

# Package ‘aspace’

January 2, 2012

**Type** Package

**Title** A collection of functions for estimating centrographic statistics and computational geometries for spatial point patterns

**Version** 3.0

**Date** 2011-03-28

**Author** Randy Bui, Ron N. Buliung, Tarmo K. Remmel

**Maintainer** Ron N. Buliung <ron.buliung@utoronto.ca>

**Description** A collection of functions for computing centrographic statistics (e.g., standard distance, standard deviation ellipse, standard deviation box) for observations taken at point locations. Separate plotting functions have been developed for each measure. Users interested in writing results to ESRI shapefiles can do so by using results from aspace functions as inputs to the `convert.to.shapefile` and `write.shapefile` functions in the `shapefiles` library. The `aspace` library was originally conceived to aid in the analysis of spatial patterns of travel behaviour (see Buliung and Remmel,2008). Major changes in the current version include (1) removal of dependencies on several external libraries (e.g., `gpclib`, `maptools`, `sp`), (2) the separation of plotting and estimation capabilities, (3) reduction in the number of functions, and (4) expansion of analytical capabilities with additional functions for descriptive analysis and visualization (e.g., standard deviation box, centre of minimum distance, central feature).

**License** GPL (>= 2)

**Depends** R (>= 2.12.2), `splancs`, `Hmisc`, `shapefiles`

**LazyData** yes

**Repository** CRAN

**Date/Publication** 2011-03-31 08:55:53

## R topics documented:

aspace-package	2
acos_d	3
activities	4
activities2	5
asin_d	6
as_radians	7
atan_d	8
calc_box	9
calc_sdd	10
calc_sde	12
centre	14
CF	15
CF2PTS	16
CMD	17
cos_d	18
distances	19
mean_centre	20
median_centre	21
plot_box	23
plot_centres	24
plot_sdd	26
plot_sde	27
r.BOX	28
r.SDD	29
r.SDE	30
sin_d	32
tan_d	33
wts	34
<b>Index</b>	<b>35</b>

---

aspace-package	<i>A collection of functions for estimating centrographic statistics and computational geometries for spatial point patterns</i>
----------------	--

---

### Description

A collection of functions for computing centrographic statistics (e.g., standard distance, standard deviation ellipse, standard deviation box) for observations taken at point locations. Separate plotting functions have been developed for each measure. Users interested in writing results to ESRI shapefiles can do so by using results from aspace functions as inputs to the `convert.to.shapefile` and `write.shapefile` functions in the `shapefiles` library. The `aspace` library was originally conceived to aid in the analysis of spatial patterns of travel behaviour (see Buliung and Rummel, 2008). Major changes in the current version include (1) removal of dependencies on several external libraries (e.g., `gpclib`, `maptools`, `sp`), (2) the separation of plotting and estimation capabilities, (3) reduction in the number of functions, and (4) expansion of analytical capabilities with additional functions

for descriptive analysis and visualization (e.g., standard deviation box, centre of minimum distance, central feature).

### Details

Package: aspace  
Type: Package  
Version: 3.0  
Date: 2011-03-28  
License: GPL ( $\geq 2.0$ )

### Author(s)

Randy Bui, Ron N. Buliung, Tarmo K. Rimmel

### References

Bachi, R. 1963. Standard distance measures and related methods for spatial analysis. Papers of the Regional Science Association 10: 83-132.

Buliung, R.N. and Rimmel, T. (2008) Open source, spatial analysis, and activity travel behaviour research: capabilities of the aspace package. Journal of Geographical Systems, 10: 191-216.

Buliung, R.N. and Kanaroglou, P.S. (2006) Urban form and household activity-travel behaviour. Growth and Change, 37: 174-201.

Ebdon, D. 1988. Statistics in Geography 2nd Edition. Oxford UK: Blackwell.

Levine, N. 2002. CrimeStat II: A Spatial Statistics Program for the Analysis of Crime Incident Locations (version 2.0) Houston TX/National Institute of Justice, Washington DC: Ned Levine & Associates.

---

acos\_d

*Compute inverse cosine with angle given in degrees*

---

### Description

Provides the functionality of acos, but for input angles measured in degrees (not radians).

### Usage

acos\_d(theta = 0)

**Arguments**

theta            A numeric angular measurement in degrees from north.

**Details**

Since the R default is to compute trigonometric functions on angular measurements stored in radians, this simple function performs the conversion from degrees, reducing the need to do so a priori, outside the function.

**Value**

Returns a numeric value for the inverse cosine of the specified angular measurement

**Note**

To reduce the need for unit conversions prior to calling trigonometric functions, this function accepts input in angular degrees rather than radians. Depending on the data source, this function may be preferred to the existing version requiring input in angular radians.

**Author(s)**

Tarmo K. Remmel

**See Also**

[sin\\_d](#), [cos\\_d](#), [tan\\_d](#), [asin\\_d](#), [atan\\_d](#)

**Examples**

```
acos_d(theta = 90)
```

---

activities

*Demo Data: x and y coordinates of 10 specified point locations*

---

**Description**

This is a simple two-column data frame (or matrix) containing x,y coordinates for a series of point locations. These data mimic UTM coordinates such that the first column contains Easting (x), and the second Northing (y) coordinates for the set of unique points.

**Usage**

```
data(activities)
```

**Format**

A data frame with 10 observations on the following 2 variables.

col1 A numeric vector of x-coordinates

col2 A numeric vector of y-coordinates

**Details**

The coordinates of the points must have the same units and projection as the specified center.

**Source**

This demonstration data has been manufactured for illustrative purposes only.

**Examples**

```
data(activities)
str(activities)
plot(activities)
```

---

activities2

*Demo Data: x and y coordinates of 10 specified point locations*

---

**Description**

This is a simple two-column data frame (or matrix) containing x,y coordinates for a series of point locations. These data mimic UTM coordinates such that the first column contains Easting (x), and the second Northing (y) coordinates for the set of unique points.

