Package ‘DGVM3D’

March 19, 2018

Type Package
Title 3D Forest Simulation Visualization Tool
Version 1.0.0
Date 2018-03-16
URL https://github.com/joergsteinkamp/DGVM3D
BugReports https://github.com/joergsteinkamp/DGVM3D/issues
Description This is a visualization tool for vegetation structure/succession in space and/or time mainly for forest gap models. However, it could also be used to visualize observed forest stands. If used for models, they should contain either individual trees or cohorts (e.g. LPJ-GUESS by Smith et al. (2014) <doi:10.5194/bg-11-2027-2014>). For a list of required and additional data fields see the vignette.
License GPL-3 | file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Collate 'DGVM3D.R' 'classes.R' 'flame.R' 'geometry.R' 'input.R' 'misc.R' 'stand.R' 'vegetation.R' 'visualize.R'
Depends R (>= 2.10), rgl (>= 0.96.0), methods, stats, grDevices,
Suggests RColorBrewer, knitr, rmarkdown, data.table, ggplot2
VignetteBuilder knitr
NeedsCompilation no
Author Steinkamp Joerg [aut, cre]
Maintainer Steinkamp Joerg <steinkamp.joerg@gmail.com>
Repository CRAN
Date/Publication 2018-03-19 14:47:13 UTC
R topics documented:

DGVM3D-package ........................................... 2
dgvm3d.options ........................................... 3
dgvm3d.succession ........................................ 4
establishTrees ........................................... 4
fire3D ....................................................... 5
getCone ..................................................... 6
getEllipsoid .............................................. 7
getFlame .................................................... 7
getHexagon .................................................. 8
grass3D ..................................................... 9
initStand ................................................... 10
Patch-class .................................................. 11
plant3D ..................................................... 11
random.disc .............................................. 12
read.LPJ ................................................... 13
row.disc .................................................... 14
snapshot .................................................... 14
Stand-class .................................................. 15
stand3D ..................................................... 15
succession ................................................... 16
sunflower.disc ............................................ 17
sunflower.radius ......................................... 17
tree3D ...................................................... 18
TriangBody-class .......................................... 18
triClose ..................................................... 19
updateStand ............................................... 20

Index ................................................. 21

DGVM3D-package 3D Forest Simulation Visualization Tool

Description

This is a visualization tool for vegetation structure/succession in space and/or time mainly for forest gap models. However, it could also be used to visualize observed forest stands. If used for models, they should contain either individual trees or cohorts (e.g. LPJ-GUESS by Smith et al. (2014) <doi:10.5194/bg-11-2027-2014>). For a list of required and additional data fields see the vignette.

Author(s)

Maintainer: Steinkamp Joerg <steinkamp.joerg@gmail.com>
References


See Also

Useful links:
- https://github.com/joergsteinkamp/DGVM3D
- Report bugs at https://github.com/joergsteinkamp/DGVM3D/issues

dgvm3d.options

set some variables used in cascading functions

Description

set some variables used in cascading functions

Usage

dgvm3d.options(x = NULL, patch.area = NULL, samples = NULL, overlap = NULL, sort.column = NULL, establish.method = NULL, color.column = NULL, verbose = NULL)

Arguments

x query character 'x' for its value.
patch.area the patch area in m^2.
samples 2 element vector. 1. number of samples to determine the next trees position. 2. max. number to repeat the sampling
overlap fraction of crownradius allowed to overlap.
sort.column 2 element vector: 1. vegetation data.frame column name to sort by. 2. "descending" (default) or "ascending".
establish.method where to place the trees: 'random', 'sunflower' or 'row'. If there are trees with positions already 'random' is applied.
color.column name of the vegetation column to create the canopy colors from.
verbose print some information.
dgvm3d.succession  

*Vegetation timeseries data from 1860-2005*

---

**Description**

A list of 3 data.frames with simulation results of a LPJ-GUESS model run without random patch disturbance at 3 locations. All patches at all locations were disturbed in 1859.

**Usage**

dgvm3d.succession

**Format**

An object of class list of length 3.

**Details**

@name dgvm3d.succession @docType data @keywords data

---

**establishTrees**  

*poplulate a patch with its vegetation*

---

**Description**

Randomly 'plant' the trees in the patch within a given radius.

**Usage**

establishTrees(vegetation = NULL, radius = 1, jitter = FALSE, ...)

