Package ‘ARPobservation’

August 29, 2016

Title Tools for Simulating Direct Behavioral Observation Recording Procedures Based on Alternating Renewal Processes

Description Tools for simulating data generated by direct observation recording. Behavior streams are simulated based on an alternating renewal process, given specified distributions of event durations and interim times. Different procedures for recording data can then be applied to the simulated behavior streams. Functions are provided for the following recording methods: continuous duration recording, event counting, momentary time sampling, partial interval recording, and whole interval recording.

Version 1.1

Maintainer James E. Pustejovsky <jepusto@gmail.com>

Author James E. Pustejovsky, with contributions from Daniel M. Swan

License GPL-3

VignetteBuilder knitr

Suggests plyr, reshape2, ggplot2, knitr, testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-11 07:38:57

R topics documented:

`ARPobservation` *ARPobservation*

Description

Tools for simulating different methods of observing alternating renewal processes

Details

ARPobservation provides a set of tools for simulating data based on direct observation of behavior. It works by first simulating a behavior stream based on an alternating renewal process, using specified distributions of event durations and interim times. Different procedures for recording data can then be applied to the simulated behavior stream.

The main function for simulating a behavior stream is `r_behavior_stream`. Currently, the event duration and interim time distributions must come from the class `eq_dist`. (See the documentation for this class for distributions that are currently implemented.)

Several different observation recording procedures can then be applied as filters to a simulated behavior stream. The following procedures are currently implemented:

- `continuous_duration_recording`
- `momentary_time_recording`
- `event_counting`
- `interval_recording`

To apply multiple procedures to the same behavior stream, use `reported_observations`. Data can also be simulated using the convenience functions `r_PIR`, `r_WIR`, `r_MTS`, `r_continuous_recording`, and `r_event_counting`. These functions wrap the behavior-stream generation step and the observation recording step into a single function. They are more memory efficient, but slightly less computationally efficient, than executing each step in turn.

Author(s)

James E. Pustejovsky <jepusto@gmail.com>

`augmented_recording` *Applies augmented interval recording to a behavior stream*

Description

Divides the observation session into intervals. Each interval is scored using partial interval recording, whole interval recording, and momentary time sampling (at the beginning of the following interval). The sum of the three scores is then recorded.

Usage

```
augmented_recording(BS, interval_length, rest_length = 0)
```

Arguments

`BS`	object of class behavior_stream
`interval_length`	time length of each interval.
`rest_length`	portion of each interval to exclude from observation. Default is zero. See details.

Details

Each behavior stream is divided into intervals of length `interval_length`. The last `rest_length` of each interval is excluded from observation. For example, for a stream length of 100, `interval_length = 20`, and `rest_length = 5`, the first interval runs from [0,15), the second interval runs from [20,35), etc.

Value

A matrix with rows equal to the number of intervals per session and columns equal to the number of behavior streams in `BS`.

Examples

```
BS <- r_behavior_stream(n = 5, mu = 3, lambda = 10,
                        F_event = F_exp(), F_interim = F_exp(), stream_length = 100)
augmented_recording(BS, interval_length = 20)
```

continuous_duration_recording *Applies continuous duration recording to a behavior stream*

Description

Calculates the proportion of session time during which behavior occurs.

Usage

```
continuous_duration_recording(BS)
```

Arguments

`BS`	object of class `behavior_stream`

Value

Vector of proportions.

Examples

```
BS <- r_behavior_stream(n = 5, mu = 3, lambda = 10,
                        F_event = F_exp(), F_interim = F_exp(), stream_length = 100)
continuous_duration_recording(BS)
```

Dunlap *Dunlap et al.(1994) data*

Description

Single case design data measured with partial interval recording from a study of the effect of providing Choice between academic activities on the disruptive behavior of three elementary school students with emotional and behavioral disorders. For this data "No Choice" is the baseline phase. Data were extracted from the figures in the publicaton.

Usage

```
Dunlap
```

Format

A data frame with 58 observations on 7 variables

,1 `Case` The participant for whom the observation took place

,2 `Phase` The level of the observation ("Choice" vs. "No Choice")

,3 `Session` The observation session # for each participant

,4 `outcome` The summary PIR measurement for the observation session

,5 `active_length` The length of the active observation interval, in seconds

,6 `rest_length` The length of the recording interval, in seconds

,7 `intervals` The total number of intervals in the observation session

References

Dunlap, G., DePerczel, M., Clarke, S., Wilson, D., Wright, S., White, R., & Gomez, A. (1994). Choice making to promote adaptive behavior for students with emotional and behavioral challenges. Journal of Applied Behavior Analysis, 27 (3), 505-518.

`eq_dist` *Constructor for class* `eq_dist`

Description

The `eq_dist` class consists of a pair of component functions for generating random variates from a specified distribution and the corresponding equilibrium distribution.

Usage

```
eq_dist(r_gen, r_eq)
```

Arguments

`r_gen`	function for generating random deviates.
`r_eq`	function for generating random deviates from the corresponding equilibrium distribution.

