

Package: NACHO (via r-universe)

October 19, 2026

Type Package

Title NanoString Quality Control Dashboard

Version 2.0.8.9000

Description NanoString nCounter data are gene expression assays where there is no need for the use of enzymes or amplification protocols and work with fluorescent barcodes (Geiss et al. (2018) <[doi:10.1038/nbt1385](https://doi.org/10.1038/nbt1385)>). Each barcode is assigned a messenger-RNA/micro-RNA (mRNA/miRNA) which after bonding with its target can be counted. As a result each count of a specific barcode represents the presence of its target mRNA/miRNA. 'NACHO' (NANoString quality Control dasHbOard) is able to analyse the exported NanoString nCounter data and facilitates the user in performing a quality control. 'NACHO' does this by visualising quality control metrics, expression of control genes, principal components and sample specific size factors in an interactive web application.

License GPL-3

Copyright See file inst/COPYRIGHTS.

URL <https://github.com/mcanouil/NACHO/>, <https://m.canouil.dev/NACHO/>

BugReports <https://github.com/mcanouil/NACHO/issues>

Depends R (>= 4.3)

Imports methods, utils, grDevices, bslib (>= 0.10.0), brand.yml (>= 0.1.0), cli (>= 3.6.0), data.table, ggplot2 (>= 4.0.0), ggforce (>= 0.5.0), ggrepel (>= 0.9.6), htmltools (>= 0.5.9), promises (>= 1.3.2), rlang (>= 1.1.0), S7 (>= 0.2.2), scales (>= 1.4.0), shiny (>= 1.11.0), yaml (>= 2.3.12)

Suggests chromote (>= 0.5.1), colorspace (>= 2.1-3), gdtools (>= 0.5.1), ggiraph (>= 0.9.6), jsonlite (>= 2.0.0), knitr (>= 1.39), later, mirai (>= 2.7.3), parallel, png (>= 0.1-9), quarto (>= 1.5.1), reactable (>= 0.4.5), rmarkdown (>= 1.16), roxygen2 (>= 8.1.0), shinytest2 (>= 0.5.1), testthat (>= 3.1.5), vdiffr (>= 1.0.9), withr, covr (>= 3.3.2), Biobase,

GEOquery, limma, NanoStringNCTools, NormqPCR, RUVSeq,
S4Vectors, SummarizedExperiment

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

SystemRequirements Quarto command-line interface (>= 1.9.18), pandoc (>= 2.11)

Config/Needs/website pdftools

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Collate 'GSE74821.R' 'NACHO-package.R' 'thresholds.R' 'qc.R'
'nacho-class.R' 'accessors.R' 'daemon.R' 'render.R'
'mod_export.R' 'qc-table.R' 'report.R' 'mod_qc_plot.R'
'mod_batch.R' 'mod_outliers.R' 'mod_overview.R'
'mod_thresholds.R' 'load_rcc.R' 'mod_data.R' 'tour.R' 'app.R'
'autoplot.R' 'stability.R' 'ruv.R' 'mirna.R' 'batch.R'
'brand.R' 'conditions.R' 'deploy.R' 'detection.R' 'interop.R'
'methods.R' 'normalise.R' 'plot_worker.R' 'read_nacho.R'
'read_rcc.R' 'utils.R' 'visualise.R' 'zzz.R'

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev make libuv1-dev zlib1g-dev

Repository <https://mcanouil.r-universe.dev>

Date/Publication 2026-10-19 19:00:24 UTC

RemoteUrl <https://github.com/mcanouil/NACHO>

RemoteRef HEAD

RemoteSha bb34296e3e39e98bd24b03bdc5b496c5a18891e5

Contents

as_summarized_experiment	3
autoplot.nacho	4
batch_diagnostics	5
deploy	6
exclude_outliers	7
filter_detected	8
GSE74821	9
housekeeping_stability	9
load_rcc	10
nacho-accessors	13
nacho-methods	14
nacho_app	15
nacho_thresholds	16

normalise	17
read_nacho	19
render	20
suggest_ruv_k	21
upgrade_nacho	22
visualise	23
write_thresholds	24

Index	26
--------------	-----------

as_summarized_experiment	
	<i>Convert between nacho objects and Bioconductor containers</i>

Description

as_summarized_experiment() gives a SummarizedExperiment with the raw and normalised counts as assays counts and normalised, [nacho_samples\(\)](#) as colData, [nacho_probes\(\)](#) as rowData, and the settings, thresholds, RCC type and provenance in metadata()[\\$nacho](#).

Usage

```
as_summarized_experiment(x)

as_nacho(x, id_colname = "IDFILE")
```

Arguments

x	A nacho object for as_summarized_experiment(); a SummarizedExperiment or a NanoStringRccSet for as_nacho().
id_colname	Name of the sample id column to create when x does not come from NACHO. When metadata(x) \$nacho\$settings exists, its id_colname wins and this argument is ignored.

Details

as_nacho() goes the other way, from a SummarizedExperiment or from a NanoStringRccSet read by NanoStringNCTools::readNanoStringRccSet(). The dates of a NanoStringRccSet are written as YYYYMMDD, as in RCC files. Its probe names lose a trailing numeric suffix such as |0, as in [load_rcc\(\)](#), so both give the same probe names for the same RCC files. It computes the quality-control metrics and the normalisation from the raw counts; settings and thresholds come from metadata()[\\$nacho](#) when present, and are the defaults otherwise. They are checked like the arguments of [normalise\(\)](#). The result always carries the current object schema. A nacho object is checked and returned as is, and a NACHO 2 object gets a hint to use [upgrade_nacho\(\)](#).

