type, and Cond2type, and Delta value (the difference of gene expression between the two conditions multiplied logFC).

```
# Downstream analysis using gene expression data
# TCGA samples from IlluminaHiSeq_RNASeqV2 with type rsem.genes.results
# save(dataBRCA, geneInfo , file = "dataGeneExpression.rda")
library(TCGAbiolinks)

# Diff.expr.analysis (DEA)
dataDEGs <- DEArnaSEQ(dataFilt[,samplesNT], dataFilt[,samplesTP],
                      "Normal", "Tumor")

# DEGs filter by abs(logFC) >=1
dataDEGsFilt <- dataDEGs[abs(dataDEGs$logFC) >= 1,]
```

```
# DEGs table with expression values in normal and tumor samples
dataDEGsFiltLevel <- CreateTabLevel(dataDEGsFilt,"Tumor","Normal",
                                    dataFilt[,samplesTP],dataFilt[,samplesNT])
```

The result is shown below:

Table 6: Table DEGs after DEA

mRNA	logFC	FDR	Tumor	Normal	Delta
FN1	2.88	1.296151e-19	347787.48	41234.12	1001017.3
COL1A1	1.77	1.680844e-08	358010.32	89293.72	633086.3
C4orf7	5.20	2.826474e-50	87821.36	2132.76	456425.4
COL1A2	1.40	9.480478e-06	273385.44	91241.32	383242.9
GAPDH	1.32	3.290678e-05	179057.44	63663.00	236255.5
CLEC3A	6.79	7.971002e-74	27257.16	259.60	185158.6
IGFBP5	1.24	1.060717e-04	128186.88	53323.12	158674.6
CPB1	4.27	3.044021e-37	37001.76	2637.72	157968.8
CARTPT	6.72	1.023371e-72	21700.96	215.16	145872.8
DCD	7.26	1.047988e-80	19941.20	84.80	144806.3

9.2 EAcomplete & EAbarplot: Enrichment Analysis

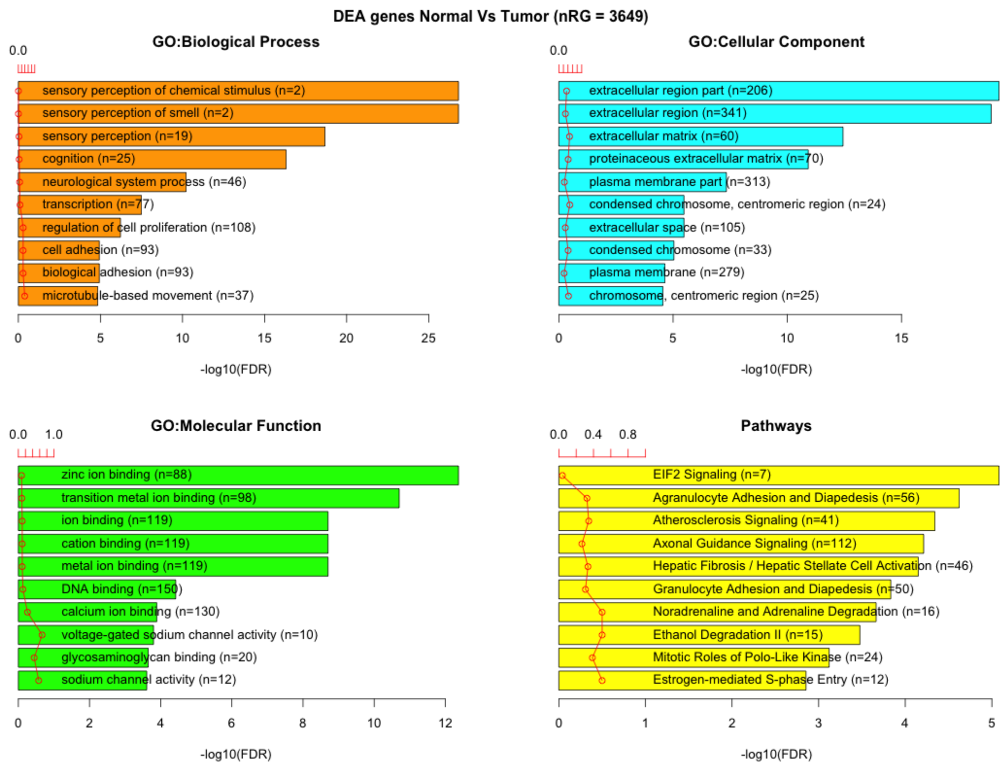
Researchers, in order to better understand the underlying biological processes, often want to retrieve a functional profile of a set of genes that might have an important role. This can be done by performing an enrichment analysis.

We will perform an enrichment analysis on gene sets using the `EAcomplete` function. Given a set of genes that are up-regulated under certain conditions, an enrichment analysis will find identify classes of genes or proteins that are over-represented using annotations for that gene set.

To view the results you can use the `EAbarplot` function as shown below.

```
library(TCGAbiolinks)
# Enrichment Analysis EA
# Gene Ontology (GO) and Pathway enrichment by DEGs list
Genelist <- rownames(dataDEGsFiltLevel)

system.time(ansEA <- EAcomplete(TFname="DEA genes Normal Vs Tumor",Genelist))

# Enrichment Analysis EA (TCGAVisualize)
# Gene Ontology (GO) and Pathway enrichment barPlot

EAbarplot(tf = rownames(ansEA$ResBP),
          GOBPTab = ansEA$ResBP,
          GOCCTab = ansEA$ResCC,
          GOMFTab = ansEA$ResMF,
          PathTab = ansEA$ResPat,
          nRGTab = Genelist,
          nBar = 10)
```

The result is shown below:



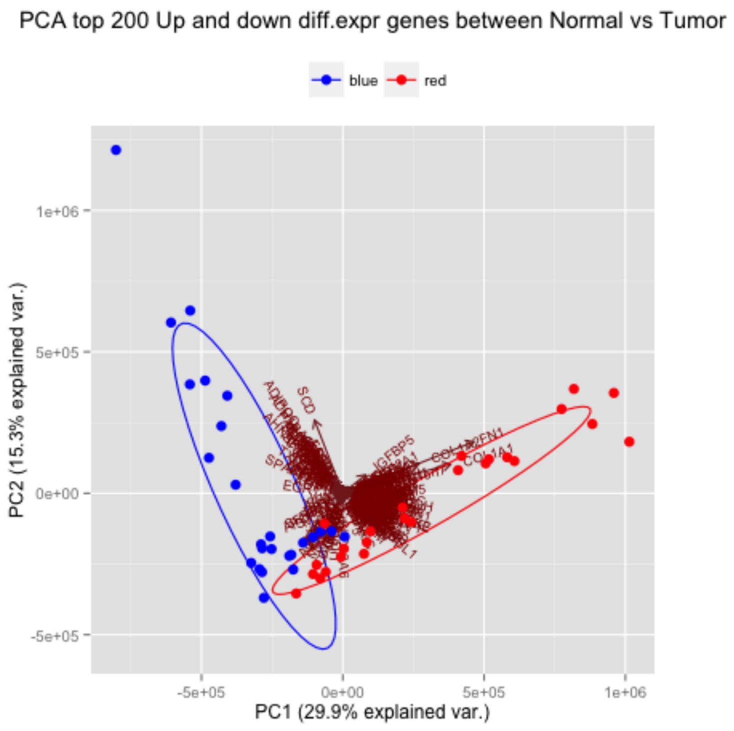
9.3 plotPCAforGroups: Principal Component Analysis plot for differentially expressed genes

In order to understand better our genes, we can perform a PCA to reduce the number of dimensions of our gene set. The function plotPCAforGroups will plot the PCA for different groups.