Details

Both functions must take arguments `n` and `mean`. Currently, the following distributions are implemented:

- `F_exp` - Exponential
- `F_gam` - Gamma
- `F_gam_mix` - Mixture of two gammas
- `F_weib` - Weibull
- `F_unif` - Uniform
- `F_const` - Constant

Value

Object of class `eq_dist` with components `r_gen` and `r_eq`.

`event_counting` *Applies event counting to a behavior stream*

Description

Calculates the number of behaviors that begin during the observation session.

Usage

```
event_counting(BS)
```

Arguments

`BS`	object of class `behavior_stream`

Value

Vector of non-negative integers.

Examples

```
BS <- r_behavior_stream(n = 5, mu = 3, lambda = 10,
                        F_event = F_exp(), F_interim = F_exp(), stream_length = 100)
event_counting(BS)
```

`F_const` *Constant (degenerate) distribution and related equilibrium distribution*

Description

Generation from a degenerate distribution and random number generation from the related equilibrium distribution, for use with `r_behavior_stream`.

Usage

```
F_const()
```

Value

Object of class `eq_dist` with components `r_gen` and `r_eq`.

The function `r_gen(n, mean)` simply returns a vector of length `n` with all values equal to `mean`.

The function `r_eq(n, mean)` generates random deviates from a uniform distribution on the interval (0, mean).

Examples

```
hist(F_const()$r_gen(1000, 2))
hist(F_const()$r_eq(1000, 2))
```

F_exp — *Exponential distribution and related equilibrium distribution*

Description

Random number generation from exponential distributions, for use with `r_behavior_stream`.

Usage

```
F_exp()
```

Value

Object of class `eq_dist` with components `r_gen` and `r_eq`.

The function `r_gen(n, mean)` generates random deviates from an exponential distribution with specified `mean`.

The function `r_eq(n, mean)` generates random deviates from an exponential distribution with specified `mean`.

Examples

```
hist(F_exp()$r_gen(1000, 3))
hist(F_exp()$r_eq(1000, 3))
```

F_gam — *Gamma distribution and related equilibrium distribution*

Description

Random number generation from a gamma distribution and the related equilibrium distribution, for use with `r_behavior_stream`.

Usage

```
F_gam(shape)
```

Arguments

`shape`	shape parameter

Value

Object of class `eq_dist` with components `r_gen` and `r_eq`.

The function `r_gen(n, mean)` generates random deviates from a gamma distribution with specified `mean` and `shape` parameters.

The function `r_eq(n, mean)` generates random deviates from the equilibrium distribuion corresponding to the gamma distribution with specified `mean` and `shape` parameters.

Examples

```
hist(F_gam(2)$r_gen(1000, 3))
hist(F_gam(2)$r_eq(1000, 3))
```

`F_gam_mix` *Mixture of two gamma distributions and related equilibrium distribution*

Description

Random number generation from a mixture of two gamma distributions and the related equilibrium distribution, for use with `r_behavior_stream`.

Usage

```
F_gam_mix(shape1, shape2, scale_ratio, mix)
```

Arguments

`shape1`	shape parameter for first mixture component, k_1
`shape2`	shape parameter for second mixture component, k_2
`scale_ratio`	ratio of first scale component to second scale component, θ_1/θ_2
`mix`	mixing proportion of first component, p

Value

Object of class `eq_dist` with components `r_gen` and `r_eq`.

The function `r_gen(n, mean)` generates random deviates from a mixture of two gamma distributions with specified `mean`, `shape1`, `shape2`, `scale_ratio`, and `mix`. The cumulative distribution function is given by

$$F(x) = p\Gamma(x; k_1, \theta_1) + (1 - p)\Gamma(x; k_2, \theta_2),$$

where $\Gamma(x; k, \theta)$ is the cumulative distribution function of a Gamma random variable with shape k and scale θ, and the scale parameters are determined by the specified `mean` and `scale_ratio`.

The function `r_eq(n, mean)` generates random deviates from the equilibrium distribuion corresponding to the mixture of gamma distributions.

Examples

```
hist(F_gam_mix(2, 2, 1 / 12, 3 / 5)$r_gen(1000, 20))
hist(F_gam_mix(2, 2, 1 / 12, 3 / 5)$r_eq(1000, 20))
```

F_unif *Uniform distribution and related equilibrium distribution*

Description

Random number generation from a uniform distribution and the related equilibrium distribution, for use with `r_behavior_stream`.

Usage

```
F_unif()
```

Value

Object of class `eq_dist` with components `r_gen` and `r_eq`.

The function `r_gen(n, mean)` generates random deviates from a uniform distribution with specified mean μ on the interval $(0, 2\mu)$. The cumulative distribution function is given by $F(x) = x/2\mu$.

The function `r_eq(n, mean)` generates random deviates from the equilibrium distribuion corresponding to a uniform distribution on the interval $(0, 2\mu)$. The cumulative distribution function is given by

$$F(x) = x(4\mu - x)/(4\mu^2).$$

Examples

```
hist(F_unif()$r_gen(1000, 2))
hist(F_unif()$r_eq(1000, 2))
```

F_weib *Weibull distribution and related equilibrium distribution*

Description

Random number generation from a Weibull distribution and the related equilibrium distribution, for use with `r_behavior_stream`.

