Working with TCGAbiolinks package

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

2015-09-23

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.

Installation

For the moment the package is in the devel branch of bioconductor repository. To install use the code below.

```
source("http://bioconductor.org/biocLite.R")
useDevel()
biocLite("TCGAbiolinks")
```

TCGAquery: Searching TCGA open-access data

TCGAquery: Searching TCGA open-access data for download

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

The next subsections will detail each of the search parameters. Below, we show some search examples:

```
query <- TCGAquery(tumor = c("LGG","GBM"), level = 3,
                   platform = c("HumanMethylation450","HumanMethylation27"),
                   samples = c("TCGA-19-4065","TCGA-E1-5322-01A-01D-1467-05"),
                   version = list(c("HumanMethylation450","LGG",1),
                                  c("HumanMethylation450","GBM",5)))
```

TCGAquery: Searching by tumor

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

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

The list of tumors is show below:

Table 1: List of tumors

abbreviation	id	name
SARC	30	Sarcoma

abbreviation	id	name
BLCA	28	Bladder Urothelial Carcinoma
GBM	1	Glioblastoma multiforme
LUSC	3	Lung squamous cell carcinoma
OV	2	Ovarian serous cystadenocarcinoma
LUAD	4	Lung adenocarcinoma
BRCA	5	Breast invasive carcinoma
COAD	6	Colon adenocarcinoma
KIRC	7	Kidney renal clear cell carcinoma
KIRP	8	Kidney renal papillary cell carcinoma
STAD	9	Stomach adenocarcinoma
HNSC	10	Head and Neck squamous cell carcinoma
LIHC	11	Liver hepatocellular carcinoma
CESC	12	Cervical squamous cell carcinoma and endocervical adenocarcinoma
LAML	13	Acute Myeloid Leukemia
SKCM	15	Skin Cutaneous Melanoma
THCA	20	Thyroid carcinoma
LGG	21	Brain Lower Grade Glioma
PRAD	22	Prostate adenocarcinoma
UCEC	23	Uterine Corpus Endometrial Carcinoma
READ	24	Rectum adenocarcinoma
DLBC	26	Lymphoid Neoplasm Diffuse Large B-cell Lymphoma
PAAD	27	Pancreatic adenocarcinoma
ESCA	29	Esophageal carcinoma
CNTL	31	Controls
KICH	32	Kidney Chromophobe
UVM	39	Uveal Melanoma
MESO	33	Mesothelioma
UCS	34	Uterine Carcinosarcoma
TGCT	40	Testicular Germ Cell Tumors
ACC	35	Adrenocortical carcinoma
LCML	36	Chronic Myelogenous Leukemia
PCPG	37	Pheochromocytoma and Paraganglioma
MISC	38	Miscellaneous
CHOL	41	Cholangiocarcinoma
THYM	42	Thymoma
FPPP	43	FFPE Pilot Phase II

`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")
```

The list of platforms is show below:

Table 2: List of tumors

displayName	id	name
Agilent Human Genome CGH Microarray 44K	34	WHG-CGH_4x44B
Agilent Whole Human Genome Microarray Kit	39	WHG-4x44K_G4112F
Agilent Whole Human Genome	35	WHG-1x44K_G4112A
Tissue Images	28	tissue_images
supplemental_clinical	82	supplemental_clinical
SOLiD curated DNA sequencing	70	SOLiD_DNASeq_curated
SOLiD curated DNA sequencing - controlled	78	SOLiD_DNASeq_Cont_curated
SOLiD automated DNA sequencing - controlled	77	SOLiD_DNASeq_Cont_automated
ABI SOLiD DNA Sequencing - Controlled	61	SOLiD_DNASeq_Cont
SOLiD automated DNA sequencing	69	SOLiD_DNASeq_automated
ABI SOLiD DNA Sequencing	42	SOLiD_DNASeq
Pathology Reports	45	pathology_reports
Multi-Center Mutations - Controlled	84	Multicenter_mutation_calling_MC3_Cont
Multi-Center Mutations	83	Multicenter_mutation_calling_MC3
Mixed curated DNA sequencing	72	Mixed_DNASeq_curated
Mixed curated DNA sequencing - controlled	80	Mixed_DNASeq_Cont_curated
Mixed automated DNA sequencing - controlled	79	Mixed_DNASeq_Cont_automated
Mixed DNA Sequencing - Controlled	63	Mixed_DNASeq_Cont
Mixed automated DNA sequencing	71	Mixed_DNASeq_automated
Mixed DNA Sequencing	62	Mixed_DNASeq
Biospecimen Metadata - Minimal Set - All Samples - Tab-delimited	37	minbiotab
Biospecimen Metadata - Minimal Set	32	minbio
Microsatellite Instability Analysis	47	microsat_i
M.D. Anderson Reverse Phase Protein Array Core	46	MDA_RPPA_Core
Affymetrix Human Mapping 250K Sty Array	26	Mapping250K_Sty
Affymetrix Human Mapping 250K Nsp Array	25	Mapping250K_Nsp
Illumina HiSeq 2000 Bisulfite-converted DNA Sequencing	64	IlluminaHiSeq_WGBS
Illumina HiSeq 2000 Total RNA Sequencing Version 2 analysis	81	IlluminaHiSeq_TotalRNASeqV2
Illumina HiSeq 2000 RNA Sequencing Version 2 analysis	58	IlluminaHiSeq_RNASeqV2
Illumina HiSeq 2000 RNA Sequencing	51	IlluminaHiSeq_RNASeq
Illumina HiSeq 2000 mRNA Digital Gene Expression	49	IlluminaHiSeq_mRNA_DGE
Illumina HiSeq 2000 miRNA Sequencing	50	IlluminaHiSeq_miRNASeq
IlluminaHiSeq curated DNA sequencing	66	IlluminaHiSeq_DNASeq_curated
IlluminaHiSeq curated DNA sequencing - controlled	74	IlluminaHiSeq_DNASeq_Cont_curated
IlluminaHiSeq automated DNA sequencing - controlled	73	IlluminaHiSeq_DNASeq_Cont_automated
Illumina HiSeq 2000 DNA Sequencing - Controlled	59	IlluminaHiSeq_DNASeq_Cont
Illumina HiSeq for Copy Number Variation	53	IlluminaHiSeq_DNASeqC
IlluminaHiSeq automated DNA sequencing	65	IlluminaHiSeq_DNASeq_automated
Illumina HiSeq 2000 DNA Sequencing	52	IlluminaHiSeq_DNASeq
Illumina GoldenGate	29	IlluminaGG
Illumina Genome Analyzer RNA Sequencing Version 2 analysis	57	IlluminaGA_RNASeqV2
Illumina Genome Analyzer RNA Sequencing	41	IlluminaGA_RNASeq
Illumina Genome Analyzer mRNA Digital Gene Expression	22	IlluminaGA_mRNA_DGE
Illumina Genome Analyzer miRNA Sequencing	43	IlluminaGA_miRNASeq
IlluminaGA curated DNA sequencing	68	IlluminaGA_DNASeq_curated
IlluminaGA curated DNA sequencing - controlled	76	IlluminaGA_DNASeq_Cont_curated
IlluminaGA automated DNA sequencing - controlled	75	IlluminaGA_DNASeq_Cont_automated

displayName	id	name
Illumina Genome Analyzer DNA Sequencing - Controlled	60	IlluminaGA_DNASeq_Cont
IlluminaGA automated DNA sequencing	67	IlluminaGA_DNASeq_automated
Illumina Genome Analyzer DNA Sequencing	40	IlluminaGA_DNASeq
Illumina DNA Methylation OMA003 Cancer Panel I	3	IlluminaDNAMethylation_OMA003_CPI
Illumina DNA Methylation OMA002 Cancer Panel I	2	IlluminaDNAMethylation_OMA002_CPI
Illumina Infinium Human DNA Methylation 450	48	HumanMethylation450
Illumina Infinium Human DNA Methylation 27	13	HumanMethylation27
Illumina 550K Infinium HumanHap550 SNP Chip	7	HumanHap550
Illumina Human1M-Duo BeadChip	16	Human1MDuo
Affymetrix Human Exon 1.0 ST Array	6	HuEx-1_0-st-v2
Affymetrix HT Human Genome U133 Array Plate Set	4	HT_HG-U133A
Agilent Human miRNA Microarray	36	H-miRNA_G4470A
Agilent Human miRNA Early Access Array	33	H-miRNA_EarlyAccess
Agilent Human miRNA Microarray Rel12.0	20	H-miRNA_8x15Kv2
Agilent 8 x 15K Human miRNA-specific microarray	12	H-miRNA_8x15K
Affymetrix Human Genome U133 Plus 2.0 Array	24	HG-U133_Plus_2
Affymetrix Human Genome U133A 2.0 Array	23	HG-U133A_2
Agilent Human Genome CGH Custom Microarray 2x415K	21	HG-CGH-415K_G4124A
Agilent Human Genome CGH Microarray 244A	5	HG-CGH-244A
Affymetrix Genome-Wide Human SNP Array 6.0	1	Genome_Wide_SNP_6
Affymetrix Genome-Wide Human SNP Array 5.0	27	GenomeWideSNP_5
Firehose Standardized Data	55	fh_stddata
Firehose Reports	56	fh_reports
Firehose Analyses	54	fh_analyses
Diagnostic Images	44	diagnostic_images
Agilent SurePrint G3 Human CGH Microarray Kit 1x1M	15	CGH-1x1M_G4447A
Biospecimen Metadata - Complete Set - All Samples - Tab-delimited	38	biotab
Biospecimen Metadata - Complete Set	30	bio
Agilent 244K Custom Gene Expression G4502A-07-3	14	AgilentG4502A_07_3
Agilent 244K Custom Gene Expression G4502A-07-2	10	AgilentG4502A_07_2
Agilent 244K Custom Gene Expression G4502A-07-1	8	AgilentG4502A_07_1
Agilent 244K Custom Gene Expression G4502A-07	18	AgilentG4502A_07
Applied Biosystems Sequence data	17	ABI
454 Life Sciences Genome Sequence data	31	454

`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.

The list of centers is show below:

Table 3: List of tumors

displayName	id	name
Washington University School of Medicine	24	genome.wustl.edu
International Genomics Consortium	26	intgen.org
Nationwide Children's Hospital	27	nationwidechildrens.org
Vanderbilt University Proteomics	30	vanderbilt.edu
University of Southern California	28	usc.edu

displayName	id	name
Broad Institute of MIT and Harvard	1	broad.mit.edu
Johns Hopkins / University of Southern California	2	jhu-usc.edu
Harvard Medical School	3	hms.harvard.edu
Lawrence Berkeley National Laboratory	4	lbl.gov
Memorial Sloan-Kettering Cancer Center	5	mskcc.org
HudsonAlpha Institute for Biotechnology	6	hudsonalpha.org
University of North Carolina	7	unc.edu
Baylor College of Medicine	8	hgsc.bcm.edu
Washington University School of Medicine	9	genome.wustl.edu
Combined GSCs	10	combined GSCs
Nationwide Children's Hospital	11	nationwidechildrens.org
Broad Institute of MIT and Harvard	12	broad.mit.edu
Washington University School of Medicine	13	genome.wustl.edu
International Genomics Consortium	14	intgen.org
Canada's Michael Smith Genome Sciences Centre	15	bcgsc.ca
Broad Institute of MIT and Harvard	16	broadinstitute.org
Institute for Systems Biology	17	systemsbiology.org
Lawrence Berkeley National Laboratory	18	lbl.gov
Memorial Sloan-Kettering Cancer Center	19	mskcc.org
University of California, Santa Cruz	20	ucsc.edu
MD Anderson	21	mdanderson.org
Rubicon Genomics	22	rubicongenomics.com
Baylor College of Medicine	23	hgsc.bcm.edu
The Johns Hopkins University	31	jhu.edu
Pacific Northwest National Lab	32	pnl.gov
MD Anderson	25	mdanderson.org
University of California, Santa Cruz	29	ucsc.edu
Wellcome Trust Sanger Institute	34	sanger.ac.uk
University of North Carolina	33	unc.edu
Canada's Michael Smith Genome Sciences Centre GSC	35	bcgsc.ca
MD Anderson GSC	36	mdanderson.org

`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 information of the data in TCGA

In case the user want to have an overview of the size and number of samples and date of old versions, you can use the TCGAquery_Version function.

The parameters of this function are:

- tumor
- platform

For example, the code below queries the version history for the IlluminaHiSeq_RNASeqV2 platform .

