# Package ‘TriDimRegression’

October 12, 2022

**Title**  
Bayesian Statistics for 2D/3D Transformations  

**Version**  
1.0.1

**Description**  
Fits 2D and 3D geometric transformations via 'Stan' probabilistic programming engine ([Stan Development Team (2021)](https://mc-stan.org)). Returns posterior distribution for individual parameters of the fitted distribution. Allows for computation of LOO and WAIC information criteria  

**License**  
GPL-3

**URL**  
https://github.com/alexander-pastukhov/tridim-regression

**BugReports**  
https://github.com/alexander-pastukhov/tridim-regression/issues

**Encoding**  
UTF-8

**LazyData**  
true

**RoxygenNote**  
7.1.1

**VignetteBuilder**  
knitr

**Biarch**  
true

**Depends**  
R (>= 3.5.0), loo

**Imports**  
methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.1.1), dplyr, future, glue, purrr, tidyr, Formula, bayesplot

**LinkingTo**  
BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

**SystemRequirements**  
GNU make

**Suggests**  
testthat, knitr, rmarkdown, ggplot2

**NeedsCompilation**  
yes

**Author**  
Alexander (Sasha) Pastukhov [cre, ctb],  
Claus-Christian Carbon [aut]
**Maintainer**  Alexander (Sasha) Pastukhov <pastukhov.alexander@gmail.com>

**Repository**  CRAN

**Date/Publication**  2021-10-05 08:30:08 UTC

---

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriDimRegression-package</td>
<td>2</td>
</tr>
<tr>
<td>CarbonExample1Data</td>
<td>4</td>
</tr>
<tr>
<td>CarbonExample2Data</td>
<td>4</td>
</tr>
<tr>
<td>CarbonExample3Data</td>
<td>5</td>
</tr>
<tr>
<td>coef.tridim_transformation</td>
<td>5</td>
</tr>
<tr>
<td>EyegazeData</td>
<td>6</td>
</tr>
<tr>
<td>Face3D_M010</td>
<td>7</td>
</tr>
<tr>
<td>Face3D_M101</td>
<td>8</td>
</tr>
<tr>
<td>Face3D_M244</td>
<td>8</td>
</tr>
<tr>
<td>Face3D_M92</td>
<td>9</td>
</tr>
<tr>
<td>Face3D_W070</td>
<td>9</td>
</tr>
<tr>
<td>Face3D_W097</td>
<td>10</td>
</tr>
<tr>
<td>Face3D_W182</td>
<td>10</td>
</tr>
<tr>
<td>Face3D_W243</td>
<td>11</td>
</tr>
<tr>
<td>fit_transformation</td>
<td>11</td>
</tr>
<tr>
<td>fit_transformation_df</td>
<td>13</td>
</tr>
<tr>
<td>FriedmanKohlerData1</td>
<td>14</td>
</tr>
<tr>
<td>FriedmanKohlerData2</td>
<td>15</td>
</tr>
<tr>
<td>is.tridim_transformation</td>
<td>15</td>
</tr>
<tr>
<td>loo.tridim_transformation</td>
<td>16</td>
</tr>
<tr>
<td>NakayaData</td>
<td>16</td>
</tr>
<tr>
<td>plot.tridim_transformation</td>
<td>17</td>
</tr>
<tr>
<td>predict.tridim_transformation</td>
<td>18</td>
</tr>
<tr>
<td>print.tridim_transformation</td>
<td>19</td>
</tr>
<tr>
<td>R2</td>
<td>19</td>
</tr>
<tr>
<td>summary.tridim_transformation</td>
<td>20</td>
</tr>
<tr>
<td>tridim_transformation-class</td>
<td>21</td>
</tr>
<tr>
<td>waic.tridim_transformation</td>
<td>21</td>
</tr>
</tbody>
</table>

---

The 'TriDimRegression' package.
Description
Fits 2D and 3D geometric transformations. Provides posterior via Stan. Includes computation of LOO and WAIC information criteria, R-squared.

To fit transformation, call the main function either via a formula that specifies dependent and independent variables with the data table or by supplying two tables one containing all independent variables and one containing all dependent variables.

