

# Package ‘cfda’

August 16, 2022

**Type** Package

**Title** Categorical Functional Data Analysis

**Version** 0.10.1

**Date** 2022-08-16

**Copyright** Inria - Université de Lille

**Description** Package for the analysis of categorical functional data.

The main purpose is to compute an encoding (real functional variable) for each state <[doi:10.3390/math9233074](https://doi.org/10.3390/math9233074)>.

It also provides functions to perform basic statistical analysis on categorical functional data.

**BugReports** <https://github.com/modal-inria/cfda/issues>

**License** AGPL-3

**Imports** msm, diagram, mgcv, parallel, pbapply

**Depends** fda, ggplot2, R (>= 3.5.0)

**Suggests** testthat, covr, knitr, rmarkdown

**Encoding** UTF-8

**VignetteBuilder** knitr

**RoxygenNote** 7.2.1

**URL** <https://modal-inria.github.io/cfda/>

**NeedsCompilation** no

**Author** Cristian Preda [aut],  
Quentin Grimonprez [aut, cre],  
Vincent Vandewalle [ctb]

**Maintainer** Quentin Grimonprez <[quentingrim@yahoo.fr](mailto:quentingrim@yahoo.fr)>

**Repository** CRAN

**Date/Publication** 2022-08-16 12:30:02 UTC

**R topics documented:**

cfda-package	2
biofam2	4
boxplot.timeSpent	5
care	6
compute_duration	8
compute_number_jumps	9
compute_optimal_encoding	10
compute_time_spent	12
convertToCfd	13
cut_data	14
estimate_Markov	16
estimate_pt	17
flours	18
generate_2State	19
generate_Markov	19
get_encoding	21
get_state	22
hist.duration	23
hist.njump	24
matrixToCfd	25
plot.fmca	26
plot.Markov	27
plot.pt	28
plotComponent	29
plotData	31
plotEigenvalues	32
predict.fmca	34
print.fmca	35
remove_duplicated_states	36
statetable	37
summary.fmca	38
summary_cfd	38
<b>Index</b>	<b>40</b>

**Description**

cfda provides functions for the analysis of categorical functional data.

The main contribution is the computation of an optimal encoding (real functional variable) of each state of the categorical functional data. This can be done using the [compute\\_optimal\\_encoding](#) function that takes in arguments the data in a specific format and a basis of functions created using

the fda package (cf. `create.basis`). The output can be analysed with `summary.fmca`, `plot.fmca`, `get_encoding`, `plotEigenvalues` and `plotComponent`.

Moreover, cfda contains functions to visualize and compute some statistics about categorical functional data. A summary of the dataset is available with `summary_cfd`. `plotData` shows a graphical representation of the dataset. Basic statistics can be computed: the number of jumps (`compute_number_jumps`), the duration (`compute_duration`), the time spent in each state (`compute_time_spent`), the probability to be in each state at any given time (`estimate_pt`), the transition table (`statetable`).

The parameters of a Markov process can be estimated using `estimate_Markov` function.

In order to test the different functions, a real dataset is provided (`biofam2`) as well as two functions for generating data: (`generate_Markov` and `generate_2State`).

## Details

See the vignette for a detailed example and mathematical background: `RShowDoc("cfda", package = "cfda")`

## References

- Deville J.C. (1982) Analyse de données chronologiques qualitatives : comment analyser des calendriers ?, Annales de l'INSEE, No 45, p. 45-104.
- Deville J.C. et Saporta G. (1980) Analyse harmonique qualitative, DIDAY et al. (editors), Data Analysis and Informatics, North Holland, p. 375-389.
- Saporta G. (1981) Méthodes exploratoires d'analyse de données temporelles, Cahiers du B.U.R.O, Université Pierre et Marie Curie, 37-38, Paris.
- Preda C, Grimonprez Q, Vandewalle V. Categorical Functional Data Analysis. The cfda R Package. Mathematics. 2021; 9(23):3074. <https://doi.org/10.3390/math9233074>

## See Also

[compute\\_optimal\\_encoding](#)

## Examples

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 5
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 5
b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)
summary(encoding)
```

```
# plot eigenvalues
plotEigenvalues(encoding, cumulative = TRUE, normalize = TRUE)

# plot the two first components
plotComponent(encoding, comp = c(1, 2))

# plot the encoding using the first harmonic
plot(encoding)

# extract the encoding using the first harmonic
encod <- get_encoding(encoding)
```

---

 biofam2

*Family life states from the Swiss Household Panel biographical survey*


---

### Description

2000 16 year-long family life sequences built from the retrospective biographical survey carried out by the Swiss Household Panel (SHP) in 2002. Data from TraMineR package.

### Usage

```
data(biofam2)
```

### Format

A data.frame containing three columns:

- *id* id of individuals (2000 different ids)
- *time* age in years where a change occurs
- *state* new state.

### Details

The biofam2 dataset derives from the biofam dataset from TraMineR package. The biofam2 format is adapted to cfda functions. The biofam data set was constructed by Müller et al. (2007) from the data of the retrospective biographical survey carried out by the Swiss Household Panel (SHP) in 2002. The data set contains sequences of family life states from age 15 to 30 (sequence length is 16). The sequences are a sample of 2000 sequences of those created from the SHP biographical survey. It includes only individuals who were at least 30 years old at the time of the survey. The biofam data set describes family life courses of 2000 individuals born between 1909 and 1972.

The eight states are defined from the combination of five basic states, namely Living with parents (Parent), Left home (Left), Married (Marr), Having Children (Child), Divorced: "Parent", "Left", "Married", "Left+Marr", "Child", "Left+Child", "Left+Marr+Child", "Divorced"

### Source

Swiss Household Panel <https://forscenter.ch/projects/swiss-household-panel/>

## References

Müller, N. S., M. Studer, G. Ritschard (2007). Classification de parcours de vie à l'aide de l'optimal matching. In XIVe Rencontre de la Société francophone de classification (SFC 2007), Paris, 5 - 7 septembre 2007, pp. 157–160.

## See Also

Other datasets: [care](#), [flours](#)

## Examples

