Package 'diffHTS'

July 28, 2026

Type Package

Title Differential Drug Sensitivity Analysis for Two-Condition High-Throughput Screens

Version 0.1.0

Description A complete workflow for large-scale, two-condition high-throughput drug screening (HTS). It compares drug sensitivity between any two experimental conditions - for example irradiated versus non-irradiated cells, cancer versus normal cell lines, or treated versus untreated samples - across many plates and experiments. The package covers the full pipeline: control-based normalisation, plate-level quality-control metrics (Z-factor, Z-prime, signal-to-background, signal-to-noise and strictly standardised mean difference), replicate-consistency checks, four-parameter logistic dose-response fitting with area under the curve (AUC) estimation, differential (delta) AUC scoring with within-plate standardisation, cut-off and sigma-based hit selection, and publication-ready heatmap, scatter and quality-control visualisations.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports tidyr, rlang, ggplot2, graphics, stats, utils

Suggests ComplexHeatmap, circlize, ggprism, ggrepel, grid, readxl, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Liang Tang [aut, cre]

Maintainer Liang Tang <xphdtangliang@gmail.com>

Repository CRAN

Date/Publication 2026-07-28 16:10:08 UTC

Contents

annotate_hit_info *Annotate hits with compound metadata*

Description

Joins external compound metadata (name, target, mechanism, library source) onto a hit table.

Usage

```
annotate_hit_info(hits, meta, by = "compound_id")
```

Arguments

hits	A hit data frame (for example from rank_hit_compound()).
meta	A metadata data frame such as hts_compound_meta.
by	Join key present in both. Default "compound_id".

Value

hits with the metadata columns merged in (left join), row order preserved.

See Also

rank_hit_compound(), export_hit_table()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
params <- merge(extract_drc_params(drc), drc$meta[, c("curve_id", "auc")])
annotate_hit_info(rank_hit_compound(params), hts_compound_meta)
```

apply_plate_layout *Stamp a plate-map layout onto measured readings*

Description

Joins a user-defined layout from read_plate_layout() onto a table of raw signal readings, matching by well address (and plate identifier when both sides carry one), and returns a standard hts_raw object. This keeps the *what was measured* (signal) and the *what each well is* (role, compound, concentration) in separate files, exactly as a screening lab records them, and lets the same signal export be re-interpreted against different layouts.

Usage

```
apply_plate_layout(
  readings,
  layout,
  well_col = "well",
  signal_col = "signal",
  plate_col = "plate_id",
  plate_id = NULL
)
```

Arguments

readings	A data frame (or hts_raw) of readings with at least a well column and a signal column.
layout	An hts_layout from read_plate_layout(), or a data frame with well, well_type, compound_id and (optionally) concentration.
well_col, signal_col	Names of the well and signal columns in readings.
plate_col	Name of the plate column. When present in both readings and a multi-plate layout, wells are matched within each plate.
plate_id	Optional plate identifier used when readings has no plate column.

Value

An hts_raw object with columns plate_id, well, row, col, well_type, compound_id, concentration and signal, ready for the rest of the pipeline.

See Also

read_plate_layout(), read_hts_plate()

Examples

```
f <- system.file("extdata", "plate_layout_example.csv", package = "diffHTS")
layout <- read_plate_layout(f, plate_id = "P01")
# Simulate a signal export for the same wells.
readings <- data.frame(well = layout$well,
                       signal = runif(nrow(layout), 1e4, 1e5))
raw <- apply_plate_layout(readings, layout, plate_id = "P01")
table(raw$well_type)
```

`baseline_subtract` *Subtract the blank background signal from a plate*

Description

Removes the per-plate instrument background by subtracting the mean blank (no-cell) signal from every well, a standard first step for luminescence and fluorescence readouts. The original signal is preserved in `signal_raw`.

Usage

```
baseline_subtract(hts_raw, blank_label = "Blank", clip_zero = TRUE)
```

Arguments

`hts_raw`	An `hts_raw` object from `read_hts_plate()`.
`blank_label`	Well-type label identifying blank wells. Default `"Blank"`.
`clip_zero`	Logical; if `TRUE` (default) background-subtracted signals are floored at zero.

Value

The input object with `signal` replaced by the background-subtracted value and a new `signal_raw` column holding the original signal.

See Also

`read_hts_plate()`, `norm_by_control()`

Examples

```
raw <- read_hts_plate(hts_primary_raw)
bs <- baseline_subtract(raw)
head(bs[, c("well_type", "signal_raw", "signal")])
```

`build_auc_matrix` *Build a compound-by-sample AUC matrix*

Description

Reshapes long AUC results into a numeric matrix with compounds as rows and samples (for example cell lines or batches) as columns, optionally standardised to Z-scores for comparability.

Usage

```
build_auc_matrix(
  data,
  row_col = "compound_id",
  col_col = "cell_line",
  value_col = "auc",
  zscore = c("none", "row", "column")
)
```

Arguments

data	A data frame with compound, sample and AUC columns (typically drc$meta after calc_drc_auc()).
row_col	Compound column. Default "compound_id".
col_col	Sample column. Default "cell_line".
value_col	AUC column. Default "auc".
zscore	Standardisation: "none" (default), "row" or "column".

Value

A numeric matrix (compounds x samples). The standardisation used is stored in attr(x, "zscore").

See Also

cluster_auc_matrix(), plot_auc_heatmap()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
build_auc_matrix(drc$meta)
```

---

calculate_qc_metrics *Plate-level quality-control metrics for high-throughput screens*

Description

Computes the standard set of assay quality metrics for a single plate from per-control-type summary statistics: the Z-factor, Z-prime (Z'), the signal-to-background ratio, the signal-to-noise ratio and the strictly standardised mean difference (SSMD).

Usage

```
calculate_qc_metrics(df)
```

Arguments

df	A data frame describing one plate with one row per control type. It must contain the columns experiment_type, mean and sd. The experiment_type column is expected to include the levels "negative_ctrl", "positive_cttl" and "sample" (an optional "edgewell" level is ignored).

Details

The metrics follow the usual HTS definitions, using the negative control and positive (kill) control populations:

$$Z\text{-}factor = 1 - \frac{3(sd_{neg} + sd_{sample})}{|m_{neg} - m_{sample}|}$$

$$Z' = 1 - \frac{3(sd_{neg} + sd_{pos})}{|m_{neg} - m_{pos}|}$$

with signal-to-background m_{neg}/m_{pos}, signal-to-noise $(m_{pos} - m_{neg})/sd_{neg}$ and $SSMD = (m_{pos} - m_{neg})/\sqrt{sd_{pos}^2 + sd_{neg}^2}$.

Value

A one-row data frame with columns z_factor, z_prime, sb_value, sn_value and ssmd_value.

See Also

flag_qc_plates()

Examples

```
plate <- data.frame(
  experiment_type = c("negative_ctrl", "positive_cttl", "sample"),
  mean = c(1.00, 0.10, 0.60),
  sd = c(0.05, 0.03, 0.20)
)
calculate_qc_metrics(plate)
```

---

calc_cv *Control coefficient of variation per plate*

Description

The percent CV (100 * SD / mean) of each control population. **What it is for:** CV% is the direct measure of pipetting and reagent-handling reproducibility within a plate. Low, stable control CVs are a precondition for every other metric; a control drifting above ~15–20% flags a liquid- handling or edge-effect problem even when the Z'-factor still passes.

Usage

```
calc_cv(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  cv_max = 20
)
```

Arguments

hts_raw	An hts_raw object.
signal_col	Name of the signal column. Default "signal".
nc_label, pc_label	Well-type labels for the negative and positive controls. Defaults "NC" and "PC".
cv_max	Acceptance threshold in percent, applied to both controls. Default 20.

Value

A data frame with one row per plate: plate_id, cv_nc, cv_pc (percent) and a logical pass (both controls at or below cv_max).

See Also

calc_plate_qc(), plot_plate_qc()

Examples

```
calc_cv(read_hts_plate(hts_primary_raw))
```

---

calc_drc_auc *Area under the dose-response curve*

Description

Computes the area under each fitted curve by trapezoidal integration of the fitted 4PL over the tested concentration range (on the log10 scale), a robust single-number summary of overall potency.

Usage

```
calc_drc_auc(drc)
```

Arguments

drc	An hts_drc object from fit_4pl_curve().

Value

The hts_drc object with an auc column added to its meta, also returned invisibly as a data frame via drc$meta.

See Also

compute_auc(), extract_drc_params()

Examples

```
drc <- fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line")
calc_drc_auc(drc)$meta
```

---

calc_plate_qc *Comprehensive per-plate quality assessment*

Description

Computes the full panel of assay-quality metrics used to judge a high-throughput screening plate, in a single per-plate table. Alongside the classic Z'-factor this reports the median/MAD-based **robust Z'-factor**, the sample-aware **Z-factor**, **SSMD**, the **signal-to-background (S/B)** and **signal-to-noise (S/N)** ratios, and the **coefficient of variation (CV%)** of each control population. All of these can then be visualised through the single entry point plot_plate_qc().

Usage

```
calc_plate_qc(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  sample_label = "compound",
  z_prime_min = 0.5
)
```

Arguments

hts_raw	An hts_raw object.
signal_col	Name of the signal column. Default "signal".
nc_label, pc_label	Well-type labels for the negative and positive controls. Defaults "NC" and "PC".
sample_label	Well-type label for the test (compound) wells, used by the sample-aware Z-factor. Default "compound".
z_prime_min	Acceptance threshold applied to the Z'-factor to set the pass flag. Default 0.5.

Details

Each metric also has its own dedicated function, useful when you only need one number (and each with its own acceptance threshold): `calc_z_prime()`, `calc_robust_z_prime()`, `calc_z_factor()`, `calc_ssmd()`, `calc_sb_ratio()`, `calc_sn_ratio()` and `calc_cv()`. `calc_plate_qc()` uses the very same formulas so the numbers agree.

What each metric tells you, with the high-signal control taken as whichever of NC/PC has the larger mean and the low-signal control as the other:

- **CV%** $= 100\,\sigma/\mu$ of each control — pipetting/reagent reproducibility; lower is better.
- **S/B** $= \mu_{hi}/\mu_{lo}$ — raw dynamic range of the assay.
- **S/N** $= (\mu_{hi} - \mu_{lo})/\sigma_{lo}$ — assay window relative to background noise.
- **Z'-factor** $= 1 - 3(\sigma_{nc} + \sigma_{pc})/|\mu_{nc} - \mu_{pc}|$ — overall control-only assay quality (`>= 0.5` excellent).
- **Robust Z'-factor**: as Z'-factor but with medians and MADs, resistant to outlier control wells.
- **Z-factor** $= 1 - 3(\sigma_s + \sigma_{nc})/|\mu_s - \mu_{nc}|$ using the sample population s — is the window large versus the library spread (often low/negative in primary screens).
- **SSMD** $= (\mu_{nc} - \mu_{pc})/\sqrt{\sigma_{nc}^2 + \sigma_{pc}^2}$ — effect-size measure of control separation for hit cut-offs.

Value

A data frame with one row per plate and the columns `plate_id`, `mean_nc`, `mean_pc`, `sd_nc`, `sd_pc`, `cv_nc`, `cv_pc`, `signal_window`, `sb` (signal-to-background), `sn` (signal-to-noise), `z_prime`, `robust_z_prime`, `z_factor`, `ssmd` and a logical `pass`.

See Also

`plot_plate_qc()`, `calc_z_prime()`, `calc_well_zscore()`

Examples

```
raw <- read_hts_plate(hts_primary_raw)
calc_plate_qc(raw)
```

---

calc_replicate_correlation

Pairwise replicate correlation between plates

Description

Computes the Pearson correlation (and its R^2) between the per-compound activity of every pair of replicate plates, the standard check that technical replicates agree.

Usage

```
calc_replicate_correlation(
  data,
  value = "viability",
  compound_col = "compound_id",
  plate_col = "plate_id"
)
```

Arguments

`data`	An `hts_norm` object or long data frame.
`value`	Name of the value column. Default `"viability"`.
`compound_col`	Name of the compound column. Default `"compound_id"`.
`plate_col`	Name of the plate column. Default `"plate_id"`.

Value

A data frame with one row per plate pair: `plate_x`, `plate_y`, `n`, `pearson_r` and `r_squared`. The wide compound-by-plate matrix is attached as `attr(x, "wide")` for use by `plot_replicate_scatter()`.

