Package 'desirability2'

July 23, 2025

Title Desirability Functions for Multiparameter Optimization

Version 0.1.0

Description In-line functions for multivariate optimization via desirability functions (Derringer and Suich, 1980, <doi:10.1080/00224065.1980.11980968>) with easy use within 'dplyr' pipelines.

License MIT + file LICENSE

URL https://desirability2.tidymodels.org, https://github.com/tidymodels/desirability2

BugReports https://github.com/tidymodels/desirability2/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, purrr, rlang (>= 1.1.0), S7, stats, tibble

Suggests covr, ggplot2, spelling, testthat (>= 3.0.0), tune

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

Collate 'checks.R' 'computations.R' 'data.R' 'in-line.R' 'desirability.R' 'desirability2-package.R' 'import-standalone-obj-type.R' 'import-standalone-types-check.R' 'overall.R' 'tune.R' 'zzz.R'

NeedsCompilation no

Author Max Kuhn [aut, cre] (ORCID: <https://orcid.org/0000-0003-2402-136X>), Posit Software, PBC [cph, fnd]

Maintainer Max Kuhn <max@posit.co>

Repository CRAN

Date/Publication 2025-07-22 22:52:28 UTC

Contents

`classification_results` *Classification results*

Description

These data are a variation of a case study at `tidymodels.org` where a penalized regression model was used for a binary classification task. The outcome metrics in `classification_results` are the areas under the ROC and PR curve, log-likelihood, and the number of predictors selected for a given amount of penalization. Two tuning parameters, `mixture` and `penalty`, were varied across 300 conditions.

Value

`classification_results`
a tibble

Source

See the `example-data` directory in the package with code that is a variation of the analysis shown at https://www.tidymodels.org/start/case-study/.

Examples

```
data(classification_results)
```

`d_overall` *Determine overall desirability*

Description

Once desirability columns have been created, determine the overall desirability using a mean (geometric by default).

Usage

```
d_overall(..., geometric = TRUE, tolerance = 0)
```

Arguments

...	One or more unquoted expressions separated by commas. To choose multiple columns using selectors, `dplyr::across()` can be used (see the example below).
geometric	A logical for whether the geometric or arithmetic mean should be used to summarize the columns.
tolerance	A numeric value where values strictly less than this value are capped at the value. For example, if users wish to use the geometric mean without completely excluding settings, a value greater than zero can be used.

Value

A numeric vector.

See Also

`d_max()`

Examples

```
library(dplyr)

# Choose model tuning parameters that minimize the number of predictors used
# while maximizing the area under the ROC curve.

classification_results |>
  mutate(
    d_feat = d_min(num_features, 1, 200),
    d_roc  = d_max(roc_auc, 0.5, 0.9),
    d_all  = d_overall(across(starts_with("d_")))
  ) |>
  arrange(desc(d_all))

# Bias the ranking toward minimizing features by using a larger scale.

classification_results |>
  mutate(
    d_feat = d_min(num_features, 1, 200, scale = 3),
    d_roc  = d_max(roc_auc, 0.5, 0.9),
    d_all  = d_overall(across(starts_with("d_")))
  ) |>
  arrange(desc(d_all))
```

`inline_desirability` *Desirability functions for in-line computations*

Description

Desirability functions map some input to a `[0, 1]` scale where zero is unacceptable and one is most desirable. The mapping depends on the situation. For example, `d_max()` increases desirability with the input while `d_min()` does the opposite. See the plots in the examples to see more examples.

Currently, only the desirability functions defined by Derringer and Suich (1980) are implemented.

Usage

```
d_max(x, low, high, scale = 1, missing = NA_real_, use_data = FALSE)

d_min(x, low, high, scale = 1, missing = NA_real_, use_data = FALSE)

d_target(
  x,
  low,
  target,
  high,
  scale_low = 1,
  scale_high = 1,
  missing = NA_real_,
  use_data = FALSE
)

d_box(x, low, high, missing = NA_real_, use_data = FALSE)

d_custom(x, x_vals, desirability, missing = NA_real_)

d_category(x, categories, missing = NA_real_)
```

Arguments

`x`	A vector of data to compute the desirability function
`low`, `high`, `target`	Single numeric values that define the active ranges of desirability.
`scale`, `scale_low`, `scale_high`	A single numeric value to rescale the desirability function (each should be great than 0.0). Values >1.0 make the desirability more difficult to satisfy while smaller values make it easier (see the examples below). `scale_low` and `scale_high` do the same for target functions with `scale_low` affecting the range below the `target` value and `scale_high` affecting values greater than `target`.
`missing`	A single numeric value on `[0, 1]` (or `NA_real_`) that defines how missing values in `x` are mapped to the desirability score.

use_data — Should the low, middle, and/or high values be derived from the data (x) using the minimum, maximum, or median (respectively)?

x_vals, desirability — Numeric vectors of the same length that define the desirability results at specific values of x. Values below and above the data in x_vals are given values of zero and one, respectively.

categories — A named vector of desirability values that match all possible categories to specific desirability values. Data that are not included in categories are given the value in missing.

