Package 'spatstat.convert'

September 17, 2026

Type Package

Title Extension to 'spatstat' for Converting Data Formats

Version 1.0-1

Date 2026-09-10

Depends R (>= 3.4.0), spatstat (>= 3.6-2), spatstat.geom (>= 3.8-2), spatstat.linnet (>= 3.5-3), sp (>= 2.1), stats, graphics, methods

Imports spatstat.utils (>= 3.2-4), spatstat.univar (>= 3.2-0)

Suggests terra (>= 1.7-83), sf (>= 1.0-21)

Maintainer Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

Description Extension to the 'spatstat' package, enabling the user to convert spatial data between formats defined in ` spatstat' and formats defined in 'sp', 'sf' and 'terra'.

License GPL (>= 2)

ByteCompile true

NeedsCompilation no

Author Adrian Baddeley [aut, cre],
Ege Rubak [aut],
Rolf Turner [aut],
Edzer Pebesma [aut],
Roger Bivand [aut],
Michael Sumner [ctb],
Matthew Lewis [ctb],
Mathieu Rajerison [ctb],
Mehdi Moradi [ctb]

Repository CRAN

Date/Publication 2026-09-17 14:40:02 UTC

Contents

`spatstat.convert-package`

Convert Spatial Data Formats

Description

Contains functions for converting spatial data between formats in the packages **spatstat**, **sf**, **sp** and **terra**.

Details

This package provides functions for converting spatial data objects between different formats supported in the packages **spatstat**, **sf**, **sp** and **terra**.

An object `X` can be converted to a new class `"foo"` using the S4 style syntax `as(X, "foo")`. If `"foo"` is a class in the **spatstat** package, then the conversion can also be written in the S3 style `as.foo(X)`.

This package depends on the packages **spatstat** and **sp**, but only Suggests the packages **sf** and **terra**. In order to convert data to or from a format defined in **sf**, the user must load **sf**. To convert data to or from a format defined in **terra**, the user must load **terra**.

Functions for converting *to* **spatstat** data types are:

`as.im.SpatialGridDataFrame` \ `as.im.SpatRaster` \ `as.linnet.SpatialLines` \ `as.linnet.sf` \ `as.owin.SpatialGridD`

Functions for converting *from* **spatstat** data types are:

`as.SpatialGridDataFrame.im` \ `as.SpatialPolygons.owin` \ `as.SpatialGridDataFrame.owin` \ `as.SpatialPoints.ppp`

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Ege Rubak <rubak@math.aau.dk>, Rolf Turner <rolfturner@posteo.net>, Edzer Pebesma <edzer.pebesma@uni-muenster.de>, Roger Bivand, Michael Sumner, Matthew Lewis, Mathieu Rajerison and Mehdi Moradi.

Examples

```
## Create a SpatialPolygons representing a region with a hole
holey <- SpatialPolygons(list(Polygons(list(Polygon(cbind(c(0,1,1,0,0),
  c(0,0,1,1,0))), Polygon(cbind(c(0.6,0.4,0.4,0.6,0.6),
  c(0.2,0.2,0.4,0.4,0.2)), hole=TRUE)), ID="ho")))
plot(holey, col="red", pbg="pink")
## convert SpatialPolygons -> owin
hoy <- as(holey, "owin")
plot(hoy)
##
```

`as.im.SpatialGridDataFrame`

Conversion Between Raster Images in the 'sp' and 'spatstat' Packages

---

Description

Functions to convert between raster image formats in the **spatstat** and **sp** packages.

Usage

```
  ## S3 method for class 'SpatialGridDataFrame'
as.im(X, ...)
  as.SpatialGridDataFrame.im(from)
```

Arguments

`X`, `from`	object to coerce from
`...`	Ignored.

Details

A `SpatialGridDataFrame` object may contain several columns of data such that several values are associated with each grid cell. In contrast an `im` object can only contain a single variable value for each cell. In `as.im.SpatialGridDataFrame()` the first data column is used. To convert another column to `im` format simply extract this column first as shown in the example below.

Value

Either a pixel image (object of class `"im"` as defined in the **spatstat** package) or an object of class `"SpatialGridDataFrame"` as defined in the package **sp**.

