Package 'MEDseq'

May 12, 2020

Type Package

Date 2020-05-12

Title Mixtures of Exponential-Distance Models with Covariates

Version 1.1.1

Description Implements a model-based clustering method for categorical life-course sequences relying on mixtures of exponential-distance models introduced by Murphy et al. (2019) <arXiv:1908.07963>. A range of flexible precision parameter settings corresponding to weighted generalisations of the Hamming distance metric are considered, along with the potential inclusion of a noise component. Gating covariates can be supplied in order to relate sequences to baseline characteristics. Sampling weights are also accommodated. The models are fitted using the EM algorithm and tools for visualising the results are also provided.

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

URL https://cran.r-project.org/package=MEDseq

BugReports https://github.com/Keefe-Murphy/MEDseq

LazyData true

Imports cluster, matrixStats, nnet, seriation, stringdist, TraMineR, WeightedCluster

Suggests knitr, rmarkdown

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Keefe Murphy [aut, cre] (<https://orcid.org/0000-0002-7709-3159>), Thomas Brendan Murphy [ctb] (<https://orcid.org/0000-0002-5668-7046>), Raffaella Piccarreta [ctb], Isobel Claire Gormley [ctb]

Maintainer Keefe Murphy <keefe.murphy@ucd.ie>

Repository CRAN

Date/Publication 2020-05-12 20:10:02 UTC

R topics documented:

`MEDseq-package` *MEDseq: Mixtures of Exponential-Distance Models with Covariates*

Description

Fits MEDseq models: mixtures of Exponential-Distance models with gating covariates and sampling weights. Typically used for clustering categorical/longitudinal life-course sequences

Usage

Fits _MEDseq_ models introduced by Murphy et al. (2019) <arXiv:1908.07963>, i.e. fits mixtures of exponential-distance models for clustering longitudinal life-course sequence data via the EM/CEM algorithm.

A family of parsimonious precision parameter constraints are accommodated. So too are sampling weights. Gating covariates can be supplied via formula interfaces.

The most important function in the **MEDseq** package is: `MEDseq_fit`, for fitting the models via EM/CEM.

`MEDseq_control` allows supplying additional arguments which govern, among other things, controls on the initialisation of the allocations for the EM/CEM algorithm and the various model selection options.

`MEDseq_compare` is provided for conducting model selection between different results from using different covariate combinations &/or initialisation strategies, etc.

`MEDseq_stderr` is provided for computing the standard errors of the coefficients for the covariates in the gating network.

A dedicated plotting function `plot.MEDseq` exists for visualising various aspects of the results, using new methods as well as some existing methods from the **TraMineR** package.

Finally, the package also contains two data sets: `biofam` and `mvad`.

Details

- Type: Package
- Package: MEDseq
- Version: 1.1.1
- Date: 2020-05-12 (this version), 2019-08-24 (original release)
- Licence: GPL (>=2)

See Also

Further details and examples are given in the associated vignette document:
`vignette("MEDseq",package = "MEDseq")`

Author(s)

Keefe Murphy [aut, cre], Thomas Brendan Murphy [ctb], Raffaella Piccarreta [ctb], Isobel Claire Gormley [ctb]

Maintainer: Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

See Also

Useful links:

- https://cran.r-project.org/package=MEDseq
- Report bugs at https://github.com/Keefe-Murphy/MEDseq

Examples

```
# Load the MVAD data
data(mvad)
mvad$Location <- factor(apply(mvad[,5:9], 1L, function(x)
                 which(x == "yes")), labels = colnames(mvad[,5:9]))
mvad          <- list(covariates = mvad[c(3:4,10:14,87)],
                      sequences = mvad[,15L:86L],
                      weights = mvad[,2])
mvad.cov      <- mvad$covariates
states        <- c("EM", "FE", "HE", "JL", "SC", "TR")
labels        <- c("Employment", "FE", "HE", "Joblessness", "School", "Training")
mvad.seq      <- seqdef(mvad$sequences, states=states, labels=labels)

# Fit a range of unweighted models without covariates
# Only consider models with a noise component
# Supply some MEDseq_control() arguments
mod1          <- MEDseq_fit(mvad.seq, G=9:10, modtype=c("CCN", "CUN", "UCN", "UUN"),
```

```
                          algo="CEM", init.z="hc", criterion="asw")

# Fit a model with weights and gating covariates
# Drop the 1st time point which was used to define the weights
mvad.seq2     <- seqdef(mvad$sequences[,-1], states=states, labels=labels)
mod2          <- MEDseq_fit(mvad.seq2, G=10, modtype="UCN", weights=mvad$weights,
                          gating=~ fmpr + gcse5eq + livboth, covars=mvad.cov)

# Examine this model in greater detail
summary(mod2, parameters=TRUE)
summary(mod2$gating)
plot(mod2, "clusters")
```

`biofam` *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.

Usage

```
data(biofam)
```

Format

A data frame with 2000 rows, 16 state variables, 1 id variable and 7 covariates and 2 weights variables.

Details

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 (in columns 10 to 25) sequences of family life states from age 15 to 30 (sequence length is 16) and a series of covariates. 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 states numbered from 0 to 7 are defined from the combination of five basic states, namely Living with parents (Parent), Left home (Left), Married (Marr), Having Children (Child), Divorced:

0 = "Parent"
1 = "Left"
2 = "Married"
3 = "Left+Marr"
4 = "Child"
5 = "Left+Child"

6 = "Left+Marr+Child"
7 = "Divorced"

The covariates are:

sex	
birthyr	(birth year)
nat_1_02	(first nationality)
plingu02	(language of questionnaire)
p02r01	(religion)
p02r04	(religious participation)
cspfaj	(father's social status)
cspmoj	(mother's social status)

Two additional weights variables are inserted for illustrative purpose ONLY (since biofam is a sub-sample of the original data, these weights are not adapted to the actual data):

wp00tbgp	(weights inflating to the Swiss population)
wp00tbgs	(weights respecting sample size)

Source

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

References

Mueller, N. S., Studer, M. and Ritschard, G. (2007). Classification de parcours de vie a l'aide de l'optimal matching. In *XIVe Rencontre de la Societe francophone de classification (SFC 2007), Paris, 5-7 septembre 2007*, pp. 157-160.