Value

A SummarizedExperiment, or a nacho object.

Examples

```
if (requireNamespace("SummarizedExperiment", quietly = TRUE)) {  
  se <- as_summarized_experiment(GSE74821)  
  as_nacho(se)  
}
```

autoplot.nacho

Plot the quality control of a nacho object

Description

Draws any of the quality-control figures of the Shiny app ([visualise\(\)](#)) and of the HTML or PDF report ([render\(\)](#)).

Value

A ggplot object.

Usage

```
autoplot(  
  object,  
  type,  
  colour = "CartridgeID",  
  size = 0.5,  
  show_legend = TRUE,  
  show_outliers = TRUE,  
  outliers_factor = 1,  
  outliers_labels = NULL,  
  dark = FALSE  
)
```

Arguments

- object: A nacho object from [load_rcc\(\)](#) or [normalise\(\)](#).
- type: The plot to draw, one of:
 - "BD": Binding Density.
 - "FoV": Field of View (imaging).
 - "PCL": Positive Control Linearity.
 - "LoD": Limit of Detection.
 - "Positive": Positive controls.
 - "Negative": Negative controls.
 - "Housekeeping": Housekeeping genes.
 - "PN": Positive controls against negative controls.
 - "ACBD": Average counts against binding density.

- "ACMC": Average counts against median counts.
 - "PCA12": Principal component 1 against 2.
 - "PCAi": Scree plot of the principal components.
 - "PCA": Planes of the first principal components.
 - "PFNF": Positive factor against negative factor.
 - "HF": Housekeeping factor.
 - "NORM": Normalisation factor.
 - "Stability": geNorm stability of the housekeeping genes, see [housekeeping_stability\(\)](#). The dashed line marks $M = 1.5$, the limit geNorm suggests for homogeneous samples. This plot ignores colour, show_legend, show_outliers and outliers_labels.
 - "RLE": Relative log expression of the normalised endogenous genes. This plot ignores size, show_outliers, outliers_factor and outliers_labels.
 - "BatchFactors": Normalisation factors by cartridge. This plot ignores show_outliers, outliers_factor and outliers_labels.
 - "PCBatch": Share of each principal component explained by cartridge and date; see [batch_diagnostics\(\)](#). Tiles with no value stay grey and unlabelled. This plot ignores colour, size, show_legend, show_outliers, outliers_factor and outliers_labels.
- colour: The column of nacho_samples(object) that colours the points.
 - size: The point size, and the line width in the "NORM" plot.
 - show_legend: If FALSE, hide the colour legend.
 - show_outliers: If TRUE, draw the flagged samples as triangles in the accent colour: rust on a light background, amber on a dark one.
 - outliers_factor: The size of the flagged samples, relative to size.
 - outliers_labels: The column of nacho_samples(object) that labels the flagged samples, or NULL for no labels. Labels imply show_outliers = TRUE.
 - dark: If TRUE, draw the plot for a dark background.

Examples

```
autoplot(GSE74821, type = "BD")
```

batch_diagnostics	<i>Batch and confounding diagnostics</i>
-------------------	--

Description

Checks whether the study design, the quality-control metrics and the main axes of variation follow technical batches such as cartridges and run dates.

Usage

```
batch_diagnostics(x, group = NULL, batch = c("CartridgeID", "Date"))
```

Arguments

x	A nacho object from <code>load_rcc()</code> or <code>normalise()</code> .
group	A column of <code>nacho_samples(x)</code> with the biological groups, or NULL to skip the design check.
batch	Columns of <code>nacho_samples(x)</code> that describe technical batches.

Details

Start with design: when a batch level holds a single group, batch and biology cannot be told apart, and no normalisation can fix it.

Value

A list:

- `design`: NULL without group; otherwise, for each batch variable, its number of levels, Cramér's V with group, the number of levels that hold a single group, and confounded, TRUE when at least one does. A level counts as holding a single group only when it has two or more samples with a group, so a cartridge with one sample never makes the design confounded. Cramér's V uses only the samples that have both a group and a batch, and is NA when the batch or group has fewer than two levels among them.
- `crosstabs`: NULL without group; otherwise the table of group against each batch variable.
- `metrics`: Kruskal-Wallis tests of each quality-control metric against each batch variable.
- `pc_batch`: the share of each principal component's variance explained by each batch variable.

`metrics` and `pc_batch` are data frames with these columns, and have no rows when there is nothing to report. A batch variable with one level gives NA in `statistic`, `p_value` and `r_squared`, and a constant metric gives NA in `statistic` and `p_value`. A batch variable whose levels each hold one sample with a value also gives NA in `r_squared`. For PlexSet data, BD and FoV are tested once per lane.

Examples

```
data(GSE74821)
batch_diagnostics(GSE74821)$pc_batch
```

deploy

Deploy (copy) the Shiny application to the specified directory

Description

Deploy (copy) the Shiny application to the specified directory

Usage

```
deploy(directory, app_name = "NACHO")
```

Arguments

directory [[character](#)] A character vector of one path to the new location.

app_name [[character](#)] A character vector defining the Shiny application name in the new location.

