cbaf

Arman Shahrisa

2017-08-22

cbaf package facilitates working with high-throughput data stored on http://www.cbioportal.org/. The official Bioconductor package that is designed for obtaining data from cBioPortal in R, is `cgdsr`. To obtain data with this package, users have to pass a multistep procedure. Besides, the index of cancers and their subgroups changes frequently, which in turn, requires changing the R code. cbaf makes this procedure automated for **RNA-seq**, **microRNA-seq**, **microarray** and **methylation** data. In addition, comparing the genetic data across multiple cancer studies/cancer study subgroups becomes much faster and easier. The results are stored as excel file(s) and multiple heatmaps.

The package consists of six main functions: `availableData()`, `obtainOneStudy()`, `obtainMultipleStudies()`, `automatedStatistics()`, `heatmapOutput()` and `xlsxOutput()`.

There are two other functions, `processOneStudy()` and `processMultipleStudies()`, that combine multiple functions in order to automatically execute multiple functions following one another. Please use these two functions instead of exceting main functions one by one. This allows functions to work with higher efficiency.

availableData()

This function scans all the cancer studies to examine presence of *RNA-seq*, *microRNA-seq*, *microarray* and *methylation* data. It requires a name to label the output result.

```
availableData("name")
```

There is another argument that enables the function to save the result as an excel file. The default value is `excelFile = TRUE`. To prevent saving excel file, set `excelFile = FALSE`.

```
availableData("name", excelFile = FALSE)
```

At first, if `excelFile == TRUE`, the function starts looking into the working directory for an excel file with the given name. If it finds the file, the following message is printed, asking the user whether or not it should proceeds.

```
[1] "An excel file with the given name already exists in the working directory. Proceeding will cause t
Proceed anyway and overwrite the file? (yes/no):      no
```

If the answer is no, the function prints a message to inform the user and then stops processing.

```
[1] "--- Function 'availableData()' was skipped. ---"
```

If the user types yes or there is no excel file with the given name, the function continues with printing a message to inform the user it is scanning all the available cancer studies. Then, a progress bar will appear in the next line.

```
[1] "Cheching the available data for every cancer study"
  |                                                                      |   0%
```

The progress bar gradually proceeds until it reaches 100%.

```
[1] "Cheching the available data for every cancer study"
  |======================================================================| 100%
```

Upon finishing, the output is accessible with the given name as a variable. if `excelFile = TRUE`, an excel file containing the results is also generated in the present directory. Both variable and excel file contain

different columns: cancer_study_id, cancer_study_name, RNA.seq, microRNA.seq, microarray of mRNA, microarray of miRNA, methylation and description.

obtainOneStudy()

This function obtains and stores the supported data for at least one group of genes across multiple subgroups of a cancer study. It can check whether or not all genes are included in different subgroups of a cancer study and, if not, looks for the alternative gene names.

It requires at least four arguments:

- **genesList**, a list that contains at least one gene group. There is no limit for the number of gene groups, users can set as many as gene groups they desire.
- **submissionName**, a character string containing name of interest. It is used for naming the process.
- **studyName**, a character string showing the desired cancer name. It is an standard cancer study name that can be found on cbioportal.org, such as `Acute Myeloid Leukemia (TCGA, NEJM 2013)`.
- **desiredTechnique**, one of the five supported high-throughput studies: `RNA-seq`, `microRNA-Seq`, `microarray.mRNA`, `microarray.microRNA` or `methylation`.
- **desiredCaseList** a numeric vector that contains the index of desired cancer subgroups, assuming the user knows index of desired subgroups. If not, desiredCaseList must be set as 'none', function will show the available subgroups and ask the user to enter the desired ones during the process. The default value is 'none'.
- **validateGenes** a logical value that, if set to be 'TRUE', function will check each cancer subgroup to find whether or not every gene has a record. If the subgroup doesn't have a record for the specific gene, function checks for alternative gene names that cbioportal might use instead of the given gene name.

