Package: ccwr (via r-universe)

October 10, 2026

Title Clone-Censor-Weight Methods for Target Trial Emulation

Version 0.0.2

Description Provides methods for clone-censor-weight analyses that emulate target trials with grace-period treatment strategies using observational time-to-event data. Tools clone participants across strategies, apply strategy-specific artificial censoring, and estimate inverse probability of censoring weights using pooled logistic or Cox models, fit weighted outcome models, and obtain subject-level bootstrap confidence intervals by repeating the complete analysis. The methods are described by Maringe et al. (2020) <doi:10.1093/ije/dyaa057> and Gaber et al. (2024) <doi:10.1002/cam4.70461>.

License MIT + file LICENSE

Copyright See inst/COPYRIGHTS

URL https://github.com/CausalInferenceLab/ccwr

BugReports https://github.com/CausalInferenceLab/ccwr/issues

Encoding UTF-8

Depends R (>= 4.4.0)

Imports dplyr, readr, survival, tibble, rlang, glue

Suggests knitr, rmarkdown, testthat (>= 3.2.0)

Config/testthat/edition 3

RoxygenNote 8.0.0

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Sang Ho Park [aut, cre], Youngrok Lee [aut], Jihyeon Baek [aut], Hye Won Yang [aut], Donghoon Jeong [aut]

Maintainer Sang Ho Park <shstat1729@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-09-03 14:38:29 UTC

RemoteUrl https://github.com/cran/ccwr

RemoteRef HEAD

RemoteSha 1fb6e3b2c29c1ea34f68a3eb7226fa0702667614

Contents

apply_logics 2
clone_arms 3
create_censoring_logics_A 4
create_final_data 5
create_policy_A 7
emul_estimate 8
emul_estimate_bootstrap 10
estimate_censoring 12
lungcancer 14
make_surv_response 15
patients13 15
read_trial_data 16
weight_cases 17

Index **19**

`apply_logics` *Apply case_when logics (e.g. policy, censoring) to clones*

Description

Apply case_when logics (e.g. policy, censoring) to clones

Usage

```
apply_logics(clones, logics)
```

Arguments

`clones`	A list of data frame. Each element of the list represents each treatment arm.
`logics`	A nested list. Each element of outer list represents each treatment arm. Each element of inner list represents each new variable to be created by applying logics. Each element of inner list containts a character vector that represents a sequence of logics to be passed into `case_when()` call to determine a value of the new variable.

Value

A list of data frame that each data frame include new variables created by the provided logics.

Examples

```
data(lungcancer)
arms <- c("Control", "Surgery")
clones <- clone_arms(lungcancer, arms)
policies <- create_policy_A(
  arms,
  treatment = "surgery",
  time_to_treatment = "timetosurgery",
  grace_period = 182.62,
  outcome = "death",
  followup = "fup_obs",
  clone_outcome = "outcome",
  clone_followup = "fup"
)
clones_policy <- apply_logics(clones, policies)
```

---

clone_arms *Duplicate data frame for each treatment arm to emulate*

Description

Duplicate data frame for each treatment arm to emulate

Usage

```
clone_arms(data, arms)
```

Arguments

data	Input data frame that contains all observations of interest. Each row represents an observation, and columns include observation identifiers, binary treatment variable (0/1), time to treatement (continuous), binary outcome variable (0/1), observed followup time (continuous), and covariates.
arms	Character vector that each element represents each arm's name.

Value

A list of data frame. Each element of list is associated with each arm.

