

# Package ‘alevinQC’

January 15, 2025

**Type** Package

**Title** Generate QC Reports For Alevin Output

**Version** 1.23.1

**Date** 2024-12-14

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

**Encoding** UTF-8

**SystemRequirements** C++11

**Depends** R (>= 4.0)

**Imports** rmarkdown (>= 2.5), tools, methods, ggplot2 (>= 3.4.0), GGally, dplyr, rjson, shiny, shinydashboard, DT, stats, utils, tximport (>= 1.17.4), cowplot, rlang, Rcpp

**RoxygenNote** 7.3.2

**Suggests** knitr, BiocStyle, testthat (>= 3.0.0), BiocManager

**VignetteBuilder** knitr

**biocViews** QualityControl, SingleCell

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

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

**License** MIT + file LICENSE

**Config/testthat/edition** 3

**LinkingTo** Rcpp

**git\_url** <https://git.bioconductor.org/packages/alevinQC>

**git\_branch** devel

**git\_last\_commit** 9ef304b

**git\_last\_commit\_date** 2024-12-21

**Repository** Bioconductor 3.21

**Date/Publication** 2025-01-15

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

**Maintainer** Charlotte Soneson <charlottesoneson@gmail.com>

## Contents

alevinQC-package . . . . .	2
checkAlevinFryInputFiles . . . . .	3
checkAlevinInputFiles . . . . .	4
plotAlevinBarcodeCollapse . . . . .	4
plotAlevinHistogram . . . . .	5
plotAlevinKneeNbrGenes . . . . .	6
plotAlevinKneeRaw . . . . .	7
plotAlevinQuant . . . . .	8
plotAlevinQuantPairs . . . . .	9
qcReport . . . . .	10
qcShiny . . . . .	12
readAlevinFryQC . . . . .	14
readAlevinQC . . . . .	15

**Index** **16**

---

alevinQC-package	<i>alevinQC</i>
------------------	-----------------

---

## Description

alevinQC

## Author(s)

**Maintainer:** Charlotte Soneson <charlottesoneson@gmail.com> (ORCID)

Authors:

- Avi Srivastava
- Rob Patro
- Dongze He

## See Also

Useful links:

- <https://github.com/csoneson/alevinQC>
- Report bugs at <https://github.com/csoneson/alevinQC/issues>

---

`checkAlevinFryInputFiles`*Check that all required input files are available for alevin-fry*

---

**Description**

Check that all required input files are available for alevin-fry

**Usage**

```
checkAlevinFryInputFiles(mapDir, permitDir, quantDir)
```

**Arguments**

<code>mapDir</code>	Path to the directory containing the map.rad file
<code>permitDir</code>	Path to the output directory of the alevin-fry generate-permit-list command.
<code>quantDir</code>	Path to the output of the alevin-fry quant command.

**Value**

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

**Author(s)**

Charlotte Soneson

**Examples**

```
checkAlevinFryInputFiles(  
  mapDir = system.file("extdata/alevinfry_example_v0.5.0/map",  
    package = "alevinQC"),  
  permitDir = system.file("extdata/alevinfry_example_v0.5.0/permit",  
    package = "alevinQC"),  
  quantDir = system.file("extdata/alevinfry_example_v0.5.0/quant",  
    package = "alevinQC"))
```

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,  
  firstSelColName = "inFirstWhiteList",  
  countCol = "collapsedFreq"  
)
```

**Arguments**

cbTable	data.frame (such as the cbTable returned by readAlevinQC or readAlevinFryQC) with barcode frequencies before and after collapsing.
firstSelColName	Character scalar indicating the name of the logical column in cbTable that corresponds to the original selection of barcodes for quantification.
countCol	Character scalar indicating the name of the column in cbTable that corresponds to the collapsed barcode frequencies.

**Value**

A ggplot object

**Author(s)**

Charlotte Soneson

**Examples**

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

---

plotAlevinHistogram *Histogram of selected summary statistic*

---

**Description**

Histogram of selected summary statistic

**Usage**

