1. Introduction

TCGA2STAT enables users to easily download TCGA data directly into a format ready for statistical analysis in the R environment. The package imports and processes both molecular profiles and clinical data for more than 30 cancer types profiled with different high-throughput platforms (Appendix A), including microarray, next generation sequencing, methylation array, SNP array, and array-CGH. The package automatically combines molecular profiles and clinical data into a data matrix ready for supervised analysis. Further, the package provides users a simple function to merge data from multiple molecular profiles or platforms for integrated statistical analysis. Most importantly, all of the package functionality is available from one simple and user-friendly interface that does not require any domain-specific knowledge to utilize.

The data imported by this package is the version-stamped standardized data sets hosted and maintained by the Broad Institute GDAC Firehose. All imported data will be the latest version available from Firehose; typically, this is the aggregated Level 3 data from all sample batches and public clinical data as available from TCGA Data Portal.

This package depends on *XML* (Lang 2013) to parse the contents and paths while importing data from Firehose. Additionally, it requires *CNTools* (Zhang) to process the segmented CNV or CNA data into gene-level data.

This package relies on the system's capability to decompress the downloaded data which is a tar. Thus, users running R in Windows system that does not support `tar` will receive an error message.

2. Licensing and availability

The TCGA2STAT package and the underlying code in the package are distributed under the GPL2.0 license. Users of the package are free to use and redistribute the package but editing is prohibited. The TCGA2STAT package is available from The Comprehensive R Archive Network (CRAN) and the package's Github.

Usage of this package to import the TCGA data and employ the data in downstream analysis constitutes to agreement to TCGA data usage policy.

3. Installation

3.1. Install from CRAN

The TCGA2STAT package is available from The Comprehensive R Archive Network (CRAN).

3.2. Install from download package

To install TCGA2STAT from the package archive file obtained from the package's Github:

```
> install.packages("TCGA2STAT_1.0.tar.gz", repos = NULL, type = "source")
```

3.3. System compatibility

This package is intended to be used in Unix and Mac OS, where `tar` archive utility is supported natively. In Windows, if `tar` is not supported, users will be given an error message stating `Error: TAR is not installed in the system. Data unzip failed.`

---

4. Import OMICs profiles

TCGA2STAT enables users to download any type of data via one single, uniform interface: the `getTCGA` function. The type of data to be downloaded can be specified via three parameters:

- `disease` - acronym for the cancer type
- `data.type` - they type of omics-profiles
- `type` - the specific type of measurement for the omics-profile

For example, the following command will return the RPKM values of genes profiled using RNA-Sequencing from ovarian cancer patients.

```
> rnaseq.ov <- getTCGA(disease="OV", data.type="RNASeq", type="RPKM")
```

Upon running, a message will be appear indicating that the RNAseq data is being imported. And, upon successful execution of the command, the number of genes imported will be given in another message; for example:

```
RNAseq data will be imported! This may take some time!
19990 genes have been imported!
```

The returned object `rnaseq.ov` will be a list containing three elements:

- `dat` - The omics-profiles data matrix with genes in rows and patients in columns; in this example, this is a matrix of RPKM values.
- `clinical` - Clinical data matrix; in this example, NULL is returned as the clinical data is not specified (default).
- `merged.dat` - If clinical data is imported, a data matrix with both omics-profiles and clinical covariates (overall survival by default); patients are in the rows with the sample ID in the first column, followed by clinical covariates, and then the omics-profiles.

As an example, the internal structure of the object returned from the above command is as follows:

```
> str(rnaseq.ov)
List of 3
 $ dat       : num [1:19990, 1:299] 0.0539 0 0.4204 3.6155 5.3981 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:19990] "AADACL3" "AADACL4" "ABCA4" "ABCB10" ...
  .. ..$ : chr [1:299] "TCGA-04-1348-01A-01R-1565-13-2" "TCGA-04-1357-01A-01R-1565-13-2" "TCGA-04-1362-01A-01R
-1565-13-2" "TCGA-04-1364-01A-01R-1565-13-2" ...
 $ clinical  : NULL
 $ merged.dat: NULL
```

The data matrix of the omics profiles returned are of dimension *gene x patients*; the row names are gene symbols and the column names are samples ID (BCR code):

```
> head(rnaseq.ov$dat[, 1:3])
        TCGA-04-1348-01A-01R-1565-13-2 TCGA-04-1357-01A-01R-1565-13-2 TCGA-04-1362-01A-01R-1565-13-2
AADACL3                         0.0539                         0.2341                         0.0573
AADACL4                         0.0000                         0.0000                         0.0000
ABCA4                           0.4204                         0.1647                         0.2120
ABCB10                          3.6155                         7.1614                         3.1549
ABCD3                           5.3981                         3.8884                         7.2949
ABL2                            1.6614                         2.4809                         1.7997
```

To obtain default clinical data together with the omics profiles, specify `clinical=TRUE`. Consider the example below:

```
# Get the RPKM of genes along with all clinical data, and combine the RPKM with overall survival (OS)
> rnaseq_os.ov <- getTCGA(disease="OV", data.type="RNASeq", type="RPKM", clinical=TRUE)
```

```
# Look at the merged RPKM-OS data matrix
> dim(rnaseq_os.ov$merged.dat)
[1]   295 19993

> head(rnaseq_os.ov$merged.dat[,1:5])
           bcr status   OS AADACL3 AADACL4
1 TCGA-04-1348      1 1483  0.0539  0.0000
2 TCGA-04-1357      0   NA  0.2341  0.0000
3 TCGA-04-1362      1 1348  0.0573  0.0000
4 TCGA-04-1364      1 1024  0.0000  0.0242
5 TCGA-04-1365      0 2329  0.0177  0.0091
6 TCGA-04-1514      1 1720  0.0019  0.0000
```

To customize the clinical variables, specify `cvars` to the desired clinical variable. Consider the example below:

```
# Get the RPKM gene expression profiles along with all clinical data, and combine the expression with age
> rnaseq_age.ov <- getTCGA(disease="OV", data.type="RNASeq", type="RPKM", clinical=TRUE, cvars="yearstobirth")
```

```
# Look at the merged RPKM-age data matrix
> head(rnaseq_age.ov$merged.dat[,1:5])
           bcr YEARSTOBIRTH AADACL3 AADACL4  ABCA4
1 TCGA-04-1348           44  0.0539  0.0000 0.4204
2 TCGA-04-1357           52  0.2341  0.0000 0.1647
3 TCGA-04-1362           59  0.0573  0.0000 0.2120
4 TCGA-04-1364           61  0.0000  0.0242 0.0560
5 TCGA-04-1365           87  0.0177  0.0091 0.0757
6 TCGA-04-1514           45  0.0019  0.0000 0.1713
```

The list of acronyms for all available cancer types is given in Appendix A at the end of this vignette. A summary of all values permitted for both `data.type` and `type` is given in Appendix B. A list of values for `cvar` and their brief descriptions is given in Appendix C. However, since the clinical variables vary from one cancer type to another, the list in Appendix C is not the complete list. Users interested in a specific clinical covariate for the particular cancer type should check the availability of the clinical covariate from the column-name of `clinical` data matrix in the list returned by `getTCGA` function.

In the subsections below, we will give examples of how to download each type of omics data along with a short description of the data obtained.

