Package 'discAUC'

October 13, 2022

Type Package

Title Linear and Non-Linear AUC for Discounting Data

Description Area under the curve (AUC; Myerson et al., 2001) <doi:10.1901/jeab.2001.76-235>
is a popular measure used in discounting research. Although the calculation of
AUC is standardized, there are differences in AUC based on some assumptions.
For example, Myerson et al. (2001) <doi:10.1901/jeab.2001.76-235>
assumed that (with delay discounting data) a researcher would impute an
indifference point at zero delay equal to the value of the larger, later outcome.
However, this practice is not clearly followed. This imputed zero-delay indifference
point plays an important role in log and ordinal versions of AUC.
Ordinal and log versions of AUC are described by Borges et al. (2016)<doi:10.1002/jeab.219>.
The package can calculate all three versions of AUC [and includes a new version: IHS(AUC)],
impute indifference points when x = 0, calculate ordinal AUC in the case of Halton
sampling of x-values, and account for probability discounting AUC.

Version 0.4.0

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat

Imports dplyr, tibble, rlang, glue

Depends R (>= 2.10)

VignetteBuilder knitr

NeedsCompilation no

Author Jonathan E. Friedel [aut, cre]
(<https://orcid.org/0000-0002-1516-330X>)

Maintainer Jonathan E. Friedel <jfriedel@georgiasouthern.edu>

Repository CRAN

Date/Publication 2021-06-02 07:40:05 UTC

R **topics documented:**

AUC   *Area Under the Curve for Discounting Data*

Description

Area Under the Curve for Discounting Data

Usage

```
AUC(
  dat,
  indiff,
  x_axis,
  prob_disc = FALSE,
  max_x_axis = NULL,
  amount,
  groupings = NULL,
  imp_zero = TRUE,
  type = "linear",
  log_base = 2
)
```

Arguments

dat	Discounting data tibble
indiff	Indifference points Variable in dat
x_axis	Delays/probabilities/social distance variable in dat
prob_disc	Boolean for whether data are probability discounting
max_x_axis	Numeric; Maximum possible value in x_axis
amount	Numeric; Maximum amount of indifference points. (A in discounting models.)
groupings	Variables for grouping (e.g., subject, experimental group) as a character or vector of characters

imp_zero	Boolean for whether indifference points at x_axis = 0 (e.g., delay = 0, odds against = 0, etc.) should be added to the data.
type	String for the type of AUC that should be calculated. Acceptable values are one of c("linear","log","ordinal")
log_base	If using logarithmic, what is the base of the log

Value

Tibble with AUC by all grouping factors. If no grouping factor specified then a tibble with one AUC will be returned.

Examples

```
AUC(examp_DD,
  indiff = "prop_indiff",
  x_axis = "delay_months",
  amount = 1,
  type = "linear",
  prob_disc = FALSE,
  groupings = c("subject", "outcome")
)
```

AUC_zeros	*Impute zero delay/100% likely indifference points*

Description

As defined by Myerson et al. (2001) the indifference point at 0 delay (100% likelihood) is set to 0. This function will add that indifference point, wherever it is missing. If the 0 delay (100% likelihood) is included in the data then it will not be overwritten.

Usage

```
AUC_zeros(dat, indiff, x_axis, amount, groupings = NULL, prob_disc = FALSE)
```

Arguments

dat	Discounting data tibble
indiff	Indifference points Variable
x_axis	Delays/probabilities/social distance variable
amount	Amount of the larger delayed/probablistic/etc. outcome (A in discounting formulas)
groupings	Variables for grouping (e.g., subject, expeirmental group) as a character or vector of characters
prob_disc	Boolean for probability discounting, if set to true function will calculate and report odds against x_axis

Value

Tibble that is grouped by groupings but in the same order as supplied to the function. If prob_disc == FALSE, then the function will add indifference points of amount at x_axis = 0. If prob_disc == TRUE, then the function will add indifference points of amount at x_axis = 1. Additionally, a orig column will be added to indicate whether the indifference point was included in the data or was imputed.

Examples

```
AUC_zeros(
  examp_DD,
  indiff = "prop_indiff",
  x_axis = "delay_months",
  amount = 1,
  groupings = c("subject", "outcome")
)

AUC_zeros(
  examp_PD,
  indiff = "prop_indiff",
  x_axis = "prob",
  amount = 1,
  groupings = c("subject", "outcome"),
  prob_disc = TRUE
)
```

---

examp_DD   *Delay discounting data*

Description

Delay discounting data with repeated measures for subjects across delayed outcomes. Data were obtained from a subset of data from DeHart et al. (2020).

