

Package: PERCEPTIONx (via r-universe)

September 19, 2026

Title Personalized Drug Response Prediction from Single-Cell Transcriptomics

Version 0.3.0

Description An R implementation of the PERCEPTION approach (Sinha et al., 2024) for predicting personalized drug response from single-cell tumor transcriptomics. It trains drug-response models on large-scale cell-line screens, predicts sensitivity at clone and patient level, and ships an interactive visualization suite plus a Shiny web application.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

VignetteBuilder knitr

Imports ggplot2, gridExtra, glmnet, caret, randomForest, pROC, rlang, Seurat, uwot, shiny, ggiraph, parallel

URL <https://wanglabcsu.github.io/PERCEPTIONx>,
<https://github.com/WangLabCSU/PERCEPTIONx>

BugReports <https://github.com/WangLabCSU/PERCEPTIONx/issues>

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggpubr, bslib, DT, plotly, waiter, thematic, readxl, callr

Config/testthat/edition 3

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev python3 zlib1g-dev

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

Date/Publication 2026-09-19 09:52:18 UTC

RemoteUrl <https://github.com/WangLabCSU/PERCEPTIONx>

RemoteRef HEAD

RemoteSha 15fbc84a04c95a1306a938ada96c02e16de98d54

Contents

add_mirrors	3
annotate_clones	3
build_clone_counts	5
build_clone_key	5
build_on_BULK_v2	6
clone_mean_expression	7
compare_performance	8
cor.test_trimmed_v0	9
each_patient_pseudo_bulk	9
export_plot_cairo	10
fdrcorr	11
feature_ranking_bulk	11
get_cellLine_list	12
get_depmap	13
get_mirrors	14
get_performance	14
get_response_matrix	15
get_significant_models	15
hypergeometric_test_for_twolists	16
list_mirrors	17
load_depmap	17
load_model	18
parse_clone_keys	20
plot_clone_distribution	20
plot_clone_umap	21
plot_clone_viability	22
plot_model_performance	24
plot_model_roc	25
plot_patient_response_panel	26
plot_response_boxplot	27
plot_roc_curve	28
plot_seurat_clustering	29
plot_tsne_biomarker_viability	30
plot_tsne_response	31
predict_drugs	33
predict_patients	34
prepare_data	35
range01	38
rank_normalization_mat	38
reset_mirrors	39
run_parallel_feature_ranking_bulk	40
run_perception_app	41
train_models	41
zscore_viability	43

Index

44

add_mirrors	<i>Add custom download mirrors</i>
-------------	------------------------------------

Description

Add custom download mirrors

Usage

```
add_mirrors(urls, position = c("first", "last"))
```

Arguments

urls	Character vector of mirror URLs to add.
position	Character. Where to add: "first" or "last". Default: "first".

Value

Invisibly returns the updated mirror list.

Examples

```
## Not run:
add_mirrors("https://my-mirror.com/https://github.com")
add_mirrors(c("https://mirror1.com/https://github.com",
              "https://mirror2.com/https://github.com"))

## End(Not run)
```

annotate_clones	<i>Annotate cells with clone IDs via Seurat clustering</i>
-----------------	--

Description

Performs Seurat clustering on a single-cell expression matrix and returns a mapping of each cell to its cluster (clone) ID. This matches the original PERCEPTIONx pipeline where Seurat clusters define transcriptional subclones.

Usage

```
annotate_clones(
  method = c("umap", "tsne"),
  expression_matrix,
  min_cells = 3,
  min_features = 200,
  nfeatures = 2000,
```

```

    dims = 10,
    resolution = 0.8,
    seed = 42,
    progress_cb = NULL,
    cluster_algorithm = 1,
    variable_selection = c("vst", "dispersion", "mvp")
  )

```

Arguments

method	Character. Dimensionality reduction method. One of "umap" (default) or "tsne". UMAP is faster and preserves global structure better; t-SNE emphasizes local neighborhoods.
expression_matrix	Matrix. Gene expression matrix with genes as rows and cells as columns. Raw counts or normalized values are both accepted.
min_cells	Integer. Minimum cells per feature. Default = 3.
min_features	Integer. Minimum features per cell. Default = 200.
nfeatures	Integer. Number of variable features. Default = 2000.
dims	Integer. Number of PCA dimensions for clustering. Default = 10.
resolution	Numeric. Clustering resolution. Default = 0.8.
seed	Integer. Random seed for reproducibility. Default = 42.
progress_cb	Optional function called with the current Seurat stage name (e.g. "pca", "clustering") between pipeline steps. Used to show live progress in the Shiny app.
cluster_algorithm	Integer. Community-detection algorithm passed to <code>Seurat::FindClusters()</code> . Default = 1 (Louvain, no extra dependency). Use 4 (Leiden) only if the leiden-base package is installed.
variable_selection	Character. Variable-feature selection method for <code>Seurat::FindVariableFeatures()</code> . One of "vst" (default, most accurate), "dispersion" or "mvp" (faster on large matrices at a small cost in feature quality).

Value

A data frame with columns: cell_id, clone_id, and dim_1, dim_2 (2D embedding coordinates for visualization).

Examples

```

## Not run:
cell_clone_map <- annotate_clones(patient_expression)
cell_clone_map <- annotate_clones("tsne", patient_expression)

## End(Not run)

```

build_clone_counts	<i>Build clone abundance table from cell-clone mapping</i>
--------------------	--

Description

Computes the number of cells per clone per patient, producing the clone_counts data frame required by predict_patients().

Usage

```
build_clone_counts(cell_clone_map, patient_ids)
```

Arguments

cell_clone_map Data frame with columns cell_id and clone_id.
patient_ids Character vector. Patient ID for each cell, in the same order as rows in cell_clone_map.

Value

A data frame with first column patients and remaining columns as clone IDs with cell counts as values.

build_clone_key	<i>Build a clone key from patient and clone id</i>
-----------------	--

Description

The inverse of [parse_clone_keys\(\)](#): pastes patient and clone identifiers into a single "patient@clone_id" key. The "@" separator is the canonical clone-key format used throughout PERCEPTIONx.

Usage

```
build_clone_key(patient, clone_id)
```

Arguments

patient Character vector of patient IDs.
clone_id Character vector of clone IDs.

Value

Character vector of "patient@clone_id" keys.

See Also

[parse_clone_keys\(\)](#) for splitting keys back apart.

build_on_BULK_v2

Build PERCEPTIONx model on bulk expression data

Description

Core training function that builds a glmnet model using bulk expression, then evaluates on pseudo-bulk, bulk test, and single-cell test data. This is the direct port of build_on_BULK_v2 from step0B.

