

# CRISPRcleanR: An R package for unsupervised identification and correction of gene independent cell responses to CRISPR-cas9 targeting

Francesco Iorio, [fi1@sanger.ac.uk](mailto:fi1@sanger.ac.uk)

October 18, 2022

## 1 Quick start

### 1.1 Installation

First, you need to install and load the devtools package. You can do this from CRAN. Invoke R and then type:

```
install.packages("devtools")  
library(devtools)
```

Secondly, install CRISPRcleanR with the following command:

```
install_github("francescojm/CRISPRcleanR")
```

### 1.2 FASTQ files alignment and creation of the count matrix

The sgRNA raw counts can be obtained directly with CRISPRcleanR starting from FASTQ or BAM files. The `ccr.FASTQ2counts` function takes as input a sgRNA library and a list of FASTQ files to return a count matrix that can be used as input for normalization step. In order to create the proper index for the alignment the sgRNA library should include a `seq` field reporting the sequences of the guides.

**IMPORTANT:** the alignment process, depending on the computer used, could take 30' or more to generate library index requested for the alignment.

Load CRISPRcleanR.

```

library(CRISPRcleanR)

## Loading required package: stringr
## Loading required package: DNACopy
## Loading required package: pROC
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
## Loading required package: withr
## Loading required package: pracma
## Loading required package: PRROC
## Loading required package: tools
##
## Attaching package: 'tools'
## The following object is masked from 'package:withr':
##
##     makevars_user
## Loading required package: BiocManager
## Bioconductor version 3.15 (BiocManager 1.30.18), R 4.2.1 (2022-06-23)
## Loading required package: ShortRead
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following object is masked from 'package:pROC':
##
##     var
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get,
##     grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
## Loading required package: BiocParallel
## Loading required package: Biostrings
## Loading required package: S4Vectors
## Loading required package: stats4

```

```

##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
## Loading required package: IRanges
## Loading required package: XVector
## Loading required package: GenomeInfoDb
##
## Attaching package: 'Biostrings'
## The following object is masked from 'package:base':
##
##     strsplit
## Loading required package: Rsamtools
## Loading required package: GenomicRanges
## Loading required package: GenomicAlignments
## Loading required package: SummarizedExperiment
## Loading required package: MatrixGenerics
## Loading required package: matrixStats
##
## Attaching package: 'MatrixGenerics'
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAugsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAugsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname)".

```

```
##
## Attaching package: 'Biobase'
## The following object is masked from 'package:MatrixGenerics':
##
##      rowMedians
## The following objects are masked from 'package:matrixStats':
##
##      anyMissing, rowMedians
## Loading required package: Rsubread
## Loading required package: Rqc
## Loading required package: ggplot2
## Loading required package: jsonlite
```

```
## load built-in library
data(KY_Library_v1.0)

## locate count file
fileList <- file.path(
  system.file("extdata", package = "CRISPRcleanR"),
  c("test_plasmid.fq.gz", "test_sample1.fq.gz", "test_sample2.fq.gz")
)

## Run the alignment and extract the raw counts
counts <- ccr.FASTQ2counts(
  FASTQfileList = fileList,
  libraryAnnotation = KY_Library_v1.0,
  fastqc_plots = FALSE
)
```

```
head(counts)
```

```
##
## A1BG_CCDS12976.1_ex3_19:58862927-58862950:-_5-1 A1BG_CCDS12976.1_ex3_19:58862927-58862950
## A1BG_CCDS12976.1_ex4_19:58863655-58863678:+_5-2 A1BG_CCDS12976.1_ex4_19:58863655-58863678
## A1BG_CCDS12976.1_ex4_19:58863697-58863720:-_5-3 A1BG_CCDS12976.1_ex4_19:58863697-58863720
## A1BG_CCDS12976.1_ex4_19:58863866-58863889:+_5-4 A1BG_CCDS12976.1_ex4_19:58863866-58863889
## A1BG_CCDS12976.1_ex5_19:58864367-58864390:-_5-5 A1BG_CCDS12976.1_ex5_19:58864367-58864390
## A1CF_CCDS7241.1_ex6_10:52588014-52588037:-_5-1 A1CF_CCDS7241.1_ex6_10:52588014-52588037
##
## gene test_plasmid test_sample1
## A1BG_CCDS12976.1_ex3_19:58862927-58862950:-_5-1 A1BG 5 10
## A1BG_CCDS12976.1_ex4_19:58863655-58863678:+_5-2 A1BG 12 7
## A1BG_CCDS12976.1_ex4_19:58863697-58863720:-_5-3 A1BG 15 14
## A1BG_CCDS12976.1_ex4_19:58863866-58863889:+_5-4 A1BG 10 5
## A1BG_CCDS12976.1_ex5_19:58864367-58864390:-_5-5 A1BG 5 2
```

```
## A1CF_CCDS7241.1_ex6_10:52588014-52588037:-_5-1 A1CF 7
## test_sample2
## A1BG_CCDS12976.1_ex3_19:58862927-58862950:-_5-1 6
## A1BG_CCDS12976.1_ex4_19:58863655-58863678:+_5-2 9
## A1BG_CCDS12976.1_ex4_19:58863697-58863720:-_5-3 17
## A1BG_CCDS12976.1_ex4_19:58863866-58863889:+_5-4 5
## A1BG_CCDS12976.1_ex5_19:58864367-58864390:-_5-5 3
## A1CF_CCDS7241.1_ex6_10:52588014-52588037:-_5-1 6
```

**IMPORTANT:** the index creation and the alignment, performed using the `Rsubread` package ([1]), will create large binary files whose size will be slightly bigger than the FASTQ files used as input, please make sure to have enough space before starting the alignment.

If `MAGEck` ([2]) is installed the creation of the BAM files can be avoided specifying "mageck" in the `aligner` parameter. In this case the count matrix will be generated using the `MAGEck count` function.

If the FASTQ files were already aligned, the BAM files can be used directly to generate the counts matrix through the `ccr.BAM2counts` function. Please refer to the function documentation to select the most appropriate parameters based on the alignment strategy that was used to generate the BAM files.

### 1.3 Raw sgRNA count normalisation and computation of sgRNAs' log fold-changes

Load `CRISPRcleanR`.

```
library(CRISPRcleanR)
```

**Step 1:** Load your sgRNA library annotation. In this example we will use a built in data frame containing the annotation of the SANGER v1.0 library, introduced in [3]:

```
data(KY_Library_v1.0)
```

To use your own library annotation you will have to store it into a data frame with the same format of the `KY_Library_v1.0` (detailed in the corresponding entry of the reference manual of the `CRISPRcleanR` package).

**Step 2:** Store the path of the tsv file containing your sgRNAs' raw counts in a temporary variable. In this example we will use counts generated upon a CRISPR-Cas9 pooled drop-out screen (described in [4]) built in this package, for an example immortalised human cancer cell line (HT-29)

```
fn<-paste(system.file('extdata',package = 'CRISPRcleanR'),
           '/HT-29_counts.tsv',sep='')

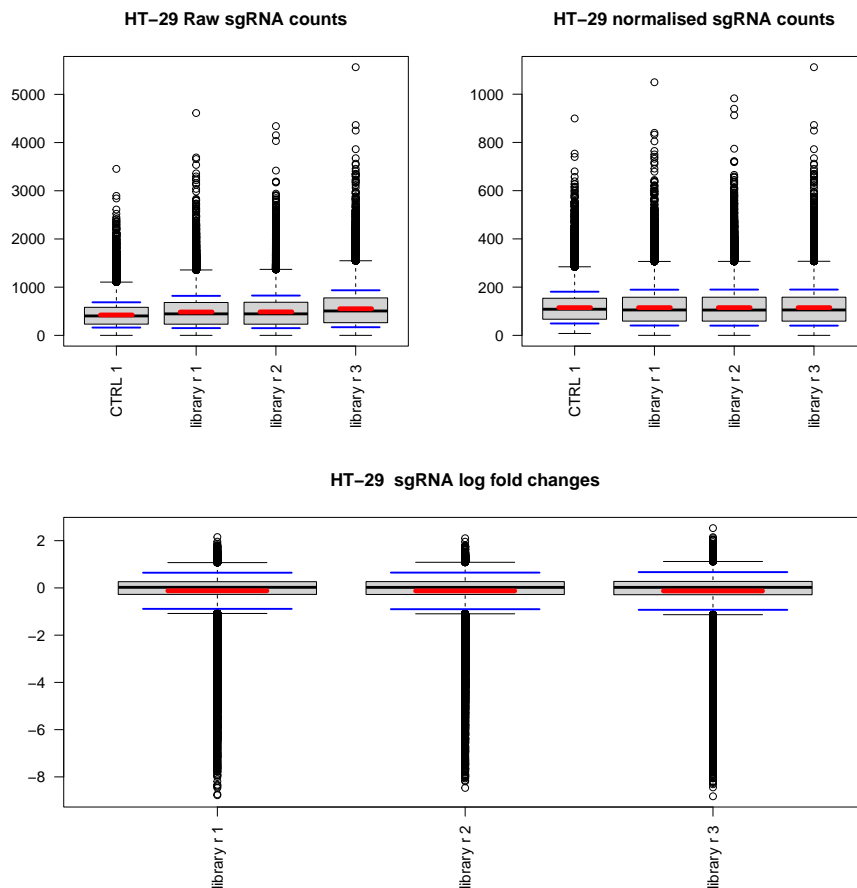
```

The tsv file with the sgRNAs' raw counts must be formatted as specified in the reference manual entry of the `ccr.NormfoldChanges` function.

