Package ‘ppmlasso’

April 5, 2022

Type Package

Title Point Process Models with LASSO-Type Penalties

Version 1.2

Date 2022-04-01

Author Ian Renner

Maintainer Ian Renner <`Ian.Renner@newcastle.edu.au`>

Description Toolkit for fitting point process models with sequences of LASSO penalties (``regularisation paths''), as described in Renner, I.W. and Warton, D.I. (2013) <doi:10.1111/j.1541-0420.2012.01824.x>. Regularisation paths of Poisson point process models or area-interaction models can be fitted with LASSO, adaptive LASSO or elastic net penalties. A number of criteria are available to judge the bias-variance tradeoff.

License GPL-3

Depends R (>= 4.0.0), spatstat.core, spatstat.geom, spatstat (>= 2.0-0)

Imports methods, data.table, lattice, stats, plyr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-05 09:00:02 UTC

R topics documented:

`ppmlasso-package` *PPM-LASSO: Point process models with LASSO-type penalties*

Description

This package contains tools to fit point process models with sequences of LASSO penalties ("regularisation paths"). Regularisation paths of Poisson point process models or area-interaction models can be fitted with LASSO, adaptive LASSO or elastic net penalties. A number of criteria are available to judge the bias-variance tradeoff.

Details

The key functions in ppmlasso are as follows:

Useful pre-analysis functions:

`findRes` Determine the optimal spatial resolution at which to perform analysis

`getEnvVar` Interpolate environmental data to species presence locations

`griddify` Ensure a matrix of environmental data is on a rectangular grid

`ppmdat` Calculate observation weights and set up design matrix for fitting

`pointInteractions` Calculate interpoint interactions for fitting area-interaction models

`sampleQuad` Set up a regular grid of quadrature points

Creating regularisation paths of point process models:

`ppmlasso` Fit a regularisation path of point process models

`plotFit` Plot the fitted intensity of a `ppmlasso` object

`plotPath` Plot the regularisation path of a `ppmlasso` object

`print.ppmlasso` Print output from a `ppmlasso` object

`predict.ppmlasso` Make predictions from a fitted point process model to new data

Checking assumptions:

`diagnose.ppmlasso` Create diagnostic residual plots of `ppmlasso` object

`envelope.ppmlasso` Create simulation envelope for goodness-of-fit checks on a `ppmlasso` object

Author(s)

Ian W. Renner

Maintainer: Ian W. Renner <Ian.Renner@newcastle.edu.au>

References

Renner, I.W. & Warton, D.I. (2013). Equivalence of MAXENT and Poisson point process models for species distribution modeling in ecology *Biometrics* **69**, 274-281.

Renner, I.W. et al (2015). Point process models for presence-only analysis. *Methods in Ecology* \& *Evolution* **6**, 366-379.

Renner, I.W., Warton, D.I., & Hui, F.K.C. (2021). What is the effective sample size of a spatial point process? *Australian & New Zealand Journal of Statistics* **63**, 144-158.

Examples

```
# Fit a regularisation path of Poisson point process models
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2, raw = TRUE)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)

# Fit a regularisation path of area-interaction models
data(BlueMountains)
ai.form  = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2, raw = TRUE)
ai.fit   = ppmlasso(ai.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1,
family = "area.inter", r = 2, availability = BlueMountains$availability, n.fits = 20,
writefile = FALSE)

# Print a ppmlasso object
print(ppm.fit, out = "model")

# Residual plot of a ppmlasso object
diagnose(ppm.fit, which = "smooth", type = "Pearson")

# Make predictions
pred.mu = predict(ppm.fit, newdata = sub.env)

# Plot the intensity from a fitted ppmlasso object
plotFit(ppm.fit)

# Plot the regularisation path from a fitted ppmlasso object
plotPath(ppm.fit)
```

`BlueMountains` *Blue Mountains eucalypt and environmental data.*

Description

This data set contains the observed presence locations of a Sydney eucalypt (`eucalypt`), the values of four environmental variables and two variables related to site accessibility throughout the region at a spatial resolution of 500m (`env`), and a matrix indicating whether locations in the region are available to the species (`availability`).

Usage

```
data(BlueMountains)
```

Format

A list with three objects:

eucalypt A data frame with a column `X` of UTM Easting coordinates (km) and a column `Y` of UTM Northing coordinates (km) of observed locations of a Sydney eucalypt

env A data frame containing environmental data in the Blue Mountains region near Sydney

availability A 301x201 matrix with UTM Northing and Easting locations stored in `dimnames` indicating whether locations are accessible or not

The data frame `env` contains the following environmental data:

X UTM Easting coordinates (km)

Y UTM Northing coordinates (km)

FC Number of fires since 1943

D_MAIN_RDS Distance from the nearest main road (m)

D_URBAN Distance from the nearest urban area (m)

RAIN_ANN Average annual rainfall (mm)

TMP_MAX Average maximum temperature (degrees Celsius)

TMP_MIN Average minimum temperature (degrees Celsius)

`diagnose-methods` *Methods for function* `diagnose`

Description

Methods for function `diagnose`

Methods

`signature(fit = "ppm")` Creates residual plots for a `ppm` object. See the help function for `diagnose.ppm` in `spatstat` for more details.