Usage

```
build_on_BULK_v2(
  infunc_drugName,
  infunc_cancerType = "PanCan",
  exclude_cancer = "PanCan",
  infunc_features,
  single_best = infunc_features[1],
  k_features = 100,
  infunc_alpha = 1,
  model_type = "glmnet",
  num_folds = 3,
  num_tree = 500,
  seed = 1,
  alpha_gradient = 0.05,
  lambda_gradient = 20,
  lambda_range = c(1e-04, 1),
  cv_method = "cv"
)
```

Arguments

infunc_drugName	Character. Drug name.
infunc_cancerType	Character. Cancer type for training. Default = "PanCan".
exclude_cancer	Character. Cancer type to exclude. Default = "PanCan".
infunc_features	Character vector. Feature gene names (ranked).
single_best	Character. Name of the single best feature. Default = infunc_features[1].
k_features	Integer. Number of top features to use. Default = 100.
infunc_alpha	Numeric. Alpha for glmnet (not used directly, tuning via alpha_gradient). Default = 1.
model_type	Character. Model type: "glmnet" or "rf". Default = "glmnet".
num_folds	Integer. Number of CV folds. Default = 3.
num_tree	Integer. Number of trees for random forest. Default = 500.

seed	Integer. Random seed for training. Default = 1.
alpha_gradient	Numeric. Step size for alpha grid in glmnet tuning. Default = 0.05.
lambda_gradient	Integer. Number of lambda values in glmnet tuning grid. Default = 20.
lambda_range	Numeric vector of length 2. Min and max lambda for tuning grid. Default = c(0.0001, 1).
cv_method	Character. Cross-validation method for caret::trainControl. Default = "cv".

Value

A list containing: model, single_best, model_performance_during_cv, performance_in_bulk, performance_in_pseudo_bulk, performance_in_scRNA, predVSgroundTruth.

clone_mean_expression *Compute clone-level mean expression from single-cell data*

Description

For each patient, groups cells by clone_id and computes the mean expression of each gene within each clone. This reduces single-cell resolution to subclone resolution, matching the original PERCEPTION pipeline.

Usage

```
clone_mean_expression(expression_matrix, cell_clone_map, patient_ids = NULL)
```

Arguments

expression_matrix	Matrix or data frame. Gene expression matrix with genes as rows and cells as columns.
cell_clone_map	Data frame with columns cell_id and clone_id, mapping each cell to its clone. Typically from Seurat clustering.
patient_ids	Character vector. Patient ID for each cell, same length as columns of expression_matrix. If NULL, all cells are assumed from one patient.

Value

A named list of matrices, one per patient. Each matrix has genes as rows and clone IDs as columns, with mean expression values.

compare_performance	<i>Compare performance of multiple trained models</i>
---------------------	---

Description

Computes and summarizes performance metrics across multiple drug models. Returns performance in cross-validation, bulk test, pseudo-bulk, and scRNA datasets.

Usage

```
compare_performance(model_list, threshold = 0.3, verbose = TRUE)
```

Arguments

<code>model_list</code>	A named list of trained model objects from <code>train_models()</code> .
<code>threshold</code>	Numeric. Minimum correlation threshold for "passing" models. Default = 0.3.
<code>verbose</code>	Logical. Whether to print summary statistics. Default = TRUE.

Value

A list containing:

perf_cv Cross-validation performance ($|\text{correlation}| = \sqrt{R^2}$; p-value not available) for each drug

perf_bulk Bulk test set performance for each drug

perf_pseudo_bulk Pseudo-bulk performance for each drug

perf_scRNA Single-cell RNA performance for each drug

summary Summary statistics of models passing threshold

Examples

```
## Not run:
models <- train_models(drug_list = c("abemaciclib", "erlotinib"))
perf <- compare_performance(models)

## End(Not run)
```

`cor.test_trimmed_v0` *Fast correlation test (generic)*

Description

A generic function for fast correlation testing. Dispatches methods based on the class of the first argument. This is a trimmed-down version of the base R `cor.test` function, optimized for speed in large-scale computations.

Usage

```
cor.test_trimmed_v0(x, ...)
```

Arguments

<code>x</code>	First argument (typically a numeric vector).
<code>...</code>	Additional arguments passed to methods.

Value

A list containing the p-value and correlation estimate. The exact structure depends on the method used.

See Also

[cor.test](#) for the full version with confidence intervals.

`each_patient_pseudo_bulk`
Compute pseudo-bulk expression for a patient

Description

Calculates weighted average of clone-level expression to produce a pseudo-bulk expression profile for a single patient. This is used for evaluating model performance on pseudo-bulk data, not for prediction.

Usage

```
each_patient_pseudo_bulk(  
  x = 1,  
  comb_viability_df,  
  Clone_Counts_per_patients,  
  clone_Level_z_expression_df  
)
```

Arguments

<code>x</code>	Integer. Index of the patient. Default = 1.
<code>comb_viability_df</code>	Data frame. Retained for backward compatibility; no longer used (clone ids are read from the expression column names).
<code>Clone_Counts_per_patients</code>	Data frame. Patient-clone abundance data.
<code>clone_Level_z_expression_df</code>	Matrix. Clone-level expression data with columns named by patient-clone identifiers.

Value

Named numeric vector of pseudo-bulk expression values.

<code>export_plot_cairo</code>	<i>Export plot to file via Cairo device</i>
--------------------------------	---

Description

High-resolution plot export using the Cairo rendering engine for superior anti-aliasing and cross-platform font rendering.

Usage

```
export_plot_cairo(
  file,
  plot,
  format = "png",
  width = 7,
  height = 5,
  res = 600,
  draw_fun = NULL
)
```

Arguments

<code>file</code>	Character. Output file path.
<code>plot</code>	A ggplot or grid object to export.
<code>format</code>	Character. One of "png", "svg", or "pdf". Default = "png".
<code>width</code>	Numeric. Plot width in inches. Default = 7.
<code>height</code>	Numeric. Plot height in inches. Default = 5.
<code>res</code>	Numeric. Output resolution (DPI) for PNG. Default = 600, minimum 96.
<code>draw_fun</code>	Function. Optional custom draw function (e.g. <code>gridExtra::grid.arrange</code> for multi-panel layouts). If NULL, <code>print(plot)</code> is used.

Value

Invisibly returns the file path. Called for its side effect of creating the file.

Examples

```
## Not run:
p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) + ggplot2::geom_point()
export_plot_cairo("plot.png", p)
export_plot_cairo("plot.pdf", p, format = "pdf")

## End(Not run)
```

fdrcorr

FDR correction for multiple testing

Description

Performs False Discovery Rate (FDR) correction using the Benjamini-Hochberg method. This is a wrapper around [p.adjust](#) with method = "fdr".

Usage

```
fdrcorr(test_list)
```

Arguments

`test_list` A numeric vector of p-values to be corrected.

Value

A numeric vector of FDR-adjusted p-values (q-values).

feature_ranking_bulk

Feature ranking for a single drug using bulk expression

Description

Computes Pearson correlation between each gene's expression and drug response, then ranks genes by absolute correlation. This matches the original `feature_ranking_bulk` function from `step0B`.

Usage

```
feature_ranking_bulk(
  infunc_drugName,
  infunc_cancerType = "PanCan",
  exclude_cancer = "PanCan",
  infunc_GOI
)
```

Arguments

infunc_drugName Character. Name of the drug.

infunc_cancerType Character. Cancer type for training. Default = "PanCan".

exclude_cancer Character. Cancer type to exclude. Default = "PanCan".

infunc_GOI Character vector. Genes of Interest to rank.

Value

Matrix of ranked features with columns: p.value, estimate.cor (sorted by abs(cor) descending)

get_cellLine_list	<i>Determine training and test cell-lines for a given drug</i>
-------------------	--

Description

Identifies which cell lines should be used for training and which should be excluded (reserved for testing) based on cancer type and single-cell data availability.