Details

The deployed app calls `NACHO::nacho_app()`, so the server needs NACHO installed. The app sets the `nacho.plot_workers` option to 1 when the option is not set, so it starts one background process for the interactive plots. That process and the plot cache belong to the R process that runs the app, and each process keeps a copy of the study, so its memory grows with the size of the study (about 170 MB for GSE74821). To change the number of processes, set the option before the app starts, for example in `.Rprofile` on the server. See the "Plot workers" section of [nacho_app\(\)](#).

Value

[\[logical\]](#) A logical indicating whether the deployment is successful (TRUE) or not (FALSE).

Examples

```
deploy(directory = tempdir())

if (interactive()) {
  shiny::runApp("NACHO")
}
```

exclude_outliers	<i>Drop outliers and normalise the other samples again</i>
------------------	--

Description

Removes the samples whose status in [nacho_qc\(\)](#) is "fail" and runs the normalisation again on the samples that are left, with the same settings and thresholds. The new factors can flag more samples; call `exclude_outliers()` again to drop those too.

Usage

```
exclude_outliers(nacho_object)
```

Arguments

nacho_object A nacho object from [load_rcc\(\)](#) or [normalise\(\)](#).

Value

A nacho object without the flagged samples.

Examples

```
data(GSE74821)
ncol(exclude_outliers(GSE74821))
```

filter_detected	<i>Keep genes detected in enough samples</i>
-----------------	--

Description

A gene is detected in a sample when its raw count is above the mean plus two standard deviations of that sample's negative controls. Control and housekeeping probes are always kept, where housekeeping probes are the probes of CodeClass Housekeeping. Endogenous genes chosen as housekeeping genes are filtered like any other gene, and the dropped ones are removed from the housekeeping_genes setting, which becomes NULL when none is left. Then the object uses its Housekeeping probes and warns, or, with no Housekeeping probe either, turns housekeeping_norm off and warns. The samples' Detection_rate and Housekeeping_detected are not recomputed after filtering. A gene's detection rate is the share of the samples with a detection limit in which it is detected, so a sample with fewer than two negative probes does not count. Endogenous genes whose detection rate is missing, because every sample with a detection limit has a missing count for them, are dropped, even with min_rate = 0. With all-zero negative controls the detection limit is 0, so every non-zero count counts as detected. An object without endogenous genes comes back unchanged. When no sample has a detection limit, filter_detected() stops with an error.

Usage

```
filter_detected(x, min_rate = 0.5)
```

Arguments

x	A nacho object from <code>load_rcc()</code> or <code>normalise()</code> .
min_rate	The smallest share of samples with a detection limit, between 0 and 1, in which an endogenous gene must be detected.

Value

A nacho object with fewer probes.

Examples

```
data(GSE74821)
dim(filter_detected(GSE74821, min_rate = 0.5))
```

GSE74821	<i>A "nacho" object with 48 samples from the GSE74821 dataset</i>
----------	---

Description

The 48 samples come from four cartridges, normalised with the "GLM" method.

Usage

GSE74821

Format

A nacho object with 48 samples and 72 probes.

Details

NanoString nCounter RUO-PAM50 Gene Expression Custom CodeSet

Source

GSE74821

housekeeping_stability	<i>Rank genes by expression stability</i>
------------------------	---

Description

Ranks candidate reference genes with geNorm (Vandesompele et al. 2002) and NormFinder (Andersen et al. 2004), after keeping the genes detected in most samples. With group, NormFinder accounts for the groups, and a Kruskal-Wallis test tells whether each gene's expression differs between them, which makes a poor reference gene.

Usage

housekeeping_stability(x, genes = NULL, group = NULL, min_detection = 0.9)

Arguments

x	A nacho object from load_rcc() or normalise() .
genes	The genes to rank; NULL ranks the Housekeeping probes.
group	A column of nacho_samples(x) with the biological groups, or NULL.
min_detection	The smallest share of samples in which a gene must be above background to be ranked.

Details

The two most stable genes share the same geNorm M by construction, so their order, ranks 1 and 2, is arbitrary. Without group, NormFinder_rho is the standard deviation of the residual variation of the gene. With group, it also counts the variation between the groups, shrunk towards zero as NormFinder does.

Value

A list with ranking, a data frame with one row per gene from the most to the least stable: Name, CodeClass, detection_rate, mean_log2, geNorm_M, geNorm_rank, NormFinder_rho and, with group, group_p_value and group_p_adjusted (Benjamini-Hochberg); and pairwise_v, the geNorm pairwise variation V for each number of genes. A V below 0.15 is the usual sign that adding a gene no longer helps.

Examples

```
data(GSE74821)
housekeeping_stability(GSE74821)$ranking
```

load_rcc

Produce a "nacho" object from RCC NanoString files

Description

This function is used to preprocess the data from NanoString nCounter.

Usage

```
load_rcc(
  data_directory,
  ssheet_csv,
  id_colname = NULL,
  housekeeping_genes = NULL,
  housekeeping_predict = FALSE,
  housekeeping_norm = NULL,
  normalisation_method = "GEO",
  background = "none",
  background_mode = "threshold",
  ruv_k = NULL,
  instrument = NULL,
  preset = "nsolver",
  n_comp = 10
)
```

