

Package ‘alevinQC’

April 15, 2020

Type Package

Title Generate QC Reports For Alevin Output

Version 1.2.0

Description Generate QC reports summarizing the output from an alevin run.
Reports can be generated as html or pdf files, or as shiny applications.

Encoding UTF-8

Depends R (>= 3.6)

Imports rmarkdown, tools, methods, ggplot2, GGally, dplyr, rjson,
shiny, shinydashboard, DT, stats, utils, tximport (>= 1.11.5),
cowplot

RoxygenNote 6.1.1

Suggests knitr, BiocStyle, testthat,

VignetteBuilder knitr

biocViews QualityControl, SingleCell

URL <https://github.com/csoneson/alevinQC>

BugReports <https://github.com/csoneson/alevinQC/issues>

License MIT + file LICENSE

git_url <https://git.bioconductor.org/packages/alevinQC>

git_branch RELEASE_3_10

git_last_commit 8ca336f

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

Author Charlotte Soneson [aut, cre] (<<https://orcid.org/0000-0003-3833-2169>>),
Avi Srivastava [aut]

Maintainer Charlotte Soneson <charlottesoneson@gmail.com>

R topics documented:

alevinQCReport	2
alevinQCShiny	3
checkAlevinInputFiles	4
plotAlevinBarcodeCollapse	4

plotAlevinKneeNbrGenes	5
plotAlevinKneeRaw	6
plotAlevinQuant	6
plotAlevinQuantPairs	7
readAlevinQC	8
Index	9

alevinQCReport	<i>Generate alevin summary report</i>
----------------	---------------------------------------

Description

Generate a report summarizing the main aspects of an alevin quantification run. The report generation assumes that alevin has been run with the `-dumpFeatures` flag to generate the necessary output files.

Usage

```
alevinQCReport(baseDir, sampleId, outputFile, outputDir = "./",
  outputFormat = NULL, showCode = FALSE, forceOverwrite = FALSE,
  knitrProgress = FALSE, quiet = FALSE, ignorePandoc = FALSE, ...)
```

Arguments

<code>baseDir</code>	Path to the output directory from the alevin run (should be the directory containing the alevin directory).
<code>sampleId</code>	Sample ID, will be used to set the title for the report.
<code>outputFile</code>	File name of the output report. The file name extension must be either <code>.html</code> or <code>.pdf</code> , and consistent with the value of <code>outputFormat</code> .
<code>outputDir</code>	Path to the output directory where the report will be generated.
<code>outputFormat</code>	The format of the output report. Either <code>"html_document"</code> or <code>"pdf_document"</code> . The file name extension of <code>outputFile</code> must be consistent with this choice.
<code>showCode</code>	Logical, whether to display the R code in the report.
<code>forceOverwrite</code>	Logical, whether to force overwrite an existing report with the same name in the output directory.
<code>knitrProgress</code>	Logical, whether to display the progress of <code>knitr</code> when generating the report.
<code>quiet</code>	Logical, whether to show progress messages.
<code>ignorePandoc</code>	Logical, determines what to do if <code>pandoc</code> or <code>pandoc-citeproc</code> is missing (if <code>Sys.which("pandoc")</code> or <code>Sys.which("pandoc-citeproc")</code> returns <code>""</code>). If <code>ignorePandoc</code> is <code>TRUE</code> , only a warning is given. The figures will be generated, but not the final report. If <code>ignorePandoc</code> is <code>FALSE</code> (default), the execution stops immediately.
<code>...</code>	Other arguments that will be passed to <code>rmarkdown::render</code> .

Details

When the function is called, a .Rmd template file will be copied into the output directory, and `rmarkdown::render` will be called to generate the final report. If there is already a .Rmd file with the same name in the output directory, the function will raise an error and stop, to avoid overwriting the existing file. The reason for this behaviour is that the copied template in the output directory will be deleted once the report is generated.

Value

Generates a summary report in the `outputDir` directory, and returns (invisibly) the name of the generated report.

Author(s)

Charlotte Soneson

Examples

```
alevinQCReport(baseDir = system.file("extdata/alevin_example_v0.14",
                                     package = "alevinQC"),
               sampleId = "example", outputFile = "alevinReport.html",
               outputDir = tempdir(), forceOverwrite = TRUE)
```

alevinQCShiny

Generate alevin summary shiny app

Description

Generate a shiny app summarizing the main aspects of an alevin quantification run. The app generation assumes that alevin has been run with the `-dumpFeatures` flag to generate the necessary output files.