```
data(biofam2)
head(biofam2)

plotData(biofam2)

# It is recommended to increase the number of cores to reduce computation time
set.seed(42)
basis <- create.bspline.basis(c(15, 30), nbasis = 4, norder = 4)
fmca <- compute_optimal_encoding(biofam2, basis, nCores = 2)

plot(fmca, harm = 1)
plot(fmca, harm = 2)
plotEigenvalues(fmca, cumulative = TRUE, normalize = TRUE)
plotComponent(fmca, comp = c(1, 2), addNames = FALSE)
```

---

boxplot.timeSpent      *Boxplot of time spent in each state*

---

## Description

Boxplot of time spent in each state

## Usage

```
## S3 method for class 'timeSpent'
boxplot(x, col = NULL, ...)
```

## Arguments

x	output of <a href="#">compute_time_spent</a> function
col	a vector containing color for each state
...	extra parameters for <code>geom_boxplot</code>

## Value

a ggplot object that can be modified using ggplot2 package.

**Author(s)**

Quentin Grimonprez

**See Also**

[compute\\_time\\_spent](#)

Other Descriptive statistics: [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# cut at Tmax = 8
d_JK2 <- cut_data(d_JK, Tmax = 8)

# compute time spent by each id in each state
timeSpent <- compute_time_spent(d_JK2)

# plot the result
boxplot(timeSpent, col = c("#8DA0CB", "#E78AC3", "#A6D854", "#FFD92F"))

# modify the plot using ggplot2
library(ggplot2)
boxplot(timeSpent, notch = TRUE, outlier.colour = "black") +
  coord_flip() +
  labs(title = "Time spent in each state")
```

---

care

*Care trajectories*

---

**Description**

Care trajectories of patients diagnosed with a serious and chronic condition

**Usage**

```
data(care)
```

**Format**

A data.frame containing three columns:

- *id* id of individuals (2929 different ids)
- *time* number of months since the diagnosis
- *state* new state.

## Details

In this study, patients were followed from the time they were diagnosed with a serious and chronic condition and their care trajectories were tracked monthly from the time of diagnosis. The status variable contains the care status of each individual for each month of follow-up. Trajectories have different lengths.

The four states are:

- D: diagnosed, but not in care
- C: in care, but not on treatment
- T: on treatment, but infection not suppressed
- S: on treatment and suppressed infection

## Source

[https://larmarange.github.io/analyse-R/data/care\\_trajectories.RData](https://larmarange.github.io/analyse-R/data/care_trajectories.RData) <https://larmarange.github.io/analyse-R/trajec-toires-de-soins.html>

## See Also

Other datasets: [biofam2](#), [flours](#)

## Examples

```
data(care)
head(care)

plotData(care)

# Individuals has not the same length. In order to compute the encoding,
# we keep individuals with at least 18 months of history and work
# with the 18 first months.
duration <- compute_duration(care)
idToKeep <- as.numeric(names(duration[duration >= 18]))
care2 <- cut_data(care[care$id %in% idToKeep, ], 18)
head(care2)

# It is recommended to increase the number of cores to reduce computation time
set.seed(42)
basis <- create.bspline.basis(c(0, 18), nbasis = 10, norder = 4)
fmca <- compute_optimal_encoding(care2, basis, nCores = 2)

plotEigenvalues(fmca, cumulative = TRUE, normalize = TRUE)
plot(fmca)
plot(fmca, addCI = TRUE)
plotComponent(fmca, addNames = FALSE)
```

---

compute_duration	<i>Compute duration of individuals</i>
------------------	--

---

**Description**

For each individual, compute the duration

**Usage**

```
compute_duration(data)
```

**Arguments**

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
------	--

**Value**

a vector containing the duration of each trajectories

**Author(s)**

Cristian Preda, Quentin Grimonprez

**See Also**

[hist.duration](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# compute duration of each individual
duration <- compute_duration(d_JK)

hist(duration)
```

---

compute\_number\_jumps *Compute the number of jumps*

---

## Description

For each individual, compute the number of jumps performed

## Usage

```
compute_number_jumps(data, countDuplicated = FALSE)
```

## Arguments

`data` data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.  
`countDuplicated` if TRUE, jumps in the same state are counted as jump

## Value

A vector containing the number of jumps for each individual

## Author(s)

Cristian Preda, Quentin Grimonprez

## See Also

[hist.njump](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

## Examples

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# compute the number of jumps
nJump <- compute_number_jumps(d_JK)
```

---

```
compute_optimal_encoding
```

*Compute the optimal encoding for each state*

---

### Description

Compute the optimal encoding for categorical functional data using an extension of the multiple correspondence analysis to a stochastic process.

### Usage

```
compute_optimal_encoding(
  data,
  basisobj,
  computeCI = TRUE,
  nBootstrap = 50,
  propBootstrap = 1,
  nCores = max(1, ceiling(detectCores()/2)),
  verbose = TRUE,
  ...
)
```

### Arguments

<code>data</code>	data.frame containing <code>id</code> , <code>id</code> of the trajectory, <code>time</code> , time at which a change occurs and <code>state</code> , associated state. All individuals must begin at the same time <code>T0</code> and end at the same time <code>Tmax</code> (use <code>cut_data</code> ).
<code>basisobj</code>	basis created using the <code>fda</code> package (cf. <code>create.basis</code> ).
<code>computeCI</code>	if TRUE, perform a bootstrap to estimate the variance of encoding functions coefficients
<code>nBootstrap</code>	number of bootstrap samples
<code>propBootstrap</code>	size of bootstrap samples relative to the number of individuals: <code>propBootstrap * number of individuals</code>
<code>nCores</code>	number of cores used for parallelization. Default is the half of cores.
<code>verbose</code>	if TRUE print some information
<code>...</code>	parameters for <code>integrate</code> function (see details).

