Package ‘NetLogoR’

March 2, 2020

Title  Build and Run Spatially Explicit Agent-Based Models

Description  Build and run spatially explicit agent-based models using only the R platform. ‘NetLogoR’ follows the same framework as the ‘NetLogo’ software (Wilensky, 1999 <http://ccl.northwestern.edu/netlogo/>) and is a translation in R of the structure and functions of ‘NetLogo’. ‘NetLogoR’ provides new R classes to define model agents and functions to implement spatially explicit agent-based models in the R environment. This package allows benefiting of the fast and easy coding phase from the highly developed ‘NetLogo’ framework, coupled with the versatility, power and massive resources of the R software. Examples of three models (Ants <http://ccl.northwestern.edu/netlogo/models/Ants>, Butterfly (Railsback and Grimm, 2012) and Wolf-Sheep-Predation <http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation>) written using ‘NetLogoR’ are available. The ‘NetLogo’ code of the original version of these models is provided alongside. A programming guide inspired from the ‘NetLogo’ Programming Guide (<https://ccl.northwestern.edu/netlogo/docs/programming.html>) and a dictionary of ‘NetLogo’ primitives (<https://ccl.northwestern.edu/netlogo/docs/dictionary.html>) equivalences are also available.

NOTE: To increment ‘time’, these functions can use a for loop or can be integrated with a discrete event simulator, such as ‘SpaDES’ (<https://cran.r-project.org/package=SpaDES>). The suggested package ‘fastshp’ can be installed with ‘install.packages("fastshp", repos = "https://rforge.net", type = "source")’.

URL  http://netlogor.predictiveecology.org,
https://github.com/PredictiveEcology/NetLogoR/

Version  0.3.7

Date  2020-02-28

Depends  R (>= 3.5), raster

Imports  abind, car, CircStats, data.table, grDevices, Hmisc, matrixStats, methods, plyr, quickPlot (>= 0.1.2), sp, SpaDES.tools, stats, rgeos
**Suggests** fastshp, knitr, magrittr, microbenchmark, rmarkdown, sf, SpaDES.core, testthat

**License** GPL-3

**Language** en-US

**Encoding** UTF-8

**VignetteBuilder** knitr, rmarkdown

**BugReports** https://github.com/PredictiveEcology/NetLogoR/issues

**ByteCompile** yes

**LazyData** true

**RoxygenNote** 7.0.2

**Additional_repositories** https://rforge.net


**NeedsCompilation** no

**Author** Sarah Bauduin [aut] (<https://orcid.org/0000-0002-3252-5894>), Eliot J B McIntire [aut, cre] (<https://orcid.org/0000-0002-6914-8316>), Alex M Chubaty [aut] (<https://orcid.org/0000-0001-7146-8135>), Her Majesty the Queen in Right of Canada, as represented by the Minister of Natural Resources Canada [cph]

**Maintainer** Eliot J B McIntire <eliot.mcintire@canada.ca>

**Repository** CRAN

**Date/Publication** 2020-03-02 10:00:08 UTC

---

**R topics documented:**

NetLogoR-package ......................................................... 5
==.agentMatrix.character-method ..................................... 5
agentClasses-class ..................................................... 6
agentMatrix ............................................................ 6
agentMatrix-class ....................................................... 7
bk ......................................................................... 8
canMove ............................................................... 10
cbind ................................................................. 11
cellFromPxcorPycor ..................................................... 11
clearPatches ........................................................... 12
coordinates.agentMatrix-method ..................................... 13
createOTurtles ......................................................... 14
createTurtles .......................................................... 15
createWorld ............................................................ 16
die ............................................................... 18
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>diffuse</td>
<td>19</td>
</tr>
<tr>
<td>downhill</td>
<td>20</td>
</tr>
<tr>
<td>dx</td>
<td>22</td>
</tr>
<tr>
<td>dy</td>
<td>23</td>
</tr>
<tr>
<td>extent.worldNLR-method</td>
<td>24</td>
</tr>
<tr>
<td>face</td>
<td>25</td>
</tr>
<tr>
<td>fargs</td>
<td>26</td>
</tr>
<tr>
<td>fd</td>
<td>28</td>
</tr>
<tr>
<td>hatch</td>
<td>29</td>
</tr>
<tr>
<td>home</td>
<td>30</td>
</tr>
<tr>
<td>inCone</td>
<td>31</td>
</tr>
<tr>
<td>initialize.agentMatrix-method</td>
<td>33</td>
</tr>
<tr>
<td>inRadius</td>
<td>34</td>
</tr>
<tr>
<td>inspect</td>
<td>35</td>
</tr>
<tr>
<td>isNLclass</td>
<td>36</td>
</tr>
<tr>
<td>layoutCircle</td>
<td>37</td>
</tr>
<tr>
<td>left</td>
<td>38</td>
</tr>
<tr>
<td>maxNof</td>
<td>40</td>
</tr>
<tr>
<td>maxOneOf</td>
<td>41</td>
</tr>
<tr>
<td>maxPxcord</td>
<td>43</td>
</tr>
<tr>
<td>maxPycord</td>
<td>44</td>
</tr>
<tr>
<td>minNof</td>
<td>45</td>
</tr>
<tr>
<td>minOneOf</td>
<td>46</td>
</tr>
<tr>
<td>minPxcord</td>
<td>48</td>
</tr>
<tr>
<td>minPycord</td>
<td>49</td>
</tr>
<tr>
<td>moveTo</td>
<td>50</td>
</tr>
<tr>
<td>neighbors</td>
<td>51</td>
</tr>
<tr>
<td>NLall</td>
<td>52</td>
</tr>
<tr>
<td>NLany</td>
<td>54</td>
</tr>
<tr>
<td>NLcount</td>
<td>55</td>
</tr>
<tr>
<td>NLdist</td>
<td>56</td>
</tr>
<tr>
<td>NLset</td>
<td>58</td>
</tr>
<tr>
<td>NLwith</td>
<td>59</td>
</tr>
<tr>
<td>NLworldIndex</td>
<td>61</td>
</tr>
<tr>
<td>nOf</td>
<td>62</td>
</tr>
<tr>
<td>noPatches</td>
<td>63</td>
</tr>
<tr>
<td>noTurtles</td>
<td>64</td>
</tr>
<tr>
<td>numLayers,worldArray-method</td>
<td>65</td>
</tr>
<tr>
<td>of</td>
<td>67</td>
</tr>
<tr>
<td>oneOf</td>
<td>68</td>
</tr>
<tr>
<td>other</td>
<td>70</td>
</tr>
<tr>
<td>patch</td>
<td>71</td>
</tr>
<tr>
<td>patchAhead</td>
<td>72</td>
</tr>
<tr>
<td>patchAt</td>
<td>74</td>
</tr>
<tr>
<td>patchDistDir</td>
<td>75</td>
</tr>
<tr>
<td>patches</td>
<td>76</td>
</tr>
<tr>
<td>patchHere</td>
<td>77</td>
</tr>
<tr>
<td>patchLeft</td>
<td>78</td>
</tr>
</tbody>
</table>
R topics documented:

patchRight ................................................................. 80
patchSet ................................................................. 81
pExist ................................................................. 82
plot.agentMatrix .............................................................. 83
PxcorPycorFromCell .......................................................... 84
randomPxcor ............................................................. 85
randomPycor ............................................................. 86
randomXcor ............................................................. 87
randomXYcor ............................................................ 88
randomYcor ............................................................. 88
raster2world ............................................................. 89
right ................................................................. 90
setXY ................................................................. 92
show,agentMatrix-method .................................................. 93
show,worldArray-method .................................................. 94
sortOn ................................................................. 94
spdf2turtles ............................................................... 96
sprou ................................................................. 97
stackWorlds ............................................................... 98
subHeadings ............................................................... 99
tExist ................................................................. 100
towards ................................................................. 101
turtle ................................................................. 103
turtles2spdf ............................................................... 104
turtlesAt ............................................................... 105
turtleSet ............................................................... 106
turtlesOn ............................................................... 107
turtlesOwn ............................................................... 109
updateList ............................................................... 110
uphill ................................................................. 111
withMax ............................................................... 112
withMin ............................................................... 114
world2raster ............................................................. 115
worldArray-class ............................................................ 116
worldHeight ............................................................ 117
worldMatrix-class .......................................................... 118
worldNLR-class ............................................................ 119
worldWidth ............................................................. 119
wrap ................................................................. 120
[ ................................................................. 122
[ [ ................................................................. 124

Index ................................................................. 125
Description

The suggested package `fastshp` can be installed with `install.packages("fastshp",repos = "https://rforge.net",type = "source")`. The examples included with the package, are located in the R package "examples" folder, which can be found at `system.file(package = "NetLogoR","examples")`. The 3 specific R examples can be opened here: `file.edit(file.path(system.file(package = "NetLogoR","examples"),"Ants","Ants.R"))`, `file.edit(file.path(system.file(package = "NetLogoR","examples"),"Butterfly","Butterfly-1.R"))`, or `file.edit(file.path(system.file(package = "NetLogoR","examples"),"Wolf-Sheep-Predation","Wolf-Sheep-Predation.R"))`.

Author(s)

Maintainer: Eliot J B McIntire <eliot.mcintire@canada.ca> (ORCID)
Authors:
- Sarah Bauduin <sarahbauduin@hotmail.fr> (ORCID)
- Alex M Chubaty <alex.chubaty@gmail.com> (ORCID)

Other contributors:
- Her Majesty the Queen in Right of Canada, as represented by the Minister of Natural Resources Canada [copyright holder]

See Also

Useful links:
- [http://netlogor.predictiveecology.org](http://netlogor.predictiveecology.org)
- [https://github.com/PredictiveEcology/NetLogoR/](https://github.com/PredictiveEcology/NetLogoR/)

==,agentMatrix,character-method

Relational Operators

Description

Binary operators which allow the comparison of values in an agentMatrix.
Usage

```r
## S4 method for signature 'agentMatrix,character'
e1 == e2

## S4 method for signature 'agentMatrix,numeric'
e1 == e2
```

Arguments

- `e1` An `agentMatrix` object.
- `e2` atomic vector, symbol, call, or other object for which methods have been written.

---

Description

Both these types can be used by NetLogoR to describe turtle agents.

Both these types can be used by NetLogoR to describe turtle agents.

Author(s)

Eliot McIntire

Eliot McIntire

---

Description

This is a fast alternative to the `SpatialPointsDataFrame`. It is meant to replace that functionality, though there are not as many methods (yet). The object is primarily a numeric matrix. Any character column passed to ... will be converted to a numeric, using `as.factor` internally, and stored as a numeric. Methods using this class will automatically convert character queries to the correct numeric alternative.

Usage

```r
agentMatrix(..., coords)

## S4 method for signature 'matrix'
agentMatrix(..., coords)

## S4 method for signature 'missing'
agentMatrix(..., coords)
```
agentMatrix-class

Arguments

... Vectors, a data.frame, or a matrix of extra columns to add to the coordinates, or a SpatialPointsDataFrame.

coords A matrix with 2 columns representing x and y coordinates

Author(s)

Eliot McIntire

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#clear-turtles

Examples

newAgent <- agentMatrix(
  coords = cbind(pxcor = c(1, 2, 5), pycor = c(3, 4, 6)),
  char = letters[c(1, 2, 6)],
  nums2 = c(4.5, 2.6, 2343),
  char2 = LETTERS[c(4, 24, 3)],
  nums = 5:7)

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
  data = runif(25))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
Examples

```r
newAgent <- new("agentMatrix",
  coords = cbind(pxcor = c(1, 2, 5), pycor = c(3, 4, 6)),
  char = letters[c(1, 2, 6)],
  nums2 = c(4.5, 2.6, 2343),
  char2 = LETTERS[c(4, 24, 3)],
  nums = 5:7)
newAgent <- new("agentMatrix",
  coords = cbind(pxcor = c(1, 2, 5), pycor = c(3, 4, 6)),
  char = letters[c(1, 2, 6)],
  nums2 = c(4.5, 2.6, 2343),
  char2 = LETTERS[c(4, 24, 3)],
  nums = 5:7)

# compare speeds -- about 5x faster
if(require(microbenchmark)) {
  microbenchmark(times = 499,
    spdf = {SpatialPointsDataFrame(
      coords = cbind(pxcor = c(1, 2, 5), pycor = c(3, 4, 6)),
      data = data.frame(
        char = letters[c(1, 2, 6)],
        nums2 = c(4.5, 2.6, 2343),
        char2 = LETTERS[c(4, 24, 3)],
        nums = 5:7))},
    agentMat = {agentMatrix(
      coords = cbind(pxcor = c(1, 2, 5),
      pycor = c(3, 4, 6)),
      char = letters[c(1, 2, 6)],
      num2 = c(4.5, 2.6, 2343),
      char2 = LETTERS[c(4, 24, 3)],
      nums = 5:7)}},
    agentMatDirect = {new("agentMatrix",
      coords = cbind(pxcor = c(1, 2, 5),
      pycor = c(3, 4, 6)),
      char = letters[c(1, 2, 6)],
      nums2 = c(4.5, 2.6, 2343),
      char2 = LETTERS[c(4, 24, 3)],
      nums = 5:7)})
}
```

bk            Move backward

Description

Move the turtles backward of their headings' directions.
Usage

bk(turtles, dist, world, torus = FALSE, out = TRUE)

## S4 method for signature 'agentMatrix,numeric'
bk(turtles, dist, world, torus = FALSE, out = TRUE)

Arguments

turtles  AgentMatrix object representing the moving agents.
dist     Numeric. Vector of distances to move. Must be of length 1 or of length turtles.
world    WorldMatrix or worldArray object.
torus    Logical to determine if the world is wrapped. Default is torus = FALSE.
out      Logical. Determine if a turtle should move when torus = FALSE and its ending
          position will be outside of the world’s extent. Default is out = TRUE.

Details

If torus = FALSE and out = TRUE, world does not need to be provided.

If a distance to move leads a turtle outside of the world’s extent and torus = TRUE, the turtle
is relocated on the other side of the world, inside its extent; if torus = FALSE and out = TRUE, the
turtle moves past the world’s extent; if torus = FALSE and out = FALSE, the turtle does not
move at all. In the event that a turtle does not move, its previous coordinates are still updated
with its position before running bk() (i.e., its current position).

If a given dist value is negative, then the turtle moves forward.

The turtles’ headings are not affected by the function (i.e., the turtles do not face backward).

Value

AgentMatrix representing the turtles with updated coordinates and updated data for their previ-
ous coordinates prevX and prevY.

Author(s)

Sarah Bauduin

References

and Computer-Based Modeling, Northwestern University. Evanston, IL.

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#back
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#jump
canMove

Can the turtles move?

Description

Report TRUE if a turtle can move the given distance without leaving the world’s extent, report FALSE otherwise.

Usage

canMove(world, turtles, dist)

## S4 method for signature 'worldNLR,agentMatrix,numeric'

canMove(world, turtles, dist)

Arguments

- world: WorldMatrix or worldArray object.
- turtles: AgentMatrix object representing the moving agents.
- dist: Numeric. Vector of distances to move. Must be of length 1 or of length turtles.

Value

Logical. Vector of length turtles.

Author(s)

Sarah Bauduin

References

cbind

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#can-move

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, world = w1)
canMove(world = w1, turtles = t1, dist = 1:10)

---

cbind

Combine R Objects by Rows or Columns

Description

Take a sequence of agentMatrix arguments and combine by columns or rows, respectively. This will take the coordinates of the first argument and remove the coordinates of the second object.

Usage

## S3 method for class 'agentMatrix'
cbind(..., deparse.level)

## S3 method for class 'agentMatrix'
rbind(..., deparse.level = 1)

Arguments

... Two agentMatrix objects
deparse.level See cbind

---

cellFromPxcorPycor

Cells numbers from patches coordinates

Description

Report the cells numbers as defined for a Raster* object given the patches coordinates pxcor and pycor.

Usage

cellFromPxcorPycor(world, pxcor, pycor)

## S4 method for signature 'worldNLR,numeric,numeric'
cellFromPxcorPycor(world, pxcor, pycor)
Arguments

world WorldMatrix or worldArray object.
pxcor Integer. Vector of patches pxcor coordinates. Must be of length 1 or of the same length as pycor.
pycor Integer. Vector of patches pycor coordinates. Must be of length 1 or of the same length as pxcor.

Value

Numeric. Vector of cells number.

Author(s)

Sarah Bauduin

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
cellFromPxcorPycor(world = w1, pxcor = 0, pycor = 9)
cellFromPxcorPycor(world = w1, pxcor = c(0, 1, 2), pycor = 0)

clearPatches WorldMatrix object with NA values for all patches.