Examples

```
data(lungcancer)
clones <- clone_arms(lungcancer, c("Control", "Surgery"))
```

`create_censoring_logics_A` *Generate censoring logic for scenario A*

Description

Generate censoring logic for scenario A

Usage

```
create_censoring_logics_A(
  arms,
  treatment,
  time_to_treatment,
  grace_period,
  followup,
  clone_censoring = ".censoring",
  clone_uncensored_followup = ".fup_uncensored"
)
```

Arguments

`arms`	A character vector of length 2. The first element represents a name of the untreated arm, and the second element represents a name of the treated arm.
`treatment`	A name of variable that represents whether each observation was treated or not in observational data. The treatment variable should exists in data frame that the return value of this function will be applied, and the treatment variable value in the data frame should be either 0 and 1, i.e. binary treatment.
`time_to_treatment`	A name of variable that represent time to treatment in the observational data. The time-to-treatment variable should exists in data frame that the return value of this function will be applied, and the value in the data frame should be either numeric value or NA if the observation was untreated in the observational data.
`grace_period`	A numeric value to represent grace period of treatment. Treatment policy is assumed to be "provide treatment within the grace period."
`followup`	A name of variable that represent follow up time in the observational data. The follow up time variable should exists in data frame that the return value of this function will be applied, and the follow up time variable value in the data frame should be numeric.
`clone_censoring`	A name of binary indicator variable to be newly created to represent whether the observation violates arm's policy or not. The new variable name should not already exist in the data frame that the return value of this function will be applied, to avoid accidental overwriting.

clone_uncensored_followup
: A name of a variable to be newly created for the strategy-specific follow-up time used by the censoring rules. It equals the observed follow-up time for a clone that remains adherent to its assigned strategy, and the artificial-censoring time for a clone that violates it. The new variable name should not already exist in the data frame that the return value of this function will be applied, to avoid accidental overwriting.

Value

A nested list. The first element of the outer list represents untreated arm, while the second element of the outer list represents treated arm. For each element of outer list, the first element of the inner list represents emulated censoring binary indicator (0/1) that represents whether the observation violated the arm's policy within the grace period. The second element of the inner list represents the strategy-specific follow-up time for the censoring process: observed follow-up for adherent clones and artificial-censoring time for non-adherent clones. Each element of the inner list represents a sequence of logics to be passed into `case_when()` when creating new variables for the emulated censoring indicator and follow-up time.

Examples

```
arms <- c("Control", "Surgery")
censoring_logics <- create_censoring_logics_A(
  arms,
  treatment = "surgery",
  time_to_treatment = "timetosurgery",
  grace_period = 182.62,
  followup = "fup_obs",
  clone_censoring = "censoring",
  clone_uncensored_followup = "fup_uncensored"
)
```

---

create_final_data *Create training data for censoring probability estimation*

Description

Create training data for censoring probability estimation

Usage

```
create_final_data(
  clones,
  clone_followup,
  clone_outcome,
  clone_censoring,
  col_ids,
  timestamp_start = "Tstart",
```

```
  id = "ID",
  timestamp_stop = "Tstop"
)
```

Arguments

clones	A list of data frame. Each element of the list represents each treatment arm. This version of clones must contain a column that represents an emulated follow up time (corresponding to clone_followup argument), an emulated outcome (correspodning to clone_outcome), and a binary indicator variable that represents whether the observation violates arm's policy or not (corresponding to clone_censoring argument).
clone_followup	A column name that represents the emulated follow up time in each arm of clones. The variable should exists in each element data frame of clones argument.
clone_outcome	A column name that represents the emulated outcome in each arm of clones. The variable should exists in each element data frame of clones argument, and the variable value should be binary (0 or 1).
clone_censoring	A column name that represent whether the observation violates arm's policy or not. The variable should exists in each element data frame of clones argument, and the variable value should be binary (0 or 1).
col_ids	A vector of column names that a combination of their values uniquely identifies each observation.
timestamp_start	A new variable name to denote start time of each subrecord of observations in a long-form data.
id	A new variable name for a unique observation identifier, to represents that multiple rows in output data frame is associated with the same observation.
timestamp_stop	A new variable name to denote end time of each subrecord of observations in a long-form data.

Value

A list of long-form data frames. Each data frame represents each clone arm. Each row of the long-form data frame represents a subrecord of each observation associated with each specific time interval. The first subrecord starts with time 0, and the rows are expanded up to clone_followup, where cut times are determined by t_events argument.