**Step 3:** Perform normalisation of raw counts and compute sgRNAs' log fold-changes (in this example we will exclude sgRNAs with less than 30 reads in the plasmid sample).

```
normANDfcs<-ccr.NormfoldChanges(fn,
                                min_reads=30,
                                EXPname='HT-29',
                                libraryAnnotation=KY_Library_v1.0)

```



This function returns a list of two data frames containing, normalised counts and log fold-changes, respectively, and it saves them as Robjects in a user defined directory (specified by the parameter `outdir`, which is set to `'./'` by default).

```
head(normANDfcs$norm_counts)

##                                sgRNA gene ERS717283.plasmid
## 1 A1BG_CCDS12976.1_ex3_19:58862927-58862950:-_5-1 A1BG      292.14621
## 2 A1BG_CCDS12976.1_ex4_19:58863655-58863678:+_5-2 A1BG      151.02032
## 3 A1BG_CCDS12976.1_ex4_19:58863697-58863720:-_5-3 A1BG      209.08503
## 4 A1BG_CCDS12976.1_ex4_19:58863866-58863889:+_5-4 A1BG      110.40106
## 5 A1BG_CCDS12976.1_ex5_19:58864367-58864390:-_5-5 A1BG       95.81979
## 6 A1CF_CCDS7241.1_ex6_10:52588014-52588037:-_5-1 A1CF       60.92889
## HT29_c904R1 HT29_c904R2 HT29_c904R3
## 1      308.05192      354.89835      305.56806
## 2      145.38048      113.16912      166.47364
## 3      280.97793      203.70441      223.03071
## 4       80.31191       64.05372       78.74023
## 5       78.71932      123.80701      103.32157
## 6       47.77762       76.50232       59.75464

head(normANDfcs$logFCs)

##                                sgRNA gene HT29_c904R1 HT29_c904R2
## 1 A1BG_CCDS12976.1_ex3_19:58862927-58862950:-_5-1 A1BG  0.07635566  0.28027938
## 2 A1BG_CCDS12976.1_ex4_19:58863655-58863678:+_5-2 A1BG -0.05472442 -0.41467098
## 3 A1BG_CCDS12976.1_ex4_19:58863697-58863720:-_5-3 A1BG  0.42548611 -0.03752168
## 4 A1BG_CCDS12976.1_ex4_19:58863866-58863889:+_5-4 A1BG -0.45663336 -0.78070106
## 5 A1BG_CCDS12976.1_ex5_19:58864367-58864390:-_5-5 A1BG -0.28197989  0.36800353
## 6 A1CF_CCDS7241.1_ex6_10:52588014-52588037:-_5-1 A1CF -0.34756262  0.32598467
## HT29_c904R3
## 1  0.06469488
## 2  0.14010902
## 3  0.09293733
## 4 -0.48496821
## 5  0.10820208
## 6 -0.02784489
```

**IMPORTANT:** if there are control replicates in your sgRNAs count file their number must be specified by in the parameter `ncontrols` (equal to 1 by default) of the `ccr.NormfoldChanges` function.

## 1.4 Genome sorting of sgRNAs' log fold-changes and their correction for gene independent responses to CRISPR-Cas9 targeting

**Step 1:** Map genome-wide sgRNAs' log fold changes (averaged across replicates) on the genome, sorted according to the of their targeted gene on the chromosomes.

```
gwSortedFCs<-
  ccr.logFCs2chromPos(normANDfcs$logFCs,KY_Library_v1.0)
```

```
head(gwSortedFCs)
```

```
##                               CHR startp  endp  genes      avgFC
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1    1 871254 871277 SAMD11 -0.12965287
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2    1 874451 874474 SAMD11  0.09329615
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3    1 874487 874510 SAMD11  0.25286616
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4    1 874693 874716 SAMD11 -0.05128489
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5    1 876601 876624 SAMD11 -0.02110076
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1    1 887388 887411  NOC2L -1.27571756
##                               BP
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1 871265.5
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2 874462.5
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3 874498.5
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4 874704.5
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5 876612.5
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1 887399.5
```

**Step 2:** Identify and correct biased sgRNAs' log fold-changes putatively due to gene independent responses to CRISPR-Cas9 targeting, using the `ccr.GWclean` function. This function calls iteratively the `ccr.cleanChrm` function, which performs the correction on an individual chromosome). In this example we are using a completely unsupervised approach and correcting chromosomal segments of equal sgRNA log fold-changes if they include sgRNAs targeting at least 3 different genes, and without making any assumption on gene essentiality, nor knowing *a priori* the copy number status of the included genes [4].

```
correctedFCs<-ccr.GWclean(gwSortedFCs,display=TRUE,label='HT-29')
```

The corrected sgRNAs fold-changes are returned in a list (as a data frame), together with the annotation of the identified segments (in another data frame) and a vector of strings containing all the genome-sorted sgRNAs' identifiers.

```
head(correctedFCs$corrected_logFCs)
```

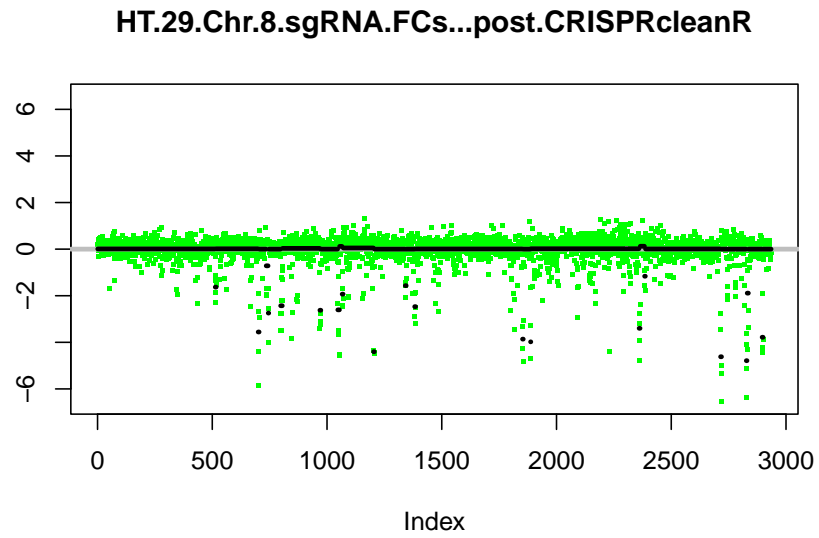
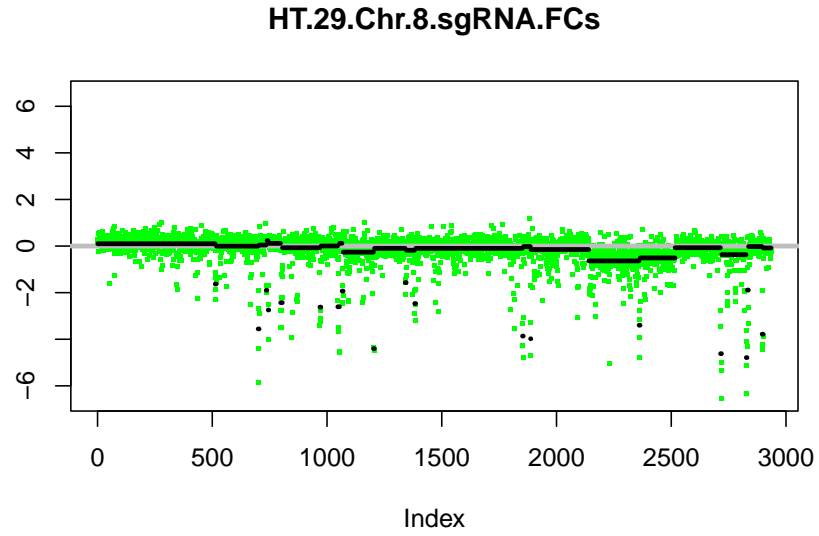
```
##                               CHR startp  endp  genes      avgFC
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1    1 871254 871277 SAMD11 -0.12965287
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2    1 874451 874474 SAMD11  0.09329615
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3    1 874487 874510 SAMD11  0.25286616
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4    1 874693 874716 SAMD11 -0.05128489
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5    1 876601 876624 SAMD11 -0.02110076
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1    1 887388 887411  NOC2L -1.27571756
##                               BP correction correctedFC
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1 871265.5          0 -0.12965287
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2 874462.5          0  0.09329615
```



```
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3 874498.5      0  0.25286616
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4 874704.5      0 -0.05128489
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5 876612.5      0 -0.02110076
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1  887399.5      0 -1.27571756
##                                     guideIdx
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1      1
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2      2
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3      3
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4      4
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5      5
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1      6
```

Details on how the data frame with the corrected sgRNAs' log fold-changes should be interpreted can be found in the entry of the `ccr.GWclean` function, of the package reference manual.

This function also produces one plot per chromosome, with segments of sgRNAs' equal log fold-change before and after the correction. An example of these plot is reported below: chromosome 8, in HT-29, showing a region containing *MYC*, which is highly biased toward consistent negative fold-changes.



### 1.5 Correcting sgRNAs' treatment counts for mean-variance modeling

In order to apply the inverse transformation described in [4], deriving corrected normalised sgRNAs' treatment counts from CRISPRcleanR corrected log fold-

changes, it is sufficient to run the function `ccr.correctCounts` as follows:

```
correctedCounts<-ccr.correctCounts('HT-29',
                                   normANDfcs$norm_counts,
                                   correctedFCs,
                                   KY_Library_v1.0,
                                   minTargetedGenes=3,
                                   OutDir='./')
```

Together with the plasmid counts, the corrected treatment counts can be used to compute depletion significance scores with a mean-variance modeling approach (such that implemented in MAGeCK[2]).

```
head(correctedCounts)
```

##	sgRNA	gene	ERS717283.plasmid
## 1	A1BG_CCDS12976.1_ex3_19:58862927-58862950:-_5-1	A1BG	292.14621
## 2	A1BG_CCDS12976.1_ex4_19:58863655-58863678:+_5-2	A1BG	151.02032
## 3	A1BG_CCDS12976.1_ex4_19:58863697-58863720:-_5-3	A1BG	209.08503
## 4	A1BG_CCDS12976.1_ex4_19:58863866-58863889:+_5-4	A1BG	110.40106
## 5	A1BG_CCDS12976.1_ex5_19:58864367-58864390:-_5-5	A1BG	95.81979
## 6	A1CF_CCDS7241.1_ex6_10:52588014-52588037:-_5-1	A1CF	60.92889
##	HT29_c904R1	HT29_c904R2	HT29_c904R3
## 1	309.77863	356.73522	307.28892
## 2	144.74469	112.89328	165.60212
## 3	280.34458	203.51866	222.73301
## 4	80.64680	64.52159	79.08797
## 5	78.22697	122.47062	102.36866
## 6	45.21299	71.83850	56.31473

This function also saves the correctedCounts as Rdata object at the location specified by the parameter `OutDir`. To run MAGeCK, using these corrected sgRNAs' counts you will need to save them as a tsv file, which will be used as input for MAGeCK. This can be done by using the following function, which also returns the path of the saved file.

```
ccr.PlainTsvFile(correctedCounts, fprefix = 'HT-29')

## [1] "./HT-29_sgRNA_count.tsv"
```

**IMPORTANT:** the corrected sgRNAs' counts are already normalised, therefore, when executing MAGeCK, the parameter `--norm-method` should be set to `none`.

