# Package 'filtro'

May 8, 2026

**Title** Feature Selection Using Supervised Filter-Based Methods

**Version** 0.2.0

**Description** Tidy tools to apply filter-based supervised feature selection methods. These methods score and rank feature relevance using metrics such as p-values, correlation, and importance scores (Kuhn and Johnson (2019) <doi:10.1201/9781315108230>).

**License** MIT + file LICENSE

**URL** https://github.com/tidymodels/filtro, https://filtro.tidymodels.org/

**BugReports** https://github.com/tidymodels/filtro/issues

**Depends** R (>= 4.1)

**Imports** cli, desirability2 (>= 0.1.0), dplyr, generics, pROC, purrr, rlang (>= 1.1.0), S7, stats, tibble, tidyr, vctrs

**Suggests** aorsf, FSelectorRcpp, knitr, modeldata, partykit, quarto, ranger, rmarkdown, testthat (>= 3.0.0), titanic

**Config/Needs/website** tidyverse/tidytemplate

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Collate** 'aaa.R' 'class_score.R' 'data.R' 'desirability2.R' 'filtro-package.R' 'import-standalone-obj-type.R' 'import-standalone-types-check.R' 'misc.R' 'score-aov.R' 'utilities.R' 'score-cor.R' 'score-cross_tab.R' 'score-forest_imp.R' 'score-info_gain.R' 'score-roc_auc.R' 'zzz.R'

**LazyData** true

**VignetteBuilder** quarto, knitr

**NeedsCompilation** no

**Author** Frances Lin [aut, cre],
Max Kuhn [aut] (ORCID: <https://orcid.org/0000-0003-2402-136X>),
Emil Hvitfeldt [aut],
Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

**Maintainer** Frances Lin <franceslinyc@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-08-26 21:40:02 UTC

# Contents



---

`ames_scores_results` *Ames exampled score results*

---

## Description

This data an ames exampled score results for, used for `show_best_desirability_prop()` as a demonstration, and created by examples in `fill_safe_values()`.

## Value

`ames_scores_results`
a tibble

## Examples

```
data(ames_scores_results)
```

---

`arrange_score` *Arrange score*

---

## Description

Arrange score

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `...` | Further arguments passed to or from other methods. |
| `target` | A numeric value specifying the target value. The default of `NULL` indicates that there is no target value. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Arrange score
ames_aov_pval_res |> arrange_score()
```

`bind_scores` *Bind score class object, including their associated metadata and scores*

## Description

Binds multiple score class objects (e.g., `score_*`), including their associated metadata and scores. See `fill_safe_values()` for binding with safe-value handling.

## Arguments

| | |
|---|---|
| `x` | A list. |
| `...` | Further arguments passed to or from other methods. |

## Value

A tibble of scores results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

# ANOVA p-value
ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Pearson correlation
ames_cor_pearson_res <-
  score_cor_pearson |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_cor_pearson_res@results

# Forest importance
set.seed(42)
```

```
ames_imp_rf_reg_res <-
  score_imp_rf |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_imp_rf_reg_res@results

# Information gain
ames_info_gain_reg_res <-
  score_info_gain |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_info_gain_reg_res@results

# Create a list
class_score_list <- list(
  ames_aov_pval_res,
  ames_cor_pearson_res,
  ames_imp_rf_reg_res,
  ames_info_gain_reg_res
)

# Bind scores
class_score_list |> bind_scores()
```

---

`class_score_list` *S7 subclass of base R's* `list` *for method dispatch*

---

## Description

`class_score_list` is an S7 subclass of S3 base R's `list`, used for method dispatch in `bind_scores()` and `fill_safe_values()`.

## Usage

```
class_score_list
```

## Format

An object of class `S7_S3_class` of length 3.

