Simulating Multi-state models with icmstate

This vignette documents the use of the `sim_weibmsm()` function for generating trajectories from an (interval-censored) Markov Multi-state model. We also show how the data set available in the `icmstate` package was generated. Currently, only Weibull hazards/intensities are supported for transitions between states.

We use the standard `R` parametrisation of the Weibull distribution with Survival function:

$$S(t) = e^{-\left(\frac{t}{\text{scale}}\right)^{\text{shape}}},$$

hazard function:

$$h(t) = \frac{\text{shape}}{\text{scale}} \left(\frac{t}{\text{scale}}\right)^{\text{shape}-1},$$

and cumulative hazard function:

$$H(t) = \left(\frac{t}{\text{scale}}\right)^{\text{shape}}$$

The mean failure time of the Weibull distribution is given by:

$$\mathbb{E}[T] = \text{scale} \cdot \Gamma(1 + \frac{1}{\text{shape}})$$

For more information, see `help(rweibull)`.

```
library(icmstate)
set.seed(1)
```

1 Simulating data

We cover the required arguments of the `sim_weibmsm()` function separately

1.1 The transition matrix (tmat)

To simulate from a multi-state model, a transition matrix must first be specified. For this, we make use of the `transMat()` function in the `mstate` package (Wreede et al. [2011]). Suppose we want to generate from an illness-death model (see Figure 1), then the transition matrix indicates we can reach states 2 and 3 from state 1 and state 3 from state 2. State 3 is absorbing and cannot be left.

```
library(mstate)
tmat_ID <- transMat(list(c(2, 3), c(3), c()), names = c("Alive", "Illness",
                                                        "Death"))
```

After using the `transMat()` function, each transition gets assigned a transition number. This number can be displayed by printing the transition matrix.

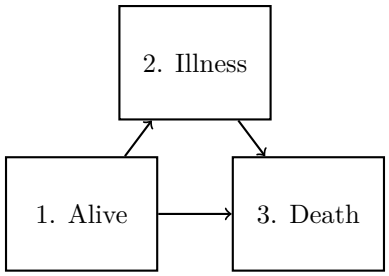


Figure 1: Graphical representation of the illness-death model.

```
tmat_ID
#>          to
#> from      Alive Illness Death
#>   Alive      NA       1     2
#>   Illness    NA      NA     3
#>   Death      NA      NA    NA
```

1.2 Choosing Weibull transition parameters (shape, scale)

The first transition is therefore from 1. Alive → 2. Illness, the second transition from 1. Alive → 3. Death and the third transition from 2. Illness → 3. Death. For each of the transitions, we need to choose a Weibull shape and scale parameter to generate transition times.

For this example, let us assume that our simulated subjects have a constant risk of becoming ill over the duration of the study. We quantify this assumption by choosing an exponential (constant) hazard rate for our first transition. This is achieved by choosing the shape equal to 1 for the first transition.

For the second transition, we assume that the rate of death from an alive state increases as you get older, which corresponds to choosing the shape for this transition larger than 1.

Finally, we assume that subjects are at greater risk of dying just after contracting the disease, and that the rate of death decreases the longer you are ill. This amounts to choosing the shape parameter smaller than 1 for the second transition.

To adhere to these choices we specify a vector of shape parameters for our transitions, so that the first entry corresponds to the Weibull shape parameter of the

```
#The first entry corresponds to shape of first transition, and so on...
shape_ID <- c(1, 1.5, 0.5)
```

Similarly, we choose scale parameters such that on average subjects become ill after 6 years, die without illness after 10 years and die after illness after 2 years.

```
scale_ID <- c(6, 10/gamma(1+1/1.5), 10/gamma(1+1/0.5))
```

1.3 Choosing/generating observation times (data, n_subj, obs_pars)

Now that we have specified the multi-state model and the transition intensities, we need to either manually determine observation times for subjects (by specifying the `data` argument) or automatically generate them (by specifying the `n_subj` and `obs_pars` arguments) using the `sim_weibmsm()` function.

1.3.1 Manually choosing observation times (data)

Manually choosing the observation times provides the user with the most freedom. To make use of this, we must supply the `data` argument to the `sim_weibmsm()` function. This must be a `data.frame` or `matrix` with named columns `time` and `id`, representing the observations times for the corresponding subjects.

Suppose we want to observe half of our $n = 100$ subjects at even times and half at uneven times. We then generate a data.frame as follows:

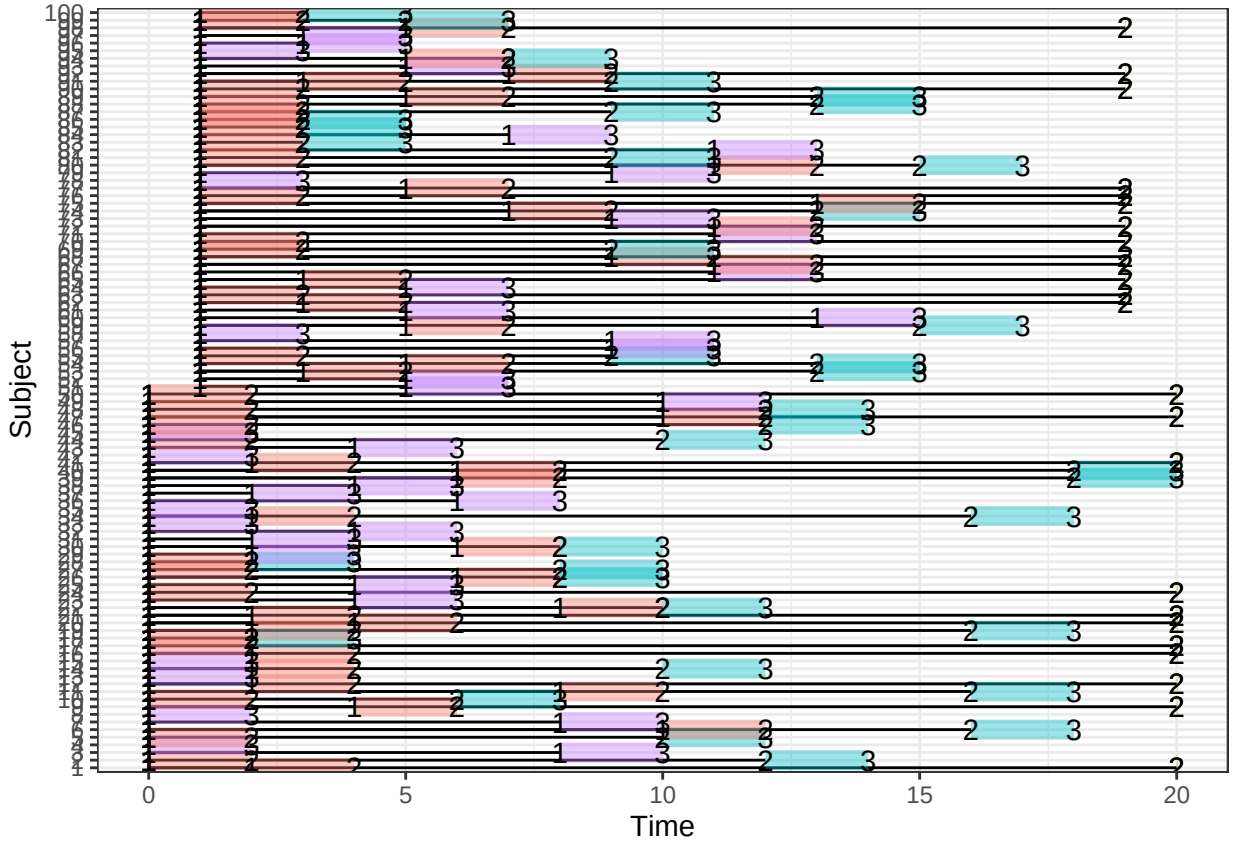
```
obs_data <- data.frame(time = c(rep(seq(0, 20, 2), 50), rep(seq(1, 19, 2), 50)),
                       id = c(rep(1:50, each = 11), rep(51:100, each = 10)))
```

Now we can generate the interval-censored data:

```
data_manual <- sim_weibmsm(data = obs_data, tmat = tmat_ID,
                           shape = shape_ID, scale = scale_ID)
```

We visualise the generated data using `visualise_msm()`:

```
visualise_msm(data_manual)
```



1.3.2 Automatically generating an observation grid (n_subj, obs_pars)

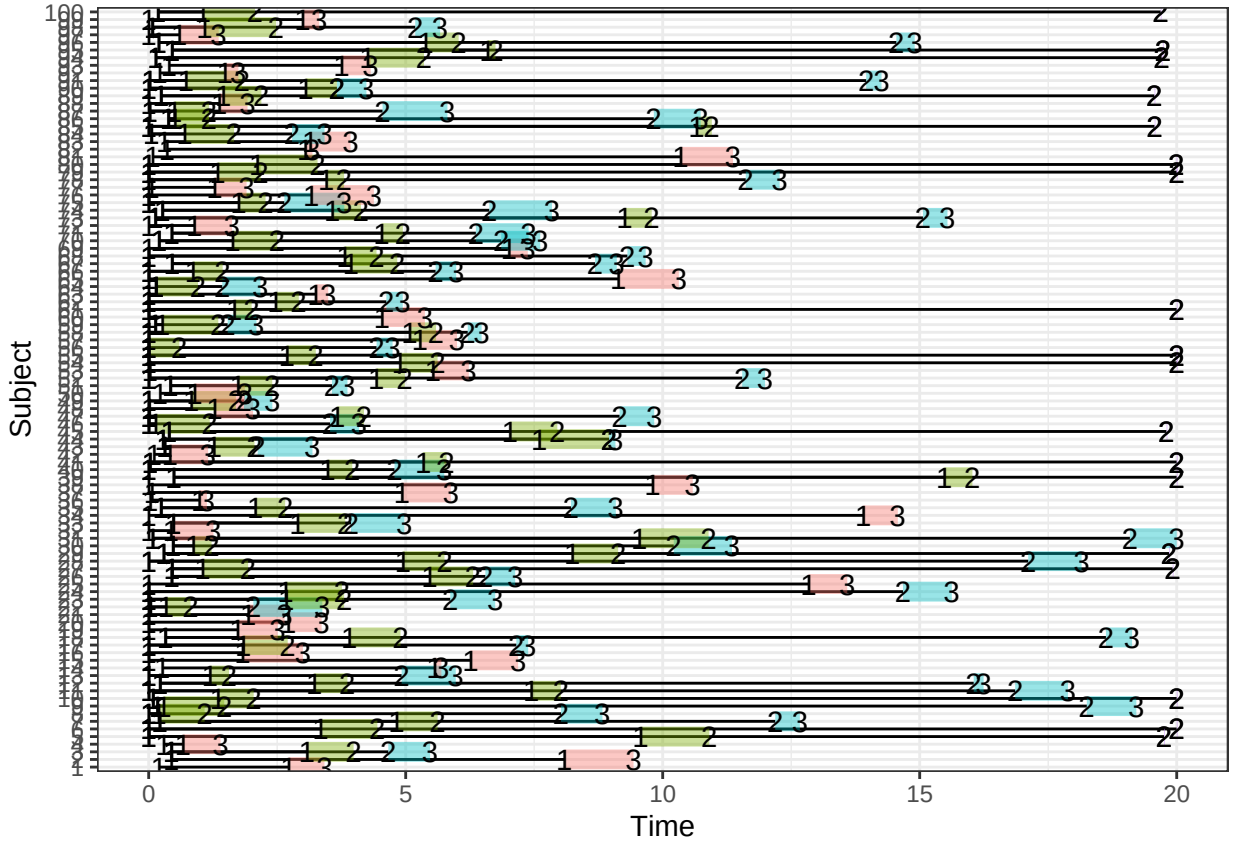
The `sim_weibmsm()` function also allows to generate observation times automatically. In that case, the `n_subj` and `obs_pars` parameters must be specified. This allows to generate `n_subj` subjects on a regular grid (planned assessment times) with uniform deviations from the planned assessment times.

