The impact of the size and the number of clusters on prediction performance of the stratified and the conditional shared gamma frailty Cox proportional hazards models

R code for the illustrative case study

Daniele Giardiello, Edoardo Ratti, Peter C. Austin

Contents

Overview

We illustrate the analysis of the case study accompanying the simulation study about the impact of the size and the number of clusters on prediction performance of the stratified and the conditional shared gamma frailty Cox proportional hazards models. The case study is the insem data from the R package `parfm` described by the book of Duchateau and Janssen The Frailty Model (2008) (example 1.8)

Recap of predictions from shared gamma frailty Cox models

Two type of predictions may be possible using frailty:

- Conditional: predictions given the frailty/random effects
  For the shared gamma frailty:

$$S_i(t) = exp[-z_i e^{x_i\beta} H_o(t)] = exp[-H_0(t)e^{x_i\beta+u_i}] = exp[-z_i H(t)] = exp[-H(t)e^{u_i}]$$

  where

$$Z_i \sim \Gamma(\theta, \theta)$$

For details see the book written by David Collett Modelling Survival Data in Medical Research, 4^{th} edition, Chapter 10 (10.2.1-10.3), 306-307. book link

- Marginal: integrating over frailty/random effects
  For the shared gamma frailty the marginal predictions may be used when the new individual does not belong to any cluster used to develop the model or to estimate predictions integrating over all frailty

Conditional predictions can be used when the new individual belongs to a cluster used to develop the model.

$$S_i^*(t) = [1 + \theta^{-1} e^{\beta x_i} H_0(t)]^{-\theta}$$

Load packages

The following libraries are needed to achieve the following goals, if you have not them installed, please use install.packages(' ') (e.g. install.packages('survival')) or use the user-friendly approach if you are using RStudio.

```
# Use pacman to check whether packages are installed, if not load
if (!require("pacman")) install.packages("pacman")
library(pacman)

pacman::p_load(
  rio,
  stringr,
  knitr,
  kableExtra,
  Hmisc,
  lattice,
  riskRegression,
  polspline,
  gtsummary,
  tidyverse,
  grid,
  gridExtra,
  webshot,
  plotly,
  ggplot2,
  ggpubr,
  patchwork,
  rms,
  webshot2,
  parfm
)
```

Import useful functions

We import useful functions

```
# Useful functions
source(here::here("Functions/predict.coxph.gammafrail.R"))

# We report the function also below

#' @description
#'
#' Function to estimate conditional and marginal
#' survival probabilities at specified time horizons
#' for shared Gamma frailty Cox proportional hazards models
#'
#' @author Daniele Giardiello
#' @param model cox model using survival::coxph including frailty gamma term
```

```r
#' It is highly recommended to center all variables or define a reference value
#' for all predictors, especially the continuous predictors
#' Example: age50 = age - 50 or age_c = age - mean(age)
#' @param newdata newdata (for validation). This should have the same variables of the model.
#' Predictors with levels should be coded with exactly the same way and levels of the predictors
#' with factors used to develop the model
#' @param cluster cluster variable. Only one variable possible
#' @param times prediction horizon time(s)
#' @examples
#' # example code
#' data(lung, package = "survival")
#' lung$age_c <- lung$age - mean(lung$age)
#' cox_frailty_lung <- survival::coxph(Surv(time, status) ~ age_c + frailty(inst,
#' distribution = "gamma"),
#' data = lung,
#' x = TRUE,
#' y = TRUE,
#' model = TRUE,
#' ties = "breslow")
#' predict.coxph.gammafrail(model = cox_frailty_lung,
#' newdata = lung_sel,
#' cluster = "inst",
#' times = c(100, 200, 300))
#' @references Collett D - Modelling Survival data in Medical Research, 4th edition chapter 10 (formula


predict.coxph.gammafrail <- function(model,
                                     newdata,
                                     cluster,
                                     times) {

  # sanity check message warning
  on.exit(base::message("
  Sanity check warning:
  please center all variables before fitting the model or assign reference values
                for continuous variable.
                For example age50 = age - 50.
                newdata should contain the centered/referenced values"))

  # Stopping messages
  if (!inherits(model, "coxph"))
    stop("`model` must be a survival::coxph fit.
         Please center all variables before fitting the model or assign reference values
         for continuous variable. For example age50 = age - 50")

  if (missing(cluster))
    stop("Please supply the name of the cluster variable (character).")

  if (!is.character(cluster) || length(cluster) != 1L)
    stop("`cluster` must be a single character string giving the cluster column name.")

  if (!cluster %in% names(newdata))
    stop("`cluster` not found in `newdata`.")
```

```r
# Start function ----
# Step0:
# Formula with all terms including cluster as it is a fixed covariate
frm_allterms <- reformulate(base::all.vars(model$formula[[3]]))

# Select only the complete cases of fixed + frailty terms
newdata <- stats::model.frame(frm_allterms,
                              data = newdata,
                              na.action = na.omit)

# Step 1: survfit for
# a baseline cumulative hazard / survival

# Survfit of the coxph with frailty
sf_model <- survival::survfit(model)

# NOTE: the survival::survfit with frailty does not accept newdata
# thus, it is important to center all variables or give them a reference.
# In this setting survfit$cumhaz will directly represent our baseline

# Step 2: linear predictors of fixed effects

# Estimate linear predictors only
# for fixed-effect effects without frailties
# This will be also useful later when
# marginal predictions are estimated


# Since with clusters size < vs >= 5
# frailty terms are treated as coefficients or not,
# I prefer calculate lp for the only fixed terms only manually

# lp calculation for marginal
# save all model coefficients
# for cluster < 5, frailty terms are included in coefficients (not practical)
coefs <- model$coefficients

# Remove frailty named "gamma"
# NOTE/WARNING:
# if a variables called gamma it must be a problem
# to be adjusted
coefs_fixed <- coefs[!base::grepl("gamma", names(coefs))]

# create formula of fixed terms only
terms_vec <- attr(terms(model$formula),
                  "term.labels")

# Remove specials using grep/grepl
# NOTE: working in progress
specials <- c("frailty", "strata", "cluster", "offset", "tt")
fixed_terms <- terms_vec[!grepl(paste(specials, collapse = "|"), terms_vec)]
```

```r
formula_fixed <- stats::reformulate(fixed_terms,
                                    response = NULL)

# create design matrix of fixed term only
X_fixed <- model.matrix(formula_fixed,
                        newdata)

X_fixed <- X_fixed[, colnames(X_fixed) != "(Intercept)"]

# estimate linear predictor X*beta fixed term only
lp_fixed <- X_fixed %*% cbind(coefs_fixed)

# this should be equivalent to this (when cluster > 5)
# lp_nofrail <- cbind(predict(model,
#                             newdata = newdata,
#                             type = "lp"))


# Step 3: linear predictors for frailty terms

# NOTE/WARNING:
# for cluster < 5 predict.coxph(model, newdata) includes also frailty
# and frailty terms must not be included for marginal predictions.
# This leads to wrong estimations of marginal predictions
# and the corresponding performance metrics in riskRegression::Score()

# For frailty with not model$frail (where clusters < 5)
# add the model$frail and keep only fixed-effect coefficients
if(is.null(model$frail)) {
  coefs_model <- model$coefficients
  model$frail <- unname(coefs_model[base::grep("gamma", names(coefs_model))])
}

# Force cluster as a factor in the newdata
# number of clusters in the newdata

# otherwise a factor
# NOTE: for newdata with one cluster, factor does not work
# newdata[[cluster]] <- base::as.character(newdata[[cluster]])
 newdata[[cluster]] <- base::factor(newdata[[cluster]],
                                    levels = unique(newdata[[cluster]]))

# newdata[[cluster]] <- base::factor(newdata[[cluster]],
#                                    levels = 1:length(model$frail))
# Create the design matrix for frailty terms
all_terms <- base::all.vars(model$formula[[3]])
frailty_terms <- c(-1, all_terms[all_terms == cluster])
X_frail <- stats::model.matrix(stats::reformulate(frailty_terms),
                               response = NULL,
                               data = newdata)

# Frailty terms linear predictors
lp_frail <- X_frail %*% cbind(model$frail)
```

```r
  # Total linear predictor
  lp_total <- lp_fixed + lp_frail


  # Cumulative hazards
  # at fixed time horizons
  # H_0(t)
  cumhaz0_thor <- stats::approx(
    x = sf_model$time,
    y = sf_model$cumhaz,
    yleft = 0, # y values below min(x). Cumulative hazard zero if below min(time)
    yright = NA, # y values above max(x). Cumulative hazard NA if above max(time)
    xout = times,
    method = "constant",
    f = 0,
    rule = 1)$y

  # NOTE:
  # stats:approx should work fine since
  # all values from survival::survfit are unique
  # for the event time of development data

  # H(t)
  cumhaz_thor <- exp(lp_total) %*% cumhaz0_thor

  # conditional S(t)
  S_cond_t <- exp(-cumhaz_thor)

  # marginal S(t)
  theta_hat <- model$history[[1]]$theta
  S_marg_t <- (1 + theta_hat * (exp(lp_fixed) %*% cumhaz0_thor))^(-1/theta_hat)

  # Output
  # NOTE: to be better defined
  res <- list("conditional" = S_cond_t,
              "marginal" = S_marg_t)

  return(res)
  # Report only conditional predictions
  #return(S_cond_t)

}

# Other ad-hoc functions
# Score function with TryCatch
Score_tryCatch <- function(predictions,
                           df,
                           t_hors,
                           frm) {
  tryCatch(

  {
    sc <-
```

```
      riskRegression::Score(
        list("prediction_model" = predictions),
        data = df,
        formula = frm,
        times = t_hors,
        metrics = "auc",
        summary = "ipa")

    # Merge AUC + IPA into a single tidy table
    score_metrics <-
      base::merge(
        sc$AUC$score[, c("model", "times", "AUC")],
        sc$Brier$score[, c("model", "times", "IPA")],
        by = c("model", "times"),
        all = TRUE)

    score_metrics_models <-
      score_metrics %>%
      dplyr::filter(model %nin% c("Null model"))

    score_metrics_models},

  error = function(msg) {
         message(paste("Prediction not possible"))

    base::expand.grid(
      times = t_hors,
      model = c("prediction_model")) %>%
      base::transform(AUC = NA_real_,
                      IPA = NA_real_)
  })

}


# smoothed calibration function

calhaz_map <- function(time,
                       status,
                       covariates,
                       t_hors,
                       predictions) {

  tryCatch(
    {

  # Hazard regression
  calhaz_mod <- polspline::hare(data = time,
                                delta = status,
                                cov = covariates)

  # estimated cumulative probability
  pred_calhaz <- polspline::phare(t_hors,
```

```
                                  cov = covariates,
                                  fit = calhaz_mod)

  # ICI/E50/E90
  ICI <- mean(abs(pred_calhaz - predictions),
              na.rm = T)

  E50 <- median(abs(pred_calhaz - predictions),
                na.rm = T)

  E90 <- quantile(abs(pred_calhaz - predictions),
                  probs = .90,
                  na.rm = T)

  res <- c(ICI, E50, E90)
  res},

  error = function(msg) {
    message(paste("Calibration not possible"))
    rep(NA, 3)})

}
```