### Details

See the vignette for the mathematical background: `RShowDoc("cfda", package = "cfda")`

Extra parameters (...) for the `integrate` function can be:

- *subdivisions* the maximum number of subintervals.
- *rel.tol* relative accuracy requested.
- *abs.tol* absolute accuracy requested.

**Value**

A list containing:

- eigenvalues eigenvalues
- alpha optimal encoding coefficients associated with each eigenvectors
- pc principal components
- F matrix containing the  $F_{(x,i)(y,j)}$
- V matrix containing the  $V_{(x,i)}$
- G covariance matrix of V
- basisobj basisobj input parameter
- pt output of `estimate_pt` function
- bootstrap Only if `computeCI = TRUE`. Output of every bootstrap run
- varAlpha Only if `computeCI = TRUE`. Variance of alpha parameters
- runTime Total elapsed time

**Author(s)**

Cristian Preda, Quentin Grimonprez

**References**

- Deville J.C. (1982) Analyse de données chronologiques qualitatives : comment analyser des calendriers ?, Annales de l'INSEE, No 45, p. 45-104.
- Deville J.C. et Saporta G. (1980) Analyse harmonique qualitative, DIDAY et al. (editors), Data Analysis and Informatics, North Holland, p. 375-389.
- Saporta G. (1981) Méthodes exploratoires d'analyse de données temporelles, Cahiers du B.U.R.O, Université Pierre et Marie Curie, 37-38, Paris.
- Preda C, Grimonprez Q, Vandewalle V. Categorical Functional Data Analysis. The cfda R Package. Mathematics. 2021; 9(23):3074. <https://doi.org/10.3390/math9233074>

**See Also**

[plot.fmca](#) [print.fmca](#) [summary.fmca](#) [plotComponent](#) [get\\_encoding](#)

Other encoding functions: [get\\_encoding\(\)](#), [plot.fmca\(\)](#), [plotComponent\(\)](#), [plotEigenvalues\(\)](#), [predict.fmca\(\)](#), [print.fmca\(\)](#), [summary.fmca\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 5
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(
  n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax,
```

```

  labels = c("A", "C", "G", "T")
)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 5
b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)
summary(encoding)

# plot the optimal encoding
plot(encoding)

# plot the two first components
plotComponent(encoding, comp = c(1, 2))

# extract the optimal encoding
get_encoding(encoding, harm = 1)

```

---

compute\_time\_spent      *Compute time spent in each state*

---

## Description

For each individual, compute the time spent in each state

## Usage

```
compute_time_spent(data)
```

## Arguments

**data**                      data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.

## Value

a matrix with K columns containing the total time spent in each state for each individual

## Author(s)

Cristian Preda, Quentin Grimonprez

## See Also

[boxplot.timeSpent](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# cut at Tmax = 8
d_JK2 <- cut_data(d_JK, Tmax = 8)

# compute time spent by each id in each state
timeSpent <- compute_time_spent(d_JK2)
```

---

convertToCfd

*Convert data to categorical functional data*


---

**Description**

Convert data to categorical functional data

**Usage**

```
convertToCfd(
  x,
  breaks,
  labels = NULL,
  include.lowest = FALSE,
  right = TRUE,
  times = NULL,
  idLabels = NULL,
  nx = 200,
  byrow = FALSE
)
```

**Arguments**

x	matrix or fd object
breaks	either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which x is to be cut.
labels	labels for the levels of the resulting category. By default, labels are constructed using "(a,b]" interval notation. If labels = FALSE, simple integer codes are returned instead of a factor.
include.lowest	logical, indicating if an 'x[i]' equal to the lowest (or highest, for right = FALSE) 'breaks' value should be included.
right	logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa.

times	vector containing values at which fd is to be evaluated
idLabels	vector containing id labels. If NULL it use the names found in the matrix or fd object
nx	Only if x is a fd object. Number of points to evaluate fd
byrow	Only if x is a matrix. If FALSE, one column = one trajectory

**Value**

a data.frame in the cfda format

**See Also**

[flours](#)

Other format: [cut\\_data\(\)](#), [matrixToCfd\(\)](#), [remove\\_duplicated\\_states\(\)](#)

**Examples**

```
# fd object
data("CanadianWeather")
temp <- CanadianWeather$dailyAv[, , "Temperature.C"]
basis <- create.bspline.basis(c(1, 365), nbasis = 8, norder = 4)
fd <- smooth.basis(1:365, temp, basis)$fd

# "Very Cold" = [-50:-10), "Cold" = [-10:0), ...
out <- convertToCfd(fd, breaks = c(-50, -10, 0, 10, 20, 50),
  labels = c("Very Cold", "Cold", "Fresh", "OK", "Hot"),
  times = 1:365)

# matrix
out2 <- convertToCfd(temp, breaks = c(-50, -10, 0, 10, 20, 50),
  labels = c("Very Cold", "Cold", "Fresh", "OK", "Hot"),
  times = 1:365, byrow = FALSE)
```

---

cut\_data

*Cut data to a maximal given time*

---

**Description**

Cut data to a maximal given time

**Usage**

```
cut_data(
  data,
  Tmax,
  prolongLastState = "all",
  NAstate = "Not observed",
  warning = FALSE
)
```

**Arguments**

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
Tmax	max time considered
prolongLastState	list of states to prolong (can be "all"). In the case where the last state of a trajectory is lesser than Tmax, we can assume that this trajectory will be in the same state at time Tmax only if it is an absorbing state. Otherwise it will add NAstate and throw a warning. Set 'prolongLastState = c()' to indicate there is no absorbing state.
NAstate	state value used when the last state is not prolonged.
warning	if TRUE, the function raises warnings when it has prolonged a trajectory with NAstate

**Value**

a data.frame with the same format as data where each individual has Tmax as last time entry.

