We assume some basic familiarity with R:

- Commands are given via *functions*. The specific values ("arguments") are given within parentheses, separated by a comma:

  ```
  foo(arg1, arg2, ...)
  ```

- The result is sent to the output ("console") window, unless it is assigned to an *object*:

  ```
  my.result <- foo(arg1, arg2, ...)
  ```

- Most calculations have the results stored in a list format. A list has components that can be accessed via the $ sign:

  ```
  my.result$comp1
  ```

1. Basic Concepts

A first look at the data The help file with background information is generated via `help(ebmt1)`.

For inspection of the actual data, you can use the functions `head`, `tail` and `View`. For summary information of the total data set you can use the functions `dim` (number of rows and columns), `str` (basic information per column) and `summary`. In `summary`, the type of summary is adapted to the data format per column. Dichotomous variables that are stored via the values zero and one are treated as numeric. The function `table` can be used to get the total number per level.

Some data preparation No further hints

Estimation of overall event time distribution You use the `survival` package, but there is no need to load it if you have attached the `mstate` package first; it is automatically loaded by `mstate`.

The Kaplan-Meier is obtained via the `survfit` function. Since you combine both events, you can use `Surv(time,stat>0)~1` as its first argument. The estimates at all time points is generated via the `summary` function. It has an argument `times` that can be specified to obtain relapse-free survival after one and five years.

The estimate is plotted via the `plot` function. Its `fun` argument is used to plot on the complementary scale of the cumulative incidence (see the help file of `plot.survfit`).

A Cox model for relapse-free survival Use the `coxph` function. The `print` command gives a short summary of the model fit, whereas the `summary` function gives more details. The `cox.zph` function tests for non-proportionality.

Log-rank test The log-rank test is performed via the `survdiff` function. The cumulative hazard is plotted using the `fun` argument in the `plot` function.

2. Competing Risks; Nonparametric Estimation

Estimation, ***(a) using standard code in the*** **`survival`** ***package*** The competing risks setting is specified in the `Surv` function via the `type` argument. The actual calculations are performed by the `survfit` function. The simplest way to show the estimates at one and five years is by specifying the `times` argument in the `summary` function.

Confidence intervals are obtained from the components `lower` and `upper` of the `survfit` or `summary` output. If you want them at specific time points, it is easier to use the `summary` function because the time points can be pre-selected. Since there are two event types, the components `lower` and `upper` are matrices with two columns. If you stored the output of the `survfit` function in an object `cuminc.1`, you can use `summary(cuminc.1, times=c(1,5))$lower)` and the same for the upper limits.

Estimation, ***(b) using the*** **`etm`** ***package*** The calculations are performed by the `etmCIF` function. You can show the results in two ways. Suppose the result of the calculations has been assiged to `cuminc.2`. One option is to apply the `summary` function on `cuminc.2` and assign the output to an R object. In that object, select the event of interest within the first, unnamed, component:

```
summ.etm <- summary(cuminc.2, ci.fun="log")
summ.etm.1 <- summ.etm[[1]][["CIF 1"]]
summ.etm.2 <- summ.etm[[1]][["CIF 2"]]
select1and5 <- c(sum(summ.etm.1$time≤1),sum(summ.etm.1$time≤5))
summ.etm.1[select1and5,]
summ.etm.2[select1and5,]
```

Another option is to use the `trprob` and `trcov` functions with as first argument `cuminc.2[[1]]`. In these functions you can specify the time points of interest, but you have to calculate the confidence intervals yourself based on the estimated standard error. As an example, for relapse as end point you obtain estimates and confidence intervals on the “log” scale in formula (1.23) on page 36 of the book via

```
est.01 <- trprob(cuminc.2[[1]],tr.choice="0 1",timepoints=c(1,5))
est.01
CI.01 <- trcov(cuminc.2[[1]], tr.choice="0 1", timepoints=c(1,5))
est.01*exp(-qnorm(0.975)*sqrt(CI.01)/est.01)
est.01*exp(qnorm(0.975)*sqrt(CI.01)/est.01)
```

Estimation, ***(c) using the weighted product-limit form*** The arguments in the `crprep` function that always need to be specified are the time variable in `Tstop` and the status variable in `status`. The `cens` argument does not have to be specified if the value 0 denotes censored individuals. You can obtain the data set with weights for both end points at the same time via the `trans` argument (default is 1). The **type** information is transferred to the weighted data set in exactly the same way as the covariables **score** and **age**: via the `keep` argument.