Import data

Insem data can be imported using the `parfm` R package. We also saved insem data in our repository in the .rds format.

```
# Import from the repository
insem <- readRDS(here::here("Data/insem.rds"))

# Import from parf package
data(insem,
     package = "parfm")
```

1. Descriptive statistics

We provide some descriptive statistics of the case study

```
insem_tab_01 <-
  insem %>%
  dplyr::select(Ureum,
                Protein,
                Parity,
                Heifer,
                Status)

label(insem_tab_01$Ureum) <- "Milk urem concentration (%) at the start of the lactation period"
label(insem_tab_01$Protein) <- "Protein concentration (%) at the start of the lactation period"
label(insem_tab_01$Parity) <- "The number of calvings"
label(insem_tab_01$Heifer) <- "Primiparous cow"
label(insem_tab_01$Status) <- "Inseminated cow"

units(insem_tab_01$Ureum) <- "%"
units(insem_tab_01$Protein) <- "%"
```

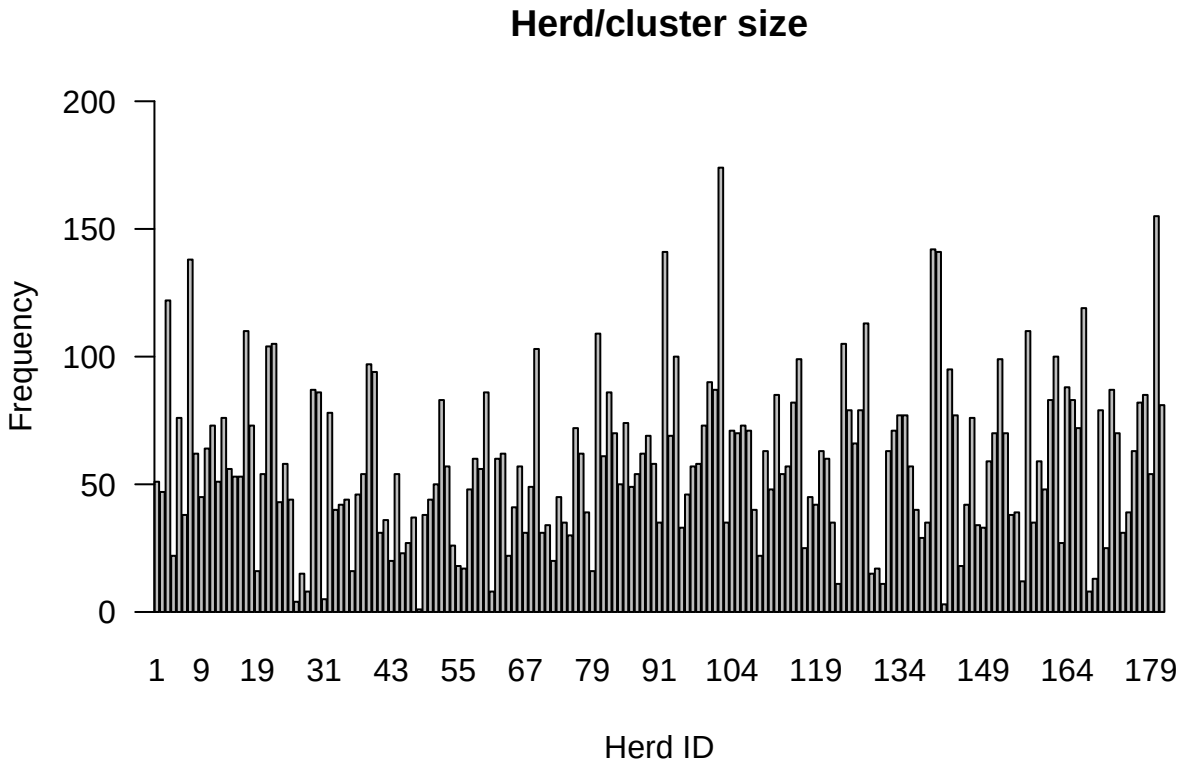
Characteristic	N = 10,513[1]
Milk urem concentration (%)	
Mean (SD)	2.58 (0.74)
Median (Range)	2.55 (0.54, 8.24)
Protein concentration (%)	
Mean (SD)	3.25 (0.34)
Median (Range)	3.21 (2.25, 6.48)
Number of calvings	
Mean (SD)	2 (2)
Median (Range)	2 (1, 14)
Primiparous cow	4,200 (40%)
Inseminated cows	9,939 (95%)

