Package 'turfR'

February 20, 2015

Type Package

Title TURF Analysis for R

Version 0.8-7

Date 2014-12-01

Description Package for analyzing TURF (Total Unduplicated Reach and Frequency) data in R. No looping in TURF algorithm results in fast processing times. Allows for individual-level weights, depth specification, and user-truncated combination set(s). Allows user to substitute Monte Carlo simulated combination set(s) after set(s) exceed a user-specified limit.

License GPL (>= 2)

Depends R (>= 3.0.0), dplyr (>= 0.3.0)

LazyData true

Author Jack Horne [aut, cre]

Maintainer Jack Horne <jack@jackhorne.net>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-02 07:51:45

R topics documented:

turfR-package *TURF Analysis for R*

Description

Package for analyzing TURF (Total Unduplicated Reach and Frequency) data in R. No looping in TURF algorithm results in fast processing times. Allows for individual-level weights, depth specification, and user-truncated combination set(s). Allows user to substitute Monte Carlo simulated combination set(s) after set(s) exceed a user-specified limit.

See `help(turf)` for additional details and references.

turf *TURF Analysis for R*

Description

Calculate weighted reach and frequency statistics from TURF data for all possible combinations of *n* choose *k*, for user-specified combination sets, or for Monte Carlo simulated subsets of combinations.

Usage

```
turf(data, n, k, combos, ...)
```

Arguments

`data`	Required. Literal character string representing name of a file in the working directory readable using `read.table(data, header=TRUE)`, or name of a data frame or matrix in R containing TURF data. Rows are individuals (respondents). Columns are (1) respondent identifier, (2) a weight variable, and a minimum of *n* columns containing only zeroes and ones, each representing an individual item in the TURF algorithm. Respondent identifiers need not be unique and weights need not sum to the total number of rows. In the absence of any weight variable, substitute a column of ones. Ones in the remaining columns indicate that the reach criterion was met for a given item by a given individual. Values other than zero or one in these columns (including `NA`) trigger an error. `data` may contain more than $n + 2$ columns, but any columns in addition to that number will be ignored.
`n`	Required. Scalar indicating the number of items to be included in the TURF algorithm; $0 < n <$ (`ncol(data)` - 1). Non-integer values are coerced using `floor(`*n*`)`.
`k`	Required. Vector of length 1 to *n* containing any values 1 to *n* indicating the combination sizes to be evaluated by the TURF algorithm. Non-integer values are coerced using `floor(`*k*`)`.

combos — Optional. List of combination sets to be evaluated by the TURF algorithm, such as that generated by `turf.combos`. Individual combination sets are *p* x *n* matrices, containing only zeroes and ones indicating items to be included in a given combination. Rows (*p*) correspond to combinations evaluated; columns (*n*) correspond to items, and must be in the same order as the items columns in `data`. Each *i*th element of `combos` should contain combinations of the size specified in the *i*th position of *k*; `length(combos)` must be equal to `length(k)`. See "details" for additional information on usage.

... — Optional. Additional arguments controlling behavior and output of the TURF algorithm. See `turf.args`. Arguments indicated here must match named arguments in `turf.args`.

Details

TURF algorithm is as originally described by Miaoulis, *et al*. (1990) and outputs *reach* and *frequency* statistics for each combination evaluated.

Reach = sum(weights[reached]) / sum(weights), where an individual is considered "reached" if the reach criterion is met for at least the number of items indicated by the `depth` argument in `turf.args`. See Markowitz (2005) for a more detailed explanation of *depth of reach*.

Frequency = sum(weights x items_reached) / sum(weights). Frequency includes all individuals, whether reached or not. Frequency among "reached" individuals may be calculated as Frequency / Reach regardless of weights.

When the `combos` argument is omitted from the call, combination sets are generated automatically using `turf.combos` and information passed to it by *n*, *k* and any additional arguments passed from `turf.args`. The user may also pass any combination sets (e.g., a user-truncated set) developed in or out of R, so long as the structure of the combination sets corresponds to that described above.

Monte Carlo simulated combination sets, when requested, are generated using the procedure described by Adler, *et al*. (2010). Sets of *n* choose *k* that result in small numbers of combinations should be run without the Monte Carlo simulation option since they require little processing time (e.g., 12 choose 7 = 792 combinations and will run in less than 1 second). Typically, even larger sets will run in a reasonable amount of time without subsetting (e.g., 18 choose 10 = 43,758 combinations and will run in less than a minute on most computers), but RAM may become a limiting factor, especially when large numbers of individuals are combined with problems involving large numbers of items. The default for substituting Monte Carlo simulated subsets of combinations is set at 10,000. For the above reasons, and because subsetting can lead to duplicate combinations especially when the original set is small, care should be taken to not subset combinations until the original set becomes unmanageably large.

The lack of looping in the algorithm permits processing at about 1,000 combinations per second.