Usage

```
get_cellLine_list(
  infunc_cancerType = "PanCan",
  infunc_drugName,
  exclude_cancer = "PanCan",
  infunc_response,
  force_add_cellLines = TRUE,
  force_add_cellLines_list = NA
)
```

Arguments

infunc_cancerType Character. Cancer type for training. Default = "PanCan".

infunc_drugName Character. Name of the drug.

exclude_cancer Character. Cancer type to exclude from training. Default = "PanCan".
 infunc_response
 Named numeric vector. Drug response data.
 force_add_cellLines
 Logical. Whether to force add additional cell lines. Default = TRUE.
 force_add_cellLines_list
 Character vector. Cell line IDs to force add. Default = NA.

Value

A list of length 2:

- [1] common_cellLines: Cell lines to use for training
- [2] cellLines2remove: Cell lines excluded (reserved for testing)

get_depmap	<i>Get the DepMap dataset</i>
------------	-------------------------------

Description

Retrieves the DepMap dataset from the package-level cache. This is the recommended way to access DepMap data after loading with `load_depmap(read = TRUE)`.

Usage

```
get_depmap()
```

Value

The DepMap list object. Errors if the data has not been loaded yet.

Examples

```
## Not run:
load_depmap(read = TRUE)
depmap <- get_depmap()

## End(Not run)
```

get_mirrors	<i>Get current download mirrors</i>
-------------	-------------------------------------

Description

Returns the current mirror list, including any user-added mirrors. User-added mirrors are tried first by default.

Usage

```
get_mirrors()
```

Value

Character vector of mirror URLs.

Examples

```
get_mirrors()
```

get_performance	<i>Load performance metrics from a saved model file</i>
-----------------	---

Description

Loads a saved model RDS file and returns only its performance portion. The file is deserialized fully first (an RDS is read as a whole), then the performance entries are extracted.

Usage

```
get_performance(filepath, drug_names = NULL)
```

Arguments

filepath	Character. Path to a single saved model RDS file.
drug_names	Character vector. Optional. Specific drug names to load. If NULL, loads all drugs in the file.

Value

A list containing performance metrics for each drug model.

Examples

```
## Not run:
# Load performance from a single model file
perf <- get_performance("models/PERCEPTIONx_models_PanCan_exPanCan_20240101_120000.RDS")

# Or use compare_performance on the result
models <- readRDS("models/PERCEPTIONx_models_PanCan_exPanCan_20240101_120000.RDS")
perf <- compare_performance(models)

## End(Not run)
```

get_response_matrix	<i>Get specific drug response data for cell-lines</i>
---------------------	---

Description

Extracts the drug response (AUC) for a given drug from the DepMap database. The function handles multiple screening batches by prioritizing MTS over HTS and selecting the batch with the fewest missing values.

Usage

```
get_response_matrix(infunc_drugName)
```

Arguments

infunc_drugName
Character string. Name of the drug (e.g., "erlotinib").

Value

A named numeric vector of AUC values for all cell lines.

get_significant_models	<i>Get best performing models</i>
------------------------	-----------------------------------

Description

Filters model list to return only models that meet performance criteria. Useful for selecting significant drug models for downstream analysis.

Usage

```
get_significant_models(
  model_list,
  min_correlation = 0.3,
  max_pvalue = 0.05,
  dataset = "scRNA"
)
```

Arguments

model_list	A named list of trained model objects.
min_correlation	Numeric. Minimum correlation threshold. Default = 0.3.
max_pvalue	Numeric. Maximum p-value threshold. Default = 0.05.
dataset	Character. Which dataset to use for filtering: "scRNA" (default), "bulk", "pseudo_bulk", or "cv".

Value

A filtered list containing only models meeting the criteria.

Examples

```
## Not run:
models <- train_models()
significant_models <- get_significant_models(models, min_correlation = 0.3, max_pvalue = 0.05)

## End(Not run)
```

hypergeometric_test_for_twolists

Hypergeometric test for gene list overlap

Description

Tests whether the overlap between two gene lists is statistically significant. It answers the question: given a background set of genes (global), a reference gene set (base_list), and a test gene set (test_list), is the overlap between test_list and base_list more than expected by chance?

Usage

```
hypergeometric_test_for_twolists(
  test_list,
  base_list,
  global,
  lower.tail = FALSE
)
```

Arguments

test_list	Character vector. The test gene list.
base_list	Character vector. The reference gene list.
global	Character vector. The background gene set.
lower.tail	Logical. If FALSE (default), calculates the probability of observing <i>greater than or equal to</i> the observed overlap (enrichment). If TRUE, calculates the probability of observing <i>less than or equal to</i> the observed overlap (depletion).

Value

Numeric p-value.

list_mirrors	<i>List current download mirrors</i>
--------------	--------------------------------------

Description

List current download mirrors

Usage

```
list_mirrors()
```

load_depmap	<i>Download filtered DepMap data</i>
-------------	--------------------------------------

Description

Downloads the filtered DepMap RDS file for training models from GitHub Release. The file contains bulk expression, scRNA expression, drug response annotations, and cell line metadata required for PERCEPTIONx model training. Supports automatic mirror fallback for users in different regions.

Usage

```
load_depmap(
  dest = perception_default_depmap_dir(),
  read = FALSE,
  mirror = FALSE,
  mirror_url = NULL,
  timeout_seconds = 300,
  retries = 1,
  force = FALSE
)
```

Arguments

dest	Directory to save the downloaded file. Default: a cache-root derived path (see options(PERCEPTIONx.cache_root)), or "." when no cache root is set.
read	Whether to read the data and assign to global environment as "DepMap". Default = FALSE.
mirror	Logical. If FALSE (default), download from GitHub directly. If TRUE, use mirror sites from get_mirrors().
mirror_url	Character. A specific mirror URL to use (e.g., "https://gh-proxy.com/https://github.com"). Overrides mirror.
timeout_seconds	Numeric, timeout for each download attempt in seconds. Default = 300.
retries	Integer, number of retries for each mirror. Default = 1.
force	Logical. If TRUE, delete an existing cached file before downloading again (use after an interrupted/failed download left a corrupt file behind). Default = FALSE (keep and reuse the cache).

Value

Invisibly returns the DepMap object if read = TRUE, otherwise NULL.

Examples

```
## Not run:
# Download from GitHub directly
load_depmap(read = TRUE)

# Use mirror sites
load_depmap(read = TRUE, mirror = TRUE)

# Force a fresh download (removes any cached copy first)
load_depmap(read = TRUE, mirror = TRUE, force = TRUE)

# Use a specific mirror
load_depmap(read = TRUE, mirror_url = "https://gh-proxy.com/https://github.com")

## End(Not run)
```

load_model

Load pre-built model of provided drugs

Description

Downloads model files from GitHub Release if not cached locally. Supports automatic mirror fallback for users in different regions.