## 2 Visualisation and assessment of Results

### 2.1 Classification performances of reference sets of genes (or sgRNAs) based on depletion log fold-changes

To perform a basic quality control assessment of your data it is possible to test the genome-wide profile of sgRNAs' depletion log fold-changes (logFCs) (or gene depletion logFCs, averaged across targeting sgRNAs) as a classifier of reference sets of core-fitness essential (CFE) and non-essential genes.

What you need for this is a named vector of sgRNAs (or gene) logFCs and two reference gene sets (respectively for positive and negative cases). In this example we make use of a precomputed essentiality profile from the builtin data object `EPLC.272HcorrectedFCs`. This is a list containing corrected sgRNAs log fold-changes and segment annotations for an example cell line (EPLC-272H), obtained using the `ccr.GWclean` function, as detailed in its reference manual entry. However the data frame containing the corrected log fold-changes, included in this list, reports also the original sgRNAs logFC (column `avgFC`), which will be used in this example.

```
data(EPLC.272HcorrectedFCs)
```

```
head(EPLC.272HcorrectedFCs$corrected_logFCs)
```

```
##                               CHR startp  endp  genes      avgFC
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1    1 871254 871277 SAMD11 -0.20295496
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2    1 874451 874474 SAMD11 -0.08917153
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3    1 874487 874510 SAMD11 -0.04417670
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4    1 874693 874716 SAMD11  0.30441537
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5    1 876601 876624 SAMD11 -0.11240079
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1    1 887388 887411 NOC2L -1.61370746
##                               BP correction correctedFC
## SAMD11_CCDS2.2_ex3_1:871254-871277:+_5-1 871265.5      0 -0.20295496
## SAMD11_CCDS2.2_ex4_1:874451-874474:-_5-2 874462.5      0 -0.08917153
## SAMD11_CCDS2.2_ex4_1:874487-874510:+_5-3 874498.5      0 -0.04417670
## SAMD11_CCDS2.2_ex5_1:874693-874716:+_5-4 874704.5      0  0.30441537
## SAMD11_CCDS2.2_ex6_1:876601-876624:-_5-5 876612.5      0 -0.11240079
## NOC2L_CCDS3.1_ex8_1:887388-887411:+_5-1  887399.5      0 -1.61370746
```

As reference gene sets we will use lists of CFE and non-essential genes assembled from multiple RNAi studies. These are used as classification template by the BAGEL algorithm to call gene depletion significance [5], and are included in the builtin data objects `BAGEL_essential` and `BAGEL_nonEssential`.

```
data(BAGEL_essential)
data(BAGEL_nonEssential)
```

```
head(BAGEL_essential)

## [1] "ACTL6A" "ACTR6" "ALYREF" "ANAPC4" "ANAPC5" "AP2S1"

head(BAGEL_nonEssential)

## [1] "ABCG8" "ACCSL" "ACTL7A" "ACTL7B" "ACTL9" "ACTRT1"
```

Finally, you will need the sgRNAs library annotation. In this case we will use the builtin object `KY_KY_Library_v1.0` (introduced in the previous section) [3]. As for the previous examples, to use a different library annotation you will have to store it in a data frame with the same format of the `KY_Library_v1.0` data frame (detailed in the corresponding entry of the reference manual of the `CRISPRcleanR` package).

```
data(KY_Library_v1.0)
```

We will start with an evaluation at the sgRNA level. As mentioned, the logFCs need to be stored a named vector:

```
FCs<-EPLC.272HcorrectedFCs$corrected_logFCs$avgFC
names(FCs)<-rownames(EPLC.272HcorrectedFCs$corrected_logFCs)
```

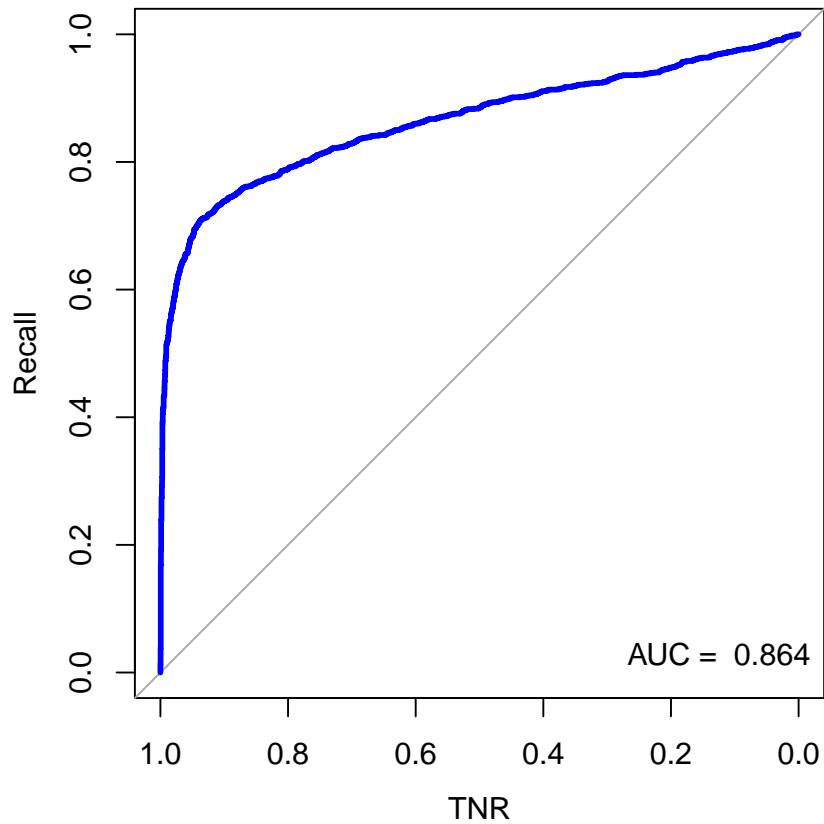
The `ccr.genes2sgRNAs` function can be used, as follows, to convert the reference sets of CFE and non-essential genes into sets of sgRNAs:

```
BAGEL_essential_sgRNAs<-
  ccr.genes2sgRNAs(KY_Library_v1.0,BAGEL_essential)
BAGEL_nonEssential_sgRNAs<-
  ccr.genes2sgRNAs(KY_Library_v1.0,BAGEL_nonEssential)
```

Following these calls, possible warning messages could appear informing you that some of the reference genes are not targeted by any sgRNAs in the considered library. This has no impact on the following steps and results.

Finally, to visualise the ROC curve quantifying the performances in classifying the considered reference sets based on their logFCs, it is sufficient to call:

```
sgRNA_level_ROC<-ccr.ROC_Curve(FCs,BAGEL_essential_sgRNAs,
                                BAGEL_nonEssential_sgRNAs)
```



To reperform the analysis at the gene level, you should first convert the profile of sgRNAs' logFCs into gene level summaries. The function `ccr.geneMeanFCs` performs this conversion by considering for each gene the average logFC across its targeting sgRNAs.

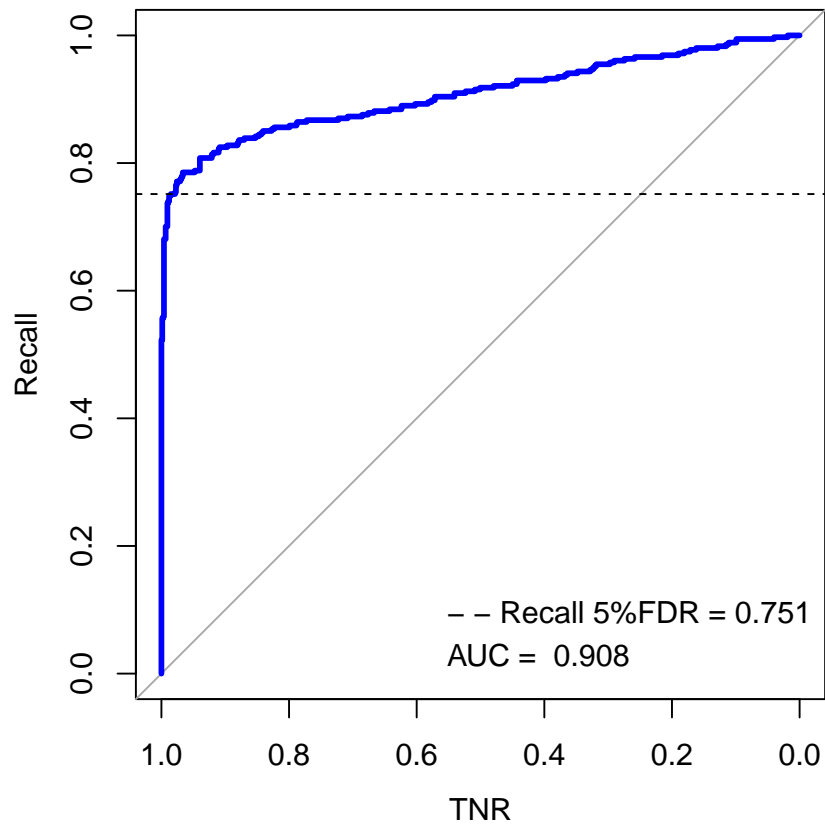
```
geneFCs<-ccr.geneMeanFCs(FCs,KY_Library_v1.0)
head(geneFCs)
```

##	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT
##	-0.2474235	-0.1550534	0.2190111	0.3736683	-0.5151889	-0.1698269

The following call reperforms the ROC analysis at the gene level and it also computes and shows Recall values at fixed False Discovery Rate (which is defined by the user and in this case is equal to 5%).