Examples

```
data(biofam, package="MEDseq")
```

---

dbs *Compute the Density-based Silhouette*

Description

Computes the Density-based Silhouette for a 'soft' clustering assignment matrix.

Usage

```
dbs(z,
    ztol = 1E-100,
    weights = NULL,
    summ = c("mean", "median"),
    ...)
```

Arguments

z	A numeric matrix such that rows correspond to observations, columns correspond to clusters, and rows sum to 1.
ztol	A small (single, numeric, non-negative) tolerance parameter governing whether small assignment probabilities are treated instead as crisp assignments. Defaults to 1E-100.
weights	An optional numeric vector giving observation-specific weights for computing the (weighted) mean/median DBS (see summ).
summ	A single character string indicating whether the (possibly weighted) "mean" (the default) or "median" DBS should be computed.
...	Catches unused arguments.

Value

A list with the following elements:

silvals A matrix where each row contains the cluster to which each observation belongs in the first column and the observation-specific DBS width in the second column.

msw Depending on the value of summ, either the mean or median DBS width.

wmsw Depending on the value of summ, either the weighted mean or weighted median DBS width.

Note

When calling MEDseq_fit, the summ argument can be passed via the ... construct, in which case it governs both the DBS and ASW criteria.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Menardi, G. (2011). Density-based Silhouette diagnostics for clustering methods. *Statistics and Computing* 21(3): 295-308.

See Also

MEDseq_fit

Examples

```
# Generate a toy z matrix
z <- abs(matrix(rnorm(50), ncol=2))
z <- z/rowSums(z)

# Return the mean DBS width
dbs(z)$msw

# For real sequence data
data(mvad)

mod <- MEDseq_fit(seqdef(mvad[,15:86]), G=10, modtype="UCN", weights=mvad$weight)

dbs(mod$z, weights=mvad$weight)
```

---

get_MEDseq_results *Extract results from a MEDseq model*

Description

Utility function for extracting results of submodels from "MEDseq" object when a range of models were run via MEDseq_fit.

Usage

```
get_MEDseq_results(x,
                   what = c("z", "MAP", "DBS", "ASW"),
                   rank = 1L,
                   criterion = c("dbs", "asw", "bic", "icl",
                                 "aic", "cv", "nec", "loglik"),
                   G = NULL,
                   modtype = NULL,
                   noise = TRUE,
                   ...)
```

Arguments

x	An object of class "MEDseq" generated by MEDseq_fit or an object of class "MEDseqCompare" generated by MEDseq_compare.
what	A character string indicating the desired results to extract.
rank	A number indicating what rank model results should be extracted from, where the rank is determined by criterion. Defaults to 1, i.e. the best model.
criterion	The criterion used to determine the ranking. Defaults to "dbs".
G	Optional argument giving the number of components in the model for which results are desired.

modtype	Optional argument the desired model type for which results are desired.
noise	A logical indicating whether models with a noise component should be considered. Defaults to TRUE.
...	Catches unused arguments.

Details

The arguments rank and criterion are invoked when one or more of the arguments G and modtype are missing. Thus, supplying G and modtype allows rank and criterion to be bypassed entirely.

Value

The desired results extracted from the MEDseq model.

Note

Arguments to this function can be supplied to plot.MEDseq via the ... construct.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

See Also

MEDseq_fit, plot.MEDseq

Examples

```
data(biofam)

# mod <- MEDseq_fit(seqdef(biofam[10:25] + 1L), G=9:10)

# Extract the MAP clustering of the best 9-cluster model according to the asw criterion
# get_MEDseq_results(mod, what="MAP", G=9, criterion="asw")
```

MEDseq_compare *Choose the best MEDseq model*

Description

Takes one or more sets of "MEDseq" models fitted by MEDseq_fit and ranks them according to a specified model selection criterion. It's possible to respect the internal ranking within each set of models, or to discard models within each set which were already deemed sub-optimal. This function can help with model selection via exhaustive or stepwise searches.

Usage

```
MEDseq_compare(...,
               criterion = c("dbs", "asw", "bic",
                             "icl", "aic", "cv", "nec"),
               pick = 10L,
               optimal.only = FALSE)

## S3 method for class 'MEDseqCompare'
print(x,
      index = seq_len(x$pick),
      digits = 3L,
      ...)
```

Arguments

`...`	One or more objects of class `"MEDseq"` outputted by `MEDseq_fit`. All models must have been fit to the same data set. A single *named* list of such objects can also be supplied. Additionally, objects of class `"MEDseqCompare"` outputted by this very function can also be supplied here. This argument is only relevant for the `MEDseq_compare` function and will be ignored for the associated `print` function.
`criterion`	The criterion used to determine the ranking. Defaults to `"dbs"`, the density-based silhouette.
`pick`	The (integer) number of models to be ranked and compared. Defaults to `10L`. Will be constrained by the number of models within the `"MEDseq"` objects supplied via `...` if `optimal.only` is `FALSE`, otherwise constrained simply by the number of `"MEDseq"` objects supplied. Setting `pick=Inf` is a valid way to select all models.
`optimal.only`	Logical indicating whether to only rank models already deemed optimal within each `"MEDeq"` object (`TRUE`), or to allow models which were deemed suboptimal enter the final ranking (`FALSE`, the default). See `details`.
`x, index, digits,`	Arguments required for the associated `print` function: `x` An object of class `"MEDseqCompare"` resulting from a call to `MEDseq_compare`. `index` A logical or numeric vector giving the indices of the rows of the table of ranked models to print. This defaults to the full set of ranked models. It can be useful when the table of ranked models is large to examine a subset via this `index` argument, for display purposes. `digits` The number of decimal places to round model selection criteria to (defaults to 3).

Details

The purpose of this function is to conduct model selection on `"MEDseq"` objects, fit to the same data set, with different combinations of gating network covariates or different initialisation settings.

Model selection will have already been performed in terms of choosing the optimal number of components and MEDseq model type within each supplied set of results, but `MEDseq_compare` will

respect the internal ranking of models when producing the final ranking if optimal.only is FALSE: otherwise only those models already deemed optimal within each "MEDseq" object will be ranked.