Arguments

- data_directory** [[character](#)] A character string of the directory where the data are stored.
- ssheet_csv** [[character](#)] or [[data.frame](#)] Either a string with the name of the CSV of the samplesheet or the samplesheet as a data.frame. Should contain a column that matches the file names in the folder. NACHO computes the columns Date, ID, BD, ScannerID, StagePosition, CartridgeID, FoV, PCL, LoD, MC, MedC, Positive_factor, Negative_factor, Detection_rate, Background, House_factor, Housekeeping_detected, Ligation_order, Ligation_R2, Ligation_NEG, Haemolysis, is_outlier, PC1, PC2, ... and W_1, W_2, ... A sample sheet column with one of these names is replaced, with a warning.
- id_colname** [[character](#)] Character string of the column in `ssheet_csv` that matches the file names in `data_directory`.
- housekeeping_genes** [[character](#)] A vector of names of the miRNAs/mRNAs that should be used as housekeeping genes. Default is `NULL`.
- housekeeping_predict** [[logical](#)] Boolean to indicate whether the housekeeping genes should be predicted (`TRUE`) or not (`FALSE`). Default is `FALSE`. Prediction picks the five most stable genes by `geNorm` on the raw counts. It is skipped for the miRNA methods of `normalisation_method`.
- housekeeping_norm** [[logical](#)] Boolean to indicate whether the housekeeping normalisation should be performed. `NULL` (the default) normalises with housekeeping genes for mRNA panels that have them. On miRNA panels, whose housekeeping mRNAs sit at background, it does so only when you pass `housekeeping_genes` or set `housekeeping_predict = TRUE`. Pass `TRUE` or `FALSE` to override this. The miRNA methods of `normalisation_method` ignore it.
- normalisation_method** [[character](#)] "GEO" (the default) or "GLM" scale samples by their positive controls, with the geometric mean or a Poisson model, then by the housekeeping genes; "RUVg" scales by the positive controls with the geometric mean, then removes `ruv_k` factors of unwanted variation estimated from the housekeeping genes, which `nacho_samples()` returns as `W_1`, `W_2`, ... for use as covariates. For miRNA panels, "stable_mirna", "total_mirna", "spike_in" and "ligation" scale by the five most stable miRNAs, the miRNAs above 50 counts, the spike-in controls or the ligation positive controls, in the order Bruker recommends for plasma and serum. These methods ignore `housekeeping_genes`, `housekeeping_predict` and `housekeeping_norm`, and give their scaling factor as `House_factor`. When a GLM fit fails, NACHO warns and uses the geometric mean instead, as `normalise()` describes.
- background** [[character](#)] How to estimate each sample's background from its negative controls: "none" (the default), "mean", "mean_2sd" (mean plus two standard deviations), "median", "max" or "geo" (geometric mean).
- background_mode** [[character](#)] What to do with that background: "threshold" (the default) raises lower counts to it, which Bruker recommends when fold changes matter, and "subtract" removes it and floors at 0.

ruv_k	[numeric] The number of unwanted factors RUVg removes; NULL uses suggest_ruv_k() . A ruv_k above two less than the number of samples, or one less than the number of control genes, is lowered with a nacho_warning_ruv_k_reduced warning. Other methods ignore it.
instrument	[character] The nCounter instrument, "max", "flex", "pro" or "sprint", which sets the binding density limits. NULL (the default) reads it from the RCC files when they say, and otherwise uses the MAX/FLEX limits with a warning.
preset	[character] "nsolver" (the default) or "legacy"; see nacho_thresholds() .
n_comp	[numeric] Number indicating the number of principal components to compute. Cannot be more than n-1 samples. Default is 10.

Value

A nacho object; read its content with [nacho_counts\(\)](#), [nacho_samples\(\)](#), [nacho_probes\(\)](#) and [nacho_qc\(\)](#).

Examples

```
if (interactive()) {
  library(GEOquery)
  library(NACHO)

  # Import data from GEO
  gse <- GEOquery::getGEO(GEO = "GSE74821")
  targets <- Biobase::pData(Biobase::phenoData(gse[[1]]))
  GEOquery::getGEOSuppFiles(GEO = "GSE74821", baseDir = tempdir())
  utils::untar(
    tarfile = file.path(tempdir(), "GSE74821", "GSE74821_RAW.tar"),
    exdir = file.path(tempdir(), "GSE74821")
  )
  targets$IDFILE <- list.files(
    path = file.path(tempdir(), "GSE74821"),
    pattern = ".RCC.gz$"
  )
  targets[] <- lapply(X = targets, FUN = iconv, from = "latin1", to = "ASCII")
  utils::write.csv(
    x = targets,
    file = file.path(tempdir(), "GSE74821", "Samplesheet.csv")
  )

  # Read RCC files and format
  nacho <- load_rcc(
    data_directory = file.path(tempdir(), "GSE74821"),
    ssheet_csv = file.path(tempdir(), "GSE74821", "Samplesheet.csv"),
    id_colname = "IDFILE"
  )
}
```

nacho-accessors	<i>Get the content of a nacho object</i>
-----------------	--

Description

These accessors are the only public way into a nacho object.

Usage

```
nacho_counts(x, normalised = FALSE, log2 = FALSE)
```

```
nacho_samples(x)
```

```
nacho_probes(x)
```

```
nacho_qc(x)
```

Arguments

<code>x</code>	A nacho object from <code>load_rcc()</code> or <code>normalise()</code> .
<code>normalised</code>	If TRUE, return the normalised counts instead of the raw counts.
<code>log2</code>	If TRUE, return $\log_2(\text{count} + 1)$.