Usage

```
F_weib(shape)
```

Arguments

shape	shape parameter

Value

Object of class eq_dist with components r_gen and r_eq.

The function r_gen(n, mean) generates random deviates from a Weibull distribution with specified mean and shape parameters.

The function r_eq(n, mean) generates random deviates from the equilibrium distribuion corresponding to the Weibull distribution with specified mean and shape parameters.

Examples

```
hist(F_gam(2)$r_gen(1000, 3))
hist(F_gam(2)$r_eq(1000, 3))
```

---

incidence_bounds *Incidence bounds and confidence interval*

Description

Calculates a bound for the log of the incidence ratio of two samples (referred to as baseline and treatment) based on partial interval recording (PIR) data, assuming that the behavior follows an Alternating Renewal Process.

Usage

```
incidence_bounds(PIR, phase, base_level, mu_U, p, active_length,
  intervals = NA, conf_level = 0.95, exponentiate = FALSE)
```

Arguments

PIR	vector of PIR measurements
phase	factor or vector indicating levels of the PIR measurements.
base_level	a character string or value indicating the name of the baseline level.
mu_U	the upper limit on the mean event duration
p	upper limit on the probability that the interim time between behavioral events is less than the active interval
active_length	length of the active observation interval
intervals	the number of intervals in the sample of observations. Default is NA
conf_level	Coverage rate of the confidence interval. Default is .95.
exponentiate	Logical value indicating if the log of the bounds and the confidence interval should be exponentiated. Default is FALSE.

Details

The incidence ratio estimate is based on the assumptions that 1) the underlying behavior stream follows an Alternating Renewal Process, 2) the average event duration is less than `mu_U`, and 3) the probability of observing an interim time less than the active interval length is less than `p`.

The `PIR` vector can be in any order corresponding to the factor or vector `phase`. The levels of `phase` can be any two levels, such as "A" and "B", "base" and "treat", or "0" and "1". If there are more than two levels in `phase` this function will not work. A value for `base_level` must be specified - if it is a chaaracter string it is case sensitive.

For all of the following variables, the function assumes that if a vector of values is provided they are constant across all observations and simply uses the first value in that vector.

`mu_U` is the upper limit on the mean event durations. This is a single value assumed to hold for both samples of behavior

`active_length` This is the total active observation length. If the intervals are 15 seconds long but 5 seconds of each interval is reserved for recording purposes, `active_length= 10`. Times are often in seconds, but can be in any time unit.

`intervals` is the number of intervals in the observations. This is a single value and is assumed to be constant across both samples and all observations. This value is only relevant if the mean of one of the samples is at the floor or ceiling of 0 or 1. In that case it will be used to truncate the sample mean. If the sample mean is at the floor or ceiling and no value for `intervals` is provided, the function will stop.

Value

A list containing two named vectors and a single named number. The first entry, `estimate_bounds`, contains the lower and upper bound for the estimate of the incidence ratio. The second entry, `estimate_SE`, contains the standard error of the estimate. The third entry, `estimate_CI`, contains the lower and upper bounds for the confidence interval of the incidence ratio.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# Estimate bounds on the incidence ratio for Ahmad from the Dunlap dataset
data(Dunlap)
with(subset(Dunlap, Case == "Ahmad"),
incidence_bounds(PIR = outcome, phase = Phase, base_level = "No Choice",
             mu_U = 10, p = .15, active_length = active_length, intervals = intervals))
```

interim_bounds *Interim bounds and confidence interval*

Description

Calculates a bound for the log of the ratio of interim time of two samples (referred to as baseline and treatment) based on partial interval recording (PIR) data, assuming that the average event durations are equal across samples and that interim times are exponentially distributed.

Usage

```
interim_bounds(PIR, phase, base_level, conf_level = 0.95, intervals = NA,
  exponentiate = FALSE)
```

Arguments

`PIR`	vector of PIR measurements
`phase`	factor or vector indicating levels of the PIR measurements.
`base_level`	a character string or value indicating the name of the baseline level.
`conf_level`	Desired coverage rate of the calculated confidence interval. Default is `.95`.
`intervals`	the number of intervals in the sample of observations. Default is `NA`
`exponentiate`	Logical value indicating if the log of the bounds and the confidence interval should be exponentiated. Default is `FALSE`.

Details

The interim ratio estimate is based on the assumptions that 1) the underlying behavior stream follows an Alternating Renewal Process, 2) the average event durations in each sample are equal, and 3) interim times follow exponential distributions.

The `PIR` vector can be in any order corresponding to the factor or vector `phase`. The levels of `phase` can be any two levels, such as "A" and "B", "base" and "treat", or "0" and "1". If there are more than two levels in `phase` this function will not work. A value for `base_level` must be specified; if it is a chaaracter string it is case sensitive.

`intervals` is the number of intervals in the observations. This is a single value and is assumed to be constant across both samples and all observations. If intervals is sent as a vector instead of a single value, the first value in the vector will be used. This value is only relevant if the mean of one of the samples is at the floor or ceiling of 0 or 1. In that case it will be used to truncate the sample mean. If the sample mean is at the floor or ceiling and no value for `intervals` is provided, the function will stop.

Value

A list with three named entries The first entry, `estimate_bounds`, contains the lower and upper bound for the estimate of the prevalence ratio. The second entry, `estimate_SE`, contains the standard errors for the upper and lower bounds. The third entry, `estimate_CI`, contains the lower and upper bounds for the confidence interval of the prevalence ratio.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# Estimate bounds on the interim time ratio for Carl from the Moes dataset
data(Moes)
with(subset(Moes, Case == "Carl"),
interim_bounds(PIR = outcome, phase = Phase, base_level = "No Choice"))
```