**Usage**

```
data(activities2)
```

**Format**

A data frame with 10 observations on the following 2 variables.

col1 A numeric vector of x-coordinates

col2 A numeric vector of y-coordinates

**Details**

The coordinates of the points must have the same units and projection as the specified center.

**Source**

This demonstration data has been manufactured for illustrative purposes only.

**Examples**

```
data(activities2)
str(activities2)
plot(activities2)
```

---

`asin_d`*Compute inverse sine with angle given in degrees*

---

**Description**

Provides the functionality of `asin`, but for input angles measured in degrees (not radians).

**Usage**

```
asin_d(theta = 0)
```

**Arguments**

`theta`            A numeric angular measurement in degrees from north.

**Details**

Since the R default is to compute trigonometric functions on angular measurements stored in radians, this simple function performs the conversion from degrees, reducing the need to do so a priori, outside the function.

**Value**

Returns a numeric value for the inverse sine of the specified angular measurement.

**Note**

To reduce the need for unit conversions prior to calling trigonometric functions, this function accepts input in angular degrees rather than radians. Depending on the data source, this function may be preferred to the existing version requiring input in angular radians.

**Author(s)**

Tarmo K. Rimmel

**See Also**

[sin\\_d](#), [cos\\_d](#), [tan\\_d](#), [acos\\_d](#), [atan\\_d](#)

**Examples**

```
asin_d(theta = 90)
```

---

as_radians	<i>Converts degrees to radians</i>
------------	------------------------------------

---

**Description**

This function converts an angular measure stored in degrees to radians. This is an alternative to the `rad` function available in the package `circular`.

**Usage**

```
as_radians(theta = 0)
```

**Arguments**

`theta` A numeric angular measurement in degrees from north.

**Details**

Achieves a very simple conversion with a convenient function call.

**Value**

Returns a numeric value for an angle in radians that is equivalent to the input `theta` in degrees.

**Note**

The purpose of this function is to reduce computer code clutter when using angular measurements in R. The simple function call ensures that degree to radian conversions are completed consistently and accurately. Since trigonometric functions in R require angular measures in radians rather than degrees, this simple function can be used for simple angular unit conversion.

**Author(s)**

Tarmo K. Rimmel

**See Also**

[sin\\_d](#), [cos\\_d](#), [tan\\_d](#), [asin\\_d](#), [acos\\_d](#), [atan\\_d](#)

**Examples**

```
as_radians(theta = 90)
```

---

`atan_d`*Compute inverse tangent with angle given in degrees*

---

**Description**

Provides the functionality of `atan`, but for input angles measured in degrees (not radians).

**Usage**

```
atan_d(theta = 0)
```

**Arguments**

`theta`            A numeric angular measurement in degrees from north.

**Details**

Since the R default is to compute trigonometric functions on angular measurements stored in radians, this simple function performs the conversion from degrees, reducing the need to do so a priori, outside the function.

**Value**

Returns a numeric value for the inverse tangent of the specified angular measurement.

**Note**

To reduce the need for unit conversions prior to calling trigonometric functions, this function accepts input in angular degrees rather than radians. Depending on data, this function may be preferred to the existing version requiring input in angular radians.

**Author(s)**

Tarmo K. Remmel

**See Also**

[sin\\_d](#), [cos\\_d](#), [tan\\_d](#), [asin\\_d](#), [acos\\_d](#)

**Examples**

```
atan_d(theta = 90)
```

calc\_box

*Calculate the Standard Deviation Box***Description**

The orthogonal dispersion of a set of points can be described using the standard deviation of the x- and y-coordinates of a set of point observations. The orthogonal dispersion can then be visualized with a Standard Deviation Box. This function computes the properties of the Standard Deviation Box (SD Box) from a set of point observations.

**Usage**

```
calc_box(id=1, filename="BOX_Output.txt", centre.xy=NULL, calccentre=TRUE,
weighted=FALSE, weights=NULL, points=activities, verbose=FALSE)
```

**Arguments**

id	A unique integer to identify a SD Box
filename	A string indicating the ASCII textfile where the box coordinates will be written
centre.xy	A vector of length 2, containing the x- and y-coordinates of the geographic centre of the SD Box
calccentre	Boolean: Set to TRUE if the mean center is to be calculated
weighted	Boolean: Set to TRUE if the weighted mean center is to be computed with weighted coordinates
weights	Weights applied to point observations, number of weights should equal the number of observations
points	A 2-column matrix or data frame containing the set of point observations input to the calc_box function
verbose	Boolean: Set to TRUE if extensive feedback is desired on the standard output

**Details**

Use the boxloc (coordinates) and boxatt(attributes) to produce shapefiles using the convert.to.shapefile and write.shapefile from the shapefiles library

**Value**

The returned result is a list:

id	Identifier for the SD Box shape - it should be unique
calccentre	Boolean: TRUE if the mean centre was estimated
weighted	Boolean: TRUE if the weighted mean centre was estimated
CENTRE.x	X-coordinate of the centre
CENTRE.y	Y-coordinate of the centre

SD.x	Orthogonal standard deviation in the x-axis
SD.y	Orthogonal standard deviation in the y-axis
Box.area	Area of the standard deviation box
NW.coord	North-west coordinates of SD Box
NE.coord	North-east coordinates of SD Box
SW.coord	South-west coordinates of SD Box
SE.coord	South-east coordinates of SD Box

**Note**

Results are stored in the `r.BOX` object (required for `plot_box`). This function can be used on its own (once) or repetitively in a loop to process grouped point data stored in a larger table. When used repetitively, be sure to increment the `id` parameter to ensure that each SD BOX has a unique identifier. The output ASCII coordinate file can be further processed using the `shapefiles` package to generate an ESRI Shapefile for SD BOX polygons.