[1] n (%)

```
gtsummary_table1 <-
  gtsummary::tbl_summary(
    data = insem_tab_01 ,
    label = list(Parity ~ "Number of calvings",
                 Heifer ~ "Primiparous cow",
                 Ureum ~ "Milk urem concentration (%) ",
                 Protein ~ "Protein concentration (%)",
                 Status ~ "Inseminated cows"),
    type = all_continuous() ~ "continuous2",
    statistic = all_continuous() ~ c(
      "{mean} ({sd})",
      "{median} ({min}, {max})"
    ),
  )


gtsummary_table1 %>%
  gtsummary::as_gt()
```

```
# Cluster size distribution
table_cluster <-
  insem %>%
  dplyr::group_by(Herd) %>%
  dplyr::summarise(n = n()) %>%
  dplyr::arrange(n)

par(xaxs = "i",
    yaxs = "i",
    las = 1)
barplot(table(insem$Herd),
        ylab = "Frequency",
        xlab = "Herd ID",
        ylim = c(0, 200),
```

```
        main = "Herd/cluster size")
```

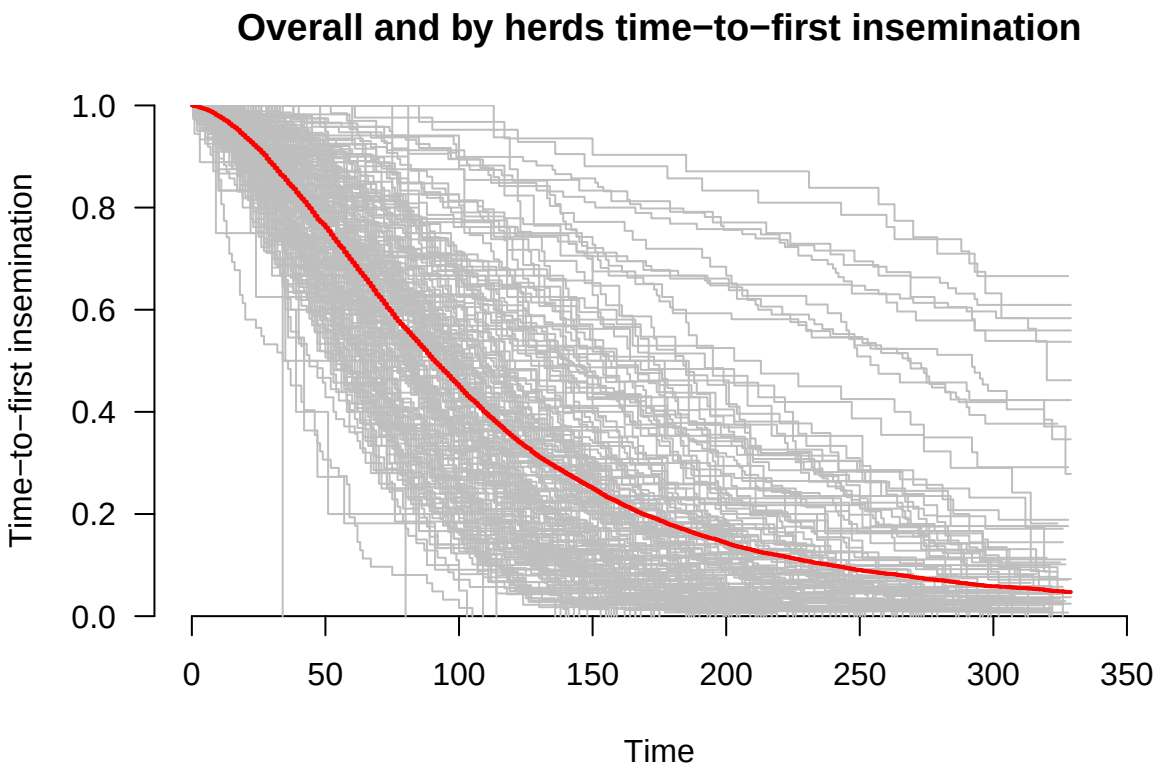



```
# kable(table_cluster) %>%
#   kableExtra::kable_styling("striped",
#                             position = "center") %>%
#     kableExtra::add_header_above(
#      c("Cluster size distribution" = 2))

# Summary cluster size
summary_cluster <-
  table_cluster %>%
  dplyr::summarise(min = min(n),
                   q25 = quantile(n, probs = .25),
                   mean = mean(n),
                   sd = sd(n),
                   median = median(n),
                   q75 = quantile(n, probs = .75),
                   max = max(n)) %>%
  round(., 1)

kable(summary_cluster) %>%
  kableExtra::kable_styling("striped") %>%
    kableExtra::add_header_above(
     c("Cluster size statistics" = 7))
```

Cluster size statistics

min	q25	mean	sd	median	q75	max
1	35	58.1	31.8	56	77	174

```r
# Event distribution per cluster
# kable(
#   addmargins(
#     table(insem$Herd,
#           insem$Status,
#           useNA = "ifany",
#           dnn = c("Cluster",
#                   "Status")))) %>%
#     kableExtra::kable_styling("striped") %>%
#   kableExtra::add_header_above(
#      c("Cluster" = 1,
#        "Status" =  2,
#        "Sum" = 1))

# survfit
sf_overall <- survival::survfit(Surv(Time,
                                     Status) ~ 1,
                                data = insem)

sf_cluster <- survival::survfit(Surv(Time,
                                     Status) ~ Herd,
                                data = insem)


par(xaxs = "i",
    yaxs = "i",
    las = 1)
plot(sf_cluster,
     col = "gray",
     xlim = c(0, 350),
     ylim = c(0, 1),
     bty = "n",
     xlab = "Time",
     ylab = "Time-to-first insemination",
     main = "Overall and by herds time-to-first insemination")
lines(sf_overall,
      lwd = 2,
      col = "red",
      conf.int = FALSE)
```




```
# Time quantiles
q_overall <- quantile(sf_overall,
                      probs = c(.25, .50, .75))$quantile
q_cluster <-
  quantile(sf_cluster,
           probs = c(.25, .50, .75))$quantile

q_surv <- rbind.data.frame("Overall" = q_overall,
                           q_cluster)

kable(q_surv[1, ]) %>%
  kableExtra::kable_styling("striped",
                            position = "center") %>%
  kableExtra::add_header_above(
    c(" " = 1,
      "Time quantile" = 3))
```

	Time quantile		
	25	50	75
Overall	52	91	150

2. Analysis

We provide the accuracy of predictions from the stratified and the shared gamma frailty Cox proportional hazards models in two scenarios

- Small cluster scenario: it includes herds with less than 50 cows
- Large cluster scenario: it includes herds with at least 50 cows

The accuracy of predictions are evaluated in terms of:

- Discrimination using time-varying AUC
- Calibration using O/E ratio, integrated calibration index (ICI), E50 and E90
- Overall performance using time-varying Index of Prediction Accuracy (IPA)

We use the Monte Carlo cross-validation procedure to assess prediction performances of the stratified and frailty Cox model.