Value

- `nacho_counts()`: a matrix with one row per probe and one column per sample.
- `nacho_samples()`: a data frame with one row per sample; the first column holds the sample ids, and the PCA scores come last (PC01, PC02, ...). `Detection_rate` (the share of endogenous genes detected) and `Housekeeping_detected` (the number of housekeeping genes detected) are NA for a sample without a detection limit, and `Housekeeping_detected` is also NA for a miRNA panel that skips housekeeping normalisation. miRNA panels add `Ligation_order`, `Ligation_R2` and `Ligation_NEG` (the ligation controls), and `Haemolysis`, the log2 ratio of miR-451a to miR-23a-3p, when those probes are present.
- `nacho_probes()`: a data frame with one row per probe: `CodeClass`, `Name`, `Accession`, `detection_rate` (the share of samples with a detection limit in which the probe is above it), `is_housekeeping` and `is_excluded`.
- `nacho_qc()`: a data frame with one row per sample: the id, lane, `CartridgeID`, each quality-control metric with its status ("pass", "fail" or NA when it cannot be assessed), the other factors, `lane_status` for PlexSet files, `n_flags` (the sample's own failing metrics, not those inherited from its lane), the overall status and a readable reason (which also names the lane metrics a sample inherits).

Examples

```
data(GSE74821)
nacho_counts(GSE74821)[1:5, 1:3]
head(nacho_samples(GSE74821)[, 1:5])
head(nacho_probes(GSE74821))
head(nacho_qc(GSE74821))
```

nacho-methods

Basic methods for a nacho object

Description

Basic methods for a nacho object

Usage

```
print(x)
format(x)
summary(object)
dim(x)
as.data.frame(x, long = FALSE)
x[i, j]
```

Arguments

- x, object: A nacho object.
- long: If TRUE, `as.data.frame()` returns one row per probe and sample (the NACHO 2 \$nacho layout) instead of one row per sample.
- i: Probes to keep.
- j: Samples to keep.

Examples

```
data(GSE74821)
print(GSE74821)
summary(GSE74821)
dim(GSE74821)
head(as.data.frame(GSE74821, long = TRUE)[, 1:5])
GSE74821[, 1:12]
```

nacho_app	<i>Run the NACHO app</i>
-----------	--------------------------

Description

Opens the quality-control app on a nacho object, or on an empty upload page. Use [visualise\(\)](#) to get the tuned object back in the R session.

Usage

```
nacho_app(x = NULL, done = FALSE)
```

Arguments

x	A nacho object from load_rcc() or normalise() , or NULL to start by uploading RCC files.
done	If TRUE, the app shows a "Done" button that closes it and returns the tuned object. Keep FALSE for a deployed app, because closing it would stop the server for every user.

Value

A Shiny app object; print it or pass it to [shiny::runApp\(\)](#).

Plot workers

When the mirai and ggiraph packages are installed, the app builds the interactive plots of a page at the same time, in background R processes. The processes and the plot cache belong to the R process that runs the app. All users of that R process share them. The app starts the processes when the first user opens a page with interactive plots, and stops them when the app stops. By default, the app starts one process for each CPU core minus one, with a maximum of four. The app from [deploy\(\)](#) starts one process.

Each process keeps a copy of the study that it draws, so its memory grows with the size of the study. For GSE74821, each process uses about 170 MB. To find the number of processes, divide the memory that you can give to the plots by the memory of one process. Set the `nacho.plot_workers` option to that number before you start the app, for example `options(nacho.plot_workers = 2)`. To build all plots in the app process, set the option to 0. If the processes do not start, the app builds the plots in its own process.

The processes use the installed NACHO package.

Examples

```
if (interactive()) {  
  shiny::runApp(nacho_app(GSE74821))  
}
```

nacho_thresholds	<i>Quality-control thresholds</i>
------------------	-----------------------------------

Description

Builds the thresholds that `load_rcc()` and `normalise()` use to flag samples.

Usage

```
nacho_thresholds(
  instrument = c("max", "flex", "pro", "sprint"),
  preset = c("nsolver", "legacy"),
  haemolysis = FALSE
)
```

Arguments

instrument	The nCounter instrument: "max", "flex", "pro" or "sprint".
preset	"nsolver" or "legacy".
haemolysis	If TRUE, flag samples of a miRNA panel whose miR-451a to miR-23a-3p log2 ratio is above 7, a sign of haemolysis in plasma and serum. <code>load_rcc()</code> does not take it, so pass the thresholds to <code>normalise(outliers_thresholds =)</code> .

Details

"nsolver" follows the current Bruker nCounter knowledge base (<https://brukerspatialbiology.com/support/>, "nCounter QC flags"): binding density 0.05 to 2.25 on MAX, FLEX and PRO and 0.1 to 1.8 on SPRINT, field of view at least 75 %, positive control linearity at least 0.95 (POS_A to POS_E, with $\log_2(\text{count} + 1)$), limit of detection at least 2, positive factor 0.3 to 3 and housekeeping factor 0.1 to 10. It also asks for at least three housekeeping genes above background, as Bruker asks (Housekeeping_detected).

"legacy" reproduces NACHO 2: binding density 0.1 to 2.25 on every instrument, positive control linearity with POS_F, positive factor 1/4 to 4 and housekeeping factor 1/11 to 11. With `background = "geo"`, `background_mode = "subtract"`, it gives back the NACHO 2 outlier calls.

The preset also chooses how negative probes are excluded and how the positive control linearity is computed, so changing it normalises again.

For miRNA panels, `Ligation_order` (the three ligation positive controls in decreasing order), `Ligation_R2` (their log2 counts on a line) and `Ligation_NEG` (the largest ligation negative minus the detection limit) are NACHO's own definitions, since Bruker publishes no threshold. "legacy" never flags them.

