Package ‘bdots’

November 10, 2022

Type Package

Title Bootstrapped Differences of Time Series

Version 1.2.0

Date 2022-11-10

Author Collin Nolte, Michael Seedorff, Jacob Oleson, Grant Brown, Joseph Cavanaugh, and Bob McMurray

Maintainer Collin Nolte <collin-nolte@uiowa.edu>

BugReports https://github.com/collinn/bdots/issues

Depends R (>= 4.0.0), data.table

Imports nlme, mvtnorm, parallel, stats, graphics, utils, ggplot2, gridExtra

LazyData TRUE


License GPL-3 | file LICENSE

URL https://github.com/collinn/bdots

NeedsCompilation no

Encoding UTF-8

RoxygenNote 7.2.1

Suggests knitr, rmarkdown, tinytest

VignetteBuilder knitr, rmarkdown

Repository CRAN

Date/Publication 2022-11-10 19:40:02 UTC
R topics documented:

ar1Solver ........................................... 3
bdotsBoot ........................................... 3
bdotsCorr ........................................... 6
bdotsFit ............................................. 6
bdotsFitter ......................................... 8
bdotsRefit .......................................... 9
bdRemove ........................................... 10
ci .................................................... 10
coef.bdotsObj ....................................... 11
coeffWriteout ....................................... 11
cohort_unrelated .................................... 12
createCurveList ..................................... 12
curveFitter ......................................... 13
df_cohort_unrelated ................................. 13
df_target ............................................ 14
doubleGauss ......................................... 14
doubleGauss2 ....................................... 15
effectiveAlpha_f .................................... 15
expCurve ............................................. 16
findModifiedAlpha .................................. 16
fwerAlpha .......................................... 17
getBootDist ......................................... 18
getFitCorforGroups ................................. 18
getSubCurveValues .................................. 19
linear ............................................... 19
logistic ............................................. 20
parTest2 ............................................ 20
plot.bdotsBootObj .................................. 21
plot.bdotsCorrObj .................................. 22
plot.bdotsObj ........................................ 22
polynomial ......................................... 23
print.bdotsBootObj .................................. 24
print.bdotsBootSummary ......................... 24
print.bdotsPars_ttest2 ............................. 25
print.bdotsSummary ................................ 25
p_adjust ............................................. 26
rbindlist.bdObjList ................................. 27
split.bdotsObj ....................................... 27
subset.bdotsBootObj ................................ 28
summary.bdotsBootObj .............................. 28
summary.bdotsObj ................................... 29
target ............................................... 29
writeCSV ........................................... 30

Index .................................................. 31
ar1Solver

*Compute AR1 correlation coefficient*

**Description**

Computes value for AR1 correlation coefficient for use in `p_adjust`.

**Usage**

```
ar1Solver(t)
```

**Arguments**

- `t` A numeric vector of t-statistics

**Value**

Estimated AR1 correlation coefficient

**See Also**

- `p_adjust`

**Examples**

```r
t <- rt(1e3, df = 1)
rho <- ar1Solver(t)
```

bdotsBoot

*Create bootstrapped curves from bdotsObj*

**Description**

Creates bootstrapped curves and performs alpha adjustment. Can perform "difference of difference" for nested comparisons.

**Usage**

```
bdotsBoot(
    formula,  # required
    bdObj,    # required
    Niter = 1000,  # default 1000
    alpha = 0.05,   # default 0.05
    padj = "oleson",  # default "oleson"
    cores = 0,  # not used
    ...     # not used
)
```
Arguments

- **formula**: See details.
- **bdObj**: An object of class `bdotsObj`
- **Niter**: Number of iterations of bootstrap to draw
- **alpha**: Significance level
- **padj**: Adjustment to make to pvalues for significance. Will be able to use anything from `p.adjust` function, but for now, just "oleson"
- **cores**: Number of cores to use in parallel. Default is zero, which uses half of what is available.
- ...: not used

Details

The formula is the only tricky part of this. There will be a minor update to how it works in the future. The three parts we will examine here are Groups, the LHS, and the RHS. For all variable names, special characters should be included with backticks, i.e., `my-var`.