The parameters of this function are: * **dataFilt** The expression matrix after normalization and quantile filter * **dataDEGsFiltLevel** The CreateTabLevel output * **ntopgenes** number of DEGs genes to plot in PCA

```r
library(TCGAbiolinks)

# normalization of genes
dataNorm <- TCGAbiolinks::RnaSeqNormalization(dataBRCA, geneInfo)

# quantile filter of genes
dataFilt <- RnaSeqFilt(dataNorm, 0.25)

# Principal Component Analysis plot for ntop selected DEGs
plotPCAforGroups(dataFilt,dataDEGsFiltLevel, ntopgenes = 200)

# boxplot of normalized data
#sampleGenes <- rownames(dataDEGsFilt[dataDEGsFilt$logFC >=1,])[1:20]
#boxplot(log(dataBRCA[sampleGenes,]), las = 2)
#boxplot(log(dataFilt[sampleGenes,]), las = 2)
```

The result is shown below:



9.4 Survival Analysis

```
library(TCGAbiolinks)
# Survival Analysis SA

clinical_patient_Cancer <- clinic("brca","clinical_patient")
dataBRCAcomplete <- log2(BRCA_rnaseqv2)

tokenStop<- 1

tabSurvKMcomplete <- NULL

for( i in 1: round(nrow(dataBRCAcomplete)/100)){
message( paste( i, "of ", round(nrow(dataBRCAcomplete)/100)))
tokenStart <- tokenStop
tokenStop <-100*i
tabSurvKM<-SurvivalKMunivariate(clinical_patient_Cancer,dataBRCAcomplete,Genelist = rownames(dataBRCAcompl
                                        Survresult = F,ThreshTop=0.67,ThreshDown=0.33)

tabSurvKMcomplete <- rbind(tabSurvKMcomplete,tabSurvKM)
}

tabSurvKMcomplete <- tabSurvKMcomplete[tabSurvKMcomplete$pvalue < 0.01,]
tabSurvKMcomplete <- tabSurvKMcomplete[!duplicated(tabSurvKMcomplete$mRNA),]
rownames(tabSurvKMcomplete) <-tabSurvKMcomplete$mRNA
tabSurvKMcomplete <- tabSurvKMcomplete[,-1]
tabSurvKMcomplete <- tabSurvKMcomplete[order(tabSurvKMcomplete$pvalue, decreasing=F),]
```

```
tabSurvKMcompleteDEGs <- tabSurvKMcomplete[rownames(tabSurvKMcomplete) %in% dataDEGsFiltLevel$mRNA,]
```

The result is shown below:

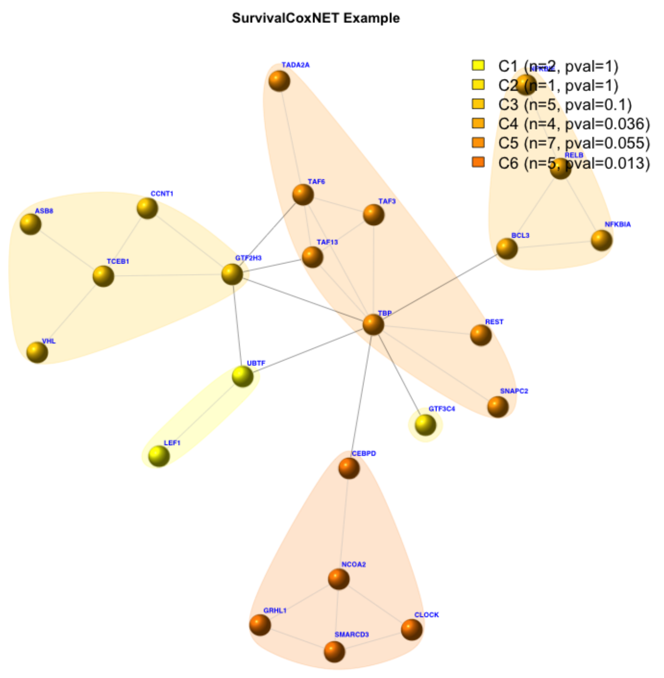
Table 7: Table KM-survival genes after SA

	pvalue	Cancer Deaths	Cancer Deaths with Top	Cancer Deaths with Down	Mean Tumor Top	Mean Tu
DCTPP1	6.204170e-08	66	46	20	13.31	
APOO	9.390193e-06	65	49	16	11.40	
LOC387646	1.039097e-05	69	48	21	7.92	
PGK1	1.198577e-05	71	49	22	15.66	
CCNE2	2.100348e-05	65	48	17	11.07	
CCDC75	2.920614e-05	74	46	28	9.47	
FGD3	3.039998e-05	69	23	46	12.30	
FAM166B	3.575856e-05	68	25	43	6.82	
MMP28	3.762361e-05	70	17	53	8.55	
ADHFE1	3.907103e-05	67	22	45	9.04	

9.5 Survival Analysis Cox Regression and dnet package

```
library(TCGAbiolinks)
# Survival Analysis SA

clinical_patient_Cancer <- clinic("brca","clinical_patient")
dataBRCAcomplete <- log2(BRCA_rnaseqv2)

tokenStop<- 1

tabSurvKMcomplete <- NULL

for( i in 1: round(nrow(dataBRCAcomplete)/100)){
message( paste( i, "of ", round(nrow(dataBRCAcomplete)/100)))
tokenStart <- tokenStop
tokenStop <-100*i
tabSurvKM<-SurvivalKMunivariate(clinical_patient_Cancer,dataBRCAcomplete,Genelist = rownames(dataBRCAcompl

tabSurvKMcomplete <- rbind(tabSurvKMcomplete,tabSurvKM)
}

tabSurvKMcomplete <- tabSurvKMcomplete[tabSurvKMcomplete$pvalue < 0.01,]
tabSurvKMcomplete <- tabSurvKMcomplete[!duplicated(tabSurvKMcomplete$mRNA),]
rownames(tabSurvKMcomplete) <-tabSurvKMcomplete$mRNA
tabSurvKMcomplete <- tabSurvKMcomplete[,-1]
tabSurvKMcomplete <- tabSurvKMcomplete[order(tabSurvKMcomplete$pvalue, decreasing=F),]

tabSurvKMcompleteDEGs <- tabSurvKMcomplete[rownames(tabSurvKMcomplete) %in% dataDEGsFiltLevel$mRNA,]

tflist <- EAGenes[EAGenes$Family == "transcription regulator","Gene"]
tabSurvKMcomplete_onlyTF <- tabSurvKMcomplete[rownames(tabSurvKMcomplete) %in% tflist,]

TabCoxNet <- SurvivalCoxNET(clinical_patient,dataBRCAcomplete,Genelist = rownames(tabSurvKMcomplete_onlyTF
```

