Package 'autoCovariateSelection'

May 7, 2026

Type Package

Title Automated Covariate Selection Using HDPS Algorithm

Version 1.0.0

Author Dennis Robert <dennis.robert.nm@gmail.com>

Maintainer Dennis Robert <dennis.robert.nm@gmail.com>

Description

Contains functions to implement automated covariate selection using methods described in the high-dimensional propensity score (HDPS) algorithm by Schneeweiss et.al. Covariate adjustment in real-world-observational-data (RWD) is important for
for estimating adjusted outcomes and this can be done by using methods such as, but not limited to, propensity score
matching, propensity score weighting and regression analysis. While these methods strive to statistically adjust for
confounding, the major challenge is in selecting the potential covariates that can bias the outcomes comparison estimates
in observational RWD (Real-World-
Data). This is where the utility of automated covariate selection comes in.
The functions in this package help to implement the three major steps of automated covariate selection as described by
Schneeweiss et. al elsewhere. These three functions, in order of the steps required to execute automated covariate
selection are, get_candidate_covariates(), get_recurrence_covariates() and get_prioritised_covariates().
In addition to these functions, a sample real-world-data from publicly available de-identified medical claims data is
also available for running examples and also for further exploration. The original article where the algorithm is described
by Schneeweiss et.al. (2009) <doi:10.1097/EDE.0b013e3181a663cc> .

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/technOslerphile/autoCovariateSelection

BugReports https://github.com/technOslerphile/autoCovariateSelection/issues

Imports purrr, data.table

Depends dplyr, R (>= 2.10)

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2020-12-14 09:50:11 UTC

Contents

`get_candidate_covariates` *Generate candidate empirical baseline covariates based on prevalence in the baseline period*

Description

`get_candidate_covariates` function generates the list of candidate empirical covariates based on their prevalence within each domains (dimensions). This is the first step in the automated covariate selection process. See 'Automated Covariate Selection' section below for more details regarding the overall process.

Usage

```
get_candidate_covariates(
  df,
  domainVarname,
  eventCodeVarname,
  patientIdVarname,
  patientIdVector,
  n = 200,
  min_num_patients = 100
)
```

Arguments

df	The input data.frame. This should contain at least 3 fields containing information on patient identifier, covariate codes and domain names of covariate codes in a long format. Any other fields containing values such as dates, treatment group are optional and will be ignored for this analysis
domainVarname	The variable(field) name which contains the domain of the covariate in the df. The domains are usually diagnosis, procedures and medications.
eventCodeVarname	The variable name which contains the covariate codes (eg:- CCS, ICD9) in the df
patientIdVarname	The variable name which contains the patient identifier in the df
patientIdVector	The 1-D vector with all the patient identifiers. The length of this vector should be equal to the number of distinct patients in the df. This vector is not really used in the function analysis per se. This is used only to return the same back as function output because the filtered df based on covars will likely not contain all patients in the input df because there could be patients for whom no records were found for any of the identified covars and they will thus be not present in the filtered df which is also an output of this function. The patientIds vector output will contain all original patients and by returning this vector, it can later be used in the next steps of automated covariate selection because each step is dependent on previous steps and information on patients who did not have any identified covars is also important for the next steps. This is why this vector is an input as well as an output, without affecting the analysis of this function.
n	The maximum number of empirical candidate baseline covariates that should be returned within each domain. By default, n is 200
min_num_patients	Minimum number of patients that should be present for each covariate to be selected for selection. To be considered for selection, a covariate should have occurred for a minimum min_num_patients in the baseline period

Details

The theoretical details of the high-dimensional propensity score (HDPS) algorithm is detailed in the publication listed below in the References section. get_candidate_covariates is the function implementing what is described in the 'Identify candidate empirical covariates' section of the article.

Value

A named list containing three R objects

- covars A 1-D vector containing the names of selected baseline covariate names from each domain. For each domain in the df, the number of covars would be equal to or less than n
- covars_data The data.frame that is filtered out of df with only the selected covars. The values of the eventCodeVarname field is prefixed with the corresponding domain name. For

example, if the event code is 19900 and the domain is 'dx', then the the covariate name will be 'dx_19900'.

