```r
# Load required packages
library(readxl)
library(survival)
```

```
## Warning: package 'survival' was built under R version 4.5.1
```

```r
library(cmprsk)
library(randomForestSRC)
```

```
## Warning: package 'randomForestSRC' was built under R version 4.5.1
```

```
##
##  randomForestSRC 3.4.1
##
##  Type rfsrc.news() to see new features, changes, and bug fixes.
##
```

```r
library(riskRegression)
```

```
## Warning: package 'riskRegression' was built under R version 4.5.1
```

```
## riskRegression version 2025.05.20
```

```r
library(prodlim)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.5.1
```

```r
library(DALEX)
```

```
## Welcome to DALEX (version: 2.4.3).
## Find examples and detailed introduction at: http://ema.drwhy.ai/
```

```r
library(iBreakDown)
library(tidyr)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.5.1
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:DALEX':
##
##     explain
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library(knitr)

# Function to generate distribution plots (histograms for continuous, bar plo
ts for categorical)
create_distribution_plots <- function(data, plot_config, ncol = 3) {
  plot_list <- list()

  for (var in names(plot_config)) {
    config <- plot_config[[var]]

    if (config$type == "histogram") {
      p <- ggplot(data, aes(x = .data[[var]])) +
        geom_histogram(bins = 30, fill = config$fill, alpha = 0.7, color = "w
hite") +
        labs(title = config$title, x = config$xlab, y = "Frequency") +
        theme_minimal() +
        theme(plot.title = element_text(size = 10, face = "bold"))
    } else if (config$type == "bar") {
      p <- ggplot(data, aes(x = factor(.data[[var]]))) +
        geom_bar(fill = config$fill, alpha = 0.7, color = "white") +
        labs(title = config$title, x = config$xlab, y = "Count") +
        theme_minimal() +
        theme(plot.title = element_text(size = 10, face = "bold"),
              axis.text.x = element_text(angle = 45, hjust = 1))
    }

    plot_list[[var]] <- p
  }

  n_plots <- length(plot_list)
  nrow <- ceiling(n_plots / ncol)
  grid.arrange(grobs = plot_list, ncol = ncol, nrow = nrow)
}

# Simulate health data with competing risks and nonlinear (quadratic) covaria
te effects
generate_health_data <- function(n, seed = 1234) {
  set.seed(seed)

  age      <- pmax(30, pmin(rnorm(n, 60, 10), 90))
  bmi      <- pmax(18, pmin(rnorm(n, 28, 4), 45))
```

```r
  smoking  <- sample(0:1, n, replace = TRUE, prob = c(0.7, 0.3))
  hba1c    <- pmax(4.5, pmin(rnorm(n, 7.0, 1.2), 12.0))

  # Quadratic effects for cause 1 (event of interest)
  lp1 <- 0.002 * age +
    0.06 * (bmi - 28)^2 +
    0.15 * (hba1c - 7)^2 +
    0.5 * smoking

  # Asymmetric nonlinear effect for cause 2 (competing event)
  lp2 <- 0.017 * age +
    0.03 * abs(bmi - 26) +
    0.3 * smoking

  shape <- 2.0
  scale_factor <- 90

  time_1 <- rweibull(n, shape = shape, scale = scale_factor * exp(-lp1 / shap
e))
  time_2 <- rweibull(n, shape = shape, scale = scale_factor * exp(-lp2 / shap
e))

  true_time <- pmin(time_1, time_2)
  obs_time <- runif(n, min = 15, max = 80)

  time <- pmin(true_time, obs_time)
  status <- ifelse(true_time <= obs_time,
                   ifelse(time_1 < time_2, 1, 2),
                   0)

  time <- pmin(time, 80)
  time <- round(time, 0)