4.1. Import gene expression

The two major platforms used to profile gene expression in TCGA cancer patients are microarray and RNA-Sequencing. Our package allows users to download data for both platforms if data is available.

4.1.1. mRNA-microarray gene expression data

Gene expression data profiled by microarray are available in different forms depending on the type of microarray employed. Specifically, three forms are available:

1. Agilent 244K Custom Gene Expression G4502A-07 (`type="G450"`)
2. Affymetrix Human Genome U133A 2.0 Array (`type="U133"`)
3. Affymetrix Human Exon 1.0 ST Array (`type="Huex"`)

The data imported from these platforms are level-3 gene-level expression data. Please also take note that only Agilent G450 is available for every cancer type; Affymetrix U133A and Human Exon ST Array are only available for certain cancers.

> Agilent 244K Custom Gene Expression G4502A

```
# Get Agilent G450 expression for ovarian cancer patients
> exp.ov <- getTCGA(disease="OV", data.type="mRNA_Array", type="G450")
```

```
# Look at the data
> dim(exp.ov$dat)
[1] 17814   597

> head(exp.ov$dat[,1:3])
        TCGA-01-0630-11A-01R-0363-07 TCGA-01-0631-11A-01R-0363-07 TCGA-01-0633-11A-01R-0363-07
ELMO2                       0.092000                    0.0081875                   -0.0953125
CREB3L1                     0.469500                    0.1473333                    0.0235000
RPS11                       0.965600                    0.7915000                    1.2695000
PNMA1                       0.815400                    0.8960000                    0.8314000
MMP2                       -0.969125                   -0.0452500                   -0.4230000
C10orf90                   -1.887200                   -1.9418000                   -1.9832000
>
```

This gene expression data from Agilent G450 should have ~17814 genes. The expression profiles are lowess-normalized log-ratio values that usually range from -10 to 10, as shown in the following histogram (Fig.1):

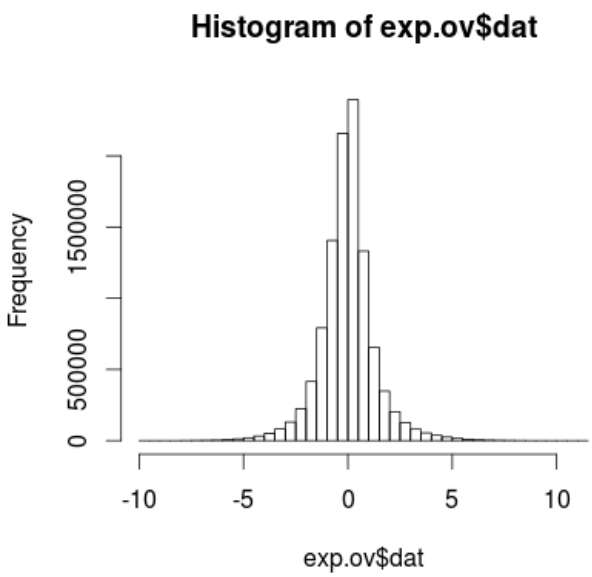

Fig.1. Histogram of Agilent G450 expression data.

Affymetrix Human Genome U133A 2.0 Array

```
# Get Affymetrix Human Genome U133A expression for ovarian cancer patients
> u133a.ov <- getTCGA(disease="OV", data.type="mRNA_Array", type="U133")
```

```
# Look at the data
> dim(u133a.ov$dat)
[1] 12042   593
> head(u133a.ov$dat[,1:3])
        TCGA-01-0628-11A-01R-0362-01 TCGA-01-0630-11A-01R-0362-01 TCGA-01-0631-11A-01R-0362-01
AACS                        5.772263                     6.428424                     5.574237
FSTL1                       8.220414                     8.496560                     8.710241
ELMO2                       4.933350                     5.430913                     4.705522
CREB3L1                     3.622720                     3.870238                     3.514235
RPS11                       9.970795                    10.387306                    10.124355
PNMA1                       7.959072                     8.356616                     8.132725
>
```

This data of ~12042 genes from Affymetrix HG-U133A is preprocessed and normalized using the RMA method, which results in profiles usually ranging from 2 to 14 as shown in the histogram below (Fig.2):

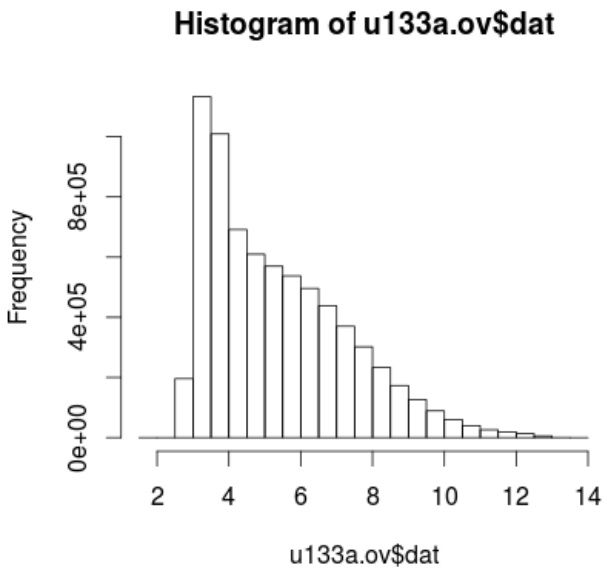

Fig.2. Histogram of Affymetrix HG-U133A expression data.

Affymetrix Human Exon 1.0 ST Array

Affymetrix Human Exon 1.0 ST Array is another common platform to profile gene level expression. Unlike HG-U133A where probes were designed to cover the entire genome, probes in Human Exon 1.0 ST Array are designed to cover exonic regions of the genes.

```
# Get Affymetrix Human Exon 1.0 ST Array expression for ovarian cancer patients
> huex.ov <- getTCGA(disease="OV", data.type="mRNA_Array", type="Huex")
```

```
# Look at the data
> dim(huex.ov$dat)
[1] 18632   594
> head(huex.ov$dat[,1:3])
         TCGA-01-0628-11A-01R-0361-03 TCGA-01-0630-11A-01R-0361-03 TCGA-01-0631-11A-01R-0361-03
C9orf152                     5.684061                     5.740597                     5.379095
ELMO2                        6.940721                     6.955458                     6.859219
RPS11                       12.112295                    11.603771                    11.435075
CREB3L1                      6.141857                     6.470617                     6.821231
PNMA1                        9.069748                     9.193657                     8.966602
MMP2                         7.993179                     8.551214                     9.651369
```

This level-3 data is quantile-normalized, consisting of ~18632 genes with values range from 2 to 14 as shown in the following histogram (Fig.3):

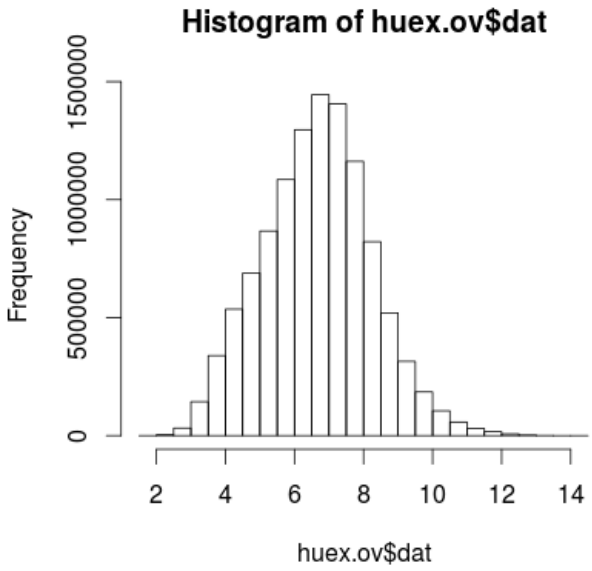

Fig.3. Histogram of Affymetrix Human Exon 1.0 ST Array data.