**Author(s)**

Randy Bui, Ron N. Buliung, Tarmo K. Rimmel

**See Also**

[plot\\_box](#), [calc\\_sde](#), [calc\\_sdd](#), [wtd.var](#)

**Examples**

```
## BOX example
calc_box(id=1, filename="BOX_Output.txt", centre.xy=NULL, calccentre=TRUE,
weighted=FALSE, weights=NULL, points=activities, verbose=FALSE)

## SD Box to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(boxloc,boxatt,"id",5)
## write.shapefile(shp, "BOX_Shape", arcgis=T)
```

---

calc\_sdd

*Calculate the Standard Distance Deviation (Standard Distance)*

---

**Description**

This function computes the Standard Distance Deviation (SDD) or Standard Distance from a set of points.

**Usage**

```
calc_sdd(id=1, filename="SDD_Output.txt", centre.xy=NULL, calccentre=TRUE,
weighted=FALSE, weights=NULL, points=activities, verbose=FALSE)
```

**Arguments**

id	A unique integer to identify a SDD estimate
filename	A string indicating the ASCII textfile where shape coordinates will be written
centre.xy	A vector of length 2, containing the x- and y-coordinates of the SDD centre
calccentre	Boolean: Set to TRUE if the mean center is to be calculated
weighted	Boolean: Set to TRUE if the weighted mean center is to be computed with weighted coordinates
weights	Weights applied to point observations, number of weights should equal the number of observations
points	A 2-column matrix or data frame containing the set of point observations input to the calc_sdd function
verbose	Boolean: Set to TRUE if extensive feedback is desired on the standard output

**Details**

Use the `sddloc` (coordinates) and `sddatt`(attributes) to produce shapefiles using the `convert.to.shapefile` and `write.shapefile` from the `shapefiles` library

**Value**

The result is a list of terms:

id	Identifier for the SDD shape - it should be unique
calccentre	Boolean: TRUE if mean centre is computed
weighted	Boolean: TRUE if the weighted mean centre is to be used instead
CENTRE.x	X-coordinate of the centre
CENTRE.y	Y-coordinate of the centre
SDD.radius	SDD value, radius of the SDD
SDD.area	Area of the SDD circle

**Note**

Results are stored in the `r.SDD` object (required for `plot_sdd`). This function can be used on its own (once) or repetitively in a loop to process grouped point data stored in a larger table. When used repetitively, be sure to increment the `id` parameter to ensure that each SDD has a unique identifier. The output ASCII coordinate file can be further processed using the `shapefiles` package to generate an ESRI Shapefile for SDD polygons.

**Author(s)**

Randy Bui, Ron Buliung, Tarmo K. Rimmel

**See Also**

[plot\\_sdd](#), [calc\\_sde](#), [calc\\_box](#)

## Examples

```
## SDD example
calc_sdd(id=1, filename="SDD_Output.txt", centre.xy=NULL, calccentre=TRUE,
weighted=FALSE, weights=NULL, points=activities, verbose=FALSE)

## SDD to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(sddloc,sddatt,"id",5)
## write.shapefile(shp, "SDD_Shape", arcgis=T)
```

---

calc\_sde

*Calculate the Standard Deviation Ellipse*

---

## Description

This function computes the Standard Deviation Ellipse (SDE) from a set of points. The SDE is a centographic measure used to characterize the dispersion of point observations along two orthogonal axes. The SDE also captures directional bias in a spatial point pattern, the ellipse will be oriented in the direction of maximum dispersion.

## Usage

```
calc_sde(id=1, filename="SDE_Output.txt", centre.xy=NULL, calccentre=TRUE,
weighted=FALSE, weights=NULL, points=activities, verbose=FALSE)
```

## Arguments

id	A unique integer to identify the shape
filename	A string indicating the ASCII textfile where shape coordinates will be written
centre.xy	A vector of length 2, containing the x- and y-coordinates of the SDE centre (Planar Coordinates Only!)
calccentre	Boolean: Set to TRUE if the mean center is to be calculated
weighted	Boolean: Set to TRUE if the weighted mean center is to be computed with weighted coordinates
weights	Weights applied to point observations, number of weights should equal the number of observations
points	A 2-column matrix or data frame containing point coordinates
verbose	Boolean: Set to TRUE if extensive feedback is desired on the standard output

## Details

Use the `sdeloc` (coordinates) and `sdeatt`(attributes) to produce shapefiles using the `convert.to.shapefile` and `write.shapefile` from the `shapefiles` library

**Value**

The returned result is a list:

id	Identifier for the SDE shape - it should be unique
calccentre	Boolean: TRUE if mean centre is computed
weighted	Boolean: TRUE if the weighted mean centre is to be used instead
CENTRE.x	X-coordinate of the centre
CENTRE.y	Y-coordinate of the centre
Sigma.x	Half-length of axis along x-axis
Sigma.y	Half-length of axis along y-axis
Major	String indicating which axis is the major elliptical axis
Minor	String indicating which axis is the minor elliptical axis
Theta	Rotation angle in degrees
Eccentricity	A measure of eccentricity (i.e., the flatness of the ellipse)
Area.sde	Area of the SDE
TanTheta	Trigonometric result
SinTheta	Trigonometric result
CosTheta	Trigonometric result
SinThetaCosTheta	Trigonometric result
Sin2Theta	Trigonometric result
Cos2Theta	Trigonometric result
ThetaCorr	Corrected theta angle for rotation of major axis from north

**Note**

Results are stored in the `r.SDE` object (required for `plot_sde`). This function can be used on its own (once) or repetitively in a loop to process grouped point data stored in a larger table. When used repetitively, be sure to increment the `id` parameter to ensure that each SDE has a unique identifier. The output ASCII coordinate file can be further processed using the `shapefiles` package to generate an ESRI Shapefile for SDE polygons..

**Author(s)**

Randy Bui, Ron N. Buliung, Tarmo K. Rimmel

**References**

See chapter 4 of the documentation manual for CrimeStat at <http://www.icpsr.umich.edu/CRIMESTAT/> and Ebdon, D. 1987. *Statistics in geography*. 2nd edition. New York, NY Basil Blackwell Ltd. 232 p.