`interval_recording` *Applies interval recording to a behavior stream*

Description

Divides the observation session into a specified number of intervals. For partial interval recording, each interval is scored according to whether the behavior is present at any point during the interval. For whole interval recording, each interval is scored according to whether the behavior is present for the duration.

Usage

```
interval_recording(BS, interval_length, rest_length = 0, partial = TRUE,
  summarize = TRUE)
```

Arguments

`BS`	object of class behavior_stream
`interval_length`	time length of each interval.
`rest_length`	portion of each interval to exclude from observation. Default is zero. See details.
`partial`	logical value indicating whether to use partial interval recording (`TRUE`) or whole interval recording (`FALSE`).
`summarize`	logical value indicating whether vector of moments should be summarized by taking their mean.

Details

Each behavior stream is divided into intervals of length `interval_length`. The last `rest_length` of each interval is excluded from observation. For example, for a stream length of 100, `interval_length = 20`, and `rest_length = 5`, the first interval runs from [0,15), the second interval runs from [20,35), etc.

Value

If `summarize = FALSE`, a matrix with rows equal to the number of intervals per session and columns equal to the number of behavior streams in `BS`. If `summarize = TRUE`, a vector of proportions of length equal to the number of behavior streams in `BS`.

Examples

```
BS <- r_behavior_stream(n = 5, mu = 3, lambda = 10,
                        F_event = F_exp(), F_interim = F_exp(), stream_length = 100)
interval_recording(BS, interval_length = 20, partial = TRUE, summarize = FALSE)
interval_recording(BS, interval_length = 20, partial = TRUE, summarize = TRUE)
colMeans(interval_recording(BS, 20, partial = TRUE, summarize = FALSE))
interval_recording(BS, interval_length = 20, rest_length = 5, partial = FALSE)
```

`logRespRatio` *Calculate log-response ratio, variance, and confidence interval*

Description

Estimates the log-response ratio (with or without bias correction), the variance of the log-response ratio, and the confidence interval for a given confidence level.

Usage

```
logRespRatio(observations, phase, base_level, conf_level = 0.95,
  bias_correct = TRUE, exponentiate = FALSE)
```

Arguments

`observations`	Vector of observations
`phase`	Factor or vector indicating levels of the PIR measurements.
`base_level`	a character string or value indicating the name of the baseline level.
`conf_level`	Desired coverage rate of the calculated confidence interval. Default is `.95`.
`bias_correct`	Logical value indicating if the bias-corrected log-response ratio should be used. Default is `TRUE`
`exponentiate`	Logical value indicating if the log-respones ratio should be exponentiated.

Details

The `observations` vector can be in any order corresponding to the factor or vector `phase`. The levels of `phase` can be any two levels, such as "A" and "B", "base" and "treat", or "0" and "1". If there are more than two levels in `phase` this function will not work. A value for `base_level` must be specified - if it is a chaaracter string it is case sensitive. If `exponentiate = TRUE`, the log-ratio and the confidence interval will be exponentiated, but the variance will be excluded from the output.

Value

If `exponentiate = FALSE`, a list with three named entries. The first entry, `lRR`, is the estimated log-response ratio. The second entry, `V_lRR`, is the estimated variance of the log-response ratio. The third entry, `CI`, is a vector containing the endpoints of a confidence interval of `conf_level` coverage rate.

If `exponentiate = TRUE`, a list with two named entries. The first entry, `RR`, is the estimated response ratio. The second entry, `CI`, is a vector containing the endpoints of a confidence interval of `conf_level` coverage rate.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# Estimate the log response ratio and its variance for Carl from Moes dataset
data(Moes)
with(subset(Moes, Case == "Carl"),
logRespRatio(observations = outcome, phase = Phase, base_level = "No Choice"))
```

`Moes` *Moes(1998) data*

Description

Single-case design data from a study that using partial interval recording (PIR) examining the impact of choice-making in a homework tutoring context on disruptive behavior. In this data "No Choice" is the baseline phase. Data were extracted from the figure in the publication.

Usage