Description

Reset all patches values to NA.

Usage

clearPatches(world)

## S4 method for signature 'worldMatrix'
clearPatches(world)

## S4 method for signature 'worldArray'
clearPatches(world)

Arguments

world WorldMatrix or worldArray object.

Value

WorldMatrix object with NA values for all patches.
### coordinates,agentMatrix-method

**Set spatial coordinates**

---

**Description**

Set spatial coordinates

**Usage**

```r
## S4 method for signature 'agentMatrix'
coordinates(obj, ...)
```

**Arguments**

- `obj` documentation needed
- `...` additional arguments that may be used by particular methods description needed

---

**Examples**

```r
w1 <- createWorld()
w1 <- NLset(world = w1, agents = patches(w1), val = runif(NLcount(patches(w1))))
w1Val <- of(world = w1, agents = patches(w1))
summary(w1Val)

w1 <- clearPatches(w1)
w1Val <- of(world = w1, agents = patches(w1))
summary(w1Val)
```

---

**Author(s)**

Sarah Bauduin

**References**


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#clear-patches
createOTurtles  

Create ordered turtles

Description

Create n turtles at the center of the world with their headings evenly distributed.

Usage

createOTurtles(n, world, breed, color)

## S4 method for signature 'numeric'
createOTurtles(n, world, breed, color)

Arguments

- **n**: Integer.
- **world**: WorldMatrix or worldArray object.
- **breed**: Character. Vector of breed names. Must be of length 1 or of length n. If missing, `breed = "turtle"` for all turtles.
- **color**: Character. Vector of color names. Must be of length n. If missing, colors are assigned using the function `rainbow(n)`.

Details

The identity of the turtles is defined by their who number. This numbering starts at 0 and increments by 1.

The coordinates from the previous time step are stored in `prevX` and `prevY`. The initial values are NA.

Value

AgentMatrix object of length n with data for the turtles being: `xcor, ycor, who, heading, prevX, prevY, breed, and color`.

Author(s)

Sarah Bauduin and Eliot McIntire

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#create-ordered-turtles
**createTurtles**

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = runif(25))
t1 <- createOTurtles(n = 10, world = w1)
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

t1 <- fd(turtles = t1, dist = 1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

**createTurtles**

Create `n` moving agents with a set of defined variables.

**Usage**

```r
createTurtles(n, coords, world, heading, breed, color)
```

**Arguments**

- **n**  
  Integer.

- **coords**  
  Matrix (n\(\times\)2) with the first column `xcor` and the second column `ycor` representing the turtles initial locations. `nrow(coords)` must be equal to 1 or to n. Given coordinates must be inside the world’s extent. If missing, turtles are put in the center of the world.

- **world**  
  `WorldMatrix` or `worldArray` object.

- **heading**  
  Numeric. Vector of values between 0 and 360. Must be of length 1 or of length `n`. If missing, a random heading is assigned to each turtle.

- **breed**  
  Character. Vector of breed names. Must be of length 1 or of length `n`. If missing, `breed = "turtle"` for all turtles.

- **color**  
  Character. Vector of color names. Must be of length `n`. If missing, colors are assigned using the function `rainbow(n)`.
createWorld

Description

Create a world of patches of class worldMatrix.

createWorld

Create a world

Details

If coords is provided, world must not be provided.

The identity of the turtles is defined by their who number. This numbering starts at 0 and increments by 1.

The coordinates from the previous time step are stored in prevX and prevY. The initial values are NA.

Value

AgentMatrix object of length n with data for the turtles being: xcor, ycor, who, heading, prevX, prevY, breed, and color.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#create-turtles

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = runif(25))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
createWorld

Usage
createWorld(minPxcor, maxPxcor, minPycor, maxPycor, data = NA)

## S4 method for signature 'numeric,numeric,numeric,numeric,ANY'
createWorld(minPxcor, maxPxcor, minPycor, maxPycor, data = NA)

## S4 method for signature 'missing,missing,missing,missing,missing'
createWorld()

Arguments

minPxcor    Integer. Minimum pxcor for the patches (world's left border).
maxPxcor    Integer. Maximum pxcor for the patches (world's right border).
minPycor    Integer. Minimum pycor for the patches (world's bottom border).
maxPycor    Integer. Maximum pycor for the patches (world's top border).
data        Vector of length 1 or length \((\text{maxPxcor} - \text{minPxcor} + 1) * (\text{maxPycor} - \text{minPycor} + 1)\). Default is NA.

Details

If data is provided, values are assigned by rows.
If no parameters value are provided, default values are: \text{minPxcor} = -16, \text{maxPxcor} = 16, \text{minPycor} = -16, and \text{maxPycor} = 16.
See help("worldMatrix-class") for more details on the worldMatrix class.

Value

WorldMatrix object composed of \((\text{maxPxcor} - \text{minPxcor} + 1) * (\text{maxPycor} - \text{minPycor} + 1)\) patches (i.e., matrix cells).

Author(s)

Sarah Bauduin, Eliot McIntire, and Alex Chubaty

References


Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = 1:25)
plot(w1)


**die**

*Kill turtles*

**Description**

Kill selected turtles.

**Usage**

```r
die(turtles, who)
```

```r
## S4 method for signature 'agentMatrix,numeric'
die(turtles, who)
```

**Arguments**

<table>
<thead>
<tr>
<th>turtles</th>
<th>AgentMatrix object representing the moving agents.</th>
</tr>
</thead>
<tbody>
<tr>
<td>who</td>
<td>Integer. Vector of the who numbers for the selected turtles.</td>
</tr>
</tbody>
</table>

**Details**

The who numbers of the remaining turtles are unchanged.

**Value**

AgentMatrix representing the turtles with the selected ones removed.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#die](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#die)

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, world = w1)
NLcount(t1)
t1 <- die(turtles = t1, who = c(2, 3, 4))
NLcount(t1)
```
diffuse  

Diffuse values in a world

Description

Each patch gives an equal share of a portion of its value to its neighbor patches.

Usage

```r
diffuse(world, pVar, share, nNeighbors, torus = FALSE)
## S4 method for signature 'worldMatrix,missing,numeric,numeric'
diffuse(world, share, nNeighbors, torus)
## S4 method for signature 'worldArray,character,numeric,numeric'
diffuse(world, pVar, share, nNeighbors, torus = FALSE)
```

Arguments

- **world**: WorldMatrix or worldArray object.
- **pVar**: Character. If the world is a worldArray object, pVar is the name of the layer to use to define the patches values. pVar must not be provided if the world is a worldMatrix object.
- **share**: Numeric. Value between 0 and 1 representing the portion of the patches values to be diffused among the neighbors.
- **nNeighbors**: Integer: 4 or 8. Represents the number of neighbor patches considered.
- **torus**: Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

What is given is lost for the patches.

If torus = TRUE, all patches have nNeighbors patches around them, which some may be on the other sides of the world. If torus = FALSE, patches located on the edges of the world have less than nNeighbors patches around them. However, each neighbor still gets 1/4 or 1/8 of the shared amount and the diffusing patch keeps the leftover.

Value

WorldMatrix or worldArray object with patches values updated.

Author(s)

Sarah Bauduin
References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#diffuse
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#diffuse4

Examples
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
data = sample(1:3, size = 25, replace = TRUE))
plot(w1)
# Diffuse 50% of each patch value to its 8 neighbors
w2 <- diffuse(world = w1, share = 0.5, nNeighbors = 8)
plot(w2)

---

downhill  Move downhill

Description
Move the turtles to their neighboring patch with the lowest value.

Usage
downhill(world, pVar, turtles, nNeighbors, torus = FALSE)

## S4 method for signature 'worldMatrix,missing,agentMatrix,numeric'
downhill(world, turtles, nNeighbors, torus)

## S4 method for signature 'worldArray,character,agentMatrix,numeric'
downhill(world, pVar, turtles, nNeighbors, torus = FALSE)

Arguments
world  WorldMatrix or worldArray object.
pVar   Character. If the world is a worldArray object, pVar is the name of the layer to use to define the patches values. pVar must not be provided if the world is a worldMatrix object.
turtles AgentMatrix object representing the moving agents.
nNeighbors Integer: 4 or 8. Represents the number of neighbor patches considered.
torus   Logical to determine if the world is wrapped. Default is torus = FALSE.
downhill

Details

If no neighboring patch has a smaller value than the patch where the turtle is currently located on, the turtle stays on this patch. It still moves to the patch center if it was not already on it.

If there are multiple neighboring patches with the same lowest value, the turtle chooses one patch randomly.

If a turtle is located on a patch on the edge of the world and torus = FALSE, it has fewer neighboring patches as options to move than nNeighbors; if torus = TRUE, the turtle can move on the other side of the world to move downhill and its choice of neighboring patches is always equals to nNeighbors.

Value

AgentMatrix representing the turtles with updated coordinates and updated data for their heading values and previous coordinates prevX and prevY.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#downhill

Examples

w1 <- createWorld(minPxcor = 1, maxPxcor = 10, minPycor = 1, maxPycor = 10, data = runif(100))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

w1 <- downhill(world = w1, turtles = t1, nNeighbors = 8)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
Description

Report the amount by which the turtles’ coordinates xcor would change if the turtles were to move forward the given distances with their current headings.

Usage

dx(turtles, dist = 1)

## S4 method for signature 'agentMatrix,numeric'
dx(turtles, dist = 1)

## S4 method for signature 'agentMatrix,missing'
dx(turtles)

Arguments

turtles AgentMatrix object representing the moving agents.
dist Numeric. Vector of distances the turtles would have to move forward to compute the increment values. Must be of length 1 or of length turtles. The default value is dist = 1.

Details

Report the sine of the turtles’ heading multiplied by the dist values. Heading 0 is north and angles are calculated in degrees in a clockwise manner.

Value

Numeric. Vector of length turtles.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#dx
**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createOTurtles(world = w1, n = 10)
dx(turtles = t1)
```

**Description**

Report the amount by which the turtles' coordinates ycor would change if the turtles were to move forward the given distances with their current headings.

**Usage**

```r
dy(turtles, dist = 1)
```

```r
## S4 method for signature 'agentMatrix,numeric'
dy(turtles, dist = 1)
```

```r
## S4 method for signature 'agentMatrix,missing'
dy(turtles)
```

**Arguments**

- `turtles` AgentMatrix object representing the moving agents.
- `dist` Numeric. Vector of distances the turtles would have to move forward to compute the increment values. Must be of length 1 or of length `turtles`. The default value is `dist = 1`.

**Details**

Report the cosine of the turtles' heading multiplied by the dist values. Heading 0 is north and angles are calculated in degrees in a clockwise manner.

**Value**

Numeric. Vector of length `turtles`.

**Author(s)**

Sarah Bauduin

**References**

extent,worldNLR-method

Bounding box and extent methods for NetLogoR classes

Description

Same as bbox and extent

Usage

```r
## S4 method for signature 'worldNLR'
extent(x, ...)
```

```
## S4 method for signature 'agentMatrix'
extent(x, ...)
```

```
.bboxCoords(coords)
```

```
## S4 method for signature 'agentMatrix'
bbox(obj)
```

```
bbox(obj) <- value
```

```
## S4 replacement method for signature 'agentMatrix,matrix'
bbox(obj) <- value
```

```
## S4 method for signature 'worldNLR'
bbox(obj)
```

Arguments

- `x`  
  Raster* or Extent object, a matrix, or a vector of four numbers

- `...`  
  Additional arguments. When `x` is a single number representing `xmin`, you can pass three additional numbers (`xmax`, `ymin`, `ymax`).

  When `x` is a Raster* object, you can pass four additional arguments to crop the extent: `r1`, `r2`, `c1`, `c2`, representing the first and last row and column number.

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createOTurtles(world = w1, n = 10)
dy(turtles = t1)
```
coords documentation needed

obj object deriving from class "Spatial", or one of classes: "Line", "Lines", "Polygon" or "Polygons", or ANY, which requires obj to be an array with at least two columns

value 2x2 matrix representing the bounding box. See bbox

---

description

Set the turtles' heading towards agents2.

Usage

```r
face(turtles, agents2, world, torus = FALSE)
```

## S4 method for signature 'agentMatrix, matrix'

```r
# face(turtles, agents2, world, torus = FALSE)
```

Arguments

- **turtles**AgentMatrix object representing the moving agents.
- **agents2**Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents, or Matrix (ncol = 2) with the first column x and the second column y representing locations coordinates.
- **world**WorldMatrix or worldArray object.
- **torus**Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

The number of agents/locations in agents2 must be equal to 1 or to the length of turtles.

If torus = FALSE, world does not need to be provided.

If torus = TRUE and the distance from one turtles to its corresponding agent/location agents2 is smaller around the sides of the world than across it, then the direction to the agent/location agents2 going around the sides of the world is given to the turtle.

If a turtle is facing its own location, its heading does not change.

Value

AgentMatrix representing the turtles with updated headings.
Author(s)
Sarah Bauduin

References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#face
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#facexy

Examples
```
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = runif(25))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
t1 <- face(turtles = t1, agents2 = cbind(x = 0, y = 0))
t1 <- fd(turtles = t1, dist = 0.5)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

### fargs

**Function arguments**

**Description**
Function arguments

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>Integer.</td>
</tr>
<tr>
<td>world</td>
<td>WorldMatrix or worldArray object.</td>
</tr>
<tr>
<td>torus</td>
<td>Logical to determine if the world is wrapped. Default is torus = FALSE.</td>
</tr>
<tr>
<td>minPxcor</td>
<td>Integer. Minimum pxcor for the patches (world’s left border).</td>
</tr>
<tr>
<td>maxPxcor</td>
<td>Integer. Maximum pxcor for the patches (world’s right border).</td>
</tr>
<tr>
<td>minPycor</td>
<td>Integer. Minimum pycor for the patches (world’s bottom border).</td>
</tr>
<tr>
<td>maxPycor</td>
<td>Integer. Maximum pycor for the patches (world’s top border).</td>
</tr>
<tr>
<td>pxcor</td>
<td>Integer. Vector of patches pxcor coordinates. Must be of length 1 or of the same length as pycor.</td>
</tr>
</tbody>
</table>
fargs

pycor  
Integer. Vector of patches pycor coordinates. Must be of length 1 or of the same length as pxcor.

cellNum  
Integer. Vector of cells number.

pVar  
Character. If the world is a worldArray object, pVar is the name of the layer to use to define the patches values. pVar must not be provided if the world is a worldMatrix object.

turtles  
AgentMatrix object representing the moving agents.

patches  
Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates.

agents  
Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

agents2  
Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents, or Matrix (ncol = 2) with the first column x and the second column y representing locations coordinates.

nNeighbors  
Integer: 4 or 8. Represents the number of neighbor patches considered.

dx  
Numeric. Vector of distances to the east (right) from the agents. If dx is negative, the distance to the west (left) is computed. dx must be of length 1 or of the same length as number of patches or turtles in agents.

dy  
Numeric. Vector of distances to the north (up) from the agents. If dy is negative, the distance to the south is computed (down). dy must be of length 1 or of the same length as number of patches or turtles in agents.

color  
Character. Vector of color names. Must be of length n. If missing, colors are assigned using the function rainbow(n).

who  
Integer. Vector of the who numbers for the selected turtles.

breed  
Characters. Vector of breed names for the selected turtles. If missing, there is no distinction based upon breed.

var  
Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles' variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

val  
Numeric or character. Vector of any length.
**fd**

*Move forward*

**Description**

Move the turtles forward with their headings as directions.

**Usage**

```r
df(turtles, dist, world, torus = FALSE, out = TRUE)
```

```
## S4 method for signature 'agentMatrix,numeric'
fd(turtles, dist, world, torus = FALSE, out = TRUE)
```

**Arguments**

- `turtles` AgentMatrix object representing the moving agents.
- `dist` Numeric. Vector of distances to move. Must be of length 1 or of length `turtles`.
- `world` WorldMatrix or worldArray object.
- `torus` Logical to determine if the world is wrapped. Default is `torus = FALSE`.
- `out` Logical. Determine if a turtle should move when `torus = FALSE` and its ending position will be outside of the world’s extent. Default is `out = TRUE`.

**Details**

If `torus = FALSE` and `out = TRUE`, `world` does not need to be provided.

If a distance to move leads a turtle outside of the world’s extent and `torus = TRUE`, the turtle is relocated on the other side of the world, inside its extent; if `torus = FALSE` and `out = TRUE`, the turtle moves past the world’s extent; if `torus = FALSE` and `out = FALSE`, the turtle does not move at all. In the event that a turtle does not move, its previous coordinates are still updated with its position before running `fd()` (i.e., its current position).