**See Also**

[plot\\_sde](#), [calc\\_sdd](#), [calc\\_box](#), [gridpts](#)

**Examples**

```
## SDE example
calc_sde(id=1, filename="SDE_Output.txt", centre.xy=NULL, calccentre=TRUE,
weighted=FALSE, weights=NULL, points=activities, verbose=FALSE)

## SDE to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(sdeloc,sdeatt,"id",5)
## write.shapefile(shp, "SDE_Shape", arcgis=T)
```

---

centre

*Demo Data: Coordinates of a single source, centre, location*

---

**Description**

This is a simple two-element vector containing x,y coordinates for a source or central location associated with a spatial point pattern. In this example, the center location represents a point of importance in an individuals daily activity pattern. Surrounding point locations are places physically contacted by an individual during a particular time interval. Demonstration data mimics UTM coordinates such that the first element represents Easting (x), and the second, Northing (y).

**Usage**

```
data(centre)
```

**Format**

The format is a two-element vector of numeric entries.

**Details**

The coordinates of the center must have the same units and projection as the remaining point observations.

**Source**

This demonstration data has been manufactured for illustrative purposes only.

**Examples**

```
data(centre)
str(centre)
plot(centre)
```

**Description**

Identifies the central feature within a set of point locations.

**Usage**

```
CF(id=1, filename="CF_Output.txt", points=activities)
```

**Arguments**

id	A unique integer to identify the CF
filename	A string indicating the ASCII textfile where the central feature coordinates will be written
points	A 2-column matrix or data frame containing the set of point observations

**Details**

Use the `cfloc` (coordinates) and `cfdata`(attributes) to produce shapefiles using the `convert.to.shapefile` and `write.shapefile` from the `shapefiles` library

**Value**

The result is a list of terms:

id	Identifier for the central feature - it should be unique
CF.x	X-coordinate of the central feature
CF.y	Y-coordinate of the central feature

**Note**

Results are stored in the `r.CF` object and can be passed through plotting functions. This function can also be used repetitively within a loop to compute multiple CF centres from different datasets.

**Author(s)**

Randy Bui, Ron Buliung

**See Also**

[mean\\_centre](#), [CMD](#), [median\\_centre](#)

**Examples**

```
## CF example
CF(id=1, filename="CF_Output.txt", points=activities)

## CF to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(cfloc,cfatt,"id",5)
## write.shapefile(shp, "CF_Shape", arcgis=T)
```

CF2PTS

*Central feature between 2 point patterns (CF2PTS) Calculator***Description**

Central feature of point2 within point1. Identifies the central feature as the point location in the first pattern that has the smallest cumulative distance to features in a second point pattern.

**Usage**

```
CF2PTS(id=1, filename="CF2PTS_Output.txt", points1=activities, points2=activities2)
```

**Arguments**

id	A unique integer to identify the CF2PTS
filename	A string indicating the ASCII textfile where the central feature coordinates will be written
points1	A 2-column matrix or data frame containing the set of point observations
points2	A 2-column matrix or data frame containing the set of point observations

**Details**

Use the `cf2ptsloc` (coordinates) and `cf2ptsatt` (attributes) to produce shapefiles using the `convert.to.shapefile` and `write.shapefile` from the `shapefiles` library

**Value**

The result is a list of terms:

id	Identifier for the central feature - it should be unique
CF2PTS.x	X-coordinate of the central feature
CF2PTS.y	Y-coordinate of the central feature

**Note**

Results are stored in the `r.CF2PTS` object and can be passed through plotting functions. This function can also be used repetitively within a loop to compute multiple CF2PTS centres from different datasets.

**Author(s)**

Randy Bui, Ron Buliung

**See Also**

[CF](#), [CMD](#), [median\\_centre](#)

**Examples**

```
## CF2PTS example
CF2PTS(id=1, filename="CF2PTS_Output.txt", points1=activities, points2=activities2)

## CF2PTS to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(cf2ptsloc,cf2ptsatt,"id",5)
## write.shapefile(shp, "CF2PTS_Shape", arcgis=T)
```

---

CMD

*Centre of Minimum Distance (CMD) Calculator*

---

**Description**

Compute the CMD within a set of point locations.

**Usage**

```
CMD(id=1, filename="CMD_Output.txt", dist=100,
points=activities)
```

**Arguments**

id	A unique integer to identify the CMD
filename	A string indicating the ASCII textfile where centre coordinates will be written
dist	Hold distance value between i and ith iterations
points	A 2-column matrix or data frame containing the set of point observations

**Details**

Use the `cmdloc` (coordinates) and `cmdatt` (attributes) to produce shapefiles using the `convert.to.shapefile` and `write.shapefile` from the `shapefiles` library

**Value**

The result is a list of terms:

id	Identifier for the CMD - it should be unique
CMD.x	X-coordinate of the CMD
CMD.y	Y-coordinate of the CMD
distance	Hold distance value between i and ith iterations (metres)
Number of Cells	Hold number of cells in each grid created for each iteration

**Note**

Results are stored in the r.CMD object and can be passed through plotting functions. The dist parameter specifies the distance threshold between i and ith iterations. This function can also be used repetitively within a loop to compute multiple CMD centres from different datasets.

**Author(s)**

Randy Bui, Ron Buliung

**See Also**

[mean\\_centre](#), [median\\_centre](#), [CF](#)

**Examples**

```
## CMD example
CMD(id=1, filename="CMD_Output.txt", dist=100,
points=activities)

## CMD to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(cmdloc,cmdatt,"id",5)
## write.shapefile(shp, "CMD_Shape", arcgis=T)
```

---

cos\_d

*Compute cosine with angle given in degrees*

---

**Description**

Provides the functionality of cos, but for input angles measured in degrees (not radians).

**Usage**

```
cos_d(theta = 0)
```

**Arguments**

theta            A numeric angular measurement in degrees from north.