4.1.2. RNA-Seq gene expression data

TCGA also provides gene expression data measured via RNA-Sequencing technology, especially via Illumina Genome Analyzer (GA) or Illumina HiSeq 2000. For either platform, the data is available in two forms: RNA-Seq (preprocessed using the first pipeline), and RNA-SeqV2 (preprocessed using version two analysis).

RNA-Seq

Our package allows the user to download RNA-Seq data from both Illumina GA and Illumina HiSeq, depending on data availability. If gene expression is profiled via both platforms, only the data from HiSeq will be imported. In addition, users have the following options for which type of data to import: raw-counts (`type="count"`) or the normalized counts (`type="RPKM"`).

```
# Get raw count, RNA-Seq data for ovarian cancer patients
> counts.ov <- getTCGA(disease="OV", data.type="RNASeq")
# or equivalently
> counts.ov <- getTCGA(disease="OV", data.type="RNASeq", type="count")
```

```
# Look at the data:
> dim(counts.ov$dat)
[1] 19990   299
> head(counts.ov$dat[,1:3])
        TCGA-04-1348-01A-01R-1565-13 TCGA-04-1357-01A-01R-1565-13 TCGA-04-1362-01A-01R-1565-13
AADACL3                           36                           64                           30
AADACL4                            0                            0                            0
ABCA4                            575                           92                          226
```

```
ABCB10                          2308                          1864                          1572
ABCD3                           4890                          1437                          5160
ABL2                            3591                          2187                          3038
```

```
# Get normalized counts, from RNA-Seq for ovarian cancer patients
> rnaseq.ov <- getTCGA(disease="OV", data.type="RNASeq", type="RPKM")
```

```
# Look at the data:
> dim(rnaseq.ov$dat)
[1] 19990   299
> head(rnaseq.ov$dat[,1:3])
        TCGA-04-1348-01A-01R-1565-13-2 TCGA-04-1357-01A-01R-1565-13-2 TCGA-04-1362-01A-01R-1565-13-2
AADACL3                         0.0539                         0.2341                         0.0573
AADACL4                         0.0000                         0.0000                         0.0000
ABCA4                           0.4204                         0.1647                         0.2120
ABCB10                          3.6155                         7.1614                         3.1549
ABCD3                           5.3981                         3.8884                         7.2949
ABL2                            1.6614                         2.4809                         1.7997
```

RNA-SeqV2

For RNA-Sequencing data from the second analysis pipeline (RNASeqV2), our package provides importation of data from Illumina HiSeq only as this is the most common profiling method. The values returned are the RSEM values for the genes, which usually contains 20501 genes, with RSEM values ranging from 0 to 10^6.

```
# Get RSEM values, RNA-SeqV2 data for ovarian cancer patients
rsem.ov <- getTCGA(disease="OV", data.type="RNASeq2")
```

```
# Look at the data
> dim(rsem.ov$dat)
[1] 20501   307
> head(rsem.ov$dat[,1:3])
      TCGA-04-1348-01A-01R-1565-13 TCGA-04-1357-01A-01R-1565-13 TCGA-04-1362-01A-01R-1565-13
A1BG                       66.4695                      65.5664                      41.6412
A1CF                        0.0000                       0.0000                       0.3310
A2BP1                       0.2689                       0.6510                       4.3025
A2LD1                     221.5219                     141.2826                     265.8161
A2ML1                       7.5289                      54.6875                       5.6263
A2M                      5899.8279                    9384.4401                    3350.4207
```

4.2. Import miRNA expression

Similarly to gene expression, expression profiles of micro-RNAs (miRNA) are available from both the microarray platforms and the sequencing platforms. However, while the miRNA-Seq data is available for all cancer types, miRNA-array data is only available for three cancer types: GBM, GBMLGG, and OV.

4.2.1. miRNA-array data

The miRNA expression data available for download with our package is the level-3 Distance Weighted Discrimination (DWD) batch adjusted miRNA expression profiled from Agilent 8 x 15K Human miRNA-specific microarray (H-miRNA_8x15K) or Agilent Human miRNA Microarray Rel12.0 (H-miRNA_8x15Kv2 - for OV only).

```
# Get miRNA expression data via microarray for ovarian cancer patients
mir.ov <- getTCGA(disease="OV", data.type="miRNA_Array")
```

```
# Look at the data:
> str(mir.ov)
List of 3
 $ dat       : num [1:799, 1:594] 4.67 4.58 4.62 4.75 4.55 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:799] "ebv-miR-BART1-3p" "ebv-miR-BART1-5p" "ebv-miR-BART10" "ebv-miR-BART10*" ...
  .. ..$ : chr [1:594] "TCGA-01-0628-11A-01T-0364-07" "TCGA-01-0630-11A-01T-0364-07" "TCGA-01-0631-11A-01T-036
4-07" "TCGA-01-0633-11A-01T-0364-07" ...
 $ clinical  : NULL
 $ merged.dat: NULL
> head(mir.ov$dat[,1:3])
                TCGA-01-0628-11A-01T-0364-07 TCGA-01-0630-11A-01T-0364-07 TCGA-01-0631-11A-01T-0364-07
ebv-miR-BART1-3p                     4.668551                     4.640936                     5.170681
ebv-miR-BART1-5p                     4.579950                     4.696028                     4.855175
ebv-miR-BART10                       4.618877                     4.684807                     4.742815
```

```
ebv-miR-BART10*          4.747920          4.730715          4.734904
ebv-miR-BART11-3p        4.553927          4.648465          4.605977
ebv-miR-BART11-5p        4.698162          4.642823          4.711533
```