```
library(TCGAbiolinks)

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

The result is shown below:

Table 4: 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"

An example of the function is below:

```
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 <- 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,clinical_brca_data,
                                            HER="Positive",
                                            gender="FEMALE",
                                            ER="Positive")
```

The result is shown below:

```
## ER Positive Samples:
##   TCGA-BH-A1ES
##   TCGA-BH-A0BZ
##   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-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"
```

`TCGAquery_subtype`: Working with molecular subtypes data.

The Cancer Genome Atlas (TCGA) Research Network has reported integrated genome-wide studies of various diseases. We have added some of the subtypes defined by these report in our package. The LGG, GBM, STAD, BRCA, READ, COAD and LUAD tumors has data added. These subtypes will be automatically added in the summarizedExperiment object through TCGAprepare. But you can also use the `TCGAquery_subtype` function to retrive that information.

```
# Check with subtypes from TCGAprepare and update examples
GBM_path_subtypes <- TCGAquery_subtype(tumor = "gbm")

LGG_path_subtypes <- TCGAquery_subtype(tumor = "lgg")

LGG_clinic <- TCGAquery_clinic(cancer = "LGG",
                               clinical_data_type = "clinical_patient")
```

A subset of the lgg subytpe is shown below:

Table 5: Table common samples among platforms from TCGAquery

	patient	IDH.1p19q.Subtype	stringAsFactor	RNASeqCluster
1	TCGA-CS-4938	IDHmut-non-codel	FALSE	R1
2	TCGA-CS-4941	IDHwt	FALSE	R2
3	TCGA-CS-4942	IDHmut-non-codel	FALSE	R1
4	TCGA-CS-4943	IDHmut-non-codel	FALSE	R1

	patient	IDH.1p19q.Subtype	stringAsFactor	RNASeqCluster
5	TCGA-CS-4944	IDHmut-non-codel	FALSE	NA
6	TCGA-CS-5390	IDHmut-codel	FALSE	R3
7	TCGA-CS-5393	IDHmut-non-codel	FALSE	R4
8	TCGA-CS-5394	IDHmut-codel	FALSE	R3
9	TCGA-CS-5395	IDHwt	FALSE	R2
10	TCGA-CS-5396	IDHmut-codel	FALSE	R3

`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)
matSamples[c(1,4,9),c(1,4,9)]
```

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

Table 6: Table common samples among platforms from TCGAquery

	AgilentG4502A_07_3	HumanMethylation450	IlluminaHiSeq_RNASeqV2
AgilentG4502A_07_3	604	224	530
HumanMethylation450	224	930	790
IlluminaHiSeq_RNASeqV2	530	790	1218

`TCGAquery_investigate`: 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 <- TCGAquery_investigate("breast", dataDEGsFiltLevelTFs, topgenes = 10)
```

The result is shown below:

Table 7: 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

TCGAquery_Social: Searching questions,answers and literature

The TCGAquery_Social 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.

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


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

The result is shown below:

Table 8: Table with number of downloads about a list of packages

Package	NumberDownload
limma	73506
edgeR	34348
survcomp	3587

Table 9: 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

question	BiostarsSite	PackageSuggested
A: Ngs Question ~ Consensus	/17535/	sigPathway

`TCGAquery_Social` with Biostar

Find most related question in biostar.

```
library(TCGAbiolinks)

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

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

The result is shown below:

Table 10: 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 11: 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

`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
- **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.

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
```

Example of use: 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")
```

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

Preparing the data for other packages

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) (Yao, L., Shen, H., Laird, P. W., Farnham, P. J., & Berman, B. P. 2015). The TCGAprepare_elmer for the DNA methylation data will Removing probes with NA values in more than 20% 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)
path <- "kirc"
query <- TCGAquery(tumor = "KIRC", level = 3, platform = "HumanMethylation450")
TCGAdownload(query, path =path)

kirc.met <- TCGAprepare(query,dir = path,
                        save = TRUE,
                        filename = "metKirc.rda",
                        summarizedExperiment = FALSE)

kirc.met <- TCGAprepare_elmer(kirc.met,
                              platform = "HumanMethylation450",
                              save = TRUE,
                              met.na.cut = 0.2)

# Step 1.2 download expression data
query.rna <- TCGAquery(tumor="KIRC",level=3, platform="IlluminaHiSeq_RNASeqV2")
TCGAdownload(query.rna,path=path,type = "rsem.genes.normalized_results")

kirc.exp <- TCGAprepare(query.rna, dir=path, save = TRUE,
                        type = "rsem.genes.normalized_results",
                        filename = "expKirc.rda", summarizedExperiment = FALSE)

kirc.exp <- TCGAprepare_elmer(kirc.exp,
```

```
                                save = TRUE,
                                platform = "IlluminaHiSeq_RNASeqV2")

# Step 2 prepare mee object
library(ELMER)
library(parallel)

geneAnnot <- txs()
geneAnnot$GENEID <- paste0("ID",geneAnnot$GENEID)
geneInfo <- promoters(geneAnnot,upstream = 0, downstream = 0)
probe <- get.feature.probe()
mee <- fetch.mee(meth = kirc.met, exp = kirc.exp, TCGA = TRUE,
                 probeInfo = probe, geneInfo = geneInfo)
save(mee,file="case4mee.rda")
```

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.

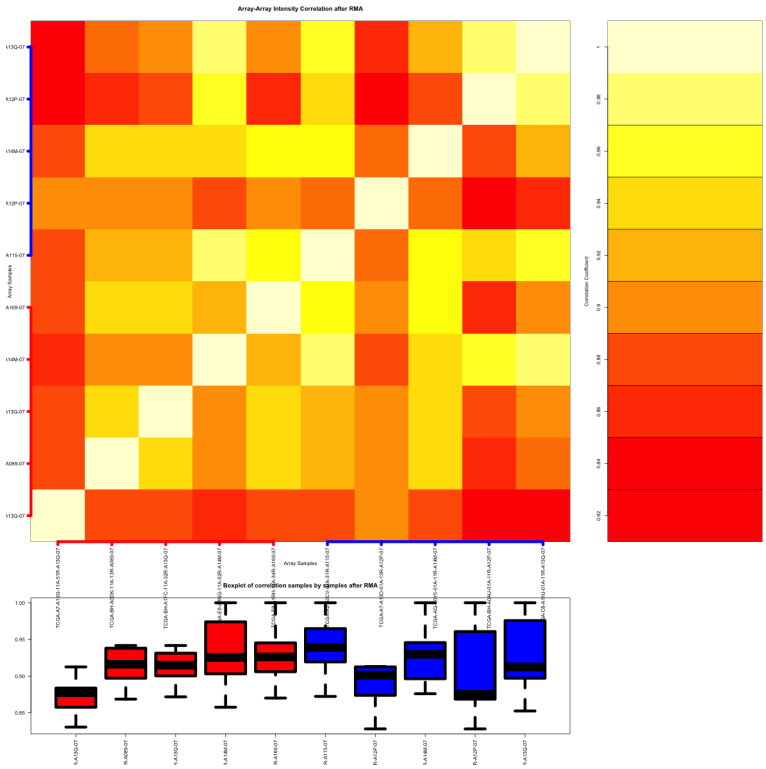
```
# You can define a list of samples to query and download providing relative TCGA barcodes.

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")

# 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 12: Example of a matrix of gene expression (10 genes in rows and 2 samples in columns)

	TCGA-AQ-A0Y5-01A-11R-A14M-07	TCGA-C8-A1HJ-01A-11R-A13Q-07
OR2W1|26692	0	0
CDK18|5129	380	1615
KNG1|3827	1	0
GOLGA8C|729786	6	1
AHCTF1|25909	6384	9417
OSGEP|55644	784	1069
KIAA0831|22863	1613	1389
ARID5A|10865	841	1248
DRG1|4733	5159	11499
CLRN1|7401	1	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)

# normalization of genes
dataNorm <- TCGAanalyze_Normalization(tabDF = dataBRCA, geneInfo =  geneInfo)
## [1] "I Need about  2.5 seconds for this Complete Normalization Upper Quantile  [Processing 80k elements
## [1] "Step 1 of 4: newSeqExpressionSet ..."
## [1] "Step 2 of 4: withinLaneNormalization ..."
## [1] "Step 3 of 4: betweenLaneNormalization ..."
## [1] "Step 4 of 4: exprs ..."
```

```
# quantile filter of genes
dataFilt <- TCGAanalyze_Filtering(tabDF = dataNorm,
                                  method = "quantile",
                                  qnt.cut =  0.25)

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

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

# Diff.expr.analysis (DEA)
dataDEGs <- TCGAanalyze_DEA(mat1 = dataFilt[,samplesNT],
                            mat2 = dataFilt[,samplesTP],
                            Cond1type = "Normal",
                            Cond2type = "Tumor",
                            fdr.cut = 0.01 ,
                            logFC.cut = 1,
                            method = "glmLRT")
## [1] "there are Cond1 type Normal in  5 samples"
## [1] "there are Cond2 type Tumor in  5 samples"
## [1] "there are  15236 features as miRNA or genes "
## [1] "I Need about  5.1 seconds for this DEA. [Processing 30k elements /s]  "
```

```
# DEGs table with expression values in normal and tumor samples
dataDEGsFiltLevel <- TCGAanalyze_LevelTab(dataDEGs,"Tumor","Normal",
```

```
                                    dataFilt[,samplesTP],dataFilt[,samplesNT])
```

The result is shown below:

Table 13: Table DEGs after DEA

mRNA	logFC	FDR	Tumor	Normal	Delta
FN1	3.22	9.095687e-05	449085.8	43415.2	1446330.92
COL1A1	2.83	2.130275e-04	410272.4	51439.0	1161966.52
GAPDH	2.56	8.106153e-04	312485.2	45901.8	800001.25
COL3A1	2.17	5.308783e-03	364673.6	71887.2	791403.35
POSTN	3.07	5.197271e-05	92039.2	10662.0	282234.80
COL11A1	9.54	3.426077e-09	18524.6	21.8	176683.41
MPZ	6.63	5.007909e-03	25115.8	208.6	166563.83
MMP11	6.38	2.928889e-09	18775.4	216.4	119730.85
SPP1	3.52	2.745116e-03	29385.8	2473.8	103402.88
BGN	2.22	6.739521e-04	42463.2	7802.2	94284.37

TCGAanalyze_EAcomplete & TCGAvisualize_EAbarplot: Enrichment Analysis

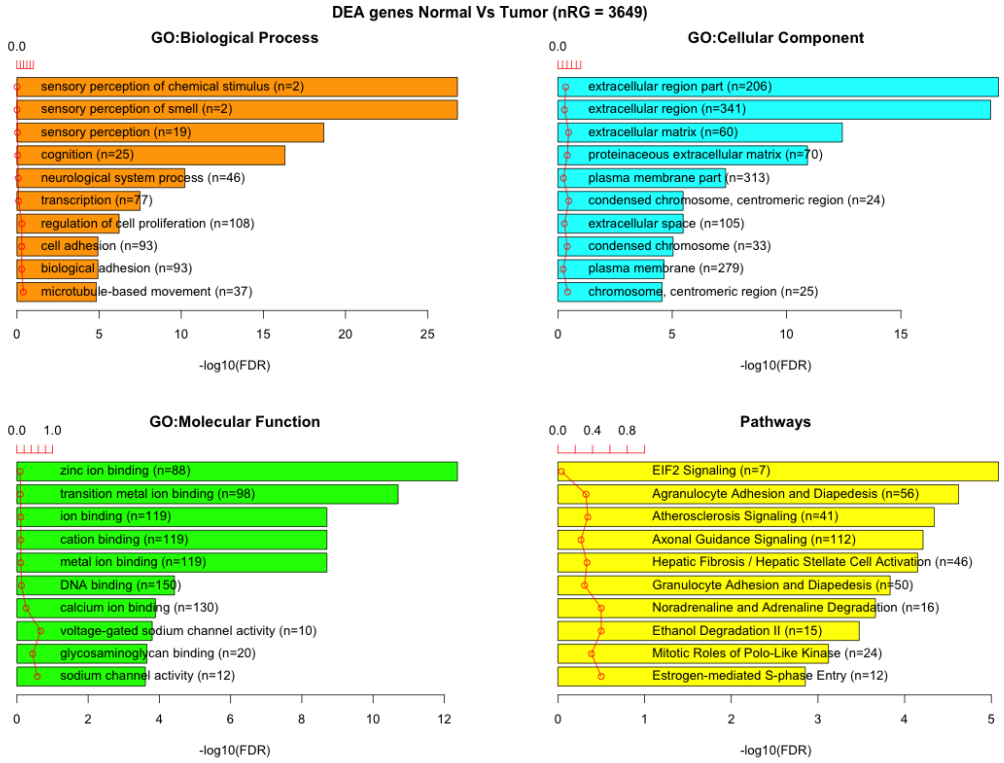
Researchers, in order to better understand the underlying biological processes, often want to retrieve a functional profile of a set of genes that might have an important role. This can be done by performing an enrichment analysis.