For example, let's say we want to observe 100 subjects every 2 days for a total duration of 20 days. Unfortunately, subjects do not come to assessments at the exactly specified times, but deviate uniformly by half a day in either direction. We can then choose `n_subj = 100` and `obs_pars = c(2, 0.5, 20)` to achieve this, where the first entry quantities the planned inter-assessment time, the second the uniform deviation and the third the maximal observation time.

```
data_auto <- sim_weibmsm(tmat = tmat_ID, shape = shape_ID, scale = scale_ID,
                         n_subj = 100, obs_pars = c(2, 0.5, 20))
```

We visualise the data:

```
visualise_msm(data_auto)
```



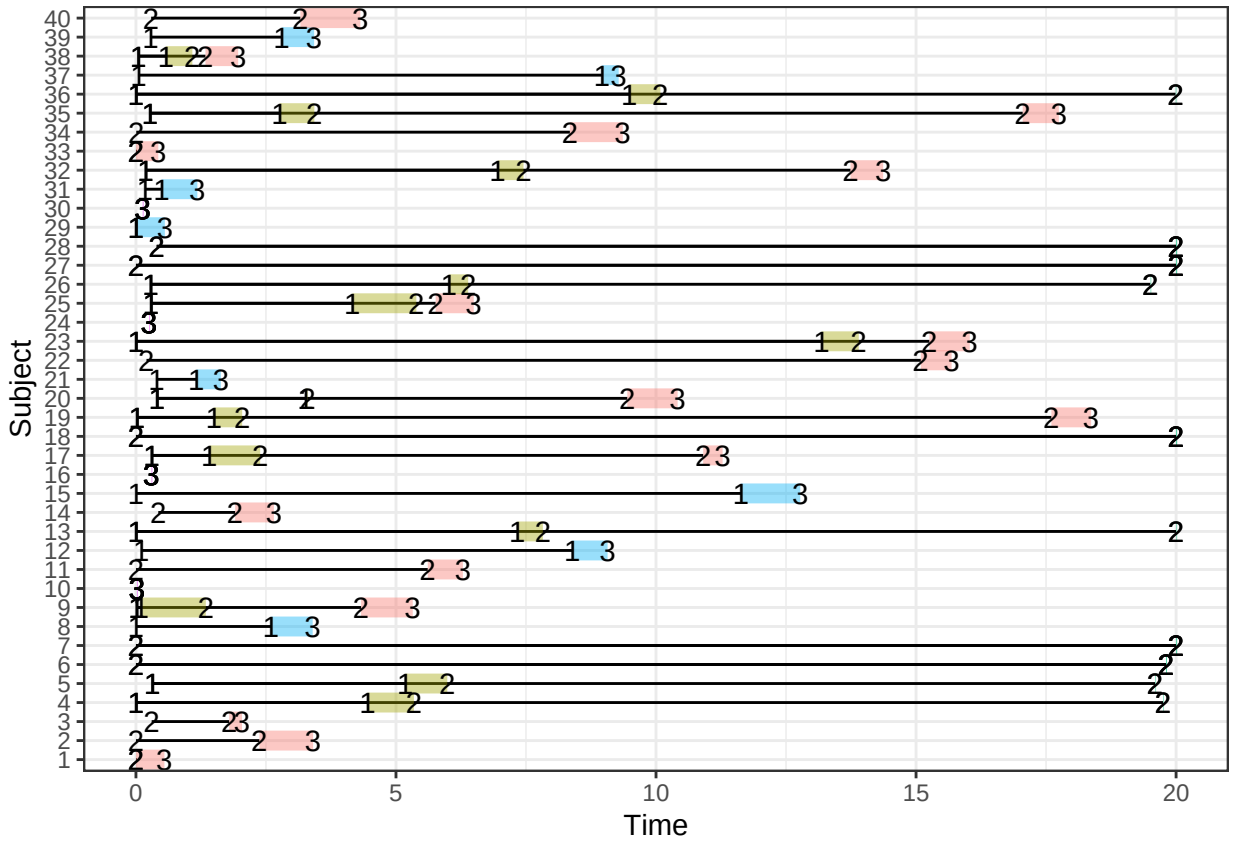
1.4 Additional possibilities (startprobs, exact, censoring)

There are some additional possibilities in the `sim_weibmsm()` function.

