# SurvivalSimulation

```r
remotes::install_github("osofr/simcausal")
```

```
## Using github PAT from envvar GITHUB_PAT
```

```
## Skipping install of 'simcausal' from a github remote, the SHA1 (d7299d4a) has not changed since last
##   Use `force = TRUE` to force installation
```

```r
library(simcausal)
```

The package sincausal allows one to simulate from user defined structural equation models. The main idea is to define a set of nodes which are causally ordered, each with their own distribution. Their distribution might depend on the values of past nodes. Therefore, the data must be simulated sequentially, following the causal ordering.

For an introduction to the package, see the github: https://github.com/osofr/simcausal

## Sample survival data (W, A, Delta, Ttilde) from scratch

```r
#Lets draw data from a parametric distribution using the package simcausal (github)
# We will assume that this is the real data from our study



D <- DAG.empty()
D <- D +
  node("W", distr = "runif", min =1, max = 25) +
  node("A", distr = "rbinom", size = 1, prob = 0.2 + W/50 ) +
  node("T", distr = "rexp", rate = 0.05*(1 - 0.5*A + 0.3*W) )+
  node("C", distr = "rexp",  rate = 0.03*(1 - 0.4*A + 0.2*W)) +
  node("Delta", distr = "rconst", const = as.numeric(T<=C)  ) +
  node("Ttilde", distr = "rconst", const = round(min(T,C,10) +1  ))
setD <- set.DAG(D)
```

```
## ...automatically assigning order attribute to some nodes...
```

```
## node W, order:1
```

```
## node A, order:2
```

```
## node T, order:3
```

```
## node C, order:4
```

```
## node Delta, order:5
```

```
## node Ttilde, order:6
```

```
## as.numeric(T <= C)
## min(cbind_mod(T, C, 10))
```

```r
# Lets draw 200 samples
dat <- sim(setD, n = 200)
```

```
## simulating observed dataset from the DAG object

## as.numeric(T <= C)
## min(cbind_mod(T, C, 10))
```

```r
head(dat)
```

```
##   ID         W A         T         C Delta Ttilde
## 1  1 21.255438 1 0.6796198  7.289365     1      2
## 2  2 12.025659 1 6.7605195 26.220218     1      8
## 3  3  1.478736 0 3.7975880 73.797266     1      5
## 4  4 12.373522 0 2.3878733  7.719747     1      3
## 5  5 20.921777 1 2.7749550 12.555266     1      4
## 6  6 12.844504 0 8.5620133  3.414262     0      4
```

```r
#Lets assume we are outcome blind.
# Thus, we are only given the outcome blind data (W, A)
baseline_data <- dat[, c("W", "A")]
#We can learn P(A |W) from the given data. For simplicty, lets use logistic regression
fit <- glm("A ~ W", family = binomial(), data = baseline_data)
#Lets store P(A=1|W) as a function of w
PA1_W <- function(w){
  as.vector(predict(fit, newdata = list(W = w), type = "response"))
}
PA1_W(0.5)
```

```
## [1] 0.1943149
```

## Simulate survival data from a known (empirical distribution) of W and estimated conditional density of binary A and parametric forms for censoring/survival

```r
# Our goal is to sample (W, A, Delta, Ttilde) survival data structure where W and A are sampled from di
# Lets estimate P(W) with the empirical distribution so that sampling from it is trivial
#Next, we assume that A is binary so that once we have drawn W = w, we can draw A using rbinom(1, size


n = 50
Wsamp = baseline_data$W
#This function samples from an empirical sample
remp <- function(n){
  emp_sample <- Wsamp
  sample(emp_sample, n, replace = T)
}

# Should be the function P(A = 1 |W).
prob_map <- PA1_W
#Make sure this map is vectorized if needed
prob_map <- Vectorize(prob_map)
#This function samples from A conditional on some value (i.e. W in this case) using our estimated condi
rTrtment <- function(n, value, prob){
  rbinom(n, size = 1, prob= prob(value))
}
```

```r
#Now, we would like to draw survival data. Specifically, we want to draw a time of death T and time of
#For this, we need to specify the conditional distributions P(T <= t| A, W) and
# P(C <= t| A, W).
# We can parametrically model these conditional distributions (as shown below) or we can estimate these


#First, let us use a parametric form for these conditional distributions of survival and censoring. We
#where the shape and scale parameters depend on A and W
D <- DAG.empty()
D <- D +
  # This node represents a sample of W's from our empirical distribution
  node("W", distr = "remp") +
    # This node represents a sample of A's from our conditional distribution of A given the previously
  node("A", distr = "rTrtment", value = W, prob =  prob_map) +
   # This node represents a drawn survival time from a parametric form of the conditional survival dist
  node("T", distr = "rweibull", shape = W + 5*A + W*A, scale = W + 5*A - 0.2*A*W) +
  # Same as above but for censoring
    node("C", distr = "rweibull", shape = W + 5*A + W*A, scale = W + 5*A - 0.205*A*W)+
  # For simplicity, lets make the times discrete by rounding up
  node("Tdiscrete", distr = "rconst", const = ceiling(T)) +
    node("Cdiscrete", distr = "rconst", const = ceiling(C)) +
  #Now we extract Ttilde and Delta
  node("Ttilde", distr = "rconst", const = min(Tdiscrete, Cdiscrete)) +
  node("Delta", distr = "rconst", const = as.numeric(Ttilde  == Tdiscrete) )


setD <- set.DAG(D)
```

