smms: analysing CAV data with simple exponential model, without covariates

Data

This is a more detailed version of the example in the `README` file. We will use the CAV dataset from the `msm` package (Jackson 2011) as an illustration. The dataset monitors a number of patients for a number of years after heart transplantation. Coronary allograft vasculopathy (CAV) is a condition potentially occurring after hear transplantation. At each time-point the patients are assigned to one of four states: well, mild CAV, severe CAV and death. The time of death is recorded precisely, but the times of entrance into the CAV-states are interval censored

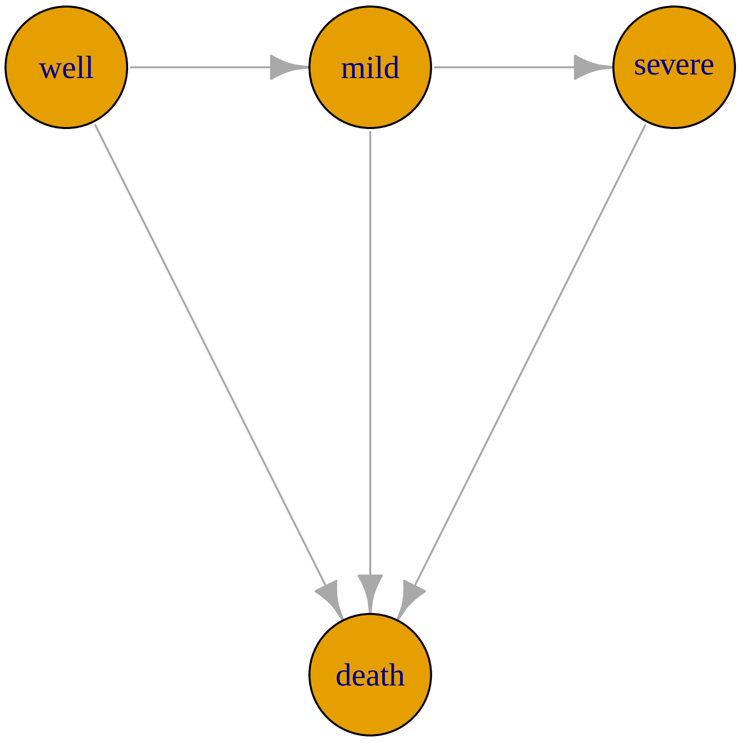
Here we see the observations belonging to two patients. Note here that the states were originally numbered from 1 to 4, but for the sake of this illustration I have changed the state names to "well", "mild", "severe" and "death". This is to demonstrate that the package accepts names as both numbers and strings. After deleting some observations that are deemed incorrect (because they appear to get better, see next paragraph), we end up with 2398 observations in 556 different patients.

```
library(smms)
library(igraph) # For specifying the multi-state graph
library(msm) # To get the CAV dataset

dd = cav
dd = dd[!is.na(dd$pdiag),]

# Remove observations where the patient appears to go back to a previous state (assumed to be impossibl
id_wrong = unique(dd$PTNUM[which(dd$state!=dd$statemax)])
dd = dd[-which(dd$PTNUM %in% id_wrong),]

dd = dd[ ,-c(2, 5, 7, 9, 10)]
colnames(dd)[1:2] <- c("patient","time") # rename relevant columns (necessary in current version)
ddo = dd

# Change state names from 1,2,3,4 to well, mild, severe, death
tab = data.frame(state=1:4,name=c("well","mild","severe","death"))
dd$state = tab$name[match(dd$state,tab$state)]

print(dd[1:11,])
##    patient     time dage pdiag  state
## 1   100002 0.000000   21   IHD   well
## 2   100002 1.002740   21   IHD   well
## 3   100002 2.002740   21   IHD   mild
## 4   100002 3.093151   21   IHD   mild
## 5   100002 4.000000   21   IHD   mild
## 6   100002 4.997260   21   IHD severe
## 7   100002 5.854795   21   IHD  death
## 8   100003 0.000000   17   IHD   well
## 9   100003 1.189041   17   IHD   well
## 10  100003 2.008219   17   IHD severe
## 11  100003 2.991781   17   IHD  death
```