Whole-blood RNA raises `LIG_NEG_A`, and to a lesser extent `LIG_NEG_B`, together with red blood cell miRNAs such as miR-451a. The ligation negatives of such samples are above the detection limit, so most samples of a whole-blood study fail `Ligation_NEG`. This pattern occurs in independent PAXgene whole-blood studies on GEO (GSE303669 and GSE317496), and not in plasma, serum or

extracellular vesicle studies. When more than half of the samples fail `Ligation_NEG`, `load_rcc()` warns about this pattern. For whole-blood samples, set a higher upper bound for `Ligation_NEG` and pass the thresholds to `normalise(outliers_thresholds = )`. Do not remove these samples with `exclude_outliers()` because of this flag only.

Each element can be changed afterwards; two numbers are a lower and an upper bound, and one number is a lower bound. Use `-Inf` or `Inf` for an open bound.

Value

A named list: `preset`, `instrument`, then one element per metric.

Examples

```
nacho_thresholds()
nacho_thresholds("sprint")
nacho_thresholds(preset = "legacy")
```

normalise	<i>(Re)normalise a nacho object</i>
-----------	-------------------------------------

Description

(Re)normalise a nacho object

Usage

```
normalise(
  nacho_object,
  housekeeping_genes = nacho_object@settings[["housekeeping_genes"]],
  housekeeping_predict = nacho_object@settings[["housekeeping_predict"]],
  housekeeping_norm = nacho_object@settings[["housekeeping_norm"]],
  normalisation_method = nacho_object@settings[["normalisation_method"]],
  background = nacho_object@settings[["background"]],
  background_mode = nacho_object@settings[["background_mode"]],
  ruv_k = nacho_object@settings[["ruv_k"]],
  n_comp = nacho_object@settings[["n_comp"]],
  outliers_thresholds = nacho_object@thresholds,
  ...
)
```

Arguments

`nacho_object` A nacho object from `load_rcc()` or `normalise()`.

`housekeeping_genes` **[character]** A vector of names of the miRNAs/mRNAs that should be used as housekeeping genes. Default is `NULL`.

housekeeping_predict	[logical] Boolean to indicate whether the housekeeping genes should be predicted (TRUE) or not (FALSE). Default is FALSE. Prediction picks the five most stable genes by geNorm on the raw counts. It is skipped for the miRNA methods of normalisation_method.
housekeeping_norm	[logical] Boolean to indicate whether the housekeeping normalisation should be performed. The default is the setting stored in nacho_object, which is TRUE or FALSE.
normalisation_method	[character] "GEO" (the default) or "GLM" scale samples by their positive controls, with the geometric mean or a Poisson model, then by the housekeeping genes; "RUVg" scales by the positive controls with the geometric mean, then removes ruv_k factors of unwanted variation estimated from the housekeeping genes, which nacho_samples() returns as W_1, W_2, ... for use as covariates. For miRNA panels, "stable_mirna", "total_mirna", "spike_in" and "ligation" scale by the five most stable miRNAs, the miRNAs above 50 counts, the spike-in controls or the ligation positive controls, in the order Bruker recommends for plasma and serum. These methods ignore housekeeping_genes, housekeeping_predict and housekeeping_norm, and give their scaling factor as House_factor. When a GLM fit fails, NACHO warns and uses the geometric mean instead, as normalise() describes.
background	[character] How to estimate each sample's background from its negative controls: "none" (the default), "mean", "mean_2sd" (mean plus two standard deviations), "median", "max" or "geo" (geometric mean).
background_mode	[character] What to do with that background: "threshold" (the default) raises lower counts to it, which Bruker recommends when fold changes matter, and "subtract" removes it and floors at 0.
ruv_k	[numeric] The number of unwanted factors RUVg removes. normalise() reuses the ruv_k stored in the object, so pass ruv_k = NULL to have suggest_ruv_k() choose again. A ruv_k above two less than the number of samples, or one less than the number of control genes, is lowered with a nacho_warning_ruv_k_reduced warning. Other methods ignore it.
n_comp	[numeric] Number indicating the number of principal components to compute. Cannot be more than n-1 samples. Default is 10.
outliers_thresholds	A list of quality-control thresholds, as returned by nacho_thresholds() .
...	Must be empty.

Details

When only the limits in outliers_thresholds change and the preset stays the same, [normalise\(\)](#) keeps the counts and stores the new thresholds; the flags follow them whenever you read [nacho_qc\(\)](#). Otherwise it computes the quality-control metrics, the normalisation factors, the normalised counts and the PCA again.

[nacho_qc\(\)](#) lists the flags, and [exclude_outliers\(\)](#) drops the flagged samples.

With `normalisation_method = "GLM"`, each sample's positive and negative control counts plus 1 are fitted against their known concentrations with a Poisson GLM and an identity link, and the positive factor is the mean slope over the sample's slope. When the fit fails for any sample, NACHO warns and uses the geometric mean for every sample, so all samples stay on the same scale. The recorded `normalisation_method` then becomes "GEO", and the failed samples are listed in `provenance$glm_fallback`.