If a given `dist` value is negative, then the turtle moves backward.

**Value**

AgentMatrix representing the turtles with updated coordinates and updated data for their previous coordinates `prevX` and `prevY`.

**Author(s)**

Sarah Bauduin

**References**

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#forward
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#jump

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
                  data = runif(25))
t1 <- createOTurtles(n = 10, world = w1)
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

t1 <- fd(turtles = t1, dist = 1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

hatch | Hatch new turtles

Description

Create new turtles from parent turtles.

Usage

```r
hatch(turtles, who, n, breed)
```

## S4 method for signature 'agentMatrix,numeric,numeric'
```r
hatch(turtles, who, n, breed)
```

Arguments

- **turtles**: AgentMatrix object representing the moving agents.
- **who**: Integer. Vector of the who numbers for the selected turtles.
- **n**: Integer. Vector of length 1 or of length who. Number of new turtles to create for each parent.
- **breed**: Character. One breed name. If missing, the created turtles are of the same breed as their parent turtle.

Details

The parent turtle must be contained in the turtles.

The created turtles inherit of all the data from the parent turtle, except for the breed if specified otherwise, and for the who numbers. The who" numbers of the turtles created take on following the highest who number among the turtles.

All new hatched turtles are placed at the end of the agentMatrix object.
Value

AgentMatrix representing the turtles with the new hatched ones.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#hatch

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, world = w1)
NLcount(t1)
t1 <- hatch(turtles = t1, who = 0, n = 2)
NLcount(t1)
```

Description

Move the turtles back home.

Usage

```r
home(world, turtles, home)
```

Arguments

- `world` WorldMatrix or worldArray object.
- `turtles` AgentMatrix object representing the moving agents.
inCone

Character. Can take one of the following options to define where to relocate the turtles:

- `home = "home0"` will place the turtles at the location $x = 0, y = 0$.
- `home = "center"` will place the turtles at the center of the world.
- `home = "pCorner"` will place the turtles at the center of the patch located in the left bottom corner of the world.
- `home = "corner"` will place the turtles at the left bottom corner of the world.

Value

`AgentMatrix` representing the turtles with updated coordinates and updated data for their previous coordinates `prevX` and `prevY`.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#home

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = runif(25))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
plot(w1)
points(t1, col = "black", pch = 16)

points(t1, col = "red", pch = 16)
t1 <- home(world = w1, turtles = t1, home = "pCorner")
```

Description

Report the agents within the "cone of vision" in front of each one of the turtles.
Usage

\texttt{inCone(turtles, radius, angle, agents, world, torus = FALSE)}

## S4 method for signature 'agentMatrix, numeric, numeric, matrix'
\texttt{inCone(turtles, radius, angle, agents, world, torus = FALSE)}

Arguments

- \texttt{turtles} \texttt{AgentMatrix} object representing the moving agents.
- \texttt{radius} Numeric. Vector of distances from turtles to locate agents. Must be of length 1 or of length \texttt{turtles}.
- \texttt{angle} Numeric. Vector of angles to define the size of the cone of vision for the turtles. The cone of vision is defined between the direction of their headings minus \texttt{angle} / 2 to the direction of their headings plus \texttt{angle} / 2. Must be of length 1 or of length \texttt{turtles}.
- \texttt{agents} Matrix (\texttt{ncol} = 2) with the first column \texttt{pxcor} and the second column \texttt{pycor} representing the patches coordinates, or \texttt{AgentMatrix} object representing the moving agents.
- \texttt{world} \texttt{WorldMatrix} or \texttt{worldArray} object.
- \texttt{torus} Logical to determine if the world is wrapped. Default is \texttt{torus = FALSE}.

Details

Agents are reported if there are within \texttt{radius} distance of the turtle and their direction from the turtle is within \([-\text{angle}, +\text{angle}]\) of the turtle’s heading.

Distances to patches are calculated to their center.

If \texttt{torus = FALSE}, \texttt{world} does not need to be provided.

If \texttt{torus = TRUE}, the \texttt{radius} distances are calculated around the sides of the world to select agents.

Value

Matrix (\texttt{ncol} = 3) with the first column \texttt{pxcor} and the second column \texttt{pycor} representing the coordinates of the patches among \texttt{agents2} within the cone of vision of each of the turtles which are represented by the \texttt{id} column, if \texttt{agents} are patches, or

Matrix (\texttt{ncol} = 2) with the first column \texttt{who} representing the \texttt{who} numbers of the turtles among \texttt{agents2} within the cone of vision of each of the turtles which are represented by the \texttt{id} column, if \texttt{agents} are turtles.

Author(s)

Sarah Bauduin

References

**initialize.agentMatrix-method**

*Initialize for agentMatrix Class*

---

### Description

To create a new `agentMatrix` object.

### Usage

```r
## S4 method for signature 'agentMatrix'
initialize(.Object = "agentMatrix", coords, ..., levelsAM)
```

### Arguments

- `.Object`  
  An object: see the “Initialize Methods” section.

- `coords`  
  2 column matrix of coordinates

- `...`  
  arguments to specify properties of the new object, to be passed to `initialize()`.

- `levelsAM`  
  A list with named character vectors. Each name should match with elements in `...`, and each character vector should be the length of unique elements in the `...` element.

---

**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#in-cone](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#in-cone)

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))

p1 <- inCone(turtles = t1, radius = 2, agents = patches(w1), angle = 90)
t2 <- inCone(turtles = turtle(t1, who = 0), radius = 2, angle = 90, agents = t1)
```
inRadius (Agents in radius)

Description
Report the patches or turtles among agents2 within given distances of each of the agents. Currently, this function multiplies radius by 1.0000001 so that the response of inRadius is inclusive.

Usage
inRadius(agents, radius, agents2, world, torus = FALSE)

## S4 method for signature 'matrix,numeric,matrix'
inRadius(agents, radius, agents2, world, torus = FALSE)

Arguments
- **agents**: Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
- **radius**: Numeric. Vector of distances from agents to locate agents2. Must be of length 1 or of length agents.
- **agents2**: Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
- **world**: WorldMatrix or worldArray object.
- **torus**: Logical to determine if the world is wrapped. Default is torus = FALSE.

Details
Distances from/to patches are calculated from/to their center.
If torus = FALSE, world does not need to be provided.
If torus = TRUE, the radius distances are calculated around the sides of the world to select agents2.

Value
Matrix (ncol = 3) with the first column pxcor and the second column pycor representing the coordinates of the patches among agents2 within radius distances for each agents which are represented by the id column, if agents2 are patches, or
Matrix (ncol = 2) with the first column who representing the who numbers of the turtles among agents2 within radius distances for each agents which are represented by the id column, if agents2 are turtles.

Author(s)
Sarah Bauduin
References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#in-radius

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))

p1 <- inRadius(agents = patch(w1, 0, 0), radius = 2, agents2 = patches(w1))
t2 <- inRadius(agents = patch(w1, 0, 0), radius = 2, agents2 = t1)
p2 <- inRadius(agents = t1, radius = 2, agents2 = patches(w1))
t3 <- inRadius(agents = turtle(t1, who = 0), radius = 2, agents2 = t1)
```

inspect

`inspect`  
*Inspect turtles*

Description

Display all variables values for the selected individuals among the turtles.

Usage

```r
inspect(turtles, who)
```

## S4 method for signature 'agentMatrix,numeric'
```
inspect(turtles, who)
```

Arguments

- `turtles`  
  AgentMatrix object representing the moving agents.

- `who`  
  Integer. Vector of the who numbers for the selected turtles.

Value

Dataframe (`nrow = length(who)`) of the variables of the selected individuals among the turtles.

Author(s)

Sarah Bauduin
References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#inspect

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createOTurtles(world = w1, n = 10)
inspect(turtles = t1, who = c(2, 3))

<table>
<thead>
<tr>
<th>isNLclass</th>
<th>Type of object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Report TRUE if the agents is of the class tested, report FALSE otherwise.

Usage

isNLclass(agents, class)

## S4 method for signature 'matrix,character'

isNLclass(agents, class)

Arguments

agents    Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

class     Character. Can take one of the following options to define the class: "agent", "agentset", "patch", "patchset", "turtle" or "turtleset".

Details

Careful! The class tested does not correspond to actual R classes.

agents is "patch" if it is a matrix (ncol = 2) with the first column pxcor and the second column pycor with only one row. agents is "patcheset" if the matrix has more than one row.

agents is "turtle" if it is an agentMatrix containing only one turtle. agents is "turtleset" if the agentMatrix contains more than one turtle.

agents is "agent" if it is either "patch" or "turtle". agents is "agentset" if it is either "patcheset" or "turtleset".
Value

Logical. TRUE if agents is of the class tested.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#is-of-type

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
                   heading = sample(1:3, size = 10, replace= TRUE))
isNLclass(agents = patches(w1), class = "patch")
isNLclass(agents = patches(w1), class = "patcheset")
isNLclass(agents = t1, class = "agentset")
isNLclass(agents = t1, class = "turtleset")

layoutCircle

Layout turtles on a circle

Description

Relocate the turtles on a circle centered on the world.

Usage

layoutCircle(world, turtles, radius, torus = FALSE)

## S4 method for signature 'worldNLR,agentMatrix,numeric'
layoutCircle(world, turtles, radius, torus = FALSE)

Arguments

world WorldMatrix or worldArray object.
turtles AgentMatrix object representing the moving agents.
radius Numeric. Radius of the circle.
torus Logical to determine if the world is wrapped. Default is torus = FALSE.
Details

The turtles point outwards.

If the radius value leads turtles outside of the world’s extent and torus = TRUE, they are relocated on the other sides of the world, inside its extent; if torus = FALSE, the turtles are located past the world’s extent.

Value

AgentMatrix representing the turtles with updated coordinates and updated data for their heading values and previous coordinates prevX and prevY.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#layout-circle

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = runif(100))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
plot(w1)
points(t1, col = "black", pch = 16)

t1 <- layoutCircle(world = w1, turtles = t1, radius = 3)
points(t1, col = "red", pch = 16)
```

Description

Rotate to the left

Rotate the turtles’s headings to the left of angle degrees.
Usage

left(turtles, angle)

## S4 method for signature 'agentMatrix,numeric'
left(turtles, angle)

Arguments

turtles AgentMatrix object representing the moving agents.
angle Numeric. Vector of angles in degrees by which to rotate the turtles’ headings. Must be of length 1 or of length turtles.

Details

If a given angle value is negative, then the turtle rotates to the right.

Value

AgentMatrix representing the turtles with updated heading values.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#left

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, world = w1)
of(agents = t1, var = "heading")
t1 <- left(turtles = t1, angle = 180)
of(agents = t1, var = "heading")
maxNof

N agents with maximum

Description

Report the n patches or turtles among agents which have their variable among the maximum values.

Usage

maxNof(agents, n, world, var)

## S4 method for signature 'matrix,numeric,worldMatrix,missing'
maxNof(agents, n, world)

## S4 method for signature 'matrix,numeric,worldArray,character'
maxNof(agents, n, world, var)

## S4 method for signature 'agentMatrix,numeric,missing,character'
maxNof(agents, n, var)

Arguments

agents  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

n  Integer.

world  WorldMatrix or worldArray object.

var  Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles’ variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

Details

world must not be provided if agents are turtles.

If there is a tie that would make the number of returned patches or turtles larger than n, it is broken randomly.
maxOneOf

Value

Matrix \((n_{col} = 2, n_{row} = n)\) with the first column \(pxcor\) and the second column \(pycor\) representing the coordinates of the \(n\) patches among the agents which have their variable values among the maximum values among the agents, or

AgentMatrix of length \(n\) representing the turtles among the agents which have their \(var\) values among the maximum values among the agents.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#max-n-of

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, 
data = sample(1:10, size = 25, replace = TRUE))
plot(w1)
p1 <- maxNof(agents = patches(w1), n = 6, world = w1)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
    heading = sample(1:5, size = 10, replace = TRUE))
t2 <- maxNof(agents = t1, n = 5, var = "heading")

Description

Report one patch or one turtle among agents which has its variable equals to the maximum value.
maxOneOf

Usage

maxOneOf(agents, world, var)

## S4 method for signature 'matrix,worldMatrix,missing'
maxOneOf(agents, world)

## S4 method for signature 'matrix,worldArray,character'
maxOneOf(agents, world, var)

## S4 method for signature 'agentMatrix,missing,character'
maxOneOf(agents, var)

Arguments

agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
world WorldMatrix or worldArray object.
var Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles' variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

Details

world must not be provided if agents are turtles.

If there are several patches or turtles among agents with their variable equal to the maximum value, one is chosen randomly. To access to all patches or turtles among agents which have their variable equal to the maximum value, use withMax().

Value

Matrix (ncol = 2, nrow = 1) with the first column pxcor and the second column pycor representing the coordinates of the patch (or of one of the patches) among the agents which has its variable equals to the maximum value among the agents, or

AgentMatrix of length 1 representing the turtle (or one of the turtles) among the agents which has its variable var equals to the maximum value among the agents.

Author(s)

Sarah Bauduin
References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#max-one-of

Examples
# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
data = sample(1:5, size = 25, replace = TRUE))
plot(w1)
p1 <- maxOneOf(agents = patches(w1), world = w1)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
heading = sample(1:3, size = 10, replace= TRUE))
t2 <- maxOneOf(agents = t1, var = "heading")

maxPxcor

<table>
<thead>
<tr>
<th>maxPxcor</th>
<th>Maximum pxcor</th>
</tr>
</thead>
</table>

Description
Report the patches maximum pxcor in the world.

Usage
maxPxcor(world)

## S4 method for signature 'worldNLR'
maxPxcor(world)

Arguments

| world | WorldMatrix or worldArray object. |

Value
Integer.

Author(s)
Sarah Bauduin
References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#max-pcor

Examples

```r
w1 <- createWorld()
maxPxcor(w1)
```

---

### maxPycor

**Maximum pycor**

**Description**

Report the patches maximum pycor in the world.

**Usage**

```r
maxPycor(world)
```

#### S4 method for signature 'worldNLR'

```r
maxPycor(world)
```

**Arguments**

- `world` WorldMatrix or worldArray object.

**Value**

Integer.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#max-pcor
Examples

```r
w1 <- createWorld()
maxPycor(w1)
```

<table>
<thead>
<tr>
<th>minNof</th>
<th>N agents with minimum</th>
</tr>
</thead>
</table>

Description

Report the n patches or turtles among agents which have their variable among the minimum values.

Usage

```r
minNof(agents, n, world, var)
```

## S4 method for signature 'matrix,numeric,worldMatrix,missing'
```
minNof(agents, n, world)
```

## S4 method for signature 'matrix,numeric,worldArray,character'
```
minNof(agents, n, world, var)
```

## S4 method for signature 'agentMatrix,numeric,missing,character'
```
minNof(agents, n, var)
```

Arguments

- `agents` Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or `AgentMatrix` object representing the moving agents.
- `n` Integer.
- `world` `WorldMatrix` or `worldArray` object.
- `var` Character. The name of the selected agents variable. If agents are patches and the world is a `worldMatrix` object, `var` must not be provided. If agents are patches and the world is a `worldArray` object, `var` is the name of the layer to use to define the patches values. If agents are turtles, `var` is one of the turtles' variable and can be equal to `xcor`, `ycor`, any of the variables created when turtles were created, as well as any variable created using `turtlesOwn()`.

Details

`world` must not be provided if agents are turtles.

If there is a tie that would make the number of returned patches or turtles larger than `n`, it is broken randomly.
**Value**

Matrix (ncol = 2, nrow = n) with the first column pxcor and the second column pycor representing the coordinates of the n patches among the agents which have their variable values among the minimum values among the agents, or

AgentMatrix of length n representing the turtles among the agents which have their var values among the minimum values among the agents.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#min-n-of

**Examples**

# Patches

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
                   data = sample(1:10, size = 25, replace = TRUE))
plot(w1)
p1 <- minNof(agents = patches(w1), n = 6, world = w1)
```

# Turtles

```r
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
                    heading = sample(1:5, size = 10, replace = TRUE))
t2 <- minNof(agents = t1, n = 5, var = "heading")
```

---

**minOneOf**

*One agent with minimum*

**Description**

Report one patch or one turtle among agents which has its variable equals to the minimum value.
Usage

minOneOf(agents, world, var)

## S4 method for signature 'matrix,worldMatrix,missing'
minOneOf(agents, world)

## S4 method for signature 'matrix,worldArray,character'
minOneOf(agents, world, var)

## S4 method for signature 'agentMatrix,missing,character'
minOneOf(agents, var)

Arguments

agents       Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
world       WorldMatrix or worldArray object.
var         Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles' variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

Details

world must not be provided if agents are turtles.

