Package ‘spuRs’

April 27, 2025

Type Package

Title Functions and Datasets for ``Introduction to Scientific Programming and Simulation Using R''

Version 2.0.3

Date 2025-04-27

Maintainer Andrew Robinson <apro@unimelb.edu.au>

Depends R (>= 2.10), MASS, lattice

Description
Provides functions and datasets from Jones, O.D., R. Maillardet, and A.P. Robinson. 2014. An Introduction to Scientific Programming and Simulation, Using R. 2nd Ed. Chapman And Hall/CRC.

License GPL-3

LazyLoad yes

NeedsCompilation no

Author Owen Jones [aut],
Robert Maillardet [aut],
Andrew Robinson [aut, cre],
Olga Borovkova [aut],
Steven Carnie [aut]

Repository CRAN

Date/Publication 2025-04-27 13:10:01 UTC

Contents

`bisection` *A function of the bisection algorithm.*

Description

Applies the bisection algorithm to find x such that ftn(x) == x.

Usage

```
bisection(ftn, x.l, x.r, tol = 1e-09)
```

Arguments

`ftn`	the function.
`x.l`	is the lower starting point.
`x.r`	is the upper starting point.
`tol`	distance of successive iterations at which algorithm terminates.

Details

We assume that `ftn` is a function of a single variable.

Value

Returns the value of x at which ftn(x) == x. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

newtonraphson, fixedpoint

Examples

```
ftn5 <- function(x) return(log(x)-exp(-x))
bisection(ftn5, 1, 2, tol = 1e-6)
```

---

booking_clerkMC *A function to simulate the harassed booking clerk Markov chain.*

Description

Simulates the harassed booking clerk Markov chain with given arrival and service rates up to t.end. The state space is (C(t),X(t),Y(t)), where C(t) represents the status of the clerk, X(t) the number of people waiting, and Y(t) the number of calls waiting. C(t) is 0 if clerk is idle, 1 if clerk is serving a person and 2 if clerk is serving a call.

Usage

```
booking_clerkMC(personArrRate,
                callArrRate,
                personServRate,
                callServRate,
                t.end)
```

Arguments

personArrRate	the person arrival rate.
callArrRate	the call arrival rate.
personServRate	the person service rate.
callServRate	the call service rate.
t.end	the time of the time period to be simulated i.e. (0,t.end).

Details

We assume that all given rates are finite and positive.

Value

Returns the matrix (t.hist, state.hist) containing the realisation of the chain.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

Examples

```
booking_clerkMC(3,6,5,8,1)
```

CMCSimulation — *A function to simulate a continuous time Markov chain.*

Description

This function simulates a continuous time finite state space Markov chain with known rate matrix Q, state space 0,1,..,n and initial state i for the time period (0,T). If plotflag is TRUE it also produces a plot.

Usage

```
CMCSimulation(Q,i,Tend,plotflag = FALSE)
```

Arguments

`Q`	the rate matrix.
`i`	the initial state.
`Tend`	the end of the simulation period (0,T).
`plotflag`	flag indicating if plot needed

Details

We assume that `Q` is well defined rate matrix.

Value

Returns the matrix (statehist,timehist) containing the realisation of the chain for the specified period. The function also produces a plot of the realisation. \

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`MCSimulation`

Examples

```
Q <- matrix(c(-24,24,0, 2,-14,12, 0,4,-4),
            nrow = 3, ncol = 3, byrow = TRUE)
CMCSimulation(Q,0,1)
```

---

fitDistances *Function to fit a model to seed transect distance/count data.*

Description

This function uses maximum likelihood to fit a nominated probability density function to the data of a seedtrap transect holder.

Usage

```
fitDistances(x, family)
```

Arguments

`x`	an object of class transectHolder
`family`	the nominated distribution, which must be one of those distributions that can be fit by `fitdistr` of the MASS package.

