Package ‘KraljicMatrix’

March 6, 2018

Type Package

Title A Quantified Implementation of the Kraljic Matrix

Version 0.2.1

Maintainer Bradley Boehmke <bradleyboehmke@gmail.com>

Date 2017-11-01

Description Implements a quantified approach to the Kraljic Matrix (Kraljic, 1983, <https://hbr.org/1983/09/purchasing-must-become-supply-management>) for strategically analyzing a firm’s purchasing portfolio. It combines multi-objective decision analysis to measure purchasing characteristics and uses this information to place products and services within the Kraljic Matrix.

URL https://github.com/koalaverse/KraljicMatrix

BugReports https://github.com/koalaverse/KraljicMatrix/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports ggplot2, dplyr, tibble, magrittr

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Bradley Boehmke [aut, cre],
Brandon Greenwell [aut],
Andrew McCarthy [aut],
Robert Montgomery [ctb]

Repository CRAN

Date/Publication 2018-03-06 22:49:03 UTC

R topics documented:

`geom_frontier` *Plotting the Pareto Optimal Frontier*

Description

The frontier geom is used to overlay the efficient frontier on a scatterplot.

Usage

```
geom_frontier(mapping = NULL, data = NULL, position = "identity",
  direction = "vh", na.rm = FALSE, show.legend = NA, inherit.aes = TRUE,
  ...)

stat_frontier(mapping = NULL, data = NULL, geom = "step",
  position = "identity", direction = "vh", na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE, quadrant = "top.right", ...)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
`data`	The data to be displayed in this layer.
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`direction`	Direction of stairs: 'vh' for vertical then horizontal, or 'hv' for horizontal then vertical.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend`	Logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
`...`	Other arguments passed on to `layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. They may also be parameters to the paired `geom/stat`.
`geom`	Use to override the default connection between `geom_frontier` and `stat_frontier`.
`quadrant`	See `get_frontier`.

Examples

```
## Not run:

# default will find the efficient front in top right quadrant
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  geom_frontier()

# change the direction of the steps
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  geom_frontier(direction = 'hv')

# use quadrant parameter to change how you define the efficient frontier
ggplot(airquality, aes(Ozone, Temp)) +
  geom_point() +
  geom_frontier(quadrant = 'top.left')

ggplot(airquality, aes(Ozone, Temp)) +
  geom_point() +
  geom_frontier(quadrant = 'bottom.right')

## End(Not run)
```

`get_frontier` *Compute the Pareto Optimal Frontier*

Description

Extract the points that make up the Pareto frontier from a set of data.

Usage

```
get_frontier(data, x, y, quadrant = c("top.right", "bottom.right",
  "bottom.left", "top.left"), decreasing = TRUE)
```

Arguments

data	A data frame.
x	A numeric vector.
y	A numeric vector.
quadrant	Chararacter string specifying which quadrant the frontier should appear in. Default is "top.right".
decreasing	Logical value indicating whether the data returned is in decreasing or ascending order (ordered by x and then y). Default is decreasing order.

Value

A data frame containing the data points that make up the efficient frontier.

See Also

geom_frontier for plotting the Pareto front

Examples

```
# default will find the Pareto optimal observations in top right quadrant
get_frontier(mtcars, mpg, wt)

# the output can be in descending or ascending order
get_frontier(mtcars, mpg, wt, decreasing = FALSE)

# use quadrant parameter to change how you define the efficient frontier
get_frontier(airquality, Ozone, Temp, quadrant = 'top.left')

get_frontier(airquality, Ozone, Temp, quadrant = 'bottom.right')
```

---

kraljic_matrix *Kraljic matrix plotting function*

Description

kraljic_matrix plots each product or service in the Kraljic purchasing matrix based on the attribute value score of x and y

Usage

```
kraljic_matrix(data, x, y)
```

Arguments

`data`	A data frame
`x`	Numeric vector of values
`y`	Numeric vector of values with compatible dimensions to `x`

Value

A Kraljic purchasing matrix plot

See Also

`SAVF_score` for computing the exponential single attribute value score for `x` and `y`

Examples

```
# Given the following \code{x} and \code{y} attribute values we can plot each
# product or service in the purchasing matrix:

# to add a new variable while preserving existing data
library(dplyr)

psc2 <- psc %>%
  mutate(x_SAVF_score = SAVF_score(x_attribute, 1, 5, .653),
         y_SAVF_score = SAVF_score(y_attribute, 1, 10, .7))

kraljic_matrix(psc2, x_SAVF_score, y_SAVF_score)
```

`kraljic_quadrant` *Kraljic quadrant assignment function*

Description

`kraljic_quadrant` assigns the Kraljic purchasing matrix quadrant based on the attribute value score of `x` and `y`

Usage

```
kraljic_quadrant(x, y)
```

Arguments

`x`	Numeric vector of values
`y`	Numeric vector of values with compatible dimensions to `x`

Value

A vector of the same length as x and y with the relevant Kraljic quadrant name

See Also

SAVF_score for computing the exponential single attribute value score for x and y

Examples