**Author(s)**

Cristian Preda

**See Also**

Other format: [convertToCfd\(\)](#), [matrixToCfd\(\)](#), [remove\\_duplicated\\_states\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
set.seed(42)
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)
tail(d_JK)

# cut at Tmax = 8
d_JK2 <- cut_data(d_JK, Tmax = 8)
tail(d_JK2)

# do not prolong any state
try(d_JK2 <- cut_data(d_JK, Tmax = 12, prolongLastState = c()))
```

---

estimate_Markov	<i>Estimate transition matrix and spent time</i>
-----------------	--

---

**Description**

Calculates crude initial values for transition intensities by assuming that the data represent the exact transition times of the Markov process.

**Usage**

```
estimate_Markov(data)
```

**Arguments**

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
------	--

**Value**

list of two elements: Q, the estimated transition matrix, and lambda, the estimated time spent in each state

**Author(s)**

Cristian Preda

**See Also**

[plot.Markov](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 100, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# estimation
mark <- estimate_Markov(d_JK)
mark$P
mark$lambda
```

---

estimate_pt	<i>Estimate probabilities to be in each state</i>
-------------	---

---

**Description**

Estimate probabilities to be in each state

**Usage**

```
estimate_pt(data, NAfterTmax = FALSE)
```

**Arguments**

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
NAfterTmax	if TRUE, return NA if $t > T_{max}$ otherwise return the state associated with $T_{max}$ (useful when individuals has different lengths)

**Value**

A list of two elements:

- t: vector of time
- pt: a matrix with  $K$  (= number of states) rows and with  $\text{length}(t)$  columns containing the probabilities to be in each state at each time.

**Author(s)**

Cristian Preda, Quentin Grimonprez

**See Also**

[plot.pt](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

d_JK2 <- cut_data(d_JK, 10)

# estimate probabilities
estimate_pt(d_JK2)
```

---

flours

*Flours dataset*

---

### Description

Resistance of dough during the kneading process

### Usage

```
data(flours)
```

### Format

flours is a list of 3 elements:

- data A matrix of size 241\*115 containing the resistance of dough (measured every 2s) during the kneading process. One dough batch = 1 column
- quality Quality of cookies baked with the associated dough (1=Good, 2=Medium, 3=Bad)
- time time values

### See Also

Other datasets: [biofam2](#), [care](#)

### Examples

```
data(flours)
```

```
matplot(flours$time, flours$data, col = flours$quality, type = "l", lty = 1)
```

```
# convert to categorical data
```

```
flours_cfd <- convertToCfd(flours$data, breaks = c(-Inf, 150, 300, 450, 600, Inf),  
                          times = flours$time)
```

```
plotData(flours_cfd, group = flours$quality)
```

```
# convert to categorical data after projecting in a basis of functions
```

```
basis <- create.bspline.basis(c(0, 480), nbasis = 10)
```

```
flours_fd <- Data2fd(flours$time, flours$data, basis)
```

```
plot(flours_fd)
```

```
flours_cfd2 <- convertToCfd(flours_fd, breaks = c(-Inf, 150, 300, 450, 600, Inf))
```

```
plotData(flours_cfd2, group = flours$quality)
```

---

generate_2State	<i>Generate data following a 2 states model</i>
-----------------	---

---

**Description**

Generate individuals such that each individual starts at time 0 with state 0 and then an unique change to state 1 occurs at a time  $t$  generated using an uniform law between 0 and 1.

**Usage**

```
generate_2State(n)
```

**Arguments**

n	number of individuals
---	-----------------------

**Value**

a data.frame with 3 columns: id, id of the trajectory, time, time at which a change occurs and state, new state.

**Author(s)**

Cristian Preda, Quentin Grimonprez

---

generate_Markov	<i>Generate Markov Trajectories</i>
-----------------	-------------------------------------

---

**Description**

Simulate individuals from a Markov process defined by a transition matrix, time spent in each time and initial probabilities.

**Usage**

```
generate_Markov(  
  n = 5,  
  K = 2,  
  P = (1 - diag(K))/(K - 1),  
  lambda = rep(1, K),  
  pi0 = c(1, rep(0, K - 1)),  
  Tmax = 1,  
  labels = NULL  
)
```

**Arguments**

n	number of trajectories to generate
K	number of states
P	matrix containing the transition probabilities from one state to another. Each row contains positive reals summing to 1.
lambda	time spent in each state
pi0	initial distribution of states
Tmax	maximal duration of trajectories
labels	state names. If NULL, integers are used

**Details**

For one individual, assuming the current state is  $s_j$  at time  $t_j$ , the next state and time is simulated as follows:

1. generate one sample,  $d$ , of an exponential law of parameter  $\text{lambda}[s\_j]$
2. define the next time values as:  $t_{j+1} = t_j + d$
3. generate the new state  $s_{j+1}$  using a multinomial law with probabilities  $Q[s\_j, ]$

**Value**

a data.frame with 3 columns: id, id of the trajectory, time, time at which a change occurs and state, new state.

**Author(s)**

Cristian Preda

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(
  n = 100, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10,
  labels = c("A", "C", "G", "T")
)

head(d_JK)
```

---

get_encoding	<i>Extract the computed encoding</i>
--------------	--------------------------------------

---

**Description**

Extract the encoding as an fd object or as a matrix

**Usage**

```
get_encoding(x, harm = 1, fdObject = FALSE, nx = NULL)
```

**Arguments**

x	Output of <a href="#">compute_optimal_encoding</a>
harm	harmonic to use for the encoding
fdObject	If TRUE returns a fd object else a matrix
nx	(Only if fdObject = TRUE) Number of points to evaluate the encoding

**Details**

The encoding is  $a_x \approx \sum_{i=1}^m \alpha_{x,i} \phi_i$ .