1.4.1 Starting state probabilities (startprobs)

In above examples, we generated data with all subjects starting in the first (initial) state. This might not always be appropriate. We can therefore specify the argument `startprobs` as a vector with length the amount of total states in the model, with each entry representing the probability of starting in that state.

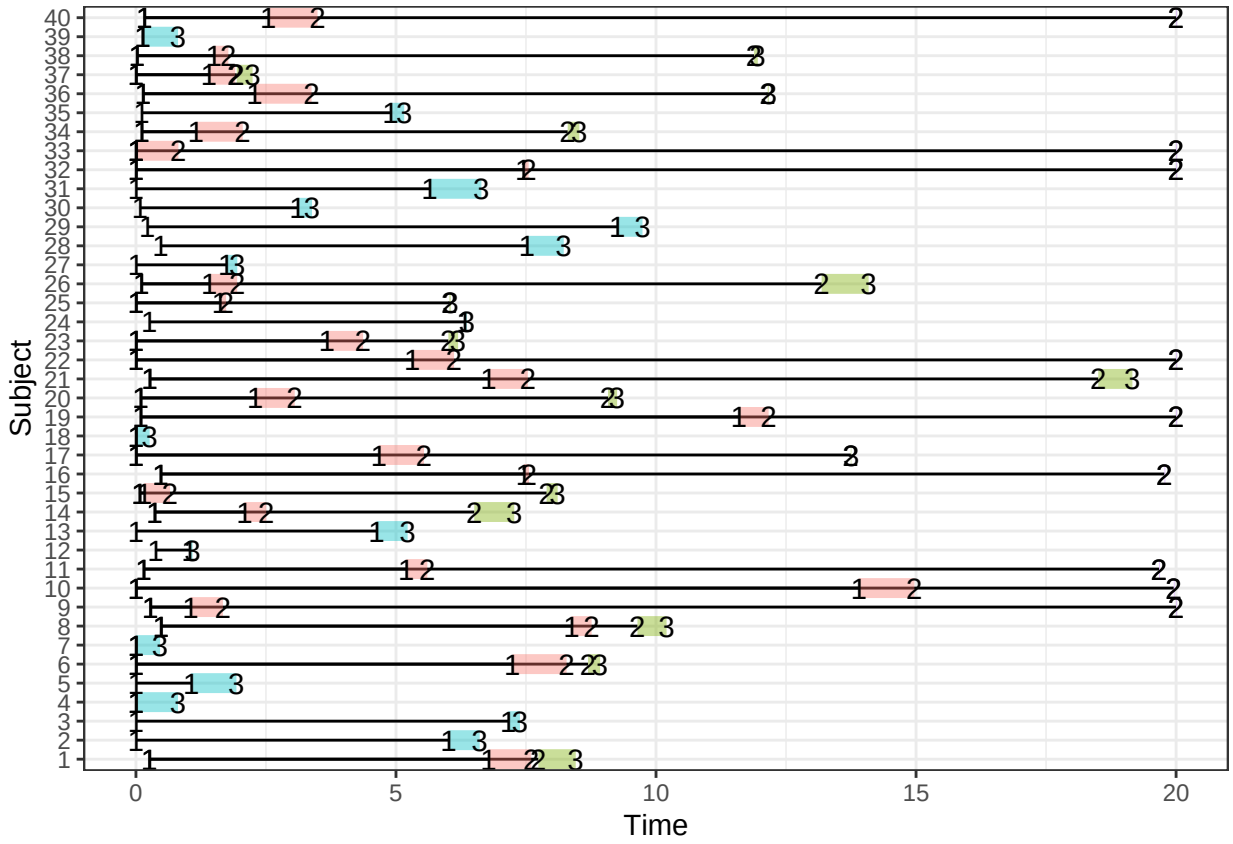
```
data_sprobs <- sim_weibmsm(tmat = tmat_ID, shape = shape_ID, scale = scale_ID,
                           n_subj = 40, obs_pars = c(2, 0.5, 20),
                           startprobs = c(0.6, 0.3, 0.1))
visualise_msm(data_sprobs)
```



1.4.2 Exactly observed states (exact)

In some multi-state models, we will have states that always have transitions into them observed exactly. Usually, these are absorbing (or death) states. One can specify the `exact` argument as a vector containing the column numbers in `tmat` representing the exactly observed states. In this case, the subjects transitioning into such a state will be additionally observed at the exact time of the corresponding transitions. In the illness-death model, death is usually observed exactly, we can generate