```
## ...automatically assigning order attribute to some nodes...

## node W, order:1

## node A, order:2

## node T, order:3

## node C, order:4

## node Tdiscrete, order:5

## node Cdiscrete, order:6

## node Ttilde, order:7

## node Delta, order:8

## The node A expression(s): W.
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how

## min(cbind_mod(Tdiscrete, Cdiscrete))
## as.numeric(Ttilde == Tdiscrete)
```

```r
#Now we simulate the data by drawing n = 200 samples
dat <- sim(setD, n = 200)
```

```
## simulating observed dataset from the DAG object
## The node A expression(s): W.
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how t
```

```
## min(cbind_mod(Tdiscrete, Cdiscrete))
## as.numeric(Ttilde == Tdiscrete)
```

```r
# only grab ID, W's, A, T.tilde, Delta
head(dat)
```

```
##   ID         W A         T         C Tdiscrete Cdiscrete Ttilde Delta
## 1  1  5.343521 0  5.218204  4.637356         6         5      5     0
## 2  2  4.814514 0  1.966661  2.380793         2         3      2     1
## 3  3 21.523171 0 21.024566 20.248110        22        21     21     0
## 4  4  7.215797 1  9.784087 10.248202        10        11     10     1
## 5  5  6.112849 0  4.792027  5.883662         5         6      5     1
## 6  6 23.214078 1 23.455675 23.157565        24        24     24     1
```

```r
#Now we can extract our simulated observed data
observed_data <- dat[, c("W", "A", "Ttilde", "Delta")]
head(observed_data)
```

```
##           W A Ttilde Delta
## 1  5.343521 0      5     0
## 2  4.814514 0      2     1
## 3 21.523171 0     21     0
## 4  7.215797 1     10     1
## 5  6.112849 0      5     1
## 6 23.214078 1     24     1
```

## Simulate data from an estimated survival distribution

Now, lets assume we have an estimated conditional survival function.

```r
#Lets assume our estimated survival is exactly the pweibull distribution we previously used for simulat
est_surv <- function(t, A, W){
  #This could be any estimated survival function (e.g. survival from cox fit)
  1 - pweibull(t, shape = W + 5*A + W*A, scale = W + 5*A - 0.2*A*W)
}
# We need the CDF for sampling
est_CDF <- function(t, A ,W){
  return(1 - est_surv(t, A, W))
}
library(stats)
#This function takes a (absolutely) continuous CDF and returns the inverse
#Note if our survival data is discrete time then we will need to drawn in a different way using the haz
# Function to numerically compute inverse:
est_cdf_inverse <- function(p, A ,W){
  f_zero <- function(t){
    est_CDF(t, A, W) - p
  }
  grid_endpoints <- c(0, 30)
  root <- stats::uniroot(f_zero, interval = grid_endpoints)$root
  return(root)
}
#We need this to be vectorized
est_cdf_inverse <- Vectorize(est_cdf_inverse)
# True inverse using qweibull
```

```r
true_inverse <- function(p , A , W){
  qweibull(p, shape = W + 5*A + W*A, scale = W + 5*A - 0.2*A*W)
}
#They are the same
true_inverse(0.5, 1, 10)
```

```
## [1] 12.8108
```

```r
est_cdf_inverse(0.5, 1, 10)
```

```
## [1] 12.8108
```

```r
#Inverse transform (CDF) sampling
n_samples <- 100
A = 1
W = 10
#draws n samples from condiitonal dist given A and W
do_sample <- function(n, A, W){
  #Draw uniform random variables and feed into inverse CDF
  est_cdf_inverse(runif(n), A = A, W= W)
}

#simulated data summary stat
samples <- do_sample(n_samples, A , W)
mean(samples)
```

```
## [1] 12.79934
```

```r
sd(samples)
```

```
## [1] 0.5964264
```

```r
# simulate from rweibull
samples <- rweibull(n_samples, shape = W + 5*A + W*A, scale = W + 5*A - 0.2*A*W)

mean(samples)
```

```
## [1] 12.69752
```

```r
sd(samples)
```

```
## [1] 0.7197063
```

```r
#As expected, the two samples have similar means and variances
#As they are sampling from the same distribution.
```

```r
#Now, lets use our "do_sample" function to simulate the W A T C data.
# We will use the parametric form of the censoring for simplicity. Though the previous work can be appl
# The only changes are in the "T" node
D <- DAG.empty()
D <- D +
  # This node represents a sample of W's from our empirical distribution
  node("W", distr = "remp") +
    # This node represents a sample of A's from our conditional distribution of A given the previously
  node("A", distr = "rTrtment", value = W, prob =  prob_map) +
   # This node represents a drawn survival time from a parametric form of the conditional survival dist
  node("T", distr = "do_sample", A = A, W = W) +
  # Same as above but for censoring
```

```
    node("C", distr = "rweibull", shape = W + 5*A + W*A, scale = W + 5*A - max(0.205*A*W, 0))+
  # For simplicity, lets make the times discrete by rounding up
  node("Tdiscrete", distr = "rconst", const = ceiling(T)) +
    node("Cdiscrete", distr = "rconst", const = ceiling(C)) +
  #Now we extract Ttilde and Delta
  node("Ttilde", distr = "rconst", const = min(Tdiscrete, Cdiscrete)) +
  node("Delta", distr = "rconst", const = as.numeric(Ttilde  == Tdiscrete) )


setD <- set.DAG(D)
```