  data.frame(
    time = time,
    status = as.integer(status),
    age = round(age, 0),
    bmi = round(bmi, 2),
    smoking = smoking,
    hba1c = round(hba1c, 2)
  )
}

# Evaluate model performance across time: AUC, Brier score, CIF, calibration
calculate_model_performance <- function(models,
                                        data,
                                        time_var = "time",
                                        event_var = "status",
                                        times = seq(6, 80, 6),
```

```
                                        fgr_model = NULL,
                                        rsf_model = NULL,
                                        csc_model1 = NULL,
                                        csc_model2 = NULL,
                                        B = 0,
                                        split.method = "none") {

  formula_obj <- as.formula(paste0("Hist(", time_var, ", ", event_var, ") ~ 1
"))

  score_result <- Score(
    object = models,
    formula = formula_obj,
    data = data,
    times = times,
    plots = "calibration",
    summary = "risks",
    se.fit = TRUE,
    B = B,
    split.method = split.method
  )

  model_colors <- c("Fine-Gray" = "#E41A1C",
                    "RSF Model" = "#377EB8",
                    "CSC Model" = "#4DAF4A",
                    "Aalen-Johansen" = "#000000")

  cif_data_list <- list()

  # Aalen-Johansen (nonparametric reference)
  aj_formula <- as.formula(paste0("Hist(", time_var, ", ", event_var, ") ~ 1"
))
  aj <- prodlim::prodlim(aj_formula, data = data)

  aj_cause1_interp <- approx(aj$time, aj$cuminc$`1`, xout = times, rule = 2)$
y
  aj_cause2_interp <- approx(aj$time, aj$cuminc$`2`, xout = times, rule = 2)$
y

  cif_data_list$AJ <- data.frame(
    Time = rep(times, 2),
    CIF = c(aj_cause1_interp, aj_cause2_interp),
    Model = "Aalen-Johansen",
    Cause = rep(c("Cause 1", "Cause 2"), each = length(times))
  )

  # RSF CIF estimates
  if (!is.null(rsf_model)) {
    pred_rsf <- predict(rsf_model, newdata = data)
```

```r
    cif_cause1_rsf_interp <- approx(pred_rsf$time.interest, apply(pred_rsf$ci
f[,,1], 2, mean), xout = times, rule = 2)$y
    cif_cause2_rsf_interp <- approx(pred_rsf$time.interest, apply(pred_rsf$ci
f[,,2], 2, mean), xout = times, rule = 2)$y

    cif_data_list$RSF <- data.frame(
      Time = rep(times, 2),
      CIF = c(cif_cause1_rsf_interp, cif_cause2_rsf_interp),
      Model = "RSF Model",
      Cause = rep(c("Cause 1", "Cause 2"), each = length(times))
    )
  }

  # Fine-Gray CIF estimates
  if (!is.null(fgr_model)) {
    fgr1 <- FGR(formula(fgr_model), data = data, cause = 1)
    fgr2 <- FGR(formula(fgr_model), data = data, cause = 2)

    pred_fgr1 <- predict(fgr1, newdata = data, times = times, cause = 1)
    pred_fgr2 <- predict(fgr2, newdata = data, times = times, cause = 2)

    cif_data_list$FGR <- data.frame(
      Time = rep(times, 2),
      CIF = c(colMeans(pred_fgr1), colMeans(pred_fgr2)),
      Model = "Fine-Gray",
      Cause = rep(c("Cause 1", "Cause 2"), each = length(times))
    )
  }

  # Cause-specific Cox CIF estimates
  if (!is.null(csc_model1) && !is.null(csc_model2)) {
    pred_csc1 <- predict(csc_model1, newdata = data, times = times, type = "a
bsRisk")
    pred_csc2 <- predict(csc_model2, newdata = data, times = times, type = "a