See Also

`calc_replicate_cv()`, `plot_replicate_scatter()`

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
calc_replicate_correlation(norm)
```

`calc_replicate_cv` *Replicate coefficient of variation per compound*

Description

Computes, for each compound, the coefficient of variation (CV, percent) across its technical replicates (repeated wells, whether on the same plate or across plates), a standard measure of assay reproducibility.

Usage

```
calc_replicate_cv(
  data,
  value = "viability",
  compound_col = "compound_id",
  group_cols = NULL
)
```

Arguments

data	An hts_norm object or a long data frame of well-level values.
value	Name of the value column on which to compute the CV. Default "viability" (more stable than inhibition, which can sit near zero).
compound_col	Name of the compound column. Default "compound_id".
group_cols	Optional additional grouping columns (for example "cell_line") that define what counts as one compound series.

Value

A data frame with one row per compound (and any group_cols): compound_id, the grouping columns, n, mean, sd and cv (percent).

See Also

calc_replicate_correlation(), filter_bad_replicate()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
cv <- calc_replicate_cv(norm)
head(cv[order(-cv$cv), ])
```

---

calc_robust_z_prime *Robust Z'-factor per plate*

Description

Median/MAD version of the Z'-factor. **What it is for:** it answers the same question as calc_z_prime() — how cleanly the positive and negative controls separate — but because it uses the median and the median absolute deviation instead of the mean and SD, a couple of splashed or crystallised control wells cannot single-handedly sink an otherwise good plate. Prefer it when control replicates occasionally contain outliers.

Usage

```
calc_robust_z_prime(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  z_prime_min = 0.5
)
```

Arguments

hts_raw	An hts_raw object.
signal_col	Name of the signal column. Default "signal".
nc_label, pc_label	Well-type labels for the negative and positive controls. Defaults "NC" and "PC".
z_prime_min	Acceptance threshold (>= 0.5 is the usual cut-off for an excellent assay). Default 0.5.

Value

A data frame with one row per plate: plate_id, median_nc, median_pc, mad_nc, mad_pc, robust_z_prime and a logical pass.

See Also

calc_z_prime(), calc_plate_qc(), plot_plate_qc()

Examples

```
calc_robust_z_prime(read_hts_plate(hts_primary_raw))
```

---

calc_sb_ratio *Signal-to-background ratio per plate*

Description

The ratio of the high-signal control mean to the low-signal control mean. **What it is for:** S/B is the quickest sanity check that the assay actually produced a usable dynamic range — that the untreated (high) and killed (low) controls are far enough apart on the raw scale. It ignores variability, so it complements but does not replace the Z'-factor; a common minimum is >= 2.

Usage

```
calc_sb_ratio(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  sb_min = 2
)
```

Arguments

hts_raw	An hts_raw object.
signal_col	Name of the signal column. Default "signal".
nc_label, pc_label	Well-type labels for the negative and positive controls. Defaults "NC" and "PC".
sb_min	Acceptance threshold. Default 2.

Value

A data frame with one row per plate: plate_id, mean_nc, mean_pc, sb (high-mean / low-mean) and a logical pass.

See Also

calc_sn_ratio(), calc_plate_qc()

Examples

```
calc_sb_ratio(read_hts_plate(hts_primary_raw))
```

---

calc_sn_ratio *Signal-to-noise ratio per plate*

Description

The assay window divided by the noise of the low-signal control. **What it is for:** S/N asks whether the separation between the two controls is large compared with the background scatter, i.e. whether a real change would rise above the noise floor. It is more informative than S/B because it accounts for variability; higher is better, with >= 3 a reasonable floor.

Usage

```
calc_sn_ratio(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  sn_min = 3
)
```

Arguments

hts_raw	An hts_raw object.
signal_col	Name of the signal column. Default "signal".
nc_label, pc_label	Well-type labels for the negative and positive controls. Defaults "NC" and "PC".
sn_min	Acceptance threshold. Default 3.

Value

A data frame with one row per plate: `plate_id`, `mean_nc`, `mean_pc`, `sn` (window / low-control SD) and a logical `pass`.

See Also

`calc_sb_ratio()`, `calc_plate_qc()`

Examples

```
calc_sn_ratio(read_hts_plate(hts_primary_raw))
```

---

calc_ssmd *SSMD per plate (strictly standardised mean difference)*

Description

The mean control difference divided by the pooled control spread. **What it is for:** SSMD is the effect-size measure of control separation, and it is the statistically principled companion to the Z'-factor for setting hit cut-offs. Its magnitude grades the assay — roughly `|SSMD| >= 2` is excellent, 1–2 good and `< 1` weak — independent of the number of control replicates.

Usage

```
calc_ssmd(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  ssmd_min = 2
)
```

Arguments

`hts_raw`	An `hts_raw` object.
`signal_col`	Name of the signal column. Default `"signal"`.
`nc_label, pc_label`	Well-type labels for the negative and positive controls. Defaults `"NC"` and `"PC"`.
`ssmd_min`	Acceptance threshold applied to `abs(ssmd)`. Default `2`.

Value

A data frame with one row per plate: `plate_id`, `mean_nc`, `mean_pc`, `sd_nc`, `sd_pc`, `ssmd` and a logical `pass`. The sign of `ssmd` follows `mean_nc - mean_pc`.

See Also

calc_z_prime(), calc_plate_qc()

Examples

```
calc_ssmd(read_hts_plate(hts_primary_raw))
```

`calc_well_zscore` *Per-well standard and robust Z-scores*

Description

Standardises every well's signal within its plate, adding both a classic **Z-score** (mean/SD based) and a **robust Z-score** (median/MAD based, which is the preferred choice in HTS because it is not distorted by the very hits the screen is looking for). Scores are computed against a per-plate reference population, by default the test-compound wells.

Usage

```
calc_well_zscore(
  hts_raw,
  signal_col = "signal",
  reference = c("compound", "negative", "all"),
  nc_label = "NC",
  sample_label = "compound"
)
```

Arguments

`hts_raw`	An `hts_raw` object.
`signal_col`	Name of the signal column. Default `"signal"`.
`reference`	One of `"compound"`, `"negative"` or `"all"`, selecting the per-plate population that defines the centre and spread. Default `"compound"`.
`nc_label, sample_label`	Well-type labels for the negative controls and test wells, used to pick the reference population.

Value

The input object with two added columns: `zscore` (mean/SD based) and `robust_zscore` (median/MAD based, MAD scaled to the normal distribution).

See Also

calc_plate_qc(), plot_plate_qc()

Examples

```
raw <- read_hts_plate(hts_primary_raw)
scored <- calc_well_zscore(raw)
head(scored[, c("plate_id", "well", "zscore", "robust_zscore")])
```

`calc_z_factor` *Z-factor per plate (sample-aware)*

Description

The screening-window quality factor computed from the **test-compound** population against the negative control. **What it is for:** unlike the Z'-factor (controls only), the Z-factor tells you whether the assay window is large enough relative to the spread of the actual library wells. Values above `0.5` indicate an excellent screen and `0`–`0.5` a marginal one; it is routinely low or negative in a single-point primary screen, because most library compounds are inactive and therefore sit at the negative-control level — that is expected, not a failure.

Usage

```
calc_z_factor(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  sample_label = "compound",
  z_factor_min = 0.5
)
```

Arguments

`hts_raw`	An `hts_raw` object.
`signal_col`	Name of the signal column. Default `"signal"`.
`nc_label, sample_label`	Well-type labels for the negative control and the test wells. Defaults `"NC"` and `"compound"`.
`z_factor_min`	Acceptance threshold. Default `0.5`.

Value

A data frame with one row per plate: `plate_id`, `mean_sample`, `mean_nc`, `sd_sample`, `sd_nc`, `z_factor` and a logical `pass`.

See Also

`calc_z_prime()`, `calc_plate_qc()`

Examples

```
calc_z_factor(read_hts_plate(hts_primary_raw))
```

`calc_z_prime` *Compute per-plate Z' factors*

Description

Calculates the Z' (Z-prime) factor for each plate from its negative- and positive-control populations and flags plates against an acceptance threshold.

Usage

```
calc_z_prime(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC",
  z_prime_min = 0.5
)
```

Arguments

`hts_raw`	An `hts_raw` object.
`signal_col`	Name of the signal column. Default `"signal"`.
`nc_label`, `pc_label`	Well-type labels for the negative and positive controls. Defaults `"NC"` and `"PC"`.
`z_prime_min`	Acceptance threshold. Default `0.5`.

Value

A data frame with one row per plate: `plate_id`, `mean_nc`, `mean_pc`, `sd_nc`, `sd_pc`, `z_prime` and a logical `pass`.

See Also

`filter_valid_plates()`, `plot_plate_qc()`

Examples

```
raw <- read_hts_plate(hts_primary_raw)
calc_z_prime(raw)
```

check_control_label *Validate control-well labelling*

Description

Checks that a data frame contains the expected negative-, positive- and blank-control labels in its well-type column, so that normalisation and QC can be carried out. It is a light-weight guard to run right after import.

Usage

```
check_control_label(data, type_col = "well_type", required = c("NC", "PC"))
```

Arguments

data	A data frame of well-level readouts.
type_col	Name of the well-type column, as a string. Default "well_type".
required	Character vector of labels that must be present. Default c("NC", "PC"); add "Blank" if background subtraction is planned.

Value

Invisibly, a list with ok (logical), present and missing (character vectors) and counts (a table of well-type frequencies). A warning is emitted when any required label is missing.

Examples

```
d <- data.frame(well_type = c("NC", "NC", "PC", "compound"))
check_control_label(d)
```

clean_compound_id *Standardise compound identifiers*

Description

Cleans free-text compound identifiers into a consistent form: trims surrounding whitespace, collapses internal whitespace, and replaces illegal or separator characters with underscores so that IDs can be matched and used safely in file names.

Usage

```
clean_compound_id(x, to_upper = TRUE)
```

Arguments

x	A character vector of compound identifiers.
to_upper	Logical; if TRUE (default) the result is upper-cased.

Value

A character vector of cleaned identifiers.

Examples

```
clean_compound_id(c("  cpd 001 ", "Drug/A", "cpd-002!"))
```

---

clean_filename *Sanitise a string for use as a file name*

Description

Removes newlines and characters that are illegal in file names on common operating systems, so that drug names or plate labels can be turned into safe output file names.

Usage

```
clean_filename(filename_str)
```

Arguments

filename_str	A character vector of candidate file names.

Value

A character vector with newlines removed, `/` and `:` replaced by `_` and `-` respectively, and the characters `* ? " < > |` stripped.

Examples

```
clean_filename("Drug A/B: 10\u00b5M?")
```

`cluster_auc_matrix` *Hierarchically cluster an AUC matrix*

Description

Performs hierarchical clustering of the compounds (rows) and samples (columns) of an AUC matrix and cuts the compound tree into groups.

Usage

```
cluster_auc_matrix(mat, k = 3, distance = "euclidean", method = "complete")
```

Arguments

`mat`	A numeric matrix from `build_auc_matrix()`.
`k`	Number of compound clusters to cut. Default `3`.
`distance`	Distance method for `stats::dist()`. Default `"euclidean"`.
`method`	Linkage method for `stats::hclust()`. Default `"complete"`.

Value

A list with `row_hclust`, `col_hclust` (or `NULL` if too few columns), `clusters` (named integer vector of compound cluster ids) and `k`.

See Also

`extract_cluster_hit()`, `plot_cluster_tree()`

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
cl <- cluster_auc_matrix(build_auc_matrix(drc$meta), k = 2)
table(cl$clusters)
```

compute_auc *Area under a four-parameter logistic dose-response curve*

Description

Integrates a fitted four_pl() curve between two concentrations to obtain the area under the curve (AUC), a robust summary of drug potency used for between-condition comparisons.

Usage

```
compute_auc(min_concentration, max_concentration, top, ic50, hill, bottom)
```

Arguments

min_concentration	Lower integration limit (minimum tested concentration).
max_concentration	Upper integration limit (maximum tested concentration).
top, ic50, hill, bottom	Four-parameter logistic parameters, as in four_pl().

Value

A single numeric value: the integrated area under the curve. When integration fails (for example a degenerate fit) NA_real_ is returned with a warning.

See Also

four_pl(), fit_dose_response()

Examples

```
compute_auc(
  min_concentration = 0.01, max_concentration = 10,
  top = 1, ic50 = 1, hill = 1, bottom = 0
)
```

compute_delta_auc *Differential (delta) AUC between two screening conditions*

Description

Contrasts drug potency between two experimental conditions by reshaping a long per-drug/per-condition AUC table to one row per drug and computing the differential AUC together with its within-experiment z-score. This is the core scoring step for two-condition screens (for example irradiated versus non-irradiated, or treated versus control).

Usage

```
compute_delta_auc(
  data,
  drug_col = "drug_name",
  condition_col = "condition",
  auc_col = "auc",
  conditions = NULL
)
```

Arguments

data	A long-format data frame with one row per drug and condition.
drug_col	Name of the drug identifier column, as a string. Default "drug_name".
condition_col	Name of the condition column, as a string. Default "condition". The column must contain exactly the two values given in conditions.
auc_col	Name of the AUC column, as a string. Default "auc".
conditions	Character vector of length two giving the baseline and treatment condition, in the order c(baseline, treatment). The delta is computed as treatment - baseline. Defaults to the two levels found in condition_col.

Details

A negative delta_auc indicates the treatment condition sensitises cells to the drug relative to baseline (lower AUC = more killing). The z-score standardises delta_auc across all drugs within the experiment so that hits can be selected on a common scale (see select_hits_sigma()).

Value

A data frame with one row per drug containing the two per-condition AUC columns (named auc_<condition>), delta_auc and zscore_delta_auc (the z-scored delta_auc across all drugs).

See Also

select_hits_cutoff(), select_hits_sigma()

Examples