Details

Each function translates the values to desirability on [0, 1].

Equations:

Maximization:

- data > high: d = 1.0
- data < low: d = 0.0
- low <= data <= high: $d = \left(\frac{data-low}{high-low}\right)^{scale}$

Minimization:

- data > high: d = 0.0
- data < low: d = 1.0
- low <= data <= high: $d = \left(\frac{data=low}{low-high}\right)^{scale}$

Target:

- data > high: d = 0.0
- data < low: d = 0.0
- low <= data <= target: $d = \left(\frac{data-low}{target-low}\right)^{scale\_low}$
- target <= data <= high: $d = \left(\frac{data-high}{target-high}\right)^{scale\_high}$

Box:

- data > high: d = 0.0
- data < low: d = 0.0
- low <= data <= high: d = 1.0

Categories:

- data = level: d = 1.0
- data != level: d = 0.0

Custom:

For the sequence of values given to the function, d_custom() will return the desirability values that correspond to data matching values in x_vals. Otherwise, linear interpolation is used for values in-between.

Data-Based Values:

By default, most of the d_*() functions require specific user inputs for arguments such as low, target and high. When use_data = TRUE, the functions can use the minimum, median, and maximum values of the existing data to estimate those values (respectively) but *only when users do not specify them*.

Value

A numeric vector on `[0, 1]` where larger values are more desirable.

References

Derringer, G. and Suich, R. (1980), Simultaneous Optimization of Several Response Variables. *Journal of Quality Technology*, 12, 214-219.

See Also

`d_overall()`

Examples

```
library(dplyr)
library(ggplot2)

set.seed(1)
dat <- tibble(x = sort(runif(30)), y = sort(runif(30)))
d_max(dat$x[1:10], 0.1, 0.75)

dat |>
  mutate(d_x = d_max(x, 0.1, 0.75))

set.seed(2)
tibble(z = sort(runif(100))) |>
  mutate(
    no_scale = d_max(z, 0.1, 0.75),
    easier   = d_max(z, 0.1, 0.75, scale = 1/2)
  ) |>
  ggplot(aes(x = z)) +
  geom_point(aes(y = no_scale)) +
  geom_line(aes(y = no_scale), alpha = .5) +
  geom_point(aes(y = easier), col = "blue") +
  geom_line(aes(y = easier), col = "blue", alpha = .5) +
  lims(x = 0:1, y = 0:1) +
  coord_fixed() +
  ylab("Desirability")

# ------------------------------------------------------------------------------
# Target example

dat |>
  mutate(
    triangle = d_target(x, 0.1, 0.5, 0.9, scale_low = 2, scale_high = 1/2)
  ) |>
  ggplot(aes(x = x, y = triangle)) +
  geom_point() +
  geom_line(alpha = .5) +
  lims(x = 0:1, y = 0:1) +
  coord_fixed() +
  ylab("Desirability")
```

```
# ------------------------------------------------------------------------------
# Box constraints

dat |>
  mutate(box = d_box(x, 1/4, 3/4)) |>
  ggplot(aes(x = x, y = box)) +
  geom_point() +
  geom_line(alpha = .5) +
  lims(x = 0:1, y = 0:1) +
  coord_fixed() +
  ylab("Desirability")

# ------------------------------------------------------------------------------
# Custom function

v_x <- seq(0, 1, length.out = 20)
v_d <- 1 - exp(-10 * abs(v_x - .5))

dat |>
  mutate(v = d_custom(x, v_x, v_d)) |>
  ggplot(aes(x = x, y = v)) +
  geom_point() +
  geom_line(alpha = .5) +
  lims(x = 0:1, y = 0:1) +
  coord_fixed() +
  ylab("Desirability")

# ------------------------------------------------------------------------------
# Qualitative data

set.seed(3)
groups <- sort(runif(10))
names(groups) <- letters[1:10]

tibble(x = letters[1:7]) |>
  mutate(d = d_category(x, groups)) |>
  ggplot(aes(x = x, y = d)) +
  geom_bar(stat = "identity") +
  lims(y = 0:1) +
  ylab("Desirability")

# ------------------------------------------------------------------------------
# Apply the same function to many columns at once (dplyr > 1.0)

dat |>
  mutate(across(c(everything()), ~ d_min(., .2, .6), .names = "d_{col}"))

# ------------------------------------------------------------------------------
# Using current data

set.seed(9015)
tibble(z = c(0, sort(runif(20)), 1)) |>
```

```
  mutate(
    user_specified = d_max(z, 0.1, 0.75),
    data_driven    = d_max(z, use_data = TRUE)
  ) |>
  ggplot(aes(x = z)) +
  geom_point(aes(y = user_specified)) +
  geom_line(aes(y = user_specified), alpha = .5) +
  geom_point(aes(y = data_driven), col = "blue") +
  geom_line(aes(y = data_driven), col = "blue", alpha = .5) +
  lims(x = 0:1, y = 0:1) +
  coord_fixed() +
  ylab("Desirability")
```