Usage

```
examp_DD
```

Format

A data frame with 360 rows and 4 variables:

subject Subject ID. Positive values are experimentally obtained. -987.987 are median indifference points. -1 and -2 values have indifference points of all 0 and all 1, respectively. These extra data were added for testing and debugging to ensure that AUC calculations will result in 0 when all indifference points are zero and 1 when all indifference points are 1.

delay_months Delay to receiving the outcome, in months

outcome Delayed outcome type (all were scaled to $100)

prop_indiff Indifference point scaled to the maximum amount of each outcome. The maximum amount was the number of servings of each outcome worth $100.

Details

Note: The DD data shares the same indifference points used in the PD data. The PD data were created by using the DD data and using probabilities instead of delays. The PD was created to demonstrate features of the discAUC package and does not represent real data.

Source

doi: 10.1002/jeab.623

---

examp_PD *Probability discounting data*

Description

Probability discounting data with repeated measures for subjects across unlikely outcomes.

Usage

```
examp_PD
```

Format

A data frame with 360 rows and 4 variables:

subject Subject ID. Positive values are experimentally obtained. -987.987 are median indifference points. -1 and -2 values have indifference points of all 0 and all 1, respectively. These extra data were added for testing and debugging to ensure that AUC calculations will result in 0 when all indifference points are zero and 1 when all indifference points are 1.

prob Probability of receiving the outcome

outcome Delayed outcome type (all were scaled to $100)

prop_indiff Indifference point scaled to the maximum amount of each outcome. The maximum amount was the number of servings of each outcome worth $100.

Details

Note: The PD data shares the same indifference points used in the DD data. The PD data were created by using the DD data and using probabilities instead of delays. The PD was created to demonstrate features of the discAUC package and does not represent real data.

Source

doi: 10.1002/jeab.623

---

prep_AUC   *AUC data peperation helper function*

Description

Helper function to take AUC tibble and preprocess for other AUC calculations

Usage

```
prep_AUC(dat, indiff = NULL, x_axis, groupings = NULL, prob_disc = FALSE)
```

Arguments

dat	Discounting data tibble
indiff	Indifference points Variable
x_axis	Delays/probabilities/social distance variable
groupings	Variables for grouping (e.g., subject, expeirmental group) as a character or vector of characters
prob_disc	Boolean for probability discounting (MAYBE NOT NECESSARY PULLED OUT ODDS AGAINST TO DIFFERENT FUNCTION)

Value

Tibble that is grouped and arranged by groupings and x_axis (or x_axis_against, if probability discounting)

Examples

```
library(dplyr)

# Prep single set of data
DD <- tibble(
  delay = c(1 / 7, 1, 2, 4, 26, 52),
  indiff = c(95, 75, 50, 20, 5, 1)
) %>%
  arrange(delay)

prep_AUC(dat = DD, indiff = "indiff", x_axis = "delay")
# Prep multiple subject data

# Create DD data disorganize by delay
DD <- tibble(
  delay = rep(c(1 / 7, 1, 2, 4, 26, 52), 2),
  indiff = c(c(95, 75, 50, 20, 5, 1), c(95, 75, 50, 20, 5, 1) + .25),
  sub = c(rep(1, 6), rep(2, 6))
) %>%
  arrange(delay)
```

```
# Group by subject and organize by subject and delay
prep_AUC(dat = DD, indiff = "indiff", x_axis = "delay", groupings = "sub")


# Probability discounting with subjects and different outcomes

# Create PD data and disorganize by probability
PD <- tibble(
  prob = rep(c(.1, 1 / 100, 1 / 300, 1 / 750, 1 / 1000, 1 / 3000), 4),
  value = rep(c(c(95, 75, 50, 20, 5, 1), c(95, 75, 50, 20, 5, 1) + .25), 2),
  sub = rep(c(rep(1, 6), rep(2, 6)), 2),
  outcome = c(rep("money", 12), rep("cigarettes", 12))
) %>%
  arrange(prob)

# Calculate odds against, organize by subject, outcome, odds against
prep_AUC(PD,
  indiff = "value",
  x_axis = "prob",
  groupings = c("sub", "outcome"),
  prob_disc = TRUE
)
```

---

prep_log_AUC   *Calculate log x_axis values for AUClog*

Description

Calculate log x_axis values for AUClog

Usage

```
prep_log_AUC(
  dat,
  x_axis,
  log_base = 2,
  type = "adjust",
  correction = 1,
  dec_offset = TRUE
)
```

Arguments

dat	Discounting data tibble. `AUC_zeroes` should be run first if zero values on the x_axis will need to be included.
x_axis	Delays/probabilities/social distance variable
log_base	Base of the logarithm

type	Type of correction to handle 0 values on x_axis. Acceptable values are "corr"., "adjust", and "IHS". "Corr" adds a set value to each x_axis value and then takes the log of those values. "Adjust" implements increasing the x_axis values by the average difference between the log values on the x_axis. "IHS" calculates the inverse hyperbolic sine, which is different than the logarithm but is highly correlated with log transformed values. The IHS transformation does not require corrections
correction	If `type == "corr"` this value is what is added to the x_axis prior to taking the log values.
dec_offset	If `TRUE`, offsets the log x_axis values if the lowest non-zero x_axis value is a decimal. This calculation is preferred because if x_axis values are negative then the log values will be negative. The negative log values can cause inconsistencies in how AUC is calculated.