If there are several patches or turtles among agents with their variable equal to the minimum value, one is chosen randomly. To access to all patches or turtles among agents which have their variable equal to the minimum value, use withMin().

Value

Matrix (ncol = 2, nrow = 1) with the first column pxcor and the second column pycor representing the coordinates of the patch (or of one of the patches) among the agents which has its variable equals to the minimum value among the agents, or

AgentMatrix of length 1 representing the turtle (or one of the turtles) among the agents which has its variable var equals to the minimum value among the agents.

Author(s)

Sarah Bauduin
References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#min-one-of

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
data = sample(1:5, size = 25, replace = TRUE))
plot(w1)
p1 <- minOneOf(agents = patches(w1), world = w1)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
heading = sample(1:3, size = 10, replace = TRUE))
t2 <- minOneOf(agents = t1, var = "heading")

---

minPxcor | Minimum pxcor
---------|-------------------

Description

Report the patches minimum pxcor in the world.

Usage

minPxcor(world)

## S4 method for signature 'worldNLR'
minPxcor(world)

Arguments

world | WorldMatrix or worldArray object.

Value

Integer.

Author(s)

Sarah Bauduin
References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#min-pcor

Examples
w1 <- createWorld()
minPxcor(w1)

---

minPycor
Minimum pycor

Description
Report the patches minimum pycor in the world.

Usage
minPycor(world)

## S4 method for signature 'worldNLR'
minPycor(world)

Arguments

world WorldMatrix or worldArray object.

Value
Integer.

Author(s)
Sarah Bauduin

References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#min-pcor
Examples

```r
w1 <- createWorld()
minPycor(w1)
```

---

**moveTo**

**Move to**

**Description**

Move the turtles to the agents' locations.

**Usage**

```r
moveTo(turtles, agents)
```

## S4 method for signature 'agentMatrix, matrix'
```r
moveTo(turtles, agents)
```

**Arguments**

- **turtles**  
  AgentMatrix object representing the moving agents.

- **agents** 
  Matrix (nrow = 2) with the first column `pxcor` and the second column `pycor` representing the patches coordinates, or 
  AgentMatrix object representing the moving agents.

**Details**

The number of agents must be equal to 1 or to length `turtles`.
The turtle's headings are not affected with this function.
If a turtle is moving to a patch location, it will be located at the patch center.

**Value**

AgentMatrix representing the turtles with updated coordinates and updated data for their previous coordinates `prevX` and `prevY`.

**Author(s)**

Sarah Bauduin

**References**

neighbors

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#move-to

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = runif(100))
t1 <- createTurtles(n = 5, coords = randomXYcor(w1, n = 5))
plot(w1)
points(t1, col = "black", pch = 16)

t1 <- moveTo(turtles = t1, agents = turtle(t1, who = 0))
points(t1, col = "red", pch = 16)

t1 <- moveTo(turtles = t1, agents = patch(w1, 9, 9))
points(t1, col = "blue", pch = 16)

neighbors

/\  Neighbors patches
\  

Description

Report the coordinates of the neighbors patches around the agents.

Usage

neighbors(world, agents, nNeighbors, torus = FALSE)

## S4 method for signature 'worldNLR,matrix,numeric'
neighbors(world, agents, nNeighbors, torus = FALSE)

Arguments

world  WorldMatrix or worldArray object.
agents  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
nNeighbors  Integer: 4 or 8. Represents the number of neighbor patches considered.
torus  Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

The patch around which the neighbors are identified, or the patch where the turtle is located on around which the neighbors are identified, is not returned.

If torus = FALSE, agents located on the edges of the world have less than nNeighbors patches around them. If torus = TRUE, all agents located on the edges of the world have nNeighbors patches around them, which some may be on the other sides of the world.
Value

Matrix (ncol = 3) with the first column pxcor and the second column pycor representing the coordinates of the neighbors patches around the agents and the third column id representing the id of the agents in the order provided.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#neighbors

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
neighbors(world = w1, agents = patch(w1, c(0,9), c(0,7)), nNeighbors = 8)
t1 <- createTurtles(n = 3, coords = randomXYcor(w1, n = 3))
neighbors(world = w1, agents = t1, nNeighbors = 4)

Description

Report TRUE if all agents have their variable equal to a given value, report FALSE otherwise.

Usage

NLall(agents, world, var, val)

## S4 method for signature 'matrix,worldMatrix,missing'
NLall(agents, world, val)

## S4 method for signature 'matrix,worldArray,character'
NLall(agents, world, var, val)

## S4 method for signature 'agentMatrix,missing,character'
NLall(agents, var, val)
Arguments

agents: Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

world: WorldMatrix or worldArray object.

var: Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles' variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

val: Numeric or character. Vector of any length.

Details

world must not be provided if agents are turtles.

Value

Logical. TRUE if all the agents have their variable equal to val, FALSE otherwise.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#all

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = runif(25))
NLall(agents = patches(w1), world = w1, val = 5)
w2 <- w1
w2 <- NLset(world = w1, agents = patches(w1), val = 5)
NLall(agents = patches(w2), world = w2, val = 5)

# Turtles
t1 <- createTurtles(n = 5, coords = cbind(xcor = 1, ycor = 1), heading = c(1, 2, 2, 1, 2))
NLall(agents = t1, var = "xcor", val = 1)
NLall(agents = t1, var = "heading", val = 2)
**Description**

Report TRUE if agents is non empty, report FALSE otherwise.

**Usage**

```r
NLany(agents)
```

```r
## S4 method for signature 'matrix'
NLany(agents)
```

**Arguments**

agents  
Matrix (`ncol = 2`) with the first column `pxcor` and the second column `pycor` representing the patches coordinates, or `AgentMatrix` object representing the moving agents.

**Value**

Logical. TRUE if there is at least one patch or one turtle in the agents, FALSE otherwise.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#any](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#any)

**Examples**

```r
# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
p1 <- noPatches()
p2 <- patch(w1, 0, 0)
NLany(p1)
NLany(p2)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
t2 <- noTurtles()
```
Description

Report the number of patches or turtles inside agents.

Usage

\texttt{NLcount(agents)}

## S4 method for signature 'matrix'
\texttt{NLcount(agents)}

Arguments

agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or \texttt{AgentMatrix} object representing the moving agents.

Value

Integer.

Author(s)

Sarah Bauduin

References


See Also

\url{https://ccl.northwestern.edu/netlogo/docs/dictionary.html#count}
Examples

```r
# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
p1 <- patches(w1)
NLcount(p1) # 25 patches

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
NLcount(t1) # 10 turtles
```

## NLdist

### Distances between agents

#### Description

Report the distances between agents and agents2.

#### Usage

```r
NLdist(agents, agents2, world, torus = FALSE, allPairs = FALSE)
```

#### Arguments

- **agents**: Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
- **agents2**: Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents, or Matrix (ncol = 2) with the first column x and the second column y representing locations coordinates.
- **world**: WorldMatrix or worldArray object.
- **torus**: Logical to determine if the world is wrapped. Default is torus = FALSE.
- **allPairs**: Logical. Only relevant if the number of agents/locations in agents and in agents2 are the same. If allPairs = FALSE, the distance between each agents with the corresponding agents2 is returned. If allPairs = TRUE, a full distance matrix is returned. Default is allPairs = FALSE.
Details

Distances from/to a patch are measured from/to its center.

If `torus = FALSE`, `world` does not need to be provided.

If `torus = TRUE`, a distance around the sides of the `world` is reported only if smaller than the one across the `world`.

Value

Numeric. Vector of distances between `agents` and `agents2` if `agents` and/or `agents2` contained one agent/location, or if `agents` and `agents2` contained the same number of agents/locations and `allPairs = FALSE`, or

Matrix of distances between `agents` (rows) and `agents2` (columns) if `agents` and `agents2` are of different lengths, or of same length and `allPairs = TRUE`.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#distance
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#distancexy

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
NLdist(agents = patch(w1, 0, 0), agents2 = patch(w1, c(1, 9), c(1, 9)))
NLdist(agents = patch(w1, 0, 0), agents2 = patch(w1, c(1, 9), c(1, 9)),
       world = w1, torus = TRUE)
t1 <- createTurtles(n = 2, coords = randomXYcor(w1, n = 2))
NLdist(agents = t1, agents2 = patch(w1, c(1,9), c(1,9)), allPairs = TRUE)
```
Set an agents variable

Description

Assign values to the agents for the selected variables.

Usage

NLset(world, turtles, agents, var, val)

## S4 method for signature 'missing,agentMatrix,agentMatrix,character'
NLset(turtles, agents, var, val)

## S4 method for signature 'worldMatrix,missing,matrix,missing'
NLset(world, agents, val)

## S4 method for signature 'worldArray,missing,matrix,character'
NLset(world, agents, var, val)

Arguments

- **world**
  WorldMatrix or worldArray object.

- **turtles**
  AgentMatrix object representing the moving agents.

- **agents**
  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

- **var**
  Character. Vector of the names of the selected agents variables. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layers to use to define the patches values. If agents are turtles, var is some of the turtles' variable and can be any of the variables created when turtles were created, as well as any variable created with turtlesOwn().

- **val**
  Numeric or character. Vector of length 1 or length NLcount(agents) if length(var) == 1, or Matrix or Dataframe (ncol = length(var), nrow = NLcount(agents)). Columns must be in the same order as var.

Details

If agents are patches, world must be provided and turtles must not be provided. If agents are turtles, turtles must be provided and world must not be provided.
Value

WorldMatrix or worldArray object with the values val assigned to the patches variables var for the agents, or

AgentMatrix representing the turtles with the values val assigned to the variables var for the agents.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#set

Examples

```
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
w1 <- NLset(world = w1, agents = patches(w1), val = 1)
    # Set the patch[0,4] to 0
w1 <- NLset(world = w1, agents = patch(w1, 0, 4), val = 0)
of(world = w1, agents = patches(w1))

# Set the heading of turtle 0 to 180

```

```
t1 <- createTurtles(n = 3, world = w1, heading = 0)
t2 <- NLset(turtles = t1, agents = turtle(t1, who = 0), var = "heading", val = 180)
of(agents = t2, var = "heading") # c(180, 0, 0)
```

Description

Report the patches or the turtles among agents which have their variable equals to specific values.
Usage

NLwith(agents, world, var, val)

## S4 method for signature 'matrix,worldMatrix,missing'
NLwith(agents, world, val)

## S4 method for signature 'matrix,worldArray,character'
NLwith(agents, world, var, val)

## S4 method for signature 'agentMatrix,missing,character'
NLwith(agents, var, val)

Arguments

agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

world WorldMatrix or worldArray object.

var Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles’ variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

val Numeric or character. Vector of any length.

Details

world must not be provided if agents are turtles.

This is equivalent in R to subsetting.

Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches among the agents which have their variable equals to any val, or AgentMatrix representing the turtles among the agents which have their variable var equals to any val.

Author(s)

Sarah Bauduin

References

**Examples**

```r
# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
                   data = sample(1:5, size = 25, replace = TRUE))
plot(w1)
p2 <- NLwith(agents = patches(w1), world = w1, val = 2)

# Turtles
t1 <- createTurtles(n = 5, coords = randomXYcor(w1, n = 5),
                    breed = c("sheep", "sheep", "wolf", "sheep", "sheperd"))
t2 <- NLwith(agents = t1, var = "breed", val = "sheep")
t3 <- NLwith(agents = t1, var = "breed", val = c("sheep", "wolf"))
```

---

**Description**

Convert vector indices or Raster* cell numbers into `worldMatrix` indices.

**Usage**

```r
NLworldIndex(world, cellNum)
```

### S4 method for signature 'worldMatrix,numeric'

```r
NLworldIndex(world, cellNum)
```

**Arguments**

- `world`  
  WorldMatrix or worldArray object.

- `cellNum`  
  Integer. Vector of cells number.

**Value**

Numeric. Vector of `worldMatrix` indices.

**Author(s)**

Eliot McIntire

---

**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#with](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#with)
Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = 1:100)
w1Ras <- world2raster(w1)
index <- 24
pxpy <- PxcorPycorFromCell(world = w1, cellNum = index)

rasValue <- as.integer(unname(w1Ras[index]))
# Not correct index:
identical(w1[index], rasValue)

# Correct index
identical(w1[NLworldIndex(w1, index)], rasValue)
```

---

### `nOf`

**N random agents**

**Description**

Report \( n \) patches or turtles randomly selected among agents.

**Usage**

```r
nOf(agents, n)
```

#### S4 method for signature 'matrix,numeric'

`nOf(agents, n)`

**Arguments**

- **agents**
  - Matrix (\( ncol = 2 \)) with the first column `pxcor` and the second column `pycor` representing the patches coordinates, or
  - Matrix (\( ncol = 3 \)) with the first column "pxcor" and the second column "pycor" representing the patches coordinates and the third column id, or
  - `AgentMatrix` object representing the moving agents, or
  - Matrix (\( ncol = 2 \)) with the first column `whoTurtles` and the second column id.

- **n**
  - Integer. Number of patches or turtles to select from agents.

**Details**

\( n \) must be less or equal the number of patches or turtles in `agents`.

If `agents` is a matrix with \( ncol = 3 \), the selection of \( n \) random patches is done per individual "id". The order of the patches coordinates returned follow the order of "id". If `agents` is a matrix (\( ncol = 2 \)) with columns `whoTurtles` and id, the selection of \( n \) random turtles (defined by their `whoTurtles`) is done per individual "id". The order of the who numbers returned follow the order of "id".
Value

Matrix (ncol = 2, nrow = n) with the first column pxcor and the second column pycor representing the coordinates of the selected patches from agents, or
Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the selected patches from agents, n per individual "id", or
AgentMatrix (nrow = n) representing the turtles selected from agents,
Integer. Vector of who numbers for the selected turtles from agents, n per individual "id".

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#n-of

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
pSelect <- nOf(agents = patches(w1), n = 5)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
tSelect <- nOf(agents = t1, n = 2)

noPatches

<table>
<thead>
<tr>
<th>noPatches</th>
<th>No patches</th>
</tr>
</thead>
</table>

Description

Report an empty patch agentset.

Usage
	noPatches()

Value

Matrix (ncol = 2, nrow = 0) with the first column pxcor and the second column pycor.
Author(s)
Sarah Bauduin

References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#no-patches

Examples
```r
p1 <- noPatches()
NLcount(p1)
```

---

noTurtles | No turtles
----------|-----------

Description
Report an empty turtle agentset.

Usage
```r
noTurtles()
```

Value
AgentMatrix with the turtle variables defined as when using createTurtles() but with 0 turtle.

Author(s)
Sarah Bauduin

References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#no-turtles
Examples

t1 <- noTurtles()
NLcount(t1)

Description

These are required to create plotting methods to work with quickPlot.

Usage

```r
## S4 method for signature 'worldArray'
numLayers(x)

## S4 method for signature 'agentMatrix'
.plotGrob(
grobToPlot,
col = NULL,
real = FALSE,
size = unit(5, "points"),
minv,
maxv,
legend = TRUE,
legendText = NULL,
length = NULL,
gp = gpar(),
gpText = gpar(),
pch = 19,
speedup = 1,
name = character(),
vp = list(),
...)

## S4 method for signature 'worldArray'
layerNames(object)

## S4 method for signature 'worldArray,.quickPlotGrob'
.identifyGrobToPlot(toPlot, sGrob, takeFromPlotObj)
```
Arguments

- **x**: A `.quickPlotObjects` object or list of these.
- **grobToPlot**: A `Raster*`, `SpatialLines*`, `SpatialPoints*`, or `SpatialPolygons*` object.
- **col**: Currently only used for the legend of a `Raster*` object.
- **real**: Logical indicating whether the data are real numbers (i.e., as opposed to integer or factor).
- **size**: The size of the `SpatialPoints`.
- **minv**: The minimum value on a `Raster*`. Required because not all Rasters have this defined internally.
- **maxv**: The maximum value on a `Raster*`. Required because not all Rasters have this defined internally.
- **legend**: Logical indicating whether a legend should be drawn. Default `TRUE`.
- **legendText**: Vector of values to use for legend value labels. Defaults to `NULL` which results in a pretty numeric representation. If `Raster*` has a Raster Attribute Table (rat; see `raster` package), this will be used by default. Currently, only a single vector is accepted.
- **length**: Numeric.
- **gp**: grid parameters, usually the output of a call to `gpar`.
- **gpText**: `gpar` object for legend label text.
- **pch**: Point character for `SpatialPoints`, as `par`.
- **speedup**: Numeric. The factor by which the number of vertices in `SpatialPolygons` and `SpatialLines*` will be subsampled. The vertices are already subsampled by default to make plotting faster.
- **name**: Character string of name of object being plotted.
- **vp**: whole viewport tree of `quickPlotGrob`
- **...**: Additional arguments. None currently implemented.
- **object**: A `Raster*`, `SpatialPoints*`, `SpatialLines*`, or `SpatialPolygons*` object; or list of these.
- **toPlot**: The object to plot. Should be a single layer if from a multi-layer object such as a RasterStack.
- **sGrob**: `quickPlot` grob object
- **takeFromPlotObj**: Logical. Should the data come from the argument passed into `Plot` (TRUE), or from the (.`quickPlotEnv`) (FALSE).
Values of an agents variable

Description

Report the agents values for the requested variable.