Value

The function returns the parameter estimates for the nominated family.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`fitdistr`, `trapTransect`

Examples

```
library(MASS)
s1 <- trapTransect(distances = 1:4, seed.counts = c(4, 3, 2, 0))
allTraps <- transectHolder(s1, family="Weibull")
fitDistances(allTraps, "exponential")
```

`fixedpoint` *A function of the fixed point algorithm.*

Description

Applies the fixed point algorithm to find x such that ftn(x) == x.

Usage

```
fixedpoint(ftn, x0, tol = 1e-09, max.iter = 100)
```

Arguments

`ftn`	the function.
`x0`	is the initial guess at the fixed point.
`tol`	distance of successive iterations at which algorithm terminates.
`max.iter`	maximum number of iterations.

Details

We assume that `ftn` is a function of a single variable.

Value

Returns the value of x at which ftn(x) == x. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`newtonraphson`, `bisection`

Examples

```
ftn1 <- function(x) return(exp(exp(-x)))
fixedpoint(ftn1, 2, tol = 1e-6)
```

`fixedpoint_show` *A function of the fixed point algorithm.*

Description

Applies the fixed point algorithm to find x such that ftn(x) == x, and plots the process.

Usage

```
fixedpoint_show(ftn, x0, xmin = x0 - 1, xmax = x0 + 1)
```

Arguments

`ftn`	the function.
`x0`	is the initial guess at the fixed point.
`xmin`	~~Describe `xmin` here~~
`xmax`	~~Describe `xmax` here~~

Details

We assume that `ftn` is a function of a single variable.

Value

Returns the value of x at which ftn(x) == x. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`fixedpoint`

`kew` *303 years of monthly rainfall data from Kew Gardens, London, U.K.*

Description

The monthly rainfall at Kew Gardens, London, U.K., from 1697 to 1999, in mm.

Usage

```
data(kew)
```

Format

A wide-format data frame with 303 observations. Each month has its own column.

Source

Data obtained from the U.S. National Climatic Data Centre, Global Historical Climatology Network data base (GHCN-Monthly Version 2, NB: not Version 3) https://www.ncei.noaa.gov/products/land-based-station/global-historical-climatology-network-monthly.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2014. An Introduction to Scientific Programming and Simulation, Using R. 2nd Ed. Chapman And Hall/CRC.

Examples

```
data(kew)
```

`MCEstimation` *A function to estimate the transition matrix for a discrete time Markov chain.*

Description

This function estimates the transition matrix for a discrete time Markov chain with state space 0,1,..,n given a realisation. The chain has n+1 states.

Usage

```
MCEstimation(statehist,n)
```

Arguments

`statehist`	the realisation of the chain.
`n`	the highest numbered state.

Details

We assume that the state space is 0,1,2...,n. n is assumed known as it cannot be reliably infered from the realisation.

Value

Returns the empirical transition matrix obtained by calculating the observed frequencies of actual transitions in the realisation.\

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

MCSimulation

Examples

```
P <- matrix(c(0.5,0.5,0,0,0.7,0.1,0.2,0,0,0.1,0.1,0.8,0,0,0.7,0.3),
            nrow = 4, ncol = 4, byrow = TRUE)
statehist<-MCSimulation(P, 0, 3000)
MCEstimation(statehist, 3)
```

MCSimulation *A function to simulate a discrete time Markov chain.*

Description

This function simulates a discrete time Markov chain with transition matrix P, state space 0,1,..,n and and initial state i for nsteps transitions.

Usage

```
MCSimulation(P,i,nsteps)
```

Arguments

P	the transition matrix.
i	the initial state.
nsteps	the number of transitions to be simulated.

Details

We assume that P is well defined transition matrix with rows summing to 1.

