Package 'mmcif'

July 17, 2022

Type Package

Title Mixed Multivariate Cumulative Incidence Functions

Version 0.1.1

Description Fits the mixed cumulative incidence functions model suggested by <doi:10.1093/biostatistics/kxx072> which decomposes within cluster dependence of risk and timing. The estimation method supports computation in parallel using a shared memory C++ implementation. A sandwich estimator of the covariance matrix is available. Natural cubic splines are used to provide a flexible model for the cumulative incidence functions.

License GPL (>= 3)

URL https://github.com/boennecd/mmcif

BugReports https://github.com/boennecd/mmcif/issues

Encoding UTF-8

RoxygenNote 7.2.0

Depends R (>= 3.5.0)

VignetteBuilder R.rsp

LinkingTo Rcpp, RcppArmadillo, testthat, psqn

Imports Rcpp, stats, alabama

Suggests testthat (>= 3.0.0), xml2, mvtnorm, R.rsp, mets

SystemRequirements C++17

Config/testthat/edition 3

NeedsCompilation yes

Author Benjamin Christoffersen [cre, aut]
(<https://orcid.org/0000-0002-7182-1346>),
Mark Clements [cph],
Alan Genz [cph],
Frank Bretz [cph],
Torsten Hothorn [cph],
R-core [cph]

Maintainer Benjamin Christoffersen <boennecd@gmail.com>

Repository CRAN

Date/Publication 2022-07-17 20:50:02 UTC

R topics documented:

`log_chol` *Computes the Log Cholesky Decomposition and the Inverse*

Description

Computes the log Cholesky decomposition and the inverse of it. The functions are provided as the log Cholesky decomposition is used in the parameterization of the covariance matrix.

Usage

```
log_chol(x)

log_chol_inv(x)
```

Arguments

`x`	A positive definite matrix or a vector with a log Cholesky decomposition.

Value

A numeric vector with the log Cholesky decomposition or a matrix with the inverse.

Examples

```
set.seed(1)
S <- drop(rWishart(1, 10, diag(10)))
log_chol(S)
stopifnot(isTRUE(all.equal(S, log_chol_inv(log_chol(S)))),
          (NCOL(S) * (NCOL(S) + 1L)) %/% 2L == length(log_chol(S)))
```

`mmcif_data` *Sets up an Object to Compute the Log Composite Likelihood*

Description

Sets up the R and C++ objects that are needed to evaluate the log composite likelihood. This reduces to a log likelihood when only clusters of size one or two are used.

Usage

```
mmcif_data(
  formula,
  data,
  cause,
  time,
  cluster_id,
  max_time,
  spline_df = 3L,
  left_trunc = NULL,
  ghq_data = NULL,
  strata = NULL,
  knots = NULL,
  boundary_quantiles = c(0.025, 0.975)
)
```

Arguments

`formula`	`formula` for covariates in the risk and trajectories.
`data`	`data.frame` with the covariate and outcome information.
`cause`	an integer vector with the cause of each outcome. If there are `n_causes` of outcome, then the vector should have values in `1:(n_causes + 1)` with `n_causes + 1` indicating censoring.
`time`	a numeric vector with the observed times.
`cluster_id`	an integer vector with the cluster id of each individual.
`max_time`	the maximum time after which there are no observed events. It is denoted by τ in the original article (Cederkvist et al., 2019).
`spline_df`	degrees of freedom to use for each spline in the cumulative incidence functions.
`left_trunc`	numeric vector with left-truncation times. `NULL` implies that there are not any individuals with left-truncation.
`ghq_data`	the default Gauss-Hermite quadrature nodes and weights to use. It should be a list with two elements called `"node"` and `"weight"`. A default is provided if `NULL` is passed.
`strata`	an integer vector or a factor vector with the strata of each individual. `NULL` implies that there are no strata.

knots
: A list of lists with knots for the splines. The inner lists needs to have elements called "knots" and "boundary_knots" which are passed to a function like ns. NULL yields defaults based on the quantiles of the observed event times. Note that the knots needs to be on the atanh((time - max_time / 2) / (max_time / 2)) scale.

boundary_quantiles
: two dimensional numerical vector with boundary quantile probabilities after which the natural cubic splines for the time transformations are restricted to be linear. Only relevant if knots is not NULL.