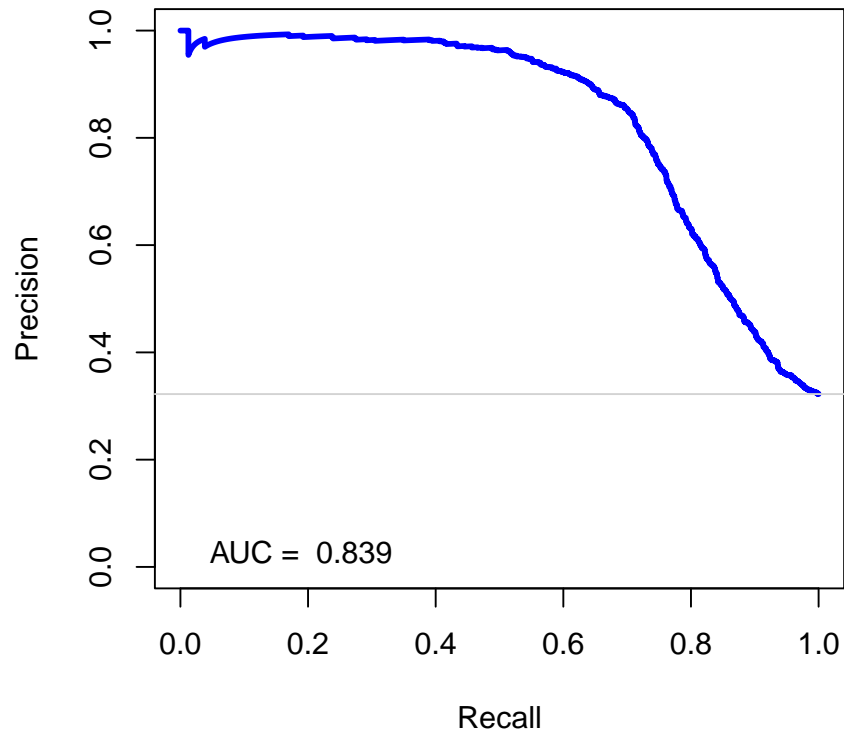
```
gene_level_ROC<-ccr.ROC_Curve(geneFCs,
                               BAGEL_essential,
```

```
BAGEL_nonEssential,  
FDRth = 0.05)
```



The same assessment can be performed at the level of Precision/Recall (PrRc) evaluation: to visualise the PrRc curve quantifying the performances in classifying the considered reference sets based on their logFCs, it is sufficient to call:

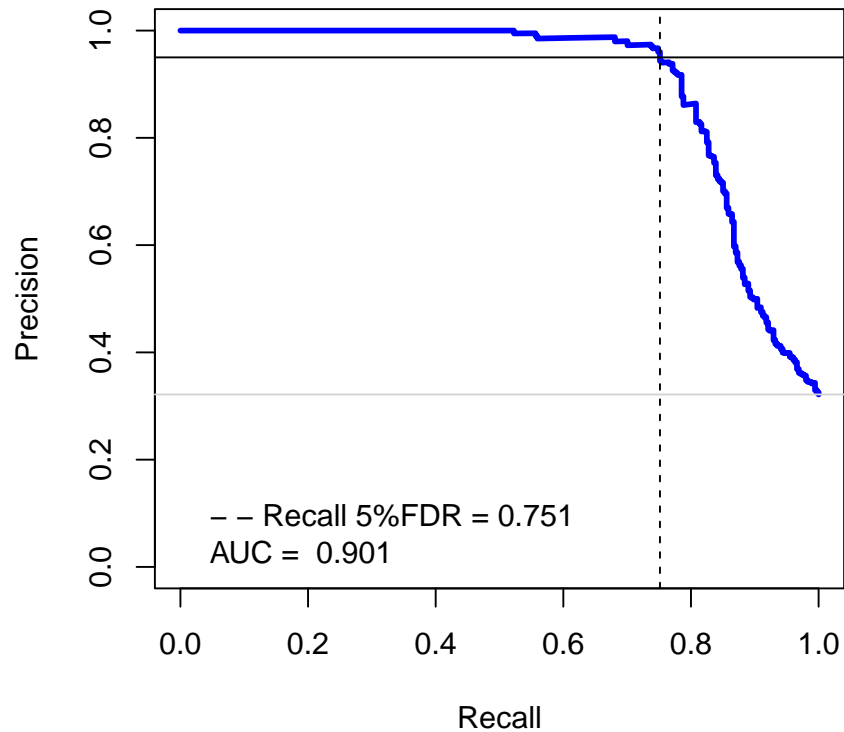
```
sgRNA_level_PrRc<-ccr.PrRc_Curve(FCs,BAGEL_essential_sgRNAs,  
BAGEL_nonEssential_sgRNAs)
```



As before, to reperform the analysis at the gene level, it is sufficient to call the following function, which also computes and shows Recall values at fixed False Discovery Rate (which is defined by the user and in this case is equal to 5%).

```
gene_level_PrRc<-ccr.PrRc_Curve(geneFCs,  
                                BAGEL_essential,  
                                BAGEL_nonEssential,  
                                FDRth = 0.05)
```





As can be seen above, when setting the parameter `FRDth` to a value different from `NULL` (its default value), this function also returns the log fold change threshold at which a classification FDR equal to the inputted value is achieved.

```

gene_level_PrRc$sigthreshold
## threshold
## -0.7683409

```

## 2.2 Depletion profile visualisation with genes signatures superimposed and recall computation

For another quick assessment of your data, it is possible to visually inspect enrichments of predefined sets of core-fitness essential genes near the top of the genome wide essentiality profiles (composed of sgRNA or gene depletion logFCs

ranked in increasing order), and to compute their classification recall at a fixed FDR (determined as detailed in the previous subsection).

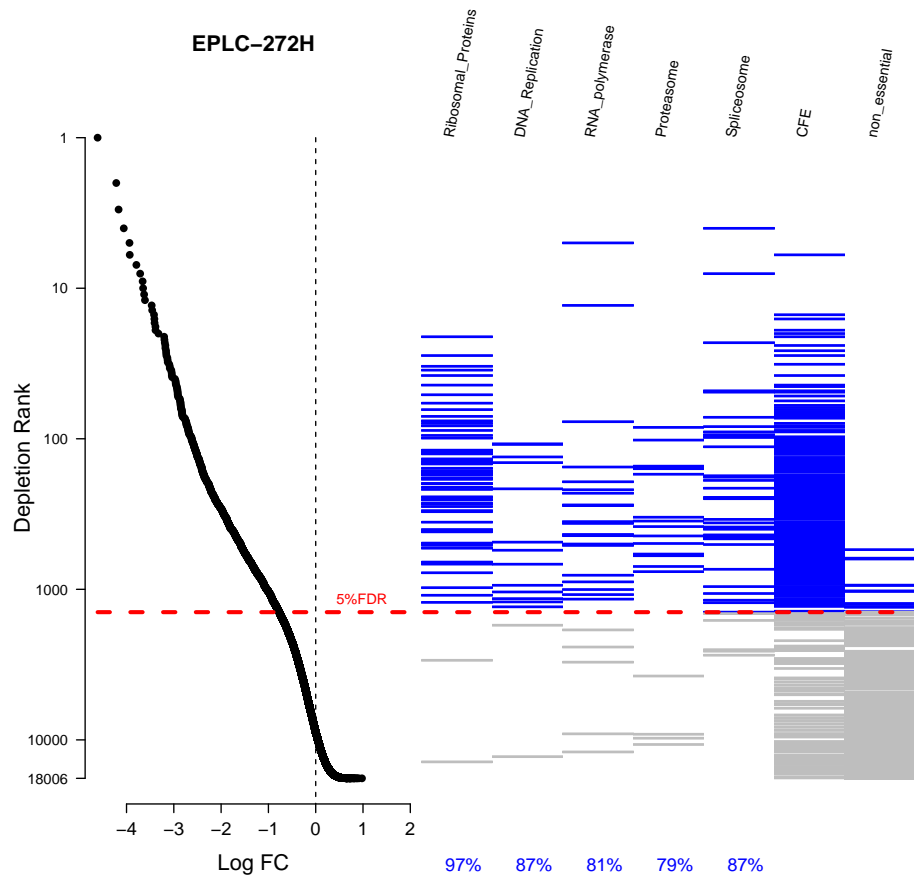
To this aim, in this example we will load additional sets of CFE genes assembled from MsigDB [6] as detailed in [4], and we will store all of them into a named list, as follows:

```
data(EssGenes.ribosomalProteins)
data(EssGenes.DNA_REPLICATION_cons)
data(EssGenes.KEGG_rna_polymerase)
data(EssGenes.PROTEASOME_cons)
data(EssGenes.SPLICEOSOME_cons)

SIGNATURES<-list(Ribosomal_Proteins=EssGenes.ribosomalProteins,
                 DNA_Replication = EssGenes.DNA_REPLICATION_cons,
                 RNA_polymerase = EssGenes.KEGG_rna_polymerase,
                 Proteasome = EssGenes.PROTEASOME_cons,
                 Spliceosome = EssGenes.SPLICEOSOME_cons,
                 CFE = BAGEL_essential,
                 non_essential = BAGEL_nonEssential)
```

Finally a visualisation of the gene essentiality profile with superimposed these signatures, can be created as follows:

```
Recall_scores<-ccr.VisDepAndSig(FCsprofile = geneFCs,
                               SIGNATURES = SIGNATURES,
                               TITLE = 'EPLC-272H',
                               pIs = 6,
                               nIs = 7)
```



**IMPORTANT:** When calling `ccr.VisDepAndSig` it is important to correctly specify the index position of the reference gene sets that are used as classification template to derive the FDR threshold, within the list of signatures. In this case the template sets are `BAGEL_essential` and `BAGEL_nonEssential`, which in the `SIGNATURE` list are in position 6 and 7, respectively (this must be specified in the `pIs` and `nIs` parameters of the `ccr.VisDepAndSig` function). This function also returns recall values at 5% FDR for all the inputted signatures.

Recall\_scores

## Ribosomal_Proteins	DNA_Replication	RNA_polymerase	Proteasome
## 0.96721311	0.86666667	0.80769231	0.78947368
## Spliceosome	CFE	non_essential	
## 0.86842105	0.75141243	0.01874163	

## 2.3 CRISPRcleanR correction assessment: Statistical tests

To evaluate the effect of the CRISPRcleanR correction on your data it is possible to inspect the logFCs' variations for sgRNAs targeting predefined sets of genes for statistically significant differences with respect to background pre/post CRISPRcleanR correction.