## Value

A list of S7 objects.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
```

```
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

# ANOVA p-value
ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Pearson correlation
ames_cor_pearson_res <-
  score_cor_pearson |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_cor_pearson_res@results

# Create a list
class_score_list <- list(
  ames_aov_pval_res,
  ames_cor_pearson_res
)
```

---

`dont_log_pvalues` *Disable -log10 transformation of p-values*

---

## Description

Disable -log10 transformation of p-values

## Usage

```
dont_log_pvalues(x)
```

## Arguments

| | |
|---|---|
| x | A score class object. |

## Value

The modified score class object with `neg_log10` set to `FALSE`.

`fill_safe_value` *Fill safe value* (singular)

## Description

Fills in safe value for missing score, with an option to apply transformation. This is a *singular* scoring method. See `fill_safe_values()` for *plural* scoring method.

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `return_results` | A logical value indicating whether to return results. |

## Details

If `transform = TRUE`, by default, all score objects use the identity transformation, except the correlation score object, which uses the absolute transformation.

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Fill safe value
ames_aov_pval_res |> fill_safe_value(return_results = TRUE)

# Fill safe value, option to transform
ames_aov_pval_res |> fill_safe_value(return_results = TRUE, transform = TRUE)
```

`fill_safe_values` *Fill safe values* (plural)

## Description

Wraps `bind_scores()`, and fills in safe values for missing scores, with an option to apply transformation. This is a *plural* scoring method. See `fill_safe_value()` for *singular* scoring method.

## Arguments

| | |
|---|---|
| `x` | A list. |
| `...` | Further arguments passed to or from other methods. |

## Details

If `transform = TRUE`, by default, all score objects use the identity transformation, except the correlation score object, which uses the absolute transformation.

## Value

A tibble of scores results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

# ANOVA p-value
ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Pearson correlation
ames_cor_pearson_res <-
  score_cor_pearson |>
  fit(Sale_Price ~ ., data = ames_subset)
```

```
ames_cor_pearson_res@results

# Forest importance
set.seed(42)
ames_imp_rf_reg_res <-
  score_imp_rf |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_imp_rf_reg_res@results

# Information gain
ames_info_gain_reg_res <-
  score_info_gain |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_info_gain_reg_res@results

# Create a list
class_score_list <- list(
  ames_aov_pval_res,
  ames_cor_pearson_res,
  ames_imp_rf_reg_res,
  ames_info_gain_reg_res
)

# Fill safe values
class_score_list |> fill_safe_values()

# Fill safe value, option to transform
class_score_list |> fill_safe_values(transform = TRUE)
```

---

`rank_best_score_dense` *Rank score based on* `dplyr::dense_rank()`*, where tied values receive the same rank and ranks are without gaps* (singular)

---

## Description

Rank score based on `dplyr::dense_rank()`, where tied values receive the same rank and ranks are without gaps *(singular)*

## Usage

```
rank_best_score_dense(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `...` | Further arguments passed to or from other methods. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Rank score
ames_aov_pval_res |> rank_best_score_dense()
```

---

# rank_best_score_min *Rank score based on* dplyr::min_rank(), *where tied values receive the same rank and ranks are with gaps* (singular)

---

## Description

Rank score based on dplyr::min_rank(), where tied values receive the same rank and ranks are with gaps *(singular)*

## Usage

```
rank_best_score_min(x, ...)
```

## Arguments

| | |
|---|---|
| x | A score class object (e.g., score_cor_pearson). |
| ... | Further arguments passed to or from other methods. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Rank score
ames_aov_pval_res |> rank_best_score_min()
```

---

# score_aov_pval *Scoring via analysis of variance hypothesis tests*

---

## Description

These two objects can be used to compute importance scores based on Analysis of Variance techniques.

## Usage

```
score_aov_pval

score_aov_fstat
```

## Format

An object of class `filtro::class_score_aov` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_aov` (inherits from `filtro::class_score`, `S7_object`) of length 1.

## Details

These objects are used when either:

- The predictors are numeric and the outcome is a factor/category, or
- The predictors are factors and the outcome is numeric.

In either case, a linear model (via `stats::lm()`) is created with the proper variable roles, and the overall p-value for the hypothesis that all means are equal is computed via the standard F-statistic. The p-value that is returned is transformed to be `-log10(p_value)` so that larger values are associated with more important predictors.

**Estimating the scores:**

In **filtro**, the `score_*` objects define a scoring method (e.g., data input requirements, package dependencies, etc). To compute the scores for a specific data set, the `fit()` method is used. The main arguments for these functions are:

`object` A score class object (e.g., `score_aov_pval`).

