

# Package ‘NetLogoR’

July 21, 2025

**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 two models from the NetLogo software repository (Ants <<http://ccl.northwestern.edu/netlogo/models/Ants>>) and Wolf-Sheep-Predation (<<http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation>>), and a third, Butterfly, from Railsback and Grimm (2012) <<https://www.railsback-grimm-abm-book.com/>>, all 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** <https://netlogor.predictiveecology.org>,  
<https://github.com/PredictiveEcology/NetLogoR/>,  
<https://groups.google.com/g/netlogor>

**Version** 1.0.5

**Date** 2023-12-21

**Depends** R (>= 4.1)

**Imports** data.table, grDevices, methods, quickPlot (>= 1.0.2), stats, terra, utils

**Suggests** CircStats, covr, fastshp, knitr, microbenchmark, raster, rmarkdown, sf, sp, SpaDES.core (>= 2.0.3), SpaDES.tools (>= 2.0.5), testthat, withr

**Additional\_repositories** <https://predictiveecology.r-universe.dev/>

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

**ByteCompile** yes

**Encoding** UTF-8

**Language** en-CA

**License** GPL-3

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr, rmarkdown

**Collate** 'Agent-classes.R' 'NetLogoR-package.R' 'worldNLR-classes-methods.R' 'helpers.R' 'agentMatrix-Class-methods.R' 'agentset-functions.R' 'function-arguments.R' 'patch-functions.R' 'plot.R' 'quickPlot.R' 'spades-functions.R' 'turtle-functions.R' 'world-functions.R'

**NeedsCompilation** no

**Author** Sarah Bauduin [aut] (ORCID: <<https://orcid.org/0000-0002-3252-5894>>), Eliot J B McIntire [aut, cre] (ORCID: <<https://orcid.org/0000-0002-6914-8316>>), Alex M Chubaty [aut] (ORCID: <<https://orcid.org/0000-0001-7146-8135>>), His Majesty the King 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** 2024-01-08 10:30:12 UTC

Contents

NetLogoR-package . . . . .	5
.bboxCoords . . . . .	6
==,agentMatrix,character-method . . . . .	6
agentMatrix . . . . .	7
agentMatrix-class . . . . .	8
bbox . . . . .	9
bk . . . . .	10
canMove . . . . .	12

cbind . . . . .	13
cellFromPxcorPycor . . . . .	13
clearPatches . . . . .	14
coordinates,agentMatrix-method . . . . .	15
createOTurtles . . . . .	16
createTurtles . . . . .	17
createWorld . . . . .	18
die . . . . .	20
diffuse . . . . .	21
downhill . . . . .	22
dx . . . . .	24
dy . . . . .	25
extent,worldNLR-method . . . . .	26
face . . . . .	27
fd . . . . .	28
hatch . . . . .	30
home . . . . .	31
inCone . . . . .	32
initialize,agentMatrix-method . . . . .	34
inRadius . . . . .	35
inspect . . . . .	36
isNLclass . . . . .	37
layoutCircle . . . . .	39
left . . . . .	40
maxNof . . . . .	41
maxOneOf . . . . .	43
maxPxcor . . . . .	45
maxPycor . . . . .	46
minNof . . . . .	47
minOneOf . . . . .	48
minPxcor . . . . .	50
minPycor . . . . .	51
moveTo . . . . .	52
neighbors . . . . .	53
NLall . . . . .	55
NLany . . . . .	56
NLcount . . . . .	57
NLdist . . . . .	58
NLset . . . . .	60
NLwith . . . . .	62
NLworldIndex . . . . .	63
nOf . . . . .	64
noPatches . . . . .	66
noTurtles . . . . .	67
numLayers.worldArray . . . . .	67
of . . . . .	68
oneOf . . . . .	70
other . . . . .	71

patch . . . . .	73
patchAhead . . . . .	74
patchAt . . . . .	76
patchDistDir . . . . .	77
patches . . . . .	78
patchHere . . . . .	79
patchLeft . . . . .	80
patchRight . . . . .	82
patchSet . . . . .	83
pExist . . . . .	84
plot.agentMatrix . . . . .	85
PxcorPycorFromCell . . . . .	86
randomPxcor . . . . .	87
randomPycor . . . . .	88
randomXcor . . . . .	89
randomXYcor . . . . .	90
randomYcor . . . . .	91
raster2world . . . . .	92
right . . . . .	93
setXY . . . . .	94
sf2turtles . . . . .	95
show,agentMatrix-method . . . . .	96
show,worldArray-method . . . . .	97
sortOn . . . . .	97
spatRast2world . . . . .	99
spdf2turtles . . . . .	100
sprout . . . . .	101
stackWorlds . . . . .	103
subHeadings . . . . .	104
tExist . . . . .	105
towards . . . . .	106
turtle . . . . .	108
turtles2sf . . . . .	109
turtles2spdf . . . . .	110
turtlesAt . . . . .	111
turtleSet . . . . .	112
turtlesOn . . . . .	113
turtlesOwn . . . . .	115
updateList . . . . .	116
uphill . . . . .	117
withMax . . . . .	119
withMin . . . . .	120
world2raster . . . . .	122
world2spatRast . . . . .	123
worldArray-class . . . . .	124
worldHeight . . . . .	124
worldMatrix-class . . . . .	125
worldNLR-class . . . . .	126

worldWidth . . . . .	126
wrap . . . . .	127
[ . . . . .	128
[[,worldArray,ANY,missing-method . . . . .	131

<b>Index</b>	<b>132</b>
--------------	------------

---

NetLogoR-package	<i>The NetLogoR package</i>
------------------	-----------------------------

---

## 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")`.

## Details

The 3 specific R examples can be opened here, but it's recommended to make copies of these for editing:

```
exPath <- file.path(system.file(package = "NetLogoR", "examples"))
file.edit(exPath, "Ants", "Ants.R")
file.edit(exPath, "Butterfly", "Butterfly-1.R")
file.edit(exPath, "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 <achubaty@for-cast.ca> ([ORCID](#))

Other contributors:

- His Majesty the King in Right of Canada, as represented by the Minister of Natural Resources Canada [copyright holder]

## See Also

Useful links:

- <https://netlogor.predictiveecology.org>
- <https://github.com/PredictiveEcology/NetLogoR/>
- <https://groups.google.com/g/netlogor>
- Report bugs at <https://github.com/PredictiveEcology/NetLogoR/issues>

---

<code>.bboxCoords</code>	<code>.bboxCoords</code> is a drop in replacement for <code>raster::bboxCoords</code> .
--------------------------	---

---

### Description

`.bboxCoords` is a drop in replacement for `raster::bboxCoords`.

### Usage