**Details**

Since the R default is to compute trigonometric functions on angular measurements stored in radians, this simple function performs the conversion from degrees, reducing the need to do so a priori, outside the function.

**Value**

Returns a numeric value for the cosine of the specified angular measurement

**Note**

To reduce the need for unit conversions prior to calling trigonometric functions, this function accepts input in angular degrees rather than radians. Depending on data, this function may be preferred to the existing version requiring input in angular radians.

**Author(s)**

Tarmo K. Remmel

**See Also**

[sin\\_d](#), [tan\\_d](#), [asin\\_d](#), [acos\\_d](#), [atan\\_d](#)

**Examples**

```
cos_d(theta = 90)
```

---

distances

*Multiple Euclidean distance calculator*

---

**Description**

Compute distances from a source location (point) to a series of destination locations (points).

**Usage**

```
distances(centre.xy = centre, destmat = activities, verbose = FALSE)
```

**Arguments**

centre.xy	Two-element vector containing x,y coordinates of the source location
destmat	Two-column matrix or data frame containing x,y coordinates of the activity locations
verbose	Boolean: Set to T if verbose output is desired

**Details**

Distance computations are strictly Euclidean between the source point and each destination point.

**Value**

A vector of distances, where each element corresponds to one of the distance between the source point and a destination (one row) from the destinations matrix.

**Note**

The order of distances in the output vector corresponds to the order of destination points in the destinations object starting at row = 1 through row = n.

**Author(s)**

Tarmo K. Remmel

**Examples**

```
data(centre)
data(activities)
distances(centre.xy=centre, destmat=activities, verbose=FALSE)
```

---

mean\_centre

*Mean Centre Calculator*

---

**Description**

Compute the mean centre from a series of point locations.

**Usage**

```
mean_centre(id=1, filename="mean_centre_Output.txt",
            weighted=FALSE, weights=NULL, points=activities)
```

**Arguments**

id	A unique integer to identify the mean centre
filename	A string indicating the ASCII textfile where centre coordinates will be written
weighted	Boolean: Set to TRUE if the weighted mean center is to be computed with weighted coordinates
weights	Weights applied to point observations, number of weights should equal the number of observations
points	A 2-column matrix or data frame containing the set of point observations

**Details**

Use the meanloc (coordinates) and meanatt(attributes) to produce shapefiles using the convert.to.shapefile and write.shapefile from the shapefiles library

**Value**

The result is a list of terms:

id	Identifier for the mean centre - it should be unique
weighted	Boolean: TRUE if the weighted mean centre is to be used instead
weights	Weights applied to point observations
CENTRE.x	X-coordinate of the mean centre
CENTRE.y	Y-coordinate of the mean centre

**Note**

Results are stored in the `r.mean` object and can be passed through plotting functions. This function can also be used repetitively within a loop to compute multiple mean centres from different datasets.

**Author(s)**

Randy Bui, Ron Buliung

**See Also**

[median\\_centre](#), [CMD](#), [CF](#)

**Examples**

```
## Mean centre example
mean_centre(id=1, filename="mean_centre_Output.txt",
weighted=FALSE, weights=NULL, points=activities)

## Mean centre to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(meanloc,meanatt,"id",5)
## write.shapefile(shp, "Mean_Shape", arcgis=T)
```

---

median\_centre

*Median Centre Calculator*

---

**Description**

Compute the median centre from a series of point locations.

**Usage**

```
median_centre(id=1, filename="median_centre_Output.txt",
points=activities)
```

**Arguments**

id	A unique integer to identify the median centre
filename	A string indicating the ASCII textfile where centre coordinates will be written
points	A 2-column matrix or data frame containing the set of point observations

**Details**

Use the `medianloc` (coordinates) and `medianatt`(attributes) to produce shapefiles using the `convert.to.shapefile` and `write.shapefile` from the `shapefiles` library

**Value**

The result is a list of terms:

id	Identifier for the median centre - it should be unique
median.x	X-coordinate of the median centre
median.y	Y-coordinate of the median centre

**Note**

Results are stored in the `r.median` object and can be passed through plotting functions. This function can also be used repetitively within a loop to compute multiple median centres from different datasets.

**Author(s)**

Randy Bui, Ron Buliung

**See Also**

[mean\\_centre](#), [CMD](#), [CF](#)

**Examples**

```
## Median centre example
median_centre(id=1, filename="median_centre_Output.txt",
points=activities)

## Median centre to shapefile example (exclude the comments below to run script)
## shp <- convert.to.shapefile(medianloc,medianatt,"id",5)
## write.shapefile(shp, "Median_Shape", arcgis=T)
```

---

plot\_box

*Plot the Standard Distance Box*


---

### Description

This function plots the standard deviation of x- and y-coordinates as a box, with the edges set, respectively, to the standard deviation of the x- and y-coordinates.

### Usage

```
plot_box(plotnew=TRUE, plothv=FALSE, plotweightedpts=FALSE,
weightedpts.col='black', weightedpts.pch=19, plotpoints=TRUE,
points.col='black', points.pch=1, plotcentre=TRUE, centre.col='black',
centre.pch=19, titletxt="Title", xaxis="Easting (m)",
yaxis="Northing (m)", box.col='black', box.lwd=2, jpeg=FALSE, ...)
```

### Arguments

plotnew	Boolean: Set to TRUE to create a new plot. Set to FALSE to overlay current plot.
plothv	Boolean: Set to TRUE if the orthogonal N-S, E-W axes are to be plotted through the centre
plotweightedpts	Boolean: Set to TRUE if the weighted point observations are to be plotted
weightedpts.col	Specify a colour for the weighted point observations
weightedpts.pch	Specify a plotting symbol for the weighted point observations
plotpoints	Boolean: Set to TRUE if the point observations are to be plotted
points.col	Specify a colour for the point observations
points.pch	Specify a plotting symbol for the point observations
plotcentre	Boolean: Set to TRUE if the mean/weighted/user-defined centre is to be plotted
centre.col	Specify a colour for the centre
centre.pch	Specify a plotting symbol for the centre
titletxt	A string to indicate the title for the plot
xaxis	A string to label the x-axis of the plot
yaxis	A string to label the y-axis of the plot
box.col	Specify a line colour for the SD Box
box.lwd	Specify a line width for the SD Box
jpeg	Boolean: Set to TRUE if the plot should be saved in JPEG format
...	Arguments to be passed to graphical parameters

**Details**

The r.BOX object (generated using the calc\_box function) is required to plot an SD Box.

