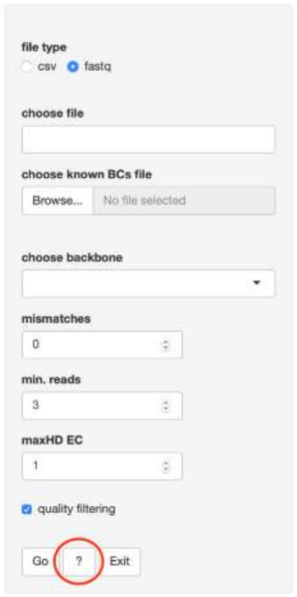
Introduction to the genBaRcode GUI

There is also a shiny-app included within the package, allowing you to use all main functionality of the package without typing any line of code at all. Or if you are well capable of programming you can also use it as a convenient method to learn about the possibilities of the package. There is an app-internal help and there is also an option to inspect the source code necessary to redo all in-app done analyses. You can start the app with the `genBaRcode_app()` command and if you already have a data file which you are dying to analyze you just need to provide the path to the directory (`dat_dir`) of this particular file and you can chose it from within the app. If you have none and no path provided, the package‘s internal example file will be available for exemplary analysis.

```r
# start Shiny app with the package internal test data file
genBaRcode_app()

# start Shiny app with access to a predefined directory
genBaRcode_app(dat_dir = "/path/to/my/data/")
```

After starting the app, the user has to provide basic informations like file type, file name, backbone structure, etc. By clicking on the button labeled with a question mark, the app internal help will be revealed (red circle).



file type
csv [✓ fastq]

choose file

choose known BCs file
Browse... No file selected

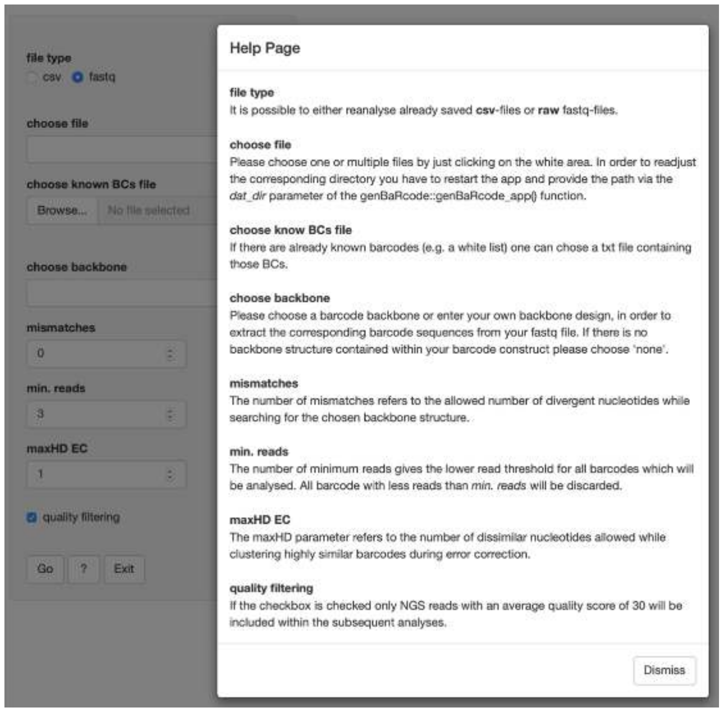
choose backbone

mismatches
0

min. reads
3

maxHD EC
1

✓ quality filtering



Help Page

file type
It is possible to either reanalyse already saved **csv**-files or **raw** fastq-files.

choose file
Please choose one or multiple files by just clicking on the white area. In order to readjust the corresponding directory you have to restart the app and provide the path via the *dat_dir* parameter of the genBaRcode::genBaRcode_app() function.

choose know BCs file
If there are already known barcodes (e.g. a white list) one can chose a txt file containing those BCs.

choose backbone
Please choose a barcode backbone or enter your own backbone design, in order to extract the corresponding barcode sequences from your fastq file. If there is no backbone structure contained within your barcode construct please choose 'none'.