```
.bboxCoords(coords)
```

### Arguments

<code>coords</code>	xy coordinates for all cells, e.g., produced by <code>raster::coordinates</code> .
---------------------	--

---

```
==,agentMatrix,character-method
```

*Relational Operators*

---

### Description

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

### Usage

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

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

### Arguments

<code>e1</code>	An <code>agentMatrix</code> object.
<code>e2</code>	atomic vector, symbol, call, or other object for which methods have been written.

### Value

A logical vector indicating the result of the element by element comparison.

---

`agentMatrix`*Create a new agentMatrix object*

---

## 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

```
agentMatrix(..., coords)

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

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

## Arguments

...	Vectors, a data.frame, or a matrix of extra columns to add to the coordinates, or a <code>SpatialPointsDataFrame</code> .
coords	A matrix with 2 columns representing x and y coordinates

## Value

An `agentMatrix` object

## 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))
```

---

agentMatrix-class      *The agentMatrix class*

---

## Description

The agentMatrix class

## Author(s)

Eliot McIntire

## Examples

```
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 (requireNamespace("microbenchmark", quietly = TRUE) &&
  requireNamespace("sp", quietly = TRUE)) {
  microbenchmark::microbenchmark(
    times = 499,
    spdf = {
      sp::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)],
    nums2 = 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
  )
}
)
}

```

bbox

*Extract or set bounding box*

## Description

These are methods for classes in NetLogoR, i.e., `agentMatrix`, `worldMatrix`, and `worldArray`. Replacement method sets the `bbox` attribute of an `agentMatrix`.

## Usage

```
bbox(obj)
```

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

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

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

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

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

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

**Arguments**

obj	object deriving from class "agentMatrix", or for bbox and extent, a "worldMatrix" or "worldArray"
value	2x2 matrix representing the bounding box. See <code>sp::bbox</code> .

**Value**

The replacement method returns the same object as supplied to `obj`, i.e., an `agentMatrix`, with the `bbox` attribute set to `value`.

**See Also**

`extent()`, `coordinates()`, `sp::bbox`

**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
)
bbox(newAgent)
extent(newAgent)
coordinates(newAgent)
```

---

bk	<i>Move backward</i>
----	----------------------

---

**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 <code>torus = FALSE</code> .
out	Logical. Determine if a turtle should move when <code>torus = FALSE</code> and its ending position will be outside of the world's extent. Default is <code>out = TRUE</code> .

## 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 previous coordinates `prevX` and `prevY`.

## Author(s)

Sarah Bauduin

## References

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning 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>>

## Examples

```
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 = 2)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

```
t1 <- bk(turtles = t1, dist = 1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
t1 <- fd(turtles = t1, dist = 0.5)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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	<i>Combine R Objects by Rows or Columns</i>
-------	---

---

**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 [base::cbind\(\)](#).

**Value**

An agentMatrix object.

---

cellFromPxcorPycor	<i>Cells numbers from patches coordinates</i>
--------------------	---

---

**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	<i>Clear world's patches</i>
--------------	------------------------------

---

**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.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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)
```

---

coordinates,agentMatrix-method

*Spatial accessors and setters for NetLogoR classes*

---

**Description**

Spatial accessors and setters for NetLogoR classes

**Usage**

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

**Arguments**

obj	object deriving from class "agentMatrix"
...	additional arguments that may be used by particular methods

**Value**

coordinates returns a matrix of coordinates of the obj.

**See Also**

[bbox\(\)](#), [extent\(\)](#)

---

createOTurtles	<i>Create ordered turtles</i>
----------------	-------------------------------

---

### 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

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

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### See Also

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



**Examples**

```

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 turtles
---------------	----------------

---

**Description**

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

**Usage**

```

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

## S4 method for signature 'numeric,matrix,missing'
createTurtles(n, coords, world, heading, breed, color)

## S4 method for signature 'numeric,missing,ANY'
createTurtles(n, coords, world, heading, breed, color)

```

**Arguments**

<code>n</code>	Integer.
<code>coords</code>	Matrix ( $n_{\text{col}} = 2$ ) with the first column <code>xcor</code> and the second column <code>ycor</code> representing the turtles initial locations. <code>nrow(coords)</code> 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.
<code>world</code>	WorldMatrix or worldArray object.
<code>heading</code>	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.
<code>breed</code>	Character. Vector of breed names. Must be of length 1 or of length $n$ . If missing, <code>breed = "turtle"</code> for all turtles.
<code>color</code>	Character. Vector of color names. Must be of length $n$ . If missing, colors are assigned using the function <code>rainbow(n)</code> .

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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

*Create a world*

---

**Description**

Create a world of patches of class worldMatrix.

**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:

```
`minPxcor` = -16`,  
`maxPxcor` = 16`, `minPycor` = -16`, and `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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**Examples**

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

---

die	<i>Kill</i> turtles
-----	---------------------

---

**Description**

Kill selected turtles.

**Usage**

```
die(turtles, who)

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

**Arguments**

turtles	AgentMatrix object representing the moving agents.
who	Integer. Vector of the who numbers for the selected turtles.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#die>

**Examples**

```
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	<i>Diffuse values in a world</i>
---------	----------------------------------

---

### Description

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

### Usage

```
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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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
if (requireNamespace("SpaDES.tools", quietly = TRUE)) {
  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.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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)

if (requireNamespace("SpaDES.tools", quietly = TRUE)) {
  t1 <- downhill(world = w1, turtles = t1, nNeighbors = 8)
  points(t1, col = of(agents = t1, var = "color"), pch = 16)
}

```

---

dx	<i>x-increment</i>
----	--------------------

---

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#dxy>

## Examples

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

---

dy	<i>y-increment</i>
----	--------------------

---

## 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

```
dy(turtles, dist = 1)

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

## 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#dxy>

**Examples**

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

---

extent, worldNLR-method

*Bounding box and extent methods for NetLogoR classes*

---

**Description**

Same as `sp::bbox` and `raster::extent`.

**Usage**

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

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

**Arguments**

<code>x</code>	object deriving from class "agentMatrix", or a "worldMatrix" or "worldArray"
<code>...</code>	Ignored.

**Value**

`bbox` returns a two-column matrix; the first column has the minimum, the second the maximum values; rows represent the spatial dimensions. `extent` returns an `SpatExtent` object from the package `terra`.

**See Also**

[bbox\(\)](#), [coordinates\(\)](#)

---

face	<i>Face something</i>
------	-----------------------

---

### Description

Set the turtles' heading towards agents2.