**Arguments**

- **vegetation**    the vegetation data.frame
- **radius**        the radius used to distribute the vegetation to
- **jitter**        add a small amount of noise to the positions. Applies only for dgvm3d.options("establish.method") = "row" or "sunflower" (default: FALSE).
- ...               additional parameters passed to jitter.

**Value**

the vegetation data.frame with the positions
Examples

```r
## Not run:
dgvm3d.options("default")
stand = initStand(npatch=1)
veg = data.frame(DBH=rep(0.5, 100))
veg$Height = veg$DBH * 35
veg$Crownarea = veg$DBH * 10
veg$LeafType = sample(1:2, nrow(veg), replace=TRUE)
veg$ShadeType = sample(1:2, nrow(veg), replace=TRUE)
stand@patches[[1]]@vegetation = establishTrees(veg, stand@hexagon@supp[['inner.radius']])
stand3D(stand)
dummy = plant3D(stand)
rot.z = rotationMatrix(0, 0, 0, 1)
rot.y = rotationMatrix(0, 1, 0, 0)
rgl.viewpoint(userMatrix = rot.y %% rot.z, fov=1)

rgl.clear()
dgvm3d.options(establish.method = "sunflower")
stand@patches[[1]]@vegetation = establishTrees(veg, stand@hexagon@supp[['inner.radius']])
stand3D(stand)
dummy = plant3D(stand)

rgl.clear()
dgvm3d.options(establish.method = "row")
stand@patches[[1]]@vegetation = establishTrees(veg, stand@hexagon@supp[['inner.radius']],
jitter=TRUE, amount=0.01)
stand3D(stand)
dummy = plant3D(stand)

## End(Not run)
```

fire3D

---

**add Fire to the stand or succession**

Description

add Fire to the stand or succession

Usage

```r
fire3D(stand = NULL, patch.id = NULL, limit = 0.5)
```

Arguments

- `stand`: the stand object
- `patch.id`: the ID of a patch if NULL all are used
- `limit`: define a lower bound below which no fire should be plotted
Examples

## Not run:
stand=snapshot(dgym3d.succession[[8]], patch.id=4, year=1905)
rgl.clear("lights")
rgl.light( theta = -25, phi = 30, specular = "#AAAAA")
fire3D(stand)

## End(Not run)

---

**getCone**

_calculate a cone_

**Description**

calculate a cone

**Usage**

getcone(radius = 0.5, height = 1, faces = 72, close = FALSE)

**Arguments**

- **radius**: the outer radius of the cone
- **height**: the height of the cone
- **faces**: number of triangular sides
- **close**: logical should the bottom side be closed.

**Value**

-a TriangBody-class

**Examples**

if (require(rgl)) {
  cone=getcone(faces=13, close=TRUE)
  triangles3d(cone@vertices[cone@id, ], col="green")
} else {
  message("the library 'rgl' is required for this example!"
}
getEllipsoid  

Calculate an ellipsoid

Description

Calculate an ellipsoid

Usage

getEllipsoid(radius = 0.5, height = 1, faces = c(6, 3))

Arguments

radius  
  x/y radius

height  
  z height

faces  
  approx. number of faces. If two values given: 1.) around z-axis; 2.) along z-axis.

Value

a TriangBody-class

Examples

if (require(rgl)) {
  ellipsoid=getEllipsoid(height=2)
  triangles3d(ellipsoid@vertices[ellipsoid@id, ], col="green")
} else {
  message("the library 'rgl' is required for this example!")
}

getFlame  

Get the shape of a single flame

Description

Get the shape of a single flame

Usage

getFlame(faces = 10, radius = 0.3, dz = 1, z.exp = 1.1, expand = 1, turn = 0)
getHexagon

Calculate a 3D hexagon

Description

Calculate a 3D hexagon

Usage

getHexagon(area = NA, outer.radius = NA, inner.radius = NA, z = c(0, 1))

Arguments

area the area of the hexagon
outer.radius the outer radius of the hexagon
inner.radius the inner radius of the hexagon
z the height of the hexagon as 2 element vector

Examples

## Not run:
center = getFlame(dz=0.8)
triangles3d(center$id[, (2 * 20 + 1):150],
   col=#6fffff, alpha=1, shininess=1, lit=FALSE)
inner = getFlame(dz=0.97, expand=2)
triangles3d(inner$id[, (2 * 20 + 1):175],
   col=#0f0f0f, alpha=0.6, shininess=1, lit=FALSE)
outer = getFlame(dz=1, expand=3)
triangles3d(outer$id[, (2*20+1):200],
   col=#c1301, alpha=0.3, shininess=10, lit=FALSE)