`show_best_desirability` *Investigate best tuning parameters*

Description

Analogs to `tune::show_best()` and `tune::select_best()` that can simultaneously optimize multiple metrics or characteristics using desirability functions.

Usage

```
show_best_desirability(x, ..., n = 5, eval_time = NULL)

select_best_desirability(x, ..., eval_time = NULL)
```

Arguments

`x`	The results of `tune_grid()` or `tune_bayes()`.
`...`	One or more desirability selectors to configure the optimization.
`n`	An integer for the number of top results/rows to return.
`eval_time`	A single numeric time point where dynamic event time metrics should be chosen (e.g., the time-dependent ROC curve, etc). The values should be consistent with the values used to create `x`. The `NULL` default will automatically use the first evaluation time used by `x`.

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.
- `categories()` is for situations where we want to set desirability for qualitative columns.

Except for `categories()`, these functions have arguments `low` and `high` to set the ranges of the metrics. For example, using:

```
minimize(rmse, low = 0.1, high = 2.0)
```

means that values above 2.0 are unacceptable and that an RMSE of 0.1 is the best possible outcome.

There is also an argument that can be used to state how important a metric is in the optimization. By default, using `scale = 1` means that desirability linearly changes between `low` and `high`. Using a `scale` value greater than 1 will make it more difficult to satisfy the criterion when suboptimal values are evaluated. Conversely, a value less than one will diminish the influence of that metric. The `categories()` does not have a scaling argument while `target()` has two (`scale_low` and `scale_high`) for ranges on either side of the `target`.

Here is an example that optimizes RMSE and the concordance correlation coefficient (a.k.a. "ccc"), with more emphasis on the former:

```
minimize(rmse, low = 0.10, high = 2.00, scale = 3.0),
maximize(ccc,  low = 0.00, high = 1.00) # scale defaults to 1.0
```

If `low`, `high`, or `target` are not specified, the observed data are used to estimate their values. For the previous example, if we were to use

```
minimize(rmse, low = 0.10, high = 2.00, scale = 3.0),
maximize(ccc)
```

and the concordance correlation coefficient statistics ranged from 0.21 to 0.35, the actual goals would end up as:

```
minimize(rmse, low = 0.10, high = 2.00, scale = 3.0),
maximize(ccc,  low = 0.21, high = 0.35)
```

More than one variable can be used in a term as long as R can parse and execute the expression. For example, you could define the Youden's J statistic using

```
maximize(sensitivity + specificity - 1)
```

(although there is a function for this metric in the **yardstick** package).

If the columns of the data set have missing values, their corresponding desirability will be missing. The overall desirability computation excludes missing values.

We advise not referencing global values or inline functions inside of these verbs.

Also note that there may be more than `n` values returned when showing the results; there may be more than one model configuration that has identical overall desirability.

Value

`show_best_desirability()` returns a tibble with `n` rows while `select_best_desirability()` returns a single row. 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_`).

References

Derringer, G. and Suich, R. (1980), Simultaneous Optimization of Several Response Variables. *Journal of Quality Technology*, 12, 214-219.

Bartz-Beielstein, T. (2025). Multi-Objective Optimization and Hyperparameter Tuning With Desirability Functions. arXiv preprint arXiv:2503.23595.

See Also

`d_max()`, `d_overall()`

Examples

```
# use pre-tuned results to demonstrate:
if (rlang::is_installed("tune")) {

  show_best_desirability(
    tune::ames_iter_search,
    maximize(rsq),
    minimize(rmse, scale = 3)
  )

  select_best_desirability(
    tune::ames_iter_search,
    maximize(rsq),
    minimize(rmse, scale = 3)
  )
}
```

Index