### Usage

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

## S4 method for signature 'agentMatrix,matrix'
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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## 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)
```

---

fd	<i>Move forward</i>
----	---------------------

---

## Description

Move the turtles forward with their headings as directions.

## Usage

```
fd(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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

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

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

## Examples

```
w1 <- createWorld(
  minPxcor = 0, maxPxcor = 4, minPycor = 0, maxPycor = 4,
  data = runif(25)
)
t1 <- create0Turtles(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	<i>Hatch new turtles</i>
-------	--------------------------

---

### Description

Create new turtles from parent turtles.

### Usage

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

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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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)
```

---

home

*Return home*


---

**Description**

Move the turtles back home.

**Usage**

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

```
## S4 method for signature 'worldNLR,agentMatrix,character'
home(world, turtles, home)
```

**Arguments**

world	WorldMatrix or worldArray object.
turtles	AgentMatrix object representing the moving agents.
home	Character. Can take one of the following options to define where to relocate the turtles: <div style="margin-left: 40px;"> <pre>`home = "home0"` will place the `turtles` at the location   `x = 0, y = 0`.</pre> <pre>`home = "center"` will place the `turtles` at the center of   the `world`.</pre> <pre>`home = "pCorner"` will place the `turtles` at the center of   the `patch` located in the left bottom corner of the `world`.</pre> <pre>`home = "corner"` will place the `turtles` at the left bottom   corner of the `world`.</pre> </div>

**Value**

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

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**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 = "black", pch = 16)

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

---

inCone

Agents *in cone*


---

**Description**

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

**Usage**

```
inCone(turtles, radius, angle, agents, world, torus = FALSE)

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

**Arguments**

turtles	AgentMatrix object representing the moving agents.
radius	Numeric. Vector of distances from turtles to locate agents. Must be of length 1 or of length turtles.



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 angle / 2 to the direction of their headings plus angle / 2. Must be of length 1 or of length turtles.
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.
torus	Logical to determine if the world is wrapped. Default is torus = FALSE.

### Details

agents are reported if there are within 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 `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 `agents`.

### Value

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

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

### Author(s)

Sarah Bauduin

### References

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### See Also

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#in-cone>

**Examples**

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

if (requireNamespace("sf", quietly = TRUE)) {
  p1 <- inCone(turtles = t1, radius = 2, agents = patches(w1), angle = 90)
  t2 <- inCone(turtles = turtle(t1, who = 0), radius = 2, angle = 90, agents = t1)
}
```

---

```
initialize,agentMatrix-method
```

*Initialize for agentMatrix Class*

---

**Description**

To create a new agentMatrix object.

**Usage**

```
## 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.

**Value**

An agentMatrix object.

---

inRadius	Agents <i>in radius</i>
----------	-------------------------

---

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### See Also

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

### Examples

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

if (requireNamespace("sf", quietly = TRUE)) {
  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 turtles*

---

### Description

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

### Usage

```
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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

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

---

isNLclass	<i>Type of object</i>
-----------	-----------------------

---

**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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## 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	<i>Layout turtles on a circle</i>
--------------	-----------------------------------

---

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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)
```

---

left

---

*Rotate to the left*


---

**Description**

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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## 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 <i>with maximum</i>
--------	------------------------------

---

## 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.

### 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 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### 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")

```

maxOneOf

*One agent with maximum***Description**

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

**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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### 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`*Maximum pxcor*

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#max-pxcor>

**Examples**

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

---

`maxPycor`*Maximum pycor*

---

**Description**

Report the patches maximum pycor in the world.

**Usage**

```
maxPycor(world)
```

```
## S4 method for signature 'worldNLR'  
maxPycor(world)
```

**Arguments**

`world`                      WorldMatrix or worldArray object.

**Value**

Integer.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

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

---

minNof	N agents <i>with minimum</i>
--------	------------------------------

---

## Description

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

## Usage

```
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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#min-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 <- minNof(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 <- 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#min-pcor>

## Examples

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

---

moveTo

*Move to*

---

## Description

Move the turtles to the agents' locations.

## Usage

```
moveTo(turtles, agents)

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

## Arguments

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`.

## 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9)
if (requireNamespace("SpaDES.tools", quietly = TRUE)) {
  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)
}
```

---

NLall	<i>All agents?</i>
-------	--------------------

---

**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





**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#any>

**Examples**

```
# 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()
NLany(t1)
NLany(t2)
```

---

NLcount

---

*Count agents*


---

**Description**

Report the number of patches or turtles inside agents.

**Usage**

```
NLcount(agents)

## S4 method for signature 'matrix'
NLcount(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**

Integer.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#count>

**Examples**

```
# 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**

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

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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

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

**Examples**

```
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)
```

---

NLset

---

*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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### 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))

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

---

NLwith

Agents *with*


---

### 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.

`val` can include `NA`.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#with>

**Examples**

```
# 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**

```
NLworldIndex(world, cellNum)

## S4 method for signature 'worldMatrix,numeric'
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

**Examples**

```
if (requireNamespace("raster", quietly = TRUE)) {
  w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = 1:100)
  w1Ras <- world2raster(w1)
  index <- 24
  pxy <- 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	<i>N random agents</i>
-----	------------------------

---

**Description**

Report n patches or turtles randomly selected among agents.

**Usage**

```
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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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	<i>No patches</i>
-----------	-------------------

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#no-patches>

**Examples**

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

---

`noTurtles`*No turtles*

---

**Description**

Report an empty turtle agentset.

**Usage**

```
noTurtles()
```

**Value**

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

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#no-turtles>

**Examples**

```
t1 <- noTurtles()
NLcount(t1)
```

---

`numLayers.worldArray`    *Methods for quickPlot*

---

**Description**

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

Usage

```
## S3 method for class 'worldArray'
numLayers(x)

## S3 method for class 'worldMatrix'
numLayers(x)

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

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

Arguments

- x                   An object or list of objects.
- object             An object from which to extract the layer names.
- 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).

Value

numLayers returns an integer representing the number of layers in a worldArray or worldMatrix (which is always 1L)

layerNames returns an character vector representing the names of the layers in a worldArray

---

of	<i>Values of an agents variable</i>
----	-------------------------------------

---

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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
# 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)
```

---

other

---

*Others*


---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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 <i>coordinates</i>
-------	----------------------------

---

## 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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)
```

---

patchAt

Patches *at*


---

### 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 (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.
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 (ncol = 2) with the first column pxcor and the second column pycor 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

## References

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## 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 `torus = FALSE` and the patch at distance `dist` and direction angle of an agent 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.

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

### 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 directions angle of agents. The order of the patches follows the order of the agents.

