Inverse probability weighting enhances absolute risk estimation in three common study designs of nosocomial infections

Supplementary table and figure

Appendix 1 - R code of Publication

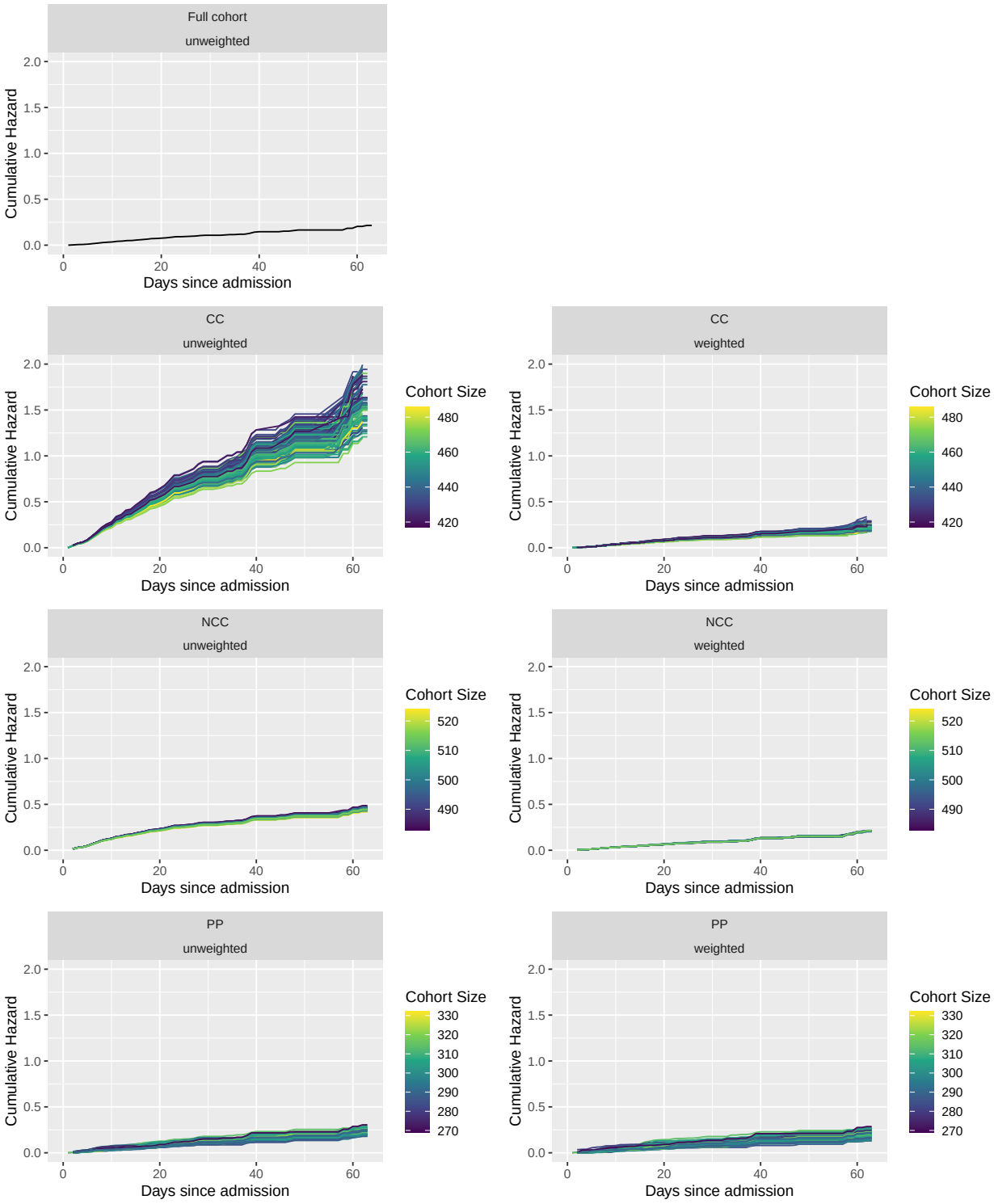
Appendix 2 - R Tutorial

Supplement Table 1 Population description.

	Full cohort	Case cohort		Nested case-control		Point prevalence	
		Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**
No. cases	129.0	129.0	0.0	129.0	0.0	46.5	4.3
No. controls	2124.0	317.0	15.0	372.9	9.0	254.3	12.1
No. total	2253.0	446.0	15.0	501.9	9.0	300.8	12.7
Length of stay in the ICU in days							
Min.	1.0	1.9	0.3	2.8	0.4	2.4	0.5
1.Q	5.0	6.3	0.5	14.2	0.7	15.0	1.2
Median	9.0	13.6	0.8	26.1	0.6	31.9	1.8
Mean	19.2	27.9	1.2	43.1	0.6	54.5	1.6
3.Q	21.0	30.3	1.6	49.7	1.6	65.0	3.2
Max.	898.0	491.5	207.2	898.0	0.0	898.0	0.0

Each study design was sampled 100 times. The mean and standard deviation (SD) over the 100 samples is reported. The sampling fraction of the CC study was 15%. For each case in the NCC study 4 controls were matched by the length of stay. ICU intensive care unit, Min. Minimum, Max. Maximum, No. Number of, Q. Quartile. Bold font identifies headings.

Supplement Figure 1 Cumulative hazard for infection for the full cohort, the Case-Cohort (CC), Nested Case-Control (NCC) and Point Prevalence (PP) design.



Each line represents one of the 100 case cohort samples. The colour of the lines represents the total number

of patients of each sample. As the number of cases is the same in all samples (n=129) in the CC and NCC design, the color can also be interpreted as the proportion of cases to controls.
In the CC design, the random occurrence of cases in or outside of the sub-cohort affects the cohort size and with it the proportion between controls and cases leading to different degrees of bias. IPW reduce this variance and lead to an estimation closer to the "true" value of the full cohort.

Appendix 1 - R Code of Publication

```
# This script was written with R version 4.1.1 (2021-08-10)
#install.packages("pacman") # run this if you do not have installed the package pacman
library(pacman)

pacman::p_load(Epi, multipleNCC, plyr, survival, lattice, mstate, dplyr, mvna,
               sandwich, psych, here, bshazard, etm, mefa, lmtest, huxtable,
               pdftools, tikzDevice, kableExtra, ggplot2, data.table, gridExtra,
               survminer, scales, mice, miceadds, tibble, cowplot, purrr )
```

Preparing Dataset

Our event of interest is the bloodstream infection (BSI). Our covariable of interest is the Charlson Comorbidity Index (CHARLSON_FIRST).
Only patients without missing values are considered in the analysis.

The dataset is prepared to fit functions for multi-state modeling. We define the states admission as state 0, bloodstream infection as state 1 and discharge as state 2. There is one row per transition in the dataset i. e. one patient could have several rows.

id: patient id
from: transition start state
to: transition end state
time: time of transition
entry: start time of a state
exit: end time of a state

```
RWdata <- read.csv(here("data/file.csv"))
RWdata$X <- NULL

# complete-case analysis
RWdata <- subset(RWdata, complete.cases(RWdata[, c("CHARLSON_FIRST")]))

# Data format for R packages
# Event of interest: Bloodstream infection

RWdata$from <- 0
RWdata$entry <- 0
RWdata$to <- ifelse(is.na(RWdata$infection_bloodstream_1st_day), 2, 1)
RWdata$exit <- NA
RWdata[RWdata$to == 2,]$exit <- RWdata[RWdata$to == 2,]$day_in_ICU
RWdata[RWdata$to == 1,]$exit <- RWdata[RWdata$to == 1,]$infection_bloodstream_1st_day

RWdata <- select(RWdata, ID, from, to, entry, exit, infection_bloodstream,
                 CHARLSON_FIRST, day_in_ICU)

RWdata <- rename(RWdata, c("id" = "ID", "event1" = "infection_bloodstream"))

RWdatac <- RWdata
```

Description of the Full Cohort

Only complete cases with no missing values for the Charlson Comorbidity Index are considered. In total 2253 patients are in the data set. Of those 129 acquired an infection.

```
# Create a function for NCC sampling
CreateSamplestat <- function(nestinfo, data, samplestat, id, set) {

  ## Adjusts samplestat vector for use in multipleNCC package.
  ## Sampled controls are given "1."
  ##
  ##
  ## Args:
  ##   nestinfo:    data frame of information for reduced cohort after
  ##                IDS sampling
  ##   joininfo:    full cohort data frame before sampling
  ##   samplestat:  vector of sampling and status (case or control) info
  ##                before adjustment
  ##   id:          id for joining data frames
  ##   set:         time-matched case-control set
  ##
  ## Returns:
  ##   data frame for full cohort with adjusted samplestat vector

  ## marks patient as sampled
  nestinfo[, samplestat] <- replace (nestinfo[, samplestat],
                                     nestinfo[[samplestat]] == 0, 1)


  # gives columns consistent names for join function
  ncc1 <- data.frame( nestinfo[[id]],  nestinfo[[samplestat]],  nestinfo[[set]])
  colnames(ncc1) <- c(id, samplestat, set)

  # removes duplicate event times
  ncc1 <- unique(ncc1[!duplicated(ncc1[-3]),] )

  # merges new samplestat vector to original data frame
  ncc1 <- join(ncc1, data, by = id, type = "full")

  # returns data frame
  ncc1

}
```

```
# Create mixin12 function
# This function is used to create a msfit object, which is needed for the calculation
# of the transition probabilities in the multi-state model for the weighted analysis
# with time-depended weights.

mixin12 <- function( coxphData, exitTime) {

# input data: coxphData should be a dataframe with two columns:
# coxphData$time and coxphData$hazard
# exitTime is a vector of timepoints
```

```r
# In our data coxphData$time are the transition times of a specific event
# coxphData$hazard are the values of the cumulative hazards at each trans. time
# exitTime is the vector, that contains the transition times of all possible events
  ## Get all timepoints only once

  allTimes <- sort(union(coxphData$time, exitTime))

  ## create dataframe with the unique time points and a hazard column
  result <- data.frame(time = allTimes, hazard = rep(-1,length(allTimes)))

  ## check if time point of allTimes has a cumulative hazards value
  for( i in seq(1, nrow(result)) ) {
    coxTime <- which(coxphData$time == result$time[i] )


    if( length(coxTime) > 0 ) {
      ## This shouldn't happen, since we have unique time points:
      if( length(coxTime) > 1 )
        stop( "error: more than one timepoint at that point" )
      # take hazard value at that time point
      result$hazard[i] = coxphData$hazard[coxTime]
    }
    else {

      ## If first timepoint is exit value, then cumHaz is 0
      ## If previous hazardvalue in results is smaller than zero, then set it to 0.
      ## This shouldn't happen.

      if( ( i == 1 ) || (result$hazard[i - 1] < 0 )){result$hazard[i] = 0}


      else  ## take previous hazard value
      {result$hazard[i] = result$hazard[i - 1]}

    }
  }

  cat( "Result: ", nrow(result), " Added from exit: ",
       nrow(result) - length(coxphData$time), "\n" )

  invisible(result)
}
```

```r
# We sample 100 random data sets with 100 randomly generated seeds
set.seed(1)
n = 100 # number of random data sets
seedlist <- sample(1:100000, n, replace = FALSE)
```

Sampling the Cohorts

CC Case Cohort Sampling

```
# Fraction of the sub-cohort (Case Cohort)
sampling_weight <- 0.15
# In the weighted anaylsis cases out side the sub-cohort enter the cohort just
# shortly before they become infected.
# That is why their entry time in the cohort is calculated by substracting the
# time of the event minus a small fraction. Here we use 0.0001 days (8.64 seconds).
# The same applies for cases occurring inside of the sub-cohort.
smallFraction <- 0.0001
```