- The sample was split into two parts to define derivation (70%) and validation samples (30%)
- The 70/30% random splits are repeated 100 times
- The repeated random splits are stratified by herds(i.e., the cluster) to obtain the same herds in both the derivation and validation samples

We excluded herds with less than 10 cows since it was not possible to obtain sufficient cows in both derivation and validation sample.

The mean of the performances metrics across the 100 random validation samples with the corresponding standard deviation are reported for each scenario and at each prediction time horizon to assess the prediction performances of the stratified and the frailty Cox models.

The fixed time horizons are defined based on the 25^{th}, 50^{th} and 75^{th} quantile fo event time overall distribution using the Kaplan-Meier distribution which corresponds to 51, 91, 150 days, respectively.

```r
# time horizons
time_hors <- c(52, 91, 150)

# reference values for continuous
# predictors
insem_ref <-
  insem %>%
  dplyr::mutate(
    Ureum_ref = Ureum - 2.5,
    Protein_ref = Protein - 2.5,
    Parity_ref = Parity - 2.5)


# Herds >= 50 cows
table_insem_50 <-
  insem_ref %>%
  dplyr::count(Herd) %>%
  dplyr::filter(n < 50 & n >= 10)

df_insem_small <-
  insem_ref %>%
  dplyr::filter(Herd %in% table_insem_50$Herd)

# Repeat B times
B <- 110
df_tbl_insem_ext <- do.call(rbind,
                        lapply(seq(1:B),
                               function(k) {
                                 cbind(id = k, df_insem_small)}))
# Nested tibble
set.seed(20250924)
df_tbl_insem <-
  df_tbl_insem_ext %>%
  dplyr::group_nest(id) %>%
  dplyr::mutate(
    # Splitting
    split = purrr::map(.x = data,
                       ~ rsample::initial_split(.x,
                                                strata = Herd,
                                                prop = 0.7,
                                                pool = 0.001)),
    # Development
    development = purrr::map(.x = split,
                             ~ rsample::training(.x)),

    # Validation
    validation = purrr::map(.x = split,
                             ~ rsample::testing(.x)),

    # Check
    unique_cluster_dev = purrr::map(.x = development,
                                    ~ unique(.x$Herd)),
```

```
    unique_cluster_val = purrr::map(.x = validation,
                                    ~ unique(.x$Herd)),

    check = purrr::map2(.x = unique_cluster_dev,
                        .y = unique_cluster_val,
                        ~ all.equal(.x, .y)))


# table(do.call(rbind, df_tbl_insem$check))
rm(df_tbl_insem_ext)
# gc()
# check
# table(do.call(rbind, df_tbl_insem$check))
# all TRUE

# Rewrite B
B <- 100

# Models + predictions + performances
# tictoc::tic()
df_tbl_insem <-
  df_tbl_insem %>%
  dplyr::filter(check == TRUE) %>%
  dplyr::select(data,
                development,
                validation) %>%

  dplyr::mutate(
    cox_strat = purrr::map(.x = development,
                           ~ survival::coxph(Surv(Time, Status) ~
                                               Heifer +
                                               Parity_ref +
                                               Ureum_ref +
                                               Protein_ref +
                                               strata(Herd),
                                             data = .x,
                                             x = T,
                                             y = T,
                                             ties = "breslow")),

    cox_frail = purrr::map(.x = development,
                           ~ survival::coxph(Surv(Time, Status) ~
                                               Heifer +
                                               Parity_ref +
                                               Ureum_ref +
                                               Protein_ref +
                                               frailty(Herd),
                                             data = .x,
                                             x = T,
                                             y = T,
                                             ties = "breslow")),

    # predictions at t25/t50/t75
```

```
# stratified cox
pred_strat = purrr::map2(.x = cox_strat,
                         .y = validation,
                         ~ 1 - pec::predictSurvProb(.x,
                                                    newdata = .y,
                                                    times = time_hors)),
# Extremes to make calibration suitable
pred_strat = purrr::map(pred_strat,
                        function(x) {
                          x <- ifelse(x == 0, 0.000001, x)
                          x <- ifelse(x == 1, 0.999999, x)
                          x}),

# separate prediction by time horizons
# due to possible missing values
pred_strat_t25 = purrr::map(.x = pred_strat,
                            ~ .x[, 1]),

pred_strat_t50 = purrr::map(.x = pred_strat,
                            ~ .x[, 2]),

pred_strat_t75 = purrr::map(.x = pred_strat,
                            ~ .x[, 3]),

pred_frail = purrr::map2(.x = cox_frail,
                         .y = validation,
                         ~ 1 - predict.coxph.gammafrail(model = .x,
                                                        newdata = .y,
                                                        cluster = "Herd",
                                                        times = time_hors)$conditional),
# Extremes to make calibration suitable
pred_frail = purrr::map(pred_frail,
                        function(x) {
                          x <- ifelse(x == 0, 0.000001, x)
                          x <- ifelse(x == 1, 0.999999, x)
                          x}),

# Score at t25/t50/t75 - frail
score_frail = purrr::map2(.x = pred_frail,
                          .y = validation,
                          ~ riskRegression::Score(
                            list("frail" = .x),
                            data = .y,
                            formula = Surv(Time, Status) ~ 1,
                            metrics = "auc",
                            summary = "ipa",
                            times = time_hors)),

# save results score - frail
score_frail_res = purrr::map(score_frail,
                             function(x) {

                               res_AUC <-
```

```
                                    x$AUC$score %>%
                                    dplyr::select(model, times, AUC)

                                  res_IPA <-
                                    x$Brier$score %>%
                                    dplyr::select(model, times, IPA) %>%
                                    dplyr::filter(model %nin% c("Null model"))

                                  res <- cbind(res_AUC,
                                               "IPA" = res_IPA$IPA)

                                  return(res)
                                }),

  # Score at t25 - strata
 score_strat_t25 = purrr::pmap(
    list(predictions = pred_strat_t25,
         df = validation,
         frm = list(as.formula(Surv(Time, Status) ~ 1)),
         t_hors = list(time_hors[1])),
    Score_tryCatch),


  # Score at t50 - strata
  score_strat_t50 = purrr::pmap(
    list(predictions = pred_strat_t50,
         df = validation,
         frm = list(as.formula(Surv(Time, Status) ~ 1)),
         t_hors = list(time_hors[2])),
    Score_tryCatch),

  # Score at t75 - strata
  score_strat_t75 = purrr::pmap(
    list(predictions = pred_strat_t75,
         df = validation,
         frm = list(as.formula(Surv(Time, Status) ~ 1)),
         t_hors = list(time_hors[3])),
    Score_tryCatch),


  # cloglog
  # cloglog - strat
  cloglog_pred_strat = purrr::map(pred_strat,
                                  function(x) {
                                    apply(x, 2, function(y) {
                                      log(-log(1 - y))
                                    })
                                  }),

 # cloglog - strat t25/t50/t75
 cloglog_pred_strat_t25 = purrr::map(.x = cloglog_pred_strat,
                                     ~ .x[, 1]),
```

```
  cloglog_pred_strat_t50 = purrr::map(.x = cloglog_pred_strat,
                                     ~ .x[, 2]),

  cloglog_pred_strat_t75 = purrr::map(.x = cloglog_pred_strat,
                                     ~ .x[, 3]),

   # cloglog
   # cloglog - frail
   cloglog_pred_frail = purrr::map(pred_frail,
                                   function(x) {
                                     apply(x, 2, function(y) {
                                       log(-log(1 - y))
                                     })
                                   }),
  # cloglog - frail t25/t50/t75
   cloglog_pred_frail_t25 = purrr::map(.x = cloglog_pred_frail,
                                       ~ .x[, 1]),

   cloglog_pred_frail_t50 = purrr::map(.x = cloglog_pred_frail,
                                       ~ .x[, 2]),

   cloglog_pred_frail_t75 = purrr::map(.x = cloglog_pred_frail,
                                       ~ .x[, 3]),


   # average expected risk
   pred_avg_strat = purrr::map(pred_strat,
                               function(x){
                                 apply(x, 2, mean)
                               }),

   pred_avg_frail = purrr::map(pred_frail,
                               function(x) {
                                apply(x, 2, mean)
                              }),

   # estimated actual risk (estimated obs proportion, observed)
   sf_obs = purrr::map(validation,
                       function(x) {
                         sf_sum <-
                           summary(survival::survfit(Surv(Time, Status) ~ 1,
                                                     data = x),
                                   extend = TRUE,
                                   times = time_hors)

                         sf_sum$surv[sf_sum$n.risk == 0] <- NA
                         sf_obs_event <- 1 - sf_sum$surv}),

  # calibration

  # OE ratio - strat
  OE_strat = purrr::map2(.x = sf_obs,
                         .y = pred_avg_strat,
```

```
                        ~ .x / .y),

 # OE ratio - frail
OE_frail = purrr::map2(.x = sf_obs,
                       .y = pred_avg_frail,
                       ~ .x / .y),

# ICI/E50/E90 - frail t25
cal_meas_frail_t25  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_frail_t25,
       t_hors = list(time_hors[1]),
       predictions = purrr::map(pred_frail, ~ .x[, 1])),
       calhaz_map),

# ICI/E50/E90 - frail t50
cal_meas_frail_t50  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_frail_t50,
       t_hors = list(time_hors[2]),
       predictions = purrr::map(pred_frail, ~ .x[, 2])),
       calhaz_map),

# ICI/E50/E90 - frail t75
cal_meas_frail_t75  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_frail_t75,
       t_hors = list(time_hors[3]),
       predictions = purrr::map(pred_frail, ~ .x[, 3])),
       calhaz_map),

# stratify
# ICI/E50/E90 - strat t25
cal_meas_strat_t25  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_strat_t25,
       t_hors = list(time_hors[1]),
       predictions = purrr::map(pred_strat, ~ .x[, 1])),
       calhaz_map),

# ICI/E50/E90 - strat t50
cal_meas_strat_t50  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_strat_t50,
       t_hors = list(time_hors[2]),
       predictions = purrr::map(pred_strat, ~ .x[, 2])),
       calhaz_map),
```

```
  # ICI/E50/E90 - strat t75
  cal_meas_strat_t75  = purrr::pmap(
    list(time = purrr::map(validation, ~ .x$Time),
         status = purrr::map(validation, ~ .x$Status),
         covariates = cloglog_pred_strat_t75,
         t_hors = list(time_hors[3]),
         predictions = purrr::map(pred_strat, ~ .x[, 3])),
         calhaz_map))

# tictoc::toc()
```