### Author(s)

Sarah Bauduin

### References

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### See Also

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#patch-at-heading-and-distance>

### Examples

```
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

*All the patches in a world*

---

### Description

Report the coordinates of all the patches in the world.

**Usage**

```
patches(world)

## S4 method for signature 'worldNLR'
patches(world)
```

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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
```

---

patchHere

Patches *here*

---

**Description**

Report the coordinates of the patches under the turtles locations.

**Usage**

```
patchHere(world, turtles)

## S4 method for signature 'worldNLR,agentMatrix'
patchHere(world, turtles)
```

**Arguments**

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

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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)
```

---

patchLeft

Patches *on the left*

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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 <i>on the right</i>
------------	-----------------------------

---

### 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 (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 right of their headings. The order of the patches follows the order of the turtles.

### Author(s)

Sarah Bauduin

## References

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

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

## Examples

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

---

patchSet

Patch *set*

---

## Description

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

## Usage

```
patchSet(...)

## S4 method for signature 'matrix'
patchSet(...)
```

## Arguments

...                      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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## 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)
```

---

pExist

*Do the patches exist?*

---

## 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## 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	<i>Basic plot methods for agentMatrix, worldMatrix, worldArray</i>
------------------	--

---

## 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*)

## Value

none; invoked for side-effect of generating a plot.

**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, 2343),
  char2 = LETTERS[c(4, 24, 3)],
  nums = 5:7
)
points(newAgent)

```

---

PxcorPycorFromCell	Patches <i>coordinates from cells numbers</i>
--------------------	---

---

**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

**Examples**

```
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**

```
randomPxcor(world, n)

## 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-pcor>

**Examples**

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

---

randomPycor

*Random pycor*


---

**Description**

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

**Usage**

```
randomPycor(world, n)

## 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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#random-pcor>

**Examples**

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



---

randomXcor	<i>Random xcor</i>
------------	--------------------

---

**Description**

Report n random xcor coordinates within the world's extent.

**Usage**

```
randomXcor(world, n)

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

**Arguments**

world	WorldMatrix or worldArray object.
n	Integer.

**Value**

Numeric. Vector of length n of xcor coordinates.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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),
  ycor = randomYcor(world = w1, n = 10)
))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

randomXYcor	<i>Random turtles coordinates</i>
-------------	-----------------------------------

---

## Description

Report n random xcor and ycor coordinates within the world's extent.

## Usage

```
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

```
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	<i>Random ycor</i>
------------	--------------------

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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),
  ycor = randomYcor(world = w1, n = 10)
))
plot(w1)
points(t1, col = of(agents = t1, var = "color"), pch = 16)
```

---

raster2world	<i>Convert a Raster* object into a worldMatrix or worldArray object</i>
--------------	---

---

## Description

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

## Usage

```
raster2world(raster)
```

```
## S4 method for signature 'ANY'
raster2world(raster)
```

## Arguments

raster                      RasterLayer or RasterStack object.

## Details

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

The number of rows and columns, as well as the cell values of the ``raster`` are kept the same. However, to match the coordinates system and resolution of a ``worldMatrix`` or ``worldArray``, the grid is shifted by a 1/2 cell to have round coordinate values at the center of the patches and patch size is equal to (1,1). The bottom left corner cell coordinates of the ``worldMatrix`` or ``worldArray`` will be (`pxcor = 0`, `pycor = 0`).

## Value

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

## Author(s)

Sarah Bauduin

## Examples

```
if (requireNamespace("raster")) {
  r1 <- raster::raster(raster::extent(c(0, 10, 0, 10)), nrows = 10, ncols = 10)
  r1[] <- runif(100)
  w1 <- raster2world(r1)
  terra::plot(r1)
  terra::plot(w1)
}
```

---

right	<i>Rotate to the right</i>
-------	----------------------------

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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	<i>Set turtles' locations</i>
-------	-------------------------------

---

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

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

---

sf2turtles

*From sf to agentMatrix*

---

**Description**

Convert a sf object into an agentMatrix object.

**Usage**

```
sf2turtles(turtles_sf)  
  
## S4 method for signature 'ANY'  
sf2turtles(turtles_sf)
```

**Arguments**

turtles\_sf      sf object of POINT geometry representing moving agents.

**Details**

If the turtles\_sf 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 turtles\_sf.

**Author(s)**

Sarah Bauduin

**Examples**

```

if (requireNamespace("sf", quietly = TRUE)) {
  turtles_sf1 <- sf::st_as_sf(
    cbind.data.frame(
      x = c(1, 2, 3), y = c(1, 2, 3),
      age = c(0, 0, 3), sex = c("F", "F", "M")
    ),
    coords = c("x", "y")
  )
  t1 <- sf2turtles(turtles_sf = turtles_sf1)
}

```

---

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	an integer vector of length up to dim(x) (or 1, for non-dimensioned objects).
...	arguments to be passed to or from other methods (currently, none used).



**Value**

show is called for its side effects. It shows all columns of data, except for the coordinates. To access those, use coordinates(). length returns a non-negative integer of length 1, except for vectors of more than  $2^{31} - 1$  elements, when it returns a double. nrow returns an integer of length 1 or NULL.

An agentMatrix object, like x, but generally smaller.

---

 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.

**Value**

show is called for its side effects. It shows key metadata elements of the worldArray or worldMatrix, plus the first 4 columns and rows of data

---

 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 a 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#sort-on>

## Examples

```
# 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")

# or
library(quickPlot)

Plot(w1)
Plot(t1, addTo = "w1")
```

---

spatRast2world	<i>Convert a SpatRaster object into a worldMatrix or worldArray object</i>
----------------	--

---

## Description

Convert a SpatRaster object into a worldMatrix object or a worldArray object depending on the number of layers of the SpatRaster object.

## Usage

```
spatRast2world(raster)

## S4 method for signature 'SpatRaster'
spatRast2world(raster)
```

## Arguments

raster                      SpatRaster object.

## Details

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

If the ``SpatRaster`` object has only one layer, a ``worldMatrix`` object will be returned. If the ``SpatRaster`` object has more than one layer, layers must have unique names and a ``worldArray`` object will be returned.

The number of rows and columns, as well as the cell values of the ``raster`` are kept the same. However, to match the coordinates system and resolution of a ``worldMatrix`` or ``worldArray``, the grid is shifted by a 1/2 cell to have round coordinate values at the center of the patches and patch size is equal to (1,1). The bottom left corner cell coordinates of the ``worldMatrix`` or ``worldArray`` will be (`pxcor = 0`, `pycor = 0`).