In the following example, *genes* consists of two gene groups K.demethylases and K.acetyltransferases, *submissionName* is `test`, *cancername* is `Breast Invasive Carcinoma (TCGA, Cell 2015)` and the *desiredTechnique* is `RNA-seq`.

```
genes <- list(K.demethylases = c("KDM1A", "KDM1B", "KDM2A"), K.acetyltransferases = c("CLOCK", "CREBBP"

obtainOneStudy(genes, "test", "Breast Invasive Carcinoma (TCGA, Cell 2015)", "RNA-seq")
```

Here is another example that shows how the index of desired subgroups can be used:

```
obtainOneStudy(genes, "test", "Breast Invasive Carcinoma (TCGA, Cell 2015)", "RNA-seq", desiredCaseList
```

First, obtainOneStudy() checks whether the requested data has already been obtained. If it can find it, function prints the following message and then stops further processing.

```
[1] "--- Function 'obtainOneStudy()' was skipped: the requested data already exist ---"
```

If there is a change in function input data, e.g. gene names, or the requested data has not been obtained, assuming `desired.case.list = "none"`, all subgroups of the requested cancer study appear on console, asking the user to choose the index of desired subgroups:

```
[1] "Please enter the numeric index of desired case list(s) for Breast Invasive Carcinoma (TCGA, Cell 2
 [1] "1. All Complete Tumors"                                "2. All Tumors"
 [3] "3. ER+ breast tumors"                                  "4. ER- breast tumors"
 [5] "5. Her2-positive breast tumors"                        "6. Invasive Ductal Cancer (Luminal A)"
 [7] "7. Invasive Ductal Cancer (PAM50 Basal-like)"          "8. Invasive Ductal Cancer (PAM50 Her2-enri
 [9] "9. Invasive Ductal Cancer (PAM50 Luminal B)"           "10. Invasive Lobular Cancer"
[11] "11. Invasive Lobular Cancer (Luminal A)"               "12. Mixed IDC/ILC breast cancer"
[13] "13. Other histologies breast cancer"                   "14. Sequenced Tumors"
[15] "15. TCGA Freeze 2015"                                  "16. Triple-negative breast tumors"
```

```
[17] "17. Tumor Samples with CNA data"                          "18. Tumor Samples with methylation data (
[19] "19. Tumor Samples with methylation data (HM450)"          "20. Tumor Samples with mRNA data (Agilent
[21] "21. Tumor Samples with mRNA data (RNA Seq V2)"            "22. Tumor Samples with RPPA data"
[23] "23. Tumor Samples with sequencing and CNA data"

Enter the numeric index(es):                  2,3,4,5
```

Then, it starts to get the data and informs the user with a progress bar.

```
[1] "*** Obtaining the requested data for test ***"
|======================================================================| 100%
```

Output is stored as a *BiocFileCache* object, which its name is combination of `bfc_` and *submissionName*. Accordingly, in our example the BiocFileCache name is `bfc_test`.

obtainMultipleStudies()

This function obtains and stores the supported data for at least one group of genes across multiple cancer studies. It can check whether or not all genes are included in each cancer study and, if not, it looks for the alternative gene names.

It requires at least four arguments:

- **genes**, a list that contains at least one group of genes. There is no limit for the number of gene groups, users can set as many as gene groups they desire.
- **submissionName**, a character string containing name of interest. It is used for naming the process.
- **cancernames**, a character vector or a matrix possessing names of desired cancer studies. The character vector contains standard cancer names that can be found on cbioportal.org, such as `Acute Myeloid Leukemia (TCGA, NEJM 2013)`. Alternatively, a matrix can be used if user prefers user-defined cancer names. In this case, the first column of matrix comprises the standard cancer names while the second column must contain the desired cancer names.
- **desiredTechnique**, one of the five supported high-throughput studies: `RNA-seq`, `microRNA-Seq`, `microarray.mRNA`, `microarray.microRNA` or `methylation`.