bsRisk")

    # Handle different output structures from riskRegression::predict
    extract_means <- function(pred, n_times) {
      if (is.matrix(pred) || is.data.frame(pred)) {
        colMeans(as.matrix(pred), na.rm = TRUE)
      } else if (is.list(pred) && "absRisk" %in% names(pred)) {
        vec <- as.numeric(unlist(pred$absRisk))
        mat <- matrix(vec[1:(length(vec) %/% n_times * n_times)], ncol = n_ti
mes, byrow = TRUE)
        colMeans(mat, na.rm = TRUE)
      } else {
        rep(NA, n_times)
      }
    }
```

```r
    cif_cause1_csc <- extract_means(pred_csc1, length(times))
    cif_cause2_csc <- extract_means(pred_csc2, length(times))

    cif_cause1_csc[1] <- ifelse(is.na(cif_cause1_csc[1]), 0, cif_cause1_csc[1
])
    cif_cause2_csc[1] <- ifelse(is.na(cif_cause2_csc[1]), 0, cif_cause2_csc[1
])

    cif_data_list$CSC <- data.frame(
      Time = rep(times, 2),
      CIF = c(cif_cause1_csc, cif_cause2_csc),
      Model = "CSC Model",
      Cause = rep(c("Cause 1", "Cause 2"), each = length(times))
    )
  }

  cif_data <- do.call(rbind, cif_data_list) %>%
    mutate(Cause_Model = paste(Cause, Model, sep = "-"))

  # AUC plot
  auc_plot <- ggplot(score_result$AUC$score, aes(x = times, y = AUC, color =
model)) +
    geom_line(linewidth = 1.2) +
    geom_ribbon(aes(ymin = lower, ymax = upper, fill = model), alpha = 0.2) +
    labs(x = "Time (Months)", y = "Time-dependent AUC") +
    theme_minimal(base_size = 14) +
    theme(legend.position = "top", legend.title = element_blank()) +
    scale_color_manual(values = model_colors) +
    scale_fill_manual(values = model_colors) +
    ylim(0, 1) + xlim(0, max(times))

  # Brier score plot
  brier_plot <- ggplot(score_result$Brier$score, aes(x = times, y = Brier, co
lor = model)) +
    geom_line(linewidth = 1.2) +
    geom_ribbon(aes(ymin = lower, ymax = upper, fill = model), alpha = 0.2) +
    labs(x = "Time (Months)", y = "Brier Score") +
    theme_minimal(base_size = 14) +
    theme(legend.position = "top", legend.title = element_blank()) +
    scale_color_manual(values = model_colors) +
    scale_fill_manual(values = model_colors) +
    ylim(0, max(score_result$Brier$score$Brier, na.rm = TRUE) * 1.1) +
    xlim(0, max(times))

  # CIF plot (with Aalen-Johansen reference)
  cif_plot <- ggplot(cif_data, aes(x = Time, y = CIF, color = Model, linetype
= Cause)) +
    geom_line(size = 1.2) +
```

```
    labs(x = "Time (Months)", y = "Cumulative Incidence") +
    theme_minimal(base_size = 14) +
    theme(legend.position = "top", legend.box = "vertical") +
    scale_color_manual(values = model_colors) +
    scale_linetype_manual(values = c("Cause 1" = "solid", "Cause 2" = "dashed
")) +
    scale_x_continuous(limits = c(0, max(times)), breaks = seq(0, max(times),
10)) +
    scale_y_continuous(limits = c(0, 1)) +
    guides(color = guide_legend(title = "Model"), linetype = guide_legend(tit
le = "Cause"))

  # Performance table
  auc_df <- score_result$AUC$score
  brier_df <- score_result$Brier$score

  performance_table <- data.frame(
    Time = times,
    FG_AUC = if("Fine-Gray" %in% auc_df$model) round(auc_df$AUC[auc_df$model
== "Fine-Gray"], 3) else NA,
    RSF_AUC = if("RSF Model" %in% auc_df$model) round(auc_df$AUC[auc_df$model
== "RSF Model"], 3) else NA,