```
Moes
```

Format

A data frame with 80 observations on 7 variables

,1 `Case` The participant for whom the observation took place

,2 `Phase` The level of the observation ("Choice" vs. "No Choice")

,3 `Session` The observation session # for each participant

,4 `outcome` The summary PIR measurement for the observation session

,5 `active_length` The length of the active observation interval, in seconds

,6 `rest_length` The length of the recording interval, in seconds

,7 `intervals` The total number of intervals in the observation session

References

Moes, D. R. (1998). Integrating choice-making opportunities within teacher-assigned academic tasks to facilitate the performance of children with autism. Research and Practice for Persons with Severe Disabilities, 23 (4), 319-328.

momentary_time_recording — *Applies momentary time recording to a behavior stream*

Description

Evaluates the presence or absence of the behavior at fixed moments in time.

Usage

```
momentary_time_recording(BS, interval_length, summarize = TRUE)
```

Arguments

`BS`	object of class behavior_stream
`interval_length`	length of interval between moments.
`summarize`	logical value indicating whether vector of moments should be summarized by taking their mean.

Value

If `summarize = FALSE`, a matrix with length `n_intervals + 1` and width equal to the number of behavior streams in `BS`. If `summarize = TRUE`, a vector of proportions of length equal to the number of behavior streams in `BS`. Note that if `summarize = TRUE`, the initial state of the behavior stream is excluded when calculating the mean, so the proportion is based on `n_intervals` values.

Examples

```
BS <- r_behavior_stream(n = 5, mu = 3, lambda = 10,
                        F_event = F_exp(), F_interim = F_exp(), stream_length = 100)
momentary_time_recording(BS, interval_length = 20, FALSE)
momentary_time_recording(BS, interval_length = 20)
colMeans(momentary_time_recording(BS, 20, FALSE)[-1,])
```

PIR_loglik — *Calculate log-likelihood*

Description

Calculates the log-likelihood of within-session PIR data

Usage

```
PIR_loglik(phi, zeta, U, c, d)
```

Arguments

`phi`	value for prevalence
`zeta`	value for incidence
`U`	a vector containing interval-level PIR data
`c`	the length of the active interval
`d`	the length of the recording interval

Details

`phi` must be a value between 0 and 1, inclusive of 0 and 1. \ codezeta must be a value greater than 0.

The vector `U` should only contain values of 1 or 0.

`c` must be some positive value in whatever time units the observation took place in (typically seconds). `d` must be some non-negative value - a `d` of zero represents a PIR observation where no time was set aside for recording.

Value

The value of the log-likelihood

Author(s)

Daniel Swan <dswan@utexas.edu>

`PIR_MOM` *Moment estimator for prevalence and incidence, with bootstrap confidence intervals*

Description

Estimates prevalance and incidence for two samples, along with the ratios of each parameter, assuming that the behavior follows an ‘. Also provides boostrap confidence intervals.

Usage

```
PIR_MOM(PIR, phase, base_level, intervals, interval_length, rest_length = 0,
  Bootstraps = 2000, conf_level = 0.95, exponentiate = FALSE,
  seed = NULL)
```

Arguments

`PIR`	vector of PIR measurements
`phase`	factor or vector indicating levels of the PIR measurements.
`base_level`	a character string or value indicating the name of the baseline level.
`intervals`	the number of intervals in the sample of observations
`interval_length`	the total length of each interval
`rest_length`	length of the portion of the interval devoted to recording. Default is `0`
`Bootstraps`	desired number of bootstrap replicates. Default is `2000`
`conf_level`	Desired coverage rate of the calculated confidence interval. Default is `.95`.
`exponentiate`	a logical indicating whether the row corresponding to the ratio of treatment to baseline should be exponentiated, with the default as `FALSE`.
`seed`	seed value set in order to make bootstrap results reproducible. Default is `null`

Details

The moment estimators are based on the assumption that the underlying behavior stream follows an Alternating Poisson Process, in which both the event durations and interim times are exponentially distributed.

Value

A dataframe with six columns and three rows corresponding to baseline, treatment, and the log ratio or ratio (depending upon the value of `exponentiate`) of treatment to baseline

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# Estimate prevalence and incidence ratios for Carl from the Moes dataset
data(Moes)
with(subset(Moes, Case == "Carl"),
PIR_MOM(PIR = outcome, phase = Phase, intervals = intervals,
interval_length = (active_length + rest_length), rest_length = rest_length,
base_level = "No Choice", seed = 149568373))
```

`prevalence_bounds` *Prevalence bounds and confidence interval*

Description

Calculates a bound for the log of the prevalence ratio of two samples (referred to as baseline and treatment) based on partial interval recording (PIR) data, assuming that the behavior follows an Alternating Renewal Process.

Usage

```
prevalence_bounds(PIR, phase, base_level, mu_L, active_length, intervals = NA,
  conf_level = 0.95, exponentiate = FALSE)
```

Arguments

`PIR`	vector of PIR measurements
`phase`	factor or vector indicating levels of the PIR measurements.
`base_level`	a character string or value indicating the name of the baseline level.
`mu_L`	the lower limit on the mean event duration
`active_length`	length of the active observation interval
`intervals`	the number of intervals in the sample of observations. Default is `NA`.
`conf_level`	Coverage rate of the confidence interval. Default is `.95`.
`exponentiate`	Logical value indicating if the log of the bounds and the confidence interval should be exponentiated. Default is `FALSE`.