For the case cohort design a sub-sample of 15 % is sampled randomly.
In the weighted analysis cases from outside the sub-sample are added to the cohort shortly before the event occurs (i. e. the name case-cohort study).
Weights are defined as the inverted probability of being sampled in the case cohort (Inverse-probability weighting, IPW).
Within sub-cohort: 1/sampling fraction of the random sample i.e. 1/0.15. Weighting changes to 1 at the time of infection as all infected patients are included.
Outside sub-cohort: Enters the analysis at time of infection, weighted 1.
In the weighted analysis cases outside the sub-cohort enter the cohort just before infection. Therefore, a very small value, here 0.000000001157407 seconds, is subtracted of the time of the event to calculate the entry time, so these patients don't contribute to the analysis before turning out to be a case. Similar applies for cases from inside the sub-cohort.They are split into two time spans separated by the time just before getting infected.[15]

```
# create directory in working directory where the sampled datasets are saved.
dir.create("data_n_samples")

for (seedi in 1:length(seedlist)){
  set.seed(seedlist[seedi])

my.data.CC <- RWdatac

# Select subcohortmembers

my.data.CC$subcohort <- rbinom(nrow(my.data.CC), 1, sampling_weight)

# creating a describing variable "type" to label cases and non cases
# in and outside of the sub-cohort.
my.data.CC$type <- NA

my.data.CC[my.data.CC$subcohort == 1 &
             my.data.CC$to == 2,]$type <- "non case subcohort"
# my.data.CC[my.data.CC$subcohort == 1 &
#             my.data.CC$to == "cens",]$type<-"non case subcohort"
my.data.CC[my.data.CC$subcohort == 0 &
             my.data.CC$to == 2,]$type <- "non case non subcohort"
# my.data.CC[my.data.CC$subcohort == 0 &
#             my.data.CC$to == "cens",]$type<-"non case non subcohort"
my.data.CC[my.data.CC$subcohort == 1 &
             my.data.CC$to == 1,]$type <- "case subcohort"
```

```
my.data.CC[my.data.CC$subcohort == 0 &
             my.data.CC$to == 1,]$type <- "case non subcohort"

case.cohort.pop <- my.data.CC

##### Calculate weights:#####

# Non cases non sub-cohort get a weight of zero
case.cohort.pop$weight <- 0

# Non cases inside the sub-cohort get the weight of 1/sampling fraction
# of the random sample i.e. 1/0.15.
case.cohort.pop[case.cohort.pop$subcohort == 1,]$weight <- 1/sampling_weight

# Cases sub-cohort change their weight from 1/sampling fraction to 1.
# This happens very shortly before the time of infection.
# Cases inside of the sub-cohort are marked with "cens2"
case.cohort.pop[case.cohort.pop$subcohort == 1 &
                  case.cohort.pop$to == 1,]$exit <-
  case.cohort.pop[case.cohort.pop$subcohort == 1 &
                    case.cohort.pop$to == 1,]$exit - smallFraction
case.cohort.pop[case.cohort.pop$subcohort == 1 &
                  case.cohort.pop$to == 1,]$to <- "cens2"

cases.in.subcohort <- my.data.CC[my.data.CC$subcohort == 1 &
                                   my.data.CC$type == "case subcohort",]
cases.in.subcohort$weight <- 1
cases.in.subcohort$entry <- cases.in.subcohort$exit - smallFraction

# cases non sub-cohort are assigned a weight of 1 at the time
# shortly before the time of infection.

cases.no.subcohort <- case.cohort.pop[case.cohort.pop$subcohort == 0 &
                                        case.cohort.pop$to == 1,]
cases.no.subcohort$weight <- 1
cases.no.subcohort$entry <- cases.no.subcohort$exit - smallFraction

# In the case.cohort dataset patients inside the cohort who get infected
# contribute as control and as case. That is why cases occurring inside the
# cohort are represented by two rows in the dataset.
case.cohort <- rbind(case.cohort.pop, cases.in.subcohort, cases.no.subcohort)
case.cohort <- case.cohort[case.cohort$weight != 0,]

case.cohort <- case.cohort[order(case.cohort$id),]

# For logistic regression we create a table with only one row per patient
# i. e. the doubled rows for cases inside the sub-cohort marked with "cens2"
# are excluded and the patients with an infection are weighted with 1.
case.cohort.glm <- case.cohort[case.cohort$to != "cens2",]
case.cohort.glm[case.cohort.glm$to == 1,]$weight <- 1

# shorter writing for following analysis
CC.glm <- case.cohort.glm
```

```r
CC <- case.cohort

##### NCC Nested Case Control Sampling #####

# In our setting we sample four controls for every case with the same length of stay.
# We break ties for event times, as otherwise to many patients have exactly
# the same value as we only count in days not in hours our minutes.
# By this 'trick' is is also possible to look at different outcomes.

my.data.NCC <- RWdatac

set.seed(seedlist[seedi])
# Break ties for event times. Otherwise to many exactly the same values.

my.data.NCC$exit_noise <- jitter(my.data.NCC$exit, amount = .001)

# Create sample indicator vector. Everyone who become infected is assigned a 2
my.data.NCC$samplestat1 <-0
my.data.NCC$samplestat1 <- ifelse(my.data.NCC$to == 1, 2, 0)

# Binary variable for event 1 (infection).
# Everyone who become infected is assigned a 1
my.data.NCC$event1 <- ifelse(my.data.NCC$to == 1, 1, 0)

# Incidence Density Sampling - generating a nested case-control study.
# Sampling 4 controls for each case

my_ncc <- ccwc(entry = 0,
               exit = exit_noise,
               fail = to == 1,
               origin = 0,
               controls = 4,
               data = my.data.NCC,
               include = list(id, from, to, exit, event1, CHARLSON_FIRST,
                              samplestat1, day_in_ICU))

# Adjusts sample indicator vector (samplestat) for use in {multipleNCC} package.
# Sampled controls are given "1."
Sample_Data <- CreateSamplestat(my_ncc, my.data.NCC, "samplestat1", "id", "Set")

# Logistic Regression Weights
Sample_Data$GLM_weights1 <- GLMprob(Sample_Data[["exit"]],
                                    Sample_Data[["samplestat1"]])

Sample_Data$weight <- 1/Sample_Data$GLM_weights1

# remove non-cases, non-sampled individuals
NCC <- subset(Sample_Data, Sample_Data[,"samplestat1"] != 0)

##### PP Point prevalence Sampling #####

# In the data we do not have dates, therefore we artificially create a PPS sample
# by randomly # picking a number between 0 and 137 for every patient.
```

```r
# This is the patient's PPS day. If the PPS day is smaller than the
# patient's days in the intensive car unit (ICU), they are sampled.

set.seed(seedlist[seedi])
my.data.pp<-RWdatac
# 99 Quartile of day_in_ICU= 136.4
my.data.pp$pps.date <- runif(n = nrow(my.data.pp), min = 0, max = 137)

RWdata.pps <- my.data.pp[my.data.pp$day_in_ICU > my.data.pp$pps.date, ]

# To correct the length-biased sampling we weight patients
# inversely to their days in ICU.

RWdata.pps$weight <- 1/RWdata.pps$day_in_ICU
RWdata.pps <- subset(RWdata.pps, weight > 0)

# create dataset with duplicates of observations equivalent to
# length-biased weighting

RWdata.pps$weight.freq <- RWdata.pps$weight
RWdata.pps$weight.freq <- RWdata.pps$weight.freq * 1/
                      min(RWdata.pps$weight.freq[RWdata.pps$weight.freq > 0]) * 100
RWdata.pps$weight.freq <- round(RWdata.pps$weight.freq)

res <- apply(RWdata.pps, 1, function(z) {
  rep(z, times = z["weight.freq"]) })

RWdata.pps.freq <- matrix(unlist(res), ncol = ncol(RWdata.pps), byrow = TRUE)
RWdata.pps.freq <- data.frame(RWdata.pps.freq)

for (i in 1:ncol(RWdata.pps.freq)) {
    RWdata.pps.freq[, i] <- as.numeric(RWdata.pps.freq[, i])
}

colnames(RWdata.pps.freq) <- names(RWdata.pps)
PP <- RWdata.pps
PP$event1 <- ifelse(PP$to == 1, 1, 0)

##### Save generated Sampling sets #####

save(RWdata, RWdatac, CC,CC.glm, my.data.CC, my_ncc, NCC, PP, RWdata.pps, seedi,
     CreateSamplestat, mixin12, seedlist, RWdata.pps.freq,
     file = paste0("data_n_samples/cohortsamples_nocens", seedi,".RData"))
# files are called with the appendix "_nocens" as no censoring was applied.

}
```

Population Description

```r
summary_los_full <- tibble()
summary_los_CC <- tibble()
summary_los_NCC <- tibble()
```

```r
summary_los_PP <- tibble()

fulldesc <- tibble()
ccdesc <- tibble()
nccdesc <- tibble()
nccdesc_c <- tibble()
ppdesc_c <- tibble()

for (seedi in 1:n){
# for (seedi in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedi, ".RData"))
  # Full cohort
  summary_los_full <- rbind(summary_los_full, summary(RWdatac$day_in_ICU))
  # CC
  summary_los_CC <- rbind(summary_los_CC, summary(CC.glm$day_in_ICU))
  # NCC
  summary_los_NCC <- rbind(summary_los_NCC, summary(NCC$day_in_ICU))
  # PP
  summary_los_PP <- rbind(summary_los_PP, summary(RWdata.pps$day_in_ICU))

  fulldesc <- rbind(fulldesc, table(RWdatac$to))
  ccdesc <- rbind(ccdesc, table(CC.glm$to))
  nccdesc <- rbind(nccdesc,table(NCC$to))
  ppdesc_c <- rbind(ppdesc_c, c(table(RWdata.pps$to)))

}

summary_los <- list(summary_los_full, summary_los_CC, summary_los_NCC, summary_los_PP)

summary_los <- lapply(summary_los, function(x){
  x <- describe(x)
  x1 <- cbind(x$mean, x$sd)
  rownames(x1) <- c("Min.", "1.Q", "Median", "Mean", "3.Q", "Max.")
  colnames(x1) <- c("Mean", "SD")
  x1 <- x1
})
names(summary_los) <- c("Full Cohort", "CC", "NCC", "PP")
summary_los <- as.data.frame(summary_los)

summary_los <- summary_los[,-2]


desc <- list(fulldesc, ccdesc, nccdesc, ppdesc_c)
desc <- lapply(desc, function(x){
  t <- cbind(x, NA, rowSums(x))
  t <- describe(t)
  t1 <- cbind(t$mean, t$sd)
  colnames(t1) <- c("Mean", "SD")
  rownames(t1) <- c("cases", "controls", "cens", "total")
  t1 <- t1
})

names(desc) <- c("Full Cohort", "CC", "NCC", "PP")
```

```r
desc <- as.data.frame(desc)
desc <- desc[,-2]

# Table

desct <- as_hux(desc, add_rownames = getOption("huxtable.add_rownames", TRUE))%>%
  theme_article()%>%
  set_align(everywhere, 1, "left")%>%
  set_align(everywhere, -1, "right")%>%
  set_number_format(everywhere, -c(1, 2), 2)%>%
  set_number_format(everywhere, 2, 0)

popdesc <- rbind(desc,summary_los)
popdesc <- as_hux(popdesc, add_rownames = getOption("huxtable.add_rownames", TRUE))%>%
  set_align(everywhere, 2:5, "right") %>%
  set_align(everywhere, 1, "left")%>%
  insert_row("Length of stay in the ICU in days", rep("", 7), after = 5)%>%
  merge_cells( 6, 1:5)%>%
  set_header_rows(c(1, 6), TRUE)%>%
  theme_article()%>%
  set_number_format(everywhere, -1, 2)

popdesc[1,2] <- "Full.Cohort"
popdesc[1,1] <- ""

popdesc <- insert_row(ht = popdesc, "", "Full cohort", "Case cohort", "",
                      "Nested case-control","", "Point prevalence", "", after = 0)
popdesc <- merge_cells(ht = popdesc, 1, 3:4)%>%
  merge_cells(1, 5:6)%>%
  merge_cells(1, 7:8)%>%
  insert_row("", "", rep(c("Mean", "SD"), 3), after = 1)%>%
  set_header_rows(2, TRUE)
popdesc <- popdesc[-3,]%>%
  theme_article()%>%
  set_align(2, everywhere, "centre" )

popdesc
```

```r
# percentage of cases in the samples in percent

desc_p <- list(fulldesc, ccdesc, nccdesc, ppdesc_c)
desc_p <- data.frame(sapply(desc_p, function(x){
  names(x) <- c("cases", "controls")
  x <- x %>% mutate(p_cases = (cases/(controls + cases) * 100))
  # x <- x %>% mutate(ratio_cases = cases/(total-cases))
  x <- x$p_cases
  x1 <- cbind(mean = mean(x), sd = sd(x))

}))

names(desc_p) <- c("Full Cohort", "CC", "NCC", "PP")
rownames(desc_p) <- c("mean", "sd")
desc_p
```

CC Population Description

```
CCtype <- tibble()
for (seedi in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedi, ".RData"))
  CCtype <- rbind(CCtype, table(CC.glm$type))
}
  CCtype <- colMeans(CCtype)
  names(CCtype) <- names(table(CC.glm$type))
# CCtype
```

The mean number of cases occurring inside the sub-cohort was 19.04. 109.96 occurred outside the sub-cohort.

PP Population Description

```
# prevalent and incident cases

PPtype <- tibble()
for (seedi in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedi, ".RData"))
  prevalent <- sum(PP$to == 1 & -PP$pps.date + PP$exit < 0)
  incident <- sum(PP$to == 1 & -PP$pps.date + PP$exit > 0)
  type <- cbind(prevalent, incident)
  PPtype <- rbind(PPtype, type)
}
PPtype <- colMeans(PPtype)
# PPtype
```

In the Point prevalence design there were 25.34 (mean) prevalent and 21.19 (mean) incident cases.