## Value

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

## Author(s)

Sarah Bauduin

## Examples

```
library(terra)
r1 <- rast(xmin = 0, xmax = 10, ymin = 0, ymax = 10, nrows = 10, ncols = 10)
r1[] <- runif(100)
w1 <- spatRast2world(r1)
terra::plot(r1)
plot(w1)

r2 <- rast(xmin = 0, xmax = 10, ymin = 0, ymax = 10, nrows = 10, ncols = 10)
r2[] <- 0
r3 <- c(r1, r2)
names(r3) <- c("layer1", "layer2")
w3 <- spatRast2world(r3)
terra::plot(r3)
plot(w3)
```

## Description

Convert a SpatialPointsDataFrame object into an agentMatrix object.

**Usage**

```
spdf2turtles(spdf)

## S4 method for signature 'ANY'
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**

```
if (requireNamespace("sp", quietly = TRUE)) {
  sp1 <- sp::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**

```
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.
patches	Matrix (ncol = 2) with the first column pxcor and the second column pycor representing the patches coordinates.
breed	Character. Vector of breed names. Must be of length 1 or of length the number of patches. If missing, breed = turtle for all the sprouted turtles.
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.
color	Character. Vector of color names. Must be of length 1, of length the number of patches or of length sum(n). If missing, colors are assigned using the function rainbow(n).
turtles	AgentMatrix object representing the moving agents.

**Details**

nrow(patches) must be equal to 1 or to n.

If ``turtles`` is provided, the new ``turtles`` are added to the ``turtles`` when returned. The ``who`` numbers of the sprouted ``turtles`` therefore follow the ones from the ``turtles``.

All new sprouted ``turtles`` are placed at the end of the ``agentMatrix`` object.

If no ``turtles`` is provided, a new ``agentMatrix`` is created and the ``who`` numbers start at 0.

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

**Value**

AgentMatrix including the new sprouted turtles.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#sprout>

**Examples**

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

---

stackWorlds

*Stack* worlds

---

**Description**

Stack multiple worldMatrix into a worldArray.

**Usage**

```
stackWorlds(...)

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

**Arguments**

... worldMatrix objects. If passed as unnamed objects, then the function will attempt to use their object names as layer names. Alternatively, to be more reliable, these can be passed as named arguments. See examples.

**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)

# pass named arguments to specify a different name than the object name
w4 <- stackWorlds(layer1 = w1, layer2 = w2)
```

---

subHeadings

*Subtract* headings

---

### 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.
angle2	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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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**

```
tExist(turtles, who, breed)

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

## S4 method for signature 'agentMatrix,numeric,character'
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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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)
```

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#towards>

<<https://ccl.northwestern.edu/netlogo/docs/dictionary.html#towardsxy>>

**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	<i>Select turtles</i>
--------	-----------------------

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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)
```

---

turtles2sf	<i>From agentMatrix to sf</i>
------------	-------------------------------

---

**Description**

Convert an agentMatrix object into an sf object.

**Usage**

```
turtles2sf(turtles)

## S4 method for signature 'ANY'
turtles2sf(turtles)
```

**Arguments**

turtles            AgentMatrix object representing the moving agents.

**Value**

sf object of POINT geometry 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))
if (requireNamespace("sf", quietly = TRUE)) {
  sf_t1 <- turtles2sf(turtles = t1)
}
```

---

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))
if (requireNamespace("sp", quietly = TRUE)) {
  sp1 <- turtles2spdf(turtles = t1)
}
```

---

turtlesAt
Turtles *at*

### Description

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

### Usage

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

```
## S4 method for signature
## 'worldNLR,agentMatrix,matrix,numeric,numeric,missing'
turtlesAt(world, turtles, agents, dx, dy, torus)
```

```
## S4 method for signature
## 'worldNLR,agentMatrix,matrix,numeric,numeric,character'
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.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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

Turtles *on*


---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
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	<i>New turtles variable</i>
------------	-----------------------------

---

**Description**

Create a new variable for the turtles.

**Usage**

```
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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

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

## Examples

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

---

updateList	<i>Update elements of a named list with elements of a second named list</i>
------------	---

---

## 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

```
updateList(x, y)

## S4 method for signature 'list,list'
updateList(x, y)

## S4 method for signature '`NULL`,list'
updateList(x, y)

## S4 method for signature 'list,`NULL`'
updateList(x, y)

## S4 method for signature '`NULL`,`NULL`'
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`.

## 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

## See Also

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

## 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)

if (requireNamespace("SpaDES.tools", quietly = TRUE)) {
  t1 <- uphill(world = w1, turtles = t1, nNeighbors = 8)
  points(t1, col = of(agents = t1, var = "color"), pch = 16)
}
```

---

withMax	Agents <i>with maximum</i>
---------	----------------------------

---

### Description

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

### Usage

```
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)
```

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### 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)
)
t2 <- withMax(agents = t1, var = "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)
```

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



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

## S4 method for signature 'agentMatrix,missing,character'
withMin(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.

### 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

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

### 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))
if (requireNamespace("raster", quietly = TRUE)) {
  r1 <- world2raster(w1)
  terra::plot(r1)
}
```

---

world2spatRast	<i>Convert a worldMatrix or worldArray object into a SpatRaster object</i>
----------------	--

---

**Description**

Convert a worldMatrix object or a worldArray object into a SpatRaster object

**Usage**

```
world2spatRast(world)

## S4 method for signature 'worldMatrix'
world2spatRast(world)

## S4 method for signature 'worldArray'
world2spatRast(world)
```

**Arguments**

world                      WorldMatrix or worldArray object.

**Details**

The SpatRaster 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**

SpatRaster object. 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 <- world2spatRast(w1)
terra::plot(r1)

w2 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = 0)
w3 <- stackWorlds(w1, w2)
r3 <- world2spatRast(w3)
terra::plot(r3)
```

---

worldArray-class	<i>The worldArray class</i>
------------------	-----------------------------

---

**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

**See Also**

[worldMatrix\(\)](#)

---

worldHeight	World <i>height</i>
-------------	---------------------

---

**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**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

```
w1 <- createWorld()
worldHeight(w1)
```

---

worldMatrix-class

*The worldMatrix class*


---

**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  $((\text{maxPxcor} - \text{minPxcor}) + 1) * ((\text{maxPycor} - \text{minPycor}) + 1)$ . 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 `Raster*` objects (i.e., by rows).

**Author(s)**

Sarah Bauduin, Eliot McIntire, and Alex Chubaty

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

[worldArray\(\)](#)

---

worldNLR-class

*The worldNLR class*

---

**Description**

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

---

worldWidth

World *width*

---

**Description**

Report the width of the world in patch number.

**Usage**

```
worldWidth(world)
```

```
## S4 method for signature 'worldNLR'
worldWidth(world)
```

**Arguments**

world                      WorldMatrix or worldArray object.

**Value**

Integer.

**Author(s)**

Sarah Bauduin

**References**

Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

**See Also**

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

**Examples**

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

---

wrap

---

*Wrap coordinates or pixels in a torus-like fashion*


---

**Description**

Generally for model development purposes.

**Usage**