`signature(fit = "ppmlasso")` Creates residual plots for a `ppmlasso` object. See the help function for `diagnose.ppm` in `spatstat` for more details.

`diagnose.ppmlasso` *Create diagnostic plots for a fitted point process model.*

Description

This function is analogous to the `diagnose.ppm` function of the `spatstat` package.

Usage

```
## S3 method for class 'ppmlasso'
diagnose(object, ...)
```

Arguments

`object`	A fitted regularisation path of point process models. The diagnostic plots will be created for the model that optimises the given criterion.
`...`	Other arguments for producing diagnostic plots, as given by the `diagnose.ppm` function of the `spatstat` package.

Details

See the help file for `diagnose.ppm` in the `spatstat` package for further details of diagnostic plots.

Author(s)

Ian W. Renner

References

Baddeley, A.J. & Turner, R. (2005). Spatstat: an R package for analyzing spatial point patterns. *Journal of Statistical Software* **12**, 1-42.

See Also

envelope.ppmlasso, for other goodness-of-fit functions inherited from spatstat.

Examples

```
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2) + poly(D_MAIN_RDS, D_URBAN, degree = 2)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)
diagnose(ppm.fit, which = "smooth", type = "Pearson")
```

---

envelope-methods *Methods for function* envelope

Description

Methods for function envelope

Methods

signature(Y = "ppmlasso") Creates Monte Carlo simulation envelopes for a given function for a ppmlasso object. See the help function for envelope in spatstat for more details.

---

envelope.ppmlasso *Calculates simulation envelopes for goodness-of-fit*

Description

This function is analogous to the envelope function of the spatstat package.

Usage

```
## S3 method for class 'ppmlasso'
envelope(Y, fun = Kest, ...)
```

Arguments

Y	A fitted regularisation path of point process models. The simulation envelopes will be calculated for the model that optimises the given criterion.
fun	The summary function to be computed for the given point process model. See the help file for the envelope function of the spatstat package for more details.
...	Other arguments for producing diagnostic plots, as given by the envelope function of the spatstat package.

Details

See the help file for envelope in the spatstat package for further details of simulation envelopes.

Author(s)

Ian W. Renner

References

Baddeley, A.J. & Turner, R. (2005). Spatstat: an R package for analyzing spatial point patterns. *Journal of Statistical Software* **12**, 1-42.

See Also

diagnose.ppmlasso, for residual plots inherited from spatstat.

Examples

```
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2) + poly(D_MAIN_RDS, D_URBAN, degree = 2)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)
envelope(ppm.fit, Kinhom, nsim = 20)
```

---

findRes *Choose spatial resolution for analysis*

Description

This function produces a plot to choose the optimal spatial resolution for analysis. A point process model is calculated for each nominated spatial resolution and the log-likelihood of all fitted models are plotted against the spatial resolutions.

Usage

```
findRes(scales, lambda = 0, coord = c("X", "Y"), sp.xy, env.grid,
formula, tol = 0.01, ...)
```

Arguments

scales	A vector of spatial resolutions for which to produce the plot.
lambda	The penalty for each fitted spatial resolution. This should be a single value such that only one point process model is computed for each spatial resolution.
coord	A vector containing the names of the longitude and latitude coordinates, used by getEnvVar.

`sp.xy`	A matrix of species locations containing at least one column representing longitude and one column representing latitude, as in `ppmlasso`.
`env.grid`	The geo-referenced matrix of environmental grids, as in `ppmlasso`.
`formula`	The formula of the fitted model, as in `ppmlasso`.
`tol`	An optional argument to specify the tolerance level of coordinate error passed to an internal call to the `griddify` function, set to 0.01 by default.
`...`	Further arguments passed to `ppmlasso`.

Details

This function produces a plot which can be used to judge an optimal spatial resolution for analysis. As the spatial resolution gets finer, the log-likelihood tends to stabilise to a constant value. The largest spatial resolution at which the log-likelihood appears to stabilise may be considered optimal for model fitting.

Value

A plot of log-likelihood versus spatial resolution.

Author(s)

Ian W. Renner

References

Renner, I.W. et al (2015). Point process models for presence-only analysis. *Methods in Ecology \& Evolution* **6**, 366-379.

Examples

```
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
scales = c(0.5, 1, 2, 4, 8, 16)
ppm.form = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2)
findRes(scales, formula = ppm.form, sp.xy = sub.euc, env.grid = sub.env)
```

`getEnvVar` *Extract environmental data to presence locations*

Description

Given a matrix of quadrature points and a list of species presences, this function extracts environmental data to presence locations using bilinear interpolation.