Value

Original data frame (a tibble) that includes an appended column with log scale version of x_axis

Correction types for handling zero x-axis values

"Corr" adds a set correction value to each x_axis value and then takes the log of those values. "Adjust" implements increasing the `x_axis` values by the average difference between the log values on the `x_axis`. "IHS" calculates the inverse hyperbolic sine for the `x_axis`, which is different than the logarithm but is highly correlated with log transformed values. The IHS transformation does not require adjustments because `IHS(0) == 0`.

Examples

```
prep_log_AUC(
  dat = examp_DD,
  x_axis = "delay_months",
  log_base = 10,
  dec_offset = TRUE,
  type = "adjust",
  correction = 1
)
```

---

prep_odds_against	*Calculate odds against values for* x_axis *for probability discounting*

Description

Calculate odds against values for x_axis for probability discounting

Usage

```
prep_odds_against(dat, x_axis, groupings = NULL)
```

Arguments

dat	Discounting data tibble. `AUC_zeroes` should be run first if zero values on the x_axis will need to be included.
x_axis	Probabilities distance variable
groupings	Variables for grouping (e.g., subject, expeirmental group) as a character or vector of characters

Value

Original data frame (a tibble) that includes an appended column odds against

Examples

```
prep_odds_against(examp_PD,
  "prob",
  groupings = c("subject", "outcome")
)
```

---

prep_ordinal	*Ordinal x-axis by grouping*

Description

Helper function to create ordinal values for x-axis variable. This helper function is designed to be used if the x-axis values are identical across every set of indifference points or if you desire ordinal x-axis values by subject. For the second case, if one subject was exposed to delays of 1 day and 1 month and a second subject was exposed to delays of 1 week and 1 month. In such a case, `prep_ordinal()` would return ordinal delays of (1, 2) for subject 1 and ordinal delays of (1, 2) for subject 2. If zeroes exist, will code as ordinal 0.

Usage

```
prep_ordinal(dat, x_axis, groupings = NULL, prob_disc = FALSE)
```

Arguments

dat	Discounting data tibble
x_axis	Delays/probabilities/social distance variable
groupings	Variables for grouping (e.g., subject, expeirmental group)
prob_disc	Boolean for probability discounting, if set to true function will calculate ordinals based on descending x_axis values which would be in line with increasing odds against.

Value

Tibble that has ordinal values for each x_axis value based on all possible x_axis values. Output tibble is arranged in the same order as original tibble.

Examples

```
library(dplyr)

PD <- tibble(
  prob = c(
    c(.05, 1 / 100, 1 / 300, 1 / 750, 1 / 1000, 1 / 3000),
    c(.1, 1 / 100, 1 / 300, 1 / 750, 1 / 1000, 1 / 4000)
  ),
  indiff = c(c(95, 75, 50, 20, 5, 1), c(95, 75, 50, 20, 5, 1) + .25),
  sub = c(rep(1, 6), rep(2, 6))
)

# Scramble data to demonstrate preserved original order
PD <- PD %>%
  mutate(scramble = rnorm(NROW(PD), 0, 1)) %>%
  arrange(scramble)

PD

prep_ordinal(PD, "prob", prob_disc = TRUE, "sub")
```

---

prep_ordinal_all   *Shared ordinal x-axis*

Description

Helper function to create ordinal values for x-axis variable. This helper function is designed to be used if the x-axis values are not identical across every set of indifference points. For example, if one subject was exposed to delays of 1 day and 1 month and a second subject was exposed to delays of 1 week and 1 month. In such a case, prep_ordinal_all() would return ordinal delays of (1, 3) for subject 1 and ordinal delays of (2, 3) for subject 2. If 0 exists, will be coded as 0.

Usage

```
prep_ordinal_all(dat, x_axis, prob_disc = FALSE)
```

Arguments

dat	Discounting data tibble
x_axis	Delays/probabilities/social distance variable
prob_disc	Boolean for probability discounting, if set to true function will calculate odrinals based on descending x_axis values which would be in line with increasing odds against.

Value

Tibble that has ordinal values for each x_axis value based on all possible x_axis values.

Examples

```
library(dplyr)

PD <- tibble(
  prob = c(
    c(.05, 1 / 100, 1 / 300, 1 / 750, 1 / 1000, 1 / 3000),
    c(.1, 1 / 100, 1 / 300, 1 / 750, 1 / 1000, 1 / 4000)
  ),
  indiff = c(c(95, 75, 50, 20, 5, 1), c(95, 75, 50, 20, 5, 1) + .25),
  sub = c(rep(1, 6), rep(2, 6))
)

prep_ordinal_all(PD, "prob", prob_disc = TRUE)
```

Index