Value

Returns the vector statehist containing the realisation of the chain for nsteps transitions.\

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

MCEstimation, CMCSimulation

Examples

```
P <- matrix(c(0.5,0.5,0,0,0.7,0.1,0.2,0,0,0.1,0.1,0.8,0,0,0.7,0.3),
            nrow = 4, ncol = 4, byrow = TRUE)
MCSimulation(P, 0, 250)
```

---

mean.transectHolder *Function to compute the mean dispersal distance along a transect of seed traps.*

Description

This function computes the mean dispersal distance along a transect of seed traps.

Usage

```
## S3 method for class transectHolder
## S3 method for class 'transectHolder'
mean(x, ...)
```

Arguments

x	an object representing a transect of seed traps.
...	further arguments passed to or from other methods.

Value

The mean seed dispersal distance is returned.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

transectHolder

Examples

```
transect.1 <- trapTransect(distances = 1:4,
                           seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                           seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                           seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                              family="Weibull")

mean(allTraps)
```

`mean.trapTransect` *Function to compute the mean dispersal distance along a transect of seed traps.*

Description

This function computes the mean dispersal distance along a transect of seed traps.

Usage

```
## S3 method for class trapTransect
## S3 method for class 'trapTransect'
mean(x, ...)
```

Arguments

`x`	an object representing a transect of seed traps.
`...`	further arguments passed to or from other methods.

Value

The mean seed dispersal distance is returned.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

trapTransect

Examples

```
s1 <- trapTransect(distances = 1:4, seed.count = c(4, 3, 2, 0))
mean(s1)
```

`newtonraphson` *A function of the Newton-Raphson algorithm.*

Description

Applies the Newton-Raphson algorithm to find x such that ftn(x)[1] == 0.

Usage

```
newtonraphson(ftn, x0, tol = 1e-09, max.iter = 100)
```

Arguments

`ftn`	the function.
`x0`	is the initial guess at the fixed point.
`tol`	distance of successive iterations at which algorithm terminates.
`max.iter`	maximum number of iterations.

Value

Returns the value of x at which ftn(x)[1] == 0. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`fixedpoint`, `bisection`

Examples

```
ftn4 <- function(x) {
  # returns function value and its derivative at x
  fx <- log(x) - exp(-x)
  dfx <- 1/x + exp(-x)
  return(c(fx, dfx))
}
newtonraphson(ftn4, 2, 1e-6)
```

newtonraphson_show *A function of the Newton-Raphson algorithm, plotting the path.*

Description

Applies the Newton-Raphson algorithm to find x such that ftn(x)[1] == 0, and plots the trace of the estimate.

Usage

```
newtonraphson_show(ftn, x0, xmin = x0 - 1, xmax = x0 + 1)
```

Arguments

`ftn`	the function.
`x0`	the initial guess of the fixed point.
`xmin`	lower limit for plotting.
`xmax`	upper limit for plotting.

Value

Returns the value of x at which ftn(x)[1] == 0. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`newtonraphson`

prime *Function to assess whether or not an integer is prime.*

Description

An inefficient, brute-force algorithm to assess whether or not an integer is prime.

Usage

```
prime(n)
```

Arguments

`n`	The integer.

Details

The function assumes that `n` is a positive integer.

Value

The function returns a logical object that is TRUE if the integer is prime.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`primesieve`

Examples

```
prime(10)
prime(7)
```

`primesieve` *Function to identify all the primes in a vector of positive integers.*

Description

This function uses the Sieve of Eratosthenes to find all the primes less than or equal to a given integer.

Usage

```
primesieve(sieved, unsieved)
```

Arguments

`sieved`	Identified primes (empty vector for initialization)
`unsieved`	Candidate integers

Details

The function assumes that `unsieved` is a vector of positive integers.

Value

Returns a vector of primes sieved (selected) from the input vector.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

prime

Examples

```
primesieve(c(), 2:200)
```

---

print.transectHolder *Function to print a transectHolder object usefullly.*

Description

This function prints the details of a transectHolder object.

Usage

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

Arguments

x	An object representing a transect of seed traps.
...	further arguments passed to or from other methods.