```
plotAlevinHistogram(  
  cbTable,  
  plotVar = "dedupRate",  
  axisLabel = plotVar,  
  colName = "inFinalWhiteList",  
  cbName = "final whitelist",  
  firstSelColName = "inFirstWhiteList"  
)
```

**Arguments**

cbTable	data.frame (such as the cbTable returned by readAlevinQC or readAlevinFryQC) containing the desired summary statistic in a column.
plotVar	Character scalar giving the name of a numeric column of cbTable to plot.
axisLabel	Character scalar giving the label of the selected statistic (will be displayed as the axis label in the plot).
colName	Character scalar giving the name of a logical column of cbTable to use for filling the bars in the histogram.
cbName	Character scalar giving the name of the set of barcodes defined by colName, used for labelling the plot legend.
firstSelColName	Character scalar indicating the name of the logical column in cbTable that corresponds to the original selection of barcodes for quantification.

**Value**

A ggplot object

**Author(s)**

Charlotte Soneson

**Examples**

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_v0.14",
                                   package = "alevinQC"))
plotAlevinHistogram(alevin$cbTable, plotVar = "dedupRate",
                    axisLabel = "Deduplication rate",
                    colName = "inFinalWhitelist",
                    cbName = "final whitelist")
```

---

plotAlevinKneeNbrGenes

*Knee plot of the number of detected genes per cell*

---

**Description**

Plot the number of detected genes per cell in decreasing order. Only cells contained in the original whitelist are considered.

**Usage**

```
plotAlevinKneeNbrGenes(cbTable, firstSelColName = "inFirstWhiteList")
```

**Arguments**

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

`firstSelColName` Character scalar indicating the name of the logical column in `cbTable` that corresponds to the original selection of barcodes for quantification.

**Value**

A ggplot object

**Author(s)**

Charlotte Soneson

**Examples**

```
alevin <- readAlevinQC(system.file("extdata/alevin_example_v0.14",  
                                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 (indicating barcodes included in the original whitelist) using color as well as a label.

**Usage**

```
plotAlevinKneeRaw(cbTable, firstSelColName = "inFirstWhiteList")
```

**Arguments**

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

`firstSelColName` Character scalar indicating the name of the logical column in `cbTable` that corresponds to the original selection of barcodes for quantification.

**Value**

A ggplot object

**Author(s)**

Charlotte Soneson

**Examples**

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

---

plotAlevinQuant      *Panel of plots with quantification summary statistics*

---

**Description**

Panel of plots with quantification summary statistics

**Usage**

```
plotAlevinQuant(
  cbTable,
  colName = "inFinalWhiteList",
  cbName = "final whitelist",
  firstSelColName = "inFirstWhiteList"
)
```

**Arguments**

cbTable	data.frame (such as the cbTable returned by readAlevinQC or readAlevinFryQC) with collapsed barcode frequencies, the total UMI count and the number of detected genes for each cell.
colName	Character scalar giving the name of a logical column of cbTable to use for coloring the points.
cbName	Character scalar giving the name of the set of barcodes defined by colName, used for labelling the plot legend.
firstSelColName	Character scalar indicating the name of the logical column in cbTable that indicates the original selection of barcodes for quantification.

**Value**

A ggplot object

**Author(s)**

Charlotte Soneson



**Examples**

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

---

plotAlevinQuantPairs *Pairs plot with quantification summary statistics*

---

**Description**

Pairs plot with quantification summary statistics

**Usage**