For inspection, you can use the same functions that were used for inspection of the `ebmt1` data set in Chapter 1.

When calculating the weighted product-limit estimate, note that the weighted data sets were created at the same time for both end points. In the output they can be separated by the `failcode` variable. You can calculate the two curves at once by specifying `failcode` at the right hand side of the formula in `survfit` and specify the `status` argument in the `Surv` function via `status==failcode`. Try to understand why.

For investigation of the estimates, you can use the strength and flexibility of the existing functions in the `survival` package. You can obtain estimates and confidence intervals for a specific event type using the selection mechanism with square brackets within the `summary` function: `cuminc.3[1]` and `cuminc.3[2]`.

Some plots (a) You can use the standard plot commands from the `survival` or `etm` package[1].

(b) Use that the sum of the cause-specific cumulative incidences is equal to the overall cumulative incidence.

(c) The `plot.survfit` function allows to choose the scale via the `fun` argument, `"identity"` for the survival scale and `"event"` for the scale of the cumulative incidence. This plot is more difficult to create based on the result from approach (b). The reason is that the `plot.etmCIF` function only plots on the scale of the cumulative incidence. You can plot on the survival scale if you distill the information from `cuminc.2[[1]]` or `summary(cuminc.2)[[1]]`.

Effect of EBMT risk score Calculations are very similar to the overall curves. The only thing that changes is the specification of the EBMT risk score variable on the right hand side of the formula.

You split the plot window into two subwindows using the command `par(mfrow=c(1,2))`. For the plots based on approach 1.(a), we provide a suggestion for the code to use. Assume that the estimates have been stored in an object named `cuminc.1.score`.

```
par(mfrow=c(1,2))
with(cuminc.1.score, plot(time[1:strata[1]], prev[1:strata[1],1], type="s",
             ylim=c(0,0.55), xlab="Time", ylab="Cumulative Incidence"))
with(cuminc.1.score, lines(time[(strata[1]+1):sum(strata[1:2])],
          prev[(strata[1]+1):sum(strata[1:2]),1],type="s",lty=2))
with(cuminc.1.score, lines(time[(sum(strata[1:2])+1):sum(strata[1:3])],
      prev[((sum(strata[1:2])+1)):sum(strata[1:3]),1], type="s",lty=3))
legend("bottomright", levels(ebmt1$score), lty=1:3,bty="n")

with(cuminc.1.score, plot(time[1:strata[1]], prev[1:strata[1],2], type="s",
             ylim=c(0,0.55), xlab="Time", ylab="Cumulative Incidence"))
with(cuminc.1.score, lines(time[(strata[1]+1):sum(strata[1:2])],
         prev[(strata[1]+1):sum(strata[1:2]),2],type="s",lty=2))
with(cuminc.1.score, lines( time[(sum(strata[1:2])+1):sum(strata[1:3])],
      prev[((sum(strata[1:2])+1)):sum(strata[1:3]),2], type="s",lty=3))
legend("bottomright", levels(ebmt1$score), lty=1:3,bty="n")
```

Note that the estimates are in the component `prev` of the `cuminc.1.score`; the first column is for relapse, the second for death. Values for all three levels of **code** have been concatenated. The problem is the selection of the relevant elements, which is most easily done via the `strata` component of the output.

Choice of the weight function in the product-limit form You obtain separate censoring weights per value of EBMT score by using the `strata` argument in the `crprep` function.

[1]Currently, in version 2.38-1 of the survival package approach (a) gives an error message when choosing the log-log scale.

For the plots, we recommend to make a separate plot per event type, and use the overlaid format for the values of EBMT risk score. Split the plot window into two subwindows using the command `par(mfrow=c(1,2))`. Making the plots with `plot.survfit` and `lines.survfit` is most easily done if you perform the calculations separately per end point.