Usage

of(world, agents, var)

## S4 method for signature 'missing,agentMatrix,character'
of(agents, var)

## S4 method for signature 'worldMatrix,matrix,missing'
of(world, agents)

## S4 method for signature 'worldArray,matrix,character'
of(world, agents, var)

Arguments

world WorldMatrix or worldArray object.
agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
var Character. Vector of the names of the selected agents variables. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layers to use to define the patches values. If agents are turtles, var is some of the turtles' variable and can be any of the variables created when turtles were created, as well as any variable created with turtlesOwn().

Details

world must be provided only if agents are patches.

Value

Vector of values for the agents if one variable is requested. The class depends of the variable class. The order of the vector follows the order of the agents, or Matrix or Dataframe (ncol = length(var), nrow = NLcount(agents)) if more than one variable is requested. The row order follows the order of the agents.

Author(s)

Sarah Bauduin
References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#of

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
data = 1:25)
of(world = w1, agents = patch(w1, c(0, 0), c(4, 0)))

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
of(agents = t1, var = "heading")

---

oneOf

**One random agent**

Description

Report one patch or turtle randomly selected among agents.

Usage

oneOf(agents)

## S4 method for signature 'matrix'

oneOf(agents)

Arguments

agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or Matrix (ncol = 3) with the first column "pxcor and the second column "pycor representing the patches coordinates and the third column id, or AgentMatrix object representing the moving agents, or Matrix (ncol = 2) with the first column whoTurtles and the second column id.
Details

If `agents` is a matrix with `ncol = 3`, the selection of one random patch is done per individual `id`. The order of the patches coordinates returned follow the order of `id`. If `agents` is a matrix (`ncol = 2`) with columns `whoTurtles` and `id`, the selection of one random turtle (defined by their `whoTurtles`) is done per individual `id`. The order of the `who` numbers returned follow the order of `id`.

Value

Matrix (`ncol = 2, nrow = 1`) with the first column `pxcor` and the second column `pycor` representing the coordinates of the selected patch from `agents`, or

Matrix (`ncol = 2`) with the first column `pxcor` and the second column `pycor` representing the coordinates of the selected patches from `agents`, one per individual `id`, or

AgentMatrix object representing the turtle selected from `agents`, or

Integer. Vector of `who` numbers for the selected turtles from `agents`, one per individual `id`.

Author(s)

Sarah Bauduin

References


See Also

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#one-of](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#one-of)

Examples

```r
# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
pSelect <- oneOf(agents = patches(w1))

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
tSelect <- oneOf(agents = t1)
```
Description

Report an agentset of the agents except specific ones.

Usage

other(agents, except)

## S4 method for signature 'matrix, matrix'
other(agents, except)

Arguments

agents  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

except  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

Details

Both agents and except must be of the same class (e.g., both patches or both turtles).

Warning: this function removes turtles only based on similar who numbers and breed names.

Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches in agents without the ones in except, or AgentMatrix representing the turtles in agents without the ones in except.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#other
Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
p1 <- other(agents = patches(w1), except = patch(w1, 0, 0))
NLcount(p1) # 99 patches

# Turtles
t1 <- createTurtles(n = 10, coords = cbind(xcor = 0, ycor = 0))
t2 <- other(agents = t1, except = turtle(t1, who = 0))
NLcount(t2) # 9 turtles

---

patch | Patches coordinates

Description

Report the coordinates of the patches at the given [x,y] locations.

Usage

patch(world, x, y, duplicate = FALSE, torus = FALSE, out = FALSE)

## S4 method for signature 'worldNLR,numeric,numeric'
patch(world, x, y, duplicate = FALSE, torus = FALSE, out = FALSE)

Arguments

world  
       WorldMatrix or worldArray object.
x       Numeric. Vector of x coordinates. Must be of same length as y.
y       Numeric. Vector of y coordinates. Must be of same length as x.
duplicate  Logical. If more than one location [x,y] fall into the same patch and duplicate == TRUE, the patch coordinates are returned the number of times the locations. If duplicate == FALSE, the patch coordinates are only returned once. Default is duplicate == FALSE.
torus       Logical to determine if the world is wrapped. Default is torus = FALSE.
out         Logical. If out = FALSE, no patch coordinates are returned for patches outside of the world’s extent, if out = TRUE, NA are returned. Default is out = FALSE.

Details

If a location [x,y] is outside the world’s extent and torus = FALSE and out = FALSE, no patch coordinates are returned; if torus = FALSE and out = TRUE, NA are returned; if torus = TRUE, the patch coordinates from a wrapped world are returned.
Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates at \([x, y]\).

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
patch(world = w1, x = c(0, 9.1, 8.9, 5, 5.3), y = c(0, 0, -0.1, 12.4, 12.4))
patch(world = w1, x = c(0, 9.1, 8.9, 5, 5.3), y = c(0, 0, -0.1, 12.4, 12.4),
duplicate = TRUE)
patch(world = w1, x = c(0, 9.1, 8.9, 5, 5.3), y = c(0, 0, -0.1, 12.4, 12.4),
torus = TRUE)
patch(world = w1, x = c(0, 9.1, 8.9, 5, 5.3), y = c(0, 0, -0.1, 12.4, 12.4),
torus = TRUE, duplicate = TRUE)

patchAhead Patches ahead

Description

Report the coordinates of the patches at the given distances of the turtles in the direction of their headings.

Usage

patchAhead(world, turtles, dist, torus = FALSE)

## S4 method for signature 'worldNLR,agentMatrix,numeric'
patchAhead(world, turtles, dist, torus = FALSE)
Arguments

world  WorldMatrix or worldArray object.
turtles  AgentMatrix object representing the moving agents.
dist  Numeric. Vector of distances from the turtles. dist must be of length 1 or of length turtles.
torus  Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

If torus = FALSE and the patch at distance dist of a turtle is outside the world’s extent, NA are returned for the patch coordinates. If torus = TRUE, the patch coordinates from a wrapped world are returned.

Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches at the distances dist and turtles’s headings directions of turtles. The order of the patches follows the order of the turtles.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-ahead

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
patchAhead(world = w1, turtles = t1, dist = 1)
Description

Report the coordinates of the patches at \((dx, dy)\) distances of the agents.

Usage

```
patchAt(world, agents, dx, dy, torus = FALSE)
```

## S4 method for signature 'worldNLR, matrix, numeric, numeric'

```
patchAt(world, agents, dx, dy, torus = FALSE)
```

Arguments

- `world`: WorldMatrix or worldArray object.
- `agents`: Matrix \((n \times 2)\) with the first column pxc or and the second column pyc representing the patches coordinates, or AgentMatrix object representing the moving agents.
- `dx`: Numeric. Vector of distances to the east (right) from the agents. If \(dx\) is negative, the distance to the west (left) is computed. \(dx\) must be of length 1 or of the same length as number of patches or turtles in agents.
- `dy`: Numeric. Vector of distances to the north (up) from the agents. If \(dy\) is negative, the distance to the south is computed (down). \(dy\) must be of length 1 or of the same length as number of patches or turtles in agents.
- `torus`: Logical to determine if the world is wrapped. Default is \(torus = FALSE\).

Details

If the patch at distance \((dx, dy)\) of an agent is outside of the world’s extent and \(torus = FALSE\), \(NA\) are returned for the patch coordinates; if \(torus = TRUE\), the patch coordinates from a wrapped world are returned.

Value

Matrix \((n \times 2)\) with the first column pxc or and the second column pyc representing the coordinates of the patches at \((dx, dy)\) distances of the agents. The order of the patches follows the order of the agents.

Author(s)

Sarah Bauduin
patchDistDir

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-at
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#at-points

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
patchCorner <- patchAt(world = w1, agents = patch(w1, 0, 0), dx = 1, dy = 1)
t1 <- createTurtles(n = 1, coords = cbind(xcor = 0, ycor = 0))
patchCorner <- patchAt(world = w1, agents = t1, dx = 1, dy = 1)

patchDistDir  Patches at given distances and directions

Description

Report the coordinates of the patches at the given distances and directions from the agents.

Usage

patchDistDir(world, agents, dist, angle, torus = FALSE)

## S4 method for signature 'worldNLR, matrix, numeric, numeric'
patchDistDir(world, agents, dist, angle, torus = FALSE)

Arguments

world  WorldMatrix or worldArray object.
agents  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.
dist  Numeric. Vector of distances from the agents. Must be of length 1 or of the same length as the number of agents.
angle  Numeric. Absolute directions from the agents. angle must be of length 1 or of the same length as the number of agents. Angles are in degrees with 0 being North.
torus  Logical to determine if the world is wrapped. Default is torus = FALSE.
Details

If \texttt{torus = FALSE} and the patch at distance \texttt{dist} and direction \texttt{angle} of an agent is outside the world's extent, \texttt{NA} are returned for the patch coordinates. If \texttt{torus = TRUE}, the patch coordinates from a wrapped world are returned.

If agents are turtles, their headings are not taken into account; the given directions \texttt{angle} are used. To find a patch at certain distance from a turtle using the turtle's heading, look at \texttt{patchAhead()}, \texttt{patchLeft()} or \texttt{patchRight()}.

Value

Matrix (\texttt{ncol = 2}) with the first column \texttt{pxcor} and the second column \texttt{pycor} representing the coordinates of the patches at the distances \texttt{dist} and directions \texttt{angle} of agents. The order of the patches follows the order of the agents.

Author(s)

Sarah Bauduin

References


See Also

\url{https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-at-heading-and-distance}

Examples

\begin{verbatim}
  w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
  p1 <- patchDistDir(world = w1, agents = patch(w1, 0, 0), dist = 1, angle = 45)
  t1 <- createTurtles(n = 1, coords = cbind(xcor = 0, ycor = 0), heading = 315)
  p2 <- patchDistDir(world = w1, agents = t1, dist = 1, angle = 45)

  patches(w1)  # All the patches in a world
\end{verbatim}

Description

Report the coordinates of all the patches in the \texttt{world}.

Usage

\begin{verbatim}
  patches(world)

  ## S4 method for signature 'worldNLR'
  patches(world)
\end{verbatim}
Arguments

world WorldMatrix or worldArray object.

Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates. The order of the patches follows the order of the cells numbers as defined for a Raster* object.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patches

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
allPatches <- patches(world = w1)
NLcount(allPatches) # 100 patches
Details

If a turtle is located outside of the world's extent, NA are returned for the patch coordinates.

Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches at the turtles location. The order of the patches follows the order of the turtles.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-here

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
patchHere(world = w1, turtles = t1)

<table>
<thead>
<tr>
<th>patchLeft</th>
<th>Patches on the left</th>
</tr>
</thead>
</table>

Description

Report the coordinates of the patches at the given distances of the turtles and given angle left of their headings.

Usage

patchLeft(world, turtles, dist, angle, torus = FALSE)

## S4 method for signature 'worldNLR,agentMatrix,numeric,numeric'
patchLeft(world, turtles, dist, angle, torus = FALSE)
**Arguments**

- **world**: WorldMatrix or worldArray object.
- **turtles**: AgentMatrix object representing the moving agents.
- **dist**: Numeric. Vector of distances from the turtles. dist must be of length 1 or of length turtles.
- **angle**: Numeric. Vector of angles in degrees by which the turtle's headings should rotate to locate the patches. Must be of length 1 or of length turtles.
- **torus**: Logical to determine if the world is wrapped. Default is torus = FALSE.

**Details**

If a given dist value is negative, then the turtle would look backward. If a given angle value is negative, then the turtle would look to the right.

If torus = FALSE and the patch at distance dist of a turtle and angle degrees to the left of its heading is outside the world's extent, NA are returned for the patch coordinates. If torus = TRUE, the patch coordinates from a wrapped world are returned.

**Value**

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches at dist distances of the turtles and angle to the left of their headings. The order of the patches follows the order of the turtles.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-lr-and-ahead

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 1, coords = cbind(xcor = 2, ycor = 2), heading = 90)
patchLeft(world = w1, turtles = t1, dist = 2, angle = 90)
```
patchRight

Patches on the right

Description

Report the coordinates of the patches at the given distances of the turtles and given angle right of their headings.

Usage

patchRight(world, turtles, dist, angle, torus = FALSE)

## S4 method for signature 'worldNLR,agentMatrix,numERIC,numERIC'
patchRight(world, turtles, dist, angle, torus = FALSE)

Arguments

world
WorldMatrix or worldArray object.

turtles
AgentMatrix object representing the moving agents.

dist
Numeric. Vector of distances from the turtles. dist must be of length 1 or of length turtles.

angle
Numeric. Vector of angles in degrees by which the turtle's headings should rotate to locate the patches. Must be of length 1 or of length turtles.

torus
Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

If a given dist value is negative, then the turtle would look backward. If a given angle value is negative, then the turtle would look to the left.

If torus = FALSE and the patch at distance dist of a turtle and angle degrees to the right of its heading is outside the world's extent, NA are returned for the patch coordinates. If torus = TRUE, the patch coordinates from a wrapped world are returned.

Value

Matrix (nrow = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches at dist distances of the turtles and angle to the right of their headings. The order of the patches follows the order of the turtles.

Author(s)

Sarah Bauduin

References

**patchSet**

**Description**

Report the patch coordinates of all the unique patches contained in the inputs.

**Usage**

```r
patchSet(…)
```

```r
## S4 method for signature 'matrix'
patchSet(…)
```

**Arguments**

```r
...
```

Matrices (ncol = 2) of patches coordinates with the first column pxcor and the second column pycor.

**Details**

Duplicate patches among the inputs are removed in the returned matrix.

**Value**

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates.

**Author(s)**

Sarah Bauduin

**References**

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-set

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
p1 <- patchAt(world = w1, agents = patch(w1, c(0,1,2), c(0,0,0)), dx = 1, dy = 1)
p2 <- patchDistDir(world = w1, agents = patch(w1, 0, 0), dist = 1, angle = 45)
p3 <- patch(world = w1, x = 4.3, y = 8)
p4 <- patchSet(p1, p2, p3)

<table>
<thead>
<tr>
<th>pExist</th>
<th>Do the patches exist?</th>
</tr>
</thead>
</table>

Description

Report TRUE if a patch exists inside the world’s extent, report FALSE otherwise.

Usage

pExist(world, pxcor, pycor)

## S4 method for signature 'worldNLR,numeric,numeric'
pExist(world, pxcor, pycor)

Arguments

world WorldMatrix or worldArray object.
pxcor Integer. Vector of patches pxcor coordinates. Must be of length 1 or of the same length as pycor.
pycor Integer. Vector of patches pycor coordinates. Must be of length 1 or of the same length as pxcor.

Value

Logical.

Author(s)

Sarah Bauduin

References

plot.agentMatrix

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#member

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
pExist(world = w1, pxcor = -1, pycor = 2)

plot.agentMatrix  Basic plot methods for agentMatrix, worldMatrix, worldArray

Description

These pass to plot, as a matrix of points (agentMatrix), as a raster (worldMatrix), or a rasterStack (worldArray). They can be modified.

Usage

## S3 method for class 'agentMatrix'
plot(x, ...)

## S3 method for class 'worldMatrix'
plot(x, ...)

## S3 method for class 'worldArray'
plot(x, ...)

## S3 method for class 'agentMatrix'
points(x, ...)

Arguments

x  an agentMatrix, worldMatrix or worldArray object

...  arguments passed to plot methods for matrix (agentMatrix) or raster (world*)

Examples

# agentMatrix
newAgent <- new("agentMatrix",
   coords = cbind(pxcor = c(1, 2, 5), pycor = c(3, 4, 6)),
   char = letters[c(1, 2, 6)],
   nums2 = c(4.5, 2.6, 2343),
   char2 = LETTERS[c(4, 24, 3)],
   nums = 5:7)
plot(newAgent)
## worldMatrix

```
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = 1:100)
plot(w1)
```

## worldArray

```
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = 1:25)
w2 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = 25:1)
w3 <- stackWorlds(w1, w2)
plot(w3)
```

# agentMatrix

```
newAgent <- new("agentMatrix",
    coords = cbind(pxcor = c(1, 2, 5), pycor = c(3, 4, 6)),
    char = letters[c(1, 2, 6)],
    nums2 = c(4.5, 2.6, 343),
    char2 = LETTERS[c(4, 24, 3)],
    nums = 5:7)
points(newAgent)
```

---

### Description

Report the patches coordinates pxcor and pycor given the cells numbers as defined for a Raster* object.

### Usage

```
PxcorPycorFromCell(world, cellNum)
```

## S4 method for signature 'worldNLR,numeric'

```
PxcorPycorFromCell(world, cellNum)
```

### Arguments

- `world` : WorldMatrix or worldArray object.
- `cellNum` : Integer. Vector of cells number.

### Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor in the order of the given cellNum.

### Author(s)

Sarah Bauduin
**randomPxcor**

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
cellNum <- cellFromPxcorPycor(world = w1, pxcor = 0, pycor = 9)
PxcorPycorFromCell(world = w1, cellNum = cellNum)
cellNum <- cellFromPxcorPycor(world = w1, pxcor = c(0, 1, 2), pycor = 0)
PxcorPycorFromCell(world = w1, cellNum = cellNum)
```

---

**randomPxcor**

Random pxcor

**Description**

Report n random pxcor coordinates within the world’s extent.