2.1 Small cluster size

Small cluster scenario						
			AUC		IPA	
model	times	pct_comp	AUC_mean	AUC_sd	IPA_mean	IPA_sd
frail	52	100	0.688	0.015	0.075	0.016
strat	52	95	0.658	0.015	0.039	0.021
frail	91	100	0.730	0.016	0.156	0.023
strat	91	78	0.723	0.016	0.146	0.026
frail	150	100	0.813	0.016	0.229	0.038

Small cluster scenario				
			O-E ratio	
model	times	pct_comp	OE_ratio_mean	OE_ratio_sd
frail	52	100	1.066	0.074
strat	52	95	1.024	0.087
frail	91	100	1.072	0.071
strat	91	78	1.023	0.045
frail	150	100	1.058	0.081

Small cluster scenario								
			ICI		E50		E90	
model	times	pct_comp	ICI_mean	ICI_sd	E50_mean	E50_sd	E90_mean	E90_sd
frail	52	100	0.045	0.032	0.046	0.035	0.070	0.035
strat	52	95	0.037	0.011	0.032	0.013	0.071	0.013
frail	91	100	0.044	0.028	0.045	0.030	0.070	0.030
strat	91	78	0.037	0.014	0.033	0.014	0.064	0.014
frail	150	100	0.045	0.027	0.047	0.030	0.068	0.030

```r
# time horizons
time_hors <- c(52, 91, 150)

# reference values for continuous
# predictors
insem_ref <-
  insem %>%
  dplyr::mutate(
    Ureum_ref = Ureum - 2.5,
    Protein_ref = Protein - 2.5,
    Parity_ref = Parity - 2.5)


# Herds >= 50 cows
table_insem_large <-
  insem_ref %>%
  dplyr::count(Herd) %>%
  dplyr::filter(n >= 50)

df_insem_large <-
  insem_ref %>%
  dplyr::filter(Herd %in% table_insem_large$Herd)

# Repeat B times
B <- 100
df_tbl_insem_ext <- do.call(rbind,
                        lapply(seq(1:B),
                               function(k) {
                                 cbind(id = k, df_insem_large)}))
# Nested tibble
set.seed(20250924)
df_tbl_insem <-
  df_tbl_insem_ext %>%
  dplyr::group_nest(id) %>%
  dplyr::mutate(
    # Splitting
    split = purrr::map(.x = data,
                       ~ rsample::initial_split(.x,
                                                strata = Herd,
                                                prop = 0.7)),
    # Development
    development = purrr::map(.x = split,
                             ~ rsample::training(.x)),

    # Validation
    validation = purrr::map(.x = split,
                             ~ rsample::testing(.x)),

    # Check
    unique_cluster_dev = purrr::map(.x = development,
                                    ~ unique(.x$Herd)),
```

```
    unique_cluster_val = purrr::map(.x = validation,
                                    ~ unique(.x$Herd)),

    check = purrr::map2(.x = unique_cluster_dev,
                        .y = unique_cluster_val,
                        ~ all.equal(.x, .y)))



rm(df_tbl_insem_ext)
# gc()
# check
# table(do.call(rbind, df_tbl_insem$check))
# all TRUE

# Models + predictions + performances
# tictoc::tic()
df_tbl_insem <-
  df_tbl_insem %>%
  dplyr::select(data,
                development,
                validation) %>%

  dplyr::mutate(
    cox_strat = purrr::map(.x = development,
                           ~ survival::coxph(Surv(Time, Status) ~
                                               Heifer +
                                               Parity_ref +
                                               Ureum_ref +
                                               Protein_ref +
                                               strata(Herd),
                                             data = .x,
                                             x = T,
                                             y = T,
                                             ties = "breslow")),

    cox_frail = purrr::map(.x = development,
                           ~ survival::coxph(Surv(Time, Status) ~
                                               Heifer +
                                               Parity_ref +
                                               Ureum_ref +
                                               Protein_ref +
                                               frailty(Herd),
                                             data = .x,
                                             x = T,
                                             y = T,
                                             ties = "breslow")),

    # predictions at t25/t50/t75
    # stratified cox
    pred_strat = purrr::map2(.x = cox_strat,
                             .y = validation,
                             ~ 1 - pec::predictSurvProb(.x,
```

```
                                                    newdata = .y,
                                                    times = time_hors)),
  # Extremes to make calibration suitable
  pred_strat = purrr::map(pred_strat,
                          function(x) {
                            x <- ifelse(x == 0, 0.000001, x)
                            x <- ifelse(x == 1, 0.999999, x)
                            x}),

  # separate prediction by time horizons
  # due to possible missing values
  pred_strat_t25 = purrr::map(.x = pred_strat,
                              ~ .x[, 1]),

  pred_strat_t50 = purrr::map(.x = pred_strat,
                              ~ .x[, 2]),

  pred_strat_t75 = purrr::map(.x = pred_strat,
                              ~ .x[, 3]),

  pred_frail = purrr::map2(.x = cox_frail,
                           .y = validation,
                           ~ 1 - predict.coxph.gammafrail(model = .x,
                                                          newdata = .y,
                                                          cluster = "Herd",
                                                          times = time_hors)$conditional),
  # Extremes to make calibration suitable
  pred_frail = purrr::map(pred_frail,
                          function(x) {
                            x <- ifelse(x == 0, 0.000001, x)
                            x <- ifelse(x == 1, 0.999999, x)
                            x}),

  # Score at t25/t50/t75 - frail
  score_frail = purrr::map2(.x = pred_frail,
                            .y = validation,
                            ~ riskRegression::Score(
                              list("frail" = .x),
                              data = .y,
                              formula = Surv(Time, Status) ~ 1,
                              metrics = "auc",
                              summary = "ipa",
                              times = time_hors)),

  # save results score - frail
  score_frail_res = purrr::map(score_frail,
                               function(x) {

                                 res_AUC <-
                                   x$AUC$score %>%
