Visualization tools in the `seqHMM` package

Satu Helske
University of Oxford, UK

July 5, 2023

Contents

1 Introduction

Visualization is a powerful tool throughout the analysis process from the first glimpses into the data to exploration and finally to presentation of the results. This vignette is supplementary material to the paper Helske and Helske (2019) on the `seqHMM` package; here we discuss visualization of sequence data and hidden Markov models in more detail and give a more examples on the plotting features of the package.

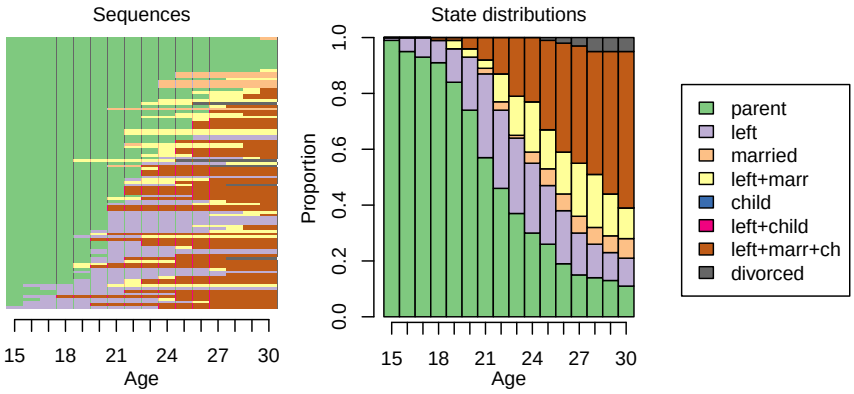

Figure 1: Sequence index plot (left) and state distribution plot (right) of annual family states for 100 individuals from the `biofam` data.

1.1 Visualizing sequences

There are many options for graphical description of sequence data. Most of them either represent sequences or summarize them. *Sequence index plot* is the most commonly used example of the former (see an illustration on the left-hand side of Figure 1). Such a graph was proposed by Scherer (2001) to show the observations of each subject in the order they appear, illustrating different states with different colours. The horizontal axis shows the time points while individuals are represented on the vertical axis; thus, each horizontal line shows the sequence of one individual.

When the number of subjects is moderate, sequence index plots give an accurate representation of the data, offering an overview on the timing of transitions and on the durations of different episodes. Sequence index plots get more complex to comprehend when the number of individuals and states increases. Sequence analysis with clustering eases interpretation by grouping similar trajectories together. Piccarreta and Lior (2010) suggested using multidimensional scaling for ordering sequences more meaningfully (similar sequences close to each other). Piccarreta (2012) proposed smoothing techniques that reduce individual noise. Similar sequences are summarized into artificial sequences that are representative to the data. Gabadinho, Ritschard, Müller, and Studer (2011) introduced *representative sequence plots* where only a few of the most representative sequences (observed or artificial) are shown. A similar approach, *relative frequency sequence plot*, was introduced by Fasang and Liao (2014). The idea is to find a representative sequence (the medoid) in equal-sized neighbourhoods to represent the relative frequencies in the data.

State distribution plots (also called tempograms or chronograms; Billari and Piccarreta, 2005; Widmer and Ritschard, 2009) summarize information in the whole data. Such graphs show the change in the prevalence of states in the

course of time (see the right-hand side of Figure 1). Also here, the horizontal axis represents time but vertical axis is now a percentage scale. These plots simplify the overall patterns but do not give information on transitions between different states. Other summary plots include, e.g., *transversal entropy plots* (Billari, 2001), which describe how evenly states are distributed at a given time point, and *mean time plots* (Gabadinho et al., 2011), which show the mean time spent in each state across the time points.

Visualizing multichannel data is not a straightforward task. The `TraMineR` package (Gabadinho et al., 2011) provides nice plotting options and summaries for simple sequence data, but there are no easy options for multichannel data. Combining states into a single-channel representation often works well if the state space is small and states at each time point are either completely observed or completely missing. In other cases it can be preferable to preserve the multichannel structure. The *stacked sequence plot* shows sequence data plotted separately for each channel (see Figure 2). The order of the subjects is kept the same in each plot and the plots are stacked on top of each other. Since the time axes are horizontally aligned, comparing timing in different life domains should be relatively easy. This approach also protects the privacy of the subjects; even though all data are shown, combining information across channels for a single individual is difficult unless the number of subjects is small. State distribution plots can then be used to show information on the prevalence and timing of combined states on a more general level.

1.2 Visualizing hidden Markov models

Markovian models are often visualized as directed graphs where vertices (nodes) present states and edges (arrows, arcs) show transition probabilities between states. We have extended this basic graph in the hidden Markov model framework by presenting hidden states as pie charts, with emission probabilities as slices, and by adjusting the thickness of edges according to transition probabilities. Such graph allows for presenting a complex model in a very efficient way, guiding the viewer to the most important aspects of the model. The graph shows the essence of the hidden states and the dynamics between them.

Figure 3 illustrates a HMM with five hidden states for the multichannel version of the `biofam` data. Following the common convention, hidden states are presented as vertices and transition probabilities are shown as edges. Initial state probabilities are given below the respective vertices. Almost all individuals (99%) start from the first hidden state of living home unmarried and without children.