**Usage**

```r
randomPxcor(world, n)
```

```r
## S4 method for signature 'worldNLR,numeric'
randomPxcor(world, n)
```

**Arguments**

- `world`: WorldMatrix or worldArray object.
- `n`: Integer.

**Value**

Integer. Vector of length n of pxcor coordinates.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-pcor](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-pcor)

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
pxcor <- randomPxcor(world = w1, n = 10)
```
randomPycor

**randomPycor**  
*Random pycor*

---

**Description**

Report *n* random pycor coordinates within the world's extent.

**Usage**

randomPycor(world, n)

```r
## S4 method for signature 'worldNLR,numeric'
randomPycor(world, n)
```

**Arguments**

- **world**  
  WorldMatrix or worldArray object.

- **n**  
  Integer.

**Value**

Integer. Vector of length *n* of pycor coordinates.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-pcor](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-pcor)

**Examples**

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
pycor <- randomPycor(world = w1, n = 10)
```
randomXcor

randomXcor \hspace{1cm} \textit{Random xcor}

**Description**

Report \( n \) random xcor coordinates within the world’s extent.

**Usage**

\[
\text{randomXcor}(\text{world}, n)
\]

\[
\text{## S4 method for signature ’worldNLR,numeric’}
\text{randomXcor}(\text{world}, n)
\]

**Arguments**

- \textit{world} \hspace{1.5cm} \textit{WorldMatrix or worldArray object.}
- \textit{n} \hspace{1.5cm} \text{Integer.}

**Value**

\text{Numeric. Vector of length \( n \) of xcor coordinates.}

**Author(s)**

Sarah Bauduin

**References**


**See Also**

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-cor

**Examples**

\[
\text{w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = runif(25))}
\]

\[
\text{t1 <- createTurtles(n = 10, coords = cbind(xcor = randomXcor(world = w1, n = 10), ycor = randomYcor(world = w1, n = 10)))}
\]

\[
\text{plot(w1)}
\]

\[
\text{points(t1, col = of(agents = t1, var = ”color”), pch = 16)}
\]
randomXYcor  
**Random turtles coordinates**

### Description
Report n random xcor and ycor coordinates within the world’s extent.

### Usage

```r
randomXYcor(world, n)
```

```
## S4 method for signature 'worldNLR,numeric'
randomXYcor(world, n)
```

### Arguments
- `world` WorldMatrix or worldArray object.
- `n` Integer.

### Value
Matrix (`ncol = 2, nrow = n`) with the first column xcor and the second column ycor.

### Author(s)
Sarah Bauduin

### Examples
```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, 
data = runif(25))
t1 <- createTurtles(n = 10, coords = randomXYcor(world = w1, n = 10))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

randomYcor  
**Random ycor**

### Description
Report n random ycor coordinates within the world’s extent.
Usage
randomYcor(world, n)

## S4 method for signature 'worldNLR,numeric'
randomYcor(world, n)

Arguments
world WorldMatrix or worldArray object.
n Integer.

Value
Numeric. Vector of length n of ycor coordinates.

Author(s)
Sarah Bauduin

References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-cor

Examples
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
data = runif(25))
t1 <- createTurtles(n = 10, coords = cbind(xcor = randomXcor(world = w1, n = 10),
ymcor = randomYcor(world = w1, n = 10)))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

raster2world
Convert a Raster* object into a worldMatrix or worldArray object

Description
Convert a RasterLayer object into a worldMatrix object or a RasterStack object into a worldArray object.
Usage

```r
raster2world(raster, method)
```

## S4 method for signature 'RasterLayer,character'

```r
raster2world(raster, method)
```

## S4 method for signature 'RasterStack,character'

```r
raster2world(raster, method)
```

Arguments

- `raster` RasterLayer or RasterStack object.
- `method` "ngb" or "bilinear" for the resample method.

Details

See `help("worldMatrix-class")` or `help("worldArray-class")` for more details on the classes.

The raster is resampled to match the coordinates system and resolution of a worldMatrix or worldArray using the chosen method. The extent will be bigger by 1 on the width and on the height.

Value

WorldMatrix or worldArray object depending on the input raster. Patches value are retained from the raster.

Author(s)

Sarah Bauduin

Examples

```r
r1 <- raster(extent(c(0, 10, 0, 10)), nrows = 10, ncols = 10)
r1[]<-runif(100)
w1 <- raster2world(r1, method = "ngb")
plot(r1)
plot(w1)
```

right

Rotate to the right

Description

Rotate the turtles’s headings to the right of angle degrees.
Usage

right(turtles, angle)

## S4 method for signature 'agentMatrix,numeric'
right(turtles, angle)

Arguments

turtles AgentMatrix object representing the moving agents.
angle Numeric. Vector of angles in degrees by which to rotate the turtles' headings. Must be of length 1 or of length turtles.

Details

If a given angle value is negative, then the turtle rotates to the left.

Value

AgentMatrix representing the turtles with updated heading values.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#right

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
t1 <- createTurtles(n = 10, world = w1)
of(agents = t1, var = "heading")
t1 <- right(turtles = t1, angle = 180)
of(agents = t1, var = "heading")
setXY  

Set turtles' locations

Description
Set the turtles xcor and ycor coordinates.

Usage
setXY(turtles, xcor, ycor, world, torus = FALSE)

## S4 method for signature 'agentMatrix,numeric,numeric,missing,ANY'
setXY(turtles, xcor, ycor, torus)

## S4 method for signature 'agentMatrix,numeric,numeric,worldNLR,logical'
setXY(turtles, xcor, ycor, world, torus = FALSE)

Arguments
- turtles: AgentMatrix object representing the moving agents.
- xcor: Numeric. Vector of x coordinates. Must be of length 1 or of length turtles.
- ycor: Numeric. Vector of y coordinates. Must be of length 1 or of length turtles.
- world: WorldMatrix or worldArray object.
- torus: Logical to determine if the world is wrapped. Default is torus = FALSE.

Details
world must be provided only if torus = TRUE.
If the given coordinates [xcor, ycor] are located outside of the world's extent and torus = TRUE, then the coordinates assigned to the turtle are the ones from a wrapped world; if torus = FALSE, the turtle is located outside of the world's extent with the given coordinates.

Value
AgentMatrix representing the turtles with updated coordinates and updated data for their previous coordinates prevX and prevY.

Author(s)
Sarah Bauduin

References
show.agentMatrix-method

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#setxy

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9,
data = runif(100))
t1 <- createTurtles(n = 5, coords = randomXYcor(w1, n = 5))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

w1 <- setXY(turtles = t1, xcor = 1:5, ycor = 1:5)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

show,agentMatrix-method

Key base R functions for agentMatrix class

Description

Slight modifications from the default versions.

Usage

## S4 method for signature 'agentMatrix'
show(object)

## S4 method for signature 'agentMatrix'
length(x)

## S4 method for signature 'agentMatrix'
nrow(x)

## S3 method for class 'agentMatrix'
head(x, n = 6L, ...)

## S3 method for class 'agentMatrix'
tail(x, n = 6L, ...)

Arguments

object An agentMatrix object.
x An agentMatrix object.
n documentation needed
... documentation needed
show,worldArray-method

Key base R functions for worldNLR classes

Description

Slight modifications from the default versions.

Usage

## S4 method for signature 'worldArray'
show(object)

## S4 method for signature 'worldMatrix'
show(object)

Arguments

object An agentMatrix object.

sortOn

Sort agents

Description

Return the agents sorted according to their value.

Usage

sortOn(agents, world, var)

## S4 method for signature 'matrix,worldMatrix,missing'
sortOn(agents, world)

## S4 method for signature 'matrix,worldArray,character'
sortOn(agents, world, var)

## S4 method for signature 'agentMatrix,missing,character'
sortOn(agents, var)
**Arguments**

- **agents**
  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or `AgentMatrix` object representing the moving agents.

- **world**
  `WorldMatrix` or `worldArray` object.

- **var**
  Character. The name of the selected agents variable. If `agents` are patches and the `world` is a `worldMatrix` object, `var` must not be provided. If `agents` are patches and the `world` is a `worldArray` object, `var` is the name of the layer to use to define the patches values. If `agents` are turtles, `var` is one of the turtles’ variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using `turtlesOwn()`.

**Details**

- `world` must not be provided if `agents` are turtles.
- The sorting of the agents is done in an increasing order.

**Value**

- Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches sorted according to their values, if `agents` are patches, or `AgentMatrix` representing the turtles sorted according to their `var` values, if `agents` are turtles.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

[https://ccl.northwestern.edu/netlogo/docs/dictionary.html#sort-on](https://ccl.northwestern.edu/netlogo/docs/dictionary.html#sort-on)

**Examples**

```r
# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
                   data = sample(1:5, size = 25, replace = TRUE))
plot(w1)
p1 <- sortOn(agents = patches(w1), world = w1)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
sortHeadingT1 <- sortOn(agents = t1, var = "heading")
```
spdf2turtles

From SpatialPointsDataFrame to agentMatrix

Description

Convert a SpatialPointsDataFrame object into an agentMatrix object.

Usage

spdf2turtles(spdf)

## S4 method for signature 'SpatialPointsDataFrame'
spdf2turtles(spdf)

Arguments

spdf SpatialPointsDataFrame object representing moving agents.

Details

If the spdf does not contain the variables created with createTurtles(), these variables will be created with the default values as in createTurtles().

Value

AgentMatrix object representing the moving agents (coordinates and data) as contained in spdf.

Author(s)

Sarah Bauduin

Examples

sp1 <- SpatialPointsDataFrame(coords = cbind(x = c(1, 2, 3), y = c(1, 2, 3)),
    data = cbind.data.frame(age = c(0, 0, 3),
    sex = c("F", "F", "M"))

t1 <- spdf2turtles(spdf = sp1)
**sprout**

_Sprout new turtles_

---

**Description**

Create \( n \) new turtles on specific patches.

**Usage**

```r
sprout(n, patches, breed, heading, color, turtles)

## S4 method for signature 'numeric,matrix'
sprout(n, patches, breed, heading, color, turtles)
```

**Arguments**

- \( n \) : Integer. Vector of length 1 or of length the number of patches. Number of new turtles to create on each patch.
- \( \text{patches} \) : Matrix (\( ncol = 2 \)) with the first column \( \text{pxcor} \) and the second column \( \text{pycor} \) representing the patches coordinates.
- \( \text{breed} \) : Character. Vector of breed names. Must be of length 1 or of length the number of patches. If missing, \( \text{breed} = \text{turtle} \) for all the sprouted turtles.
- \( \text{heading} \) : Numeric. Vector of values between 0 and 360. Must be of length 1 or of length the number of patches. If missing, a random heading is assigned to each sprouted turtle.
- \( \text{color} \) : Character. Vector of color names. Must be of length 1, of length the number of patches or of length \( \text{sum(n)} \). If missing, colors are assigned using the function `rainbow(n)`.
- \( \text{turtles} \) : AgentMatrix object representing the moving agents.

**Details**

\( \text{nrow(patches)} \) must be equal to 1 or to \( n \).

If \( \text{turtles} \) is provided, the new turtles are added to the \( \text{turtles} \) when returned. The who numbers of the sprouted turtles therefore follow the ones from the \( \text{turtles} \). All new sprouted turtles are placed at the end of the \( \text{AgentMatrix} \) object. If no \( \text{turtles} \) is provided, a new \( \text{AgentMatrix} \) is created and the who numbers start at 0.

If \( \text{turtles} \) is provided and had additional variables created with `turtlesOwn()`, \( \text{NA} \) is given for these variables for the new sprouted turtles.

**Value**

AgentMatrix including the new sprouted turtles.
**Author(s)**
Sarah Bauduin

**References**

**See Also**
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#sprout

**Examples**
```r
t1 <- sprout(patches = cbind(pxcor = 2, pycor = 2), n = 3)
t2 <- sprout(patches = cbind(pxcor = 3, pycor = 3), n = 3, turtles = t1)
```

---

**Description**
Stack multiple `worldMatrix` into a `worldArray`.

**Usage**
```r
stackWorlds(...)
```

## S4 method for signature 'worldMatrix'
```r
stackWorlds(...)
```

**Arguments**

... `worldMatrix` objects.

**Details**
The `worldMatrix` objects must all have the same extents.

**Value**

`worldArray` object.

**Author(s)**
Sarah Bauduin
Examples
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = 1:25)
w2 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4, data = 25:1)
w3 <- stackWorlds(w1, w2)
plot(w3)

Description
Compute the difference between headings.

Usage
subHeadings(angle1, angle2, range360 = FALSE)
## S4 method for signature 'numeric,numeric'
subHeadings(angle1, angle2, range360 = FALSE)
## S4 method for signature 'agentMatrix,numeric'
subHeadings(angle1, angle2, range360 = FALSE)
## S4 method for signature 'numeric,agentMatrix'
subHeadings(angle1, angle2, range360 = FALSE)
## S4 method for signature 'agentMatrix,agentMatrix'
subHeadings(angle1, angle2, range360 = FALSE)

Arguments
angle1 AgentMatrix object representing the moving agents, or Numeric. Vector of angles.
ageangle2 AgentMatrix object representing the moving agents, or Numeric. Vector of angles.
range360 Logical. If range360 = TRUE, returned values are between 0 and 360 degrees; if range360 = FALSE, returned values are between -180 and 180 degrees. Default is range360 = FALSE.

Details
This function does the opposite as the one in NetLogo where angle1 is the target heading. angle1 and angle2 must be of the same length or if different, one of them must be of length 1. Positive values mean clockwise rotations, negative value mean counterclockwise rotations.
Value

Numeric. Vector of the smallest angles in degrees by which angle1 could be rotated to produce angle2 (i.e., the target heading).

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#subtract-headings

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createOTurtles(n = 10, world = w1)
subHeadings(angle1 = t1, angle2 = 0)
```

---

**tExist**

*Do the turtle exist?*

**Description**

Report TRUE if a turtle exists inside the turtles, report FALSE otherwise.

**Usage**

```r
tExist(turtles, who, breed)
```

**Arguments**

- `turtles` AgentMatrix object representing the moving agents.
- `who` Integer. Vector of the who numbers for the selected turtles.
- `breed` Characters. Vector of breed names for the selected turtles. If missing, there is no distinction based upon breed.
Value

Logical. Vector of TRUE or FALSE if the who numbers with any of the breed, if provided, exist or not inside the turtles.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#member

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
     breed = c(rep("sheep", 5), rep("wolf", 5)))
tExist(turtles = t1, who = 3, breed = "sheep")
tExist(turtles = t1, who = 9, breed = "sheep")
tExist(turtles = t1, who = 9, breed = c("sheep", "wolf"))
tExist(turtles = t1, who = c(3, 9))

towards

Directions towards

Description

Report the directions of each agents towards each corresponding agents2.

