

# Package: detectPanel (via r-universe)

September 16, 2026

**Type** Package

**Title** Leakage-Aware Discovery of Small Biomarker Panels

**Version** 0.1.3

**Author** Fuhao Jiang [aut, cre]

**Maintainer** Fuhao Jiang <emr39515@gmail.com>

**Description** Discovers small binary-classification biomarker panels from count or expression matrices while prioritizing detectability, expression stability, and univariate discrimination. Candidate filtering and panel selection can be repeated inside nested cross-validation to reduce information leakage. The package provides shared resampling splits, exhaustive small-panel search, logistic model fitting with an automatic ridge fallback for unstable separation-prone fits, out-of-fold evaluation, selection-frequency summaries, and optional 'DESeq2' differential-expression support. The nested model-selection workflow follows Varma and Simon (2006) <doi:10.1186/1471-2105-7-91>, and the optional differential-expression analysis uses Love, Huber, and Anders (2014) <doi:10.1186/s13059-014-0550-8>.

**License** MIT + file LICENSE

**URL** <https://github.com/Emr-27/detectPanel>

**BugReports** <https://github.com/Emr-27/detectPanel/issues>

**Encoding** UTF-8

**Depends** R (>= 4.2.0)

**Imports** ggplot2, graphics, stats, utils

**Suggests** DESeq2, knitr, rmarkdown, pheatmap, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Repository** <https://emr-27.r-universe.dev>

**Date/Publication** 2026-09-05 18:47:56 UTC  
**RemoteUrl** <https://github.com/emr-27/detectpanel>  
**RemoteRef** HEAD  
**RemoteSha** 7cf33fb2df99493f19edbf58470db72bc22882da

**Contents**

|                                           |           |
|-------------------------------------------|-----------|
| detectPanel-package . . . . .             | 2         |
| confusion_metrics . . . . .               | 3         |
| detectPanel . . . . .                     | 3         |
| export_results . . . . .                  | 5         |
| fit_panel . . . . .                       | 6         |
| flag_pca_outliers . . . . .               | 7         |
| make_repeated_stratified_splits . . . . . | 8         |
| nested_validate_panels . . . . .          | 9         |
| plot_marker_expression . . . . .          | 10        |
| predict.detectPanel_result . . . . .      | 11        |
| run_deseq2 . . . . .                      | 12        |
| score_detectability . . . . .             | 13        |
| validate_assay_data . . . . .             | 14        |
| <b>Index</b>                              | <b>16</b> |

---

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| detectPanel-package | <i>Leakage-Aware Discovery of Small Biomarker Panels</i> |
|---------------------|----------------------------------------------------------|

---

**Description**

Tools for detectability-aware candidate scoring, shared-split panel search, nested cross-validation, logistic model fitting with a separation-safe ridge fallback, prediction, plotting, and result export.

**Details**

The recommended entry point is `discover_panel()`. Candidate selection and panel search are repeated inside every outer training split.

**Author(s)**

Fuhao Jiang <[emr39515@gmail.com](mailto:emr39515@gmail.com)>

---

|                   |                                         |
|-------------------|-----------------------------------------|
| confusion_metrics | <i>Compute Confusion-Matrix Metrics</i> |
|-------------------|-----------------------------------------|

---

**Description**

Computes fixed-threshold classification metrics without optimizing on the evaluation data.

**Usage**

```
confusion_metrics(response, probability, threshold = 0.5)
```

**Arguments**

|             |                                       |
|-------------|---------------------------------------|
| response    | Non-missing binary 0/1 outcome.       |
| probability | Predicted probability for class 1.    |
| threshold   | Probability classification threshold. |

**Value**

A one-row data frame with AUC, sensitivity, specificity, accuracy, balanced accuracy, and confusion counts.

**Examples**

```
response <- c(0, 0, 1, 1)
probability <- c(0.1, 0.4, 0.7, 0.9)
confusion_metrics(response, probability, threshold = 0.5)
```

---

|             |                                                        |
|-------------|--------------------------------------------------------|
| detectPanel | <i>Discover, Validate, and Refit a Biomarker Panel</i> |
|-------------|--------------------------------------------------------|

---

**Description**

Runs nested validation and refits the most frequently selected exact panel on all samples.