    CSC_AUC = if("CSC Model" %in% auc_df$model) round(auc_df$AUC[auc_df$model
== "CSC Model"], 3) else NA,
    FG_Brier = if("Fine-Gray" %in% brier_df$model) round(brier_df$Brier[brier
_df$model == "Fine-Gray"], 3) else NA,
    RSF_Brier = if("RSF Model" %in% brier_df$model) round(brier_df$Brier[brie
r_df$model == "RSF Model"], 3) else NA,
    CSC_Brier = if("CSC Model" %in% brier_df$model) round(brier_df$Brier[brie
r_df$model == "CSC Model"], 3) else NA
  )

  performance_table$AJ_CIF1 <- round(cif_data_list$AJ$CIF[cif_data_list$AJ$Ca
use == "Cause 1"], 3)
  performance_table$AJ_CIF2 <- round(cif_data_list$AJ$CIF[cif_data_list$AJ$Ca
use == "Cause 2"], 3)

  if (!is.null(fgr_model)) {
    performance_table$FG_CIF1 <- round(cif_data_list$FGR$CIF[cif_data_list$FG
R$Cause == "Cause 1"], 3)
    performance_table$FG_CIF2 <- round(cif_data_list$FGR$CIF[cif_data_list$FG
R$Cause == "Cause 2"], 3)
  }

  if (!is.null(rsf_model)) {
    performance_table$RSF_CIF1 <- round(cif_data_list$RSF$CIF[cif_data_list$R
SF$Cause == "Cause 1"], 3)
    performance_table$RSF_CIF2 <- round(cif_data_list$RSF$CIF[cif_data_list$R
SF$Cause == "Cause 2"], 3)
```

```r
  }

  if (!is.null(csc_model1) && !is.null(csc_model2)) {
    performance_table$CSC_CIF1 <- round(cif_data_list$CSC$CIF[cif_data_list$C
SC$Cause == "Cause 1"], 3)
    performance_table$CSC_CIF2 <- round(cif_data_list$CSC$CIF[cif_data_list$C
SC$Cause == "Cause 2"], 3)
  }

  # Return results
  list(
    score = score_result,
    auc_plot = auc_plot,
    brier_plot = brier_plot,
    cif_plot = cif_plot,
    calib_plot = function() plotCalibration(score_result),
    performance_table = performance_table,
    cif_data = cif_data
  )
}

# Generate data and fit models
set.seed(1234)
n <- 500
train_data <- generate_health_data(n * 0.8)
test_data <- generate_health_data(n * 0.2)

survival_vars <- c("time", "status")
time_var <- survival_vars[1]
event_var <- survival_vars[2]

continuous_vars <- c("age", "bmi", "hba1c")
categorical_vars <- c("smoking")
predictor_vars <- c(continuous_vars, categorical_vars)
formula_rhs <- paste(predictor_vars, collapse = " + ")

hist_formula <- as.formula(paste("Hist(", time_var, ", ", event_var, ") ~", f
ormula_rhs))
surv_formula <- as.formula(paste("Surv(", time_var, ", ", event_var, ") ~", f
ormula_rhs))

fg_model <- FGR(hist_formula, data = train_data, cause = 1)
rsf_model <- rfsrc(surv_formula, data = train_data, ntree = 100, cause = 1, i
mportance = TRUE)
csc_model1 <- CSC(hist_formula, data = train_data, cause = 1)
csc_model2 <- CSC(hist_formula, data = train_data, cause = 2)

# Evaluate performance on test set at 66 months
results1 <- calculate_model_performance(
```

```
  models = list("Fine-Gray" = fg_model, "RSF Model" = rsf_model, "CSC Model"
= csc_model1),
  data = test_data,
  time_var = time_var,
  event_var = event_var,
  times = seq(2, 66, 2),
  fgr_model = fg_model,
  rsf_model = rsf_model,
  csc_model1 = csc_model1,
  csc_model2 = csc_model2,
  split.method = "none"
)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Output results
print(results1$auc_plot)
```

```
print(results1$brier_plot)
```

```
print(results1$cif_plot)
```

```
results1$calib_plot()
```

```
## Warning in plotCalibration(score_result): Time point not specified, use ma
x of
## the available times: 66
```

```
kable(results1$performance_table, digits = 3, caption = "Model Performance at