Usage

```
alevinQCShiny(baseDir, sampleId)
```

Arguments

<code>baseDir</code>	Path to the output directory from the alevin run (should be the directory containing the alevin directory).
<code>sampleId</code>	Sample ID, will be used set the title for the app.

Value

A shiny app.

Author(s)

Charlotte Soneson

Examples

```
app <- alevinQCShiny(baseDir = system.file("extdata/alevin_example_v0.14",
                                          package = "alevinQC"),
                   sampleId = "example")
if (interactive()) {
  shiny::runApp(app)
}
```

checkAlevinInputFiles *Check that all required input files are available*

Description

Check that all required input files are available

Usage

```
checkAlevinInputFiles(baseDir)
```

Arguments

baseDir Path to the output directory from the alevin run (should be the directory containing the alevin directory).

Value

Returns nothing, raises an error if any of the required files are missing.

Author(s)

Charlotte Soneson

Examples

```
checkAlevinInputFiles(system.file("extdata/alevin_example_v0.14",
                                   package = "alevinQC"))
```

plotAlevinBarcodeCollapse
Summary plot of cell barcode collapsing

Description

Plot the original frequency of each cell barcode in the original whitelist against the frequency after collapsing similar cell barcodes.

Usage

```
plotAlevinBarcodeCollapse(cbTable)
```

Arguments

cbTable data.frame (such as the cbTable returned by readAlevinQC) with barcode frequencies before and after collapsing.

Value

A ggplot object

Author(s)

Charlotte Soneson

Examples

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_pre0.14",
                                   package = "alevinQC"))
plotAlevinBarcodeCollapse(alevin$cbTable)
```

plotAlevinKneeNbrGenes

Knee plot of the number of detected genes per cell

Description

Plot the number of detected genes per cell in decreasing order.

Usage

```
plotAlevinKneeNbrGenes(cbTable)
```

Arguments

cbTable data.frame (such as the cbTable returned by readAlevinQC) with the number of detected genes per cell.

Value

A ggplot object

Author(s)

Charlotte Soneson

Examples

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_pre0.14",
                                   package = "alevinQC"))
plotAlevinKneeNbrGenes(alevin$cbTable)
```

plotAlevinKneeRaw *Knee plot of raw cell barcode frequencies*

Description

Plot the raw cell barcode frequencies in decreasing order, and indicate a predetermined breakpoint using color as well as a label.

Usage

```
plotAlevinKneeRaw(cbTable)
```

Arguments

cbTable data.frame with raw barcode frequencies (such as the cbTable returned by readAlevinQC).

Value

A ggplot object

Author(s)

Charlotte Soneson

Examples

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_pre0.14",  
                                package = "alevinQC"))  
plotAlevinKneeRaw(alevin$cbTable)
```

plotAlevinQuant *Panel of plots with quantification summary statistics*

Description

Panel of plots with quantification summary statistics

Usage

```
plotAlevinQuant(cbTable)
```

Arguments

cbTable data.frame (such as the cbTable returned by readAlevinQC) with collapsed barcode frequencies, the total UMI count and the number of detected genes for each cell.

Value

A ggplot object

Author(s)

Charlotte Soneson

Examples

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_v0.14",  
                                package = "alevinQC"))  
plotAlevinQuant(alevin$cbTable)
```

`plotAlevinQuantPairs` *Pairs plot with quantification summary statistics*

Description

Pairs plot with quantification summary statistics

Usage

```
plotAlevinQuantPairs(cbTable)
```

Arguments

`cbTable` data.frame (such as the `cbTable` returned by `readAlevinQC`) with collapsed barcode frequencies, the total UMI count and the number of detected genes for each cell.

Value

A ggmatrix object

Author(s)

Charlotte Soneson

Examples

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_pre0.14",  
                                package = "alevinQC"))  
plotAlevinQuantPairs(alevin$cbTable)
```

`readAlevinQC`*Read alevin data required to generate summary report*

Description

Read all alevin output files required to generate the summary report or shiny app.

Usage

```
readAlevinQC(baseDir)
```

Arguments

`baseDir` Path to the output directory from the alevin run (should be the directory containing the alevin directory).

Value

A list collecting all necessary information for generating the summary report/shiny app.

Author(s)

Charlotte Soneson

Examples

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_v0.14",  
                                package = "alevinQC"))
```


Index

[alevinQCReport](#), 2
[alevinQCShiny](#), 3

[checkAlevinInputFiles](#), 4

[plotAlevinBarcodeCollapse](#), 4
[plotAlevinKneeNbrGenes](#), 5
[plotAlevinKneeRaw](#), 6
[plotAlevinQuant](#), 6
[plotAlevinQuantPairs](#), 7

[readAlevinQC](#), 8