Details

The prevalence ratio estimate is based on the assumptions that 1) the underlying behavior stream follows an Alternating Renewal Process and 2) the average event duration is greater than `mu_L`.

The `PIR` vector can be in any order corresponding to the factor or vector `phase`. The levels of `phase` can be any two levels, such as "A" and "B", "base" and "treat", or "0" and "1". If there are more than two levels in `phase` this function will not work. A value for `base_level` must be specified - if it is a chaaracter string it is case sensitive.

For all of the following variables, the function assumes that if a vector of values is provided they are constant across all observations and simply uses the first value in that vector.

`mu_L` is the lower limit on the mean event durations. This is a single value assumed to hold for both samples of behavior

`active_length` This is the total active observation length. If the intervals are 15 seconds long but 5 seconds of each interval is reserved for recording purposes, `active_length= 10`. Times are often in seconds, but can be in any time unit.

`intervals` is the number of intervals in the observations. This is a single value and is assumed to be constant across both samples and all observations. This value is only relevant if the mean of one of the samples is at the floor or ceiling of 0 or 1. In that case it will be used to truncate the sample mean. If the sample mean is at the floor or ceiling and no value for `intervals` is provided, the function will stop.

Value

A list with three named entries. The first entry, `estimate_bounds`, contains the lower and upper bound for the estimate of the prevalence ratio. The second entry, `estimate_SE`, contains the standard error of the estimate. The third entry, `estimate_CI`, contains the lower and upper bounds for the confidence interval of the prevalence ratio.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# Estimate bounds on the prevalence ratio for Carl from Moes dataset
data(Moes)
with(subset(Moes, Case == "Carl"),
 prevalence_bounds(PIR = outcome, phase = Phase, base_level = "No Choice",
 mu_L = 10, active_length = active_length, intervals = intervals))
```

`reported_observations` *Applies multiple recording procedures to a behavior stream*

Description

This is a convenience function that allows multiple recording procedures to be applied to a single behavior stream. Results are reported either per behavior stream or as summary statistics, averaged over multiple behavior streams.

Usage

```
reported_observations(BS, data_types = c("C", "M", "E", "P", "W"),
  interval_length = 1, rest_length = 0, n_aggregate = 1)
```

Arguments

`BS`	object of class behavior_stream
`data_types`	list of recording procedures to apply to the behavior stream. See details.
`interval_length`	time length of each interval used to score momentary time recording and interval recording procedures.
`rest_length`	portion of each interval to exclude from observation for interval recording. See documentation for `interval_recording`.
`n_aggregate`	number of observations over which to calculate summary statistics.

Details

The following recording procedures are currently implemented

- `C` - continuous duration recording
- `M` - momentary time recording
- `E` - event counting
- `P` - partial interval recording
- `W` - whole interval recording

Value

If `n_aggregate = 1`, a data frame with one column per procedure listed in `data_types` and length equal to the number of behavior streams in `BS`. If `n_aggregate > 1`, a list containing two data frames: one with sample means and one with sample variances, both taken across `n_aggregate` behavior streams.

Examples

```
BS <- r_behavior_stream(n = 50, mu = 3, lambda = 10,
                        F_event = F_exp(), F_interim = F_exp(), stream_length = 100)
reported_observations(BS, interval_length = 10)
reported_observations(BS, interval_length = 10, n_aggregate = 5)
```

`r_behavior_stream` *Generates random behavior streams*

Description

Random generation of behavior streams (based on an alternating renewal process) of a specified length and with specified mean event durations, mean interim times, event distribution, and interim distribution.

Usage

```
r_behavior_stream(n, mu, lambda, F_event, F_interim, stream_length,
  equilibrium = TRUE, p0 = 0, tuning = 2)
```

Arguments

`n`	number of behavior streams to generate
`mu`	vector of mean event durations
`lambda`	vector of mean interim time
`F_event`	distribution of event durations. Must be of class `eq_dist`.
`F_interim`	distribution of interim times. Must be of class `eq_dist`.
`stream_length`	length of behavior stream

`equilibrium`	logical; if `TRUE`, then equilibrium initial conditions are used; if `FALSE`, then `p0` is used to determine initial state and normal generating distributions are used for event durations and interim times.
`p0`	vector of initial state probabilities. Only used if `equilibrium = FALSE`, in which case default is zero (i.e., behavior stream always starts with an interim time).
`tuning`	controls the size of the chunk of random event durations and interim times. Adjusting this may be useful in order to speed computation time .

Details

Generates behavior streams by repeatedly drawing random event durations and random interim times from the distributions as specified, until the sum of the durations and interim times exceeds the requested stream length. The vectors `mu`, `lambda`, and `p0` are recycled to length `n`.

Value

An object of class `behavior_stream` containing two elements.

Examples

```
# default equilibrium initial conditions
r_behavior_stream(n = 5, mu = 3, lambda = 10,
                  F_event = F_exp(), F_interim = F_exp(),
                  stream_length = 100)