### Groups

The Groups are the values input in `group` in the `bdotsFit` function, which are columns of the dataset used. These will be denoted $G_i$ Within each group, we will designate the unique values within each group as $v_j$, ..., whereby $G_i(v_1, v_2)$ will designate unique two unique values within $G_i$. The possible values of $v_i$ will be implied by the group with which they are associated.

For example, if we have groups `vehicle` and `color`, we could specify that we are interested in all red cars and trucks with the expression `vehicle(car, truck) + color(red)`.

### Formula

#### Bootstrapped difference of curves

This illustrates the case in which we are taking a simple bootstrapped difference between two curves within a single group.

If only one group was provided in `bdotsFit`, we can take the bootstrapped difference between two values within the group with

$y \sim \text{Group1}(val1, val2)$

If more than two groups were provided, we must specify within which values of the other groups we would like to compare the differences from Group1 in order to uniquely identify the observations. This would be

$y \sim \text{Group1}(val1, val2) + \text{Group2}(val1)$

For example, bootstrapping the differences between cars and trucks when `color` was provided as a second group, we would need $y \sim \text{vehicle}(car, truck) + \text{color}(red)$.

#### Bootstrapped difference of difference curves

This next portion illustrates the case in which we are interested in studying the difference between the differences between two groups, which we will call the `innerGroup` and the `outerGroup` following a nested container metaphor. Here, we must use caution as the order of these differences matter.

Using again the vehicle example, we can describe this in two ways:
1. We may be interested in comparing the difference between red trucks and cars (d_red) with the difference between blue trucks and cars (d_blue). In this case, we will be finding the difference between cars and trucks twice (one for blue, one for red). The vehicle type is the innerGroup, nested within the outerGroup, in this case, color.

2. We may also be interested in comparing the difference between red trucks and blue trucks (d_truck) with the difference between red and blue cars (d_car). Here, innerGroup is the color and outerGroup is the vehicle.

As our primary object of interest here is not the difference in outcome itself, but the difference of the outcome within two groups, the LHS of the formula is written \texttt{difs}(y, \texttt{Group1}(val1, val2)), where \texttt{Group1} is the innerGroup. The RHS is then used to specify the groups of which we want to take the outer difference of. The syntax here is the same as above. Together, then, the formula looks like
\[
difs(y, \texttt{Group1}(val1, val2)) \sim \texttt{Group2}(val1, val2)
\]
in the case in which only two grouping variables were provided to \texttt{bdotsFit} and
\[
difs(y, \texttt{Group1}(val1, val2)) \sim \texttt{Group2}(val1, val2) + \texttt{Group3}(val1) + \ldots
\]
is used to uniquely identify the sets of differences when three or more groups were provided.

**Value**

Object of class `bdotsBootObj`

**Examples**

```r
## Not run:

## fit <- bdotsFit(cohort_unrelated, ...)

boot1 <- bdotsBoot(formula = \texttt{difs}(
\texttt{Fixations, LookType(Cohort, Unrelated_Cohort)} \sim \texttt{Group}(50, 65),
bdObj = fit,
N.iter = 1000,
alpha = 0.05,
p.adj = "oleson",
cores = 4)

boot2 <- bdotsBoot(formula = \texttt{Fixations} \sim \texttt{Group}(50, 65) + \texttt{LookType(Cohort)},
bdObj = fit,
N.iter = 1000,
alpha = 0.05,
p.adj = "oleson",
cores = 4)

## End(Not run)
```
bdotsCorr  
___

**Correlation with fixed value in bdots**

**Description**

Find the correlation of a fixed value with the bdots fitted curves at each time point.

**Usage**

```
bdotsCorr(bdObj, val, ciBands = FALSE, method = "pearson")
```

**Arguments**

- `bdObj` Object of class ‘bdotsObj’
- `val` Character string of fixed value for correlation in dataset from ‘bdotsFit’
- `ciBands` Boolean for including confidence intervals
- `method` Arguments for ‘cor’ or ‘cor.test’. The default option us ‘method = "pearson"’

bdotsFit  
___

**Fit nlme curves to grouped observations**

**Description**

Creates observation level curves to use in bdotsBoot.