Vertices are drawn as pie charts where the slices represent emitted observations or – in a multichannel case – combinations of observed states across channels. The size of the slice is proportional to the emission probability of the observed state (or in a multichannel model, the product of the emission probabilities across channels). In this model, the hidden states are very close to observed states, the largest emission probabilities for the combined observations are 0.94 or higher in all states. For emphasizing the relevant information, ob-

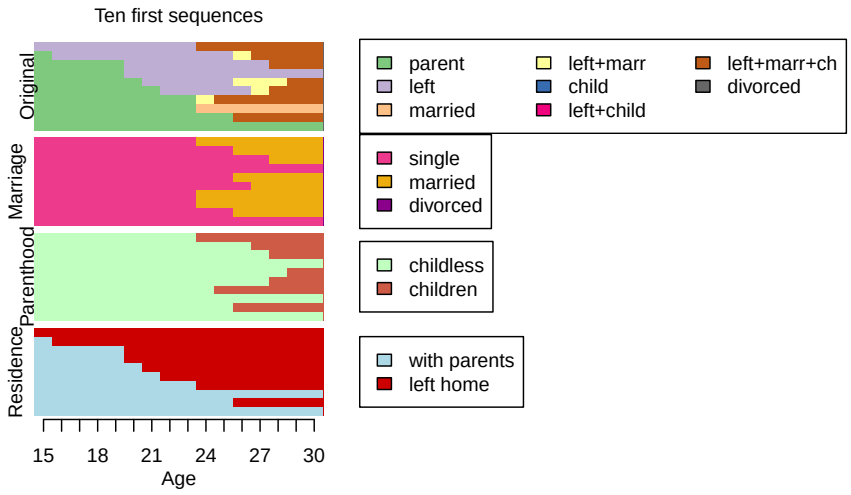

Figure 2: Stacked sequence plot of the first ten individuals in the `biofam` data. The top plot shows the original sequences, and the three bottom plots show the sequences in the separate channels for the same individuals. The sequences are in the same order in each plot, i.e., the same row always matches the same individual.

servations with small emission probabilities can be combined into one category – here we've combined states with emission probabilities less than 2%.

The width of the edge depends on the probability of the transition; the most probable transitions are thus easy to detect. Here the most probable transition between two hidden states is the transition into parenthood (typically married/childless/left home → married/children/left home) with a probability of 0.19.

1.3 Example data

As an example we use the `biofam` data available in the `TraMineR` package (Gabadinho et al., 2011). It is a sample of 2000 individuals born in 1909–1972, constructed from the Swiss Household Panel survey in 2002 (Müller, Studer, and Ritschard, 2007). The data set contains sequences of annual family life statuses from age 15 to 30. Eight observed states are defined from the combination of five basic states: living with parents, left home, married, having children, and divorced. To show a more complex example, we have split the original data into three separate channels representing different life domains: marriage, parenthood, and residence. The data for each individual now includes three parallel sequences constituting of two or three states each: single/married/divorced, childless/parent, and living with parents / having left home. This three-channel version of the data is stored as a new data object called `biofam3c`.

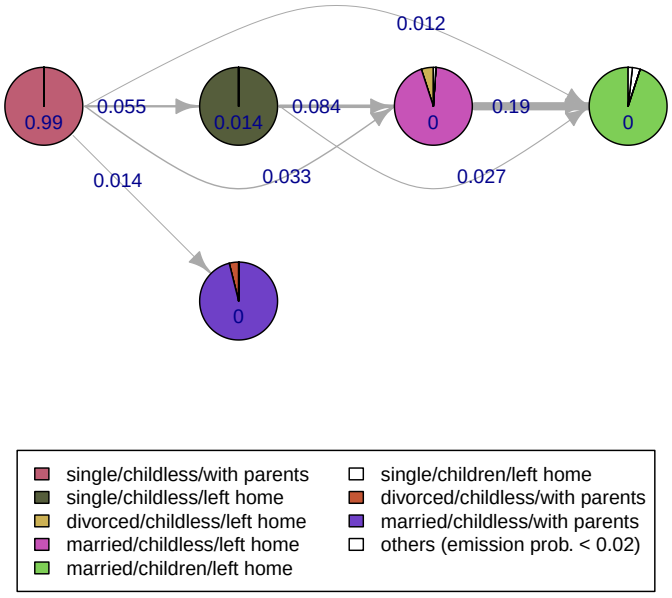

Figure 3: Illustrating a left-to-right hidden Markov model for the multichannel `biofam` data as a directed graph. Pies represent the hidden states, with emission probabilities of combined observations as slices. Arrows illustrate transition probabilities between the hidden states. Probabilities of starting in each state are shown next to the pies.

2 Tools for visualizing multichannel sequence data

2.1 `ssplot`: stacked sequence plots

The `ssplot` function is the simplest way of plotting multichannel sequence data in `seqHMM`. It can be used to illustrate state distributions or sequence index plots. The former is the default option, since index plots can take a lot of time and memory if data are large.

Data types. The `ssplot` function accepts two types of objects, either state sequence objects of type `stslist` from the `seqdef` function or a hidden Markov model object of class `hmm` from the `build_hmm` function.

As an example of the former, we start by preparing three state sequence objects from the `biofam3c` data. Here we set the start of the sequence at age 15 and define the states in each channel with the `alphabet` argument. The colour palette is stored as an attribute and may be modified accordingly.

```
library("seqHMM")

data("biofam3c")
marr_seq <- seqdef(
  biofam3c$married,
  start = 15,
  alphabet = c("single", "married", "divorced")
)
child_seq <- seqdef(
  biofam3c$children,
  start = 15,
  alphabet = c("childless", "children")
)
left_seq <- seqdef(
  biofam3c$left,
  start = 15,
  alphabet = c("with parents", "left home")
)

attr(marr_seq, "cpal") <- c(
  "violetred2", "darkgoldenrod2",
  "darkmagenta"
)
attr(child_seq, "cpal") <- c("darkseagreen1", "coral3")
attr(left_seq, "cpal") <- c("lightblue", "red3")
```

The `ssplot` function is designed for multichannel data but also works for single-channel sequences. For multichannel data, when using `stslist` objects the different channels are given as a list. Figure 4 illustrates a default plot.