4.2.2. miRNA-Seq data

miRNA expression was profiled for most cancer types using Illumina HiSeq 2000 miRNA Sequencing. Similar to RNA-Seq, two types of miRNA-Seq data can be downloaded: raw counts (`type="count"`) or RPMMM (`type="rpmmm"`). Examples below show how to obtain each type of miRNA-Seq data:

```
# Get miRNA expression read counts for ovarian cancer patients
mirseq.ov <- getTCGA(disease="OV", data.type="miRNASeq")
# or equivalently
mirseq.ov <- getTCGA(disease="OV", data.type="miRNASeq", type="count")
#
# Get miRNA expression RPMMM values for ovarian cancer patients
mirseqn.ov <- getTCGA(disease="OV", data.type="miRNASeq", type="rpmmm")
```

```
> str(mirseq.ov)
List of 3
 $ dat       : int [1:705, 1:461] 124097 247488 124173 523763 133064 39686 31230 47 11523 1375 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:705] "hsa-let-7a-1" "hsa-let-7a-2" "hsa-let-7a-3" "hsa-let-7b" ...
  .. ..$ : chr [1:461] "TCGA-04-1331-01A-01R-1569-13" "TCGA-04-1332-01A-01R-1564-13" "TCGA-04-1336-01A-01R-156
4-13" "TCGA-04-1337-01A-01R-1564-13" ...
 $ clinical  : NULL
 $ merged.dat: NULL

> head(mirseq.ov$dat[,1:3])
             TCGA-04-1331-01A-01R-1569-13 TCGA-04-1332-01A-01R-1564-13 TCGA-04-1336-01A-01R-1564-13
hsa-let-7a-1                       124097                         9682                         3037
hsa-let-7a-2                       247488                        19634                         6274
hsa-let-7a-3                       124173                         9825                         3132
hsa-let-7b                         523763                        88962                        66321
hsa-let-7c                         133064                        22926                         5265
hsa-let-7d                          39686                         8955                        14718

> head(mirseqn.ov$dat[,1:3])
             TCGA-04-1331-01A-01R-1569-13-1 TCGA-04-1332-01A-01R-1564-13-1 TCGA-04-1336-01A-01R-1564-13-1
hsa-let-7a-1                      30713.962                      11269.88                       3953.866
hsa-let-7a-2                      61253.190                      22854.04                       8168.112
hsa-let-7a-3                      30732.772                      11436.33                       4077.546
hsa-let-7b                       129631.151                     103552.07                      86343.214
hsa-let-7c                        32933.291                      26685.94                       6854.496
hsa-let-7d                         9822.271                      10423.65                      19161.343
```

4.3. Import mutation data

The level-3 files of called mutations from the Broad Firehose are saved in Mutation Annotation Format (MAF). Each MAF file lists mutations found for the particular patient. Since each patient may have different mutations, the number of genes in each MAF file vary from patient to patient. Hence, the MAF files are not in a format ready for statistical analysis. To solve this, our package aggregates the obtained MAF files into a matrix of dimension *gene x patient*, with a value of one in cell(i,j) if a mutation is found in gene-i from patient-j, and a zero otherwise.

Our package permits two options for the mutation data:

- Only somatic non-silent mutations called (`type="somatic"`) which is the default. This will only return those mutations with “somatic”" as the mutation-status and not with “silent” as the variant-classification in the original MAF files.
- All mutations called, (`type="all"`).

Here are some example on downloading somatic mutation data:

```
# Get somatic non-silent mutations for ovarian cancer patients
mut.ov <- getTCGA(disease="OV", data.type="Mutation")
# or equivalently
mut.ov <- getTCGA(disease="OV", data.type="Mutation", type="somatic")
```

```
> str(mut.ov)
List of 3
 $ dat       : num [1:5821, 1:232] 1 1 1 1 1 1 1 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5821] "PPP2R5A" "TBC1D4" "ICAM1" "CPS1" ...
  .. ..$ : chr [1:232] "TCGA-24-1471" "TCGA-13-0899" "TCGA-23-1021" "TCGA-04-1347" ...
```

```
 $ clinical  : NULL
 $ merged.dat: NULL

> head(mut.ov$dat[,1:6])
        TCGA-24-1471 TCGA-13-0899 TCGA-23-1021 TCGA-04-1347 TCGA-10-0930 TCGA-23-1027
PPP2R5A            1            0            0            0            0            0
TBC1D4             1            0            0            0            0            0
ICAM1              1            0            0            0            0            0
CPS1               1            0            0            0            0            0
EP300              1            0            0            0            0            0
PEBP4              1            0            0            0            0            0
```

Note that in the example below, the number of genes is much larger if one is importing all mutations; but the data matrix is considerably sparser.

```
# Get all mutations called for ovarian cancer patients
> allmut.ov <- getTCGA(disease="OV", data.type="Mutation", type="all")
```

```
> str(allmut.ov)
List of 3
 $ dat       : num [1:10060, 1:316] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:10060] "HRNR" "LHX9" "PPP2R5A" "ZP4" ...
  .. ..$ : chr [1:316] "TCGA-24-1471" "TCGA-13-0899" "TCGA-23-1021" "TCGA-04-1347" ...
 $ clinical  : NULL
 $ merged.dat: NULL
```

4.4. Import methylation expression

DNA methylation profiles were obtained either via Illumina Infinium HumanMethylation27 BeadChip or the HumanMethylation450 BeadChip. The former platform probes for ~27,000 CpG sites (thus commonly known 27K) and the latter probes for ~450,000 CpG sites. Our package allows users to import either methylation data when available, via `type="27K"` (default) or `type="450K"`. For example:

```
# Get 27K methylation profiles for ovarian cancer patients
methyl.ov <- getTCGA(disease="OV", data.type="Methylation", type="27K")
# or equivalently
methyl.ov <- getTCGA(disease="OV", data.type="Methylation")

# Get 450K methylation profiles for ovarian cancer patients
methyl45.ov <- getTCGA(disease="OV", data.type="Methylation", type="450K")
```

Note that the matrix of the methylation profiles returned is NOT aggregated at the gene level. Each row in the data matrix (`dat`) is a probe from the methylation assay, which represents a CpG site. Since genes often contain more than one CpG site and each CpG site can differ significantly in the methylation level, gene level aggregation is less desirable. Hence, our package returns the probe-level data.

```
# Look at the data
> str(methyl.ov)
List of 4
 $ dat       : num [1:27578, 1:612] 0.7994 0.339 0.0281 0.6012 NA ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:27578] "cg00000292" "cg00002426" "cg00003994" "cg00005847" ...
  .. ..$ : chr [1:612] "TCGA-01-0628-11A-01D-0383-05" "TCGA-01-0630-11A-01D-0383-05" "TCGA-01-0631-11A-01D-038
3-05" "TCGA-01-0633-11A-01D-0383-05" ...
 $ cpgs      :'data.frame':  27578 obs. of  3 variables:
  ..$ Gene_Symbol       : chr [1:27578] "ATP2A1" "SLMAP" "MEOX2" "HOXD3" ...
  ..$ Chromosome        : chr [1:27578] "16" "3" "7" "2" ...
  ..$ Genomic_Coordinate: int [1:27578] 28890100 57743543 15725862 177029073 148822837 93862594 93813777 11980
953 89290921 0 ...
 $ clinical  : NULL
 $ merged.dat: NULL

> head(methyl.ov$dat[,1:3])
           TCGA-01-0628-11A-01D-0383-05 TCGA-01-0630-11A-01D-0383-05 TCGA-01-0631-11A-01D-0383-05
cg00000292                   0.79940858                   0.62039417                   0.91148841
cg00002426                   0.33900444                   0.18030460                   0.29385049
cg00003994                   0.02811930                   0.03607298                   0.06159123
cg00005847                   0.60116497                   0.64955777                   0.47394908
cg00006414                           NA                           NA                           NA
cg00007981                   0.01881682                   0.01803597                   0.02904295

> head(methyl.ov$cpgs)
            Gene_Symbol Chromosome Genomic_Coordinate
cg00000292        ATP2A1         16           28890100
```

```
cg00002426          SLMAP          3          57743543
cg00003994          MEOX2          7          15725862
cg00005847          HOXD3          2         177029073
cg00006414 ZNF425;ZNF398          7         148822837
cg00007981          PANX1         11          93862594
```

As shown above, the returned list for methylation data has one additional element, `cpgs`, which contains the probe's genomic information. Each row of `cpgs` is a CpG site and contains gene symbols and genomic coordinates in columns; rows in `cpgs` are of the same number and order as the methylation profiles data `dat`.