                                   dplyr::select(model, times, AUC)

                                 res_IPA <-
```

```
                                    x$Brier$score %>%
                                    dplyr::select(model, times, IPA) %>%
                                    dplyr::filter(model %nin% c("Null model"))

                                  res <- cbind(res_AUC,
                                               "IPA" = res_IPA$IPA)

                                  return(res)
                                }),

 # Score at t25 - strata
score_strat_t25 = purrr::pmap(
   list(predictions = pred_strat_t25,
        df = validation,
        frm = list(as.formula(Surv(Time, Status) ~ 1)),
        t_hors = list(time_hors[1])),
   Score_tryCatch),


 # Score at t50 - strata
 score_strat_t50 = purrr::pmap(
   list(predictions = pred_strat_t50,
        df = validation,
        frm = list(as.formula(Surv(Time, Status) ~ 1)),
        t_hors = list(time_hors[2])),
   Score_tryCatch),

 # Score at t75 - strata
 score_strat_t75 = purrr::pmap(
   list(predictions = pred_strat_t75,
        df = validation,
        frm = list(as.formula(Surv(Time, Status) ~ 1)),
        t_hors = list(time_hors[3])),
   Score_tryCatch),


 # cloglog
 # cloglog - strat
 cloglog_pred_strat = purrr::map(pred_strat,
                                 function(x) {
                                   apply(x, 2, function(y) {
                                     log(-log(1 - y))
                                   })
                                 }),

# cloglog - strat t25/t50/t75
cloglog_pred_strat_t25 = purrr::map(.x = cloglog_pred_strat,
                                    ~ .x[, 1]),

cloglog_pred_strat_t50 = purrr::map(.x = cloglog_pred_strat,
                                    ~ .x[, 2]),

cloglog_pred_strat_t75 = purrr::map(.x = cloglog_pred_strat,
```

```
                                    ~ .x[, 3]),

 # cloglog
 # cloglog - frail
 cloglog_pred_frail = purrr::map(pred_frail,
                                 function(x) {
                                   apply(x, 2, function(y) {
                                     log(-log(1 - y))
                                   })
                                 }),
# cloglog - frail t25/t50/t75
 cloglog_pred_frail_t25 = purrr::map(.x = cloglog_pred_frail,
                                     ~ .x[, 1]),

 cloglog_pred_frail_t50 = purrr::map(.x = cloglog_pred_frail,
                                     ~ .x[, 2]),

 cloglog_pred_frail_t75 = purrr::map(.x = cloglog_pred_frail,
                                     ~ .x[, 3]),


 # average expected risk
 pred_avg_strat = purrr::map(pred_strat,
                             function(x){
                               apply(x, 2, mean)
                             }),

 pred_avg_frail = purrr::map(pred_frail,
                             function(x) {
                              apply(x, 2, mean)
                            }),

 # estimated actual risk (estimated obs proportion, observed)
 sf_obs = purrr::map(validation,
                     function(x) {
                       sf_sum <-
                         summary(survival::survfit(Surv(Time, Status) ~ 1,
                                                   data = x),
                                 extend = TRUE,
                                 times = time_hors)

                       sf_sum$surv[sf_sum$n.risk == 0] <- NA
                       sf_obs_event <- 1 - sf_sum$surv}),

# calibration

# OE ratio - strat
OE_strat = purrr::map2(.x = sf_obs,
                       .y = pred_avg_strat,
                       ~ .x / .y),

 # OE ratio - frail
OE_frail = purrr::map2(.x = sf_obs,
```

```
                        .y = pred_avg_frail,
                        ~ .x / .y),

# ICI/E50/E90 - frail t25
cal_meas_frail_t25  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_frail_t25,
       t_hors = list(time_hors[1]),
       predictions = purrr::map(pred_frail, ~ .x[, 1])),
       calhaz_map),

# ICI/E50/E90 - frail t50
cal_meas_frail_t50  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_frail_t50,
       t_hors = list(time_hors[2]),
       predictions = purrr::map(pred_frail, ~ .x[, 2])),
       calhaz_map),

# ICI/E50/E90 - frail t75
cal_meas_frail_t75  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_frail_t75,
       t_hors = list(time_hors[3]),
       predictions = purrr::map(pred_frail, ~ .x[, 3])),
       calhaz_map),

# stratify
# ICI/E50/E90 - strat t25
cal_meas_strat_t25  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_strat_t25,
       t_hors = list(time_hors[1]),
       predictions = purrr::map(pred_strat, ~ .x[, 1])),
       calhaz_map),

# ICI/E50/E90 - strat t50
cal_meas_strat_t50  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
       covariates = cloglog_pred_strat_t50,
       t_hors = list(time_hors[2]),
       predictions = purrr::map(pred_strat, ~ .x[, 2])),
       calhaz_map),

# ICI/E50/E90 - strat t75
cal_meas_strat_t75  = purrr::pmap(
  list(time = purrr::map(validation, ~ .x$Time),
       status = purrr::map(validation, ~ .x$Status),
```

```
          covariates = cloglog_pred_strat_t75,
          t_hors = list(time_hors[3]),
          predictions = purrr::map(pred_strat, ~ .x[, 3])),
          calhaz_map))

# tictoc::toc()
```

2.2 Large cluster size

Large cluster scenario						
			AUC		IPA	
model	times	pct_comp	AUC_mean	AUC_sd	IPA_mean	IPA_sd
frail	52	100	0.670	0.010	0.067	0.009
strat	52	100	0.656	0.009	0.055	0.009
frail	91	100	0.706	0.008	0.122	0.011
strat	91	97	0.701	0.008	0.124	0.011
frail	150	100	0.776	0.010	0.162	0.022

Large cluster scenario				
			O-E ratio	
model	times	pct_comp	OE_ratio_mean	OE_ratio_sd
frail	52	100	1.101	0.037
strat	52	100	1.015	0.045
frail	91	100	1.097	0.042
strat	91	97	1.011	0.024
frail	150	100	1.095	0.040

Large cluster scenario								
			ICI		E50		E90	
model	times	pct_comp	ICI_mean	ICI_sd	E50_mean	E50_sd	E90_mean	E90_sd
frail	52	100	0.050	0.030	0.053	0.031	0.066	0.040
strat	52	100	0.026	0.047	0.023	0.039	0.048	0.087
frail	91	100	0.050	0.032	0.052	0.032	0.066	0.049
strat	91	97	0.020	0.009	0.019	0.010	0.036	0.018
frail	150	100	0.048	0.035	0.050	0.031	0.067	0.067

3. Additional investigations

We additionally estimate predictions from the stratified and the shared gamma frailty Cox model using all data without splitting.