Examples

```
data(lungcancer)
arms <- c("Control", "Surgery")
clones <- clone_arms(lungcancer, arms)
policies <- create_policy_A(
  arms, "surgery", "timetosurgery", 182.62, "death", "fup_obs",
  clone_outcome = "outcome", clone_followup = "fup"
)
```

```
clones_policy <- apply_logics(clones, policies)
censoring_logics <- create_censoring_logics_A(
  arms, "surgery", "timetosurgery", 182.62, "fup_obs",
  clone_censoring = "censoring",
  clone_uncensored_followup = "fup_uncensored"
)
clones_censored <- apply_logics(clones_policy, censoring_logics)
clones_final <- create_final_data(
  clones_censored,
  clone_followup = "fup",
  clone_outcome = "outcome",
  clone_censoring = "censoring",
  col_ids = "id"
)
```

`create_policy_A` *Generate clone policy for scenario A*

Description

Generate clone policy for scenario A

Usage

```
create_policy_A(
  arms,
  treatment,
  time_to_treatment,
  grace_period,
  outcome,
  followup,
  clone_outcome = ".outcome",
  clone_followup = ".fup"
)
```

Arguments

`arms`	A character vector of length 2. The first element represents a name of the untreated arm, and the second element represents a name of the treated arm.
`treatment`	A name of variable that represents whether each observation was treated or not in observational data. The treatment variable should exists in data frame that the return value of this function will be applied, and the treatment variable value in the data frame should be either 0 and 1, i.e. binary treatment.
`time_to_treatment`	A name of variable that represent time to treatment in the observational data. The time-to-treatment variable should exists in data frame that the return value of this function will be applied, and the value in the data frame should be either numeric value or NA if the observation was untreated in the observational data.

grace_period	A numeric value to represent grace period of treatment. Treatment policy is assumed to be "provide treatment within the grace period."
outcome	A name of variable that represent outcome in the observational data. The outcome variable should exists in data frame that the return value of this function will be applied, and the outcome variable value in the data frame should be either 0 and 1, i.e. binary outcome.
followup	A name of variable that represent follow up time in the observational data. The follow up time variable should exists in data frame that the return value of this function will be applied, and the follow up time variable value in the data frame should be numeric.
clone_outcome	A name of variable to be newly created to represent emulated outcome in cloned data frame. The new variable name should not already exist in the data frame that the return value of this function will be applied, to avoid accidental overwriting.
clone_followup	A name of variable to be newly created to represent emulated follow up time in cloned data frame. The new variable name should not already exist in the data frame that the return value of this function will be applied, to avoid accidental overwriting.

Value

A nested list. The first element of the outer list represents untreated arm, while the second element of the outer list represents treated arm. For each element of outer list, the first element of the inner list represents emulated outcome, and the second element of the inner list represents emulated follow up time. Each element of the inner list represents a sequence of logics to be passed into `case_when()` when creating new variables for emulated outcome and follow up time.

Examples

```
arms <- c("Control", "Surgery")
policies <- create_policy_A(
  arms,
  treatment = "surgery",
  time_to_treatment = "timetosurgery",
  grace_period = 182.62,
  outcome = "death",
  followup = "fup_obs",
  clone_outcome = "outcome",
  clone_followup = "fup"
)
```

---

emul_estimate *Estimate the emulated trial effect*

Description

Estimate the emulated trial effect

Usage

```
emul_estimate(
  clones_weighted,
  method = c("Cox", "logistic", "KM"),
  cluster = "id",
  weights = NULL,
  predictors = NULL,
  outcome = "outcome",
  time_start = "Tstart",
  time_stop = "Tstop",
  arm = "arms"
)
```

Arguments

clones_weighted	A named list of weighted clone data frames, or a single data frame containing an arm column.
method	Analysis method: "Cox", "logistic", or "KM".
cluster	Column name used for robust clustering in Cox models.
weights	Optional weight column name. Unquoted column names are also accepted when they exist in the data.
predictors	Optional adjustment predictors. For KM, predictors define additional strata rather than covariate adjustment.
outcome	Column name for the outcome indicator.
time_start	Column name for interval start time.
time_stop	Column name for interval stop time.
arm	Column name for treatment arm after binding clone lists.

