Package ‘CoOL’

February 23, 2021

Type Package

Title Causes of Outcome Learning

Version 1.0.1

Date 2021-02-23

Maintainer Andreas Rieckmann <aric@sund.ku.dk>


URL https://bioconductor.org

License GPL-2

Imports Rcpp, data.table, pROC, graphics, mltools, stats, plyr, ggplot2, ClustGeo, wesanderson

Suggests ggtree, imager

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation yes

Author Andreas Rieckmann [aut, cre], Piotr Dworzynski [aut], Leila Arras [ctb], Claus Thorn Ekstrom [aut]

Repository CRAN

Date/Publication 2021-02-23 15:10:02 UTC
**Description**

This function binary encodes the exposure data set so that each category is coded 0 and 1 (e.g. the variable sex will be two variables men (1/0) and women (0/1)).

**Usage**

```
CoOL_0_binary_encode_exposure_data(exposure_data)
```

**Arguments**

- `exposure_data` The exposure data set.

**Value**

Data frame with the expanded exposure data, where all variables are binary encoded.
References


Examples

#See the example under CoOL_0_working_example

---

CoOL_0_common_simulation

Common example

---

Description

To reproduce the common causes example.

Usage

CoOL_0_common_simulation(n)

Arguments

n  number of observations for the synthetic data.

Value

A data frame with the columns Y, A, B, C, D, E, F and n rows.

References

CoOL_0_complex_simulation

**Complex example**

Description
To reproduce the complex example.

Usage
CoOL_0_complex_simulation(n)

Arguments
n  number of observations for the synthetic data.

Value
A data frame with the columns Y, Physically_active, Low_SES, Mutation_X, LDL, Night_shifts, Air_pollution and n rows.

References

CoOL_0_confounding_simulation

**Confounding example**

Description
To reproduce the confounding example.

Usage
CoOL_0_confounding_simulation(n)

Arguments
n  number of observations for the synthetic data.

Value
A data frame with the columns Y, A, B, C, D, E, F and n rows.
References


---

CoOL_0_mediation_simulation

**Mediation example**

**Description**

To reproduce the mediation example.

**Usage**

CoOL_0_mediation_simulation(n)

**Arguments**

- **n**
  
  number of observations for the synthetic data.

**Value**

A data frame with the columns Y, A,B,C, D, E, F and n rows.

**References**


---

CoOL_0_working_example

**CoOL working example with sex, drug A, and drug B**

**Description**

To reproduce the CoOL working example with sex, drug A, and drug B.

**Usage**

CoOL_0_working_example(n)

**Arguments**

- **n**
  
  number of observations for the synthetic data.
Value

A data frame with the columns Y, sex, drug_a, drug_b and rows equal to n.