Methods

coerce `signature(from = "SpatialGridDataFrame", to = "im")`

coerce `signature(from = "im", to = "SpatialGridDataFrame")`

Warning

In **spatstat** all spatial objects are assumed to be planar. This means that **spatstat** is not designed to work directly with geographic (longitude and latitude) coordinates. If a **sp** object is declared to have geographic (unprojected) coordinates **maptools** refuses to convert directly to **spatstat** format. Rather, these should be projected first using e.g. `spTransform`. If you know what you are doing, and really want to force coercion, you can overwrite the `proj4string` of the **sp** object with `NA`, `proj4string(x) <- CRS(NA)`, which will fool the system to think that the data is in local planar coordinates. This is probably not a good idea!

Author(s)

Edzer Pebesma <edzer.pebesma@uni-muenster.de> and Roger Bivand. Adapted by Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

Other converters between **sp** and **spatstat**: `as.ppp.SpatialPoints`, `as.psp.SpatialLines`, `as.owin.SpatialPolygons`, `as.SpatialPolygons.tess`.

Examples

```
run <- TRUE
qt <- TRUE
plo <- interactive()
if (run) {
  ## Extract an example SpatialGridDataFrame and plot it
  data(meuse.grid) # A data.frame
  gridded(meuse.grid) = ~x+y # Now a SpatialPixelsDataFrame
  fullgrid(meuse.grid) <- TRUE # Finally a SpatialGridDataFrame
  mg_dist <- meuse.grid["dist"] # A SpatialGridDataFrame with a single column
  image(mg_dist, axes=TRUE)
  ## Convert to im format and plot it
  mg_im <- as(mg_dist, "im")
  if(plo) {
   plot(mg_im)
  }
  ## Convert back to SpatialGridDataFrame and plot it again
  mg2 <- as.SpatialGridDataFrame.im(mg_im)
  if(plo) {
   image(mg2, axes=TRUE)
  }
}
run <- run && require(terra, quietly=qt)
if (run) {
  ## Convert SpatialGridDataFrame -> SpatRaster -> im and plot it
  r <- as(mg2, "SpatRaster")
  rim <- as.im(r)
  if(plo) plot(rim)
  ## image with categorical values
  rr <- rast(nrows=2, ncols=3)
  values(rr) <- factor(letters[11:16])
```

```
  ii <- as.im(rr)
  if(plo) plot(ii)
}
```

`as.im.SpatRaster` *Convert 'terra' Raster Image to 'spatstat' Image*

Description

Given a raster image in the `SpatRaster` format of the **terra** package, convert it to a pixel image of class `im` in the **spatstat** package.

Usage

```
 ## S3 method for class 'SpatRaster'
as.im(X, ...)
```

Arguments

`X`	A raster image object of class `SpatRaster` in the **terra** package.
`...`	Ignored.

Details

The function `as.im` is generic. This file describes the method for `as.im` for objects of class `"SpatRaster"` in the **terra** package.

The object `X` is converted to a pixel image of class `"im"` recognised by the **spatstat** family of packages.

Value

A pixel image object of class `"im"` recognised by the **spatstat** family of packages.

Author(s)

Robert Hijmans, Ege Rubak <rubak@math.aau.dk> and Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

`as.im`.

The reverse operation is `rast`.

Examples

```
qt <- TRUE
if(require(terra, quietly=qt)) {
    f <- system.file("ex/elev.tif", package="terra")
    x <- rast(f)
    plot(x)
    z <- as.im(x)
    plot(z)
}
```

`as.linnet.sf` *Convert 'sf' Object to Linear Network in 'spatstat'*

Description

Convert an object of class `sf` or `sfc` (from package **sf**), representing a collection of polygonal lines, into an object of class `linnet` (from package **spatstat**), representing a network.

Usage

```
  ## S3 method for class 'sf'
as.linnet(X, ..., fuse=TRUE)
  ## S3 method for class 'sfc'
as.linnet(X, ..., fuse=TRUE)
```

Arguments

`X`	Object of class `sf` or `sfc` to be converted.
`...`	Ignored.
`fuse`	Logical value indicating whether to join different curves which have a common vertex.

Details

These functions convert an object of class `sf` or `sfc` into an object of class `linnet`. They can be used both as S3 methods for the **spatstat** generic function `as.linnet` which can be called in the syntax `as.linnet(X)`, and as S4 coercion methods which can be called in the syntax `as(X, "linnet")`.

An object of class `sf` or `sfc` (from package **sp**) represents a list of lists of the coordinates of lines, such as a list of all roads in a city. An object of class `linnet` in the **spatstat** package represents a linear network, such as a road network.