**Value**

a fd object or a list of two elements y, a matrix with nx rows containing the encoding of the state and x, the vector with time values.

**Author(s)**

Cristian Preda

**See Also**

Other encoding functions: [compute\\_optimal\\_encoding\(\)](#), [plot\\_fmca\(\)](#), [plotComponent\(\)](#), [plotEigenvalues\(\)](#), [predict\\_fmca\(\)](#), [print\\_fmca\(\)](#), [summary\\_fmca\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 6
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 6
```

```

b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)

# extract the encoding using 1 harmonic
encodFd <- get_encoding(encoding, fdObject = TRUE)
encodMat <- get_encoding(encoding, nx = 200)

```

---

get\_state                      *Extract the state of each individual at a given time*

---

### Description

Extract the state of each individual at a given time

### Usage

```
get_state(data, t, NAAfterTmax = FALSE)
```

### Arguments

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
t	time at which extract the state
NAAfterTmax	if TRUE, return NA if $t > T_{max}$ otherwise return the state associated with $T_{max}$ (useful when individuals has different lengths)

### Value

a vector containing the state of each individual at time t

### Author(s)

Cristian Preda, Quentin Grimonprez

### Examples

```

# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# get the state of each individual at time t = 6
get_state(d_JK, 6)

```

```
# get the state of each individual at time t = 12 (> Tmax)
get_state(d_JK, 12)
# if NAafterTmax = TRUE, it will return NA for t > Tmax
get_state(d_JK, 12, NAafterTmax = TRUE)
```

---

hist.duration	<i>Plot the duration</i>
---------------	--------------------------

---

## Description

Plot the duration

## Usage

```
## S3 method for class 'duration'
hist(x, breaks = NULL, ...)
```

## Arguments

x	output of <a href="#">compute_duration</a> function
breaks	number of breaks. If not given, use the Sturges rule
...	parameters for <a href="#">geom_histogram</a>

## Value

a ggplot object that can be modified using [ggplot2](#) package.

## Author(s)

Quentin Grimonprez

## See Also

[compute\\_duration](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

## Examples

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)
```

```
# compute duration of each individual
duration <- compute_duration(d_JK)

hist(duration)

# modify the plot using ggplot2
library(ggplot2)
hist(duration) +
  labs(title = "Distribution of the duration")
```

---

hist.njump

*Plot the number of jumps*

---

## Description

Plot the number of jumps

## Usage

```
## S3 method for class 'njump'
hist(x, breaks = NULL, ...)
```

## Arguments

x	output of <a href="#">compute_number_jumps</a> function
breaks	number of breaks. If not given, use the Sturges rule
...	parameters for <a href="#">geom_histogram</a>

## Value

a ggplot object that can be modified using ggplot2 package.

## Author(s)

Quentin Grimonprez

## See Also

[compute\\_number\\_jumps](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

nJump <- compute_number_jumps(d_JK)

hist(nJump)

# modify the plot using ggplot2
library(ggplot2)
hist(nJump, fill = "#984EA3") +
  labs(title = "Distribution of the number of jumps")
```

---

matrixToCfd

*Convert a matrix to a cfda data.frame*


---

**Description**

Convert a matrix to a cfda data.frame

**Usage**

```
matrixToCfd(X, times = NULL, labels = NULL, byrow = FALSE)
```

**Arguments**

X	matrix containing the states
times	time values. If NULL, it uses a sequence of integers starting with 1
labels	id labels. If NULL, it uses the matrix colnames
byrow	if FALSE, one column = one trajectory

**Value**

a data.frame in the cfda format

**See Also**

Other format: [convertToCfd\(\)](#), [cut\\_data\(\)](#), [remove\\_duplicated\\_states\(\)](#)

**Examples**

```
x <- matrix(c("a", "b", "c", "c",
             "c", "a", "a", "a",
             "b", "c", "a", "b"), ncol = 4, byrow = TRUE,
           dimnames = list(NULL, paste0("ind", 1:4)))
matrixToCfd(x)
```

---

plot.fmca

*Plot the optimal encoding*


---

### Description

Plot the optimal encoding

### Usage

```
## S3 method for class 'fmca'
plot(
  x,
  harm = 1,
  states = NULL,
  addCI = FALSE,
  coeff = 1.96,
  col = NULL,
  nx = 128,
  ...
)
```

### Arguments

x	output of <a href="#">compute_optimal_encoding</a> function
harm	harmonic to use for the encoding
states	states to plot (default = NULL, it plots all states)
addCI	if TRUE, plot confidence interval (only when computeCI = TRUE in <a href="#">compute_optimal_encoding</a> )
coeff	the confidence interval is computed with +/- coeff * the standard deviation
col	a vector containing color for each state
nx	number of time points used to plot
...	not used

### Details

The encoding for the harmonic h is  $a_x^{(h)} \approx \sum_{i=1}^m \alpha_{x,i}^{(h)} \phi_i$ .

### Value

a ggplot object that can be modified using ggplot2 package.

### Author(s)

Quentin Grimonprez

**See Also**

Other encoding functions: [compute\\_optimal\\_encoding\(\)](#), [get\\_encoding\(\)](#), [plotComponent\(\)](#), [plotEigenvalues\(\)](#), [predict.fmca\(\)](#), [print.fmca\(\)](#), [summary.fmca\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 6
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 6
b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)

# plot the encoding produced by the first harmonic
plot(encoding)

# modify the plot using ggplot2
library(ggplot2)
plot(encoding, harm = 2, col = c("red", "blue", "darkgreen", "yellow")) +
  labs(title = "Optimal encoding")
```

