Working with TCGAbiolinks package

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

2015-08-11

Contents

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.

`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
- **level** Options: 1,2,3,"mage-tab"
- **center**
- **version** List of Platform/Tumor/Version to be changed

`TCGAquery`: some filtering examples

`TCGAquery`: 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.
## ========================================================
```

TCGAquery: 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")
```

TCGAquery: 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.
## ======================================================
```

TCGAquery: 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.
## ======================================================
```

TCGAquery: 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")
```

TCGAquery_version: Retrieve versions of the data in TCGA

Query version for a specific platform for example IlluminaHiSeq_RNASeqV2

```
library(TCGAbiolinks)

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

The result is shown below:

Table 1: 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

TCGAquery: Searching old versions

The results from TCGAquery are always the last one from the TCGA data portal. As we have a preprocessed table you should always update TCGAbiolinks package. We intent to update the database constantly.

In case you want an old version of the files we have the version parameter that should be a list of triple values(platform,tumor,version). For example the code below will get the LGG and GBM tumor for platform HumanMethylation450 but for the LGG/HumanMethylation450, we want the version 5 of the files instead of the latest. This could take some seconds.

```
query <- TCGAquery(tumor = c("LGG","GBM"),platform = c("HumanMethylation450"),
                     level = 3,
                      version = list(c("HumanMethylation450","LGG",1)))
```

TCGAquery_clinic & TCGAquery_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.

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

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

The parameters of TCGAquery_clinicFilt are:

- **barcode** List of barcodes
- **clinical_patient_data** clinical patient data obtained with clinic function Ex: clinical_patient_data <- TCGAquery_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"

```
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-A0-A0J5-05A-11R-1789-07", "TCGA-BH-A0B4-11A-12R-1789-07",
         "TCGA-B6-A1KN-60A-13R-1789-07", "TCGA-A0-A0J5-01A-11R-1789-07",
         "TCGA-A0-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 <- TCGAquery_SampleTypes(bar,"TP")
S2 <- TCGAquery_SampleTypes(bar,"NB")

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

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

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

The result is shown below:

```
## ER Positive Samples:
##
##
## HER Positive Samples:
##
##
## 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
## character(0)
```

TCGAquery_subtypes: Working with molecular subtypes data.

```
# Check with subtypes from TCGAprepare and update examples
require(xlsx)

GBM_path_subtypes <- TCGAquery_subtypes(tumor = "gbm",path ="../dataGBM")

LGG_path_subtypes <- TCGAquery_subtypes(tumor = "lgg",path ="../dataLGG")

LGG_clinic <- TCGAquery_clinic(cancer = "LGG",clinical_data_type = "clinical_patient")
# table(LGG_clinic$ldh1_mutation_found)
```

TCGAquery_integrate: 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 TCGAquery_integrate 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 <- TCGAquery_integrate(query)
```

The result of the 3 platforms of TCGAquery_integrate result is shown below:

Table 2: Table common samples among platforms from TCGAquery

	AgilentG4502A_07_3	HumanMethylation450	IlluminaHiSeq_RNASeqV2
AgilentG4502A_07_3	604	0	530
HumanMethylation450	0	0	0
IlluminaHiSeq_RNASeqV2	530	0	1218

TCGAquery: some examples

Some search examples are shown below:

```
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")
```

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
- **force** Download again if file already exists? Default: FALSE

TCGAdownload: 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

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*

TCGAdownload: Table of types available for downloading

- **RNASeqV2:** junction_quantification,rsem.genes.results, rsem.isoforms.results, rsem.genes.normalized_results, rsem.isoforms.normalized_results, bt.exon_quantification
- **RNASeq:** exon.quantification,spljxn.quantification, gene.quantification
- **genome_wide_snp_6:** hg18.seg,hg19.seg,nocnv_hg18.seg,nocnv_hg19.seg

TCGAprepare: Preparing the data

You can easily read the downloaded data using the TCGAprepare function. This function will prepare the data into a SummarizedExperiment (Huber, Wolfgang and Carey, Vincent J and Gentleman, Robert and Anders, Simon and Carlson, Marc and Carvalho, Benilton S and Bravo, Hector Corrada and Davis, Sean and Gatto, Laurent and Girke, Thomas and others 2015) 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.
- **samples** List of samples 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`
- **reannotate** Reannotate genes? Source http://grch37.ensembl.org/. Default: `FALSE`. (For the moment only working for methylation data)

In order to add useful information to reasearches we added in the colData of the summarizedExperiment the subtypes classification for the LGG and GBM samples that can be found in the TCGA publication section We intend to add more tumor types in the future.

Also in the metadata of the objet we added the parameters used in TCGAprepare, the query matrix used for preparing, and file information (name,creation time and modification time) in order to help the user know which samples, versions, and parameters they used.

TCGAprepare: 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.

```
library(TCGAbiolinks)
library(SummarizedExperiment)
head(assay(dataREAD,"normalized_count"))
```

```
##              TCGA-DY-A1DE-01A-11R-A155-07 TCGA-DY-A0XA-01A-11R-A155-07
## A1BG|1                            13.6732                      13.0232
## A1CF|29974                        53.4379                     140.5455
## A2M|2                           5030.4792                    1461.9358
## A2ML1|144568                       0.0000                      18.2001
## A4GALT|53947                     170.1189                      89.9895
## A4GNT|51146                        0.9805                       0.0000
```

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.

```
library(TCGAbiolinks)
class(dataREAD_df)
```

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

```
dim(dataREAD_df)
```

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

```
head(dataREAD_df)
```

```
##              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
```

TCGAprepare: Table of types available for the TCGAprepare

- **RNASeqV2:** junction_quantification,rsem.genes.results, rsem.isoforms.results, rsem.genes.normalized_results, rsem.isoforms.normalized_results, bt.exon_quantification
- **RNASeq:** exon.quantification,spljxn.quantification, gene.quantification
- **genome_wide_snp_6:** hg18.seg,hg19.seg,nocnv_hg18.seg,nocnv_hg19.seg

TCGAprepare: Preparing the data with parameter - 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 DNA 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 DNA 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")
# This function will take a lot of time depens on internet connection
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)

########## gene 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)
```

TCGAprepare: Preparing the data with CNV data (Genome_Wide_SNP_6)

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