Analysis

Cox Regression

Cox Regression - Event 1 (Blood Stream Infection)

```
list_cox.uw.e1 <- list(Full = list(), CC = list(), NCC = list(), PP = list())

for (seedx in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedx, ".RData"))

datasets.u <- list(RWdatac, CC.glm, NCC, PP)
# CC: For the unweighted analysis we use the dataset with one row per patient.
# In the weighted analysis we use the dataset where several rows per patient are possible.

# NCC we need to stratify for the matched pairs (Set) in the analysis
coxmodel.ncc.event1 <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST +
```

```
                              strata(Set), NCC, model = TRUE)

# apply coxph model to all datasets
coxmodels.event1 <- lapply(datasets.u[-3],
                           function(data){
                             coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST,
                                   data)
                             })

coxmodels.event1[[4]] <- coxmodel.ncc.event1
coxmodels.event1<-coxmodels.event1[c(1, 2, 4, 3)]

list_cox.uw.e1[[1]][[seedx]] <- coxmodels.event1[[1]]
list_cox.uw.e1[[2]][[seedx]] <- coxmodels.event1[[2]]
list_cox.uw.e1[[3]][[seedx]] <- coxmodels.event1[[3]]
list_cox.uw.e1[[4]][[seedx]] <- coxmodels.event1[[4]]

}
```

Full Cohort and unweighted Cox Regression

```


# apply weighted coxph model with robust variance to all datasets
# weight NCC = 1/GLM_weights1
# weighted CC.glm dataset  for CC
# because of left truncation in the case cohort design, the entry time
# is needed in the coxph. For the others study designs the entry time is 0.

# coxmodel for CC and PP

#### change back datasets.w[-2]############
list_cox.w.e1 <- list(CC = list(), NCC = list(), PP = list())
for (seedx in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedx, ".RData"))

  datasets.w <- list(CC, NCC, PP)
  coxmodels.event1.w <-
    lapply(datasets.w,
      function(data) {
        coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST + cluster(id),
        data, weights=weight)
        })
  list_cox.w.e1[[1]][[seedx]] <- coxmodels.event1.w[[1]]
  list_cox.w.e1[[2]][[seedx]] <- coxmodels.event1.w[[2]]
  list_cox.w.e1[[3]][[seedx]] <- coxmodels.event1.w[[3]]

}
```

Weighted Cox regression

Cox Regression - Event 2 (Discharge)

```
list_cox.uw.e2 <- list(Full = list(), CC = list(), NCC = list(), PP = list())
for (seedx in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedx, ".RData"))

  datasets.u <- list(RWdatac, CC.glm, NCC, PP)

  # apply coxph model to all datasets
  coxmodels.event2 <-
    lapply(datasets.u,
       function(data){
         coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST, data)})

  list_cox.uw.e2[[1]][[seedx]] <- coxmodels.event2[[1]]
  list_cox.uw.e2[[2]][[seedx]] <- coxmodels.event2[[2]]
  list_cox.uw.e2[[3]][[seedx]] <- coxmodels.event2[[3]]
  list_cox.uw.e2[[4]][[seedx]] <- coxmodels.event2[[4]]
}
```

Full Cohort and unweighted Cox Regression

Weighted Cox Regression For CC we do not calculate robust variances as only event 1 (cases) could contribute two times to the analysis. CC includes the entry variable as data are left censored. NCC and PP need robust variances.

```
list_cox.w.e2 <- list(CC = list(), NCC = list(), PP = list())
for (seedx in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedx, ".RData"))

  # In the CC design only the sub-cohort is used for the analysis of the discharge hazard
  coxmodel.cc.event2.w <- coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST,
                                data = my.data.CC, subset = subcohort == 1, model = TRUE)

  datasets.w.e2 <- list(NCC, PP)
  coxmodels.event2.w <- lapply(datasets.w.e2,
                               function(data){
                                 coxph(Surv(exit, to == 2) ~ CHARLSON_FIRST + cluster(id),
                                       data, weights = weight)
                                 })
  coxmodels.event2.w <- list(coxmodel.cc.event2.w, coxmodels.event2.w[[1]],
                             coxmodels.event2.w[[2]])

  list_cox.w.e2[[1]][[seedx]] <- coxmodels.event2.w[[1]]
  list_cox.w.e2[[2]][[seedx]] <- coxmodels.event2.w[[2]]
  list_cox.w.e2[[3]][[seedx]] <- coxmodels.event2.w[[3]]

}
```

```
listcox <- list(list_cox.uw.e1, list_cox.uw.e2, list_cox.w.e1, list_cox.w.e2)
```

```
titleshort <- list("Full", "CC", "NCC", "PP")
namelist <- list(event1_unweighted = titleshort, event2_unweighted = titleshort,
                 event1_weighted = titleshort[2:4], event2_weighted = titleshort[2:4])

# for the weighted analysis we extract the covariance matrix( robust se)
# and beta coefficients and pool those

vcovlist <- list()
betalist <- list()
sumcoxlist <- list()

for (i in 1:length(listcox)){
  vcovlist[[i]] <- list()
  betalist[[i]] <- list()
  sumcoxlist[[i]] <- list()

  for (no_settings in 1:length(listcox[[i]])){

      betalist[[i]][[no_settings]] <- lapply(listcox[[i]][[no_settings]], coef)
      vcovlist[[i]][[no_settings]] <- lapply(listcox[[i]][[no_settings]], vcov)
      sumcoxlist[[i]][[no_settings]] <- list()
      sumcoxlist[[i]][[no_settings]] <- pool_mi(qhat = betalist[[i]][[no_settings]],
                                                u = vcovlist[[i]][[no_settings]])
      sumcoxlist[[i]][[no_settings]] <- summary(sumcoxlist[[i]][[no_settings]],
                                                exp = FALSE)
  }
}

names(sumcoxlist) <- names(namelist)
for (i in 1:length(sumcoxlist)){
  names(sumcoxlist[[i]]) <- namelist[[i]]
}

# sumcoxlist

names(listcox) <- names(namelist)
for (i in 1:length(listcox)){
  names(listcox[[i]]) <- namelist[[i]]
}
```

```
coxtable <- list()
for (i in 1:length(sumcoxlist)){
  coxtable[[i]] <- tibble(NA)
  for (no_setting in 1:length(sumcoxlist[[i]])){
    t <- t(sumcoxlist[[i]][[no_setting]][1:2])
    coxtable[[i]] <- cbind(coxtable[[i]], t)
  }
  coxtable[[i]] <- coxtable[[i]][-1]
  names(coxtable[[i]]) <- namelist[[i]]
}
names(coxtable) <- names(namelist)
```

```
coxevent1 <- cbind(coxtable[[1]][1:2], coxtable[[3]][1],coxtable[[1]][3],
                   coxtable[[3]][2], coxtable[[1]][4], coxtable[[3]][3])
coxevent2 <- cbind(coxtable[[2]][1:2], coxtable[[4]][1],coxtable[[2]][3],
                   coxtable[[4]][2], coxtable[[2]][4], coxtable[[4]][3])

coxlist_r <- list(coxevent1, coxevent2)

titlelist <- list("Hazard ratio - Infection",
                  "Hazard ratio - Discharge/Death without Infection")
plotlist <- list()
yaxis <- list(c(0.93, 1.2), c(0.9, 1.1))


labelw <- as_labeller(c("w" = "weighted",
                        "uw" = "unweighted"))

coxlist_p <- list()
for (i in 1:length(coxlist_r)){

  tcox <- coxlist_r[[i]][1:2,]
  tcox <- data.frame(lapply(tcox, as.numeric))
  tcox <- as.data.frame(t(tcox))
  colnames(tcox) <- c("estimate", "se")
  tcox$dgroup <- c("F", "CC", "CC", "NCC", "NCC", "PP", "PP")
  tcox <- mutate(tcox, row = row_number())
  tcox$weighting <- c("uw", "uw", "w", "uw", "w", "uw", "w")
  estimfull <- unlist(tcox[1,1])
  errorfull <- unlist(tcox[1,2])
  coxlist_p[[i]]<- tcox

 plot <- ggplot(
    tcox, aes(x = reorder(dgroup, row), y = exp(estimate),
              colour = weighting, group = weighting)) +
   geom_hline(yintercept = exp(estimfull),  colour='grey25')+
   geom_point(size = 4, position = position_dodge(0.6)) +
    annotate("rect", xmin = 0, xmax = Inf,
             ymin = exp(estimfull - errorfull * 1.96),
             ymax = exp(estimfull + errorfull * 1.96),
             alpha = 0.5, fill = "lightgrey")+
    geom_point(size = 4, position = position_dodge(0.6)) +
    scale_color_manual(values = c("grey60", "black"), name = "weighting",
                       labels = c("unweighted", "weighted")) +
    geom_errorbar(aes(ymax = exp(estimate + se * 1.96),
                      ymin = exp(estimate - se * 1.96)),
                  position = position_dodge(0.6)) +
    scale_y_continuous(labels = scales::number_format(accuracy = 0.01)) +
    theme_classic() +
    ggtitle(titlelist[[i]]) +
    labs(x = "Study designs", y = "Hazard ratio estimate")

  plotlist[[i]] <- plot
}
```

```
do.call("grid.arrange", c(plotlist, nrow = 1))
```

```
# calculate confidence intervals and exp() the results
coxlistp <- lapply(coxlist_p, function(x){
  x$CI_95hi <- exp(x$estimate + x$se * 1.96)
  x$CI_95lo <- exp(x$estimate - x$se * 1.96)
  x$exp_e <- exp(x$estimate)
  x$p <- x$exp_e/x[1,1]
  x <- x
})
coxlistp
round(coxlistp[[2]]$exp_e, 3)
```

Pooling Cox Results

Logistic Regression

```
list_log_uw.e1 <- list(Full = list(), CC = list(), NCC = list(), PP = list())

for (seedi in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedi, ".RData"))

  datasets.u <- list(RWdatac, CC.glm, NCC, PP)

  glm.models.event1 <-
    lapply(datasets.u,
           function(data){
             glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"), data)
             })

list_log_uw.e1[[1]][[seedi]] <- glm.models.event1[[1]]
list_log_uw.e1[[2]][[seedi]] <- glm.models.event1[[2]]
list_log_uw.e1[[3]][[seedi]] <- glm.models.event1[[3]]
list_log_uw.e1[[4]][[seedi]] <- glm.models.event1[[4]]

}
```

Full Cohort and unweighted Logistic Regression

```
list_log_w.e1 <- list(CC = list(), NCC = list(), PP = list())
for (seedi in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedi, ".RData"))

  # apply model with weights
  datasets.glm.w <- list(CC.glm, NCC, PP)
```

```
  glm.models.event1.w <-
    lapply(datasets.glm.w,
           function(data){
             lr.w <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                         data, weights = weight)
             summary.lr.w <- summary(lr.w)
             # save the robust covariance matrix
             cov.mat <- vcovHC(eval(summary.lr.w$call))
             lr.w$qr$qr[1:lr.w$rank, 1:lr.w$rank] <- chol(solve(cov.mat))
             data <- lr.w
             })
list_log_w.e1[[1]][[seedi]] <- glm.models.event1.w[[1]]
list_log_w.e1[[2]][[seedi]] <- glm.models.event1.w[[2]]
list_log_w.e1[[3]][[seedi]] <- glm.models.event1.w[[3]]
}
```

Weighted Logistic Regression

```
# logistic regression pool results
listlog <- list(list_log_uw.e1, list_log_w.e1)

sumloglist <- list()
listloglist <- list()
for (i in 1: length(listlog)){
  sumloglist[[i]] <- list()
  listloglist[[i]] <- list()
  for(no_settings in 1:length(listlog[[i]])){
  listloglist[[i]][[no_settings]] <- pool(listlog[[i]][[no_settings]])
  sumloglist[[i]][[no_settings]] <- summary(listloglist[[i]][[no_settings]], exp = FALSE)
  }
}

names(sumloglist) <- names(namelist[c(1, 3)])
for (i in 1:length(sumloglist)){
  names(sumloglist[[i]]) <- namelist[c(1, 3)][[i]]
}

# sumloglist

logtable <- list()
for (i in 1:length(sumloglist)){
  logtable[[i]] <- tibble(NA)
  for (no_setting in 1:length(sumloglist[[i]])){
    t <- t(sumloglist[[i]][[no_setting]])
    logtable[[i]] <- cbind(logtable[[i]], t)

  }
  logtable[[i]] <- logtable[[i]][, -1]
  names(logtable[[i]]) <- c(rep(namelist[[i + 1]], each = 2))
}
```

```
logtablex <- as_hux(logtable)%>%
  theme_basic()

logtablex
```