Value

A fitted model object: "coxph" for Cox, "glm" for logistic, or "survfit" for KM.

Examples

```
data(lungcancer)
arms <- c("Control", "Surgery")
clones <- clone_arms(lungcancer, arms)
policies <- create_policy_A(
  arms, "surgery", "timetosurgery", 182.62, "death", "fup_obs",
  clone_outcome = "outcome", clone_followup = "fup"
)
clones_policy <- apply_logics(clones, policies)
censoring_logics <- create_censoring_logics_A(
  arms, "surgery", "timetosurgery", 182.62, "fup_obs",
  clone_censoring = "censoring",
  clone_uncensored_followup = "fup_uncensored"
```

```
)
clones_censored <- apply_logics(clones_policy, censoring_logics)
clones_final <- create_final_data(
  clones_censored, "fup", "outcome", "censoring", "id"
)
clones_estimated <- estimate_censoring(
  clones_final,
  predictors = c("age", "sex"),
  method = "pooled_logit"
)
clones_weighted <- weight_cases(clones_estimated)
fit <- emul_estimate(
  clones_weighted,
  method = "Cox",
  weights = "weight_Cox",
  predictors = c("age", "sex")
)
```

`emul_estimate_bootstrap` *Bootstrap a complete clone-censor-weight analysis*

Description

Resamples subjects from the original subject-level data and repeats the complete analysis in every bootstrap replicate: cloning, strategy-specific artificial censoring, person-time expansion, censoring-model estimation, weighting, and outcome-model estimation.

Usage

```
emul_estimate_bootstrap(
  data,
  arms,
  id,
  treatment,
  time_to_treatment,
  grace_period,
  outcome,
  followup,
  censoring_predictors = NULL,
  censoring_method = c("pooled_logit", "stabilized_logit", "Cox"),
  numerator_predictors = NULL,
  censoring_time_spline_df = NULL,
  method = c("Cox", "logistic"),
  predictors = NULL,
  n_bootstrap = 200,
  conf_level = 0.95,
  eps = 1e-06,
```

```
  seed = NULL
)
```

Arguments

data	A data frame with one row per subject.
arms	A character vector of length two. The first element names the strategy of not receiving treatment during the grace period; the second names the strategy of receiving treatment during the grace period.
id	Name of the column that uniquely identifies subjects.
treatment	Name of the binary treatment column.
time_to_treatment	Name of the numeric time-to-treatment column.
grace_period	Positive numeric length of the treatment grace period.
outcome	Name of the binary outcome column.
followup	Name of the numeric follow-up-time column.
censoring_predictors	Optional character vector of predictors for the denominator censoring model.
censoring_method	Censoring model passed to estimate_censoring(): "pooled_logit", "stabilized_logit", or "Cox".
numerator_predictors	Optional character vector of predictors for the numerator model when censoring_method = "stabilized_logit".
censoring_time_spline_df	Degrees of freedom for the natural cubic spline of interval start time in pooled-logistic censoring models. Set to an integer of at least 2 to use a spline. The default NULL uses a linear time term. Ignored when censoring_method = "Cox".
method	Outcome analysis method: "Cox" or "logistic".
predictors	Optional adjustment predictors for the outcome model.
n_bootstrap	Number of bootstrap resamples.
conf_level	Confidence level for the percentile interval.
eps	Probability floor passed to estimate_censoring() and weight_cases().
seed	Optional random seed.

Value

A list containing the full-sample effect estimate, percentile confidence limits, bootstrap estimates, and analysis settings. Effects are returned on the exponentiated coefficient scale: a hazard ratio for Cox models and an odds ratio for logistic models.