As such if two sets of results are supplied when optimal.only is FALSE, the 1st, 2nd and 3rd best models could all belong to the first set of results, meaning a model deemed suboptimal according to one set of covariates could be superior to one deemed optimal under another set of covariates.

Value

A list of class "MEDseqCompare", for which a dedicated print function exists, containing the following elements (each of length pick, and ranked according to criterion, where appropriate):

data	The name of the data set to which the models were fitted.
optimal	The single optimal model (an object of class "MEDseq") among those supplied, according to the chosen criterion.
pick	The final number of ranked models. May be different (i.e. less than) the supplied pick value.
MEDNames	The names of the supplied "MEDseq" objects.
modelNames	The MEDseq model names (denoting the constraints or lack thereof on the precision parameters).
G	The optimal numbers of components.
df	The numbers of estimated parameters.
iters	The numbers of EM/CEM iterations.
bic	BIC values, ranked according to criterion.
icl	TCL values, ranked according to criterion.
aic	AIC values, ranked according to criterion.
cv	Cross-validated log-likelihood values, ranked according to criterion.
nec	NEC values, ranked according to criterion.
dbs	(Weighted) mean/median DBS values, ranked according to criterion.
asw	(Weighted) mean/median ASW values, ranked according to criterion.
loglik	Maximal log-likelihood values, ranked according to criterion.
gating	The gating formulas.
algo	The algorithm used for fitting the model - either "EM", "CEM", "cemEM".
equalPro	Logical indicating whether mixing proportions were constrained to be equal across components.
weights	Logical indicating whether the given model was fitted with sampling weights.
noise	Logical indicating the presence/absence of a noise component. Only displayed if at least one of the compared models has a noise component.
noise.gate	Logical indicating whether gating covariates were allowed to influence the noise component's mixing proportion. Only printed for models with a noise component, when at least one of the compared models has gating covariates.
equalNoise	Logical indicating whether the mixing proportion of the noise component for equalPro models is also equal (TRUE) or estimated (FALSE).

Note

The `criterion` argument here need not comply with the criterion used for model selection within each `"MEDseq"` object, but be aware that a mismatch in terms of `criterion` *may* require the optimal model to be re-fit in order to be extracted, thereby slowing down `MEDseq_compare`.

If random starts had been used via `init.z="random"` the `optimal` model may not necessarily correspond to the highest-ranking model in the presence of a criterion mismatch, due to the randomness of the initialisation.

A dedicated `print` function exists for objects of class `"MEDseqCompare"` and `plot.MEDseq` can also be called on objects of class `"MEDseqCompare"`.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

See Also

`MEDseq_fit`, `plot.MEDseq`

Examples

```
data(biofam)
seqs <- seqdef(biofam[10:25] + 1L)
covs <- biofam[2:3]

# Fit a range of models
# m1   <- MEDseq_fit(seqs, G=9:10)
# m2   <- MEDseq_fit(seqs, G=9:10, gating=~sex, covars=covs)
# m3   <- MEDseq_fit(seqs, G=9:10, gating=~birthyr, covars=covs)
# m4   <- MEDseq_fit(seqs, G=9:10, gating=~sex + birthyr, covars=covs)

# Rank only the optimal models. Examine the best model and its gating network
# (comp <- MEDseq_compare(m1, m2, m3, m4, optimal.only=TRUE))
# (best <- comp$optimal)
# (summ <- summary(best, classification = TRUE, gating=TRUE))

# Examine all models visited, including those already deemed suboptimal
# Only print models with gating covariates & 10 components
# comp2 <- MEDseq_compare(m1, m2, m3, m4, pick=Inf)
# print(comp2, comp2$gating != "None" & comp2$G == 10)
```

MEDseq_control *Set control values for use with MEDseq_fit*

Description

Supplies a list of arguments (with defaults) for use with MEDseq_fit.

Usage

```
MEDseq_control(algo = c("EM", "CEM", "cemEM"),
               init.z = c("kmedoids", "hc", "random", "list"),
               z.list = NULL,
               dist.mat = NULL,
               unique = TRUE,
               criterion = c("dbs", "asw", "bic", "icl", "aic", "cv", "nec"),
               tau0 = NULL,
               noise.gate = TRUE,
               do.nec = FALSE,
               do.cv = FALSE,
               nfolds = 10L,
               nstarts = 1L,
               stopping = c("aitken", "relative"),
               equalPro = FALSE,
               equalNoise = FALSE,
               tol = c(1E-05, 1E-08),
               itmax = c(.Machine$integer.max, 100L),
               opti = c("mode", "medoid", "first", "GA"),
               ordering = c("none", "decreasing", "increasing"),
               MaxNWts = 1000L,
               verbose = TRUE,
               ...)
```

Arguments

algo	Switch controlling whether models are fit using the "EM" (the default) or "CEM" algorithm. The option "cemEM" allows running the EM algorithm starting from convergence of the CEM algorithm.
init.z	The method used to initialise the cluster labels. Defaults to "kmedoids". Other options include Ward hierarchical clustering ("hc"), "random" initialisation, and a user-supplied "list".
z.list	A user supplied list of initial cluster allocation matrices, with number of rows given by the number of observations, and numbers of columns given by the range of component numbers being considered. Only relevant if init.z == "z.list". These matrices are allowed correspond to both soft or hard clusterings, and will be internally normalised so that the rows sum to 1.

`dist.mat` An optional distance matrix to use for initialisation when `init.z` is one of "kmedoids" or "hc". Defaults to a Hamming distance matrix. This is an experimental feature and should only be tampered with by expert users.

`unique` A logical indicating whether the model is fit only to the unique observations (defaults to `TRUE`). When there are covariates, this means all unique combinations of covariate and sequence patterns, otherwise only the sequence patterns.

When `weights` *are not* supplied to `MEDseq_fit` and `isTRUE(unique)`, weights are given by the occurrence frequency of the corresponding sequences, and the model is then fit to the unique observations only.

When `weights` *are* supplied and `isTRUE(unique)`, the weights are summed for each set of duplicate observations and assigned to one retained copy of each corresponding unique sequence. Hence, observations with different weights that are otherwise duplicates are treated as duplicates and significant computational gains can be made.