For the 2D data, you can fit "translation" (2 parameters for translation only), "euclidean" (4 parameters: 2 for translation, 1 for scaling, and 1 for rotation), "affine" (6 parameters: 2 for translation and 4 that jointly describe scaling, rotation and sheer), or "projective" (8 parameters: affine plus 2 additional parameters to account for projection). For 3D data, you can fit "translation" (3 for translation only), "euclidean_x", "euclidean_y", "euclidean_z" (5 parameters: 3 for translation scale, 1 for rotation, and 1 for scaling), "affine" (12 parameters: 3 for translation and 9 to account for scaling, rotation, and sheer), and "projective" (15 parameters: affine plus 3 additional parameters to account for projection). For details on transformation matrices and computation of scale and rotation parameters please see vignette("transformation_matrices", package = "TriDimRegression")

Once the data is fitted, you can extract the transformation coefficients via coef function and the matrix itself via transformation_matrix. Predicted data, either based on the original data or on the new data, can be generated via predict. Bayesian R-squared can be computed with or without adjustment via R2 function. In all three cases, you have choice between summary (mean + specified quantiles) or full posterior samples. loo and waic provide corresponding measures that can be used for comparison via loo::loo_compare() function.

References

See Also
fit_transformation fit_transformation_df tridim_transformation vignette("transformation_matrices", package = "TriDimRegression") vignette("calibration", package = "TriDimRegression") vignette("comparing_faces", package = "TriDimRegression")

Examples
# Fitting via formula
euc2 <- fit_transformation(depV1 + depV2 ~ indepV1 + indepV2, NakayaData, 'euclidean')
aff2 <- fit_transformation(depV1 + depV2 ~ indepV1 + indepV2, NakayaData, 'affine')
prj2 <- fit_transformation(depV1 + depV2 ~ indepV1 + indepV2, NakayaData, 'projective')

# summary of transformation coefficients
coef(euc2)

# statistical comparison via WAIC criterion
loo::loo_compare(waic(euc2), waic(aff2), waic(prj2))
# Fitting via two tables

euc2 <- fit_transformation_df(NakayaData[, 1:2], NakayaData[, 3:4], 'euclidean')

tr3 <- fit_transformation_df(Face3D_W070, Face3D_W097, transformation = 'translation')

CarbonExample1Data  

Carbon, C. C. (2013), data set #1

Description

Example 1 from the domain of aesthetics to show how the method can be utilized for assessing the similarity of two portrayed persons, actually the Mona Lisa in the world famous Louvre version and the only recently re-discovered Prado version.

Usage

CarbonExample1Data

Format

A data frame with 36 observations on the following 4 variables:

- **depV1, depV2** numeric vectors, dependent variables
- **indepV1, indepV2** numeric vectors, independent variables

Source

doi: 10.18637/jss.v052.c01

CarbonExample2Data  

Carbon, C. C. (2013), data set #2

Description

Example 2 originates from the area of geography and inspects the accuracy of different maps of the city of Paris which were created over the last 350 years as compared to a recent map.

Usage

CarbonExample2Data

Format

A data frame with 13 observations on the following 4 variables:

- **depV1, depV2** numeric vectors, dependent variables
- **indepV1, indepV2** numeric vectors, independent variables
**Source**

doi: 10.18637/jss.v052.c01

---

**Description**

Example 3 focuses on demonstrating how good a cognitive map recalculated from averaged cognitive distance data fits with a related real map.

**Usage**

CarbonExample3Data

**Format**

A data frame with 10 observations on the following 4 variables:

- `depV1` numeric vectors, dependent variables
- `depV2` numeric vectors, dependent variables
- `indepV1` numeric vectors, independent variables
- `indepV2` numeric vectors, independent variables

**Source**

doi: 10.18637/jss.v052.c01

---

**coef.tridim_transformation**

*Posterior distributions for transformation coefficients in full or summarized form.*

**Description**

Posterior distributions for transformation coefficients in full or summarized form.

**Usage**

```r
## S3 method for class 'tridim_transformation'
coef(
  object,
  summary = TRUE,
  probs = c(0.055, 0.945),
  convert_euclidean = FALSE,
  ...
)
```

Arguments

object An object of class tridim_transformation.

summary Whether summary statistics should be returned instead of raw sample values. Defaults to TRUE.

probs The percentiles used to compute summary, defaults to 89% credible interval.

convert_euclidean Whether to convert matrix coefficients to scale(phi) and rotation(theta). Defaults to FALSE.

... Unused

Value

If summary=FALSE, a list with matrix iterationsN x dimensionsN for each variable. If summary=TRUE, a data.frame with columns "dvindex" with mean for each dependent variable plus optional quantiles columns with names "dvindex_quantile".