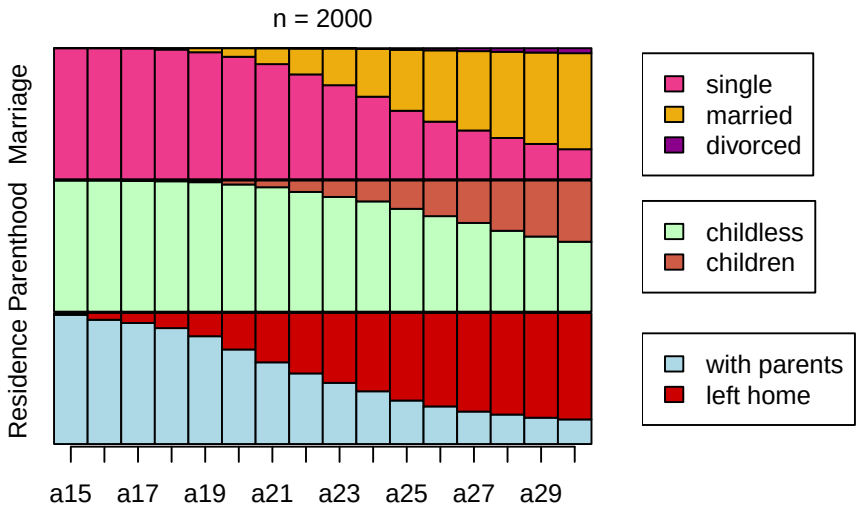

Figure 4: Stacked sequence plot of annual state distributions in the three-channel `biofam` data. This is the default output of the `ssplot` function.

```
ssplot(
  x = list(
    "Marriage" = marr_seq, "Parenthood" = child_seq,
    "Residence" = left_seq
  )
)
```

Observed and hidden state sequences. The `ssplot` function may be asked to plot observed sequences, hidden state sequences, or both. In the case of `hmm` objects, the most probable hidden state sequences are computed from the data **Choosing plots.** The `ssplot` function is able to also hidden state paths in addition to or instead of observed sequences. The type of sequences is set with the `plots` argument. Figure 5 shows state distributions for the observations as well as hidden states.

```
data("hmm_biofam")

ssplot(x = hmm_biofam, plots = "both")
```

Sequence index plots are called with the `type = "I"` argument in a similar way to the `seqplot` function in `TraMineR`. This type of plot shows the sequences as a whole. As the index plot is often difficult to interpret as such, sequences may be ordered in a more meaningful way.

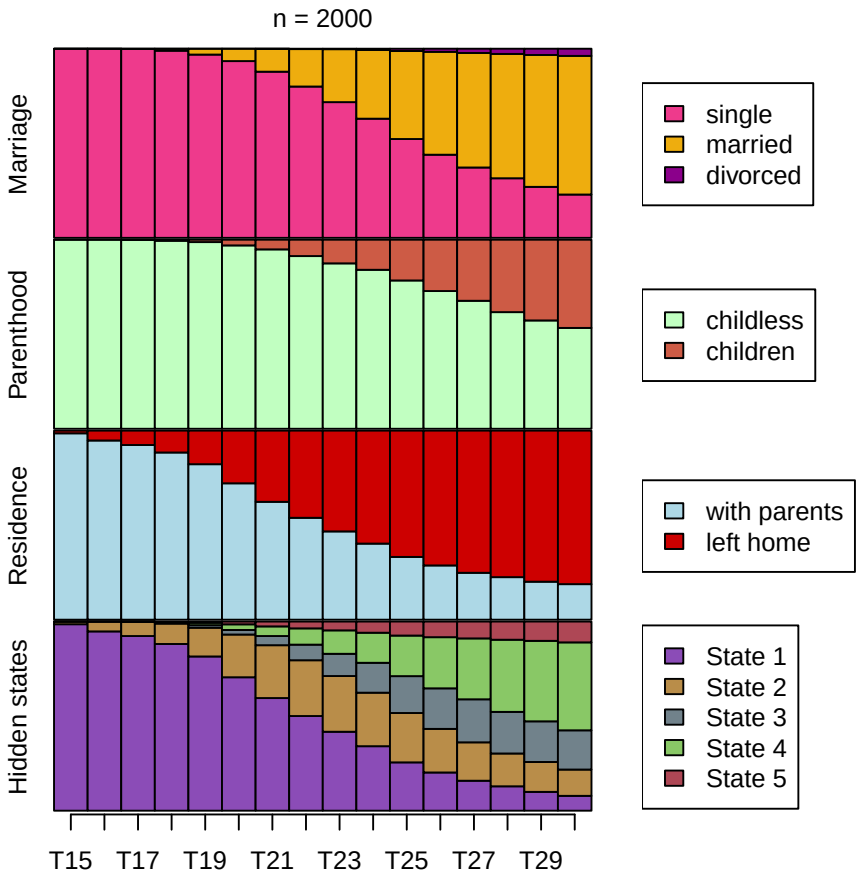

Figure 5: Stacked sequence plot of observations and hidden state paths using a hidden Markov model object.

