Package: RoundingMatters (via r-universe)

September 30, 2024

Type Package

Title Tools for adjusting for rounding problems in metastudies about p-hacking and publication bias

Version 0.1.0

Author Sebastian Kranz, Peter Puetz

Maintainer Sebastian Kranz <sebastian.kranz@uni-ulm.de>

Description Tools for adjusting for rounding problems in metastudies about p-hacking and publication bias

License GPL (>= 2.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends tidyr, dplyr, restorepoint, ggplot2

Repository https://skranz.r-universe.dev

RemoteUrl https://github.com/skranz/RoundingMatters

RemoteRef main

RemoteSha ca9b1e2a086d7107167092bc112e26eb968de4cc

Contents

`absz.density` *Density estimates for absolute z-statistics assuming that z-statistics are symmetrically distributed around 0*

Description

Avoids downward bias at the left hand side where abs(z)=0.

Usage

```
absz.density(
  z,
  at = NULL,
  bw = 0.1,
  adjust = 1,
  kernel = "epanechnikov",
  n = 1024,
  weights = NULL,
  ...
)
```

Arguments

`z`	vector of z-statistics (or absolute z-statistics)
`at`	vector of points where density shall be evaluated. If NULL return a function (by calling `approxfun`) that allows evaluate the density at arbitrary points.
`bw`, `adjust`, `kernel`, `n`, `weights`, `...`	arguments passed to `stats::density`

`absz.density.ratio` *Perform kernel estimates of two densities of absolute z-statistics and their ratio.*

Description

Add by default bootstrap standard errors and confidence intervals

Usage

```
absz.density.ratio(
  z.num,
  z.denom,
  at,
  bootstrap = TRUE,
  B = 1000,
  ci.level = 0.95,
  bw = 0.1,
  kernel = "epanechnikov",
  return.as = c("long", "wide")[1],
  weights.num = NULL,
  weights.denom = NULL,
  ...
)
```

Arguments

`z.num`	observed z-statistics forming numerator density
`z.denom`	observed z-statistics forming denominator density
`at`	position where density shall be evaluated
`bootstrap`	if TRUE add bootstrap SE and CI for all measures
`B`	number of bootstrap repetitions
`ci.level`	Confidence level. Default 0.95.
`weights.num`	weights for z.num (optional)
`weights.denom`	weights for z.denom (optional)
`...`	arguments for absz.density

`as.perc` *Convert numbers like 0.421 to 42.1%*

Description

Convert numbers like 0.421 to 42.1%

Usage

```
as.perc(x, digits = 1)
```

Arguments

`x`	a vector of floating point numbers
`digits`	to how many decimal digits shall the percentage be rounded?

`deround.z.density.adjust` *Draw derounded z assuming missing digits of mu and sigma are uniformly distributed, but adjust for estimated density of z using rejection sampling*

Description

Draw derounded z assuming missing digits of mu and sigma are uniformly distributed, but adjust for estimated density of z using rejection sampling

Usage

```
deround.z.density.adjust(
  z.pdf,
  mu,
  sigma,
  mu.dec = pmax(num.deci(mu), num.deci(sigma)),
  sigma.dec = mu.dec,
  max.rejection.rounds = 10000,
  verbose = TRUE,
  just.uniform = rep(FALSE, length(mu)),
  z.min = 0,
  z.max = 5
)
```

Arguments

z.pdf	An estimated density of the derounded z-statistics (e.g. using only observations with many significant digits) normalized such that its highest values is 1. Best use make.z.pdf to create such a normalized pdf from a vector of observed z-statistics.
mu	Reported coefficient, possibly rounded
sigma	Reported standard error, possibly rounded.
mu.dec	Number of decimal places mu is reported to. Usually, we would assume that mu and sigma are rounded to the same number of decimal places. Since trailing zeros may not be detected, we set the default mu.dec=pmax(num.deci(mu),num.deci(sigma)).
sigma.dec	By default equal to mu.dec.
max.rejection.rounds	A limit how often the rejection sampler redraws to avoid an infinite loop.
verbose	If TRUE cat an r for each resampling draw to see how the function progresses.