Examples

euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2, 
data = NakayaData, 
transformation = 'euclidean')

# full posterior distribution 
transform_posterior <- coef(euc2, summary=FALSE)

# coefficients' summary with 89% CI 
coef(euc2)

# scale and rotation coefficients 
coef(euc2, convert_euclidean=TRUE)

EyegazeData Eye gaze calibration data

Description

A dataset containing a monocular eye gaze recording with calibration sequence. Courtesy of Bamberger Baby Institut: BamBI.

Usage

EyegazeData
Format

A data frame with 365 rows and 6 variables:

- **time** sample timestamp, in milliseconds
- **x, y** recorded gaze, in internal eye tracker units
- **target_x, target_y** location of the calibration target on the screen, in pixels
- **target** index of the target within the sequence

Source

[https://www.uni-bamberg.de/entwicklungspychologie/transfer/babyforschung-bambi/](https://www.uni-bamberg.de/entwicklungspychologie/transfer/babyforschung-bambi/).

---

**Face3D_M010**

*Face landmarks, male, #010*

---

Description

Face landmarks, male, #010

Usage

Face3D_M010

Format

A data frame with 64 landmarks on the following 3 variables:

- **x, y, z** numeric vectors, coordinates of face landmarks

Source

Face3D_M101  
*Face landmarks, male, #101*

**Description**

Face landmarks, male, #101

**Usage**

Face3D_M101

**Format**

A data frame with 64 landmarks on the following 3 variables:

- x, y, z numeric vectors, coordinates of face landmarks

**Source**


---

Face3D_M244  
*Face landmarks, male, #244*

**Description**

Face landmarks, male, #244

**Usage**

Face3D_M244

**Format**

A data frame with 64 landmarks on the following 3 variables:

- x, y, z numeric vectors, coordinates of face landmarks

**Source**

## Face3D_M92

**Description**

Face landmarks, male, #092

**Usage**

Face3D_M92

**Format**

A data frame with 64 landmarks on the following 3 variables:

- \( x, y, z \) numeric vectors, coordinates of face landmarks

**Source**


## Face3D_W070

**Description**

Face landmarks, female, #070

**Usage**

Face3D_W070

**Format**

A data frame with 64 landmarks on the following 3 variables:

- \( x, y, z \) numeric vectors, coordinates of face landmarks

**Source**

Face3D_W097  

Face landmarks, female, #097

Description
Face landmarks, female, #097

Usage
Face3D_W097

Format
A data frame with 64 landmarks on the following 3 variables:

\( x, y, z \) numeric vectors, coordinates of face landmarks

Source

Face3D_W182  

Face landmarks, female, #182

Description
Face landmarks, female, #182

Usage
Face3D_W182

Format
A data frame with 64 landmarks on the following 3 variables:

\( x, y, z \) numeric vectors, coordinates of face landmarks

Source
Face3D_W243

Face landmarks, female, #243

Description

Face landmarks, female, #243

Usage

Face3D_W243

Format

A data frame with 64 landmarks on the following 3 variables:

\( x, y, z \) numeric vectors, coordinates of face landmarks

Source


fit_transformation

Fitting Bidimensional or Tridimensional Regression / Geometric Transformation Models via Formula.

Description

Fits Bidimensional or Tridimensional regression / geometric transformation models using Stan engine. The formula described dependent and independent numeric variables in the data. See also fit_transformation_df.

For the 2D data, you can fit "translation" (2 parameters for translation only), "euclidean" (4 parameters: 2 for translation, 1 for scaling, and 1 for rotation), "affine" (6 parameters: 2 for translation and 4 that jointly describe scaling, rotation and sheer), or "projective" (8 parameters: affine plus 2 additional parameters to account for projection).

For 3D data, you can fit "translation" (3 for translation only), "euclidean_x", "euclidean_y", "euclidean_z" (5 parameters: 3 for translation scale, 1 for rotation, and 1 for scaling), "affine" (12 parameters: 3 for translation and 9 to account for scaling, rotation, and sheer), and "projective" (15 parameters: affine plus 3 additional parameters to account for projection). transformations.