Specifying the model graph

Here we assume a four-state illness death model, since we consider CAV to be irreversible (so we do not allow for patients to move back to less severe states). It is convenient to stick to the same state names as in the dataset when specifying the model graph.

```
# Specify the graph:
gg = graph_from_literal("well"--+"mild"--+"severe"--+"death", "well"--+"death", "mild"--+"death")
par(mar=c(1,1,1,1))
plot(gg,layout=layout_with_sugiyama(gg,layers=c(1,1,1,2))$layout,vertex.size=40)
```



Specifying parametric models

Then, the user has to specify parametric models for all transition times (meaning one for each edge in the graph). In the current version of the package, these models have to be specified by providing density functions (in a specific format detailed below), as well as the corresponding survival functions. The functions will look like the following if one chooses to use simple exponential models for all transitions (i.e. meaning that we are fitting a homogeneous Markov model which could have been fitted with the `msm`package too - but this is for the sake of a simple illustration):

```
f_01 = function(param, x, tt){dexp(tt,exp(param[1]))}
f_12 = function(param, x, tt){dexp(tt,exp(param[2]))}
f_23 = function(param, x, tt){dexp(tt,exp(param[3]))}
f_03 = function(param, x, tt){dexp(tt,exp(param[4]))}
```

```
f_13 = function(param, x, tt){dexp(tt,exp(param[5]))}

S_01 = function(param, x, tt){1-pexp(tt,exp(param[1]))}
S_12 = function(param, x, tt){1-pexp(tt,exp(param[2]))}
S_23 = function(param, x, tt){1-pexp(tt,exp(param[3]))}
S_03 = function(param, x, tt){1-pexp(tt,exp(param[4]))}
S_13 = function(param, x, tt){1-pexp(tt,exp(param[5]))}
```

Important to note:

- **the names of the functions**: these have to be in the form `f_ij` and `S_ij` with i indicating the source state (in the internal numbering system) and j indicating the receiving state. More details on the naming convention below.
- **the arguments of the functions**: these should always be given as `(param, x, tt)` as above. `x` will point to the vector of measured covariates (for a patient) when these are present. When there are no covariates, like in this example, `x` should still be present as an argument, but will not be called within the functions.
- **the scale of the parameters**: for the sake of stable optimisation it is convenient that the parameters live on the real line (instead of the positive half-line as in the common parameterisation of the exponential distribution). Therefore, we include an exponential transformation of the `param` vector, and we recommend that transformation for all positive parameters.
- **the ordering of the parameters**: `param`denotes the full parameter vector for the model.
- Survival functions should be written so that they return 1 when `tt` is negative. Using build-in R CDFs for distributions over the positive half-line will ensure this. Otherwise, if the user codes the survival functions herself, she should ensure that they return 1 when `tt` is negative.

As we saw above, the user has to follow a strict naming convention when specifying the densities and survival functions: within the package, the states are numbered from 0 to $k-1$ (k being the number of states), in a specific order which depends on the graph. To find out how the user defined state names relate to the internal numbering system, use the `names_of_survival_density` function. This always recommended before specifying the model:

```
print(names_of_survival_density(gg))
##   edge_name survival_name density_name from_prev to_prev  type
## 1        01          S_01         f_01      well    mild trans
## 2        03          S_03         f_03      well   death   abs
## 3        12          S_12         f_12      mild  severe trans
## 4        13          S_13         f_13      mild   death   abs
## 5        23          S_23         f_23    severe   death   abs
```

Here we see for example that the density for the edge between “mild” and “severe” should be named `f_12`(as we do above).