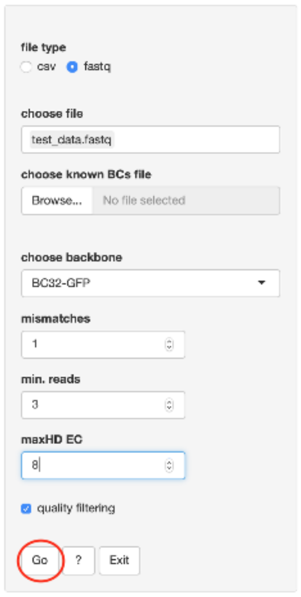
mismatches
The number of mismatches refers to the allowed number of divergent nucleotides while searching for the chosen backbone structure.

min. reads
The number of minimum reads gives the lower read threshold for all barcodes which will be analysed. All barcode with less reads than *min. reads* will be discarded.

maxHD EC
The maxHD parameter refers to the number of dissimilar nucleotides allowed while clustering highly similar barcodes during error correction.

quality filtering
If the checkbox is checked only NGS reads with an average quality score of 30 will be included within the subsequent analyses.

If no user specific input file containing folder was specified, the app will automatically make the example data file available which is included within the package. The following parameter choices would be appropriate.



file type
csv [✓ fastq]

choose file
test_data.fastq

choose known BCs file
Browse... No file selected

choose backbone
BC32-GFP

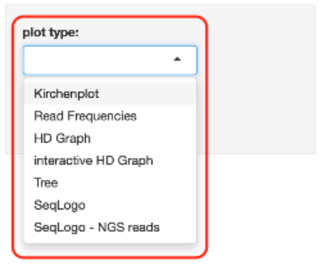
mismatches
1

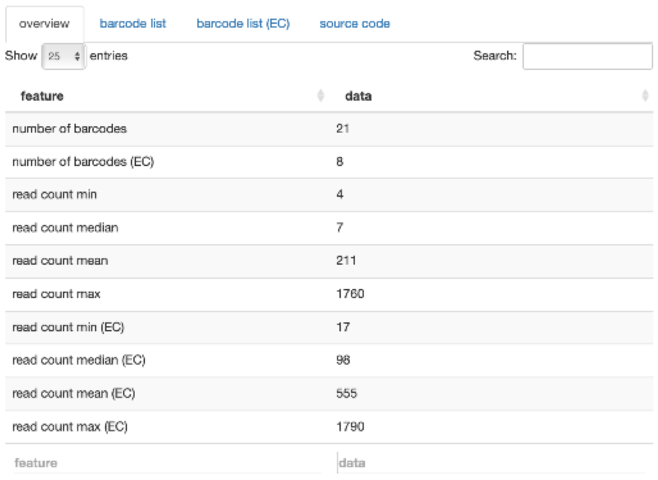
min. reads
3

maxHD EC
8

[✓] quality filtering

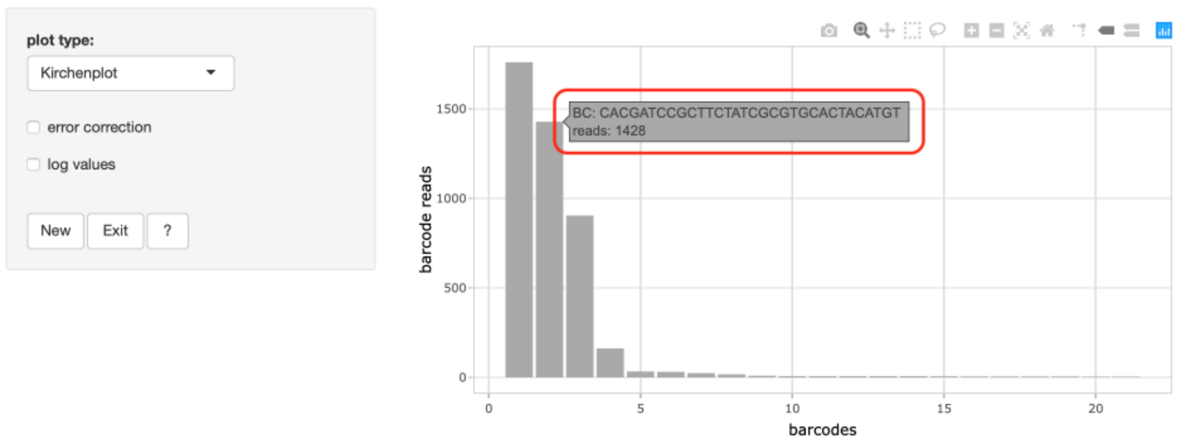
After starting the analysis by clicking the go button, a progress-bar will appear, unsurprisingly indicating the progress made so far. Then a dropdown menu with a variety of different plot types to choose from, an empty plot area and a table containing the most basic meta data will be visible.





feature	data
number of barcodes	21
number of barcodes (EC)	8
read count min	4
read count median	7
read count mean	211
read count max	1760
read count min (EC)	17
read count median (EC)	98
read count mean (EC)	555
read count max (EC)	1790
feature	data

After choosing a particular plot, the plot will be created and can instantaneously be modified. You can hover over certain parts of the plot to reveal additional informations and modify the displayed data, e.g. displaying the raw or error corrected data or change the scaling of axes.