Value

An object of class mmcif which is needed for the other functions in the package.

References

Cederkvist, L., Holst, K. K., Andersen, K. K., & Scheike, T. H. (2019). *Modeling the cumulative incidence function of multivariate competing risks data allowing for within-cluster dependence of risk and timing*. Biostatistics, Apr 1, 20(2), 199-217.

See Also

mmcif_fit, mmcif_start_values and mmcif_sandwich.

Examples

```
if(require(mets)){
  # prepare the data
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
  })

  # select the DZ twins and re-code the status
  prt_use <- subset(prt, zyg == "DZ") |>
    transform(status = ifelse(status == 0, 3L, status))

  # randomly sub-sample
  set.seed(1)
  prt_use <- subset(
    prt_use, id %in% sample(unique(id), length(unique(id)) %/% 10L))

  mmcif_obj <- mmcif_data(
    ~ country - 1, prt_use, status, time, id, max_time,
    2L, strata = country)
}
```

`mmcif_fit` *Fits a Mixed Competing Risk Model*

Description

Fits mixed cumulative incidence functions model by maximizing the log composite likelihood function.

Usage

```
mmcif_fit(
  par,
  object,
  n_threads = 1L,
  control.outer = list(itmax = 100L, method = "nlminb", kkt2.check = FALSE, trace =
    FALSE),
  control.optim = list(eval.max = 10000L, iter.max = 10000L),
  ghq_data = object$ghq_data,
  ...
)
```

Arguments

`par`	numeric vector with parameters. This is using a log Cholesky decomposition for the covariance matrix.
`object`	an object from `mmcif_data`.
`n_threads`	the number of threads to use.
`control.outer, control.optim, ...`	arguments passed to `auglag`.
`ghq_data`	the Gauss-Hermite quadrature nodes and weights to use. It should be a list with two elements called `"node"` and `"weight"`. The argument can also be a list with lists with different sets of quadrature nodes. In this case, fits are successively made using the previous fit as the starting value. This may reduce the computation time by starting with fewer quadrature nodes.

Value

The output from `auglag`.

References

Cederkvist, L., Holst, K. K., Andersen, K. K., & Scheike, T. H. (2019). *Modeling the cumulative incidence function of multivariate competing risks data allowing for within-cluster dependence of risk and timing*. Biostatistics, Apr 1, 20(2), 199-217.

See Also

mmcif_data, mmcif_start_values and mmcif_sandwich.

Examples

```
if(require(mets)){
  # prepare the data
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
  })

  # select the DZ twins and re-code the status
  prt_use <- subset(prt, zyg == "DZ") |>
    transform(status = ifelse(status == 0, 3L, status))

  # randomly sub-sample
  set.seed(1)
  prt_use <- subset(
    prt_use, id %in% sample(unique(id), length(unique(id)) %/% 10L))

  n_threads <- 2L
  mmcif_obj <- mmcif_data(
    ~ country - 1, prt_use, status, time, id, max_time,
    2L, strata = country)

  # get the staring values
  start_vals <- mmcif_start_values(mmcif_obj, n_threads = n_threads)

  # estimate the parameters
  ests <- mmcif_fit(start_vals$upper, mmcif_obj, n_threads = n_threads)

  # show the estimated covariance matrix of the random effects
  tail(ests$par, 10L) |> log_chol_inv() |> print()

  # gradient is ~ zero
  mmcif_logLik_grad(
    mmcif_obj, ests$par, is_log_chol = TRUE, n_threads = n_threads) |>
    print()
}
```

---

mmcif_logLik *Evaluates the Log Composite Likelihood and its Gradient*

Description

Evaluates the log composite likelihood and its gradient using adaptive Gauss-Hermite quadrature.

Usage

```
mmcif_logLik(
  object,
  par,
  ghq_data = object$ghq_data,
  n_threads = 1L,
  is_log_chol = FALSE
)

mmcif_logLik_grad(
  object,
  par,
  ghq_data = object$ghq_data,
  n_threads = 1L,
  is_log_chol = FALSE
)
```

Arguments

object	an object from mmcif_data.
par	numeric vector with parameters. This is either using a log Cholesky decomposition for the covariance matrix or the covariance matrix.
ghq_data	the Gauss-Hermite quadrature nodes and weights to use. It should be a list with two elements called "node" and "weight". A default is provided if NULL is passed.
n_threads	the number of threads to use.
is_log_chol	logical for whether a log Cholesky decomposition is used for the covariance matrix or the full covariance matrix.

