Package ‘bdots’
March 27, 2021

Type Package
Title Bootstrapped Differences of Time Series
Version 1.0.1
Date 2021-03-12
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, mvtorm, parallel, stats, graphics, utils
LazyData TRUE


License GPL-3 | file LICENSE
URL https://github.com/collinn/bdots
NeedsCompilation no
Encoding UTF-8
RoxygenNote 7.1.1
Suggests knitr, rmarkdown, tinytest
VignetteBuilder knitr, rmarkdown
Repository CRAN
Date/Publication 2021-03-27 07:10:03 UTC

R topics documented:

ar1Solver ................................................................. 2
bdotsBoot ............................................................... 3
bdotsFit ................................................................. 5
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

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,
  bdObj,
  Niter = 1000,
  alpha = 0.05,
  padj = "oleson",
  cores = 0,
  ...
)

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.

## 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 blue 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 bootstraped 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

\[ y \sim \text{vehicle(car, truck) + color(red)}. \]

### 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 `difs(y, Group1(val1, val12))`, where Group1 is the innerGroup. The RHS is then used to specify the groups of which we want to take the inner difference of. The syntax here is the same as above. Together, then, the formula looks like

\[ \text{difs}(y, \text{Group1(val1, val12)}) \sim \text{Group2(val1, val12)} \]
bdotsFit 5

in the case in which only two grouping variables were provided to bdotsFit and
diffs(y,Group1(val1,val2)) ~ Group2(val1,val2) + Group3(val1) + ...is used to uniquely identify the sets of differences when three or more groups were provided.

Value

Object of class ‘bdotsBootObj’

Examples

## Not run:
## fit <- bdotsFit(cohort_unrelated, ...)

boot1 <- bdotsBoot(formula = diffs(Fixations, LookType(Cohort, Unrelated_Cohort)) ~ Group(50, 65),
bdObj = fit,
N.iter = 1000,
alpha = 0.05,
p.adj = "oleson",
cores = 4)

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

## End(Not run)

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,
bdotsFit

numRefits = 0,
cores = 0,
verbose = FALSE,
returnX = NULL,
...
)

Arguments

data  Dataset used
subject Column name of dataset containing subject identifiers
time  Column name containing time variable
y  Column name containing outcome of interest
group Character vector containing column names of groups. Can be greater than one
curveType See details/vignette
cor  Boolean. Autocorrelation?
numRefits Integer indicating number of attempts to fit an observation if the first attempt fails
cores  number of cores. Default is 0, indicating half cores available
verbose currently not used
returnX  Boolean. Return data with bdObj? Currently not implemented
...

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 function. See the vignette on customizing this.

Value

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

Examples

## 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)
bdotsFitter  Fits Individual Subject Curve

## End(Not run)

**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  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.
... not used

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**

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

Arguments

  object  A bdotsObj
  ...  not used

Value
Returns matrix of model coefficients for observations in object

c_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
c_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.

curveFitter

**Curve Fitter**

**Description**

Used in bdotsFit

**Usage**

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**  
*df_cohort_unrelated dataset*

---

**Description**

*df_cohort_unrelated dataset - need to include details*

**Usage**

*df_cohort_unrelated*

**Format**

An object of class `data.frame` with 78156 rows and 5 columns.

---

**df_target**  
*df_target dataset*

---

**Description**

*df_target dataset - need to include details*

**Usage**

*df_target*

**Format**

An object of class `data.frame` with 37575 rows and 4 columns.
**doubleGauss**

*Double Gauss curve function for nlme*

**Description**

Double Gauss function used in fitting nlme curve for observations

**Usage**

```r
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 gnls
- **concave**: Boolean
- **...**: just in case

**Details**

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