Usage

towards(agents, agents2, world, torus = FALSE)

## S4 method for signature 'matrix,matrix'
towards(agents, agents2, world, torus = FALSE)
Towards

Arguments

agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

agents2 Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents, or Matrix (ncol = 2) with the first column x and the second column y representing locations coordinates.

world WorldMatrix or worldArray object.

torus Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

agents and agents2 must have the same number of agents/locations or if different, one of them must have only one agent/location. If agents and agents2 have the same number of agents/locations, the directions are calculated for each pair agents[i] and agents2[i] and not for each agents towards every single agents2.

If torus = FALSE, world does not need to be provided.

If torus = TRUE and the distance from one agents to its corresponding agents2 is smaller around the sides of the world than across it, then the direction to agents2 going around the sides of the world is returned.

The direction from a patch to its location returns 0; the direction from a turtle to its location returns the turtle’s heading.

Value

Numeric. Vector of angles in degrees of length equal to the largest number of agents/locations between agents and agents2.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#towards
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#towardsxy
turtle

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4)
towards(agents = patches(w1), agents2 = cbind(x = 0, y = 0))
t1 <- createTurtles(n = 10, world = w1)
towards(agents = t1, agents2 = cbind(x = 0, y = 0))

---

turtle  Select turtles

Description

Report the individuals among turtles based on their who numbers and breed.

Usage

turtle(turtles, who, breed)

## S4 method for signature 'agentMatrix,numeric,missing'
turtle(turtles, who)

## S4 method for signature 'agentMatrix,numeric,character'
turtle(turtles, who, breed)

Arguments

- **turtles**: AgentMatrix object representing the moving agents.
- **who**: Integer. Vector of the who numbers for the selected turtles.
- **breed**: Characters. Vector of breed names for the selected turtles. If missing, there is no distinction based upon breed.

Details

If no turtle matches the given who numbers, with potentially one of the given breed, inside turtles, then an empty agentMatrix is returned.

If there are duplicates who numbers among the turtles, the first matching turtle with the requested who number is returned.

Value

AgentMatrix of the selected turtles sorted in the order of the who numbers requested. If breed was provided, the turtles selected are of one of the breed.

Author(s)

Sarah Bauduin
References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#turtle

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
t2 <- turtle(t1, who = 2)

turtles2spdf  From agentMatrix to SpatialPointsDataFrame

Description

Convert an agentMatrix object into a SpatialPointsDataFrame object.

Usage

turtles2spdf(turtles)

## S4 method for signature 'agentMatrix'
turtles2spdf(turtles)

Arguments

turtles            AgentMatrix object representing the moving agents.

Value

SpatialPointsDataFrame object representing the moving agents (coordinates and data) as contained in turtles.

Author(s)

Sarah Bauduin

Examples

  t1 <- createTurtles(n = 10, coords = cbind(xcor = 1:10, ycor = 1:10))
  sp1 <- turtles2spdf(turtles = t1)
turtlesAt

**Description**

Report the individuals among turtles that are located on the patches at \((dx, dy)\) distances of the agents.

**Usage**

```r
 turtlesAt(world, turtles, agents, dx, dy, breed, torus = FALSE)
```

## S4 method for signature

```r
'worldNLR,agentMatrix,matrix,numeric,numeric,missing'
```

```r
turtlesAt(world, turtles, agents, dx, dy, torus)
```

## S4 method for signature

```r
'worldNLR,agentMatrix,matrix,numeric,numeric,character'
```

```r
turtlesAt(world, turtles, agents, dx, dy, breed, torus = FALSE)
```

**Arguments**

- `world`  
  WorldMatrix or worldArray object.

- `turtles`  
  AgentMatrix object representing the moving agents.

- `agents`  
  Matrix \((ncol = 2)\) with the first column `pxcor` and the second column `pycor` representing the patches coordinates, or AgentMatrix object representing the moving agents.

- `dx`  
  Numeric. Vector of distances to the east (right) from the agents. If `dx` is negative, the distance to the west (left) is computed. `dx` must be of length 1 or of the same length as number of patches or turtles in agents.

- `dy`  
  Numeric. Vector of distances to the north (up) from the agents. If `dy` is negative, the distance to the south is computed (down). `dy` must be of length 1 or of the same length as number of patches or turtles in agents.

- `breed`  
  Characters. Vector of breed names for the selected turtles. If missing, there is no distinction based upon breed.

- `torus`  
  Logical to determine if the world is wrapped. Default is `torus = FALSE`.

**Details**

If the patch at distance \((dx, dy)\) of an agent is outside of the world's extent and `torus = FALSE`, no turtle is returned; if `torus = TRUE`, the turtle located on the patch whose coordinates are defined from the wrapped `world` is returned.

**Value**

AgentMatrix representing the individuals among turtles of any of the given breed, if specified, which are located on the patches at \((dx, dy)\) distances of the agents.
turtleSet

Author(s)
Sarah Bauduin

References

See Also
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#turtles-at
https://ccl.northwestern.edu/netlogo/docs/dictionary.html#at-points

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 10, coords = cbind(xcor = 0:9, ycor = 0:9),
breed = c(rep("sheep", 5), rep("wolf", 5)))
t2 <- turtlesAt(world = w1, turtles = t1, agents = turtle(t1, who = 0),
  dx = 1, dy = 1)
t3 <- turtlesAt(world = w1, turtles = t1,
  agents = patch(w1, c(3,4,5), c(3,4,5)), dx = 1, dy = 1,
  breed = "sheep")

turtleSet Create a turtle agentset

Description
Report a turtle agentset containing all unique turtles provided in the inputs.

Usage
turtleSet(...)  

## S4 method for signature 'agentMatrix'
turtleSet(...)

Arguments
...

AgentMatrix objects representing the moving agents.
Details

Duplicated turtles are identified based only on their who numbers. The turtle chosen for a who number is the first one given in the inputs. To keep all turtles from the inputs, use NLset() to reassign who numbers in some of the inputs, prior using turtleSet(), to avoid turtles with duplicated who numbers.

Value

AgentMatrix object containing all the unique turtles.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#turtle-set

Examples

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10), breed = "sheep")
t2 <- createTurtles(n = 2, coords = randomXYcor(w1, n = 2), breed = "wolf")
t2 <- NLset(turtles = t2, agents = t2, var = "who", val = c(10, 11))
t3 <- createTurtles(n = 1, coords = randomXYcor(w1, n = 1), breed = "sheperd")
t3 <- NLset(turtles = t3, agents = t3, var = "who", val = 12)
t4 <- turtleSet(t1, t2, t3)
```

---

**turtlesOn**

<table>
<thead>
<tr>
<th>turtlesOn</th>
<th>Turtles on</th>
</tr>
</thead>
</table>

Description

Report the individuals among turtles that are on the same patches as the agents.
Usage

turtlesOn(world, turtles, agents, breed, simplify = TRUE)

## S4 method for signature 'worldNLR,agentMatrix,matrix,missing'
turtlesOn(world, turtles, agents, simplify)

## S4 method for signature 'worldNLR,agentMatrix,matrix,character'
turtlesOn(world, turtles, agents, breed, simplify = TRUE)

Arguments

world WorldMatrix or worldArray object.
turtles AgentMatrix object representing the moving agents.
agents Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or
AgentMatrix object representing the moving agents.
breed Characters. Vector of breed names for the selected turtles. If missing, there is no distinction based upon breed.
simplify Logical. If simplify = TRUE, all turtles on the same patches as any agents are returned; if simplify = FALSE, the turtles are evaluated for each agents's patches individually.

Details

The agents must be located inside the world's extent.

Value

AgentMatrix representing any individuals from turtles of any of the given breed, if specified, located on the same patches as any of the agents, if simplify = TRUE, or
Matrix (ncol = 2) with the first column whoTurtles and the second column id showing which turtles are on the same patches as which agents represented by id, if simplify = FALSE. id represents and follows the order of the agents. id does not represent the who numbers of the agents if agents are turtles.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#turtles-on
Example

```r
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9,
data = runif(100))
t1 <- createTurtles(n = 500, coords = randomXYcor(w1, n = 500))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

t2 <- turtlesOn(world = w1, turtles = t1, agents = patch(w1, 2, 2))
```

---

### turtlesOwn

**New turtles variable**

<table>
<thead>
<tr>
<th>turtlesOwn</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new variable for the turtles.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

```r
turtlesOwn(turtles, tVar, tVal)
```

```
## S4 method for signature 'agentMatrix,character,missing'
turtlesOwn(turtles, tVar)
```

```
## S4 method for signature 'agentMatrix,character,ANY'
turtlesOwn(turtles, tVar, tVal)
```

**Arguments**

- **turtles**: AgentMatrix object representing the moving agents.
- **tVar**: Character. the name of the turtles variable to create.
- **tVal**: Vector representing the values of tVar. Must be of length 1 or of length turtles. If missing, NA is given.

**Value**

AgentMatrix representing the turtles with the new variable tVar added.

**Author(s)**

Sarah Bauduin

**References**

updateList

See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#turtles-own

Examples

```r
  t1 <- createTurtles(n = 5, coords = cbind(xcor = 0, ycor = 0))
  t1 <- turtlesOwn(turtles = t1, tVar = "sex", tVal = c("F", "F", "F", "M", "M"))
```

updateList

Update elements of a named list with elements of a second named list

Description

Merge two named list based on their named entries. Where any element matches in both lists, the
value from the second list is used in the updated list. Subelements are not examined and are simply
replaced. If one list is empty, then it returns the other one, unchanged.

Usage

```r
updateList(x, y)
```

Arguments

- `x`, `y` a named list

Value

A named list, with elements sorted by name. The values of matching elements in list `y` replace the
values in list `x`.

Author(s)

Alex Chubaty
Examples

L1 <- list(a = "hst", b = NA_character_, c = 43)
L2 <- list(a = "gst", c = 42, d = list(letters))
updateList(L1, L2)

updateList(L1, NULL)
updateList(NULL, L2)
updateList(NULL, NULL) # should return empty list

uphill

Move uphill

Description

Move the turtles to their neighboring patch with the highest value.

Usage

uphill(world, pVar, turtles, nNeighbors, torus = FALSE)

## S4 method for signature 'worldMatrix,missing,agentMatrix,numeric'
uphill(world, turtles, nNeighbors, torus)

## S4 method for signature 'worldArray,character,agentMatrix,numeric'
uphill(world, pVar, turtles, nNeighbors, torus = FALSE)

Arguments

world WorldMatrix or worldArray object.
pVar Character. If the world is a worldArray object, pVar is the name of the layer to use to define the patches values. pVar must not be provided if the world is a worldMatrix object.
turtles AgentMatrix object representing the moving agents.
nNeighbors Integer: 4 or 8. Represents the number of neighbor patches considered.
torus Logical to determine if the world is wrapped. Default is torus = FALSE.

Details

If no neighboring patch has a larger value than the patch where the turtle is currently located on, the turtle stays on this patch. It still moves to the patch center if it was not already on it.

If there are multiple neighboring patches with the same highest value, the turtle chooses one patch randomly.

If a turtle is located on a patch on the edge of the world and torus = FALSE, it has fewer neighboring patches as options to move than nNeighbors; if torus = TRUE, the turtle can move on the other side of the world to move uphill and its choice of neighboring patches is always equals to nNeighbors.
withMax

Value

AgentMatrix representing the turtles with updated coordinates and updated data for their heading values and previous coordinates prevX and prevY.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#uphill

Examples

```r
w1 <- createWorld(minPxcor = 1, maxPxcor = 10, minPycor = 1, maxPycor = 10, data = runif(100))
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)

t1 <- uphill(world = w1, turtles = t1, nNeighbors = 8)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

withMax Agents with maximum

Description

Report the patches or turtles among agents which have their variable equals to the maximum value.

Usage

```r
withMax(agents, world, var)
```

## S4 method for signature 'matrix,worldMatrix,missing'
```
withMax(agents, world)
```

## S4 method for signature 'matrix,worldArray,character'
```
withMax(agents, world, var)
```

## S4 method for signature 'agentMatrix,missing,character'
```
withMax(agents, var)
```
withMax

Arguments

agents  Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates, or AgentMatrix object representing the moving agents.

world  WorldMatrix or worldArray object.

var  Character. The name of the selected agents variable. If agents are patches and the world is a worldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles’ variable and can be equal to xcor, ycor, any of the variables created when turtles were created, as well as any variable created using turtlesOwn().

Details

world must not be provided if agents are turtles.

Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches among the agents which have their variable equal to the maximum value among the agents, or AgentMatrix representing the turtles among the agents which have their variable var equal to the maximum value among the agents.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#with-max

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
                 data = sample(1:5, size = 25, replace = TRUE))
plot(w1)
p1 <- withMax(agents = patches(w1), world = w1)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
                 heading = sample(1:3, size = 10, replace= TRUE))
withMin

\[ \text{t2} \leftarrow \text{withMax}(\text{agents} = \text{t1}, \text{var} = \text{"heading"}) \]

---

**withMin**  
**Agents with minimum**

**Description**

Report the patches or turtles among agents which have their variable equals to the minimum value.

**Usage**

```
withMin(agents, world, var)
```

---

**Arguments**

- **agents**
  - Matrix (ncol = 2) with the first column `pxcor` and the second column `pycor` representing the patches coordinates, or `AgentMatrix` object representing the moving agents.
- **world**
  - WorldMatrix or worldArray object.
- **var**
  - Character. The name of the selected agents variable. If agents are patches and the world is a WorldMatrix object, var must not be provided. If agents are patches and the world is a worldArray object, var is the name of the layer to use to define the patches values. If agents are turtles, var is one of the turtles’ variable and can be equal to `xcor`, `ycor`, any of the variables created when turtles were created, as well as any variable created using `turtlesOwn()`.

---

**Details**

world must not be provided if agents are turtles.
Value

Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the coordinates of the patches among the agents which have their variable equal to the minimum value among the agents, or

AgentMatrix representing the turtles among the agents which have their variable var equal to the minimum value among the agents.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#with-min

Examples

# Patches
w1 <- createWorld(minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
                 data = sample(1:5, size = 25, replace = TRUE))
plot(w1)
p1 <- withMin(agents = patches(w1), world = w1)

# Turtles
t1 <- createTurtles(n = 10, coords = randomXYcor(w1, n = 10),
                    heading = sample(1:3, size = 10, replace= TRUE))
t2 <- withMin(agents = t1, var = "heading")

world2raster

Convert a worldMatrix or worldArray object into a Raster* object

Description

Convert a worldMatrix object into a RasterLayer object or a worldArray object into a RasterStack object
Usage

world2raster(world)

## S4 method for signature 'worldMatrix'
world2raster(world)

## S4 method for signature 'worldArray'
world2raster(world)

Arguments

world                  WorldMatrix or worldArray object.

Details

The Raster* returned has the same extent and resolution as the world with round coordinates at the center of the cells and coordinates x . 5 at the edges of the cells.

Value

RasterLayer or RasterStack object depending on the input world. Patches value are retained from the world.

Author(s)

Sarah Bauduin

Examples

w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = runif(100))
r1 <- world2raster(w1)
plot(r1)

---

worldArray-class  The worldArray class

Description

This is an s4 class extension of array. It is a collection of several worldMatrix objects with the same extent (i.e., same values for all their slots) stacked together. It is used to keep more than one value per patch.

Author(s)

Sarah Bauduin, Eliot McIntire, and Alex Chubaty
worldHeight

See Also

worldMatrix

worldHeight  World height

Description

Report the height of the world in patch number.

Usage

worldHeight(world)

## S4 method for signature 'worldNLR'

worldHeight(world)

Arguments

world WorldMatrix or worldArray object.

Value

Integer.

Author(s)

Sarah Bauduin

References


See Also

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#world-dim

Examples

w1 <- createWorld()
worldHeight(w1)
Description

This is an s4 class extension of matrix with 7 additional slots. A worldMatrix object can be viewed as a grid composed of squared patches (i.e., matrix cells). Patches have two spatial coordinates pxcor and pycor, representing the location of their center. pxcor and pycor are always integer and increment by 1. pxcor increases as you move right and pycor increases as you move up. pxcor and pycor can be negative if there are patches to the left or below the patch [pxcor = 0, pycor = 0].

Details

The first four slots of the worldMatrix are: minPxcor, maxPxcor, minPycor, maxPycor which represent the minimum and maximum patches coordinates in the worldMatrix. The slot extent is similar to a Raster* extent. Because pxcor and pycor represent the spatial location at the center of the patches and the resolution of them is 1, the extent of the worldMatrix is equal to xmin = minPxcor -0.5, xmax = maxPxcor + 0.5, ymin = minPycor -0.5, and ymax = maxPycor + 0.5. The number of patches in a worldMatrix is equal to ((maxPxcor -minPxcor) + 1) * ((maxPycor -minPycor) + 1). The slot res is equal to 1 as it is the spatial resolution of the patches. The last slot pCoords is a matrix representing the patches coordinates of all the matrix cells in the order of cells in a Raster* (i.e., by rows).