```
auc_long <- data.frame(
  drug_name = rep(c("DrugA", "DrugB", "DrugC"), each = 2),
  condition = rep(c("Gy0", "Gy2"), times = 3),
  auc = c(5.0, 3.1, 4.2, 4.0, 6.0, 2.0)
)
compute_delta_auc(auc_long, conditions = c("Gy0", "Gy2"))
```

`convert_conc_log10` *Convert concentrations to a log10 scale*

Description

Transforms a concentration vector to `log10`, the scale on which dose-response curves are usually drawn and fitted. Non-positive values (for example the zero-dose controls) become `NA`.

Usage

```
convert_conc_log10(x, shift_min = FALSE)
```

Arguments

`x`	Numeric vector of concentrations (factors/characters are coerced).
`shift_min`	Logical; if `TRUE` the minimum finite `log10` value is subtracted so that the series starts at zero (matching the pipeline's `log10(conc) - min(log10(conc))` convention). Default `FALSE`.

Value

A numeric vector of `log10` concentrations.

Examples

```
convert_conc_log10(c(0.01, 0.1, 1, 10, 100))
convert_conc_log10(c(0.01, 0.1, 1, 10, 100), shift_min = TRUE)
```

`detect_outlier_wells` *Flag outlier wells by the IQR rule*

Description

Detects anomalous wells (for example contaminated, bubble or crystallisation artefacts) using the interquartile-range rule within groups of comparable wells, and optionally removes them.

Usage

```
detect_outlier_wells(
  hts_raw,
  signal_col = "signal",
  group_cols = c("plate_id", "well_type"),
  k = 1.5,
  filter = FALSE
)
```

Arguments

`hts_raw`	An `hts_raw` object.
`signal_col`	Name of the signal column to test. Default `"signal"`.
`group_cols`	Columns defining comparable well groups within which the IQR is computed. Default `c("plate_id", "well_type")`.
`k`	IQR multiplier for the fences. Default `1.5`.
`filter`	Logical; if `TRUE` outlier wells are dropped instead of only flagged. Default `FALSE`.

Value

The input object with a logical `outlier` column added (or with outlier rows removed when `filter = TRUE`).

See Also

`calc_z_prime()`, `filter_valid_plates()`

Examples

```
raw <- read_hts_plate(hts_primary_raw)
flagged <- detect_outlier_wells(raw)
sum(flagged$outlier)
```

export_hit_table *Export a hit summary table*

Description

Writes a complete hit table (all QC and pharmacology columns) to a CSV file.

Usage

```
export_hit_table(hits, file, ...)
```

Arguments

hits	A hit data frame.
file	Output path.
...	Passed to utils::write.csv().

Value

The file path, invisibly.

See Also

rank_hit_compound(), annotate_hit_info()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
params <- merge(extract_drc_params(drc), drc$meta[, c("curve_id", "auc")])
export_hit_table(rank_hit_compound(params), tempfile(fileext = ".csv"))
```

extract_cluster_hit *Extract the most active compound cluster*

Description

Returns the compounds belonging to the cluster with the most extreme mean activity, that is the lowest mean AUC (strongest potency) by default.

Usage

```
extract_cluster_hit(mat, cluster_result, direction = c("low", "high"))
```

Arguments

mat	The AUC matrix used for clustering.
cluster_result	The list returned by cluster_auc_matrix().
direction	"low" (default) selects the lowest-AUC cluster (most active); "high" selects the highest-AUC cluster.

Value

A character vector of compound ids in the selected cluster.

See Also

cluster_auc_matrix()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
m <- build_auc_matrix(drc$meta)
extract_cluster_hit(m, cluster_auc_matrix(m, k = 2))
```

---

extract_drc_params *Extract dose-response parameters*

Description

Pulls the fitted potency and efficacy parameters from every curve of an hts_drc object.

Usage

```
extract_drc_params(drc)
```

Arguments

drc	An hts_drc object from fit_4pl_curve().

Value

A data frame with one row per curve: keys plus ic50, emax (top), bottom, hill, rsq and pass. Curves that failed to fit yield NA parameters.

See Also

fit_4pl_curve(), calc_drc_auc()

Examples

```
drc <- fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line")
head(extract_drc_params(drc))
```

---

filter_bad_replicate *Drop compounds with poor replicate reproducibility*

Description

Removes compounds whose replicate coefficient of variation exceeds a threshold. When a correlation table is supplied it also checks that the replicate plates are globally concordant, warning if any pair falls below the R^2 threshold.

Usage

```
filter_bad_replicate(cv, cv_max = 15, cor_result = NULL, r2_min = 0.8)
```

Arguments

cv	A data frame from calc_replicate_cv().
cv_max	Maximum acceptable CV (percent). Default 15.
cor_result	Optional data frame from calc_replicate_correlation().
r2_min	Minimum acceptable pairwise R^2. Default 0.8.

Value

The subset of cv for compounds passing the CV threshold, with the removed compound ids stored in attr(x, "removed").

See Also

calc_replicate_cv(), calc_replicate_correlation()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
cv <- calc_replicate_cv(norm)
good <- filter_bad_replicate(cv, cv_max = 20)
attr(good, "removed")
```

filter_low_quality_curve *Drop low-quality dose-response curves*

Description

Removes curves that failed to fit or whose R^2 falls below the threshold, leaving only trustworthy dose-response models.

Usage

```
filter_low_quality_curve(drc, rsq_min = NULL)
```

Arguments

drc	An hts_drc object.
rsq_min	Minimum acceptable R^2. Defaults to the value stored at fit time.

Value

The filtered hts_drc object; removed curve ids are stored in attr(drc, "removed").

See Also

fit_4pl_curve()

Examples

```
drc <- fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line")
clean <- filter_low_quality_curve(drc)
nrow(clean$meta)
```

filter_valid_plates *Keep only plates that pass the Z' threshold*

Description

Removes whole plates whose Z' factor falls below the acceptance threshold, returning a clean data set for downstream normalisation.

Usage

```
filter_valid_plates(
  hts_raw,
  z_prime_min = 0.5,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC"
)
```

Arguments

hts_raw	An hts_raw object.
z_prime_min	Acceptance threshold. Default 0.5.
signal_col, nc_label, pc_label	Passed to calc_z_prime().

Value

The hts_raw object restricted to passing plates, with the Z' summary table stored in attr(x, "zprime") and the dropped plate ids in attr(x, "dropped_plates").

See Also

calc_z_prime()

Examples

```
raw <- read_hts_plate(hts_primary_raw)
clean <- filter_valid_plates(raw)
attr(clean, "dropped_plates")
```

---

fit_4pl_curve *Fit four-parameter logistic curves per compound*

Description

Fits a 4PL dose-response model to every compound (optionally split by grouping columns such as cell line), reusing the package's self-contained fitter, and records the goodness of fit R^2. No external fitting package is required.

Usage

```
fit_4pl_curve(
  data,
  response = "response",
  concentration = "concentration",
  compound_col = "compound_id",
  group_cols = NULL,
  rsq_min = 0.85
)
```

Arguments

data	An hts_dose object from import_dose_response() (or a long data frame with compound_id, concentration, response).
response, concentration, compound_col	Column names. Default "response", "concentration", "compound_id".
group_cols	Optional grouping columns.
rsq_min	Curves with R^2 below this are flagged (not dropped here; use filter_low_quality_curve()). Default 0.85.

Value

An hts_drc object: a list with meta (one row per curve: keys, n, rsq, converged, pass), fits (named list of dsr_4pl models) and data (the input). group_cols is stored as an attribute.

See Also

extract_drc_params(), calc_drc_auc(), plot_single_drc()

Examples

```
dr <- import_dose_response(hts_dose_response, response_col = "viability",
  group_cols = "cell_line")
drc <- fit_4pl_curve(dr, group_cols = "cell_line")
head(drc$meta)
```

---

fit_dose_response *Fit a four-parameter logistic dose-response model*

Description

Fits a 4PL curve for a single drug under one condition by non-linear least squares, entirely in base R (no external fitting package required). Robust starting values are derived from the data and refined with a bounded L-BFGS-B optimisation over several initial IC50 guesses, falling back to a derivative-free Nelder-Mead search if needed. The model uses the same parameterisation as four_pl(), which matches the dr4pl package convention (UpperLimit, IC50, Slope, LowerLimit).

Usage

```
fit_dose_response(data, response, concentration, bounds = NULL)

## S3 method for class 'dsr_4pl'
coef(object, ...)

## S3 method for class 'dsr_4pl'
predict(object, newdata = NULL, ...)

## S3 method for class 'dsr_4pl'
fitted(object, ...)

## S3 method for class 'dsr_4pl'
residuals(object, ...)

## S3 method for class 'dsr_4pl'
print(x, ...)
```

Arguments

data	A data frame containing at least the response and concentration columns for one drug and one condition.
response	Name of the (normalised) response column, as a string.
concentration	Name of the concentration column, as a string.
bounds	Optional named list overriding the box constraints used by the bounded optimiser. Recognised names are top, bottom, ic50 and hill, each a numeric length-two vector c(lower, upper).
object	A dsr_4pl object.
...	Ignored, for S3 method consistency.
newdata	Optional numeric vector of concentrations at which to predict. Defaults to the concentrations used for fitting.
x	A dsr_4pl object.

Details

The fitted parameters can be passed directly to compute_auc() to obtain a potency summary, for example do.call(compute_auc, c(list(min(x), max(x)), as.list(coef(fit)))).

Value

An object of class dsr_4pl: a list with elements coefficients (named numeric vector top, ic50, hill, bottom), fitted, residuals, convergence (logical), sse, data and call. The companion methods coef(), predict(), fitted(), residuals() and print() are provided for this class.

Functions

- `coef(dsr_4pl)`: Extract the fitted 4PL coefficients (`top`, `ic50`, `hill`, `bottom`).
- `predict(dsr_4pl)`: Predict responses from a fitted model.
- `fitted(dsr_4pl)`: Extract fitted values.
- `residuals(dsr_4pl)`: Extract residuals.
- `print(dsr_4pl)`: Print a short summary of the fitted model.

See Also

`four_pl()`, `compute_auc()`

Examples

```
one <- subset(
  screen_doseresponse,
  drug_name == "DrugA" & condition == "Gy0" & experiment_type == "sample"
)
fit <- fit_dose_response(one, "normalized_cell_count", "concentration")
coef(fit)
compute_auc(
  min(one$concentration), max(one$concentration),
  top = coef(fit)[["top"]], ic50 = coef(fit)[["ic50"]],
  hill = coef(fit)[["hill"]], bottom = coef(fit)[["bottom"]]
)
```

`flag_qc_plates` *Flag plates that fail quality-control thresholds*

Description

Adds pass/fail flags to a table of plate quality metrics such as the one produced by `calculate_qc_metrics()`, using conventional acceptance thresholds.

Usage

```
flag_qc_plates(qc, z_prime_min = 0.4, sb_min = 3, cv_max = 10)
```

Arguments

`qc`	A data frame of plate quality metrics. Columns referenced by the active thresholds must be present (`z_prime`, `sb_value` and, if supplied, `cv_negative_ctrl`).
`z_prime_min`	Minimum acceptable Z-prime. Plates below this are flagged `"Bad"`. Default `0.4`.
`sb_min`	Minimum acceptable signal-to-background ratio. Default `3`.
`cv_max`	Maximum acceptable coefficient of variation (percent) of the negative control. Only applied when a `cv_negative_ctrl` column is present. Default `10`.

Value

The input data frame with added character flag columns `z_primer_flag`, `sb_flag` and (when applicable) `cv_flag`, each taking the value `"Bad"` or `"Normal"`.

See Also

`calculate_qc_metrics()`

Examples

```
qc <- data.frame(
  plateID = c("P01", "P02"),
  z_prime = c(0.62, 0.20),
  sb_value = c(5.1, 2.4),
  cv_negative_ctrl = c(6, 14)
)
flag_qc_plates(qc)
```

---

four_pl *Four-parameter logistic (4PL) dose-response function*

Description

Evaluates the classic four-parameter logistic model used to describe drug dose-response curves in high-throughput screens.

Usage

```
four_pl(x, top, ic50, hill, bottom)
```

Arguments

`x`	Numeric vector of drug concentrations (on the same scale used to fit the model, typically the raw concentration).
`top`	Response of the upper asymptote (e.g. viability with no drug).
`ic50`	Concentration producing the half-maximal response.
`hill`	Hill slope controlling the steepness of the curve.
`bottom`	Response of the lower asymptote (e.g. viability at saturating drug).