The result is shown below:

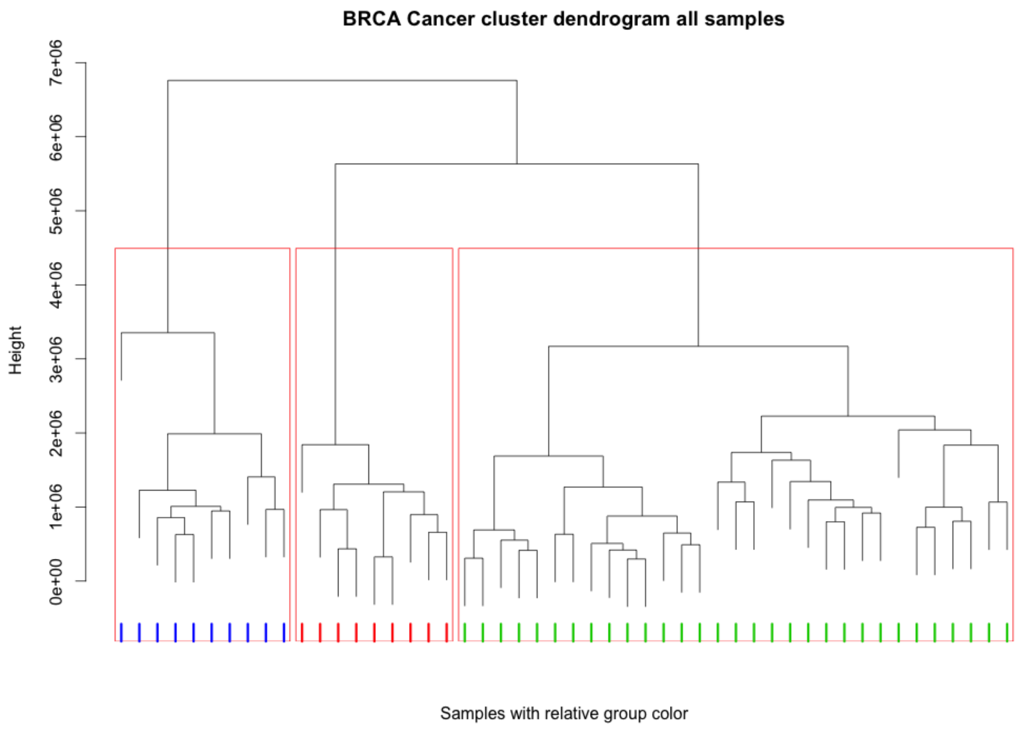


9.6 TCGA Downstream Analysis Integration

```
library(TCGAbiolinks)
library(genefilter)
library(clue)

BRCArnaseqV2 <- dataBRCA
BRCArnaseqV2MostVar <- varFilter(BRCArnaseqV2, var.func = IQR,  var.cutoff = 0.75,
                                 filterByQuantile = TRUE)

wData <- t(BRCArnaseqV2MostVar)
ddist <- dist(wData, method = "euclidean")
sHc <- hclust(ddist, method = "ward.D")

plot(sHc, labels = FALSE, main ="BRCA Cancer cluster dendrogram all samples",
   xlab = "Samples with relative group color",sub="")

rect.hclust(sHc, k=3, border="red")
tabCluster <- as.matrix(cutree(sHc, k = 3))
colnames(tabCluster)<-"Cluster"
tabCluster<-cbind(Sample = rownames(tabCluster),Color = rownames(tabCluster), tabCluster)
tabCluster<-as.data.frame(tabCluster)
tabCluster<-tabCluster[order(tabCluster$Cluster,decreasing = FALSE),]
tabCluster<-as.data.frame(tabCluster)
tabCluster$Color<-as.character(tabCluster$Color)
```

```r
ccol <- palette()[1 + 1:3]

for( cc in 1:3){
  tabCluster[tabCluster[, "Cluster"] == cc, "Color"] <- ccol[cc]
}

tabCluster <- tabCluster[sHc$labels, ]

rug(which(tabCluster[sHc$order, "Color"] == "blue"), col = "blue", lwd = 3)
rug(which(tabCluster[sHc$order, "Color"] == "green3"), col = "green3", lwd = 3)
rug(which(tabCluster[sHc$order, "Color"] == "red"), col = "red", lwd = 3)
```