If `fuse=FALSE`, each line in `X` will be treated as if it were disconnected from the others. The result is a network that consists of many disconnected sub-networks, equivalent to the list of lines.

If `fuse=TRUE` (the default), the code will search for identical pairs of vertices occurring in different lines, and will treat them as identical vertices, effectively joining the two lines objects at this common vertex.

Value

A linear network (object of class `linnet` as defined in the **spatstat** package).

Methods

coerce `signature(from = "sf", to = "linnet")`

coerce `signature(from = "sfc", to = "linnet")`

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

`as.linnet`

Examples

```
qt <- TRUE
if(require(sf, quietly=qt)) {
  nc <- st_read(system.file("shape/nc.shp", package="sf"))
  b <- st_sfc(st_cast(st_geometry(nc)[[4]], "MULTILINESTRING"))
  ## convert to 'spatstat' network format
  z <- as.linnet(b)
}
```

---

as.linnet.SpatialLines

Convert SpatialLines to Linear Network

Description

Convert an object of class `SpatialLines` or `SpatialLinesDataFrame` (from package **sp**), representing a collection of polygonal lines, into an object of class `linnet` (from package **spatstat**), representing a network.

Usage

```
  ## S3 method for class 'SpatialLines'
as.linnet(X, ..., fuse = TRUE)
  ## S4 method for signature 'SpatialLines,linnet'
coerce(from, to = "linnet", strict = TRUE)
  ## S4 method for signature 'SpatialLinesDataFrame,linnet'
coerce(from, to = "linnet",
    strict = TRUE)
```

Arguments

X, from	Object of class SpatialLines or SpatialLinesDataFrame to be converted.
to	output object of class linnet.
strict	logical flag. If TRUE, the returned object must be strictly from the target class.
...	Ignored.
fuse	Logical value indicating whether to join different curves which have a common vertex.

Details

This function converts an object of class SpatialLines or SpatialLinesDataFrame into an object of class linnet. It is not a method for the **spatstat** generic function as.linnet, but like other S4 coercion functions for **sp** classes to **spatstat** classes, it may be called directly as a function.

An object of class SpatialLines or SpatialLinesDataFrame (from package **sp**) represents a list of lists of the coordinates of lines, such as a list of all roads in a city. An object of class linnet in the **spatstat** package represents a linear network, such as a road network.

If fuse=FALSE, each "Line" object in X will be treated as if it were disconnected from the others. The result is a network that consists of many disconnected sub-networks, equivalent to the list of "Line" objects.

If fuse=TRUE (the default), the code will search for identical pairs of vertices occurring in different "Line" objects, and will treat them as identical vertices, effectively joining the two "Line" objects at this common vertex.

If X belongs to class SpatialLinesDataFrame, the associated columns of data in the auxiliary data frame slot(X, "data") will be copied to the output as the marks attached to the line segments of the network. See the Examples.

Value

A linear network (object of class linnet as defined in the **spatstat** package).

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

as.linnet

Examples

```
s1 <- cbind(c(1,2,3,4),c(3,2,2,1))
s2 <- s1 + 0.05
S <- SpatialLines(list(Lines(list(Line(s1), Line(s2)), ID="a")))
L <- as.linnet(S, fuse=FALSE)
print(max(vertexdegree(L)))
## extract data associated with each network segment
head(marks(as.psp(L)))
```

`as.owin.SpatialGridDataFrame` *Coercion Between 'spatstat' Observation Windows and 'sp' Formats*

Description

Functions to convert between observation window (`owin`) objects in the **spatstat** package and various formats in the **sp** package. Coercion can use the S3 or S4 syntax.

Usage

```
## S3 method for class 'SpatialPolygons'
as.owin(W, ..., fatal=TRUE)
## S3 method for class 'SpatialGridDataFrame'
as.owin(W, ..., fatal=TRUE)
## S3 method for class 'SpatialPixelsDataFrame'
as.owin(W, ..., fatal=TRUE)
as.SpatialPolygons.owin(from)
as.SpatialGridDataFrame.owin(from)
```

Arguments

`W`	Object of class `SpatialPolygons`, `SpatialGridDataFrame` or `SpatialPixelsDataFrame` in the **sp** package.
`from`	Object of class `owin` in the **spatstat** package.
`...`	ignored
`fatal`	Logical value specifying whether conversion errors should cause a fatal error.