Values in the methylation data are the beta values. They range from zero to one (as shown in the histogram in Fig.4 below), with values greater than 0.5 representing hypermethylation, values less than 0.5 representing hypomethylation.

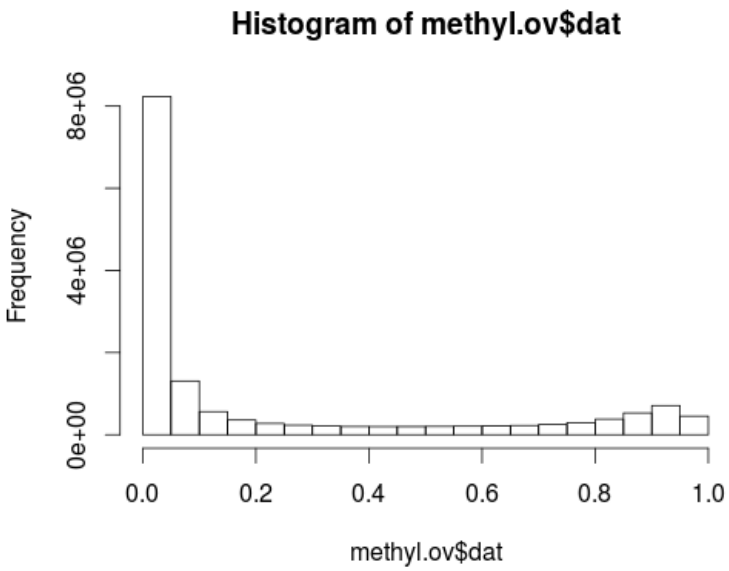

Fig.4. Histogram of methylation data.

4.5. Import copy number changes

Two major high-throughput platforms used to profile copy number changes are SNP-arrays and array comparative genomic hybridization (aCGH). Our package permits users to download both of these data types, although only SNP-array data is available for most cancer types. The copy number changes downloaded with our package are aggregated at the gene level; specifically, we merge the level-3 segments provided from TCGA (with reference genome build hg19) at the gene level. Values in the `dat` matrix of the returned list object are the log-ratio of copy number between samples to reference samples. Thus, a positive log-ratio indicates a copy number gain and negative value indicates a DNA copy number loss.

Two forms of copy number changes profiled using SNP-arrays (Affymetrix Genome-Wide Human SNP Array 6.0) are available: `data.type="CNA_SNP"` gives somatic copy number alterations (CNAs), which are somatic changes to chromosome structure that result in gains or losses in copies of sections of DNA; `data.type="CNV_SNP"` gives copy number variants (CNVs), which are the copy number ratios from somatic cells minus the ratios from germlines. Examples of these are given below:

```
# Get SNP array CNA for ovarian cancer patients
cnasnp <- getTCGA(disease="OV", data.type="CNA_SNP")
# or
# Get SNP array CNV for ovarian cancer patients
cnvsnp <- getTCGA(disease="OV", data.type="CNV_SNP")
```

```
# Look at the data
> head(cnvsnp$dat[,1:3])
        TCGA-01-0628-11A-01D-0356-01 TCGA-01-0630-11A-01D-0356-01 TCGA-01-0631-11A-01D-0356-01
A1BG                         -0.0043                       0.0002                      -0.0797
A2M                           0.0028                      -0.0003                       0.1000
A2MP1                         0.0028                      -0.0003                       0.1000
NAT1                          0.0047                       0.0050                      -0.0045
NAT2                          0.0047                       0.0050                      -0.0045
SERPINA3                      0.0024                       0.0031                       0.0501
```

Copy number changes profiled via array-CGH are measured with two different types of arrays: Agilent Human Genome CGH Microarray 244A (`type="244A"`) and Agilent Human Genome CGH Custom Microarray 2x415K (`type="415K"`). Unlike CNA/CNV profiled with SNP-arrays that are available for most cancer types, aCGH-profiled CNA data are available for only a few cancer types and most of them only for Custom Array 2x415. Examples of how to obtain this data are given below:

```
# Get aCGH CNA for ovarian cancer patients
cnacgh <- getTCGA(disease="OV", data.type="CNA_CGH")
```

```
# or equivalently
cnacgh <- getTCGA(disease="OV", data.type="CNA_CGH", type="415K")
```

```
# Look at the data
> names(cnacgh)
[1] "dat"        "clinical"   "merged.dat"
> head(cnacgh$dat[,1:3])
         TCGA-04-1353-01A-01D-1046-02 TCGA-04-1353-11B-01D-1046-02 TCGA-04-1369-01A-02D-1046-02
A1BG                          -0.7638                      -0.2282                      -0.5277
A2M                           -0.0284                      -0.0774                       0.0367
A2MP1                         -0.0284                      -0.0774                       0.0367
NAT1                          -0.8137                      -0.1329                      -1.1487
NAT2                          -0.8137                      -0.1329                      -1.1487
SERPINA3                      -0.2948                      -0.0776                      -0.1730
```

While merging the segments of copy number changes at gene-level, we filter out chromosome Y by default, which means the above three commands are equivalent to the following:

```
cnasnp <- getTCGA(disease="OV", data.type="CNA_SNP", filter="Y")
cnvsnp <- getTCGA(disease="OV", data.type="CNV_SNP", filter="Y")
cnacgh <- getTCGA(disease="OV", data.type="CNA_CGH", type="415K", filter="Y")
```

To include all genes, give an empty string to `filter`, for example:

```
cnacgh.gbm <- getTCGA(disease="GBM", data.type="CNA_CGH", type="415K", filter="")
```

Since there are certain gene symbols shared by two chromosomal locations in the sex chromosomes, we suffixed these genes by the chromosome to eliminate ambiguities. For example, in the `cnacgh.gbm$dat` object obtained from the above command, “SLC25A6_X” and “SLC25A6_Y” represent SLC25A6 genes in chromosome X and SLC25A6 in chromosome Y respectively.