The normalisation runs in this order: raw counts, negative probe exclusion, background (background and background_mode), positive control factor (`normalisation_method`), then content factor (housekeeping genes when `housekeeping_norm` is TRUE). RUVg works on $\log_2(\text{count} + 1)$ after the positive factor and returns counts on the count scale, floored at 0. It replaces the housekeeping scaling, so `housekeeping_norm` has no effect and `House_factor` is not computed, and it corrects only the endogenous and housekeeping probes. The miRNA methods "stable_mirna", "total_mirna", "spike_in" and "ligation" also ignore `housekeeping_norm`, `housekeeping_genes` and `housekeeping_predict`. They store their factor in `House_factor`, which is judged against the `House_factor` threshold, and the probes they used in `provenance$content_probes`. Normalised counts are otherwise neither rounded nor floored; use `nacho_counts(x, normalised = TRUE, log2 = TRUE)` for $\log_2(\text{count} + 1)$.

Value

A nacho object.

Examples

```
data(GSE74821)
GSE74821_geo <- normalise(GSE74821, normalisation_method = "GEO")
nacho_qc(GSE74821_geo)
```

read_nacho	<i>Read a saved NACHO object</i>
------------	----------------------------------

Description

Reads an .rds file written with `saveRDS()` from any NACHO version. NACHO 2 objects go through `upgrade_nacho()`. NACHO 3 objects are rebuilt with the current class, because each saved S7 object carries a copy of the class it was made with.

Usage

```
read_nacho(path)
```

Arguments

path Path to an .rds file.

Details

Only read files you trust: `readRDS()` can run code stored in a crafted file, notably in R before 4.4.0.

Value

A nacho object.

See Also

[upgrade_nacho\(\)](#)

Examples

```
path <- tempfile(fileext = ".rds")
saveRDS(GSE74821, path)
read_nacho(path)
```

render

Render the quality-control report of a nacho object

Description

Writes a quality-control report with Quarto. The report starts with a cover that gives the study details. Then comes a decision summary, with the flagged samples and notes on how the data were processed. A table gives each setting and limit, with its meaning and its source. Each plot of [autoplot\(\)](#) and each table has a short "How to read this". A methods appendix gives the citation and the package versions.

Usage

```
render(
  x,
  format = c("html", "typst"),
  output_dir = ".",
  colour = "CartridgeID",
  group = NULL,
  size = 1,
  show_legend = TRUE,
  outliers_factor = 1,
  outliers_labels = NULL,
  title = NULL,
  author = NULL
)
```

Arguments

x	A nacho object from load_rcc() or normalise() .
format	"html" (the default) for a self-contained HTML file, or "typst" for a PDF.
output_dir	The directory to write nacho-report.html or nacho-report.pdf to. It is created when it does not exist.

colour	The column of <code>nacho_samples(x)</code> that colours the samples.
group	The column of <code>nacho_samples(x)</code> with the biological groups, or NULL. With a group, the report checks whether batches and groups are confounded; see batch_diagnostics() .
size	The point size.
show_legend	If FALSE, hide the colour legends.
outliers_factor	The size of flagged samples, relative to size.
outliers_labels	The column of <code>nacho_samples(x)</code> that labels the flagged samples, or NULL for no labels.
title	The report title. NULL or empty uses "NanoString quality-control report". A missing value (NA) is an error.
author	Who prepared the report, as one string, for example "Jane Doe, Genomics Core". NULL or empty leaves it out. A missing value (NA) is an error.

Details

The report needs the Quarto command-line interface 1.9.18 or newer, and the quarto, knitr and rmarkdown packages. RStudio and Positron bundle Quarto; elsewhere, install it from <https://quarto.org/docs/get-started/>. The PDF goes through Typst, which Quarto bundles, so no LaTeX is needed.

The PDF is tagged and declares PDF/UA-1. When Typst compiles the PDF, it checks the PDF/UA-1 rules that it can enforce. When Quarto shows the warning "verapdf is not installed", you can ignore it: it only means that Quarto did not do the extra validation step.

Value

The path of the report, invisibly.

Examples

```
if (interactive()) {
  data(GSE74821)
  render(GSE74821, output_dir = tempdir())
  render(GSE74821, format = "typst", output_dir = tempdir())
}
```

suggest_ruv_k

Suggest how many unwanted factors RUVg should remove

Description

For each `k` from 0 to `max_k`, corrects the log2 positive-normalised counts with RUVg, using the housekeeping genes as controls, and reports the mean interquartile range of the relative log expression (RLE) and the share of variance on the first principal component. The suggested `k` is the smallest whose RLE spread is within 5 % of the smallest one: removing more factors risks removing biology.

Usage

```
suggest_ruv_k(x, max_k = 5)
```

Arguments

x A nacho object from `load_rcc()` or `normalise()`.

max_k The largest k to try. It is lowered to one less than the number of housekeeping genes, and to two less than the number of samples, when those are smaller. A k above that cap fits the samples exactly and flattens the RLE spread, so it is never tried. With one or two samples only k = 0 is tried.

Value

A data frame with k, rle_iqr, pc1_variance and suggested. pc1_variance is NA when the corrected values have no variance, as with one sample.

Examples

```
data(GSE74821)
suggest_ruv_k(GSE74821)
```

upgrade_nacho

Convert a NACHO 2 object to NACHO 3

Description

NACHO 3 stores data in an S7 nacho object instead of the NACHO 2 list. `upgrade_nacho()` converts a NACHO 2 object once: it keeps the raw counts, the sample sheet, the settings and the thresholds, and computes the quality-control metrics, the normalised counts, the PCA and the outlier flags again with the NACHO 3 definitions.

Usage

```
upgrade_nacho(x)
```

Arguments

x A NACHO 2 object, as returned by `load_rcc()` or `normalise()` in NACHO 2.