Selected Time Points")
```

```
# Distribution plots
auto_plot_config <- function(continuous_vars, categorical_vars) {
  all_vars <- c(continuous_vars, categorical_vars)
  types <- c(rep("histogram", length(continuous_vars)), rep("bar", length(cat
egorical_vars)))
  fills <- hcl.colors(length(all_vars), "Set2")

  setNames(
    Map(function(var, type, fill) {
      title <- tools::toTitleCase(var)
      list(type = type, fill = fill, title = title, xlab = title)
```

```
    }, all_vars, types, fills),
    all_vars
  )
}

plot_config1 <- auto_plot_config(continuous_vars, c(categorical_vars, "status
"))
create_distribution_plots(test_data, plot_config1, ncol = 3)
```

```
# Variable importance from RSF
vi_rsf <- rsf_model$importance
print(vi_rsf)

# Prediction wrapper for DALEX (CIF at fixed time point)
predict_rfsrc <- function(model, newdata, type = "cif", cause = 1, time = 66)
{
  pred <- predict.rfsrc(model, newdata = newdata, cause = cause)
  time_idx <- which.min(abs(pred$time.interest - time))
  if (type == "cif") {
    return(pred$cif[, time_idx, cause])
  } else if (type == "chf") {
    return(pred$chf[, time_idx, cause])
  } else {
    stop("Invalid 'type'. Use 'cif' or 'chf'.")
  }
}

predict_dalex <- function(model, newdata) {
  predict_rfsrc(model, newdata = newdata, type = "cif", cause = 1, time = 66)
}

# DALEX explainer for RSF
explainer_rsf <- DALEX::explain(
  model = rsf_model,
  data = train_data[, setdiff(names(train_data), survival_vars)],
  y = predict_rfsrc(rsf_model, train_data, type = "cif", cause = 1, time = 66
),
  predict_function = predict_dalex,
  label = "RSF Model"
)

## Preparation of a new explainer is initiated
##   -> model label       :  RSF Model
##   -> data              :  400  rows  4  cols
##   -> target variable   :  400  values
##   -> predict function  :  predict_dalex
##   -> predicted values  :  No value for predict function target column. (
default  )
```

```
##   -> model_info        :  package Model of class: rfsrc package unrecogniz
ed , ver. Unknown , task regression (  default  )
##   -> predicted values  :  numerical, min =  0.1381661 , mean =  0.4779825
, max =  0.9113953
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -5.551115e-16 , mean =  -1.339
207e-17 , max =  4.440892e-16
##   A new explainer has been created!

# Partial dependence plots (PDPs)
pdp_num <- model_profile(explainer_rsf, variables = continuous_vars)
pdp_plot <- plot(pdp_num, title = NULL, subtitle = NULL) + theme_minimal(base
_size = 12)

dp_cat <- variable_profile(explainer_rsf, variable = categorical_vars,
                           type = "accumulated", variable_type = "categorical
")
pdpCat_plot <- plot(dp_cat, title = NULL, subtitle = NULL)

# SHAP-like breakdown for a single patient
new_patient <- test_data[1, setdiff(names(test_data), survival_vars)]
bd <- predict_parts(explainer_rsf, new_observation = new_patient, type = "bre
ak_down")
shap_plot <- plot(bd, title = NULL, subtitle = NULL) + theme_minimal(base_siz
e = 12)

# Feature importance plot
vi <- model_parts(explainer_rsf, type = "variable_importance")
vi_plot <- plot(vi) + ggtitle("Feature Importance for 66-Month CIF Prediction
(Cause 1)")

# Arrange interpretability plots
grid.arrange(pdp_plot, pdpCat_plot, shap_plot, ncol = 3)
```