```
# Given the following \code{x} and \code{y} attribute values  we can determine
# which quadrant each product or service falls in:

# to add a new variable while preserving existing data
library(dplyr)

psc2 <- psc %>%
  mutate(x_SAVF_score = SAVF_score(x_attribute, 1, 5, .653),
         y_SAVF_score = SAVF_score(y_attribute, 1, 10, .7))

psc2 %>%
  mutate(quadrant = kraljic_quadrant(x_SAVF_score, y_SAVF_score))
```

MAVF_score *Multi-attribute value function*

Description

MAVF_score computes the multi-attribute value score of x and y given their respective weights

Usage

```
MAVF_score(x, y, x_wt, y_wt)
```

Arguments

x	Numeric vector of values
y	Numeric vector of values with compatible dimensions to x
x_wt	Swing weight for x
y_wt	Swing weight for y

Value

A vector of the same length as x and y with the multi-attribute value scores

See Also

`MAVF_sensitivity` to perform sensitivity analysis with a range of x and y swing weights

`SAVF_score` for computing the exponential single attribute value score

Examples

```
# Given the following \code{x} and \code{y} attribute values with \code{x} and
# \code{y} swing weight values of 0.65 and 0.35 respectively, we can compute
# the multi-attribute utility score:

x_attribute <- c(0.92, 0.79, 1.00, 0.39, 0.68, 0.55, 0.73, 0.76, 1.00, 0.74)
y_attribute <- c(0.52, 0.19, 0.62, 1.00, 0.55, 0.52, 0.53, 0.46, 0.61, 0.84)

MAVF_score(x_attribute, y_attribute, x_wt = .65, y_wt = .35)
```

MAVF_sensitivity *Multi-attribute value function sensitivity analysis*

Description

`MAVF_sensitivity` computes summary statistics for multi-attribute value scores of x and y given a range of swing weights for each attribute

Usage

```
MAVF_sensitivity(data, x, y, x_wt_min, x_wt_max, y_wt_min, y_wt_max)
```

Arguments

`data`	A data frame
`x`	Variable from data frame to represent x attribute values
`y`	Variable from data frame to represent y attribute values
`x_wt_min`	Lower bound anchor point for x attribute swing weight
`x_wt_max`	Upper bound anchor point for x attribute swing weight
`y_wt_min`	Lower bound anchor point for y attribute swing weight
`y_wt_max`	Upper bound anchor point for y attribute swing weight

Details

The sensitivity analysis performs a Monte Carlo simulation with 1000 trials for each product or service (row). Each trial randomly selects a weight from a uniform distribution between the lower and upper bound weight parameters and calculates the mult-attribute utility score. From these trials, summary statistics for each product or service (row) are calculated and reported for the final output.

Value

A data frame with added variables consisting of sensitivity analysis summary statistics for each product or service (row).

See Also

`MAVF_score` for computing the multi-attribute value score of x and y given their respective weights

`SAVF_score` for computing the exponential single attribute value score

Examples

```
# Given the following data frame that contains \code{x} and \code{y} attribute
# values for each product or service contract, we can compute how the range of
# swing weights for each \code{x} and \code{y} attribute influences the multi-
# attribute value score.

df <- data.frame(contract = 1:10,
              x_attribute = c(0.92, 0.79, 1.00, 0.39, 0.68, 0.55, 0.73, 0.76, 1.00, 0.74),
              y_attribute = c(0.52, 0.19, 0.62, 1.00, 0.55, 0.52, 0.53, 0.46, 0.61, 0.84))

MAVF_sensitivity(df, x_attribute, y_attribute, .55, .75, .25, .45)
```

---

psc — *Product and service contracts*

Description

A dataset containing a single value score for the x attribute (i.e. supply risk) and y attribute (i.e. profit impact) of 200 product and service contracts (PSC). The variables are as follows:

Usage

```
psc
```

Format

A tibble with 200 rows and 3 variables:

PSC Contract identifier for each product and service

x_attribute x attribute score, from 1 (worst) to 5 (best) in .01 increments

y_attribute y attribute score, from 1 (worst) to 10 (best) in .01 increments

SAVF_plot *Plot the single attribute value curve*

Description

SAVF_plot plots the single attribute value curve along with the subject matter desired values for comparison

Usage

```
SAVF_plot(desired_x, desired_v, x_low, x_high, rho)
```

Arguments

desired_x	Elicited input x value(s)
desired_v	Elicited value score related to elicited input value(s)
x_low	Lower bound anchor point (can be different than min(x))
x_high	Upper bound anchor point (can be different than max(x))
rho	Exponential constant for the value function

Value

A plot that visualizes the single attribute value curve along with the subject matter desired values for comparison

See Also

SAVF_plot_rho_error for plotting the rho squared error terms

SAVF_score for computing the exponential single attribute value score

Examples