```
## ...automatically assigning order attribute to some nodes...

## node W, order:1

## node A, order:2

## node T, order:3

## node C, order:4

## node Tdiscrete, order:5

## node Cdiscrete, order:6

## node Ttilde, order:7

## node Delta, order:8

## The node A expression(s): W.
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how

## max(cbind_mod(0.205 * A * W, 0))
## min(cbind_mod(Tdiscrete, Cdiscrete))
## as.numeric(Ttilde == Tdiscrete)
```

```
#Now we simulate the data by drawing n = 200 samples
dat <- sim(setD, n = 200)
```

```
## simulating observed dataset from the DAG object
## The node A expression(s): W.
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how
##  One of the formulas evaluated to unknown data type. Make sure the distribution function knowns how

## max(cbind_mod(0.205 * A * W, 0))
## min(cbind_mod(Tdiscrete, Cdiscrete))
## as.numeric(Ttilde == Tdiscrete)
```

```
head(dat)
```

```
##   ID         W A         T         C Tdiscrete Cdiscrete Ttilde Delta
## 1  1  1.699926 0  1.325294  0.490954         2         1      1     0
## 2  2 15.522414 1 17.478178 17.136919        18        18     18     1
## 3  3  3.682359 0  4.194934  3.150684         5         4      4     0
## 4  4 12.030841 1 14.531806 14.465729        15        15     15     1
## 5  5  4.631800 1  8.757633  8.814486         9         9      9     1
## 6  6  3.682359 0  2.774327  4.481006         3         5      3     1
```

```r
#Now we can extract our simulated observed data
observed_data <- dat[, c("W", "A", "Ttilde", "Delta")]
head(observed_data)
```

```
##           W A Ttilde Delta
## 1  1.699926 0      1     0
## 2 15.522414 1     18     1
## 3  3.682359 0      4     0
## 4 12.030841 1     15     1
## 5  4.631800 1      9     1
## 6  3.682359 0      3     1
```

```r
# We now have our survival data using a custom conditional survvial fit.
```

## Simulation of survival times via hazard estimates for censoring and survival

```r
time_points = 1:10
hazard_f <- function(t, A, W){
  #some hazard of t, A and W
  plogis(-2 - 0.2*A + W/100)
}
hazard_f <- Vectorize(hazard_f)

hazard_g <- function(t, A, W){
  #some hazard of t, A and W
  plogis(-6)
}
hazard_g <- Vectorize(hazard_f)
D <- DAG.empty()
D <- D +
  node("W", distr = "runif", min =1, max = 25) +
  node("A", distr = "rbinom", size = 1, prob = 0.2 + W/50 ) +
  node("dNt", t = time_points,  distr = "rbern",  p = hazard_f(t, A, W)  , EFU  = T)+  node("dAt", t = t

# Lets draw 200 samples
setD <- set.DAG(D, vecfun = c("hazard_f", "hazard_g"))
```

```
## ...automatically assigning order attribute to some nodes...

## node W, order:1

## node A, order:2

## node dNt_1, order:3

## node dAt_1, order:4

## node dNt_2, order:5

## node dAt_2, order:6

## node dNt_3, order:7

## node dAt_3, order:8

## node dNt_4, order:9
## node dAt_4, order:10

## node dNt_5, order:11

## node dAt_5, order:12

## node dNt_6, order:13

## node dAt_6, order:14

## node dNt_7, order:15

## node dAt_7, order:16

## node dNt_8, order:17

## node dAt_8, order:18

## node dNt_9, order:19

## node dAt_9, order:20

## node dNt_10, order:21

## node dAt_10, order:22

## [1] "current list of user-defined vectorized functions: hazard_f,hazard_g"
```

```
dat <- sim(setD, n = 200)
```

```
## simulating observed dataset from the DAG object
```

```
head(dat)
```

```
##   ID         W A dNt_1 dAt_1 dNt_2 dAt_2 dNt_3 dAt_3 dNt_4 dAt_4 dNt_5 dAt_5
## 1  1 14.985488 0     0     0     0     0     0     0     0     0     0     0
## 2  2 12.195101 1     0     0     0     0     0     1    NA    NA    NA    NA
## 3  3  4.833145 0     0     0     0     1    NA    NA    NA    NA    NA    NA
## 4  4 20.974849 1     0     1    NA    NA    NA    NA    NA    NA    NA    NA
## 5  5  9.624771 1     0     0     0     0     1    NA    NA    NA    NA    NA
## 6  6 19.428912 0     1    NA    NA    NA    NA    NA    NA    NA    NA    NA
##   dNt_6 dAt_6 dNt_7 dAt_7 dNt_8 dAt_8 dNt_9 dAt_9 dNt_10 dAt_10
## 1     0     0     0     0     0     0     0     0      1     NA
## 2    NA    NA    NA    NA    NA    NA    NA    NA     NA     NA
## 3    NA    NA    NA    NA    NA    NA    NA    NA     NA     NA
## 4    NA    NA    NA    NA    NA    NA    NA    NA     NA     NA
## 5    NA    NA    NA    NA    NA    NA    NA    NA     NA     NA
## 6    NA    NA    NA    NA    NA    NA    NA    NA     NA     NA
```

```
dNt_data <- dat[, grep( "dNt", colnames(dat))]
survival_times <- apply(dNt_data,1, function(row){
  result = time_points[!is.na(row) & row ==1]
  if(length(result)==0){
    return(NA)
  }
  return(result)
})

dAt_data <- dat[, grep( "dAt", colnames(dat))]
censor_times <- apply(dAt_data,1, function(row){
  result = time_points[!is.na(row) & row ==1]
  if(length(result)==0){
    return(max(time_points))
  }
  return(result)
})

#NA means censoring happened first
head(data.frame(cbind(survival_times, censor_times)))
```

```
##   survival_times censor_times
## 1             10           10
## 2             NA            3
## 3             NA            2
## 4             NA            1
## 5              3           10
## 6              1           10
```

## Estimating hazards

```
library(sl3)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
library(simcausal)
D <- DAG.empty()
D <- D +
  node("W", distr = "runif", min =1, max = 25) +
  node("A", distr = "rbinom", size = 1, prob = 0.2 + W/50 ) +
  node("T", distr = "rexp", rate = 0.05*(1 - 0.5*A + 0.3*W) )+
  node("C", distr = "rexp",  rate = 0.03*(1 - 0.4*A + 0.2*W)) +
  node("Delta", distr = "rconst", const = as.numeric(T<=C)  ) +
  node("Ttilde", distr = "rconst", const = round(min(T,C,10) +1  ))
setD <- set.DAG(D)
```