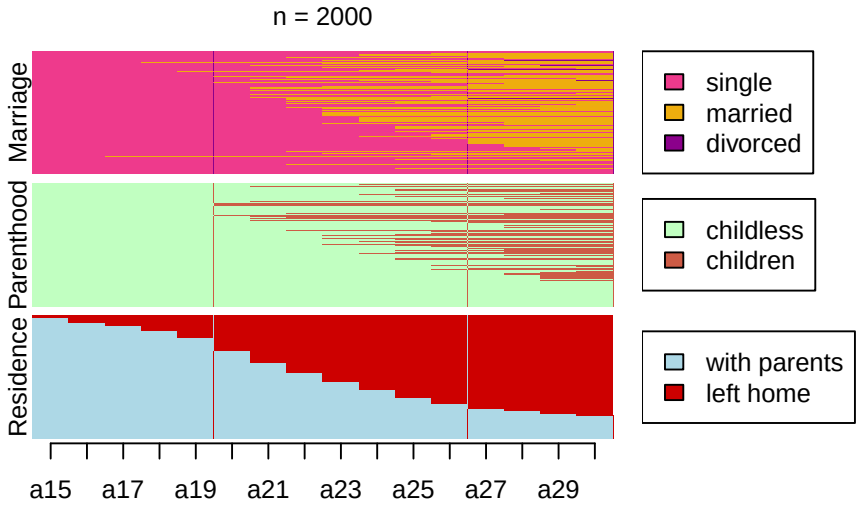

Figure 6: Sequence index plot showing observed sequences sorted by the third channel, residence.

Sorting. The `sortv` argument may be given a sorting variable or a sorting method. A sorting method may be one of `from.start`, `from.end`, `mds.obs`, and `mds.hidden`. The `from.start` and `from.end` arguments sort the sequences by the elements of the alphabet in the channel defined by the `sort.channel` argument (the first channel by default), starting from the start of the end of the sequences. The `mds.obs` and `mds.hidden` arguments sort the sequences according to the scores of multidimensional scaling for the observed data or hidden states paths, respectively. The `tlim` argument may be used to select only certain subjects for the plot. Figure 6 shows sequences sorted by the third channel, residence.

```
ssplot(
  hmm_biofam,
  type = "I", sortv = "from.start", sort.channel = 3
)
```

The labels and positions of the ssplot may be modified in many ways.

Title. By default, the function shows the number of subjects in the plot. Additional title printed in front of this is given with the `title` argument. This may be suppressed by `title.n = FALSE`.

Legend. The `with.legend` argument defines if and where the legend for the states is plotted. The default value `"right"` creates separate legends for each

channel (and hidden states, when relevant) and positiones them on the right-hand side of the plot. Other possible values are `"bottom"`, `"right.combined"`, and `"bottom.combined"`, of which the last two create one combined legend of the states in all channels in the selected position. The legend may be supressed altogether with the value `FALSE`. The `ncol.legend` argument gives the number of columns in (each of) the legend(s) and the `legend.prop` argument sets the proportion of the graphic area that is used for plotting the legend (0.3 by default).

Axes. The user may choose to suppress or plot both the a axis (`TRUE` by default) and the y axis (`FALSE` by default). Both may be given optional labels with the `xlab` and `ylab` arguments and the distance of the labels from the plot may be modified with the `xlab.pos` and `ylab.pos` arguments (note: for hidden states, the label for the y axis is given with the `hidden.states.title` argument). The labels for the ticks of the x axis are modified with the `xtlab` argument.

Text sizes. The sizes of the title, legend, axis labels, and axis tick labels are modified with the `cex.title`, `cex.legend`, `cex.lab`, and `cex.axis` arguments, respectively.

2.2 `ssp`: saving stacked sequence plots

The user may also pre-define function arguments with the `ssp` function and then use this object for plotting with a simple `plot` method. After defining one ssp, modifications are easy to do with the `update` function (see an example in the next section).

```
ssp_def <- ssp(
  hmm_biofam,
  plots = "both", type = "I", sortv = "mds.hidden",
  ylab.pos = c(1, 2),
  title = "Family trajectories", title.n = FALSE,
  xtlab = 15:30, xlab = "Age",
  ncol.legend = c(2, 1, 1), legend.prop = 0.37
)

plot(ssp_def)
```

Figure 7 shows the plot from the saved ssp. Here, the sequences were sorted according to multidimensional scaling scores computed from the most probable hidden state sequences. Labels and their positions as well as the legends were also modified.

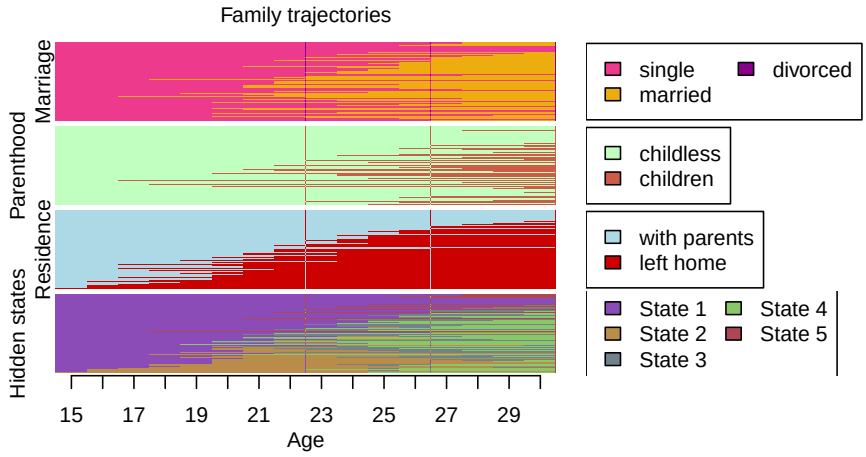

Figure 7: Example on saving `ssp` objects. Sequences are sorted according to multidimensional scaling scores.