`formula` A standard R formula with a single outcome on the right-hand side and one or more predictors (or `.`) on the left-hand side. The data are processed via `stats::model.frame()`

`data` A data frame containing the relevant columns defined by the formula.

`...` Further arguments passed to or from other methods.

`case_weights` A quantitative vector of case weights that is the same length as the number of rows in `data`. The default of `NULL` indicates that there are no case weights.

Missing values are removed for each predictor/outcome combination being scored.

In cases where the underlying computations fail, the scoring proceeds silently, and a missing value is given for the score.

## Value

An S7 object. The primary property of interest is in `results`. This is a data frame of results that is populated by the `fit()` method and has columns:

- `name`: The name of the score (e.g., `aov_fstat` or `aov_pval`).
- `score`: The estimates for each predictor.
- `outcome`: The name of the outcome column.
- `predictor`: The names of the predictor inputs.

These data are accessed using `object@results` (see examples below).

## See Also

Other class score metrics: `score_cor_pearson`, `score_imp_rf`, `score_info_gain`, `score_roc_auc`, `score_xtab_pval_chisq`

## Examples

```
# Analysis of variance where `class` is the class predictor and the numeric
# predictors are the outcomes/responses

cell_data <- modeldata::cells
cell_data$case <- NULL

# ANOVA p-value
cell_p_val_res <-
  score_aov_pval |>
  fit(class ~ ., data = cell_data)
cell_p_val_res@results

# ANOVA raw p-value
natrual_units <- score_aov_pval |> dont_log_pvalues()
cell_pval_natrual_res <-
  natrual_units |>
  fit(class ~ ., data = cell_data)
cell_pval_natrual_res@results

# ANOVA t/F-statistic
cell_t_stat_res <-
  score_aov_fstat |>
  fit(class ~ ., data = cell_data)
cell_t_stat_res@results

# ------------------------------------------------------------------------------
library(dplyr)

# Analysis of variance where `chem_fp_*` are the class predictors and
# `permeability` is the numeric outcome/response

permeability <-
  modeldata::permeability_qsar |>
  # Make the problem a little smaller for time; use 50 predictors
  select(1:51) |>
  # Make the binary predictor columns into factors
  mutate(across(starts_with("chem_fp"), as.factor))

perm_p_val_res <-
  score_aov_pval |>
  fit(permeability ~ ., data = permeability)
perm_p_val_res@results

# Note that some `lm()` calls failed and are given NA score values. For
# example:
table(permeability$chem_fp_0007)

perm_t_stat_res <-
  score_aov_fstat |>
  fit(permeability ~ ., data = permeability)
perm_t_stat_res@results
```

`score_cor_pearson` *Scoring via correlation coefficient*

## Description

These two objects can be used to compute importance scores based on correlation coefficient.

## Usage

```
score_cor_pearson

score_cor_spearman
```

## Format

An object of class `filtro::class_score_cor` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_cor` (inherits from `filtro::class_score`, `S7_object`) of length 1.

## Details

These objects are used when:

- The predictors are numeric and the outcome is numeric.

In this case, a correlation coefficient (via `stats::cov.wt()`) is computed with the proper variable roles. Values closer to 1 or -1 (i.e., `abs(cor_pearson)` closer to 1) are associated with more important predictors.

**Estimating the scores:**

In **filtro**, the `score_*` objects define a scoring method (e.g., data input requirements, package dependencies, etc). To compute the scores for a specific data set, the `fit()` method is used. The main arguments for these functions are:

`object` A score class object (e.g., `score_cor_pearson`).

`formula` A standard R formula with a single outcome on the right-hand side and one or more predictors (or `.`) on the left-hand side. The data are processed via `stats::model.frame()`

`data` A data frame containing the relevant columns defined by the formula.

`...` Further arguments passed to or from other methods.

`case_weights` A quantitative vector of case weights that is the same length as the number of rows in `data`. The default of `NULL` indicates that there are no case weights.

Missing values are removed for each predictor/outcome combination being scored.

In cases where the underlying computations fail, the scoring proceeds silently, and a missing value is given for the score.