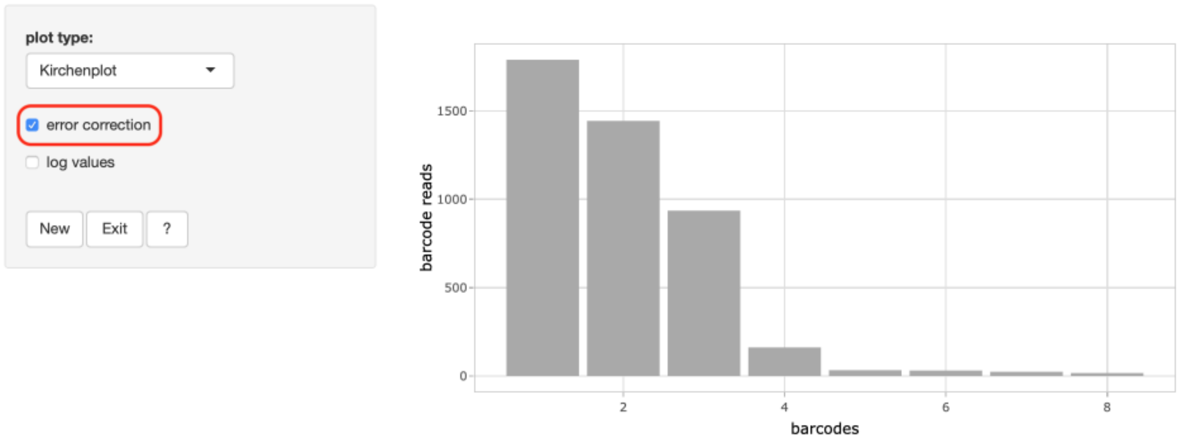


plot type: Kirchenplot

☐ error correction

☐ log values

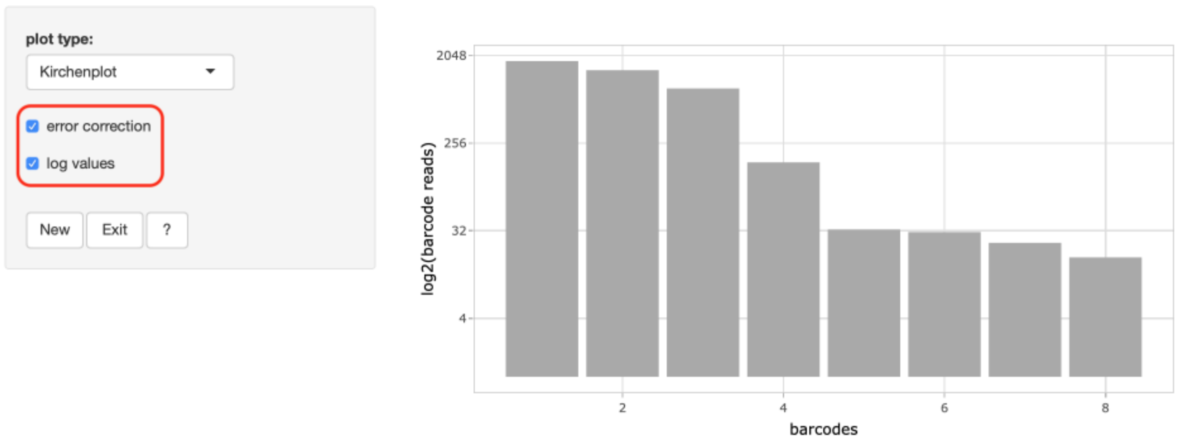
BC: CACGATCCGCTTCTATCGCGTGCACTACATGT
reads: 1428