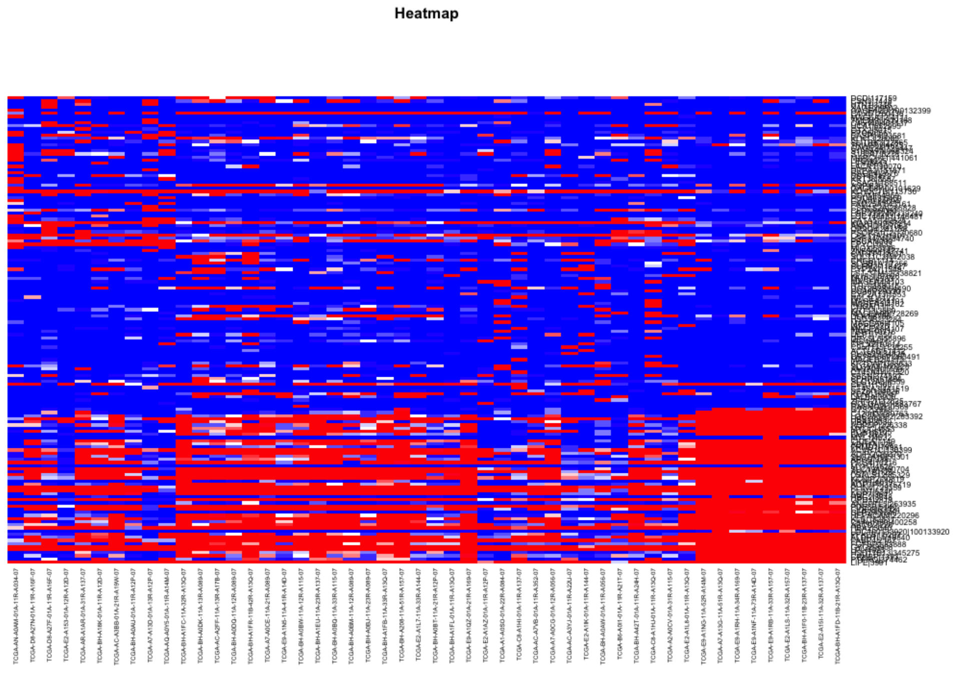
The result is shown below:



```r
library(TCGAbiolinks)

### Differential analysis
GroupBlueData <- BRCArnaseqV2[, as.character(tabCluster[tabCluster$Color == "blue", "Sample"])]
GroupGreen3Data <- BRCArnaseqV2[, as.character(tabCluster[tabCluster$Color == "green3", "Sample"])]
GroupRedData <- BRCArnaseqV2[, as.character(tabCluster[tabCluster$Color == "red", "Sample"])]

DEGsBlue <- DEArnaSEQ(cbind(GroupGreen3Data, GroupRedData),
                      GroupBlueData, "GroupOther", "GroupBlue")
DEGsGreen3 <- DEArnaSEQ(cbind(GroupBlueData, GroupRedData),
                        GroupGreen3Data, "GroupOther", "GroupGreen3")
DEGsRed <- DEArnaSEQ(cbind(GroupBlueData, GroupGreen3Data),
                     GroupRedData, "GroupOther", "GroupRed")


dataDEGs <- DEArnaSEQ(dataFilt[,samplesNT], dataFilt[,samplesTP], "Normal", "Tumor")
```

```
# DEGs filter by abs(logFC) >=1
dataDEGsFilt <- dataDEGs[abs(dataDEGs$logFC) >= 1,]

dataDEGsFiltLevel <- CreateTabLevel(dataDEGsFilt,"Tumor","Normal",
                                    [,samplesTP],dataFilt[,samplesNT])

DEGsBlueLevel <- CreateTabLevel(DEGsBlue, "GroupBlue", "GroupOther",
  GroupBlueData, cbind(GroupGreen3Data, GroupRedData), typeOrder = TRUE)
DEGsGreen3Level <- CreateTabLevel(DEGsGreen3, "GroupGreen3", "GroupOther",
  GroupGreen3Data, cbind(GroupBlueData, GroupRedData), typeOrder = TRUE)
DEGsRedLevel <- CreateTabLevel(DEGsRed, "GroupRed", "GroupOther",
  GroupRedData, cbind(GroupBlueData, GroupGreen3Data), typeOrder = TRUE)

blueDEGs <- DEGsBlueLevel[DEGsBlueLevel$FDR < 0.01 & DEGsBlueLevel$logFC >= 1, ]
blueDEGs <- blueDEGs[order(blueDEGs$FDR), ]
green3DEGs <- DEGsGreen3Level[DEGsGreen3Level$FDR < 0.01 & DEGsGreen3Level$logFC >= 1, ]
green3DEGs <- green3DEGs[order(green3DEGs$FDR), ]
redDEGs <- DEGsRedLevel[DEGsRedLevel$FDR < 0.01 & DEGsRedLevel$logFC >= 1, ]
redDEGs <- redDEGs[order(redDEGs$FDR), ]

blueDEGsSpec <- blueDEGs[setdiff(rownames(blueDEGs),
                                 union(rownames(green3DEGs), rownames(redDEGs))), ]
green3DEGsSpec <- green3DEGs[setdiff(rownames(green3DEGs),
                                     union(rownames(blueDEGs), rownames(redDEGs))), ]
redDEGsSpec <- redDEGs[setdiff(rownames(redDEGs),
                               union(rownames(blueDEGs), rownames(green3DEGs))), ]

blueDEGsSpec <- blueDEGsSpec[1:50, ]
green3DEGsSpec <- green3DEGsSpec[1:50, ]
redDEGsSpec <- redDEGsSpec[1:50, ]

tabCluster <- tabCluster[order(tabCluster$Color), ]

MfiltQuantileOrdered <- BRCArnaseqV2[c(rownames(blueDEGsSpec), rownames(green3DEGsSpec),
  rownames(redDEGsSpec)), rownames(tabCluster)]

MRactivity <- t(MfiltQuantileOrdered)

HMactivity <- MRactivity
thresholdquantile<-0.75
HMactivity[ HMactivity >= quantile(HMactivity,thresholdquantile)]<- quantile(HMactivity,thresholdquantile)

summary(as.vector(HMactivity))
quantile(HMactivity, 0.15); quantile(HMactivity, 0.85)
HMactivity[HMactivity <= quantile(HMactivity, 0.15)] <- quantile(HMactivity, 0.15)
HMactivity[HMactivity >= quantile(HMactivity, 0.85)] <- quantile(HMactivity, 0.85)

column_annotation <- matrix(" ", nrow = nrow(HMactivity), ncol = 1)
column_annotation[, 1] <- tabCluster$Color

row_annotation <- matrix(" ", nrow = 1, ncol = ncol(HMactivity))
row_annotation[1, ] <- c(rep("blue", nrow(blueDEGsSpec)),
                         rep("green3", nrow(green3DEGsSpec)),
                        rep("red", nrow(redDEGsSpec)))
library("GMD")

png("BRCA_heatmap.png", width = 1200, height = 800)
heatmap.3(t(HMactivity), ColSideColors = column_annotation, RowSideColors = row_annotation,
          key = FALSE, Colv = NA, Rowv = NA,
          scale = "none",
          #col = greenred(75),
          dendrogram = "none",
          #labRow = NA, labCol = NA,
          margins = c(1, 6), side.height.fraction = 0.25, keysize = 1.4, cexRow = 1.6)
dev.off()
```

The result is shown below:



9.7 TCGA downstream methylation analysis

Some downstream analysis from methylation data can be done with `TCGAbiolinks`. An example is shown below. Firstly, we search, download and prepare data from the HumanMethylation450 platform for the GBM tumor and also get the clinical information from the patients. In this step, we will have a SummarizedExperiment object, where the rows are the probes and the columns the samples. For more information about this object you can take a look in the documentation with the command `?SummarizedExperiment`.

```
library(TCGAbiolinks)

# Getting the data
query <- TCGAQuery(tumor = "gbm", platform = "HumanMethylation450", level = 3)
TCGADownload(query,path=".")
data <- TCGAPrepare(query = query,dir = ".",save = T)
```

```
clinical <- clinic("gbm","clinical_patient")

#Preprocessing
## We will remove probes with NA level
data <- subset(data,subset=(rowSums(is.na(assay(data)))==0))

# For the analysis we remove X and Y chromosome, because gender
# should not influentiate the analysis
# We will remove the rs probes that should not be used in the methylation analysis
idx <- !(grepl("chrX|chrY|chrNA",as.vector(seqnames(data))))
data <- subset(data,subset=idx)
```

As an example, we divided the data into groups in order to analyze the data.

```
# random split of pacients into groups
clinical$group <- c(rep("group1",nrow(clinical)/4),
                    rep("group2",nrow(clinical)/4),
                    rep("group3",nrow(clinical)/4),
                    rep("group4",nrow(clinical)-3*(floor(nrow(clinical)/4))))

 colData(data)$group <- c(rep("group1",ncol(data)/2), rep("group2",ncol(data)/2))
```