```
plotAlevinQuantPairs(
  cbTable,
  colName = "inFinalWhiteList",
  firstSelColName = "inFirstWhiteList"
)
```

**Arguments**

cbTable	data.frame (such as the cbTable returned by readAlevinQC or readAlevinFryQC) with collapsed barcode frequencies, the total UMI count and the number of detected genes for each cell.
colName	Character scalar giving the name of a logical column of cbTable to use for coloring the points.
firstSelColName	Character scalar indicating the name of the logical column in cbTable that corresponds to the original selection of barcodes for quantification.

**Value**

A ggmatrix object

**Author(s)**

Charlotte Soneson

**Examples**

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

---

`qcReport`*Generate alevin/alevin-fry summary report*

---

**Description**

Generate a report summarizing the main aspects of an alevin/alevin-fry quantification run. The report generation assumes that alevin/alevin-fry 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,  
  customCBList = list(),  
  ...  
)
```

```
simpleafQCReport(  
  simpleafQuantDir,  
  sampleId,  
  outputFile,  
  outputDir = "./",  
  outputFormat = NULL,  
  showCode = FALSE,  
  forceOverwrite = FALSE,  
  knitrProgress = FALSE,  
  quiet = FALSE,  
  ignorePandoc = FALSE,  
  customCBList = list(),  
  ...  
)
```

```
alevinFryQCReport(  
  mapDir,  
  permitDir,  
  quantDir,  
  sampleId,  
  outputFile,
```

```

    outputDir = "./",
    outputFormat = NULL,
    showCode = FALSE,
    forceOverwrite = FALSE,
    knitrProgress = FALSE,
    quiet = FALSE,
    ignorePandoc = FALSE,
    customCBList = list(),
    ...
)

```

## Arguments

<code>baseDir</code>	(Only used for alevin output) 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> , <code>"pdf_document"</code> , <code>"BiocStyle::html_document"</code> or <code>"BiocStyle::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 knitr when generating the report.
<code>quiet</code>	Logical, whether to show progress messages.
<code>ignorePandoc</code>	Logical, determines what to do if pandoc or pandoc-citeproc 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>customCBList</code>	Named list with custom set(s) of barcodes to provide summary statistics/plots for, in addition to the whitelists generated by alevin.
<code>...</code>	Other arguments that will be passed to <code>rmarkdown::render</code> .
<code>simpleafQuantDir</code>	(Only used for simpleaf output) Path to the output directory from the simpleaf run (should be the directory containing the <code>af_map</code> and <code>af_quant</code> directories).
<code>mapDir</code>	(Only used for alevin-fry output) Path to the output directory from the salmon alevin run (should be the directory containing the <code>map.rad</code> file).
<code>permitDir</code>	(Only used for alevin-fry output) Path to the output directory from the permit list generation step (should be the directory containing the <code>all_freq.tsv</code> file).
<code>quantDir</code>	(Only used for alevin-fry output) Path to the output directory from the alevin-fry quantification step (should be the directory containing the alevin directory).

## 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)  
  
alevinFryQCReport(  
  mapDir = system.file("extdata/alevinfry_example_v0.5.0/map",  
                      package = "alevinQC"),  
  permitDir = system.file("extdata/alevinfry_example_v0.5.0/permit",  
                          package = "alevinQC"),  
  quantDir = system.file("extdata/alevinfry_example_v0.5.0/quant",  
                        package = "alevinQC"),  
  sampleId = "example", outputFile = "alevinFryReport.html",  
  outputDir = tempdir(), forceOverwrite = TRUE)  
  
simpleafQCReport(  
  simpleafQuantDir = system.file("extdata/alevinfry_example_piscem_v0.6.0",  
                                package = "alevinQC"),  
  sampleId = "example", outputFile = "simpleafReport.html",  
  outputDir = tempdir(), forceOverwrite = TRUE)
```

---

qcShiny

*Generate alevin/alevin-fry summary shiny app*

---

## Description

Generate a shiny app summarizing the main aspects of an alevin/alevin-fry 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, customCBLIST = list(), addStopButton = TRUE)
```

```
alevinFryQCShiny(mapDir, permitDir, quantDir, sampleId, addStopButton = TRUE)
```

```
simpleafQCShiny(simpleafQuantDir, sampleId, addStopButton = TRUE)
```

