Package: tvgeom (via r-universe)

November 5, 2024

Title The Time-Varying (Right-Truncated) Geometric Distribution

Version 1.0.1

Description Probability mass (d), distribution (p), quantile (q), and random number generating (r and rt) functions for the time-varying right-truncated geometric (tvgeom) distribution. Also provided are functions to calculate the first and second central moments of the distribution. The tvgeom distribution is similar to the geometric distribution, but the probability of success is allowed to vary at each time step, and there are a limited number of trials. This distribution is essentially a Markov chain, and it is useful for modeling Markov chain systems with a set number of time steps.

License MIT + file LICENSE

Copyright 2019 Conservation Science Partners, Inc.

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Suggests ggthemes, ggplot2, dplyr, tidyr, purrr, knitr, rmarkdown, magrittr, gridExtra, testthat

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Vincent Landau [aut, cre], Luke Zachmann [ctb], Conservation Science Partners, Inc. [cph]

Maintainer Vincent Landau <vincent@csp-inc.org>

Repository CRAN

Date/Publication 2019-12-07 01:40:02 UTC

Contents

---

tvgeom   *tvgeom: A package for the time-varying geometric probability distri-*
                                bution

Description

The tvgeom package provides two categories of important functions: probability distribution functions (d, p, q, r) and moments (tvgeom_mean and tvgeom_var).

Density (dtvgeom), distribution function (qtvgeom), quantile function (ptvgeom), and random number generation (rtvgeom and rttvgeom for sampling from the *full* and *truncated* distribution, respectively) for the time-varying, right-truncated geometric distribution with parameter prob.

Usage

```
dtvgeom(x, prob, log = FALSE)

ptvgeom(q, prob, lower.tail = TRUE, log.p = FALSE)

qtvgeom(p, prob, lower.tail = TRUE, log.p = FALSE)

rtvgeom(n, prob)

rttvgeom(n, prob, lower = 0, upper = length(prob) + 1)
```

Arguments

x, q	vector of quantiles representing the trial at which the first success occurred.
prob	vector of the probability of success for each trial.
log, log.p	logical; if TRUE, probabilities, p, are given as log(p). Defaults to FALSE.
lower.tail	logical; if FALSE, ptvgeom returns
p	vector of probabilities at which to evaluate the quantile function.
n	number of observations to sample.
lower	lower value (exclusive) at which to truncate the distribution for random number generation. Defaults to 0, in which case the distribution is not left-truncated.
upper	upper value (inclusive) at which to truncate the distribution for random number generation. Defaults to length(prob), in which case the distribution is not right-truncated.

Details

The time-varying geometric distribution describes the number of independent Bernoulli trials needed to obtain one success. The probability of success, prob, may vary for each trial. It has mass

$$p(x) = prob[x] * prod(1 - prob[1 : (x - 1)])$$

with support $x = 1, 2, ..., n + 1$, where n equals then length of prob. For i in $prob, 0 \leq i \leq 1$. The n+1 case represents the case that the event did not happen in the first n trials.

Value

dtvgeom gives the probability mass, qtvgeom gives the quantile functions, ptvgeom gives the distribution function, rtvgeom generates random numbers, and rttvgeom gives random numbers from the distribution truncated at bounds provided by the user. $P(X > x)$ instead of $P(X \leq x)$. Defaults to TRUE.

Package functions

The tvgeom functions ...

Examples

```
# What's the probability that a given number of trials, n, are needed to get
# one success if `prob` = `p0`, as defined below...?
p0 <- .15 # the probability of success

# Axis labels (for plotting purposes, below).
x_lab <- "Number of trials, n"
y_lab <- sprintf("P(success at trial n | prob = %s)", p0)

# Scenario 1: the probability of success is constant and we invoke functions
# from base R's implementation of the geometric distribution.
y1 <- rgeom(1e3, p0) + 1 # '+1' b/c dgeom parameterizes in terms of failures
x1 <- seq_len(max(y1))
z1 <- dgeom(x1 - 1, p0)
plot(table(y1) / 1e3,
  xlab = x_lab, ylab = y_lab, col = "#00000020",
  bty = "n", ylim = c(0, p0)
)
lines(x1, z1, type = "l")

# Scenario 2: the probability of success is constant, but we use tvgeom's
# implementation of the time-varying geometric distribution. For the purposes
# of this demonstration, the length of vector `prob` (`n_p0`) is chosen to be
# arbitrarily large *relative* to the distribution of n above (`y1`) to
# ensure we don't accidentally create any censored observations!
n_p0 <- max(y1) * 5
p0_vec <- rep(p0, n_p0)
y2 <- rtvgeom(1e3, p0_vec)
x2 <- seq_len(max(max(y1), max(y2)))
z2 <- dtvgeom(x2, p0_vec) # dtvgeom is parameterized in terms of successes
points(x2[x2 <= max(y1)], z2[x2 <= max(y1)],
  col = "red", xlim = c(1, max(y1))
)

# Scenario 3: the probability of success for each process varies over time
# (e.g., chances increase linearly by `rate` for each subsequent trial until
# chances saturate at `prob` = 1).
rate <- 1.5
prob_tv <- numeric(n_p0)
for (i in 1:length(p0_vec)) {
  prob_tv[i] <- ifelse(i == 1, p0_vec[i], rate * prob_tv[i - 1])
```

```
  }
  prob_tv[prob_tv > 1] <- 1
  y3 <- rtvgeom(1e3, prob_tv)
  x3 <- seq_len(max(y3))
  z3 <- dtvgeom(x3, prob_tv)
  plot(table(y3) / 1e3,
    xlab = x_lab, col = "#00000020", bty = "n",
    ylim = c(0, max(z3)),
    ylab = sprintf("P(success at trial n | prob = %s)", "`prob_tv`")
  )
  lines(x3, z3, type = "l")
```

---

tvgeom-moments	*Moments for the The Time-Varying (Right-Truncated) Geometric Distribution*

Description

Functions to calculate first moment `tvgeom_mean()` and second central moment `tvgeom_var()` for the time-varying geometric distribution.

Usage

```
  tvgeom_mean(prob)

  tvgeom_var(prob)
```

Arguments

prob   vector of the probability of success for each trial/time step.

Value

`tvgeom_mean` returns the moment (the mean), and `tvgeom_var` returns the second central moment (the variance).

Examples

```
  tvgeom_mean(prob = rep(0.1, 5))
  tvgeom_var(prob = rep(0.1, 5))
```

Index