```
wrap(obj, bounds, withHeading)

## S4 method for signature 'ANY'
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**

```

if (requireNamespace("terra")) {
  xrange <- yrange <- c(-50, 50)
  hab <- terra::rast(terra::ext(c(xrange, yrange)))
  hab[] <- runif(terra::ncell(hab))

  # 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 <- agentMatrix(coords = starts, data = data.frame(x1 = x1, y1 = y1))

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

  if (interactive()) {
    library(quickPlot)
    clearPlot()
    Plot(hab, zero.color = "white", axes = "L")
    Plot(agent, addTo = "hab")
  }
  if (requireNamespace("SpaDES.tools") &&
      requireNamespace("CircStats")) {
    for (i in 1:10) {
      agent <- SpaDES.tools::crw(
        agent = agent,
        extent = terra::ext(hab), stepLength = ln,
        stddev = sd, lonlat = FALSE, torus = TRUE
      )
      if (interactive()) Plot(agent, addTo = "hab", axes = TRUE)
    }
  }
}

```



## Description

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

## Usage

```
## 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 = FALSE]

## 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 = FALSE]

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

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

## S4 method for signature 'agentMatrix,missing,numeric,ANY'
```

```

x[i, j, ..., drop = FALSE]

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

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

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

## S4 replacement method for signature 'agentMatrix,numeric,character,data.frame'
x[i, j] <- value

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

## S4 replacement method for signature 'agentMatrix,missing,numeric,character'
x[i, j] <- value

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

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

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

```

## Arguments

<code>x</code>	A <code>agentMatrix</code> object from which to extract element(s) or in which to replace element(s).
<code>i</code>	Indices specifying elements to extract or replace.
<code>j</code>	see <code>i</code> .
<code>...</code>	other named arguments
<code>drop</code>	not implemented
<code>value</code>	Any R object
<code>name</code>	A literal character string or a <code>name()</code> (possibly backtick quoted).

## Value

An `agentMatrix` when full row(s), full column(s) or element(s) at specific row(s) and column(s) is/are extracted.

**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.

---

```
[[,worldArray,ANY,missing-method
```

*Subsetting and replacing for worldArray class*

---

**Description**

Subsetting and replacing for worldArray class

**Usage**

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

## S4 replacement method for signature 'worldArray,ANY,missing'
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.

**Value**

The replacement method returns the original object, but with updated elements. The accessor method extracts the entire layer.

**Examples**

```
w1 <- createWorld(minPxcor = 0, maxPxcor = 9, minPycor = 0, maxPycor = 9, data = runif(100))
w2 <- createWorld(0, 9, 0, 9, data = runif(100))
w3 <- createWorld(0, 9, 0, 9, data = runif(100) + 2) # add 2 so different range
a1 <- stackWorlds(w1, w2)
a1[[2]]
a1[[2]] <- w3
```

# Index

.bboxCoords, 6  
 .identifyGroBToPlot, worldArray, .quickPlotGroB-method ([, 128  
   (numLayers.worldArray), 67  
 ==, agentMatrix, character-method, 6  
 ==, agentMatrix, numeric-method  
   (==, agentMatrix, character-method),  
   6  
 [, 128  
 [, agentMatrix, logical, missing, ANY-method  
   ([, 128  
 [, agentMatrix, missing, character, ANY-method  
   ([, 128  
 [, agentMatrix, missing, missing, missing-method  
   ([, 128  
 [, agentMatrix, missing, numeric, ANY-method  
   ([, 128  
 [, agentMatrix, numeric, character, ANY-method  
   ([, 128  
 [, agentMatrix, numeric, missing, ANY-method  
   ([, 128  
 [, agentMatrix, numeric, numeric, ANY-method  
   ([, 128  
 [, worldArray, missing, missing, ANY-method  
   ([, 128  
 [, worldArray, numeric, numeric, ANY-method  
   ([, 128  
 [, worldMatrix, missing, missing, ANY-method  
   ([, 128  
 [, worldMatrix, numeric, numeric, ANY-method  
   ([, 128  
 [<- ([, 128  
 [<-, agentMatrix, missing, character, character-method  
   ([, 128  
 [<-, agentMatrix, missing, numeric, character-method  
   ([, 128  
 [<-, agentMatrix, missing, numeric, numeric-method  
   ([, 128  
 [<-, agentMatrix, numeric, character, character-method  
   ([, 128  
 [<-, agentMatrix, numeric, character, data.frame-method  
   ([, 128  
 [<-, agentMatrix, numeric, missing, numeric-method  
   ([, 128  
 [<-, agentMatrix, numeric, numeric, character-method  
   ([, 128  
 [<-, agentMatrix, numeric, numeric, numeric-method  
   ([, 128  
 [<-, worldArray, missing, missing, matrix-method  
   ([, 128  
 [<-, worldArray, numeric, numeric, matrix-method  
   ([, 128  
 [<-, worldMatrix, missing, missing, ANY-method  
   ([, 128  
 [<-, worldMatrix, numeric, numeric, ANY-method  
   ([, 128  
 [[, worldArray, ANY, missing-method, 131  
 [[<-, worldArray, ANY, missing-method  
   ([[, worldArray, ANY, missing-method),  
   131  
 \$([, worldArray, ANY, missing-method),  
   131  
 \$, agentMatrix-method ([, 128  
 \$, worldArray-method  
   ([, worldArray, ANY, missing-method),  
   131  
 agentMatrix, 7  
 agentMatrix, matrix-method  
   (agentMatrix), 7  
 agentMatrix, missing-method  
   (agentMatrix), 7  
 agentMatrix-class, 8  
 all (NLall), 55  
 any (NLany), 56  
 base::cbind(), 13  
 bbox, 9  
 bbox(), 15, 26  
 bbox, agentMatrix-method (bbox), 9