Careful: The methods [] and [] <- retrieve or assign values for the patches in the given order of the patches coordinates provided. When no patches coordinates are provided, the values retrieved or assigned is done in the order of the cell numbers as defined in in Raster* objects (i.e., by rows).

Author(s)

Sarah Bauduin, Eliot McIntire, and Alex Chubaty

References


See Also

worldArray
The `worldNLR` class is the union of the `worldMatrix` and `worldArray` classes. Mostly used for building function purposes.

**Author(s)**
Sarah Bauduin, and Eliot McIntire

---

Report the width of the world in patch number.

**Usage**

```
worldWidth(world)
```

**Arguments**

- `world` WorldMatrix or worldArray object.

**Value**

Integer.

**Author(s)**

Sarah Bauduin

**References**


**See Also**

https://ccl.northwestern.edu/netlogo/docs/dictionary.html#world-dim
Examples

```r
w1 <- createWorld()
worldWidth(w1)
```

---

**wrap**

Wrap coordinates or pixels in a torus-like fashion

---

**Description**

Generally for model development purposes.

**Usage**

```r
wrap(obj, bounds, withHeading)
```

## S4 method for signature 'matrix,Extent,missing'
```r
wrap(obj, bounds)
```

## S4 method for signature 'SpatialPoints,ANY,missing'
```r
wrap(obj, bounds)
```

## S4 method for signature 'matrix,Raster,missing'
```r
wrap(obj, bounds)
```

## S4 method for signature 'matrix,Raster,missing'
```r
wrap(obj, bounds)
```

## S4 method for signature 'matrix,matrix,missing'
```r
wrap(obj, bounds)
```

## S4 method for signature 'SpatialPointsDataFrame,Extent,logical'
```r
wrap(obj, bounds, withHeading)
```

## S4 method for signature 'SpatialPointsDataFrame,Raster,logical'
```r
wrap(obj, bounds, withHeading)
```

## S4 method for signature 'SpatialPointsDataFrame,matrix,logical'
```r
wrap(obj, bounds, withHeading)
```

**Arguments**

- **obj** A `SpatialPoints*` object, or matrix of coordinates.
- **bounds** Either a `Raster*`, `Extent`, or `bbox` object defining bounds to wrap around.
- **withHeading** Logical. If `TRUE`, then the previous points must be wrapped also so that the subsequent heading calculation will work. Default `FALSE`. See details.
Details

If withHeading used, then obj must be a SpatialPointsDataFrame that contains two columns, x1 and y1, with the immediately previous agent locations.

Value

Same class as obj, but with coordinates updated to reflect the wrapping.

Author(s)

Eliot McIntire

Examples

library(quickPlot)
library(raster)

xrange <- yrange <- c(-50, 50)
hab <- raster(extent(c(xrange, yrange)))
hab[] <- 0

# initialize agents
N <- 10

# previous points
x1 <- rep(0, N)
y1 <- rep(0, N)

# initial points
starts <- cbind(x = stats::runif(N, xrange[1], xrange[2]),
    y = stats::runif(N, yrange[1], yrange[2]))

# create the agent object
agent <- SpatialPointsDataFrame(coords = starts, data = data.frame(x1, y1))

ln <- rlnorm(N, 1, 0.02) # log normal step length
sd <- 30 # could be specified globally in params

if (interactive()) {
clearPlot()
    Plot(hab, zero.color = "white", axes = "L")
} if (requireNamespace("SpaDES.tools")) {
    for (i in 1:10) {
        agent <- SpaDES.tools::crw(agent = agent,
                        extent = extent(hab), stepLength = ln,
                        stddev = sd, lonlat = FALSE, torus = TRUE)
        if (interactive()) Plot(agent, addTo = "hab", axes = TRUE)
    }
}
Extract or Replace Parts of an Object

Description

Operators acting on vectors, matrices, arrays and lists to extract or replace parts.

Usage

```r
## S4 method for signature 'worldMatrix,numeric,numeric,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'worldMatrix,missing,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'worldMatrix,numeric,numeric,ANY'
x[i, j] <- value

## S4 replacement method for signature 'worldMatrix,missing,missing,ANY'
x[i, j] <- value

## S4 method for signature 'worldArray,numeric,numeric,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'worldArray,missing,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'worldArray,numeric,numeric,matrix'
x[i, j] <- value

## S4 replacement method for signature 'worldArray,missing,missing,matrix'
x[i, j] <- value

## S4 method for signature 'agentMatrix,numeric,numeric,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'agentMatrix,logical,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'agentMatrix,numeric,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'agentMatrix,missing,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'agentMatrix,missing,character,ANY'
x[i, j, ..., drop = TRUE]
```
## S4 method for signature 'agentMatrix,numeric,character,ANY'
\[ x[i, j, \ldots, \text{drop} = \text{TRUE}] \]

## S4 method for signature 'agentMatrix,missing,numeric,ANY'
\[ x[i, j, \ldots, \text{drop} = \text{TRUE}] \]

## S4 replacement method for signature 'agentMatrix,numeric,numeric,numeric'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,missing,numeric,numeric'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,numeric,missing,numeric'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,numeric,character,data.frame'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,numeric,numeric,character'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,missing,numeric,character'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,missing,character,character'
\[ x[i, j] \leftarrow \text{value} \]

## S4 replacement method for signature 'agentMatrix,numeric,character,character'
\[ x[i, j] \leftarrow \text{value} \]

## S4 method for signature 'agentMatrix'
\[ x$name \]

### Arguments

- **x**: A `agentMatrix` object from which to extract element(s) or in which to replace element(s).
- **i**: Indices specifying elements to extract or replace.
- **j**: see i.
- **...**: other named arguments
- **drop**: not implemented
- **value**: Any R object
- **name**: documentation needed
Note

Extract methods for agentMatrix class will generally maintain the agentMatrix class. This means that there will still be coordinates, character columns represented as numerics etc. $ is for extracting the raw columns and does not maintain the agentMatrix class. [] will extract all values, and result in a data.frame with the correct character and numeric columns.

[[ ]] is for extracting the raw columns and does not maintain the agentMatrix class. [] will extract all values, and result in a data.frame with the correct character and numeric columns.

[[ ]] Subsetting for worldArray class

Description

These function similarly to [[ for RasterStack objects.

Usage

## S4 method for signature 'worldArray,ANY,ANY'
x[[i]]

## S4 replacement method for signature 'worldArray,ANY,ANY'
x[[i]] <- value

## S4 method for signature 'worldArray'
x$name

Arguments

x A worldArray object.
i Index number or layer name specifying a subset of layer(s) from the worldArray.
value A replacement worldMatrix layer for one of the current layers in the worldArray.
name Layer name, normally without back ticks, unless has symbols.
Index

*Topic **integral**
  fargs, 26
  .bboxCoords (extent, wordNLR-method), 24
  .identifyGrobToPlot, worldArray, quickPlotGrob-method (numLayers, worldArray-method), 65
  .plotGrob, agentMatrix-method (numLayers, worldArray-method), 65
  ==, agentMatrix, character-method, 5
  ==, agentMatrix, numeric-method (==, agentMatrix, character-method), 5
  [, 122
  [, agentMatrix, logical, missing, ANY-method, 122
  [, agentMatrix, missing, character, ANY-method, 122
  [, agentMatrix, missing, missing, missing-method, 122
  [, agentMatrix, missing, numeric, ANY-method, 122
  [, agentMatrix, numeric, character, ANY-method, 122
  [, agentMatrix, numeric, missing, ANY-method, 122
  [, worldArray, missing, missing, ANY-method, 122
  [, worldArray, numeric, numeric, ANY-method, 122
  [<=[], 122
  [<=[], agentMatrix, missing, numeric, character-method ([], 122
  [<=[], agentMatrix, missing, numeric, numeric-method ([], 122
  [<=[], agentMatrix, numeric, character, character-method ([], 122
  [<=[], agentMatrix, numeric, character, data.frame-method ([], 122
  [<=[], agentMatrix, numeric, missing, numeric-method ([], 122
  [<=[], agentMatrix, numeric, numeric, character-method ([], 122
  [<=[], agentMatrix, numeric, numeric, numeric-method ([], 122
  [<=[], worldArray, missing, missing, matrix-method ([], 122
  [<=[], worldArray, numeric, numeric, matrix-method ([], 122
  [<=[], worldMatrix, missing, missing, ANY-method ([], 122
  [<=[], worldMatrix, numeric, numeric, ANY-method ([], 122
  [][, 124
  [[], worldArray, ANY, ANY-method ([[], 124
  [<=][[], 124
  [<=][[, worldArray, ANY, ANY-method [[[], 124
  $([[], 124
  $, agentMatrix-method ([], 122
  $, worldArray-method [[[], 124
agentClasses (agentClasses-class), 6
agentClasses-class, 6
agentMatrix, 6
agentMatrix-class, 7
agentMatrix, matrix-method (agentMatrix), 6
agentMatrix, missing-method (agentMatrix), 6
any (NLany), 54
bbox, 24, 25
downhill, worldArray, character, agentMatrix, numeric-method
  (downhill), 20
downhill, worldMatrix, numeric, missing, agentMatrix, numeric-method
  (downhill), 20
dx, 22
dx, agentMatrix, missing-method (dx), 22
dx, agentMatrix, numeric-method (dx), 22
dy, 23
dy, agentMatrix, missing-method (dy), 23
dy, agentMatrix, numeric-method (dy), 23
extent, 24
extent, agentMatrix-method
  (extent, worldNLR-method), 24
extent, worldNLR-method, 24
extent, worldNLR-method
extent, worldNLR-method, 24

bk, 8
die, 18
face, 25
die, agentMatrix, numeric-method (die), 18
die, agentMatrix, numeric-method (die), 18
diffuse, 19
diffuse, worldArray, character, numeric, numeric-method
  (diffuse), 19
diffuse, worldArray, character, numeric, numeric-method
  (diffuse), 19
diffuse, worldMatrix, missing, numeric, numeric-method
  (diffuse), 19
diffuse, worldMatrix, missing, numeric, numeric-method
  (diffuse), 19
dist (NLdist), 56
downhill, 20
INDEX

layoutCircle, 37
layoutCircle, worldNLR, agentMatrix, numeric-method (layoutCircle), 37
left, 38
left, agentMatrix, numeric-method (left), 38
length, agentMatrix-method (show, agentMatrix-method), 93
maxNof, 40
maxNof, agentMatrix, numeric, missing, character-method (maxNof), 40
maxNof, matrix, numeric, worldArray, character-method (maxNof), 40
maxOneOf, 41
maxOneOf, agentMatrix, missing, character-method (maxOneOf), 41
maxOneOf, matrix, worldArray, character-method (maxOneOf), 41
maxOneOf, matrix, worldMatrix, missing-method (maxOneOf), 41
maxPxcor, 43
maxPxcor, worldNLR-method (maxPxcor), 43
maxPycor, 44
maxPycor, worldNLR-method (maxPycor), 44
minNof, 45
minNof, agentMatrix, numeric, missing, character-method (minNof), 45
minNof, matrix, numeric, worldArray, character-method (minNof), 45
minNof, matrix, numeric, worldMatrix, missing-method (minNof), 45
minOneOf, 46
minOneOf, agentMatrix, missing, character-method (minOneOf), 46
minOneOf, matrix, worldArray, character-method (minOneOf), 46
minOneOf, matrix, worldMatrix, missing-method (minOneOf), 46
minPxcor, 48
minPxcor, worldNLR-method (minPxcor), 48
minPycor, 49
minPycor, worldNLR-method (minPycor), 49
moveTo, 50
moveTo, agentMatrix, matrix-method (moveTo), 50
neighbors, 51
theo,

NetLogo (NetLogoR-package), 5
NetLogoR-package, 5
NLall, 52
NLall, agentMatrix, missing, character-method (NLall), 52
NLall, matrix, worldArray, character-method (NLall), 52
NLall, matrix, worldMatrix, missing-method (NLall), 52
NLAny, matrix-method (NLAny), 54
NLcount, 55
NLcount, matrix-method (NLcount), 55
NLdist, 56
NLdist, matrix, matrix-method (NLdist), 56
NLset, 58
NLset, missing, agentMatrix, agentMatrix, character-method (NLset), 58
NLset, worldArray, missing, matrix, character-method (NLset), 58
NLset, worldMatrix, missing, matrix, missing-method (NLset), 58
NLwith, 59
NLwith, agentMatrix, missing, character-method (NLwith), 59
NLwith, matrix, worldArray, character-method (NLwith), 59
NLwith, matrix, worldMatrix, missing-method (NLwith), 59
NLworldIndex, 61
NLworldIndex, worldMatrix, numeric-method (NLworldIndex), 61
n1Of, 62
n1Of, matrix, numeric-method (n1Of), 62
n2Of, 67
n2Of, missing, agentMatrix, character-method (n2Of), 67
nrow, agentMatrix-method (show, agentMatrix-method), 93
numLayers, worldArray-method, 65
of, 67
of, missing, agentMatrix, character-method (of), 67
of, worldArray, matrix, character-method (of), 67
of, worldMatrix, matrix, missing-method
(of), 67
oneOf, 68
oneOf, matrix-method (oneOf), 68
other, 70
other, matrix, matrix-method (other), 70

patch, 71
patch, worldNLR, numeric, numeric-method
(patch), 71
patchAhead, 72
patchAhead, worldNLR, agentMatrix, numeric-method
(patchAhead), 72
patchAt, 74
patchAt, worldNLR, matrix, numeric, numeric-method
(patchAt), 74
patchDistDir, 75
patchDistDir, worldNLR, matrix, numeric, numeric-method
(patchDistDir), 75
patches, 76
patches, worldNLR-method (patches), 76
patchHere, 77
patchHere, worldNLR, agentMatrix-method
(patchHere), 77
patchLeft, 78
patchLeft, worldNLR, agentMatrix, numeric, numeric-method
(patchLeft), 78
patchRight, 80
patchRight, worldNLR, agentMatrix, numeric, numeric-method
(patchRight), 80
patchSet, 81
patchSet, matrix-method (patchSet), 81
pExist, 82
pExist, worldNLR, numeric, numeric-method
(pExist), 82
plot, agentMatrix, 83
plot, worldArray (plot, agentMatrix), 83
plot, worldMatrix (plot, agentMatrix), 83
points, agentMatrix (plot, agentMatrix), 83
PxcorPycorFromCell, 84
PxcorPycorFromCell, worldNLR, numeric-method
(PxcorPycorFromCell), 84

randomPxcor, 85
randomPxcor, worldNLR, numeric-method
(randomPxcor), 85
randomPxcor, 86
randomXcor, 87
randomXcor, worldNLR, numeric-method
(randomXcor), 87
randomXYcor, 88
randomXYcor, worldNLR, numeric-method
(randomXYcor), 88
randomYcor, 88
randomYcor, worldNLR, numeric-method
(randomYcor), 88
raster2world, 89
raster2world, RasterLayer, character-method
(raster2world), 89
raster2world, RasterStack, character-method
(raster2world), 89
rbind (cbind), 11
right, agentMatrix, numeric-method
(right), 90

set, (NLset), 58
setXY, 92
setXY, agentMatrix, numeric, numeric, missing, ANY-method
(setXY), 92
setXY, agentMatrix, numeric, numeric, worldNLR, logical-method
(setXY), 92
show, agentMatrix-method, 93
show, Array-method, 94
show, worldMatrix-method
(show, worldArray-method), 94
sortOn, 94
sortOn, agentMatrix, missing, character-method
(sortOn), 94
sortOn, matrix, worldArray, character-method
(sortOn), 94
sortOn, matrix, worldMatrix, missing-method
(sortOn), 94
spdf2turtles, 96
spdf2turtles, SpatialPointsDataFrame-method
(spdf2turtles), 96
sprout, 97
sprout, numeric, matrix-method (sprout), 97

stackWorlds, 98
stackWorlds, worldMatrix-method
(stackWorlds), 98
subHeadings, 99
wrap, matrix, matrix, missing-method  
  (wrap). 120
wrap, matrix, Raster, missing-method  
  (wrap). 120
wrap, SpatialPoints, ANY, missing-method  
  (wrap). 120
wrap, SpatialPointsDataFrame, Extent, logical-method  
  (wrap). 120
wrap, SpatialPointsDataFrame, matrix, logical-method  
  (wrap). 120
wrap, SpatialPointsDataFrame, Raster, logical-method  
  (wrap). 120