Value

A numeric vector with either the log composite likelihood or the gradient of it.

Examples

```
if(require(mets)){
  # prepare the data
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
```

```
})

# select the DZ twins and re-code the status
prt_use <- subset(prt, zyg == "DZ") |>
  transform(status = ifelse(status == 0, 3L, status))

# randomly sub-sample
set.seed(1)
prt_use <- subset(
  prt_use, id %in% sample(unique(id), length(unique(id)) %/% 10L))

n_threads <- 2L
mmcif_obj <- mmcif_data(
  ~ country - 1, prt_use, status, time, id, max_time,
  2L, strata = country)

# get the staring values
start_vals <- mmcif_start_values(mmcif_obj, n_threads = n_threads)

# compute the log composite likelihood and the gradient at the starting
# values
mmcif_logLik(
  mmcif_obj, start_vals$upper, is_log_chol = TRUE, n_threads = n_threads) |>
  print()
mmcif_logLik_grad(
  mmcif_obj, start_vals$upper, is_log_chol = TRUE, n_threads = n_threads) |>
  print()
}
```

`mmcif_pd_cond` *Computes Marginal Measures Using Two Observations*

Description

Computes bivariate figures such as conditional CIFs, survival probabilities, and hazards or bivariate CIFs, densities, and survival probabilities.

Usage

```
mmcif_pd_cond(
  par,
  object,
  newdata,
  cause,
  time,
  left_trunc = NULL,
  ghq_data = object$ghq_data,
  strata = NULL,
```

```
  which_cond,
  type_cond = "derivative",
  type_obs = "cumulative"
)

mmcif_pd_bivariate(
  par,
  object,
  newdata,
  cause,
  time,
  left_trunc = NULL,
  ghq_data = object$ghq_data,
  strata = NULL,
  use_log = FALSE,
  type = c("cumulative", "cumulative")
)
```

Arguments

par	numeric vector with the model parameters.
object	an object from mmcif_data.
newdata	a data.frame with data for the observations. It needs to have two rows.
cause	an integer vector with the cause of each outcome. If there are n_causes of outcome, then the vector should have values in 1:(n_causes + 1) with n_causes + 1 indicating censoring.
time	a numeric vector with the observed times.
left_trunc	numeric vector with left-truncation times. NULL implies that there are not any individuals with left-truncation.
ghq_data	the Gauss-Hermite quadrature nodes and weights to use. It should be a list with two elements called "node" and "weight". A default is provided if NULL is passed.
strata	an integer vector or a factor vector with the strata of each individual. NULL implies that there are no strata.
which_cond	an integer with value one or two for the index of the individual that is being conditioned on.
type_cond	a character for the type of outcome that is being conditioned on. "derivative" for the derivative of a CIF or "cumulative" for a CIF or the survival probability.
type_obs	a character the type of conditional measure. It can be "derivative" for the derivative of a CIF, "cumulative" for a CIF or the survival probability, and "hazard" for the hazard.
use_log	a logical for whether the returned output should be on the log scale.
type	a 2D character vector for the type of measures for each observation. The elements can be "derivative" for the derivative of a CIF or "cumulative" for a CIF or the survival probability.

Value

A numeric scalar with the requested quantity.

See Also

mmcif_pd_univariate and mmcif_fit.

