

Package: ppmData (via r-universe)

October 19, 2026

Type Package

Title Quasirandom quardature scheme using dirichlet tessellation for spatial Poisson point process models

Version 1.0.0

Maintainer Skipton Woolley <skip.woolley@csiro.au>

Description This package generates a set of quasi-random points for a quadrature scheme, we calculate each points weights using a dirichlet tessellation (voronoi) algorithm. The resulting object can be used in existing single and multiple species models in order to extend them to Point Processes.

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 7.2.3

URL <https://github.com/skiptoniam/ppmData>

BugReports <https://github.com/skiptoniam/ppmData/issues>

Imports methods, mgcv, parallel, sf, randtoolbox, Rcpp, spatstat.geom

Suggests spatstat, knitr, testthat (>= 3.0.0), rmarkdown

Depends R (>= 4.0.0), terra (>= 1.7.29)

Config/testthat/edition 3

SystemRequirements C++11

NeedsCompilation yes

VignetteBuilder knitr

LinkingTo Rcpp

Encoding UTF-8

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository <https://fdzul.r-universe.dev>

Date/Publication 2023-05-08 03:36:40 UTC

RemoteUrl <https://github.com/skiptoniam/ppmData>

RemoteRef HEAD
RemoteSha 650ec854cbb0a6d347e66a703810151c144933aa

Contents

as.owin.ppmData	2
as.ppp	3
as.spatstat	3
dirTess	4
gridQuad	5
gridWeights	5
plot.dirTess	6
plot.ppmData	7
polygonise	8
ppmData	9
print.ppmData	11
pseudoRandomQuad	12
pseudoRandomWeights	13
quasiSampFromHyperRect	14
rast_to_sf	14
raster_datasets	15
snails	16
terra2im	16
Index	18

as.owin.ppmData	<i>Convert a ppmData object to an owin from the SpatStat package.</i>
-----------------	---

Description

Convert a ppmData object to an owin from the SpatStat package.

Usage

```
## S3 method for class 'ppmData'  
as.owin(W, ..., fatal = TRUE)
```

Arguments

W	Window A ppmData object. Uses the saved window in the ppmData object.
...	Unused.
fatal	Unused.

as.ppp

*Function to covert a spatstat quadrature***Description**

Function to covert a spatstat quadrature

Usage

```
## S3 method for class 'ppmData'
as.ppp(X, ..., fatal = TRUE)
```

Arguments

X	A ppmData object
fatal	Logical value specifying what to do if the data cannot be converted. See Details.
...	Ignored

Examples

```
## Not run:
library(ppmData)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
preds <- rast(lst)
window <- preds[[1]]
presences <- subset(snails,SpeciesID %in% "Tasmaphena sinclairi")
object <- ppmData(npoints = 10000, presences=presences, window = window, covariates = preds)
X <- as.ppp(object)

## End(Not run)
```

as.spatstat

*Function to covert a spatstat quadrature and spatstat data (covariates)***Description**

This function will convert the ppmData to a spatstat object that can be used in the spatstat library.
This currently only works for unmarked point process

Usage

```
as.spatstat(object, ...)
```

Arguments

object	A ppmData object
...	Ignored

Examples

```
## Not run:
library(ppmData)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
preds <- rast(lst)
window <- preds[[1]]
presences <- subset(snails,SpeciesID %in% "Tasmaphena sinclairi")
object <- ppmData(npoints = 10000, presences=presences, window = window, covariates = preds)
ss.object <- as.spatstat(object)

## now you should be able to fit a ppm from spatstat
ss.form <- ss.object$Q~1+poly(min_temp_coldest_month,2)+poly(annual_mean_precip,2)
ppmfit <- ppm(ss.form,data=ss.object$covariates,Poisson())
pred <- predict(ppmfit, covariates = ss.object$covariates,type="trend")

## End(Not run)
```

dirTess

dirTess

Description

Computes the Dirichlet tessellation using the dual-graph of the Delaunay triangulation. The Delaunay triangulation is calculated internally for this function using a sweep algorithm.

Usage

```
dirTess(coords, bbox = NULL)
```

Arguments

coords	The coordinates of the points to triangulate.
bbox	Optional - doesn't actually do anything anymore :)

Examples

```
coords <- matrix(runif(2000),ncol=2)
tess <- dirTess(coords)
```

gridQuad

*gridQuad***Description**

A relatively quick way to generate psuedo-random quadrature set for ppm models.