```
# Given the single attribute x is bounded between 1 and 5 and the subject matter experts
# prefer x values of 3, 4, & 5 provide a utility score of .75, .90 & 1.0 respectively,
# the preferred rho is 0.54. We can visualize this value function:

SAVF_plot(desired_x = c(3, 4, 5),
          desired_v = c(.75, .9, 1),
          x_low = 1,
          x_high = 5,
          rho = 0.54)
```

SAVF_plot_rho_error *Plot the rho squared error terms*

Description

SAVF_plot_rho_error plots the squared error terms for the rho search space to illustrate the preferred rho that minimizes the squared error between subject matter desired values and exponentially fitted scores

Usage

```
SAVF_plot_rho_error(desired_x, desired_v, x_low, x_high, rho_low = 0,
  rho_high = 1)
```

Arguments

desired_x	Elicited input x value(s)
desired_v	Elicited value score related to elicited input value(s)
x_low	Lower bound anchor point (can be different than min(x))
x_high	Upper bound anchor point (can be different than max(x))
rho_low	Lower bound of the exponential constant search space for a best fit value function
rho_high	Upper bound of the exponential constant search space for a best fit value function

Value

A plot that visualizes the squared error terms for the rho search space

See Also

SAVF_preferred_rho for identifying the preferred rho value

SAVF_score for computing the exponential single attribute value score

Examples

```
# Given the single attribute x is bounded between 1 and 5 and the subject matter experts
# prefer x values of 3, 4, & 5 provide a utility score of .75, .90 & 1.0 respectively, we
# can visualize the error terms for rho values between 0-1:

SAVF_plot_rho_error(desired_x = c(3, 4, 5),
                    desired_v = c(.75, .9, 1),
                    x_low = 1,
                    x_high = 5,
                    rho_low = 0,
                    rho_high = 1)
```

SAVF_preferred_rho *Identify preferred rho*

Description

SAVF_preferred_rho computes the preferred rho that minimizes the squared error between subject matter input desired values and exponentially fitted scores

Usage

```
SAVF_preferred_rho(desired_x, desired_v, x_low, x_high, rho_low = 0,
  rho_high = 1)
```

Arguments

desired_x	Elicited input x value(s)
desired_v	Elicited value score related to elicited input value(s)
x_low	Lower bound anchor point (can be different than min(x))
x_high	Upper bound anchor point (can be different than max(x))
rho_low	Lower bound of the exponential constant search space for a best fit value function
rho_high	Upper bound of the exponential constant search space for a best fit value function

Value

A single element vector that represents the rho value that best fits the exponential utility function to the desired inputs

See Also

SAVF_plot_rho_error for plotting the rho squared error terms

SAVF_score for computing the exponential single attribute value score

Examples

```
# Given the single attribute x is bounded between 1 and 5 and the subject matter experts
# prefer x values of 3, 4, & 5 provide a utility score of .75, .90 & 1.0 respectively, we
# can search for a rho value between 0-1 that provides the best fit utility function:

SAVF_preferred_rho(desired_x = c(3, 4, 5),
                   desired_v = c(.75, .9, 1),
                   x_low = 1,
                   x_high = 5,
                   rho_low = 0,
                   rho_high = 1)
```

`SAVF_score` *Single attribute value function*

Description

SAVF_score computes the exponential single attribute value score of x

Usage

```
SAVF_score(x, x_low, x_high, rho)
```

Arguments

x	Numeric vector of values to score
x_low	Lower bound anchor point (can be different than min(x))
x_high	Upper bound anchor point (can be different than max(x))
rho	Exponential constant for the value function

Value

A vector of the same length as x with the exponential single attribute value scores

See Also

SAVF_plot for plotting single attribute scores

SAVF_preferred_rho for identifying the preferred rho

Examples

```
# The single attribute x is bounded between 1 and 5 and follows an exponential
# utility curve with rho = .653

x <- runif(10, 1, 5)
x
## [1] 2.964853 1.963182 1.223949 1.562025 4.381467 2.286030 3.071066
## [8] 4.470875 3.920913 4.314907

SAVF_score(x, x_low = 1, x_high = 5, rho = .653)
## [1] 0.7800556 0.5038275 0.1468234 0.3315217 0.9605856 0.6131944 0.8001003
## [8] 0.9673124 0.9189685 0.9553165
```

%>% *Pipe functions*

Description

Like dplyr, KraljicMatrix also uses the pipe function, `%>%` to turn function composition into a series of imperative statements.

Arguments

`lhs, rhs`	An R object and a function to apply to it

Examples

```
# given the following \code{psc2} data set
psc2 <- dplyr::mutate(psc, x_SAVF_score = SAVF_score(x_attribute, 1, 5, .653),
                           y_SAVF_score = SAVF_score(y_attribute, 1, 10, .7))

# you can use the pipe operator to re-write the following:
kraljic_matrix(psc2, x_SAVF_score, y_SAVF_score)

# as
psc2 %>% kraljic_matrix(x_SAVF_score, y_SAVF_score)
```

Index