For details on transformation matrices and computation of scale and rotation parameters please see vignette("transformation_matrices", package = "TriDimRegression")
Usage

```r
## S3 method for class 'formula'
fit_transformation(
  formula,
  data,
  transformation,
  priors = NULL,
  chains = 1,
  cores = NULL,
  ...
)
```

Arguments

- **formula**: a symbolic description of the model to be fitted in the format \(X_{dep} + Y_{dep} \sim X_{ind} + Y_{ind}\), where \(X_{dep}\) and \(Y_{dep}\) are dependent and \(X_{ind}\) and \(Y_{ind}\) are independent variables.
- **data**: a data frame containing variables for the model.
- **transformation**: the transformation to be used: "translation" (both 2D and 3D), "euclidean" (2D), "euclidean_x", "euclidean_y", "euclidean_z" (3D, rotation about, respectively, x, y, and z axis), "affine" (2D and 3D), or "projective" (2D and 3D).
- **priors**: named list of parameters for prior distributions of parameters \(a\) (translation, normal distribution), \(b\) (all other parameters, normal distribution), and \(sigma\) (residual variance, exponential). E.g., `list("a" = c(0, 10), "b" = c(0, 1), "sigma" = 1)`. Default priors are \(a = c(0, \text{max\_absolute\_difference\_in\_means}(d, iv)) / 2\), \(b = c(0, \text{max\_absolute\_difference\_in\_means}(d, iv)) / 2\), and \(sigma = 1 * \text{sd}(dv)\).
- **chains**: Number of chains for sampling.
- **cores**: Number of CPU cores to use for sampling. If omitted, all available cores are used.
- **...**: Additional arguments passed to `sampling` function.

Value

A `tridim_transformation` object

See Also

- `fit_transformation_df`

Examples

```r
# Geometric transformations of 2D data
euc2 <- fit_transformation(depV1 + depV2 ~ indepV1 + indepV2,
                           NakayaData, "euclidean")
aff2 <- fit_transformation(depV1 + depV2 ~ indepV1 + indepV2,
                           NakayaData, "affine")
```
fit_transformation_df

NakayaData, 'affine')
prj2 <- fit_transformation(depV1 + depV2 ~ indepV1 + indepV2,
NakayaData, 'projective')

# summary of transformation coefficients
coeff(euc2)

# statistical comparison via WAIC criterion
loo::loo_compare(waic(euc2), waic(aff2), waic(prj2))

fit_transformation_df  Fitting Bidimensional or Tridimensional Regression / Geometric Transformation Models via Two Tables.

Description

Fits Bidimensional or Tridimensional regression / geometric transformation models using Stan engine. Two sets of coordinates are supplied via iv (for an independent variable) and dv (for the dependent one). The two tables must have the same dimensions (both N×2 or N×3).

For the 2D data, you can fit "translation" (2 for translation only), "euclidean" (4 parameters: 2 for translation, 1 for scaling, and 1 for rotation), "affine" (6 parameters: 2 for translation and 4 that jointly describe scaling, rotation and sheer), or "projective" (8 parameters: affine plus 2 additional parameters to account for projection).

For 3D data, you can fit "translation" (3 for translation only), "euclidean_x", "euclidean_y", "euclidean_z" (5 parameters: 3 for translation scale, 1 for rotation, and 1 for scaling), "affine" (12 parameters: 3 for translation and 9 to account for scaling, rotation, and sheer), and "projective" (15 parameters: affine plus 3 additional parameters to account for projection). transformations.

For details on transformation matrices and computation of scale and rotation parameters please see vignette("transformation_matrices", package = "TriDimRegression")

Usage

fit_transformation_df(
  iv,
  dv,
  transformation,
  priors = NULL,
  chains = 1,
  cores = NULL,
  ...
)

Arguments

iv  a data frame containing independent variable, must by numeric only, N×2 or N×3.
dv  a data frame containing dependent variable, must by numeric only, N×2 or N×3.
transformation  the transformation to be used: "translation" (both 2D and 3D), "euclidean" (2D), "euclidean_x", "euclidean_y", "euclidean_z" (3D, rotation about, respectively, x, y, and z axis), "affine" (2D and 3D), or "projective" (2D and 3D).
priors  named list of parameters for prior distributions of parameters a (translation, normal distribution), b (all other parameters, normal distribution), and sigma (residual variance, exponential). E.g., list("a" = c(0, 10), "b" = c(0, 1), "sigma" = 1). Default priors are "a" = c(0, max_absolute_difference_in_means(d, iv)) / 2), "b" = c(0, max_absolute_difference_in_means(d, iv)) / 2), "sigma" = 1 * sd(dv).
chains  Number of chains for sampling.
cores  Number of CPU cores to use for sampling. If omitted, all available cores are used.
...  Additional arguments passed to sampling function.