**Usage**

```
bdotsFit(
  data,
  subject,
  time,
  y,
  group,
  curveType = doubleGauss(concave = TRUE),
  cor = TRUE,
  numRefits = 0,
  cores = 0,
  verbose = FALSE,
  ...
)
```
bdotsFit

Arguments

data  
subject  
time  
y  
group  
curveType  
cor  
numRefits  
cores  
verbose  
...

Details

This is step one of the three step bdots process. Things should be more or less straight forward. The only tricky part involves curveType. For now know that one can use doubleGauss(concave = TRUE/FALSE) or logistic(). Should be passed in as a call. See the vignette on customizing this.

Value

Object of class 'bdotsObj', inherits from data.table

Examples

```r
## Not run:
res <- bdotsFit(data = cohort_unrelated,
                subject = "Subject",
                time = "Time",
                y = "Fixations",
                group = c("Group", "LookType"),
                curveType = doubleGauss(concave = TRUE),
                cor = TRUE,
                numRefits = 2,
                cores = 0,
                verbose = FALSE)

## End(Not run)
```
**bdotsFitter**

*Fits Individual Subject Curve*

---

**Description**

The one subject version of bdotsFit

**Usage**

```r
bdotsFitter(
  dat,  
  curveType, 
  rho,    
  numRefits = 0, 
  verbose, 
  getCovOnly = NULL, 
  params = NULL, 
  splitVars = NULL, 
  datVarNames = NULL, 
  ...
)
```

**Arguments**

- **dat** data for single subject/group combo
- **curveType** this is actually a function. Should rename
- **rho** correlation coefficient
- **numRefits** number of refit attempts
- **verbose** not used
- **getCovOnly** only find covariance matrix from starting parameter values
- **params** starting parameters, if wanting to add manually
- **splitVars** variables used to identify group. Might combine with datVarNames
- **datVarNames** character vector indicating response and time values from parent call
- **...** not used
bdotsRefit

Refit Observations Returned from bdotsFit

Description
Refit Observations Returned from bdotsFit

Usage

bdotsRefit(
  bdObj,
  fitCode = 1L,
  quickRefit = FALSE,
  numRefits = 2L,
  paramDT = NULL,
  ...
)

Arguments
bdObj An object of class 'bdotsObj' returned from bdotsFit
fitCode A length one integer indicating observations to refit. See Details
quickRefit A Boolean indicating if a quick refit should be used. If TRUE, rather than prompting
  the user for adjustments for each observation, bdotsRefit will jitter the parameters of all
  observations indicated by fitCode and attempt to refit. Between the original and the refitted
  curve, this will place priority on the higher fitCode. If these are equal, R2 will take precedence.
  Otherwise, the original fit will be kept.
numRefits Integer indicating the number of refit attempts after jittering parameters, either
  with quickRefit or when done individually
paramDT A data.table or data.frame that matches the what is returned by coefWriteout(bdObj).
  That is, it should have columns uniquely identifying observations with subjects and
  groups, as well as named columns for the parameters. NA parameters are OK. Can also be a
  subset of the original rows. Note, if this argument is not NULL, the remaining arguments will be ignored.
...

Details
fitCode indicates lower bound on observations to refit. For example, if fitCode = 4, bdotsRefit
will prompt user to refit all observations with fitCode = 4, 5, 6. The quickRefit option will attempt to
jitter and refit all observations selected by fitCode. Otherwise, the user will be prompted through
a menu to individually refit observations

Value
Returns bdObj with updated fits
bdRemove  

bdots Remove Function

Description

Remove observations with a specified fitCode and optionally all pairs

Usage

bdRemove(bdObj, fitCode = 6L, removePairs = TRUE)