Usage

```
gridQuad(npoints, window, coord = c("X", "Y"), control)
```

Arguments

npoints	The number of background points to use. The default is 1000, but it is generally recommended to use more when fitting a ppm.
window	A SpatRaster from terra package which will represent the extent and resolution of the point process model.
coord	The names of the coordinates. Default is c("X","Y").
control	A list of control options passed from ppmData

Author(s)

Skipton Woolley

Examples

```
library(ppmData)
library(terra)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
window <- terra::rast(lst[1])
res <- gridQuad(10000,window,control=list(quiet=FALSE))
plot(window)
points(res[,1:2],pch=".")
```

gridWeights

*gridWeights***Description**

Quick way to generate psuedo-random points using the terra package. Weights are taken as the area(window)/npoints. The default unit is km², but other units can be used such as meters squared "m" or hectars "ha". There is an internal check to see if the input data is in lon/lat and if so, it returns the area of the cell in which the point lies.

Usage

```
gridWeights(
  presences,
  quadrature,
  window,
  coord = c("X", "Y"),
  mark.id,
  unit = c("geo", "m", "km", "ha")
)
```

Arguments

presences	The observed presence locations
quadrature	The quadrature locations
window	A SpatRaster from terra package which will represent the extent and resolution of the point process model.
coord	The names of the coordinates. Default is c("X","Y").
mark.id	character Name of the species ID column
unit	The scale of the area weights, default is kilometers squared "km" but meters squared "m" or hectares "ha" can be used.

Author(s)

Skipton Woolley

Examples

```
library(ppmData)
library(terra)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
window <- terra::rast(lst[1])
presences <- subset(snails,SpeciesID %in% "Tasmaphena sinclairi")[,1:2]
quadrature <- gridQuad(10000,window,control=list(quiet=FALSE))
res <- gridWeights(presences,quadrature,window)
plot(window)
points(res[res$presence==1,1:2],pch=16,cex=0.5,col='blue')
points(res[res$presence==0,1:2],pch='.')

```

plot.dirTess

plot a Dirichlet tessellation

Description

plot a Dirichlet tessellation

Usage

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

Arguments

x	dirTess object
...	Additional plotting arguments

plot.ppmData	<i>Plot a ppmData object</i>
--------------	------------------------------

Description

Plot a ppmData object

Usage

```
## S3 method for class 'ppmData'
plot(
  x,
  main = "Quadrature Scheme",
  rcols = gray(0.5, 0.75),
  pch.cols = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#E6AB02", "#A6761D",
    "#666666"),
  pchs = 15:20,
  plt.legend = TRUE,
  ...
)
```

Arguments

x	A ppmData object.
main	Title of the plot, default is 'Quadrature Scheme'
rcols	Vector of colours or colour codes for raster
pch.cols	Vector of point shapes
pchs	Vector of pch codes
plt.legend	boolean plot the legend or not, default is TRUE.
...	Ignored

`polygonise`

Convert the Dirichlet tessellation to polygons and clip if so desired.

Description

Convert the Dirichlet tessellation to polygons and clip if so desired.

Usage

```
polygonise(
  x,
  window = NULL,
  clippy = TRUE,
  crs = sf::st_crs("EPSG:4326"),
  unit,
  ...
)
```

Arguments

<code>x</code>	Dirichlet tessellation object
<code>window</code>	polygon A polygon to clip the dirichlet tessellation - intersection between boundary polygons and polygon
<code>clippy</code>	Logical Clip to the bounding box or polygon if <code>polyclip = TRUE</code> .
<code>crs</code>	CRS Projection of coordinates default is "EPSG:4326" (lon/lat wgs84)
<code>unit</code>	Character The type of area to return. The default is "geo" and returns the area based on the euclidean distance between geographic coordinates. This will default to the values of the raster and presence coordinate system. Alternatively, meters squared "m", kilometers squared "km", or hectares "ha" can be used.
<code>...</code>	Additional arguments for a <code>polygonise</code> function

Examples

```
library(sf)
coords <- matrix(runif(2000), ncol=2)
tess <- dirTess(coords)
tess.polys <- polygonise(tess)
plot(st_geometry(tess.polys$polygons), col=hcl.colors(100))
points(coords,col='tomato',pch=16, cex=.3)
```

ppmData	<i>Develop a quadrature scheme using quasi-random sampling for spatial point processes.</i>
---------	---