Details

The model is

$$f(x) = \text{bottom} + \frac{\text{top} - \text{bottom}}{1 + (x/\text{ic50})^{\text{hill}}}$$

Value

A numeric vector of predicted responses, the same length as `x`.

See Also

`compute_auc()`, `fit_dose_response()`

Examples

```
four_pl(x = c(0.01, 0.1, 1, 10), top = 1, ic50 = 1, hill = 1, bottom = 0)
```

---

generate_hts_report *Generate an HTML screening report*

Description

Renders a standard HTML analysis report from the bundled R Markdown template, embedding the supplied QC, dose-response and heatmap results. Requires the suggested **rmarkdown** package.

Usage

```
generate_hts_report(
  params = list(),
  output_file = tempfile(fileext = ".html"),
  template = NULL,
  quiet = TRUE
)
```

Arguments

`params`	A named list of objects made available to the report template (passed as `params`), for example `list(hits = ranked, drc = drc)`.
`output_file`	Output HTML path. Default a temporary file.
`template`	Path to an `.Rmd` template. Defaults to the package template in `inst/rmd/report_template.Rmd`.
`quiet`	Passed to `rmarkdown::render()`. Default `TRUE`.

Value

The path to the rendered report, invisibly.

See Also

`export_hit_table()`

Examples

```
## Not run:
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
params <- merge(extract_drc_params(drc), drc$meta[, c("curve_id", "auc")])
generate_hts_report(list(hits = rank_hit_compound(params)))

## End(Not run)
```

---

hts_compound_meta *Example compound metadata*

Description

A small, simulated compound annotation table used to demonstrate hit annotation. Keyed by `compound_id` to join onto screening results.

Usage

```
hts_compound_meta
```

Format

A data frame with one row per compound and the following columns:

compound_id Compound identifier such as `"CPD001"` (character).

compound_name Human-readable compound name (character).

target Nominal molecular target (character).

moa Mechanism of action (character).

library_source Source library the compound came from (character).

Details

The data are simulated (see `data-raw/make_datasets.R`) and do not correspond to any real compound.

hts_dose_response *Example gradient dose-response secondary screen*

Description

A small, simulated dose-response confirmation screen: ten compounds tested across a concentration gradient in three cell lines with two replicates each. Used to demonstrate the dose-response fitting, AUC and clustering modules.

Usage

```
hts_dose_response
```

Format

A data frame with one row per measurement and the following columns:

compound_id Compound identifier such as `"CPD001"` (character).

cell_line Cell line, `"A549"`, `"H1299"` or `"MRC5"` (character).

concentration Compound concentration (micromolar) (numeric).

replicate Technical replicate index, `1` or `2` (integer).

viability Percent viability relative to vehicle control (numeric).

Details

The data are simulated (see `data-raw/make_datasets.R`); the cell line `"MRC5"` is made comparatively resistant.

hts_pal *Colourblind-safe qualitative palette for diffHTS*

Description

Returns the package's low-saturation, colourblind-safe qualitative palette (an Okabe-Ito ordering). Use it whenever a plot maps a discrete variable to colour so that figures stay consistent and accessible.

Usage

```
hts_pal(n = NULL)
```

Arguments

`n`	Number of colours to return (recycled if `n` exceeds the eight base hues). If `NULL` (default) the full palette is returned.

Value

A character vector of hex colours.

See Also

`theme_hts()`

Examples

```
hts_pal(3)
```

`hts_plate_384` *Example single 384-well screening plate (realistic layout)*

Description

A single simulated 384-well plate (`"EXP87P01"`) whose layout mirrors a real cell-based high-throughput screen (see `EXP87/`). The two outermost rings of wells (rows A, B, O, P and columns 1, 2, 23, 24) are kept free of compound to avoid edge-evaporation artefacts. They still hold medium but no live cells, so the instrument reads them at the same low, "dead" level as the kill control; they are stored as `"edgewell"` wells (not omitted). The controls sit inset from the true plate edge in the two innermost flanking columns (3 and 22), arranged in a rotationally balanced *diagonal* pattern: column 3 holds the positive (kill) control in its top half (rows C-H) and the negative (DMSO) control in its bottom half (rows I-N), while column 22 mirrors it (negative top, positive bottom). This diagonal split averages out spatial/edge signal gradients. Columns 4-21 across the interior rows C-N hold the test compounds. Used to demonstrate plate-layout heatmaps with control circling on a 384-well plate.

Usage

```
hts_plate_384
```

Format

A data frame with one row per well (all 384) and the following columns:

plate_id Plate identifier, `"EXP87P01"` (character).

well Well address such as `"C03"` (character).

row Plate row letter, `"A"`-`"P"` (character).

col Plate column number, `1`-`24` (integer).

well_type Well role: `"PC"` (positive control), `"NC"` (negative control), `"compound"` or `"edgewell"` (empty buffer) (character).

compound_id Compound identifier for test wells, `NA` for controls and edge wells (character).

concentration Compound concentration (micromolar); `NA` for controls and edge wells (numeric).

signal Raw luminescence read-out (numeric).

Details

The data are simulated (see `data-raw/make_datasets.R`) and do not correspond to any real compound. All 384 wells are stored; the empty edge wells carry only a low background/no-cell reading.

`hts_primary_raw` *Example single-concentration primary screen (raw signals)*

Description

A small, simulated primary high-throughput screen in 96-well plate format with raw readout signals, used to demonstrate the pre-QC, normalisation, replicate and primary-hit modules. The layout follows good HTS practice: the outer ring of wells (row A, row H and columns 1 and 12) is a compound-free evaporation buffer stored as `"Blank"` wells, the two inset columns 2 and 11 carry the controls in a rotationally balanced *diagonal* pattern (positive controls top-left/bottom-right, negative controls top-right/bottom-left), and the 48 compounds fill the interior block (rows B-G, columns 3-10). Plates `"P01"` and `"P02"` are good replicates (Z-prime around 0.8) while `"P03"` is a failed plate.

Usage

```
hts_primary_raw
```

Format

A data frame with one row per well and the following columns:

plate_id Plate identifier, `"P01"`, `"P02"` or `"P03"` (character).

well Well identifier such as `"A1"` (character).

row Plate row letter, `"A"`-`"H"` (character).

col Plate column number, `1`-`12` (integer).

well_type Well role: `"NC"` (negative), `"PC"` (positive), `"Blank"` or `"compound"` (character).

compound_id Compound identifier such as `"CPD001"`, `NA` for control and blank wells (character).

concentration Compound concentration (micromolar), `NA` for controls (numeric).

signal Raw instrument signal (numeric).

Details

The data are simulated (see `data-raw/make_datasets.R`) and do not correspond to any real compound or cell line.

`import_dose_response` *Import gradient dose-response data*

Description

Standardises a secondary (dose-response) screening table into the columns expected by the fitting engine, matching each compound to its concentration gradient and response, ready for `fit_4pl_curve()`.

Usage

```
import_dose_response(
  data,
  compound_col = "compound_id",
  concentration_col = "concentration",
  response_col = "viability",
  group_cols = NULL
)
```

Arguments

`data`	A data frame, or a path to a csv/txt/xlsx file.
`compound_col, concentration_col, response_col`	Names of the compound, concentration and response columns in `data`.
`group_cols`	Optional grouping columns kept alongside each curve (for example `"cell_line"`).

Value

An `hts_dose` object (data frame) with columns `compound_id`, `concentration`, `response`, any `group_cols`, and canonical class.

See Also

`fit_4pl_curve()`

Examples

```
dr <- import_dose_response(hts_dose_response, response_col = "viability",
  group_cols = "cell_line")
head(dr)
```

merge_plate_data *Combine normalised plates into one data set*

Description

Row-binds several normalised plates (or a list of them) into a single `hts_norm` object holding the activity of the whole screen, keeping only the columns shared by all inputs.

Usage

```
merge_plate_data(...)
```

Arguments

`...`	One or more `hts_norm`/data-frame objects, or a single list of them.

Value

A combined `hts_norm` object.

See Also

`norm_by_control()`

Examples

```
n1 <- norm_by_control(read_hts_plate(
  hts_primary_raw[hts_primary_raw$plate_id == "P01", ]))
n2 <- norm_by_control(read_hts_plate(
  hts_primary_raw[hts_primary_raw$plate_id == "P02", ]))
merged <- merge_plate_data(n1, n2)
table(merged$plate_id)
```

norm_by_control *Normalise wells to plate controls*

Description

Converts raw or background-subtracted signals to percent inhibition and percent viability using each plate's own negative (NC) and positive (PC) controls, so that readouts are comparable across plates. Inhibition is $100 \times (m_{NC} - x)/(m_{NC} - m_{PC})$ and viability is its complement relative to the control window.

Usage

```
norm_by_control(
  hts_raw,
  signal_col = "signal",
  nc_label = "NC",
  pc_label = "PC"
)
```

Arguments

`hts_raw`	An `hts_raw` object (typically after `baseline_subtract()`).
`signal_col`	Name of the signal column to normalise. Default `"signal"`.
`nc_label, pc_label`	Well-type labels for the negative and positive controls. Defaults `"NC"` and `"PC"`.

Value

An `hts_norm` object: the input data with added numeric columns `inhibition` (percent) and `viability` (percent). Multiple plates are normalised independently.

See Also

`merge_plate_data()`, `plot_plate_heatmap_inhibition()`

Examples

```
raw <- baseline_subtract(read_hts_plate(hts_primary_raw))
norm <- norm_by_control(raw)
summary(norm$inhibition)
```

`plate_layout` *Standard microplate coordinate map*

Description

Builds the full row/column coordinate table for a standard microplate, ordered by row then column. This is useful for joining plate readouts onto a complete layout so that empty wells are represented explicitly.

Usage

```
plate_layout(format = 96)

plate_layout_384()

plate_layout_1536()
```

Arguments

format	Plate format (number of wells): one of 6, 12, 24, 48, 96, 384 or 1536. Default 96.

Value

A data frame with one row per well and the columns well (for example "A1"), row (row letter), col (column number) and row_index (numeric row position, A = 1).

See Also

plate_layout_384(), plate_layout_1536()

Examples

```
head(plate_layout(96))
nrow(plate_layout(384))
```

---

plot_auc_heatmap *AUC clustering heatmap*

Description

Draws a clustered heatmap of the AUC matrix. Uses **ComplexHeatmap** (with **circlize** for the colour scale) when available for a publication-quality figure with a diverging navy-white-firebrick palette matching plot_delta_auc_heatmap(), otherwise falls back to base stats::heatmap().

Usage

```
plot_auc_heatmap(
  mat,
  title = "AUC heatmap",
  cluster_rows = TRUE,
  cluster_columns = TRUE
)
```

Arguments

mat	A numeric matrix from build_auc_matrix().
title	Plot title. Default "AUC heatmap".
cluster_rows, cluster_columns	Whether to cluster rows/columns. Default TRUE.

Value

Invisibly, the drawn object. Called for its side effect (a plot on the active graphics device).

See Also

build_auc_matrix(), plot_cluster_tree()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
plot_auc_heatmap(build_auc_matrix(drc$meta, zscore = "row"))
```

---

plot_batch_drc_overlay

Overlay multiple dose-response curves

Description

Overlays the fitted curves of several compounds for side-by-side comparison of activity.

Usage

```
plot_batch_drc_overlay(drc, curve_ids = NULL)
```

Arguments

drc	An hts_drc object.
curve_ids	Curve identifiers to overlay. Defaults to all fitted curves.

Value

A ggplot2::ggplot object.

See Also

plot_single_drc()

Examples

```
drc <- fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line")
plot_batch_drc_overlay(drc, drc$meta$curve_id[1:4])
```

plot_cluster_tree *Plot the compound clustering dendrogram*

Description

Draws the hierarchical clustering tree of compounds from an AUC matrix.

Usage

```
plot_cluster_tree(cluster_result, main = "Compound clustering")
```

Arguments

cluster_result	The list returned by cluster_auc_matrix().
main	Plot title. Default "Compound clustering".

Value

Invisibly NULL; called for its side effect (a base-graphics plot).

See Also

cluster_auc_matrix()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
m <- build_auc_matrix(drc$meta)
plot_cluster_tree(cluster_auc_matrix(m, k = 2))
```

plot_condition_scatter *Scatter plot comparing two conditions or cell lines*

Description

Plots a per-drug scatter of a score (for example delta AUC) under two conditions or cell lines, colouring points by quadrant and labelling the most extreme drugs in each quadrant. This is useful for spotting drugs that behave differently between, say, a cancer and a normal cell line.