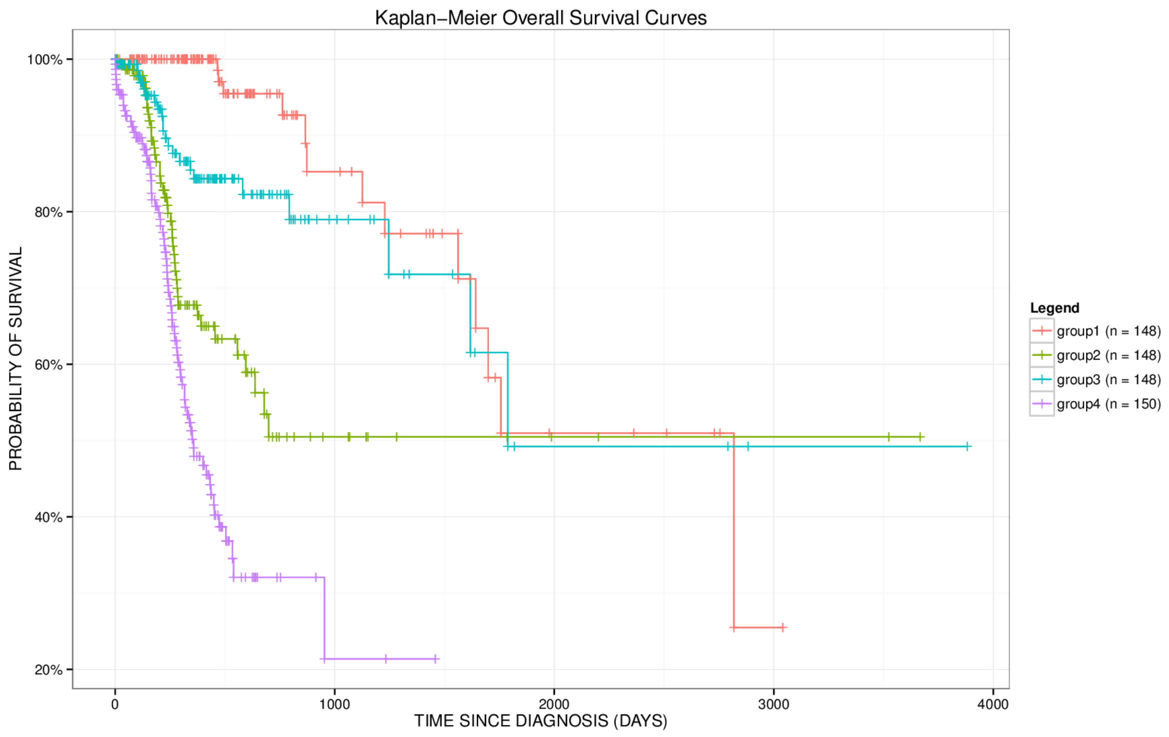
Using the clinical data, it is possible to create a survival plot with the function `survivalPlot` as follows:

```
# survival using the column group
survivalAnalysis(clinical,"group")
```

The arguments of `survivalAnalysis` are:

- **clinical_patient** TCGA Clinical patient with the information days_to_death
- **clusterCol** Column with groups to plot. This is a mandatory field, the caption will be based in this column
- **legend** Legend title of the figure
- **cutoff** xlim This parameter will be a limit in the x-axis. That means, that patients with days_to_deth > cutoff will be set to Alive.
- **main** main title of the plot
- **ylab** y-axis text of the plot
- **xlab** x-axis text of the plot
- **filename** The name of the pdf file
- **color** Define the colors of the lines.

The result is shown below:



9.8 `meanMethylationAnalysis`: Mean Methylation Analysis

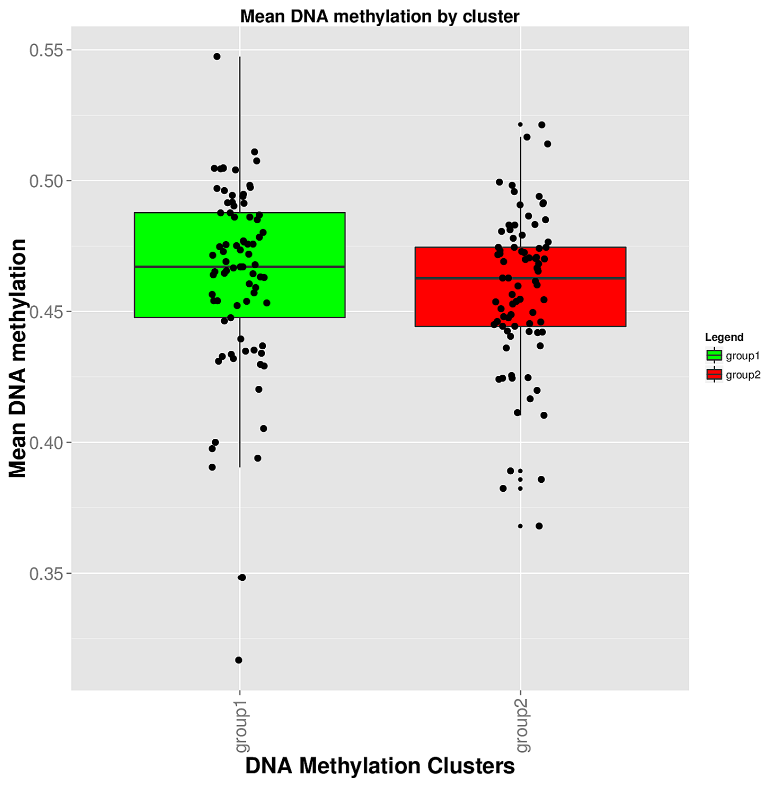
Using the data and calculating the mean methylation per group, it is possible to create a mean methylation boxplot with the function `meanMethylationAnalysis` as follows:

```
meanMethylationAnalysis(data,"group")
```

The arguments of `meanMethylationAnalysis` are:

- **data** SummarizedExperiment object obtained from `TCGAPrepare`
- **groupCol** Columns in colData(data) that defines the groups. If no columns defined a columns called "Patients" will be used
- **sort** Sort by mean methylation? False by default
- **filename** The name of the pdf that will be save
- **legend** Legend title of the figured
- **ylab** y-axis text of the plot
- **xlab** x-axis text of the plot
- **filename** The name of the pdf file
- **color** Define the colors of the lines.

The result is shown below:



9.9 `volcanoAnalysis`: Volcano plots for methylation data

Finally, we want to know which are the probes that are different methylated and are significant. For that we use a volcano plot that compares the methylation data between the two groups. Firstly, it calculates the difference between the mean methylation of each group for each probes. After, it calculates the p-value using the wilcoxon test using the Benjamini-Hochberg adjustment method. With both values, it is possible to analyse the data.