plot type: Kirchenplot

☑ error correction

☐ log values

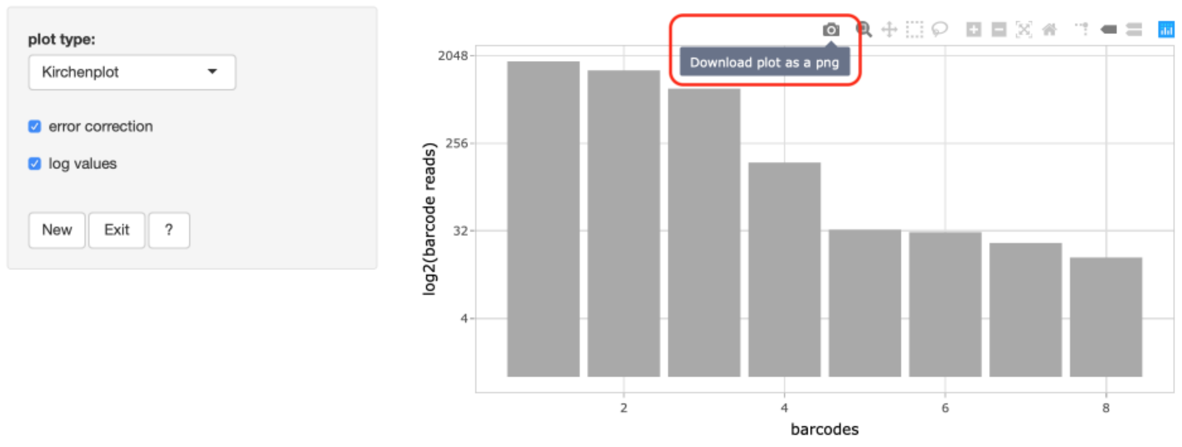


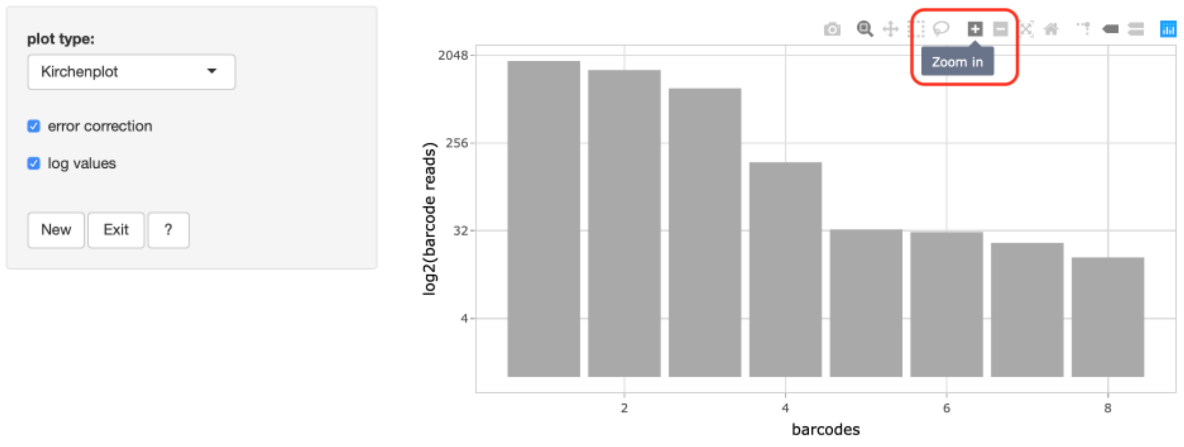
plot type: Kirchenplot

☑ error correction

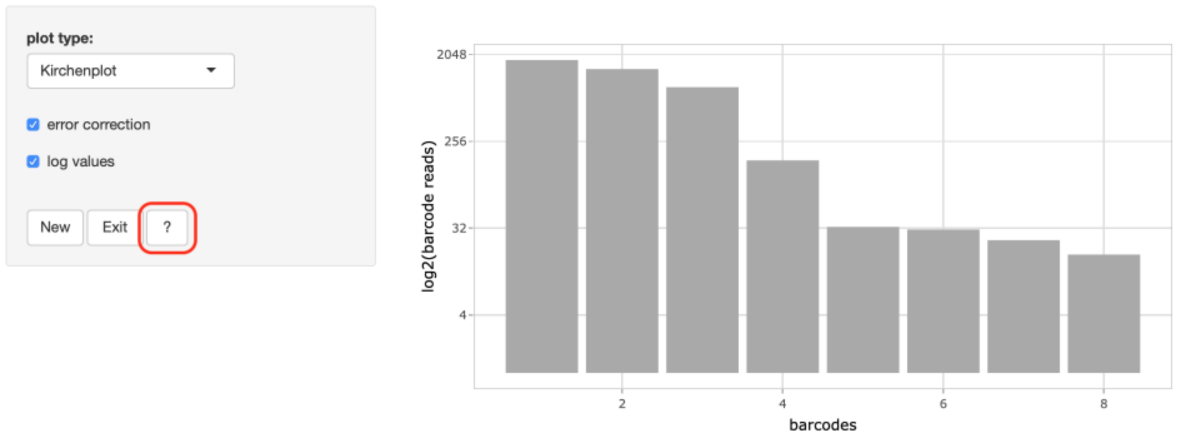
☑ log values

Additionally, since the `ggplotly` package was used there are a lot of further options available like zooming in and out, saving the entire plot as a *png* file or to box-select certain parts of the plot.



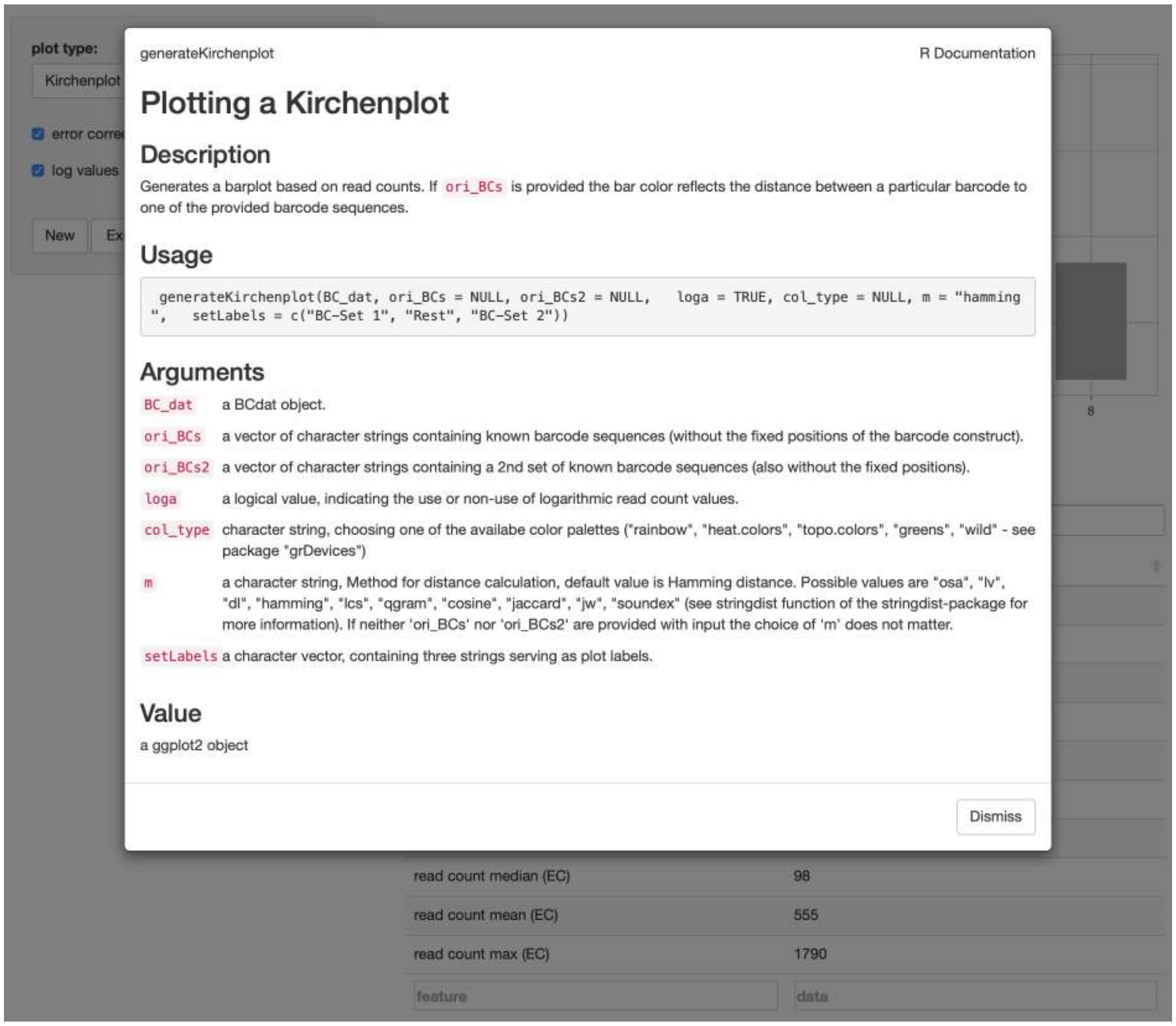


If there are questions regarding the already chosen plot type, there is also a button labeled with a question mark available, explaining all the necessary details regarding the specific plot.


plot type: Kirchenplot

☑ error correction

☑ log values


generateKirchenplot — R Documentation

Plotting a Kirchenplot

Description

Generates a barplot based on read counts. If `ori_BCs` is provided the bar color reflects the distance between a particular barcode to one of the provided barcode sequences.