Usage

```
plot_condition_scatter(data, x, y, drug_col = "drug_name", label_n = 5)
```

Arguments

`data`	A data frame with one row per drug.
`x, y`	Names of the two numeric score columns to place on the x and y axes, as strings.
`drug_col`	Name of the drug identifier column, as a string. Default `"drug_name"`.
`label_n`	Number of most-extreme drugs (by distance from the origin) to label per quadrant. Default 5.

Details

Point labelling uses **ggrepel** when available and falls back to `ggplot2::geom_text()` otherwise. The axes are centred on the origin with a fixed aspect ratio so the four quadrants are equal in size.

Value

A ggplot2::ggplot object.

See Also

`plot_delta_auc_heatmap()`

Examples

```
df <- data.frame(
  drug_name = paste0("Drug", 1:8),
  MRC5 = c(-1, 0.5, -0.2, 1.2, -0.8, 0.3, -1.5, 0.9),
  A549 = c(-1.2, 0.1, 0.7, -0.9, -1.1, 1.0, -0.4, 0.6)
)
plot_condition_scatter(df, x = "MRC5", y = "A549")
```

`plot_control_scatter` *Scatter plot of control readouts across plates*

Description

Draws, for each plate, the individual negative- and positive-control well readouts (for example normalised cell counts) so that plate-to-plate control behaviour and separation can be inspected at a glance. This is the standard QC view used to confirm that negative controls stay high and positive controls stay low across an entire screen.

Usage

```
plot_control_scatter(
  data,
  plate_col = "plateID",
  neg_cols = grep("^neg_ctrl", names(data), value = TRUE),
  pos_cols = grep("^pos_ctrl", names(data), value = TRUE),
  colors = c(`Negative control` = "#0072B2", `Positive control` = "#D55E00"),
  y_title = "Normalized cell count"
)
```

Arguments

data	A data frame with one row per plate and several negative- and positive-control replicate columns (wide format, such as screen_plate_qc).
plate_col	Name of the plate identifier column, as a string. Default "plateID".
neg_cols, pos_cols	Character vectors naming the negative- and positive-control replicate columns. By default columns matching "^neg_ctrl" and "^pos_ctrl" are used.
colors	Named character vector giving the point colours for the two control groups.
y_title	Y-axis title. Default "Normalized cell count".

Value

A ggplot2::ggplot object: control readouts (points, dodged and jittered) per plate, coloured by control type, with the y-axis expanded to include zero.

See Also

plot_qc_boxplot(), plot_qc_circular_heatmap()

Examples

```
plot_control_scatter(screen_plate_qc)
```

---

plot_cv_distribution *Histogram of replicate CV values*

Description

Shows the distribution of per-compound replicate CVs across the library, with the acceptance threshold marked.

Usage

```
plot_cv_distribution(cv, cv_max = 15, bins = 30)
```

Arguments

cv	A data frame from calc_replicate_cv().
cv_max	Threshold to draw as a reference line. Default 15.
bins	Number of histogram bins. Default 30.

Value

A ggplot2::ggplot object.

See Also

calc_replicate_cv(), filter_bad_replicate()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
plot_cv_distribution(calc_replicate_cv(norm))
```

---

plot_delta_auc_heatmap

Heatmap of differential AUC across conditions or cell lines

Description

Draws a diverging heatmap of differential (delta) AUC values with drugs on one axis and conditions/cell lines on the other, using **ComplexHeatmap**. Colours are centred at zero so that sensitising (negative) and antagonising (positive) responses are visually separated.

Usage

```
plot_delta_auc_heatmap(
  data,
  drug_col = "drug_name",
  value_cols = NULL,
  cell_types = NULL,
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  name = "delta AUC"
)
```

Arguments

`data`	A data frame with one row per drug: a drug identifier column and one numeric delta-AUC column per condition or cell line.
`drug_col`	Name of the drug identifier column, as a string. Default `"drug_name"`.
`value_cols`	Character vector of the delta-AUC column names to display. Defaults to every numeric column other than `drug_col`.
`cell_types`	Optional named character vector mapping each column in `value_cols` to a group label (for example `"Cancer"` or `"Normal"`) used to annotate the heatmap.
`cluster_rows, cluster_columns`	Logical, whether to cluster drugs and conditions respectively. Defaults `TRUE`.
`name`	Legend title. Default `"delta AUC"`.

Details

Requires the suggested packages **ComplexHeatmap** and **circlize**.

Value

A ComplexHeatmap::Heatmap object, which is drawn when printed. Rows with missing values are dropped before plotting.

See Also

`plot_condition_scatter()`

Examples

```
if (requireNamespace("ComplexHeatmap", quietly = TRUE) &&
    requireNamespace("circlize", quietly = TRUE)) {
  ht <- plot_delta_auc_heatmap(
    screen_delta_auc,
    value_cols = c("A549", "H1299", "H1975", "MRC5")
  )
}
```

---

plot_dose_response_curves

Plot dose-response curves for two conditions

Description

Fits a four-parameter logistic model to each condition of a single drug and overlays the observed responses with the smooth fitted curves, coloured by condition. This is the key visual for comparing drug sensitivity between two conditions (for example irradiated versus non-irradiated cells), making a potency shift between conditions immediately apparent.

Usage

```
plot_dose_response_curves(
  data,
  drug = NULL,
  response = "normalized_cell_count",
  concentration = "concentration",
  condition = "condition",
  drug_col = "drug_name",
  n_points = 200,
  colors = c("#0072B2", "#D55E00", "#009E73", "#CC79A7"),
  shapes = c(19, 15, 17, 18),
  ref_line = 0.5,
  legend_title = "Condition"
)
```

Arguments

data	A long-format data frame with one row per well/replicate, containing drug, condition, concentration and response columns.
drug	Optional drug name to display. If NULL (default) data is assumed to already contain a single drug.
response	Name of the (normalised) response column, as a string. Default "normalized_cell_count".
concentration	Name of the concentration column, as a string. Default "concentration".
condition	Name of the condition column, as a string. Default "condition".
drug_col	Name of the drug identifier column, as a string. Default "drug_name".
n_points	Number of points used to draw each smooth fitted curve. Default 200.
colors	Character vector of colours, one per condition (recycled from the pipeline palette by default).
shapes	Numeric vector of point shapes, one per condition.
ref_line	Numeric response level at which to draw a horizontal dashed reference line (for example the 50 % effect level). Set to NA to omit.
legend_title	Legend title. Default "Condition".

Details

The styling reproduces the plots used throughout the screening pipeline: a logarithmic concentration axis, distinct colour and point shape per condition, a horizontal 50 % reference line, and the fitted IC50/AUC of each condition reported in the subtitle.

Only concentrations that are finite and strictly positive are used (a log scale cannot show zero or negative doses). A condition with fewer than four usable observations is plotted as points only, with a warning, and no curve is fitted for it. Fitting is performed with fit_dose_response(), so no external fitting package is required. The figure uses the package's modern-academic theme_hts() look with a colourblind-safe condition palette.

Value

A ggplot2::ggplot object with a base-10 logarithmic concentration axis, observed points and fitted 4PL curves coloured by condition.

See Also

`fit_dose_response()`, `four_pl()`, `compute_auc()`

Examples

```
one <- subset(
  screen_doseresponse,
  drug_name == "DrugA" & experiment_type == "sample"
)
plot_dose_response_curves(one, drug = "DrugA")
```

---

plot_hit_bar_count *Hit-count bar chart*

Description

Bar chart of the number of hit versus non-hit compounds from a primary screen.

Usage

```
plot_hit_bar_count(hits)
```

Arguments

hits	A data frame from `select_primary_hit()`.

Value

A ggplot2::ggplot object.

See Also

`summarize_primary_hit()`

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
plot_hit_bar_count(select_primary_hit(norm))
```

plot_hit_stratify_bar *Stratified hit-count bar chart*

Description

Bar chart of the number of Strong, Moderate and Weak hits.

Usage

```
plot_hit_stratify_bar(ranked)
```

Arguments

ranked	A data frame from `rank_hit_compound()` with a `tier` column.

Value

A ggplot2::ggplot object.

See Also

`rank_hit_compound()`

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
params <- merge(extract_drc_params(drc), drc$meta[, c("curve_id", "auc")])
plot_hit_stratify_bar(rank_hit_compound(params))
```

plot_ic50_auc_cor *IC50 versus AUC scatter*

Description

Scatter plot of IC50 against AUC across curves, a compact view for triaging strong, moderate and weak compounds. IC50 marks the potency (the concentration for half-maximal effect) while AUC integrates potency *and* efficacy over the whole tested range; plotting them together shows whether the two agree and flags compounds that are potent but only weakly efficacious (or vice versa). A regression trend line and the Pearson correlation (computed on the log10 IC50, matching the log x-axis) are drawn on the figure by default.

Usage

```
plot_ic50_auc_cor(
  params,
  auc_col = "auc",
  ic50_col = "ic50",
  add_trend = TRUE,
  annotate_cor = TRUE
)
```

Arguments

params	A data frame from extract_drc_params() joined with an auc column (for example merge(extract_drc_params(drc), calc_drc_auc(drc)$meta)), or the meta of an hts_drc after calc_drc_auc() plus IC50.
auc_col, ic50_col	Column names. Default "auc", "ic50".
add_trend	Add a linear regression trend line with a 95% confidence band. Default TRUE.
annotate_cor	Annotate the Pearson correlation coefficient and p-value on the plot. Default TRUE.

Value

A ggplot2::ggplot object.

See Also

extract_drc_params(), calc_drc_auc()

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
params <- merge(extract_drc_params(drc), drc$meta[, c("curve_id", "auc")])
plot_ic50_auc_cor(params)
```

---

plot_inhibition_hist *Histogram of compound inhibition on a plate*

Description

Shows the distribution of percent inhibition across the compound wells of a plate (controls excluded), the standard view for judging overall hit density.

Usage

```
plot_inhibition_hist(
  hts_norm,
  plate = NULL,
  value = "inhibition",
  compound_label = "compound",
  bins = 30
)
```

Arguments

hts_norm	An hts_norm object.
plate	Optional plate identifier; if NULL (default) all compound wells are pooled.
value	Name of the value column. Default "inhibition".
compound_label	Well-type label marking compound wells. Default "compound".
bins	Number of histogram bins. Default 30.

Value

A ggplot2::ggplot object.

See Also

norm_by_control()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
plot_inhibition_hist(norm, plate = "P01")
```

---

plot_plate_heatmap *Plate-layout heatmap of well readouts*

Description

Draws a microplate as a grid of wells (rows lettered top-to-bottom, columns numbered left-to-right) coloured by a per-well readout such as normalised cell count, so that drug-treated wells and their positive/negative controls can be inspected in their physical plate positions. Missing wells are shown as empty cells, making layout errors and edge effects easy to spot. This abstracts the per-plate heatmap.2() plate maps used in the screening pipeline, but is built on **ggplot2** and needs no external heatmap package.

Usage

```
plot_plate_heatmap(
  data,
  fill = "normalized_cell_count",
  well_col = NULL,
  row_col = "plate_rows",
  col_col = "plate_columns",
  label = NULL,
  control_col = NULL,
  nc_values = c("NC", "negative_ctrl"),
  pc_values = c("PC", "positive_ctrl"),
  circle_size = NULL,
  plate = NULL,
  plate_col = "plateID",
  plate_type = NULL,
  title = NULL,
  na_value = "grey90"
)
```

Arguments

data	A long-format data frame with one row per well.
fill	Name of the column mapped to tile colour, as a string. Default "normalized_cell_count". Numeric columns use a continuous viridis scale; character/factor columns use a discrete viridis scale (useful for showing the well-type layout).
well_col	Optional name of a single well-identifier column such as "A1" or "B12". If supplied it is split into a row letter and column number and row_col/col_col are ignored.
row_col, col_col	Names of the row (letter) and column (number) columns used when well_col is NULL. Defaults "plate_rows" and "plate_columns". Any non-digit characters in the column values (for example an "X" prefix) are stripped.
label	Optional name of a column whose values are printed inside each tile (for example a drug name or a QC flag). Default NULL.
control_col	Optional name of a column identifying each well's role (for example "well_type" or "experiment_type"). When supplied, the negative- and positive-control wells are circled on the plot so that they can be told apart from the drug-treated wells at a glance. Default NULL (no wells circled).
nc_values, pc_values	Character vectors of the labels in control_col that mark negative and positive controls. Defaults cover the package's canonical labels ("NC"/"PC") and the common long forms ("negative_ctrl"/"positive_ctrl").
circle_size	Size of the control-marking circles. When NULL (default) it is scaled to the plate so that 96- and 384-well plates are both legible.
plate	Optional plate identifier; if supplied data is filtered to this plate. When NULL and several plates are present the plot is facetted by plate_col.

plate_col	Name of the plate identifier column, as a string. Default "plateID".
plate_type	Plate format, either 96 or 384, controlling the size of the drawn grid. When NULL (default) it is inferred from the observed rows and columns.
title	Optional plot title.
na_value	Colour used for wells with no data. Default "grey90".

Details

Duplicate wells (for example replicates sharing a position) are summarised by their mean for numeric fill and concatenated for categorical fill/label.

Value

A ggplot2::ggplot object: one tile per well, coloured by fill, with row A at the top and column 1 at the top-left, drawn with a fixed aspect ratio. When control_col is supplied the negative- and positive-control wells are outlined with coloured circles.

See Also

plot_qc_boxplot(), plot_control_scatter()

Examples