Description

ppmData is a package for setting up quadrature to implement spatial Poisson Point process models and extensions. The approach uses quasi-random sampling (Grafston & Tille, 2013, Foster et al., 2018) to generate a quadrature scheme for numerical approximation of a Poisson point process model (Berman & Turner 1992; Warton & Shepard 2010). Quasi-random sampling quadrature are form of spatially-balanced survey design or point stratification that aims to reduce the frequency of placing samples close to each other (relative to pseudo-random or grid designs). A quasi-random quadrature design improves efficiency of background point sampling (and subsequent modelling) by reducing the amount of spatial auto-correlation between data implying that each sample is providing as much unique information as possible (Grafston & Tille, 2013, Foster et al., 2018) and thus reducing low errors for geostatistical prediction (Diggle & Ribeiro, 2007). Because the quasi-random design is not on a regular grid we use Dirichlet tessellation to generate polygons for each point in the quadrature scheme. Areal weights are then derived from these polygons.

Usage

```
ppmData(
  presences,
  window = NULL,
  covariates = NULL,
  npoints = NULL,
  coord = c("X", "Y"),
  mark.id = "SpeciesID",
  quad.method = c("quasi.random", "pseudo.random", "grid"),
  interpolation = c("simple", "bilinear"),
  unit = c("geo", "m", "km", "ha"),
  na.rm = FALSE,
  control = list()
)
```

Arguments

presences	a three column matrix or data.frame object giving the coordinates of each species' presence in (should be a matrix of nsites * 3) with the three columns being c("X","Y","SpeciesID"), where X is longitude, Y is latitude and SpeciesID is a factor which associated each occurrence to a species.
window	SpatRaster a raster object giving the region where to generate the quadrature scheme. Windows with NA cells are ignored and masked out of returned data. If NULL, a rectangle bounding the extent of presences will be used as the default window.

covariates	SpatRaster A terra raster object containing covariates for modelling the point process. These layers should match the resolution and extent of the window provided. If NULL, only the coordinates of the presences and quadrature points are returned for the ppmData object.
npoints	Integer The number of quadrature points to generate. If NULL, the number of quadrature points is calculated based on linear scaling. In reality, the number of quadrature points needed to approximate the log-likelihood function will depend on the data and likelihood function being approximated. Typically, the more quadrature the better the estimate, but there is a trade off between computational efficiency and accuracy. See Warton & Shepard (2010) or Renner et al., 2015 for useful discussions on the location and number of quadrature points required to converge a ppm likelihood.
coord	Character These are the users name of site coordinates. The default is c('X','Y'). This should match the name of the coordinates in the presences data set.
mark.id	Character This is the column name of the mark ID. The default is "SpeciesID". But this should be changed to match the user's data. If this column contains multiple "species" then a marked quadrature scheme will be created.
quad.method	Character The quadrature generation method. Default is "quasi.random" for quasi-random, "pseudo.random" for pseudo-random (regular random) and "grid" for a regular grid at a set resolution (with respect to the original window resolution).
interpolation	Character The interpolation method to use when extracting covariate data at a presence or quadrature location. Default is "bilinear", can also use "simple", this is based on the terra package extract .
unit	Character The type of area to return. The default is "geo" and returns the area based on the euclidean distance between geographic coordinates. This will default to the values of the raster and presence coordinate system. Alternatively, meters squared "m", kilometers squared "km", or hectares "ha" can be used.
na.rm	Boolean Remove NA data from covariates. Only works for single species models.
control	list A list of control parameters for using ppmData. See details for uses of control parameters.

Details

The approach uses quasi-random sampling to generate a quadrature scheme based (e.g Berman & Turner 1992; Warton & Shepard 2010; Foster et al, 2017). The weights each quasi-random point in the quadrature scheme is calculated using a Dirichlet tessellation (Turner 2020). To improve computational efficiency we have rewritten the Delaunay triangulation and Dirichlet tessellation in c++ using a sweep algorithm. The control has a bunch of parameters you can use to tweak the ppmData object.