Value

R object:

A list of 2 elements

turf — A list of `length(k)` elements, each of which is comprised of a matrix whose columns are *reach* (`rchX`), *frequency* (`frqX`), and *n* indicator variables representing the items included in a given combination. Rows represent combinations

evaluated. Sorting and the number of rows returned are controlled by the `sort` and `keep` arguments passed from `turf.args`.

`call` The call to `turf` as a literal character string

Written to Console:

Combination size(s) evaluated, as *k* of *n*; time in seconds required to evaluate all combinations of a given size; total time in seconds required to complete the function call.

References

Adler, T.J., Smith, C. & Dumont, J. 2010. Optimizing product portfolios using discrete choice modeling and TURF. In: S. Hess, A. Daly (Eds), *Choice modeling: the state-of-the-art and the state-of-practice; proceedings from the Inaugural International Choice Modeling Conference*. Emerald Publishing Group Ltd., pp. 485-497.

Krieger, A.M. & Green, P.E. 2000. Turf revisited: Enhancements to toal unduplicated reach and frequency analysis. *Marketing Research*, 12, 30-36.

Markowitz, L. 2005. Going beyond TURF to complement and extend existing product lines. *Ipsos-Insight*, November 2005.

Miaoulis, G., Free, V. & Parsons, H. 1990. TURF: A new planning approach for product line extensions. *Marketing Research*, March, pp. 28-40.

Examples

```
##Example 1
##Evaluate all combinations of 3, 4, 5 and 6 items drawn from 10 items.
data(turf_ex_data)
ex1 <- turf(turf_ex_data, 10, 3:6)

##Output to Console:
##3 of 10: 0.105068 sec
##4 of 10: 0.1420949 sec
##5 of 10: 0.1511021 sec
##6 of 10: 0.1160791 sec
##total time elapsed: 0.518347 sec


##Example 2
##Pass additional arguments
data(turf_ex_data)
ex2 <- turf(turf_ex_data, 10, 3:6, depth=2, keep=20, mc=TRUE, nsims=100)

##Output to Console:
##3 of 10: 0.03802586 sec
##4 of 10: 0.03702521 sec
##5 of 10: 0.0380249 sec
##6 of 10: 0.03802609 sec
##total time elapsed: 0.156105 sec
```

```
##Example 3
##Customize combos, exclude item 10 from all combinations of size 3
data(turf_ex_data)
psims <- colSums(turf_ex_data[,-c(1:2)]) / sum(turf_ex_data[,2])
psims <- psims / sum(psims)
combos <- turf.combos(10, 3, mc=TRUE, nsims=100, psims=psims)
combos[[1]] <- combos[[1]][-which(combos[[1]][,10]==1),]
ex3 <- turf(turf_ex_data, 10, 3, combos)

##Output to Console:
##3 of 10: 0.02001405 sec
##total time elapsed: 0.02001405 sec
```

`turf.args` *Arguments used in TURF Analysis*

Description

INTERNAL: A list of named arguments that can be passed into `turf` and `turf.combos` to control the behavior and output of the TURF algorithm.

Usage

```
turf.args(depth=1L, keep=0, mc=FALSE, nsims=10000, psims=NULL, sort="d")
```

Arguments

`depth`	Scalar indicating *depth of reach*, see Markowitz (2005). `depth` is an integer, greater than or equal to 1; non-integer values are coerced using `floor(depth)`.
`keep`	Scalar indicating the number of combinations of each size to keep in the output object generated by `turf`. `keep` is an integer, greater than or equal to zero; non-integer values are coerced using `floor(keep)`. If `keep` == 0 (default), *all* combinations evaluated by the TURF algorithm are kept; if `keep` >= 1, the lesser of `keep` or the number of evaluated combinations of a given size are kept.
`mc`	Logical indicating whether or not Monte Carlo simulated subsets of combinations should be substituted for all possible combinations of n choose k or user-specified combinations. When TRUE, subsets are generated using the method described by Adler, *et al*. (2010), only when `nsims` is also less than n choose k or the number of user-specified combinations.
`nsims`	Scalar indicating the number of combination sets of a given size to generate when `mc` == TRUE. `nsims` is an integer, greater than or equal to 1; non-integer values are coerced using `floor(nsims)`. This argument is ignored when `mc` == FALSE.
`psims`	Vector of length n indicating the probabilities of sampling each item into a combination set when `mc` == TRUE. Probabilities need not sum to 1 (see `sample` for

additional details). All values must be greater than or equal to 0; for combinations of size *k*, there must be at least *k* non-zero values in `psims`. Default varies depending on calling function; when called from `turf`, default is the probability of each item being reached; when called from `turf.combos`, default is `rep(1/n, n)`. `psims` is ignored when `mc` == FALSE.