Details

These functions convert an object of class `owin` in the **spatstat** package into an object of class `SpatialPolygons`, `SpatialGridDataFrame` or `SpatialPixelsDataFrame` in the **sp** package, and vice versa.

An observation window object of class `owin` in the **spatstat** packages defines a planar region. It is typically used to represent a sampling region where data were observed. It comes in three different formats: a simple rectangle, a polygon (vector format) or a binary mask (TRUE/FALSE grid; raster format). These can all be coerced to polygonal form internally in **spatstat** and then converted to `SpatialPolygons`, which is what `as.SpatialPolygons.owin()` does. For objects of class `SpatialPolygons` (and `SpatialPolygonsDataFrame`) the **sp** polygons are simply extracted and cast into **spatstat**'s polygon format inside the `owin` object. For `SpatialPixelsDataFrame` and `SpatialGridDataFrame` the grid is extracted and cast into **spatstat**'s mask format inside the `owin` object. In all cases, any data apart from the spatial region itself are discarded.

Value

A spatial window (object of class "owin" as defined in the **spatstat** package) or a collection of polygons (object of class "SpatialPolygons" as defined in the package **sp**) or a spatial grid (object of class "SpatialGridDataFrame" as defined in the package **sp**).

Methods

coerce signature(from = "SpatialPolygons", to = "owin")

coerce signature(from = "SpatialPixelsDataFrame", to = "owin")

coerce signature(from = "SpatialGridDataFrame", to = "owin")

coerce signature(from = "owin", to = "SpatialPolygons")

Warning

In **spatstat** all spatial objects are assumed to be planar. This means that **spatstat** is not designed to work directly with geographic (longitude and latitude) coordinates. If a **sp** object is declared to have geographic (unprojected) coordinates **maptools** refuses to convert directly to **spatstat** format. Rather, these should be projected first using e.g. spTransform. If you know what you are doing, and really want to force coercion, you can overwrite the proj4string of the **sp** object with an empty string, proj4string(x) <- "", which will fool the system to think that the data is in local planar coordinates. This is probably not a good idea!

Note

When coercing a SpatialPolygons object to an owin object, full topology checking is enabled by default. To avoid checking, set spatstat.options(checkpolygons=FALSE). To perform the checking later, owinpolycheck(W, verbose=TRUE).

Author(s)

Edzer Pebesma <edzer.pebesma@uni-muenster.de> and Roger Bivand. Adapted by Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

as.SpatialPolygons.tess

Examples

```
## SpatialPixelsDataFrame -> owin
data(meuse.grid) # A data.frame
gridded(meuse.grid) = ~x+y # Now a SpatialPixelsDataFrame
mg_owin <- as(meuse.grid, "owin")
mg_owin
## SpatialGridDataFrame -> owin
fullgrid(meuse.grid) <- TRUE # Now a SpatialGridDataFrame
mg_owin2 <- as(meuse.grid, "owin")
## SpatialPolygons region with a hole
ho_sp <- SpatialPolygons(list(Polygons(list(Polygon(cbind(c(0,1,1,0,0),
```

```
  c(0,0,1,1,0))), Polygon(cbind(c(0.6,0.4,0.4,0.6,0.6),
  c(0.2,0.2,0.4,0.4,0.2)), hole=TRUE)), ID="ho")))
plot(ho_sp, col="red", pbg="pink")
## SpatialPolygons -> owin
ho <- as(ho_sp, "owin")
plot(ho)
## Define owin directly and check they are identical
ho_orig <- owin(poly=list(list(x=c(0,1,1,0), y=c(0,0,1,1)),
                    list(x=c(0.6,0.4,0.4,0.6), y=c(0.2,0.2,0.4,0.4))))
identical(ho, ho_orig)
## owin -> SpatialPolygons
ho_sp1 <- as(ho, "SpatialPolygons")
all.equal(ho_sp, ho_sp1, check.attributes=FALSE)
```

`as.ppp.sf` *Convert Point Pattern Data*

Description

Given a dataset of spatial points in the **sf** package, convert it to a spatial point pattern object in the **spatstat** package.

Usage

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

Arguments

`X`	An object of class `sf` in the **sf** package representing a point pattern.
`...`	Ignored.
`fatal`	Logical value specifying whether an error should be caused if the conversion cannot be performed.
`warn`	Logical value specifying whether a warning should be issued if the conversion cannot be performed.

Details

The function `as.ppp` is generic. This file describes the method for `as.ppp` for objects of class `sf` in the **sf** package.