Usage

```
getEnvVar(sp.xy, env.grid, env.scale, coord = c("X", "Y"), envfilename = "SpEnvData",
tol = 0.01, writefile = TRUE)
```

Arguments

`sp.xy`	A matrix of species locations containing at least one column representing longitude and one column representing latitude.
`env.grid`	The geo-referenced matrix of environmental grids.
`env.scale`	The spatial resolution of the quadrature scheme from which the environmental data is extracted.
`coord`	A vector containing the names of the longitude and latitude coordinates, as in `sampleQuad`.
`envfilename`	An optional argument containing the name of the saved file. Setting `envfilename = "SpEnvData"` will save a matrix `sp.dat` containing the species presence locations and the interpolated environmental data to the file `"SpEnvData.RData"`.
`tol`	An optional argument to specify the tolerance level of coordinate error passed to an internal call to the `griddify` function, set to 0.01 by default.
`writefile`	A logical argument to determine whether the output should be written to a file or not. If `TRUE` (the default), the output will be saved with the file name as described in the `envfilename` argument.

Details

At a given species location with coordinates (x, y), the interpolated value of the environmental variable z is calculated as a weighted average of z at four reference quadrature points $(x^{(1)}, y^{(1)})$, $(x^{(1)}, y^{(2)})$, $(x^{(2)}, y^{(1)})$ and $(x^{(2)}, y^{(2)})$ that form a square of nominated side length `env.scale` surrounding (x, y).

Value

A matrix containing locations of species presences in the first two columns and the interpolated environmental data in the remaining columns.

Author(s)

Ian W. Renner

Examples

```
data(BlueMountains)
species.env = getEnvVar(BlueMountains$eucalypt, env.grid = BlueMountains$env, env.scale = 0.5,
envfilename = NA, writefile = FALSE)
```

griddify — *Ensure that a geo-referenced matrix of environmental grids is rectangular*

Description

This function ensures that the coordinates of the supplied geo-referenced matrix of environmental grids constitutes a rectangular grid.

Usage

```
griddify(envframe, tol = 0.01, coord = c("X", "Y"))
```

Arguments

`envframe`	The geo-referenced matrix of environmental grids.
`tol`	The tolerance level within which to correct coordinate errors.
`coord`	A vector containing the names of the longitude and latitude coordinates.

Details

The functions in the `ppmlasso` package require a set of quadrature points along a rectangular grid. At times a set of quadrature points with a desired spatial resolution of $x_\delta \times y_\delta$ will have some minor machine error in some coordinates such that the coordinates as supplied do not consistute a rectangular grid. The `griddify` function corrects this error as follows:

Let $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, y_2, \ldots, y_n\}$ be the supplied coordinates contained in `envframe`. The function first determines the spatial resolution $x_\delta \times y_\delta$ based on the median of the differences in the unique values of x_i and y_i as well as the coordinates of a rectangular grid with this spatial resolution $\{x_1^{grid}, x_2^{grid}, \ldots, x_n^{grid}\}$ and $\{y_1^{grid}, y_2^{grid}, \ldots, y_n^{grid}\}$. Given the tolerance ϵ supplied to `tol`, any coordinate x_i for which $0 < \left|x_i - x_i^{grid}\right| \leq epsilon \times x_\delta$ will be adjusted to x_i^{grid}. Likewise, any coordinate y_i for which $0 < \left|y_i - y_i^{grid}\right| < epsilon \times y_\delta$ will be adjusted to y_i^{grid}.

Any environmental variables contained in `envframe` are left unchanged.

Value

A data frame containing the coordinates on a rectangular grid as well as any environmental variables left unchanged.

Author(s)

Ian W. Renner

Examples

```
X = seq(0, 5, 1)
Y = seq(1, 11, 2)
XY = expand.grid(X, Y) # generate 1 x 2 rectangular grid
names(XY) = c("X", "Y")
#move some coordinates off of rectangular grid
XY$X[1] = XY$X[1] - 0.01
XY$Y[1] = XY$Y[1] - 0.01
XY$X[7] = XY$X[7] + 0.01
XY$Y[7] = XY$Y[7] + 0.01

#generate environmental variables
XY$V1 = 0.1*XY$X + 0.2*XY$Y + rnorm(36, 0, 1)
XY$V2 = -0.2*XY$X + 0.1*XY$Y + 0.05*XY$X*XY$Y + rnorm(36, 0, 5)

XY_grid = griddify(XY)
```

`plotFit` *Plot the predicted intensity of a fitted* `ppmlasso` *model*

Description

This function produces a levelplot of the predicted intensity from a fitted `ppmlasso` model.