## End(Not run)
Value

a `TriangBody-class`

Examples

```r
if (require(rgl)) {
    hexagon <- getHexagon(area=dgvm3d.options("patch.area"), z=c(0, -2))
    triangles3d(hexagon@vertices[hexagon@id, ], col="brown")
} else {
    message("the library 'rgl' is required for this example!"
}
```

Description

Plant the grass on the patch

Usage

```r
grass3D(grass = NULL, kind = NULL, offset = c(0, 0, 0), col = "green",
    opacity.threshold = c(0.2, 2), height.scale = 0.1)
```

Arguments

- `grass`: vegetation data.frame
- `kind`: so far only a hexagon is allowed (TriangBody)
- `offset`: the patch offset
- `col`: the color to use for grass
- `opacity.threshold`: no grass is drawn below the lower values of LAI and full opacity is used above the upper value
- `height.scale`: scale the LAI by this factor as height for the hexagon.
initStand

*Initialize the model Stand*

**Description**

Initialize the model Stand

**Usage**

```r
initStand(npatch = 1, year = 2000, soil = c(0, -0.5, -1.5), z = 0,
        layout = "square", composition = "spatial", dist = 0.05)
```

**Arguments**

- `npatch`: number of patches
- `year`: the initialization year
- `soil`: a vector or matrix of soil depths.
- `z`: the height of each patch.
- `layout`: patch layout ('square' or 'linear'), a two element vector with number of rows/columns. A matrix for layout (not yet ready).
- `composition`: 'spatial' or 'temporal'
- `dist`: the fractional distance between the hexagons

**Details**

If soil is a matrix, the number of columns must be equal to npatch. In that way each patch can have its own soil depth. The patches represented as hexagons can either be arranged in a square or in a line. The later one for example to represent a time series (succession).

**Value**

a *Stand-class*

**Examples**

```r
## Not run:
stand &lt;- initStand(npatch=9, z=sort(rnorm(9, sd=2)))
stand3D(stand)

stand &lt;- initStand(npatch=9, z=sort(rnorm(9, sd=2)), layout='linear')
stand3D(stand)

## End(Not run)
```
**Patch-class**

*One model patch*

---

**Description**

This defines the basic class

**Slots**

- `id` unique ID
- `pid` the patch id in the vegetation data.frame
- `soil` vector of soil layer depth
- `vegetation` the vegetation data.frame
- `color.table` lookup table for coloring

---

**plant3D**

*Plant the trees of an already created patch/stand*

---

**Description**

Plant the trees of an already created patch/stand

**Usage**

```r
plant3D(stand = NULL, patch.id = NULL, crown.opacity = 1)
```

**Arguments**

- `stand` the stand for plantation
- `patch.id` one or several specific patches only
- `crown.opacity` alpha value for the green tree crowns. Setting it to something different than 1 slows down the rendering substantially!

**Value**

the updated stand
## Examples

```r
## Not run:
stand = initStand(npatch=2)
stand3D(stand, 1)
veg = data.frame(DBH=rep(0.4, 50))
veg$Height = veg$DBH * 35
veg$Crownarea = veg$DBH * 5
veg$LeafType = sample(1:2, nrow(veg), replace=TRUE)
veg$ShadeType = sample(1:2, nrow(veg), replace=TRUE)
stand@patches[[1]]@vegetation = establishTrees(veg, stand@hexagon@supp[['inner.radius']])
dummy = plant3D(stand, 1)

stand3D(stand, 2)
veg = data.frame(DBH=rep(0.5, 100) * rgamma(100, 2.5, 9))
veg$Height = veg$DBH * 35 * rbeta(nrow(veg),10,1)
veg$Crownarea = veg$DBH * 5 * rnorm(nrow(veg), 1, 0.1)
veg$LeafType = sample(1:2, nrow(veg), replace=TRUE)
veg$ShadeType = sample(1:2, nrow(veg), replace=TRUE)
stand@patches[[2]]@vegetation = establishTrees(veg, stand@hexagon@supp[['inner.radius']])
dummy = plant3D(stand, 2)
```

## End(Not run)

---

## random.disc

### Random distribution in a circle

#### Description

Random distribution in a circle

#### Usage

`random.disc(n, strict = FALSE)`

#### Arguments

- `n` total number of trees
- `strict` should the value 2\pi be excluded

#### Value

data.frame of positions
Prepare the output table from LPJ-GUESS for visualization

**Description**

Stand ID and Patch ID start counting at 0 in the standard output. Here the value of 1 is added, to be consistent with R.