To this aim, in this example we will use the builtin data object `HT.29correctedFCs` containing corrected sgRNAs' logFCs and segment annotations for an example cell line (HT-29), obtained using the `ccr.GWclean` function, as detailed in its reference manual entry.

```
data(HT.29correctedFCs)
```

The function `ccr.perf_statTests` performs this analysis, saving pdf figures in a user defined location ('/' by default).

Particularly, this functions assess statistical difference respect to background population pre/post CRISPRcleanR correction of logFCs for sgRNAs targeting respectively:

- copy number (CN) deleted genes according to the GDSC1000 repository
- CN deleted genes (gistic score = -2) according to the CCLE repository
- non expressed genes (FPKM lower than 0.05)
- genes with gistic score = 1
- genes with gistic score = 2
- non expressed genes (FPKM lower than 0.05) with gistic score = 1
- non expressed genes (FPKM lower than 0.05) with gistic score = 2
- genes with minimal CN = 2, according to the GDSC1000
- genes with minimal CN = 4, according to the GDSC1000
- genes with minimal CN = 8, according to the GDSC1000
- genes with minimal CN = 10, according to the GDSC1000
- non expressed genes (FPKM lower than 0.05) with minimal CN = 2, according to the GDSC1000
- non expressed genes (FPKM lower than 0.05) with minimal CN = 4, according to the GDSC1000
- non expressed genes (FPKM lower than 0.05) with minimal CN = 8, according to the GDSC1000

- non expressed genes (FPKM lower than 0.05) with minimal CN = 10, according to the GDSC1000

The call should be as follows:

```
RES<-ccr.perf_statTests('HT-29',libraryAnnotation = KY_Library_v1.0,
                        correctedFCs = HT.29correctedFCs$corrected_logFCs,
                        GDSC.geneLevCNA = NULL,
                        CCLE.gisticCNA = NULL,
                        RNAseq.fpkms = NULL)

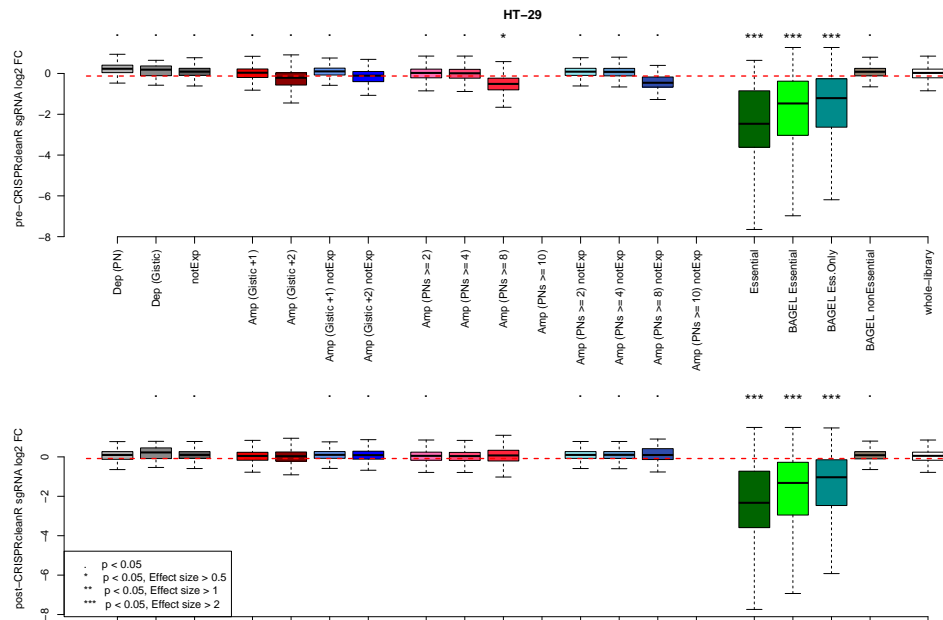
## [1] "Testing sgRNAs targeting: Dep (PN) genes"
## [1] "Testing sgRNAs targeting: Dep (Gistic) genes"
## [1] "Testing sgRNAs targeting: notExp genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +1) genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +2) genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +1) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +2) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 2) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 4) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 8) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 10) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 2) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 4) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 8) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 10) notExp genes"
## [1] "Testing sgRNAs targeting: Essential genes"
## [1] "Testing sgRNAs targeting: BAGEL Essential genes"
## [1] "Testing sgRNAs targeting: BAGEL Ess.Only genes"
## [1] "Testing sgRNAs targeting: BAGEL nonEssential genes"
```

This saves a pdf with the following figure in a user defined location ('./' by default).

Leaving the parameters `GDSC.geneLevCNA`, `CCLE.gisticCNA`, and `RNAseq.fpkms` to their default NULL value will force this function to use the respective builtin data objects encompassing data only for 15 cell lines, used in [4] and in this package to assess the performances of CRISPRcleanR.

**IMPORTANT:** To analyse data from screening a different cell line ad-hoc `GDSC.geneLevCNA`, `CCLE.gisticCNA`, and `RNAseq.fpkms` data objects should be assembled and passed to this function. These should have the same format of the respective builtin counterparts, detailed in their user reference manual entries, which contains also additional infos on how to derive this datasets for 1,000 human cancer cell lines from the GDSC1000 data portal [7]).

Comprehensive statistical scores (detailed in the user reference manual) are



also returned in output by this function.

Another example, analysing in the same way the essentiality profile of the EPLC-272H cell line (included in a corresponding builtin R object) is reported below,

```
RES<-ccr.perf_statTests('EPLC-272H',libraryAnnotation = KY_Library_v1.0,
                        correctedFCs = EPLC.272HcorrectedFCs$corrected_logFCs)

## [1] "No gistic CNA scores available for this cell line"
## [1] "Testing sgRNAs targeting: Dep (PN) genes"
## [1] "Testing sgRNAs targeting: Dep (Gistic) genes"
## [1] "Testing sgRNAs targeting: notExp genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +1) genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +2) genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +1) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (Gistic +2) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 2) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 4) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 8) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 10) genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 2) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 4) notExp genes"
## [1] "Testing sgRNAs targeting: Amp (PNs >= 8) notExp genes"
```

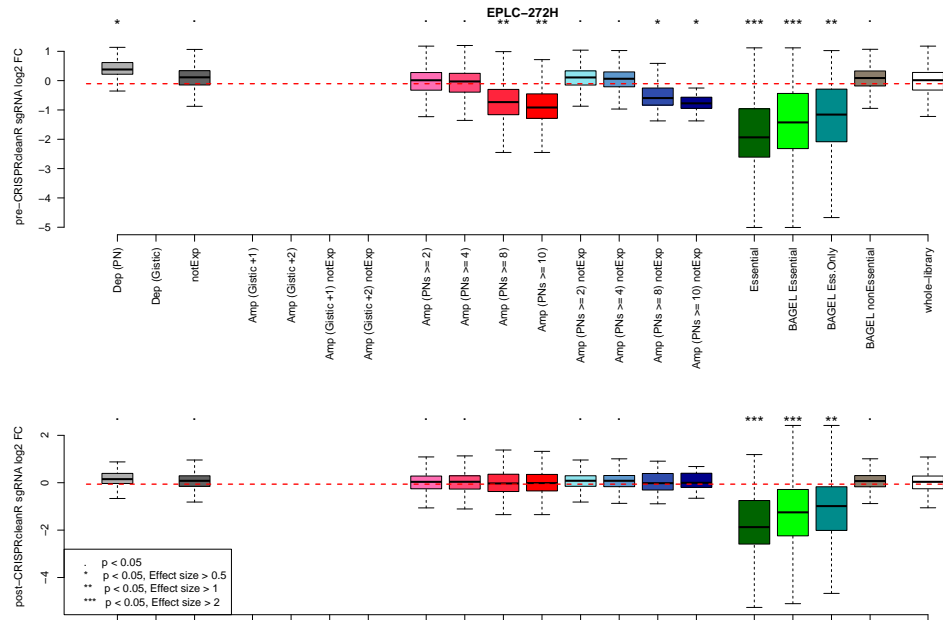
```
## [1] "Testing sgRNAs targeting: Amp (PNs >= 10) notExp genes"
## [1] "Testing sgRNAs targeting: Essential genes"
## [1] "Testing sgRNAs targeting: BAGEL Essential genes"
## [1] "Testing sgRNAs targeting: BAGEL Ess.Only genes"
## [1] "Testing sgRNAs targeting: BAGEL nonEssential genes"
```

#### RES\$PVALS

```
##               Dep (PN) Dep (Gistic)      notExp Amp (Gistic +1)
## pre-CRISPRcleanR 5.145194e-27      NA 0.00000e+00      NA
## post-CRISPRcleanR 3.219730e-06      NA 5.69841e-173      NA
##               Amp (Gistic +2) Amp (Gistic +1) notExp Amp (Gistic +2) notExp
## pre-CRISPRcleanR      NA      NA      NA      NA
## post-CRISPRcleanR      NA      NA      NA      NA
##               Amp (PNs >= 2) Amp (PNs >= 4) Amp (PNs >= 8) Amp (PNs >= 10)
## pre-CRISPRcleanR 1.580669e-18 1.780173e-57 1.991668e-137 1.476429e-42
## post-CRISPRcleanR 4.951065e-05 1.593962e-02 2.036767e-01 9.122396e-02
##               Amp (PNs >= 2) notExp Amp (PNs >= 4) notExp
## pre-CRISPRcleanR 2.268570e-311 2.387775e-52
## post-CRISPRcleanR 7.683083e-168 4.710481e-49
##               Amp (PNs >= 8) notExp Amp (PNs >= 10) notExp Essential
## pre-CRISPRcleanR 3.575550e-07 4.600584e-05 2.736956e-175
## post-CRISPRcleanR 1.871438e-01 4.089818e-01 3.550035e-159
##               BAGEL Essential BAGEL Ess.Only BAGEL nonEssential
## pre-CRISPRcleanR 2.898286e-297 1.591525e-205 8.174798e-81
## post-CRISPRcleanR 5.759565e-270 2.745117e-183 7.856856e-48
```