```
## ...automatically assigning order attribute to some nodes...

## node W, order:1

## node A, order:2

## node T, order:3

## node C, order:4

## node Delta, order:5

## node Ttilde, order:6

## as.numeric(T <= C)
## min(cbind_mod(T, C, 10))
```

```
# Lets draw 200 samples
dat <- sim(setD, n = 200)
```

```
## simulating observed dataset from the DAG object

## as.numeric(T <= C)
## min(cbind_mod(T, C, 10))
```

```r
head(dat)
```

```
##   ID         W A         T         C Delta Ttilde
## 1  1  7.243682 0  8.882407 23.941410     1     10
## 2  2 22.321508 1  4.950192  3.827694     0      5
## 3  3  4.198056 0 10.477247 15.039372     1     11
## 4  4 12.385979 0  2.183238  8.753820     1      3
## 5  5 15.535858 1  4.661168  1.571518     0      3
## 6  6  9.605092 0 26.735238  3.877388     0      5
```

```r
data <- dat[,c("ID", "W", "A", "Delta", "Ttilde")]
head(data)
```

```
##   ID         W A Delta Ttilde
## 1  1  7.243682 0     1     10
## 2  2 22.321508 1     0      5
## 3  3  4.198056 0     1     11
## 4  4 12.385979 0     1      3
## 5  5 15.535858 1     0      3
## 6  6  9.605092 0     0      5
```

```r
time_range <- range(data$Ttilde)
times <- seq.int(min(time_range), max(time_range))
list_of_data <- lapply(times, function(t) {
  #copy data
  data <- data
  data$t <- t
  # Outcome is 1 if and only if person dies at time t
  data$dN <- as.numeric(data$Ttilde == t & data$Delta == 1)
  data$dC <- as.numeric(data$Ttilde == t & data$Delta == 0)
  return(data)
})


data_pooled <- do.call(rbind, list_of_data)
head(data_pooled)
```

```
##   ID         W A Delta Ttilde t dN dC
## 1  1  7.243682 0     1     10 1  0  0
## 2  2 22.321508 1     0      5 1  0  0
## 3  3  4.198056 0     1     11 1  0  0
## 4  4 12.385979 0     1      3 1  0  0
## 5  5 15.535858 1     0      3 1  0  0
## 6  6  9.605092 0     0      5 1  0  0
```

```r
# Estimate survival hazard
covariates <- c("W", "A", "t")
outcome <-  "dN"
# Create sl3 regression task. The task essentially keeps track of the information needed for regression
# E.g. data, covariate names, outcome names, time and ids for cross validation.
task_survival <- sl3_Task$new(data_pooled, covariates = covariates, outcome = outcome, outcome_type = "

covariates <- c("W", "A", "t")
outcome <-  "dC"
task_censoring <- sl3_Task$new(data_pooled, covariates = covariates, outcome = outcome, outcome_type =
head(task_survival$data)
```

```
##              W A t dN ID
## 1:  7.243682 0 1  0  1
## 2: 22.321508 1 1  0  2
## 3:  4.198056 0 1  0  3
## 4: 12.385979 0 1  0  4
## 5: 15.535858 1 1  0  5
## 6:  9.605092 0 1  0  6
```

```
# For estimation, we only keep rows corresponding to times where the person is still alive/not censored
keep_surv <-  data_pooled$t <= data_pooled$Ttilde
data_pooled_training_surv <- data_pooled[keep_surv,]
covariates <- c("W", "A", "t")
outcome <-  "dN"
task_survival_training <- sl3_Task$new(data_pooled_training_surv, covariates = covariates, outcome = ou
# or cleaner
task_survival_training <- task_survival[keep_surv]
head(task_survival_training$data)
```

```
##              W A t dN ID
## 1:  7.243682 0 1  0  1
## 2: 22.321508 1 1  0  2
## 3:  4.198056 0 1  0  3
## 4: 12.385979 0 1  0  4
## 5: 15.535858 1 1  0  5
## 6:  9.605092 0 1  0  6
```

```
tail(task_survival_training$data)
```

```
##              W A  t dN  ID
## 1:  3.238178 0 11  1 138
## 2:  2.599679 0 11  1 139
## 3:  2.222611 0 11  1 157
## 4:  2.852906 1 11  1 174
## 5: 14.274613 1 11  1 189
## 6:  1.574208 0 11  0 197
```

```
dim(task_survival_training$data)
```

```
## [1] 876   5
```