In both cases, the results will be unchanged, but setting `unique` to `TRUE` can often be much faster.

`criterion` When either `G` or `modtype` is a vector, `criterion` governs how the 'best' model is determined when gathering output. Note that all criteria will be returned in any case, if possible.

`tau0` Prior mixing proportion for the noise component. If supplied, a noise component will be added to the model in the estimation, with `tau0` giving the prior probability of belonging to the noise component for *all* observations. Typically supplied as a scalar in the interval (0, 1), e.g. `0.1`. Can be supplied as a vector when gating covariates are present and `noise.gate` is `TRUE`.

`noise.gate` A logical indicating whether gating network covariates influence the mixing proportion for the noise component, if any. Defaults to `TRUE`, but leads to greater parsimony if `FALSE`. Only relevant in the presence of a noise component; only effects estimation in the presence of gating covariates.

`do.nec` A logical indicating whether the normalised entropy criterion (NEC) should also be computed (for models with more than one component). Defaults to `FALSE`. When `TRUE`, models with `G=1` are fitted always.

`do.cv` A logical indicating whether cross-validated log-likelihood scores should also be computed (see `nfolds`). Defaults to `FALSE` due to significant computational burden incurred.

`nfolds` The number of folds to use when `isTRUE{do.cv}`.

`nstarts` The number of random initialisations to use when `init.z="random"`. Defaults to `1`. Results will be based on the random start yielding the highest estimated log-likelihood.

`stopping` The criterion used to assess convergence of the EM/CEM algorithm. The default (`"aitken"`) uses Aitken's acceleration method, otherwise the `"relative"` change in log-likelihood is monitored (which may be less strict).

`equalPro` Logical variable indicating whether or not the mixing proportions are to be constrained to be equal in the model. Default: `equalPro = FALSE`. Only relevant when `gating` covariates are *not* supplied within `MEDseq_fit`, otherwise ignored. In the presence of a noise component, only the mixing proportions for the

non-noise components are constrained to be equal (by default, see equalNoise), after accounting for the noise component.

equalNoise — Logical which is only invoked when isTRUE(equalPro) and gating covariates are not supplied. Under the default setting (FALSE), the mixing proportion for the noise component is estimated, and remaining mixing proportions are equal; when TRUE all components, including the noise component, have equal mixing proportions.

tol — A vector of length two giving relative convergence tolerances for 1) the log-likelihood of the EM/CEM algorithm, and 2) optimisation in the multinomial logistic regression in the gating network, respectively. The default is c(1e-05,1e-08). If only one number is supplied, it is used as the tolerance in both cases.

itmax — A vector of length two giving integer limits on the number of iterations for 1) the EM/CEM algorithm, and 2) the multinomial logistic regression in the gating network, respectively. The default is c(.Machine$integer.max,100).

opti — Character string indicating how central sequence parameters should be estimated. The default "mode" is exact and thus this experimental argument should only be tampered with by expert users. The option "medoid" fixes the central sequence(s) to be one of the observed sequences (like k-medoids). The other options "first" and "GA" use the first-improvement and genetic algorithms, respectively, to mutate the medoid. Pre-computation of the Hamming distance matrix for the observed sequences speeds-up computation of all options other than "mode".

ordering — Experimental feature that should only be tampered with by experienced users. Allows sequences to be reordered on the basis of the column-wise entropy when opti is "first" or "GA".

MaxNWts — The maximum allowable number of weights in the call to multinom for the multinomial logistic regression in the gating network. There is no intrinsic limit in the code, but increasing MaxNWts will probably allow fits that are very slow and time-consuming. It may be necessary to increase MaxNWts when categorical concomitant variables with many levels are included or the number of components is high.

verbose — Logical indicating whether to print messages pertaining to progress to the screen during fitting. By default is TRUE if the session is interactive, and FALSE otherwise. If FALSE, warnings and error messages will still be printed to the screen, but everything else will be suppressed.

... — Catches unused arguments, and also allows the optional arguments ztol and summ to be passed to dbs (ztol and summ) and the ASW computation (summ).

Details

MEDseq_control is provided for assigning values and defaults within MEDseq_fit. While the criterion argument controls the choice of the optimal number of components and MEDseq model type (in terms of the constraints or lack thereof on the precision parameters), MEDseq_compare is provided for choosing between fits with different combinations of covariates or different initialisation settings.

Value

A named list in which the names are the names of the arguments and the values are the values supplied to the arguments.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

Menardi, G. (2011). Density-based Silhouette diagnostics for clustering methods. *Statistics and Computing* 21(3): 295-308.

See Also

MEDseq_fit, dbs, wcKMedoids, pam, agnes, hclust, seqdist, multinom, MEDseq_compare

Examples

```
data(mvad)

# The CC MEDseq model is equivalent to k-medoids when the CEM
# algorithm is employed, mixing proportions are constrained,
# and the central sequences are restricted to the observed sequences
ctrl   <- MEDseq_control(algo="CEM", equalPro=TRUE, opti="medoid", criterion="asw")

# Note that ctrl must be explicitly named 'ctrl'
mod  <- MEDseq_fit(seqdef(mvad[,15:86]), G=8, modtype="CC", weights=mvad$weight, ctrl=ctrl)

# Alternatively, specify the control arguments directly
mod   <- MEDseq_fit(seqdef(mvad[,15:86]), G=8, modtype="CC", weights=mvad$weight,
                    algo="CEM", equalPro=TRUE, opti="medoid", criterion="asw")

# Note that supplying control arguments via a mix of the ... construct and the named argument
# 'control' or supplying MEDseq_control output without naming it 'control' can throw an error
```

MEDseq_fit *MEDseq: Mixtures of Exponential-Distance Models with Covariates*

Description

Fits MEDseq models: mixtures of Exponential-Distance models with gating covariates and sampling weights. Typically used for clustering categorical/longitudinal life-course sequences. Additional arguments are available via the function MEDseq_control.