The object `X` is converted to a spatial point pattern of class `"ppp"` recognised by the **spatstat** family of packages.

Value

A spatial point pattern (object of class `"ppp"`) recognised by the **spatstat** family of packages.

Author(s)

Ege Rubak <rubak@math.aau.dk>, Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Edzer Pebesma <edzer.pebesma@uni-muenster.de>.

See Also

as.ppp.

Examples

```
qt <- TRUE
if(require(sf, quietly=qt)) {
  ## point pattern dataset in 'sp' format
  data(meuse)
  coordinates(meuse) = ~x+y
  ## convert to 'sf' format
  maas <- st_as_sf(meuse)
  ## convert to 'spatstat' format
  z <- as.ppp(maas)
}
```

---

as.ppp.SpatialPoints *Coercion between point pattern data in the 'sp' and 'spatstat' packages*

Description

Functions to convert between **spatstat**s planar point pattern (ppp) format and **sp**s SpatialPoints and SpatialPointsDataFrame as well as one-way conversion from SpatialGridDataFrame to ppp. S4-style as() coercion can be used as well.

Usage

```
## S3 method for class 'SpatialPoints'
as.ppp(X, W=NULL, ..., fatal=TRUE)
## S3 method for class 'SpatialPointsDataFrame'
as.ppp(X, W=NULL, ..., fatal=TRUE)
as.SpatialPoints.ppp(from)
as.SpatialPointsDataFrame.ppp(from)
as.SpatialGridDataFrame.ppp(from)
```

Arguments

from, X	object to coerce from
W	Window (object of class "owin") for the point pattern.
...	Ignored.
fatal	Logical value specifying whether a conversion error should result in a fatal error.

Details

The main conversion is between **sp**s `SpatialPoints`/`SpatialPointsDataFrame` and **spatstat**s ppp. Conversion between `SpatialGridDataFrame` and ppp should rarely be used; using `as.owin.SpatialGridDataFrame` is more transparent.

Value

A spatial point pattern (object of class `"ppp"` as defined in the **spatstat** package) or an object of class `"SpatialPoints"`, `"SpatialPointsDataFrame"` or `"SpatialGridDataFrame"` as defined in the package **sp**.

Methods

coerce `signature(from = "SpatialPoints", to = "ppp")`

coerce `signature(from = "SpatialPointsDataFrame", to = "ppp")`

coerce `signature(from = "ppp", to = "SpatialGridDataFrame")`

coerce `signature(from = "ppp", to = "SpatialPointsDataFrame")`

coerce `signature(from = "ppp", to = "SpatialPoints")`

Warning

In **spatstat** all spatial objects are assumed to be planar. This means that **spatstat** is not designed to work directly with geographic (longitude and latitude) coordinates. If a **sp** object is declared to have geographic (unprojected) coordinates **maptools** refuses to convert directly to **spatstat** format. Rather, these should be projected first using e.g. `spTransform`. If you know what you are doing, and really want to force coercion, you can overwrite the `proj4string` of the **sp** object with `NA`, `proj4string(x) <- CRS(NA)`, which will fool the system to think that the data is in local planar coordinates. This is probably not a good idea!

Note

The ppp format requires an observation window which is the sampling region. The **sp** formats contain no such information and by default the bounding box of the points is simply used. This is almost never the correct thing to do! Rather, information about the sampling region should be converted into **spatstat**s `owin` format and assigned as the observation window. Usually conversion from ppp to **sp** format simply discards the `owin`. However, `as.SpatialGridDataFrame.ppp` actually first discards the points(!), second checks that the corresponding `owin` is in a grid format (matrix of TRUE/FALSE for inside/outside sampling region), and finally converts the TRUE/FALSE grid to a `SpatialGridDataFrame`.

Author(s)