```
# 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")

# 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")
```

TCGAanalyze: Analyze data from TCGA.

You can easily analyze data using following functions:

TCGAanalyze_Preprocessing Preprocessing of Gene Expression data (IlluminaHiSeq_RNASeqV2).

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

```
# 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 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)
```

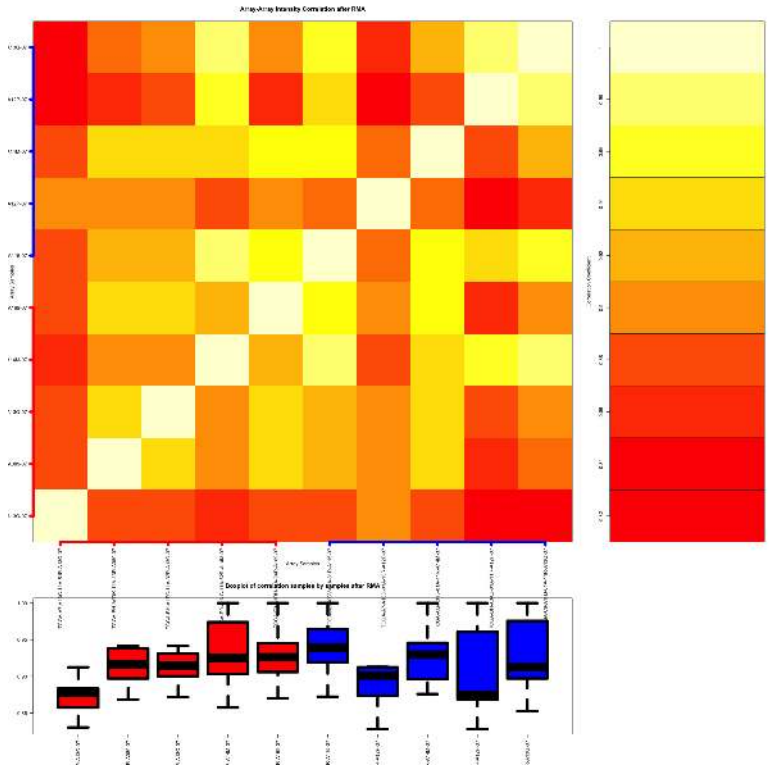
```
# Prepare expression matrix with gene id in rows and samples (barcode) in columns
# rsem.genes.results as values
BRCARnaseq_assay <- TCGAprepare(query,"../dataBrca",type = "rsem.genes.results")

BRCAMatrix <- assay(BRCARnaseq_assay,"raw_counts")

# For gene expression if you need to see a boxplot correlation and AAIC plot
# to define outliers you can run

BRCARnaseq_CorOutliers <- TCGAanalyze_Preprocessing(BRCARnaseq_assay)
```

The result is shown below:

Table 3: Example of a m
7 samples in columns)

	TCGA-A2-A0CV-01A-31R-A115-07	TCGA-C8-A1HJ-01A-11R-A13Q-07	TCGA-BH-A1FC-11A-32R-A13Q-0
ANXA2P3|305	8.00	4.00	11.0
NECAP1|25977	2439.00	4012.00	1158.0
A2M|2	64314.72	386443.55	67114.8
CDRT15P|94158	17.00	3.00	5.0
RASL11B|65997	179.00	103.00	91.0
OR1C1|26188	0.00	0.00	0.0
PDE1B|5153	305.00	400.00	667.0
HBEGF|1839	2118.00	1318.00	698.0
MAPK7|5598	870.00	768.01	721.0
SCARNA21|677763	0.00	0.00	0.0

The result from TCGAanalyze_Preprocessing is shown below:



TCGAanalyze_DEA & TCGAanalyze_LevelTab Differential expression analysis (DEA)

Perform DEA (Differential expression analysis) to identify differentially expressed genes (DEGs) using the TCGAanalyze_DEA function.

TCGAanalyze_DEA performs DEA using following functions from R *edgeR*:

1. edgeR::DGEList converts the count matrix into an edgeR object.
2. edgeR::estimateCommonDisp each gene gets assigned the same dispersion estimate.
3. edgeR::exactTest performs pair-wise tests for differential expression between two groups.
4. edgeR::topTags takes the output from exactTest(), adjusts the raw p-values using the False Discovery Rate (FDR) correction, and returns the top differentially expressed genes.

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 TCGAanalyze_LevelTab 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 <- TCGAanalyze_DEA(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 <- TCGAanalyze_LevelTab(dataDEGsFilt,"Tumor","Normal",
                                dataFilt[,samplesTP],dataFilt[,samplesNT])
```

The result is shown below:

Table 4: 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

TCGAanalyze_EAcomplete & TCGAvisualize_EAbarplot: Enrichment Analysis

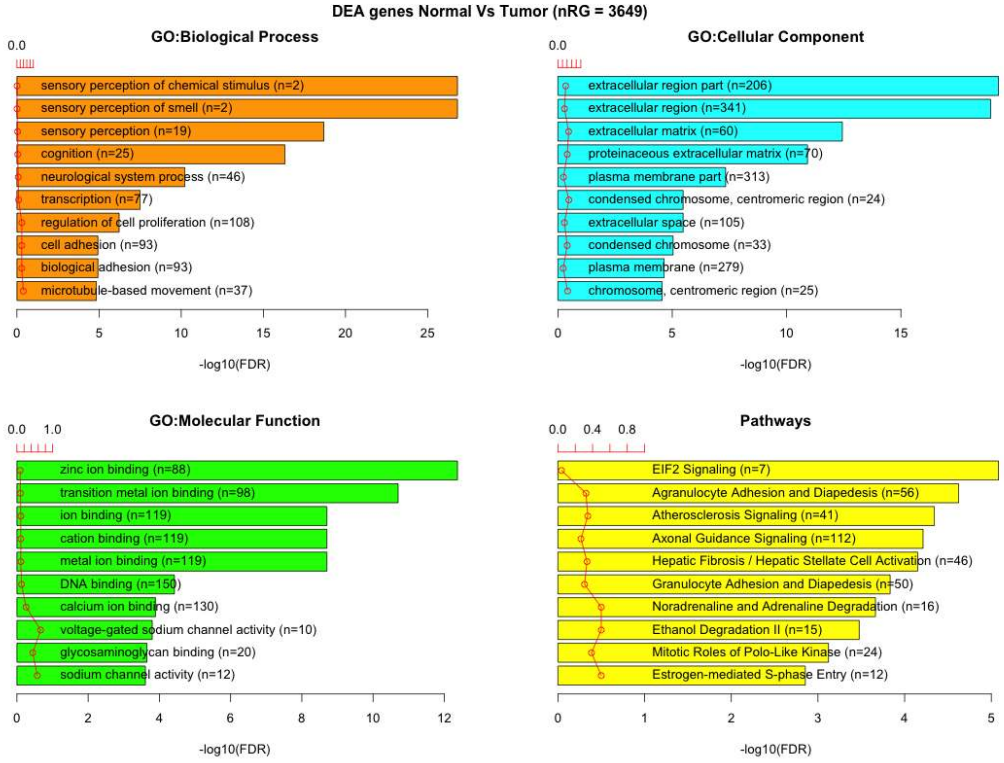
Researchers, in order to better understand the underlying biological processes, often want to retrieve a functional profile of a set of genes that might have an important role. This can be done by performing an enrichment analysis.

