

# Package ‘TOP’

January 30, 2026

**Title** TOP Constructs Transferable Model Across Gene Expression Platforms

**Version** 1.10.0

**Date** 2022-11-09

**Description** TOP constructs a transferable model across gene expression platforms for prospective experiments. Such a transferable model can be trained to make predictions on independent validation data with an accuracy that is similar to a re-substituted model. The TOP procedure also has the flexibility to be adapted to suit the most common clinical response variables, including linear response, binomial and Cox PH models.

**License** GPL-3

**URL** <https://github.com/Harry25R/TOP>

**BugReports** <https://github.com/Harry25R/TOP/issues>

**biocViews** Software, Survival, GeneExpression

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Imports** assertthat, caret, ClassifyR, directPA, doParallel, dplyr, ggnewscale, ggplot2, ggraph, ggrepel, ggthemes, glmnet, Hmisc, igraph, latex2exp, limma, magrittr, methods, plotly, pROC, purrr, reshape2, stats, stringr, survival, tibble, tidygraph, tidyr, statmod

**Suggests** knitr, rmarkdown, BiocStyle, Biobase, curatedOvarianData, ggbeeswarm, ggsci, survminer, tidyverse

**VignetteBuilder** knitr

**LazyData** false

**git\_url** <https://git.bioconductor.org/packages/TOP>

**git\_branch** RELEASE\_3\_22

**git\_last\_commit** 37ed69c

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2026-01-29

**Author** Harry Robertson [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-9203-3894>>),  
Nicholas Robertson [aut]

**Maintainer** Harry Robertson <[harry.robertson@sydney.edu.au](mailto:harry.robertson@sydney.edu.au)>

Contents

|                                  |           |
|----------------------------------|-----------|
| coefNetworkPlot . . . . .        | 2         |
| expit . . . . .                  | 3         |
| filterFeatures . . . . .         | 3         |
| pairwise_col_diff . . . . .      | 4         |
| performance_TOP . . . . .        | 5         |
| predict_TOP . . . . .            | 6         |
| ROC_Plot . . . . .               | 7         |
| simplenetworkPlot . . . . .      | 7         |
| Surv_TOP_CI . . . . .            | 8         |
| TOP_coefPlot . . . . .           | 9         |
| TOP_data_binary . . . . .        | 10        |
| TOP_lambdaPlot . . . . .         | 10        |
| TOP_model . . . . .              | 11        |
| TOP_survival . . . . .           | 12        |
| TOP_survivalPrediction . . . . . | 14        |
| <b>Index</b>                     | <b>15</b> |

---

|                 |                        |
|-----------------|------------------------|
| coefNetworkPlot | <i>coefNetworkPlot</i> |
|-----------------|------------------------|

---

Description

coefNetworkPlot

Usage

```
coefNetworkPlot(TOP_model, nFeatures = 20, s = "lambda.min")
```

Arguments

|           |                                                                                |
|-----------|--------------------------------------------------------------------------------|
| TOP_model | A Transferable Omics Prediction model. The output from the TOP_model function. |
| nFeatures | The number of features that will be plotted. Default: 20                       |
| s         | Lambda value for the lasso model. Default is "lambda.min"                      |

Value

A coefNetwork plot

Examples

```
data(TOP_data_binary, package = "TOP")

x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3
```

```

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
coefNetworkPlot(model)
#' @import ggplot2

```

expit

*The expit function***Description**

The expit function

**Usage**

```
expit(x)
```

**Arguments**

x                      numeric

**Value**

The expit of x

**Examples**

```
curve(expit, from = -5, to = 5)
```

filterFeatures

*filterFeatures***Description**

A function that implements feature selection, using limma, from a list of data frames with corresponding labels.

**Usage**

```

filterFeatures(
  x_list,
  y_list,
  contrast = NULL,
  nFeatures = 50,
  combinationMethod = "OSP"
)

```

**Arguments**

|                                |                                                                                                                             |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>x_list</code>            | A list of data frames, with columns corresponding to features and rows corresponding to observations.                       |
| <code>y_list</code>            | A list of factor labels.                                                                                                    |
| <code>contrast</code>          | A character vector describing which order of levels to contrast in <code>y_list</code> ("disease - control"), Default: NULL |
| <code>nFeatures</code>         | Number of features to return, Default: 50                                                                                   |
| <code>combinationMethod</code> | Which p-value combination method to use, Default: 'OSP' Options are 'Stouffer', 'OSP', 'Fisher', 'maxP'.                    |

**Details**

`contrast` must be a character vector of length 1. If `contrast` is NULL, the first level of the first factor in `y_list` will be used as the reference level.

**Value**

A vector of feature names.

**Examples**

```
data(TOP_data_binary, package = "TOP")
x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3

x_list <- list(x1, x2, x3)
y_list <- list(TOP_data_binary$y1, TOP_data_binary$y2, TOP_data_binary$y3)
y_list <- y_list <- lapply(y_list, function(x) {
  x <- factor(x, levels = c("1", "0"), labels = c("Yes", "No"))
})

filterFeatures(
  x_list, y_list,
  contrast = "Yes - No", nFeatures = 10, combinationMethod = "OSP"
)
```

---

|                                |                                                           |
|--------------------------------|-----------------------------------------------------------|
| <code>pairwise_col_diff</code> | <i>Compute pairwise difference between matrix columns</i> |
|--------------------------------|-----------------------------------------------------------|

---

**Description**

Compute pairwise difference between matrix columns

**Usage**

```
pairwise_col_diff(x)
```

**Arguments**

`x` A data matrix of size  $n$  times  $p$ . Where rows are observations and columns are features.