Examples

```
if(require(mets)){
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
  })

  # select the DZ twins and re-code the status
  prt_use <- subset(prt, zyg == "DZ") |>
    transform(status = ifelse(status == 0, 3L, status))

  # Gauss Hermite quadrature nodes and weights from fastGHQuad::gaussHermiteData
  ghq_data <- list(
    node = c(-3.43615911883774, -2.53273167423279, -1.75668364929988, -1.03661082978951,
             -0.342901327223705, 0.342901327223705, 1.03661082978951, 1.75668364929988,
             2.53273167423279, 3.43615911883774),
   weight = c(7.6404328552326e-06, 0.00134364574678124, 0.0338743944554811, 0.240138611082314,
              0.610862633735326,0.610862633735326, 0.240138611082315, 0.033874394455481,
              0.00134364574678124, 7.64043285523265e-06))

  # setup the object for the computation
  mmcif_obj <- mmcif_data(
    ~ country - 1, prt_use, status, time, id, max_time,
    2L, strata = country, ghq_data = ghq_data)

  # previous estimates
  par <- c(0.727279974859164, 0.640534073288067, 0.429437766165371, 0.434367104339573,
           -2.4737847536253, -1.49576564624673, -1.89966050143904, -1.58881346649412,
           -5.5431198001029, -3.5328359024178, -5.82305147022587, -3.4531896212114,
           -5.29132887832377, -3.36106297109548, -6.03690322125729, -3.49516746825624,
           2.55000711185704, 2.71995985605891, 2.61971498736444, 3.05976391058032,
           -5.97173564860957, -3.37912051983482, -5.14324860374941, -3.36396780694965,
           -6.02337246348561, -3.03754644968859, -5.51267338700737, -3.01148582224673,
           2.69665543753264, 2.59359057553995, 2.7938341786374, 2.70689750644755,
           -0.362056555418564, 0.24088005091276, 0.124070380635372, -0.246152029808377,
         -0.0445628476462479, -0.911485513197845, -0.27911988106887, -0.359648419277058,
           -0.242711959678559, -6.84897302527358)

  # the test data we will use
  test_dat <- data.frame(
```

```
  country = factor(c("Norway", "Norway"), levels(prt_use$country)),
  status = c(1L, 2L), time = c(60, 75))

# probability that both experience the event prior to the two times
mmcif_pd_bivariate(
  par = par, object = mmcif_obj, newdata = test_dat, cause = status,
  strata = country, ghq_data = ghq_data, time = time, type =
    c("cumulative", "cumulative")) |>
  print()

# density that one experiences an event at the point and the other
# experiences an event prior to the point
mmcif_pd_bivariate(
  par = par, object = mmcif_obj, newdata = test_dat, cause = status,
  strata = country, ghq_data = ghq_data, time = time, type =
    c("derivative", "cumulative")) |>
  print()

# probability that both survive up to the passed points
mmcif_pd_bivariate(
  par = par, object = mmcif_obj, newdata = test_dat, cause = c(3L, 3L),
  strata = country, ghq_data = ghq_data, time = time, type =
    c("cumulative", "cumulative")) |>
  print()

# conditional hazard given that the other experiences an event prior to time
mmcif_pd_cond(
  par = par, object = mmcif_obj, newdata = test_dat, cause = status,
  strata = country, ghq_data = ghq_data, time = time, which_cond = 1L,
  type_cond = "cumulative", type_obs = "hazard") |>
  print()

# conditional CIF given that the other experiences an event prior to the
# time
mmcif_pd_cond(
  par = par, object = mmcif_obj, newdata = test_dat, cause = c(2L, 2L),
  strata = country, ghq_data = ghq_data, time = time, which_cond = 1L,
  type_cond = "cumulative", type_obs = "cumulative") |>
  print()

# same but given that the other experiences the event at the point
mmcif_pd_cond(
  par = par, object = mmcif_obj, newdata = test_dat, cause = c(2L, 2L),
  strata = country, ghq_data = ghq_data, time = time, which_cond = 1L,
  type_cond = "derivative", type_obs = "cumulative") |>
  print()
}
```

`mmcif_pd_univariate` *Computes Marginal Measures for One Observation*

Description

Computes the marginal cumulative incidence functions (CIF), marginal survival function or the derivative of the CIF.

Usage

```
mmcif_pd_univariate(
  par,
  object,
  newdata,
  cause,
  time,
  left_trunc = NULL,
  ghq_data = object$ghq_data,
  strata = NULL,
  use_log = FALSE,
  type = "cumulative"
)
```

Arguments

par	numeric vector with the model parameters.
object	an object from mmcif_data.
newdata	a data.frame with data for the observation. It needs to have one row.
cause	an integer vector with the cause of each outcome. If there are n_causes of outcome, then the vector should have values in 1:(n_causes + 1) with n_causes + 1 indicating censoring.
time	a numeric vector with the observed times.
left_trunc	numeric vector with left-truncation times. NULL implies that there are not any individuals with left-truncation.
ghq_data	the Gauss-Hermite quadrature nodes and weights to use. It should be a list with two elements called "node" and "weight". A default is provided if NULL is passed.
strata	an integer vector or a factor vector with the strata of each individual. NULL implies that there are no strata.
use_log	a logical for whether the returned output should be on the log scale.
type	a character for the type of measures for the observation. It can have value "derivative" for the derivative of a CIF or "cumulative" for a CIF or the survival probability.