We provide the corresponding plots of predictions at different time horizons from the stratified versus frailty model in the small and large cluster scenario.

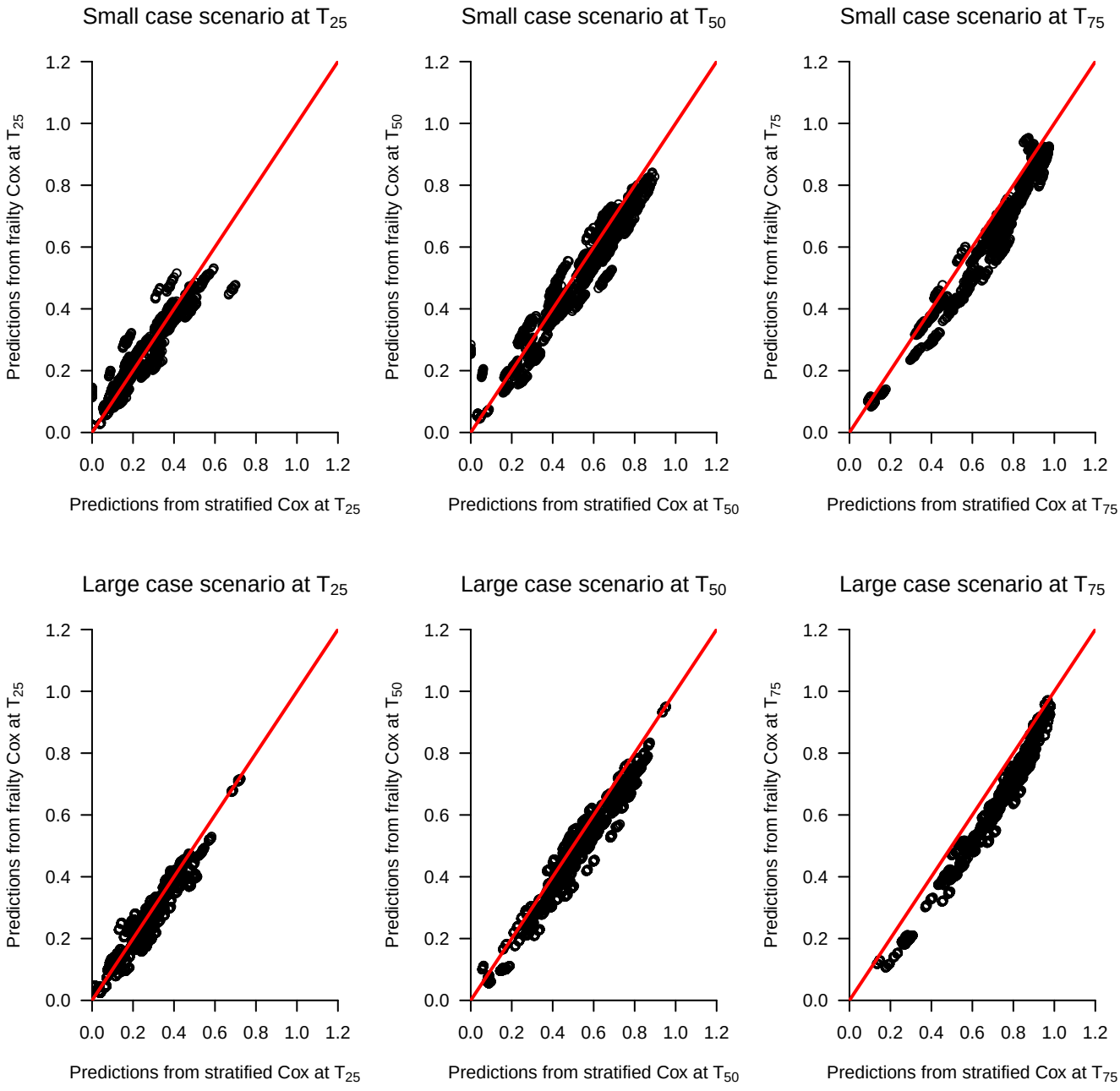
```
# Herds < 50 (herds >= 10)
table_insem_small <-
  insem %>%
  dplyr::count(Herd) %>%
```

```r
  dplyr::filter(n < 50 & n >= 10)

# Herds >= 50 cows
table_insem_large <-
  insem %>%
  dplyr::count(Herd) %>%
  dplyr::filter(n >= 50)

df_insem <-
  insem %>%
  # dplyr::filter(Herd %in% table_insem_50$Herd) %>%
  # reference values for all continuous variable
  # to estimate predictions from frailty correctly
  dplyr::mutate(
    Ureum_ref = Ureum - 2.5,
    Protein_ref = Protein - 2.5,
    Parity_ref = Parity - 2.5)

# Small scenario overall data
df_insem_small <-
  df_insem %>%
  dplyr::filter(Herd %in% table_insem_small$Herd)

# Large scenario overall data
df_insem_large <-
  df_insem %>%
  dplyr::filter(Herd %in% table_insem_large$Herd)

# Stratified Cox - small
cox_strat_small <-
  survival::coxph(Surv(Time, Status) ~
                    Heifer +
                    Parity_ref +
                    Ureum_ref +
                    Protein_ref +
                    strata(Herd),
      data = df_insem_small,
      x = T,
      y = T,
      ties = "breslow")

# Frailty Cox - small
cox_frail_small <-
  survival::coxph(Surv(Time, Status) ~
                    Heifer +
                    Parity_ref +
                    Ureum_ref +
                    Protein_ref +
                    frailty(Herd),
      data = df_insem_small,
      x = T,
      y = T,
      ties = "breslow")
```

```r
# Stratified Cox - large
cox_strat_large <-
  survival::coxph(Surv(Time, Status) ~
                    Heifer +
                    Parity_ref +
                    Ureum_ref +
                    Protein_ref +
                    strata(Herd),
      data = df_insem_large,
      x = T,
      y = T,
      ties = "breslow")

# Frailty Cox - small
cox_frail_large <-
  survival::coxph(Surv(Time, Status) ~
                    Heifer +
                    Parity_ref +
                    Ureum_ref +
                    Protein_ref +
                    frailty(Herd),
      data = df_insem_large,
      x = T,
      y = T,
      ties = "breslow")

## Predictions
# at 25th, 50th, 75th overall percentiles
time_hors <- c(52, 91, 150)
# time_hors <- q_surv[1, ]

# Predictions
# Stratified small
pred_strat_small <- 1 - pec::predictSurvProb(cox_strat_small,
                                             newdata = df_insem_small,
                                             times = time_hors)

# Frailty small
pred_frail_small <- 1 - predict.coxph.gammafrail(model = cox_frail_small,
                                                 newdata = df_insem_small,
                                                 cluster = "Herd",
                                                 times = time_hors)$conditional

# Predictions
# Stratified large
pred_strat_large <- 1 - pec::predictSurvProb(cox_strat_large,
                                             newdata = df_insem_large,
                                             times = time_hors)

# Frailty large
pred_frail_large <- 1 - predict.coxph.gammafrail(model = cox_frail_large,
                                                 newdata = df_insem_large,
                                                 cluster = "Herd",
```

```
                                                  times = time_hors)$conditional

# Plot predictions
# from stratify and frailty model

# Small scenario at T25, T50 and T75
oldpar <-
  par(
    mfrow = c(2, 3),
    xaxs = "i",
    yaxs = "i",
    las = 1)
plot(pred_strat_small[, 1],
     pred_frail_small[, 1],
     xlim = c(0, 1.2),
     xlab = expression("Predictions from stratified Cox at T"[25]),
     ylim = c(0, 1.2),
     ylab = expression("Predictions from frailty Cox at T"[25]),
     main = expression("Small case scenario at T"[25]),
     bty = "n")
abline(a = 0,
       b = 1,
       lwd = 2,
       col = "red")

par(xaxs = "i",
    yaxs = "i",
    las = 1)
plot(pred_strat_small[, 2],
     pred_frail_small[, 2],
     xlim = c(0, 1.2),
     xlab = expression("Predictions from stratified Cox at T"[50]),
     ylim = c(0, 1.2),
     ylab = expression("Predictions from frailty Cox at T"[50]),
     main = expression("Small case scenario at T"[50]),
     bty = "n")
abline(a = 0,
       b = 1,
       lwd = 2,
       col = "red")

par(xaxs = "i",
    yaxs = "i",
    las = 1)
plot(pred_strat_small[, 3],
     pred_frail_small[, 3],
     xlim = c(0, 1.2),
     xlab = expression("Predictions from stratified Cox at T"[75]),
     ylim = c(0, 1.2),
     ylab = expression("Predictions from frailty Cox at T"[75]),
     main = expression("Small case scenario at T"[75]),
     bty = "n")
abline(a = 0,
```

```
       b = 1,
       lwd = 2,
       col = "red")

# Large scenario at T25, T50 and T75
par(xaxs = "i",
    yaxs = "i",
    las = 1)
plot(pred_strat_large[, 1],
     pred_frail_large[, 1],
     xlim = c(0, 1.2),
     xlab = expression("Predictions from stratified Cox at T"[25]),
     ylim = c(0, 1.2),
     ylab = expression("Predictions from frailty Cox at T"[25]),
     main = expression("Large case scenario at T"[25]),
     bty = "n")
abline(a = 0,
       b = 1,
       lwd = 2,
       col = "red")

par(xaxs = "i",
    yaxs = "i",
    las = 1)
plot(pred_strat_large[, 2],
     pred_frail_large[, 2],
     xlim = c(0, 1.2),
     xlab = expression("Predictions from stratified Cox at T"[50]),
     ylim = c(0, 1.2),
     ylab = expression("Predictions from frailty Cox at T"[50]),
     main = expression("Large case scenario at T"[50]),
     bty = "n")
abline(a = 0,
       b = 1,
       lwd = 2,
       col = "red")

par(xaxs = "i",
    yaxs = "i",
    las = 1)
plot(pred_strat_large[, 3],
     pred_frail_large[, 3],
     xlim = c(0, 1.2),
     xlab = expression("Predictions from stratified Cox at T"[75]),
     ylim = c(0, 1.2),
     ylab = expression("Predictions from frailty Cox at T"[75]),
     main = expression("Large case scenario at T"[75]),
     bty = "n")
abline(a = 0,
       b = 1,
       lwd = 2,
       col = "red")
```