Usage

```
load_model(
  ...,
  dest = perception_default_model_dir(),
  read = FALSE,
  mirror = FALSE,
  mirror_url = NULL,
  timeout_seconds = 30,
  retries = 0,
  force = FALSE,
  all = FALSE
)
```

Arguments

...	One or more drug names (e.g., "erlotinib", "gefitinib").
dest	Directory to save downloaded models. Default: a cache-root derived path (see <code>options(PERCEPTIONx.cache_root)</code>), or <code>"/models"</code> when no cache root is set.
read	Whether to read and return the downloaded model(s) as a named list. Default = FALSE (download only).
mirror	Logical. If FALSE (default), download from GitHub directly. If TRUE, use mirror sites from <code>get_mirrors()</code> .
mirror_url	Character. A specific mirror URL to use (e.g., <code>"https://gh-proxy.com/https://github.com"</code>). Overrides mirror.
timeout_seconds	Numeric, timeout for each download attempt in seconds. Default = 30.
retries	Integer, number of retries for each mirror. Default = 0.
force	Logical. If TRUE, delete an existing cached model file before downloading it again (use after an interrupted/failed download left a corrupt file behind). Default = FALSE (keep and reuse the cache).
all	Logical. If TRUE, download all 44 pre-trained models (useful to pre-cache the full model set on a deployment server). Overrides any drug names given in ... Default = FALSE.

Value

If `read = TRUE`, a named list of model objects (names = drug names). If `read = FALSE`, invisibly returns NULL (files are saved to disk only). The returned list can be directly passed to `predict_drugs()`, `compare_performance()`, or `get_significant_models()`.

Examples

```
## Not run:
# Download and load models from GitHub directly
models <- load_model("erlotinib", "gefitinib", read = TRUE)

# Use mirror sites
```

```
models <- load_model("erlotinib", "gefitinib", read = TRUE, mirror = TRUE)

# Use a specific mirror
models <- load_model("erlotinib", read = TRUE,
                    mirror_url = "https://gh-proxy.com/https://github.com")

## End(Not run)
```

parse_clone_keys	<i>Parse clone keys in "Patient@Clone" format</i>
------------------	---

Description

Splits a character vector of clone identifiers (e.g. "PAT_001@0") into a data frame with `patient` and `clone_id` columns.

Usage

```
parse_clone_keys(x)
```

Arguments

`x` Character vector of clone keys in "patient@clone_id" format.

Value

A data frame with columns `patient` and `clone_id`.

See Also

[build_clone_key\(\)](#) for the inverse operation.

plot_clone_distribution	<i>Plot clone distribution as stacked bar</i>
-------------------------	---

Description

Visualizes the proportion of each clone across patients as a stacked bar plot. Useful for understanding tumor heterogeneity and clonal architecture.

Usage

```
plot_clone_distribution(
  clone_distribution,
  response_var = NULL,
  base_size = 15,
  tooltip = TRUE,
  tooltip_col = NULL
)
```

Arguments

clone_distribution	Data frame with columns: patients, clones, weights.
response_var	Character. Optional column name for response annotation. If provided, facets by response. Default = NULL.
base_size	Numeric. Base font size. Default = 15.
tooltip	Logical. If TRUE (default) and ggiraph is installed, bar segments get hover tooltips (clone + proportion).
tooltip_col	Character. Optional existing column used as the tooltip text. Default = NULL (auto-builds a rich tooltip).

Value

A ggplot object.

Examples

```
## Not run:
# After computing clone weights
clone_dist <- data.frame(
  patients = c("P1", "P1", "P1", "P2", "P2", "P2"),
  clones = c("c1", "c2", "c3", "c1", "c2", "c3"),
  weights = c(0.3, 0.5, 0.2, 0.6, 0.3, 0.1)
)
plot_clone_distribution(clone_dist)

## End(Not run)
```

plot_clone_umap

Plot UMAP colored by clone identity

Description

Single cells in the 2D embedding colored by clone identity – the spatial analogue of the paper’s Extended Data Fig. 8a, showing at a glance which transcriptional subclones sit where.

Usage

```
plot_clone_umap(
  tsne_data,
  clone_col = "clone_id",
  title = NULL,
  color_label = "Clone",
  point_size = NULL,
  base_size = 11,
  tooltip = TRUE,
  tooltip_col = NULL
)
```

Arguments

tsne_data	Data frame with columns: X, Y, and the clone column.
clone_col	Character. Name of the column holding clone ids. Default = "clone_id".
title	Character. Plot title. Default = NULL.
color_label	Character. Legend label. Default = "Clone".
point_size	Numeric. Point size. If NULL (default), auto-adapts to the number of cells to avoid overplotting.
base_size	Numeric. Base font size. Default = 11.
tooltip	Logical. If TRUE (default) and ggiraph is installed, points get hover tooltips (clone id).
tooltip_col	Character. Optional existing column used as the tooltip text. Default = NULL (auto-builds from the clone id).

Value

A ggplot object.

plot_clone_viability *Plot clone-level viability (lollipop plot)*

Description

Visualizes predicted drug sensitivity for each clone within patients. Each clone is represented as a point with a stem (lollipop style). Useful for identifying resistant clones within heterogeneous tumors.

Usage

```
plot_clone_viability(
  clone_viability,
  viability_var = "comb_viability",
  weights_var = NULL,
  response_var = NULL,
  drug = NULL,
  base_size = 11,
  y_limits = c(-3, 1.2),
  viridis_scale = FALSE,
  tooltip = TRUE,
  tooltip_col = NULL
)
```

Arguments

clone_viability	Data frame with columns: patient, clone_id, and the viability column named by viability_var. Optionally also weights_var (clone proportion) and response_var (clinical response) columns.
viability_var	Character. Column name for viability values. Default = "comb_viability".
weights_var	Character. Optional column name for clone weights (point size). Default = NULL.
response_var	Character. Optional column for response annotation. Default = NULL.
drug	Character. Drug name, used as plot title. Default = NULL.
base_size	Numeric. Base font size. Default = 11.
y_limits	Numeric vector. Y-axis limits. Default = c(-3, 1.2).
viridis_scale	Logical. If TRUE, uses a viridis sequential scale (dark = low/sensitive, yellow = high/resistant). If FALSE (default), uses the diverging red-blue scale (blue = low/sensitive, white = neutral, red = high/resistant) with data-driven limits, so every value keeps a real color – nothing clips to grey/NA outside a fixed window.
tooltip	Logical. If TRUE (default) and ggiraph is installed, points get hover tooltips (clone + viability + proportion).
tooltip_col	Character. Optional existing column used as the tooltip text. Default = NULL (auto-builds a rich tooltip).

Details

Facet strategy is adaptive:

- ≤ 12 patients: single-row grid, strips on the bottom (45 deg).
- > 12 patients: facet_wrap grid, one compact panel per patient.

Value

A ggplot object.

Examples

```
## Not run:
clone_kill <- data.frame(
  patient = c("P1", "P1", "P2", "P2"),
  clone_id = c("c1", "c2", "c1", "c2"),
  comb_viability = c(-0.5, 0.8, -1.2, 0.3)
)
plot_clone_viability(clone_kill, viability_var = "comb_viability")

## End(Not run)
```

plot_model_performance

Plot model performance across datasets

Description

Visualizes the number of drugs achieving different correlation thresholds across bulk, pseudo-bulk, and single-cell datasets.