Similarly, by providing a vector of characters with more than one chromosome to `filter`, the function will filter out any genes belonging to the specified chromosomes. For example, to get CNA autosomal chromosomes only (excluding sex chromosomes):

```
cnacgh <- getTCGA(disease="OV", data.type="CNA_CGH", filter=c("X", "Y"))
```

5. Miscellaneous Functions

5.1. Combine multiple OMICs profiles

Our package includes a function named `OMICSBind` that allows users to combine two matrices of omics-profiles for the same set of patients into a single object. In order to combine more than two types of omics-profiles together, users can call the `OMICSBind` function multiple times. In the example below, we show how to combine the RNA-Seq data from the second pipeline analysis, methylation profiles from 27K, and mutation data for ovarian cancer patients:

```
# Get the RNA-Seq, methylation, and mutation profiles for OV cancer patients
seq <- getTCGA(disease="OV", data.type="RNASeq2")
meth <- getTCGA(disease="OV", data.type="Methylation", type="27K")
mut <- getTCGA(disease="OV", data.type="Mutation", type="all")

# Now, merge the three OMICs-data into one R object
# step 1: merge RNA-Seq and mutation data
m1 <- OMICSBind(dat1 = seq$dat, dat2 = mut$dat)
# step 2: further concatenate the methylation data to the merged data-object
m2 <- OMICSBind(dat1 = m1$merged.data, dat2 = meth$dat)
```

```
# Look at the data
> dim(seq$dat)
[1] 20501   307
> dim(mut$dat)
[1] 10060   316
> dim(meth$dat)
[1] 27578   612
> dim(m1$merged.data)
[1] 30561   185
> dim(m2$merged.data)
[1] 58139   185
```

As shown above, the final merged data matrix `m2$merged.data` has 185 samples. These are the tumor samples of patients common in RNA-Seq, mutation, and methylation data. On the other hand, `m2$merged.data` has 58139 rows, which is the combination of 20501 genes from

RNA-Seq, 10060 genes from mutation, and 30561 CpG sites from methylation data. In order to distinguish gene symbols from different omics-profile, we affix the gene names with *d1* and *d2* in the merged data. Specifically, *d1* is affixed to gene names of the first omics-data input to `OMICSBind` function, *d2* is affixed to the names of the second input object:

```
> str(m1)
List of 3
 $ merged.data: num [1:30561, 1:185] 66.469 0 0.269 221.522 7.529 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:30561] "d1.A1BG" "d1.A1CF" "d1.A2BP1" "d1.A2LD1" ...
  .. ..$ : chr [1:185] "TCGA-04-1348" "TCGA-04-1357" "TCGA-04-1362" "TCGA-04-1364" ...
 $ X          : num [1:20501, 1:185] 66.469 0 0.269 221.522 7.529 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20501] "d1.A1BG" "d1.A1CF" "d1.A2BP1" "d1.A2LD1" ...
  .. ..$ : chr [1:185] "TCGA-04-1348" "TCGA-04-1357" "TCGA-04-1362" "TCGA-04-1364" ...
 $ Y          : num [1:10060, 1:185] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:10060] "d2.HRNR" "d2.LHX9" "d2.PPP2R5A" "d2.ZP4" ...
  .. ..$ : chr [1:185] "TCGA-04-1348" "TCGA-04-1357" "TCGA-04-1362" "TCGA-04-1364" ...
> str(m2)
List of 3
 $ merged.data: num [1:58139, 1:185] 66.469 0 0.269 221.522 7.529 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:58139] "d1.d1.A1BG" "d1.d1.A1CF" "d1.d1.A2BP1" "d1.d1.A2LD1" ...
  .. ..$ : chr [1:185] "TCGA-04-1348" "TCGA-04-1357" "TCGA-04-1362" "TCGA-04-1364" ...
 $ X          : num [1:30561, 1:185] 66.469 0 0.269 221.522 7.529 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:30561] "d1.d1.A1BG" "d1.d1.A1CF" "d1.d1.A2BP1" "d1.d1.A2LD1" ...
  .. ..$ : chr [1:185] "TCGA-04-1348" "TCGA-04-1357" "TCGA-04-1362" "TCGA-04-1364" ...
 $ Y          : num [1:27578, 1:185] 0.7991 0.0968 0.0218 0.8795 NA ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:27578] "d2.cg00000292" "d2.cg00002426" "d2.cg00003994" "d2.cg00005847" ...
  .. ..$ : chr [1:185] "TCGA-04-1348" "TCGA-04-1357" "TCGA-04-1362" "TCGA-04-1364" ...
```

5.2. Separating OMICs profiles by sample types

The TCGA OMICs profiles downloaded for a cancer type often include samples of different types, such as from tumor tissue, normal tissue, or other controls. The type of sample is encoded in the sample's BCR code. Users who are interested in studying only tumor samples, for example, would need to parse the codes to extract only these samples. To spare users the difficulties of decoding the sample ID (BCR code) to extract particular types of samples, our package includes a function, `SampleSplit` to accomplish this task in a user-friendly manner. Specifically, `SampleSplit` will take the omics data matrix as input and separate the matrix according to the sample types. The object returned is a list of three matrices corresponding to the omics profiles of primary solid tumor, recurrent solid tumor, and normal tissues/blood samples.

```
# Get the RNA-Seq V2 profiles for LUSC cancer patients
lusc.rnaseq2 <- getTCGA(disease="LUSC", data.type="RNASeq2")

# Split the OMICs data by sample types
lusc.rnaseq2.bytype <- SampleSplit(lusc.rnaseq2$dat)
```

```
# Look at the data
> str(lusc.rnaseq2.bytype)
List of 3
 $ primary.tumor  : num [1:20501, 1:501] 742 0 0 170 128 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20501] "A1BG" "A1CF" "A2BP1" "A2LD1" ...
  .. ..$ : chr [1:501] "TCGA-18-3406-01A-01R-0980-07" "TCGA-18-3407-01A-01R-0980-07" "TCGA-18-3408-01A-01R-098
0-07" "TCGA-18-3409-01A-01R-0980-07" ...
 $ recurrent.tumor: num[1:20501, 0 ]
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20501] "A1BG" "A1CF" "A2BP1" "A2LD1" ...
  .. ..$ : NULL
 $ normal         : num [1:20501, 1:51] 49.49 0 1.04 105.22 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20501] "A1BG" "A1CF" "A2BP1" "A2LD1" ...
  .. ..$ : chr [1:51] "TCGA-22-4593-11A-01R-1820-07" "TCGA-22-4609-11A-01R-2125-07" "TCGA-22-5471-11A-01R-1635
-07" "TCGA-22-5472-11A-11R-1635-07" ...
```

5.3. Getting matched tumor-normal OMICs profiles

Our package also includes a utility function, `TumorNormalMatch` to prepare the downloaded omics-profiles for pairwise analysis between tumor tissues and normal tissues/blood samples in a user-friendly manner. In the example below, we show how to first get the RNA-Seq data of analysis pipeline 2 from LUSC patients and then split the data into tumor tissues and normal tissues/blood samples for the same set of patients.

```
# Get RNA-SeqV2 data for LUSC patients
> lusc.rnaseq2 <- getTCGA(disease="LUSC", data.type="RNASeq2")
# tumor-normal matched profiles
> lusc.rnaseq2.tum.norm <- TumorNormalMatch(lusc.rnaseq2$dat)
```

This command will return a list of two omics-profile data matrices with dimension of *gene x patient*. Both matrices have the same dimension and order for both columns (patient) and rows (gene). Thus, enabling users to make direct pairwise comparisons.

```
# Look at the data
> str(lusc.rnaseq2.tum.norm)
List of 2
 $ primary.tumor: num [1:20501, 1:51] 94.67 0 3.46 38.73 209.32 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20501] "A1BG" "A1CF" "A2BP1" "A2LD1" ...
  .. ..$ : chr [1:51] "TCGA-22-4593" "TCGA-22-4609" "TCGA-22-5471" "TCGA-22-5472" ...
 $ normal       : num [1:20501, 1:51] 49.49 0 1.04 105.22 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20501] "A1BG" "A1CF" "A2BP1" "A2LD1" ...
  .. ..$ : chr [1:51] "TCGA-22-4593" "TCGA-22-4609" "TCGA-22-5471" "TCGA-22-5472" ...
```

6. Example: Canonical correlation analysis

The major objective of developing `TCGA2STAT` package is to provide statisticians with a set of user friendly tools to obtain and prepare omics-profiles from TCGA cancer patients for statistical analysis. In this example, we show how `TCGA2STAT` can enable easy and user-friendly integrated analysis of two types of TCGA omics profiles using canonical correlation analysis.