Pool log Results

```
logevent1or <- cbind(logtable[[1]][c(2, 3), c(seq(2, 8, 2))],
                     logtable[[2]][c(2, 3), c(seq(2, 6, 2))])
logevent1or <-logevent1or[, c(1, 2, 5, 3, 6, 4, 7)]

  tlog <- logevent1or
  tlog <- data.frame(lapply(tlog, as.numeric))
  tlog <- as.data.frame(t(tlog))
  colnames(tlog) <- c("estimate", "se")
  tlog$dgroup <- c("F", "CC", "CC", "NCC", "NCC", "PP", "PP")
  tlog <- mutate(tlog, row = row_number())
  tlog$weighting <- c("uw", "uw", "w", "uw", "w", "uw", "w")
  estimfull <- unlist(tlog[1, 1])
  errorfull <- unlist(tlog[1, 2])

 plotor <- ggplot(
    tlog, aes(x = reorder(dgroup, row), y = exp(estimate),
              colour = weighting, group = weighting)) +
   geom_hline(yintercept = exp(estimfull),  colour = 'grey25')+
    geom_point(size = 4, position = position_dodge(0.6)) +
    annotate("rect", xmin = 0, xmax = Inf,
             ymin = exp(estimfull - errorfull * 1.96),
             ymax = exp(estimfull + errorfull * 1.96),
             alpha = 0.5, fill = "lightgrey") +
    geom_point(size = 4, position = position_dodge(0.6)) +
    scale_color_manual(values = c("grey60", "black"), name = "weighting",
                       labels = c("unweighted", "weighted")) +
    geom_errorbar(aes(ymax = exp(estimate + se * 1.96),
                      ymin = exp(estimate - se * 1.96)),
                  position = position_dodge(0.6)) +
    theme_classic() +
    ggtitle("Odds ratio for Infection") +
    labs(x = "Study designs", y = "Odds ratio estimate")

plotor
```

```
logevent1 <- cbind(logtable[[1]][c(2, 3), c(seq(1, 8, 2))],
                   logtable[[2]][c(2, 3), c(seq(1, 6, 2))])
logevent1 <- logevent1[, c(1, 2, 5, 3, 6, 4, 7)]

  tlog <- logevent1
  tlog <- data.frame(lapply(tlog, as.numeric))
```

```
  tlog <- as.data.frame(t(tlog))
  colnames(tlog) <- c("estimate", "se")
  tlog$dgroup <- c("F", "CC", "CC", "NCC", "NCC", "PP", "PP")
  tlog <- mutate(tlog, row = row_number())
  tlog$weighting <- c("uw", "uw", "w", "uw", "w", "uw", "w")
  estimfull <- unlist(tlog[1, 1])
  errorfull <- unlist(tlog[1, 2])

 plotbo <- ggplot(
    tlog, aes(x = reorder(dgroup, row), y = exp(estimate),
              colour = weighting, group = weighting)) +
   geom_hline(yintercept = exp(estimfull),  colour = 'grey25')+
   geom_point(size = 4, position = position_dodge(0.6)) +
    annotate("rect", xmin = 0, xmax = Inf,
             ymin = exp(estimfull - errorfull * 1.96),
             ymax = exp(estimfull + errorfull * 1.96),
             alpha = 0.5, fill = "lightgrey")+
    geom_point(size = 4, position = position_dodge(0.6)) +
    scale_color_manual(values = c("grey60", "black"),
                       name = "weighting",
                       labels = c("unweighted", "weighted")) +
    geom_errorbar(aes(ymax = exp(estimate + se * 1.96),
                      ymin = exp(estimate - se * 1.96)),
                  position = position_dodge(0.6)) +
   scale_y_continuous(labels = scales::number_format(accuracy = 0.01), trans = "log2",
                      breaks = c(round(seq(min(exp(tlog$estimate - tlog$se * 1.96)),
                                           max(exp(tlog$estimate + tlog$se * 1.96)),
                                           by = 0.05), 2)))+
    theme_classic() +
    ggtitle("Baseline odds for Infection") +
    labs(x = "Study designs", y = "Baseline odds estimate")

plotbo
```

Plot OR

Results

```
par(mfrow = c(1, 1))
empty <- ggplot() + theme_void()
legend_plot <- cowplot::get_legend(plotor)
#
# tiff("plots/without_censoring/fig5.tiff", units = "in",
#      width = 10, height = 8, res = 1200)
plot_hr_or_bo<-grid.arrange(arrangeGrob(
                            plotlist[[1]] + theme(legend.position = "none"),
                            plotlist[[2]] + theme(legend.position = "none"),
                            legend_plot,
                            plotor + theme(legend.position = "none"),
                            plotbo + theme(legend.position = "none"),
                            empty, ncol = 3, widths = c(2.5, 2.5, 1)
```

```
                                ))
print(plot_hr_or_bo)
# dev.off()
```

Residuals - Cox Assumptions Testing

Numbers of datasets (of 100) with a significant Schoenfield test result (p<0.05).

```
# Testing the cox model assumption
## Testing proportional hazard assumption
titleshort <- list("Full", "CC", "NCC", "PP")
namelist <- list(event1_unweighted = titleshort,
                 event2_unweighted = titleshort,
                 event1_weighted = titleshort[2:4],
                 event2_weighted = titleshort[2:4])
```

```
listschoenfeld <- list()
tschoenfeld <- list()
library(tibble)

for (i in 1:length(listcox)){
    listschoenfeld[[i]] <- list()
    tschoenfeld[[i]] <- list()
    for (no_settings in 1:length(listcox[[i]])){
    listschoenfeld[[i]][[no_settings]] <- list()
    tschoenfeld[[i]][[no_settings]] <- tibble()

    for (seeds in 1:length(listcox[[i]][[no_settings]])){
      schoenf <- cox.zph(listcox[[i]][[no_settings]][[seeds]])$table

      listschoenfeld[[i]][[no_settings]][[seeds]] <- schoenf
      schoenf <- schoenf[1, ]

      tschoenfeld[[i]][[no_settings]] <- rbind(tschoenfeld[[i]][[no_settings]], schoenf)
      names(tschoenfeld[[i]][[no_settings]]) <- c("chisq", "df", "p")
    }
  }
}
```

```
coxassumption<-data.frame()
for (i in 1:length(tschoenfeld)){
  for (no_settings in 1: length(tschoenfeld[[i]])){
    coxassumption[i, no_settings] <- with(tschoenfeld[[i]][[no_settings]],
                                          sum(p < 0.05))
  }
}
rownames(coxassumption) <- c("event 1 uw", "event 2 uw", "event 1 w", "event 2 w")


coxassumption[3:4, 2:4] <- coxassumption[3:4, 1:3]
coxassumption[3:4, 1] <- NA
colnames(coxassumption) <- c("Full", "CC", "NCC", "PP")
```

```
coxassumptionx <- as_hux(coxassumption, add_colnames = getOption("huxtable.add_colnames",
                                                                 TRUE),
                         add_rownames = getOption("huxtable.add_rownames", TRUE))%>%
  theme_article()%>%
  set_align(everywhere, 1, "left")
coxassumptionx[1,1] <- ""
coxassumptionx
```

Multi-state Modelling - Calculating Cumulative Hazards and Transition Probabilites (alias Cumulative Incidence Function)

We will demonstrate the analysis with the full cohort first and will than perform it again jointly for all three study designs.

As a first step, we define which transitions are possible in a transition matrix (traCRetm). We use this matrix to compute the Nelson-Aalen estimator of the cumulative transition hazards in a multi-state model.[6] For each possible transition, it computes an estimate of the cumulative hazard (see: mvna{mvna}).
In our case we have the transition from admission (state 0) to bloodstream infection (state 1) or from admission to discharge (state 2).
The hazard function for both events (infection and discharge) is estimated non-parametrically (no assumption about the distribution of failure times) with the bshazard{bshazard} function using a survival object. Left truncation is considered by including the entry time (variable "entry").

Lastly the empirical transition matrix (Aalen-Johansen estimator) of the transition probability matrix of the multi-state model is computed using the etm {etm} function.

```
# matrix describing the possible transitions between the states
traCRetm <- matrix(FALSE, 3, 3)
traCRetm[1,2] <- TRUE
traCRetm[1,3] <- TRUE

# transition names
traCR <- matrix(NA, 3, 3)
traCR[1,2] <- 1
traCR[1,3] <- 2

# calculating the cumulative hazards for these transitions from the data.
# If you have censoring , choose "cens" as the last argument.
# (Nelson-Aalen estimator)
# The dataset used here should contain one row per transition
# (possibly several rows per patient).
# Specifying an entry and exit time in the data enables to account for left-truncation.
cumHazRUSdata <- mvna(RWdatac, c(0:2), traCRetm, "cens")

# estimating the hazard function for state 1 (infection) and 2 (discharge)
# non-parametrically (no assumption about the distribution of failure times)
# smoothing of the hazard function
bs2 <- bshazard(Surv(entry, exit, to == 2) ~ 1, data = RWdatac)
bs1 <- bshazard(Surv(entry, exit, to == 1) ~ 1, data = RWdatac)

# computation of the empirical transition matrix (Aalen-Johansen estimator)
```

```
# of the transition probability matrix of the multi-state model.
# It considers competing risks. Account for censoring if applicable for your data.
TransProbRUSdata <- etm::etm(RWdatac, as.character(0:2),
                             traCRetm, "cens" , s = 0, covariance = F)
```