#### RES\$EFFsizes

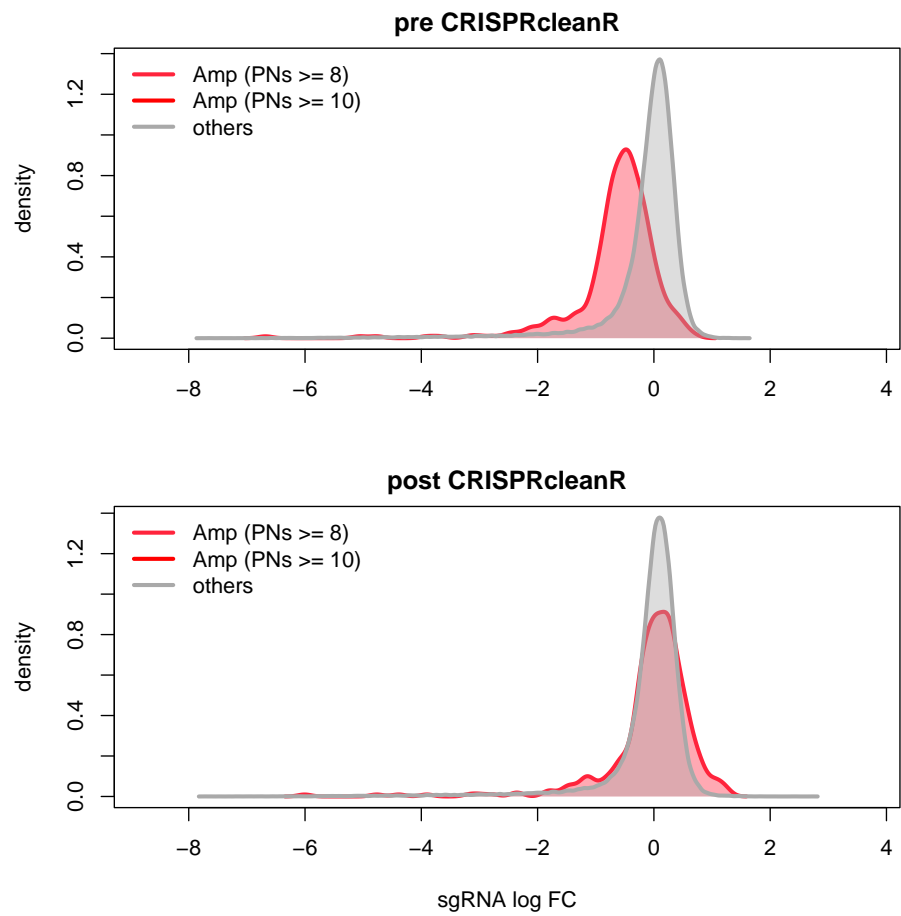
```
##               Dep (PN) Dep (Gistic)      notExp Amp (Gistic +1)
## pre-CRISPRcleanR 0.7269806      NA 0.3141706      NA
## post-CRISPRcleanR 0.2874528      NA 0.2168481      NA
##               Amp (Gistic +2) Amp (Gistic +1) notExp Amp (Gistic +2) notExp
## pre-CRISPRcleanR      NA      NA      NA      NA
## post-CRISPRcleanR      NA      NA      NA      NA
##               Amp (PNs >= 2) Amp (PNs >= 4) Amp (PNs >= 8) Amp (PNs >= 10)
## pre-CRISPRcleanR 0.10987878 0.1142627 1.0609873 1.34598769
## post-CRISPRcleanR 0.05019186 0.0170022 0.0426818 0.09939445
##               Amp (PNs >= 2) notExp Amp (PNs >= 4) notExp
## pre-CRISPRcleanR 0.3054604 0.2039852
## post-CRISPRcleanR 0.2183225 0.1916572
##               Amp (PNs >= 8) notExp Amp (PNs >= 10) notExp Essential
## pre-CRISPRcleanR 0.6678038 0.9848744 2.859216
## post-CRISPRcleanR 0.1506673 0.1617632 2.945718
##               BAGEL Essential BAGEL Ess.Only BAGEL nonEssential
## pre-CRISPRcleanR 2.223645 1.915641 0.2455203
## post-CRISPRcleanR 2.244647 1.900852 0.1799946
```

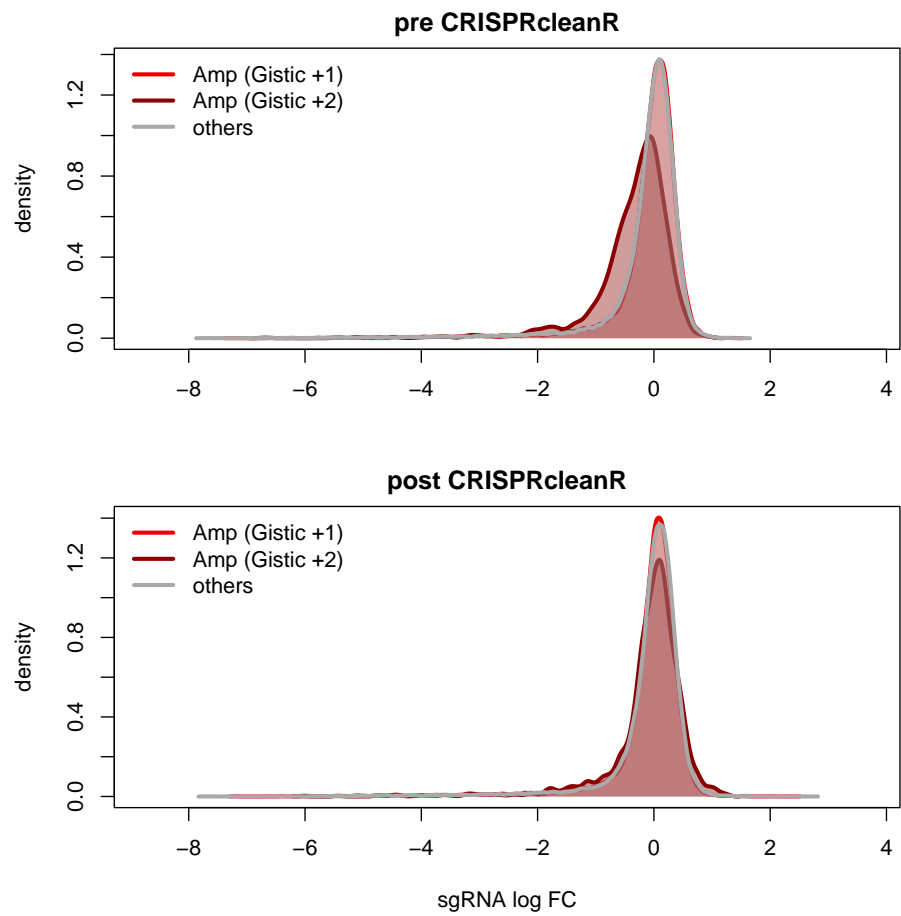


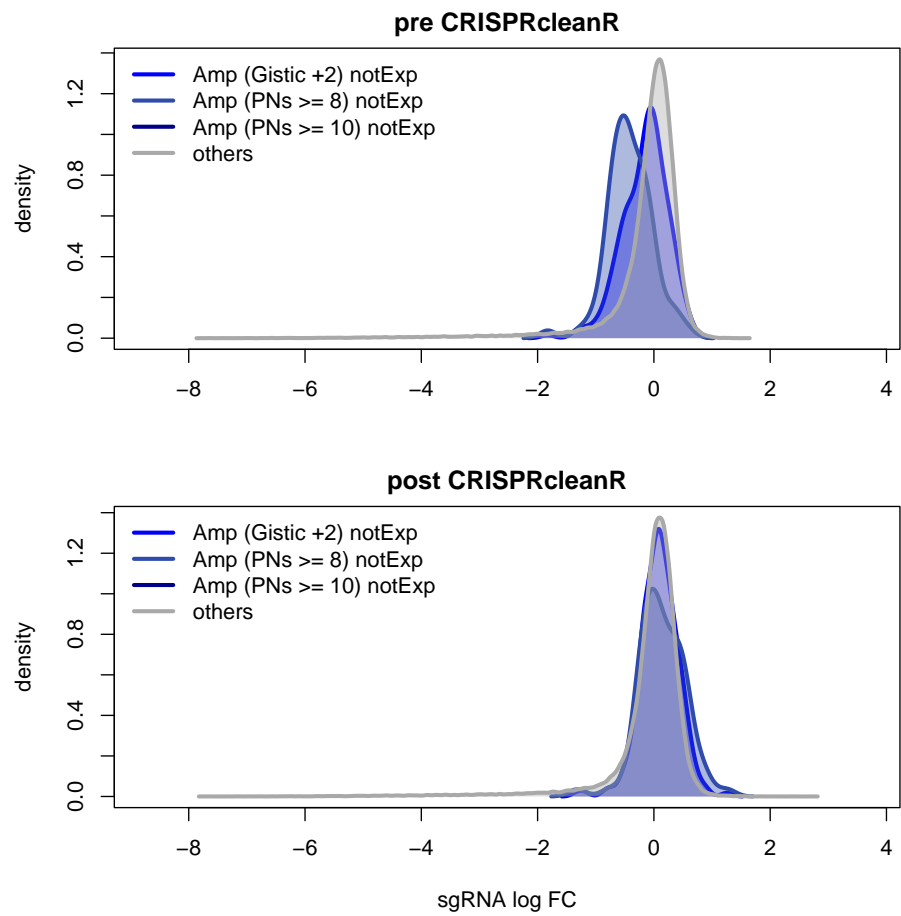
To inspect the variation induced by the CRISPRcleanR correction on the logFCs' distributions of sgRNAs targeting defined sets of genes prior/post CRISPRcleanR correction, the following function can be also used (and will produce the following density plots):