Details

The print function simply uses `str` on the transectHolder object.

Value

This function is called for its side-effect, which is to print the object informatively.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

transectHolder

Examples

```
transect.1 <- trapTransect(distances = 1:4,
                          seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                          seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                          seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                           family="Weibull")
allTraps
```

print.trapTransect *Function to print a trapTransect object usefullly.*

Description

This function prints the details of a trapTransect object.

Usage

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

Arguments

x	An object representing a transect of seed traps.
...	further arguments passed to or from other methods.

Details

The print function simply uses `str` on the trapTransect object.

Value

This function is called for its side-effect, which is to print the object informatively.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`trapTransect`

Examples

```
s1 <- trapTransect(distances = 1:4, seed.count = c(4, 3, 2, 0))
s1
```

`RK4adapt` *A function which uses the Fourth order Runge-Kutta method with adaptive step size to solve a system of ODE's.*

Description

This function simulates a discrete time Markov chain with transition matrix P, state space 0,1,..,n and and initial state i for nsteps transitions.

Usage

```
RK4adapt(dydt, t0, y0, t1, h0 = 1, tol = 1e-10, ...)
```

Arguments

`dydt`	a function giving the gradient of y(t).
`t0`	initial value of t.
`y0`	initial value of y(t).
`t1`	system solved up to time t1.
`h0`	initial step size
`tol`	tolerance for adapting step size.
`...`	pass arguments to function dydt.

Details

We assume that `P` is well defined transition matrix with rows summing to 1.

Value

Returns a list with elements t, a vector giving times, and y, a matrix whose rows give the solution at successive times.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

Examples

```
LV <- function(t=NULL, y, a, b, g, e, K=Inf)
  c(a*y[1]*(1 - y[1]/K) - b*y[1]*y[2], g*b*y[1]*y[2] - e*y[2])

xy <- RK4adapt(LV, 0, c(100, 50), 200, 1, tol=1e-3,
               a=0.05, K=Inf, b=0.0002, g=0.8, e=0.03)

par(mfrow = c(2,1))
plot(xy$y[,1], xy$y[,2], type='p',
     xlab='prey', ylab='pred', main='RK4, adaptive h')
plot(xy$t, xy$y[,1], type='p', xlab='time',
     ylab='prey circles pred triangles', main='RK4, adaptive h')
points(xy$t, xy$y[,2], pch=2)
par(mfrow=c(1,1))
```

`sd.transectHolder` *Function to compute the sd dispersal distance along a transect of seed traps.*

Description

This function computes the standard deviation of the dispersal distances along a transect of seed traps.

Usage

```
sd.transectHolder(transectHolder)
```

Arguments

`transectHolder` an object representing a transect of seed traps.

Value

The standard deviation of the seed dispersal distances is returned.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`transectHolder`

Examples

```
transect.1 <- trapTransect(distances = 1:4,
                        seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                        seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                        seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                          family="Weibull")

mean(allTraps)

sd.transectHolder(allTraps)
```

---

simulate.transectHolder

Function to simulate a modelled seed rain from a transectHolder

---

Description

This function simulates a two-dimensional seed rain according to the model stored in a transectHolder object. The angle of the seed location from the parent plant is uniformly distributed on [0, 2 pi).

Usage

```
## S3 method for class transectHolder
## S3 method for class 'transectHolder'
simulate(object, nsim=1, seed=NULL, ...)
```

Arguments

`object`	the transectHolder object for simulation
`nsim`	the number of seeds to simulate.
`seed`	if not NULL, set the seed to this value before simulation.
`...`	additional optional arguments (ignored here).

Value

A dataframe with `n` rows with the following components:

`distances`	seed distances to parent plant
`angles`	seed angles to parent plant, in radians
`x`	x-location of seed
`y`	y-location of seed

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`transectHolder`

Examples

```
transect.1 <- trapTransect(distances = 1:4,
                        seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                        seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                        seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                           family="Weibull")
allTraps
simulate(allTraps, nsim=5, seed=123)
```

`transectHolder` *Function to construct an object representing a collection of trapTransect objects.*

Description

This function constructs a transectHolder object given a collection of `trapTransect` objects and a nominated probability density function to fit to the seed count profile.