Function also contains two other options:

- **cancerCode**, if `TRUE`, will force the function to use the standard abbreviated cancer names instead of complete cancer names. For example, `laml_tcga_pub` is the shortened name for `Acute Myeloid Leukemia (TCGA, NEJM 2013)`.
- **validateGenes** , if `TRUE`, causes the function to check all cancer studies to find which genes from the input data are available. In addition, function checks for alternative gene names that cbioportal might use instead of the given gene name.

In the following example, *genes* consists of two gene groups K.demethylases and K.acetyltransferases, *submissionName* is `test`, *cancername* has complete name of five cancer studies and the desired high-throughput study is `RNA-seq`.

```
genes <- list(K.demethylases = c("KDM1A", "KDM1B", "KDM2A"), K.acetyltransferases = c("CLOCK", "CREBBP"


cancernames <- c("Acute Myeloid Leukemia (TCGA, Provisional)", "Adrenocortical Carcinoma (TCGA, Provisi


obntainMultipleStudies(genes, "test2", cancernames, "RNA-seq")
```

The following code shows how to create a matrix with the desired names for `cancernames`:

```
cancernames <- matrix(c("Acute Myeloid Leukemia (TCGA, Provisional)", "acute myeloid leukemia", "Adreno
```

An example of how more options for `processMultipleStudies()` can be altered:

```
ObtainMultipleStudies(genes, "test2", cancernames, "RNA-seq", cancerCode = TRUE, validateGenes = FALSE)
```

Function starts by checking whether or not the requested data has already been obtained and, if the requested data exists, it prints the following message and then prevents further processing.

```
[1] "--- Function 'obtainMultipleStudies()' was skipped: the requested data already exist ---"
```

If there is a change in function input data, e.g. gene names, or the requested data has not been obtained, it proceeds by getting the data and informs the user with a progress bar.

```
[1] "*** Obtaining the requested data for test2 ***"
|======================================================================| 100%
```

The output is stored as a BiocFileCache object, which its name is combination of `bfc_` and *submissionName*. Accordingly, in our example the BiocFileCache name is `bfc_test2`.

automatedStatistics()

The function calculates the statistics of the data obtained by `obtainOneStudy()` or `obtainMultipleStudies()` functions. Based on user's preference, these statistics can include *frequency percentage*, *frequency ratio*, *mean value* and *median value* of samples greater than specific value. Furthermore, it can look for the genes that comprise the highest values in each cancer and list the top 5 genes for *frequency percentage*, *mean value* and *median value*.

It requires at least two arguments:

- **submissionName**, a character string containing name of interest. It is used for naming the process and should be the same as submissionName for either of `obtainOneStudy()` or `obtainMultipleStudies()` functions.
- **obtainedDataType**, a character string that identifies the type of input data produced by the previous function. Two options are available: `single study` for `obtainOneStudy()` and `multiple studies` for `obtainMultipleStudies()`. The function uses `obtainedDataType` and `submissionName` to construct the name of the BiocFileCach object and then finds the appropriate data inside it. Default value is `multiple studies`.

Function also contains four other options:

- **calculate**, a character vector that contains the desired statistical procedures. Default input is `c("frequencyPercentage", "frequencyRatio", "meanValue", "medianValue")`. This will tell the function to compute the followings:
  - *frequencyPercentage*, which is the percentage of samples having the value greather than specific cutoff divided by the total sample size for every study / study subgroup
  - *frequency ratio*, which shows the number of selected samples divided by the total number of samples that give the frequency percentage. It shows the selected and total sample sizes.
  - *Mean Value*, which contains mean value of selected samples for each study.
  - *Median Value*, which shows the median value of selected samples for every study.
- **topGenes**, a logical value that, if set as TRUE, causes the function to create three data.frame that contain the five top genes for each cancer. To get all the three data.frames, *frequencyPercentage*, *meanValue* and *median* must have been included for **calculate**.
- **cutoff**, a number used to limit samples to those that are greater than this number (cutoff). The default value for methylation data is 0.6 while gene expression studies use default value of 2. For methylation

studies, it is *observed/expected ratio*, for the rest, it is *z-score*. To change the cutoff to any desired number, change the option to `cutoff = desiredNumber` in which desiredNumber is the number of interest.