- `patientIds` The list of patient ids present in the original input `df`. This is exactly the same as the input `patientIdVector`

Automated Covariate Selection

The three steps in automated covariate selection are listed below with the functions implementing the methodology

1. Identify candidate empirical covariates: `get_candidate_covariates`
2. Assess recurrence: `get_recurrence_covariates`
3. Prioritize covariates: `get_prioritised_covariates`

Author(s)

Dennis Robert <dennis.robert.nm@gmail.com>

References

Schneeweiss S, Rassen JA, Glynn RJ, Avorn J, Mogun H, Brookhart MA. High-dimensional propensity score adjustment in studies of treatment effects using health care claims data Epidemiology. 2009;20(4):512-522. doi:10.1097/EDE.0b013e3181a663cc

Examples

```
library("autoCovariateSelection")
data(rwd)
head(rwd, 3)
#select distinct elements that are unique for each patient - treatment and outcome
basetable <- rwd %>% select(person_id, treatment, outcome_date) %>% distinct()
head(basetable, 3)
patientIds <- basetable$person_id
step1 <- get_candidate_covariates(df = rwd,  domainVarname = "domain",
eventCodeVarname = "event_code", patientIdVarname = "person_id",
patientIdVector = patientIds,n = 100, min_num_patients = 10)
out1 <- step1$covars_data #this will be input to get_recurrence_covariates() function
```

---

get_prioritised_covariates

Generate the prioritised covariates from the global list of binary recurrence covariates using multiplicative bias ranking

---

Description

get_prioritised_covariates function assesses the recurrence of each of the identified candidate empirical covariates based on their frequency of occurrence for each patient in the baseline period and generates three binary recurrence covariates for each of the identified candidate empirical covariates. This is the third and final step in the automated covariate selection process. The previous step of assessing recurrence and generating the binary recurrence covariates is done using the get_recurrence_covariates function. See 'Automated Covariate Selection'section below for more details regarding the overall process.

Usage

```
get_prioritised_covariates(
  df,
  patientIdVarname,
  exposureVector,
  outcomeVector,
  patientIdVector,
  k = 500
)
```

Arguments

df	The input data.frame. Ideally this should be the output recurrence_data from the get_recurrence_covariates function
patientIdVarname	The variable name which contains the patient identifier in the df
exposureVector	The 1-D exposure (treatment/intervention) vector. The length of this vector should be equal to that of patientIdVector and outcomeVector. Also, this should be a binary vector with value of 1 for patients primary cohort 1 and 0 for those in comparator cohort. The order of this vector should resonate the order of patients in outcomeVector and patientIdVector
outcomeVector	The 1-D outcome vector indicating whether or not the patient experienced the outcome of interest (value = 1) or not (value =0). The length of this vector should be equal to that of patientIdVector and exposureVector. The order of elements in this vector should resonate with the order of patients in exposureVector and patientIdVector
patientIdVector	The 1-D vector with all the patient identifiers. This should contain all the patient IDs in the original two cohorts with its length and order equal to and resonating with that of exposureVector and outcomeVector
k	The maximum number of prioritised covariates that should be returned by the function. By default, this is 500 as described in the original paper

Details

To prioritise covariates across data dimensions (domains) should be assessed by their potential for controlling confounding that is not conditional on exposure and other covariates. This means that

the association of the covariates with the outcomes (relative risk) should be taken into consideration for quantifying the 'potential' for confounding. Relative risk weighted by the ratio of prevalence of the covariates between the two exposure groups is known as multiplicative bias. The other way to do this would be to use the absolute risk and this would have been the rather straight-forward procedure to quantify the potential for confounding. However, this method would invariably down-weight the association between the covariate and the outcome if the outcome prevalence is small and the exposure prevalence is high which is a common phenomenon seen with comparative effective research using real-world-data by retrospective cohort studies. The multiplicative bias term balances this and generates a quantity for each covariate that is reflective of its confounding potential. By ranking the multiplicative bias, the objective is to choose the top `k` number of covariates from this procedure. `k`, by default, is 500 as described in the original paper. For further theoretical details of the algorithm please refer to the original article listed below in the `References` section. `get_recurrence_covariates` is the function implementing what is described in the 'Prioritise Covariates' section of the article.