References


Examples

```r
while (FALSE) {
  library(CoOL)
  set.seed(1)
  data <- CoOL_0_working_example(n=10000)
  outcome_data <- data[,1]
  exposure_data <- data[,-1]
  exposure_data <- CoOL_0_binary_encode_exposure_data(exposure_data)
  model <- CoOL_1_initiate_neural_network(inputs=ncol(exposure_data),
                                          output = outcome_data,hidden=5)
  model <- CoOL_2_train_neural_network(lr = 1e-4,X_train=exposure_data,
                                       Y_train=outcome_data,X_test=exposure_data, Y_test=outcome_data,
                                       model=model, epochs=1000,patience = 200, input_parameter_reg = 1e-3
                                       ) # Train the non-negative model (The model can be retrained)
  model <- CoOL_2_train_neural_network(lr = 1e-5,X_train=exposure_data,
                                       Y_train=outcome_data,X_test=exposure_data, Y_test=outcome_data, model=model,
                                       epochs=1000,patience = 100, input_parameter_reg = 1e-3)
  # Train the non-negative model (The model can be retrained)
  model <- CoOL_2_train_neural_network(lr = 1e-6,X_train=exposure_data,
                                       Y_train=outcome_data,X_test=exposure_data, Y_test=outcome_data, model=model,
                                       epochs=1000,patience = 50, input_parameter_reg = 1e-3)
  # Train the non-negative model (The model can be retrained)
  plot(model$train_performance,type='l',yaxs='i',ylab="Mean squared error",
       xlab="Epochs",main="A) Performance during training\n\n", ylim=quantile(model$train_performance,c(0,.975))) # Model performance
  CoOL_3_plot_neural_network(model,names(exposure_data),5/max(model[[1]]),
                           title = "B) Model connection weights\nand intercepts") # Model visualization
  CoOL_4_AUC(outcome_data,exposure_data,model,
              title = "C) Receiver operating\ncharacteristic curve") # AUC
  risk_contributions <- CoOL_5_layerwise_relevance_propagation(exposure_data,model
                                                               ) # Risk contributions
  CoOL_6_dendrogram(risk_contributions,number_of_subgroups = 3,
                     title = "D) Dendrogram with 3 sub-groups") # Dendrogram
  sub_groups <- CoOL_6_sub_groups(risk_contributions,number_of_subgroups = 3
                                   ) # Assign sub-groups
  CoOL_7_prevalence_and_mean_risk_plot(risk_contributions,sub_groups,
                                        title = "E) Prevalence and mean risk of sub-groups") # Prevalence and mean risk plot
  CoOL_8_mean_risk_contributions_by_sub_group(risk_contributions,
                                              sub_groups,outcome_data = outcome_data,exposure_data = exposure_data,
                                              model=model,exclude_below = 0.01) # Mean risk contributions by sub-groups
}
```
CoOL_1_initiate_neural_network

*Initiates a non-negative neural network*

**Description**

This function initiates a non-negative neural network. The one-hidden layer non-negative neural network is designed to resemble a DAG with hidden synergistic components. With the model, we intend to learn the various synergistic interactions between the exposures and outcome. The model needs to be non-negative and estimate the risk on an additive scale. Neural networks include hidden activation functions (if the sum of the input exceeds a threshold, information is passed on), which can model minimum threshold values of interactions between exposures. We need to specify the upper limit of the number of possible hidden activation functions and through model fitting, the model may be able to learn both stand-alone and synergistically interacting factors.

**Usage**

```r
CoOL_1_initiate_neural_network(inputs, output, hidden = 10)
```

**Arguments**

- `inputs`: The number of exposures.
- `output`: The output variable is used to calculate the mean of it used to initiate the baseline risk.
- `hidden`: Number of hidden nodes.

**Details**

The non-negative neural network can be denoted as:

\[
P(Y = 1|X^+) = \sum_j \left( w^{+}_{j,k} \cdot \text{ReLU}_j \left( \sum_i (w^{+}_{i,j} \cdot X_i^+) + b_j^- \right) \right) + R_b
\]

**Value**

A list with connection weights, bias weights and meta data.

**References**


**Examples**

#See the example under CoOL_0_working_example
Description

This function trains the non-negative neural network. Fitting the model is done in a step-wise procedure one individual at a time, where the model estimates individual’s risk of the disease outcome, estimates the prediction’s residual error and adjusts the model parameters to reduce this error. By iterating through all individuals for multiple epochs (one complete iterations through all individuals is called an epoch), we end with parameters for the model, where the errors are smallest possible for the full population. The model fit follows the linear expectation that synergism is a combined effect larger than the sum of independent effects. The initial values, derivatives, and learning rates are described in further detail in the Supplementary material. The non-negative model ensures that the predicted value cannot be negative. The model does not prevent estimating probabilities above 1, but this would be unlikely, as risks of disease and mortality even for high risk groups in general are far below 1. The use of a test dataset does not seem to assist deciding on the optimal number of epochs possibly due to the constrains due to the non-negative assumption. We suggest splitting data into a train and test data set, such that findings from the train data set can be confirmed in the test data set before developing hypotheses.