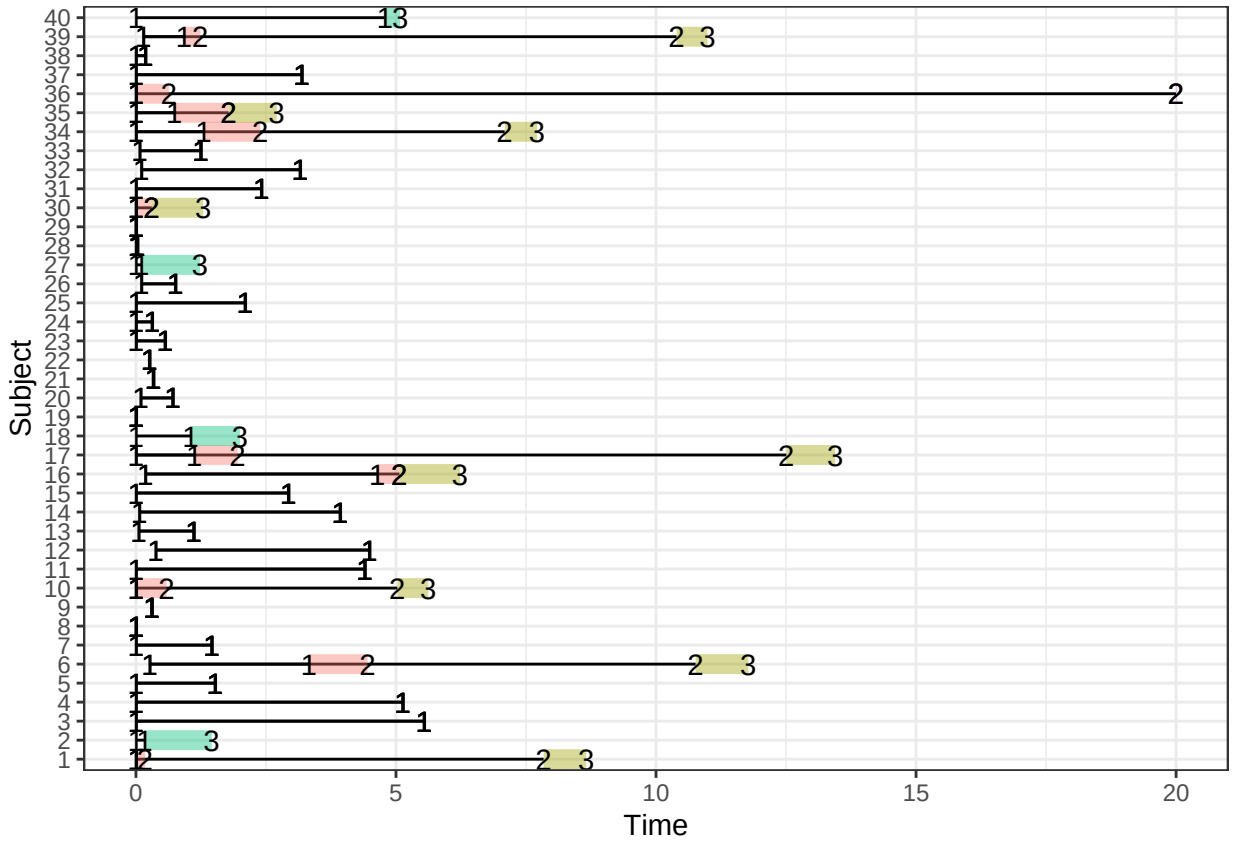
```
data_exact <- sim_weibmsm(tmat = tmat_ID, shape = shape_ID, scale = scale_ID,
                          n_subj = 40, obs_pars = c(2, 0.5, 20),
                          exact = c(3))
visualise_msm(data_exact)
```



1.4.3 Weibull censoring

It might be desirable to allow for right-censoring in certain states. In that case, it is possible to specify the `censshape` and `censscale` parameters indicating the Weibull shapes and scales of the censoring hazard in each of the states. If censoring is undesires in any of the states, leave either the shape or scale as `NA`.

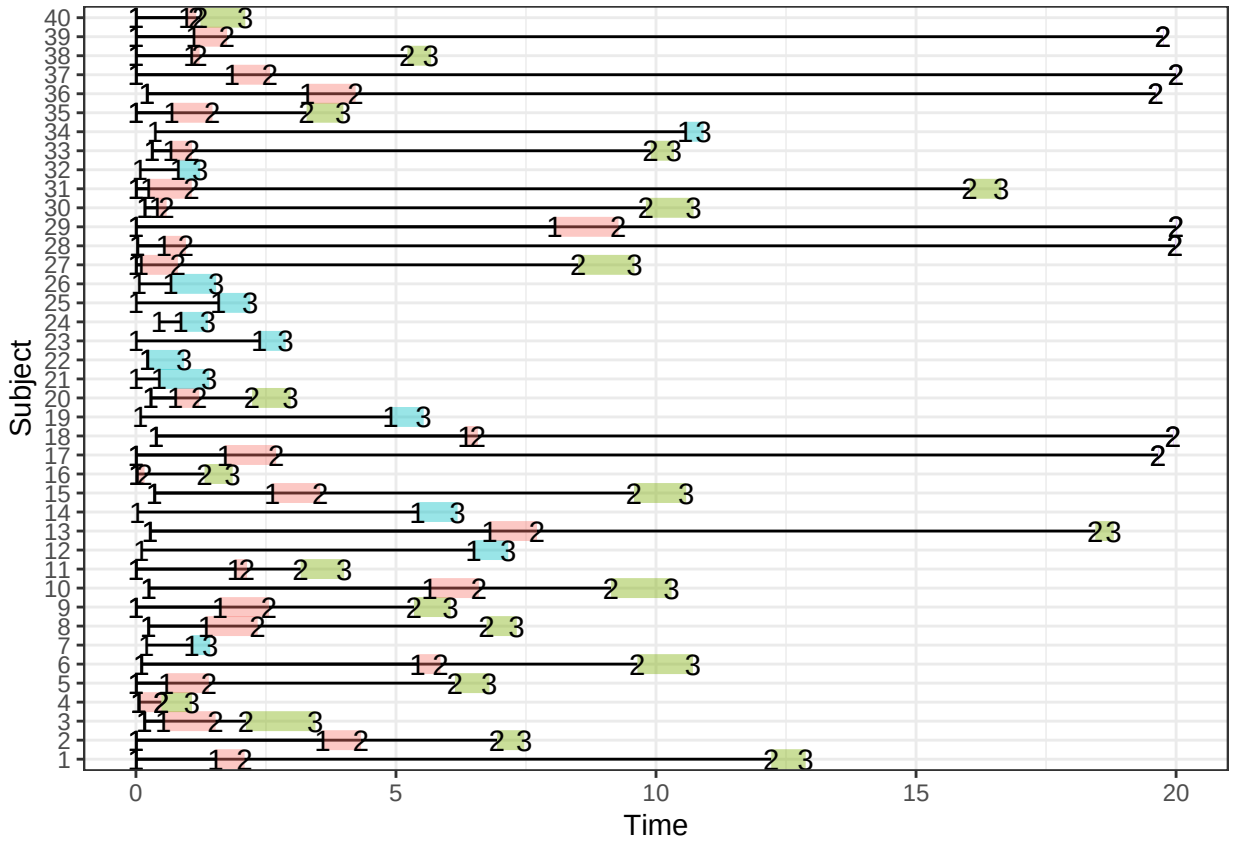
```
data_cens <- sim_weibmsm(tmat = tmat_ID, shape = shape_ID, scale = scale_ID,
                         n_subj = 40, obs_pars = c(2, 0.5, 20),
                         censshape = c(1, NA, NA), censscale = c(3, NA, NA))
visualise_msm(data_cens)
```