# non-equilibrium initial conditions
r_behavior_stream(n = 5, mu = 3, lambda = 10,
                  F_event = F_gam(3), F_interim = F_gam(3),
                  stream_length = 100,
                  equilibrium = FALSE, p0 = 0.5)
```

`r_continuous_recording` *Generates random samples of continuously recorded behavior streams*

Description

Random generation of behavior streams (based on an alternating renewal process) of a specified length and with specified mean event durations, mean interim times, event distribution, and interim distribution, summarized as the total proportion of time the behavior of interest occurred.

Usage

```
r_continuous_recording(n, mu, lambda, stream_length, F_event, F_interim,
  equilibrium = TRUE, p0 = 0, tuning = 2)
```

Arguments

`n`	number of behavior streams to generate
`mu`	mean event duration
`lambda`	mean interim time
`stream_length`	length of behavior stream
`F_event`	distribution of event durations. Must be of class `eq_dist`.
`F_interim`	distribution of interim times. Must be of class `eq_dist`.
`equilibrium`	logical; if `TRUE`, then equilibrium initial conditions are used; if `FALSE`, then `p0` is used to determine initial state and normal generating distributions are used for event durations and interim times.
`p0`	Initial state probability. Only used if `equilibrium = FALSE`, in which case default is zero (i.e., behavior stream always starts with an interim time).
`tuning`	controls the size of the chunk of random event durations and interim times. Adjusting this may be useful in order to speed computation time .

Details

Generates behavior streams by repeatedly drawing random event durations and random interim times from the distributions as specified, until the sum of the durations and interim times exceeds the requested stream length. Then applies a continuous recording filter to the generated behavior streams.

Value

A vector of proportions of length `n`.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
r_continuous_recording(n = 5, mu = 2, lambda = 4, stream_length = 20,
                       F_event = F_exp(), F_interim = F_exp())
```

`r_event_counting` *Generates random samples of event counts*

Description

Random generation of behavior streams (based on an alternating renewal process) of a specified length and with specified mean event durations, mean interim times, event distribution, and interim distribution, summarized as the the total number of behaviors that began during the recording session

Usage

```
r_event_counting(n, mu, lambda, stream_length, F_event, F_interim,
  equilibrium = TRUE, p0 = 0, tuning = 2)
```

Arguments

`n`	number of behavior streams to generate
`mu`	mean event duration
`lambda`	mean interim time
`stream_length`	length of behavior stream
`F_event`	distribution of event durations. Must be of class `eq_dist`.
`F_interim`	distribution of interim times. Must be of class `eq_dist`.
`equilibrium`	logical; if `TRUE`, then equilibrium initial conditions are used; if `FALSE`, then `p0` is used to determine initial state and normal generating distributions are used for event durations and interim times.
`p0`	Initial state probability. Only used if `equilibrium = FALSE`, in which case default is zero (i.e., behavior stream always starts with an interim time).
`tuning`	controls the size of the chunk of random event durations and interim times. Adjusting this may be useful in order to speed computation time .

Details

Generates behavior streams by repeatedly drawing random event durations and random interim times from the distributions as specified, until the sum of the durations and interim times exceeds the requested stream length. Then applies an event counting filter to the generated behavior streams.

Value

A vector of behavior counts of length `n`.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
r_event_counting(n = 5, mu = 2, lambda = 4, stream_length = 20,
                 F_event = F_exp(), F_interim = F_exp())
```

r_MTS — *Generates random momentary time sampling behavior streams*

Description

Random generation of behavior streams (based on an alternating renewal process) of a specified length and with specified mean event durations, mean interim times, event distribution, and interim distribution, which are then coded as momentary time sampling data with given interval length between moments.

Usage

```
r_MTS(n, mu, lambda, stream_length, F_event, F_interim, interval_length,
  summarize = FALSE, equilibrium = TRUE, p0 = 0, tuning = 2)
```

Arguments

`n`	number of behavior streams to generate
`mu`	mean event duration
`lambda`	mean interim time
`stream_length`	length of behavior stream
`F_event`	distribution of event durations. Must be of class `eq_dist`.
`F_interim`	distribution of interim times. Must be of class `eq_dist`.
`interval_length`	length of time between moments
`summarize`	logical value indicating whether the vector of moments should be summarized by taking their mean, excluding the first moment in each row.
`equilibrium`	logical; if `TRUE`, then equilibrium initial conditions are used; if `FALSE`, then `p0` is used to determine initial state and normal generating distributions are used for event durations and interim times.
`p0`	Initial state probability. Only used if `equilibrium = FALSE`, in which case default is zero (i.e., behavior stream always starts with an interim time).
`tuning`	controls the size of the chunk of random event durations and interim times. Adjusting this may be useful in order to speed computation time.

Details

Generates behavior streams by repeatedly drawing random event durations and random interim times from the distributions as specified, until the sum of the durations and interim times exceeds the requested stream length. Then applies a momentary time sampling filter to the generated behavior streams.

Value

If `summarize = FALSE`, a matrix of logicals with rows equal to `n` and length equal to `(stream_length/interval_length) +`
If `summarize = TRUE`, a vector of means of length `n`.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# A set of unsummarized MTS observations
r_MTS(n = 5, mu = 2, lambda = 4, stream_length = 20,
      F_event = F_exp(), F_interim = F_exp(), interval_length = 1)

# A set of summarized MTS observations
r_MTS(n = 5, mu = 2, lambda = 4, stream_length = 20,
      F_event = F_exp(), F_interim = F_exp(),
      interval_length = 1, summarize = TRUE)
```