Usage

```
MEDseq_fit(seqs,
           G = 1L:9L,
           modtype = c("CC", "UC", "CU", "UU",
                       "CCN", "UCN", "CUN", "UUN"),
           gating = NULL,
           weights = NULL,
           ctrl = MEDseq_control(...),
           covars = NULL,
           ...)

## S3 method for class 'MEDseq'
summary(object,
        classification = TRUE,
        parameters = FALSE,
        network = FALSE,
        ...)

## S3 method for class 'MEDseq'
print(x,
      digits = 3L,
      ...)
```

Arguments

seqs	A state-sequence object of class "stslist" as created by the seqdef function in the **TraMineR** package.
G	A positive integer vector specifying the numbers of mixture components (clusters) to fit. Defaults to G=1:9.
modtype	A vector of character strings indicating the type of MEDseq models to be fitted, in terms of the constraints or lack thereof on the precision parameters. By default, all valid model types are fitted (except some only where G > 1 or G > 2, see note). The models are named "CC", "CU", "UC", "UU", CCN", "CUN", "UCN", and "UUN". The first letter denotes whether the precision parameters are constrained/unconstrained across clusters. The second letter denotes whether the precision parameters are constrained/unconstrained across sequence positions (i.e. time points). The third letter denotes whether one of the components is constrained to have zero-precision/infinite variance. Such a noise component assumes sequences in that cluster follow a uniform distribution.
gating	A formula for determining the model matrix for the multinomial logistic regression in the gating network when fixed covariates enter the mixing proportions. Defaults to ~1, i.e. no covariates. This will be ignored where G=1. Continuous, categorical, and/or ordinal covariates are allowed. Logical covariates will be coerced to factors. Interactions, transformations, and higher order terms are permitted: the latter **must** be specified explicitly using the AsIs operator (I). The specification of the LHS of the formula is ignored. Intercept terms are included by default.

`weights` Optional numeric vector containing observation-specific sampling weights, which are accounted for in the model fitting and other functions where applicable. See the `unique` argument to `MEDseq_control` to see how incorporating weights also yields computational benefits.

`ctrl` A list of control parameters for the EM/CEM and other aspects of the algorithm. The defaults are set by a call to `MEDseq_control`.

`covars` An optional data frame (or a matrix with named columns) in which to look for the covariates in the `gating` network formula, if any. If not found in `covars`, any supplied `gating` covariates are taken from the environment from which `MEDseq_fit` is called. Try to ensure the names of variables in `covars` do not match any of those in `seqs`.

`...` Catches unused arguments (see `MEDseq_control`).

`x, object, digits, classification, parameters, network`
Arguments required for the `print` and `summary` functions: `x` and `object` are objects of class `"MEDseq"` resulting from a call to `MEDseq_fit`, while `digits` gives the number of decimal places to round to for printing purposes (defaults to 3). `classification`, `parameters`, and `network` are logicals which govern whether a table of the MAP classification of observations, the mixture component parameters, and the gating network coefficients are printed, respectively.

Details

The function effectively allows 8 different MEDseq precision parameter settings for models with or without gating network covariates. By constraining the mixing proportions to be equal (see `equalPro` in `MEDseq_control`) an extra special case is facilitated in the latter case.

While model selection in terms of choosing the optimal number of components and the MEDseq model type is performed within `MEDseq_fit`, using one of the `criterion` options within `MEDseq_control`, choosing between multiple fits with different combinations of covariates or different initialisation settings can be done by supplying objects of class `"MEDseq"` to `MEDseq_compare`.

Value

A list (of class `"MEDseq"`) with the following named entries (of which some may be missing, depending on the `criterion` employed), mostly corresponding to the chosen optimal model (as determined by the `criterion` within `MEDseq_control`):

`call` The matched call.

`data` The input data, `seqs`.

`modtype` A character string denoting the MEDseq model type at which the optimal `criterion` occurs.

`G` The optimal number of mixture components according to `criterion`.

`params` A list with the following named components:

`theta` A matrix with `G` rows and P columns, where P is the number of sequence positions, giving the central sequences of each cluster. The mean of the noise component is not reported, as it does not contribute in any way to the likelihood.

`lambda` A matrix of precision parameters. Will contain `1` row if the 1st letter of `modtype` is "C" and `G` columns otherwise. Will contain `1` column if the 2nd letter of `modtype` is "C" and P columns otherwise, where P is the number of sequence positions. Precision parameter values of zero are reported for the noise component, if any. Note that values of `Inf` are also possible, corresponding to zero-variance, which is most likely under the "UU" or "UUN" models.

`tau` The mixing proportions: either a vector of length `G` or, if `gating` covariates were supplied, a matrix with an entry for each observation (rows) and component (columns).

`gating` An object of class `"MEDgating"` and either `"multinom"` or `"glm"` (for single-component models) giving the `multinom` regression coefficients of the `gating` network. If `gating` covariates were *NOT* supplied (or the best model has just one component), this corresponds to a RHS of `~1`, otherwise the supplied `gating` formula. As such, a fitted `gating` network is always returned even in the absence of supplied covariates. If there is a noise component (and the option `noise.gate=TRUE` is invoked), its coefficients are those for the *last* component. **Users are cautioned against making inferences about statistical significance from summaries of the coefficients in the gating network. Users are instead advised to use the function** `MEDseq_stderr`.

`z` The final responsibility matrix whose `[i,k]`-th entry is the probability that observation *i* belongs to the *k*-th component. If there is a noise component, its values are found in the *last* column.

`MAP` The vector of cluster labels for the chosen model corresponding to `z`, i.e. `max.col(z)`. Observations belonging to the noise component, if any, will belong to component `0`.

`DBS` A matrix of *all* (weighted) mean/median DBS values with `length{G}` rows and `length(modtype)` columns. See `note` and `dbs`.

`DBSvals` A list of lists giving the observation-specific DBS values for *all* fitted models. The first level of the list corresponds to numbers of components, the second to the MEDseq model types.

`dbs` The (weighted) mean/median DBS value corresponding to the optimal model. May not necessarily be the optimal DBS.

`dbsvals` Observation-specific DBS values corresponding to the optimum model, which may not be optimal in terms of DBS.

`ASW` A matrix of *all* (weighted) mean/median ASW values with `length{G}` rows and `length(modtype)` columns. See `note`.

`ASWvals` A list of lists giving the observation-specific ASW values for *all* fitted models. The first level of the list corresponds to numbers of components, the second to the MEDseq model types.