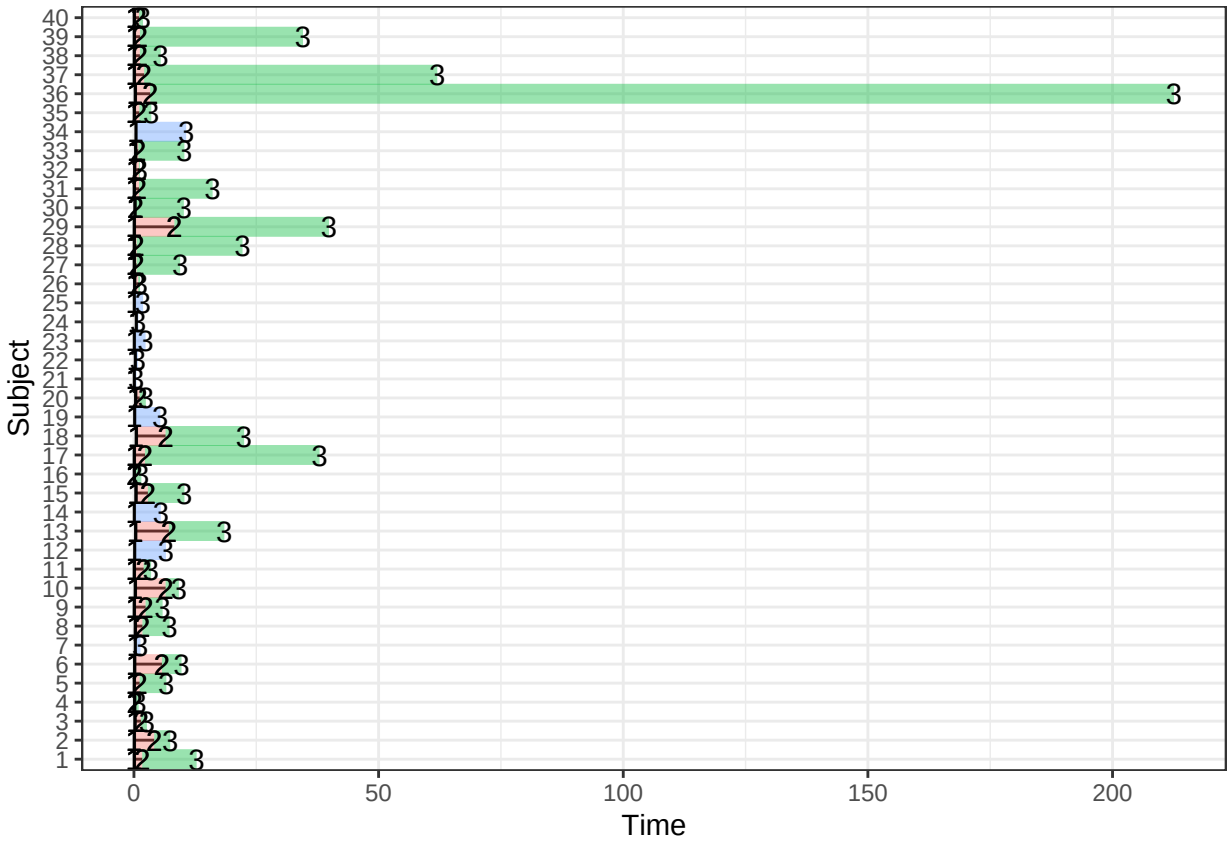
1.4.4 True trajectories

When generating interval-censored data, it is often interesting to know the true underlying trajectory the subjects have taken. This can be achieved by setting the `true_trajec` argument to `TRUE` in the function. We then recover a list with both the observed as well as the true trajectory.

```
data_true <- sim_weibmsm(tmat = tmat_ID, shape = shape_ID, scale = scale_ID,
                         n_subj = 40, obs_pars = c(2, 0.5, 20),
                         true_trajec = TRUE)
visualise_msm(data_true$observed)
```




```
visualise_msm(data_true$true)
```



2 Compare simulated data with true intensities

In this section, we check whether the underlying hazards can be recovered from simulated data.

2.1 True trajectories

We simulate from an illness-death model with everyone starting in state 1 and recover the true trajectories.

```
data_ID <- sim_weibmsm(tmat = tmat_ID, shape = shape_ID, scale = scale_ID,
                       n_subj = 2000, obs_pars = c(2, 0.5, 20),
                       true_trajec = TRUE)
```

We would like to fit a simple survival model on the true trajectory. For this, we first need to transform the data

```
library(survival)
tmat2_ID <- to.trans2(tmat_ID)
dat_true <- data_ID$true
```

We check whether the resulting curves from the fit correspond to the true values.