```
# A single 96-well plate coloured by normalised cell count.
plot_plate_heatmap(subset(screen_plate_layout, plateID == "EXP99_Gy0"))

# Show the well-type layout instead, both plates side by side.
plot_plate_heatmap(screen_plate_layout, fill = "experiment_type")

# Circle the negative and positive controls on a raw 96-well plate.
plot_plate_heatmap(
  subset(hts_primary_raw, plate_id == "P01"),
  fill = "signal", well_col = "well", plate_col = "plate_id",
  control_col = "well_type"
)

# A realistic 384-well plate (the outer 2-well ring is a compound-free buffer
# that reads at the low, no-cell level; controls sit on a diagonal in the two
# inset columns 3 and 22).
plot_plate_heatmap(hts_plate_384, fill = "signal", well_col = "well",
  control_col = "well_type", plate_col = "plate_id", plate_type = 384)
```

plot_plate_heatmap_inhibition

Percent-inhibition plate heatmap

Description

Draws a plate's normalised percent-inhibition values in plate layout. A thin wrapper around `plot_plate_heatmap()` for `hts_norm` objects.

Usage

```
plot_plate_heatmap_inhibition(hts_norm, plate = NULL, value = "inhibition")
```

Arguments

`hts_norm`	An `hts_norm` object from `norm_by_control()`.
`plate`	Plate identifier to show. Defaults to the first plate.
`value`	Name of the value column to colour by. Default `"inhibition"`.

Value

A ggplot2::ggplot object.

See Also

`plot_plate_heatmap()`, `plot_inhibition_hist()`

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
plot_plate_heatmap_inhibition(norm, plate = "P01")
```

plot_plate_heatmap_raw

Raw-signal plate heatmap

Description

Draws a single plate's raw (or background-subtracted) readouts in plate layout, optionally marking flagged outlier wells. A thin wrapper around `plot_plate_heatmap()` for `hts_raw` objects.

Usage

```
plot_plate_heatmap_raw(
  hts_raw,
  plate = NULL,
  signal_col = "signal",
  mark_outliers = TRUE
)
```

Arguments

hts_raw	An hts_raw object.
plate	Plate identifier to show. Defaults to the first plate.
signal_col	Name of the signal column to colour by. Default "signal".
mark_outliers	Logical; if TRUE and an outlier column is present, flagged wells are labelled with "X". Default TRUE.

Value

A ggplot2::ggplot object.

See Also

plot_plate_heatmap(), plot_plate_qc()

Examples

```
raw <- detect_outlier_wells(read_hts_plate(hts_primary_raw))
plot_plate_heatmap_raw(raw, plate = "P01")
```

---

plot_plate_qc *Visualise any plate quality-control metric*

Description

A single entry point for plotting the plate-quality metrics produced by calc_plate_qc() (and the per-well scores from calc_well_zscore()). Choose which assessment to draw through the metric argument; the acceptance threshold, axis label and pass/fail colouring are set automatically for each metric but can be overridden.

Usage

```
plot_plate_qc(qc, metric = "z_prime", threshold = NULL, plate_col = "plate_id")
```

Arguments

qc	A data frame from calc_plate_qc() for the plate-level metrics, or from calc_well_zscore() for the per-well zscore/robust_zscore distributions.
metric	The metric to plot: one of "z_prime", "robust_z_prime", "z_factor", "ssmd", "sb", "sn", "cv_nc", "cv_pc", "cv" (both control CVs side by side), or "zscore"/"robust_zscore" for per-well distributions. Default "z_prime".
threshold	Optional acceptance threshold drawn as a reference line; defaults to the standard cut-off for the chosen metric.
plate_col	Name of the plate-identifier column. Default "plate_id".

Value

A ggplot2::ggplot object.

See Also

calc_plate_qc(), calc_well_zscore(), plot_plate_heatmap_raw()

Examples

```
raw <- read_hts_plate(hts_primary_raw)
qc <- calc_plate_qc(raw)
plot_plate_qc(qc, metric = "z_prime")
plot_plate_qc(qc, metric = "ssmd")
plot_plate_qc(qc, metric = "cv")
plot_plate_qc(calc_well_zscore(raw), metric = "robust_zscore")
```

---

plot_primary_inhibition_rank

Ranked primary-activity curve

Description

Draws the whole library ranked by activity, with the hit cut-off marked, the standard view of how many compounds clear the primary threshold.

Usage

```
plot_primary_inhibition_rank(hits)
```

Arguments

hits	A data frame from select_primary_hit().

Value

A ggplot2::ggplot object.

See Also

select_primary_hit()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
plot_primary_inhibition_rank(select_primary_hit(norm))
```

---

plot_primary_secondary_cor

Primary versus secondary activity correlation

Description

Scatter of primary-screen activity against secondary-screen AUC for the same compounds, the standard check that primary hits reproduce in the dose-response confirmation screen.

Usage

```
plot_primary_secondary_cor(
  primary,
  secondary,
  by = "compound_id",
  primary_col = "activity",
  secondary_col = "auc"
)
```

Arguments

primary	A data frame with a compound column and a primary-activity column.
secondary	A data frame with a compound column and a secondary-activity (AUC) column.
by	Join key. Default "compound_id".
primary_col	Primary-activity column. Default "activity".
secondary_col	Secondary-activity column. Default "auc".

Value

A ggplot2::ggplot object.

See Also

select_primary_hit(), calc_drc_auc()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
primary <- select_primary_hit(norm)
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
sec <- aggregate(auc ~ compound_id, drc$meta, mean)
plot_primary_secondary_cor(primary, sec)
```

`plot_qc_boxplot` *Quality-control boxplot of control wells across plates*

Description

Draws grouped boxplots of a per-well readout across plates, split by control type, to inspect assay consistency and separation between negative and positive controls.

Usage

```
plot_qc_boxplot(
  data,
  plate_col = "plateID",
  value_col = "normalized_cell_count",
  group_col = "experiment_type",
  y_title = "Normalized cell count"
)
```

Arguments

`data`	A long-format data frame of well-level readouts.
`plate_col`	Name of the plate identifier column, as a string. Default `"plateID"`.
`value_col`	Name of the numeric readout column, as a string. Default `"normalized_cell_count"`.
`group_col`	Name of the control-type / grouping column, as a string. Default `"experiment_type"`.
`y_title`	Axis title for the readout. Default `"Normalized cell count"`.

Value

A ggplot2::ggplot object.

See Also

`calculate_qc_metrics()`

Examples

```
plot_qc_boxplot(screen_doseresponse)
```

plot_qc_circular_heatmap *Circular heatmap of control readouts across plates*

Description

Draws a large circular (**circlize**) heatmap of the control-well readouts for every plate, grouped into sectors by experiment. Each ring is one plate and each cell one control well, so systematic control problems across a whole screen become visible at once. This mirrors the summary `circos.heatmap` used in the pipeline to display negative- and positive-control cell counts.

Usage

```
plot_qc_circular_heatmap(
  data,
  value_cols = grep("_ctrl_", names(data), value = TRUE),
  plate_col = "plateID",
  split_col = "experimentID",
  col_breaks = c(0, 0.5, 1),
  col_colors = c("navy", "white", "firebrick3"),
  name = "Norm cell counts",
  show_sector_labels = TRUE
)
```

Arguments

`data`	A data frame with one row per plate and numeric control columns, such as screen_plate_qc.
`value_cols`	Character vector of the control-value columns to display. By default all columns matching `"_ctrl_"` are used.
`plate_col`	Name of the plate identifier column (used as row names), as a string. Default `"plateID"`.
`split_col`	Optional name of a grouping column (experiment) used to split the ring into sectors. Default `"experimentID"`.
`col_breaks, col_colors`	Numeric breakpoints and matching colours passed to `circlize::colorRamp2()`. Defaults map `0/0.5/1` to navy/white/firebrick.
`name`	Legend title. Default `"Norm cell counts"`.
`show_sector_labels`	Logical, whether to label each sector. Default `TRUE`.

Details

Requires the suggested package **circlize**; when **ComplexHeatmap** is also installed a colour legend is added.

Value

Invisibly NULL. Called for the side effect of drawing to the active graphics device.

See Also

plot_control_scatter(), plot_delta_auc_heatmap()

Examples

```
if (requireNamespace("circlize", quietly = TRUE)) {
  plot_qc_circular_heatmap(screen_plate_qc)
}
```

---

plot_qc_metric_circular

Circular plot of a per-plate QC metric

Description

Draws a circular (**circlize**) scatter of a QC metric with one sector per experiment and one point per plate, coloured by whether the plate passes a threshold. This reproduces the whole-screen Z'-factor overview used in the pipeline, where many plates across several experiments are compared on a single ring.

Usage

```
plot_qc_metric_circular(
  data,
  metric = "z_prime",
  sector_col = "experimentID",
  x_col = "plateNumber",
  label_col = "plateID",
  threshold = 0.5,
  transform = TRUE,
  colors = c(Bad = "#D55E00", Normal = "#0072B2"),
  bar_lwd = 4,
  x_margin = 0.8,
  show_labels = TRUE,
  center_label = NULL
)
```

Arguments

data	A data frame with one row per plate, such as screen_plate_qc.
metric	Name of the metric column, as a string. Default "z_prime".
sector_col	Name of the column that defines the sectors (experiments), as a string. Default "experimentID".
x_col	Name of the numeric within-sector position column (plate order), as a string. Default "plateNumber".
label_col	Name of the column holding the plate identifier printed on the outside of each bar, as a string. Default "plateID"; if the column is absent the values of x_col are used instead.
threshold	Numeric pass/fail cut-off. Plates whose metric is below threshold are drawn in bad_color. Default 0.5.
transform	Logical; if TRUE (default) strongly negative metric values (< -1) are compressed with -log10(-x) for display, matching the pipeline's Z'-factor plot.
colors	Named character vector with entries "Bad" and "Normal" giving the bar colours.
bar_lwd	Line width of each plate bar. Default 4.
x_margin	Numeric angular padding (in x_col units) added to each side of every sector so the first and last plate bars do not touch the sector edges. Default 0.8.
show_labels	Logical; if TRUE (default) the plate identifier from label_col is printed radially just outside each bar.
center_label	Text drawn in the centre of the ring. Defaults to metric.

Details

Requires the suggested package **circlize**. Each experiment is a sector and each plate a thin vertical bar drawn at its own plateNumber position, rising from a zero baseline to the metric value, so the bars never overlap. An angular x_margin keeps the first and last bars clear of the sector edges, the plate identifier is printed radially just outside each bar, and a short Y-axis on the first sector plus a dashed reference line at threshold in every sector give the scale. The design scales to large campaigns of roughly ten plates per experiment.

Value

Invisibly NULL. Called for the side effect of drawing to the active graphics device.

See Also

plot_qc_metric_trend(), plot_qc_circular_heatmap()

Examples

```
if (requireNamespace("circlize", quietly = TRUE)) {
  plot_qc_metric_circular(screen_plate_qc, metric = "z_prime")
}
```

`plot_qc_metric_trend` *Trend plot of a QC metric across plates*

Description

Plots a single QC metric (for example Z', Z-factor or negative-control CV) for every plate in acquisition order, colouring each plate by a pass/fail flag and marking the overall mean. This mirrors the per-metric QC review plots used in the screening pipeline and makes drifting or failing plates easy to spot.

Usage

```
plot_qc_metric_trend(
  data,
  metric = "z_prime",
  plate_col = "plateID",
  flag_col = NULL,
  threshold = 0.5,
  direction = c("min", "max"),
  colors = c(Bad = "#D55E00", Normal = "#0072B2")
)
```

Arguments

`data`	A data frame with one row per plate, such as screen_plate_qc.
`metric`	Name of the metric column to plot, as a string. Default `"z_prime"`.
`plate_col`	Name of the plate identifier column, as a string. Default `"plateID"`.
`flag_col`	Optional name of a column of `"Bad"`/`"Normal"` flags used to colour points. If `NULL` (default) plates are flagged on the fly using `threshold` and `direction`.
`threshold`	Numeric cut-off used to flag plates when `flag_col` is `NULL`. Default `0.5`.
`direction`	Either `"min"` (values below `threshold` fail, the default) or `"max"` (values above `threshold` fail).
`colors`	Named character vector giving the colours for `"Bad"` and `"Normal"` plates.