First, we import methylation profiles and RNA-Seq gene expression data for LUSC patients via the `getTCGA` function from our package:

```
# Get data
> lusc.methyl <- getTCGA(disease="LUSC", data.type="Methylation")
> lusc.rnaseq2 <- getTCGA(disease="LUSC", data.type="RNASeq2", clinical=TRUE)
```

To reduce computation time, we only include CpG sites and genes that are at the top 1% of variance across patients:

```
# Filter data
> met.var <- apply(lusc.methyl$dat, 1, var)
> met.data <- subset(lusc.methyl$dat, met.var >= quantile(met.var, 0.99, na.rm=T) & !is.na(met.var))

> rnaseq2.var <- apply(log10(1+lusc.rnaseq2$dat), 1, var)
> rnaseq.data <- subset(log10(1+lusc.rnaseq2$dat), rnaseq2.var >= quantile(rnaseq2.var, 0.99, na.rm=T) & !is.n
a(rnaseq2.var))
```

This will give us 234 CpG sites and 206 genes. Since canonical correlation analysis requires two different types of features measured on the same set of samples, we use the `OMICSBind` function in the package to get R objects with methylation profiles and gene expression profiles for the same set of LUSC patients:

```
> met.rnaseq2 <- OMICSBind(dat1 = rnaseq.data, dat2= met.data)
```

```
# Look at the data
> str(met.rnaseq2)
List of 3
 $ merged.data: num [1:440, 1:130] 2.112 1.654 0.252 2.874 1.391 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:440] "d1.A2ML1" "d1.ADAM23" "d1.ADCY8" "d1.ADH1B" ...
  .. ..$ : chr [1:130] "TCGA-18-3406" "TCGA-18-3407" "TCGA-18-3408" "TCGA-18-3409" ...
 $ X          : num [1:206, 1:130] 2.112 1.654 0.252 2.874 1.391 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:206] "d1.A2ML1" "d1.ADAM23" "d1.ADCY8" "d1.ADH1B" ...
  .. ..$ : chr [1:130] "TCGA-18-3406" "TCGA-18-3407" "TCGA-18-3408" "TCGA-18-3409" ...
 $ Y          : num [1:234, 1:130] 0.7561 0.6541 0.642 0.024 0.0258 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:234] "d2.cg00061629" "d2.cg00202711" "d2.cg00332153" "d2.cg00463577" ...
  .. ..$ : chr [1:130] "TCGA-18-3406" "TCGA-18-3407" "TCGA-18-3408" "TCGA-18-3409" ...
```

Now that the data is ready, we carry out canonical correlation analysis using the `CCA` package. We apply regularized canonical correlation analysis as the number of features for both sets of omics-profiles are larger than the number of samples.

```
library(CCA)
# run regularized-cca
> lusc.cc <- rcc(t(met.rnaseq2$X), t(met.rnaseq2$Y), 0.75025, 0.5005)
```

```
# compute the canonical loadings
> lusc.cc2 <- comput(t(met.rnaseq2$X), t(met.rnaseq2$Y), lusc.cc)
```

We can visualize the results of the analysis. In order to generate plots shown in our paper, we wrote plotting functions that can be found here: script. First, we visualize how the gene expression profiles correlate with the methylation profiles:

```
> plotNice2(lusc.cc2, d1=1, d2=2, XY="X", cex=0.7)
```

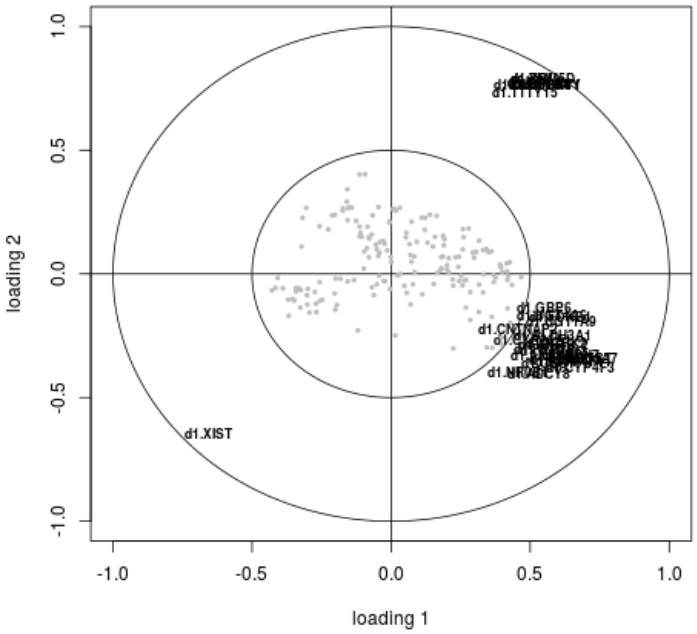

Fig.5. Canonical loadings of genes.

(Note - fill in XXX) From the plot in Figure 5, we can see that gene XIST is highly correlated with the set of CpG sites. Similarly, we can see how methylation profiles correlate with the gene expression profiles:

```
> plotNice2(lusc.cc2, d1=1, d2=2, XY="Y", cex=0.7)
```

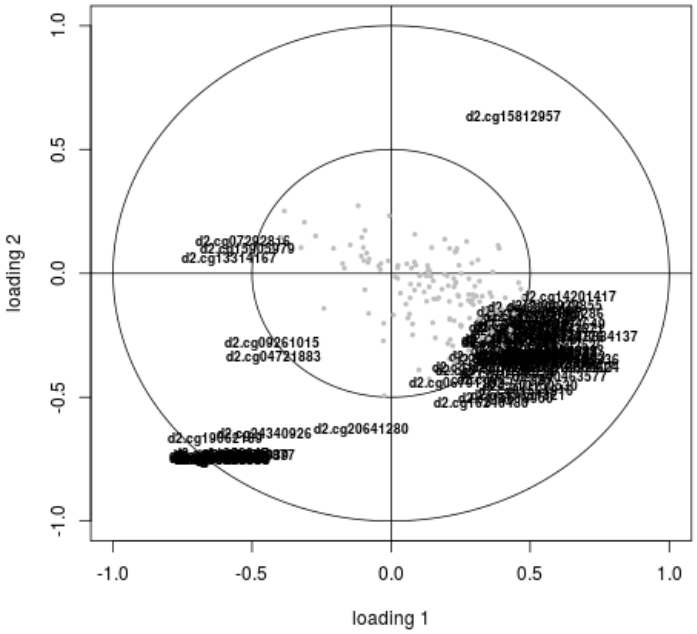

Fig.6. Canonical loadings of CpG sites.

