

# Package: vec2dtransf (via r-universe)

September 17, 2024

**Type** Package

**Title** 2D Cartesian Coordinate Transformation

**Version** 1.1.3

**Date** 2023-08-22

**Author** German Carrillo

**Maintainer** German Carrillo <geotux\_tuxman@linuxmail.org>

**Description** Applies affine and similarity transformations on vector spatial data (sp objects). Transformations can be defined from control points or directly from parameters. If redundant control points are provided Least Squares is applied allowing to obtain residuals and RMSE.

**License** GPL (>= 2)

**Collate** Class-Cartesian2DCoordinateTransformation.R  
Cartesian2DCoordinateTransformation-methods.R  
Class-SimilarityTransformation.R Class-AffineTransformation.R  
SimilarityTransformation-methods.R  
AffineTransformation-methods.R

**Depends** sp, methods

**URL** <https://github.com/gacarrillor/vec2dtransf>

**BugReports** <https://github.com/gacarrillor/vec2dtransf/issues>

**Repository** <https://gacarrillor.r-universe.dev>

**RemoteUrl** <https://github.com/gacarrillor/vec2dtransf>

**RemoteRef** HEAD

**RemoteSha** 71be7b34f4feb0181d3bb76b8d8c9a4b3509f701

## Contents

vec2dtransf-package . . . . .	2
AffineTransformation . . . . .	3
AffineTransformation-class . . . . .	4

applyTransformation . . . . .	5
calculateParameters . . . . .	6
Cartesian2DCoordinateTransformation-class . . . . .	7
control.points . . . . .	8
getParameters . . . . .	8
getResiduals . . . . .	9
getRMSE . . . . .	10
plotGridTransformation . . . . .	11
SimilarityTransformation . . . . .	12
SimilarityTransformation-class . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

---

vec2dtransf-package     *2D Cartesian Coordinate Transformation*

---

## Description

This package provides classes for defining and applying both affine and similarity transformations on vector spatial data (sp objects). Transformations can be defined from control points or directly from parameters. If redundant control points are provided Least Squares is applied allowing to obtain residuals and RMSE.

## Details

This package helps to define a transformation object (either similarity or affine) from control points or directly from parameters. Similarity transformations can rotate, shift and scale geometries whereas affine transformations can rotate, shift, scale (even applying different factors on each axis) and skew geometries.

If control points are provided, the 'calculateParameters' method must be used in order to calculate the transformation parameters. If there are more control points than required (2 for similarity and 3 for affine) Least Squares is applied and therefore residuals as well as Root Mean Square Error (RMSE) are available via 'getResiduals' and 'getRMSE' methods. Parameters can always be accessed via 'getParameters' method.

Once parameters are known, sp objects can be transformed by using the 'applyTransformation' method. Supported sp objects are SpatialPoints, SpatialPointsDataFrame, SpatialLines, SpatialLinesDataFrame, SpatialPolygons and SpatialPolygonsDataFrame.

Additionally, the method 'plotGridTransformation' can plot a grid of points to see the effect of the transformation on a given extent.

## Author(s)

German Carrillo (<http://geotux.tuxfamily.org>)  
 Maintainer: German Carrillo <geotux\_tuxman@linuxmail.org>

## References

Iliffe, J. and Lott, R. Datums and map projections: For remote sensing, GIS and surveying. Section 4.5. pp.109-117,135-137, 2008.

UC Davis Soil Resource Laboratory. Case Study: Fixing Bad TIGER Line data with R and Post-GIS. <URL: <http://casoilresource.lawr.ucdavis.edu/drupal/node/433>>

Knippers, R. 2D Cartesian coordinate transformations. 2009.  
<URL: <http://kartoweb.itc.nl/geometrics/Coordinate%20transformations/coordtrans.html>>

Pebesma, E. and Bivand, R. sp package:  
<URL: <http://cran.r-project.org/web/packages/sp/index.html>>

---

AffineTransformation *Create an AffineTransformation object*

---

## Description

Creates objects of class AffineTransformation from control points or directly from parameters

## Usage

```
AffineTransformation(controlPoints = data.frame(),  
  parameters = numeric())
```

## Arguments

**controlPoints** data.frame containing control point coordinates arranged in four (4) columns: X source, Y source, X target, Y target.