Value

A named list containing two R objects

- `autoselected_covariate_df` A `data.frame` in wide format containing the auto-selected prioritised covariates and their values (1 or 0) for each patients
- `multiplicative_bias`The absolute log of the multiplicative bias term for each of the auto-selected prioritised covariates

Automated Covariate Selection

The three steps in automated covariate selection are listed below with the functions implementing the methodology

1. Identify candidate empirical covariates: `get_candidate_covariates`
2. Assess recurrence: `get_recurrence_covariates`
3. Prioritize covariates: `get_prioritised_covariates`

Author(s)

Dennis Robert <dennis.robert.nm@gmail.com>

References

Schneeweiss S, Rassen JA, Glynn RJ, Avorn J, Mogun H, Brookhart MA. High-dimensional propensity score adjustment in studies of treatment effects using health care claims data Epidemiology. 2009;20(4):512-522. doi:10.1097/EDE.0b013e3181a663cc

Examples

```
library("autoCovariateSelection")
data(rwd)
head(rwd, 3)
basetable <- rwd %>% select(person_id, treatment, outcome_date) %>% distinct()
head(basetable, 3)
```

```
patientIds <- basetable$person_id
step1 <- get_candidate_covariates(df = rwd,  domainVarname = "domain",
eventCodeVarname = "event_code" , patientIdVarname = "person_id",
patientIdVector = patientIds,n = 100, min_num_patients = 10)
out1 <- step1$covars_data
all.equal(patientIds, step1$patientIds) #should be TRUE
step2 <- get_recurrence_covariates(df = out1,
patientIdVarname = "person_id", eventCodeVarname = "event_code",
patientIdVector = patientIds)
out2 <- step2$recurrence_data
out3 <- get_prioritised_covariates(df = out2,
patientIdVarname = "person_id", exposureVector = basetable$treatment,
outcomeVector = ifelse(is.na(basetable$outcome_date), 0,1),
patientIdVector = patientIds, k = 10)
```

`get_recurrence_covariates` *Generate the binary recurrence covariates for the identified candidate empirical covariates*

Description

`get_recurrence_covariates` function assesses the recurrence of each of the identified candidate empirical covariates based on their frequency of occurrence for each patient in the baseline period and generates three binary recurrence covariates for each of the identified candidate empirical covariates. This is the second step in the automated covariate selection process. The first step of identifying empirical candidate covariates is done via `get_candidate_covariates` function. See 'Automated Covariate Selection'section below for more details regarding the overall process.

Usage

```
get_recurrence_covariates(
  df,
  patientIdVarname,
  eventCodeVarname,
  patientIdVector
)
```

Arguments

`df`	The input `data.frame`. Ideally this should be the output `covars_data` from `get_candidate_covariates`
`patientIdVarname`	The variable name which contains the patient identifier in the `df`
`eventCodeVarname`	The variable name which contains the covariate codes (eg:- CCS, ICD9) in the `df`

`patientIdVector`
: The 1-D vector with all the patient identifiers. This should contain all the patient IDs in the original two cohorts. This vector can simply be the `patientIds` output vector of the `get_candidate_covariates` function. of the function

Details

The recurrence covariates are generated based on the frequency (counts) of occurrence of each empirical candidate covariates that got generated by the `generate_candidate_covariates` function. This is done by looking at the baseline period of each patients and assessing whether the covariate occurred only once or sporadically or frequently. That is, a maximum of three recurrence covariates for each candidate covariate is created and returned.

- `once` Indicates whether or not the covariate occurred more than or equal to 1 number of times for the patient
- `sporadic` Indicates whether or not the covariate occurred more than or equal to median (median of non-zero occurrences of the candidate covariate) number of times for the patient.
- `frequent` Indicates whether or not the covariate occurred more than or equal to upper quartile (75th percentile of non-zero occurrences of the candidate covariate) number of times for the patient

Note that if two or all three covariates are identical for any of the binary recurrence covariates, only the distinct recurrence covariate is returned. For example, if once == sporadic == frequent for the candidate covariate (median and upper quartile both are 1), then only the 'once' recurrence covariate is returned. If once != sporadic == frequent, then 'once' and 'sporadic' is returned. If once == sporadic != frequent, then 'once' and 'frequent' are returned. If none of three recurrence covariates are identical, then all three are returned. The theoretical details of the algorithm implemented is detailed in the publication listed below in the `References` section. `get_recurrence_covariates` is the function implementing what is described in the 'Assess Recurrence' section of the article.