`asw` The (weighted) mean/median ASW value corresponding to the optimal model. May not necessarily be the optimal ASW.

`aswvals` Observation-specific ASW values corresponding to the optimum model, which may not be optimal in terms of ASW.

BIC	A matrix of *all* BIC values with length{G} rows and length(modtype) columns. See note.
ICL	A matrix of *all* ICL values with length{G} rows and length(modtype) columns. See note.
AIC	A matrix of *all* AIC values with length{G} rows and length(modtype) columns. See note.
LOGLIK	A matrix of *all* maximal log-likelihood values with length{G} rows and length(modtype) columns. See note.
DF	A matrix giving the numbers of estimated parameters (i.e. the number of 'used' degrees of freedom) for *all* visited models, with length{G} rows and length(modtype) columns. Subtract these numbers from the sample size to get the degrees of freedom. See note.
ITERS	A matrix giving the total number of EM/CEM iterations for *all* visited models, with length{G} rows and length(modtype) columns. See note.
NEC	A matrix of *all* NEC values with length{G} rows and length(modtype) columns, if available. See note and the argument do.nec to MEDseq_control.
CV	A matrix of *all* cross-validated log-likelihood values with length{G} rows and length(modtype) columns, if available. See note and the arguments do.cv and nfolds to MEDseq_control.
bic	The BIC value corresponding to the optimal model. May not necessarily be the optimal BIC.
icl	The ICL value corresponding to the optimal model. May not necessarily be the optimal ICL.
aic	The AIC value corresponding to the optimal model. May not necessarily be the optimal AIC.
loglik	The vector of increasing log-likelihood values for every EM/CEM iteration under the optimal model. The last element of this vector is the maximum log-likelihood achieved by the parameters returned at convergence.
df	The number of estimated parameters in the optimal model (i.e. the number of 'used' degrees of freedom). Subtract this number from the sample size to get the degrees of freedom.
iters	The total number of EM/CEM iterations for the optimal model.
nec	The NEC value corresponding to the optimal model, if available. May not necessarily be the optimal NEC.
cv	The cross-validated log-likelihood value corresponding to the optimal model, if available. May not necessarily be the optimal one.
ZS	A list of lists giving the z matrices for *all* fitted models. The first level of the list corresponds to numbers of components, the second to the MEDseq model types.
uncert	The uncertainty associated with the classification.
covars	A data frame gathering the set of covariates used in the gating network, if any. Will contain zero columns in the absence of gating covariates. Supplied gating covariates will be excluded if the optimal model has only one component. May have fewer columns than covariates supplied via the covars argument also, as only the included covariates are gathered here.

Note

Where DBS, ASW, BIC, ICL, AIC, LOGLIK, DF, ITERS, NEC, and CV contain NA entries, this corresponds to a model which was not run; for instance a UU model is never run for single-component models as it is equivalent to CU, while a UCN model is never run for two-component models as it is equivalent to CCN. As such, one can consider the value as not really missing, but equivalent to the corresponding value. On the other hand, -Inf represents models which were terminated due to error, for which a log-likelihood could not be estimated. These objects all inherit the class "MEDCriterion" for which a dedicated printing functions exists.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

See Also

seqdef, MEDseq_control, MEDseq_compare, plot.MEDseq, MEDseq_stderr, I

Examples

```
# Load the MVAD data
data(mvad)
mvad$Location <- factor(apply(mvad[,5:9], 1L, function(x)
                 which(x == "yes")), labels = colnames(mvad[,5:9]))
mvad          <- list(covariates = mvad[c(3:4,10:14,87)],
                      sequences = mvad[,15L:86L],
                      weights = mvad[,2])
mvad.cov      <- mvad$covariates
states        <- c("EM", "FE", "HE", "JL", "SC", "TR")
labels        <- c("Employment", "Further Education", "Higher Education",
                   "Joblessness", "School", "Training")
mvad.seq      <- seqdef(mvad$sequences, states=states, labels=labels)

# Fit an exponential-distance model without clustering
mod0          <- MEDseq_fit(mvad.seq, G=1)

# Fit a range of unweighted mixture models without covariates
# Only consider models with a noise component
# Supply some MEDseq_control() arguments

# mod1        <- MEDseq_fit(mvad.seq, G=9:10, modtype=c("CCN", "CUN", "UCN", "UUN"),
#                           algo="CEM", init.z="hc", criterion="asw")

# Fit a model with weights and gating covariates
# Drop the 1st time point which was used to define the weights
mvad.seq2     <- seqdef(mvad$sequences[,-1], states=states, labels=labels)
```

```
mod2          <- MEDseq_fit(mvad.seq2, G=10, modtype="UCN", weights=mvad$weights,
                            gating=~ fmpr + gcse5eq + livboth, covars=mvad.cov)

# Examine this model in greater detail
summary(mod2, parameters=TRUE)
summary(mod2$gating)
plot(mod2, "clusters")
```

MEDseq_meantime *Compute the mean time spent in each sequence category*

Description

Computes the mean time (per cluster) spent in each sequence category (i.e. state value) for a fitted MEDseq model.

Usage

```
MEDseq_meantime(x,
                MAP = FALSE,
                norm = TRUE)
```

Arguments

x	An object of class "MEDseq" generated by MEDseq_fit or an object of class "MEDseqCompare" generated by MEDseq_compare.
MAP	A logical indicating whether to use the MAP classification in the computation of the averages, or the 'soft' clustering assignments given by x$z. Defaults to FALSE, but is always TRUE for models fitted by the CEM algorithm (see MEDseq_control).
norm	A logical indicating whether the mean times are normalised to sum to the sequence length within each cluster (defaults to TRUE). Otherwise, when FALSE, entries give the total (weighted) number of times a given sequence category was observed in a given cluster.

Details

Models with weights, covariates, &/or a noise component are also accounted for.

Value

A matrix with sequence category and cluster-specific mean times, giving clusters on the rows, corresponding cluster sizes in the first column, and sequence categories in the remaining columns.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

See Also

MEDseq_fit, MEDseq_control

Examples

```
data(biofam)

mod <- MEDseq_fit(seqdef(biofam[10:25] + 1L), G=10, modtype="UUN")

MEDseq_meantime(mod)
MEDseq_meantime(mod, MAP=TRUE, norm=FALSE)
```

MEDseq_news *Show the NEWS file*

Description

Show the NEWS file of the MEDseq package.