```r
y ~ (time < mu) * (exp(-1 * (time -mu)^2 / (2 * sig1^2)) * (ht-base1) + base1) +
    (mu <= time) * (exp(-1 * (time -mu)^2 / (2 * sig2^2)) * (ht-base2) + base2)
```

**doubleGauss2**

*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
\[ y \sim (\text{time} < \mu) \times (\exp(-1 \times (\text{time} - \mu) ^ 2 / (2 \times \sigma_1 ^ 2)) \times (h - \text{base}_1) + \text{base}_1) \times (\mu \leq \text{time}) \times (\exp(-1 \times (\text{time} - \mu) ^ 2 / (2 \times \sigma_2 ^ 2)) \times (h - \text{base}_2) + \text{base}_2) \]

---

**effectiveAlpha_f**  
**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

---

**findModifiedAlpha**  
**Find modified alpha**

**Description**

find modified alpha

**Usage**

```r
findModifiedAlpha(
  rho,
  n,
  df,
  alpha = 0.05,
  errorAcc = 0.001,
  gradDiff = ifelse(cores > 3, 0.5, 0.1),
  cores = 0,
  verbose = FALSE,
  method = "t"
)
```
The function `fwerAlpha` performs family-wise alpha calculation. It takes three arguments:

- `rho`: correlation coefficient
- `k`: bounds of non-critical region
- `n`: number of observations

The function returns an effective alpha, given the 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'. The function performs the following expression:

\[ 1 - P(I_t)P(I_t | I_{t-1})^{N-1} \]
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**

- `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**

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

---

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, plotDiffs = TRUE, group = NULL, ciBands = TRUE, ...)
```

**Arguments**

- `x`: An object of class bdotsBootObj
- `alpha`: Significance level for plotting confidence intervals. To readjust areas of significance will value different than alpha used in bdotsBoot is computationally expensive and is currently not an option but will be offered soon.
- `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
plot.bdotsObj

  ciBands  Boolean indicating whether or not to include confidence intervals around fitted
curves (currently only option is TRUE)
...
  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
**polynomial**

*Polynomial curve function for nlme*

**Description**

Logistic function used in fitting nlme curve for observations

**Usage**

```r
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.

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

---

**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.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**
```r
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
split.bdotsObj

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`

---

**rbindlist.bdotsObjList**  *rbindlist for bdotsObjects*

**Description**

Similar to `data.table::rbindlist`, but preserves `bdotsObject` attributes

**Usage**

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

**Arguments**

- `x` `bdotsObject`
- `...` for compatibility with `data.table`

---

**split.bdotsObj**  *Split object of class bdotsObj*

**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, ...)
```
Subset a nested group bdotsBoot objects

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. Anaglogous 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

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.

Usage

```r
## S3 method for class 'bdotsBootObj'
subset(x, group, adjustAlpha = NULL, ...)
```
**summary.bdotsBootObj**  
*Summary for bdotsBootObj*

**Description**

Provides summary information for bdotsBootObj

**Usage**

```r
## 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)`

---

**summary.bdotsObj**  
*Summary for bdotsObj*

**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)`
Description

target dataset - need to include details

Usage

target

Format

An object of class data.frame with 25050 rows and 4 columns.
Index

* datasets
  - ci, 9
  - cohort_unrelated, 11
  - df_cohort_unrelated, 12
  - df_target, 12
  - target, 23

- ar1Solver, 2, 20
- bdotsBoot, 3
- bdotsFit, 5
- bdotsFitter, 7
- bdotsRefit, 8
- bdRemove, 9

- ci, 9
- coef.bdotsObj, 10
- coefWriteout, 10
- cohort_unrelated, 11
- curveFitter, 11

- df_cohort_unrelated, 12
- df_target, 12
- doubleGauss, 13
- doubleGauss2, 13
- effectiveAlpha_f, 14

- findModifiedAlpha, 14
- fwerAlpha, 15

- logistic, 16

- p_adjust, 3, 19
- plot.bdotsBootObj, 16
- plot.bdotsObj, 17
- polynomial, 18
- print.bdotsBootSummary, 18
- print.bdotsSummary, 19

- rbindlist.bdObjList, 20

- split.bdotsObj, 20
- subset.bdotsBootObj, 21
- summary.bdotsBootObj, 22
- summary.bdotsObj, 22

- target, 23