Fitting the model

Now we have everything in place in order to fit the multi-state model we have specified above:

```
startval <- c(-2.5,-1.1,-1.2,-3.1,-2.8)

mlo <- smms(startval,dd,gg, mc_cores = 1, hessian_matrix = T)
```

One needs some start values for the optimisation, and we will soon add a function which calculates good starting values for a given model. Increasing the number of cores will make optimisation faster (but will not work on Windows machines). Here we choose to compute the hessian matrix too, which takes a bit more time. With one core this might take something like 5 minutes to compute on an ordinary laptop.

We can compute AIC, and look at the estimated parameters and approximate 95% confidence intervals.

```
# Compute AIC (higher values are better with this definition)
aic <- (-2*mlo$opt$objective)-2*length(mlo$opt$par) #-2887.1

# Look at estimates and 95% confidence intervals.
# On the -Inf to Inf scale:
print(round(est_ci(mlo$opt$par,mlo$hess),2))
##   estimate lower.ci upper.ci
## 1    -2.51    -2.66    -2.36
## 2    -1.11    -1.36    -0.86
## 3    -1.24    -1.48    -1.00
## 4    -3.11    -3.33    -2.90
## 5    -2.76    -3.67    -1.84

# On the 0 to Inf scale (on the transition intensity scale):
round(exp(est_ci(mlo$opt$par,mlo$hess)),2)
##   estimate lower.ci upper.ci
## 1     0.08     0.07     0.09
## 2     0.33     0.26     0.42
## 3     0.29     0.23     0.37
## 4     0.04     0.04     0.06
## 5     0.06     0.03     0.16
```

Figures for interepretation and diagnostics

In the smms package, we have included code for computing various functions of interest, after having fitted a model. Confidence bands are available for all these functions, but computing these bands require differentiating the functions with respect to the parameters and that can be time-consuming for big models.

Occupancy probabilities

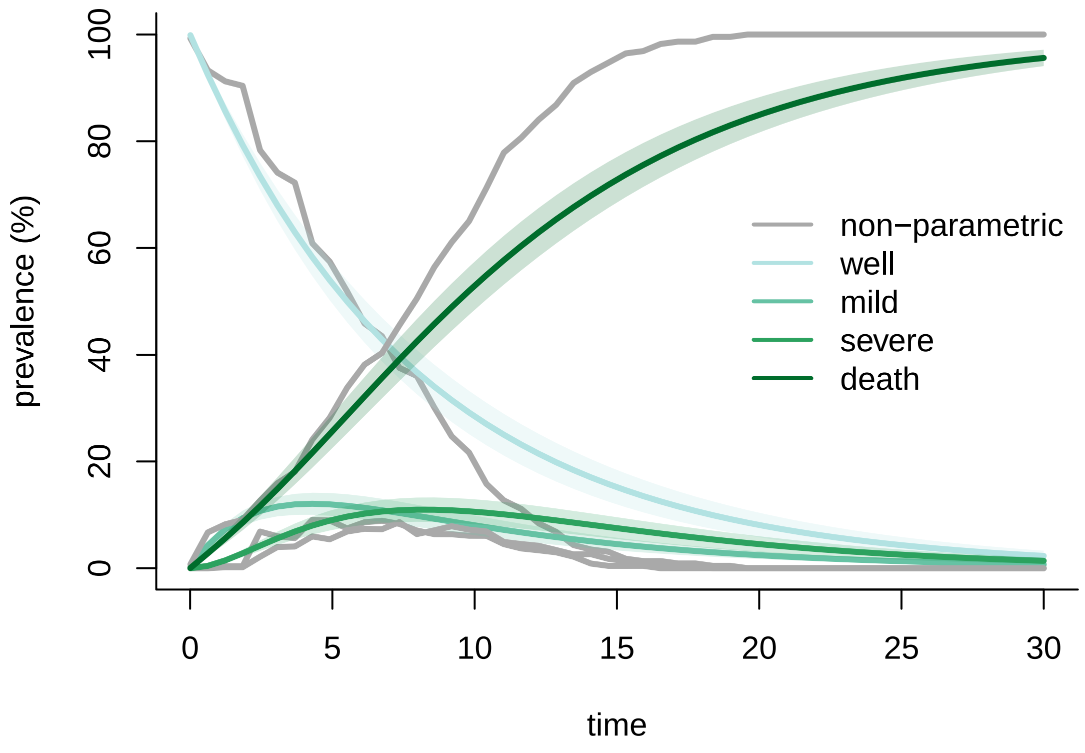
Gives the probability that a patient is found in state i at time t. The state should be provided using the user-defined names:

```
tval <- seq(0.01,30,length=50) # a sequence of time-points over which to compute the state occupancies
p0_ci <- occupancy_prob_ci_band("well",tval,mlo$opt$par,gg,hessian=mlo$hess)
#for computing the confidence bands, the hessian needs to be provided (but there exists a function
# computing only the occupancy probabilities too)
p1_ci <- occupancy_prob_ci_band("mild",tval,mlo$opt$par,gg,hessian=mlo$hess)
p2_ci <- occupancy_prob_ci_band("severe",tval,mlo$opt$par,gg,hessian=mlo$hess)
p3_ci <- occupancy_prob_ci_band("death",tval,mlo$opt$par,gg,hessian=mlo$hess)
```