**Author(s)**

Randy Bui, Ron N. Buliung, Tarmo K. Remmel

**See Also**

[plot\\_sdd](#), [plot\\_sde](#)

**Examples**

```
plot_box(plotnew=TRUE, plothv=FALSE, plotweightedpts=FALSE,
plotpoints=TRUE, plotcentre=TRUE, titletxt="Title",
xaxis="Easting (m)", yaxis="Northing (m)")
```

---

plot\_centres

*Plot centres*

---

**Description**

This function plots various centre of a set of point observations.

**Usage**

```
plot_centres(plotnew=FALSE, plotsDE=FALSE, xaxis="Easting (m)", yaxis="Northing (m)",
robject=NULL, plotweightedpts=FALSE, weightedpts.col='black', weightedpts.pch=19,
plotpoints=TRUE, points.col='black', points.pch=1, plotcentre=FALSE, centre.col='black',
centre.pch=19, plotcentral=FALSE, central.col='green', central.pch=19,
plotCF2PTS=FALSE, CF2PTS.col='orange', CF2PTS.pch=19, plotmedian=FALSE,
median.col='blue', median.pch=17, plotCMD=FALSE, CMD.col='red', CMD.pch=17, ...)
```

**Arguments**

plotnew	Boolean: Set to TRUE to create a new plot. Set to FALSE to overlay current plot.
plotsDE	Boolean: Set to TRUE if the centres for the SDE are to be plotted
xaxis	A string to label the x-axis of the plot
yaxis	A string to label the y-axis of the plot
robject	Specify the results object from the computation function. Can be either r.SDD, r.SDE, or r.BOX.
plotweightedpts	Boolean: Set to TRUE if the weighted point observations are to be plotted
weightedpts.col	Specify a colour for the weighted point observations

weightedpts.pch	Specify a plotting symbol for the weighted point observations
plotpoints	Boolean: Set to TRUE if the point observations are to be plotted
points.col	Specify a colour for the point observations
points.pch	Specify a plotting symbol for the point observations
plotcentre	Boolean: Set to TRUE if the mean/weighted/user-defined centre is to be plotted
centre.col	Specify a colour for the centre
centre.pch	Specify a plotting symbol for the centre
plotcentral	Boolean: Set to TRUE if the central feature is to be highlighted
central.col	Specify a colour for the central feature
central.pch	Specify a plotting symbol for the central feature
plotCF2PTS	Boolean: Set to TRUE if the central feature between 2 point patterns is to be highlighted
CF2PTS.col	Specify a colour for the central feature
CF2PTS.pch	Specify a plotting symbol for the central feature
plotmedian	Boolean: Set to TRUE if the median centre is to be plotted
median.col	Specify a colour for the median centre
median.pch	Specify a plotting symbol for the median centre
plotCMD	Boolean: Set to TRUE if the centre of minimum distance is to be plotted
CMD.col	Specify a colour for the centre of minimum distance
CMD.pch	Specify a plotting symbol for the centre of minimum distance
...	Arguments to be passed to graphical parameters

### Details

The results object, for example, `r.SDD` object (generated in `calc_sdd` function) is required to plot the centres for the SDD.

### Author(s)

Randy Bui, Ron N. Buliung, Tarmo K. Rimmel

### See Also

[plot\\_sde](#), [plot\\_box](#)

### Examples

```
plot_centres(plotnew=FALSE, plotSDE=FALSE, robject=NULL, plotweightedpts=FALSE,
xaxis="Easting (m)", yaxis="Northing (m)",
weightedpts.col='black', weightedpts.pch=19, plotpoints=TRUE,
points.col='black', points.pch=1, plotcentre=FALSE, centre.col='black',
centre.pch=19, plotcentral=FALSE, central.col='green', central.pch=19,
plotCF2PTS=FALSE, CF2PTS.col='orange', CF2PTS.pch=19,
plotmedian=FALSE, median.col='blue', median.pch=17, plotCMD=FALSE,
CMD.col='red', CMD.pch=17)
```

---

plot\_sdd

*Plot the Standard Distance Deviation (Standard Distance)*


---

### Description

This function plots the SDD as a circle with radius (standard distance), centred on a mean/weighted-mean/user-defined centre of a set of point observations.

### Usage

```
plot_sdd(plotnew=TRUE, plothv=FALSE, plotweightedpts=FALSE,
weightedpts.col='black', weightedpts.pch=19, plotpoints=TRUE,
points.col='black', points.pch=1, plotcentre=TRUE, centre.col='black',
centre.pch=19, titletxt="Title", xaxis="Easting (m)",
yaxis="Northing (m)", sdd.col='black', sdd.lwd=2, jpeg=FALSE, ...)
```

### Arguments

plotnew	Boolean: Set to TRUE to create a new plot. Set to FALSE to overlay current plot.
plothv	Boolean: Set to TRUE if the orthogonal N-S, E-W axes are to be plotted through the centre
plotweightedpts	Boolean: Set to TRUE if the weighted point observations are to be plotted
weightedpts.col	Specify a colour for the weighted point observations
weightedpts.pch	Specify a plotting symbol for the weighted point observations
plotpoints	Boolean: Set to TRUE if the point observations are to be plotted
points.col	Specify a colour for the point observations
points.pch	Specify a plotting symbol for the point observations
plotcentre	Boolean: Set to TRUE if the mean/weighted/user-defined centre is to be plotted
centre.col	Specify a colour for the centre
centre.pch	Specify a plotting symbol for the centre
titletxt	A string to indicate the title on the plot
xaxis	A string to label the x-axis of the plot
yaxis	A string to label the y-axis of the plot
sdd.col	Specify a line colour for the SDD circle
sdd.lwd	Specify a line width for the SDD circle
jpeg	Boolean: Set to TRUE if the plot should be saved in JPEG format
...	Arguments to be passed to graphical parameters

**Details**

The r.SDD object (generated in calc\_sdd function) is required to plot the SDD circle.