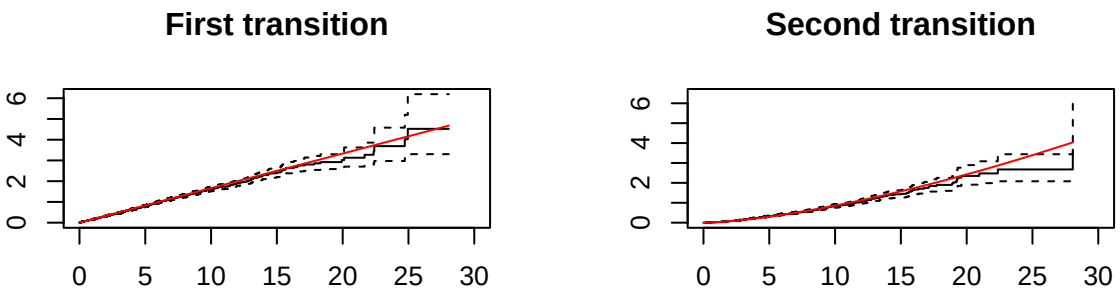
```
opar <- par(no.readonly = TRUE)
par(mfrow = c(2,2))
```

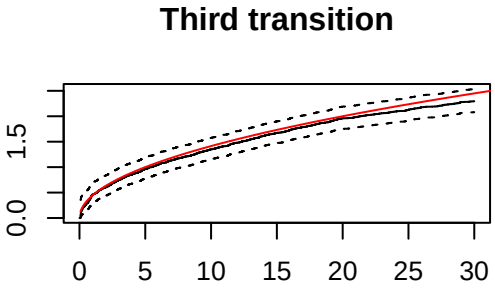
```r
dat_surv <- reshape(dat_true, direction = "wide", idvar = "id", timevar = "state")
dat_surv$status <- rep(1, nrow(dat_surv))

#From 1 to 2
dat1 <- dat_surv
dat1[is.na(dat1[, "time.2"]), "status"] <- 0
dat1[is.na(dat1[, "time.2"]), "time.2"] <- dat1[is.na(dat1[, "time.2"]), "time.3"]
dat1 <- dat1[!dat1[, "time.2"] == 0,]
first_trans <- survfit(Surv(time.1, time.2, status) ~ 1, data = dat1)
plot(first_trans, fun = "cumhaz", xlim = c(0, 30), main = "First transition")
lines(first_trans$time, -pweibull(first_trans$time, shape_ID[1], scale_ID[1], lower = FALSE, log = TRUE)

#From 1 to 3
dat2 <- dat_surv
dat2[!is.na(dat2[, "time.2"]), "status"] <- 0
dat2[!is.na(dat2[, "time.2"]), "time.3"] <- dat2[!is.na(dat2[, "time.2"]), "time.2"]
second_trans <- survival::survfit(Surv(time.3, status) ~ 1, data = dat2)
plot(second_trans, fun = "cumhaz", xlim = c(0, 30), main = "Second transition")
lines(second_trans$time, -pweibull(first_trans$time, shape_ID[2], scale_ID[2], lower = FALSE, log = TRU

third_trans <- survfit(Surv(time.2, time.3, status) ~ 1, data = subset(dat_surv, !is.na(time.2)))
plot(third_trans, fun = "cumhaz", xlim = c(0, 30), main = "Third transition")
lines(third_trans$time, -pweibull(third_trans$time, shape_ID[3], scale_ID[3], lower = FALSE, log = TRUE)
par(opar)
```