- bbox, ANY-method (bbox), 9
- bbox, SpatExtent-method (bbox), 9
- bbox, worldNLR-method (bbox), 9
- bbox<- (bbox), 9
- bbox<-, agentMatrix, matrix-method (bbox), 9
- bk, 10
- bk, agentMatrix, numeric-method (bk), 10
- canMove, 12
- canMove, worldNLR, agentMatrix, numeric-method (canMove), 12
- cbind, 13
- cellFromPxcorPycor, 13
- cellFromPxcorPycor, worldNLR, numeric, numeric-method (cellFromPxcorPycor), 13
- clearPatches, 14
- clearPatches, worldArray-method (clearPatches), 14
- clearPatches, worldMatrix-method (clearPatches), 14
- coordinates(), 10, 26
- coordinates, agentMatrix-method, 15
- count (NLcount), 57
- createOTurtles, 16
- createOTurtles, numeric-method (createOTurtles), 16
- createTurtles, 17
- createTurtles, numeric, matrix, missing-method (createTurtles), 17
- createTurtles, numeric, missing, ANY-method (createTurtles), 17
- createWorld, 18
- createWorld, missing, missing, missing, missing, missing-method (createWorld), 18
- createWorld, numeric, numeric, numeric, numeric, ANY-method (createWorld), 18
- die, 20
- die, agentMatrix, numeric-method (die), 20
- diffuse, 21
- diffuse, worldArray, character, numeric, numeric-method (diffuse), 21
- diffuse, worldMatrix, missing, numeric, numeric-method (diffuse), 21
- dist (NLdist), 58
- downhill, 22
- downhill, worldArray, character, agentMatrix, numeric-method (downhill), 22
- downhill, worldMatrix, missing, agentMatrix, numeric-method (downhill), 22
- dx, 24
- dx, agentMatrix, missing-method (dx), 24
- dx, agentMatrix, numeric-method (dx), 24
- dy, 25
- dy, agentMatrix, missing-method (dy), 25
- dy, agentMatrix, numeric-method (dy), 25
- extent(), 10, 15
- extent, agentMatrix-method (extent, worldNLR-method), 26
- extent, worldNLR-method, 26
- face, 27
- face, agentMatrix, matrix-method (face), 27
- fd, 28
- fd, agentMatrix, numeric-method (fd), 28
- hatch, 30
- hatch, agentMatrix, numeric, numeric-method (hatch), 30
- head (show, agentMatrix-method), 96
- home, 31
- home, worldNLR, agentMatrix, character-method (home), 31
- inCone, 32
- inCone, agentMatrix, numeric, numeric, matrix-method (inCone), 32
- initialize, agentMatrix-method, 34
- inRadius, 35
- inRadius, matrix, numeric, matrix-method (inRadius), 35
- inspect, 36
- inspect, agentMatrix, numeric-method (inspect), 36
- isNLclass, 37
- isNLclass, matrix, character-method (isNLclass), 37
- layerNames, worldArray-method (numLayers.worldArray), 67
- layoutCircle, 39
- layoutCircle, worldNLR, agentMatrix, numeric-method (layoutCircle), 39
- left, 40
- left, agentMatrix, numeric-method (left), 40

- length, agentMatrix-method
  - (show, agentMatrix-method), 96
- maxNof, 41
- maxNof, agentMatrix, numeric, missing, character-method
  - (maxNof), 41
- maxNof, matrix, numeric, worldArray, character-method
  - (maxNof), 41
- maxNof, matrix, numeric, worldMatrix, missing-method
  - (maxNof), 41
- maxOneOf, 43
- maxOneOf, agentMatrix, missing, character-method
  - (maxOneOf), 43
- maxOneOf, matrix, worldArray, character-method
  - (maxOneOf), 43
- maxOneOf, matrix, worldMatrix, missing-method
  - (maxOneOf), 43
- maxPxcor, 45
- maxPxcor, worldNLR-method (maxPxcor), 45
- maxPycor, 46
- maxPycor, worldNLR-method (maxPycor), 46
- minNof, 47
- minNof, agentMatrix, numeric, missing, character-method
  - (minNof), 47
- minNof, matrix, numeric, worldArray, character-method
  - (minNof), 47
- minNof, matrix, numeric, worldMatrix, missing-method
  - (minNof), 47
- minOneOf, 48
- minOneOf, agentMatrix, missing, character-method
  - (minOneOf), 48
- minOneOf, matrix, worldArray, character-method
  - (minOneOf), 48
- minOneOf, matrix, worldMatrix, missing-method
  - (minOneOf), 48
- minPxcor, 50
- minPxcor, worldNLR-method (minPxcor), 50
- minPycor, 51
- minPycor, worldNLR-method (minPycor), 51
- moveTo, 52
- moveTo, agentMatrix, matrix-method
  - (moveTo), 52
- name(), 130
- neighbors, 53
- neighbors, worldNLR, matrix, numeric-method
  - (neighbors), 53
- NetLogoR (NetLogoR-package), 5
- NetLogoR-package, 5
- NLall, 55
- NLall, agentMatrix, missing, character-method
  - (NLall), 55
- NLall, matrix, worldArray, character-method
  - (NLall), 55
- NLall, matrix, worldMatrix, missing-method
  - (NLall), 55
- NLany, 56
- NLany, matrix-method (NLany), 56
- NLcount, 57
- NLcount, matrix-method (NLcount), 57
- NLdist, 58
- NLdist, matrix, matrix-method (NLdist), 58
- NLset, 60
- NLset, missing, agentMatrix, agentMatrix, character-method
  - (NLset), 60
- NLset, worldArray, missing, matrix, character-method
  - (NLset), 60
- NLset, worldMatrix, missing, matrix, missing-method
  - (NLset), 60
- NLwith, 62
- NLwith, agentMatrix, missing, character-method
  - (NLwith), 62
- NLwith, matrix, worldArray, character-method
  - (NLwith), 62
- NLwith, matrix, worldMatrix, missing-method
  - (NLwith), 62
- NLworldIndex, 63
- NLworldIndex, worldMatrix, numeric-method
  - (NLworldIndex), 63
- nOf, 64
- nOf, matrix, numeric-method (nOf), 64
- noPatches, 66
- noTurtles, 67
- nrow, agentMatrix-method
  - (show, agentMatrix-method), 96
- numLayers.worldArray, 67
- numLayers.worldMatrix
  - (numLayers.worldArray), 67
- of, 68
- of, missing, agentMatrix, character-method
  - (of), 68
- of, worldArray, matrix, character-method
  - (of), 68
- of, worldMatrix, matrix, missing-method
  - (of), 68
- oneOf, 70
- oneOf, matrix-method (oneOf), 70