Arguments

- **bdObj**: bdots object
- **fitCode**: min fitCode to remove. Default is 6, which removes all subjects with NULL fits (fitCode = 5 would remove 5 and 6)
- **removePairs**: Boolean. Remove subject pairs is one of pair is removed. Default is TRUE to retain paired t-test

Details

This function is used to remove all bdots observations with a fit code equal to or larger than the argument passed to fitCode without refitting. If removePairs = TRUE, all entries for a subject will be removed if their fit failed in any of the groups in which they were a member

---

**ci**

**ci dataset**

Description

ci dataset - need to include details

Usage

```r
  ci
```

Format

An object of class `data.frame` with 108216 rows and 5 columns.
coef.bdotsObj

---

**Extract bdotsFit Model Coefficients**

**Description**

Returns coefficient matrix for bdotsFit object

**Usage**

```r
## S3 method for class 'bdotsObj'
coef(object, ...)
```

**Arguments**

- `object` A bdotsObj
- `...` not used

**Value**

Returns matrix of model coefficients for observations in object

---

coefWriteout

---

**Create data.table with bdotsObj parameters**

**Description**

Creates an object of class data.table that matches parameter values for each observation. This can then be passed to the bdotsRefit function

**Usage**

`coefWriteout(bdObj)`

**Arguments**

- `bdObj` An object returned from bdotsFit or bdotsRefit

**Value**

A data.table matching parameter values to observations
Examples

```r
## Not run:
fit <- bdotsFit(data = cohort_unrelated,
subject = "Subject",
time = "Time",
y = "Fixations",
group = c("Group", "LookType"),
curveType = doubleGauss(concave = TRUE),
cor = TRUE,
numRefits = 2,
cores = 0,
verbose = FALSE)
parDT <- coefWriteout(fit)

## End(Not run)
```

cohort_unrelated  cohort_unrelated dataset

Description

cohort_unrelated dataset - need to include details

Usage

cohort_unrelated

Format

An object of class data.frame with 50100 rows and 6 columns.

createCurveList  Create old bdots curve list

Description

Create old bdots curve list

Usage

createCurveList(x, prs, splitGroups)

Arguments

- `x` list of group distributions
- `prs` named list of parsed call
- `splitGroups` splitGroups
curveFitter  

**Description**

Used in bdotsFit

**Usage**

```r
curveFitter(dat, ff, params, rho, numRefits = 0, getCovOnly = NULL, ...)
```

**Arguments**

- `dat` data used in building curve
- `ff` formula used in building curve
- `params` starting parameters
- `rho` correlation coefficient
- `numRefits` number of refit attempts
- `getCovOnly` only find covariance matrix from starting parameter values
- `...` don't know that this is used, can maybe get rid of it

**df_cohort_unrelated**  

**Description**

`df_cohort_unrelated` dataset - need to include details

**Usage**

```r
df_cohort_unrelated
```

**Format**

An object of class `data.frame` with 78156 rows and 5 columns.
Double Gauss curve function for nlme

Description

Double Gauss function used in fitting nlme curve for observations

Usage

doubleGauss(dat, y, time, params = NULL, concave = TRUE, ...)

Arguments

dat subject data to be used
y outcome variable, character vector
time time variable, character vector
params NULL unless user wants to specify starting parameters for gnlm
concave Boolean
...
just in case

Details

User should only have to worry about setting concavity of this function

\[ y \sim (\text{time} < \mu) \times \exp\left(-1 \times (\text{time} - \mu)^2 / (2 \times \text{sig1^2})\right) \times \left((\text{ht} - \text{base1}) + \text{base1}\right) + (\mu \leq \text{time}) \times \exp\left(-1 \times (\text{time} - \mu)^2 / (2 \times \text{sig2^2})\right) \times \left((\text{ht} - \text{base2}) + \text{base2}\right) \]
**DoubleGauss2 curve function for nlme**

**Description**

DoubleGauss2 function used in fitting nlme curve for observations

**Usage**

```r
doubleGauss2(dat, y, time, params = NULL, concave = TRUE, ...)
```

**Arguments**

- `dat` : subject data to be used
- `y` : outcome variable, character vector
- `time` : time variable, character vector
- `params` : NULL unless user wants to specify starting parameters for gnls
- `concave` : Boolean
- `...` : just in case