Usage

```
plot_model_performance(
  performance_list,
  threshold_range = NULL,
  base_size = 20,
  highlight_threshold = 0.3,
  tooltip = TRUE
)
```

Arguments

performance_list	Named list of model performance objects (from train_perception_models).
threshold_range	Numeric vector. Correlation thresholds to evaluate. Default = NULL (auto: from 0.1 up to at least 0.6, extended to the highest observed correlation so the curve always reaches 0).
base_size	Numeric. Base font size. Default = 20.
highlight_threshold	Numeric. Threshold to highlight with vertical line. Default = 0.3.
tooltip	Logical. If TRUE (default) and ggiraph is installed, points get hover tooltips (dataset, threshold, drug count).

Value

A ggplot object.

Examples

```
## Not run:
models <- train_perception_models(c("abemaciclib", "erlotinib"), ...)
plot_model_performance(models)

## End(Not run)
```

plot_model_roc	<i>Plot validation ROC curves for trained models</i>
----------------	--

Description

For each model and each validation dataset (bulk, pseudo-bulk, single-cell), the observed response is stratified into the top vs bottom 50\ (resistant vs sensitive – the PERCEPTION paper’s convention, Extended Data Fig. 4C) and a ROC curve of the predicted viability is drawn, one curve per dataset, with the AUC annotated in the legend. Higher AUC = the model stratifies better. This is the most informative single-model summary after training.

Usage

```
plot_model_roc(performance_list, base_size = 13)
```

Arguments

performance_list	Named list of model objects from train_models(), each with a \$predVSgroundTruth element.
base_size	Numeric. Base font size. Default = 13.

Value

A ggplot object: ROC curves per validation dataset, faceted by drug when more than one model is provided.

Examples

```
## Not run:
models <- train_models(drug_list = "abemaciclib", ...)
plot_model_roc(models)

## End(Not run)
```

plot_patient_response_panel

Complete patient response visualization pipeline

Description

Generates a comprehensive visualization panel for patient drug response prediction, including clone distribution, clone-level viability, response boxplot, and ROC curve. This is a convenience function that combines multiple plot functions.

Usage

```
plot_patient_response_panel(
  clone_distribution,
  clone_viability,
  exp_vs_pred,
  response_col = "response",
  viability_col = "comb_viability",
  predicted_col = "predicted_viability",
  weights_col = "weights",
  layout_matrix = NULL
)
```

Arguments

clone_distribution	Data frame. Clone weights per patient.
clone_viability	Data frame. Viability scores per clone.
exp_vs_pred	Data frame. Predicted vs observed response.
response_col	Character. Response column name. Default = "response".
viability_col	Character. Viability column name. Default = "comb_viability".
predicted_col	Character. Predicted values column name. Default = "predicted_viability".
weights_col	Character. Weights column name. Default = "weights".
layout_matrix	Matrix. Layout for grid.arrange. Default = NULL (auto).

Value

A gtable object from grid.arrange.

Examples

```
## Not run:
# After running prediction pipeline
panel <- plot_patient_response_panel(
  clone_distribution = clone_dist_df,
```

```

    clone_viability = clone_kill_df,
    exp_vs_pred = response_df
  )
  ggsave(panel, filename = "patient_response.pdf", height = 15, width = 10)

## End(Not run)

```

plot_response_boxplot *Plot predicted vs observed response boxplot*

Description

Creates a combined violin + box + jitter plot comparing predicted viability between response groups (e.g., Responders vs Non-Responders) with a statistical significance bracket.

Usage

```

plot_response_boxplot(
  exp_vs_pred,
  response_var = "response",
  predicted_var = "predicted_viability",
  y_label = "Predicted Viability (z-score)",
  base_size = 15,
  compare_method = "wilcox.test",
  alternative = "less",
  tooltip = TRUE
)

```

Arguments

exp_vs_pred	Data frame with columns: response, predicted_viability.
response_var	Character. Column name for response labels. Default = "response".
predicted_var	Character. Column name for predicted values. Default = "predicted_viability".
y_label	Character. Y-axis label. Default = "Predicted Viability (z-score)".
base_size	Numeric. Base font size. Default = 15.
compare_method	Character. Statistical test method. One of "wilcox.test" (default) or "t.test".
alternative	Character. Alternative hypothesis direction. Default = "less".
tooltip	Logical. If TRUE (default) and ggiraph is installed, jittered points get hover tooltips.

Value

A ggplot object.

Examples

```
## Not run:
exp_pred <- data.frame(
  response = factor(c("R", "NR", "R", "NR")),
  predicted_viability = c(0.8, 0.2, 0.7, 0.3)
)
plot_response_boxplot(exp_pred)

## End(Not run)
```

plot_roc_curve

Plot ROC curve with AUC annotation

Description

Generates a ROC curve from predicted vs observed response with AUC value annotation.

Usage

```
plot_roc_curve(
  response,
  predictor,
  smooth_curve = TRUE,
  base_size = 15,
  auc_digits = 3,
  title = NULL,
  tooltip = TRUE
)
```

Arguments

response	Factor or numeric. True response labels (e.g., "R"/"NR" or 0/1).
predictor	Numeric. Predicted values (e.g., viability scores).
smooth_curve	Logical. Whether to smooth the ROC curve. Default = TRUE.
base_size	Numeric. Base font size. Default = 15.
auc_digits	Integer. Number of digits for AUC display. Default = 3.
title	Character. Plot title. Default = NULL.
tooltip	Logical. If TRUE (default) and ggiraph is installed, curve points get hover tooltips (FPR / TPR).

Value

A ggplot object.

Examples

```
## Not run:
response <- factor(c("R", "NR", "R", "NR", "R"))
predictor <- c(0.8, 0.2, 0.7, 0.3, 0.9)
plot_roc_curve(response, predictor)

## End(Not run)
```

plot_seurat_clustering

Run Seurat clustering and plot 2D embedding

Description

Performs Seurat clustering on an expression matrix and generates a 2D embedding visualization (UMAP or t-SNE). Useful for identifying subclones within patient tumor samples.

Usage

```
plot_seurat_clustering(
  method = c("umap", "tsne"),
  expression_matrix,
  min_cells = 3,
  min_features = 200,
  nfeatures = 2000,
  dims = 10,
  resolution = 0.8,
  seed = 1
)
```

Arguments

method	Character. Dimensionality reduction method. One of "umap" (default) or "tsne".
expression_matrix	Matrix. Gene expression matrix (genes as rows, cells as columns).
min_cells	Integer. Minimum cells per feature. Default = 3.
min_features	Integer. Minimum features per cell. Default = 200.
nfeatures	Integer. Number of variable features. Default = 2000.
dims	Integer. Number of PCA dimensions for clustering. Default = 10.
resolution	Numeric. Clustering resolution. Default = 0.8.
seed	Integer. Random seed. Default = 1.

Value

A list containing:

```
seurat_object   Seurat object with clustering results
embedding_plot  ggplot 2D embedding visualization
cluster_ids     Named vector of cluster IDs per cell
```

Examples

```
## Not run:
result <- plot_seurat_clustering(patient_expression)
result$embedding_plot
result$cluster_ids

## End(Not run)
```

```
plot_tsne_biomarker_viability
```

Plot UMAP side-by-side for biomarker and viability

Description

Creates a side-by-side comparison of biomarker expression and predicted viability in UMAP space. Useful for visualizing correlation between marker and response.