Usage

```
plotFit(fit, pred.data = data.frame(X = fit$x[fit$pres == 0],
Y = fit$y[fit$pres == 0], 1, scale(fit$data[fit$pres == 0, -1],
center = -fit$s.means/fit$s.sds, scale = 1/fit$s.sds)),
coord = c("X", "Y"), asp = "iso", ylab = "", xlab = "",
col.regions = heat.colors(1024)[900:1], cuts = length(col.regions),
cex = 1.4, main.text = paste(toupper(fit$criterion), "fit"), cex.color = 1.4)
```

Arguments

`fit`	A fitted `ppmlasso` object.
`pred.data`	A data frame which defines the coordinates and the environmental variables for which the predicted intensity will be calculated. By default, this uses the coordinates and environmental variables from the object supplied to the `fit` argument.
`coord`	A vector containing the names of the longitude and latitude coordinates, as in `sampleQuad`.
`asp`	Aspect of the plot, with "iso" as default. See the documentation for the `levelplot` function of the `lattice` pacakge for more details.
`ylab`	Label for the y-axis. Blank by default.
`xlab`	Label for the x-axis. Blank by default

col.regions	A vector of colours to define the intensity gradient. See the documentation for the levelplot function of the lattice pacakge for more details.
cuts	The number of levels the color gradient for the intensity surface is divided into. See the documentation for the levelplot function of the lattice pacakge for more details.
cex	Character size for axis labels.
main.text	Title of the plot.
cex.color	Character size for the colorkey labels.

Details

This function will compute the predicted intensity of a fitted ppmlasso object using the model within the regularisation path which optimises the criterion specified in the call to ppmlasso.

Author(s)

Ian W. Renner

Examples

```
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2) + poly(D_MAIN_RDS, D_URBAN, degree = 2)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)
plotFit(ppm.fit)
```

---

plotPath *Plot of the regularisation path of a* ppmlasso *model*

Description

This function produces a trace plot of the coefficient estimates of a fitted ppmlasso object and identifies the models which optimise the various penalisation criteria.

Usage

```
plotPath(fit, colors = c("gold", "green3", "blue", "brown", "pink"), logX = TRUE)
```

Arguments

fit	A fitted ppmlasso object.
colors	A vector of colours for each criterion: "aic", "bic", "hqc", "gcv", and "nlgcv".
logX	A logical argument to indicate whether the plot should utilise a logarithmic scale on the x-axis (the default) or not.

Details

A fitted ppmlasso object contains a matrix called betas which stores the coefficient estimates of each of the n.fits models fitted. This function produces a traceplot of these coefficient estimates for each environmental variable and highlights the models which optimise each of the penalisation criteria.

Author(s)

Ian W. Renner

Examples

```
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2) + poly(D_MAIN_RDS, D_URBAN, degree = 2)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)
plotPath(ppm.fit)
```

---

pointInteractions *Calculate point interactions for area-interaction models*

Description

This function calculates point interactions at presence locations and quadrature points required for fitting a regularisation path of area-interaction models.

Usage

```
pointInteractions(dat.ppm, r, availability = NA)
```

Arguments

dat.ppm	A design matrix generated using the ppmdat function.
r	The radius of point interactions.
availability	An optional binary matrix used in calculating point interactions indicating whether locations are available (availability = 1) or not (availability = 0). If no such matrix is provided, availability is automatically generated with all values set to 1 at a special resolution of half of r.

Details

Theoretically, the point interaction $t(y)$ at a point y is calculated as the proportion of available area in a circular region Y or radius r centred at y that overlaps with circles of radius r centred at other presence locations (Baddeley & Turner, 2005).

This function discretises the study region at the same spatial resolution as `availability` by defining the matrix `occupied`, a fine grid of locations spanning the study region initialised to zero. The values of `occupied` within a distance of `r` of each presence location are then augmented by `1`, such that `occupied` then contains the total number of presence locations with which each grid location interacts. To prevent unavailable areas from being included in the calculation of point interactions, the values of `occupied` at grid locations for which `availability = 0` are set to zero.

$t(y)$ is then estimated as the proportion of available grid locations within Y that overlap circular regions around other presence locations.

The availability matrix is particularly useful for regions that have inaccessible areas (**e.g.** due to the presence of ocean or urban areas).

Finer resolutions of the `availability` matrix will yield more precise estimates but at a cost of greater computation time.

Value

A vector of point interactions corresponding to the locations contained in the `dat.ppm` argument.

Author(s)

Ian W. Renner

References

Baddeley, A.J. & Turner, R. (2005). Spatstat: an `R` package for analyzing spatial point patterns. *Journal of Statistical Software* **12**, 1-42.

See Also

`ppmlasso` for fitting a regularisation path of area-interaction models

Examples

```
data(BlueMountains)
species.ppm = ppmdat(sp.xy = BlueMountains$eucalypt, back.xy = BlueMountains$env,
sp.scale = 1, datfilename = NA, writefile = FALSE) # generate design matrix
species.int = pointInteractions(species.ppm, 2, BlueMountains$availability)
```

ppmdat *Prepare data for model fitting*

Description

This function prepares the data for model fitting. In particular, it determines observation weights and sets up the design matrix required for fitting a regularisation path.