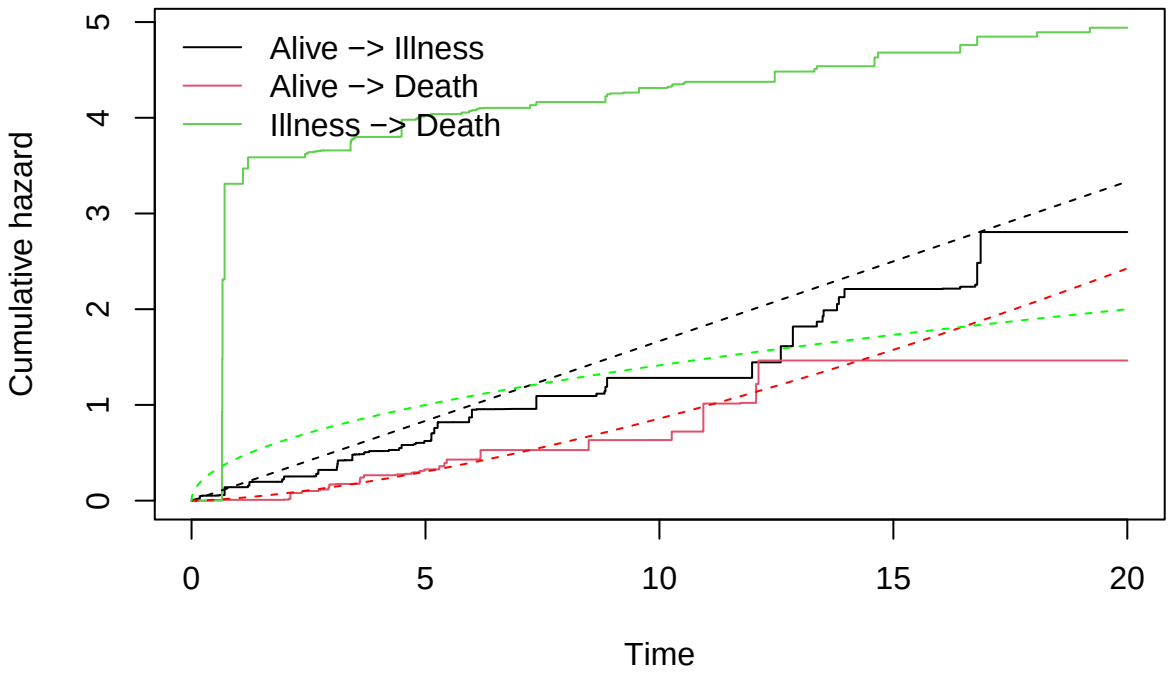


The true curves are inside the confidence intervals of the fitted model, so we generate correct survival times.

2.2 Interval-censored trajectories

We can also use the EM algorithm to recover the underlying intensities. For computational reasons, we fit the model only on the first 100 subjects and restrict ourselves to 30 iterations.

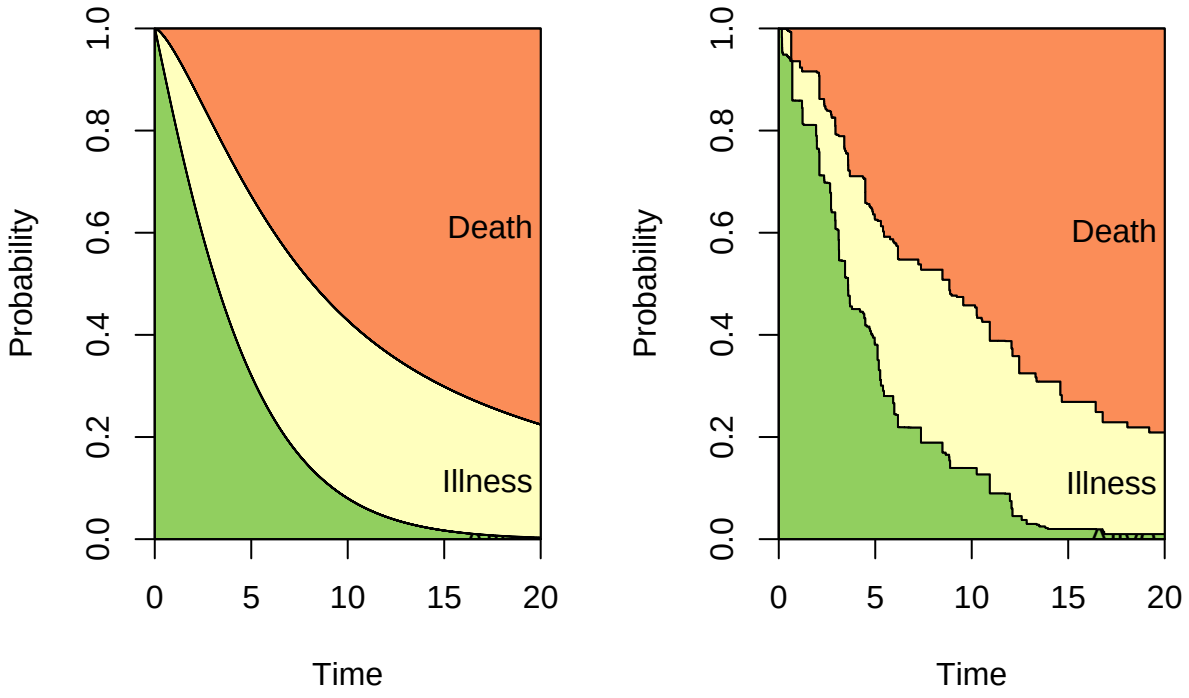
```
EM_fit <- npmsm(subset(data_ID$observed, id < 100), tmat = tmat_ID, tol = 1e-4,
                maxit = 30)
plot(EM_fit)
tseq <- seq(0, 20, 0.01)
lines(tseq, (tseq/scale_ID[1])^shape_ID[1], col = "black", lty = 2)
lines(tseq, (tseq/scale_ID[2])^shape_ID[2], col = "red", lty = 2)
lines(tseq, (tseq/scale_ID[3])^shape_ID[3], col = "green", lty = 2)
```



It looks like the first two transition intensities are recovered quite well (especially for such a short run and small sample size). The intensity of the third transition does not seem to be estimated well. The reason for this is that no subjects start in the illness state, and we can see that the cumulative intensity is estimated to be 0 in early time points. If we move the estimated intensity upward to match the true intensity at time 2, we recover the underlying intensity quite well.

Instead of comparing the cumulative intensities, we can compare the transition probabilities. The `probtrans_weib()` function allows to determine true transition probabilities for multi-state models with Weibull transitions.

```
par(mfrow = c(1, 2))
#Plot true transition probabilities
plot(probtrans_weib(tmat_ID, seq(0, 20, 0.01), shapes = shape_ID,
                    scales = scale_ID))
#Plot estimated transition probabilities
plot(transprob(EM_fit, predt = 0))
```




```
par(opar)
```

These look a lot more similar!

References

Liesbeth C. de Wreede, Marta Fiocco, and Hein Putter. mstate: An r package for the analysis of competing risks and multi-state models. *Journal of Statistical Software*, 38(7), 2011. ISSN 1548-7660. doi: 10.18637/jss.v038.i07.