**Usage**

```r
read.LPJ(file = "vegstruct.out", stand.id = 1, patch.id = NULL,
year = NULL, lon = NULL, lat = NULL, grass = TRUE)
```

**Arguments**

- `file`: the filename to be read
- `stand.id`: the stand ID default to 0.
- `patch.id`: if a single patch should be used (default all)
- `year`: if a single year should be used (default all)
- `lon`: if a single longitude should be used (default all). Should be defined, if more than one gridpoint is in the output.
- `lat`: as above
- `grass`: should grasses be included (so far they are not yet further processed).

**Value**

individual vegetation data.frame with equal individuals from each cohort.

**Examples**

```r
## Not run:
dgvm3d.locations = read.table("gridlist.txt",
col.names=c("Lon", "Lat", "Name"), sep="\t",
stringsAsFactors=FALSE)
dgvm3d.succession=list()
for (i in 1:nrow(dgvm3d.locations)) {
  dgvm3d.succession[[dgvm3d.locations$Name[i]]] =
  read.LPJ("vegstruct.out",
  lon=dgvm3d.locations$Lon[i],
  lat=dgvm3d.locations$Lat[i])
  dgvm3d.succession[i] = dgvm3d.succession[i][!(dgvm3d.succession[i]$Year %% 5) &
  dgvm3d.succession[i]$Year > 1859, ]
}

## End(Not run)
```
**row.disc**

*row-wise distribution of points in a disc*

**Description**

row-wise distribution of points in a disc

**Usage**

```r
row.disc(n)
```

**Arguments**

- `n` number of points

**Value**

data.frame of x and y positions

**Examples**

```r
par(mfrow=c(2,2), mai=c(0, 0, 0.5, 0))
for (n in sample(500, 4)) {
  ret=row.disc(n)
  plot(cos(seq(0, 2 * pi, length.out=7)) * 1.154701,
       sin(seq(0, 2 * pi, length.out=7)) * 1.154701,
       type="l", axes = FALSE, ylab = "", xlab="", main=n)
  points(ret)
}
```

**snapshot**

*Visualize a snapshot of patches.*

**Description**

Visualize a snapshot of patches.

**Usage**

```r
snapshot(vegetation, stand.id = 1, patch.id = NULL, year = 2000)
```

**Arguments**

- `vegetation` the data.frame of individual trees
- `stand.id` the stand to take a snapshot off.
- `patch.id` all patches (default) or just one.
- `year` which year to take the snapshot off.
Stand-class

Value

a Stand-class.

Examples

```r
## Not run:
stand=snapshot(dgvm3d.succession[1])
```

## End(Not run)

---

Stand-class

One model stand consisting of several patches

Description

One model stand consisting of several patches

Slots

- `patches` list of patches in one stand
- `area` the area of each patch
- `year` the year of the current patch vegetation
- `hexagon` a TriangBody-class Hexagon definition used for all patches
- `layout` either 'linear' or 'square'
- `composition` either 'spatial' or 'temporal'. Has no effect yet.
- `patch.pos` the position of the patch hexagon centers

---

stand3D

3D view of the stands

Description

Uses rgl to visualize a single, if patch.id is given, or all patch soil hexagons

Usage

```r
stand3D(stand, patch.id = NULL)
```

Arguments

- `stand` the Stand-class to visualize
- `patch.id` the patch IDs to create. Default: all.
succession

Value
None

See Also
initStand for examples

---

**succession**

create a temporal succession

**Description**
create a temporal succession

**Usage**

```r
succession(vegetation, stand.id = 1, patch.id = NULL, init.year = 1901,
years = seq(1950, 2000, 10))
```

**Arguments**

- `vegetation` data.frame
- `stand.id` the Stand ID
- `patch.id` the patch ID, if NULL all available ones are considered
- `init.year` year, when to initialize the tree positions
- `years` the years to be included

**Value**
a stand object

**Examples**

```r
## Not run:
stand=succesion(dgvm3d.succession[[3]], init.year=1865, years=c(1865, seq(1875, 2000, 25)),
               patch.id=sample(1:12, 3))
stand3D(stand)
stand=plant3D(stand)
```

## End(Not run)
**sunflower.disc**

return the positions

**Description**

return the positions

**Usage**

sunflower.disc(n, alpha = 0)