**Details**

User should only have to worry about setting concavity of this function. Presently only work for time series scaled out to 2000ms

\[
y \sim (\text{time} < \mu) \times \left( \exp(-1 \times (\text{time} - \mu)^2 / (2 \times \text{sigma1}^2)) \times (\text{ht} - \text{base1}) + \text{base1} \right) + (\mu \leq \text{time}) \times \left( \exp(-1 \times (\text{time} - \mu)^2 / (2 \times \text{sigma2}^2)) \times (\text{ht} - \text{base2}) + \text{base2} \right)
\]

---

**Effective Alpha Functional**

**Description**

Functional that returns function for computing effective alpha for given parameters and distribution

**Usage**

```r
effectiveAlpha_f(rho, n = 10, df = NULL, method = "norm")
```

**Arguments**

- `rho` : Correlation coefficient
- `n` : Number of observations
- `df` : Degrees of freedom if method = "t"
- `method` : Character string. Determines distribution for adjusted alpha can be either "norm" for normal distribution or "t" for t-dist
expCurve

__Exponential curve function__

**Description**

Exponential function used in fitting nlme curve for observations

**Usage**

\[
\text{expCurve}(\text{dat, y, time, params = NULL, ...})
\]

**Arguments**

- **dat** subject data to be used
- **y** outcome variable
- **time** time variable
- **params** NULL unless user wants to specify starting parameters for gnls
- **...** just in case

**Details**

Remove any values of zero, or jitter, before using with bdotsFit

\[
y \sim x_0 \exp(k \beta)
\]

findModifiedAlpha

__Find modified alpha__

**Description**

find modified alpha

**Usage**

\[
\text{findModifiedAlpha}(\rho, n, df, \\
alpha = 0.05, \\
errorAcc = 0.001, \\
gradDiff = ifelse(cores > 3, 0.5, 0.1), \\
cores = 0, \\
verbose = FALSE, \\
method = "t"
)
\]
**fwerAlpha**

**Arguments**

- `rho` correlation coefficient
- `n` number of observations
- `df` degrees of freedom if `method == "t"`
- `alpha` starting alpha from which to adjust
- `errorAcc` acceptable error for `alphastar`
- `gradDiff` gradient steps in algorithm
- `cores` number of cores. Default is zero, or half of what’s available
- `verbose` will probably remove this
- `method` either "t" or "norm"

**Description**

Family wise alpha calculation

**Usage**

`fwerAlpha(rho, k, n = 10)`

**Arguments**

- `rho` Correlation coefficient
- `k` Bounds of non-critical region
- `n` Number of observations

**Details**

Returns effective alpha, given number of tests and the correlation coefficient. This isn't explicitly checked, but there is no reason this function should take any non-scalar values. Derivation of this can be found on pg 12 of Jake’s 'Detecting time-specific differences'. This function performs the expression

\[
1 - P(I_t)P(I_t \mid I_{t-1})^{N-1}
\]
getBootDist

Create Distribution for Groups

Description

Create Distribution for Groups

Usage

getBootDist(x, b = 1000)

Arguments

x A subset object from bdObj
b Number of bootstraps

getFitCorforGroups

Get Fit Correlations

Description

Helper function for finding correlation of fixed value and fitted values within group

Usage

getFitCorforGroups(x, val, ciBands = FALSE, method = "pearson")

Arguments

x A split object of class ‘bdObj’ split by identifiers
val Fixed value from dataset
.ciBands boolean for including cibands
method method for correlation function
**getSubCurveValues**

*Return fitted values*

**Description**

Returns fitted values at observed times

**Usage**

```r
getSubCurveValues(bd, origNames = TRUE, origTime = TRUE)
```

**Arguments**

- `bd`: Single row of bdObj
- `origNames`: use original names for y and time, or use "y" and "time"
- `origTime`: Boolean. Do I actually want fitted values at observed times for that subject, or data.table with fitted values at the union of times