**Arguments**

baseDir	(Only used for alevin output) Path to the output directory from the alevin run (should be the directory containing the alevin directory).
sampleId	Sample ID, will be used set the title for the app.
customCBLIST	Named list with custom set(s) of barcodes to provide summary statistics/plots for, in addition to the whitelists generated by alevin.
addStopButton	Logical scalar. If TRUE (default), will add a dropdown menu with a button to stop the app (by calling <code>shiny::stopApp</code> ) and return a list with the information displayed in the app.
mapDir	(Only used for alevin-fry output) Path to the output directory from the salmon alevin run (should be the directory containing the <code>map.rad</code> file).
permitDir	(Only used for alevin-fry output) Path to the output directory from the permit list generation step (should be the directory containing the <code>all_freq.tsv</code> file).
quantDir	(Only used for alevin-fry output) Path to the output directory from the alevin-fry quantification step (should be the directory containing the alevin directory).
simpleafQuantDir	(Only used for simpleaf output) Path to the output directory from the simpleaf run (should be the directory containing the <code>af_map</code> and <code>af_quant</code> directories).

**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)
}

app <- alevinFryQCShiny(
  mapDir = system.file("extdata/alevinfry_example_v0.5.0/map",
    package = "alevinQC"),
```

```

    permitDir = system.file("extdata/alevinfry_example_v0.5.0/permit",
                           package = "alevinQC"),
    quantDir = system.file("extdata/alevinfry_example_v0.5.0/quant",
                           package = "alevinQC"),
    sampleId = "example")
if (interactive()) {
  shiny::runApp(app)
}

app <- simpleafQCShiny(
  simpleafQuantDir = system.file("extdata/alevinfry_example_piscem_v0.6.0",
                                 package = "alevinQC"),
  sampleId = "example")
if (interactive()) {
  shiny::runApp(app)
}

```

---

readAlevinFryQC

*Read alevin-fry data required to generate summary report*


---

### Description

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

### Usage

```
readAlevinFryQC(mapDir, permitDir, quantDir)
```

### Arguments

mapDir	Path to the output directory from the salmon alevin run (should be the directory containing the alevin folder).
permitDir	Path to the output directory from the generate-permit-list and collate runs.
quantDir	Path to the output directory from the alevin-fry quant run (should be the directory containing the alevin folder).

### Value

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

### Author(s)

Charlotte Soneson

**Examples**

```
alevinfry <- readAlevinFryQC(  
  mapDir = system.file("extdata/alevinfry_example_v0.5.0/map",  
    package = "alevinQC"),  
  permitDir = system.file("extdata/alevinfry_example_v0.5.0/permit",  
    package = "alevinQC"),  
  quantDir = system.file("extdata/alevinfry_example_v0.5.0/quant",  
    package = "alevinQC"))
```

---

readAlevinQC	<i>Read alevin data required to generate summary report</i>
--------------	---

---

**Description**

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

**Usage**

```
readAlevinQC(baseDir, customCBLlist = list())
```

**Arguments**

baseDir	Path to the output directory from the alevin run (should be the directory containing the alevin directory).
customCBLlist	Named list with custom set(s) of barcodes to provide summary statistics/plots for, in addition to the whitelists generated by alevin.

**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

## \* **internal**

- alevinQC-package, [2](#)
  
- alevinFryQCReport (qcReport), [10](#)
- alevinFryQCShiny (qcShiny), [12](#)
- alevinQC (alevinQC-package), [2](#)
- alevinQC-package, [2](#)
- alevinQCReport (qcReport), [10](#)
- alevinQCShiny (qcShiny), [12](#)
  
- checkAlevinFryInputFiles, [3](#)
- checkAlevinInputFiles, [4](#)
  
- plotAlevinBarcodeCollapse, [4](#)
- plotAlevinHistogram, [5](#)
- plotAlevinKneeNbrGenes, [6](#)
- plotAlevinKneeRaw, [7](#)
- plotAlevinQuant, [8](#)
- plotAlevinQuantPairs, [9](#)
  
- qcReport, [10](#)
- qcShiny, [12](#)
  
- readAlevinFryQC, [14](#)
- readAlevinQC, [15](#)
  
- simpleafQCReport (qcReport), [10](#)
- simpleafQCShiny (qcShiny), [12](#)