Usage

```
ppmdat(sp.xy, sp.scale, back.xy, coord = c("X","Y"), sp.dat = getEnvVar(sp.xy = sp.xy,
env.scale = sp.scale, env.grid = back.xy, coord = coord, writefile = writefile),
sp.file = NA, quad.file = NA, datfilename = "PPMDat", writefile = TRUE)
```

Arguments

sp.xy	A matrix of species locations containing at least one column representing longitude and one column representing latitude.
sp.scale	The spatial resolution at which to sample quadrature points.
back.xy	The geo-referenced matrix of environmental grids.
coord	A vector containing the names of the longitude and latitude coordinates, as in sampleQuad.
sp.dat	A matrix of species presence locations and the corresponding environmental data, as generated by getEnvVar.
sp.file	The name of a saved file containing a matrix of species presence locations and the corresponding environmental data, as generated by getEnvVar.
quad.file	The name of a fie containing the quadrature points created from the sampleQuad function.
datfilename	An optional argument containing the name of the saved file. Setting datfilename = "PPMDat" will save a matrix dat.ppm containing the species presence locations and quadrature points, the environmental data, as well as an indicator for whether the locations correspond to presence locations or quadrature points and the corresponding quadrature weights to the file "PPMDat.RData".
writefile	A logical argument to determine whether the output should be written to a file or not. If TRUE (the default), the output will be saved with the file name as described in the datfilename argument.

Details

This function will call the sampleQuad and getEnvVar functions to generate a quadrature scheme and interpolate environmental data to presence locations. Alternatively, the quadrature scheme may be directly supplied to the quad.file argument, and the matrix of presence locations and associated environmental data may be directly supplied to either the sp.dat argument (as an object in the workspace) or to the sp.file argument (as the name of a saved file containing this matrix).

Value

A matrix `dat.ppm` with columns representing the latitude and longitude of presence locations and quadrature points along with the associated environmental data, as well as a column `Pres` indicating whether either point corresponds to a presence location or a quadrature point, and a column `wt` of observation weights.

Author(s)

Ian W. Renner

See Also

`sampleQuad` for generating a regular grid of quadrature points.

`getEnvVar` for interpolating environmental data to species presence locations.

Examples

```
data(BlueMountains)
species.ppm = ppmdat(sp.xy = BlueMountains$eucalypt, back.xy = BlueMountains$env,
sp.scale = 1, datfilename = NA, writefile = FALSE)
```

`ppmlasso` *Fit point process models with LASSO penalties*

Description

The ppmlasso function fits point process models (either Poisson or area-interaction models) with a sequence of LASSO, adaptive LASSO or elastic net penalties (a "regularisation path").

Usage

```
ppmlasso(formula, sp.xy, env.grid, sp.scale, coord = c("X", "Y"),
data = ppmdat(sp.xy = sp.xy, sp.scale = sp.scale, back.xy = env.grid, coord = c("X","Y"),
sp.file = NA, quad.file = NA, datfilename = "PPMDat", writefile = writefile), lamb = NA,
n.fits = 200, ob.wt = NA, criterion = "bic", alpha = 1, family = "poisson", tol = 1.e-9,
gamma = 0, init.coef = NA, mu.min = 1.e-16, mu.max = 1/mu.min, r = NA, interactions = NA,
availability = NA, max.it = 25, min.lamb = -10, standardise = TRUE, n.blocks = NA,
block.size = sp.scale*100, seed = 1, writefile = TRUE)
```

Arguments

`formula`	The formula of the fitted model. For a point process model, the correct form is `~ variables`.
`sp.xy`	A matrix of species locations containing at least one column representing longitude and one column representing latitude. Environmental variables are interpolated to the locations of `sp.xy` using the `getEnvVar` function, unless the `data` argument is supplied.