Usage

```
generateKirchenplot(BC_dat, ori_BCs = NULL, ori_BCs2 = NULL,   loga = TRUE, col_type = NULL, m = "hamming
",   setLabels = c("BC-Set 1", "Rest", "BC-Set 2"))
```

Arguments

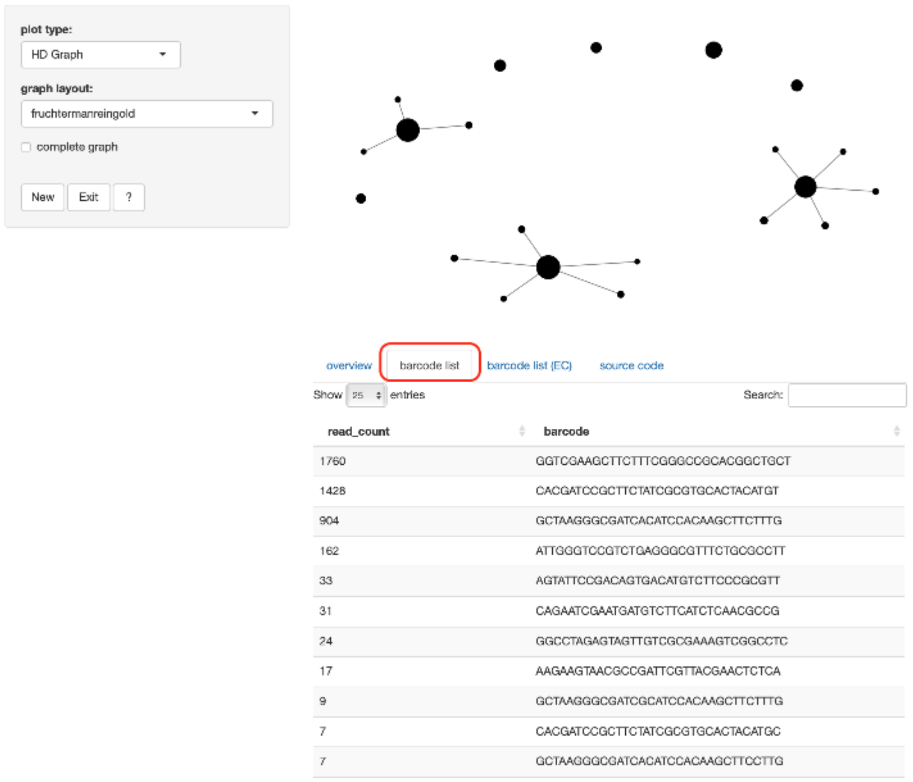
`BC_dat`	a BCdat object.
`ori_BCs`	a vector of character strings containing known barcode sequences (without the fixed positions of the barcode construct).
`ori_BCs2`	a vector of character strings containing a 2nd set of known barcode sequences (also without the fixed positions).
`loga`	a logical value, indicating the use or non-use of logarithmic read count values.
`col_type`	character string, choosing one of the availabe color palettes ("rainbow", "heat.colors", "topo.colors", "greens", "wild" - see package "grDevices")
`m`	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information). If neither 'ori_BCs' nor 'ori_BCs2' are provided with input the choice of 'm' does not matter.
`setLabels`	a character vector, containing three strings serving as plot labels.