With canonical correlation analysis, we can also investigate patterns among patients by projecting onto the canonical loadings. Here, we are interested in determining whether the estimated canonical loadings separate patients into distinct groups or cancer subtypes. First, we project onto the first two canonical loadings and visualize the patients below:

```
> plt.indiv(lusc.cc, d1=1, d2=2, ind.names=rep("+", nrow(t(met.rnaseq2$X))))
```

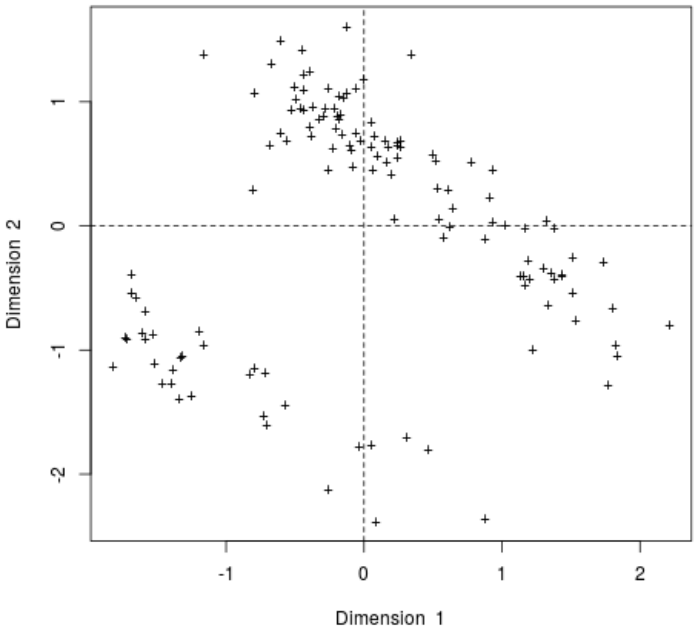

Fig.7. Patient clusters.

In Figure 7, we can see that patients are well-clustered into two groups. We re-plot this with two different colors in order to clearly see the two groups:

```
score <- lusc.cc2$xscores[,1:2]

# Parameters were estimated based on visual inspection
grp1 <- rownames(score)[(score[,1]<0 & score[,2]<0) | (score[,1]>0 & score[,2]< -1.5)]
grp2 <- setdiff(rownames(score), grp1)
grp <- ifelse(rownames(score) %in% grp1, 1, 2)

colors = colors()[c(153,220)]
plotNice2.indiv(lusc.cc2, d1=1, d2=2, cols=colors[grp], cex=2.5,pchs=c(20,20)[grp])
```

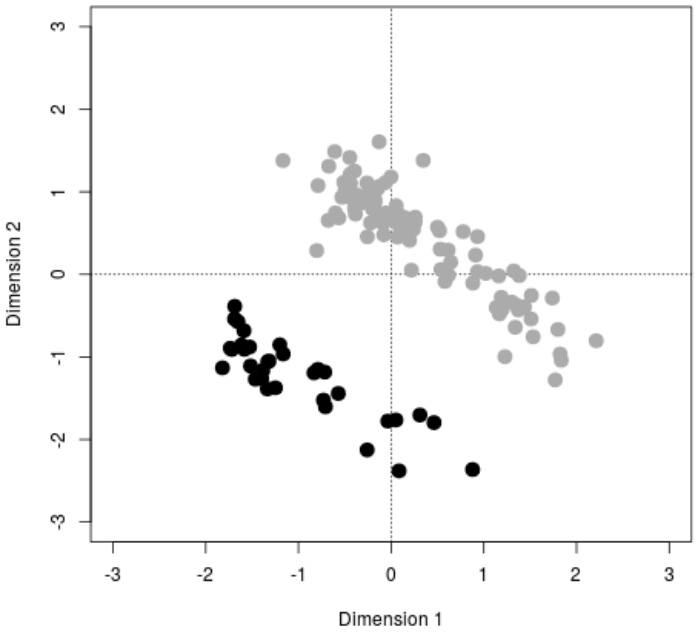

Fig.8. Nicer plot of patient clusters as in Fig.7.

Next, we investigate whether the two patient groups found by our analysis are clinically meaningful by seeing whether the groups exhibit differing survival outcomes. To this end, we perform a Kaplan-Meier analysis on the overall patient survival time to determine whether the survival probabilities between the two identified patient subgroups are significantly different:

```
# Get survival data from the imported RNASeq2 object
os.score <- lusc.rnaseq2$merged.dat[,1:3]
```

```
# prepare the data to make sure they match the two clusters from above
os <- os.score[,-1]
rownames(os) <- os.score[,1]
os <- os[rownames(lusc.cc2$xscores),]
plotdat <- data.frame(grp, os)
plotdat <- plotdat[order(plotdat[,1]),]

# KM-plot
mypamr.plotsurvival(plotdat[,1], plotdat[,3]/365, plotdat[,2], c(colors()[c(153,220)]), lty=c(2,1), lwd=c(2,2)
)
```

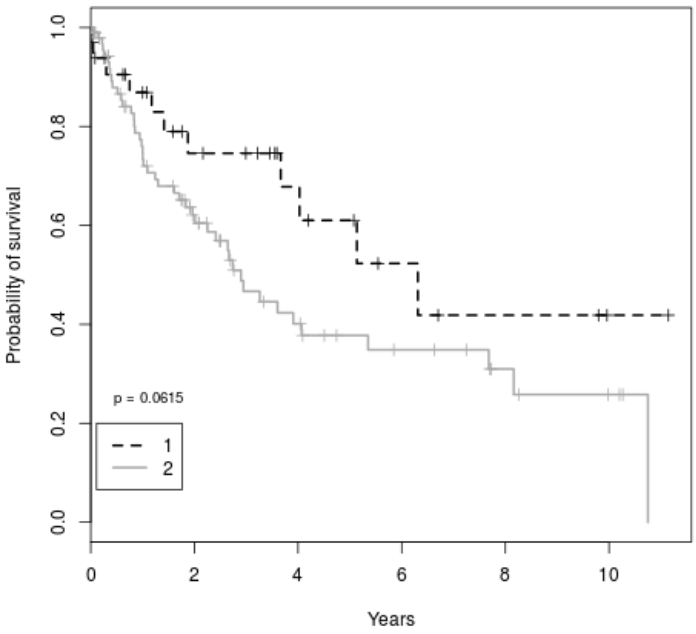

Fig.9. KM-plot of patients' overall survival.

From the survival plot in Figure 9, we can conclude that the canonical loadings of gene expression and methylation profiles separate patients into two clinically distinct subgroups: the '*black group*' is the low-risk group and the '*gray group*' is the high-risk group. The difference in overall survival between these two groups is border-line statistically significant (log-rank p-value < 0.06).

7. Appendix

7.1. Appendix A

Summary of cancer types and omics-profiles. Get File

7.2. Appendix B

Values permitted for `data.type` and `type` in `getTCGA` function. Get File

7.3. Appendix C

Summary of clinical variables common in most cancer types. Get File

8. Package Archive

10/09/2015 TCGA2STAT

10/19/2015 TCGA2STAT_1.2

References

Lang, Duncan Temple. 2013. *XML: Tools for Parsing and Generating XML Within R and S-Plus.* http://CRAN.R-project.org/package=XML.

Zhang, Jianhua. *CNTools: Convert Segment Data into a Region by Sample Matrix to Allow for Other High Level Computational Analyses.*