---

plot.Markov

*Plot the transition graph*


---

**Description**

Plot the transition graph between the different states. A node corresponds to a state with the mean time spent in this state. Each arrow represents the probability of transition between states.

**Usage**

```
## S3 method for class 'Markov'
plot(x, ...)
```

**Arguments**

x                    output of [estimate\\_Markov](#) function  
...                   parameters of [plotmat](#) function from [diagram](#) package (see details).

**Details**

Some useful extra parameters:

- `main` main title.
- `dtext` controls the position of arrow text relative to arrowhead (default = 0.3).
- `relsize` scaling factor for size of the graph (default = 1).
- `box.size` size of label box, one value or a vector with dimension = number of rows of `x$P`.
- `box.cex` relative size of text in boxes, one value or a vector with dimension=number of rows of `x$P`.
- `arr.pos` relative position of arrowhead on arrow segment/curve.

**Value**

No return value, called for side effects

**Author(s)**

Cristian Preda

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 100, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# estimation
mark <- estimate_Markov(d_JK)

# transition graph
plot(mark)
```

---

plot.pt

*Plot probabilities*

---

**Description**

Plot the probabilities of each state at each given time

**Usage**

```
## S3 method for class 'pt'
plot(x, col = NULL, ribbon = FALSE, ...)
```

**Arguments**

x	output of <a href="#">estimate_pt</a>
col	a vector containing color for each state
ribbon	if TRUE, use ribbon to plot probabilities
...	only if ribbon = TRUE, parameter addBorder, if TRUE, add black border to the ribbons.

**Value**

a ggplot object that can be modified using ggplot2 package.

**Author(s)**

Quentin Grimonprez

**See Also**

[estimate\\_pt](#)

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plotData\(\)](#), [statetable\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

d_JK2 <- cut_data(d_JK, 10)

pt <- estimate_pt(d_JK2)

plot(pt, ribbon = TRUE)
```

---

plotComponent

*Plot Components*

---

**Description**

Plot Components

**Usage**

```
plotComponent(
  x,
  comp = c(1, 2),
  addNames = TRUE,
  nudge_x = 0.1,
  nudge_y = 0.1,
  size = 4,
  ...
)
```

**Arguments**

x	output of <a href="#">compute_optimal_encoding</a> function
comp	a vector of two elements indicating the components to plot
addNames	if TRUE, add the id labels on the plot
nudge_x, nudge_y	horizontal and vertical adjustment to nudge labels by
size	size of labels
...	geom_point parameters

**Value**

a ggplot object that can be modified using ggplot2 package.

**Author(s)**

Quentin Grimonprez

**See Also**

Other encoding functions: [compute\\_optimal\\_encoding\(\)](#), [get\\_encoding\(\)](#), [plot\\_fmca\(\)](#), [plotEigenvalues\(\)](#), [predict\\_fmca\(\)](#), [print\\_fmca\(\)](#), [summary\\_fmca\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 6
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 6
b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
```

```

encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)

plotComponent(encoding, comp = c(1, 2))

# modify the plot using ggplot2
library(ggplot2)
plotComponent(encoding, comp = c(1, 2), shape = 23) +
  labs(title = "Two first components")

```

---

plotData

*Plot categorical functional data*


---

### Description

Plot categorical functional data

### Usage

```

plotData(
  data,
  group = NULL,
  col = NULL,
  addId = TRUE,
  addBorder = TRUE,
  sort = FALSE,
  nCol = NULL
)

```

### Arguments

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
group	vector, of the same length as the number individuals of data, containing group index. Groups are displayed on separate plots. If group = NA, the corresponding individuals in data is ignored.
col	a vector containing color for each state (can be named)
addId	If TRUE, add id labels
addBorder	If TRUE, add black border to each individual
sort	If TRUE, id are sorted according to the duration in their first state
nCol	number of columns when group is given

### Value

a ggplot object that can be modified using ggplot2 package. On the plot, each row represents an individual over [0:Tmax]. The color at a given time gives the state of the individual.

**Author(s)**

Cristian Preda, Quentin Grimonprez

**See Also**

Other Descriptive statistics: `boxplot.timeSpent()`, `compute_duration()`, `compute_number_jumps()`, `compute_time_spent()`, `estimate_pt()`, `hist.duration()`, `hist.njump()`, `plot.pt()`, `statetable()`, `summary_cfd()`

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# add a line with time Tmax at the end of each individual
d_JKT <- cut_data(d_JK, Tmax = 10)

plotData(d_JKT)

# modify the plot using ggplot2
library(ggplot2)
plotData(d_JKT, col = c("red", "blue", "green", "brown")) +
  labs(title = "Trajectories of a Markov process")

# use the group variable: create a group with the 3 first variables and one with the others
group <- rep(1:2, c(3, 7))
plotData(d_JKT, group = group)

# use the group variable: remove the id number 5 and 6
group[c(5, 6)] <- NA
plotData(d_JKT, group = group)
```

---

plotEigenvalues

*Plot Eigenvalues*

---

**Description**

Plot Eigenvalues

**Usage**

```
plotEigenvalues(x, cumulative = FALSE, normalize = FALSE, ...)
```

**Arguments**

x                    output of `compute_optimal_encoding` function  
cumulative        if TRUE, plot the cumulative eigenvalues  
normalize         if TRUE eigenvalues are normalized for summing to 1  
...                `geom_point` parameters

**Value**

a ggplot object that can be modified using ggplot2 package.