- **round**, a logical value that forces the function to round all the calculated values to two decimal places. The default value is `TRUE`.

In the following example, *submissionName* is `test`, and the *obtainedDataType* is `multiple studies`.

```
automatedStatistics("test", obtainedDataType = "multiple studies")
```

The following is the same as previous but excludes *mean value* and *median value*. Note that top genes for these two statistics will also be skipped.

```
automatedStatistics("test", obtainedDataType = "multiple studies", calculate = c("frequencyPercentage",
```

Function starts by checking whether or not output data for the previous function exists. If not, an error message will appear:

```
Error: Please run one of the obtainSingleStudy() or obtainMultipleStudies() functions first
```

If it can find the desired data but one of the two previous function (either `obtainOneStudy()` or `obtainMultipleStudies()`) was skipped before, function then looks for its own output. If the requested output already exists, function will stop further processing, printing the following message:

```
[1] "--- Function 'automatedStatistics()' was skipped: the requested data already exist ---"
```

If there is a change in function input data, e.g. change in cutoff value, or the output result of automatedStatistics() does not exist, function continues by computing the requested statistics and informs the user with a progress bar.

```
[1] "*** Performing the requested statistical analyses for test2 ***"
|======================================================================| 100%
```

The output is stored as a new section of the previous BiocFileCache object which is created by one of the two former functions (`obtainOneStudy()` or `obtainMultipleStudies()`).

heatmapOutput()

This function prepares heatmap for *frequency percentage*, *mean value* and *median value* data provided by `automatedStatistics()` function. Heatmaps for every gene group are stored in separate folder.

It requires at least one arguments:

- **submissionName**, a character string containing name of interest. It is used for naming the process and should be the same as submissionName for either of `obtainOneStudy()` or `obtainMultipleStudies()` functions.

Function also contains thirteen other options:

- **shortenStudyNames** a logical value that causes the function to remove the last part of cancer names aiming to shorten them. The removed segment usually contains the name of scientific group that has conducted the experiment.
- **genelimit** if large number of genes exist in at least one gene group, this option can be used to limit the number of genes that are shown on heatmap. For instance, `genelimit=50` will limit the heatmap to 50 genes that show the most variation across multiple study / study subgroups. The default value is `none`.
- **resolution** This option can be used to adjust the resolution of the output heatmaps as 'dot per inch'. The defalut resolution is 600.
- **RowCex** a number that specifies letter size in heatmap row names.

- **ColCex** a number that specifies letter size in heatmap column names.
- **heatmapMargines** a numeric vector that is used to set heatmap margins. The default value is `heatmapMargines=c(15,07)`.
- **angleForYaxisNames** a number that determines the angle with which the studies/study subgroups names are shown on heatmaps. The default value is 45 degree.
- **heatmapColor** a character string that defines heatmap color. The default value is "RdBu". "redgreen" is also a popular color in genomic studies. To see the rest of colors, please type `library(RColorBrewer)` and then `display.brewer.all()`.
- **reverseColor** a logical value that reverses the color gradient for heatmap(s).
- **transposedHeatmap** a logical value that transposes heatmap rows to columns and vice versa.
- **simplify** a logical value that tells the function whether or not to change values under *simplifictionCuttoff* to zero. The purpose behind this option is to facilitate seeing the candidate genes. Therefore, it is not suited for publications.
- **simplifictionCuttoff** a logical value that, if `simplify.visulization = TRUE`, needs to be set as a desired cutoff for *simplify.visulization*. It has the same unit as *cutoff*.
- **genesToDrop** a character vector. Gene names within this vector will be omitted from heatmap.