Usage

```
plot_tsne_biomarker_viability(
  tsne_data,
  biomarker_var = "biomarker_scaled",
  viability_var = "viability_scaled",
  biomarker_label = "Biomarker Exp",
  viability_label = "Drug Viability",
  nrow = 1,
  base_size = 8
)
```

Arguments

```
tsne_data      Data frame with X, Y coordinates and both biomarker/viability columns.
biomarker_var  Character. Column name for biomarker expression. Default = "biomarker_scaled".
viability_var  Character. Column name for viability values. Default = "viability_scaled".
biomarker_label
                Character. Legend label for biomarker. Default = "Biomarker Exp".
```

viability_label	Character. Legend label for viability. Default = "Drug Viability".
nrow	Integer. Number of rows in arrangement. Default = 1.
base_size	Numeric. Base font size. Default = 8.

Value

A gtable object from grid.arrange.

Examples

```
## Not run:
tsne_data <- data.frame(
  X = lung_tsne$X,
  Y = lung_tsne$Y,
  biomarker_scaled = range01(rank(MDM2_expression)),
  viability_scaled = range01(rank(-viability_pred))
)
plot_tsne_biomarker_viability(tsne_data)

## End(Not run)
```

plot_tsne_response	<i>Plot UMAP with drug response overlay</i>
--------------------	---

Description

Visualizes single cells in UMAP space with color overlay representing either biomarker expression or predicted drug sensitivity.

Usage

```
plot_tsne_response(
  tsne_data,
  color_var = "viability_scaled",
  title = NULL,
  color_label = "Predicted Viability",
  point_size = NULL,
  colors = NULL,
  palette = c("viridis", "diverging"),
  midpoint = 0,
  limits = NULL,
  base_size = 11,
  tooltip = TRUE,
  tooltip_col = NULL
)
```

Arguments

tsne_data	Data frame with columns: X, Y (coordinates), and optional biomarker/viability columns.
color_var	Character. Name of the column to use for color mapping. Default = "viability_scaled".
title	Character. Plot title. Default = NULL.
color_label	Character. Legend label for color. Default = "Predicted Viability".
point_size	Numeric. Point size. If NULL (default), auto-adapts to the number of cells to avoid overplotting.
colors	Character vector. Custom gradient colors (low, mid, high) for the sequential palette. Default = NULL (uses built-in viridis).
palette	Character. One of "viridis" (sequential, default) or "diverging" (blue-white-red centered at midpoint).
midpoint	Numeric. Center value for diverging palette. Default = 0.
limits	Numeric vector of length 2. Optional fixed scale limits (e.g. c(0, 1) for 0-1 expression) to pin the color at each end. Default = NULL (data-driven limits – recommended, so extreme values always keep a real color instead of clipping to grey/NA).
base_size	Numeric. Base font size for theme. Default = 11.
tooltip	Logical. If TRUE (default) and ggiraph is installed, points get hover tooltips (cell id if present, else the colored value).
tooltip_col	Character. Optional existing column used as the tooltip text (e.g. "cell_id"). Default = NULL (auto-builds from color_var).

Value

A ggplot object.

Examples

```
## Not run:
# After predicting viability for single cells
tsne_data <- data.frame(
  X = lung_tSNE$X,
  Y = lung_tSNE$Y,
  viability_scaled = range01(rank(-viability_pred))
)
plot_tsne_response(tsne_data, color_var = "viability_scaled")

## End(Not run)
```

predict_drugs	<i>Predict drug response for cells or clones</i>
---------------	--

Description

Given a trained model (or list of models) and a rank-normalized expression matrix, predicts viability scores for each cell/sample across one or more drugs. This function merges the former `viability_from_model` (single drug) and `viability_in_each_dataset` (multi-drug) into a unified interface.

Usage

```
predict_drugs(model_list, expr)
```

Arguments

<code>model_list</code>	A named list of model objects (each with a <code>\$model</code> element), or a single model object. From <code>train_models()</code> or <code>load_model()</code> .
<code>expr</code>	Matrix or data frame. Rank-normalized expression matrix with genes as rows and cells/samples as columns.

Details

If a small fraction ($\leq 50\%$) `expr` (e.g. genes filtered out during scRNA QC), they are imputed with the neutral rank value 0.5 and a warning is issued. If more than half of the features are missing, prediction stops with an error.

Value

A matrix with cells/samples as rows and drugs as columns, containing predicted viability scores. Lower values indicate higher drug sensitivity.

Examples

```
## Not run:
# Single drug
models <- load_model("erlotinib", read = TRUE)
pred <- predict_drugs(models, expr_rnorm)

# Multiple drugs
models <- train_models(drug_list = c("abemaciclib", "erlotinib"),
                      cancer_type = "PanCan", exclude_cancer = "PanCan", GOI = GOI)
pred <- predict_drugs(models, expr_rnorm)

## End(Not run)
```

predict_patients	<i>Predict drug response at patient level</i>
------------------	---

Description

Aggregates clone-level drug response predictions to patient-level using various weighting strategies based on clone abundance. This function merges the former `each_patient_viability` (single drug) and `each_patient_viabilityv2` (multi-drug) into a unified interface.

Usage

```
predict_patients(
  clone_pred,
  prepared_data,
  clone_counts = NULL,
  mode = "weighted_max",
  zscore = TRUE
)
```

Arguments

<code>clone_pred</code>	Matrix from <code>predict_drugs()</code> , with clones as rows and drugs as columns. Row names should match clone column names from <code>prepare_data()</code> .
<code>prepared_data</code>	List from <code>prepare_data()</code> , containing <code>\$clone_viability_template</code> and <code>\$clone_counts</code> . Alternatively, you can pass <code>clone_counts</code> directly as a data frame (legacy mode).
<code>clone_counts</code>	Optional. Data frame with patients as rows and clone IDs as columns. Only needed if <code>prepared_data</code> is not a list from <code>prepare_data()</code> .
<code>mode</code>	Character. Aggregation method: "weighted_max" Maximum of weighted viability across clones. Default. "max" Maximum viability across clones (most resistant clone) "weighted_average" Weighted average of clone viability by clone abundance "min" Minimum viability across clones (most sensitive clone) "average" Average of clone viability
<code>zscore</code>	Logical. Whether to z-score scale drug columns across patients before aggregation. Default = TRUE. Matches the original PERCEPTION pipeline. An already-standardized <code>comb_viability</code> column (the output of the IDA combination step) is never scaled again.

Value

A data frame with patients as rows and drugs as columns, containing aggregated viability scores.

Examples

```
## Not run:
# Simple workflow: pass prepare_data() output directly
prepared <- prepare_data(patient_scRNA)
clone_pred <- predict_drugs(models, prepared$clone_expression)
patient_pred <- predict_patients(clone_pred, prepared)

# Legacy workflow: manually build clone_viability_matrix
clone_viability_df <- data.frame(
  patient = patient_ids,
  clone_id = clone_ids,
  clone_pred
)
patient_pred <- predict_patients(clone_viability_df, clone_counts)

## End(Not run)
```

```
prepare_data
```

```
Prepare patient data for PERCEPTIONx prediction
```

Description

End-to-end preprocessing pipeline that takes raw single-cell expression data and produces a rank-normalized subclone expression matrix and clone counts table, ready for direct use with `predict_drugs()` and `predict_patients()`.