**Details**

Given a single row of bdObj, this returns fitted values at the observed times to use in conjunction with whatever else

---

**linear**

*Linear curve function*

**Description**

Linear function used in fitting nlme curve for observations

**Usage**

```r
linear(dat, y, time, params = NULL, ...)
```

**Arguments**

- `dat`: subject data to be used
- `y`: outcome variable
- `time`: time variable
- `params`: NULL unless user wants to specify starting parameters for gnlS
- `...`: just in case

**Details**

Don’t use this function please

\[ y \sim \text{slope} \times \text{time} + \text{intercept} \]
logistic  

*Logistic curve function for nlme*

**Description**

Logistic function used in fitting nlme curve for observations

**Usage**

```r
logistic(dat, y, time, params = NULL, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>subject data to be used</td>
</tr>
<tr>
<td>y</td>
<td>outcome variable</td>
</tr>
<tr>
<td>time</td>
<td>time variable</td>
</tr>
<tr>
<td>params</td>
<td>NULL unless user wants to specify starting parameters for gnls</td>
</tr>
<tr>
<td>...</td>
<td>just in case</td>
</tr>
</tbody>
</table>

**Details**

\[
y \sim \text{mini} + \frac{\text{peak} - \text{mini}}{1 + \exp(4 \times \text{slope} \times (\text{cross} - (\text{time})) / (\text{peak} - \text{mini})))}
\]

parTest2  

*Parameter t-test*

**Description**

Perform t-test on curve parameters of bdotsFit object

**Usage**

```r
parTest2(bdObj, group, vals = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdObj</td>
<td>Object of class bdObj</td>
</tr>
<tr>
<td>group</td>
<td>Length one character of grouping column in which to perform t-test</td>
</tr>
<tr>
<td>vals</td>
<td>Character vector of values within grouping column in which to perform the test. If NULL, it will do all pairwise tests</td>
</tr>
</tbody>
</table>

**Details**

Performs pairwise t-test. Currently only tests at alpha = 0.95. Also currently only allows t-test within single grouping column. Ability to test across grouping columns to come later
Value
List of t-test results of class {bdotsPars_ttest}

Examples

```r
## Not run:
res <- bdotsFit(data = cohort_unrelated,
subject = "Subject",
time = "Time",
y = "Fixations",
group = c("Group", "LookType"),
curveType = doubleGauss(concave = TRUE),
cor = TRUE,
numRefits = 2,
cores = 0,
verbose = FALSE)
tstats <- parTest(res, group = "LookType", vals = c("Cohort", "Unrelated_Cohort"))
## End(Not run)
```

plot.bdotsBootObj
Plot for object of class bdotsBootObj

Description
Allows a number of different but also unstable option for plotting an object of class bdotsBoot

Usage

```r
## S3 method for class 'bdotsBootObj'
plot(x, alpha = NULL, ciBands = TRUE, plotDiffs = TRUE, group = NULL, ...)
```

Arguments

- `x`: An object of class bdotsBootObj
- `alpha`: Significance level for plotting confidence intervals.
- `ciBands`: Boolean indicating whether or not to include confidence intervals around fitted curves
- `plotDiffs`: Boolean to plot difference curve
- `group`: Specify group to plot if difference of difference was used. The user can also subset the bdotsBootObj prior to plotting. Currently not used
- `...`: ignore for now, but will eventually allow plot parameters

Details
This plot function is also a bit unstable and is expected to change
plot.bdotsObj

Plot a bdotsFit object

Description
Plot individual fits or model fit parameters from an object of class 'bdotsObj'. These functions are not very stable.

Usage
## S3 method for class 'bdotsObj'
plot(x, fitCode = NULL, gridSize = NULL, plotfun = "fits", ...)

