Package: bcscr (via r-universe)

November 24, 2024

Type Package

Title Beginning Computer Science with R

Version 0.1.2

Maintainer Homer White <homerhanumat@gmail.com>

Description Functions and datasets to accompany the text Beginning Computer Science with R (https://homerhanumat.github.io/r-notes).

Imports data.tree, ggplot2, ggExtra, TurtleGraphics, R6, purrr, shiny, shinydashboard, htmltools

Suggests DiagrammeR

License GPL (>=3)

Encoding UTF-8

NeedsCompilation no

LazyData true

URL https://github.com/homerhanumat/bcscr

RoxygenNote 7.2.3

Config/pak/sysreqs make libicu-dev zlib1g-dev

Repository https://homerhanumat.r-universe.dev

RemoteUrl https://github.com/homerhanumat/bcscr

RemoteRef HEAD

RemoteSha 981a01a32998ad8ad66c2e4e229aeda69a0a5400

Contents

`collatz` *Collatz Numbers*

Description

Find and graph the Collatz sequence from a given initial number.

Usage

```
collatz(n, limit = 10000)
```

Arguments

`n`	the initial integer
`limit`	maximum number of members of the Collatz sequence to compute

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## Not run:
collatz(1757)

## End(Not run)
```

courtSim *Appeals Court Simulation*

Description

In this version all five judges vote independently.

Usage

```
courtSim(reps = 10000,
                seed = NULL,
                table = FALSE,
                probs = c(0.95, 0.94, 0.90, 0.90, 0.80))
```

Arguments

reps	number of simulations to perform
seed	The user may provide a seed-value for random-number generation.
table	Does the user want a table of the results?
probs	Chance for each judge to make the right decision on any given case.

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
courtSim(seed = 3030)
```

courtSim2 *Appeals Court Simulation (Version 2)*

Description

In this version the weakest judge always votes with the strongest one.

Usage

```
courtSim2(reps = 10000,
                seed = NULL,
                table = FALSE,
                probs = c(0.95, 0.94, 0.90, 0.90, 0.80))
```

Arguments

reps	number of simulations to perform
seed	The user may provide a seed-value for random-number generation.
table	Does the user want a table of the results?
probs	Chance for each judge to make the right decision on any given case.

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
courtSim2(seed = 3030)
```

---

distExplore *Exploring Major Probability Distributions*

Description

Manipulate parameters of a chosen distribution.

Usage

```
distExplore(options = NULL)
```

Arguments

options	Options that will be passed to shiny::shinyApp.

Value

side effects

Author(s)

Homer White <hwhite0@georgetowncollege.edu>

Examples

```
## distExplore()
```

drunkenSim *Drunken-Turtle Simulation*

Description

A drunken turtle starts at the origin and takes unit steps, turning through a randdom angle after each step. We are interested in the distribution of the number of close-returns to the origin, for a fixed number of steps and a fixed measure of closeness.

Usage

```
drunkenSim(steps = 1000, reps = 10000, close = 0.5,
                        seed = NULL, table = FALSE)
```

Arguments

steps	the number of steps the turtle will take
reps	number of simulations to perform
close	the distance from the origin that counts as "close"
seed	The user may provide a seed-value for random-number generation.
table	Does the user want a table of the results?

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
drunkenSim(seed = 3030)
```

drunkenSim2 *Drunken-Turtle Simulation (Graph Version)*

Description

A drunken turtle starts at the origin and takes unit steps, turning through a randdom angle after each step. We graph the distance from the origin as a function of step-number.

Usage

```
drunkenSim2(steps = 1000, seed = NULL)
```

Arguments

steps	the number of steps the turtle will take
seed	The user may provide a seed-value for random-number generation.