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.

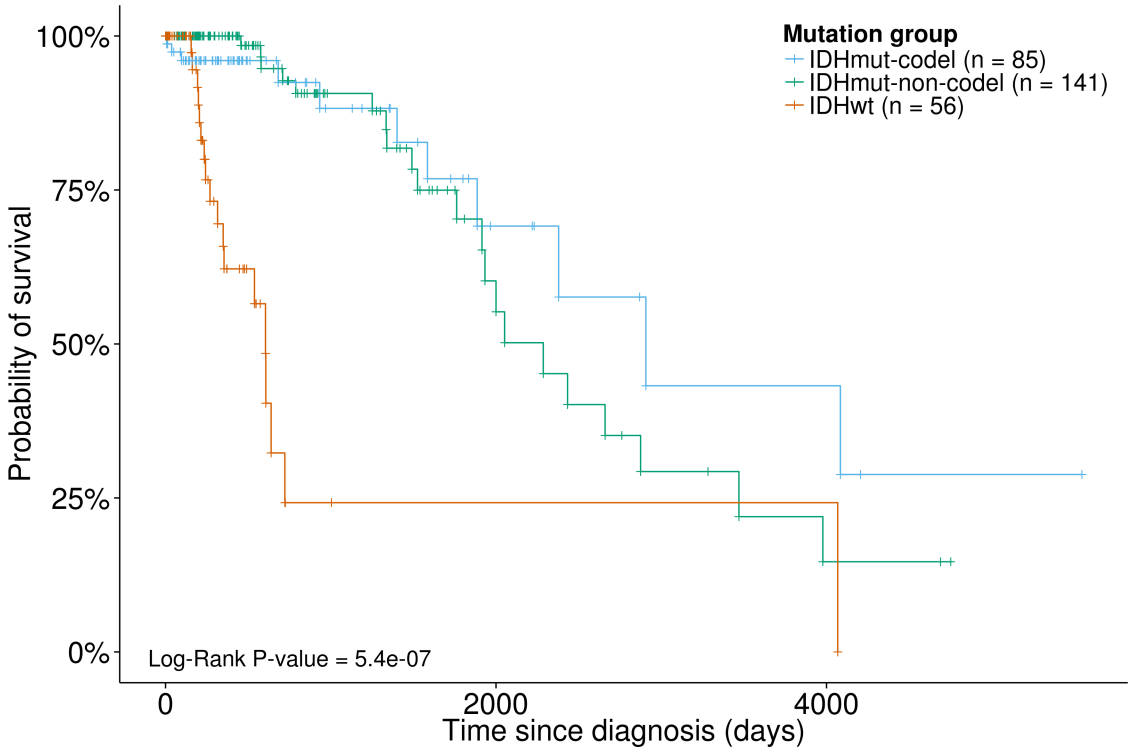
Using the clinical data, it is possible to create a survival plot with the function `TCGAanalyze_survival` as follows:

```
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:




```
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 14: Table KM-survival genes after SA

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

	Mean Tumor Top	Mean Tumor Down	Mean Normal
DCTPP1	13.31	12.01	11.74
APOO	11.40	10.17	10.01
LOC387646	7.92	4.64	5.90
PGK1	15.66	14.18	14.28
CCNE2	11.07	8.23	7.03
CCDC75	9.47	-Inf	9.74
FGD3	12.30	8.57	8.90
FAM166B	6.82	-Inf	7.52
MMP28	8.55	-Inf	9.06
ADHFE1	9.04	6.13	10.10

`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.

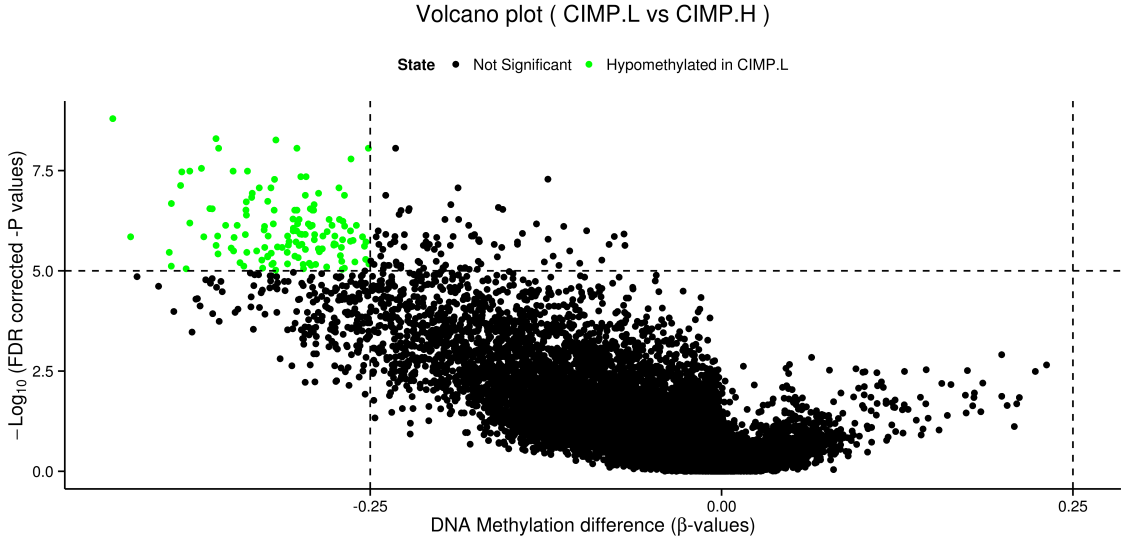
The arguments of volcanoPlot are:

- **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("Not Significant", "Hypermethylated in group1", "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*
- **save** save the object with the results?
- **cores** use multiple cores for non-parametric test

```
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



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

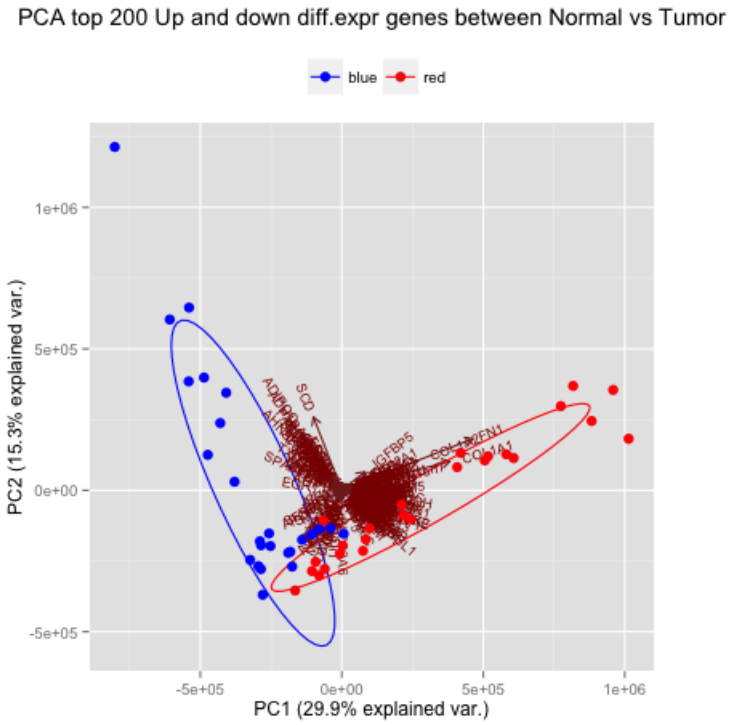
In order to understand better our genes, we can perform a PCA to reduce the number of dimensions of our gene set. The function `TCGAvisualize_PCA` will plot the PCA for different groups.

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

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

# quantile filter of genes
dataFilt <- TCGAanalyze_Filtering(tabDF = dataNorm,
                                  method = "quantile",
                                  qnt.cut =  0.25)

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

The result is shown below:



TCGAvisualize_SurvivalCoxNET Survival Analysis: Cox Regression and dnet package

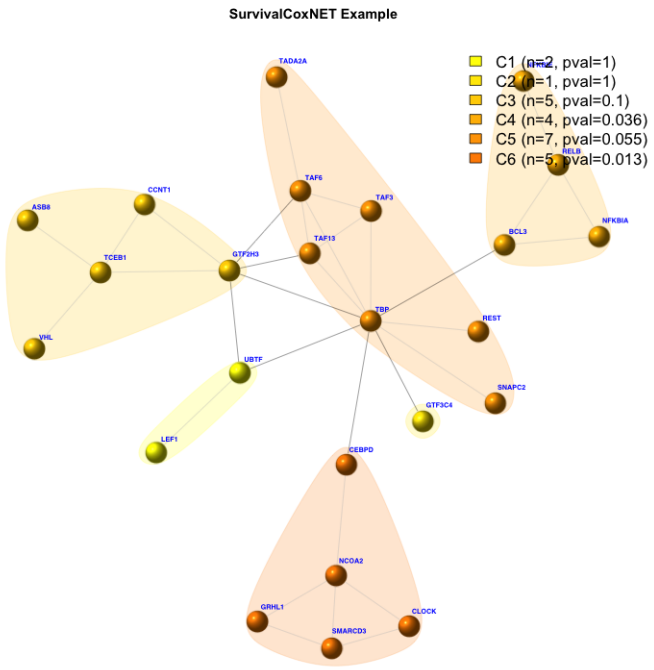
TCGAvisualize_SurvivalCoxNET can help an user to identify a group of survival genes that are significant from univariate Kaplan Meier Analysis and also for Cox Regression. It shows in the end a network build with community of genes with similar range of pvalues from Cox regression (same color) and that 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(tabSurvKMcompleteDEGs),
                            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

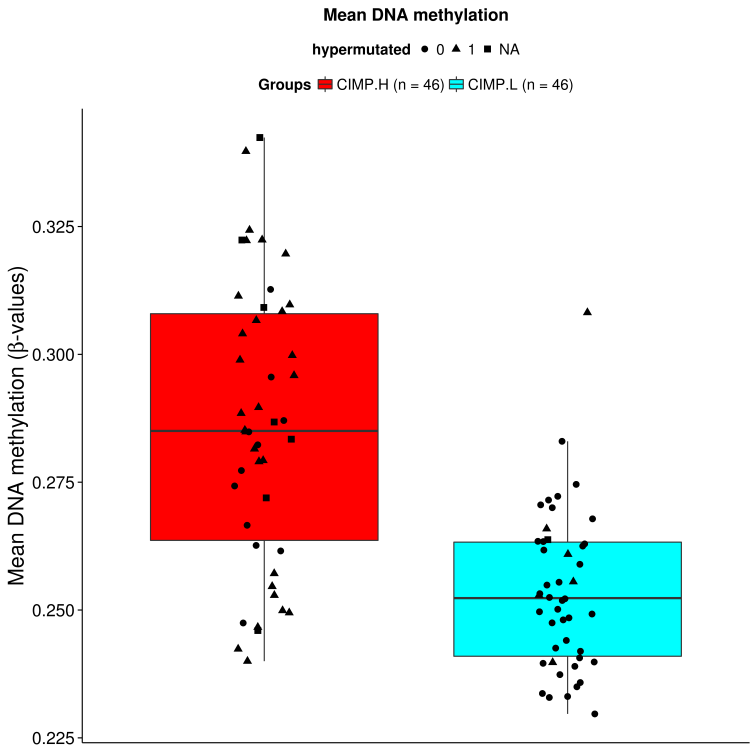
Using the data and calculating the mean DNA methylation per group, it is possible to create a mean DNA methylation boxplot with the function TCGAvisualize_meanMethylation as follows:

```
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:



`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

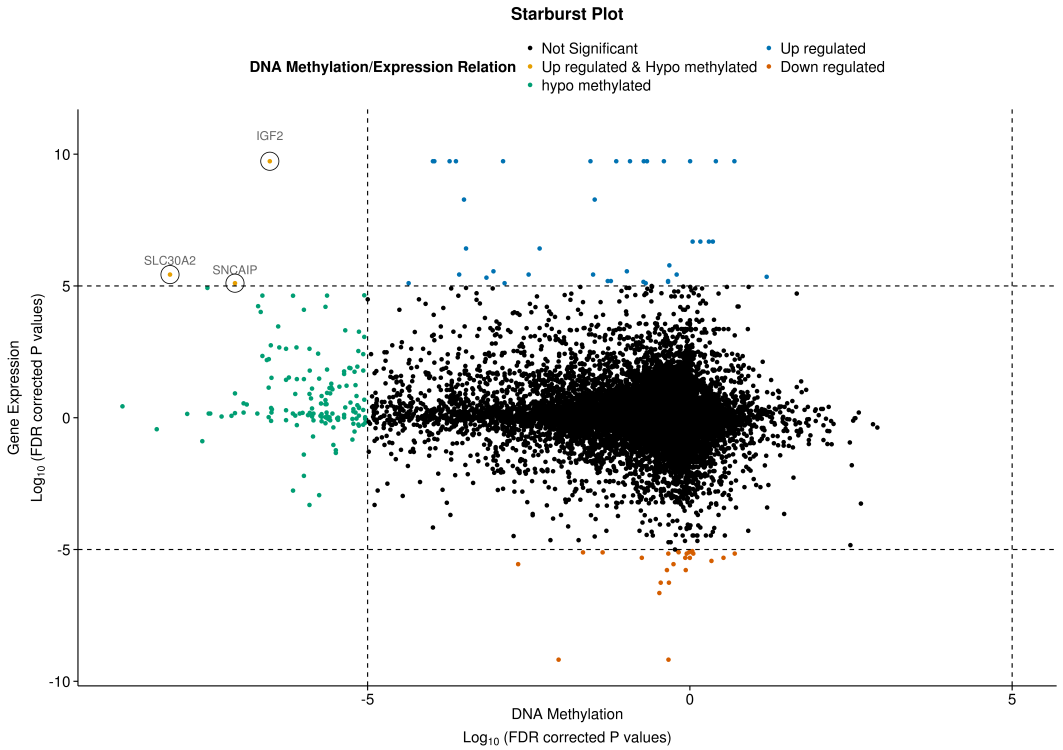
dashed line shows the FDR-adjusted P value of 0.01.

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.

```
starburst <- TCGAvisualize_starburst(coad.SummarizeExperiment,
                                     different.experssion.analysis.data,
                                     group1 = "CIMP.H",
                                     group2 = "CIMP.L",
                                     met.p.cut = 10^-5,
                                     exp.p.cut=10^-5,
                                     names = TRUE)
```

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). The case study 3, shows the complete pipeline for creating this figure.



TCGA Downstream Analysis: Case Studies

Introduction

This vignette shows a complete workflow of the TCGAbiolinks package. The code is divided in 4 case study:

- 1. Expression pipeline (BRCA)
- 2. Expression pipeline (GBM)
- 3. Methylation pipeline - LGG/OV
- 4. Elmer pipeline - KIRC

Parameters definition

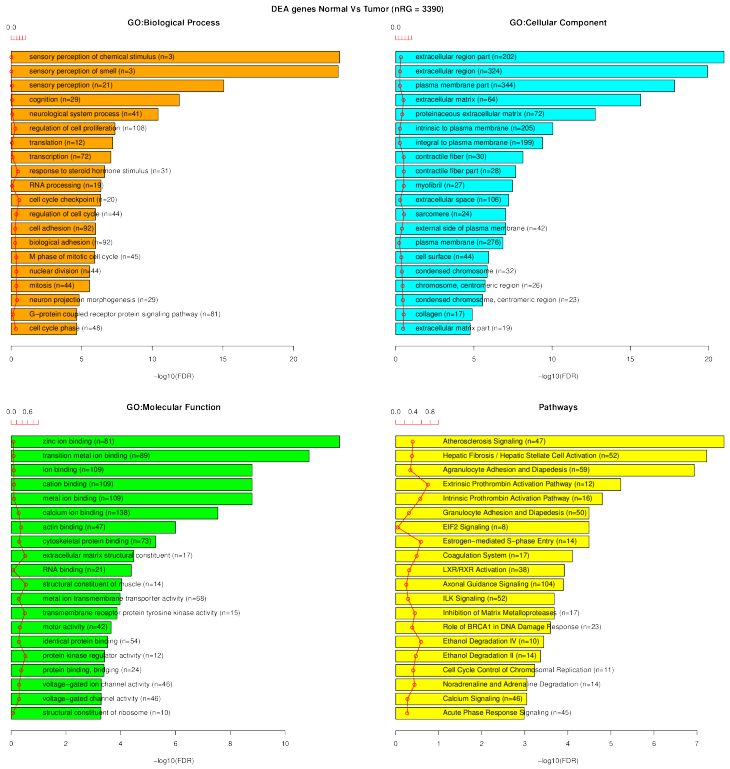
```
PlatformCancer <- "IlluminaHiSeq_RNASeqV2"
dataType <- "rsem.genes.results"
pathGBM<- "../dataGBM"
pathLGG <- "../dataLGG"

library(BiocInstaller)
useDevel()  # we need Devel for SummarizedExperiment package
library(SummarizedExperiment)
library(TCGAbiolinks)
```

Case study n. 1: Pan Cancer downstream analysis BRCA

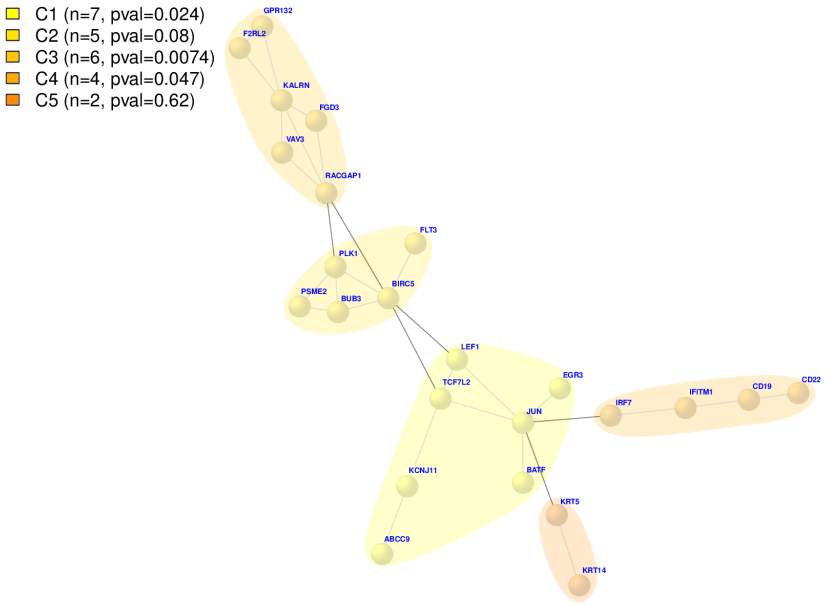
```
library(TCGAbiolinks)
cancer <- "BRCA"
PlatformCancer <- "IlluminaHiSeq_RNASeqV2"
dataType <- "rsem.genes.results"
pathCancer <- paste0("../data",cancer)

# Result....Function.....parameters...p1...pn.................................time execution

datQuery <- TCGAquery(tumor = cancer, platform = PlatformCancer, level = "3")  # time = 0.093s
lsSample <- TCGAquery_samplesfilter(query = datQuery)
dataSubt <- TCGAquery_subtype(tumor = cancer)
dataSmTP <- TCGAquery_SampleTypes(barcode = lsSample$IlluminaHiSeq_RNASeqV2,
                                  typesample = "TP")
dataSmTN <- TCGAquery_SampleTypes(barcode = lsSample$IlluminaHiSeq_RNASeqV2,
                                  typesample ="NT")
dataClin <- TCGAquery_clinic(cancer = cancer,
                             clinical_data_type = "clinical_patient")         # time = 2.606s

            TCGAdownload(data = datQuery, path = pathCancer, type = dataType,
                         samples =c(dataSmTP,dataSmTN))

dataAssy <- TCGAprepare(query = datQuery,
                        dir = pathCancer,
                        type = dataType,
                        save = TRUE,
                        summarizedExperiment = TRUE,
                        samples = c(dataSmTP,dataSmTN),
                        filename = paste0(cancer,"_",PlatformCancer,".rda"))  #time = 1178.353s

dataPrep <- TCGAanalyze_Preprocessing(object = dataAssy,
                                      cor.cut = 0.6,
                                      path = pathCancer,
                                      cancer = cancer )                      #time = 50.372s

dataNorm <- TCGAanalyze_Normalization(tabDF = dataPrep,
                                      geneInfo = geneInfo,
                                      method = "gcContent")                # time = 407.991s

dataFilt <- TCGAanalyze_Filtering(tabDF = dataNorm,
                                  method = "quantile",
                                  qnt.cut =  0.25)                        # time = 0.833s

dataDEGs <- TCGAanalyze_DEA(mat1 = dataFilt[,dataSmTN],
                            mat2 = dataFilt[,dataSmTP],
                            Cond1type = "Normal",
                            Cond2type = "Tumor",
                            fdr.cut = 0.01 ,
                            logFC.cut = 1,
                            method = "glmLRT")                            #time = 215.273s

ansEA <- TCGAanalyze_EAcomplete(TFname="DEA genes Normal Vs Tumor",
                                RegulonList = rownames(dataDEGs))    # time = 69.488s
```

```
TCGAvisualize_EAbarplot(tf = rownames(ansEA$ResBP),
                        GOBPTab = ansEA$ResBP,
                        GOCCTab = ansEA$ResCC,
                        GOMFTab = ansEA$ResMF,
                        PathTab = ansEA$ResPat,
                        nRGTab = rownames(dataDEGs),
                        nBar = 20)

system.time(dataSurv<-TCGAanalyze_SurvivalKM(clinical_patient = dataClin,
                                             dataGE = dataFilt,
                                             Genelist = rownames(dataDEGs),
                                             Survresult = FALSE,
                                             ThreshTop = 0.67,
                                             ThreshDown = 0.33,
                                             p.cut = 0.05))               # time = 175.664s

require(dnet)  # to change
org.Hs.string <- dRDataLoader(RData = "org.Hs.string")

TabCoxNet <- TCGAvisualize_SurvivalCoxNET(dataClin,
                                          dataFilt,
                                          Genelist = rownames(dataSurv),
                                          scoreConfidence = 700,
                                          org.Hs.string = org.Hs.string,
                                          titlePlot = "Case Study n.1 dnet")

# Convert images from pdf to png.
library(animation)
ani.options(outdir = getwd())

im.convert("TCGAvisualize_EAbarplot_Output.pdf",
           output = "case1_EA.png",
           extra.opts="-density 150")

im.convert("SurvivalCoxNETOutput.pdf",
           output = "case1_dnet.png",
           extra.opts="-density 150")
```

The figures resulted from the code above are shown below.