---

r_PIR — *Generates random partial interval recording behavior streams*

Description

Random generation of behavior streams (based on an alternating renewal process) of a specified length and with specified mean event durations, mean interim times, event distribution, and interim distribution, which are then coded as partial interval recording data with given interval length and rest length.

Usage

```
r_PIR(n, mu, lambda, stream_length, F_event, F_interim, interval_length,
  rest_length = 0, summarize = FALSE, equilibrium = TRUE, p0 = 0,
  tuning = 2)
```

Arguments

`n`	number of behavior streams to generate
`mu`	mean event duration
`lambda`	mean interim time
`stream_length`	length of behavior stream
`F_event`	distribution of event durations. Must be of class `eq_dist`.
`F_interim`	distribution of interim times. Must be of class `eq_dist`.
`interval_length`	total interval length
`rest_length`	length of any recording time in each interval
`summarize`	logical value indicating whether the behavior streams should by summarized by taking their mean
`equilibrium`	logical; if `TRUE`, then equilibrium initial conditions are used; if `FALSE`, then `p0` is used to determine initial state and normal generating distributions are used for event durations and interim times.

`p0`	Initial state probability. Only used if `equilibrium = FALSE`, in which case default is zero (i.e., behavior stream always starts with an interim time).
`tuning`	controls the size of the chunk of random event durations and interim times. Adjusting this may be useful in order to speed computation time .

Details

Generates behavior streams by repeatedly drawing random event durations and random interim times from the distributions as specified, until the sum of the durations and interim times exceeds the requested stream length. Then applies a partial interval recording filter to the generated behavior streams.

Value

If `summarize = FALSE`, a matrix with rows equal to `n` and a number of columns equal to the number intervals per session. If `summarize = TRUE` a vector of means of length `n`.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# An unsummarized set of PIR observations
r_PIR(n = 5, mu = 2, lambda = 4, stream_length = 20,
       F_event = F_exp(), F_interim = F_exp(),
       interval_length = 1, rest_length = 0)

# A summarized set of of PIR observations
r_PIR(n = 5, mu = 2, lambda = 4, stream_length = 20,
       F_event = F_exp(), F_interim = F_exp(),
       interval_length = 1, rest_length = 0,
       summarize = TRUE)
```

`r_WIR` *Generates random whole interval recording behavior streams*

Description

Random generation of behavior streams (based on an alternating renewal process) of a specified length and with specified mean event durations, mean interim times, event distribution, and interim distribution, which are then coded as whole interval recording data with given interval length and rest length.

Usage

```
r_WIR(n, mu, lambda, stream_length, F_event, F_interim, interval_length,
  rest_length = 0, summarize = FALSE, equilibrium = TRUE, p0 = 0,
  tuning = 2)
```

Arguments

`n`	number of behavior streams to generate
`mu`	mean event duration
`lambda`	mean interim time
`stream_length`	length of behavior stream
`F_event`	distribution of event durations. Must be of class `eq_dist`.
`F_interim`	distribution of interim times. Must be of class `eq_dist`.
`interval_length`	total interval length
`rest_length`	length of any recording time in each interval
`summarize`	logical value indicating whether the behavior streams should by summarized by taking their mean
`equilibrium`	logical; if `TRUE`, then equilibrium initial conditions are used; if `FALSE`, then `p0` is used to determine initial state and normal generating distributions are used for event durations and interim times.
`p0`	Initial state probability. Only used if `equilibrium = FALSE`, in which case default is zero (i.e., behavior stream always starts with an interim time).
`tuning`	controls the size of the chunk of random event durations and interim times. Adjusting this may be useful in order to speed computation time .

Details

Generates behavior streams by repeatedly drawing random event durations and random interim times from the distributions as specified, until the sum of the durations and interim times exceeds the requested stream length. Then applies a whole interval recording filter to the generated behavior streams.

Value

If `summarize = FALSE`, a matrix with rows equal to `n` and a number of columns equal to the number intervals per session. If `summarize = TRUE` a vector of means of length `n`.

Author(s)

Daniel Swan <dswan@utexas.edu>

Examples

```
# An unsummarized set of WIR observations
r_WIR(n = 5, mu = 2, lambda = 4, stream_length = 20,
       F_event = F_exp(), F_interim = F_exp(),
       interval_length = 1, rest_length = 0)

# A summarized set of of WIR observations
r_WIR(n = 5, mu = 2, lambda = 4, stream_length = 20,
       F_event = F_exp(), F_interim = F_exp(),
       interval_length = 1, rest_length = 0,
       summarize = TRUE)
```

Index