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## Not run:
drunkenSim2(seed = 3030)

## End(Not run)
```

---

fuel *Speed and Fuel Efficiency (British Ford Escort)*

Description

A British Ford Escort was driven along a prescribed course. Each drive was done at a different speed, and the fuel efficiency was recorded for each drive.

Format

A data frame with 15 observations on the following 2 variables.

speed in kilometers per hour.

efficiency fuel efficiency, measured in liters of fuel required to travel 100 kilometers.

Source

The Basic Practice of Statistics, by Moore and McCabe.

kdExplore *Exploring Kernel Density Estimation*

Description

See how density plots are built from kernels.

Usage

```
kdExplore(data, options = NULL)
```

Arguments

data	A vector of numerical values from which to form the density plot.
options	Options that will be passed to shiny::shinyApp.

Value

side effects

Author(s)

Homer White <hwhite0@georgetowncollege.edu>

Examples

```
## small custom dataset:
## myData <- c(1, 3, 5, 6, 6.2, 7, 9)
## kdExplore(myData)

## random exponential data:
## kdExplore(rexp(50, rate = 0.2))
```

m111survey *MAT 111 Survey*

Description

Results of a survey of MAT 111 students at Georgetown College.

- height. How tall are you, in inches?
- ideal_ht. A numeric vector How tall would you LIKE to be, in inches?
- sleep. How much sleep did you get last night?
- fastest. What is the highest speed at which you have ever driven a car?
- weight_feel. How do you feel about your weight?

- love_first. Do you believe in love at first sight?
- extra_life. Do you believe in extraterrestrial life?
- seat. When you have a choice, where do you prefer to sit in a classroom?
- GPA. What is your college GPA?
- enough_Sleep. Do you think you get enough sleep?
- sex. What sex are you?
- diff. Your ideal height minus your actual height.

Format

A data frame with 71 rows and 12 variables

Source

Georgetown College, MAT 111.

`make_val_tree` *Building Random Trees*

Description

Utility function to make small directed acyclic graphs, using the data.tree package. Nodes are provided with randomly-selected numerical values.

Usage

```
make_val_tree(min_children = 1,
                      max_children = 4,
                      min_depth = 3,
                      values = 1:5,
                      fillcolor = "palegreen",
                      fontcolor = "black",
                      seed = NULL)
```

Arguments

`min_children`	Minimum number of children when node will not be a leaf.
`max_children`	Maximum number of children when node will not be a leaf.
`min_depth`	Nodes at less than this specified depth will not be leaves.
`values`	Numerical vector of possible values to associate with each node.
`fillcolor`	Fill color for nodes.
`fontcolor`	Fonot color for text showing node value and node name.
`seed`	Option to set the random seed.

Value

A tree object of class "Node" and "R6"

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## Not run:
tr <- make_val_tree(
  values = 1:10,
  seed = 4040,
  fillcolor = "lightblue"
)
library(DiagrammeR)
plot(tr)

## End(Not run)
```

NamePhone *Names and Phone Numbers*

Description

Sample data for regular expression practice.

- name. Last name followed by first.
- phone. Phone number with area code, in several formats.

Format

A data frame with 50 rows and 2 variables

---

numberNeededSim *Number-Needed Simulation*

Description

You pick random numbers from 0 to 1, until their sum exceeds some target number. What's the expected value of the number of numbers you have to pick?

Usage

```
numberNeededSim(target = 1, reps = 1000,
                seed = NULL, table = TRUE)
```

Arguments

target	the target number
reps	number of simulations to perform
seed	The user may provide a seed-value for random-number generation.
table	Does the user want a table of the results?

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
numberNeededSim(seed = 3030)
```

Ocean *Whales in an Ocean*

Description

R6 Object for simulating a population of whales.

Format

R6Class object.

Value

Object of R6Class with methods for simulation.

Properties

- dimensions: A vector of length two giving the dimensions of the ocean.
- males: A list of R6 objects of class Male containing the current population of male whales.
- females: A list of R6 objects of class Female containing the current population of female whales.
- malePop: Current number of males in the population.
- femalePop: Current number of females in the population.
- starveParameter: Helps determine probability for each whale to die by starvation in the current generation.
- distance: Computes distance between any two whales.

Methods

- new: Instantiates an Ocean object. Parameters are:
  - dims: A numeric vector of length 2 setting the length and width of the ocean.
  - males: An integer giving the number of males (to be created with defaults) or a list of Male whale objects.
  - females: An integer giving the number of females (to be created with defaults) or a list of Female whale objects.
  - starve: A non-negative number, used to determine probability that an individual starves in a given generation. The larger the value, the lower the carrying-capacity of the population will be.
- advance: Advances the simulation by one generation. Takes no arguments.
- plot: Plots the current population. Takes no arguments.

Methods

Public methods:

- Ocean$distance()
- Ocean$new()
- Ocean$starvationProbability()
- Ocean$advance()
- Ocean$plot()
- Ocean$clone()

Method distance():

Usage:

```
Ocean$distance(a, b)
```

Method new():

Usage:

```
Ocean$new(dims = c(100, 100), males = 10, females = 10, starve = 5)
```

Method starvationProbability():

Usage:

```
Ocean$starvationProbability(popDensity)
```

Method advance():

Usage:

```
Ocean$advance()
```

Method plot():

Usage:

```
Ocean$plot()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
Ocean$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## Not run:
library(ggplot2)
oceanSim <- function(
  steps = 100, males = 10,
  females = 10, starve = 5,
  animate = FALSE, seed = NULL
  ) {
 if ( !is.null(seed) ) {
   set.seed(seed)
  }
  ocean <- Ocean$new(dims = c(100, 100), males = males,
                     females = females, starve = starve)
  population <-numeric(steps)
  for ( i in 1:steps ) {
    population[i] <- ocean$malePop + ocean$femalePop
    if ( animate ) ocean$plot()
    if ( population[i] == 0 ) break
    ocean$advance()
    if ( animate ) {
        ocean$plot()
        Sys.sleep(0.5)
      }
  }
  pop <- population[1:i]
  df <- data.frame(time = 1:length(pop),
                   pop)
  ggplot(df, aes(x = time, y = pop)) + geom_line() +
    labs(x = "Time", y = "Whale Population")
}
oceanSim(seed = 5050)

## End(Not run)
```

---

qqExplore *Exploring Quantile-Quantile Plots*

Description

Understand why qq-plots bend the way they do.

Usage

```
qqExplore(data, options = NULL)
```

Arguments

data	A vector of numerical values from which to form the qq-plot.
options	Options that will be passed to shiny::shinyApp.

Value

side effects

Author(s)

Homer White <hwhite0@georgetowncollege.edu>

Examples

```
## bimodal data:
## bimodal <- c(rnorm(50, 5, 1), rnorm(50, 10, 1))
## qqExplore(bimodal)

## random exponential data:
## qqExplore(rexp(300, rate = 0.2))
```

---

railtrail *Volume of Users of a Rail Trail*

Description

This data table is modifed slightly from mosaicData::RailTrail, (see http://cran.r-project.org/web/packages/mosaicData/mosai
Description below is drawn from the mosaicData help file.

Usage

```
data(railtrail)
```

Format

A data frame with 90 observations on the following variables.

- hightemp daily high temperature (in degrees Fahrenheit)
- lowtemp daily low temperature (in degrees Fahrenheit)
- avgtemp average of daily low and daily high temperature (in degrees Fahrenheit)
- season spring, summer or fall
- cloudcover measure of cloud cover (in oktas)
- precip measure of precipitation (in inches)
- volume estimated number of trail users that day (number of breaks recorded)
- weekday logical indicator of whether the day was a non-holiday weekday
- dayType one of "weekday" or "weekend"

Details

The Pioneer Valley Planning Commission (PVPC) collected data north of Chestnut Street in Florence, MA for ninety days from April 5, 2005 to November 15, 2005. Data collectors set up a laser sensor, with breaks in the laser beam recording when a rail-trail user passed the data collection station.

There is a potential for error when two users trigger the infrared beam at exactly the same time since the counter would only logs one of the crossings. The collectors left the motion detector out during the winter, but because the counter drops data when the temperature falls below 14 degrees Fahrenheit, there is no data for the cold winter months.

Source

Pioneer Valley Planning Commission

References

http://www.fvgreenway.org/pdfs/Northampton-Bikepath-Volume-Counts

Examples

```
data(railtrail)
```

`triangleSim` *Chance of a Triangle*

Description

Break a unit length at two random points: what's the chance that the three segments produced can form a triangle?

Usage

```
triangleSim(reps = 10000, table = FALSE, seed = NULL)
```

Arguments

`reps`	number of simulations to perform
`table`	Does the user want a table of the results?
`seed`	The user may provide a seed-value for random-number generation.

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
triangleSim(seed = 3030)
```

---

turtle_bounce *Bouncing Turtle (Turtle Graphics)*

Description

A turtle walks randomly, but bounces back from the edge of its containing field.