- `quasirandom.samples` integer This sets the total number of samples to consider in the BAS step (rejection sampling). The default is set to NULL and the function internally generates 10 times the total number of quadrature points needed. This means if 10000 quadrature points are required for ppmData, then a halton sequence of 100000 quasi-random numbers are drawn and then thinned according to the inclusion probabilities. The more `quasirandom.samples` selected the slower the quasi-random quadrature scheme will be to generate.

- `buffer.NA` boolean If extract from `extract` returns NA for point extract, do you want us to attempt to use buffer to calculate cells which are NA.
- `buffer.size` numeric If you call 'buffer.NA' what is the range of the buffer in meters.
- `mc.cores` integer The number of cores to use in the processing. default is `parallel::detectCores()-1`
- `quiet` boolean If TRUE, do not print messages. Default is FALSE.

Author(s)

Skip Woolley <skip.woolley@csiro.au> & Scott Foster <scott.foster@data61.csiro.au>

References

Diggle, P. J., P. J. Ribeiro, Model-based Geostatistics. Springer Series in Statistics. Springer, 2007.

Foster, S.D., Monk, J., Lawrence, E., Hayes, K.R., Hosack, G.R. and Przeslawski, R., 2018. Statistical considerations for monitoring and sampling. Field manuals for marine sampling to monitor Australian waters, pp.23-41.

Grafstrom, Anton, and Yves Tille. Doubly balanced spatial sampling with spreading and restitution of auxiliary totals. *Environmetrics* 24.2 (2013): 120-131.

Warton, D. I., and L. C. Shepherd. Poisson point process models solve the pseudo-absence problem for presence-only data #'in ecology. *The Annals of Applied Statistics* 4.3 (2010): 1383-1402.

Examples

```
## Not run:
library(ppmData)
library(terra)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
preds <- rast(lst)
window <- preds[[1]]
presences <- subset(snails,SpeciesID %in% "Tasmaphena sinclairi")
quad <- ppmData(npoints = 1000, presences=presences, window = window, covariates = preds)

## End(Not run)
```

<code>print.ppmData</code>	<i>Print a summary of ppmData object.</i>
----------------------------	---

Description

Print a summary of ppmData object.

Usage

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

Arguments

<code>x</code>	A ppmData object.
<code>...</code>	Ignored

<code>pseudoRandomQuad</code>	<i>psuedoRandomQuad</i>
-------------------------------	-------------------------

Description

A relatively quick way to generate psuedo-random quadrature set for ppm models.

Usage

```
pseudoRandomQuad(npoints, window, coord = c("X", "Y"), control)
```

Arguments

<code>npoints</code>	The number of background points to use. The default is 1000, but it is generally recommended to use more when fitting a ppm.
<code>window</code>	A SpatRaster from terra package which will represent the extent and resolution of the point process model.
<code>coord</code>	The names of the coordinates. Default is <code>c("X","Y")</code> .
<code>control</code>	A list of control options passed from ppmData

Author(s)

Skipton Woolley

Examples

```
library(ppmData)
library(terra)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
window <- rast(lst[1])
res <- pseudoRandomQuad(10000,window,control=list(quiet=FALSE))
plot(window)
points(res[,1:2],pch=".")
```

pseudoRandomWeights *random_background_weights*

Description

Quick way to generate psuedo-random points using the terra package. Weights are taken as the $\text{area}(\text{window})/\text{npoints}$. The default unit is km^2 , but other units can be used such as meters squared "m" or hectares "ha". There is an internal check to see if the input data is in lon/lat and if so, it returns the area of the cell in which the point lies.

Usage

```
pseudoRandomWeights(
  presences,
  quadrature,
  window,
  coord,
  mark.id,
  unit = c("geo", "m", "km", "ha")
)
```

Arguments

presences	The observed presence locations
quadrature	The quadrature locations
window	A SpatRaster from terra package which will represent the extent and resolution of the point process model.
coord	The names of the coordinates. Default is c("X","Y").
mark.id	character Name of the species ID column
unit	The scale of the area weights, default is kilometers squared "km" but meters squared "m" or hectares "ha" can be used.

Author(s)

Skipton Woolley

Examples

```
library(ppmData)
library(terra)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
window <- terra::rast(lst[1])
presences <- subset(snails,SpeciesID %in% "Tasmaphena sinclairi")[,1:2]
quadrature <- pseudoRandomQuad(1000,window,control=list(quiet=FALSE))
res <- pseudoRandomWeights(presences,quadrature>window,unit="geo")
```

```
plot(window)
points(res[res$presence==1,1:2],pch=16,cex=0.5,col='blue')
points(res[res$presence==0,1:2],pch='.')