When you are done, use `par(mfrow=c(1,1))` to return to one overall plot window.

Log-rank tests No hints.

3. Multi-state models; nonparametric estimation

Define the structure No hints

Create the stacked data set Before you can perform the calculations, you need to do two things.

1. Define the transitions. In `etm` and `mstate` you use a transition matrix. The format of the matrix differs between both (see Sections 3.7.1 and 3.7.3 of the book). Note that `mstate` has a function `trans.illdeath` that helps in creating the matrix in an illness death model. In `msSurv`, the construction is slightly more involved. See the explanation in Section 3.7.2 of the book.
2. Transform the data to long format. You can use the functions `etmprep` or `msprep` to create the required long format. This is explained in Sections 3.7.1 and 3.7.3 of the book.

   `msSurv` does not have a function to create the data set in the required transition-based long format. One option is to use the `etmprep` function, as was explained in Section 3.7.2. An alternative is to create the data set ourselves. For an illness-death model, this is not too difficult. You can use the following code. Try to understand what it does.

```
ebmt.msSurv <- with(ebmt1, data.frame(id=patid, start=0,
                                   stop=pmin(srv,rel), start.stage=1,
                             end.stage=ifelse(relstat==1,2,3*srvstat)))
tmp <- with(subset(ebmt1, relstat==1), data.frame(id=patid,
          start=rel, stop=srv, start.stage=2, end.stage=3*srvstat))
ebmt.msSurv <- rbind(ebmt.msSurv, tmp)
ebmt.msSurv <- ebmt.msSurv[order(ebmt.msSurv$id,ebmt.msSurv$start),]
```

Estimation of cumulative hazard If you use the `etm` package, it is easiest to calculate the Nelson-Aalen estimates via the `mvna` function in the companion package with the same name. The resulting estimates can be plotted via the `plot.mvna` function.

In the `msSurv` package you use the `msSurv` function. It is the workhorse that computes everything. For the actual plotting you need to distill the relevant components from its output. Suppose the output has been stored in `Prob.msSurv`. Then you can obtain the estimates using `et(Prob.msSurv)` (for the observed transition times) and `cumsum(I.dA(Prob.msSurv))` (for the cumulative hazard).

If you use the `mstate` package, the stacked format allows to calculate all cumulative hazards via the basic `coxph` function. However, for plotting it is recommended to first apply the `msfit` function to the output of the `coxph` function.

Estimation of transition probabilities The state prevalences are the state occupation probabilities $\mathrm{P}_{1,k}(0,t)$ for $k = 1, 2, 3$. They are basically transition probabilities.

In the `etm` package, you first apply the `etm` function. Next, you use the `summary.etm` function to store all estimates in a format for further analysis. The result is a list with names equal to the transitions, let's call it `Prob.etm.summ`. Since you cannot select the time points at which you want the numeric values, you have to use a little trick. We show the code for one of the transitions. You obtain the probabilities for the other transitions in the same way.

```
tail(subset(Prob.etm.summ[["transplant relapse"]],time≤5*365.25),1)
```

In the `msSurv` package, any estimate can be obtained by selecting the appropriate slot in the output of the `msSurv` function. The estimates of the state occupation probabilities at five years are best obtained via the `SOPt` slot.

```
SOPt(Prob.msSurv, t=5*365.25, covar=TRUE)
```

You need to calculate the confidence intervals yourself, based on the estimated variance in the output of `SOPt`.

In the `mstate` package, you use the `probtrans` function to compute the estimates. The result is a list, let's call it `Prob.mst`. The first component gives the estimates for the transitions out of state 1, which is what you need. Since you cannot select the time points at which you want the numeric values, you have to use a little trick.

```
Est.5yrs <- tail(subset(Prob.mst[[1]],time≤5*365.25),1)
```

You need to calculate the confidence intervals yourself, based on the information in `Est.5yrs`.