**Usage**

```
detectPanel(...)

discover_panel(
  counts, metadata, outcome, positive = NULL, sample_id = NULL,
  exclude = character(), exclude_regex = NULL, panel_size = 3L,
  candidate_n = 10L, transform_method = c("log2_cpm", "log2", "none"),
  prior_count = 1, detection_threshold = 10,
  detectability_weights = c(0.5, 0.35, 0.15),
```

```

preselection_weights = c(0.5, 0.5), min_mean = 100,
min_median = 20, min_detection = 0.8, min_group_detection = 0.5,
min_auc = 0.8, allow_fallback = FALSE, outer_v = 5L,
outer_repeats = 2L, inner_v = 5L, inner_repeats = 3L,
near_best_tolerance = 0.01, seed = 123L, max_combinations = 50000L
)

```

## Arguments

|                                                                   |                                                          |
|-------------------------------------------------------------------|----------------------------------------------------------|
| counts                                                            | Non-negative feature-by-sample matrix.                   |
| metadata                                                          | Sample metadata data frame.                              |
| outcome                                                           | Name of the binary outcome column.                       |
| positive                                                          | Positive-class label.                                    |
| sample_id                                                         | Optional sample-ID column in metadata.                   |
| exclude                                                           | Exact feature names to remove.                           |
| exclude_regex                                                     | Optional feature-name regular expression to remove.      |
| panel_size                                                        | Number of features per panel.                            |
| candidate_n                                                       | Maximum candidate-pool size inside each outer split.     |
| transform_method                                                  | Modeling transformation.                                 |
| prior_count                                                       | Offset used before log transformation.                   |
| detection_threshold                                               | Count threshold defining detection.                      |
| detectability_weights                                             | Weights for abundance, detection, and stability.         |
| preselection_weights                                              | Weights for detectability and individual AUC.            |
| min_mean, min_median, min_detection, min_group_detection, min_auc | Training-only candidate filters.                         |
| allow_fallback                                                    | Whether to fill undersized candidate pools by rank.      |
| outer_v, outer_repeats                                            | Outer validation settings.                               |
| inner_v, inner_repeats                                            | Inner panel-search settings.                             |
| near_best_tolerance                                               | AUC tolerance before tie breaking.                       |
| seed                                                              | Random seed.                                             |
| max_combinations                                                  | Maximum combinations per inner search.                   |
| ...                                                               | Arguments passed from detectPanel() to discover_panel(). |

## Value

A detectPanel\_result object containing nested validation and a final fitted model.

**Examples**

```

set.seed(1)
y <- rep(c("Control", "Case"), each = 12)
x <- matrix(rpois(10 * 24, 50), 10,
            dimnames = list(paste0("m", 1:10), paste0("s", 1:24)))
x[1:3, y == "Case"] <- x[1:3, y == "Case"] + 50
meta <- data.frame(group = y, row.names = colnames(x))
fit <- detectPanel(x, meta, "group", positive = "Case",
                  candidate_n = 6, min_mean = 5, min_median = 2,
                  min_detection = 0.2, min_group_detection = 0.1, min_auc = 0.55,
                  outer_v = 3, outer_repeats = 1, inner_v = 3, inner_repeats = 1)
fit$final_panel

```

export\_results

*Export detectPanel Results***Description**

Writes nested metrics, predictions, selection frequencies, audits, and optional final-model files.

**Usage**

```
export_results(x, path, save_model = TRUE)
```

**Arguments**

|            |                                                    |
|------------|----------------------------------------------------|
| x          | A detectPanel_result or detectPanel_nested object. |
| path       | Output directory.                                  |
| save_model | Whether to save an RDS object.                     |

**Value**

The normalized path, invisibly.