Usage

CoOL_2_train_neural_network(
  X_train,
  Y_train,
  X_test,
  Y_test,
  model,
  lr = c(1e-04, 1e-05, 1e-06),
  epochs = 2000,
  patience = 100,
  monitor = TRUE,
  plot_and_evaluation_frequency = 50,
  input_parameter_reg = 0.001,
  spline_df = 10,
  restore_par_options = TRUE,
  drop_out = 0
)

Arguments

X_train The exposure data for the training data.
Y_train The outcome data for the training data.
X_test The exposure data for the test data (currently the training data is used).
Y_test The outcome data for the test data (currently the training data is used).
model  The fitted non-negative neural network.

lr    Learning rate (several LR can be provided, such that the model training will train for each LR and continue to the next).

epochs Epochs.

patience The number of epochs allowed without an improvement in performance.

monitor Whether a monitoring plot will be shown during training.

plot_and_evaluation_frequency The interval for plotting the performance and checking the patience.

input_parameter_reg Regularisation decreasing parameter value at each iteration for the input parameters.

spline_df Degrees of freedom for the spline fit for the performance plots.

restore_par_options Restore par options.

drop_out To drop connections if their weights reaches zero.

Value
An updated list of connection weights, bias weights and meta data.

References

Examples
#See the example under CoOL_0_working_example

CoOL_3_plot_neural_network

Plotting the non-negative neural network

Description
This function plots the non-negative neural network.

Usage
CoOL_3_plot_neural_network(
  model,
  names,
  arrow_size = NA,
  title = "Model connection weights and intercepts",
  restore_par_options = TRUE
)
Arguments

model The fitted non-negative neural network.
names Labels of each exposure.
arrow_size Define the arrow_size for the model illustration in the reported training progress.
title Title on the plot.
restore_par_options Restore par options.

Value

A plot visualizing the connection weights.

References


Examples

#See the example under CoOL_0_working_example

---

CoOL_4_AUC | Plot the ROC AUC

Description

Plot the ROC AUC

Usage

CoOL_4_AUC(
  outcome_data,
  exposure_data,
  model,
  title = "Receiver operating\ncharacteristic curve",
  restore_par_options = TRUE
)

Arguments

outcome_data The outcome data.
exposure_data The exposure data.
model The fitted non-negative neural network.
title Title on the plot.
restore_par_options Restore par options.
**Value**

A plot of the ROC and the ROC AUC value.

**References**


**Examples**

#See the example under CoOL_0_working_example

---

**Description**

Predict the risk of the outcome using the fitted non-negative neural network.

**Usage**

CoOL_4_predict_risks(X, model)

**Arguments**

- `X` The exposure data.
- `model` The fitted the non-negative neural network.

**Value**

A vector with the predicted risk of the outcome for each individual.

**References**


**Examples**

#See the example under CoOL_0_working_example
CoOL_5_layerwise_relevance_propagation

Layer-wise relevance propagation of the fitted non-negative neural network

Description

Calculates risk contributions for each exposure and a baseline using layer-wise relevance propagation of the fitted non-negative neural network and data.

Usage

CoOL_5_layerwise_relevance_propagation(X, model)

Arguments

X  The exposure data.
model  The fitted the non-negative neural network.

Details

For each individual:

\[ P(Y = 1|X^+) = R^b + \sum R^X_i \]

The below procedure is conducted for all individuals in a one by one fashion. The baseline risk, $R^b$, is simply parameterised in the model. The decomposition of the risk contributions for exposures, $R^X_i$, takes 3 steps:

Step 1 - Subtract the baseline risk, $R^b$:

\[ R^X_k = P(Y = 1|X^+) - R^b \]

Step 2 - Decompose to the hidden layer:

\[ R^X_j = \frac{H_j w_{j,k}}{\sum_j (H_j w_{j,k})} R^X_k \]

Where $H_j$ is the value taken by each of the $ReLU()_j$ functions for the specific individual.

Step 3 - Hidden layer to exposures:

\[ R^X_i = \sum_j \left( \frac{X^+_i w_{i,j}}{\sum_i (X^+_i w_{i,j})} R^X_j \right) \]

This creates a dataset with the dimensions equal to the number of individuals times the number of exposures plus a baseline risk value, which can be termed a risk contribution matrix. Instead of exposure values, individuals are given risk contributions, $R^X_i$. 
Value

A data frame with the risk contribution matrix [number of individuals, risk contributors + the baseline risk].