In the following example, *submissionName* is `test`.

```
heatmapOutput(submissionName = "test")
```

The following is the same as previous but uses more options.

```
heatmapOutput("test", shortenStudyNames = TRUE, heatmapMargines = c(13,5), heatmapColor = "redgreen", g
```

Function starts by checking whether or not output data of the previous function exists. If not, an error message will appear:

```
Error: Please run one of the obtainSingleStudy() or obtainMultipleStudies() functions and then the auto
```

Otherwise, it continues by preparing and storing the requested heatmaps and informs the user with a progress bar.

```
[1] "*** Preparing the requested heatmaps for test ***"
|======================================================================| 100%
```

If there is no change in the options or the requested heatmaps already exist, it avoids storing the heatmaps. The number of skipped heatmaps is then printed:

```
"--- 1 out of 1 heatmaps was skipped: It already exists. ---"
```

xlsxOutput()

This function exports the output of `automatedStatistics()` and the *gene validation* result of one of the `obtainOneStudy()` or `obtainMultipleStudies()` functions as an excel file. For every gene group, an excel file will be generated and stored in the same folder as heatmaps.

It requires one argument:

- **submissionName**, a character string containing name of interest. It is used for naming the process and should be the same as submissionName for either of `obtainOneStudy()` or `obtainMultipleStudies()` functions.

In the following example, *submissionName* is `test`.

```
xlsxOutput("test")
```

Function starts by checking whether or not output data of the previous function exists. If not, an error message will appear:

```
Error: Please run one of the obtainSingleStudy() or obtainMultipleStudies() functions and then the auto
```

Otherwise, it continues by preparing and storing the requested heatmaps and informs the user with a progress bar.

```
[1] "*** Preparing the requested excel file(s) for test ***"
|======================================================================| 100%
```

If there is no change in the input parameters or the requested excel files already exist, it avoids rewriting them. The number of skipped excel files is then printed:

```
"--- 1 out of 1 excel file was skipped: It already exists. ---"
```

processOneStudy()

This function combines four of the mentioned functions for the ease of use. It is recommended that users only use this parent function to obtain and process gene data across multiple subsections of a cancer study so that child functions work with maximum efficiency. `processOneStudy()` uses the following functions:

- **obtainOneStudy()**
- **automatedStatistics()**
- **heatmapOutput()**
- **xlsxOutput()**

The function code along with all available options is as follows:

```
processOneStudy(genesList, submissionName, studyName, desiredTechnique, desiredCaseList = FALSE, valida
```

To get more information about the function options, please refer to the child function to whom they correspond, for example `genesList` lies within `obtainMultipleStudies()` function. The following is an example showing how this function can be used:

```
genes <- list(K.demethylases = c("KDM1A", "KDM1B", "KDM2A", "KDM2B", "KDM3A", "KDM3B", "JMJD1C", "KDM4A

processOneStudy(genes, "test", "Breast Invasive Carcinoma (TCGA, Cell 2015)", "RNA-seq", desiredCaseLis
```

The output excel files and heatmaps are stored in separate folders for every gene group. Ultimately, all the folders are located inside another folder, which it's name is the combination of *submissionName* and "output for single study", for example in our example it is: "test output for single study".

processMultipleStudies()

This function combines four of the mentioned above functions for the ease of use. It is recommended that users only use this parent function to obtain and process gene data across multiple cancer studies for maximum efficiency. `processMultipleStudies()` uses the following functions:

- **obtainMultipleStudies()**
- **automatedStatistics()**
- **heatmapOutput()**
- **xlsxOutput()**

The function code along with all available options is as follows:

```
processMultipleStudies(genesList, submissionName, studiesNames, desiredTechnique, cancerCode = FALSE, va
```

To obtain more information about the function options, please refer to the child function to whom they correspond, for example `genesList` lies within `ontainOneStudy()` function. The following example shows how this function can be used:

```
genes <- list(K.demethylases = c("KDM1A", "KDM1B", "KDM2A", "KDM2B", "KDM3A", "KDM3B", "JMJD1C", "KDM4A"

studies <- c("Acute Myeloid Leukemia (TCGA, Provisional)", "Adrenocortical Carcinoma (TCGA, Provisional)

processMultipleStudies(genes, "test2", studies, "RNA-seq", calculate = c("frequencyPercentage", "freque
```

The output excel file and heatmaps are stored in separate folders for every gene group. Ultimately, all the folders are located inside another folder, which it's name that is the combination of *submissionName* and "output for multiple studies", for example "test output for multiple studies".