Examples

```
data(lungcancer)
boot <- emul_estimate_bootstrap(
  lungcancer,
  arms = c("Control", "Surgery"),
  id = "id",
  treatment = "surgery",
  time_to_treatment = "timetosurgery",
  grace_period = 182.62,
  outcome = "death",
  followup = "fup_obs",
  censoring_predictors = c("age", "sex"),
  predictors = c("age", "sex"),
  n_bootstrap = 3,
  seed = 1
)
```

---

estimate_censoring *Estimate censoring probabilities*

Description

Estimate censoring probabilities

Usage

```
estimate_censoring(
  clones,
  predictors = NULL,
  method = c("Cox", "pooled_logit", "stabilized_logit"),
  numerator_predictors = NULL,
  censoring = "censoring",
  id = "id",
  time_start = "Tstart",
  time_stop = "Tstop",
  time_spline_df = NULL,
  eps = 1e-06
)
```

Arguments

clones	A named list of long-form clone data frames.
predictors	Optional character vector of denominator model predictors.
method	Censoring model. "Cox" fits a Cox censoring model, "pooled_logit" fits a pooled logistic denominator model, and "stabilized_logit" additionally fits a numerator model.

`numerator_predictors`
: Optional character vector of numerator model predictors for stabilized pooled-logit weights. When `NULL`, `predictors` are used.

`censoring`
: Column name for the censoring indicator.

`id`
: Column name for the subject identifier.

`time_start`
: Column name for interval start time.

`time_stop`
: Column name for interval stop time.

`time_spline_df`
: Degrees of freedom for the natural cubic spline of `time_start` in pooled-logistic censoring models. Set to an integer of at least 2 to use a spline. The default `NULL` uses a linear time term. Ignored when `method = "Cox"`.

`eps`
: Small probability floor to avoid division by zero.

Value

A named list of clone data frames with censoring probability columns added. All methods add `P_uncens`; pooled-logit methods also add `p_cens_den`, and stabilized pooled logit adds `p_cens_num` and `P_uncens_num`.

Examples

```
data(lungcancer)
arms <- c("Control", "Surgery")
clones <- clone_arms(lungcancer, arms)
policies <- create_policy_A(
  arms, "surgery", "timetosurgery", 182.62, "death", "fup_obs",
  clone_outcome = "outcome", clone_followup = "fup"
)
clones_policy <- apply_logics(clones, policies)
censoring_logics <- create_censoring_logics_A(
  arms, "surgery", "timetosurgery", 182.62, "fup_obs",
  clone_censoring = "censoring",
  clone_uncensored_followup = "fup_uncensored"
)
clones_censored <- apply_logics(clones_policy, censoring_logics)
clones_final <- create_final_data(
  clones_censored, "fup", "outcome", "censoring", "id"
)
clones_estimated <- estimate_censoring(
  clones_final,
  predictors = c("age", "sex"),
  method = "pooled_logit"
)
```

`lungcancer` *Simulated lung cancer patients data by Maringe et al. (2020)*

Description

The dataset is from the supplementary files of Maringe et al. (2020), doi:10.1093/ije/dyaa057. The dataset is a set of 200 simulated lung cancer patients. These patients are followed up for a year following their cancer diagnosis: 106 of them received surgery within six months of their diagnosis and 48 died in the year.

Usage

```
lungcancer
```

Format

`lungcancer`**:**
A data frame with 200 rows and 12 columns:

id patient identifier
fup_obs observed follow-up time (time to death or 1 year if censored alive)
death observed event of interest (all-cause death) 1: dead, 0:alive
timetosurgery time to surgery (NA if no surgery)
surgery observed treatment 1 if the patient received surgery within 6 month, 0 otherwise
age age at diagnosis
sex patient's sex
perf performance status at diagnosis
stage stage at diagnosis
deprivation deprivation score
charlson Charlson's comorbidity index
emergency route to diagnosis

Details

See `system.file("COPYRIGHTS", package = "ccwr")` for the source and copyright notice.

Source

doi:10.1093/ije/dyaa057

Examples

```
data(lungcancer)
```

make_surv_response *Construct a survival response*

Description

Construct a survival response

Usage

```
make_surv_response(data, follow_up, event)
```

Arguments

data	A data frame with follow-up and event columns.
follow_up	The name of the follow-up time column.
event	The name of the event indicator column.