Creation of informative plots In the `etm` package, the `plot.etm` function does not allow to plot in stacked format. You first need to create a data frame with all relevant information from the output of `summary.etm`. Next, use the standard `plot` and `lines` functions. The second plot is easily made with the `plot.etm` function.

In the `msSurv` package, use the `et` and `AJs` slots from the output of the `msSurv` function in the standard `plot` and `lines` functions. Note that the `AJs` slot is an array. The first dimension of the `AJs` slot defines the outgoing state, the second the incoming state, and the third dimension contains the estimates at all time points. Hence, the plot of the probability to remain in the transplant state is made via

```
plot(et(Prob.msSurv), AJs(Prob.msSurv)[1,1,], type="s")
```

The other estimates are added to this plot.

In the `mstate` package, the standard `plot.probtrans` function plots the transition probabilities out of state 1. There is no pre-defined function to plot the estimates for all three transitions at once. You are recommended to first create a data frame with all relevant information from the output of the `probtrans` function and next use the standard `plot` and `lines` functions.

Time from transplantation to death The basic idea is the same as in the previous exercises.

4. Regression. Cause-Specific/Transition Hazard

Competing risks, separate analyses A proportional hazards model on the relapse-specific hazard can be fitted by censoring individuals when they experience the competing event. Only the individuals with `stat` equal to 1 had a relapse as a first event. Similarly for death as a first event: censor individuals when they had a relapse.

Competing risks, combined analysis When using `Webmt1` you need to restrict to the rows that have `count` equal to 1. In the analysis, the event type as given in the `failcode` column is included as stratum variable.

Since EBMT score is a categorical variable, you can create one extra column that contains all the type-specific information. One way to do this is as follows

```
Webmt1$score.comb <- with(Webmt1, factor(
                           (as.numeric(score)-1)*(failcode==1)+
                 3*(as.numeric(score)-1)*(failcode==2), labels=
   c("Low","Medium.Rel","High.Rel","Medium.Death","High.Death")
))
```

Why do you need only five levels, not six? Internally, this information is transformed into four columns with dummy variables.

You can create these type-specific variables yourself via

```
Webmt1$Medium.Rel <- with(Webmt1,
                ifelse(score=="Medium risk"&failcode==1, 1, 0))
Webmt1$High.Rel <- with(Webmt1,
                  ifelse(score=="High risk"&failcode==1, 1, 0))
Webmt1$Medium.Mort <- with(Webmt1,
                ifelse(score=="Medium risk"&failcode==2, 1, 0))
Webmt1$High.Mort <- with(Webmt1,
                  ifelse(score=="High risk"&failcode==2, 1, 0))
```

or you can use the `expand.covs` function.

Competing risks, test for equality of effects For the test per level, you can use the output from the previous exercise.

For the likelihood ratio test, you need to compare the model that allows the effects to differ by event type with the one in which the effect of EBMT score is assumed to be equal for both event types. This model can be obtained by using the original `score` variable. Both models are compared via the `anova` function.

Multi-state model, create and inspect the stacked data set If you have performed the analyses of Section 3.8 with the `mstate` package, the data set in stacked long format has been created already. However, you need to create it again because you need to include the variables `score` (EBMT score) and `yrel` (year of relapse) via the `keep` argument of the `msprep` function.

For the summary based on `ebmt1`, use the `table` function on the variables `stat` and `srvstat`.

Multi-state model, create transition-specific covariables In order to create transition-specific covariables with short names, you change the value in the `longnames` argument in `msprep`.

Multi-state model, general model Similar to the competing risks setting, you allow for a separate baseline hazard per transition by including `trans` as stratum variable.

Multi-state model, same hazard ratios? This is similar to what you did in the competing risks analysis. In order to perform the likelihood ratio test, you first need to fit a model in which the effect of EBMT score is assumed equal over all transitions, still allowing for separate baseline hazards. Next, you use the `anova` function.

Multi-state model, proportional baseline hazards You can define `rel.surv` as 0 if no relapse has occurred and 1 if a relapse has occurred. It can be created via

```
msebmt$rel.srv <- 0
msebmt$rel.srv[msebmt$trans==3] <- 1
```

The model can be fitted using a stratified Cox regression model with strata defined by the receiving state (`to` in the stacked data set), and with `rel.srv` as additional covariable. Testing for proportionality can be done using the `cox.zph` function.