Value

A ggplot2::ggplot object: one bar per plate in plate order, coloured by flag, with a dashed line at the mean and a solid line at `threshold`.

See Also

`calculate_qc_metrics()`, `flag_qc_plates()`

Examples

```
plot_qc_metric_trend(screen_plate_qc, metric = "z_prime")
```

`plot_replicate_scatter` *Replicate scatter plot(s) with regression fit and correlation*

Description

Plots the activity of replicate plates against one another in the house style used across the screening reports: points coloured by well type (compounds, negative/positive controls, blanks) at 50% opacity, a linear regression line with a shaded 95% confidence band, and a Pearson R (with p-value) annotated on each panel. When more than two replicate plates are present and no specific pair is requested, every pairwise combination (replicate 1 vs 2, 1 vs 3, 2 vs 3, ...) is drawn together in a single faceted figure, mirroring the `combn()`-based `reps_corr_plot()` workflow in the source analyses.

Usage

```
plot_replicate_scatter(
  data,
  plate_x = NULL,
  plate_y = NULL,
  value = "viability",
  plate_col = "plate_id",
  key_col = NULL,
  compound_col = "compound_id",
  color_col = NULL,
  palette = .well_type_palette,
  line_color = "lightblue",
  band_color = "lightgray",
  show_identity = TRUE
)
```

Arguments

`data`	An `hts_norm`/long data frame, or a data frame from `calc_replicate_correlation()` (its wide matrix is used).
`plate_x, plate_y`	Plate identifiers for the two axes. When both are `NULL` (default) all pairwise combinations of the available plates are drawn; supply both to draw a single pair.
`value`	Name of the value column when `data` is well-level. Default `"viability"`.
`plate_col`	Name of the plate column. Default `"plate_id"`.
`key_col`	Column identifying matching observations across replicate plates. Defaults to the well-address column `"well"` when present (so that control wells are paired and displayed), otherwise the compound column.
`compound_col`	Compound column, used as the pairing key when no `well` column is available. Default `"compound_id"`.

color_col	Column used to colour points, typically the well-role column. Defaults to "well_type" when present; set to NULL to disable colouring.
palette	Named vector mapping well-type levels to colours. Defaults to the report palette (compounds blue, negative controls black, positive controls red, blanks gold).
line_color, band_color	Colour of the regression line and of its 95% confidence band. Default light blue on light grey, as in the reports.
show_identity	Whether to draw the dotted $y = x$ identity line. Default TRUE.

Value

A single ggplot2::ggplot object. With three or more plates it is faceted, one panel per replicate pair. The annotated R and p-value are computed over every point shown in the panel.

See Also

calc_replicate_correlation()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
# All pairwise replicate comparisons at once, points coloured by well type:
plot_replicate_scatter(norm)
# A single chosen pair:
plot_replicate_scatter(norm, plate_x = "P01", plate_y = "P02")
```

---

plot_sigma_hits *Distribution plot of sigma-based hits*

Description

Histogram of the per-compound activity with the library centre and the 1, 2 and 3 sigma cut-offs marked, bars coloured by how many sigma each compound clears. Optionally overlays the fitted normal curve so the departure from normality (the hit tail) stands out.

Usage

```
plot_sigma_hits(hits, bins = 30, show_normal = TRUE)
```

Arguments

hits	A data frame from select_sigma_hits().
bins	Number of histogram bins. Default 30.
show_normal	Overlay the fitted normal density curve. Default TRUE.

Value

A ggplot2::ggplot object.

See Also

`select_sigma_hits()`, `summarize_sigma_hits()`

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
plot_sigma_hits(select_sigma_hits(norm))
```

---

plot_single_drc *Plot a single dose-response curve*

Description

Plots one compound's observed points and fitted 4PL curve on a log10 concentration axis, annotating IC50, AUC and R^2.

Usage

```
plot_single_drc(drc, curve_id = NULL, color = "#0072B2")
```

Arguments

`drc`	An `hts_drc` object (ideally after `calc_drc_auc()`).
`curve_id`	Curve identifier (from `drc$meta$curve_id`). Defaults to the first curve.
`color`	Curve colour. Default `"#0072B2"`.

Value

A ggplot2::ggplot object.

See Also

`plot_batch_drc_overlay()`

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
plot_single_drc(drc, drc$meta$curve_id[1])
```

`rank_hit_compound` *Rank and stratify hit compounds*

Description

Combines potency, efficacy and quality metrics into a single activity score and assigns each compound to a Strong, Moderate or Weak tier, the final prioritisation step of the pipeline. Lower AUC and IC50 and higher Emax count as more active; lower replicate CV and higher fit R^2 raise confidence.

Usage

```
rank_hit_compound(
  data,
  auc_col = "auc",
  ic50_col = "ic50",
  emax_col = "emax",
  cv_col = "cv",
  rsq_col = "rsq",
  weights = NULL,
  probs = c(1/3, 2/3)
)
```

Arguments

`data`	A per-compound data frame of metrics.
`auc_col`, `ic50_col`, `emax_col`, `cv_col`, `rsq_col`	Column names for the metrics to use; set any to `NULL` to omit it. Metrics absent from `data` are silently skipped.
`weights`	Optional named numeric vector of per-metric weights (names matching the metric column names). Defaults to equal weights.
`probs`	Two increasing probabilities splitting Weak/Moderate/Strong by score quantile. Default `c(1/3, 2/3)`.

Value

`data` with added `score` (numeric) and `tier` (ordered factor Weak < Moderate < Strong), sorted by descending score.

See Also

`annotate_hit_info()`, `plot_hit_stratify_bar()`

Examples

```
drc <- calc_drc_auc(fit_4pl_curve(import_dose_response(hts_dose_response,
  response_col = "viability", group_cols = "cell_line"),
  group_cols = "cell_line"))
params <- merge(extract_drc_params(drc), drc$meta[, c("curve_id", "auc")])
ranked <- rank_hit_compound(params)
table(ranked$tier)
```

`read_hts_plate` *Import a plate readout file into a standard HTS object*

Description

Reads an instrument export (csv, txt/tsv or Excel) or an in-memory data frame of well-level readouts and standardises it into an `hts_raw` object with a fixed set of columns, ready for the rest of the pipeline. Well coordinates are parsed from the well identifier and control wells are canonicalised to the labels `"NC"`, `"PC"` and `"Blank"`.

Usage

```
read_hts_plate(
  file,
  plate_id = NULL,
  well_col = "well",
  signal_col = "signal",
  compound_col = "compound_id",
  conc_col = "concentration",
  type_col = "well_type",
  plate_col = "plate_id",
  nc_label = "NC",
  pc_label = "PC",
  blank_label = "Blank",
  sep = "\t",
  sheet = 1
)
```

Arguments

`file`	A path to a `.csv`, `.txt`/`.tsv` or `.xlsx`/`.xls` file, or an existing data frame of well-level readings.
`plate_id`	Optional plate identifier used when the data has no plate column.
`well_col`, `signal_col`, `compound_col`, `conc_col`, `type_col`, `plate_col`	Names of the well, signal, compound, concentration, well-type and plate columns in `file`. Missing optional columns are filled with `NA`.

`nc_label`, `pc_label`, `blank_label`
: Character vectors of the labels that mark negative controls, positive controls and blanks (matched case-insensitively in `type_col`, or in `compound_col` when `type_col` is absent).

`sep`
: Field separator for `.txt`/`.tsv` files. Default tab.

`sheet`
: Worksheet passed to **readxl** for Excel files. Default `1`.

Details

Reading Excel files requires the suggested package **readxl**.

Value

An `hts_raw` object (a data frame subclass) with columns `plate_id`, `well`, `row`, `col`, `well_type`, `compound_id`, `concentration` and `signal`.

See Also

`baseline_subtract()`, `calc_z_prime()`, `norm_by_control()`

Examples

```
raw <- read_hts_plate(hts_primary_raw)
table(raw$well_type)
```

---

read_plate_layout *Read a user-defined plate-map layout file*

Description

Reads a plate map that **you author yourself** in Excel or CSV to declare, for every well, whether it holds a negative control, a positive control, a blank (empty/edge) well or a test compound. Nothing about the layout is fixed by the package: you are free to place the controls, blanks and compounds in whichever wells your assay uses, and `apply_plate_layout()` then stamps those roles onto the measured readings.

Usage

```
read_plate_layout(
  file,
  plate_id = NULL,
  nc_labels = c("NC", "DMSO", "vehicle", "negative_ctrl", "neg_ctrl"),
  pc_labels = c("PC", "kill_ctrl", "positive_ctrl", "pos_ctrl"),
  blank_labels = c("Blank", "edgewell", "empty", "blank", "na", ""),
  concentration_file = NULL,
  sheet = 1,
  sep = "\t"
)
```

Arguments

file	Path to the plate-map grid (.csv, .txt/.tsv or .xlsx/.xls).
plate_id	Optional plate identifier attached to every well of the layout, so the same map can be matched to a specific plate.
nc_labels, pc_labels, blank_labels	Character vectors of the cell labels that mark negative controls, positive controls and blank/empty wells. All other non-empty labels become compound identifiers.
concentration_file	Optional path to a second grid (same layout) giving the concentration of each well.
sheet, sep	Worksheet (Excel) and field separator (.txt/.tsv) passed to the reader.

Details

The expected file is a **grid** that mirrors the physical plate: the first column holds the row letters (A, B, ...), the remaining column headers are the plate column numbers (1, 2, ... 24), and each cell contains the label for that well. Labels are matched case-insensitively against nc_labels, pc_labels and blank_labels; any other non-empty label is treated as a compound identifier. An optional second grid of the same shape supplies per-well concentrations.

Value

An hts_layout data frame with one row per well and the columns plate_id, well, row, col, well_type ("NC", "PC", "Blank" or "compound"), compound_id and concentration.

See Also

apply_plate_layout(), read_hts_plate(), check_control_label()

Examples

```
f <- system.file("extdata", "plate_layout_example.csv", package = "diffHTS")
layout <- read_plate_layout(f, plate_id = "P01")
table(layout$well_type)
```

---

screen_delta_auc *Example differential-AUC hit table*

Description

A small, simulated table of differential (delta) AUC values with one row per drug and one column per cell line, used to demonstrate hit selection and heatmap/scatter visualisation across several cancer cell lines and a normal (MRC5) cell line.

Usage

```
screen_delta_auc
```

Format

A data frame with one row per drug and the following columns:

drug_name Drug identifier (character).

A549 Delta AUC in the A549 lung cancer cell line (numeric).

H1299 Delta AUC in the H1299 lung cancer cell line (numeric).

H1975 Delta AUC in the H1975 lung cancer cell line (numeric).

H2030 Delta AUC in the H2030 lung cancer cell line (numeric).

MRC5 Delta AUC in the MRC5 normal lung fibroblast cell line (numeric).

Details

The data are simulated (see `data-raw/make_datasets.R`) and do not correspond to any real compound. Negative values indicate treatment-induced sensitisation.

See Also

screen_doseresponse

`screen_doseresponse` *Example two-condition dose-response screen*

Description

A small, simulated high-throughput drug-screening dataset in long (tidy) format, mimicking a two-condition experiment in which each drug is tested across a concentration series under a baseline (`"Gy0"`) and a treatment (`"Gy2"`) condition, alongside negative and positive control wells. It is sized to run the package examples and vignette quickly.

Usage

```
screen_doseresponse
```

Format

A data frame with one row per well and the following columns:

plateID Plate identifier (character).

drug_name Drug identifier, or a control label such as `"DMSO"` (negative) or `"kill_ctrl"` (positive).

experiment_type Well role: `"sample"`, `"negative_ctrl"` or `"positive_cttl"`.

condition Experimental condition, `"Gy0"` (baseline) or `"Gy2"` (treatment).

concentration Drug concentration (micromolar).

normalized_cell_count Viability normalised to the negative control (roughly 0-1).

Details

The data are simulated (see `data-raw/make_datasets.R`) and do not correspond to any real compound or cell line.

See Also

screen_delta_auc

`screen_plate_layout` *Example 96-well plate layout with well positions*

Description

A small, simulated 96-well plate (8 rows by 12 columns) assayed under two conditions (`"Gy0"` and `"Gy2"`), in long (one row per well) format. The layout follows good HTS practice: the outer ring (row A, row H, columns 1 and 12) is an empty evaporation buffer, the two flanking columns (2 and 11) carry the positive (kill) and negative (DMSO) controls in a rotationally balanced diagonal pattern, and the interior block (rows B-G, columns 3-10) holds an 8-point dose series of one drug per row. It is used to demonstrate the plate-layout heatmap `plot_plate_heatmap()`.

Usage