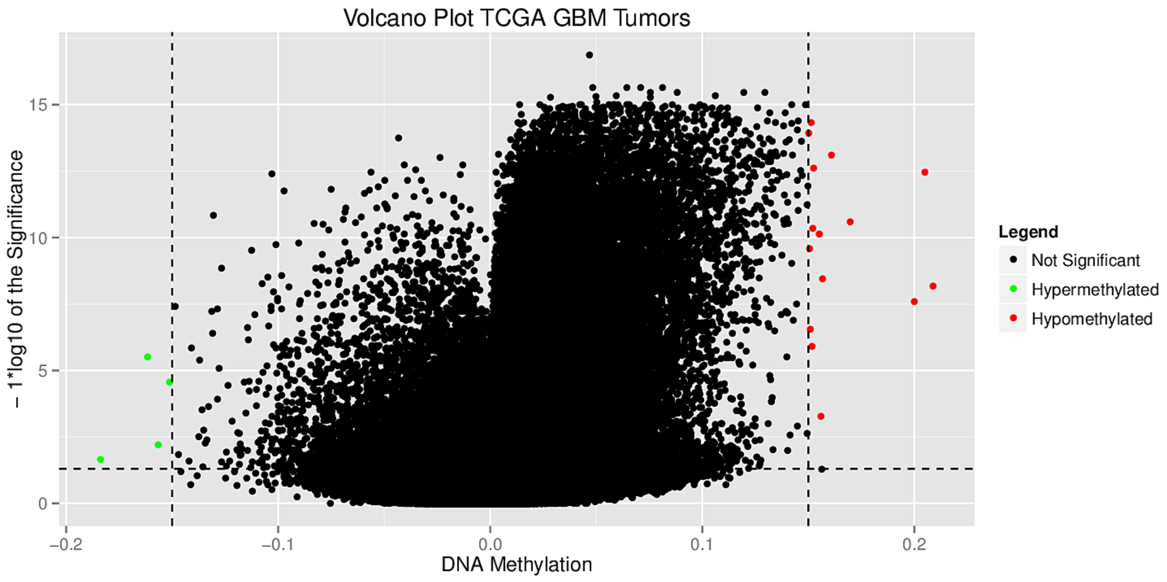
```
data <- volcanoAnalysis(data,groupCol="group")
```

The output will be an graph such as the figure below. The function will return the SummarizedExperiment with the values of p-value, p-value adjusted, diffmean and the group it belongs in the graph (1 = non significant, 2 = hypomethylated, 3 = hypermethylated). This values can be view/acessed using the `rowRanges` acessesor.

The arguments of volcanoPlot are:

- **data** SummarizedExperiment object obtained from `TCGAPrepare`
- **groupCol** Columns in colData(data) that defines the groups. If no columns defined a columns called "Patients" will be used
- **group1** In case our object has more than 2 groups, you should set the name of the group
- **group2** In case our object has more than 2 groups, you should set the name of the group
- **filename** The name of the pdf that will be save
- **legend** Legend title of the figured
- **ylab** y-axis text of the plot
- **xlab** x-axis text of the plot
- **filename** The name of the pdf file
- **color** Define the colors of the lines.
- **label** vector of labels to be used in the figure
- **xlim** x limits to cut image
- **ylim** y limits to cut image

- **p.cut** p values threshold
- **diffmean.cut** diffmean threshold



9.10 `starburstAnalysis`: Analyzing expression and methylation together

The starburst plot is proposed to combine information from two volcano plots, and is applied for a study of DNA methylation and gene expression. In order to reproduce this plot, we will use the `starburstAnalysis` function.

The parameters of this function are: * **met** SummarizedExperiment with methylation data obtained from the `TCGAPrepare` and processed by `volcanoAnalysis` function. Expected colData columns: diffmean, p.value.adj and p.value * **exp** SummarizedExperiment with methylation data obtained from the `TCGAPrepare` function and processed by `DEArnaSEQ` function. Expected colData columns: diffmean, p.value.adj and p.value

```
nrows <- 20000; ncols <- 20
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GenomicRanges::GRanges(rep(c("chr1", "chr2"), c(5000, 15000)),
                    IRanges::IRanges(floor(runif(20000, 1e5, 1e6)), width=100),
                     strand=sample(c("+", "-"), 20000, TRUE),
                     probeID=sprintf("ID%03d", 1:20000),
                     Gene_Symbol=sprintf("ID%03d", 1:20000))
colData <- S4Vectors::DataFrame(Treatment=rep(c("ChIP", "Input"), 5),
                     row.names=LETTERS[1:20],
                     group=rep(c("group1","group2"),c(10,10)))
data <- SummarizedExperiment::SummarizedExperiment(
          assays=S4Vectors::SimpleList(counts=counts),
          rowRanges=rowRanges,
          colData=colData)
met <- data
exp <- data
rowRanges(met)$diffmean <- c(runif(20000, -0.1, 0.1))
```

```r
rowRanges(met)$p.value <- c(runif(20000, 0, 1))
rowRanges(met)$p.value.adj <- c(runif(20000, 0, 1))
rowRanges(exp)$diffmean <- c(runif(20000, -0.1, 0.1))
rowRanges(exp)$p.value <- c(runif(20000, 0, 1))
rowRanges(exp)$p.value.adj <- c(runif(20000, 0, 1))
result <- starburstPlot(met,exp,p.cut = 0.01)
starburstAnalysis(met,exp)
```

10 Searching questions, answers and literature

10.1 `TCGAinvestigate`: Find most studied TFs in pubmed

Find most studied TFs in pubmed related to a specific cancer, disease, or tissue

```r
# First perform DEGs with TCGAanalyze
# See previous section
library(TCGAbiolinks)

# Select only transcription factors (TFs) from DEGs
TFs <- EAGenes[EAGenes$Family =="transcription regulator",]
TFs_inDEGs <- intersect(TFs$Gene, dataDEGsFiltLevel$mRNA )
dataDEGsFiltLevelTFs <- dataDEGsFiltLevel[TFs_inDEGs,]

# Order table DEGs TFs according to Delta decrease
dataDEGsFiltLevelTFs <- dataDEGsFiltLevelTFs[order(dataDEGsFiltLevelTFs$Delta,decreasing = TRUE),]

# Find Pubmed of TF studied related to cancer
tabDEGsTFPubmed <- TCGAinvestigate("breast", dataDEGsFiltLevelTFs, topgenes = 10)
```

The result is shown below:

Table 8: Table with
cancer

mRNA	logFC	FDR	Tumor	Normal	Delta	Pubmed	PMID
MUC1	2.46	0	38498.56	6469.40	94523.36	827	26016502; 25986064; 25982681; 25973571; 25964555; 259
FOS	-2.46	0	14080.32	66543.24	34627.41	513	26011749; 25956506; 25824986; 25788839; 25784959; 257
MDM2	1.41	0	16132.28	4959.92	22824.14	441	26042602; 26001071; 25814188; 25803170; 25744307; 257
GATA3	1.58	0	29394.60	8304.72	46410.03	180	26028330; 26008846; 25994056; 25906123; 25851711; 258
FOXA1	1.45	0	16176.96	5378.88	23465.63	167	26008846; 25995231; 25994056; 25762479; 25755696; 257
EGR1	-2.44	0	16073.08	74947.28	39275.29	77	25703326; 24980816; 24742492; 24675512; 24294184; 242
TOB1	1.43	0	17765.96	6260.08	25476.30	13	25798844; 23589165; 23162636; 21937081; 20132413; 195
MAGED1	1.18	0	20850.16	8244.32	24633.09	6	24225485; 23884293; 22935435; 21618523; 19639218; 159
PTRF	-1.72	0	15200.12	44192.52	26104.62	5	25945613; 23214712; 21913217; 20427576; 20306477
ILF2	1.27	0	22250.32	7854.44	28246.23	0	0

10.2 `TCGASocial`: Searching questions,answers and literature

The `TCGASocial` function has two type of searches, one that searches for most downloaded packages in CRAN or BioConductor and one that searches the most related question in biostar.

10.2.1 `TCGASocial` with BioConductor

Find most downloaded packages in CRAN or BioConductor

```r
library(TCGAbiolinks)

# Define a list of package to find number of downloads
listPackage <-c("limma","edgeR","survcomp")

tabPackage <- TCGAsocial(siteToFind ="bioconductor.org",listPackage)


# define a keyword to find in support.bioconductor.org returing a table with suggested packages
tabPackageKey <- TCGAsocial(siteToFind ="support.bioconductor.org" ,KeyInfo = "tcga")
```

The result is shown below:

Table 9: Table with number of downloads about a list of packages

Package	NumberDownload
limma	70621
edgeR	34070
survcomp	3813

Table 10: Find most related question in support.bioconductor.org with keyword = tcga

question	BiostarsSite	PackageSuggested
A: Calculating Ibd Using R Package	/55481/	TIN
A: Mirna Seq Blood Time Course Data Without Replicate And Paired Control	/96836/	timecourse
A: How Can I Use Geo To Understand The Regulation Of My Gene?	/14513/	SIM;TIN
A: How To Identify Rotamer States From A Pdb ?	/96579/	SIM
A: Pathway Analysis In R	/14316/	sigPathway
A: Constructing A Nj Tree From A Binary Matrix With 11 Taxa And 370.000 Characters.	/54599/	sigPathway
A: Ngs Question ~ Consensus	/17535/	sigPathway
A: Is There An R Library Similar To Libraries Like Bioperl, Biopython Or Bioruby (	/17287/	sigPathway
A: How to read .bam file in Rsamtools R package?	/97978/	Rsamtools
A: Best Practices/Softwares To Calculate Ka/Ks Ratio	/5817/#	les
A: Trouble With Local Psiblast	/79246/	les
A: R Package For Annotations Of Genomic Regions	/43313/	les
A: Question About Medip Methylation Array	/89357/	LEA;MEDIPS
A: Visualizing Genes On A Chromosome According To Their Position	/53783/	geneplotter;ggbio;Gviz
A: Rna-Seq Time Course Data	/13105/	DESeq;edge;edgeR;les;ro
A: Enrichment Analysis Without Differentially Expressed Protein List	/45212/	clusterProfiler
A: How Do I Draw A Heatmap In R With Both A Color Key And Multiple Color Side Bars?	/18211/	clusterProfiler
A: Find Out The Genes That Correspond To My Coordinates	/47826/	ChIPpeakAnno
A: Peaks And Nearby Genes	/8675/#	biomaRt;ChIPpeakAnno
Mirna Sequence Using Biomart R Package	/96700/	biomaRt
A: What Is Your Favorite Visualizer For Acgh Or Snp Microarray Data?	/988/#9	beadarray;beadarraySNP
A: Annotating Expression Profile Data	/60694/	AnnotationDbi;LEA
A: How To Calculate 95% Ci In Genotypes And In Alleles By Using Hardy–Weinberg Test	/46269/	AnnotationDbi;LEA
A: Is There Any R Or R / Bioconductor Package That Can Make Circular Plots Like Per	/17728/	AnnotationDbi;LEA
A: To Calculate The Energy From Secondary Structure Dot Bracket Notation Of Rna	/16394/	AnnotationDbi;LEA

question	BiostarsSite	PackageSuggested
A: How Can I Identify Orthologous Contigs Between Two De Novo Transcriptome Assembl	/15073/	AnnotationDbi;LEA
A: How To Download Dataset For The Microarray Data Analysis From Ncbi For Affymetri	/81867/	affy;canceR;HELP;LEA;l
A: How to generate a Venn diagram	/102393	0
A: How to Normalize the Microarray Data Obtained from ncbi?	/141595	0
A: CNV calling for illumina 550k array	/108029	0
A: Error: could not find function "heatmap.2"	/106843	0
A: Extracting Probeset IDs from .CELfiles	/135942	0
A: Is there a way to access the data stored in a .ab1 file ?	/122709	0
A: Bam to nucleotide frequencies	/109798	0
A: How can I programmatically download the GEO DataSets of a given accession?	/113070	0
A: Gene Regulatory Network using micro array data	/121070	0
A: R programming question: insert alternately	/139129	0
A: Ignoring N.s on Each Side of the Chromosome	/146513	0
* MISSING ***	NA	0
A: r3Cseq rat genome	/135732	0

10.2.2 `TCGASocial` with Biostar

Find most related question in biostar.

```r
library(TCGAbiolinks)

# Find most related question in biostar with TCGA
tabPackage1 <- TCGAsocial(siteToFind ="biostars.org",KeyInfo = "TCGA")

# Find most related question in biostar with package
tabPackage2 <- TCGAsocial(siteToFind ="biostars.org",KeyInfo = "package")
```

The result is shown below:

Table 11: Find most related question in biostar with TCGA

question	BiostarsSite	PackageSuggested
A: Question About Tcga Snp-Array Data	/88541/	LEA;PROcess;ROC
A: Cnv Data	/95763/	DNAcopy;HELP
A: Cnv Data	/95763/	DNAcopy;HELP
A: Where To Find Test Datasets For Data Classification Problems	/60664/	convert;GEOquery;LEA;rMAT;roar;SIM
A: How to get public cancer RNA-seq data?	/137370	0
A: Microarray And Epigenomic Data For Same Cancer Cell Line?	/95724/	0