Value
A tridim_transformation object

See Also
fit_transformation

Examples

# Geometric transformations of 2D data
euc2 <- fit_transformation_df(NakayaData[,1:2], NakayaData[,3:4], transformation = "euclidean")
tr3 <- fit_transformation_df(Face3D_W070, Face3D_W097, transformation = "translation")

Description

Usage
FriedmanKohlerData1

Format
A data frame with 4 observations on the following 4 variables:

depV1, depV2  numeric vectors, dependent variables
indepV1, indepV2  numeric vectors, independent variables
FriedmanKohlerData2

Source
doi: 10.1037/1082989X.8.4.468

FriedmanKohlerData2  Friedman & Kohler (2003), data set #2

Description

Usage
FriedmanKohlerData2

Format
A data frame with 4 observations on the following 4 variables:

- depV1, depV2  numeric vectors, dependent variables
- indepV1, indepV2  numeric vectors, independent variables

Source
doi: 10.1037/1082989X.8.4.468

is.tridim_transformation

Description
Checks if argument is a tridim_transformation object

Usage
is.tridim_transformation(x)

Arguments
x  An R object

Value
Logical
Description

Computes an efficient approximate leave-one-out cross-validation via loo library. It can be used for a model comparison via loo::loo_compare() function.

Usage

```r
## S3 method for class 'tridim_transformation'
loo(x, ...)
```

Arguments

- `x`: A `tridim_transformation` object
- `...`: unused

Value

A named list, see `loo::loo()` for details.

Examples

```r
euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2, NakayaData, transformation = 'euclidean')
aff2 <- fit_transformation(depV1+depV2~indepV1+indepV2, NakayaData, transformation = 'affine')
loo::loo_compare(loo(euc2), loo(aff2))
```

Description

Format

A data frame with 19 observations on the following 4 variables:

- **depV1, depV2** numeric vectors, dependent variables
- **indepV1, indepV2** numeric vectors, independent variables

Source


Description

Posterior interval plots for key parameters. Uses `bayesplot::mcmc_intervals`.

Usage

```r
## S3 method for class 'tridim_transformation'
plot(x, convert_euclidean = FALSE, ...)
```

Arguments

- `x` A `tridim_transformation` object
- `convert_euclidean` Whether to convert matrix coefficients to scale(\(\phi\)) and rotation(\(\theta\)). Defaults to `FALSE`.
- `...` Extra parameters to be passed to `bayesplot::mcmc_intervals()`

Value

A ggplot object produced by `bayesplot::mcmc_intervals()`

Examples

```r
euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2,
data = NakayaData,
transformation = 'euclidean')
plot(euc2)

# same but for converted coefficients
plot(euc2, convert_euclidean=TRUE)
```
predict.tridim_transformation

Computes posterior samples for the posterior predictive distribution.

Description

Predicted values based on the bi/tridimensional regression model object.

Usage

## S3 method for class 'tridim_transformation'
predict(object, newdata = NULL, summary = TRUE, probs = NULL, ...)

Arguments

- **object**: An object of class `tridim_transformation`
- **newdata**: An optional two column data frame with independent variables. If omitted, the fitted values are used.
- **summary**: Whether summary statistics should be returned instead of raw sample values. Defaults to `TRUE`.
- **probs**: The percentiles used to compute summary, defaults to `NULL` (no CI).
- **...**: Unused

Value

If summary=FALSE, a numeric matrix iterationsN x observationsN x variablesN. If summary=TRUE, a data.frame with columns "dvindex" with mean for each dependent variable plus optional quantiles columns with names "dvindex_quantile".

See Also

- `fit_transformation`

Examples

euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2,
    NakayaData, transformation = 'euclidean')

# prediction summary
predictions <- predict(euc2)

# full posterior prediction samples
predictions <- predict(euc2, summary=FALSE)
print.tridim_transformation

Prints out tridim_transformation object

Description

Prints out tridim_transformation object

Usage

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

Arguments

x

A tridim_transformation object

...

Unused

Value

Nothing, console output only.