NOTE: comparisons only among clusters where predictions were possible for both models

Reproducibility ticket

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
```

```
##
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## time zone: Europe/Rome
## tzcode source: internal
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] parfm_2.7.8               optimx_2025-4.9
##  [3] survival_3.8-3            webshot2_0.1.2
##  [5] rms_6.8-0                 patchwork_1.2.0
##  [7] ggpubr_0.6.0              plotly_4.10.4
##  [9] webshot_0.5.5             gridExtra_2.3
## [11] lubridate_1.9.4           forcats_1.0.0
## [13] dplyr_1.1.4               purrr_1.0.2
## [15] readr_2.1.5               tidyr_1.3.1
## [17] tibble_3.2.1              ggplot2_3.5.2
## [19] tidyverse_2.0.0           gtsummary_1.7.2
## [21] polspline_1.1.25          riskRegression_2025.05.20
## [23] lattice_0.22-5            Hmisc_5.2-3
## [25] kableExtra_1.4.0          knitr_1.50
## [27] stringr_1.5.1             rio_1.2.3
## [29] pacman_0.5.1
##
## loaded via a namespace (and not attached):
##   [1] RColorBrewer_1.1-3   rstudioapi_0.17.1    jsonlite_1.8.9
##   [4] shape_1.4.6.1        magrittr_2.0.3       TH.data_1.1-3
##   [7] farver_2.1.2         nloptr_2.2.1         rmarkdown_2.29
##  [10] vctrs_0.6.5          base64enc_0.1-3      rstatix_0.7.2
##  [13] htmltools_0.5.8.1    broom_1.0.9          Formula_1.2-5
##  [16] parallelly_1.45.1    pracma_2.4.4         htmlwidgets_1.6.4
##  [19] sandwich_3.1-1       zoo_1.8-14           gt_1.0.0
##  [22] commonmark_2.0.0     lifecycle_1.0.4      cmprsk_2.2-12
##  [25] iterators_1.0.14     pkgconfig_2.0.3      Matrix_1.6-5
##  [28] R6_2.6.1             fastmap_1.2.0        future_1.67.0
##  [31] digest_0.6.35        numDeriv_2016.8-1.1  colorspace_2.1-1
##  [34] ps_1.9.1             rprojroot_2.1.0      timechange_0.3.0
##  [37] httr_1.4.7           abind_1.4-8          compiler_4.3.3
##  [40] here_1.0.1           withr_3.0.2          htmlTable_2.4.3
##  [43] backports_1.5.0      carData_3.0-5        ggsignif_0.6.4
##  [46] MASS_7.3-60.0.1      lava_1.8.1           quantreg_6.1
##  [49] tools_4.3.3          chromote_0.5.1       foreign_0.8-90
##  [52] future.apply_1.20.0  nnet_7.3-20          glue_1.7.0
##  [55] mets_1.3.5           nlme_3.1-164         promises_1.3.2
##  [58] checkmate_2.3.2      cluster_2.1.8.1      generics_0.1.4
```

```
##  [61] gtable_0.3.6         tzdb_0.5.0           sn_2.1.1
##  [64] websocket_1.4.2      data.table_1.16.2    hms_1.1.3
##  [67] xml2_1.3.6           car_3.1-3            foreach_1.5.2
##  [70] pillar_1.11.0        markdown_2.0         later_1.4.2
##  [73] splines_4.3.3        SparseM_1.84-2       tidyselect_1.2.1
##  [76] litedown_0.7         svglite_2.1.3        stats4_4.3.3
##  [79] xfun_0.52            expm_1.0-0           stringi_1.8.4
##  [82] lazyeval_0.2.2       yaml_2.3.10          pec_2025.06.24
##  [85] evaluate_1.0.4       codetools_0.2-20     msm_1.8.2
##  [88] cli_3.6.2            rpart_4.1.24         systemfonts_1.2.3
##  [91] processx_3.8.6       Rcpp_1.1.0           globals_0.18.0
##  [94] parallel_4.3.3       MatrixModels_0.5-4   listenv_0.9.1
##  [97] glmnet_4.1-10        viridisLite_0.4.2    broom.helpers_1.15.0
## [100] mvtnorm_1.3-3        timereg_2.0.6        scales_1.4.0
## [103] prodlim_2025.04.28   rlang_1.1.6          multcomp_1.4-28
## [106] mnormt_2.1.1
```