**Arguments**

- **n**: total number of trees
- **alpha**: smoothing factor for boundary points

**Value**

position data.frame

---

**sunflower.radius**

Calc the current radius

**Description**

Calc the current radius

**Usage**

sunflower.radius(k, n, b)

**Arguments**

- **k**: current value
- **n**: total number
- **b**: number of boundary points

**Value**

radius
**TriangBody-class**

---

**tree3D**

*draw a single tree*

---

**Description**

draw a single tree

**Usage**

```r
tree3D(tree = NULL, offset = c(0, 0, 0), col = c("#22BB22", "33FF33"), opacity = 1, faces = 19)
```

**Arguments**

- `tree`: one column of the `Patch-class` vegetation data.frame slot
- `offset`: x/y center and surface (z) of the respective patch
- `col`: crown colors for the shade classes
- `opacity`: alpha value for the tree crown (heavy impacting performance)
- `faces`: number of faces/triangles used per stem and tree cone 3-times for ellipsoid.

---

**TriangBody-class**

*a Information to draw a triangular object*

---

**Description**

*a Information to draw a triangular object*

**Slots**

- `vertices`: the vertices of the object
- `id`: the column indices of the vertex matrix to draw the triangular body
- `supp`: supplementary information
Description

Method 'circular' (default) used the most triangles so far by going round in the circle and connecting the next three vertices. 'fix' uses vertex id 1 and creates triangles to all other points round. 'planar' always flips the triangles.

Usage

triClose(n, method = "circular", center = NA)

Arguments

n       number of vertices.
method  Method how to organize the triangles 'circular', 'planar', 'fix' and 'center'.
center  The center vertex ID for the central point (method 'center' only; default NA).

Value

A vector of indices for the polygon vertices.

Examples

par(mfrow=c(2,2))
for (m in c("plan", "fix", "center", "")) {
  faces <- sample(12:20, 1)
  vertices <- sapply(seq(0, 2*pi*(faces-1)/faces, length.out=faces),
                      function(x) c(sin(x), cos(x)))
  tri = triClose(faces, method=m)
  if (m == "center") {
    tri[is.na(tri)] = faces + 1
    vertices = cbind(vertices, c(mean(vertices[1,]), mean(vertices[2,])))
  }
  plot(vertices[1,1:faces], vertices[2,1:faces], type="b")
  text(x=1.05*vertices[1,], y=1.05*vertices[2,], labels=1:faces, adj=0.5)
  for (i in seq(1, length(tri)), 3))
    polygon(vertices[1,tri[i:(i+2)]], vertices[2,tri[i:(i+2)]],
            col=rgb(runif(1), runif(1), runif(1)))
}
par(mfrow=c(2,2))
for (faces in c(6, 12, 13, 25)) {
  vertices <- sapply(seq(0, 2*pi*(faces-1)/faces, length.out=faces),
                      function(x) c(sin(x), cos(x)))
  tri = triClose(faces, method=m)
  plot(vertices[1,], vertices[2,], type="b")
  text(x=1.05*vertices[1,], y=1.05*vertices[2,], labels=1:faces, adj=0.5)
for (i in seq(1, length(tri), 3))
  polygon(vertices[1,tri[i:(i+2)]], vertices[2,tri[i:(i+2)]],
  col=rgb(runif(1), runif(1), runif(1)))
}

updateStand

Description

Removes those individuals with the shortest distance to any neighbour and adds new individuals randomly.

Usage

updateStand(stand, vegetation, year = NULL)

Arguments

- **stand**: stand to update
- **vegetation**: new vegetation data.frame
- **year**: the next year

Value

- **stand**
Index

*Topic **datasets**
  dgvm3d.succession, 4

DGVM3D (DGVM3D-package), 2
DGVM3D-package, 2
dgvm3d.options, 3
dgvm3d.succession, 4

establishTrees, 4

fire3D, 5

ggetCone, 6
ggetEllipsoid, 7
ggetFlame, 7
ggetHexagon, 8
grass3D, 9

initStand, 10, 16

Patch-class, 11
plant3D, 11

random.dic, 12
read.LPJ, 13
rgl, 15
row.dic, 14

snapshot, 14
Stand-class, 15
stand3D, 15
succession, 16
sunflower.dic, 17
sunflower.radius, 17

tree3D, 18
TriangBody-class, 18
triClose, 19

updateStand, 20