Usage

```
MEDseq_news()
```

Value

The MEDseq NEWS file, provided the session is interactive.

Examples

```
MEDseq_news()
```

MEDseq_stderr *MEDseq gating network standard errors*

Description

Computes standard errors of the gating network coefficients in a fitted MEDseq model using either the Weighted Likelihood Bootstrap or Jackknife methods.

Usage

```
MEDseq_stderr(mod,
              method = c("WLBS", "Jackknife"),
              N = 1000L,
              symmetric = TRUE)
```

Arguments

mod	A fitted model of class "MEDseq" generated by MEDseq_fit.
method	The method used to compute the standard errors (defaults to "WLBS", the Weighted Likelihood Bootstrap).
N	The (integer) number of samples to use when the "WLBS" method is employed. Defaults to 1000L. Not relevant when method="Jackknife", in which case N is always the number of observations. Must be > 1.
symmetric	A logical indicating whether symmetric draws from the uniform Dirichlet distribution are used for the WLBS method in the presence of existing sampling weights. Defaults to TRUE; when FALSE, the concentration parameters of the Dirichlet distribution are given by the sampling weights. Only relevant when method="WLBS" for models with existing sampling weights.

Value

A list with the following two elements:

Coefficients The original matrix of estimated coefficients (coef(mod$gating)).

Std. Errors The matrix of corresponding standard error estimates.

Note

A progress bar is displayed as the function iterates over the N samples. The function may take a long time to run for large N. The function terminates immediately if mod$G == 1.

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

O'Hagan, A., Murphy, T. B., Scrucca, L., and Gormley, I. C. (2019). Investigation of parameter uncertainty in clustering using a Gaussian mixture model via jackknife, bootstrap and weighted likelihood bootstrap. *Computational Statistics* 34(4): 1779-1813.

See Also

MEDseq_fit

Examples

```
# Load the MVAD data
data(mvad)
mvad$Location <- factor(apply(mvad[,5:9], 1L, function(x)
                 which(x == "yes")), labels = colnames(mvad[,5:9]))
mvad          <- list(covariates = mvad[c(3:4,10:14,87)],
                      sequences = mvad[,15L:86L],
                      weights = mvad[,2])
mvad.cov      <- mvad$covariates
states        <- c("EM", "FE", "HE", "JL", "SC", "TR")
labels        <- c("Employment", "Further Education", "Higher Education",
                   "Joblessness", "School", "Training")
mvad.seq      <- seqdef(mvad$sequences, states=states, labels=labels)

# Fit a model with weights and gating covariates
# Drop the 1st time point which was used to define the weights
# mvad.seq2   <- seqdef(mvad$sequences[,-1], states=states, labels=labels)
# mod         <- MEDseq_fit(mvad.seq2, G=10, modtype="UCN", weights=mvad$weights,
#                           gating=~ fmpr + gcse5eq + livboth, covars=mvad.cov)

# Estimate standard errors using 100 WLBS samples
# (std        <- MEDseq_stderr(mod, N=100))
```

`mvad` *MVAD: Transition from school to work*

Description

The data comes from a study by McVicar and Anyadike-Danes on transition from school to work. The data consist of static background characteristics and a time series sequence of 72 monthly labour market activities for each of a cohort of 712 individuals in the Status Zero Survey. The individuals were followed up from July 1993 to June 1999. The monthly states are recorded in columns 15 (`Jul.93`) to 86 (`Jun.99`).

Usage

```
data(mvad)
```

Format

A data frame containing 712 rows, 72 state variables, 1 id variable and 13 covariates.

Details

States are:

employment	(EM)
FE	further education (FE)
HE	higher education (HE)
joblessness	(JL)
school	(SC)
training	(TR)

The data set contains also ids (id) and sample weights (weights) as well as the following binary covariates:

male
catholic
Belfast, N.Eastern, Southern, S.Eastern, Western (location of school, one of five Education and Library Board areas in Northern Ireland)
Grammar (type of secondary education, 1=grammar school)
funemp (father's employment status at time of survey, 1=father unemployed)
gcse5eq (qualifications gained by the end of compulsory education, 1=5+ GCSEs at grades A-C, or equivalent)
fmpr (SOC code of father's current or most recent job at time of survey, 1=SOC1 (professional, managerial or related))
livboth (living arrangements at time of first sweep of survey (June 1995), 1=living with both parents)

Source

McVicar and Anyadike-Danes (2002)

References

McVicar, D. (2000). Status 0 four years on: young people and social exclusion in Northern Ireland. *Labour Market Bulletin*, 14, 114-119.

McVicar, D. and Anyadike-Danes, M. (2002). Predicting successful and unsuccessful transitions from school to work by using sequence methods. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 165(2): 317-334.

Examples

```
data(mvad, package="MEDseq")
```

`plot.MEDseq` *Plot MEDseq results*

Description

Produces a range of plots of the results of fitted `MEDseq` models.

Usage

```
## S3 method for class 'MEDseq'
plot(x,
       type = c("clusters", "mean", "precision", "gating",
                "dbs", "asw", "bic", "icl", "aic", "nec",
                "cv", "LOGLIK", "dbsvals", "aswvals",
                "uncert.bar", "uncert.profile", "loglik",
                "d", "f", "Ht", "i", "I"),
       seriate = c("observations", "both", "clusters", "none"),
       preczero = TRUE,
       log.scale = FALSE,
       ...)
```

Arguments

`x`	An object of class `"MEDseq"` generated by `MEDseq_fit` or an object of class `"MEDseqCompare"` generated by `MEDseq_compare`.
`type`	A character string giving the type of plot requested:
	`"clusters"` Visualise the data set with sequences grouped into their respective clusters. See `seriate`.
	`"mean"` Visualise the central sequences. See `seriate`. The central sequence for the noise component, if any is not shown as it doesn't contribute in any way to the likelihood.
	`"precision"` Visualise the central sequence parameters in the form of a heatmap. Values of `0` and `Inf` are shown in `grey` and `black` respectively (see `log.scale`).
	`"gating"` Visualise the gating network, i.e. the observation index (by default) against the mixing proportions for that observation, coloured by cluster. See `seriate`. The optional argument `x.axis` can be passed via the `...` construct to change the x-axis against which mixing proportions are plotted (only advisable for models with a single gating network covariate, when `x.axis` is a quantity related to the gating network of the fitted model).
	`"dbs"` Plots all (weighted) mean/median DBS values in a fitted `MEDseq` object.
	`"asw"` Plots all (weighted) mean/median ASW values in a fitted `MEDseq` object.
	`"bic"` Plots all BIC values in a fitted `MEDseq` object.