Value

A nacho object.

See Also

[read_nacho\(\)](#)

Examples

```
# A NACHO 2 object is converted; GSE74821 is already a NACHO 3 object,
# so it comes back unchanged.
data(GSE74821)
identical(upgrade_nacho(GSE74821), GSE74821)
```

visualise

Explore and tune quality control in the NACHO app

Description

Opens `nacho_app()` on `nacho_object`. Change thresholds and normalisation settings in the sidebar, then click "Done" to close the app and get the tuned object back.

Usage

```
visualise(nacho_object)
```

Arguments

`nacho_object` A nacho object from `load_rcc()` or `normalise()`.

Value

The tuned nacho object, invisibly, once you click "Done".

Examples

```
if (interactive()) {
  data(GSE74821)
  tuned <- visualise(GSE74821)
  nacho_qc(tuned)
}

if (interactive()) {
  library(GEOquery)
  library(NACHO)

  # Import data from GEO
  gse <- GEOquery::getGEO(GEO = "GSE74821")
  targets <- Biobase::pData(Biobase::phenoData(gse[[1]]))
  GEOquery::getGEOSuppFiles(GEO = "GSE74821", baseDir = tempdir())
  utils::untar(
    tarfile = file.path(tempdir(), "GSE74821", "GSE74821_RAW.tar"),
    exdir = file.path(tempdir(), "GSE74821")
  )
  targets$IDFILE <- list.files(
    path = file.path(tempdir(), "GSE74821"),
    pattern = ".RCC.gz$"
```

```

)
targets[] <- lapply(X = targets, FUN = iconv, from = "latin1", to = "ASCII")
utils::write.csv(
  x = targets,
  file = file.path(tempdir(), "GSE74821", "Samplesheet.csv")
)

# Read RCC files and format
nacho <- load_rcc(
  data_directory = file.path(tempdir(), "GSE74821"),
  ssheet_csv = file.path(tempdir(), "GSE74821", "Samplesheet.csv"),
  id_colname = "IDFILE"
)
visualise(nacho)

# Drop the outliers and normalise the other samples again
visualise(exclude_outliers(nacho))

# Normalise with the "GLM" method, then drop the outliers
nacho_glm <- normalise(nacho, normalisation_method = "GLM")
visualise(exclude_outliers(nacho_glm))
}

```

write_thresholds

Write and read quality-control thresholds

Description

write_thresholds() saves thresholds to a YAML file, and read_thresholds() reads them back. Use the file to apply the same quality-control policy to each study, or to load thresholds that you tuned in [nacho_app\(\)](#). The Export page of the app writes the same file.

Usage

```
write_thresholds(thresholds, file)
```

```
read_thresholds(file)
```

Arguments

thresholds	A list of thresholds, as returned by nacho_thresholds() or found in the @thresholds property of a nacho object.
file	The path of the YAML file.

Details

In the file, `.inf` and `-.inf` are open bounds, and `.na.character` is an unknown instrument. You can edit the file by hand. `read_thresholds()` reads whole numbers such as 80 as numbers, and checks the thresholds as `normalise()` does. If the file does not give a miRNA threshold (Ligation_order, Ligation_R2, Ligation_NEG or Haemolysis), `read_thresholds()` uses the value of the preset in the file.

Value

`write_thresholds()` returns file, invisibly. `read_thresholds()` returns the list of thresholds.

Examples

```
file <- tempfile(fileext = ".yaml")
thresholds <- nacho_thresholds("sprint")
thresholds$FoV <- 80
write_thresholds(thresholds, file)
read_thresholds(file)
unlink(file)
```

Index

* datasets

GSE74821, 9

as_nacho (as_summarized_experiment), 3

as_summarized_experiment, 3

autoplot(), 20

autoplot.nacho, 4

batch_diagnostics, 5

batch_diagnostics(), 5, 21

character, 7, 11, 12, 17, 18

data.frame, 11

deploy, 6

deploy(), 15

exclude_outliers, 7

exclude_outliers(), 17, 18

filter_detected, 8

GSE74821, 9

housekeeping_stability, 9

housekeeping_stability(), 5

load_rcc, 10

load_rcc(), 3, 4, 6–9, 13, 15–17, 20, 22, 23

logical, 7, 11, 18

nacho-accessors, 13

nacho-methods, 14

nacho_app, 15

nacho_app(), 7, 23, 24

nacho_counts (nacho-accessors), 13

nacho_counts(), 12

nacho_probes (nacho-accessors), 13

nacho_probes(), 3, 12

nacho_qc (nacho-accessors), 13

nacho_qc(), 7, 12, 18

nacho_samples (nacho-accessors), 13

nacho_samples(), 3, 11, 12, 18

nacho_thresholds, 16

nacho_thresholds(), 12, 18, 24

normalise, 17

normalise(), 3, 4, 6–9, 11, 13, 15–18, 20, 22, 23, 25

normalize (normalise), 17

numeric, 12, 18

read_nacho, 19

read_nacho(), 22

read_thresholds (write_thresholds), 24

readRDS(), 19

render, 20

render(), 4

saveRDS(), 19

shiny::runApp(), 15

suggest_ruv_k, 21

suggest_ruv_k(), 12, 18

upgrade_nacho, 22

upgrade_nacho(), 3, 19, 20

visualise, 23

visualise(), 4, 15

visualize (visualise), 23

write_thresholds, 24