We will perform an enrichment analysis on gene sets using the `TCGAanalyze_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 `TCGAvisualize_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 <- TCGAanalyze_EAcomplete(TFname="DEA genes Normal Vs Tumor",Genelist))

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

TCGAvisualize_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:



TCGAanalyze_survival Survival Analysis: Cox Regression and dnet package

When analyzing survival times, different problems come up than the ones dis- cussed so far. One question is how do we deal with subjects dropping out of a study. For example, assume that we test a new cancer drug. While some subjects die, others may believe that the new drug is not effective, and decide to drop out of the study before the study is finished. A similar problem would be faced when we investigate how long a machine lasts before it breaks down.

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

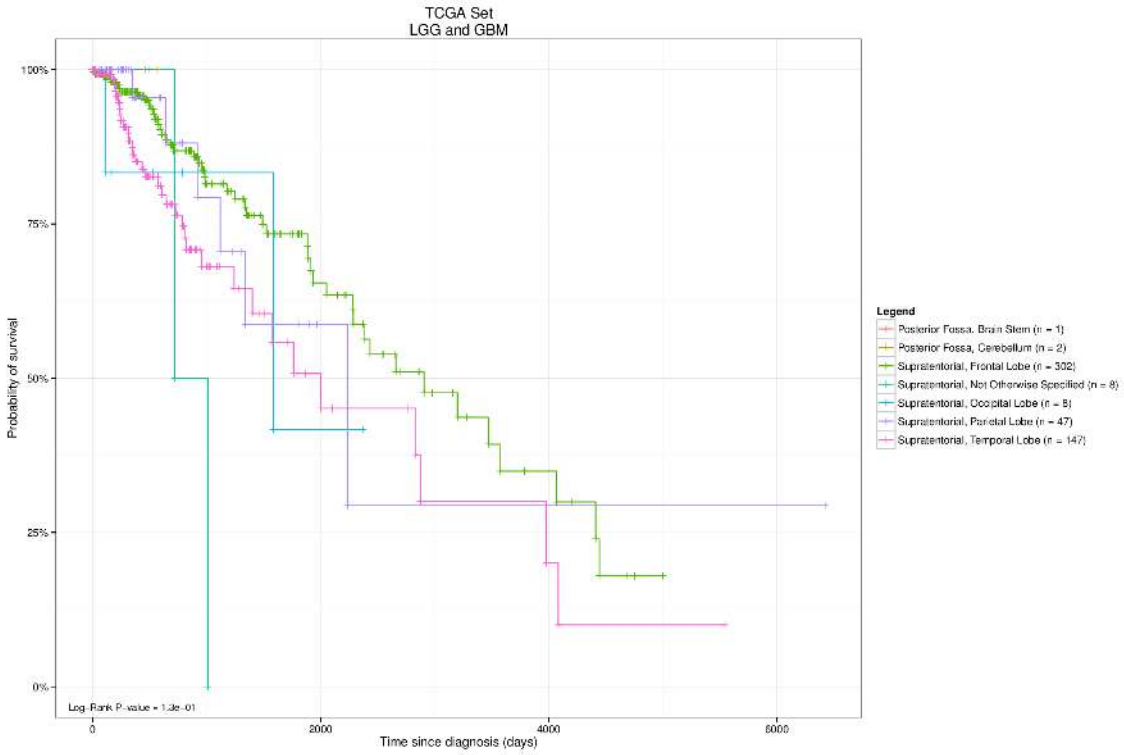
```
clin.gbm <- TCGAquery_clinic("gbm", "clinical_patient")
clin.lgg <- TCGAquery_clinic("lgg", "clinical_patient")

TCGAanalyze_survival(plyr::rbind.fill(clin.lgg,clin.gbm),
                     "radiation_therapy",
                     main = "TCGA Set\nLGG and GBM",height = 10, width=10)
```