```
# For estimation, we only keep rows corresponding to times where the person is still alive/not censored
# We usually assume that we observe whether someone is alive before they get censored Therefore, if som
### This done so we have a time ordering in the likelihood. Only needed for discrete time.
keep_censoring <-  data_pooled$t <= data_pooled$Ttilde & (data_pooled$dN != 1)
task_censoring_training <- task_censoring[keep_censoring]
head(task_censoring_training$data)
```

```
##              W A t dC ID
## 1:  7.243682 0 1  0  1
## 2: 22.321508 1 1  0  2
## 3:  4.198056 0 1  0  3
## 4: 12.385979 0 1  0  4
## 5: 15.535858 1 1  0  5
## 6:  9.605092 0 1  0  6
```

```
tail(task_censoring_training$data)
```

```
##             W A  t dC  ID
## 1: 1.574208 0 10  0 197
## 2: 2.670341 0 11  1   9
## 3: 6.862854 0 11  1  28
## 4: 4.829887 1 11  1  77
## 5: 1.123334 0 11  1 101
## 6: 1.574208 0 11  1 197
```

```
dim(task_censoring_training$data)
```

```
## [1] 739   5
```

```
# We now have 4 tasks, 2 for estimation and 2 for prediction.
# Removing rows is needed so that the estimation is correct.
# However we want the hazard predictions at all time points for all people, so dont remove rows.
task_survival
```

```
## A sl3 Task with 2200 obs and these nodes:
## $covariates
## [1] "W" "A" "t"
##
## $outcome
## [1] "dN"
##
## $id
## [1] "ID"
##
## $weights
## NULL
##
## $offset
## NULL
##
## $time
## [1] "t"
```

```
task_survival_training
```

```
## A sl3 Task with 876 obs and these nodes:
## $covariates
## [1] "W" "A" "t"
##
## $outcome
## [1] "dN"
##
## $id
## [1] "ID"
##
## $weights
## NULL
##
## $offset
## NULL
##
## $time
## [1] "t"
```

```
task_censoring
```

```
## A sl3 Task with 2200 obs and these nodes:
## $covariates
## [1] "W" "A" "t"
##
## $outcome
## [1] "dC"
##
## $id
## [1] "ID"
##
## $weights
## NULL
##
## $offset
## NULL
##
## $time
## [1] "t"
```

```
task_censoring_training
```

```
## A sl3 Task with 739 obs and these nodes:
## $covariates
## [1] "W" "A" "t"
##
## $outcome
## [1] "dC"
##
## $id
## [1] "ID"
##
## $weights
## NULL
##
## $offset
## NULL
##
## $time
## [1] "t"
```

```
#survival training
# Now, we need a binomial learner.
lrnr_glm <- make_learner(Lrnr_glm, family = binomial())
# train using training task
lrnr_glm_trained <- lrnr_glm$train(task_survival_training)
# predict using full task
hazard_survival_preds <- lrnr_glm_trained$predict(task_survival)
long_preds <- (data.frame(id = task_survival$id, t = task_survival$time, preds = hazard_survival_preds)
print(head(long_preds))
```

```
##   id t      preds
## 1  1 1 0.08158023
## 2  2 1 0.17333385
## 3  3 1 0.06742447
## 4  4 1 0.11170278
## 5  5 1 0.11702910
## 6  6 1 0.09436690
```

```
wide_preds <- reshape(long_preds, timevar = "t", idvar = "id", direction = "wide")
colnames(wide_preds) <- c("id", times)
head(wide_preds)
```

```
##   id          1          2          3         4         5         6         7
## 1  1 0.08158023 0.09550785 0.11152451 0.1298416 0.1506570 0.1741415 0.2004230
## 2  2 0.17333385 0.19952289 0.22857476 0.2604803 0.2951354 0.3323293 0.3717384
## 3  3 0.06742447 0.07914364 0.09269727 0.1082990 0.1261616 0.1464863 0.1694504
## 4  4 0.11170278 0.13004485 0.15088716 0.1744002 0.2007113 0.2298865 0.2619129
## 5  5 0.11702910 0.13611175 0.15775010 0.1821033 0.2092819 0.2393298 0.2722069
## 6  6 0.09436690 0.11021554 0.12834868 0.1489657 0.1722401 0.1983035 0.2272282
##           8         9        10        11
## 1 0.2295684 0.2615655 0.2963072 0.3335788
## 2 0.4129305 0.4553787 0.4984853 0.5416144
## 3 0.1951913 0.2237888 0.2552472 0.2894787
## 4 0.2966822 0.3339786 0.3734739 0.4147314
## 5 0.3077729 0.3457780 0.3858603 0.4275523
## 6 0.2590089 0.2935460 0.3306335 0.3699529
```