Edzer Pebesma <edzer.pebesma@uni-muenster.de> and Roger Bivand

Examples

```
## Convert SpatialPointsDataFrame into a marked ppp
data(meuse)
coordinates(meuse) = ~x+y
meuse_ppp <- as(meuse, "ppp")
meuse_ppp # Window is the bounding rectangle
plot(meuse_ppp, which.marks = "zinc")
## Convert SpatialPoints into an unmarked ppp
meuse2 <- as(meuse, "SpatialPoints")
as(meuse2, "ppp")
## Get sampling region in grid format and assign it as observation window
data(meuse.grid)
gridded(meuse.grid) <- ~x+y
mg_owin <- as(meuse.grid, "owin")
Window(meuse_ppp) <- mg_owin
meuse_ppp # Window is now a binary image mask (TRUE/FALSE grid)
plot(meuse_ppp, which.marks = "zinc")
## Convert marked ppp back to SpatialPointsDataFrame
rev_ppp_SPDF <- as.SpatialPointsDataFrame.ppp(meuse_ppp)
summary(rev_ppp_SPDF)
## Convert marked ppp back to SpatialPoints (discarding marks)
rev_ppp_SP <- as.SpatialPoints.ppp(meuse_ppp)
summary(rev_ppp_SP)
## Convert marked ppp back to SpatialGridDataFrame (extracting window grid)
rev_ppp_SGDF <- as.SpatialGridDataFrame.ppp(meuse_ppp)
summary(rev_ppp_SGDF)
```

---

as.psp.SpatialLines *Coercion between sp objects and spatstat psp objects*

Description

Functions to convert between **spatstat**s planar segment pattern (psp) format and various **sp** line formats. S4-style as() coercion can be used as well.

Usage

```
## S3 method for class 'Line'
as.psp(x, ..., window=NULL, marks=NULL, fatal)
## S3 method for class 'Lines'
as.psp(x, ..., window=NULL, marks=NULL, fatal)
## S3 method for class 'SpatialLines'
as.psp(x, ..., window=NULL, marks=NULL,
                  characterMarks=FALSE, fatal)
## S3 method for class 'SpatialLinesDataFrame'
as.psp(x, ..., window=NULL,
                  marks=NULL, fatal)
as.SpatialLines.psp(from)
```

Arguments

x, from	object to coerce from
...	ignored
window	window of class owin as defined in the spatstat package
marks	marks as defined in the spatstat package
characterMarks	default FALSE, if TRUE, do not convert NULL marks to factor from character
fatal	formal coercion argument; ignored

Value

A spatial pattern of line segments (object of class "psp" as defined in the **spatstat** package) or a collection of lines (object of class "SpatialLines" or "SpatialLinesDataFrame" in the package **sp**).

Methods

coerce signature(from = "Line", to = "psp")

coerce signature(from = "Lines", to = "psp")

coerce signature(from = "SpatialLines", to = "psp")

coerce signature(from = "SpatialLinesDataFrame", to = "psp")

coerce signature(from = "psp", to = "SpatialLines")

Warning

In **spatstat** all spatial objects are assumed to be planar. This means that **spatstat** is not designed to work directly with geographic (longitude and latitude) coordinates. If a **sp** object is declared to have geographic (unprojected) coordinates **maptools** refuses to convert directly to **spatstat** format. Rather, these should be projected first using e.g. spTransform. If you know what you are doing, and really want to force coercion, you can overwrite the proj4string of the **sp** object with NA, proj4string(x) <- CRS(NA), which will fool the system to think that the data is in local planar coordinates. This is probably not a good idea!

Author(s)

Edzer Pebesma <edzer.pebesma@uni-muenster.de> and Roger Bivand

Examples

```
data(meuse.riv)
mr <- Line(meuse.riv)
mr_psp <- as(mr, "psp")
mr_psp
plot(mr_psp)
xx_back <- as(mr_psp, "SpatialLines")
plot(xx_back)
```

as.SpatialPolygons.tess *Coercion of spatstat tess object to sp SpatialPolygons object*

Description

This function coerces **spatstat**s tessellation objects of class `tess` to **sp**s `SpatialPolygons` class. S4-style as() coercion works as well.

Usage

```
as.SpatialPolygons.tess(from)
```

Arguments

`from`	**spatstat** object of class `tess` to coerce from

Value

An object of class `"SpatialPolygons"` as defined in the package **sp**.

Methods

coerce `signature(from = "tess", to = "SpatialPolygons")`

Author(s)

Edzer Pebesma <edzer.pebesma@uni-muenster.de> and Roger Bivand

Examples

```
A <- dirichlet(runifpoint(6))
A_sp <- as(A, "SpatialPolygons")
plot(A_sp)
text(coordinates(A_sp), labels=row.names(A_sp), cex=0.6)
```

ppp-class *Virtual Class 'ppp' and Friends*

Description

Virtual S4 class definition for S3 classes in the spatstat package to allow S4-style coercion to these classes

Objects from the Class

A virtual Class: No objects may be created from it.

Author(s)

Edzer Pebesma <edzer.pebesma@uni-muenster.de>

Index