The arguments of TCGAanalyze_survival 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:

```
## Warning in readPNG("survival.png"): libpng warning: iCCP: profile 'icc':
## 0h: PCS illuminant is not D50
```




```
library(TCGAbiolinks)
# Survival Analysis SA

clinical_patient_Cancer <- TCGAquery_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<-TCGAanalyze_SurvivalKM(clinical_patient_Cancer,dataBRCAcomplete,
                                  Genelist = rownames(dataBRCAcomplete)[tokenStart:tokenStop],
                                          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 5: 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	

`TCGAanalyze_DMR`: Differentially methylated regions Analysis

We will search for differentially methylated CpG sites using the `TCGAanalyze_DMR` function. In order to find these regions we use the beta-values (methylation values ranging from 0.0 to 1.0) to compare two groups.

Firstly, it calculates the difference between the mean DNA methylation of each group for each probes.

Secondly, it calculates the p-value using the wilcoxon test adjusting by the Benjamini-Hochberg method. The default parameters was set to require a minimum absolute beta-values difference of 0.2 and a p-value adjusted of < 0.01.

After these analysis, we save a volcano plot (x-axis:diff mean methylation, y-axis: significance) that will help the user identify the differentially methylated CpG sites and return the object with the calculus in the rowRanges.

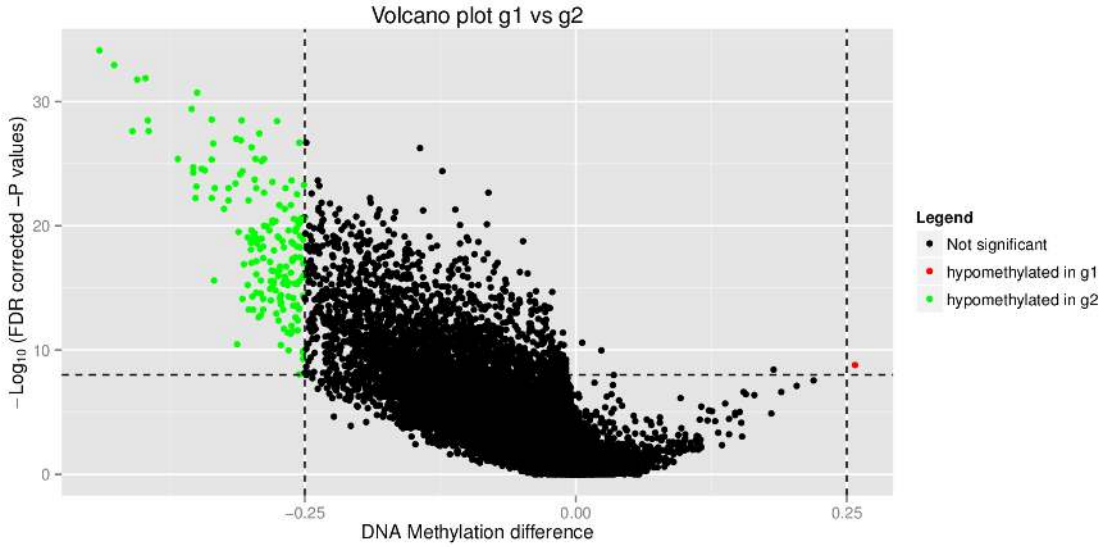
The arguments of volcanoPlot are:

- **data** SummarizedExperiment obtained from the TCGAPrepare
- **groupCol** Columns with the groups inside the SummarizedExperiment object. (This will be obtained by the function colData(data))
- **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** pdf filename. Default: volcano.pdf
- **legend** Legend title
- **color** vector of colors to be used in graph
- **title** main title. If not specified it will be "Volcano plot (group1 vs group2)
- **ylab** y axis text
- **xlab** x axis text
- **xlim** x limits to cut image
- **ylim** y limits to cut image
- **label** vector of labels to be used in the figure. Example: c("1" = "Not Significant", "2" = "Hypermethylated in group1", "3" = "Hypomethylated in group1"))
- **p.cut** p values threshold. *Default: 0.01*
- **diffmean.cut** diffmean threshold. *Default: 0.2*
- **adj.method** Adjusted method for the p-value calculation
- **paired** Wilcoxon paired parameter. *Default: FALSE*
- **overwrite** Overwrite the pvalues and diffmean values if already in the object for both groups? *Default: FALSE*

```
data <- TCGAanalyze_DMR(data, groupCol = "cluster.meth",subgroupCol = "disease",
                            group.legend  = "Groups", subgroup.legend = "Tumor",
                            print.pvalue = TRUE)
```

The output will be a plot such as the figure below. The green dots are the probes that are hypomethylated in group 2 compared to group 1, while the red dots are the hypermethylated probes in group 2 compared to group 1

```
## Warning in readPNG("volcano.png"): libpng warning: iCCP: profile 'icc': 0h:
## PCS illuminant is not D50
```



Also, the `TCGAanalyze_DMR` function will save the plot as pdf and return the same SummarizedExperiment that was given as input with the values of p-value, p-value adjusted, diffmean and the group it belongs in the graph (non significant, hypomethylated, hypermethylated) in the rowRanges. The collumns will be (where group1 and group2 are the names of the groups):

- diffmean.group1.group2 (mean.group2 - mean.group1)
- diffmean.group2.group1 (mean.group1 - mean.group2)
- p.value.group1.group2
- p.value.adj.group1.group2
- status.group1.group2 (Status of probes in group2 in relation to group1)
- status.group2.group1 (Status of probes in group1 in relation to group2)

This values can be view/acessed using the `rowRanges` acessesor (`rowRanges(data)`).

Observation: Calling the same function again, with the same arguments will only plot the results, as it was already calculated. With you want to have them recalculated, please set `overwrite` to `TRUE` or remove the calculated collumns.

`TCGAvisualize`: Visualize results from analysis functions with TCGA's data.

You can easily visualize results from soome following functions:

TCGAvisualize_PCA: Principal Component Analysis plot for differentially expressed genes

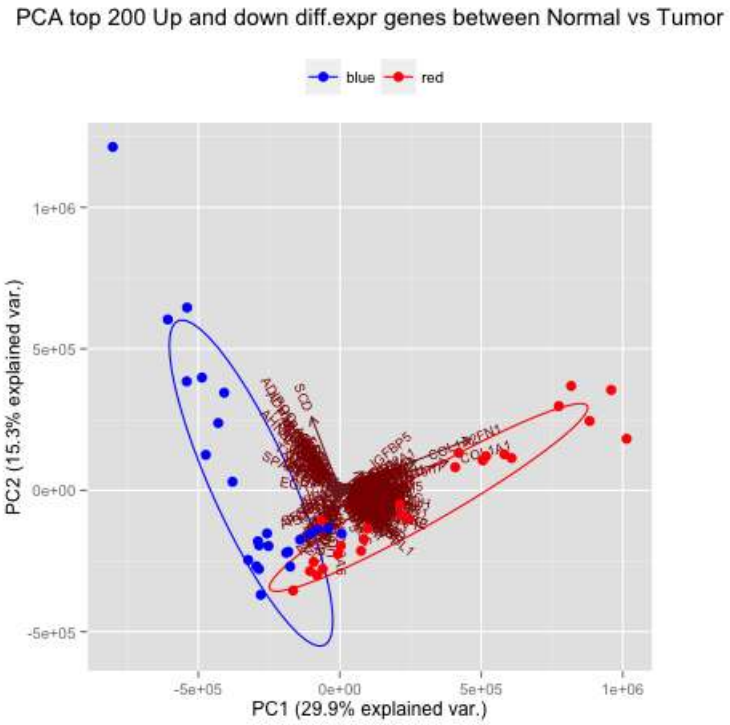
In order to understand better our genes, we can perform a PCA to reduce the number of dimensions of our gene set. The function TCGAvisualize_PCA will plot the PCA for different groups.

The parameters of this function are:

- **dataFilt** The expression matrix after normalization and quantile filter
- **dataDEGsFiltLevel** The TCGAanalyze_LevelTab output
- **ntopgenes** number of DEGs genes to plot in PCA