**Author(s)**

Quentin Grimonprez

**See Also**

Other encoding functions: `compute_optimal_encoding()`, `get_encoding()`, `plot_fmca()`, `plotComponent()`, `predict_fmca()`, `print_fmca()`, `summary_fmca()`

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 6
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 6
b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)

# plot eigenvalues
plotEigenvalues(encoding, cumulative = TRUE, normalize = TRUE)

# modify the plot using ggplot2
library(ggplot2)
plotEigenvalues(encoding, shape = 23) +
  labs(caption = "Jukes-Cantor model of nucleotide replacement")
```

---

predict.fmca	<i>Predict the principal components for new trajectories</i>
--------------	--

---

### Description

Predict the principal components for new trajectories

### Usage

```
## S3 method for class 'fmca'  
predict(  
  object,  
  newdata = NULL,  
  nCores = max(1, ceiling(detectCores()/2)),  
  verbose = TRUE,  
  ...  
)
```

### Arguments

object	output of <a href="#">compute_optimal_encoding</a> function.
newdata	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state. All individuals must begin at the same time T0 and end at the same time Tmax (use <a href="#">cut_data</a> )..
nCores	number of cores used for parallelization. Default is the half of cores.
verbose	if TRUE print some information
...	parameters for <a href="#">integrate</a> function (see details).

### Value

principal components for the individuals

### Author(s)

Quentin Grimonprez

### See Also

Other encoding functions: [compute\\_optimal\\_encoding\(\)](#), [get\\_encoding\(\)](#), [plot.fmca\(\)](#), [plotComponent\(\)](#), [plotEigenvalues\(\)](#), [print.fmca\(\)](#), [summary.fmca\(\)](#)

**Examples**

```

# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
Tmax <- 6
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(
  n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax,
  labels = c("A", "C", "G", "T")
)
d_JK2 <- cut_data(d_JK, Tmax)

# create basis object
m <- 6
b <- create.bspline.basis(c(0, Tmax), nbasis = m, norder = 4)

# compute encoding
encoding <- compute_optimal_encoding(d_JK2, b, computeCI = FALSE, nCores = 1)

# predict principal components
d_JK_predict <- generate_Markov(
  n = 5, K = K, P = PJK, lambda = lambda_PJK, Tmax = Tmax,
  labels = c("A", "C", "G", "T")
)
d_JK_predict2 <- cut_data(d_JK, Tmax)

pc <- predict(encoding, d_JK_predict2, nCores = 1)

```

---

print.fmca

*Print a fmca object*


---

**Description**

Print a fmca object

**Usage**

```

## S3 method for class 'fmca'
print(x, n = 6, ...)

```

**Arguments**

x	fmca object (see <a href="#">compute_optimal_encoding</a> function)
n	maximal number of rows and cols to print
...	Not used.

**Value**

No return value, called for side effects

**See Also**

Other encoding functions: [compute\\_optimal\\_encoding\(\)](#), [get\\_encoding\(\)](#), [plot.fmca\(\)](#), [plotComponent\(\)](#), [plotEigenvalues\(\)](#), [predict.fmca\(\)](#), [summary.fmca\(\)](#)

---

remove\_duplicated\_states

*Remove duplicated states*

---

**Description**

Remove duplicated consecutive states from data. If for an individual there is two or more consecutive states that are identical, only the first is kept. Only time when the state changes are kept.

**Usage**

```
remove_duplicated_states(data, keep.last = TRUE)
```

**Arguments**

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
keep.last	if TRUE, keep the last state for every individual even if it is a duplicated state.

**Value**

data without duplicated consecutive states

**Author(s)**

Quentin Grimonprez

**See Also**

Other format: [convertToCfd\(\)](#), [cut\\_data\(\)](#), [matrixToCfd\(\)](#)

**Examples**

```
data <- data.frame(  
  id = rep(1:3, c(10, 3, 8)), time = c(1:10, 1:3, 1:8),  
  state = c(rep(1:5, each = 2), 1:3, rep(1:3, c(1, 6, 1)))  
)  
out <- remove_duplicated_states(data)
```

---

statetable	<i>Table of transitions</i>
------------	-----------------------------

---

**Description**

Calculates a frequency table counting the number of times each pair of states were observed in successive observation times.

**Usage**

```
statetable(data, removeDiagonal = FALSE)
```

**Arguments**

`data` data.frame containing `id`, `id` of the trajectory, `time`, time at which a change occurs and `state`, associated state.

`removeDiagonal` if TRUE, does not count transition from a state `i` to `i`

**Value**

a matrix of size  $K \times K$  containing the number of transition for each pair

**Author(s)**

Quentin Grimonprez

**See Also**

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [summary\\_cfd\(\)](#)

**Examples**

```
# Simulate the Jukes-Cantor model of nucleotide replacement
K <- 4
PJK <- matrix(1 / 3, nrow = K, ncol = K) - diag(rep(1 / 3, K))
lambda_PJK <- c(1, 1, 1, 1)
d_JK <- generate_Markov(n = 10, K = K, P = PJK, lambda = lambda_PJK, Tmax = 10)

# table of transitions
statetable(d_JK)
```

---

summary\_fmca

*Object Summaries*


---

**Description**

Summary of a fmca object

**Usage**