Usage

```
turtle_bounce(side = 60, step= 10)
```

Arguments

side	side-lengths of the containing square
step	length of one turtle step (side-length/2 must be a multiple of step).

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## turtle_bounce(side = 80, step = 10)
```

---

turtle_drunk *Drunken Turtle (Turtle Graphics)*

Description

A turtle takes steps of a fixed length, but turns at a random angle after each step.

Usage

```
turtle_drunk(side, step)
```

Arguments

side	side-lengths of the containing square
step	length of one turtle step

Value

side effects

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## turtle_drunk(side = 100, step = 10)
```

`Whale` *Whales in an Ocean*

Description

R6 Objects for modelling a whale. Female and Male inherit from whale. Use Female and Male if you want to provide custom lists of male and female whales when you instantiate an ocean.

Format

`R6Class` object.

Value

Object of `R6Class` with methods for simulation.

Methods

- `new`: Instantiates an Whale object. Parameters are:
  - `position`: A numeric vector of length 2 giving the initial position. (Make sure that it's within the dimensions of the ocean.)
  - `age`: Initial age of the whale.
  - `lifespan`: Lifespan of the whale.
  - `range`: Distance at which a female can detect an eligible male.
  - `maturity`: Age of whale at which reproduction is possible.
  - `stepSize`: Number of units the whale move sin each gneration.

Methods

Public methods:

- `Whale$new()`
- `Whale$move()`
- `Whale$clone()`

Method `new()`:

Usage:

```
Whale$new(
  position = NA,
  age = 3,
  lifespan = 40,
  range = 5,
  maturity = 10,
  stepSize = 5
)
```

Method `move()`**:**

Usage:

```
Whale$move(dims, r = self$stepSize)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Whale$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Super class

`bcscr::Whale` -> `Male`

Methods

Public methods:

- `Male$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Male$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Super class

`bcscr::Whale` -> `Female`

Methods

Public methods:

- `Female$maleNear()`
- `Female$mate()`
- `Female$clone()`

Method `maleNear()`**:**

Usage:

```
Female$maleNear(males, dist)
```

Method `mate()`**:**

Usage:

```
Female$mate()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Female$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Homer White <homerhanumat@gmail.com>

Examples

```
## Not run:
initialMales <- vector(mode = "list", length = 10)
ages <- c(rep(3, 5), c(rep(10, 5)))
for (i in 1:10) {
  initialMales[[i]] <- Male$new(
    position = runif(2, min = 0, max = 100),
    age = ages[i],
    lifespan = 40,
    range = 12,
    maturity = 10,
    stepSize = 7
  )
}
initialFemales <- vector(mode = "list", length = 10)
for (i in 1:10) {
  initialFemales[[i]] <- Female$new(
    position = runif(2, min = 0, max = 100),
    age = ages[i],
    lifespan = 40,
    maturity = 10,
    range = 12,
    stepSize = 3
  )
}

library(ggplot2)
oceanSim <- function(
  steps = 100,
  males = 10,
  females = 10,
  starve = 5,
```

```
  animate = FALSE,
  seed = NULL
  ) {
 if ( !is.null(seed) ) {
   set.seed(seed)
  }
  ocean <- Ocean$new(dims = c(100, 100), males = males,
                     females = females, starve = starve)
  population <-numeric(steps)
  for ( i in 1:steps ) {
    population[i] <- ocean$malePop + ocean$femalePop
    if ( animate ) ocean$plot()
    if ( population[i] == 0 ) break
    ocean$advance()
    if ( animate ) {
        ocean$plot()
        Sys.sleep(0.5)
      }
  }
  pop <- population[1:i]
  df <- data.frame(time = 1:length(pop),
                   pop)
  ggplot(df, aes(x = time, y = pop)) + geom_line() +
    labs(x = "Time", y = "Whale Population")
}
oceanSim(males = initialMales, females = initialFemales, seed = 5050)

## End(Not run)
```

Index