Table 12: Find most related question in biostar with package

question	BiostarsSite	PackageSuggested
A: Calculating Ibd Using R Package	/55481/	TIN
A: Mirna Seq Blood Time Course Data Without Replicate And Paired Control	/96836/	timecourse
A: How Can I Use Geo To Understand The Regulation Of My Gene?	/14513/	SIM;TIN
A: How To Identify Rotamer States From A Pdb ?	/96579/	SIM
A: Pathway Analysis In R	/14316/	sigPathway
A: Constructing A Nj Tree From A Binary Matrix With 11 Taxa And 370.000 Characters.	/54599/	sigPathway
A: Ngs Question ~ Consensus	/17535/	sigPathway
A: Is There An R Library Similar To Libraries Like Bioperl, Biopython Or Bioruby (	/17287/	sigPathway

question	BiostarsSite	PackageSuggested
A: How to read .bam file in Rsamtools R package?	/97978/	Rsamtools
A: Best Practices/Softwares To Calculate Ka/Ks Ratio	/5817/#	les
A: Trouble With Local Psiblast	/79246/	les
A: R Package For Annotations Of Genomic Regions	/43313/	les
A: Question About Medip Methylation Array	/89357/	LEA;MEDIPS
A: Visualizing Genes On A Chromosome According To Their Position	/53783/	geneplotter;ggbio;Gviz
A: Rna-Seq Time Course Data	/13105/	DESeq;edge;edgeR;les;r
A: Enrichment Analysis Without Differentially Expressed Protein List	/45212/	clusterProfiler
A: How Do I Draw A Heatmap In R With Both A Color Key And Multiple Color Side Bars?	/18211/	clusterProfiler
A: Find Out The Genes That Correspond To My Coordinates	/47826/	ChIPpeakAnno
A: Peaks And Nearby Genes	/8675/#	biomaRt;ChIPpeakAnno
Mirna Sequence Using Biomart R Package	/96700/	biomaRt
A: What Is Your Favorite Visualizer For Acgh Or Snp Microarray Data?	/988/#9	beadarray;beadarraySNP
A: Annotating Expression Profile Data	/60694/	AnnotationDbi;LEA
A: How To Calculate 95% Ci In Genotypes And In Alleles By Using Hardy–Weinberg Test	/46269/	AnnotationDbi;LEA
A: Is There Any R Or R / Bioconductor Package That Can Make Circular Plots Like Per	/17728/	AnnotationDbi;LEA
A: To Calculate The Energy From Secondary Structure Dot Bracket Notation Of Rna	/16394/	AnnotationDbi;LEA
A: How Can I Identify Orthologous Contigs Between Two De Novo Transcriptome Assembl	/15073/	AnnotationDbi;LEA
A: How To Download Dataset For The Microarray Data Analysis From Ncbi For Affymetri	/81867/	affy;canceR;HELP;LEA;
A: How to generate a Venn diagram	/102393	0
A: How to Normalize the Microarray Data Obtained from ncbi?	/141595	0
A: CNV calling for illumina 550k array	/108029	0
A: Error: could not find function "heatmap.2"	/106843	0
A: Extracting Probeset IDs from .CELfiles	/135942	0
A: Is there a way to access the data stored in a .ab1 file ?	/122709	0
A: Bam to nucleotide frequencies	/109798	0
A: How can I programmatically download the GEO DataSets of a given accession?	/113070	0
A: Gene Regulatory Network using micro array data	/121070	0
A: R programming question: insert alternately	/139129	0
A: Ignoring N.s on Each Side of the Chromosome	/146513	0
* MISSING ***	NA	0
A: r3Cseq rat genome	/135732	0

10.2.3 Session Information

```
sessionInfo()
```

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-redhat-linux-gnu (64-bit)
## Running under: Fedora 22 (Twenty Two)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=pt_BR.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] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] TCGAbiolinks_0.99.1        png_0.1-7
##  [3] SummarizedExperiment_0.3.2 Biobase_2.29.1
##  [5] GenomicRanges_1.21.16      GenomeInfoDb_1.5.8
##  [7] IRanges_2.3.14             S4Vectors_0.7.10
##  [9] BiocGenerics_0.15.3        BiocStyle_1.7.4
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.2-6
##  [2] rjson_0.2.15
##  [3] hwriter_1.3.2
##  [4] futile.logger_1.4.1
##  [5] XVector_0.9.1
##  [6] rstudioapi_0.3.1
##  [7] roxygen2_4.1.1
##  [8] hexbin_1.27.0
##  [9] AnnotationDbi_1.31.17
## [10] xml2_0.1.1
## [11] splines_3.2.1
## [12] R.methodsS3_1.7.0
## [13] DESeq_1.21.0
## [14] geneplotter_1.47.0
## [15] knitr_1.10.5
## [16] Rsamtools_1.21.13
## [17] annotate_1.47.1
## [18] R.oo_1.19.0
## [19] supraHex_1.7.1
## [20] graph_1.47.2
## [21] httr_1.0.0
## [22] Matrix_1.2-2
## [23] TxDb.Hsapiens.UCSC.hg19.knownGene_3.1.3
## [24] limma_3.25.13
## [25] formatR_1.2
## [26] htmltools_0.2.6
## [27] tools_3.2.1
## [28] igraph_1.0.1
## [29] gtable_0.1.2
## [30] reshape2_1.4.1
## [31] ShortRead_1.27.5
## [32] Rcpp_0.11.6
## [33] Biostrings_2.37.2
## [34] ape_3.3
## [35] nlme_3.1-121
## [36] rtracklayer_1.29.12
## [37] exactRankTests_0.8-28
## [38] stringr_1.0.0
## [39] proto_0.3-10
## [40] rvest_0.2.0
## [41] devtools_1.8.0
## [42] XML_3.98-1.3
## [43] edgeR_3.11.2
## [44] zlibbioc_1.15.0
## [45] MASS_7.3-42
## [46] scales_0.2.5
## [47] aroma.light_2.5.2
## [48] rversions_1.0.2
## [49] lambda.r_1.1.7
## [50] RColorBrewer_1.1-2
## [51] yaml_2.1.13
## [52] curl_0.9.1
## [53] memoise_0.2.1
## [54] ggplot2_1.0.1
## [55] downloader_0.4
## [56] biomaRt_2.25.1
## [57] reshape_0.8.5
## [58] latticeExtra_0.6-26
## [59] stringi_0.5-5
## [60] RSQLite_1.0.0
## [61] highr_0.5
## [62] genefilter_1.51.0
## [63] GenomicFeatures_1.21.13
## [64] BiocParallel_1.3.34
## [65] chron_2.3-47
## [66] matrixStats_0.14.2
## [67] bitops_1.0-6
## [68] dnet_1.0.6
## [69] evaluate_0.7
## [70] lattice_0.20-31
## [71] GenomicAlignments_1.5.11
## [72] GGally_0.5.0
## [73] plyr_1.8.3
## [74] magrittr_1.5
## [75] R6_2.1.0
## [76] DBI_0.3.1
## [77] survival_2.38-3
## [78] RCurl_1.95-4.7
## [79] EDASeq_2.3.2
## [80] futile.options_1.0.0
## [81] rmarkdown_0.7
## [82] data.table_1.9.4
## [83] git2r_0.10.1
## [84] Rgraphviz_2.13.0
## [85] digest_0.6.8
## [86] xtable_1.7-4
## [87] R.utils_2.1.0
## [88] munsell_0.4.2
```