`env.grid`	The geo-referenced matrix of environmental grids. This matrix is used to generate quadrature points using the `sampleQuad` function, interpolate environmental data to the species locations of `sp.xy` using the `getEnvVar` function, and calculate observation weights using the `ppmdat` function, unless the `data` argument is supplied. This creates a data matrix `data` which provides the variables for the `formula` argument.
`sp.scale`	The spatial resolution at which to define the regular grid of quadrature points. `sampleQuad` will subsample from the rows of `data` that coincide with a regular grid at a resolution of `sp.scale`.
`coord`	A vector containing the names of the longitude and latitude coordinates.
`data`	An optional data matrix generated from the `ppmdat` function. Supplying a matrix to `data` is an alternative way of providing the environmental variables used in the `formula` argument, instead of specifying `sp.xy` and `env.grid`.
`lamb`	A vector of penalty values that will be used to create the regularisation path. If `lamb = NA`, the penalty values are automatically generated from the `data` and the `n.fits` argument.
`n.fits`	The number of models fitted in the regularisation path. If `lamb = NA`, the `n.fits` penalty values will be equally spaced on a logarithmic scale from e^{-10} to λ_{max}, the smallest penalty that shrinks all parameter coefficients to zero.
`ob.wt`	Quadrature weights, usually inherited from the `ppmdat` function.
`criterion`	The penalisation criteria to be optimised by the regularisation path. The options include `"aic"`, `"bic"`, `"blockCV"`, `"hqc"`, `"gcv"`, `"nlgcv"` and `"msi"`.
`alpha`	The elastic net parameter. The form of the penalty is $$\alpha * \lambda * \sum_{j=1}^{p} \lvert\beta_j\rvert + (1-\alpha) * \lambda * \sum_{j=1}^{p} (\beta_j)^2.$$ The default value `alpha = 1` corresponds to a LASSO penalty, while `alpha = 0` corresponds to a ridge regression penalty.
`family`	The family of models to be fitted – `family = "poisson"` for Poisson point process models or `family = "area.inter"` for area-interaction models.
`tol`	The convergence threshold for the descent algorithm. The algorithm continues for a maximum of `max.it` iterations until the difference in likelihood between successive fits falls below `tol`.
`gamma`	The exponent of the adaptive weights for the adaptive LASSO penalty. The default value `gamma = 0` corresponds to a normal LASSO penalty.
`init.coef`	The initial coefficients used for an adaptive LASSO penalty.
`mu.min`	The threshold for small fitted values. Any fitted value less than the threshold is set to `mu.min`.
`mu.max`	The threshold for large fitted values. Any fitted value larger than the threshold will be set to `mu.max`.
`r`	The radius of point interactions, required if `family = "area.inter"`.

`interactions`	A vector of point interactions calculated from the `pointInteractions` function necessary for fitting area-interaction models. If `interactions` = `NA` and `family` = `"area.inter"`, point interactions will be automatically calculated for radius `r` to the locations of `data`.
`availability`	An optional binary matrix used in calculating point interactions indicating whether locations are available (1) or not (0). See `pointInteractions` for more details.
`max.it`	The maximum number of iterations of the descent algorithm for fitting the model.
`min.lamb`	The power x of smallest penalty e^x among the `n.fits` models.
`standardise`	A logical argument indicating whether the environmental variables should be standardised to have mean 0 and variance 1. It is recommended that variables are standardised for analysis.
`n.blocks`	This argument controls the number of cross validation groups into which the spatial blocks are divided if the `criterion` argument is set to `"blockCV"`. See details.
`block.size`	The length of the edges for the spatial blocks created if the `criterion` argument is set to `"blockCV"`. Only square spatial blocks are currently supported. See details.
`seed`	The random seed used for controlling the allocation of spatial blocks to cross validation groups if the `criterion` argument is set to `"blockCV"`.
`writefile`	A logical argument passed to the `ppmdat` function to determine whether its output should be written to a file or not, set to `TRUE` by default. See the documentation for `ppmdat` for details.

Details

This function fits a regularisation path of point process models provided a list of species locations and a geo-referenced grid of environmental data. It is assumed that Poisson point process models (Warton & Shepherd, 2010) fit intensity as a log-linear model of environmental covariates, and that area-interaction models (Widom & Rowlinson, 1970; Baddeley & van Lieshout, 1995) fit conditional intensity as a log-linear model of environmental covariates and point interactions. Parameter coefficients are estimated by maximum likelihood for Poisson point process models and by maximum pseudolikelihood (Besag, 1977) for area-interaction models. The expressions for both the likelihood and pseudolikelihood involve an intractable integral which is approximated using a quadrature scheme (Berman & Turner, 1992).

Each model in the regularisation path is fitted by extending the Osborne descent algorithm (Osborne, 2000) to generalised linear models with penalised iteratively reweighted least squares.

Three classes of penalty $p(\beta)$ are available for the vector of parameter coefficients β:

For the LASSO (Tibshirani, 1996), $p(\beta) = \lambda * \sum_{j=1}^{p} |\beta_j|$

For the adaptive LASSO (Zou, 2006), $p(\beta) = \lambda * \sum_{j=1}^{p} w_j * |\beta_j|$, where $w_j = 1/|\hat{\beta}_{init,j}|^{\gamma}$ for some initial estimate of parameters $\hat{\beta}_{init}$.

For the elastic net (Zou & Hastie, 2005), $\alpha * \lambda * \sum_{j=1}^{p} |\beta_j| + (1-\alpha) * \lambda * \sum_{j=1}^{p} (\beta_j)^2$. Note that this form of the penalty is a restricted case of the general elastic net penalty.

There are various criteria available for managing the bias-variance tradeoff (Renner, 2013). The default choice is BIC, the Bayesian Information Criterion, which has been shown to have good performance.