2.3 `gridplot`: multiple stacked sequence plots

The `gridplot` function combines several `ssp` figures into one. It is useful for showing different features for the same subjects or the same features for different groups. The dimensions of the grid, widths and heights of the cells, and positions of the legends are easily modified.

```
ssp_f <- ssp(
  list(
    marr_seq[biofam3c$covariates$sex == "woman", ],
    child_seq[biofam3c$covariates$sex == "woman", ],
    left_seq[biofam3c$covariates$sex == "woman", ]
  ),
  type = "I", sortv = "mds.obs", with.legend = FALSE,
  title = "Women", xtlab = 15:30, ylab.pos = c(1, 2, 1),
  ylab = c("Married", "Children", "Residence")
)

ssp_m <- update(
  ssp_f,
  title = "Men",
  x = list(
    marr_seq[biofam3c$covariates$sex == "man", ],
    child_seq[biofam3c$covariates$sex == "man", ],
    left_seq[biofam3c$covariates$sex == "man", ]
  )
```

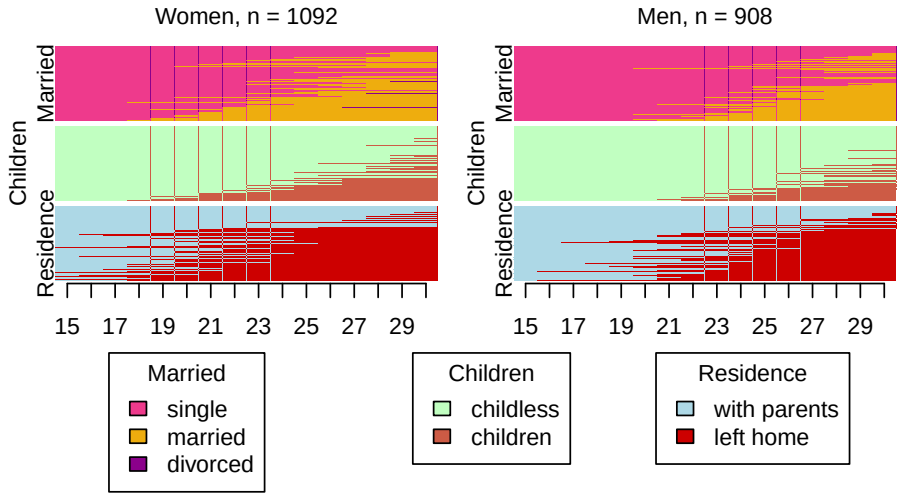

Figure 8: Showing state distribution plots for women and men in the `biofam` data. Two figures were defined with the `ssp` function and then combined into one figure with the `gridplot` function.

```
)

gridplot(
  list(ssp_f, ssp_m),
  ncol = 2, nrow = 2, byrow = TRUE,
  legend.pos = "bottom", legend.pos2 = "top",
  row.prop = c(0.65, 0.35)
)
```

Figure 8 illustrates an example of a gridplot showing sequence index plots for women and men. Sequences were sorted using multidimensional scaling scores for observations. The `legend.pos` function defines the position of the legend(s) relative to the whole plot. It may be one of `"bottom"` or `"right"` or a numerical vector of grid cells.

```
ssp_f2 <- update(ssp_f, type = "d", title = FALSE)

ssp_m2 <- update(ssp_m, type = "d", title = FALSE)

gridplot(
  list(ssp_f, ssp_f2, ssp_m, ssp_m2),
```

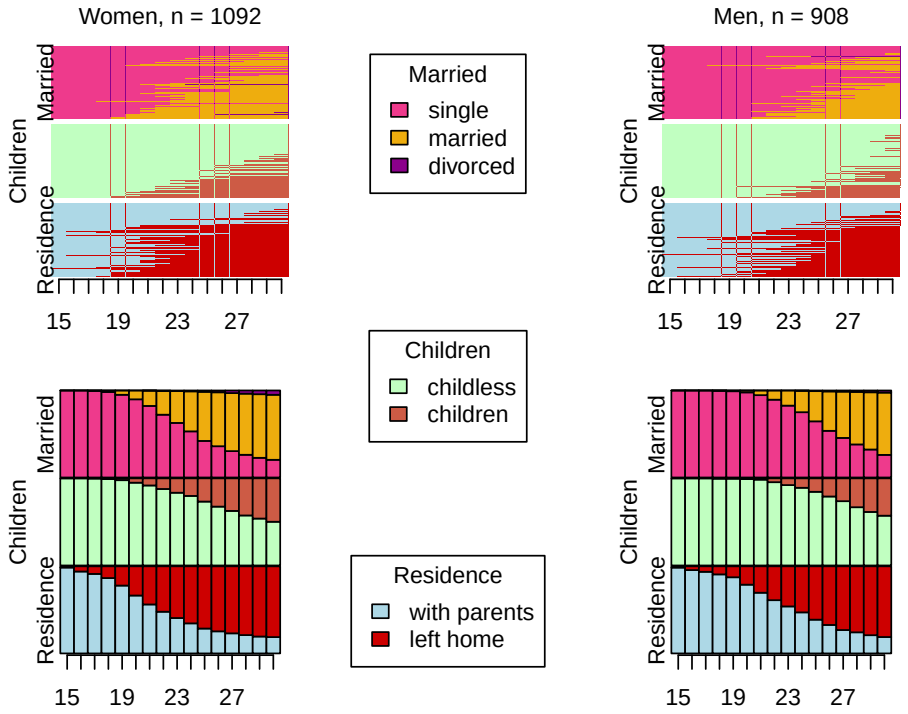

Figure 9: Another example of `gridplot`. Showing sequences and state distributions for women and men.

```
  ncol = 3, nrow = 2,
  legend.pos = 3:4
)
```

Figure 9 shows an example of manual positioning of legends into grid cells 3 and 4.