Plot the occupancy probabilities (the green lines are the fitted state occupancies, the grey lines are the non-parametric estimates):

```
# msm package with covariates (to get non-parameteric prevalence curves)
oneway4.q <- rbind(c(0, 0.25, 0, 0.25), c(0, 0, 0.25, 0.25),c(0, 0, 0, 0.5), c(0, 0, 0, 0))
rownames(oneway4.q) <- colnames(oneway4.q) <- c("Well", "Mild","Severe", "Death")
cav.msm <- msm(state ~ time, subject = patient, data = ddo,qmatrix = oneway4.q, death = 4,method = "BFGS

prev_np <- prevalence.msm(cav.msm,times=tval)
```

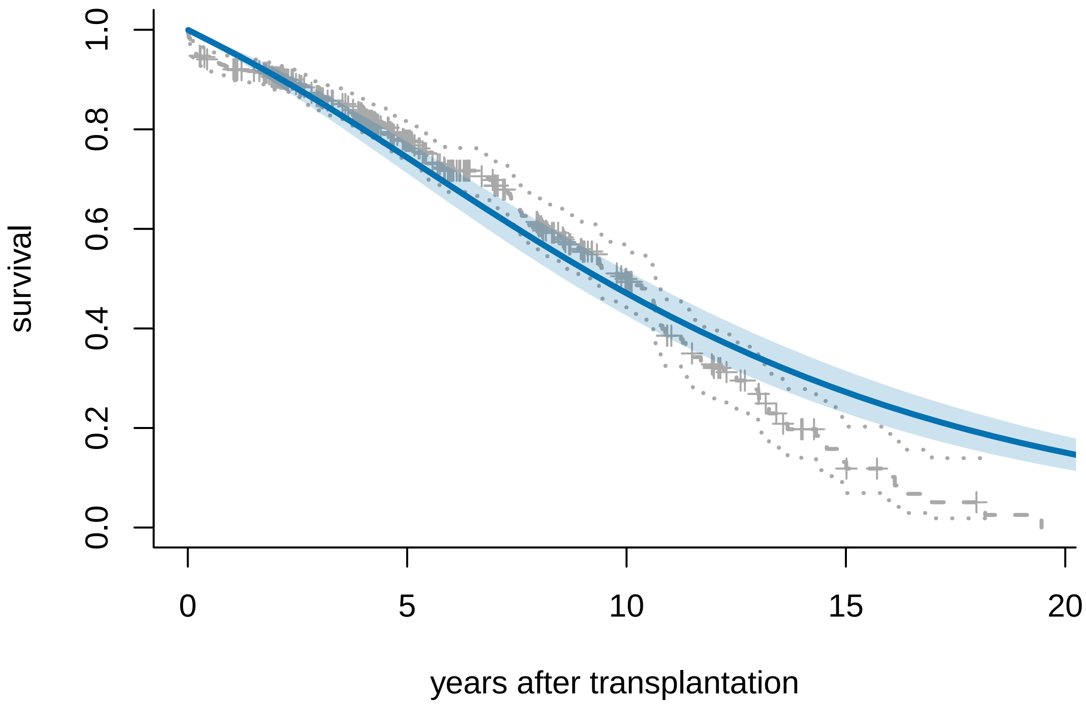
```r
# Plot
par(bty="l")
par(mar=c(4,4,1,2))
par(cex=1)
plot(tval,prev_np$`Observed percentages`[,1],type="l",ylim=c(0,100),lwd=3,xlab="time",col="dark grey",
     ylab="prevalence (%)")
polygon(c(tval, rev(tval)), c(p0_ci$upper*100, rev(p0_ci$lower*100)),
        col = adjustcolor("#b2e2e2",alpha.f=0.2), border=NA)
lines(tval,p0_ci$est*100,col="#b2e2e2",lwd=3)

lines(tval,prev_np$`Observed percentages`[,2],col="dark grey",lwd=3)
polygon(c(tval, rev(tval)), c(p1_ci$upper*100, rev(p1_ci$lower*100)),
        col = adjustcolor("#66c2a4",alpha.f=0.2), border=NA)
lines(tval,p1_ci$est*100,col="#66c2a4",lwd=3)


lines(tval,prev_np$`Observed percentages`[,3],col="dark grey",lwd=3)
polygon(c(tval, rev(tval)), c(p2_ci$upper*100, rev(p2_ci$lower*100)),