```

quasiSampFromHyperRect

quasiSampFromHyperRect

Description

Copied from MBHdesign as it wasn't an exported function.

Usage

```
quasiSampFromHyperRect(nSampsToConsider, randStartType = 2, designParams)
```

Arguments

nSampsToConsider	Number of samples to consider
randStartType	Random start type, default is 2.
designParams	A list of design parameters for the sampler.

Author(s)

Scott Foster

rast_to_sf

Convert a terra SpatRaster object to an sf object Convert a terra SpatRaster* object to an sf object. This creates a binary mask for the raster NA vs data and then converts it to POLYGONS. For use internally to turn a rast object in to a mask for polygon clipping.*

Description

Convert a terra SpatRaster* object to an sf object Convert a terra SpatRaster* object to an sf object. This creates a binary mask for the raster NA vs data and then converts it to POLYGONS. For use internally to turn a rast object in to a mask for polygon clipping.

Usage

```
rast_to_sf(x, ...)
```

Arguments

x	rast* to be converted into an object class sf
...	further specifications, see st_as_sf

raster_datasets	<i>Raster data set of environmental and social covariates clipped to Tasmania</i>
-----------------	---

Description

Raster data set of environmental and social covariates clipped to Tasmania

Details

The a subset of the ANUCLIM v6 layers are included in the package, but have been clipped to the Tasmanian extent. A distance from roads layer is also included which is calculated as the distance from main roads, based on the road network dataset from OpenStreetMap (OSM) across the geographic area of Tasmania

- precip_of_driest_month.tif Precipitation of the driest month
- min_temp_coldest_month.tif Minimum temperature of the coldest month
- max_temp_coldest_month.tif Maximum temperature of the warmest month
- annual_mean_temp.tif Annual mean temperature
- annual_mean_precip.tif Annual mean precipitation
- distance_from_main_roads.tif Distance from main roads

References

Xu, Tingbao, and Michael Hutchinson. "ANUCLIM version 6.1 user guide." The Australian National University, Fenner School of Environment and Society, Canberra 90 (2011).

OpenStreetMap, (2020): OpenStreetMap - Road Network (Australia) 2020; accessed from AURIN.

Examples

```
## Not run:
## This is how you would load these data
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
covariates <- rast(lst)

## End(Not run)
```

snails	<i>Presence only observations of snails from the family Rhytididae located in Tasmania, Australia.</i>
--------	--

Description

Presence only observations of snails from the family Rhytididae located in Tasmania, Australia.

Usage

```
snails
```

Format

A data frame with columns:

X Longitude.

Y Latitude.

SpeciesID Species taxonomic name.

Source

Atlas of Living Australia. <http://www.ala.org/>

Examples

```
## Not run:
snails

## End(Not run)
```

terra2im	<i>convert at terra raster to a spatstat image taken from maptools and converted to terra</i>
----------	---

Description

convert at terra raster to a spatstat image taken from maptools and converted to terra

Usage

```
terra2im(rast, factor.col.name = NULL)
```

Arguments

`rast` rast A terra raster to convert to a spatstat image
`factor.col.name` string Names of factors in the raster to convert to window

Examples

```
library(ppmData)
library(terra)
path <- system.file("extdata", package = "ppmData")
lst <- list.files(path=path,pattern='*.tif',full.names = TRUE)
covariates <- rast(lst[1])
im.r <- terra2im(covariates)

plot(im.r)
```

Index

* datasets

snails, [16](#)

as.owin.ppmData, [2](#)

as.ppp, [3](#)

as.spatstat, [3](#)

dirTess, [4](#)

extract, [10](#), [11](#)

gridQuad, [5](#)

gridWeights, [5](#)

plot.dirTess, [6](#)

plot.ppmData, [7](#)

polygonise, [8](#)

ppmData, [9](#)

print.ppmData, [11](#)

pseudoRandomQuad, [12](#)

pseudoRandomWeights, [13](#)

quasiSampFromHyperRect, [14](#)

rast_to_sf, [14](#)

raster_datasets, [15](#)

snails, [16](#)

st_as_sf, [14](#)

terra2im, [16](#)