- other, [71](#)
- other,matrix,matrix-method (other), [71](#)
- patch, [73](#)
- patch,worldNLR,numeric,numeric-method  
(patch), [73](#)
- patchAhead, [74](#)
- patchAhead,worldNLR,agentMatrix,numeric-method  
(patchAhead), [74](#)
- patchAt, [76](#)
- patchAt,worldNLR,matrix,numeric,numeric-method  
(patchAt), [76](#)
- patchDistDir, [77](#)
- patchDistDir,worldNLR,matrix,numeric,numeric-method  
(patchDistDir), [77](#)
- patches, [78](#)
- patches,worldNLR-method (patches), [78](#)
- patchHere, [79](#)
- patchHere,worldNLR,agentMatrix-method  
(patchHere), [79](#)
- patchLeft, [80](#)
- patchLeft,worldNLR,agentMatrix,numeric,numeric-method  
(patchLeft), [80](#)
- patchRight, [82](#)
- patchRight,worldNLR,agentMatrix,numeric,numeric-method  
(patchRight), [82](#)
- patchSet, [83](#)
- patchSet,matrix-method (patchSet), [83](#)
- pExist, [84](#)
- pExist,worldNLR,numeric,numeric-method  
(pExist), [84](#)
- plot.agentMatrix, [85](#)
- plot.worldArray (plot.agentMatrix), [85](#)
- plot.worldMatrix (plot.agentMatrix), [85](#)
- points.agentMatrix (plot.agentMatrix),  
[85](#)
- PxcorPycorFromCell, [86](#)
- PxcorPycorFromCell,worldNLR,numeric-method  
(PxcorPycorFromCell), [86](#)
- randomPxcor, [87](#)
- randomPxcor,worldNLR,numeric-method  
(randomPxcor), [87](#)
- randomPycor, [88](#)
- randomPycor,worldNLR,numeric-method  
(randomPycor), [88](#)
- randomXcor, [89](#)
- randomXcor,worldNLR,numeric-method  
(randomXcor), [89](#)
- randomXYcor, [90](#)
- randomXYcor,worldNLR,numeric-method  
(randomXYcor), [90](#)
- randomYcor, [91](#)
- randomYcor,worldNLR,numeric-method  
(randomYcor), [91](#)
- raster2world, [92](#)
- raster2world,ANY-method (raster2world),  
[92](#)
- rbind (cbind), [13](#)
- right, [93](#)
- right,agentMatrix,numeric-method  
(right), [93](#)
- set (NLset), [60](#)
- setXY, [94](#)
- setXY,agentMatrix,numeric,numeric,missing,ANY-method  
(setXY), [94](#)
- setXY,agentMatrix,numeric,numeric,worldNLR,logical-method  
(setXY), [94](#)
- sf2turtles, [95](#)
- sf2turtles,ANY-method (sf2turtles), [95](#)
- show,agentMatrix-method, [96](#)
- show,worldArray-method, [97](#)
- show,worldMatrix-method  
(show,worldArray-method), [97](#)
- sortOn, [97](#)
- sortOn,agentMatrix,missing,character-method  
(sortOn), [97](#)
- sortOn,matrix,worldArray,character-method  
(sortOn), [97](#)
- sortOn,matrix,worldMatrix,missing-method  
(sortOn), [97](#)
- spatRast2world, [99](#)
- spatRast2world,SpatRaster-method  
(spatRast2world), [99](#)
- spdf2turtles, [100](#)
- spdf2turtles,ANY-method (spdf2turtles),  
[100](#)
- sprout, [101](#)
- sprout,numeric,matrix-method (sprout),  
[101](#)
- stackWorlds, [103](#)
- stackWorlds,worldMatrix-method  
(stackWorlds), [103](#)
- subHeadings, [104](#)
- subHeadings,agentMatrix,agentMatrix-method  
(subHeadings), [104](#)

- subHeadings,agentMatrix,numeric-method  
(subHeadings), 104
- subHeadings,numeric,agentMatrix-method  
(subHeadings), 104
- subHeadings,numeric,numeric-method  
(subHeadings), 104
- tail (show,agentMatrix-method), 96
- tExist, 105
- tExist,agentMatrix,numeric,character-method  
(tExist), 105
- tExist,agentMatrix,numeric,missing-method  
(tExist), 105
- towards, 106
- towards,matrix,matrix-method (towards),  
106
- turtle, 108
- turtle,agentMatrix,numeric,character-method  
(turtle), 108
- turtle,agentMatrix,numeric,missing-method  
(turtle), 108
- turtles2sf, 109
- turtles2sf,ANY-method (turtles2sf), 109
- turtles2spdf, 110
- turtles2spdf,agentMatrix-method  
(turtles2spdf), 110
- turtlesAt, 111
- turtlesAt,worldNLR,agentMatrix,matrix,numeric,numeric,character-method  
(turtlesAt), 111
- turtlesAt,worldNLR,agentMatrix,matrix,numeric,numeric,missing-method  
(turtlesAt), 111
- turtleSet, 112
- turtleSet,agentMatrix-method  
(turtleSet), 112
- turtlesOn, 113
- turtlesOn,worldNLR,agentMatrix,matrix,character-method  
(turtlesOn), 113
- turtlesOn,worldNLR,agentMatrix,matrix,missing-method  
(turtlesOn), 113
- turtlesOwn, 115
- turtlesOwn,agentMatrix,character,ANY-method  
(turtlesOwn), 115
- turtlesOwn,agentMatrix,character,missing-method  
(turtlesOwn), 115
- updateList, 116
- updateList,list,list-method  
(updateList), 116
- updateList,list,NULL-method  
(updateList), 116
- updateList,NULL,list-method  
(updateList), 116
- updateList,NULL,NULL-method  
(updateList), 116
- uphill, 117
- uphill,worldArray,character,agentMatrix,numeric-method  
(uphill), 117
- uphill,worldMatrix,missing,agentMatrix,numeric-method  
(uphill), 117
- with (NLwith), 62
- withMax, 119
- withMax,agentMatrix,missing,character-method  
(withMax), 119
- withMax,matrix,worldArray,character-method  
(withMax), 119
- withMax,matrix,worldMatrix,missing-method  
(withMax), 119
- withMin, 120
- withMin,agentMatrix,missing,character-method  
(withMin), 120
- withMin,matrix,worldArray,character-method  
(withMin), 120
- withMin,matrix,worldMatrix,missing-method  
(withMin), 120
- world2raster, 122
- world2raster,numeric,character-method  
world2raster,worldArray-method  
(world2raster), 122
- world2raster,numeric,missing-method  
world2raster,worldMatrix-method  
(world2raster), 122
- world2spatRast, 123
- world2spatRast,worldArray-method  
(world2spatRast), 123
- world2spatRast,worldMatrix-method  
(world2spatRast), 123
- worldArray (worldArray-class), 124
- worldArray(), 126
- worldArray-class, 124
- worldHeight, 124
- worldHeight,worldNLR-method  
(worldHeight), 124
- worldMatrix (worldMatrix-class), 125
- worldMatrix(), 124
- worldMatrix-class, 125
- worldNLR (worldNLR-class), 126
- worldNLR-class, 126
- worldWidth, 126



worldWidth,worldNLR-method  
    (worldWidth), [126](#)  
wrap, [127](#)  
wrap,ANY-method (wrap), [127](#)