Calculation for the Full Cohort

```
# We use the Cox model not adjusted for covariables

coxphlist_uw_e1 <- list(Full = list(), CC = list(), NCC = list(), PP = list())
coxphlist_w_e1 <- list(CC = list(), NCC = list(), PP = list())
coxphlist_uw_e2 <- list(Full = list(), CC = list(), NCC = list(), PP = list())
coxphlist_w_e2 <- list(CC = list(), NCC = list(), PP = list())

for (seedi in 1:length(seedlist)){
  load(paste0("data_n_samples/cohortsamples_nocens", seedi, ".RData"))

  datasets.u <- list(RWdatac, CC.glm, NCC, PP)
  datasets.w <- list(CC, NCC, PP)
  datasets.w.e2 <- list(NCC, PP)

  # unweighted
  coxphlist_uw_e1_temp <- list()
  for (i in 1:length(datasets.u)){
    coxphlist_uw_e1_temp[[i]]<- coxph(Surv(entry, exit, to == 1) ~ 1,
                                      data = datasets.u[[i]] )

  }
  coxphlist_uw_e1[[1]][[seedi]] <- coxphlist_uw_e1_temp[[1]]
  coxphlist_uw_e1[[2]][[seedi]] <- coxphlist_uw_e1_temp[[2]]
  coxphlist_uw_e1[[3]][[seedi]] <- coxphlist_uw_e1_temp[[3]]
  coxphlist_uw_e1[[4]][[seedi]] <- coxphlist_uw_e1_temp[[4]]

  # weighted
  coxphlist_w_e1_temp <-
    lapply(datasets.w,
           function(data){
             coxph(Surv(entry, exit, to == 1) ~ 1, robust = TRUE, id = id,
                   data, weights = weight)
             })

  coxphlist_w_e1[[1]][[seedi]] <- coxphlist_w_e1_temp[[1]]
  coxphlist_w_e1[[2]][[seedi]] <- coxphlist_w_e1_temp[[2]]
  coxphlist_w_e1[[3]][[seedi]] <- coxphlist_w_e1_temp[[3]]

  ## Event 2
  # unweighted
  coxphlist_uw_e2_temp <- list()
  for (i in 1:length(datasets.u)){
    coxphlist_uw_e2_temp[[i]] <- coxph(Surv(entry, exit, to == 2) ~ 1,
                                       data = datasets.u[[i]] )
```

```
  }

  coxphlist_uw_e2[[1]][[seedi]] <- coxphlist_uw_e2_temp[[1]]
  coxphlist_uw_e2[[2]][[seedi]] <- coxphlist_uw_e2_temp[[2]]
  coxphlist_uw_e2[[3]][[seedi]] <- coxphlist_uw_e2_temp[[3]]
  coxphlist_uw_e2[[4]][[seedi]] <- coxphlist_uw_e2_temp[[4]]

  # weighted
  # for CC: no weighting no robust variance as second event is only calculated
  # by sub-cohort only
  coxphlist_w_e2_CC <-
    coxph(Surv(entry, exit, to == 2) ~ 1, data = my.data.CC, subset = subcohort == 1)

  coxphlist_w_e2_temp <-
    lapply(datasets.w.e2,
           function(data){
             coxph(Surv(exit, to == 2) ~ 1, robust = TRUE, id = id,
                   data, weights = weight)
             })

  coxphlist_w_e2_temp <- list(coxphlist_w_e2_CC,
                              coxphlist_w_e2_temp[[1]],
                              coxphlist_w_e2_temp[[2]])

  coxphlist_w_e2[[1]][[seedi]] <- coxphlist_w_e2_temp[[1]]
  coxphlist_w_e2[[2]][[seedi]] <- coxphlist_w_e2_temp[[2]]
  coxphlist_w_e2[[3]][[seedi]] <- coxphlist_w_e2_temp[[3]]
}
```

```
# From the Cox models the predicted survival curve and cumulative hazards are calculated

coxlist <- list(coxphlist_uw_e1, coxphlist_uw_e2, coxphlist_w_e1, coxphlist_w_e2)
cumhazlist <- list()
for (i in  1: length(coxlist)){
  cumhazlist[[i]] <- list()
  for (no_settings in 1:length(coxlist[[i]])){
    cumhazlist[[i]][[no_settings]] <- list()
    for (seed in 1:length(coxlist[[i]][[no_settings]])){
      cumhazlist[[i]][[no_settings]][[seed]] <-
        basehaz(coxlist[[i]][[no_settings]][[seed]], centered = FALSE)
    }
  }
}

# we prepare for the mixin 2 function to create a msfit object

timeslist <- list()

for (seed in 1:length(cumhazlist[[3]][[1]])){
  timeslist[[seed]] <- list()
  for (i in 1:3){
```

```
  timeslist[[seed]][[i]] <- list()
  timeslist[[seed]][[i]]$times <- cumhazlist[[3]][[i]][[seed]]$time
  timeslist[[seed]][[i]]$FinalTime <- c(timeslist[[seed]][[i]]$times,
                                        timeslist[[seed]][[i]]$times)
  timeslist[[seed]][[i]]$colTrans <- c(rep(1, length(timeslist[[seed]][[i]]$times)),
                                       rep(2, length(timeslist[[seed]][[i]]$times)))

timeslist[[seed]][[4]] <- list()
timeslist[[seed]][[4]]$times <- cumhazlist[[1]][[1]][[seed]]$time
timeslist[[seed]][[4]]$FinalTime <- c(timeslist[[seed]][[4]]$times,
                                      timeslist[[seed]][[4]]$times)
timeslist[[seed]][[4]]$colTrans <- c(rep(1, length(timeslist[[seed]][[4]]$times)),
                                     rep(2, length(timeslist[[seed]][[4]]$times)))
  }

timeslist[[seed]] <- timeslist[[seed]][c(4, 1, 2, 3)]
timeslist[[seed]] <- list(timeslist[[seed]], timeslist[[seed]],
                          timeslist[[seed]][2:4], timeslist[[seed]][2:4])
}

# applying the mixin12 function to prepare an msfit object
mixinobjects <- list()

for (i in 1:length(cumhazlist)) {
  mixinobjects[[i]] <- list()
  for (no_settings in 1:length(cumhazlist[[i]])){
    mixinobjects[[i]][[no_settings]] <- list()
    for(seed in 1:length(cumhazlist[[i]][[no_settings]])){
      mixinobjects[[i]][[no_settings]][[seed]] <- list()
      mixinobjects[[i]][[no_settings]][[seed]] <-
        mixin12(cumhazlist[[i]][[no_settings]][[seed]],
                                    unlist(timeslist[[seed]][[i]][[no_settings]]$times))
    }
  }
}


# combine competing risks
## unweighted
jointmixinobjects_uw <- list()
for (no_settings in 1: length(mixinobjects[[1]])){
  jointmixinobjects_uw[[no_settings]] <- list()
  for (seed in 1:length(mixinobjects[[1]][[no_settings]])){
    jointmixinobjects_uw[[no_settings]][[seed]] <- rbind(
      mixinobjects[[1]][[no_settings]][[seed]], mixinobjects[[2]][[no_settings]][[seed]])
  }
}

## weighted
jointmixinobjects_w <- list()
for (no_settings in 1: length(mixinobjects[[3]])){
  jointmixinobjects_w[[no_settings]] <- list()
  for (seed in 1:length(mixinobjects[[3]][[no_settings]])){
```

```
    jointmixinobjects_w[[no_settings]][[seed]] <- rbind(
      mixinobjects[[3]][[no_settings]][[seed]], mixinobjects[[4]][[no_settings]][[seed]])
  }
}

# create msfit objects
## unweighted
msfitobjects_uw <- list()
for (no_settings in 1:length(jointmixinobjects_uw)){
  msfitobjects_uw[[no_settings]] <- list()
  for(seed in 1:length(jointmixinobjects_uw[[no_settings]])){
  msfitobjects_uw[[no_settings]][[seed]] <- list(
    Haz = data.frame(time = timeslist[[seed]][[1]][[no_settings]]$FinalTime,
                     Haz = jointmixinobjects_uw[[no_settings]][[seed]]$hazard,
                     trans = timeslist[[seed]][[1]][[no_settings]]$colTrans),
    trans = traCR)
  class(msfitobjects_uw[[no_settings]][[seed]]) <- "msfit"
  }
}

## weighted
msfitobjects_w <- list()
for (no_settings in 1:length(jointmixinobjects_w)){
  msfitobjects_w[[no_settings]] <- list()
  for(seed in 1:length(jointmixinobjects_w[[no_settings]])){
  msfitobjects_w[[no_settings]][[seed]] <- list(
    Haz = data.frame(time = timeslist[[seed]][[3]][[no_settings]]$FinalTime,
                     Haz = jointmixinobjects_w[[no_settings]][[seed]]$hazard,
                     trans = timeslist[[seed]][[3]][[no_settings]]$colTrans),
    trans = traCR)
  class(msfitobjects_w[[no_settings]][[seed]]) <- "msfit"
  }
}

# From the msfit objects (the estimated cumulative hazards for each transition)
# the transition probabilities in our multi-state model are calculated.
transprob_uw <- list()
for (no_settings in 1:length(msfitobjects_uw)){
  transprob_uw[[no_settings]] <- list()
  for (seed in 1:length(msfitobjects_uw[[no_settings]])){
  transprob_uw[[no_settings]][[seed]] <- probtrans(
    msfitobjects_uw[[no_settings]][[seed]], predt = 0, variance = FALSE)
  }
}


transprob_w <- list()
for (no_settings in 1:length(msfitobjects_w)){
  transprob_w[[no_settings]] <- list()
  for(seed in 1:length(msfitobjects_w[[no_settings]])){
    transprob_w[[no_settings]][[seed]] <- probtrans(
      msfitobjects_w[[no_settings]][[seed]], predt = 0, variance = FALSE)
  }
```

```
}
```

Joint Calculation of Cumulative Hazards and Transition Probabilities

Multi-state Plots

```
lwd <- 3
mylinetypes <- c(1:3, 6)
mycolors <- c("red", rep("black", 3))
a <- 0.1


#### Cumulative Hazard Plots

#, dev = "tikz"}
# pdf(here("plots/Realworlddata_cum_haz.pdf"),
# width = 15, height = 8.27)

# Cumulative hazard: BSI / weighted
# tiff("plots/without_censoring/fig3.tiff", units = "in",
#       width = 10, height = 10, res = 1200)

par(mar = c(2, 3, 1, 1), oma = c(3, 2, 1, 1))
layout (matrix(c(1, 1,
                 2, 3,
                 4, 4,
                 5, 6), 4, 2, byrow = TRUE), heights = c(0.5, 3, 1.5, 3))
plot.new()
text(y = 0.5, x = 0.5, "Infection", cex = 2.5, font = 2)

xlimmax = 60

# plot unweighted cum hazard
plot(1, type = "n",  xlim = c(0, xlimmax), ylim = c(0, 1.7), cex.axis = 2)

for (i in 1:length(cumhazlist[[1]])){
  for (seed in 1:1){

  lines(cumhazlist[[1]][[i]][[seed]]$time, cumhazlist[[1]][[i]][[seed]]$hazard,
        type = "l", lwd = lwd, lty = mylinetypes[[i]], col = alpha(mycolors[[i]], 1))
  }
   lines(cumhazlist[[1]][[1]][[1]]$time, cumhazlist[[1]][[1]][[1]]$hazard,
        lty = 1, lwd = lwd, type = "l", col = "black")
}
legend("topleft", c("Full", "CC", "NCC" , "PP"),
       lty = mylinetypes[c(1:4)], col = "black", lwd = 2, cex = 2)
mtext("Cumulative Hazard", side = 2, line = 3, font = 1, cex = 1.25)
mtext("unweighted", side = 3, line = 1, font = 2, cex = 1.25)
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)

# plot weighted cum hazard
```

```
plot(1, type = "n",
     xlim = c(0, xlimmax), ylim = c(0, 1.7), cex.axis = 2)

for (i in 1:length(cumhazlist[[3]])){
  for(seed in 1:1){

  lines(cumhazlist[[3]][[i]][[seed]]$time, cumhazlist[[3]][[i]][[seed]]$hazard,
        type = "l", lwd = lwd, lty = mylinetypes[[i + 1]], col = alpha("black", 1))
  }
  lines(cumhazlist[[1]][[1]][[1]]$time, cumhazlist[[1]][[1]][[1]]$hazard,
        lty = 1, lwd = lwd, type = "l", col = "black")
}

mtext("weighted", side = 3, line = 1, font = 2, cex = 1.25)
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)

plot.new()
text(0.5, 0.3, "Discharge/Death without Infection", cex = 2.5, font = 2)


# plot unweighted cum hazard
plot(1, type = "n",  xlim = c(0, xlimmax), ylim = c(0, 4), cex.axis = 2)

for (i in 1:length(cumhazlist[[2]])){
  for (seed in 1:1){
  lines(cumhazlist[[2]][[i]][[seed]]$time, cumhazlist[[2]][[i]][[seed]]$hazard,
        type = "l", lwd = lwd, lty = mylinetypes[[i]], col = alpha(mycolors[[i]], 1))
  }
  lines(cumhazlist[[2]][[1]][[1]]$time, cumhazlist[[2]][[1]][[1]]$hazard,
        lty = 1, lwd = lwd, type = "l", col = "black")
}

mtext("Cumulative Hazard", side = 2, line = 3, font = 1, cex = 1.25)
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)
mtext("unweighted", side = 3, line = 1, font = 2, cex = 1.25)

# plot weighted cum hazard
plot(1, type = "n",
     xlim = c(0, xlimmax), ylim = c(0, 4), cex.axis = 2)

for (i in 1:length(cumhazlist[[4]])){
  for (seed in 1:1){
  lines(cumhazlist[[4]][[i]][[seed]]$time, cumhazlist[[4]][[i]][[seed]]$hazard,
        type = "l", lwd = lwd, lty = mylinetypes[[i + 1]], col = alpha("black", 1))
  }
  lines(cumhazlist[[2]][[1]][[1]]$time, cumhazlist[[2]][[1]][[1]]$hazard,
        lty = 1, lwd = lwd, type = "l", col = "black")
}

mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)
mtext("weighted", side = 3, line = 1, font = 2, cex = 1.25)

# dev.off()
```

```r
# prepair dataset for ggplot
cumhazt <- cumhazlist
names(cumhazt) <- names(namelist)
for(i in 1:length(cumhazlist)){
    names(cumhazt[[i]]) <- namelist[[i]]

    for(seed in 1:length(cumhazt[[i]])){
    names(cumhazt[[i]][[seed]]) <- paste0("seed_", 1:length(cumhazt[[i]][[seed]]))
    }
}

for (i in 1:length(cumhazt)){
  for(no_settings in 1:length(cumhazt[[i]])){
    cumhazt[[i]][[no_settings]] <- bind_rows(cumhazt[[i]][[no_settings]],
                                             .id="seed")
  }
 cumhazt[[i]] <- bind_rows(cumhazt[[i]], .id="study_type")
}
cumhazt <- bind_rows(cumhazt, .id="event")

baseline <- subset(cumhazt, event == "event1_unweighted" &
                     study_type == "Full" &
                     seed == "seed_1")
baseline_e1w <- baseline
baseline_e1w$event <- "event1_weighted"

baseline2 <- subset(cumhazt, event == "event2_unweighted" &
                      study_type == "Full" &
                      seed == "seed_1")
baseline_e2w <- baseline2
baseline_e2w$event <- "event2_weighted"

cumhazt <- rbind(cumhazt, baseline_e1w, baseline_e2w)
```

```r
desc3 <- data.frame()
desc2 <- list(fulldesc, ccdesc, nccdesc, ppdesc_c)
for (i in 1:length(desc2)){
  desc2[[i]] <- cbind(desc2[[i]], rowSums(desc2[[i]]))
  names(desc2[[i]]) <- c("1", "2", "total")
  desc3 <- rbind(desc3, desc2[[i]])
}
desc3$study_type <- c(rep(c("Full", "CC", "NCC", "PP"), each = n))
desc3$seed <- paste0("seed_", 1:n )
cumhaztm <- merge(desc3, cumhazt, by = c("seed", "study_type"))



#### Cumulative Incidence function


# , dev = "tikz"}
# pdf(here("plots/Realworlddata_transprob_infection.pdf"),
# width = 15, height = 8.27)
```

```r
# xlimmax = 60

cexl <- 2
cexa <- 2
cexh <- 1.5
cexm <- 3

# tiff("plots/without_censoring/fig4.tiff", units = "in",
#        width = 10, height = 10, res = 1200)

par(mar = c(2, 3, 1, 1), oma = c(3, 2, 1, 1))
layout (matrix(c(1, 1,
                 2, 3,
                 4, 4,
                 5, 6), 4, 2, byrow = TRUE), heights = c(0.5, 3, 1.5, 3))
plot.new()
text(y = 0.5, x = 0.5,"Infection", cex = 2.5, font = 2)

# plot unweighted probtrans
plot(1, type = "n",  xlim = c(0, xlimmax), ylim = c(0, 0.3),
     # main = "unweighted", ylab = "Cumulative incidence function",
     xlab = "Days since admission", cex.axis = cexa, cex.lab = cexl)

for (i in 1:length(transprob_uw)){
   for (seed in 1:1){
  lines(transprob_uw[[i]][[seed]][[1]]$time,
        transprob_uw[[i]][[seed]][[1]]$pstate2, type = "l",
        lwd = lwd,
        lty = mylinetypes[[i]],
        col = alpha(mycolors[[i]], 1))
   }
  lines(transprob_uw[[1]][[1]][[1]]$time,
        transprob_uw[[1]][[1]][[1]]$pstate2,
        lty = 1, lwd = lwd, type = "l", col = "black")
}
# axis(side = 2, at = pretty(c(0,0.3), 3), labels = pretty(c(0, 0.3), 3))
mtext("Cumulative incidence function", side = 2, line = 3, font = 1, cex = 1.25)
mtext("unweighted", side = 3, line = 1, font = 2, cex = 1.25)
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)

# plot weighted probtrans
plot(1, type = "n",
     xlim = c(0, xlimmax), ylim = c(0, 0.3),
     # main = "weighted", ylab = "",
     xlab = "Days since admission", cex.axis = cexa, cex.lab = cexl)

for (i in 1:length(transprob_w)){
   for (seed in 1:1){
  lines(transprob_w[[i]][[seed]][[1]]$time,
        transprob_w[[i]][[seed]][[1]]$pstate2, type = "l",
        lwd = lwd,
        lty = mylinetypes[[i + 1]],
        col = alpha("black", 1))
```

```
   }
  lines(transprob_uw[[1]][[1]][[1]]$time,
        transprob_uw[[1]][[1]][[1]]$pstate2,
        lty = 1, lwd = lwd, type = "l", col = "black")
}


legend("topleft", c("Full", "CC", "NCC" , "PP"),
       lty = 1:4, col = "black",
       lwd = 2,  cex = cexl)
mtext("weighted", side = 3, line = 1, font = 2, cex = 1.25)
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)

# mtext("Infection", outer = TRUE, cex = cexh, side = 3, line = -0.5)

# par(mfrow = c(1, 2), oma = c(0, 0, 3, 0))
plot.new()
text(0.5, 0.3, "Discharge/Death without Infection", cex=2.5, font = 2)

# plot unweighted probtrans
plot(1, type = "n",  xlim = c(0, xlimmax), ylim = c(0, 1),
     # main = "unweighted",
     ylab ="Cumulative incidence function",
     xlab ="Days since admission",  cex.axis = cexa, cex.lab = cexl)

for (i in 1:length(transprob_uw)){
  for (seed in 1:1){
  lines(transprob_uw[[i]][[seed]][[1]]$time,
        transprob_uw[[i]][[seed]][[1]]$pstate3, type = "l",
        lwd = lwd, lty = mylinetypes[[i]],
        col = alpha(mycolors[[i]], 1))
  }
  lines(transprob_uw[[1]][[1]][[1]]$time,
        transprob_uw[[1]][[1]][[1]]$pstate3,
        lty = 1, lwd = lwd, type = "l", col = "black")
}

mtext("Cumulative incidence function", side = 2, line = 3, font = 1, cex = 1.25)
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)
mtext("unweighted", side = 3, line = 1, font = 2, cex = 1.25)

# plot weighted probtrans
plot(1, type = "n",
     xlim = c(0, xlimmax), ylim = c(0, 1),
     # main = "weighted",
     xlab = "Days since admission", ylab = "" , cex.axis = cexa, cex.lab = cexl)

for (i in 1:length(transprob_w)){
  for (seed in 1:1){
  lines(transprob_w[[i]][[seed]][[1]]$time,
        transprob_w[[i]][[seed]][[1]]$pstate3, type = "l",
        lwd = lwd, lty = mylinetypes[[i + 1]], col = alpha("black", 1))
  }
```

```
  lines(transprob_uw[[1]][[1]][[1]]$time,
        transprob_uw[[1]][[1]][[1]]$pstate3,
        lty = 1, lwd = lwd, type = "l", col = "black")
}
mtext("Days since admission", side = 1, line = 3, font = 1, cex = 1.25)
mtext("weighted", side = 3, line = 1, font = 2, cex = 1.25)

# dev.off()
```