**Examples**

```

set.seed(6)
group <- rep(c("Control", "Case"), each = 10)
counts <- matrix(rpois(6 * 20, 40), 6,
                 dimnames = list(paste0("gene", 1:6), paste0("s", 1:20)))
counts[1:2, group == "Case"] <- counts[1:2, group == "Case"] + 35
metadata <- data.frame(group = group, row.names = colnames(counts))
fit <- discover_panel(counts, metadata, "group", positive = "Case",
                     panel_size = 2, candidate_n = 4, min_mean = 5, min_median = 5,
                     min_detection = 0.5, min_group_detection = 0.5, min_auc = 0.55,
                     outer_v = 2, outer_repeats = 1, inner_v = 2, inner_repeats = 1)
export_results(fit, file.path(tempdir(), "detectPanel-example"),
              save_model = FALSE)

```

fit\_panel

*Fit and Predict with a Biomarker Panel***Description**

Stores preprocessing and scaling parameters needed to predict new samples. Ordinary logistic regression is attempted first. If it is non-convergent or shows separation-like numerical behavior, an internal L2-penalized logistic fit is used as a deterministic fallback.

**Usage**

```
fit_panel(assay, outcome, features, positive = NULL,
          transform_method = c("log2_cpm", "log2", "none"), prior_count = 1,
          threshold_method = c("youden", "fixed"), fixed_threshold = 0.5)

## S3 method for class 'detectPanel_fit'
predict(object, newdata,
        type = c("response", "class", "link"), ...)
```

**Arguments**

|                  |                                               |
|------------------|-----------------------------------------------|
| assay            | Feature-by-sample count or expression matrix. |
| outcome          | Binary outcome aligned to columns.            |
| features         | Features included in the logistic model.      |
| positive         | Positive-class label.                         |
| transform_method | Input transformation.                         |
| prior_count      | Log offset.                                   |
| threshold_method | Training threshold method.                    |
| fixed_threshold  | Fixed probability threshold.                  |
| object           | A fitted detectPanel_fit object.              |
| newdata          | Feature-by-new-sample matrix.                 |
| type             | Prediction scale.                             |
| ...              | Additional arguments, currently unused.       |

**Value**

A fitted model object or prediction vector. The fitted object records `fit_method` and `ridge_lambda` when the internal ridge fallback is used.

**Examples**

```

set.seed(4)
outcome <- rep(c("Control", "Case"), each = 8)
assay <- matrix(rpois(4 * 16, 40), 4,
  dimnames = list(paste0("gene", 1:4), paste0("s", 1:16)))
assay[1:2, outcome == "Case"] <- assay[1:2, outcome == "Case"] + 25
fit <- fit_panel(assay, outcome, c("gene1", "gene2"), positive = "Case")
predict(fit, assay[, 1:3], type = "response")

```

---

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| flag_pca_outliers | <i>Flag Potential PCA Outliers Without Deleting Samples</i> |
|-------------------|-------------------------------------------------------------|

---

**Description**

Provides non-destructive PCA quality-control flags for review alongside independent QC evidence.

**Usage**

```

flag_pca_outliers(expression, group = NULL, n_components = 5L,
  threshold = NULL)

```

**Arguments**

|              |                                              |
|--------------|----------------------------------------------|
| expression   | Transformed feature-by-sample matrix.        |
| group        | Optional sample group vector.                |
| n_components | Maximum number of principal components used. |
| threshold    | Optional positive robust-distance threshold. |

**Value**

A sample-level data frame with PCA coordinates, robust distance, and a flag.

**Examples**

```

set.seed(7)
expression <- matrix(rnorm(5 * 10), 5,
  dimnames = list(paste0("gene", 1:5), paste0("s", 1:10)))
flag_pca_outliers(expression, group = rep(c("Control", "Case"), each = 5),
  n_components = 2, threshold = 4)

```

---

make\_repeated\_stratified\_splits

*Shared Resampling Splits and Exhaustive Panel Search*


---

## Description

All panels are evaluated on the same folds to permit fair paired comparison. A repeated-CV AUC is retained only when every sample in that repeat receives a finite out-of-fold prediction; failed folds are not silently dropped.