Usage

```
prepare_data(
  method = c("umap", "tsne"),
  expression_matrix,
  patient_mapping = NULL,
  cell_col = "cell_id",
  patient_col = "patient_id",
  parse_patient = FALSE,
  patient_sep = "_",
  patient_pos = 1,
  genes_to_use = NULL,
  seurat_resolution = 0.8,
  seurat_dims = 10,
  seurat_nfeatures = 2000,
  seurat_min_cells = 3,
  seurat_min_features = 200,
  seurat_seed = 42,
  skip_clustering = FALSE,
  progress_cb = NULL
)
```

Arguments

method	Character. Dimensionality reduction method passed to <code>annotate_clones()</code> . One of "umap" (default) or "tsne". UMAP is faster and preserves global structure; t-SNE emphasizes local neighborhoods.
expression_matrix	Matrix. Gene expression matrix with genes as rows and cells as columns.
patient_mapping	List or data frame. Patient-cell mapping in one of two formats: List format Named list where each element is a patient ID and contains a character vector of cell IDs. Example: <code>list(Patient_1 = c("Cell_1", "Cell_2"), Patient_2 = c("Cell_3"))</code> Data frame format Metadata with cell ID and patient ID columns. Specify column names via <code>cell_col</code> and <code>patient_col</code> . Example: <code>data.frame(cell_id = c("Cell_1", "Cell_2"), patient_id = c("P1", "P1"))</code> If NULL, all cells are assigned to a single patient "patient1".
cell_col	Character. Cell ID column name in <code>patient_mapping</code> data frame. Default = "cell_id". Only used when <code>patient_mapping</code> is a data frame.
patient_col	Character. Patient ID column name in <code>patient_mapping</code> data frame. Default = "patient_id". Only used when <code>patient_mapping</code> is a data frame.
parse_patient	Logical. If TRUE, parse patient ID from <code>cell_col</code> using separator. Default = FALSE. Auto-enabled if <code>patient_sep</code> or <code>patient_pos</code> is provided. Useful when cell names contain patient info (e.g., "P11_M_Barcode").
patient_sep	Character. Separator to split <code>cell_col</code> for parsing patient ID. Default = "_". Providing this parameter auto-enables <code>parse_patient</code> .
patient_pos	Integer. Position of patient ID after splitting. Default = 1 (first element). Providing this parameter auto-enables <code>parse_patient</code> .
genes_to_use	Character vector. Genes to retain in the output matrix. If NULL, all genes in the expression matrix are used.
seurat_resolution	Numeric. Clustering resolution. Default = 0.8.
seurat_dims	Integer. PCA dimensions for clustering. Default = 10.
seurat_nfeatures	Integer. Variable features count. Default = 2000.
seurat_min_cells	Integer. Minimum cells per feature. Default = 3.
seurat_min_features	Integer. Minimum features per cell. Default = 200. Auto-adjusted to 10% of gene count if the expression matrix has fewer genes.
seurat_seed	Integer. Random seed. Default = 42.
skip_clustering	Logical. If TRUE, skip the Seurat clustering step and treat every column of <code>expression_matrix</code> as one pre-defined clone. Use this when you already have a clone-level expression matrix (e.g. from a published study). Rank normalization and clone counts are still applied. No UMAP/t-SNE embedding is produced in this mode.

progress_cb Optional function called as `progress_cb(phase, i, n, drug)` with the current pipeline stage (e.g. "seurat-pca", "mapping"). Used to show live progress in the Shiny app.

Details

The pipeline performs:

1. Seurat clustering to define transcriptional subclones
2. Cell-to-patient and cell-to-clone annotation
3. Clone-level mean expression computation
4. Rank normalization of clone expression
5. Clone abundance table construction

Value

A named list with:

clone_expression_rnorm Matrix. Rank-normalized clone-level expression (genes as rows, patient_clone as columns). Ready for `predict_drugs()`.

clone_counts Data frame. Clone abundance per patient. Ready for `predict_patients()`.

cell_clone_map Data frame. Cell-to-clone mapping with columns `cell_id`, `clone_id`, `patient`, `dim_1`, `dim_2`.

clone_viability_df_template Data frame. Template with `patient` and `clone_id` columns, ready to merge with `predict_drugs()` output.

umap_coords Data frame. 2D embedding coordinates per cell (`cell_id`, `dim_1`, `dim_2`). Ready for `plot_tsne_response()`.

reduction_method Character. The method used ("umap" or "tsne").

Examples

```
## Not run:
# List format (same as Rmd)
prepared <- prepare_data(
  expression_matrix = patient_scRNA,
  patient_mapping = cell_names_list,
  genes_to_use = GOI
)

# Or data frame format (from metadata)
metadata <- data.frame(cell_id = colnames(patient_scRNA), patient_id = patient_ids)
prepared <- prepare_data(patient_scRNA, metadata)

# Parse patient ID from cell names (e.g., "P11_M_Barcode" -> "P11")
metadata <- data.frame(Cell = c("P11_M_Barcode1", "P12_M_Barcode2"))
prepared <- prepare_data(
  patient_scRNA, metadata,
  cell_col = "Cell",          # Custom column name
  parse_patient = TRUE,      # Parse from Cell column
)
```

```

    patient_sep = "_",          # Split by "_"
    patient_pos = 1            # Take first element
  )
  # Result: Patient IDs = "P11", "P12"

  # Use directly with prediction functions
  clone_pred <- predict_drugs(models, prepared$clone_expression_rnorm)
  patient_pred <- predict_patients(clone_pred, prepared)

## End(Not run)

```

range01

Change range to 0-1

Description

Scales a numeric vector to the 0-1 range using the 5th and 95th percentiles as thresholds to handle outliers.

Usage

```
range01(x)
```

Arguments

x A numeric vector.

Value

A numeric vector scaled between 0 and 1.

rank_normalization_mat

Rank-normalize each column of a matrix

Description

Converts each column to ranks divided by column length, producing values in the range 0 to 1 (exclusive of 0, inclusive of 1). This is a **crucial preprocessing step** when using your own expression data with PERCEPTIONx.

Usage

```
rank_normalization_mat(mat)
```

Arguments

`mat` A numeric matrix. Rows = genes, columns = cells/samples.

Value

A rank-normalized matrix of the same dimensions, with values between 0 and 1.

Why rank normalization?

PERCEPTIONx models are trained on DepMap expression data that has been rank-normalized. The model coefficients capture the relationship between **relative gene expression ranks** and drug response, not absolute expression values. Therefore, any new expression data fed into the model must undergo the same rank normalization to ensure compatibility.

How it works

For each column (cell/sample), every gene's expression value is replaced by its rank within that column, divided by the total number of genes:

$$x_{ij}^{norm} = \frac{\text{rank}(x_{ij})}{n}$$

where n is the number of rows (genes) and ties are resolved by averaging. This transforms each column into a uniform distribution over $(0, 1]$, making the data robust to batch effects, library size differences, and outliers.

Important

If you provide your own expression data that has **not** been rank-normalized, predictions will be unreliable. Always run `rank_normalization_mat()` on your raw expression matrix before passing it to `predict_drugs()`.