**parameters** A vector of six (6) parameters for representing the transformation, namely: a, b, c, d, e and f, where  
 $x' = ax + by + c$   
 $y' = dx + ey + f$

## Details

Both controlPoints and parameters are optional, but one has to be given. In the case of the latter, the name of the argument has to be specified, e.g., AffineTransformation(parameters=c(1,2,3,4,5,6))

## Value

Object of the class AffineTransformation

## Author(s)

German Carrillo

**See Also**

[AffineTransformation-class](#)

**Examples**

```
at <- AffineTransformation(parameters=c(1,2,3,4,5,6))
```

---

```
AffineTransformation-class
      Class "AffineTransformation"
```

---

**Description**

Class to define affine transformations to be applied on `sp` objects. Affine transformations can rotate, shift, scale (even applying different factors on each axis) and skew geometries.

**Objects from the Class**

Objects can be created by calls to the function [AffineTransformation](#).

**Slots**

**controlPoints:** Object of class "data.frame" containing control point coordinates arranged in four (4) columns: X source, Y source, X target, Y target.

**parameters:** Object of class "numeric". A vector of six (6) parameters for representing the transformation, namely: a, b, c, d, e and f, where

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

**residuals:** Object of class "matrix". Only set if Least Squares is applied, i.e., if more than three (3) control points were provided. Residuals are the difference between transformed source coordinates and target coordinates of control points.

**rmse:** Object of class "numericOrNULL". Only set if Least Squares is applied, i.e., if more than three (3) control points were provided. Root Mean Square Error, useful when comparing two transformations. It measures the general deviation of transformed source coordinates with respect to target coordinates of control points. It has the same units as the coordinates, usually meters.

**Extends**

Class "[Cartesian2DCoordinateTransformation](#)", directly.

**Methods**

**calculateParameters** signature(object = "AffineTransformation"):  
Calculate transformation parameters from control points.

**Author(s)**

German Carrillo

**References**

Iliffe, J. and Lott, R. Datums and map projections: For remote sensing, GIS and surveying. Section 4.5.4. pp.115-117, 2008.

**See Also**

[AffineTransformation](#)

**Examples**

```
showClass("AffineTransformation")
```

---

applyTransformation    *Apply the transformation to an sp object*

---

**Description**

The main functionality of this package. Applies the transformation to an sp object, which can have geometry type point, line or polygon and can have an associated data.frame.

**Usage**

```
applyTransformation(object, sp.object)
```

**Arguments**

object	Object of the class SimilarityTransformation or AffineTransformation
sp.object	An object of type: SpatialPoints, SpatialPointsDataFrame, SpatialLines, SpatialLinesDataFrame, SpatialPolygons or SpatialPolygonsDataFrame

**Details**

The transformation only affects sp object geometries. Attributes, IDs and other sp object characteristics are preserved.

The transformation can only be applied either on projected spatial data or on spatial data with unknown projection information. Geographic data cannot be transformed due to the nature of the methods (2D Cartesian).

**Value**

A transformed sp.object

**Author(s)**

German Carrillo

**References**

Pebesma, E. and Bivand, R. sp package:  
<URL: <http://cran.r-project.org/web/packages/sp/index.html>>

**See Also**

[sp](#), [calculateParameters](#),  
[SimilarityTransformation-class](#),  
[AffineTransformation-class](#)

**Examples**

```
# From the sp examples:
x <- c(1,1,1,2,2,2,3,3,3)
y <- c(1,2,3,1,2,3,1,2,3)
xy <- cbind(x,y)
S <- SpatialPoints(xy)

st=SimilarityTransformation(parameters=c(1,0,0.2,-0.2))
trS <- applyTransformation(st, S)
trS # Transformed sp object, have a look at the coordinates
```

---

calculateParameters    *Calculate parameters*

---

**Description**

Calculates transformation parameters from control points

**Usage**

```
calculateParameters(object)
```

**Arguments**

object            Object of the class SimilarityTransformation or AffineTransformation

**Details**

If control points are provided, the 'calculateParameters' method must be used in order to calculate the transformation parameters. If there are more control points than required (2 for similarity and 3 for affine) Least Squares is applied and therefore residuals as well as Root Mean Square Error (RMSE) are available via 'getResiduals' and 'getRMSE' methods. Parameters can always be accessed via 'getParameters' method.

**Value**

Modifies the original object

**Author(s)**

German Carrillo

**References**

Illife, J. and Lott, R. Datums and map projections: For remote sensing, GIS and surveying. Section 4.5. pp.109-117, 2008.