The `gridplot` function uses the first list object for defining the legends. If the legends are different for different ssp figures, the common legend may be suppressed with the `with.legend = FALSE` argument.

```
ssp_f3 <- update(
  ssp_f,
  with.legend = TRUE, legend.prop = 0.4, ylab.pos = 1,
  cex.lab = 0.9, cex.axis = 0.8, cex.legend = 0.9
)

ssp_f4 <- ssp(
  biofam_seq[biofam3c$covariates$sex == "woman", ],
```

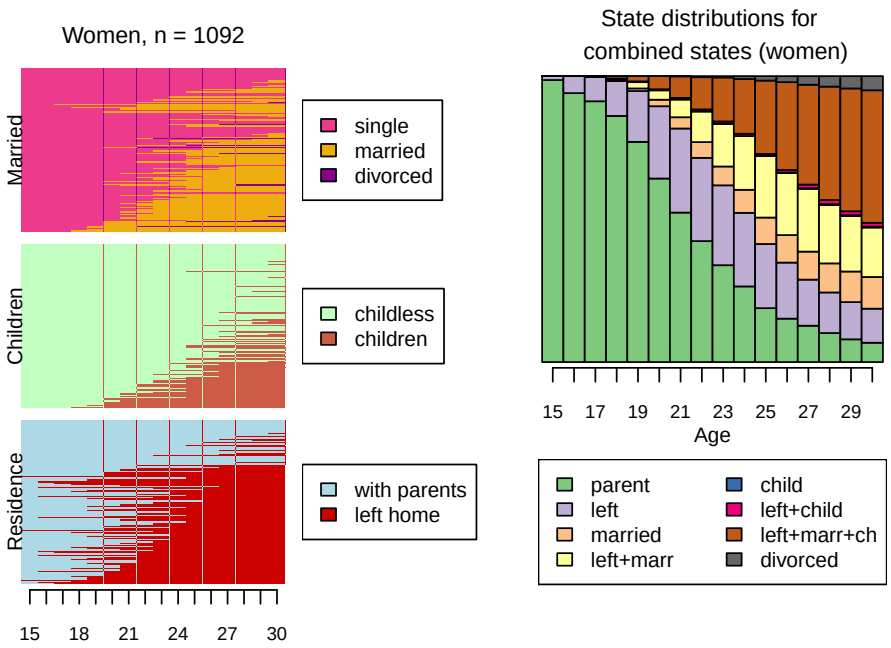

Figure 10: Another example of `gridplot`. Showing three-channel sequences and state distributions of combined states for women.

```
  type = "d", title.n = FALSE, xtlab = 15:30,
  title = "State distributions for \n combined states (women)",
  title.pos = 1.5, ylab = "", xlab = "Age",
  with.legend = "bottom", ncol.legend = 2,
  cex.lab = 0.9, cex.axis = 0.8, cex.legend = 0.9
)

gridplot(list(ssp_f3, ssp_f4),
  with.legend = FALSE, ncol = 2,
  col.prop = c(0.55, 0.45)
)
```

2.4 `mssplot`: ssp for mixture HMMs

The `mssplot` draws stacked sequence plots of observed sequences and/or most probable hidden state paths for clusters based on a mixture hidden Markov model of class `mhmm`. The most probable cluster (submodel) for each subject is chosen according to the most probable path of hidden states. By default, the function plots all clusters but the user may choose a subset of clusters with the

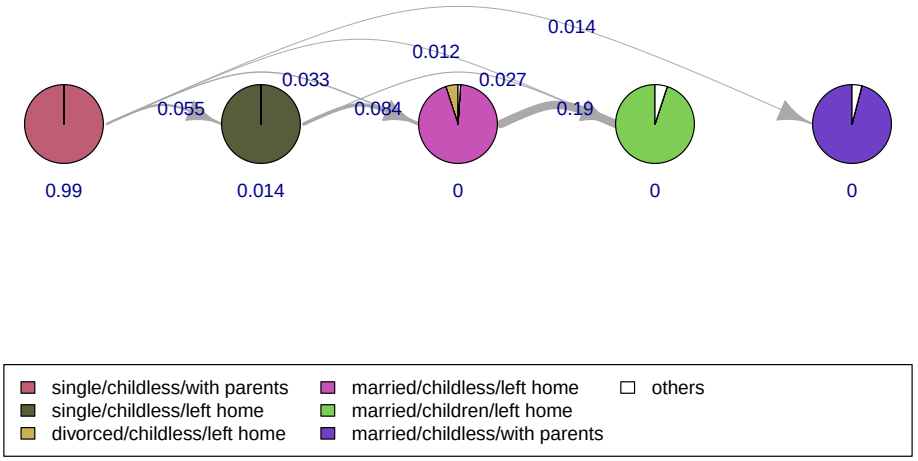

Figure 11: A default plot of a hidden Markov model.