## Usage

```
make_repeated_stratified_splits(outcome, v = 5L, repeats = 10L, seed = 123L)
```

```
search_panels(expression, outcome, candidates, panel_size = 3L,
  splits = NULL, candidate_table = NULL, v = 5L, repeats = 10L,
  seed = 123L, near_best_tolerance = 0.01, max_combinations = 50000L)
```

## Arguments

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| outcome             | Binary vector.                                          |
| v                   | Number of folds.                                        |
| repeats             | Number of repeated fold assignments.                    |
| seed                | Random seed.                                            |
| expression          | Transformed feature-by-sample matrix.                   |
| candidates          | Candidate feature names.                                |
| panel_size          | Number of features per panel.                           |
| splits              | Reusable splits from make_repeated_stratified_splits(). |
| candidate_table     | Optional candidate scoring table.                       |
| near_best_tolerance | AUC tolerance before expression-aware tie breaking.     |
| max_combinations    | Safety limit for exhaustive search.                     |

## Value

A list of splits or a detectPanel\_search object.

**Examples**

```
set.seed(3)
outcome <- rep(0:1, each = 8)
expression <- matrix(rnorm(4 * 16), 4,
  dimnames = list(paste0("gene", 1:4), paste0("s", 1:16)))
expression[1:2, outcome == 1] <- expression[1:2, outcome == 1] + 1.5
splits <- make_repeated_stratified_splits(outcome, v = 2, repeats = 1)
search_panels(expression, outcome, rownames(expression), panel_size = 2,
  splits = splits)
```

---

 nested\_validate\_panels

*Nested Validation of Biomarker Panel Discovery*


---

**Description**

Repeats candidate selection and panel search within every outer training split. If every outer split fails, the error message includes each split identifier and its original stage-specific failure message.

**Usage**

```
nested_validate_panels(counts, metadata, outcome, positive = NULL,
  sample_id = NULL, exclude = character(), exclude_regex = NULL,
  panel_size = 3L, candidate_n = 10L,
  transform_method = c("log2_cpm", "log2", "none"), prior_count = 1,
  detection_threshold = 10, detectability_weights = c(0.5, 0.35, 0.15),
  preselection_weights = c(0.5, 0.5), min_mean = 100,
  min_median = 20, min_detection = 0.8, min_group_detection = 0.5,
  min_auc = 0.8, allow_fallback = FALSE, outer_v = 5L,
  outer_repeats = 2L, inner_v = 5L, inner_repeats = 3L,
  near_best_tolerance = 0.01, seed = 123L, max_combinations = 50000L)
```

**Arguments**

counts, metadata, outcome, positive, sample\_id, exclude, exclude\_regex  
 See discover\_panel().

panel\_size, candidate\_n, transform\_method, prior\_count  
 Model and search settings.

detection\_threshold, detectability\_weights, preselection\_weights  
 Marker-scoring settings.

min\_mean, min\_median, min\_detection, min\_group\_detection, min\_auc  
 Training-only candidate filters.

allow\_fallback Whether to fill undersized candidate pools by rank.

outer\_v, outer\_repeats, inner\_v, inner\_repeats  
 Nested resampling settings.

near\_best\_tolerance, seed, max\_combinations  
 Search controls.

**Value**

A detectPanel\_nested object with predictions, metrics, selection frequencies, audits, and errors.

**Examples**

```
set.seed(5)
group <- rep(c("Control", "Case"), each = 10)
counts <- matrix(rpois(6 * 20, 40), 6,
  dimnames = list(paste0("gene", 1:6), paste0("s", 1:20)))
counts[1:2, group == "Case"] <- counts[1:2, group == "Case"] + 35
metadata <- data.frame(group = group, row.names = colnames(counts))
nested_validate_panels(counts, metadata, "group", positive = "Case",
  panel_size = 2, candidate_n = 4, min_mean = 5, min_median = 5,
  min_detection = 0.5, min_group_detection = 0.5, min_auc = 0.55,
  outer_v = 2, outer_repeats = 1, inner_v = 2, inner_repeats = 1)
```

---

plot\_marker\_expression

*Diagnostic Plotting Helpers*

---

**Description**

Creates expression, ROC, volcano, and optional heatmap views from package results.