Value

An object of class "Surv".

Examples

```
data(lungcancer)
surv_response <- make_surv_response(
  lungcancer,
  follow_up = "fup_obs",
  event = "death"
)
```

patients13 *13 types of patient records in Maringe et al. (2020)*

Description

The dataset is from Figure 2 in Maringe et al. (2020), doi:10.1093/ije/dyaa057. The dataset illustrates all possible censoring mechanisms with 13 types of patients records that could be seen in the cancer registry data, when allowing at most one treatment (surgery in this example) for each patient and outcome event (death in this example) happens at most once for each patient.

Usage

```
patients13
```

Format

`patients13`:

A data frame with 13 rows and 5 columns:

id patient identifier

surgery observed treatment 1 if the patient received surgery during follow-up, 0 otherwise

time_to_surgery time to surgery in days (NA if no surgery)

death observed event at the latest follow-up, 1: dead, 0: alive

followup observed follow-up time (time to death or time to latest followup with 1 year (365 days) at maximum)

Details

See `system.file("COPYRIGHTS", package = "ccwr")` for the source and copyright notice.

Source

doi:10.1093/ije/dyaa057

Examples

```
data(patients13)
```

---

read_trial_data *Read trial-style data from CSV*

Description

Read trial-style data from CSV

Usage

```
read_trial_data(file, show_col_types = FALSE)
```

Arguments

`file`	Path to a CSV file.
`show_col_types`	Passed to `readr::read_csv()`.

Value

A tibble.

Examples

```
csv_file <- tempfile(fileext = ".csv")
writeLines(
  c(
    "id,follow_up,event,treatment",
    "1,10,1,A",
    "2,12,0,B"
  ),
  con = csv_file
)
trial_data <- read_trial_data(csv_file)
```

`weight_cases` *Add inverse probability of censoring weights*

Description

Add inverse probability of censoring weights

Usage

```
weight_cases(
  clones,
  uncensored_prob = "P_uncens",
  numerator_uncensored_prob = "P_uncens_num",
  weight = "weight_Cox",
  eps = 1e-06
)
```

Arguments

`clones`	A named list of clone data frames returned by `estimate_censoring()`.
`uncensored_prob`	Column name for denominator uncensoring probability.
`numerator_uncensored_prob`	Column name for numerator uncensoring probability. When this column exists, stabilized weights are created.
`weight`	Column name for the output weight.
`eps`	Small probability floor to avoid division by zero.

Value

A named list of clone data frames with the weight column added.

Examples

```
data(lungcancer)
arms <- c("Control", "Surgery")
clones <- clone_arms(lungcancer, arms)
policies <- create_policy_A(
  arms, "surgery", "timetosurgery", 182.62, "death", "fup_obs",
  clone_outcome = "outcome", clone_followup = "fup"
)
clones_policy <- apply_logics(clones, policies)
censoring_logics <- create_censoring_logics_A(
  arms, "surgery", "timetosurgery", 182.62, "fup_obs",
  clone_censoring = "censoring",
  clone_uncensored_followup = "fup_uncensored"
)
clones_censored <- apply_logics(clones_policy, censoring_logics)
clones_final <- create_final_data(
  clones_censored, "fup", "outcome", "censoring", "id"
)
clones_estimated <- estimate_censoring(
  clones_final,
  predictors = c("age", "sex"),
  method = "pooled_logit"
)
clones_weighted <- weight_cases(clones_estimated)
```

Index

∗ **datasets**
lungcancer, 14
patients13, 15

apply_logics, 2

clone_arms, 3
create_censoring_logics_A, 4
create_final_data, 5
create_policy_A, 7

emul_estimate, 8
emul_estimate_bootstrap, 10
estimate_censoring, 12
estimate_censoring(), *11*

lungcancer, 14

make_surv_response, 15

patients13, 15

read_trial_data, 16
readr::read_csv(), *16*

weight_cases, 17
weight_cases(), *11*