Value

A numeric scalar with the requested quantity.

See Also

mmcif_pd_bivariate and mmcif_pd_cond.

Examples

```
if(require(mets)){
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
  })

  # select the DZ twins and re-code the status
  prt_use <- subset(prt, zyg == "DZ") |>
    transform(status = ifelse(status == 0, 3L, status))

  # Gauss Hermite quadrature nodes and weights from fastGHQuad::gaussHermiteData
  ghq_data <- list(
    node = c(-3.43615911883774, -2.53273167423279, -1.75668364929988, -1.03661082978951,
             -0.342901327223705, 0.342901327223705, 1.03661082978951, 1.75668364929988,
             2.53273167423279, 3.43615911883774),
   weight = c(7.6404328552326e-06, 0.00134364574678124, 0.0338743944554811, 0.240138611082314,
              0.610862633735326,0.610862633735326, 0.240138611082315, 0.033874394455481,
               0.00134364574678124, 7.64043285523265e-06))

  # setup the object for the computation
  mmcif_obj <- mmcif_data(
    ~ country - 1, prt_use, status, time, id, max_time,
    2L, strata = country, ghq_data = ghq_data)

  # previous estimates
  par <- c(0.727279974859164, 0.640534073288067, 0.429437766165371, 0.434367104339573,
           -2.4737847536253, -1.49576564624673, -1.89966050143904, -1.58881346649412,
           -5.5431198001029, -3.5328359024178, -5.82305147022587, -3.4531896212114,
           -5.29132887832377, -3.36106297109548, -6.03690322125729, -3.49516746825624,
           2.55000711185704, 2.71995985605891, 2.61971498736444, 3.05976391058032,
           -5.97173564860957, -3.37912051983482, -5.14324860374941, -3.36396780694965,
           -6.02337246348561, -3.03754644968859, -5.51267338700737, -3.01148582224673,
           2.69665543753264, 2.59359057553995, 2.7938341786374, 2.70689750644755,
           -0.362056555418564, 0.24088005091276, 0.124070380635372, -0.246152029808377,
         -0.0445628476462479, -0.911485513197845, -0.27911988106887, -0.359648419277058,
           -0.242711959678559, -6.84897302527358)

  # the test data we will use
  test_dat <- data.frame(country = factor("Norway", levels(prt_use$country)),
                         status = 2L)

  # compute the CIF
  mmcif_pd_univariate(
    par = par, object = mmcif_obj, newdata = test_dat, cause = status,
    strata = country, ghq_data = ghq_data, time = 75, type = "cumulative") |>
    print()
```

```
  # compute the derivative of the CIF
  mmcif_pd_univariate(
    par = par, object = mmcif_obj, newdata = test_dat, cause = status,
    strata = country, ghq_data = ghq_data, time = 75, type = "derivative") |>
    print()

  # compute the survival probability
  mmcif_pd_univariate(
    par = par, object = mmcif_obj, newdata = test_dat, cause = 3L,
    strata = country, ghq_data = ghq_data, time = 75, type = "cumulative") |>
    print()
}
```

`mmcif_sandwich` *Computes the Sandwich Estimator*

Description

Computes the sandwich estimator of the covariance matrix. The parameter that is passed is using the log Cholesky decomposition. The Hessian is computed using numerical differentiation with Richardson extrapolation to refine the estimate.

Usage

```
mmcif_sandwich(
  object,
  par,
  ghq_data = object$ghq_data,
  n_threads = 1L,
  eps = 0.01,
  scale = 2,
  tol = 1e-08,
  order = 3L
)
```

Arguments

`object`	an object from `mmcif_data`.
`par`	numeric vector with the parameters to compute the sandwich estimator at.
`ghq_data`	the Gauss-Hermite quadrature nodes and weights to use. It should be a list with two elements called `"node"` and `"weight"`. A default is provided if `NULL` is passed.
`n_threads`	the number of threads to use.
`eps`	determines the step size in the numerical differentiation using `max(sqrt(.Machine$double.eps), |par[i]| * eps)` for each parameter `i`.