Usage

```
transectHolder(..., family = "exponential")
```

Arguments

`...`	one or more `trapTransect` objects
`family`	the probability density function to fit to the distance count profiles.

Details

This function is a constructor.

The nominated distribution, which must be one of those distributions that can be fit by `fitdistr` of the MASS package.

Value

A transectHolder object, which is a list comprising

`transects` a list one or more trapTransect objects,

`family` the name of the distribution to which the transect data has been fit,

`parameters` the estimated parameters for that distribution,

`rng` the corresponding random number generator for simulations.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

trapTransect

Examples

```
transect.1 <- trapTransect(distances = 1:4,
                           seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                           seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                           seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                              family="Weibull")
allTraps
```

---

trapTransect — *Function to construct an object representing a transect of seedtraps.*

Description

This function constructs a trapTransect object given a vector of trap distances from the parent plant, a vector of trap seed counts corresponding to the trap distances, and a single trap area.

Usage

```
trapTransect(distances, seed.counts, trap.area = 0.0001)
```

Arguments

`distances` A vector of trap distances from the parent plant.

`seed.counts` A vector of seed counts in each trap.

`trap.area` The surface area of each trap.

Details

This function is a constructor.

Value

A trapTransect object, which is a list comprising three objects:

`distances`	A vector of trap distances from the parent plant.
`seed.counts`	A vector of seed counts in each trap.
`trap.area`	The surface area of each trap.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`mean.trapTransect`, `print.trapTransect`

Examples

```
s1 <- trapTransect(distances = 1:4, seed.counts = c(4, 3, 2, 0))
s1
mean(s1)
```

---

treg

`treeg` *Grand fir tree growth data from northern and central Idaho, USA.*

Description

A sample of 66 grand fir (*Abies grandis*) trees was selected from national forests around northern and central Idaho. The trees were selected to be dominant in their environment, with no visible evidence of crown damage, forks, broken tops, etc. For each tree the habitat type and the national forest from which it came were recorded. We have data from nine national forests and six different habitat types.

Usage

```
data(treeg)
```

Format

A data frame with 542 observations on the following 6 variables.

tree.ID Tree number.

forest National forest number.

habitat Habitat code (see Details).

dbh.in Bole diameter at 1.37 m, in inches

height.ft Tree height, in feet.

age Age at which measurement was taken.

Details

For each tree the height, diameter and age were measured (age is measured using tree rings), then the tree was split lengthways, which allows you to determine the height and diameter of the tree at any age. In this instance height and diameter were recorded for the age the tree was felled and then at ten year periods going back in time. The diameter of the tree was measured at a height of 1.37 m (4'6"), which is called *breast height* in forestry. The height refers to the height of the main trunk only.

The habitats corresponding to codes 1 through 5 are: Ts/Pach; Ts/Op; Th/Pach; AG/Pach and PA/Pach. These codes refer to the climax tree species, which is the most shade-tolerant species that can grow on the site, and the dominant understorey plant, respectively. Ts refers to *Thuja plicata* and *Tsuga heterophylla*, Th refers to just *Thuja plicata*, AG is *Abies grandis*, PA is *Picea engelmanii* and *Abies lasiocarpa*, Pach is *Pachistima myrsinites*, and Op is the nasty *Oplopanaz horridurn*. Grand fir is considered a major climax species for AG/Pach, a major seral species for Th/Pach and PA/Pach, and a minor seral species for Ts/Pach and Ts/Op. Loosely speaking, a community is *seral* if there is evidence that at least some of the species are temporary, and *climax* if the community is self-regenerating (Daubenmire, 1952).