Case Study n.1 dnet



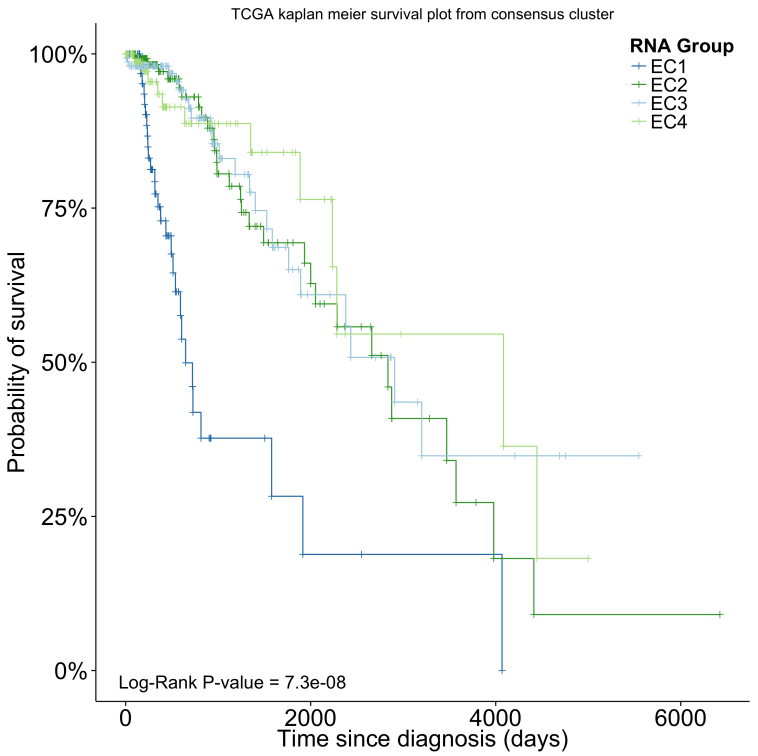
Case study n. 2: Pan Cancer downstream analysis LGG

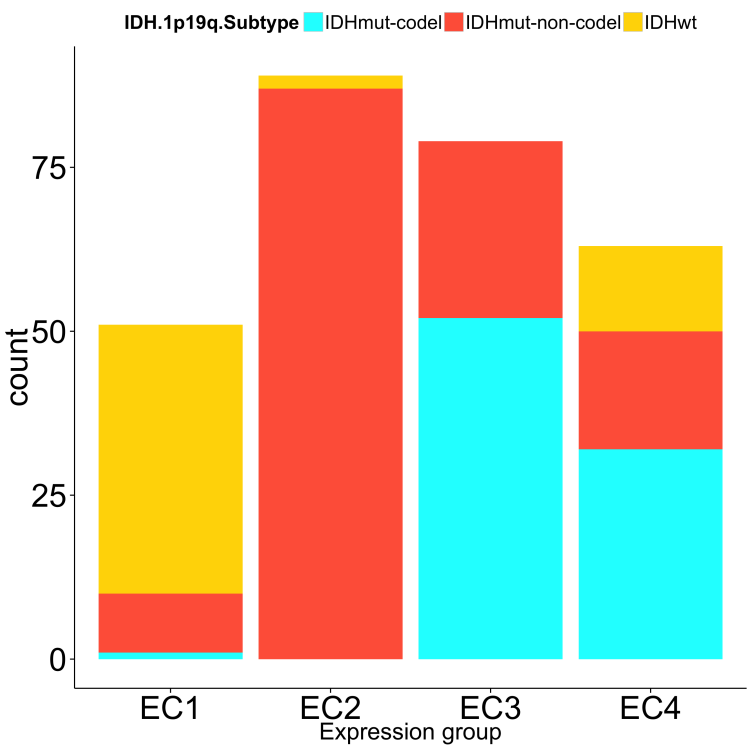
```
library(TCGAbiolinks)
cancer <- "LGG"
PlatformCancer <- "IlluminaHiSeq_RNASeqV2"
dataType <- "rsem.genes.results"
pathCancer <- paste0("../data",cancer)

# Result....Function.....parameters...p1...pn.................................time execution

datQuery <- TCGAquery(tumor = cancer, platform = PlatformCancer, level = "3")  # time = 0.093s
lsSample <- TCGAquery_samplesfilter(query = datQuery)
dataSubt <- TCGAquery_subtype(tumor = cancer)
dataSmTP <- TCGAquery_SampleTypes(barcode = lsSample$IlluminaHiSeq_RNASeqV2,
                                  typesample = "TP")
dataClin <- TCGAquery_clinic(cancer = cancer,
                             clinical_data_type = "clinical_patient")

            TCGAdownload(data = datQuery, path = pathCancer, type = dataType,
                         samples = dataSmTP )

dataAssy <- TCGAprepare(query = datQuery,
                        dir = pathCancer,
                        type = dataType,
                        save = TRUE,
                        summarizedExperiment = TRUE,
                        samples = dataSmTP,
                        filename = paste0(cancer,"_",PlatformCancer,".rda"))

dataPrep <- TCGAanalyze_Preprocessing(object = dataAssy,cor.cut = 0.6)        # time = 13.028s
dataNorm <- TCGAanalyze_Normalization(tabDF = dataPrep,
                                      geneInfo = geneInfo,
                                      method = "gcContent")                   # time = 165.577s

datFilt1 <- TCGAanalyze_Filtering(tabDF = dataNorm,method = "varFilter")
datFilt2 <- TCGAanalyze_Filtering(tabDF = datFilt1,method = "filter1")
datFilt <- TCGAanalyze_Filtering(tabDF = datFilt2,method = "filter2")

data_Hc1 <- TCGAanalyze_Clustering(tabDF = datFilt,method = "hclust", methodHC = "ward.D2")
data_Hc2 <- TCGAanalyze_Clustering(tabDF = datFilt,
                                   method = "consensus",
                                   methodHC = "ward.D2")        # time = 207.389
# deciding number of tree to cuts
cut.tree <-4
paste0(c("EC"),(1:cut.tree))

## consensusClusters contains barcodes for 4 groups
ans <- hclust(ddist <- dist(datFilt), method = "ward.D2")
hhc <- data_Hc2[[cut.tree]]$consensusTree
consensusClusters<-data_Hc2[[cut.tree]]$consensusClass
sampleOrder <- consensusClusters[hhc$order]

consensusClusters <- as.factor(data_Hc2[[cut.tree]]$clrs[[1]])
names(consensusClusters) <- attr(ddist, "Labels")
```

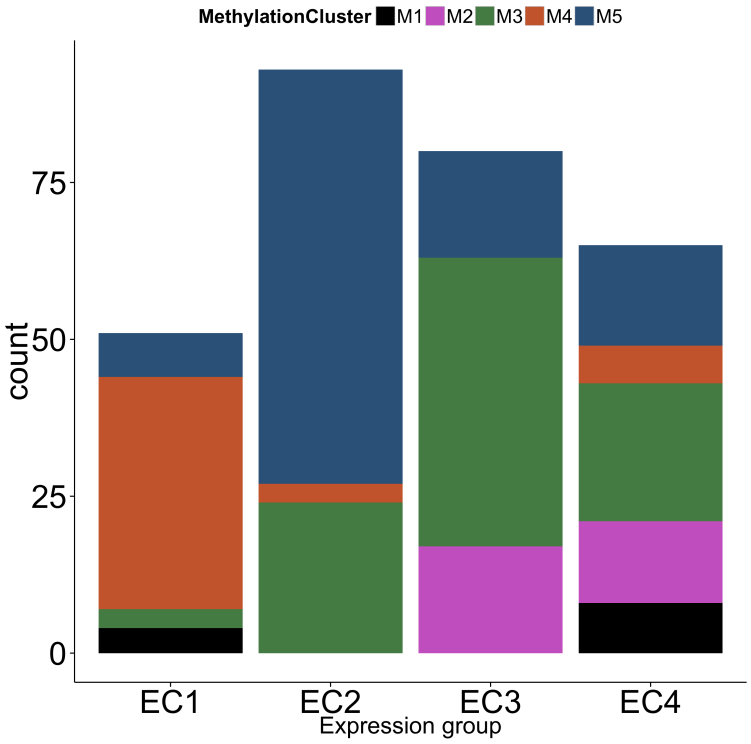
```
names(consensusClusters) <- substr(names(consensusClusters),1,12)

# adding information about gropus from consensus Cluster in clinical data
dataClin <- cbind(dataClin, groupsHC = matrix(0,nrow(dataClin),1))
rownames(dataClin) <- dataClin$bcr_patient_barcode

for( i in 1: nrow(dataClin)){
    currSmp <- dataClin$bcr_patient_barcode[i]
    dataClin[currSmp,"groupsHC"] <- as.character(consensusClusters[currSmp])
}

# plotting survival for groups EC1, EC2, EC3, EC4

TCGAanalyze_survival(data = dataClin,
                     clusterCol = "groupsHC",
                     main = "TCGA kaplan meier survival plot from consensus cluster",
                     height = 10,
                     width=10,
                     legend = "RNA Group",
                     labels=paste0(c("EC"),(1:cut.tree)),
                     color = as.character(levels(consensusClusters)),
                     filename = "case2_surv.png")
dev.off()
TCGAvisualize_BarPlot(DFfilt = datFilt,
                      DFclin = dataClin,
                      DFsubt = dataSubt,
                      data_Hc2 = data_Hc2,
                      Subtype = "IDH.1p19q.Subtype",
                      cbPalette = c("cyan","tomato","gold"),
                      filename = "case2_Idh.png",
                      height = 10,
                      width=10,
                      dpi =300)

TCGAvisualize_BarPlot(DFfilt = datFilt,
                      DFclin = dataClin,
                      DFsubt = dataSubt,
                      data_Hc2 = data_Hc2,
                      Subtype = "MethylationCluster",
                      cbPalette = c("black","orchid3","palegreen4","sienna3", "steelblue4"),
                      filename = "case2_Met.png",
                      height = 10,
                      width=10,
                      dpi =300)

TCGAvisualize_BarPlot(DFfilt = datFilt,
                      DFclin = dataClin,
                      DFsubt = dataSubt,
                      data_Hc2 = data_Hc2,
                      Subtype = "AGE",
                      cbPalette = c("yellow2","yellow3","yellow4"),
                      filename = "case2_Age.png",
                      height = 10,
```

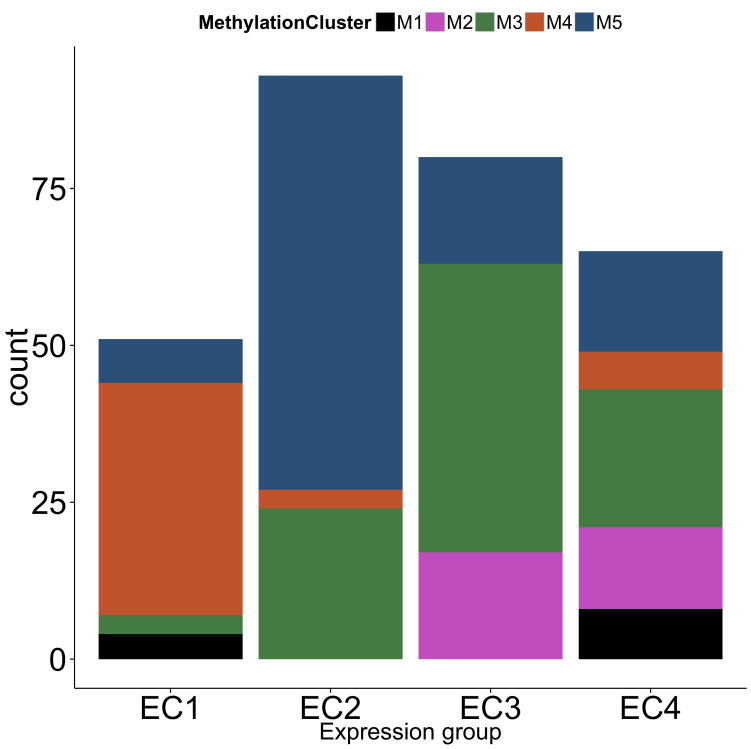
```
                        width=10,
                        dpi =300)

dev.off()
pdf(file="case2_Heatmap2.pdf")
TCGAvisualize_Heatmap(DFfilt = datFilt,
                        DFclin = dataClin,
                        DFsubt = dataSubt,
                        data_Hc2= data_Hc2)
dev.off()


# Convert images from pdf to png.
library(animation)
ani.options(outdir = getwd())

im.convert("case2_Heatmap2.pdf",
           output = "case2_Heatmap2.png",
           extra.opts="-density 150")
```