```
# It looks like the hazard probability increases with time.
survival_preds <- data.frame(cbind(wide_preds[,1], t(apply(1 -wide_preds[,-1],1, cumprod))))
colnames(survival_preds) <- c("id", times)
head(survival_preds)
```

```
##   id         1         2         3         4         5         6         7
## 1  1 0.9184198 0.8307035 0.7380597 0.6422288 0.5454726 0.4504832 0.3601960
## 2  2 0.8266662 0.6617273 0.5104732 0.3775050 0.2660899 0.1776604 0.1116172
## 3  3 0.9325755 0.8587681 0.7791627 0.6947801 0.6071256 0.5181900 0.4303825
## 4  4 0.8882972 0.7727787 0.6561763 0.5417390 0.4330059 0.3334637 0.2461252
## 5  5 0.8829709 0.7627882 0.6424583 0.5254645 0.4154943 0.3160541 0.2300220
## 6  6 0.9056331 0.8058183 0.7023926 0.5977602 0.4948019 0.3966809 0.3065438
##            8          9         10          11
## 1 0.27750637 0.20492027 0.14420092 0.096098540
## 2 0.06552707 0.03568744 0.01789778 0.008204083
## 3 0.34637556 0.26886060 0.20023467 0.142270995
## 4 0.17310426 0.11529114 0.07223290 0.042275649
## 5 0.15922749 0.10417012 0.06397501 0.036622342
## 6 0.22714626 0.16046839 0.10741217 0.067674723
```

```
#censoring training
# Now, we need a binomial learner.
lrnr_glm <- make_learner(Lrnr_glm)
# train using training task
lrnr_glm_trained <- lrnr_glm$train(task_censoring_training)
# predict using full task
hazard_censoring_preds <- lrnr_glm_trained$predict(task_censoring)
long_preds <- (data.frame(id = task_censoring$id, t = task_censoring$time, preds = hazard_censoring_pre
print(head(long_preds))
```

```
##   id t      preds
## 1  1 1 0.04891348
## 2  2 1 0.08873778
## 3  3 1 0.03998976
## 4  4 1 0.06839009
## 5  5 1 0.05739248
## 6  6 1 0.05710120
```

```
wide_preds <- reshape(long_preds, timevar = "t", idvar = "id", direction = "wide")
colnames(wide_preds) <- c("id", times)
head(wide_preds)
```

```
##   id          1          2          3          4          5          6
## 1  1 0.04891348 0.05731762 0.06706393 0.07832978 0.09130295 0.10617723
## 2  2 0.08873778 0.10324165 0.11980445 0.13861367 0.15983966 0.18362313
## 3  3 0.03998976 0.04693633 0.05502042 0.06440280 0.07525768 0.08777049
## 4  4 0.06839009 0.07985965 0.09306061 0.10818714 0.12543237 0.14497963
## 5  5 0.05739248 0.06715060 0.07842980 0.09141788 0.10630870 0.12329589
## 6  6 0.05710120 0.06681330 0.07804058 0.09097058 0.10579701 0.12271367
##           7         8         9        10        11
## 1 0.1231463 0.1423953 0.1640901 0.1883642 0.2153043
## 2 0.2100607 0.2391894 0.2709717 0.3052826 0.3419005
## 3 0.1021340 0.1185426 0.1371845 0.1582319 0.1818279
## 4 0.1669915 0.1915964 0.2188741 0.2488400 0.2814303
## 5 0.1425645 0.1642801 0.1885760 0.2155383 0.2451908
## 6 0.1419060 0.1635404 0.1877515 0.2146271 0.2441933
```

```
# It looks like the hazard probability increases with time.
censoring_survival_preds <- data.frame(cbind(wide_preds[,1], t(apply(1 -wide_preds[,-1],1, cumprod))))
colnames(censoring_survival_preds) <- c("id", times)
head(censoring_survival_preds)
```

```
##   id         1         2         3         4         5         6         7
## 1  1 0.9510865 0.8965725 0.8364448 0.7709263 0.7005384 0.6261572 0.5490483
## 2  2 0.9112622 0.8171820 0.7192800 0.6195779 0.5205448 0.4249607 0.3356932
## 3  3 0.9600102 0.9149509 0.8646099 0.8089266 0.7480487 0.6823921 0.6126966
## 4  4 0.9316099 0.8572119 0.7774392 0.6933303 0.6063642 0.5184538 0.4318764
## 5  5 0.9426075 0.8793109 0.8103467 0.7362665 0.6579950 0.5768669 0.4946262
## 6  6 0.9428988 0.8799006 0.8112327 0.7374344 0.6594160 0.5784966 0.4964045
##           8         9        10        11
## 1 0.4708664 0.3936019 0.3194614 0.2506800
## 2 0.2553990 0.1861931 0.1293516 0.0851262
## 3 0.5400660 0.4659773 0.3922448 0.3209238
## 4 0.3491304 0.2727148 0.2048525 0.1472008
## 5 0.4133689 0.3354175 0.2631222 0.1986070
## 6 0.4152223 0.3372637 0.2648777 0.2001964
```

```
#survival
#Lets use xgboost now with depth 3
lrnr_xgboost <- make_learner(Lrnr_xgboost, max.depth = 3)
lrnr_xgboost_trained <- lrnr_xgboost$train(task_survival_training)
hazard_survival_preds <- lrnr_xgboost_trained$predict(task_survival)
long_preds <- (data.frame(id = task_survival$id, t = task_survival$time, preds = hazard_survival_preds)