An alternative criterion useful when data are sparse is MSI, the maximum score of the intercept model (Renner, in prep). For a set of m presence locations, the MSI penalty is $\lambda_{MSI} = \lambda_{max}/\sqrt{m}$, where λ_{max} is the smallest penalty that shrinks all environmental coefficients to zero. The MSI penalty differs from the other criteria in that does not require an entire regularisation path to be fitted.

It is also possible to control the magnitude of the penalty by spatial cross validation by setting the `criterion` argument to `"blockCV"`. The study region is then divided into square blocks with edge lengths controlled by the `block.size` argument, which are assigned to one of a number of cross validation groups controlled by the `n.groups` argument. The penalty which maximises the predicted log-likelihood is chosen.

Value

An object of class `"ppmlasso"`, with elements:

`betas`	A matrix of fitted coefficients of the `n.fits` models.
`lambdas`	A vector containing the `n.fits` penalty values.
`likelihoods`	A vector containing the likelihood of `n.fits` fitted models.
`pen.likelihoods`	A vector containing the penalised likelihood of `n.fits` fitted models.
`beta`	A vector containing the coefficients of the model that optimises the specified `criterion`.
`lambda`	The penalty value of the model that optimises the specified `criterion`.
`mu`	A vector of fitted values from the model that optimises the specified `criterion`.
`likelihood`	The likelihood of the model that optimises the specified `criterion`.
`criterion`	The specified `criterion` of the function call.
`family`	The specified `family` of the function call.
`gamma`	The specified `gamma` of the function call.
`alpha`	The specified `alpha` of the function call.
`init.coef`	The specified `init.coef` of the function call.
`criterion.matrix`	A matrix with `n.fits` rows corresponding to the observed values of AIC, BIC, HQC, GCV, and non-linear GCV.
`data`	The design matrix. For the point process models fitted with this function, `mu` = `e^{data*beta}`.
`pt.interactions`	The calculated point interactions.
`wt`	The vector of quadrature weights.
`pres`	A vector indicating presence (1) or quadrature point (0).
`x`	A vector of point longitudes.
`y`	A vector of point latitudes.
`r`	The radius of point interactions.
`call`	The function call.

`formula`	The `formula` argument.
`s.means`	If `standardise` = `TRUE`, the means of each column of `data` prior to standardisation.
`s.sds`	If `standardise` = `TRUE`, the standard deviations of each column of `data` prior to standardisation.
`cv.group`	The cross validation group associated with each point in the data set.
`n.blocks`	The number of cross validation groups specified.

Author(s)

Ian W. Renner

References

Baddeley, A.J. & van Lieshout, M.N.M. (1995). Area-interaction point processes. *Annals of the Institute of Statistical Mathematics* **47**, 601-619.

Berman, M. & Turner, T.R. (1992). Approximating point process likelihoods with GLIM. *Journal of the Royal Statistics Society, Series C* **41**, 31-38.

Besag, J. (1977). Some methods of statistical analysis for spatial data. *Bulletin of the International Statistical Institute* **47**, 77-91.

Osborne, M.R., Presnell, B., & Turlach, B.A. (2000). On the lasso and its dual. *Journal of Computational and Graphical Statistics* **9**, 319-337.

Renner, I.W. & Warton, D.I. (2013). Equivalence of MAXENT and Poisson point process models for species distribution modeling in ecology. *Biometrics* **69**, 274-281.

Renner, I.W. (2013). Advances in presence-only methods in ecology. `https://unsworks.unsw.edu.au/fapi/datastream/unsworks:11510/SOURCE01`

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B* **58**, 267-288.

Warton, D.I. & Shepherd, L.C. (2010). Poisson point process models solve the "pseudo-absence problem" for presence-only data in ecology. *Annals of Applied Statistics* **4**, 1383-1402.

Widom, B. & Rowlinson, J.S. (1970). New model for the study of liquid-vapor phase transitions. *The Journal of Chemical Physics* **52**, 1670-1684.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association* **101**, 1418-1429.

Zou, H. & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B* **67**, 301-320.

See Also

`print.ppmlasso` for printing features of the fitted regularisation path.

`predict.ppmlasso` for predicting intensity for a set of new data.

`envelope.ppmlasso` for constructing a K-envelope of the model which optimises the given criterion from the `spatstat` package.

`diagnose.ppmlasso` for diagnostic plots from the `spatstat` package.

Examples

```
# Fit a regularisation path of Poisson point process models
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)

#Fit a regularisation path of area-interaction models
data(BlueMountains)
ai.form  = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2)
ai.fit   = ppmlasso(ai.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1,
family = "area.inter", r = 2, availability = BlueMountains$availability, n.fits = 20,
writefile = FALSE)
```

---

ppmlasso-class *Class* "ppmlasso"

Description

A class ppmlasso which represents a point process model with a LASSO-type penalty.