## Value

An S7 object. The primary property of interest is in `results`. This is a data frame of results that is populated by the `fit()` method and has columns:

- `name`: The name of the score (e.g., `score_cor_pearson` or `score_cor_spearman`).
- `score`: The estimates for each predictor.
- `outcome`: The name of the outcome column.
- `predictor`: The names of the predictor inputs.

These data are accessed using `object@results` (see examples below).

## See Also

Other class score metrics: `score_aov_pval`, `score_imp_rf`, `score_info_gain`, `score_roc_auc`, `score_xtab_pval_chisq`

## Examples

```
library(dplyr)

ames <- modeldata::ames

# Pearson correlation
ames_cor_pearson_res <-
  score_cor_pearson |>
  fit(Sale_Price ~ ., data = ames)
ames_cor_pearson_res@results

# Spearman correlation
ames_cor_spearman_res <-
  score_cor_spearman |>
  fit(Sale_Price ~ ., data = ames)
ames_cor_spearman_res@results
```

---

# score_imp_rf *Scoring via random forests*

---

## Description

Three different random forest models can be used to measure predictor importance.

## Usage

```
score_imp_rf

score_imp_rf_conditional

score_imp_rf_oblique
```

## Format

An object of class `filtro::class_score_imp_rf` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_imp_rf` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_imp_rf` (inherits from `filtro::class_score`, `S7_object`) of length 1.

## Details

These objects are used when either:

- The predictors are numeric and the outcome is a factor/category, or
- The predictors are factors and the outcome is numeric.

In either case, a random forest, conditional random forest, or oblique random forest (via `ranger::ranger()`, `partykit::cforest()`, or `aorsf::orsf()`) is created with the proper variable roles, and the feature importance scores are computed. Larger values are associated with more important predictors.

When a predictor's importance score is 0, `partykit::cforest()` may omit its name from the results. In cases like these, a score of 0 is assigned to the missing predictors.

**Estimating the scores:**

In **filtro**, the `score_*` objects define a scoring method (e.g., data input requirements, package dependencies, etc). To compute the scores for a specific data set, the `fit()` method is used. The main arguments for these functions are:

`object` A score class object (e.g., `score_imp_rf`).

`formula` A standard R formula with a single outcome on the right-hand side and one or more predictors (or `.`) on the left-hand side. The data are processed via `stats::model.frame()`

`data` A data frame containing the relevant columns defined by the formula.

`...` Further arguments passed to or from other methods.

`case_weights` A quantitative vector of case weights that is the same length as the number of rows in `data`. The default of `NULL` indicates that there are no case weights.

Missing values are removed by case-wise deletion.

In cases where the underlying computations fail, the scoring proceeds silently, and a missing value is given for the score.

## Value

An S7 object. The primary property of interest is in `results`. This is a data frame of results that is populated by the `fit()` method and has columns:

- `name`: The name of the score (e.g., `imp_rf`).
- `score`: The estimates for each predictor.
- `outcome`: The name of the outcome column.
- `predictor`: The names of the predictor inputs.

These data are accessed using `object@results` (see examples below).

## See Also

Other class score metrics: score_aov_pval, score_cor_pearson, score_info_gain, score_roc_auc, score_xtab_pval_chisq

## Examples

```
library(dplyr)

# Random forests for classification task

cells_subset <- modeldata::cells |>
  # Use a small example for efficiency
  dplyr::select(
    class,
    angle_ch_1,
    area_ch_1,
    avg_inten_ch_1,
    avg_inten_ch_2,
    avg_inten_ch_3
  ) |>
  slice(1:50)

# Random forest
set.seed(42)
cells_imp_rf_res <- score_imp_rf |>
  fit(class ~ ., data = cells_subset)
cells_imp_rf_res@results

# Conditional random forest
cells_imp_rf_conditional_res <- score_imp_rf_conditional |>
  fit(class ~ ., data = cells_subset, trees = 10)
cells_imp_rf_conditional_res@results

# Oblique random forest
cells_imp_rf_oblique_res <- score_imp_rf_oblique |>
  fit(class ~ ., data = cells_subset)
cells_imp_rf_oblique_res@results

# ------------------------------------------------------------------------

# Random forests for regression task

ames_subset <- modeldata::ames |>
  # Use a small example for efficiency
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
```

```
  ) |>
  slice(1:50)
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

set.seed(42)
ames_imp_rf_regression_task_res <-
  score_imp_rf |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_imp_rf_regression_task_res@results
```

---

`score_info_gain` *Scoring via entropy-based filters*

---

## Description

Three different information theory (entropy) scores can be computed.