print(head(long_preds))
```

```
##   id t      preds
## 1  1 1 0.03690691
## 2  2 1 0.15022910
## 3  3 1 0.02784358
## 4  4 1 0.15780196
## 5  5 1 0.05994054
## 6  6 1 0.12961829
```

```
wide_preds <- reshape(long_preds, timevar = "t", idvar = "id", direction = "wide")
colnames(wide_preds) <- c("id", times)
head(wide_preds)
```

```
##   id          1         2         3         4         5         6         7
## 1  1 0.03690691 0.1498530 0.1498530 0.1717766 0.1592090 0.1592090 0.1755167
## 2  2 0.15022910 0.2102288 0.2102288 0.1794837 0.2104129 0.2104129 0.2104129
## 3  3 0.02784358 0.1037678 0.1037678 0.1199001 0.1106210 0.1106210 0.1226780
## 4  4 0.15780196 0.2601473 0.2601473 0.2990735 0.3170545 0.3170545 0.3170545
## 5  5 0.05994054 0.1568912 0.1568912 0.1576724 0.2037124 0.2037124 0.2037124
## 6  6 0.12961829 0.1081733 0.1081733 0.1248951 0.1499119 0.1499119 0.1499119
##           8         9        10        11
## 1 0.1755167 0.3292992 0.2919398 0.5120776
## 2 0.2104129 0.4367060 0.3943253 0.6559693
## 3 0.1226780 0.1203666 0.1203666 0.7061907
## 4 0.3170545 0.5776691 0.5345914 0.6939998
## 5 0.2037124 0.4297926 0.3876212 0.6495877
## 6 0.1499119 0.2891274 0.2545955 0.4650696
```

```
survival_preds <- data.frame(cbind(wide_preds[,1], t(apply(1 -wide_preds[,-1],1, cumprod))))
colnames(survival_preds) <- c("id", times)
head(survival_preds)
```

```
##   id         1         2         3         4         5         6         7
## 1  1 0.9630931 0.8187707 0.6960755 0.5765060 0.4847210 0.4075491 0.3360174
## 2  2 0.8497709 0.6711246 0.5300349 0.4349023 0.3433933 0.2711389 0.2140878
## 3  3 0.9721564 0.8712778 0.7808672 0.6872411 0.6112179 0.5436044 0.4769161
## 4  4 0.8421980 0.6231025 0.4610040 0.3231299 0.2206801 0.1507125 0.1029284
## 5  5 0.9400595 0.7925724 0.6682247 0.5628641 0.4482017 0.3568974 0.2841930
## 6  6 0.8703817 0.7762296 0.6922623 0.6058021 0.5149852 0.4377827 0.3721539
##            8          9         10          11
## 1 0.27704075 0.18581146 0.13156570 0.064193858
## 2 0.16904094 0.09521975 0.05767219 0.019841002
## 3 0.41840898 0.36804651 0.32374599 0.095119581
## 4 0.07029453 0.02968755 0.01381684 0.004227955
## 5 0.22629932 0.12903754 0.07901986 0.027689530
## 6 0.31636358 0.22489420 0.16763715 0.089674210
```

```
# And hal9001 (highly adaptive lasso)


lrnr_hal9001<- make_learner(Lrnr_hal9001, max_degree = 2)
lrnr_hal9001_trained <- lrnr_hal9001$train(task_survival_training)
hazard_survival_preds <- lrnr_hal9001_trained$predict(task_survival)
long_preds <- (data.frame(id = task_survival$id, t = task_survival$time, preds = hazard_survival_preds)
print(head(long_preds))
```

```
##   id t      preds
## 1  1 1 0.09545087
## 2  2 1 0.10839041
## 3  3 1 0.09545087
## 4   4 1 0.09646455
## 5   5 1 0.09646455
## 6   6 1 0.09545087
```

```
wide_preds <- reshape(long_preds, timevar = "t", idvar = "id", direction = "wide")
colnames(wide_preds) <- c("id", times)
head(wide_preds)
```

```
##   id          1         2         3         4         5         6         7
## 1  1 0.09545087 0.1533482 0.1533482 0.1533482 0.1533482 0.1533482 0.1533482
## 2  2 0.10839041 0.2101678 0.2101678 0.2101678 0.2248603 0.2248603 0.2248603
## 3  3 0.09545087 0.1266970 0.1266970 0.1266970 0.1266970 0.1266970 0.1266970
## 4  4 0.09646455 0.1894228 0.1894228 0.1894228 0.2030378 0.2030378 0.2030378
## 5  5 0.09646455 0.1894228 0.1894228 0.1894228 0.2030378 0.2030378 0.2030378
## 6  6 0.09545087 0.1533482 0.1533482 0.1533482 0.1572914 0.1572914 0.1572914
##           8         9        10        11
## 1 0.1533482 0.2800161 0.2800161 0.7311340
## 2 0.2248603 0.3838195 0.3838195 0.8132697
## 3 0.1266970 0.1266970 0.1266970 0.5035703
## 4 0.2030378 0.3536077 0.3536077 0.7927442
## 5 0.2030378 0.3536077 0.3536077 0.7927442
## 6 0.1572914 0.2861157 0.2861157 0.7370013
```

```
survival_preds <- data.frame(cbind(wide_preds[,1], t(apply(1 -wide_preds[,-1],1, cumprod))))
colnames(survival_preds) <- c("id", times)
head(survival_preds)
```

```