---

deround.z.uniform *Draw derounded z assuming missing digits of mu and sigma are uniformly distributed*

Description

Draw derounded z assuming missing digits of mu and sigma are uniformly distributed

Usage

```
deround.z.uniform(
  mu,
  sigma,
  mu.dec = pmax(num.deci(mu), num.deci(sigma)),
  sigma.dec = mu.dec
)
```

Arguments

mu	Reported coefficient, possibly rounded
sigma	Reported standard error, possibly rounded.
mu.dec	Number of decimal places mu is reported to. Usually, we would assume that mu and sigma are rounded to the same number of decimal places. Since trailing zeros may not be detected, we set the default mu.dec=pmax(num.deci(mu),num.deci(sigma)).
sigma.dec	By default equal to mu.dec.

dsr.ab.df *Create an ab.df for the dsr approach*

Description

The resulting data frame is required for derounding b simulting rounding (dsr) approach. It contains a row for all considered combinations of z and s and window half-width h in h.seq. The columns share.below and share.above indicate which share of derounded z-statistics are inside the window and either fall below or above the threshold z0, respectively. Note that 1-share.above-share.below is the share of derounded z-statistics that fall outside the considered window.

Usage

```
dsr.ab.df(
  dat,
  h.seq = c(0.05, 0.075, 0.1, 0.2, 0.3, 0.4, 0.5),
  z0 = 1.96,
  min.n = 10000,
  min.rounds = 5,
  verbose = TRUE
)
```

Arguments

dat	the data frame that should contain at least the columns z and num.deci (number fo decimal places of mu and sigma, maximum of both)
h.seq	all considered window half-widths
z0	the significance threshold. Can be a single number or a vector with one element per row of dat.
min.n	how many z values shall be minimally rounded to compute the derounded z-distribution for each observation.
min.rounds	how many repetitions of rounding z-values shall there be at least (even if min.n is already reached).
verbose	Shall some progress information be shown? (This function can take a while).

dsr.mark.obs *Finds observations in dat for which we shall perform dsr adjustment*

Description

Adds to dat the logical columns dsr.adjust and dsr.compute. dsr.adjust==TRUE means that z-statistics of this observation will be adjusted by dsr. The adjustment statistics only depend on the reported z value and significant s of sigma. We thus don't need to compute the distribution for all rows with dsr.adjust==TRUE. If dsr.compute==TRUE we shall cmpute the derounded distribution for this observations.

Usage

```
dsr.mark.obs(
  dat,
  h.seq = c(0.05, 0.075, 0.1, 0.2, 0.3, 0.4, 0.5),
  z0 = 1.96,
  s.max = 100,
  no.deround = NULL
)
```

Arguments

dat	the data frame, must have columns z and s
h.seq	vector of considered windows half-width. We mark an observation for adjustment if it is at risk of missclassification, wrong inclusion, or wrong exclusion for any considered window size.
z0	the signficance threshold for z (default=1.96).
s.max	only mark observations for adjustment who have s <= s.max. Default value is 100.
no.deround	a logical vector indicating columns that shall never be derounded

---

make.z.pdf *Compute a normalized pdf from a vector of z-statistics*

Description

The PDF is normalized such that the point of highest density is 1

Usage

```
make.z.pdf(
  z,
  bw = 0.05,
  kernel = "gaussian",
  n = 512,
  dat,
  min.s = 100,
  z.min = 0,
  z.max = 5,
  show.hist = FALSE,
  ...
)
```

Arguments

z	a vector of z statistics. Usually, you would select all values from dat whose mu and sigma have sufficiently many significant digits
...	other parameters passed to stats::density

min.max.z — *Compute minimum and maximum possible values of z given rounded mu and sigma*

Description

Compute minimum and maximum possible values of z given rounded mu and sigma

Usage

```
## S3 method for class 'max.z'
min(
  mu,
  sigma,
  mu.dec = pmax(num.deci(mu), num.deci(sigma)),
  sigma.dec = mu.dec
)
```

Arguments

mu	Vector of reported estimated coefficients
sigma	Vector of reported standard errors
mu.dec	Number of reported decimal digits for mu. By default the maximum of the
long	if TRUE (default return results in a long format)

num.deci — *Get the number of significand digits of a floating point number using the character presentation of those numbers of R*

Description

Get the number of significand digits of a floating point number using the character presentation of those numbers of R

Usage

```
num.deci(x)
```

Arguments

x	a numeric vector

num.sig.digits — *Get the number of significand digits of a floating point number using the character presentation of those numbers of R*

Description

We assume that trailing zeros left of the decimal point are significant digits while trailing zeros right of the decimal point are not significant digits

Usage

```
num.sig.digits(x)
```

Arguments

x	a numeric vector

rightmost.sig.digit — *Get the last significant digit(s) of a floating point number*

Description

Get the last significant digit(s) of a floating point number

Usage

```
rightmost.sig.digit(x, r1 = 1, r2 = 1)
```

Arguments

x	The vector of floating point numbers
r1	Starting position from right
r2	Ending position from right