```
library(TCGAbiolinks)

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

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

# Principal Component Analysis plot for ntop selected DEGs
TCGAvisualize_PCA(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:



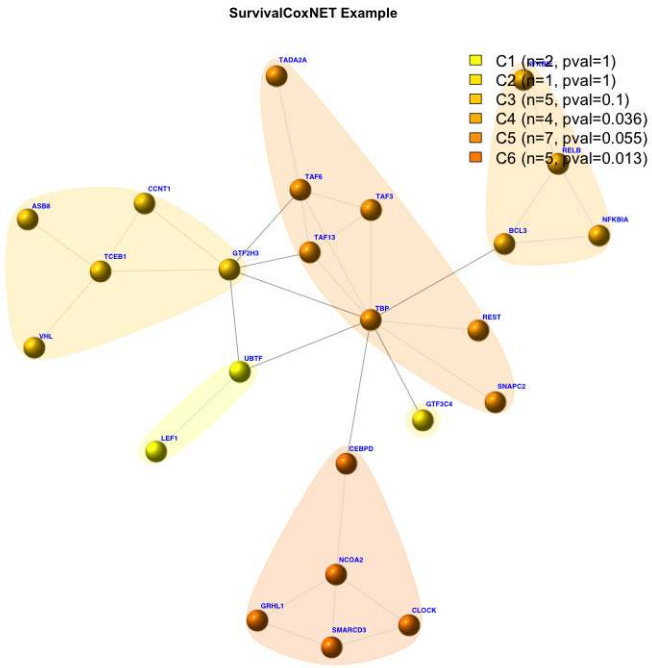
TCGAvisualize_SurvivalCoxNET Survival Analysis: Cox Regression and dnet package

TCGAvisualize_SurvivalCoxNET can help an user to identify a group of survival genes that are significant from univariate Kaplan Meier Analysis and also for Cox Regression. It shows in the end a network build with community of genes with similar range of pvalues from Cox regression (same color) and that interaction among those genes is already validated in literatures using the STRING database (version 9.1).

```
library(TCGAbiolinks)
# Survival Analysis SA

clinical_patient_Cancer <- TCGAquery_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<-TCGAanalyze_SurvivalKM(clinical_patient_Cancer,
                                  dataBRCAcomplete,
                                  Genelist = rownames(dataBRCAcomplete)[tokenStart:tokenStop],
                                  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,]

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

TabCoxNet <- TCGAvisualize_SurvivalCoxNET(clinical_patient_Cancer,dataBRCAcomplete,
                             Genelist = rownames(tabSurvKMcomplete_onlyTF),
                             scoreConfidence = 700,titlePlot = "TCGAvisualize_SurvivalCoxNET Example")
```

In particular the survival analysis with kaplan meier and cox regression allow user to reduce the feature / number of genes significant for survival. And using 'dnet' pipeline with 'TCGAvisualize_SurvivalCoxNET' function the user can further filter those genes according some already validated interaction according STRING database. This is important because the user can have an idea about the biology inside the survival discrimination and further investigate in a sub-group of genes that are working in as synergistic effect influencing the risk of survival. In the following picture the user can see some community of genes with same color and survival pvalues.

The result is shown below:



`TCGAvisualize_meanMethylation`: Sample Mean DNA Methylation Analysis

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

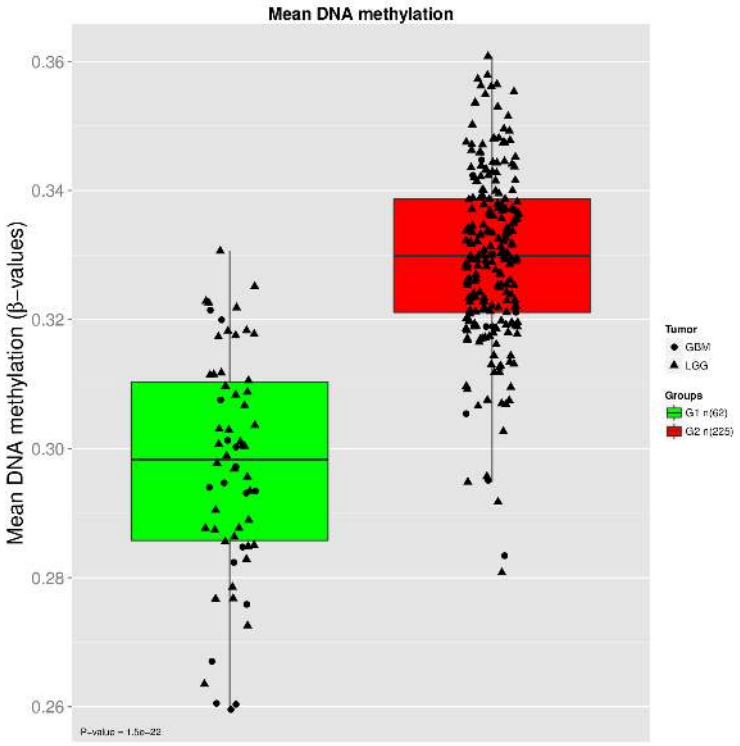
```
TCGAvisualize_meanMethylation(data,"group")
```

The arguments of `TCGAvisualize_meanMethylation` 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
- **subgroupCol** Columns in colData(data) that defines the subgroups.
- **shapes** Shape vector of the subgroups. It must have the size of the levels of the subgroups. Example: shapes = c(21,23) if for two levels
- **filename** The name of the pdf that will be saved
- **subgroup.legend** Name of the subgroup legend. **DEFAULT: subgroupCol**
- **group.legend** Name of the group legend. **DEFAULT: groupCol**
- **color** vector of colors to be used in graph
- **title** main title in the plot
- **ylab** y axis text in the plot
- **print.pvalue** Print p-value for two groups in the plot
- **xlab** x axis text in the plot
- **labels** Labels of the groups

The result is shown below:

```
## Warning in readPNG("meanmet.png"): libpng warning: iCCP: profile 'icc': 0h:
## PCS illuminant is not D50
```



`TCGAvisualize_starburst`: 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 `TCGAvisualize_starburst` function.