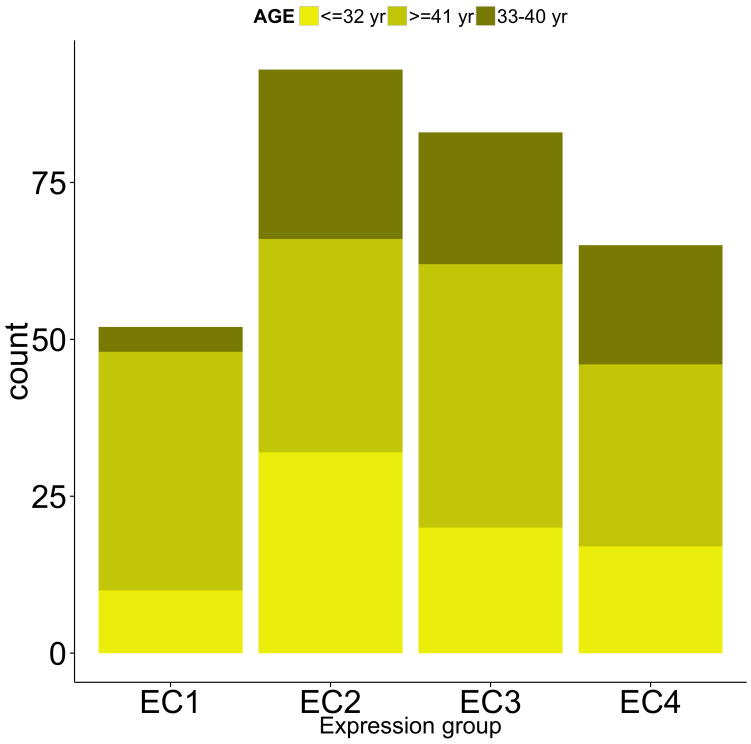
The figures resulted from the code above are shown below.

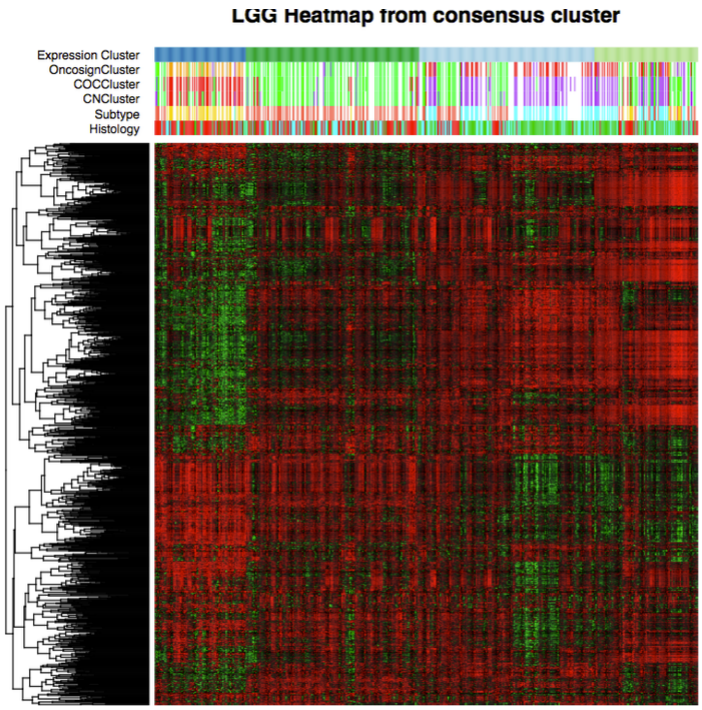



IDH.1p19q.Subtype
IDHmut-codel
IDHmut-non-codel
IDHwt
count
0
25
50
75
EC1
EC2
EC3
EC4
Expression group


MethylationCluster
M1
M2
M3
M4
M5
count
0
25
50
75
EC1
EC2
EC3
EC4
Expression group


MethylationCluster M1 M2 M3 M4 M5
75
50
25
0
count
EC1
EC2
EC3
EC4
Expression group


AGE <=32 yr >=41 yr 33-40 yr
75
50
25
0
count
EC1
EC2
EC3
EC4
Expression group



Case study n. 3: Integration of methylation and expression for COAD

In recent years, it was discovered that there is a relationship between DNA methylation and gene expression and the study of this relationship is often difficult to accomplish.

This case study will show the steps to conduct a study of the relationship between the two types of data.

First we downloaded COAD methylation data for HumanMethylation27k and HumanMethylation450k platforms, and COAD expression data for IlluminaGA_RNASeqV2.

TCGAbiolinks adds by default the classifications already published by researchers. We will use this classification to do our examples. We selected the groups CIMP.L and CIMP.H to do a expression and DNA methylation comparison.

Firstly, we do a DMR (different methylated region) analysis, which will give the difference of DNA methylation for the probes of the groups and their significance value. The output can be seen by a volcano plot.

Secondly, we do a DEA (differential expression analysis) which will give the fold change of gene expression and their significance value.

Finally, using both previous analysis we do a starburst plot to select the genes that are Candidate Biologically Significant.

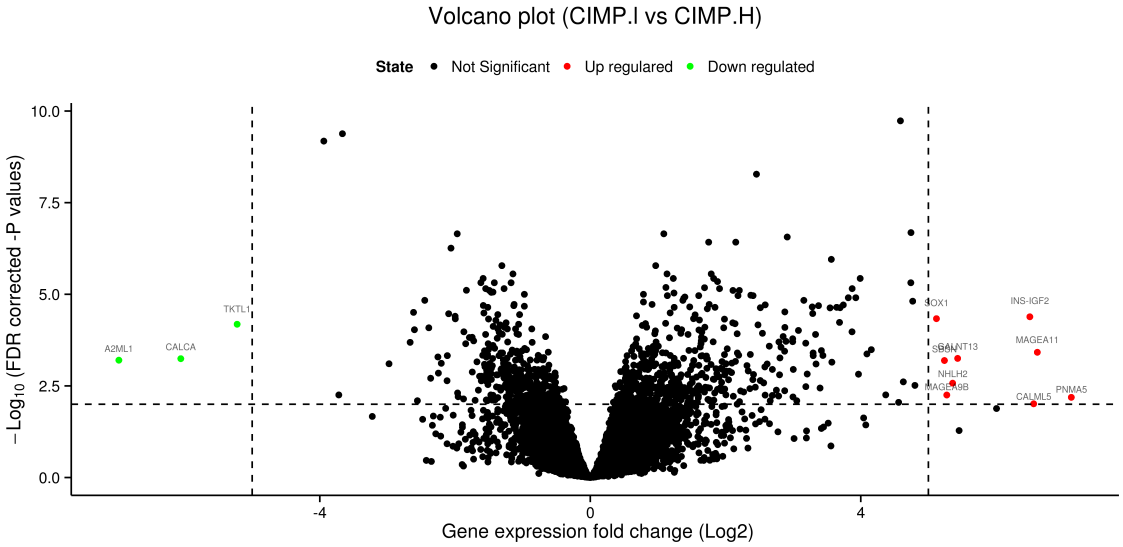
Observation: over the time, the number of samples has increased and the clinical data updated. We used only the samples that had a classification in the examples.

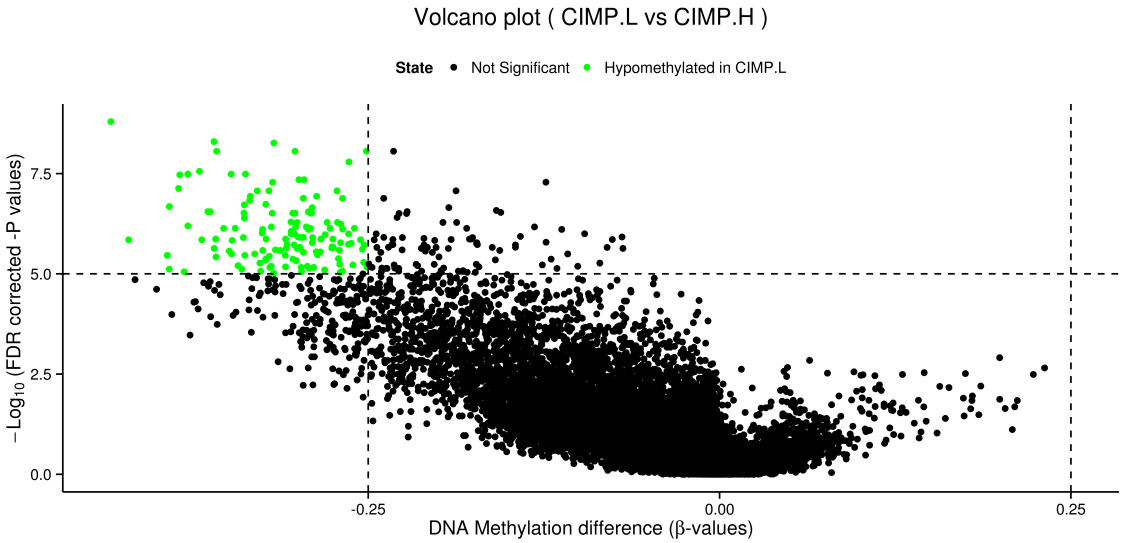
```
#-----------------------------------
# STEP 1: Search, download, prepare |
#-----------------------------------
# 1.1 - Methylation
# ---------------------------------
query.met <- TCGAquery(tumor = c("coad"),
                       platform = c("HumanMethylation27",
                                    "HumanMethylation450"),
                       level = 3)
```