```
## S3 method for class 'fmca'
summary(object, n = 6, ...)
```

**Arguments**

object	fmca object (see <a href="#">compute_optimal_encoding</a> function)
n	maximal number of rows and cols to print
...	Not used.

**Value**

No return value, called for side effects

**See Also**

Other encoding functions: [compute\\_optimal\\_encoding\(\)](#), [get\\_encoding\(\)](#), [plot\\_fmca\(\)](#), [plotComponent\(\)](#), [plotEigenvalues\(\)](#), [predict\\_fmca\(\)](#), [print\\_fmca\(\)](#)

---

summary\_cfd

*Summary*


---

**Description**

Get a summary of the data.frame containing categorical functional data

**Usage**

```
summary_cfd(data, max.print = 10)
```

**Arguments**

data	data.frame containing id, id of the trajectory, time, time at which a change occurs and state, associated state.
max.print	maximal number of states to display

**Value**

a list containing:

- nRow number of rows
- nInd number of individuals
- timeRange minimal and maximal time value
- uniqueStart TRUE, if all individuals have the same time start value
- uniqueEnd TRUE, if all individuals have the same time start value
- states vector containing the different states
- visit number of individuals visiting each state

**Author(s)**

Quentin Grimonprez

**See Also**

Other Descriptive statistics: [boxplot.timeSpent\(\)](#), [compute\\_duration\(\)](#), [compute\\_number\\_jumps\(\)](#), [compute\\_time\\_spent\(\)](#), [estimate\\_pt\(\)](#), [hist.duration\(\)](#), [hist.njump\(\)](#), [plot.pt\(\)](#), [plotData\(\)](#), [statetable\(\)](#)

**Examples**

```
data(biofam2)
summary_cfd(biofam2)
```

# Index

- \* **Descriptive statistics**
  - boxplot.timeSpent, 5
  - compute\_duration, 8
  - compute\_number\_jumps, 9
  - compute\_time\_spent, 12
  - estimate\_pt, 17
  - hist.duration, 23
  - hist.njump, 24
  - plot.pt, 28
  - plotData, 31
  - statetable, 37
  - summary\_cfd, 38
- \* **datasets**
  - biofam2, 4
  - care, 6
  - flours, 18
- \* **data**
  - biofam2, 4
  - care, 6
  - flours, 18
- \* **encoding functions**
  - compute\_optimal\_encoding, 10
  - get\_encoding, 21
  - plot\_fmca, 26
  - plotComponent, 29
  - plotEigenvalues, 32
  - predict\_fmca, 34
  - print\_fmca, 35
  - summary\_fmca, 38
- \* **format**
  - convertToCfd, 13
  - cut\_data, 14
  - matrixToCfd, 25
  - remove\_duplicated\_states, 36
- \* **package**
  - cfda-package, 2
- biofam2, 3, 4, 7, 18
- boxplot.timeSpent, 5, 8, 9, 12, 17, 23, 24, 29, 32, 37, 39
- care, 5, 6, 18
- cfda-package, 2
- compute\_duration, 3, 6, 8, 9, 12, 17, 23, 24, 29, 32, 37, 39
- compute\_number\_jumps, 3, 6, 8, 9, 12, 17, 23, 24, 29, 32, 37, 39
- compute\_optimal\_encoding, 2, 3, 10, 21, 26, 27, 30, 33–36, 38
- compute\_time\_spent, 3, 5, 6, 8, 9, 12, 17, 23, 24, 29, 32, 37, 39
- convertToCfd, 13, 15, 25, 36
- create.basis, 3, 10
- cut\_data, 10, 14, 14, 25, 34, 36
- estimate\_Markov, 3, 16, 27
- estimate\_pt, 3, 6, 8, 9, 11, 12, 17, 23, 24, 29, 32, 37, 39
- flours, 5, 7, 14, 18
- generate\_2State, 3, 19
- generate\_Markov, 3, 19
- get\_encoding, 3, 11, 21, 27, 30, 33, 34, 36, 38
- get\_state, 22
- hist.duration, 6, 8, 9, 12, 17, 23, 24, 29, 32, 37, 39
- hist.njump, 6, 8, 9, 12, 17, 23, 24, 29, 32, 37, 39
- integrate, 10, 34
- matrixToCfd, 14, 15, 25, 36
- plot\_fmca, 3, 11, 21, 26, 30, 33, 34, 36, 38
- plot.Markov, 16, 27
- plot.pt, 6, 8, 9, 12, 17, 23, 24, 28, 32, 37, 39
- plotComponent, 3, 11, 21, 27, 29, 33, 34, 36, 38
- plotData, 3, 6, 8, 9, 12, 17, 23, 24, 29, 31, 37, 39

`plotEigenvalues`, [3](#), [11](#), [21](#), [27](#), [30](#), [32](#), [34](#),  
[36](#), [38](#)  
`predict.fmca`, [11](#), [21](#), [27](#), [30](#), [33](#), [34](#), [36](#), [38](#)  
`print.fmca`, [11](#), [21](#), [27](#), [30](#), [33](#), [34](#), [35](#), [38](#)  
  
`remove_duplicated_states`, [14](#), [15](#), [25](#), [36](#)  
  
`statetable`, [3](#), [6](#), [8](#), [9](#), [12](#), [17](#), [23](#), [24](#), [29](#), [32](#),  
[37](#), [39](#)  
`summary.fmca`, [3](#), [11](#), [21](#), [27](#), [30](#), [33](#), [34](#), [36](#), [38](#)  
`summary_cfd`, [3](#), [6](#), [8](#), [9](#), [12](#), [17](#), [23](#), [24](#), [29](#), [32](#),  
[37](#), [38](#)