`scale`	scaling factor in the Richardson extrapolation. Each step is smaller by a factor `scale`.
`tol`	relative convergence criteria in the extrapolation given by `max(tol, |g[i]| * tol)` with `g` being the gradient and for each parameter `i`.
`order`	maximum number of iteration of the Richardson extrapolation.

Value

The sandwich estimator along attributes called

- `"meat"` for the "meat" of the sandwich estimator.
- `"hessian"` for the Hessian of the log composite likelihood.
- `"res vcov"` which is the sandwich estimator where the last elements are the upper triangle of the covariance matrix of the random effects rather than the log Cholesky decomposition of the matrix.

References

Cederkvist, L., Holst, K. K., Andersen, K. K., & Scheike, T. H. (2019). *Modeling the cumulative incidence function of multivariate competing risks data allowing for within-cluster dependence of risk and timing*. Biostatistics, Apr 1, 20(2), 199-217.

See Also

`mmcif_fit` and `mmcif_data`.

Examples

```
if(require(mets)){
  # prepare the data
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
  })

  # select the DZ twins and re-code the status
  prt_use <- subset(prt, zyg == "DZ") |>
    transform(status = ifelse(status == 0, 3L, status))

  # randomly sub-sample
  set.seed(1)
  prt_use <- subset(
    prt_use, id %in% sample(unique(id), length(unique(id)) %/% 10L))

  n_threads <- 2L
  mmcif_obj <- mmcif_data(
```

```
  ~ country - 1, prt_use, status, time, id, max_time,
  2L, strata = country)

# get the staring values
start_vals <- mmcif_start_values(mmcif_obj, n_threads = n_threads)

# estimate the parameters
ests <- mmcif_fit(start_vals$upper, mmcif_obj, n_threads = n_threads)

# get the sandwich estimator
vcov_est <- mmcif_sandwich(
  mmcif_obj, ests$par, n_threads = n_threads, order = 2L)

# show the parameter estimates along with the standard errors
rbind(Estimate = ests$par,
      SE = sqrt(diag(vcov_est))) |>
  print()

# show the upper triangle of the covariance matrix and the SEs
rbind(`Estimate (vcov)` = tail(ests$par, 10) |> log_chol_inv() |>
        (\(x) x[upper.tri(x, TRUE)])() ,
      SE = attr(vcov_est, "res vcov") |> diag() |> sqrt() |> tail(10)) |>
  print()
}
```

`mmcif_start_values` *Finds Staring Values*

Description

Fast heuristic for finding starting values for the mixed cumulative incidence functions model.

Usage

```
mmcif_start_values(object, n_threads = 1L, vcov_start = NULL)
```

Arguments

`object`	an object from `mmcif_data`.
`n_threads`	the number of threads to use.
`vcov_start`	starting value for the covariance matrix of the random effects. `NULL` yields the identity matrix.

Value

A list with

- an element called `"full"` with the starting value where the last components are the covariance matrix.

- an element called "upper" the staring values where the covariance matrix is stored as a log Cholesky decomposition. This is used e.g. for optimization with mmcif_fit.

Examples

```
if(require(mets)){
  # prepare the data
  data(prt)

  # truncate the time
  max_time <- 90
  prt <- within(prt, {
    status[time >= max_time] <- 0
    time <- pmin(time, max_time)
  })

  # select the DZ twins and re-code the status
  prt_use <- subset(prt, zyg == "DZ") |>
    transform(status = ifelse(status == 0, 3L, status))

  # randomly sub-sample
  set.seed(1)
  prt_use <- subset(
    prt_use, id %in% sample(unique(id), length(unique(id)) %/% 10L))

  n_threads <- 2L
  mmcif_obj <- mmcif_data(
    ~ country - 1, prt_use, status, time, id, max_time,
    2L, strata = country)

  # get the staring values
  start_vals <- mmcif_start_values(mmcif_obj, n_threads = n_threads)

  # the starting values
  print(start_vals)
}
```

Index