Source

These data were kindly supplied by Dr Al Stage, Principal Mensurationist (retired), USDA Forest Service Foresct Sciences Laboratory, Moscow, ID, USA.

References

R. Daubenmire, 1952. Forest Vegetation of Northern Idaho and Adjacent Washington, and Its Bearing on Concepts of Vegetation Classification, *Ecological Monographs* **22**, 301–330.

A. R. Stage, 1963. A mathematical approach to polymorphic site index curves for grand fir. *Forest Science* **9**, 167–180.

Examples

```
data(treeg)
```

`trees` *von Guttenberg Norway spruce tree measurement data*

Description

These are a subset of the von Guttenberg data, a set of measurements on Norway spruce (*Picea abies* [L.] Karst) in several different locations and site categories.

Usage

```
data(trees)
```

Format

A data frame with 1200 observations on the following 3 variables.

ID A factor identifying the tree by location, site, and tree number.

Age The age at which the tree was measured.

Vol The bole volume of the tree, in cubic dm.

Source

These data were kindly provided by Professor Boris Zeide, University of Arkanasa, Monticello, AK, USA, and are further documented in Zeide (1993).

References

A.R. von Guttenberg. 1915. Growth and yield of spruce in Hochgebirge. Franz Deuticke, Wien. (In German)

B. Zeide, 1993. Analysis of growth equations. *Forest Science* **39** 594–616.

Examples

```
data(trees)
```

`ufc` *Upper Flat Creek forest cruise tree data*

Description

These are a subset of the tree measurement data from the Upper Flat Creek unit of the University of Idaho Experimental Forest, which was measured in 1991.

Usage

```
data(ufc)
```

Format

A data frame with 336 observations on the following 5 variables.

plot plot label

tree tree label

species species kbd with levels `DF`, `GF`, `WC`, `WL`

dbh.cm tree diameter at 1.37 m. from the ground, measured in centimetres.

height.m tree height measured in metres

Details

The inventory was based on variable radius plots with 6.43 sq. m. per ha. BAF (Basal Area Factor). The forest stand was 121.5 ha. This version of the data omits errors, trees with missing heights, and uncommon species. The four species are Douglas-fir, grand fir, western red cedar, and western larch.

Source

The data are provided courtesy of Harold Osborne and Ross Appelgren of the University of Idaho Experimental Forest.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`ufc.plots`

Examples

```
data(ufc)
```

`ufc.plots` *Upper Flat Creek forest cruise plot data*

Description

These are a subset of the plot measurement data from the Upper Flat Creek unit of the University of Idaho Experimental Forest, which was measured in 1991.

Usage

```
data(ufc.plots)
```

Format

A data frame with 144 observations on the following 6 variables.

plot plot label

north.n northerly plot count

east.n easterly plot count

north northerly coordinate

east easterly coordinate

vol.m3.ha total above-ground merchantable volume, in cubic metres per hectare.

Source

The data are provided courtesy of Harold Osborne and Ross Appelgren of the University of Idaho Experimental Forest.

References

Jones, O.D., R. Maillardet, and A.P. Robinson. 2009. An Introduction to Scientific Programming and Simulation, Using R. Chapman And Hall/CRC.

See Also

`ufc`

Examples

```
data(ufc.plots)
```

---

vol.m3 *Function to compute the volume of a tree bole assuming a particular shape.*

Description

This function computes the volume of a tree bole given its basal diameter and length, assuming that the bole is a frustum of a geometric solid.

Usage

```
vol.m3(dbh.cm, height.m, multiplier = 0.5)
```

Arguments

`dbh.cm`	basal diameter in cm.
`height.m`	height in m.
`multiplier`	shape, expressed as a multiplier.

Details

Commonly-used shapes are:

1/3 conoid

1/2 second-degree parabaloid

1 cylinder

Value

The volume is returned, in units of cubic metres.

Examples

```
vol.m3(30, 30)
vol.m3(30, 30, 1)
```

Index