UC Davis Soil Resource Laboratory. Case Study: Fixing Bad TIGER Line data with R and PostGIS. <URL: <http://casoilresource.lawr.ucdavis.edu/drupal/node/433>>

**See Also**

[SimilarityTransformation](#), [AffineTransformation](#), [getResiduals](#), [getRMSE](#), [getParameters](#)

**Examples**

```
data(control.points)
at <- AffineTransformation(control.points[2:5])
getParameters(at) # Still unknown
calculateParameters(at)
getParameters(at) # Now there are parameters!
```

---

Cartesian2DCoordinateTransformation-class

*Class "Cartesian2DCoordinateTransformation"*

---

**Description**

Virtual base class for cartesian 2D coordinate transformations such as similarity and affine. Please read the section 'See Also' for further documentation on slots and methods.

**Objects from the Class**

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

**Author(s)**

German Carrillo

**See Also**

[SimilarityTransformation-class](#), [AffineTransformation-class](#)

---

<code>control.points</code>	<i>Sample control points data.frame</i>
-----------------------------	---

---

**Description**

This data.frame contains control points for defining a 2D cartesian coordinate transformation. The control points are in both the source and target coordinates.

**Usage**

```
data(control.points)
```

**Format**

A data.frame consisting on 16 control points with ID, coordinates X and Y from the source dataset and coordinates X and Y from the target dataset.

---

<code>getParameters</code>	<i>Get tranformation parameters</i>
----------------------------	-------------------------------------

---

**Description**

Retrieves the transformation parameters

**Usage**

```
getParameters(object)
```

**Arguments**

<code>object</code>	Object of the class <code>SimilarityTransformation</code> or <code>AffineTransformation</code>
---------------------	--

**Details**

If the transformation object was initialized with control points, parameters can be retrieved only after calling the method `'calculateParameters'`.

**Value**

A numeric vector with the transformation parameters.

**Author(s)**

German Carrillo



## References

Iliffe, J. and Lott, R. Datums and map projections: For remote sensing, GIS and surveying. Section 4.9.3-4. pp.135-137, 2008.

## See Also

[SimilarityTransformation-class](#) [AffineTransformation-class](#)

## Examples

```
# From parameters, similarity transformation
st <- SimilarityTransformation(parameters=c(1,2,3,4))
getParameters(st)

# From parameters, affine transformation
at <- AffineTransformation(parameters=c(1,2,3,4,5,6))
getParameters(at)

# From control points, affine transformation
data(control.points)
at2 <- AffineTransformation(control.points[2:5])
getParameters(at2) # Still unknown parameters
calculateParameters(at2)
getParameters(at2)
```

---

getResiduals

*Get residuals*

---

## Description

Retrieves the residuals if redundant control points were provided when creating the transformation object

## Usage

```
getResiduals(object)
```

## Arguments

object            Object of the class SimilarityTransformation or AffineTransformation

## Details

Residuals are determined only if Least Squares is applied, i.e., if more than the required control points were provided. Residuals are the difference between transformed source coordinates and target coordinates of control points.

**Value**

2-column matrix with coordinate differences

**Author(s)**

German Carrillo

**References**

Iliffe, J. and Lott, R. Datums and map projections: For remote sensing, GIS and surveying. Section 4.9.3-4. pp.135-137, 2008.

**See Also**

[SimilarityTransformation-class](#) [AffineTransformation-class](#)

**Examples**

```
data(control.points)
at <- AffineTransformation(control.points[2:5])
calculateParameters(at)
getResiduals(at)
```

---

getRMSE

*Get RMSE*

---

**Description**

Retrieves the Root Mean Square Error (RMSE) if redundant control points were provided when creating the transformation object

**Usage**

```
getRMSE(object)
```

**Arguments**

object            Object of the class `SimilarityTransformation` or `AffineTransformation`

**Details**

RMSE is determined only if Least Squares is applied, i.e., if more than the required control points were provided. The RMSE is useful when comparing two transformations. It measures the general deviation of transformed source coordinates with respect to target coordinates of control points. It has the same units as the coordinates, usually meters.