**Usage**

```
plot_marker_expression(expression, metadata, features, outcome,
  sample_id = NULL)

plot_marker_roc(expression, outcome, features, positive = NULL)

plot_volcano(results, feature_col = "feature",
  lfc_col = "log2FoldChange", padj_col = "padj",
  padj_cut = 0.05, lfc_cut = 1)

plot_panel_heatmap(expression, features, metadata = NULL,
  annotation_cols = NULL, scale = "row", ...)
```

**Arguments**

|            |                                                                |
|------------|----------------------------------------------------------------|
| expression | Transformed feature-by-sample matrix.                          |
| metadata   | Sample metadata aligned by row names.                          |
| features   | Features to display.                                           |
| outcome    | Outcome column name or a binary vector, depending on function. |
| sample_id  | Optional metadata sample-ID column.                            |
| positive   | Positive-class label.                                          |

results            Differential-expression result data frame.  
 feature\_col, lfc\_col, padj\_col  
                   Differential-expression column names.  
 padj\_cut, lfc\_cut  
                   Volcano significance cutoffs.  
 annotation\_cols  
                   Metadata columns used as heatmap annotations.  
 scale             Heatmap scaling option.  
 ...                Additional arguments passed to `pheatmap::pheatmap()`.

### Value

A ggplot object or the object returned by `pheatmap::pheatmap()`.

### Examples

```

set.seed(8)
group <- rep(c("Control", "Case"), each = 5)
expression <- matrix(rnorm(3 * 10), 3,
  dimnames = list(paste0("gene", 1:3), paste0("s", 1:10)))
expression[1, group == "Case"] <- expression[1, group == "Case"] + 2
metadata <- data.frame(group = group, row.names = colnames(expression))
plot_marker_expression(expression, metadata, "gene1", "group")
plot_marker_roc(expression, group, "gene1", positive = "Case")
de_results <- data.frame(feature = rownames(expression),
  log2FoldChange = c(2, -1.5, 0.2), padj = c(0.01, 0.03, 0.8))
plot_volcano(de_results)
if (requireNamespace("pheatmap", quietly = TRUE)) {
  plot_panel_heatmap(expression, c("gene1", "gene2"), metadata,
    annotation_cols = "group", silent = TRUE)
}

```

---

predict.detectPanel\_result

*Predict samples using a fitted detectPanel result*

---

### Description

Applies the final fitted panel model from a `detectPanel_result` object to new count data.

### Usage

```

## S3 method for class 'detectPanel_result'
predict(object, newdata,
  type = c("response", "class"), ...)

```

**Arguments**

|         |                                                                          |
|---------|--------------------------------------------------------------------------|
| object  | A detectPanel_result object returned by <a href="#">discover_panel</a> . |
| newdata | A feature-by-sample numeric matrix.                                      |
| type    | Prediction output type: probabilities or classes.                        |
| ...     | Additional arguments.                                                    |

**Value**

A vector of predicted probabilities or classes.

**Examples**

```
set.seed(9)
group <- rep(c("Control", "Case"), each = 10)
counts <- matrix(rpois(6 * 20, 40), 6,
  dimnames = list(paste0("gene", 1:6), paste0("s", 1:20)))
counts[1:2, group == "Case"] <- counts[1:2, group == "Case"] + 35
metadata <- data.frame(group = group, row.names = colnames(counts))
fit <- discover_panel(counts, metadata, "group", positive = "Case",
  panel_size = 2, candidate_n = 4, min_mean = 5, min_median = 5,
  min_detection = 0.5, min_group_detection = 0.5, min_auc = 0.55,
  outer_v = 2, outer_repeats = 1, inner_v = 2, inner_repeats = 1)
predict(fit, counts[, 1:3], type = "response")
```

---

run\_deseq2

---

*Optional DESeq2 Differential-Expression Adapter*


---

**Description**

Produces differential-expression evidence separately from predictive validation.