Value

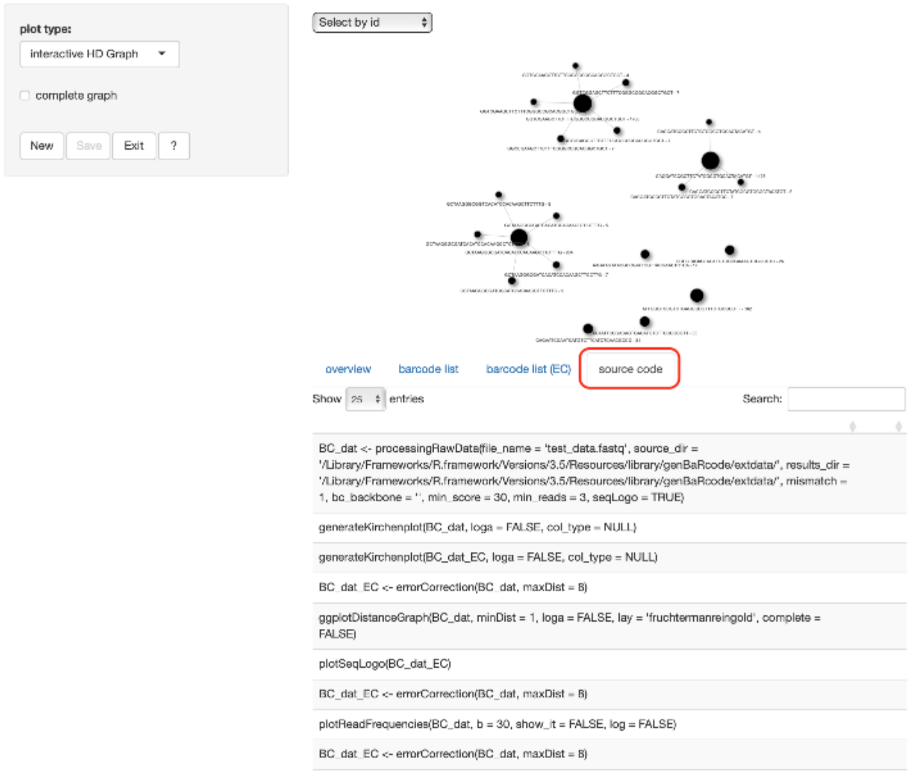
a ggplot2 object

read count median (EC)	98
read count mean (EC)	555
read count max (EC)	1790
feature	data

The included tables on the lower right side contain the meta-data, the barcode sequences, their read-counts and the raw source code necessary to redo all of the analysis steps done within the app directly within the R console. Here you can see the exemplary barcode list before the error correction and the corresponding source code.