Value

A named list containing two R objects

- `recurrence_data` A `data.frame` containing all the binary recurrence covariates for all the patients in wide format. This means that this `data.frame` will have a dimension with number of rows equal to number of distinct patients and number of columns equal to number of binary recurrence covariates plus 1 (for the patient Id variable). The binary recurrence covariate is prefixed with a 'rec_' to indicate that the covariate is a 'reccurrence covariate' and suffixed with '_once', '_sporadic' or '_frequent'. See `details` section above for details.
- `patientIds` The list of patient ids present in the original input `df`. This is exactly the same as the input `patientIdVector`

Automated Covariate Selection

The three steps in automated covariate selection are listed below with the functions implementing the methodology

1. Identify candidate empirical covariates: `get_candidate_covariates`
2. Assess recurrence: `get_recurrence_covariates`
3. Prioritize covariates: `get_prioritised_covariates`

Author(s)

Dennis Robert <dennis.robert.nm@gmail.com>

References

Schneeweiss S, Rassen JA, Glynn RJ, Avorn J, Mogun H, Brookhart MA. High-dimensional propensity score adjustment in studies of treatment effects using health care claims data Epidemiology. 2009;20(4):512-522. doi:10.1097/EDE.0b013e3181a663cc

Examples

```
library("autoCovariateSelection")
data(rwd)
head(rwd, 3)
basetable <- rwd %>% select(person_id, treatment, outcome_date) %>% distinct()
head(basetable, 3)
patientIds <- basetable$person_id
step1 <- get_candidate_covariates(df = rwd,  domainVarname = "domain",
eventCodeVarname = "event_code" , patientIdVarname = "person_id",
patientIdVector = patientIds,n = 100, min_num_patients = 10)
out1 <- step1$covars_data
all.equal(patientIds, step1$patientIds) #should return  TRUE
step2 <- get_recurrence_covariates(df = out1, patientIdVarname = "person_id",
eventCodeVarname = "event_code", patientIdVector = patientIds)
out2 <- step2$recurrence_data
```

---

get_relative_risk *Compute relative risk for each of the covariates with respect to outcomes occurred*

Description

`get_relative_risk` function is a helper function used within the `get_prioritised_covariates` function. This function computes the prevalence in the exposed and that in the unexposed and simply returns the relative risk for all the covariates in the input `data.frame`

Usage

```
get_relative_risk(df, outcomeVec)
```

Arguments

`df`	The input `data.frame`. Ideally this should be the output `recurrence_data` from the `get_recurrence_covariates` function. The first column should be the patient identifier column and all other columns should be binary covariates. The values of these binary columns should be 1 indicating occurrence of covariate and 0 indicating no occurrence of the covariate.

`outcomeVec` The 1-D outcome vector indicating whether or not the patient experienced the outcome of interest (value = 1) or not (value =0). The length of this vector should be equal to the number of rows of `df`. The order of elements in this vector should resonate with the order of patients in `df`

Value

A 1-D vector containing relative risk of the association between the covariate (confounder) and the outcome. Thus, the length of this vector will be equal to the number of covariates.

Author(s)

Dennis Robert <dennis.robert.nm@gmail.com>

`rwd` *Sample Data for autoCovariateSelection*

Description

This is data contains Medicare claims data of a small sample of 1000 patients from the publicly available CMS Medicare De-SynPUF data. It contains all data from three domains - diagnosis, procedures and medications. The diagnosis codes are ICD9 codes, procedures are CPT4/HCPCS codes and medications are NDC codes.

Usage

```
rwd
```

Format

A data frame with 69333 rows and 9 variables:

person_id patient_identifier

index_date Date of first exposure. For one patient, there will only be one index_date

event_date Date at which event_code occurred for the patient

event_code The medical coding of the event. These are ICD9, CPT4, HCPCS or NDC codes depending on the `domain`

event_concept_id Another identifier for the `event_code`. This is irrelevant for this package and you can ignore it

domain The domain to which the `event_code` belongs to. The three unique values are dx (for diagnosis), px (for procedure) and rx (for medication)

treatment Binary indicator treatment allocation based on exposure. 1 indicates primary cohort and 0 for control/comparator cohort

outcome_date Date in which the outcome occurred. `NA` indicates no outcome occurred. In this sample data, the outcome is death

last_enrollment_date Last enrolled date of the patient. This field is irrelevant for this package and you can ignore it ...

Index