## Usage

```
score_info_gain

score_gain_ratio

score_sym_uncert
```

## Format

An object of class `filtro::class_score_info_gain` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_info_gain` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_info_gain` (inherits from `filtro::class_score`, `S7_object`) of length 1.

## Details

These objects are used when either:

- The predictors are numeric and the outcome is a factor/category, or
- The predictors are factors and the outcome is numeric.

In either case, an entropy-based filter (via `FSelectorRcpp::information_gain()`) is applied with the proper variable roles. Depending on the chosen method, information gain, gain ratio, or symmetrical uncertainty is computed. Larger values are associated with more important predictors.

**Estimating the scores:**

In **filtro**, the `score_*` objects define a scoring method (e.g., data input requirements, package dependencies, etc). To compute the scores for a specific data set, the `fit()` method is used. The main arguments for these functions are:

`object` A score class object (e.g., `score_info_gain`).

`formula` A standard R formula with a single outcome on the right-hand side and one or more predictors (or `.`) on the left-hand side. The data are processed via `stats::model.frame()`

`data` A data frame containing the relevant columns defined by the formula.

`...` Further arguments passed to or from other methods.

`case_weights` A quantitative vector of case weights that is the same length as the number of rows in `data`. The default of `NULL` indicates that there are no case weights.

Missing values are removed for each predictor/outcome combination being scored.

In cases where the underlying computations fail, the scoring proceeds silently, and a missing value is given for the score.

## Value

An S7 object. The primary property of interest is in `results`. This is a data frame of results that is populated by the `fit()` method and has columns:

- `name`: The name of the score (e.g., `info_gain`).
- `score`: The estimates for each predictor.
- `outcome`: The name of the outcome column.
- `predictor`: The names of the predictor inputs.

These data are accessed using `object@results` (see examples below).

## See Also

Other class score metrics: `score_aov_pval`, `score_cor_pearson`, `score_imp_rf`, `score_roc_auc`, `score_xtab_pval_chisq`

## Examples

```
library(dplyr)

# Entropy-based filter for classification tasks

cells_subset <- modeldata::cells |>
  dplyr::select(
    class,
    angle_ch_1,
    area_ch_1,
    avg_inten_ch_1,
    avg_inten_ch_2,
    avg_inten_ch_3
  )
```

```
# Information gain
cells_info_gain_res <- score_info_gain |>
  fit(class ~ ., data = cells_subset)
cells_info_gain_res@results

# Gain ratio
cells_gain_ratio_res <- score_gain_ratio |>
  fit(class ~ ., data = cells_subset)
cells_gain_ratio_res@results

# Symmetrical uncertainty
cells_sym_uncert_res <- score_sym_uncert |>
  fit(class ~ ., data = cells_subset)
cells_sym_uncert_res@results

# ------------------------------------------------------------------------------

# Entropy-based filter for regression tasks

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

regression_task <- score_info_gain
regression_task@mode <- "regression"

ames_info_gain_regression_task_res <-
  regression_task |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_info_gain_regression_task_res@results
```

---

`score_roc_auc` *Scoring via area under the Receiver Operating Characteristic curve (ROC AUC)*

---

## Description

The area under the ROC curves can be used to measure predictor importance.

## Usage

```
score_roc_auc
```

## Format

An object of class `filtro::class_score_roc_auc` (inherits from `filtro::class_score`, `S7_object`) of length 1.

## Details

This objects are used when either:

- The predictors are numeric and the outcome is a factor/category, or
- The predictors are factors and the outcome is numeric.

In either case, a ROC curve (via `pROC::roc()` or `pROC::multiclass.roc()`) is created with the proper variable roles, and the area under the ROC curve is computed (via `pROC::auc()`). Values higher than 0.5 (i.e., `max(roc_auc, 1 - roc_auc)` > 0.5) are associated with more important predictors.