`which.plots` argument. This is an interactive plot that shows several plots in a row. The default is to change the plot by pressing Enter. With the `ask = TRUE` argument the user gets a menu from which to choose the next cluster for plotting.

3 Tools for hidden Markov models

3.1 `plot.hmm`: plotting hidden Markov models

A basic HMM graph is easily called with the `plot` method. Figure 11 illustrates the default plot.

```
plot(hmm_biofam)
```

A simple default plot is a convenient way of visualizing the models during the analysis process, but for publishing it is often better to modify the plot to get an output that best illustrates the structure of the model at hand.

Layout. The default layout positions hidden states horizontally. This may be changed with the `layout` argument. The user may choose to position hidden states vertically or use a layout function from the `igraph` package. It is also possible to position the vertices manually by giving a two-column numerical matrix of x and y coordinates.

Vertices. By default, the vertices are drawn as pie charts showing the emission probabilities as slices. The pies may be omitted with the `pie = FALSE` argument. Initial probabilities are shown as vertex labels by default. Instead of these, the user may choose to print the names of the hidden states, use own labels, or omit the labels altogether with the `vertex.label` argument. The distance and the positions of the labels are modified with the `vertex.label.dist` and `vertex.label.pos` arguments. The size of the vertices is modified with the `vertex.size` argument.

The colour palette for (combinations of) observed states is set with the `cpal` argument. Observations with small emission probabilities may be combined into one state with the `combine.slices` argument which sets the threshold for printing observed states. By defafult, observed states with emission probabilities less than 0.05 are combined. The colour and label for the combined state is modified with the `combined.slice.color` and `combined.slice.label` arguments, respectively.

Edges. By default, the plotting method draws transition probabilities between different states and omits transitions to same states. Self-loops may be drawn with the `loops` argument. The `edge.curved` argument tells the curvature of edges. These may be different for different edges; setting curvature to 0 or FALSE draws straight edges. The widths of the edges are modified with the `edge.width` and `cex.edge.width` arguments. The former sets the widths (by default, proportional to transition probabilities) and the latter sets an expansion factor to all edges. The size of the arrows is modified with the `edge.arrow.size` argument. The `trim` argument omits edges with transition probabilities less than the specified value (0 by default, i.e., no trimming).

Legend. The `with.legend` argument defines if and where the legend is plotted. The `ltext` argument may be used to modify the labels shown in the legend. Similarly to ssp figures, the legend may be modified with the `legend.prop`, `ncol.legend`, and `cex.legend` arguments.

Texts. Font families for labels are modified with the `vertex.label.family` and `edge.label.family` arguments. Printing of model parameters may be modified with the `label.signif`, `label.scientific`, and `label.max.length` arguments. The first rounds labels to the specified number of significant digits, the second one defines if scientific notation should be used to describe small numbers, and the last argument sets the maximum number of digits for labels. These three arguments are omitted for user-given labels.

Other modifications. The plotting method also accepts other arguments in the `plot.igraph` function in the `igraph` package.

Figure 3 in Section 1.2 is an example of manual positioning of vertices. Here we have modified edge curvature and width, legend properties, and combined slices. The code for creating the figure is shown here.

```
plot(hmm_biofam,
  layout = matrix(c(
    1, 2, 3, 4, 2,
    1, 1, 1, 1, 0
  ), ncol = 2),
  xlim = c(0.5, 4.5), ylim = c(-0.5, 1.5), rescale = FALSE,
  edge.curved = c(0, -0.8, 0.6, 0, 0, -0.8, 0),
  cex.edge.width = 0.8, edge.arrow.size = 1,
  legend.prop = 0.3, ncol.legend = 2,
  vertex.label.dist = 1.1, combine.slices = 0.02,
  combined.slice.label = "others (emission prob. < 0.02)"
)
```

Figure 12 shows an example of using a layout function from the `igraph` package. This is based on the same model as the other figures but here we have omitted pie graphs and instead show the names of the hidden states printed within the circles. This figure also shows the self-loops for transitions. We have omitted transition probabilities smaller than 0.01 and printed labels with three significant digits.

```
require("igraph")
set.seed(1234)
plot(hmm_biofam,
  layout = layout_nicely, pie = FALSE,
  vertex.size = 30, vertex.label = "names", vertex.label.dist = 0,
  edge.curved = FALSE, edge.width = 1,
  loops = TRUE, edge.loop.angle = -pi / 8,
  trim = 0.01, label.signif = 3,
  xlim = c(-1, 1.3)
)
```

3.2 `plot.mhmm`: plotting mixture hidden Markov models

The `plot.mhmm` function shows the submodels of a `mhmm` object similarly to the `mssplot` function. Also here the user may choose which submodels to plot with the `which.plots` argument and ask for a menu from which to choose the next plot with the `ask` argument. Instead of an interactive mode it is also possible to plot all submodels in the same figure similarly to the `gridplot` function. Note, however, that such a plot requires opening a large window.