##   id         1         2         3         4         5         6         7
## 1  1 0.9045491 0.7658381 0.6483982 0.5489675 0.4647843 0.3935104 0.3331663
## 2  2 0.8916096 0.7042220 0.5562172 0.4393182 0.3405330 0.2639606 0.2046064
## 3  3 0.9045491 0.7899454 0.6898617 0.6024582 0.5261286 0.4594696 0.4012562
## 4  4 0.9035354 0.7323853 0.5936548 0.4812031 0.3835006 0.3056355 0.2435799
## 5  5 0.9035354 0.7323853 0.5936548 0.4812031 0.3835006 0.3056355 0.2435799
## 6  6 0.9045491 0.7658381 0.6483982 0.5489675 0.4626196 0.3898535 0.3285329
##           8          9         10         11
## 1 0.2820758 0.20309006 0.14622158 0.03931402
## 2 0.1585985 0.09772531 0.06021644 0.01124423
## 3 0.3504182 0.30602126 0.26724927 0.13267047
## 4 0.1941240 0.12548026 0.08110948 0.01681041
## 5 0.1941240 0.12548026 0.08110948 0.01681041
## 6 0.2768575 0.19764423 0.14109513 0.03710784
```

```
# We can also do the estimation without sl3
covariates <- c("W", "A", "t")
outcome <-  "dN"
id <- data_pooled$ID

data_pooled_glm <- data_pooled[, c(covariates, outcome)]
data_pooled_training_surv_glm <- data_pooled_training_surv[, c(covariates, outcome)]
head(data_pooled)
```

```
##   ID         W A Delta Ttilde t dN dC
## 1  1  7.243682 0     1     10 1  0  0
## 2  2 22.321508 1     0      5 1  0  0
## 3  3  4.198056 0     1     11 1  0  0
## 4  4 12.385979 0     1      3 1  0  0
## 5  5 15.535858 1     0      3 1  0  0
## 6  6  9.605092 0     0      5 1  0  0
```

```r
head(data_pooled_training_surv)
```

```
##   ID         W A Delta Ttilde t dN dC
## 1  1  7.243682 0     1     10 1  0  0
## 2  2 22.321508 1     0      5 1  0  0
## 3  3  4.198056 0     1     11 1  0  0
## 4  4 12.385979 0     1      3 1  0  0
## 5  5 15.535858 1     0      3 1  0  0
## 6  6  9.605092 0     0      5 1  0  0
```

```r
# need to add intercpt manually
x_train = cbind(rep(1, nrow(data_pooled_training_surv_glm)), data_pooled_training_surv_glm[, covariates
colnames(x_train) <- c("intercept", covariates)
y_train = data_pooled_training_surv_glm[, outcome]
fit <- glm.fit(x_train, y_train, family = binomial())
x_pred = cbind(rep(1, nrow(data_pooled_glm)), data_pooled_glm[, covariates])
coefs <- fit$coefficients
coefs
```

```
##  intercept          W          A          t
## -3.0836198  0.0675968 -0.1603265  0.1729025
```

```r
preds <- data.frame(t = data_pooled_glm$t, id =id, preds = plogis(as.matrix(x_pred) %*% coefs ))
wide_preds <- reshape(preds, idvar = "id", timevar = "t", direction = "wide")
head(wide_preds)
```

```
##   id    preds.1    preds.2    preds.3   preds.4   preds.5   preds.6   preds.7
## 1  1 0.08158023 0.09550785 0.11152451 0.1298416 0.1506570 0.1741415 0.2004230
## 2  2 0.17333385 0.19952289 0.22857476 0.2604803 0.2951354 0.3323293 0.3717384
## 3  3 0.06742447 0.07914364 0.09269727 0.1082990 0.1261616 0.1464863 0.1694504
## 4  4 0.11170278 0.13004485 0.15088716 0.1744002 0.2007113 0.2298865 0.2619129
## 5  5 0.11702910 0.13611175 0.15775010 0.1821033 0.2092819 0.2393298 0.2722069
## 6  6 0.09436690 0.11021554 0.12834868 0.1489657 0.1722401 0.1983035 0.2272282
##     preds.8   preds.9  preds.10  preds.11
## 1 0.2295684 0.2615655 0.2963072 0.3335788
## 2 0.4129305 0.4553787 0.4984853 0.5416144
## 3 0.1951913 0.2237888 0.2552472 0.2894787
## 4 0.2966822 0.3339786 0.3734739 0.4147314
## 5 0.3077729 0.3457780 0.3858603 0.4275523
## 6 0.2590089 0.2935460 0.3306335 0.3699529
```

```r
survival_preds <- data.frame(cbind(wide_preds[,1], t(apply(1 -wide_preds[,-1],1, cumprod))))
colnames(survival_preds) <- c("id", times)
head(survival_preds)
```

```
##   id         1         2         3         4         5         6         7
## 1  1 0.9184198 0.8307035 0.7380597 0.6422288 0.5454726 0.4504832 0.3601960
## 2  2 0.8266662 0.6617273 0.5104732 0.3775050 0.2660899 0.1776604 0.1116172
## 3  3 0.9325755 0.8587681 0.7791627 0.6947801 0.6071256 0.5181900 0.4303825
## 4  4 0.8882972 0.7727787 0.6561763 0.5417390 0.4330059 0.3334637 0.2461252
## 5  5 0.8829709 0.7627882 0.6424583 0.5254645 0.4154943 0.3160541 0.2300220
## 6  6 0.9056331 0.8058183 0.7023926 0.5977602 0.4948019 0.3966809 0.3065438
##            8          9         10          11
## 1 0.27750637 0.20492027 0.14420092 0.096098540
## 2 0.06552707 0.03568744 0.01789778 0.008204083
## 3 0.34637556 0.26886060 0.20023467 0.142270995
## 4 0.17310426 0.11529114 0.07223290 0.042275649
## 5 0.15922749 0.10417012 0.06397501 0.036622342
## 6 0.22714626 0.16046839 0.10741217 0.067674723
```