The function creates Starburst plot for comparison of DNA methylation and gene expression. The log10 (FDR-corrected P value) is plotted for beta value for DNA methylation (x axis) and gene expression (y axis) for each gene. The black dashed line shows the FDR-adjusted P value of 0.01.

The parameters of this function are:

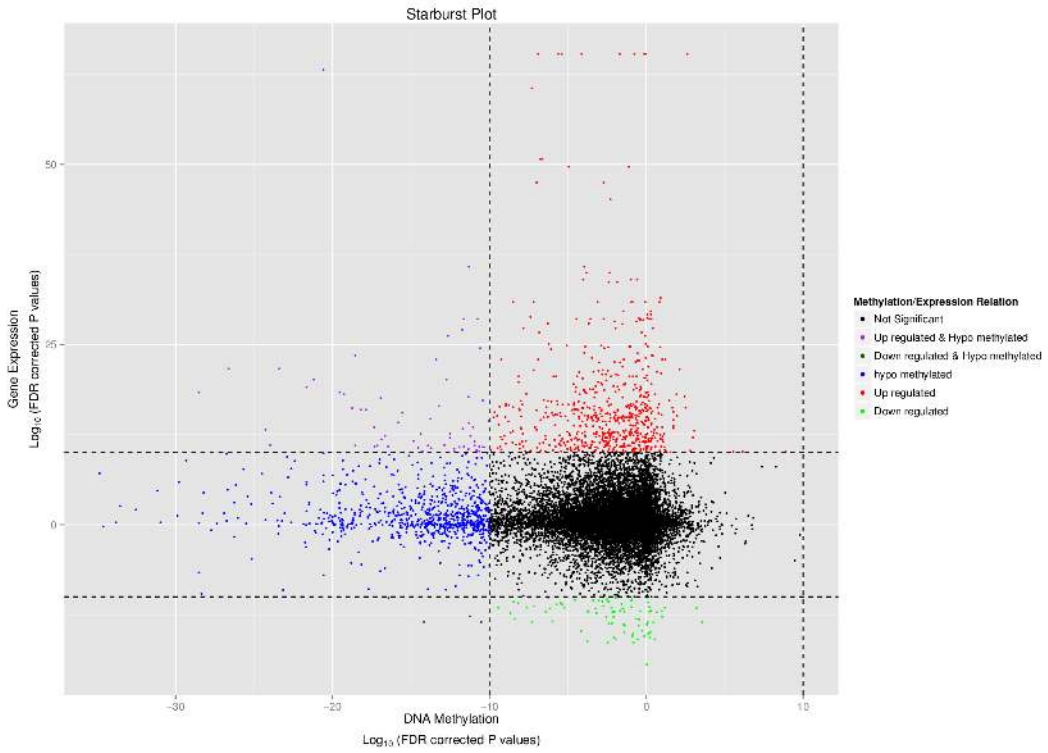
- **met** SummarizedExperiment with methylation data obtained from the `TCGAprepare` and processed by `TCGAanalyze_DMR` function. Expected colData columns: diffmean and p.value.adj
- **exp** Matrix with expression data obtained from the `TCGAanalyze_DEA` function. Expected colData columns: logFC, FDR
- **filename** pdf filename
- **legend** legend title
- **color** vector of colors to be used in graph
- **label** vector of labels to be used in graph
- **title** main title
- **ylab** y axis text
- **xlab** x axis text
- **xlim** x limits to cut image
- **ylim** y limits to cut image
- **p.cut** p value cut-off
- **group1** The name of the group 1 Obs: Column p.value.adj.group1.group2 should exist
- **group2** The name of the group 2. Obs: Column p.value.adj.group1.group2 should exist
- **exp.p.cut** expression p value cut-off
- **met.p.cut** methylation p value cut-off

- **diffmean.cut** If set, the probes with diffmean higher than methylation cut-off will be highlighted in the plot. And the data frame return will be subseted.
- **logFC.cut** If set, the probes with expression fold change higher than methylation cut-off will be highlighted in the plot. And the data frame return will be subseted.

```
resut <- TCGAvisualize_starburst(met,exp,"g1","g2",met.p.cut = 10^-10)
```

As result the function will a plot the figure below and return a matrix with The Gene_symbol and it status in relation to expression(up regulated/down regulated) and methylation (Hyper/Hypo methylated).

```
## Warning in readPNG("starburst.png"): libpng warning: iCCP: profile 'icc':
## 0h: PCS illuminant is not D50
```



TCGAinvestigate: Searching questions, answers and literature

TCGAinvestigate: Find most studied TFs in pubmed

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

```
# 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 6: Table with most studied TF in pubmed related to a specific cancer

mRNA	logFC	FDR	Tumor	Normal	Delta	Pubmed	PMID
MUC1	2.46	0	38498.56	6469.40	94523.36	827	26016502; 25986064; 25982681;
FOS	-2.46	0	14080.32	66543.24	34627.41	513	26011749; 25956506; 25824986;
MDM2	1.41	0	16132.28	4959.92	22824.14	441	26042602; 26001071; 25814188;
GATA3	1.58	0	29394.60	8304.72	46410.03	180	26028330; 26008846; 25994056;
FOXA1	1.45	0	16176.96	5378.88	23465.63	167	26008846; 25995231; 25994056;
EGR1	-2.44	0	16073.08	74947.28	39275.29	77	25703326; 24980816; 24742492;
TOB1	1.43	0	17765.96	6260.08	25476.30	13	25798844; 23589165; 23162636;
MAGED1	1.18	0	20850.16	8244.32	24633.09	6	24225485; 23884293; 22935435;
PTRF	-1.72	0	15200.12	44192.52	26104.62	5	25945613; 23214712; 21913217;
ILF2	1.27	0	22250.32	7854.44	28246.23	0	0

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.

TCGAsocial with BioConductor

Find most downloaded packages in CRAN or BioConductor