`sort` Character indicating sort order of TURF output: "d" (default) causes combinations to be sorted in descending order of reach, then frequency; "a" causes combinations to be sorted in ascending order of reach, then frequency; "n" leaves combinations unsorted (i.e., they remain in the same order as the rows in each element of `combos`). `sort` is executed before `keep`. When `sort` == "n", `keep` is ignored (i.e., all combinations evaluated are kept in the output). No other values are permitted.

Details

Defaults set by this function may be modified by specifying any of the arguments as name-value pairs in place of the `(...)` argument when calling `turf` or `turf.combos`. It is never necessary to call `turf.args` directly.

Value

A list consisting of the named arguments.

References

Adler, T.J., Smith, C. & Dumont, J. 2010. Optimizing product portfolios using discrete choice modeling and TURF. In: S. Hess, A. Daly (Eds), *Choice modeling: the state-of-the-art and the state-of-practice; proceedings from the Inaugural International Choice Modeling Conference*. Emerald Publishing Group Ltd., pp. 485-497.

Krieger, A.M. & Green, P.E. 2000. Turf revisited: Enhancements to toal unduplicated reach and frequency analysis. *Marketing Research*, 12, 30-36.

Markowitz, L. 2005. Going beyond TURF to complement and extend existing product lines. *Ipsos-Insight*, November 2005.

Miaoulis, G., Free, V. & Parsons, H. 1990. TURF: A new planning approach for product line extensions. *Marketing Research*, March, pp. 28-40.

Examples

```
##INTERNAL USE ONLY
```

turf.combos *Combination sets used in TURF analysis*

Description

Generate combination sets to be used in TURF analysis, formatted for use in `turf`. Can be used to generate all possible combinations of *n* choose *k* or Monte Carlo simulated subsets of combinations involving the same *n* items.

Usage

```
turf.combos(n, k, ...)
```

Arguments

`n`	Required. Scalar indicating the number of items to be included in the TURF algorithm; $0 < n <$ `(ncol(data) - 1)`. Non-integer values are coerced using `floor(`*n*`)`.
`k`	Required. Vector of length 1 to *n* containing any values 1 to *n* indicating the combination sizes to be evaluated by the TURF algorithm. Non-integer values are coerced using `floor(`*k*`)`.
`...`	Optional. Additional arguments controlling behavior and output of the TURF algorithm. See `turf.args`. Arguments indicated here must match named arguments in `turf.args`.

Details

Generally, it is only necessary to call `turf.combos` independently of `turf` if the user wishes to manually modify combination sets that will be evaluated. See `turf` and `turf.args` for additional information on usage.

Value

A list of `length(`*k*`)` elements, each of which is comprised of a *p* x *n* matrix of zeroes and ones. Rows (*p*) are individual combinations to be evaluted by the TURF algorithm. Columns (*n*) correspond to items and must be in the same order as the items variables in `data` submitted to TURF. Ones indicate that an item is included in a given combination. Each *i*th element contains only combinations of the size indicated by the *i*th position of *k*.

References

Adler, T.J., Smith, C. & Dumont, J. 2010. Optimizing product portfolios using discrete choice modeling and TURF. In: S. Hess, A. Daly (Eds), *Choice modeling: the state-of-the-art and the state-of-practice; proceedings from the Inaugural International Choice Modeling Conference*. Emerald Publishing Group Ltd., pp. 485-497.

Krieger, A.M. & Green, P.E. 2000. Turf revisited: Enhancements to toal unduplicated reach and frequency analysis. *Marketing Research*, 12, 30-36.

Markowitz, L. 2005. Going beyond TURF to complement and extend existing product lines. *Ipsos-Insight*, November 2005.

Miaoulis, G., Free, V. & Parsons, H. 1990. TURF: A new planning approach for product line extensions. *Marketing Research*, March, pp. 28-40.

Examples

```
##Example 1
##Generate all combinations of 3, 4, 5 and 6 items drawn from 10 items.
ex1 <- turf.combos(10, 3:6)
colSums(ex1[[2]])

##Example 2
##Pass additional arguments
data(turf_ex_data)
psims <- colSums(turf_ex_data[,-c(1:2)]) / sum(turf_ex_data[,2])
psims <- psims / sum(psims)
ex2 <- turf.combos(10, 4, mc=TRUE, nsims=100, psims=psims)
colSums(ex2[[1]])
```

---

turf_ex_data *Example data set for TURF*

Description

Data used in examples and to test functionality of turfR.

Usage

```
turf_ex_data
```

Format

A data frame containing 180 observations (rows) and 12 variables (columns), all numeric. Columns are `respid`: a unique row identifier; `weight`: an individual-level weight variable that sums to 180 (i.e., the number of rows, mean = 1); and `item_1` to `item_10`: 10 binary-coded (0/1) variables representing TURF data for 10 items.

Source

Generated using `runif` and related functions.

Index