**Estimating the scores:**

In **filtro**, the `score_*` objects define a scoring method (e.g., data input requirements, package dependencies, etc). To compute the scores for a specific data set, the `fit()` method is used. The main arguments for these functions are:

`object` A score class object (e.g., `score_cor_pearson`).

`formula` A standard R formula with a single outcome on the right-hand side and one or more predictors (or `.`) on the left-hand side. The data are processed via `stats::model.frame()`

`data` A data frame containing the relevant columns defined by the formula.

`...` Further arguments passed to or from other methods.

`case_weights` A quantitative vector of case weights that is the same length as the number of rows in `data`. The default of `NULL` indicates that there are no case weights. NOTE case weights cannot be used when a multiclass ROC is computed.

Missing values are removed for each predictor/outcome combination being scored.

In cases where the underlying computations fail, the scoring proceeds silently, and a missing value is given for the score.

## Value

An S7 object. The primary property of interest is in `results`. This is a data frame of results that is populated by the `fit()` method and has columns:

- `name`: The name of the score (e.g., `roc_auc`).
- `score`: The estimates for each predictor.
- `outcome`: The name of the outcome column.
- `predictor`: The names of the predictor inputs.

These data are accessed using `object@results` (see examples below).

## See Also

Other class score metrics: score_aov_pval, score_cor_pearson, score_imp_rf, score_info_gain, score_xtab_pval_chisq

## Examples

```
library(dplyr)

# ROC AUC where the numeric predictors are the predictors and
# `class` is the class outcome/response

cells_subset <- modeldata::cells |>
  dplyr::select(
    class,
    angle_ch_1,
    area_ch_1,
    avg_inten_ch_1,
    avg_inten_ch_2,
    avg_inten_ch_3
  )

cells_roc_auc_res <- score_roc_auc |>
  fit(class ~ ., data = cells_subset)
cells_roc_auc_res@results

# ------------------------------------------------------------------------------

# ROC AUC where `Sale_Price` is the numeric predictor and the class predictors
# are the outcomes/responses

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_roc_auc_res <- score_roc_auc |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_roc_auc_res@results
# TODO Add multiclass example
```

---

score_xtab_pval_chisq *Scoring via the chi-squared test or Fisher's exact test*

---

## Description

These two objects can be used to compute importance scores based on chi-squared test or Fisher's exact test.

## Usage

```
score_xtab_pval_chisq

score_xtab_pval_fisher
```

## Format

An object of class `filtro::class_score_xtab` (inherits from `filtro::class_score`, `S7_object`) of length 1.

An object of class `filtro::class_score_xtab` (inherits from `filtro::class_score`, `S7_object`) of length 1.

## Details

These objects are used when:

- The predictors are factors and the outcome is a factor.

In this case, a contingency table (via `table()`) is created with the proper variable roles, and the cross tabulation p-value is computed using either the chi-squared test (via `stats::chisq.test()`) or Fisher's exact test (via `stats::fisher.test()`). The p-value that is returned is transformed to be `-log10(p_value)` so that larger values are associated with more important predictors.

### Estimating the scores:

In **filtro**, the `score_*` objects define a scoring method (e.g., data input requirements, package dependencies, etc). To compute the scores for a specific data set, the `fit()` method is used. The main arguments for these functions are:

`object` A score class object (e.g., `score_xtab_pval_chisq`).

`formula` A standard R formula with a single outcome on the right-hand side and one or more predictors (or `.`) on the left-hand side. The data are processed via `stats::model.frame()`

`data` A data frame containing the relevant columns defined by the formula.

`...` Further arguments passed to or from other methods.

`case_weights` A quantitative vector of case weights that is the same length as the number of rows in `data`. The default of `NULL` indicates that there are no case weights.

Missing values are removed for each predictor/outcome combination being scored.

In cases where the underlying computations fail, the scoring proceeds silently, and a missing value is given for the score.