        col = adjustcolor("#2ca25f",alpha.f=0.2), border=NA)
lines(tval,p2_ci$est*100,col="#2ca25f",lwd=3)

lines(tval,prev_np$`Observed percentages`[,4],col="dark grey",lwd=3)
polygon(c(tval, rev(tval)), c(p3_ci$upper*100, rev(p3_ci$lower*100)),
        col = adjustcolor("#006d2c",alpha.f=0.2), border=NA)
lines(tval,p3_ci$est*100,col="#006d2c",lwd=3)

legend("right",legend=c("non-parametric","well","mild","severe","death"),
      col=c("dark grey","#b2e2e2","#66c2a4","#2ca25f","#006d2c"),lwd=2,bty="n",cex=1)
```



Overall survival

The overall survival curve gives the probability of not having reached an absorbing state at time t. The grey line is the Kaplan-Meier estimator, the blue line is the fitted line from the model.

```r
tval <- seq(0.01,30,length=50)
So <- overall_survival_ci_band(tval,mlo$opt$par,gg,hessian=mlo$hess)

par(bty="l")
par(mar=c(4,4,2,2))
par(cex=1)
plot.survfit.msm(cav.msm, col.surv="dark grey",lwd.surv=2,xlab="years after transplantation",
                 ylab="survival",main=" ",legend.pos=c(30,2),col=NULL,lwd=3)
# using msm package for the nonparametric estimate (for now)
polygon(c(tval, rev(tval)), c(So$upper, rev(So$lower)),
        col = adjustcolor("#0571b0",alpha.f=0.2), border=NA)
lines(tval,So$est,col="#0571b0",lwd=3)
```