```
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 7: Table with number of downloads about a list of packages

Package	NumberDownload
limma	71307
edgeR	33842
survcomp	3679

Table 8: Find most related question in support.bioconductor.org with keyword = tcga

question	BiostarsSite	PackageSuggested
A: Calculating Ibd Using R Package	/55481/	TIN
A: How To Identify Rotamer States From A Pdb ?	/96579/	SIM
A: Pathway Analysis In R	/14316/	sigPathway
A: Ngs Question ~ Consensus	/17535/	sigPathway

`TCGAsocial` with Biostar

Find most related question in biostar.

```
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 9: 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 10: Find most related question in biostar with package

question	BiostarsSite	PackageSuggested
A: Calculating Ibd Using R Package	/55481/	TIN
A: Pathway Analysis In R	/14316/	sigPathway
A: Ngs Question ~ Consensus	/17535/	sigPathway

TCGA Downstream Analysis some workflows and pipelines

Downstream Analysis n.1 IlluminaHiSeq_RNASeqV2 data

After preparing the gene expression from TCGA data using the `TCGAprepare` function, you can do a normalization of genes using the function `TCGAanalyze_Normalization`, do a quantile filter of genes with the `TCGAanalyze_Filtering` function.

TCGAanalyze_Normalization allows user to normalize mRNA transcripts and miRNA, using R *EDASeq* package. Normalization for RNA-Seq Numerical and graphical summaries of RNA-Seq read data. Within-lane normalization procedures to

adjust for GC-content effect (or other gene-level effects) on read counts: loess robust local regression, global-scaling, and full-quantile normalization (Risso, Davide and Schwartz, Katja and Sherlock, Gavin and Dudoit, Sandrine 2011). Between-lane normalization procedures to adjust for distributional differences between lanes (e.g., sequencing depth): global-scaling and full-quantile normalization (Bullard, James H and Purdom, Elizabeth and Hansen, Kasper D and Dudoit, Sandrine 2010).

For istance returns all mRNA or miRNA with mean across all samples, higher than the threshold defined quantile mean across all samples.

Also, in order to classify your samples (barcode) you can use the `TCGAquery_SampleTypes` 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::TCGAanalyze_Normalization(dataBRCA, geneInfo)

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

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

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

Downstream Analysis n.2 IlluminaHiSeq_RNASeq data

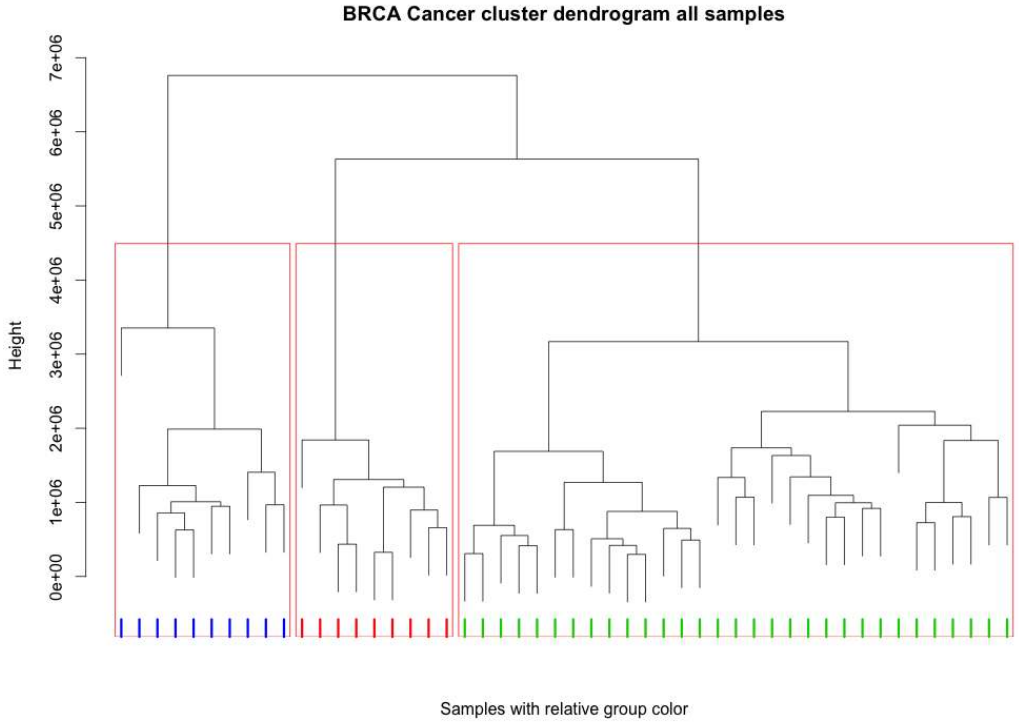
You can easily search TCGA samples, download and prepare a matrix of gene expression.

```
# 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 <- TCGAquery_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")
```

Downstream Analysis n.3 LGG and GBM Integration (Heatmap and Cluster)

```
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)

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:

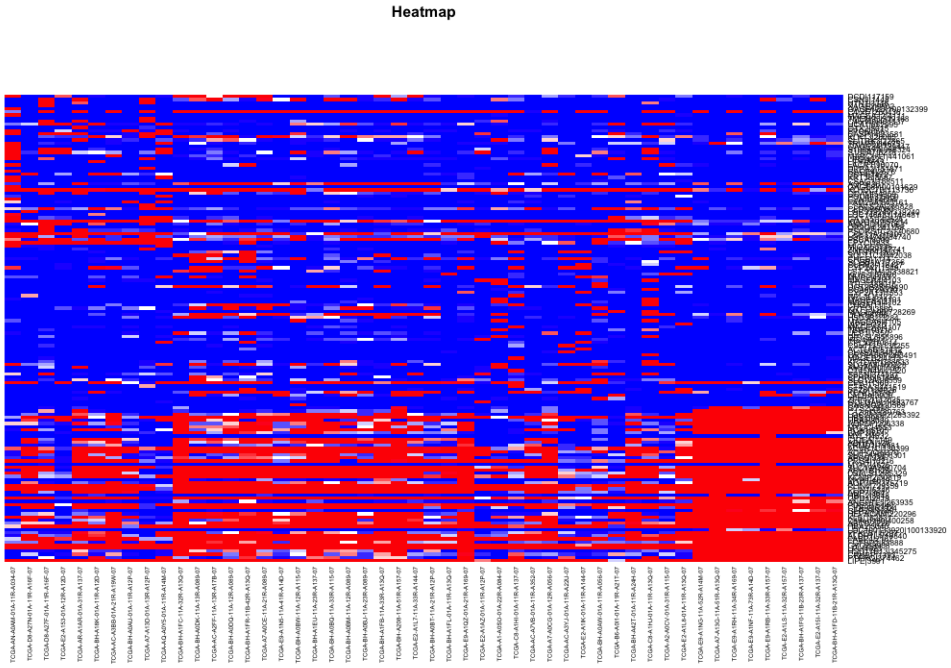



```
library(TCGAbiolinks)


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


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

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

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

DEGsBlueLevel <- TCGAanalyze_LevelTab(DEGsBlue, "GroupBlue", "GroupOther", GroupBlueData,
```

```
    cbind(GroupGreen3Data, GroupRedData), typeOrder = TRUE)
DEGsGreen3Level <- TCGAanalyze_LevelTab(DEGsGreen3, "GroupGreen3", "GroupOther",
    GroupGreen3Data, cbind(GroupBlueData, GroupRedData), typeOrder = TRUE)
DEGsRedLevel <- TCGAanalyze_LevelTab(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:



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=en_US.UTF-8
```

```
##  [9] LC_ADDRESS=en_US.UTF-8        LC_TELEPHONE=en_US.UTF-8
## [11] LC_MEASUREMENT=pt_BR.UTF-8    LC_IDENTIFICATION=en_US.UTF-8
##
## attached base packages:
##  [1] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] stringr_1.0.0              png_0.1-7
##  [3] SummarizedExperiment_0.3.2 Biobase_2.29.1
##  [5] GenomicRanges_1.21.17      GenomeInfoDb_1.5.9
##  [7] IRanges_2.3.17             S4Vectors_0.7.12
##  [9] BiocGenerics_0.15.5        TCGAbiolinks_0.99.1
## [11] BiocStyle_1.7.6
##
## loaded via a namespace (and not attached):
##  [1] httr_1.0.0
##  [2] edgeR_3.11.2
##  [3] splines_3.2.1
##  [4] R.utils_2.1.0
##  [5] highr_0.5
##  [6] aroma.light_2.5.2
##  [7] latticeExtra_0.6-26
##  [8] xlsxjars_0.6.1
##  [9] coin_1.0-24
## [10] Rsamtools_1.21.14
## [11] yaml_2.1.13
## [12] RSQLite_1.0.0
## [13] lattice_0.20-33
## [14] limma_3.25.15
## [15] downloader_0.4
## [16] chron_2.3-47
## [17] digest_0.6.8
## [18] RColorBrewer_1.1-2
## [19] XVector_0.9.1
## [20] rvest_0.2.0
## [21] colorspace_1.2-6
## [22] Matrix_1.2-2
## [23] htmltools_0.2.6
## [24] R.oo_1.19.0
## [25] plyr_1.8.3
## [26] XML_3.98-1.3
## [27] devtools_1.8.0
## [28] ShortRead_1.27.5
## [29] biomaRt_2.25.1
## [30] genefilter_1.51.0
## [31] zlibbioc_1.15.0
## [32] mvtnorm_1.0-3
## [33] xtable_1.7-4
## [34] scales_0.2.5
## [35] supraHex_1.7.2
## [36] BiocParallel_1.3.48
## [37] git2r_0.10.1
## [38] annotate_1.47.4
```

```
## [39] ggplot2_1.0.1
## [40] GenomicFeatures_1.21.14
## [41] hexbin_1.27.0
## [42] proto_0.3-10
## [43] survival_2.38-3
## [44] magrittr_1.5
## [45] memoise_0.2.1
## [46] evaluate_0.7
## [47] GGally_0.5.0
## [48] R.methodsS3_1.7.0
## [49] nlme_3.1-121
## [50] MASS_7.3-43
## [51] xml2_0.1.1
## [52] hwriter_1.3.2
## [53] graph_1.47.2
## [54] tools_3.2.1
## [55] data.table_1.9.4
## [56] formatR_1.2
## [57] matrixStats_0.14.2
## [58] xlsx_0.5.7
## [59] munsell_0.4.2
## [60] AnnotationDbi_1.31.17
## [61] lambda.r_1.1.7
## [62] rversions_1.0.2
## [63] Biostrings_2.37.3
## [64] DESeq_1.21.0
## [65] futile.logger_1.4.1
## [66] RCurl_1.95-4.7
## [67] rjson_0.2.15
## [68] igraph_1.0.1
## [69] bitops_1.0-6
## [70] rmarkdown_0.7
## [71] dnet_1.0.7
## [72] gtable_0.1.2
## [73] DBI_0.3.1
## [74] reshape_0.8.5
## [75] roxygen2_4.1.1
## [76] curl_0.9.2
## [77] R6_2.1.0
## [78] reshape2_1.4.1
## [79] EDASeq_2.3.2
## [80] GenomicAlignments_1.5.12
## [81] knitr_1.10.5
## [82] rtracklayer_1.29.14
## [83] futile.options_1.0.0
## [84] Rgraphviz_2.13.0
## [85] ape_3.3
## [86] TxDb.Hsapiens.UCSC.hg19.knownGene_3.1.3
## [87] modeltools_0.2-21
## [88] rJava_0.9-7
## [89] stringi_0.5-5
## [90] Rcpp_0.12.0
## [91] geneplotter_1.47.0
```

References

Bullard, James H and Purdom, Elizabeth and Hansen, Kasper D and Dudoit, Sandrine. 2010. "Evaluation of Statistical Methods for Normalization and Differential Expression in MRNA-Seq Experiments."

Huber, Wolfgang and Carey, Vincent J and Gentleman, Robert and Anders, Simon and Carlson, Marc and Carvalho, Benilton S and Bravo, Hector Corrada and Davis, Sean and Gatto, Laurent and Girke, Thomas and others. 2015. "Orchestrating High-Throughput Genomic Analysis with Bioconductor."

Risso, Davide and Schwartz, Katja and Sherlock, Gavin and Dudoit, Sandrine. 2011. "GC-Content Normalization for RNA-Seq Data."