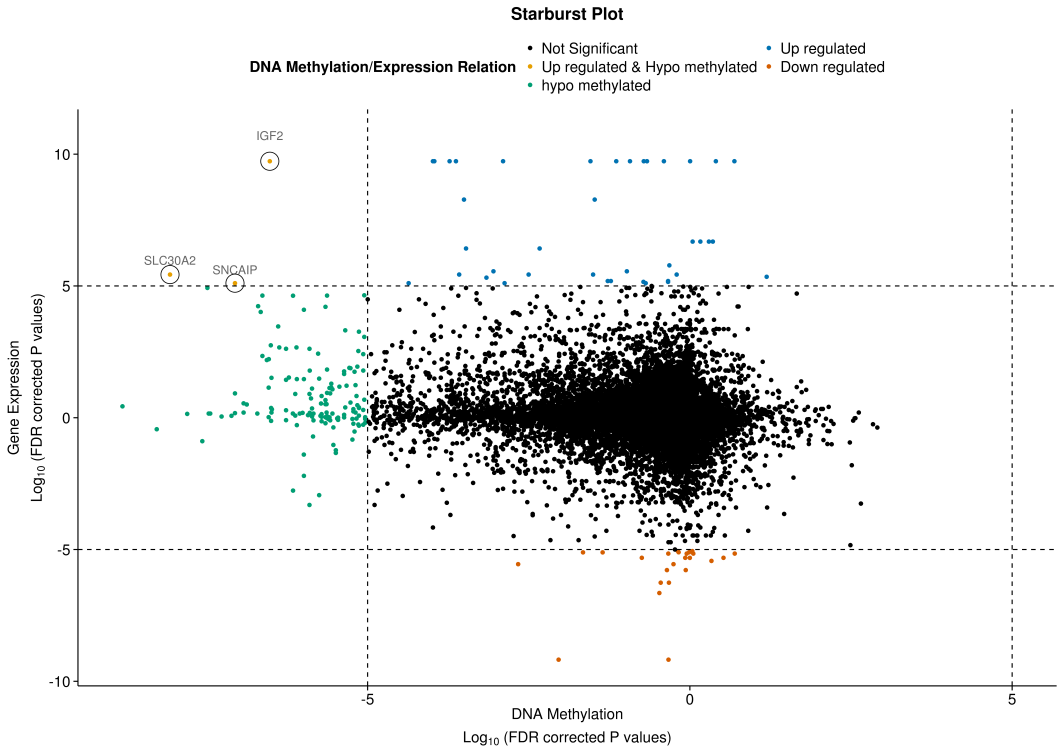
```
TCGAdownload(query.met, path = "/dados/ibm/comparing/biolinks/coad/")

coad.met <- TCGAprepare(query = query.met,
                        dir = "/dados/ibm/comparing/biolinks/coad/",
                        save = TRUE,
                        filename = "metcoad.rda",
                        reannotate = TRUE)

#-----------------------------------
# 1.2 - Expression
# ----------------------------------

coad.query.exp <- TCGAquery(tumor = "coad",
                       platform = "IlluminaGA_RNASeqV2",
                       level = 3)

TCGAdownload(coad.query.exp,
             path = "/dados/ibm/comparing/biolinks/RNA/",
             type = "rsem.genes.results")

coad.exp <- TCGAprepare(query = coad.query.exp,
                        dir = "/dados/ibm/comparing/biolinks/RNA/",
                        type = "rsem.genes.results",
                        save = T,filename = "coadexp.rda")


# removing the samples without classification
coad.met <- subset(coad.met,select = !(colData(coad.met)$methylation_subtype %in% c(NA)))

#-------------------------------------------
# STEP 2: Analysis
#-------------------------------------------
# 2.1 - Mean methylation of samples
# ------------------------------------------
TCGAvisualize_meanMethylation(coad.met,
                              groupCol = "methylation_subtype",
                              subgroupCol = "hypermutated",
                              group.legend  = "Groups",
                              subgroup.legend = "hypermutated",
                              filename = "coad_mean.png")

#-----------------------------------------------
# 2.2 - DMR - Methylation analysis - volcano plot
# ----------------------------------------------
coad.aux <- subset(coad.met,
                   select = colData(coad.met)$methylation_subtype %in% c("CIMP.L","CIMP.H"))


# na.omit
coad.aux <- subset(coad.aux,subset = (rowSums(is.na(assay(coad.aux))) == 0))

# Volcano plot
coad.aux <- TCGAanalyze_DMR(coad.aux, groupCol = "methylation_subtype",
```

```
                              group1 = "CIMP.H",
                              group2="CIMP.L",
                              p.cut = 10^-5,
                              diffmean.cut = 0.25,
                              legend = "State",
                              plot.filename = "coad_CIMPHvsCIMPL_metvolcano.png")

save(coad.aux,file = "coad_pvalue.rda")

#---------------------------------------------
# 2.3 - DEA - Expression analysis - volcano plot
# --------------------------------------------

coad.exp.aux <- subset(coad.exp,
                       select = colData(coad.exp)$methylation_subtype %in% c("CIMP.H","CIMP.L"))

idx <- colData(coad.exp.aux)$methylation_subtype %in% c("CIMP.H")
idx2 <- colData(coad.exp.aux)$methylation_subtype %in% c("CIMP.L")

dataPrep <- TCGAanalyze_Preprocessing(object = coad.exp.aux,
                                      cor.cut = 0.6,
                                      cancer = "coad",
                                      path=".")

dataNorm <- TCGAanalyze_Normalization(tabDF = dataPrep,
                                      geneInfo = geneInfo,
                                      method = "gcContent")

dataFilt <- TCGAanalyze_Filtering(tabDF = dataNorm,
                                  qnt.cut = 0.25,
                                  method='quantile')

dataDEGs <- TCGAanalyze_DEA(mat1 = dataFilt[,idx],
                            mat2 = dataFilt[,idx2],
                            Cond1type = "CIMP.H",
                            Cond2type = "CIMP.l",
                            method = "glmLRT")

TCGAVisualize_volcano(dataDEGs$logFC,dataDEGs$FDR,
                      filename = "Case3_volcanoexp.png",
                      x.cut = 5,
                      names = rownames(dataDEGs),
                      xlab = " Gene expression fold change (Log2)",
                      legend = "State",
                      title = "Volcano plot (CIMP.l vs CIMP.H)")

#----------------------------------------
# 2.4 - Starburst plot
# ---------------------------------------
starburst <- TCGAvisualize_starburst(coad.aux, dataDEGs,"CIMP.H","CIMP.L",
                                     filename = "starburst.png",
                                     met.p.cut = 10^-5,
                                     exp.p.cut = 10^-5,
                                     names = TRUE)
```

The figures resulted from the code above are shown below.







Case study n. 4: Elmer pipeline - KIRC

One of the biggest problems related to the study data is the preparation phase, which often consists of successive steps in order to prepare it to a format acceptable by certain algorithms and software.

With the object of assisting users in this arduous step, TCGAbiolinks offers in data preparation stage, the toPackage argument, which aims to prepare the data in order to obtain the correct format for different packages.

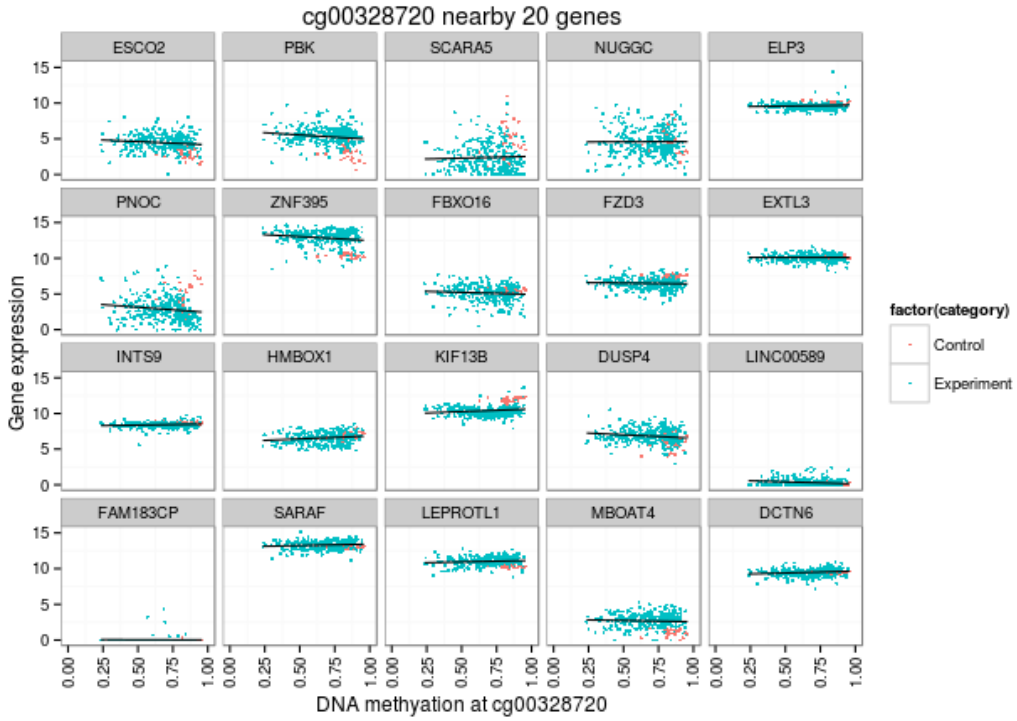
An example of package to perform DNA methylation and expression analysis is ELMER (Huber, Wolfgang and Carey, Vincent J and Gentleman, Robert and Anders, Simon and Carlson, Marc and Carvalho, Benilton S and Bravo, Hector Corrada and Davis, Sean and Gatto, Laurent and Girke, Thomas and others 2015). We will present this case study the study KIRC by TCGAbiolinks and ELMER integration. For more information, please consult ELMER package.

```
#----------------------------------
# STEP 1: Search, download, prepare |
#----------------------------------
# Step 1.1 download methylation data|
# ----------------------------------
path <-  "."
query <- TCGAquery(tumor = "KIRC",level = 3, platform = "HumanMethylation450")
TCGAdownload(query, path = path)
kirc.met <- TCGAprepare(query,dir = path,
                        save = TRUE,
                        filename = "metKirc.rda",
                        summarizedExperiment = FALSE)

kirc.met <- TCGAprepare_elmer(kirc.met,
                              platform = "HumanMethylation450",
                              save = TRUE,
                              met.na.cut = 0.2)
```

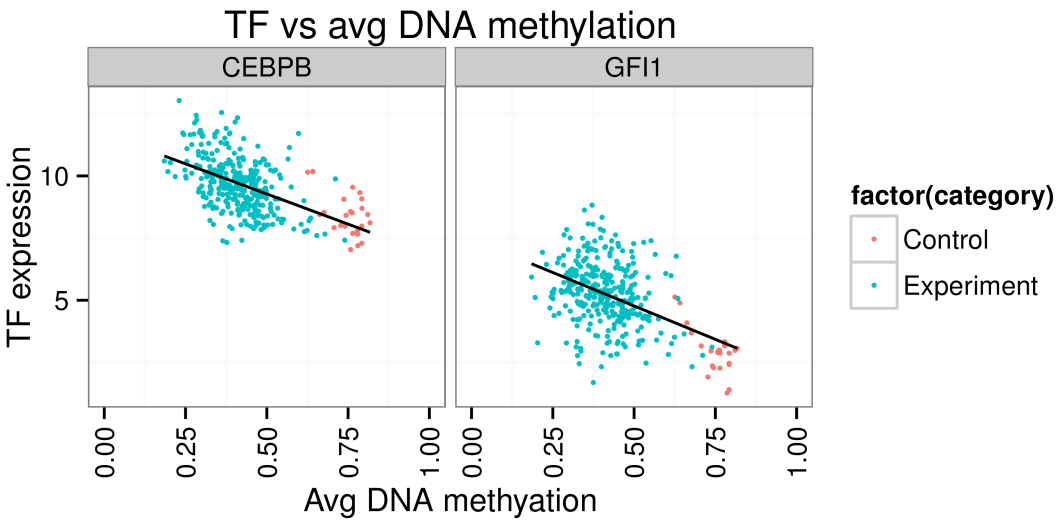
```
# Step 1.2 download expression data
query.rna <- TCGAquery(tumor="KIRC",level=3, platform="IlluminaHiSeq_RNASeqV2")
TCGAdownload(query.rna,path=path,type = "rsem.genes.normalized_results")
kirc.exp <- TCGAprepare(query.rna, dir=path, save = TRUE,
                        type = "rsem.genes.normalized_results",
                        filename = "expKirc.rda", summarizedExperiment = FALSE)

kirc.exp <- TCGAprepare_elmer(kirc.exp,
                              save = TRUE,
                              platform = "IlluminaHiSeq_RNASeqV2")
#----------------------------------
# STEP 2: ELMER integration        |
#----------------------------------
# Step 2.1 prepare mee object      |
# ---------------------------------
library(ELMER)
library(parallel)

geneAnnot <- txs()
geneAnnot$GENEID <- paste0("ID",geneAnnot$GENEID)
geneInfo <- promoters(geneAnnot,upstream = 0, downstream = 0)
probe <- get.feature.probe()
mee <- fetch.mee(meth = kirc.met, exp = kirc.exp, TCGA = TRUE,
                 probeInfo = probe, geneInfo = geneInfo)

#--------------------------------------
# STEP 3: Analysis                    |
#--------------------------------------
# Step 3.1: Get diff methylated probes |
#--------------------------------------
Sig.probes <- get.diff.meth(mee ,cores=detectCores(),
                            dir.out ="kirc",diff.dir="hypo",
                            pvalue = 0.01)

#-----------------------------------------------
# Step 3.2: Identifying putative probe-gene pairs |
#-----------------------------------------------
# Collect nearby 20 genes for Sig.probes
nearGenes <- GetNearGenes(TRange=getProbeInfo(mee, probe=Sig.probes),
                          cores=detectCores(),
                          geneAnnot=getGeneInfo(mee))

# Identify significant probe-gene pairs
Hypo.pair <- get.pair(mee=mee,
                      probes=Sig.probes,
                      nearGenes=nearGenes,
                      permu.dir="./kirc/permu",
                      dir.out="./kirc/",
                      cores=detectCores(),
                      label= "hypo",
                      permu.size=10000,
                      Pe = 0.001)
```

```
Sig.probes.paired <- fetch.pair(pair=Hypo.pair,
                                probeInfo = getProbeInfo(mee),
                                geneInfo = getGeneInfo(mee))

#------------------------------------------------------------
# Step 3.3: Motif enrichment analysis on the selected probes |
#------------------------------------------------------------
enriched.motif <- get.enriched.motif(probes=Sig.probes.paired,
                                     dir.out="kirc", label="hypo",
                                     background.probes = probe$name)

#------------------------------------------------------------
# Step 3.4: Identifying regulatory TFs                      |
#------------------------------------------------------------
TF <- get.TFs(mee=mee,
              enriched.motif=enriched.motif,
              dir.out="kirc",
              cores=detectCores(), label= "hypo")
```