**Author(s)**

Randy Bui, Ron N. Buliung, Tarmo K. Remmel

**See Also**

[plot\\_sde](#), [plot\\_box](#)

**Examples**

```
plot_sdd(plotnew=TRUE, plothv=FALSE, plotweightedpts=FALSE,
plotpoints=TRUE, plotcentre=TRUE, titletxt="Title",
xaxis="Easting (m)", yaxis="Northing (m)")
```

---

plot\_sde

*Plot the Standard Deviation Ellipse*


---

**Description**

This function plots the SDE as an ellipse centred on the mean/weighted/user-defined centre of a set of point observations. The plot characterizes the dispersion of point observations along two orthogonal axes.

**Usage**

```
plot_sde(plotnew=TRUE, plotSDEaxes=FALSE, plotweightedpts=FALSE,
weightedpts.col='black', weightedpts.pch=19, plotpoints=TRUE,
points.col='black', points.pch=1, plotcentre=TRUE, centre.col='black',
centre.pch=19, titletxt="Title", xaxis="Easting (m)",
yaxis="Northing (m)", sde.col='black', sde.lwd=2, jpeg=FALSE, ...)
```

**Arguments**

plotnew	Boolean: Set to TRUE to create a new plot. Set to FALSE to overlay current plot.
plotSDEaxes	Boolean: Set to TRUE if the orthogonal axes through the centroid are to be plotted
plotweightedpts	Boolean: Set to TRUE if the weighted point observations are to be plotted
weightedpts.col	Specify a colour for the weighted point observations
weightedpts.pch	Specify a plotting symbol for the weighted point observations

plotpoints	Boolean: Set to TRUE if the point observations are to be plotted
points.col	Specify a colour for the point observations
points.pch	Specify a plotting symbol for the point observations
plotcentre	Boolean: Set to TRUE if the mean/weighted/user-defined centre is to be plotted
centre.col	Specify a colour for the centre
centre.pch	Specify a plotting symbol for the centre
titletxt	A string to indicate the title on the plot
xaxis	A string to label the x-axis of the plot
yaxis	A string to label the y-axis of the plot
sde.col	Specify a line colour for the SDE circle
sde.lwd	Specify a line width for the SDE circle
jpeg	Boolean: Set to TRUE if the plot should be saved in JPEG format
...	Arguments to be passed to graphical parameters

### Details

The r.SDE object (generated in `calc_sde` function) is required to plot the SDE circle.

### Author(s)

Randy Bui, Ron N. Buliung, Tarmo K. Remmel

### See Also

[plot\\_sdd](#), [plot\\_box](#)

### Examples

```
plot_sde(plotnew=TRUE, plotSDEaxes=FALSE, plotweightedpts=FALSE,
plotpoints=TRUE, plotcentre=TRUE, titletxt="Title",
xaxis="Easting (m)", yaxis="Northing (m)")
```

---

r.BOX

*Demo Data: Standard Deviation Box Output Object*

---

### Description

Results from the Standard Deviation Box Calculator (`calc_box`) are stored in a list object. This object is required for the plot function (`plot_box`).

### Usage

```
data(r.BOX)
```

**Format**

The list object contains the following results:

**id** Identifier for the SD box

**points** a simple two-column data frame (or matrix) containing x,y coordinates for a series of point locations.

**calcentre** Boolean: Indicates whether the mean centre was computed

**weighted** Boolean: TRUE if the weighted mean centre is to be used instead

**weights** Weights applied to point observations

**CENTRE.x** Actual, used x-coordinate of centre

**CENTRE.y** Actual, used y-coordinate of centre

**SDD** Standard deviation distance value

**SDx** Orthogonal standard deviation in x-direction

**SDy** Orthogonal standard deviation in y-direction

**Box.area** Area of orthogonal standard deviation box

**NW.coord** Coordinates of the north-west extent of the SD Box

**NE.coord** Coordinates of the north-east extent of the SD Box

**SW.coord** Coordinates of the south-west extent of the SD Box

**SE.coord** Coordinates of the south-east extent of the SD Box

**Details**

The coordinates of the points must have the same units and projection as the specified center.

**Source**

This demonstration data has been manufactured for illustrative purposes only.

**Examples**

```
data(r.BOX)
str(r.BOX)
```

---

r.SDD

*Demo Data: Standard Deviation Distance Output Object*

---

**Description**

Results from the Standard Deviation Distance Calculator (`calc_sdd`) are stored in a list object. This object is required for the plot function (`plot_sdd`).

## Format

The list object contains the following results:

**id** Identifier for the SDD estimation - it should be unique

**points** a simple two-column data frame (or matrix) containing x,y coordinates for a series of point locations.

**coordsSDD** coordsSDD value, coordinates of the SDD

**SDD** SDD value, radius of the SDD

**calcentre** Boolean: TRUE if mean centre is computed

**weighted** Boolean: TRUE if the weighted mean centre is to be used instead

**weights** Weights applied to point observations

**CENTRE.x** X-coordinate of the centre

**CENTRE.y** Y-coordinate of the centre

**SDD.area** Area of the SDD circle

## Details

The coordinates of the points must have the same units and projection as the specified center.

## Source

This demonstration data has been manufactured for illustrative purposes only.

## Examples

```
data(r.SDD)
str(r.SDD)
```

---

r.SDE

*Demo Data: Standard Deviation Ellipse Output Object*

---

## Description

Results from the Standard Deviation Ellipse Calculator (`calc_sde`) are stored in a list object. This object is required for the plot function (`plot_sde`).