Examples

euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2,
    data = NakayaData,
    transformation = 'euclidean')
euc2

R2


Description


Usage

## S3 method for class 'tridim_transformation'
R2(object, summary = TRUE, probs = c(0.055, 0.945), ...)
Arguments

- `object`: An object of class `tridim_transformation`
- `summary`: Whether summary statistics should be returned instead of raw sample values. Defaults to `TRUE`.
- `probs`: The percentiles used to compute summary, defaults to 89% credible interval.
- `...`: Unused.

Value

A vector of values or a data.frame with summary

Examples

```r
euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2, 
  NakayaData, transformation = "euclidean")
R2(euc2)

summary(euc2)
```

Summary for a tridim_transformation object

Description

Summary for a tridim_transformation object

Usage

```r
## S3 method for class 'tridim_transformation'
summary(object, ...)
```

Arguments

- `object`: A `tridim_transformation` object
- `...`: Unused

Value

Nothing, console output only.

Examples

```r
euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2, 
  data = NakayaData, 
  transformation = "euclidean")

summary(euc2)
```
tridim_transformation-class

Class tridim_transformation.

Description
Geometric transformations fitted with the `fit_transformation` function represented as a `tridim_transformation` object with information about transformation, data dimension, call formula, and fitted `stanfit` object.

Details
See methods(class = "tridim_transformation") for an overview of available methods.

Slots
- `transformation` A string with the transformation name.
- `formula` A `formula` object.
- `Ndim` An integer with data dimension, either 2 or 3.
- `data` A list containing variables used for the sampling.
- `stanmodel` A `stanmodel` used for sampling.
- `stanfit` a `stanfit` object.

See Also
- `fit_transformation`

waic.tridim_transformation

Computes widely applicable information criterion (WAIC).

Description
Computes widely applicable information criterion via loo library. It can be used for a model comparison via `loo::loo_compare()` function.

Usage
```r
## S3 method for class 'tridim_transformation'
waic(x, ...)
```

Arguments
- `x` A `tridim_transformation[tridim_transformation-class]` object
- `...` unused
Value

A named list, see \texttt{loo::waic()} for details.

Examples

\begin{verbatim}
euc2 <- fit_transformation(depV1+depV2~indepV1+indepV2, NakayaData, transformation = 'euclidean')
aff2 <- fit_transformation(depV1+depV2~indepV1+indepV2, NakayaData, transformation = 'affine')
loo::loo_compare(waic(euc2), waic(aff2))
\end{verbatim}
Index

* datasets
  CarbonExample1Data, 4
  CarbonExample2Data, 4
  CarbonExample3Data, 5
  EyegazeData, 6
  Face3D_M010, 7
  Face3D_M101, 8
  Face3D_M244, 8
  Face3D_M92, 9
  Face3D_W070, 9
  Face3D_W097, 10
  Face3D_W182, 10
  Face3D_W243, 11
  FriedmanKohlerData1, 14
  FriedmanKohlerData2, 15
  NakayaData, 16

  bayesplot::mcmc_intervals(), 17

  CarbonExample1Data, 4
  CarbonExample2Data, 4
  CarbonExample3Data, 5
  coef, 3
  coef.tridim_transformation, 5

  EyegazeData, 6

  Face3D_M010, 7
  Face3D_M101, 8
  Face3D_M244, 8
  Face3D_M92, 9
  Face3D_W070, 9
  Face3D_W097, 10
  Face3D_W182, 10
  Face3D_W243, 11
  fit_transformation, 3, 11, 14, 18, 21
  fit_transformation_df, 3, 11, 12, 13
  formula, 21
  FriedmanKohlerData1, 14
  FriedmanKohlerData2, 15

  is.tridim_transformation, 15
  loo, 3, 21
  loo.tridim_transformation, 16
  loo::loo(), 16
  loo::loo_compare(), 3, 21
  loo::waic(), 22

  NakayaData, 16

  plot.tridim_transformation, 17
  predict, 3
  predict.tridim_transformation, 18
  print.tridim_transformation, 19

  R2, 3, 19

  sampling, 12, 14, 21
  stanfit, 21
  stanmodel, 21
  summary.tridim_transformation, 20

  transformation_matrix, 3
  tridim_transformation, 3, 6, 12, 14, 16–20

  tridim_transformation
    (tridim_transformation-class), 21
  tridim_transformation-class, 21

  TriDimRegression
    (TriDimRegression-package), 2
  TriDimRegression-package, 2

  waic, 3
  waic.tridim_transformation, 21