**Usage**

```
run_deseq2(counts, metadata, design, contrast, min_total_count = 10, ...)
```

**Arguments**

|                 |                                                                 |
|-----------------|-----------------------------------------------------------------|
| counts          | Feature-by-sample count matrix.                                 |
| metadata        | Sample metadata aligned by row names.                           |
| design          | Design formula.                                                 |
| contrast        | Contrast passed to <code>DESeq2::results()</code> .             |
| min_total_count | Minimum total feature count.                                    |
| ...             | Additional arguments passed to <code>DESeq2::results()</code> . |

**Value**

A list containing a fitted DESeq2 object and result data frame.

**Examples**

```
if (requireNamespace("DESeq2", quietly = TRUE)) {
  set.seed(10)
  ngenes <- 300L
  nsamples <- 8L
  base_mean <- exp(seq(log(10), log(1000), length.out = ngenes))
  counts <- matrix(
    rbinom(ngenes * nsamples, mu = rep(base_mean, nsamples), size = 10),
    nrow = ngenes,
    dimnames = list(paste0("gene", seq_len(ngenes)),
                     paste0("s", seq_len(nsamples))))
  metadata <- data.frame(
    condition = factor(rep(c("Control", "Case"), each = nsamples / 2),
                      levels = c("Control", "Case")),
    row.names = colnames(counts))
  result <- run_deseq2(counts, metadata, ~ condition,
    c("condition", "Case", "Control"))
  head(result$results)
}
```

---

|                     |                                                 |
|---------------------|-------------------------------------------------|
| score_detectability | <i>Score and Preselect Candidate Biomarkers</i> |
|---------------------|-------------------------------------------------|

---

**Description**

Combines abundance, detection, robust stability, and fixed-direction rank AUC.

**Usage**

```
score_detectability(counts, group = NULL, detection_threshold = 10,
  weights = c(abundance = 0.5, detection = 0.35, stability = 0.15),
  stability_transform = c("log2", "none"))

evaluate_markers(expression, outcome, positive = NULL)

preselect_markers(detectability, marker_metrics, min_mean = 100,
  min_median = 20, min_detection = 0.8, min_group_detection = 0.5,
  min_auc = 0.8, weights = c(detectability = 0.5, auc = 0.5),
  fallback_n = 0L)
```

**Arguments**

|                                                                   |                                                                                |
|-------------------------------------------------------------------|--------------------------------------------------------------------------------|
| counts                                                            | Non-negative feature-by-sample count matrix.                                   |
| group                                                             | Optional grouping vector.                                                      |
| detection_threshold                                               | Value defining detection.                                                      |
| weights                                                           | Non-negative weights summing to one.                                           |
| stability_transform                                               | Transformation before robust dispersion scoring.                               |
| expression                                                        | Transformed feature-by-sample matrix.                                          |
| outcome                                                           | Binary outcome aligned to columns.                                             |
| positive                                                          | Positive-class label.                                                          |
| detectability                                                     | Output from <code>score_detectability()</code> .                               |
| marker_metrics                                                    | Output from <code>evaluate_markers()</code> .                                  |
| min_mean, min_median, min_detection, min_group_detection, min_auc | Candidate filters.                                                             |
| fallback_n                                                        | Optional number of ranked markers returned when hard filters are insufficient. |

**Value**

A feature-level data frame.