`rounding.risk.s.thresholds` *Compute thresholds for the significant s of the reported standard deviation such that we can rule-out the errors: misclassification, wrong inclusion, wrong exclusion*

Description

Compute thresholds for the significant s of the reported standard deviation such that we can rule-out the errors: misclassification, wrong inclusion, wrong exclusion

Usage

```
rounding.risk.s.thresholds(z, z0 = z0, h = 0.2)
```

Arguments

z	a vector of z statistics
z0	significance threshold. Can be a single number or a vector of length z
h	half-width of considered window around z0

Value

A data frame with the columns "z", "s.misclass", "s.include", "s.exclude" specifying for each z value the corresponding thresholds.

`rounding.risks` *Assess for observations with reported z-statistic z and a signficand of s for the standard error whether it is at risk of the errors: misclassification, wrong inclusion, wrong exclusion*

Description

Assess for observations with reported z-statistic z and a signficand of s for the standard error whether it is at risk of the errors: misclassification, wrong inclusion, wrong exclusion

Usage

```
rounding.risks(z, s, z0 = 1.96, h = 0.2)
```

Arguments

z	a vector of z statistics
s	vector of corresponding significands of the standard error
z0	significance threshold. Can be a single number like 1.96 or a vector of length z
h	half-width of considered window around z0

Value

A data frame with risk of missclassification information for each observations. We illustrate the columns for the misclassification risk: "s.misclass" is the threshold for the significand s above which we can rule out misclassification risk `risk.misclass = s < s.misclass` indicates whether the observation is at risk of misclassification `risk.misclass.below = risk.misclass & z < z0` indicates whether the observation is at risk of misclassification and below the significance threshold the other columns should be self-explainable given this info.

`rounding.risks.summary` *Summary statistics for rounding risks for different thresholds*

Description

Summary statistics for rounding risks for different thresholds

Usage

```
rounding.risks.summary(rr.dat, s.thresh = 0:100, long = TRUE)
```

Arguments

`rr.dat`	A data frame returned from a call to `rounding.risks`.
`long`	if TRUE (default return results in a long format)
`s.tresh`	a vector of considered s thresholds

`sample.uniform.z.deround` *Sample derounded z from the uniformely derounded distributon for a given single value of mu and sigma*

Description

Sample derounded z from the uniformely derounded distributon for a given single value of mu and sigma

Usage

```
sample.uniform.z.deround(
  n,
  mu,
  sigma,
  mu.dec = pmax(num.deci(mu), num.deci(sigma)),
  sigma.dec = mu.dec
)
```

Arguments

n	Number of sample draws
mu	Reported coefficient, possibly rounded
sigma	Reported standard error, possibly rounded.
mu.dec	Number of decimal places mu is reported to. Usually, we would assume that mu and sigma are rounded to the same number of decimal places. Since trailing zeros may not be detected, we set the default `mu.dec=pmax(num.deci(mu.round),num.deci(sigma.roun`
sigma.dec	By default equal to mu.dec.

---

set.last.digit.zero *Sets the last digit of a number x to zero*

Description

Sets the last digit of a number x to zero

Usage

```
set.last.digit.zero(x)
```

Arguments

x	a numeric vector

---

significand *Get the significands of a numeric vector using the character presentation of those numbers of R*

Description

The significand is the integer of all significand digits, e.g. the significand of 0.012 is 12.

Usage

```
significand(x, num.deci = NULL)
```

Arguments

x	a numeric vector.
num.deci	If not NULL a vector that states the number reported decimal places for x. This can be used if we know that there were addtional trailing zeros.

stat_abszdensity *ggplot2 density lines for absolute z-statistics assuming that they are symmetrically distributed around 0*

Description

Unlike normal [geom_density] or [stat_density] the density estimate does not go artificially decrease at the left bound 0. Note that this function only works nicely if the data starts left with 0. Possibly atoms at z=0 should ideally be removed.