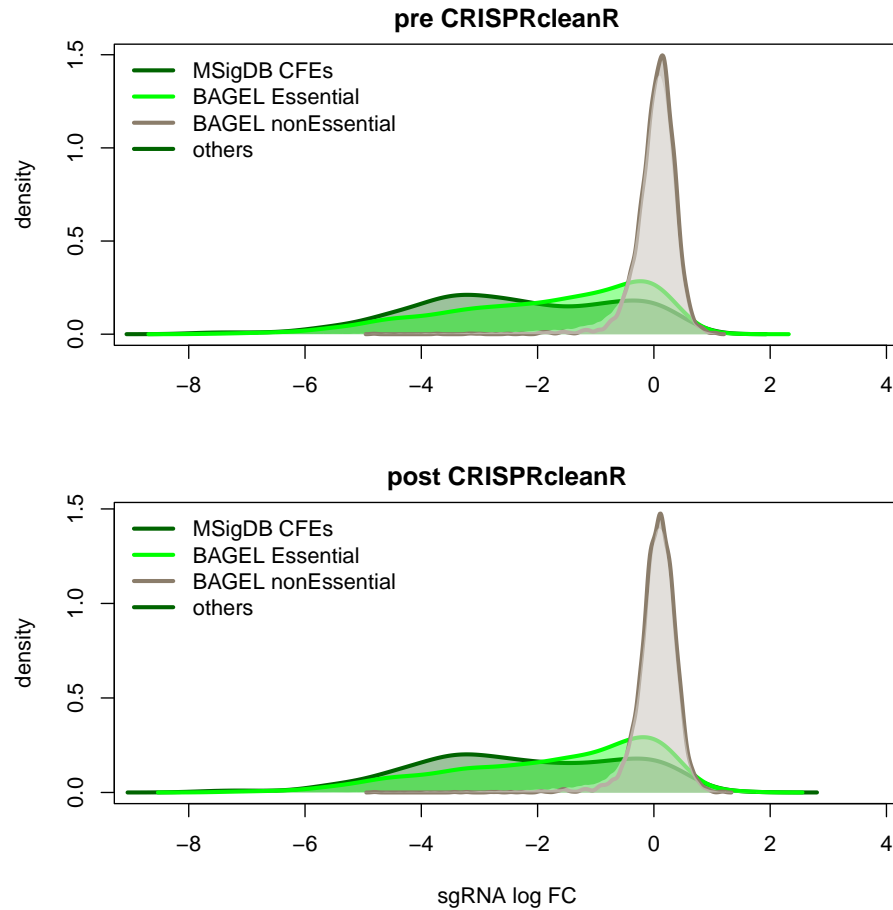
```
ccr.perf_distributions('HT-29', HT.29correctedFCs$corrected_logFCs,
  libraryAnnotation = KY_Library_v1.0)
```











**IMPORTANT:** The instructions provided regarding what CN/transcriptional data object to pass to the `ccr.perf_statTests` apply also to this function.

Additional infos on how to use this function can be found in the user reference manual.

## 2.4 Recall variations following CRISPRcleanR correction for reference, copy number amplified, and non expressed genes

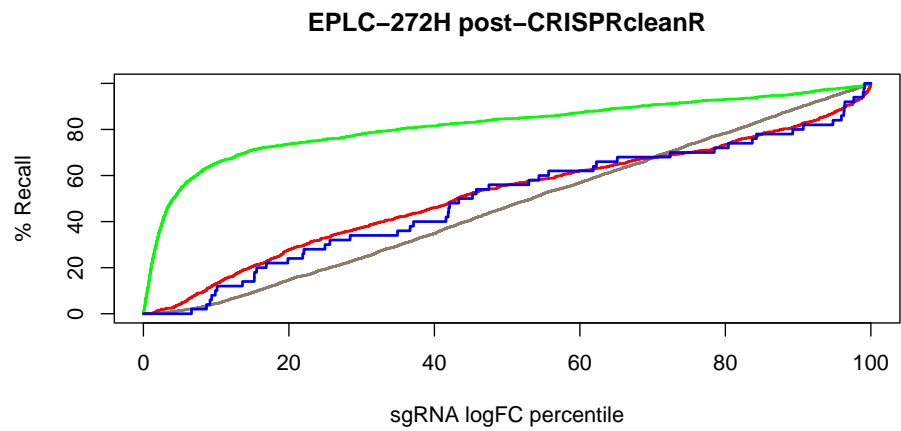
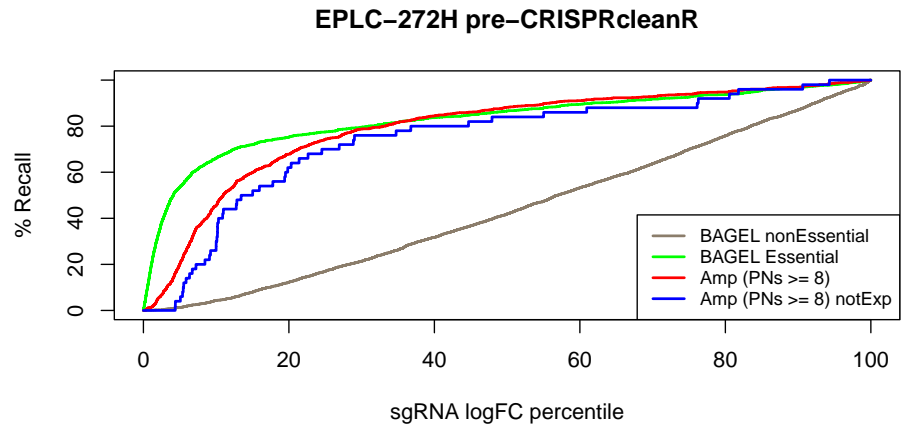
A final analysis that can be done with the CRISPRcleanR package in order to evaluate the effect of its correction on the classification recall for predefined gene sets can be performed by calling the function, which can work at the sgRNA

```

ccr.RecallCurves('EPLC-272H',EPLC.272HcorrectedFCs$corrected_logFCs,
                  libraryAnnotation=KY_Library_v1.0)

## [1] "No gistic CNA scores available for this cell line"

```



```

##          BAGEL nonEssential BAGEL Essential Amp (PNs >= 8)
## pre-CRISPRcleanR          0.4405845      0.8282859      0.7921870
## post-CRISPRcleanR         0.4663982      0.8149893      0.5109027
##          Amp (PNs >= 8) notExp
## pre-CRISPRcleanR          0.7447993
## post-CRISPRcleanR         0.4924973

```

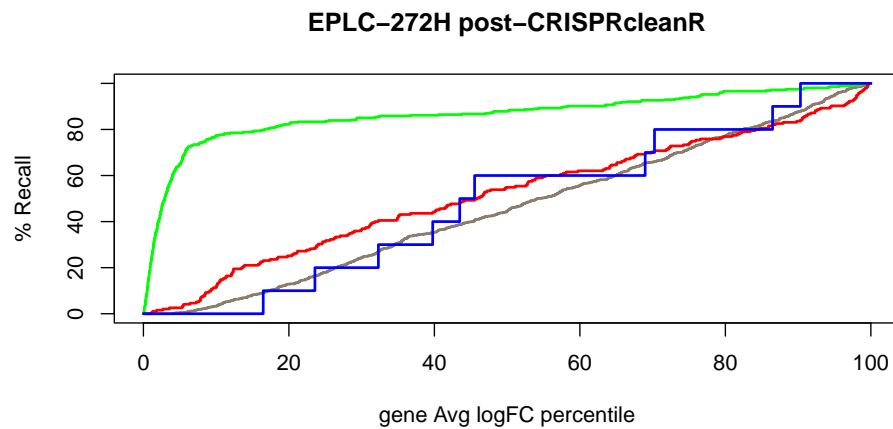
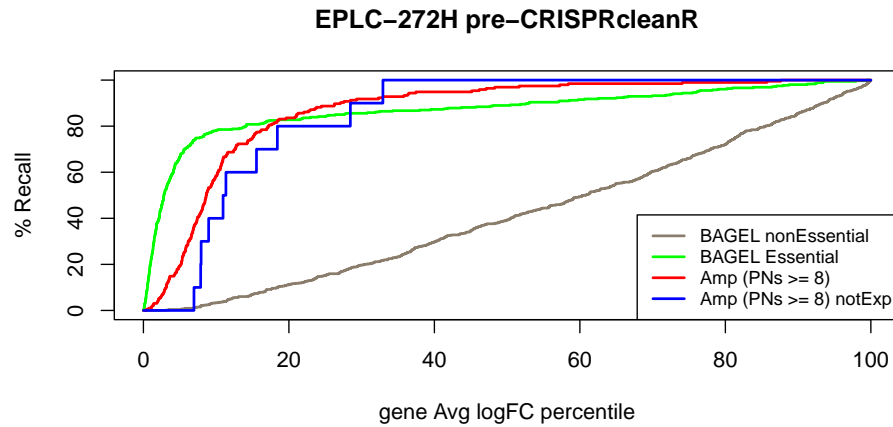
as well as the gene level:

```

ccr.RecallCurves('EPLC-272H',EPLC.272HcorrectedFCs$corrected_logFCs,
                  libraryAnnotation=KY_Library_v1.0,GeneLev = TRUE)

## [1] "No gistic CNA scores available for this cell line"

```



	BAGEL nonEssential	BAGEL Essential	Amp (PNs >= 8)
pre-CRISPRcleanR	0.4188711	0.8704539	0.8697322
post-CRISPRcleanR	0.4570109	0.8627785	0.5156109
	Amp (PNs >= 8) notExp		
pre-CRISPRcleanR	0.8506553		
post-CRISPRcleanR	0.4828279		

**IMPORTANT:** The instructions provided regarding what CN/transcriptional data object to pass to the `ccr.perf_statTests` apply also to this function.

Additional infos on how to use this function can be found in the user reference manual.