Transition probabilities

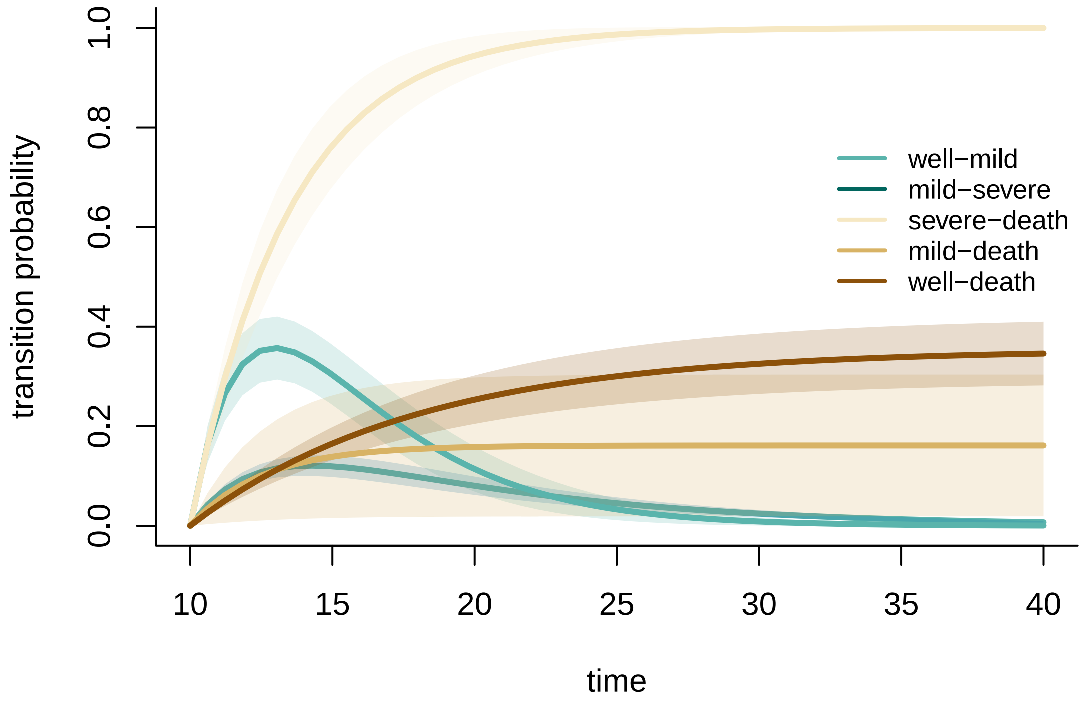
The probability that a patient is found in state i at a time-point t given that she was in state j at a prior time-point v. In the current implementation, t can be a vector of time-points while v is a single number. The user has to specify a particular edge or transition for which the function computes the transition probability, for example the transition between "mild" and "severe".

```r
tval <- seq(10,40,length=50)
vt <- 10
t01_ci <- transition_prob_ci_band("well-mild",tval,vt,mlo$opt$par,gg,hessian=mlo$hess)
t12_ci <- transition_prob_ci_band("mild-severe",tval,vt,mlo$opt$par,gg,hessian=mlo$hess)
t23_ci <- transition_prob_ci_band("severe-death",tval,vt,mlo$opt$par,gg,hessian=mlo$hess)
```

```
t13_ci <- transition_prob_ci_band("mild-death",tval,vt,mlo$opt$par,gg,hessian=mlo$hess)
t03_ci <- transition_prob_ci_band("well-death",tval,vt,mlo$opt$par,gg,hessian=mlo$hess)
```

Plot the transition probabilities (the colored lines are the fitted transition probabilities):

```
par(bty="l")
par(mar=c(4,4,2,2))
par(cex=1)
plot(tval,t01_ci$est,type="l",ylim=c(0,1),col="#5ab4ac",lwd=3,ylab="transition probability",xlab="time")
polygon(c(tval, rev(tval)), c(t01_ci$upper, rev(t01_ci$lower)),
        col = adjustcolor("#0571b0",alpha.f=0.2), border=NA)
polygon(c(tval, rev(tval)), c(t12_ci$upper, rev(t12_ci$lower)),
        col = adjustcolor("#5ab4ac",alpha.f=0.2), border=NA)
polygon(c(tval, rev(tval)), c(t23_ci$upper, rev(t23_ci$lower)),
        col = adjustcolor("#f6e8c3",alpha.f=0.2), border=NA)
polygon(c(tval, rev(tval)), c(t13_ci$upper, rev(t13_ci$lower)),
        col = adjustcolor("#d8b365",alpha.f=0.2), border=NA)
polygon(c(tval, rev(tval)), c(t03_ci$upper, rev(t03_ci$lower)),
        col = adjustcolor("#8c510a",alpha.f=0.2), border=NA)
lines(tval,t12_ci$est,lwd=3,col="#5ab4ac")
lines(tval,t23_ci$est,lwd=3,col="#f6e8c3")
lines(tval,t13_ci$est,lwd=3,col="#d8b365")
lines(tval,t03_ci$est,lwd=3,col="#8c510a")
legend(x=32,y=0.8,legend=c("well-mild","mild-severe","severe-death","mild-death","well-death"),
       col=c("#5ab4ac","#01665e","#f6e8c3","#d8b365","#8c510a"),lwd=2,bty="n",cex=0.8)
```