## Value

An S7 object. The primary property of interest is in `results`. This is a data frame of results that is populated by the `fit()` method and has columns:

- `name`: The name of the score (e.g., `pval_chisq`).
- `score`: The estimates for each predictor.
- `outcome`: The name of the outcome column.
- `predictor`: The names of the predictor inputs.

These data are accessed using `object@results` (see examples below).

## See Also

Other class score metrics: `score_aov_pval`, `score_cor_pearson`, `score_imp_rf`, `score_info_gain`, `score_roc_auc`

## Examples

```
# Binary factor example

library(titanic)
library(dplyr)

titanic_subset <- titanic_train |>
  mutate(across(c(Survived, Pclass, Sex, Embarked), as.factor)) |>
  select(Survived, Pclass, Sex, Age, Fare, Embarked)

# Chi-squared test
titanic_xtab_pval_chisq_res <- score_xtab_pval_chisq |>
  fit(Survived ~ ., data = titanic_subset)
titanic_xtab_pval_chisq_res@results

# Chi-squared test adjusted p-values
titanic_xtab_pval_chisq_p_adj_res <- score_xtab_pval_chisq |>
  fit(Survived ~ ., data = titanic_subset, adjustment = "BH")

# Fisher's exact test
titanic_xtab_pval_fisher_res <- score_xtab_pval_fisher |>
  fit(Survived ~ ., data = titanic_subset)
titanic_xtab_pval_fisher_res@results

# Chi-squared test where `class` is the multiclass outcome/response

hpc_subset <- modeldata::hpc_data |>
  dplyr::select(
    class,
    protocol,
    hour
  )

hpc_xtab_pval_chisq_res <- score_xtab_pval_chisq |>
```

```
    fit(class ~ ., data = hpc_subset)
hpc_xtab_pval_chisq_res@results
```

---

`show_best_desirability_num` *Show best desirability scores, based on number of predictors* (plural)

---

## Description

Similar to `show_best_desirability_prop()` that can simultaneously optimize multiple scores using desirability functions. See `show_best_score_num()` for *singular* scoring method.

## Usage

```
show_best_desirability_num(x, ..., num_terms = 5)
```

## Arguments

| | |
|---|---|
| `x` | A tibble or data frame returned by `fill_safe_values()`. |
| `...` | One or more desirability selectors to configure the optimization. |
| `num_terms` | An integer value specifying the number of predictors to consider. |

## Details

See `show_best_desirability_prop()` for details.

## Value

A tibble with `num_terms` number of rows. When showing the results, the metrics are presented in "wide format" (one column per metric) and there are new columns for the corresponding desirability values (each starts with `.d_`).

## Examples

```
library(desirability2)
library(dplyr)

# Remove outcome
ames_scores_results <- ames_scores_results |>
  dplyr::select(-outcome)
ames_scores_results

show_best_desirability_num(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1)
)
```

```
show_best_desirability_num(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1),
  maximize(imp_rf)
)

show_best_desirability_num(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1),
  maximize(imp_rf),
  maximize(infogain)
)

show_best_desirability_num(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1),
  maximize(imp_rf),
  maximize(infogain),
  num_terms = 2
)

show_best_desirability_num(
  ames_scores_results,
  target(cor_pearson, low = 0.2, target = 0.255, high = 0.9)
)

show_best_desirability_num(
  ames_scores_results,
  constrain(cor_pearson, low = 0.2, high = 1)
)
```

---

`show_best_desirability_prop` *Show best desirability scores, based on proportion of predictors* (plural)

---

## Description

Analogous to, and adapted from `desirability2::show_best_desirability()` that can simultaneously optimize multiple scores using desirability functions. See `show_best_score_prop()` for *singular* filtering method.

## Usage

```
show_best_desirability_prop(x, ..., prop_terms = 1)
```

## Arguments

| | |
|---|---|
| x | A tibble or data frame returned by `fill_safe_values()`. |
| ... | One or more desirability selectors to configure the optimization. |
| prop_terms | A numeric value specifying the proportion of predictors to consider. |

## Details

Desirability functions might help when selecting the best model based on more than one performance metric. The user creates a desirability function to map values of a metric to a `[0, 1]` range where 1.0 is most desirable and zero is unacceptable. After constructing these for the metric of interest, the overall desirability is computed using the geometric mean of the individual desirabilities.