Usage

```
stat_abszdensity(
  mapping = NULL,
  data = NULL,
  geom = "line",
  position = "stack",
  ...,
  bw = "nrd0",
  adjust = 1,
  kernel = "epanechnikov",
  n = 512,
  trim = FALSE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

bw	The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in [stats::bw.nrd()].
adjust	A multiplicate bandwidth adjustment. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, ‘adjust = 1/2‘ means use half of the default bandwidth.
kernel	Kernel. See list of available kernels in [density()].
n	number of equally spaced points at which the density is to be estimated, should be a power of two, see [density()] for details
trim	If ‘FALSE‘, the default, each density is computed on the full range of the data. If ‘TRUE‘, each density is computed over the range of that group: this typically means the estimated x values will not line-up, and hence you won’t be able to stack density values. This parameter only matters if you are displaying multiple densities in one plot or if you are manually adjusting the scale limits.

Computed variables

density density estimate

count density * number of points - useful for stacked density plots

scaled density estimate, scaled to maximum of 1

ndensity alias for ‘scaled‘, to mirror the syntax of [‘stat_bin()‘]

---

study.with.derounding *Analysis with derounded z-statistics for different window half-widths around z0*

Description

This is the main function you will call if you want to perform a publication bias / p-hacking analysis with derounded z-statistics. It allows flexible combinations of how a single derounded z vector is drawn, which statistics are computed for each combination of window h and derounded z-draw and how those statistics are aggregated over multiple replications.

Usage

```
study.with.derounding(
  dat,
  h.seq = c(0.05, 0.075, 0.1, 0.2, 0.3, 0.4, 0.5),
  window.fun = window.t.ci,
  mode = c("reported", "uniform", "zda", "dsr")[1],
  alt.mode = c("uniform", "reported")[1],
  make.z.fun = NULL,
  z0 = ifelse(has.col(dat, "z0"), dat[["z0"]], 1.96),
  repl = 1,
  aggregate.fun = "median",
  ab.df = NULL,
  z.pdf = NULL,
  max.s = 100,
  common.deci = TRUE,
  verbose = TRUE
)
```

Arguments

dat	a data frame containing all observations. Each observation is a test from a regression table in some article. It must have the columns mu (reported coefficient) and sigma (reported standard error). The optional column no.deround can specify rows whose z statistic shall never be derounded. dat can also have the columns z, num.deci, mu.deci and sigma.deci. If those columns do not exist, they will be computed from mu and sigma.
h.seq	All considered half-window sizes

window.fun	The function that computes for each draw of a derounded z vector and a window h the statistics of interest. Examples are window.t.ci (DEFAULT) or window.binom.test. Not that our implementation of dsr derounding (or any other derounding using ab.df) does not draw derounded z, but only creates a logical vector above indicating which draws are above or below the z0 threshold. This means if you write a custom function, it should essentially work on that vector.
mode	Mode how a single draw of derounded z is computed: "reported", "uniform","zda","dsr" or some custom name (requires ab.df to be defined)
alt.mode	Either "uniform" (DEFAULT) or "reported". Some derounding modes like "zda" and "dsr" cannot be well defined (or are too time-consuming to compute) for observations with many significant digits or outlier z-statistics. alt.mode specifies how z values shall be selected for those observations.
z0	The significance threshold for z
repl	Number of replications of each derounding draw.
aggregate.fun	How shall multiple replications be aggregated. Not yet implemented. Currently we always take the medians of each variale returned by window.fun of all replications.
ab.df	Required if mode=="dsr" or some custom mode. See e.g. dsr.ab.df.
z.pdf	Required if mode=="zda". Should be generated via make.z.pdf.
max.s	Used if mode=="zda". Specifies the maximum significand for which zda derounding shall be performed. For observations with larger significand s, uniform derounding will be performed.
common.deci	Shall we assume that mu and sigma are given with the same number of decimal places. If TRUE (Default) take the column num.deci i present in dat or create it as the pairwise maximum of the decimal places of mu and sigma. If FALSE, either use the columns mu.deci and sigma.deci if present in dat or generate them from mu and sigma.

---

window.binom.test *Apply on windows one-sided binomiminal test with H0: z <= z0*

Description

Apply on windows one-sided binomiminal test with H0: z <= z0

Usage

```
## S3 method for class 'binom.test'
window(above = z >= z0, h = NA, ci.level = 0.95, z, z0, ...)
```

`window.binom.test.2s` *Apply on windows two sided binomiminal test with H0: z = z0*

Description

Apply on windows two sided binomiminal test with H0: z = z0

Usage

```
## S3 method for class 'binom.test.2s'
window(above = z >= z0, h = NA, ci.level = 0.9, z, z0, ...)
```

`window.t.ci` *Window function returning estimated probability that a z-statistic is above a threshold z0 in a window with half-width h around z0 and t-test confidence intervals*

Description

Can be used as argument `window.fun` in `compute.with.derounding`

Usage

```
## S3 method for class 't.ci'
window(above = z >= z0, h = NA, ci.level = 0.95, z, z0, ...)
```

Index