Multi-state model, time trend in relapse? In order to get rid of the missing values in `yrel`, you need to replace them by a fixed value, e.g. by 0:

```
msebmt$yrel1.3[is.na(msebmt$yrel1.3)] <- 0
msebmt$yrel2.3[is.na(msebmt$yrel2.3)] <- 0
```

You need to add `yrel1.3` and `yrel2.3` to the model.

For the effect of relapse in the latest period, you can redefine the `yrel2.3` variable. We store this data set in an object `tmp`. Understand what is happening here:

```
tmp <- msebmt
tmp <- within(tmp,{
    yrel2.3[!is.na(yrel)&yrel=="1993-1996"&trans==3] <- 1
    yrel2.3[!is.na(yrel)&yrel=="2000-"] <- 0
   })
```

5. Regression; Translation to Cumulative Scale

Competing risks, from cause-specific to cumulative Use the `mstate` package to translate results form a proportional cause-specfic hazards model to the cumulative scale. The same steps are used as in Chapter . First you fit the proportional hazards model using the stacked data set. It is easiest to use the model based on the stacked data set that you already fitted in Chapter . Before you can go to the next step and use the `msfit` function, you first need to define the transition matrix for the competing risks setting and create data for three new individuals, one for each value of the EBMT risk score. Finally, you use the `probtrans` function to compute the cause-specific cumulative incidence.

When plotting the results, we recommend to first plot the nonparametric estimates and then add the ones based on the model using the standard `lines` function. Note that the first component of the list that is created by `probtrans` contains all estimates.

Competing risks, effects on subdistribution hazards You can use the `Webmt.score` object that you created in Chapter . You only need to add the covariables to this data set that allow you to fit the model for both event types at once. One way is to create type-specific covariables via the `expand.covs` function.

The basic structure of the model is the same as when fitting a proportional cause-specific hazards model. The only difference is that individuals that experience the competing event remain included with a weight specified via the `weights` argument.

To predict the cause-specific cumulative incidence for the three EBMT score groups, you can use the functionality in the `survival` package. You can specify all covariable combinations in a single data frame with six rows. Each row represents a combination of a value of risk score and event type. Next, you calculate predicted survival curves via the `newdata` argument in `survfit`.

When plotting the results, it is easiest to use the `plot` and `lines` functions from the `survival` package.

Proportional hazards can be tested for via the `cox.zph` function.

Multi-state analysis, cumulative transition hazards Again use the `msfit` function from the `mstate` package to perform the computations. Before you can use the `msfit` function, you need to create a data frame that represents an individual with Low risk score. This data frame contains values for all covariables used in the model on which it is based. It has as many rows as there are transitions in the model, and the values in row k refer to the k-th transition. You also need to define a **strata** column.

Use the `str` function to investigate the structure of the output of `msfit` and use the `summary.msfit` function to summarize results.

There is a `plot` method for `msfit` objects that makes it easy to create the graph.

Multi-state analysis, impact of year of relapse on hazard You can use the `time` and `Haz` components of `H0` to `H4` in the standard `plot` function.

Multi-state analysis, transition probabilities The result of `msfit` can be used as input for `probtrans`. The output of `probtrans` is a list with S elements (S being the number of states) with element `[[g]]` containing estimates of either $P_{gh}(s,t)$ for all event times $t > s$ in case of fixed history prediction at time s, or $P_{gh}(s,t)$ for all event times $s < t$ in case of fixed horizon prediction at time t. You call `probtrans` with `HvH` as input.

You can use the `plot.probtrans` function to obtain the state occupation probabilities. The argument `type` distinguishes between different types of plots (have a look at the help function for more information).

Multi-state analysis, impact of year of relapse on transition probability Basically, you need to create new data frames in which `yrel1.3` and `yrel2.3` are changed according to the calendar period of interest. After this, you apply `msfit` and `probtrans` again.