The verbs that can be used in ... (and their arguments) are:

- `maximize()` when larger values are better, such as the area under the ROC score.
- `minimize()` for metrics such as RMSE or the Brier score.
- `target()` for cases when a specific value of the metric is important.
- `constrain()` is used when there is a range of values that are equally desirable.

## Value

A tibble with `prop_terms` proportion of rows. When showing the results, the metrics are presented in "wide format" (one column per metric) and there are new columns for the corresponding desirability values (each starts with `.d_`).

## Examples

```
library(desirability2)
library(dplyr)

# Remove outcome
ames_scores_results <- ames_scores_results |>
  dplyr::select(-outcome)
ames_scores_results

show_best_desirability_prop(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1)
)

show_best_desirability_prop(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1),
  maximize(imp_rf)
)

show_best_desirability_prop(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1),
```

```
  maximize(imp_rf),
  maximize(infogain)
)

show_best_desirability_prop(
  ames_scores_results,
  maximize(cor_pearson, low = 0, high = 1),
  maximize(imp_rf),
  maximize(infogain),
  prop_terms = 0.2
)

show_best_desirability_prop(
  ames_scores_results,
  target(cor_pearson, low = 0.2, target = 0.255, high = 0.9)
)

show_best_desirability_prop(
  ames_scores_results,
  constrain(cor_pearson, low = 0.2, high = 1)
)
```

---

`show_best_score_cutoff` *Show best score, based on based on cutoff value* (singular)

---

## Description

Show best score, based on based on cutoff value *(singular)*

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `...` | Further arguments passed to or from other methods. |
| `cutoff` | A numeric value specifying the cutoff value. |
| `target` | A numeric value specifying the target value. The default of `NULL` indicates that there is no target value. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)
```

```
ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Show best score
ames_aov_pval_res |> show_best_score_cutoff(cutoff = 130)
```

---

`show_best_score_dual` *Show best score, based on number or proportion of predictors with optional cutoff value* (singular)

---

## Description

Show best score, based on number or proportion of predictors with optional cutoff value *(singular)*

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `...` | Further arguments passed to or from other methods. |
| `prop_terms` | A numeric value specifying the proportion of predictors to consider. |
| `num_terms` | An integer value specifying the number of predictors to consider. |
| `cutoff` | A numeric value specifying the cutoff value. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
```

```
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Show best score
ames_aov_pval_res |> show_best_score_dual(prop_terms = 0.5)
ames_aov_pval_res |> show_best_score_dual(prop_terms = 0.5, cutoff = 130)

ames_aov_pval_res |> show_best_score_dual(num_terms = 2)
ames_aov_pval_res |> show_best_score_dual(num_terms = 2, cutoff = 130)
```

---

`show_best_score_num` *Show best score, based on number of predictors* (singular)

---

## Description

Show best score, based on number of predictors *(singular)*

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `...` | Further arguments passed to or from other methods. |
| `num_terms` | An integer value specifying the number of predictors to consider. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
```

```
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Show best score
ames_aov_pval_res |> show_best_score_num(num_terms = 2)
```

---

`show_best_score_prop` *Show best score, based on proportion of predictors* (singular)

---

## Description

Show best score, based on proportion of predictors *(singular)*

## Arguments

| | |
|---|---|
| `x` | A score class object (e.g., `score_cor_pearson`). |
| `...` | Further arguments passed to or from other methods. |
| `prop_terms` | A numeric value specifying the proportion of predictors to consider. |

## Value

A tibble of score results.

## Examples

```
library(dplyr)

ames_subset <- modeldata::ames |>
  dplyr::select(
    Sale_Price,
    MS_SubClass,
    MS_Zoning,
    Lot_Frontage,
    Lot_Area,
    Street
  )
ames_subset <- ames_subset |>
```

```
  dplyr::mutate(Sale_Price = log10(Sale_Price))

ames_aov_pval_res <-
  score_aov_pval |>
  fit(Sale_Price ~ ., data = ames_subset)
ames_aov_pval_res@results

# Show best score
ames_aov_pval_res |> show_best_score_prop(prop_terms = 0.2)
```

# Index