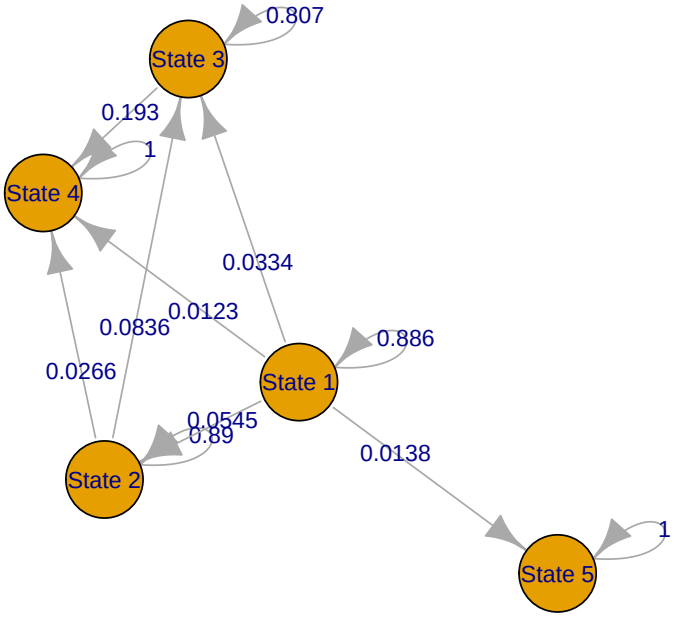

Figure 12: Another example of `plot.hmm`.

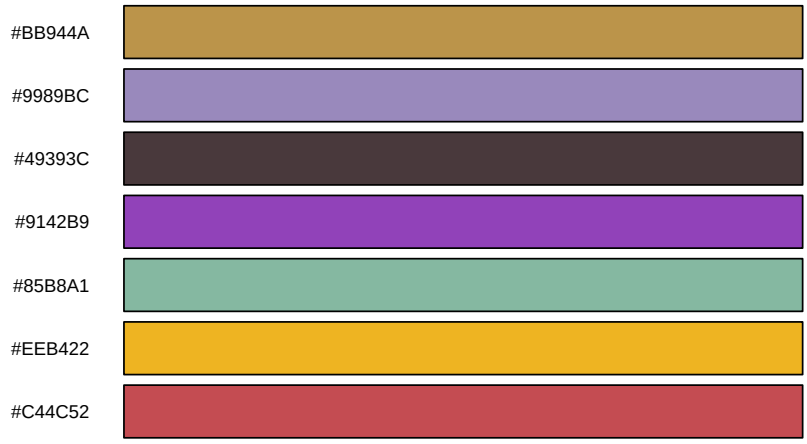

Figure 13: Helper function for plotting colour palettes with their names.

4 Helper tools

4.1 `mc_to_sc_data` and `mc_to_sc`: from multichannel to single-channel

We also provide a function `mc_to_sc_data` for the easy conversion of multichannel sequence data into a single channel representation. Plotting combined data is often useful in addition to (or instead of) showing separate channels. A similar function called `mc_to_sc` converts multichannel HMMs into single-channel representations.

4.2 `colorpalette`: ready-made colour palettes

The `colorpalette` data is a list of ready-made colour palettes with distinct colours. By default, the `seqHMM` package uses these palettes when determining colours for new state sequence objects or for plotting. A colour palette with n colours is called with `colorpalette[[n]]`.

4.3 `plot_colors`: show colours of colour palettes

The `plot_colors` function plots colors and their labels for easy visualization of a colorpalette. Figure 13 shows an example from `colorpalette` with seven colours.

```
plot_colors(colorpalette[[7]])
```

4.4 `separate_mhmm`: reorganize MHMM into a list of HMMs

The `separate_mhmm` function reorganizes the parameters of an `mhmm` object into a list where each list component is an object of class `hmm` consisting of parameters of the corresponding cluster. This gives more possibilities for plotting. For example, the user may define ssp figures for each cluster defined by an MHMM for plotting with the `gridplot` function.

References

Francesco C. Billari. The analysis of early life courses: Complex descriptions of the transition to adulthood. *Journal of Population Research*, 18(2):119–142, 2001.

Francesco C Billari and Raffaella Piccarreta. Analyzing demographic life courses through sequence analysis. *Mathematical Population Studies*, 12(2):81–106, 2005. doi: 10.1080/08898480590932287.

Anette Eva Fasang and Tim Futing Liao. Visualizing sequences in the social sciences: Relative frequency sequence plots. *Sociological Methods & Research*, 43(4):643–676, 2014. doi: 10.1177/0049124113506563.

Alexis Gabadinho, Gilbert Ritschard, Nicolas S. Müller, and Matthias Studer. Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software*, 40(4):1–37, 2011. doi: 10.18637/jss.v040.i04. URL `http://www.jstatsoft.org/v40/i04/paper`.

Satu Helske and Jouni Helske. Mixture hidden Markov models for sequence data: The seqHMM package in R. *Journal of Statistical Software*, 88(3):1–32, 2019. doi: 10.18637/jss.v088.i03.

Nicolas Séverin Müller, Matthias Studer, and Gilbert Ritschard. Classification de parcours de vie à l'aide de l'optimal matching. *XIVe Rencontre de la Société francophone de classification (SFC 2007)*, pages 157–160, 2007.

R. Piccarreta and O. Lior. Exploring sequences: a graphical tool based on multi-dimensional scaling. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(1):165–184, 2010. doi: 10.1111/j.1467-985X.2009.00606.x.

Raffaella Piccarreta. Graphical and smoothing techniques for sequence analysis. *Sociological Methods & Research*, 41(2):362–380, 2012. doi: 10.1177/0049124112452394.

Stefani Scherer. Early career patterns: A comparison of Great Britain and West Germany. *European Sociological Review*, 17(2):119–144, 2001. doi: 10.1093/esr/17.2.119.

Eric D Widmer and Gilbert Ritschard. The de-standardization of the life course: Are men and women equal? *Advances in Life Course Research*, 14(1):28–39, 2009. doi: 10.1016/j.alcr.2009.04.001.