Writing out the log-likelihood

The `smms`package contains functions for writing out the formula for the log-likelihood belonging to a particular graph in latex format. The user needs to provide the graph, and specify whether the time of entrance into the absorbing state is observed exactly or not.

```
write_loglikelihood(gg,abs_exact = T)
```

This function will write a txt file with the log-likelihood formula to the working directory.

$$
\begin{aligned}
\ell_n(\theta) = & \sum_{k=0}^{n_0} \log\{S_{01}(t_{0M})S_{03}(t_{0M})\} + \sum_{k=0}^{n_{01}} \log\{\int_{t_{0M}}^{t_{1m}} f_{01}(s)S_{03}(s)S_{12}(t_{1M}-s)S_{13}(t_{1M}-s)\,ds\}+ \\
& \sum_{k=0}^{n_{012}} \log\{\int_{t_{0M}}^{t_{1m}} \int_{t_{1M}-s}^{t_{2m}-s} f_{01}(s)f_{12}(u)S_{03}(s)S_{13}(u)S_{23}(t_{2M}-s-u)\,du\,ds\}+ \\
& \sum_{k=0}^{n_{02}} \log\{\int_{t_{0M}}^{t_{2m}} \int_{0}^{t_{2m}-s} f_{01}(s)f_{12}(u)S_{03}(s)S_{13}(u)S_{23}(t_{2M}-s-u)\,du\,ds\}+ \\
& \sum_{k=0}^{n_{0123}} \log\{\int_{t_{0M}}^{t_{1m}} \int_{t_{1M}-s}^{t_{2m}-s} f_{01}(s)f_{12}(u)f_{23}(t_{3m}-s-u)S_{03}(s)S_{13}(u)\,du\,ds\}+ \\
& \sum_{k=0}^{n_{03}} \log\{\int_{t_{0M}}^{t_{3m}} \int_{0}^{t_{3m}-s} f_{01}(s)f_{12}(u)f_{23}(t_{3m}-s-u)S_{03}(s)S_{13}(u)\,du\,ds+ \\
& \int_{t_{0M}}^{t_{3m}} f_{01}(s)f_{13}(t_{3m}-s)S_{03}(s)S_{12}(t_{3m}-s)\,ds + f_{03}(t_{3m})S_{01}(t_{3m})\}+ \\
& \sum_{k=0}^{n_{013}} \log\{\int_{t_{0M}}^{t_{1m}} \int_{t_{1M}-s}^{t_{3m}-s} f_{01}(s)f_{12}(u)f_{23}(t_{3m}-s-u)S_{03}(s)S_{13}(u)\,du\,ds+ \\
& \int_{t_{0M}}^{t_{1m}} f_{01}(s)f_{13}(t_{3m}-s)S_{03}(s)S_{12}(t_{3m}-s)\,ds\}+ \\
& \sum_{k=0}^{n_{023}} \log\{\int_{t_{0M}}^{t_{2m}} \int_{0}^{t_{2m}-s} f_{01}(s)f_{12}(u)f_{23}(t_{3m}-s-u)S_{03}(s)S_{13}(u)\,du\,ds\}.
\end{aligned}
$$