Methods

diagnose signature(object = "ppmlasso"): Produce diagnostic plots for a fitted point process model.

envelope signature(Y = "ppmlasso"): Produce a Monte Carlo simulation envelope for a summary function of a fitted point process model.

predict signature(object = "ppmlasso"): Calculate the predicted intensity for a fitted point process model to a set of data.

print signature(x = "ppmlasso"): Print the details of a fitted point process model.

Author(s)

Ian W. Renner

Examples

```
showClass("ppmlasso")
```

predict-methods *Methods for function* `predict`

Description

Methods for function `predict`

Methods

`signature(object = "ppmlasso")` Creates a vector of predicted intensities for an object of class `ppmlasso`.

predict.ppmlasso *Prediction to new data from a fitted regularisation path*

Description

Given a fitted regularisation path produced by `ppmlasso`, this function will predict the intensity for a new set of data.

Usage

```
## S3 method for class 'ppmlasso'
predict(object, ..., newdata, interactions = NA)
```

Arguments

`object`	A fitted regularisation path produced by `ppmlasso`.
`...`	Additional arguments impacting the prediction calculations.
`newdata`	A data frame of new environmental data for which predicted intensities are computed.
`interactions`	A vector of point interactions for predictions of area-interaction models.

Value

A vector of predicted intensities corresponding to the environmental data provided in the `newdata` argument.

Author(s)

Ian W. Renner

See Also

`ppmlasso` for fitting a regularisation path of point process models.

Examples

```
data(BlueMountains)
sub.env = BlueMountains$env[BlueMountains$env$Y > 6270 & BlueMountains$env$X > 300,]
sub.euc = BlueMountains$eucalypt[BlueMountains$eucalypt$Y > 6270 & BlueMountains$eucalypt$X > 300,]
ppm.form = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2, raw = TRUE)
ppm.fit  = ppmlasso(ppm.form, sp.xy = sub.euc, env.grid = sub.env, sp.scale = 1, n.fits = 20,
writefile = FALSE)
pred.mu  = predict(ppm.fit, newdata = sub.env)
```

`print-methods` *Methods for function* `print`

Description

Methods for function `print`

Methods

`signature(x = "ppmlasso")` Prints output for a `ppmlasso` object with details controlled by arguments of the `print.ppmlasso` function.

`print.ppmlasso` *Print a fitted regularisation path*

Description

This function prints output from a fitted regularisation path.

Usage

```
## S3 method for class 'ppmlasso'
print(x, ..., output = c("all", "path", "model", "interaction"))
```

Arguments

`x`	A regularisation path fitted by `ppmlasso`.
`...`	Further arguments controlling the printed output.
`output`	This argument controls what output is printed to the screen. If `output` includes `"path"`, information about the entire regularisation path is printed. If `output` includes `"model"`, information about the model that optimises the given criterion is printed. If `output` includes `"interaction"`, information about the point interactions is printed. Setting `output = "all"` will print all available information.

Value

N/A

Author(s)

Ian W. Renner

See Also

`ppmlasso` for fitting regularisation paths.

Examples

```
# Fit a regularisation path of Poisson point process models
data(BlueMountains)
ppm.form = ~ poly(FC, TMP_MIN, TMP_MAX, RAIN_ANN, degree = 2)
ppm.fit  = ppmlasso(ppm.form, sp.xy = BlueMountains$eucalypt,
env.grid = BlueMountains$env, sp.scale = 1, n.fits = 20, writefile = FALSE)
print(ppm.fit)
```

`sampleQuad` *Generate regular grid of quadrature points with environmental data*

Description

This function generates a regular grid of quadrature points and associated environmental data at a nominated spatial resolution.

Usage

```
sampleQuad(env.grid, sp.scale, coord = c("X", "Y"), quadfilename = "Quad", tol = 0.01,
writefile = TRUE)
```

Arguments

`env.grid`	The geo-referenced matrix of environmental grids. It must have a vector of longitude and a vector of latitude.
`sp.scale`	The spatial resolution at which to sample quadrature points.
`coord`	A vector containing the names of the longitude and latitude coordinates.
`quadfilename`	An optional argument containing the prefix of the name of the saved file. The default is `"Quad"` so that a matrix generated at a spatial resolution of 1 would be saved in the file `"Quad1.RData"`. A file is saved for every resolution given in `sp.scale`.
`tol`	An optional argument to specify the tolerance level of coordinate error passed to an internal call to the `griddify` function, set to 0.01 by default.
`writefile`	A logical argument to determine whether the output should be written to a file or not. If `TRUE` (the default), the output will be saved with the file name as described in the `quadfilename` argument.

Value

The output is a matrix of quadrature points at the spatial resolution supplied to `sp.scale`. If a vector of resolutions is supplied, the output is a list of matrices for each spatial resolution.

Author(s)

Ian W. Renner

Examples

```
data(BlueMountains)
quad.1 = sampleQuad(env.grid = BlueMountains$env, sp.scale = 1, quadfilename = NA,
writefile = FALSE)
```

Index