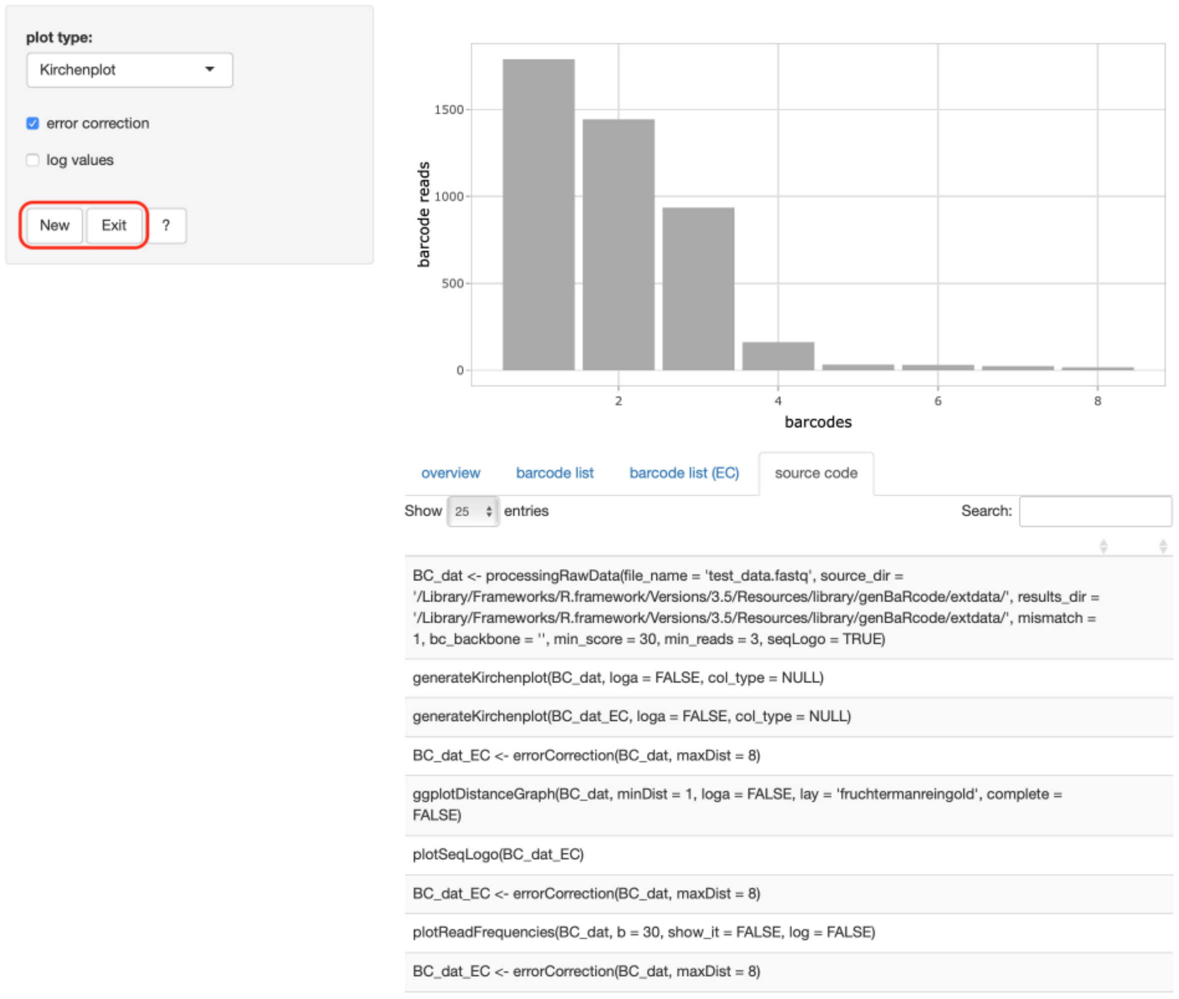


read_count	barcode
1760	GGTCGAAGCTTCTTTCGGGCCGCACGGCTGCT
1428	CACGATCCGCTTCTATCGCGTGCACTACATGT
904	GCTAAGGGCGATCACATCCACAAGCTTCTTTG
162	ATTGGGTCCGTCTGAGGGCGTTTCTGCGCCTT
33	AGTATTCCGACAGTGACATGTCTTCCCGCGTT
31	CAGAATCGAATGATGTCTTCATCTCAACGCCG
24	GGCCTAGAGTAGTTGTCGCGAAAGTCGGCCTC
17	AAGAAGTAACGCCGATTCGTTACGAACTCTCA
9	GCTAAGGGCGATCGCATCCACAAGCTTCTTTG
7	CACGATCCGCTTCTATCGCGTGCACTACATGC
7	GCTAAGGGCGATCACATCCACAAGCTTCCTTG



```
BC_dat <- processingRawData(file_name = 'test_data.fastq', source_dir = '/Library/Frameworks/R.framework/Versions/3.5/Resources/library/genBaRcode/extdata/', results_dir = '/Library/Frameworks/R.framework/Versions/3.5/Resources/library/genBaRcode/extdata/', mismatch = 1, bc_backbone = '', min_score = 30, min_reads = 3, seqLogo = TRUE)
generateKirchenplot(BC_dat, loga = FALSE, col_type = NULL)
generateKirchenplot(BC_dat_EC, loga = FALSE, col_type = NULL)
BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8)
ggplotDistanceGraph(BC_dat, minDist = 1, loga = FALSE, lay = 'fruchtermanreingold', complete = FALSE)
plotSeqLogo(BC_dat_EC)
BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8)
plotReadFrequencies(BC_dat, b = 30, show_it = FALSE, log = FALSE)
BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8)
```

Finally, if you decide to start another analysis or to exit the app entirely, there are the appropriate buttons available.



```
BC_dat <- processingRawData(file_name = 'test_data.fastq', source_dir =
'/Library/Frameworks/R.framework/Versions/3.5/Resources/library/genBaRcode/extdata/', results_dir =
'/Library/Frameworks/R.framework/Versions/3.5/Resources/library/genBaRcode/extdata/', mismatch =
1, bc_backbone = '', min_score = 30, min_reads = 3, seqLogo = TRUE)

generateKirchenplot(BC_dat, loga = FALSE, col_type = NULL)

generateKirchenplot(BC_dat_EC, loga = FALSE, col_type = NULL)

BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8)

ggplotDistanceGraph(BC_dat, minDist = 1, loga = FALSE, lay = 'fruchtermanreingold', complete =
FALSE)

plotSeqLogo(BC_dat_EC)

BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8)

plotReadFrequencies(BC_dat, b = 30, show_it = FALSE, log = FALSE)

BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8)
```