References


Examples

#See the example under CoOL_0_working_example

Dendrogram and sub-groups

Description

Calculates presents a dendrogram coloured by the pre-defined number of sub-groups and provides the vector with sub-groups.

Usage

CoOL_6_dendrogram(
  risk_contributions,
  number_of_subgroups = 3,
  title = "Dendrogram",
  colours = NA
)

Arguments

risk_contributions
  The risk contributions.
number_of_subgroups
  The number of sub-groups chosen (Visual inspection is necessary).
title
  The title of the plot.
colours
  Colours indicating each sub-group.

Value

A dendrogram illustrating similarities between individuals based on their risk contributions.

Examples

#See the example under CoOL_0_working_example
CoOL_6_individual_effects_matrix

*Risk contribution matrix based on individual effects (had all other exposures been set to zero)*

**Description**

Estimating the risk contribution for each exposure if each individual had been exposed to only one exposure, with the value the individual actually had.

**Usage**

CoOL_6_individual_effects_matrix(X, model)

**Arguments**

- **X**:
  - The exposure data.
- **model**:
  - The fitted the non-negative neural network.

**Value**

A matrix [Number of individuals, exposures] with the estimated individual effects by each exposure had all other values been set to zero.

**References**


**Examples**

#See the example under CoOL_0_working_example

---

CoOL_6_sub_groups

*Assign sub-groups*

**Description**

Calculates presents a dendrogram coloured by the pre-defined number of sub-groups and provides the vector with sub-groups.

**Usage**

CoOL_6_sub_groups(risk_contributions, number_of_subgroups = 3)
**Arguments**

- `risk_contributions`  
  The risk contributions.
- `number_of_subgroups`  
  The number of sub-groups chosen (Visual inspection is necessary).

**Value**

A vector `[number of individuals]` with an assigned sub-group.

**References**


**Examples**

#See the example under CoOL_0_working_example

---

`CoOL_6_sum_of_individual_effects`

*Predict the risk based on the sum of individual effects*

**Description**

By summing through the risk as if each individual had been exposed to only one exposure, with the value the individual actually had.

**Usage**

`CoOL_6_sum_of_individual_effects(X, model)`

**Arguments**

- `X`  
  The exposure data.
- `model`  
  The fitted the non-negative neural network.

**Value**

A value the sum of individual effects, had there been no interactions between exposures.

**References**

CoOL_7_prevalence_and_mean_risk_plot

Examples

#See the example under CoOL_0_working_example

CoOL_7_prevalence_and_mean_risk_plot

Description

This plot shows the prevalence and mean risk for each sub-group. Its distribution hits at sub-groups with great public health potential.

Usage

CoOL_7_prevalence_and_mean_risk_plot(
  risk_contributions,
  sub_groups,
  title = "Prevalence and mean risk
of sub-groups",
  y_max = NA,
  restore_par_options = TRUE,
  colours = NA
)

Arguments

risk_contributions
  The risk contributions.
sub_groups
  The vector with the sub-groups.
title
  The title of the plot.
y_max
  Fix the axis of the risk of the outcome.
restore_par_options
  Restore par options.
colours
  Colours indicating each sub-group.

Value

A plot with prevalence and mean risks by sub-groups.

References

Mean risk contributions by sub-groups

Description
Table with the mean risk contributions by sub-groups.

Usage
CoOL_8_mean_risk_contributions_by_sub_group(
  risk_contributions,
  sub_groups,
  exposure_data,
  outcome_data,
  model,
  exclude_below = 0.001,
  restore_par_options = TRUE,
  colours = NA
)

Arguments
risk_contributions  The risk contributions.
sub_groups  The vector with the sub-groups.
exposure_data  The exposure data.
outcome_data  The outcome data.
model  The trained non-negative model.
exclude_below  A lower cut-off for which risk contributions shown.
restore_par_options  Restore par options.
colours  Colours indicating each sub-group.

References

Examples
#See the example under CoOL_0_working_example
The default analysis for computational phase of CoOL

Description

The analysis and plots presented in the main paper. We recommend using View(CoOL_default) and View() on the many sub-functions to understand the steps and modify to your own research question. 3 sets of training will run with a learning rate of 1e-4 and a patience of 200 epochs, a learning rate of 1e-5 and a patience of 100 epochs, and a learning rate of 1e-6 and a patience of 50 epochs.