**Value**

A number representing the RMSE

**Author(s)**

German Carrillo

**See Also**

[SimilarityTransformation-class](#), [AffineTransformation-class](#)

**Examples**

```
data(control.points)
at <- AffineTransformation(control.points[2:5])
calculateParameters(at)
getRMSE(at)
```

---

plotGridTransformation  
*Plot grid transformation*

---

**Description**

Plots a grid transformed to visualize the effects of the transformation on a given extent

**Usage**

```
plotGridTransformation(object, bbox, numberOfPoints)
```

**Arguments**

object	Object of the class <code>SimilarityTransformation</code> or <code>AffineTransformation</code>
bbox	An SP <code>bbox</code> object, i.e. a 2x2 matrix with coordinates
numberOfPoints	Number of points to represent the grid, perfect squares are recommended.

**Details**

The grid is made out of points over the extent defined by `bbox`. If `numberOfPoints` has not an integer square root the grid will contain less points.

The grid is transformed using the transformation parameters and displayed over the source grid to see the effects of the transformation.

The source grid is drawn in red whereas the transformed one is green.

This code is based on "Case Study: Fixing Bad TIGER Line data with R and PostGIS" (See References).

**Author(s)**

German Carrillo

**References**

UC Davis Soil Resource Laboratory. Case Study: Fixing Bad TIGER Line data with R and PostGIS.  
 <URL: <http://casoilresource.lawr.ucdavis.edu/drupal/node/433>>

**See Also**

[applyTransformation](#)

**Examples**

```
# From the sp examples:
x <- c(1,1,1,2,2,2,3,3,3)
y <- c(1,2,3,1,2,3,1,2,3)
xy <- cbind(x,y)
S <- SpatialPoints(xy)

st=SimilarityTransformation(parameters=c(1,0,0.2,-0.2))
plotGridTransformation(st, bbox(S), 25)
```

---

SimilarityTransformation

*Create an SimilarityTransformation object*

---

**Description**

Creates objects of class SimilarityTransformation from control points or directly from parameters

**Usage**

```
SimilarityTransformation(controlPoints = data.frame(),
  parameters = numeric())
```

**Arguments**

controlPoints	data.frame containing control point coordinates arranged in four (4) columns: X source, Y source, X target, Y target.
parameters	A vector of four (4) parameters for representing the transformation, namely: a, b, c and d, where $x' = ax + by + c$ $y' = ay - bx + d$

**Details**

Both controlPoints and parameters are optional, but one has to be given. In the case of the latter, the name of the argument has to be specified, e.g., SimilarityTransformation(parameters=c(1,2,3,4))

**Value**

Object of the class SimilarityTransformation

**Author(s)**

German Carrillo

**See Also**

[SimilarityTransformation-class](#)

**Examples**

```
st <- SimilarityTransformation(parameters=c(1,2,3,4))
```

---

SimilarityTransformation-class

*Class "SimilarityTransformation"*

---

**Description**

Class to define similarity transformations to be applied on sp objects. Similarity transformations can rotate, shift and scale geometries.

**Objects from the Class**

Objects can be created by calls to the function [SimilarityTransformation](#).

**Slots**

**controlPoints:** Object of class "data.frame" containing control point coordinates arranged in four (4) columns: X source, Y source, X target, Y target.

**parameters:** Object of class "numeric". A vector of four (4) parameters for representing the transformation, namely: a, b, c and d, where  $x' = ax + by + c$   $y' = ay - bx + d$

**residuals:** Object of class "matrix". Only set if Least Squares is applied, i.e., if more than two (2) control points were provided. Residuals are the difference between transformed source coordinates and target coordinates of control points.

**rmse:** Object of class "numericOrNULL". Only set if Least Squares is applied, i.e., if more than two (2) control points were provided. Root Mean Square Error, useful when comparing two transformations. It measures the general deviation of transformed source coordinates with respect to target coordinates of control points. It has the same units as the coordinates, usually meters.

**Extends**

Class "[Cartesian2DCoordinateTransformation](#)", directly.

**Methods**

**calculateParameters** signature(object = "SimilarityTransformation"):  
Calculate transformation parameters from control points.

**Author(s)**

German Carrillo

**References**

Iliffe, J. and Lott, R. Datums and map projections: For remote sensing, GIS and surveying. Section 4.5.3. pp.113-115, 2008.

**See Also**

[SimilarityTransformation](#)

**Examples**

```
showClass("SimilarityTransformation")
```



`vec2dtransf` (`vec2dtransf`-package), [2](#)  
`vec2dtransf`-package, [2](#)