Arguments
- **x**: An object of class 'bdotsObj' returned from bdotsFit
- **fitCode**: Currently not used
- **gridSize**: Length one numeric indicating size of plot grid. Default is 2x2. For right now, they are square
- **plotfun**: Plot either subject fits or model parameters with "fits" or "pars"
- **...**: Ignore for now (other args to plot.generic)
Details

Right now, these functions are a bit unstable and expected to change. The largest current issue is with the placement of the legend, which cannot be adjusted. If you are running into issues with seeing things correctly, try making the "Plots" window in RStudio larger before running this function.

Value

This will return a list of all of the plots rendered.

Description

Polynomial function used in fitting nlme curve for observations.

Usage

polynomial(dat, y, time, degree, raw = TRUE, params = NULL, ...)

Arguments

dat subject data to be used
y outcome variable
time time variable
degree degree of polynomial
raw Boolean, use raw polynomials?
params NULL unless user wants to specify starting parameters for gnls
... just in case

Details

It’s recommended that one uses raw polynomials for this function for numerical stability. As inference is not performed on the parameters themselves, this should have minimal consequences.

\[ y \sim mini + (peak - mini) / (1 + \exp(4 \times slope \times (cross - (time)) / (peak - mini))) \]
print.bdotsBootObj

Print `bdotsBootObj`

Description
Prints argument. Really, just the summary function

Usage
```r
## S3 method for class 'bdotsBootObj'
print(x, ...)
```

Arguments
- `x` An object of class `bdotsBootObj`
- `...` Top secret alpha one code red

Details
Generic for printing `bdotsBootObj`

print.bdotsBootSummary

Print `bdotsBoot Summary`

Description
That’s pretty much it. This is a print method, so there is likely not much need to call it directly

Usage
```r
## S3 method for class 'bdotsBootSummary'
print(x, ...)
```

Arguments
- `x` generic name, but this will be an object of `bdotsBootSummary`
- `...` ignored for now
print.bdotsPars_ttest2

Print Parameter Test Summary

Description

Print Parameter Test Summary

Usage

```r
## S3 method for class 'bdotsPars_ttest2'
print(x, ...)
```

Arguments

- `x` object to be printed
- `...` not used

Details

That’s pretty much it. This is a print method, so there is likely not much need to call it directly

print.bdotsSummary

Print bdotsObj Summary

Description

Print bdotsObj Summary

Usage

```r
## S3 method for class 'bdotsSummary'
print(x, ...)
```

Arguments

- `x` object to be printed
- `...` not used

Details

That’s pretty much it. This is a print method, so there is likely not much need to call it directly
p_adjust

Adjust P-values for Multiple Comparisons

Description

Identical to stats::p.adjust, but includes method = "oleson"

Usage

p_adjust(p, method = "oleson", n = length(p), alpha = 0.05, df, rho, cores = 0)

Arguments

- **p**: numeric vector of p-values (possibly with NAs).
- **method**: correction method, a character string. Can be any of the methods in p.adjust.methods, with the additional value method = "oleson"
- **n**: number of comparisons, must be at least length(p); only set this (to non-default) when you know what you are doing!
- **alpha**: adjustment to be made with method oleson
- **df**: degrees of freedom, if using method = "oleson"
- **rho**: AR1 correlation coefficient, if using method = "oleson"
- **cores**: number of cores for use in parallel, only valid for method = "oleson". Default is zero, using half of the available cores

Details

This function works identically to the function p.adjust, with the additional option to use method = "oleson". For this option, user must include a value for df, alpha. If method = "oleson" and no value is given for rho, 0.9 will be used. To compute a value for rho from t-statistics, use ar1Solver.

Value

Returns a vector of adjusted p-values just as in p.adjust, but with additional attributes for alphastar and rho.