Appendix 2 - R Tutorial

```
# This script is written as a Rmd file which was written with R version 4.1.1 (2021-08-10)
```

```
# install.packages("pacman") # run this if you do not have installed the package pacman
library(pacman)
pacman::p_load(etm, survival, sandwich, lmtest, mvna, bshazard, plyr, dplyr,
               multipleNCC, here, mstate, lattice, gridExtra)
```

```
# load your data. We named the data frame:
my.data
```

Preparing Dataset

The dataset is prepared to fit functions for multi-state modeling. We define the states admission as state 0, bloodstream infection as state 1 and discharge/death as state 2. There is one row per transition in the dataset i. e. one patient could have several rows. For the time variables time, entry, and exit we take the timescale in days since ICU admission.

id: patient id
from: transition start state
to: transition end state
time: time of transition
entry: start time of a state
exit: end time of a state

Sampling the Cohorts and IPW Weighting

CC Case Cohort Sampling and weights

```
# Set the fraction of the sub-cohort (Case Cohort)
sampling_weight <- 0.15 # 15 % in the example
# In the weighted analysis cases outside the sub-cohort enter the cohort just
# shortly before they become infected.
# That is why their entry time in the cohort is calculated by subtracting the
# time of the event minus a small fraction. Here we use 0.0001 days (8.64 seconds).
# The same applies for cases occurring inside of the sub-cohort.
smallFraction <- 0.0001
```

For the case cohort design a sub-sample of 15 % is sampled randomly.
In the weighted analysis cases from outside the sub-sample are added to the cohort shortly before the event occurs (i. e. the name case-cohort study).
Weights are defined as the inverted probability of being sampled in the case cohort (Inverse-probability weighting, IPW).
Within sub-cohort: 1/sampling fraction of the random sample i.e. 1/0.15. Weighting changes to 1 at the time of infection as all infected patients are included.
Outside sub-cohort: Enters the analysis at time of infection, weighted 1.
In the weighted analysis cases outside the sub-cohort enter the cohort just before infection. Therefore, a very small value, here 0.000000001157407 seconds, is subtracted of the time of the event to calculate the entry time, so these patients don't contribute to the analysis before turning out to be a case. Similar applies for cases from inside the sub-cohort.They are split into two time spans separated by the time just before getting infected.[15]

```
my.data.CC <- my.data

# Select subcohortmembers
set.seed(24388)
my.data.CC$subcohort <- rbinom(nrow(my.data.CC), 1, sampling_weight)

# creating a describing variable "type" to label cases and non cases
# in and outside of the sub-cohort.
my.data.CC$type <- NA

my.data.CC[my.data.CC$subcohort == 1 &
             my.data.CC$to == 2,]$type <- "non case subcohort"
# my.data.CC[my.data.CC$subcohort == 1 &
#            my.data.CC$to == "cens",]$type<-"non case subcohort"
my.data.CC[my.data.CC$subcohort == 0 &
             my.data.CC$to == 2,]$type <- "non case non subcohort"
# my.data.CC[my.data.CC$subcohort == 0 &
#            my.data.CC$to == "cens",]$type<-"non case non subcohort"
my.data.CC[my.data.CC$subcohort == 1 &
             my.data.CC$to == 1,]$type <- "case subcohort"
my.data.CC[my.data.CC$subcohort == 0 &
             my.data.CC$to == 1,]$type <- "case non subcohort"

case.cohort.pop <- my.data.CC

##### Calculate weights:#####

# Controls not in the sub-cohort get a weight of zero
case.cohort.pop$weight <- 0

# Controls inside the sub-cohort get the weight of 1/sampling fraction
# of the random sample i.e. 1/0.15.
case.cohort.pop[case.cohort.pop$subcohort == 1,]$weight <- 1/sampling_weight

# Cases in the sub-cohort change their weight from 1/sampling fraction to 1.
# This happens very shortly before the time of infection.
# Cases inside of the sub-cohort are marked with "cens2"
case.cohort.pop[case.cohort.pop$subcohort == 1 & case.cohort.pop$to == 1,]$exit <-
  case.cohort.pop[case.cohort.pop$subcohort == 1 &
                    case.cohort.pop$to == 1,]$exit - smallFraction
```

```
case.cohort.pop[case.cohort.pop$subcohort == 1 &
                  case.cohort.pop$to == 1,]$to <- "cens2"

cases.in.subcohort <- my.data.CC[my.data.CC$subcohort == 1 &
                                   my.data.CC$type == "case subcohort",]
cases.in.subcohort$weight <- 1
cases.in.subcohort$entry <- cases.in.subcohort$exit - smallFraction

# cases not in the sub-cohort are assigned a weight of 1 at the time
# shortly before the time of infection.

cases.no.subcohort <- case.cohort.pop[case.cohort.pop$subcohort == 0 &
                                        case.cohort.pop$to == 1,]
cases.no.subcohort$weight <- 1
cases.no.subcohort$entry <- cases.no.subcohort$exit - smallFraction

# In the case.cohort dataset patients inside the cohort who get infected
# contribute as control and as case. That is why cases occurring inside the
# cohort are represented by two rows in the dataset.
case.cohort <- rbind(case.cohort.pop, cases.in.subcohort, cases.no.subcohort)
case.cohort <- case.cohort[case.cohort$weight != 0,]

case.cohort <- case.cohort[order(case.cohort$id),]

# For logistic regression we create a table with only one row per patient
# i. e. the doubled rows for cases inside the sub-cohort marked with "cens2"
# are excluded and the patients with an infection are weighted with 1.
case.cohort.glm <- case.cohort[case.cohort$to != "cens2",]
case.cohort.glm[case.cohort.glm$to == 1,]$weight <- 1

# shorter writing for following analysis
CC.glm <- case.cohort.glm
CC <- case.cohort
```

NCC Nested Case Control Sampling and weights

In our setting we sample four controls for every case with the same length of stay. We break ties for event times, as otherwise too many patients have exactly the same value as we only count in days not in hours our minutes.

```
# Create a function for NCC sampling
CreateSamplestat <- function(nestinfo, data, samplestat, id, set) {

  ## Adjusts samplestat vector for use in multipleNCC package.
  ## Sampled controls are given "1."
  ##
  ##
  ## Args:
  ##   nestinfo:    data frame of information for reduced cohort after
  ##                IDS sampling
  ##   joininfo:    full cohort data frame before sampling
  ##   samplestat:  vector of sampling and status (case or control) info
  ##                before adjustment
```

```
  ##   id:          id for joining data frames
  ##   set:         time-matched case-control set
  ##
  ## Returns:
  ##   data frame for full cohort with adjusted samplestat vector

  ## marks patient as sampled
  nestinfo[, samplestat] <- replace (nestinfo[, samplestat],
                                     nestinfo[[samplestat]] == 0, 1)

  # gives columns consistent names for join function
  ncc1 <- data.frame( nestinfo[[id]],  nestinfo[[samplestat]],  nestinfo[[set]])
  colnames(ncc1) <- c(id, samplestat, set)

  # removes duplicate event times
  ncc1 <- unique(ncc1[!duplicated(ncc1[-3]),] )

  # merges new samplestat vector to original data frame
  ncc1 <- join(ncc1, data, by = id, type = "full")

  # returns data frame
  ncc1

}
```

```
my.data.NCC <- my.data

# Break ties for event times. Otherwise to many exactly the same values.

my.data.NCC$exit_noise <- jitter(my.data.NCC$exit, amount = .001)

# Create sample indicator vector. Everyone who become infected is assigned a 2
my.data.NCC$samplestat1 <-0
my.data.NCC$samplestat1 <- ifelse(my.data.NCC$to == 1, 2, 0)

# Binary variable for event 1 (infection).
# Everyone who become infected is assigned a 1
my.data.NCC$event1 <- ifelse(my.data.NCC$to == 1, 1, 0)

# Incidence Density Sampling - generating a nested case-control study.
# Sampling 4 controls for each case

my_ncc <- ccwc(entry = 0,
               exit = exit_noise,
               fail = to == 1,
               origin = 0,
               controls = 4,
               data = my.data.NCC,
               include = list(id, from, to, exit, event1, CHARLSON_FIRST,
                              samplestat1, day_in_ICU))

# Adjusts sample indicator vector (samplestat) for use in {multipleNCC} package.
# Sampled controls are given "1."
```

```
Sample_Data <- CreateSamplestat(my_ncc, my.data.NCC, "samplestat1", "id", "Set")

# Logistic Regression Weights. Calculates the probability to be included
# into the study for each patient.
Sample_Data$GLM_weights1 <- GLMprob(Sample_Data[["exit"]],
                                    Sample_Data[["samplestat1"]])

Sample_Data$weight <- 1/Sample_Data$GLM_weights1

# remove non-cases, non-sampled individuals
NCC <- subset(Sample_Data, Sample_Data[,"samplestat1"] != 0)
```