Usage

CoOL_default(
  data,
  sub_groups = 3,
  exclude_below = 0.01,
  input_parameter_reg = 0.001,
  hidden = 10,
  monitor = TRUE,
  epochs = 10000
)

Arguments

- **data**: A data.frame(cbind(outcome_data,exposure_data)).
- **sub_groups**: Define the number of expected sub-groups.
- **exclude_below**: Risk contributions below this value are not shown in the table.
- **input_parameter_reg**: The regularization of the input parameters.
- **hidden**: The number of synergy-functions.
- **monitor**: Whether monitoring plots will be shown in R.
- **epochs**: The maximum number of epochs.

Value

A series of plots across the full Causes of Outcome Learning approach.

References

cpp_train_network_relu

Examples

```r
# Not run
while (FALSE) {
  #See the example under CoOL_0_working_example for a more detailed tutorial
  library(CoOL)
  data <- CoOL_0_working_example(n=10000)
  CoOL_default(data)
}
```

-----------------------------------------------

cpp_train_network_relu

*Function used as part of other functions*

Description

Non-negative neural network

Usage

```r
cpp_train_network_relu(
  x, y, testx, testy, W1_input, B1_input, W2_input, B2_input, lr = 0.01,
  maxepochs = 100, input_parameter_reg = 1e-06, drop_out = 0L
)
```

Arguments

- **x**: A matrix of predictors for the training dataset of shape (nsamples, nfeatures)
- **y**: A vector of output values for the training data with a length similar to the number of rows of x
- **testx**: A matrix of predictors for the test dataset of shape (nsamples, nfeatures)
- **testy**: A vector of output values for the test data with a length similar to the number of rows of x
- **W1_input**: Input-hidden layer weights of shape (nfeatures, hidden)
- **B1_input**: Biases for the hidden layer of shape (1, hidden)
- **W2_input**: Hidden-output layer weights of shape (hidden, 1)
B2_input Bias for the output layer (the baseline risk) of shape (1, 1)
lr Initial learning rate
maxepochs The maximum number of epochs
input_parameter_reg Regularisation decreasing parameter value at each iteration for the input parameters
drop_out To drop connections if their weights reach zero.

Value
A list of class "SCL" giving the estimated matrices and performance indicators

Author(s)
Andreas Rieckmann, Piotr Dworzynski, Leila Arras, Claus Ekstrøm

random Function used as part of other functions

Description
Function used as part of other functions

Usage
random(r, c)

Arguments
r rows in matrix
c columns in matrix

rcpprelu Function used as part of other functions

Description
relu-function

Usage
rcpprelu(x)

Arguments
x input in the relu function
**rcpprelu_neg**

*Function used as part of other functions*

**Description**

Negative ReLU function

**Usage**

`rcpprelu_neg(x)`

**Arguments**

- `x`  input in the negative ReLU function

**relu**

*Function used as part of other functions*

**Description**

Function used as part of other functions

**Usage**

`relu(input)`

**Arguments**

- `input`  input in the ReLU function
Index

CoOL_0_binary_encode_exposure_data, 2
CoOL_0_common_simulation, 3
CoOL_0_complex_simulation, 4
CoOL_0_con founding_simulation, 4
CoOL_0_mediation_simulation, 5
CoOL_0_working_example, 5
CoOL_1_initiate_neural_network, 7
CoOL_2_train_neural_network, 8
CoOL_3_plot_neural_network, 9
CoOL_4_AUC, 10
CoOL_4_predict_risks, 11
CoOL_5_layerwise_relevance_propagation, 12
CoOL_6_dendrogram, 13
CoOL_6_individual_effects_matrix, 14
CoOL_6_sub_groups, 14
CoOL_6_sum_of_individual_effects, 15
CoOL_7_prevalence_and_mean_risk_plot, 16
CoOL_8_mean_risk_contributions_by_sub_group, 17
CoOL_default, 18
cpp_train_network_relu, 19
random, 20
rcpprelu, 20
rcpprelu_neg, 21
relu, 21