```
screen_plate_layout
```

Format

A data frame with one row per well and the following columns:

plateID Plate identifier such as `"EXP99_Gy0"` (character).

condition Experimental condition, `"Gy0"` or `"Gy2"`.

well Well identifier such as `"A1"` (character).

plate_rows Plate row letter, `"A"`-`"H"` (character).

plate_columns Plate column number, `1`-`12` (integer).

drug_name Drug identifier, or a control label (`"DMSO"`, `"kill_ctrl"`), or `NA` for empty buffer wells.

experiment_type Well role: `"sample"`, `"negative_ctrl"`, `"positive_ctrl"` or `"empty"` (evaporation-buffer ring).

concentration Drug concentration (micromolar), `NA` for controls and empty wells.

normalized_cell_count Viability normalised to the negative control; empty buffer wells hold no live cells and read near the kill control.

Details

The data are simulated (see `data-raw/make_datasets.R`) and do not correspond to any real compound or cell line.

See Also

plot_plate_heatmap()

---

screen_plate_qc *Example per-plate quality-control summary*

Description

A small, simulated per-plate quality-control (QC) table spanning several experiments (cell lines), each with a few plates. Every plate carries its negative- and positive-control replicate readouts together with the QC metrics derived from them, so the table can drive the control scatter, the circular control heatmap and the circular Z'-factor plot. A couple of plates are deliberately noisy so that passing and failing plates are both present.

Usage

```
screen_plate_qc
```

Format

A data frame with one row per plate and the following columns:

plateID Plate identifier such as "EXP76_01" (character).

experimentID Experiment identifier (character), the plate prefix.

cell_line Cell line assayed on the experiment (character).

plateNumber Plate order within the experiment (integer).

z_factor Assay Z-factor from sample and negative-control wells.

z_prime Z-prime factor from negative- and positive-control wells.

sb_value Signal-to-background ratio.

sn_value Signal-to-noise value.

ssmd_value Strictly standardised mean difference.

cv_negative_ctrl Coefficient of variation of the negative controls (percent).

neg_ctrl_1, neg_ctrl_2, neg_ctrl_3, neg_ctrl_4, neg_ctrl_5, neg_ctrl_6 Negative-control well readouts.

pos_ctrl_1, pos_ctrl_2, pos_ctrl_3, pos_ctrl_4, pos_ctrl_5, pos_ctrl_6 Positive-control well readouts.

Details

The data are simulated (see data-raw/make_datasets.R) and do not correspond to any real compound or cell line.

See Also

plot_control_scatter(), plot_qc_metric_circular(), plot_qc_circular_heatmap()

`select_hits_cutoff` *Select hits by an absolute cut-off across conditions*

Description

Nominates drugs whose score is below a fixed cut-off in a required number of the supplied score columns. This implements the cut-off based hit-selection strategy used to intersect differential responses across several cancer cell lines.

Usage

```
select_hits_cutoff(
  data,
  score_cols,
  cutoff = 0,
  min_pass = length(score_cols),
  drug_col = "drug_name"
)
```

Arguments

`data`	A data frame with one row per drug: a drug identifier column plus one numeric score column per condition or cell line (for example `delta_auc` or a z-scored value).
`score_cols`	Character vector of the score column names to test.
`cutoff`	Numeric threshold. A drug passes a column when its value is strictly below `cutoff`. Default `0`.
`min_pass`	Minimum number of `score_cols` that must be below `cutoff` for a drug to be selected. Defaults to all of them.
`drug_col`	Name of the drug identifier column, as a string. Default `"drug_name"`.

Value

The subset of `data` (rows) meeting the criterion, with an added integer column `n_pass` giving the number of score columns below `cutoff`. Rows containing missing scores in the tested columns are dropped.

See Also

`select_hits_sigma()`

Examples

```
df <- data.frame(
  drug_name = c("DrugA", "DrugB", "DrugC"),
  A549 = c(-0.5, 0.2, -0.9),
  H1299 = c(-0.3, -0.1, -0.8)
```

```
)
select_hits_cutoff(df, score_cols = c("A549", "H1299"), cutoff = 0,
                   min_pass = 2)
```

`select_hits_sigma` *Select hits by a sigma (z-score) threshold*

Description

Nominates drugs whose standardised score exceeds a given number of standard deviations from the mean, in the desired direction. This implements the sigma-based (for example 2-sigma or 3-sigma) hit-selection strategy applied to z-scored differential AUC values.

Usage

```
select_hits_sigma(
  data,
  score_col = "zscore_delta_auc",
  n_sigma = 3,
  direction = c("less", "greater"),
  standardize = FALSE,
  drug_col = "drug_name"
)
```

Arguments

`data`	A data frame with one row per drug.
`score_col`	Name of the numeric (typically already z-scored) score column, as a string. Default `"zscore_delta_auc"`.
`n_sigma`	Number of standard deviations defining the threshold. Default `3`.
`direction`	Either `"less"` (default, select strongly negative scores, i.e. sensitising drugs) or `"greater"` (select strongly positive scores).
`standardize`	Logical. If `TRUE`, the score column is z-scored before thresholding; if `FALSE` (default) the column is assumed to already be on a standardised (sigma) scale and is thresholded directly.
`drug_col`	Name of the drug identifier column, as a string. Default `"drug_name"`.

Value

The subset of `data` whose score is beyond the sigma threshold in the requested direction, ordered by score.

See Also

`select_hits_cutoff()`, `compute_delta_auc()`

Examples

```
df <- data.frame(
  drug_name = paste0("Drug", 1:6),
  zscore_delta_auc = c(-3.4, -1.2, 0.1, 0.5, 1.1, -2.8)
)
select_hits_sigma(df, n_sigma = 2, direction = "less")
```

---

select_primary_hit *Select primary-screen hits*

Description

Flags active compounds from single-concentration primary-screen data using either a fixed activity threshold or a percentile rank of the whole library.

Usage

```
select_primary_hit(
  data,
  value = "inhibition",
  method = c("threshold", "percentile"),
  threshold = 50,
  percentile = 0.95,
  compound_col = "compound_id"
)
```

Arguments

data	An hts_norm object or long data frame.
value	Name of the activity column. Default "inhibition".
method	Selection mode: "threshold" (default) keeps compounds with activity at or above threshold; "percentile" keeps the top fraction.
threshold	Activity cut-off for method = "threshold". Default 50.
percentile	Upper fraction to keep for method = "percentile" (for example 0.95 keeps the top 5 percent). Default 0.95.
compound_col	Name of the compound column. Default "compound_id".

Value

A data frame with one row per compound: compound_id, activity, is_hit. The applied cut-off is stored in attr(x, "cutoff") and the method in attr(x, "method").

See Also

summarize_primary_hit(), plot_primary_inhibition_rank()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
hits <- select_primary_hit(norm, threshold = 50)
sum(hits$is_hit)
```

`select_sigma_hits` *Select primary-screen hits by distance from the library distribution*

Description

Treats the per-compound activity of the whole library as an approximately normal "inactive" distribution and flags compounds sitting far out in the tail. Each compound receives a z-score (how many standard deviations it lies from the library centre) and is binned by whether it clears 1, 2 or 3 sigma. By default the centre and spread are estimated *robustly* (median and MAD), the HTS-standard choice: a handful of genuine hits would otherwise inflate a plain mean/SD and mask themselves (with too wide an SD nothing reaches 3 sigma).

Usage

```
select_sigma_hits(
  data,
  value = "inhibition",
  method = c("robust", "sd"),
  n_sigma = 3,
  direction = c("greater", "less", "two.sided"),
  compound_col = "compound_id"
)
```

Arguments

`data`	An `hts_norm` object or long data frame.
`value`	Name of the activity column. Default `"inhibition"`.
`method`	How to estimate the distribution centre and spread: `"robust"` (default) uses the median and MAD (scaled by 1.4826 to match the SD of a normal); `"sd"` uses the plain mean and standard deviation.
`n_sigma`	Number of sigma a compound must clear to be called a hit. Default `3`.
`direction`	Tail to test: `"greater"` (default, high activity), `"less"` (low activity) or `"two.sided"` (either tail, scored on `abs(z)`).
`compound_col`	Name of the compound column. Default `"compound_id"`.

Value

A data frame with one row per compound: `compound_id`, `activity`, `sigma` (signed z-score), `sigma_level` (0-3, how many sigma it clears in the tested direction), `band` (an ordered-factor label) and `is_hit` (`sigma_level >= n_sigma`). The distribution `center`, `scale`, `method`, `n_sigma` and `direction` are stored as attributes.

See Also

summarize_sigma_hits(), plot_sigma_hits(), select_primary_hit()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
hits <- select_sigma_hits(norm, n_sigma = 3)
sum(hits$is_hit)
```

---

summarize_plate_setup *Summarise how each plate in a run is set up*

Description

Reports, for every plate in an hts_raw object, how the plate is laid out: its physical format, the number of wells of each role (negative/positive controls, blanks and compounds), how many distinct compounds it carries, and which columns the controls and compounds occupy. The positions are **read back from your own layout** (see read_plate_layout()) rather than assumed by the package: because a screening run normally spans **several plates at once**, this is the quickest way to confirm that every plate carries the layout you designed before QC and normalisation.

Usage

```
summarize_plate_setup(
  hts_raw,
  plate_col = "plate_id",
  type_col = "well_type",
  compound_col = "compound_id",
  col_col = "col",
  nc_label = "NC",
  pc_label = "PC",
  blank_label = "Blank",
  compound_label = "compound"
)
```

Arguments

hts_raw	An hts_raw object from read_hts_plate() (or any well-level data frame with plate, well-coordinate and well-type columns).
plate_col, type_col, compound_col, col_col	Names of the plate, well-type, compound and column-index columns. Defaults match read_hts_plate() output.
nc_label, pc_label, blank_label, compound_label	Well-type labels for the four roles.

Value

A data frame with one row per plate and the columns `plate_id`, `n_wells`, `plate_format`, `n_rows`, `n_cols`, `n_nc`, `n_pc`, `n_blank`, `n_compound`, `n_unique_compound`, `nc_columns`, `pc_columns`, `blank_columns` and `compound_columns` (control/compound positions given as compact column ranges such as `"3-11"`).

See Also

`read_hts_plate()`, `check_control_label()`, `calc_z_prime()`

Examples

```
raw <- read_hts_plate(hts_primary_raw)
# One row per plate; the run holds three 96-well plates here.
summarize_plate_setup(raw)
```

---

summarize_primary_hit *Summarise primary-screen hits*

Description

Reports the number of compounds screened, the number of hits and the hit rate.

Usage

```
summarize_primary_hit(hits)
```

Arguments

`hits`	A data frame from `select_primary_hit()`.

Value

A one-row data frame with `n_compounds`, `n_hits`, `hit_rate` (fraction) and `cutoff`.

See Also

`select_primary_hit()`

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
summarize_primary_hit(select_primary_hit(norm))
```

summarize_sigma_hits *Summarise sigma-based hits*

Description

Counts how many compounds clear 1, 2 and 3 sigma from the library centre and compares each count to what a perfectly normal, hit-free distribution would predict, so enrichment of the tail over chance is obvious.

Usage

```
summarize_sigma_hits(hits)
```

Arguments

hits	A data frame from select_sigma_hits().

Value

A data frame with one row per sigma level (1, 2, 3): sigma, n_beyond (compounds clearing it), frac_beyond, expected_frac (normal-theory tail probability) and expected_n (expected count).

See Also

select_sigma_hits(), plot_sigma_hits()

Examples

```
norm <- norm_by_control(baseline_subtract(read_hts_plate(hts_primary_raw)))
summarize_sigma_hits(select_sigma_hits(norm))
```

theme_hts *Modern academic ggplot2 theme for diffHTS figures*

Description

A minimal, publication-ready theme: pure-white background, an L-shaped axis (left and bottom only), light dashed horizontal major grid lines with no minor grid, restrained dark-grey typography and generous margins. Pair it with hts_pal() for discrete colours or a viridis/muted-blue gradient for continuous ones.

Usage

```
theme_hts(
  base_size = 12,
  base_family = "",
  legend = "bottom",
  grid = c("y", "x", "both", "none")
)
```

Arguments

base_size	Base font size in points. Default 12.
base_family	Base font family; defaults to the device sans-serif.
legend	Legend position passed to ggplot2::theme(). Default "bottom".
grid	Which major grid lines to keep: "y" (default), "x", "both" or "none".

Value

A ggplot2::theme object that can be added to any ggplot.

See Also

hts_pal(), plot_plate_qc()

Examples

```
library(ggplot2)
ggplot(mtcars, aes(factor(cyl), mpg)) +
  geom_boxplot() +
  theme_hts()
```

---

windows_to_linux_path *Convert a Windows path to a forward-slash path*

Description

Replaces backslashes with forward slashes so that Windows-style paths can be used consistently across platforms.

Usage

```
windows_to_linux_path(path)
```

Arguments

path	A character vector of file paths.

Value

A character vector of the same length with backslashes replaced by forward slashes.

Examples

```
windows_to_linux_path("C:\\data\\EXP99\\raw.txt")
```

Index