Examples

```
raw_expr <- matrix(c(10, 200, 50, 30, 300, 80), nrow = 3, ncol = 2)
rownames(raw_expr) <- c("GENE_A", "GENE_B", "GENE_C")
norm_expr <- rank_normalization_mat(raw_expr)
```

reset_mirrors	<i>Reset mirrors to default</i>
---------------	---------------------------------

Description

Reset mirrors to default

Usage

```
reset_mirrors()
```

`run_parallel_feature_ranking_bulk`*Parallel feature ranking for multiple drugs*

Description

Runs `feature_ranking_bulk` for a list of drugs.

Usage

```
run_parallel_feature_ranking_bulk(  
  infunc_DrugsToUse,  
  id_cancerType = "PanCan",  
  infunc_exclude_cancer = "PanCan",  
  infunc_GOI,  
  ncores = 4  
)
```

Arguments

<code>infunc_DrugsToUse</code>	Character vector. Drug names to rank features for.
<code>id_cancerType</code>	Character. Cancer type. Default = "PanCan".
<code>infunc_exclude_cancer</code>	Character. Cancer type to exclude. Default = "PanCan".
<code>infunc_GOI</code>	Character vector. Genes of Interest.
<code>ncores</code>	Integer. Kept for API compatibility; ignored (see NOTE above).

Details

NOTE: feature ranking is fully vectorized (~0.15 s per drug over 15k genes), so this deliberately runs **SERIALLY**. The old Windows PSOCK path serialized the 567 MB DepMap into every worker (`clusterExport`) and then still failed — workers are clean R sessions without the package's internal helpers — before silently falling back to serial; that cost ~10x for zero benefit.

Value

A list of feature ranking results, one per drug.

run_perception_app	<i>Launch PERCEPTIONx Shiny Dashboard</i>
--------------------	---

Description

Starts the interactive PERCEPTIONx dashboard for drug response prediction. The dashboard provides a graphical interface for data loading, model training, prediction, and visualization.

Usage

```
run_perception_app()
```

Details

Under a Shiny Server deployment the server executes this function and expects a shiny app *object* back (a nested runApp() would fail), so the function detects Shiny Server via the SHINY_PORT environment variable it sets for every R worker (the same signal the shiny package itself uses internally) and returns the app object instead of starting its own server. This lets the same one-liner work both locally and inside apps/<name>/app.R.

Value

A Shiny app object. When run interactively (no SHINY_PORT environment variable set) the app is also launched in the user's browser.

Examples

```
## Not run:  
library(PERCEPTIONx)  
run_perception_app()  
  
## End(Not run)
```

train_models	<i>Train PERCEPTIONx models for multiple drugs</i>
--------------	--

Description

This function runs the complete PERCEPTIONx training pipeline for a list of drugs. It performs feature ranking, model training with hyperparameter tuning across different k_features values, and selects the best model per drug based on single-cell test performance.

Usage

```
train_models(
  drug_list = NULL,
  cancer_type = "PanCan",
  exclude_cancer = "PanCan",
  GOI = NULL,
  k_features_values = NULL,
  ncores = 4,
  output_dir = "./models",
  model_type = "glmnet",
  num_folds = 3,
  num_tree = 500,
  seed = 1,
  alpha_gradient = 0.05,
  lambda_gradient = 20,
  lambda_range = c(1e-04, 1),
  cv_method = "cv",
  progress_cb = NULL
)
```

Arguments

drug_list	Character vector. Names of drugs to train models for. If NULL, uses the 44 FDA-approved drugs from the paper.
cancer_type	Character. Cancer type for training. Default = "PanCan".
exclude_cancer	Character. Cancer type to exclude from training. Default = "PanCan".
GOI	Character vector. Genes of Interest to use as features.
k_features_values	Numeric vector. Feature counts to test during tuning. If NULL, automatically calculated from expression_rnorm dimensions.
ncores	Integer. Number of CPU cores for parallel feature ranking. Default = 4.
output_dir	Character. Directory to save trained model RDS file. Default = "./models".
model_type	Character. Model type: "glmnet" or "rf". Default = "glmnet".
num_folds	Integer. Number of CV folds. Default = 3.
num_tree	Integer. Number of trees for random forest. Default = 500.
seed	Integer. Random seed for training. Default = 1.
alpha_gradient	Numeric. Step size for alpha grid in glmnet tuning. Default = 0.05.
lambda_gradient	Integer. Number of lambda values in glmnet tuning grid. Default = 20.
lambda_range	Numeric vector of length 2. Min and max lambda for tuning grid. Default = c(0.0001, 1).
cv_method	Character. Cross-validation method. Default = "cv".
progress_cb	Function. Optional progress callback called as progress_cb(phase, i, n, drug) where phase is "rank", "train", or "done". Useful for progress bars in interactive sessions.

Value

A named list of trained model objects, one per drug. Also saved as a single RDS file.

Examples

```
## Not run:
load_depmap(read = TRUE)
available_genes <- intersect(rownames(DepMap$expression_20Q4),
                             rownames(DepMap$scRNA_complete))

set.seed(123)
GOI_100 <- sample(available_genes, 100)
train_models("abemaciclib", "PanCan", "PanCan", GOI_100, ncores = 1)

## End(Not run)
```

zscore_viability	<i>Z-score scale viability values across patients</i>
------------------	---

Description

Applies base R scale() to each drug column across all patients, centering to mean 0 and standard deviation 1. This ensures comparability of predicted viability scores between different drugs before patient-level aggregation. Matches the original PERCEPTION pipeline.

Usage

```
zscore_viability(clone_viability_df, cols = NULL)
```

Arguments

clone_viability_df	Data frame. Must have columns 'patient' and 'clone_id', plus one or more drug columns with predicted viability values.
cols	Character vector of column names to scale. Default NULL scales every drug column. Use this to leave already-standardized columns (e.g. a pre-combined comb_viability) untouched.

Value

A data frame with the same structure, but drug columns z-score scaled.

Index

add_mirrors, 3
annotate_clones, 3, 36

build_clone_counts, 5
build_clone_key, 5
build_clone_key(), 20
build_on_BULK_v2, 6

clone_mean_expression, 7
compare_performance, 8
cor.test, 9
cor.test_trimmed_v0, 9

each_patient_pseudo_bulk, 9
export_plot_cairo, 10

fdrcorr, 11
feature_ranking_bulk, 11

get_cellLine_list, 12
get_depmap, 13
get_mirrors, 14
get_performance, 14
get_response_matrix, 15
get_significant_models, 15

hypergeometric_test_for_twolists, 16

list_mirrors, 17
load_depmap, 17
load_model, 18

p.adjust, 11
parse_clone_keys, 20
parse_clone_keys(), 5
plot_clone_distribution, 20
plot_clone_umap, 21
plot_clone_viability, 22
plot_model_performance, 24
plot_model_roc, 25
plot_patient_response_panel, 26

plot_response_boxplot, 27
plot_roc_curve, 28
plot_seurat_clustering, 29
plot_tsne_biomarker_viability, 30
plot_tsne_response, 31
predict_drugs, 33, 39
predict_patients, 34
prepare_data, 35

range01, 38
rank_normalization_mat, 38
reset_mirrors, 39
run_parallel_feature_ranking_bulk, 40
run_perception_app, 41

train_models, 41

zscore_viability, 43