**Examples**

```
set.seed(2)
group <- rep(c("Control", "Case"), each = 6)
counts <- matrix(rpois(5 * 12, 30), 5,
  dimnames = list(paste0("gene", 1:5), paste0("s", 1:12)))
counts[1:2, group == "Case"] <- counts[1:2, group == "Case"] + 30
detectability <- score_detectability(counts, group, detection_threshold = 5)
marker_metrics <- evaluate_markers(log2(counts + 1), group, positive = "Case")
preselect_markers(detectability, marker_metrics, min_mean = 5,
  min_median = 5, min_detection = 0.5, min_group_detection = 0.5,
  min_auc = 0.6)
```

---

|                     |                                                   |
|---------------------|---------------------------------------------------|
| validate_assay_data | <i>Validate, Filter, and Transform Assay Data</i> |
|---------------------|---------------------------------------------------|

---

**Description**

Core utilities for reproducible assay preparation without hard-coded dataset rules.

**Usage**

```
validate_assay_data(assay, metadata, outcome, positive = NULL, sample_id = NULL)
exclude_features(assay, features = character(), regex = NULL, ignore_case = TRUE)
transform_assay(assay, method = c("log2_cpm", "log2", "none"), prior_count = 1)
```

**Arguments**

|             |                                                   |
|-------------|---------------------------------------------------|
| assay       | Numeric feature-by-sample matrix.                 |
| metadata    | Sample metadata.                                  |
| outcome     | Binary outcome column name.                       |
| positive    | Positive-class label.                             |
| sample_id   | Optional metadata sample-ID column.               |
| features    | Exact feature names to remove.                    |
| regex       | Optional feature-name regular expression.         |
| ignore_case | Whether regular-expression matching ignores case. |
| method      | Transformation method.                            |
| prior_count | Positive log offset.                              |

**Value**

Aligned data, a filtering audit, or a transformed matrix, depending on the function.

**Examples**

```
assay <- matrix(1:24, nrow = 3,
  dimnames = list(c("gene1", "gene2", "control_gene"), paste0("s", 1:8)))
metadata <- data.frame(group = rep(c("Control", "Case"), each = 4),
  row.names = colnames(assay))
validated <- validate_assay_data(assay, metadata, "group", positive = "Case")
filtered <- exclude_features(validated$assay, regex = "control")
transform_assay(filtered$assay, method = "log2")
```

# Index

- \* **package**
  - detectPanel-package, [2](#)
- \* **prediction**
  - predict.detectPanel\_result, [11](#)
- confusion\_metrics, [3](#)
- detectPanel, [3](#)
- detectPanel-package, [2](#)
- discover\_panel, [12](#)
- discover\_panel (detectPanel), [3](#)
- evaluate\_markers (score\_detectability),  
[13](#)
- exclude\_features (validate\_assay\_data),  
[14](#)
- export\_results, [5](#)
- fit\_panel, [6](#)
- flag\_pca\_outliers, [7](#)
- make\_repeated\_stratified\_splits, [8](#)
- nested\_validate\_panels, [9](#)
- plot.detectPanel\_nested  
(nested\_validate\_panels), [9](#)
- plot.detectPanel\_result (detectPanel), [3](#)
- plot\_marker\_expression, [10](#)
- plot\_marker\_roc  
(plot\_marker\_expression), [10](#)
- plot\_panel\_heatmap  
(plot\_marker\_expression), [10](#)
- plot\_volcano (plot\_marker\_expression),  
[10](#)
- predict.detectPanel\_fit (fit\_panel), [6](#)
- predict.detectPanel\_result, [11](#)
- preselect\_markers  
(score\_detectability), [13](#)
- print.detectPanel\_fit (fit\_panel), [6](#)
- print.detectPanel\_nested  
(nested\_validate\_panels), [9](#)
- print.detectPanel\_result (detectPanel),  
[3](#)
- print.detectPanel\_search  
(make\_repeated\_stratified\_splits),  
[8](#)
- run\_deseq2, [12](#)
- score\_detectability, [13](#)
- search\_panels  
(make\_repeated\_stratified\_splits),  
[8](#)
- summary.detectPanel\_fit (fit\_panel), [6](#)
- summary.detectPanel\_nested  
(nested\_validate\_panels), [9](#)
- summary.detectPanel\_result  
(detectPanel), [3](#)
- transform\_assay (validate\_assay\_data),  
[14](#)
- validate\_assay\_data, [14](#)