PP Point prevalence Sampling and weights

In our data we do not have dates, therefore we artificially create a PPS sample by randomly picking a number between 0 and 137 for every patient. In our dataset this was the 99 Quartile of days spent in the ICU. This is the patient's PPS day. If the PPS day is smaller than the patient's days in the intensive car unit (ICU), they are sampled.

```
my.data.pp <- my.data
# 99 Quartile of day_in_ICU= 136.4
my.data.pp$pps.date <- runif(n = nrow(my.data.pp), min = 0, max = 137)

my.data.pps <- my.data.pp[my.data.pp$day_in_ICU > my.data.pp$pps.date , ]

# To correct the length-biased sampling we weight patients
# inversely to their days in ICU.

my.data.pps.cens <- my.data.pps
my.data.pps$weight <- 1/my.data.pps$day_in_ICU
my.data.pps <- subset(my.data.pps, weight > 0)

PP <- my.data.pps
PP$event1 <- ifelse(PP$to == 1, 1, 0)
```

Full Cohort - Cause-Specific Cox Models and Logistic Regression

We estimate cause-specific cox proportional hazards models to access direct and indirect effects of the covariate CHARLSON_FIRST, the Charlson Comorbidity Index (CCI) at admission on the event infection and the combined endpoint discharge and death.

```
coxmodel.full.event1 <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST,
                              my.data)
coxmodel.full.event2 <- coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST,
                              my.data)

log.full <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                my.data)
exp(cbind(HR = coef(coxmodel.full.event1), confint(coxmodel.full.event1)))
exp(cbind(HR = coef(coxmodel.full.event2), confint(coxmodel.full.event2)))
exp(cbind(OR = coef(log.full), confint(log.full)))
```

CC Case Cohort Study

Cox Regression

```
# CC: For the unweighted analysis we use the dataset with one row per patient.
# In the weighted analysis we use the dataset where several rows per patient are possible.
# unweighted
coxmodel.cc.event1.uw <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST,
                               CC.glm)
# weighted
coxmodel.cc.event1.w <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST + cluster(id),
                              CC.glm, weights = weight)
# unweighted
coxmodel.cc.event2.uw <- coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST,
                               CC.glm)

# weighted
# In the CC design only the sub-cohort is used for the analysis of the discharge hazard
coxmodel.cc.event2.w <- coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST,
                              data = my.data.CC, subset = subcohort == 1, model = TRUE)

exp(cbind(HR = coef(coxmodel.cc.event1.uw), confint(coxmodel.cc.event1.uw)))
exp(cbind(HR = coef(coxmodel.cc.event1.w), confint(coxmodel.cc.event1.w)))
exp(cbind(HR = coef(coxmodel.cc.event2.uw), confint(coxmodel.cc.event2.uw)))
exp(cbind(HR = coef(coxmodel.cc.event2.w), confint(coxmodel.cc.event2.w)))
```

Logistic Regression

Two estimate the odds ratio of the CCI for infection we performed a logistic regression.

```
# unweighted
log.cc.uw <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                 CC.glm)
exp(cbind(OR = coef(log.cc.uw), confint(log.cc.uw)))

# weighted
log.cc.w <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                CC.glm, weights = weight)

robu_cov_1.cc <- vcovHC(log.cc.w)
#coeftest(log.cc.w, robu_cov_1.cc)
exp(cbind(OR = coef(log.cc.w), confint(log.cc.w)))
```

NCC Nested Case-control Study

Cox Regression

```
# Event 1
# unweighted
# NCC we need to stratify for the matched pairs (Set) in the analysis
```

```
coxmodel.ncc.event1.uw <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST +
                                strata(Set), NCC, model = TRUE)
# weighted
coxmodel.ncc.event1.w <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST +
                               cluster(id), NCC, weights = weight)
# Event 2
# unweighted
coxmodel.ncc.event2.uw <- coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST,
                                NCC)
# weighted
coxmodel.ncc.event2.w <- coxph(Surv(exit, to == 2) ~ CHARLSON_FIRST +
                               cluster(id), NCC, weights = weight)

exp(cbind(HR = coef(coxmodel.ncc.event1.uw), confint(coxmodel.ncc.event1.uw)))
exp(cbind(HR = coef(coxmodel.ncc.event1.w), confint(coxmodel.ncc.event1.w)))
exp(cbind(HR = coef(coxmodel.ncc.event2.uw), confint(coxmodel.ncc.event2.uw)))
exp(cbind(HR = coef(coxmodel.ncc.event2.w), confint(coxmodel.ncc.event2.w)))
```

Logistic Regression

```
# unweighted
log.ncc.uw <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                  NCC)
exp(cbind(OR = coef(log.ncc.uw), confint(log.ncc.uw)))

# weighted
log.ncc.w <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                 NCC, weights = weight)

robu_cov_1.ncc <- vcovHC(log.ncc.w)
#coeftest(log.ncc.w, robu_cov_1.ncc)
exp(cbind(OR = coef(log.ncc.w), confint(log.ncc.w)))
```

PP Point prevalence study

Cox Regression

```
# Event 1
# unweighted
coxmodel.pp.event1.uw <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST,
                               PP)
# weighted
coxmodel.pp.event1.w <- coxph(Surv(entry, exit, to == 1) ~ CHARLSON_FIRST +
                              cluster(id), PP, weights = weight)
# Event 2
# unweighted
coxmodel.pp.event2.uw <- coxph(Surv(entry, exit, to == 2) ~ CHARLSON_FIRST,
                               PP)
# weighted
```

```
coxmodel.pp.event2.w <- coxph(Surv(exit, to == 2) ~ CHARLSON_FIRST +
                              cluster(id), PP, weights = weight)

exp(cbind(HR = coef(coxmodel.pp.event1.uw), confint(coxmodel.pp.event1.uw)))
exp(cbind(HR = coef(coxmodel.pp.event1.w), confint(coxmodel.pp.event1.w)))
exp(cbind(HR = coef(coxmodel.pp.event2.uw), confint(coxmodel.pp.event2.uw)))
exp(cbind(HR = coef(coxmodel.pp.event2.w), confint(coxmodel.pp.event2.w)))
```

Logistic Regression

```
# unweighted
log.pp.uw <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"), PP)
exp(cbind(OR = coef(log.pp.uw), confint(log.pp.uw)))

# weighted
log.pp.w <- glm(to == 1 ~ CHARLSON_FIRST, family = binomial(link = "logit"),
                PP , weights = weight)

robu_cov_1.pp <- vcovHC(log.pp.w)
#coeftest(log.pp.w, robu_cov_1.pp)
exp(cbind(OR = coef(log.pp.w), confint(log.pp.w)))
```

Multi-state Modelling - Calculating Cumulative Hazards and Transition Probabilites (alias Cumulative Incidence Function)

We will demonstrate the analysis with the full cohort first and will than perform it again for all three study designs.

As a first step, we define which transitions are possible in a transition matrix (traCRetm). We use this matrix to compute the Nelson-Aalen estimator of the cumulative transition hazards in a multi-state model.[6] For each possible transition, it computes an estimate of the cumulative hazard (see: mvna{mvna}). In our case we have the transition from admission (state 0) to bloodstream infection (state 1) or from admission to discharge (state 2).
The hazard function for both events (infection and discharge) is estimated non-parametrically (no assumption about the distribution of failure times) with the bshazard{bshazard} function using a survival object. Left truncation is considered by including the entry time (variable "entry").

Lastly the empirical transition matrix (Aalen-Johansen estimator) of the transition probability matrix of the multi-state model is computed using the etm {etm} function.

For more detailed information on "Competing risks and multistate models with R" we recommend the book of the same name by Beyersmann et al.[5]

Calculation for the Full Cohort without Weighting

```
# matrix describing the possible transitions between the states
traCRetm <- matrix(FALSE, 3, 3)
traCRetm[1,2] <- TRUE
traCRetm[1,3] <- TRUE
```

```r
# transition names
traCR <- matrix(NA, 3, 3)
traCR[1,2] <- 1
traCR[1,3] <- 2

# calculating the cumulative hazards for these transitions from the data.
# If you have censoring , choose "cens" as the last argument.
# (Nelson-Aalen estimator)
# The dataset used here should contain one row per transition
# (possibly several rows per patient).
# Specifying an entry and exit time in the data enables to account for left-truncation.
cumHaz.my.data <- mvna(my.data, c(0:2), traCRetm, "cens")

# estimating the hazard function for state 1 (infection) and 2 (discharge/death)
# non-parametrically (no assumption about the distribution of failure times)
# smoothing of the hazard function
bs2 <- bshazard(Surv(entry, exit, to == 2) ~ 1, data = my.data)
bs1 <- bshazard(Surv(entry, exit, to == 1) ~ 1, data = my.data)

# computation of the empirical transition matrix (Aalen-Johansen estimator)
# of the transition probability matrix of the multi-state model.
# It considers competing risks. Account for censoring if applicable for your data.
TransProb.my.data <- etm::etm(data = my.data, state.names = as.character(0:2),
                              tra = traCRetm, cens.name = "cens" , s = 0, covariance = F)
```

```r
# plot
par(mfrow = c(1,2))
  plot(cumHaz.my.data,
       xlim = c(0,60), ylim = c(0,3),
       xlab ="Days since admission",
       lty = 1, col = c(1,4),
       legend = FALSE)
  legend("topleft", c("Infection full cohort",
                      "Discharge/Death w/o Infection full cohort"),
         col = c(1, 4), lty = c(1,1),
         bty = "n", lwd = 2, cex = 0.9)
  plot(TransProb.my.data, xlim = c(0,60),
       xlab="Days since admission",
       ylab="Cumulative incidence function",
       tr.choice = c("0 1", "0 2"),
       lty = 1, col = c(1,4), legend = FALSE)
```