## 2.5 CRISPRcleanR correction assessment: impact on phenotype and possible distortion

To evaluate the effect of the CRISPRcleanR correction on the genes showing a significant loss/gain-of-fitness effect (fitness genes) in the uncorrected data, a comparison of fitness gene sets (computed with MAGeCK [2]) before/after CRISPRcleanR correction can be performed as follows

Before running the below example let us clean the R environment with the following command:

```
rm(list=ls())
```

Subsequently, as detailed above, we load the reference sgRNA library annotation, we load and normalise sgRNA counts for the EPLC-272H example cell line.

```
data(KY_Library_v1.0)

fn<-paste(system.file('extdata', package = 'CRISPRcleanR'),
           '/EPLC-272H_counts.tsv', sep='')

normANDfcs<-ccr.NormfoldChanges(fn,min_reads=30,
                                EXPname='EPLC-272H',
                                libraryAnnotation = KY_Library_v1.0)
```

Then, we save the normalised counts as plain tsv file (suitable for MAGeCK):

```
uncorrected_fn<-
  ccr.PlainTsvFile(sgRNA_count_object = normANDfcs$norm_counts,
                   fprefix = 'EPLC-272H')
```

At this point we execute MAGeCK on the uncorrected normalised counts. **IMPORTANT:** This requires python and MAGeCK (v0.5.3). Additionally, as these counts are already normalised, we should specify that the normalisation shouldn't be reperformed by MAGeCK.

Run-time information displayed by MAGeCK (not reported in this document) will be visualised and all the outputted files will be saved in the working directory (by default), unless different specified in the corresponding argument (see reference manual entry for this function for further details).

```

uncorrected_gs_fn<-
  ccr.ExecuteMageck(mgckInputFile = uncorrected_fn,
                    expName = 'EPLC-272H',
                    normMethod = 'none')

```

This function returns in output the path of the gene summary file outputted by MAGeCK.

```

uncorrected_gs_fn

## [1] "./EPLC-272H.gene_summary.txt"

```

As detailed in the previous sections of this document, we now correct the sgRNA counts of the example cell lines with CRISPRcleanR and save them in a plain tsv file (suitable to be used by MAGeCK)

```

gwSortedFCs<-ccr.logFCs2chromPos(normANDfcs$logFCs,KY_Library_v1.0)

correctedFCs<-ccr.GWclean(gwSortedFCs,display=FALSE,label='EPLC-272H')

correctedCounts<-ccr.correctCounts('EPLC-272H',normANDfcs$norm_counts,
                                   correctedFCs,
                                   KY_Library_v1.0,
                                   minTargetedGenes=3,
                                   OutDir='./')

corrected_fn<-ccr.PlainTsvFile(sgRNA_count_object = correctedCounts,
                              fprefix = 'EPLC-272H_ccleaned')

```

Then we execute MAGeCK on the corrected sgRNA counts.

```

corrected_gs_fn<-ccr.ExecuteMageck(mgckInputFile = corrected_fn,
                                   expName = 'EPLC-272H_ccleaned',
                                   normMethod = 'none')

```

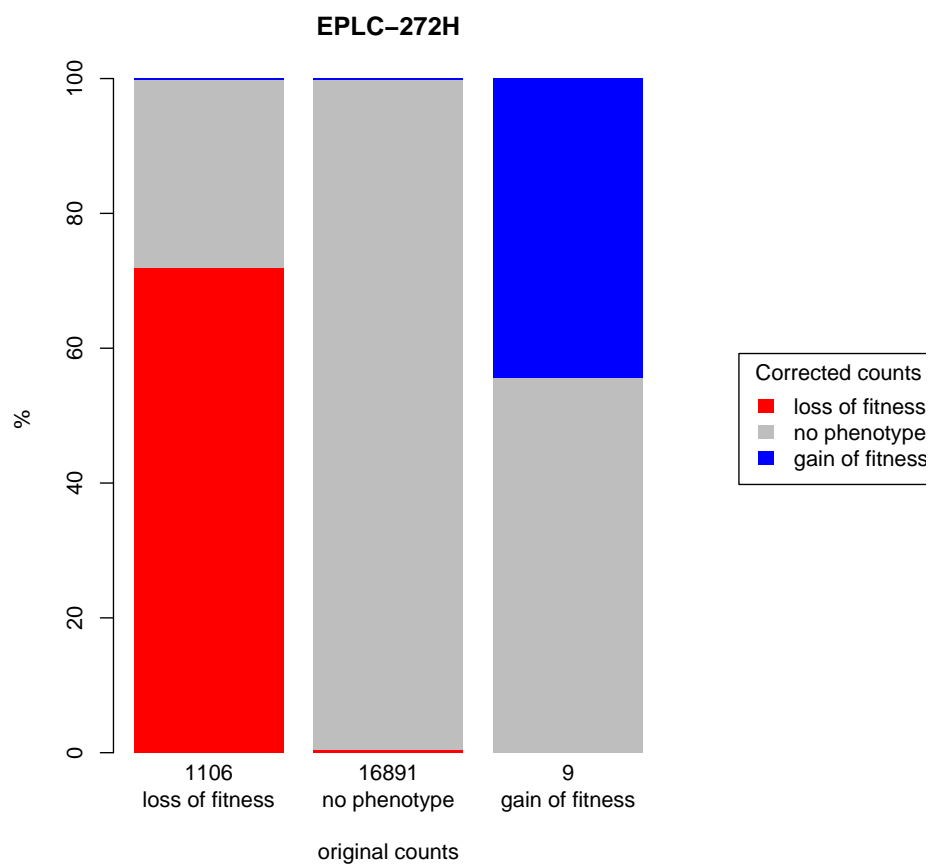
Finally, we assess the impact of CRISPRcleanR correction on gain/loss-of-fitness genes with the following call, which will produce bar/pie charts reported below

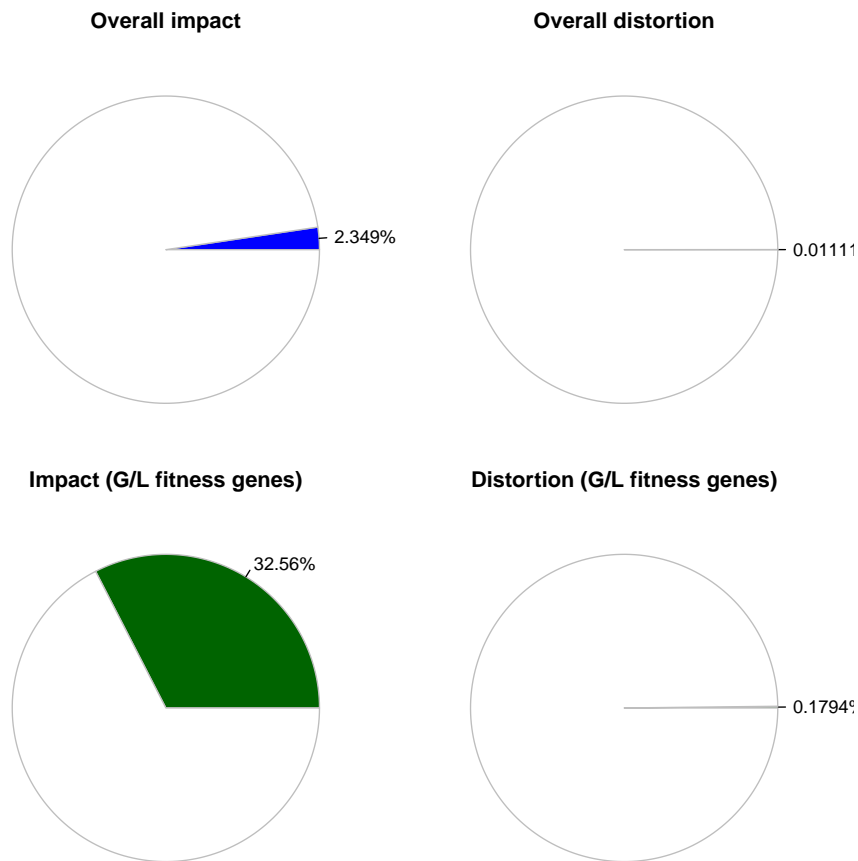
```

RES<-ccr.impactOnPhenotype(MO_uncorrectedFile = uncorrected_gs_fn,
                           MO_correctedFile = corrected_gs_fn,
                           expName = 'EPLC-272H')

```







### 3 CRISPRcleanR analysis pipeline

To facilitate the whole analysis process all the normalization, correction and QC steps can run sequentially through the `ccr.AnalysisPipeline` function.

The function takes as input FASTQ, BAM or counts files and performs all steps up to the visualisation of the genes signatures. All the results will be saved in the output path organized in different subfolders: **data**, including all the data table in TSV format; **pdf**, including all the plots exported in PDF format. If the output is based on FASTQ files, an optional **bam** subfolder will be available to store the output of the alignment process.

All the usual parameters required for the different steps can be supplied directly to the `ccr.AnalysisPipeline` function.

```
fn <- file.path(
  system.file("extdata", package = "CRISPRcleanR"),
  "HT-29_counts.tsv"
)

## Run the alignment and extract the raw counts
suppressWarnings(ccr.AnalysisPipeline(
  file_counts = fn,
  outdir='./HT29_pipeline/',
  EXPname = 'HT29',
  library_builtin = "KY_Library_v1.0",
  run_mageck = FALSE,
  ncontrols = 1
))
```

```
list.files('./HT29_pipeline/')

## [1] "data" "HT29_correctedCounts.RData"
## [3] "HT29_foldChanges.Rdata" "HT29_normCounts.Rdata"
## [5] "pdf"
```

## References

- [1] Yang Liao, Gordon K Smyth, and Wei Shi. “The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads”. In: *Nucleic Acids Research* 47.8 (), e47.
- [2] Wei Li et al. “MAGeCK enables robust identification of essential genes from genome-scale CRISPR/Cas9 knockout screens.” In: *Genome Biology* 15.12 (2014), p. 554.
- [3] Konstantinos Tzelepis et al. “A CRISPR Dropout Screen Identifies Genetic Vulnerabilities and Therapeutic Targets in Acute Myeloid Leukemia.” In: *Cell reports* 17.4 (Oct. 2016), pp. 1193–1205.
- [4] Francesco Iorio et al. “Unsupervised correction of gene-independent cell responses to CRISPR-Cas9 targeting”. In: *revision* 0.0 (), pp. 0–0.
- [5] Traver Hart and Jason Moffat. “BAGEL: a computational framework for identifying essential genes from pooled library screens.” In: *BMC bioinformatics* 17 (Apr. 2016), p. 164.
- [6] A Subramanian et al. “Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles”. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.43 (2005), p. 15545.

- [7] Francesco Iorio et al. “A Landscape of Pharmacogenomic Interactions in Cancer.” In: *Cell* (July 2016).