## Usage

```
data(r.SDE)
```

**Format**

The list object contains the following results:

- id** Identifier for the SDE estimate - it should be unique
- points** a simple two-column data frame (or matrix) containing x,y coordinates for a series of point locations.
- coordsSDE** coordsSDE value, coordinates of the SDE
- calcentre** Boolean: TRUE if mean centre is computed
- CENTRE.x** X-coordinate of the centre
- CENTRE.y** Y-coordinate of the centre
- Sigma.x** Half-length of axis along x-axis
- Sigma.y** Half-length of axis along y-axis
- Major** String indicating which axis is the major elliptical axis
- Minor** String indicating which axis is the minor elliptical axis
- Theta** Rotation angle in degrees
- Eccentricity** A measure of eccentricity (i.e., the flatness of the ellipse)
- Area.sde** Area of the SDE
- TanTheta** Trigonometric result
- SinTheta** Trigonometric result
- CosTheta** Trigonometric result
- SinThetaCosTheta** Trigonometric result
- Sin2Theta** Trigonometric result
- Cos2Theta** Trigonometric result
- ThetaCorr** Corrected theta angle for rotation of major axis from north
- weighted** Boolean: TRUE if the weighted mean centre is to be used instead
- weights** Weights applied to point observations

**Details**

The coordinates of the points must have the same units and projection as the specified center.

**Source**

This demonstration data has been manufactured for illustrative purposes only.

**Examples**

```
data(r.SDE)
str(r.SDE)
```

---

`sin_d`*Compute sine with angle given in degrees*

---

**Description**

Provides the functionality of `sin`, but for input angles measured in degrees (not radians).

**Usage**

```
sin_d(theta = 0)
```

**Arguments**

`theta` A numeric angular measurement in degrees from north.

**Details**

Since the R default is to compute trigonometric functions on angular measurements stored in radians, this simple function performs the conversion from degrees, reducing the need to do so a priori, outside the function.

**Value**

Returns a numeric value for the sine of the specified angular measurement

**Note**

To reduce the need for unit conversions prior to calling trigonometric functions, this function accepts input in angular degrees rather than radians. Depending on data, this function may be preferred to the existing version requiring input in angular radians.

**Author(s)**

Tarmo K. Rimmel

**See Also**

[cos\\_d](#), [tan\\_d](#), [asin\\_d](#), [acos\\_d](#), [atan\\_d](#)

**Examples**

```
sin_d(theta = 90)
```

---

tan_d	<i>Compute tangent with angle given in degrees</i>
-------	--

---

**Description**

Provides the functionality of tan, but for input angles measured in degrees (not radians).

**Usage**

```
tan_d(theta = 0)
```

**Arguments**

theta            A numeric angular measurement in degrees from north.

**Details**

Since the R default is to compute trigonometric functions on angular measurements stored in radians, this simple function performs the conversion from degrees, reducing the need to do so a priori, outside the function.

**Value**

Returns a numeric value for the tangent of the specified angular measurement

**Note**

To reduce the need for unit conversions prior to calling trigonometric functions, this function accepts input in angular degrees rather than radians. Depending on data, this function may be preferred to the existing version requiring input in angular radians.

**Author(s)**

Tarmo K. Rimmel

**See Also**

[sin\\_d](#), [cos\\_d](#), [asin\\_d](#), [acos\\_d](#), [atan\\_d](#)

**Examples**

```
tan_d(theta = 45)
```

---

`wts`*Weights vector*

---

**Description**

This is a single column vector for weighting the importance of point locations.

**Usage**

```
data(wts)
```

**Format**

A single column vector of numeric values.

**Details**

The weights can be specified according to any reasonable criteria specified by the user

**Source**

This demonstration data has been manufactured for illustrative purposes only.

**Examples**

```
data(wts)
str(wts)
plot(wts)
```

# Index

## \*Topic **arith**

aspace-package, 2  
calc\_box, 9  
calc\_sdd, 10  
calc\_sde, 12  
CF, 15  
CF2PTS, 16  
CMD, 17  
distances, 19  
mean\_centre, 20  
median\_centre, 21  
plot\_box, 23  
plot\_centres, 24  
plot\_sdd, 26  
plot\_sde, 27

## \*Topic **array**

acos\_d, 3  
as\_radians, 7  
asin\_d, 6  
atan\_d, 8  
cos\_d, 18  
sin\_d, 32  
tan\_d, 33

## \*Topic **datasets**

activities, 4  
activities2, 5  
centre, 14  
r.BOX, 28  
r.SDD, 29  
r.SDE, 30  
wts, 34

acos\_d, 3, 6–8, 19, 32, 33  
activities, 4  
activities2, 5  
as\_radians, 7  
asin\_d, 4, 6, 7, 8, 19, 32, 33  
aspace (aspace-package), 2  
aspace-package, 2  
atan\_d, 4, 6, 7, 8, 19, 32, 33

calc\_box, 9, 11, 14  
calc\_sdd, 10, 10, 14  
calc\_sde, 10, 11, 12  
centre, 14  
CF, 15, 17, 18, 21, 22  
CF2PTS, 16  
CMD, 15, 17, 17, 21, 22  
cos\_d, 4, 6–8, 18, 32, 33  
  
distances, 19  
  
gridpts, 14  
  
mean\_centre, 15, 18, 20, 22  
median\_centre, 15, 17, 18, 21, 21  
  
plot\_box, 10, 23, 25, 27, 28  
plot\_centres, 24  
plot\_sdd, 11, 24, 26, 28  
plot\_sde, 14, 24, 25, 27, 27  
  
r.BOX, 28  
r.SDD, 29  
r.SDE, 30  
  
sin\_d, 4, 6–8, 19, 32, 33  
  
tan\_d, 4, 6–8, 19, 32, 33  
  
wtd.var, 10  
wts, 34