CC - Cumulative Hazards and Transition Probabilites

For CC it is a little more complicated, as weights are time-dependent. We used this approach also for NCC and PP study design for the results in the paper.

```r
# unweighted
cumHaz.cc.uw <- mvna(data = CC.glm, tra = traCRetm,
                     state.names = c("0","1","2"), cens.name = "cens")
TransProb.cc.uw <- etm(data = CC.glm, tra = traCRetm,
                       state.names = c("0","1","2"), cens.name = "cens", s = 0)
```

```r
# weighted

## Coxmodels
c01uw <- coxph(Surv(exit, to == 1) ~ 1, data = CC.glm)
c01w <- coxph(Surv(entry, exit, to == 1) ~ 1, data = CC,
              weights = weight)

c02uw <- coxph(Surv(exit, to == 2) ~ 1, data = CC.glm)
c02w <- coxph(Surv(entry, exit, to == 2) ~ 1, data = my.data.CC,
              subset = subcohort == 1)

# estimating the hazard function for state 1 (infection) and 2 (discharge)
# non-parametrically (no assumption about the distribution of failure times)
# smoothing of the hazard function
basehaz01uw <- basehaz(c01uw, centered = FALSE)
basehaz01w <- basehaz(c01w, centered = FALSE)

basehaz02uw <- basehaz(c02uw, centered = FALSE)
basehaz02w <- basehaz(c02w, centered = FALSE)
```

```r
# This function is used to create a msfit object.

# input data: coxphData should be a dataframe with two columns:
# coxphData$time and coxphData$hazard
# exitTime is a vector of timepoints

# In our data coxphData$time are the transition times of a specific event
# coxphData$hazard are the values of the cumulative hazards at each trans. time
# exitTime is the vector, that contains the transition times of all possible events

mixin12 <- function( coxphData, exitTime) {

  ## Get all timepoints only once
  allTimes <- sort(union(coxphData$time, exitTime))

  ## create dataframe with the unique time points and a hazard column
  result <- data.frame( time=allTimes, hazard=rep(-1,length(allTimes)) )

  ## check if time point of allTimes has a cumulative hazards value
  for( i in seq(1,nrow(result)) ) {
    coxTime <- which(coxphData$time == result$time[i] )


    if( length(coxTime) > 0 ) {
      ## This shouldn't happen, since we have unique time points:
      if( length(coxTime) > 1 )
        stop( "error: more than one timepoint at that point" )
      # take hazard value at that time point
      result$hazard[i] = coxphData$hazard[coxTime]
    }
    else {

      ## If first timepoint is exit value, then cumHaz is 0
```

```
      ## If previous hazardvalue in results is smaller than zero, then set it to 0.
      ## This shouldn't happen.

      if( ( i == 1 ) || (result$hazard[i-1] < 0 )){result$hazard[i] = 0}


      else  ## take previous hazard value
      {result$hazard[i] = result$hazard[i-1]}

    }
  }

  cat("Result: ", nrow(result), " Added from exit: ", nrow(result) -
        length(coxphData$time), "\n" )

  invisible(result)
}
```

```
# Preparation of the data to obtain the msfit object:
times <- basehaz01w$time
FinalTime <- c(times, times)
colTrans <- c(rep(1, length(times)), rep(2, length(times)))

haz01w <- mixin12(basehaz01w, times)
haz02w <- mixin12(basehaz02w, times)
hazard0w <- rbind(haz01w, haz02w)
hazard0msfit_w <- data.frame(time = FinalTime,
                             Haz = hazard0w$hazard,
                             trans = colTrans)
res0w <- list(Haz = hazard0msfit_w, trans = traCR)
class(res0w) <- "msfit"

haz01uw <- mixin12(basehaz01uw, times)
haz02uw <- mixin12(basehaz02uw, times)
hazard0uw <- rbind(haz01uw, haz02uw)
hazard0msfit_uw <- data.frame(time = FinalTime,
                              Haz = hazard0uw$hazard,
                              trans = colTrans)
res0uw <- list(Haz = hazard0msfit_uw, trans = traCR)
class(res0uw) <- "msfit"


# From the msfit objects (the estimated cumulative hazards for each transition)
# the transition probabilities in our multi-state model are calculated.

TransProb.cc.uw <- probtrans(res0uw, predt = 0, variance = FALSE)
TransProb.cc.w <- probtrans(res0w, predt = 0, variance = FALSE)
```

```
# plot
par(mfrow = c(1,2))
plot(cumHaz.my.data, tr.choice = c("0 1", "0 2"), xlim = c(0, 60),
     lty = c(1,1), lwd = c(1,1),
     col = c(1,4), ylim = c(0,3),
```

```
     xlab = "Days since admission", legend = FALSE)

lines(cumHaz.cc.uw$`0 1`$time, cumHaz.cc.uw$`0 1`$na, lty = 2, col = 6)
lines(cumHaz.cc.uw$`0 2`$time, cumHaz.cc.uw$`0 2`$na, lty = 2, col = 4)
lines(basehaz01w$time, basehaz01w$hazard, lty = 4, col = 6)
lines(basehaz02w$time, basehaz02w$hazard, lty = 4, col = 4)
legend("topleft", c("Infection full cohort",
                    "Infection unweighted",
                    "Infection weighted",
                    "Discharge/Death w/o Infection full cohort",
                    "Discharge/Death w/o Infection unweighted",
                    "Discharge/Death w/o Infection weighted"),
       col = c(1, 6, 6, 4, 4, 4), lty = c(1, 2, 4, 1, 2, 4),
       bty = "n", lwd = 2, cex = 0.9)

plot(TransProb.my.data, xlim = c(0, 60), ylim = c(0, 1), lty = c(1,1), col = c(1,4),
     ylab = "Cumulative incidence function",
     xlab = "Days since admission", tr.choice = c("0 1", "0 2"), legend = FALSE)
lines(TransProb.cc.uw[[1]]$time, TransProb.cc.uw[[1]]$pstate2, lty = 2, col = 6)
lines(TransProb.cc.uw[[1]]$time, TransProb.cc.uw[[1]]$pstate3, lty = 2, col = 4)
lines(TransProb.cc.w[[1]]$time, TransProb.cc.w[[1]]$pstate2, lty = 4, col = 6)
lines(TransProb.cc.w[[1]]$time, TransProb.cc.w[[1]]$pstate3, lty = 4, col = 4)
legend("topleft", c("Infection full cohort",
                    "Infection unweighted",
                    "Infection weighted",
                    "Discharge/Death w/o Infection full cohort",
                    "Discharge/Death w/o Infection unweighted",
                    "Discharge/Death w/o Infection weighted"),
       col = c(1, 6, 6, 4, 4, 4), lty = c(1, 2, 4, 1, 2, 4),
       bty = "n", lwd = 2, cex = 0.9)
```

For NCC and PP there is also an easier approach. By replicating the patients according to their weights a weighted cohort is generated which can be analysed with common functions.

NCC - Cumulative Hazards and Transition Probabilites

```
# unweighted
cumHaz.ncc.uw <- mvna(data = NCC, tra = traCRetm,
                      state.names = c("0","1","2"), cens.name = "cens")
TransProb.ncc.uw <- etm(data = NCC, tra = traCRetm,
                        state.names = c("0","1","2"), cens.name = "cens", s = 0)

# weigthed
### repeat data according to weights to generate weighted cohort sample
# to prevent rounding to have a to great effect, weights are scaled

NCC$weight.freq <- NCC$weight
NCC$weight.freq <- NCC$weight.freq * 1/
                      min(NCC$weight.freq[NCC$weight.freq > 0]) * 10
NCC$weight.freq <- round(NCC$weight.freq)
```

```
res <- apply(NCC, 1, function(z) {
  rep(z, times = z["weight.freq"]) })

NCC.w <- matrix(unlist(res), ncol = ncol(NCC), byrow = TRUE)
NCC.w <- data.frame(NCC.w)

for (i in 1:ncol(NCC.w)) {
    NCC.w[, i] <- as.numeric(NCC.w[, i])
}
colnames(NCC.w) <- names(NCC)
###

cumHaz.ncc.w <- mvna(data = NCC.w, tra = traCRetm,
                     state.names = c("0","1","2"), cens.name = "cens")
TransProb.ncc.w <- etm(data = NCC.w, tra = traCRetm,
                       state.names = c("0","1","2"), cens.name = "cens", s = 0)
```

```
# plot
par(mfrow = c(1,2))
plot(cumHaz.my.data, tr.choice = c("0 1", "0 2"),
     xlim = c(0, 60), lty = c(1,1), lwd = c(1,1),
     col = c(1,4), ylim = c(0,3),
     xlab ="Days since admission", legend = FALSE)
lines(cumHaz.ncc.uw$`0 1`$time, cumHaz.ncc.uw$`0 1`$na, lty = 2, col = 6)
lines(cumHaz.ncc.uw$`0 2`$time, cumHaz.ncc.uw$`0 2`$na, lty = 2, col = 4)
lines(cumHaz.ncc.w$`0 1`$time, cumHaz.ncc.w$`0 1`$na, lty = 4, col = 6)
lines(cumHaz.ncc.w$`0 2`$time, cumHaz.ncc.w$`0 2`$na, lty = 4, col = 4)
legend("topleft", c("Infection full cohort",
                    "Infection unweighted",
                    "Infection weighted",
                    "Discharge/Death w/o Infection full cohort",
                    "Discharge/Death w/o Infection unweighted",
                    "Discharge/Death w/o Infection weighted"),
       col = c(1, 6, 6, 4, 4, 4), lty = c(1, 2, 4, 1, 2, 4),
       bty = "n", lwd = 2, cex = 0.9)


sTransncc.w <- summary(TransProb.ncc.w)
sTransncc.uw <- summary(TransProb.ncc.uw)

plot(TransProb.my.data,
     xlim = c(0, 60),
     ylim = c(0, 1), lty = c(1,1), col = c(1,4),
     ylab="Cumulative incidence function",
     xlab="Days since admission", tr.choice = c("0 1", "0 2"), legend = FALSE)
lines(sTransncc.uw$`0 1`$time, sTransncc.uw$`0 1`$P, lty = 2, col = 6)
lines(sTransncc.uw$`0 2`$time, sTransncc.uw$`0 2`$P, lty = 2, col = 4)
lines(sTransncc.w$`0 1`$time, sTransncc.w$`0 1`$P, lty = 4, col = 6)
lines(sTransncc.w$`0 2`$time, sTransncc.w$`0 2`$P, lty = 4, col = 4)
legend("topleft", c("Infection full cohort",
                    "Infection unweighted",
                    "Infection weighted",
                    "Discharge/Death w/o Infection full cohort",
```

```
                    "Discharge/Death w/o Infection unweighted",
                    "Discharge/Death w/o Infection weighted"),
       col = c(1, 6, 6, 4, 4, 4), lty = c(1, 2, 4, 1, 2, 4),
       bty = "n", lwd = 2, cex = 0.9)
```

PP - Cumulative Hazards and Transition Probabilites

```
# unweighted
cumHaz.pp.uw <- mvna(data = PP, tra = traCRetm,
                     state.names = c("0","1","2"), cens.name = "cens")
TransProb.pp.uw <- etm(data = PP, tra = traCRetm,
                       state.names = c("0","1","2"), cens.name = "cens", s = 0)

# weighted
### repeat data according to weights to generate weighted cohort sample
# as weights are below 1 we need to take care especially
my.data.pps$weight.freq <- my.data.pps$weight
my.data.pps$weight.freq <- my.data.pps$weight.freq * 1/
                       min(my.data.pps$weight.freq[my.data.pps$weight.freq > 0]) * 100
my.data.pps$weight.freq <- round(my.data.pps$weight.freq)

res <- apply(my.data.pps, 1, function(z) {
  rep(z, times = z["weight.freq"]) })

PP.w <- matrix(unlist(res), ncol = ncol(my.data.pps), byrow = TRUE)
PP.w <- data.frame(PP.w)

for (i in 1:ncol(PP.w)) {
    PP.w[, i] <- as.numeric(PP.w[, i])
}

colnames(PP.w) <- names(my.data.pps)
####

cumHaz.pp.w <- mvna(data = PP.w, tra = traCRetm,
                    state.names = c("0","1","2"), cens.name = "cens")
TransProb.pp.w <- etm(data = PP.w, tra = traCRetm,
                      state.names = c("0","1","2"), cens.name = "cens", s = 0)
```

```
# plot
par(mfrow = c(1,2))
plot(cumHaz.my.data, tr.choice = c("0 1", "0 2"),
     xlim = c(0, 60), lty = c(1,1), lwd = c(1,1),
     col = c(1,4), ylim = c(0,3),
     xlab = "Days since admission", legend = FALSE)
lines(cumHaz.pp.uw$`0 1`$time, cumHaz.pp.uw$`0 1`$na, lty = 2, col = 6)
lines(cumHaz.pp.uw$`0 2`$time, cumHaz.pp.uw$`0 2`$na, lty = 2, col = 4)
lines(cumHaz.pp.w$`0 1`$time, cumHaz.pp.w$`0 1`$na, lty = 4, col = 6)
lines(cumHaz.pp.w$`0 2`$time, cumHaz.pp.w$`0 2`$na, lty = 4, col = 4)
legend("topleft", c("Infection full cohort",
                    "Infection unweighted",
```

```
                    "Infection weighted",
                    "Discharge/Death w/o Infection full cohort",
                    "Discharge/Death w/o Infection unweighted",
                    "Discharge/Death w/o Infection weighted"),
       col = c(1, 6, 6, 4, 4, 4), lty = c(1, 2, 4, 1, 2, 4),
       bty = "n", lwd = 2, cex = 0.9)

sTransPP.w <- summary(TransProb.pp.w)
sTransPP.uw <- summary(TransProb.pp.uw)

plot(TransProb.my.data, xlim = c(0, 60), ylim = c(0, 1),
     lty = c(1,1), col = c(1,4),
     ylab = "Cumulative incidence function",
     xlab = "Days since admission", tr.choice = c("0 1", "0 2"), legend = FALSE)
lines(sTransPP.uw$`0 1`$time, sTransPP.uw$`0 1`$P, lty = 2, col = 6)
lines(sTransPP.uw$`0 2`$time, sTransPP.uw$`0 2`$P, lty = 2, col = 4)
lines(sTransPP.w$`0 1`$time, sTransPP.w$`0 1`$P, lty = 4, col = 6)
lines(sTransPP.w$`0 2`$time, sTransPP.w$`0 2`$P, lty = 4, col = 4)
legend("topleft", c("Infection full cohort",
                    "Infection unweighted",
                    "Infection weighted",
                    "Discharge/Death w/o Infection full cohort",
                    "Discharge/Death w/o Infection unweighted",
                    "Discharge/Death w/o Infection weighted"),
       col = c(1, 6, 6, 4, 4, 4), lty = c(1, 2, 4, 1, 2, 4),
       bty = "n", lwd = 2, cex = 0.9)
```

References

5. Beyersmann J, Allignol A, Schumacher M. Competing Risks and Multistate Models with R. New York, NY: Springer New York; 2012.

6. Aalen OO, Johansen S. An Empirical Transition Matrix for Non-Homogeneous Markov Chains Based on Censored Observations. Scandinavian Journal of Statistics. 1978;5(3):141–150. Available from: http://www.jstor.org/stable/4615704.

15. Barlow WE, Ichikawa L, Rosner D, Izumi S. Analysis of Case-Cohort Designs. J Clin Epidemiol. 1999;52(12):1165-1172. doi:10.1016/S0895-4356(99)00102-X