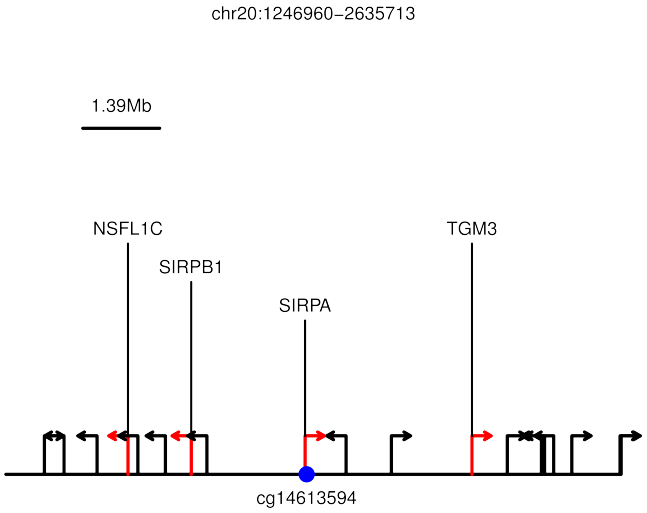
From this analysis it is possible to verify the relation between a probe and nearby genes. The result of this is show by the ELMER scatter plot.



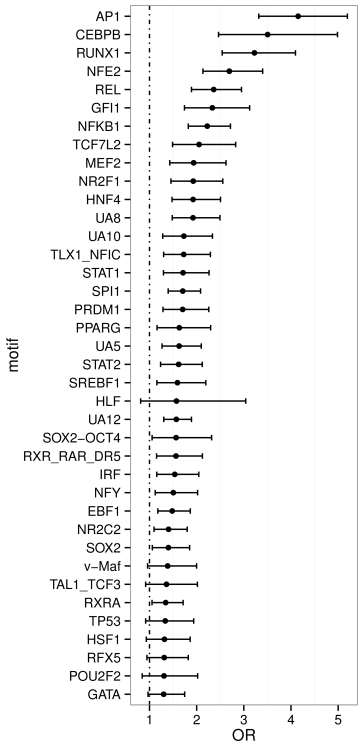
Each scatter plot showing the average methylation level of sites with the AP1 motif in all KIRC samples plotted against the expression of the transcription factor CEBPB and GFI1 respectively.

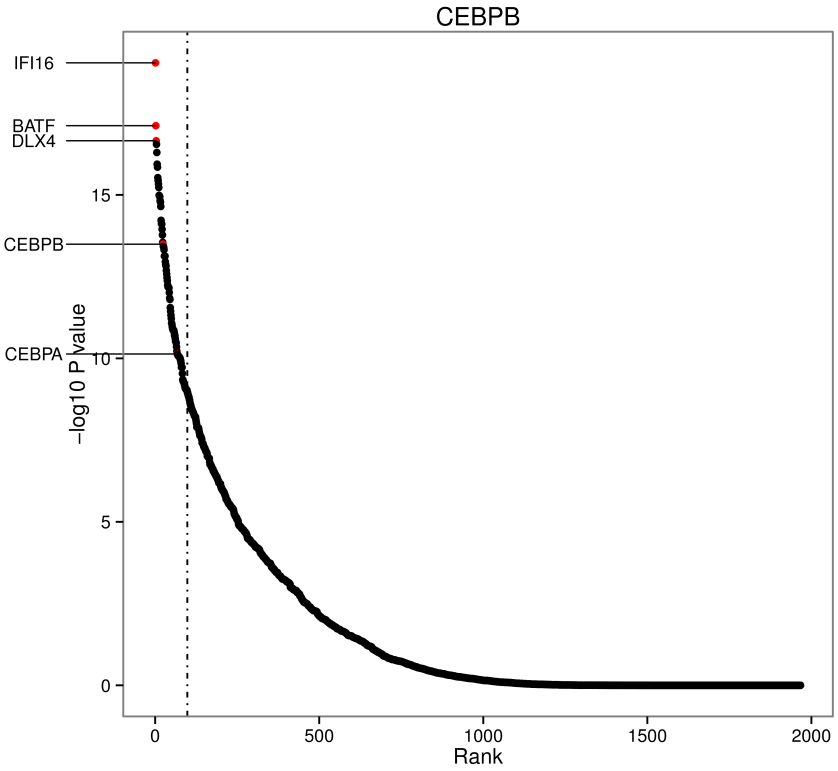


The schematic plot shows probe colored in blue and the location of nearby 20 genes, The genes significantly linked to the probe were shown in red.



The plot shows the odds ratio (x axis) for the selected motifs with OR above 1.3 and lower boundary of OR above 1.3. The range shows the 95% confidence interval for each Odds Ratio.


AP1
CEBPB
RUNX1
NFE2
REL
GFI1
NFKB1
TCF7L2
MEF2
NR2F1
HNF4
UA8
UA10
TLX1_NFIC
STAT1
SPI1
PRDM1
PPARG
UA5
STAT2
SREBF1
HLF
UA12
SOX2–OCT4
RXR_RAR_DR5
IRF
NFY
EBF1
NR2C2
SOX2
v–Maf
TAL1_TCF3
RXRA
TP53
HSF1
RFX5
POU2F2
GATA
motif
1
2
3
4
5
OR


CEBPB
IFI16
BATF
DLX4
CEBPB
CEBPA
15
10
5
0
−log10 P value
0
500
1000
1500
2000
Rank

Session Information

```
sessionInfo()
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.3 LTS
##
## 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=pt_BR.UTF-8
##  [9] LC_ADDRESS=pt_BR.UTF-8        LC_TELEPHONE=pt_BR.UTF-8
## [11] LC_MEASUREMENT=pt_BR.UTF-8    LC_IDENTIFICATION=pt_BR.UTF-8
##
## attached base packages:
##  [1] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] png_0.1-7                  SummarizedExperiment_0.3.8
##  [3] Biobase_2.29.1             GenomicRanges_1.21.25
##  [5] GenomeInfoDb_1.5.15        IRanges_2.3.21
##  [7] S4Vectors_0.7.17           BiocGenerics_0.15.6
##  [9] stringr_1.0.0              TCGAbiolinks_0.99.3
## [11] BiocStyle_1.7.7
##
## loaded via a namespace (and not attached):
##   [1] TH.data_1.0-6
##   [2] colorspace_1.2-6
##   [3] rjson_0.2.15
##   [4] hwriter_1.3.2
##   [5] modeltools_0.2-21
##   [6] futile.logger_1.4.1
##   [7] XVector_0.9.4
##   [8] roxygen2_4.1.1
##   [9] hexbin_1.27.1
##  [10] affyio_1.39.0
##  [11] AnnotationDbi_1.31.18
##  [12] mvtnorm_1.0-3
##  [13] coin_1.1-0
##  [14] codetools_0.2-14
##  [15] splines_3.2.2
##  [16] R.methodsS3_1.7.0
##  [17] doParallel_1.0.8
##  [18] DESeq_1.21.0
##  [19] geneplotter_1.47.0
##  [20] knitr_1.11
##  [21] heatmap.plus_1.3
##  [22] Rsamtools_1.21.17
##  [23] rJava_0.9-7
##  [24] annotate_1.47.4
```

```
##  [25] cluster_2.0.3
##  [26] R.oo_1.19.0
##  [27] supraHex_1.7.2
##  [28] graph_1.47.2
##  [29] httr_1.0.0
##  [30] assertthat_0.1
##  [31] Matrix_1.2-2
##  [32] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.1
##  [33] limma_3.25.15
##  [34] formatR_1.2.1
##  [35] htmltools_0.2.6
##  [36] tools_3.2.2
##  [37] igraph_1.0.1
##  [38] gtable_0.1.2
##  [39] affy_1.47.1
##  [40] dplyr_0.4.3
##  [41] reshape2_1.4.1
##  [42] ShortRead_1.27.5
##  [43] Rcpp_0.12.1
##  [44] Biostrings_2.37.8
##  [45] gdata_2.17.0
##  [46] ape_3.3
##  [47] preprocessCore_1.31.0
##  [48] nlme_3.1-122
##  [49] rtracklayer_1.29.27
##  [50] iterators_1.0.7
##  [51] xlsxjars_0.6.1
##  [52] proto_0.3-10
##  [53] rvest_0.2.0
##  [54] gtools_3.5.0
##  [55] devtools_1.9.1
##  [56] XML_3.98-1.3
##  [57] xlsx_0.5.7
##  [58] edgeR_3.11.2
##  [59] zlibbioc_1.15.0
##  [60] MASS_7.3-44
##  [61] zoo_1.7-12
##  [62] scales_0.3.0
##  [63] aroma.light_2.5.3
##  [64] BiocInstaller_1.19.14
##  [65] sandwich_2.3-3
##  [66] lambda.r_1.1.7
##  [67] RColorBrewer_1.1-2
##  [68] yaml_2.1.13
##  [69] memoise_0.2.1
##  [70] ggplot2_1.0.1
##  [71] downloader_0.4
##  [72] biomaRt_2.25.2
##  [73] reshape_0.8.5
##  [74] latticeExtra_0.6-26
##  [75] stringi_0.5-5
##  [76] RSQLite_1.0.0
##  [77] highr_0.5.1
```

```
##  [78] genefilter_1.51.0
##  [79] foreach_1.4.2
##  [80] GenomicFeatures_1.21.24
##  [81] caTools_1.17.1
##  [82] BiocParallel_1.3.52
##  [83] chron_2.3-47
##  [84] matrixStats_0.14.2
##  [85] bitops_1.0-6
##  [86] dnet_1.0.7
##  [87] evaluate_0.8
##  [88] lattice_0.20-33
##  [89] GenomicAlignments_1.5.17
##  [90] GGally_0.5.0
##  [91] plyr_1.8.3
##  [92] magrittr_1.5
##  [93] R6_2.1.1
##  [94] gplots_2.17.0
##  [95] multcomp_1.4-1
##  [96] DBI_0.3.1
##  [97] survival_2.38-3
##  [98] RCurl_1.95-4.7
##  [99] EDASeq_2.3.2
## [100] futile.options_1.0.0
## [101] KernSmooth_2.23-15
## [102] rmarkdown_0.8
## [103] data.table_1.9.6
## [104] Rgraphviz_2.13.0
## [105] ConsensusClusterPlus_1.23.0
## [106] digest_0.6.8
## [107] xtable_1.7-4
## [108] R.utils_2.1.0
## [109] munsell_0.4.2
```

References

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."

Yao, L., Shen, H., Laird, P. W., Farnham, P. J., & Berman, B. P. 2015. "Inferring Regulatory Element Landscapes and Transcription Factor Networks from Cancer Methylomes."