"icl" Plots all ICL values in a fitted MEDseq object.

"aic" Plots all AIC values in a fitted MEDseq object.

"nec" Plots all NEC values in a fitted MEDseq object.

"cv" Plots all cross-validated log-likelihood values in a fitted MEDseq object.

"LOGLIK" Plots all maximal log-likelihood values in a fitted MEDseq object.

"dbsvals" Silhouette plot using observations-specific DBS values for the optimal model (coloured by cluster).

"aswvals" Silhouette plot using observations-specific ASW values for the optimal model (coloured by cluster).

"uncert.bar" Plot the observation-specific clustering uncertainties in the form of a bar plot.

"uncert.profile" Plot the observation-specific clustering uncertainties in the form of a profile plot.

"loglik" Plot the log-likelihood at every iteration of the EM/CEM algorithm used to fit the model.

Also available are the following options which act as wrappers to types of plots produced by the seqplot function in the **TraMineR** package.

"d" State distribution plots (by cluster).

"f" Sequence frequency plots (by cluster).

"Ht" Transversal entropy plots (by cluster).

"i" Selected sequence index plots (by cluster).

"I" Whole set index plots (by cluster).

seriate Switch indicating whether seriation should be used to improve the visualisation by re-ordering the "observations" within clusters (the default), the "clusters", "both", or "none". See seriate. The options "clusters" and "both" are only invoked when type is one of "clusters", "mean", "precision", "gating", "d", "f", "Ht", "i", or "I". Additionally, the options "observations" and "both" are only invoked when type is one of "clusters" or "gating".

preczero Logical indicating whether central sequence parameter positions corresponding to zero-valued precision parameters (if any!) should also be suppressed for the non-noise components. Defaults to TRUE; noise-component means are never shown regardless of the value of preczero.

log.scale Logical indicating whether precision parameter heatmaps should be plotted on the log-scale when type="precision". The behaviour of 0 or Inf values remains unchanged; only strictly-positive finite entries are effected. Heavily imbalanced values are more likely for the "UU" and "UUN" model types, thus log.scale defaults to TRUE in those instances and FALSE otherwise.

... Catches unused arguments, and allows arguments to get_MEDseq_results to be passed when type is one of "clusters", "dbsvals", "aswvals", "uncert.bar", "uncert.profile", "d", "f", "Ht", "i", or "I", as well as the x.axis argument when type="gating". Also allows additional arguments to the TraMineR function seqplot to be used.

Details

The type options related to model selection criteria plot values for *all* fitted models in the "MEDseq" object x. The remaining type options plot results for the optimal model, by default. However, arguments to get_MEDseq_results can be passed via the ... construct to plot corresponding results for suboptimal models in x when type is one of "clusters", "d", "f", "Ht", "i", or "I".

Value

The visualisation according to type of the results of a fitted MEDseq model.

Note

Every type of plot respects the sampling weights, if any. Those related to seqdef plots from **TraMineR** may be too wide to display in the preview panel. The same is also true when type is "dbsvals" or "aswvals".

Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

References

Murphy, K., Murphy, T. B., Piccarreta, R., and Gormley, I. C. (2019). Clustering longitudinal life-course sequences using mixtures of exponential-distance models. *To appear*. <arXiv:1908.07963>.

Gabadinho, A., Ritschard, G., Mueller, N. S., and Studer, M. (2011). Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software* 40(4): 1-37.

See Also

MEDseq_fit, seqplot, dbs, get_MEDseq_results, seriate

Examples

```
# Load the MVAD data
data(mvad)
mvad$Location <- factor(apply(mvad[,5:9], 1L, function(x)
                   which(x == "yes")), labels = colnames(mvad[,5:9]))
mvad          <- list(covariates = mvad[c(3:4,10:14,87)],
                      sequences = mvad[,15L:86L],
                      weights = mvad[,2])
mvad.cov      <- mvad$covariates
states        <- c("EM", "FE", "HE", "JL", "SC", "TR")
labels        <- c("Employment", "Further Education", "Higher Education",
                   "Joblessness", "School", "Training")
mvad.seq      <- seqdef(mvad$sequences, states=states, labels=labels)

# Fit an exponential-distance model without clustering
mod0          <- MEDseq_fit(mvad.seq, G=1)
```

```
# Show the central sequence and precision parameters
plot(mod0, type="mean")
plot(mod0, type="precision")

# Fit a range of unweighted mixture models without covariates
# Only consider models with a noise component
# mod1        <- MEDseq_fit(mvad.seq, G=9:10, modtype=c("CCN", "CUN", "UCN", "UUN"))

# Plot the DBS values for all fitted models
# plot(mod1, "dbs")

# Plot the clusters of the optimal model
# plot(mod1, "clusters")

# Plot the clusters of the best UUN model
# plot(mod1, "clusters", modtype="UUN")

# Fit a model with weights and gating covariates
# Drop the 1st time point which was used to define the weights
# mvad.seq2   <- seqdef(mvad$sequences[,-1], states=states, labels=labels)
# mod2        <- MEDseq_fit(mvad.seq2, G=10, modtype="UCN", weights=mvad$weights,
#                           gating=~ fmpr + gcse5eq + livboth, covars=mvad.cov)

# Plot the central sequences & precision parameters of this model
# plot(mod2, "mean")
# plot(mod2, "precision")

# Plot the clustering uncertainties in the form of a barplot
# plot(mod2, "uncert.bar")

# Plot the observation-specific DBS values and the sequence frequencies by cluster
# Note that these plots may not display properly in the preview panel
# plot(mod2, "dbsvals")
# plot(mod2, "Ht")
```

Index