See Also

ar1Solver
**Description**

Similar to data.table::rbindlist, but preserves botsObjects attributes

**Usage**

```r
## S3 method for class 'bdObjList'
rbindlist(x, ...)
```

**Arguments**

- `x` : bdotsObject
- `...` : for compatability with data.table

---

**Description**

Analogous to other splitting functions, but retains necessary attributes across the split object. As of now, it can only be unsplit with bdots::rbindlist

**Usage**

```r
## S3 method for class 'bdotsObj'
split(x, f, drop = FALSE, by, ...)
```

**Arguments**

- `x` : Object of class bdotsObj
- `f` : For consistency with generic, but is not used
- `drop` : logical. Default FALSE will not drop empty list elements caused by factor levels not referred by that factor. Analogous to data.table::split
- `by` : Character vector of column names on which to split. Usually will be Subject or one of the fitted groups
- `...` : not used
subset.bdotsBootObj  Subset a nested group bdotsBoot objects

Description

Subset a nested group bdotsBoot objects

Usage

## S3 method for class 'bdotsBootObj'
subset(x, group, adjustAlpha = NULL, ...)

Arguments

- **x**: An object returned from bdotsBoot
- **group**: A group to subset. Must be an outer group
- **adjustAlpha**: currently not used. Will give option to recompute adjusted alpha
- **...**: Not used

Details

This function is used to subset a bdotsBootObject that was fit to compute the difference of differences. This allows the user to subset out the outer group in the comparison for plotting and investigation.

summary.bdotsBootObj  Summary for bdotsBootObj

Description

Provides summary information for bdotsBootObj

Usage

## S3 method for class 'bdotsBootObj'
summary(object, ...)

Arguments

- **object**: An object of class bdotsObj
- **...**: Ignored for now

Value

Returns an object of class "bdotsBootSummary". There is some summarized information included if assigned to an object, i.e., `summ <- summary(bdBootObj)` then `str(summ)`
Description

Provides summary information for bdotsObj

Usage

```r
## S3 method for class 'bdotsObj'
summary(object, ...)
```

Arguments

- `object`: An object of class bdotsObj
- `...`: not used

Value

Returns an object of class "bdotsSummary". There is some summarized information included if assigned to an object, i.e., ‘summ <- summary(bdObj)’ then ‘str(summ)’

target

Description

target dataset - need to include details

Usage

target

Format

An object of class `data.frame` with 25050 rows and 4 columns.
**writeCSV**

*Write fits from bdotsBoot to csv file*

---

**Description**

The function is used to write out columns for each group for which a curve was bootstrapped

**Usage**

```r
writeCSV(bootObj, file, alpha = 0.05, ...)
```

**Arguments**

- `bootObj` An object of class `bdotsBootObj`
- `file` file name to write out csv
- `alpha` alpha level for upper/lower CI
- `...` Other arguments passed to `data.table::fread`

**Details**

This is potentially useful for constructing plots in a separate application. There is an additional column, `Significant` indicating if a particular time point was considered significant between the difference curves. For difference of difference objects, this only indicates significance for the outer difference.
Index

* datasets
  - ci, 10
  - cohort_unrelated, 12
  - df_cohort_unrelated, 13
  - df_target, 14
  - target, 29

ar1Solver, 3, 26

bdotsBoot, 3
bdotsCorr, 6
bdotsFit, 6
bdotsFitter, 8
bdotsRefit, 9
bdRemove, 10

ci, 10
coef.bdotsObj, 11
coefWriteout, 11
cohort_unrelated, 12
createCurveList, 12
curveFitter, 13
df_cohort_unrelated, 13
df_target, 14
doubleGauss, 14
doubleGauss2, 15
effectiveAlpha_f, 15
expCurve, 16

findModifiedAlpha, 16
fwerAlpha, 17

getBootDist, 18
getFitCorforGroups, 18
getSubCurveValues, 19

linear, 19
logistic, 20
p_adjust, 3, 26

parTest2, 20
plot.bdotsBootObj, 21
plot.bdotsCorrObj, 22
plot.bdotsObj, 22
polynomial, 23
print.bdotsBootObj, 24
print.bdotsBootSummary, 24
print.bdotsPars_ttest2, 25
print.bdotsSummary, 25

rbindlist.bdObjList, 27

split.bdotsObj, 27
subset.bdotsBootObj, 28
summary.bdotsBootObj, 28
summary.bdotsObj, 29

target, 29

writeCSV, 30