**Value**

A matrix of size  $n$  times  $(p \text{ choose } 2)$ , where each column is the difference between two of the original columns.

**Examples**

```
n <- 1
p <- 4
x <- matrix(rep(seq_len(p), n), nrow = n, ncol = p, byrow = TRUE)
colnames(x) <- paste0("X", seq_len(p))
pairwise_col_diff(x)
```

---

performance\_TOP

*performance\_TOP*


---

**Description**

A function to calculate the external performance of the Transferable Omics Prediction model.

**Usage**

```
performance_TOP(TOP_model, newx, newy, covariates = NULL, s = "lambda.min")
```

**Arguments**

`TOP_model` This is the output of the function `TOP_model`.

`newx` A matrix of the new data to be predicted. With the same number of feature columns as the original data.

`newy` A vector of the true labels that are being predicted. With the same number of samples as `newx`.

`covariates` A data.frame of the same covariates as the original TOP model, Default: NULL

`s` Lambda used in the lasso model, Default: 'lambda.min'

**Value**

A confusion matrix that displays the performance of the classifier.

**Examples**

```
data(TOP_data_binary, package = "TOP")
x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2

x_list <- list(x1,x2)
y_list <- list(TOP_data_binary$y1, TOP_data_binary$y2)
```

```

model <- TOP_model(x_list, y_list)

x3 <- TOP_data_binary$x3
y3 <- TOP_data_binary$y3

performance_TOP(model$models, newx = x3, newy = y3)

```

---

predict\_TOP

*Predict using the Trasferable Omics Prediction model.*


---

## Description

A prediction function for the Trasferable Omics Prediction model.

## Usage

```
predict_TOP(TOP_model, newx, covariates = NULL, s = "lambda.min")
```

## Arguments

|            |                                                                                                          |
|------------|----------------------------------------------------------------------------------------------------------|
| TOP_model  | The output from the TOP_model function.                                                                  |
| newx       | A matrix of the new data to be predicted. The columns should be features and the rows should be samples. |
| covariates | A data frame of the same covariates that were used in the TOP model, Default: NULL                       |
| s          | Lambda value for the lasso model, Default: 'lambda.min'                                                  |

## Value

A vector of predictions for the new data.

## Examples

```

data(TOP_data_binary, package = "TOP")

x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
predictions <- predict_TOP(model$models, newx = x3)

```

ROC\_Plot

*ROC\_Plot***Description**

A function visualizes the performance of a classifier by plotting the Receiver Operating Characteristic (ROC) curve.

**Usage**

```
ROC_Plot(roc_list)
```

**Arguments**

`roc_list`            A list of roc objects from the pROC package

**Value**

A ROC Plot

**Examples**

```
data(TOP_data_binary, package = "TOP")
x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
pred <- predict_TOP(model$models, newx = x3)
roc <- pROC::roc(y3, pred)
ROC_Plot(list(roc))
```

simplenetworkPlot

*simplenetworkPlot***Description**

simplenetworkPlot

**Usage**

```
simplenetworkPlot(TOP_model, nFeatures = 50, s = "lambda.min")
```

Arguments

|           |                                                                                |
|-----------|--------------------------------------------------------------------------------|
| TOP_model | A Transferable Omics Prediction model. The output from the TOP_model function. |
| nFeatures | The number of features that will be plotted. Default: 20                       |
| s         | Lambda value for the lasso model. Default is "lambda.min"                      |

Value

A simple network plot

Examples

```
data(TOP_data_binary, package = "TOP")

x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
simplenetworkPlot(model)
```

---

|             |                                                       |
|-------------|-------------------------------------------------------|
| Surv_TOP_CI | Create a function to calculate the concordance index. |
|-------------|-------------------------------------------------------|

---

Description

FUNCTION\_DESCRIPTION

Usage

```
Surv_TOP_CI(TOP_survival, newx, newy)
```

Arguments

|              |                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------|
| TOP_survival | A TOP_survival model. See <a href="#">TOP_survival</a> .                                                        |
| newx         | A new data.frame to predict the survival time.                                                                  |
| newy         | A data.frame, where the first columns in each data frame is the time and the second column is the event status. |

Value

An object of class concordance

**Examples**

```
data(TOP_data_binary, package = "TOP")
time <- rpois(300, c(600, 1000))
surv <- sample(c(0, 1), 300, replace = TRUE)
y <- data.frame(time, surv)

batch <- rep(paste0("y", 1:3), c(100, 100, 100))
y_list <- y |> split(batch)

x_list <- list(TOP_data_binary$x1, TOP_data_binary$x2, TOP_data_binary$x3)

surv_model <- TOP_survival(x_list[-3], y_list[-3], nFeatures = 10)
Surv_TOP_CI(surv_model, newx = x_list[[3]], newy = y_list[[3]])
```

TOP\_coefPlot

*TOP\_coefPlot***Description**

TOP\_coefPlot

**Usage**

```
TOP_coefPlot(TOP_model, nFeatures = 20, s = "lambda.min")
```

**Arguments**

|           |                                                                                |
|-----------|--------------------------------------------------------------------------------|
| TOP_model | A Transferable Omics Prediction model. The output from the TOP_model function. |
| nFeatures | The number of features that will be plotted. Default: 20                       |
| s         | Lambda value for the lasso model, Default: 'lambda.min'                        |

**Value**

A TOP coeff plot

**Examples**

```
data(TOP_data_binary, package = "TOP")

x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
TOP_coefPlot(model)
```

---

|                 |                                |
|-----------------|--------------------------------|
| TOP_data_binary | <i>A simulated binary data</i> |
|-----------------|--------------------------------|

---

### Description

A simulated binary data

### Usage

```
data("TOP_data_binary")
```

### Format

A list with columns:

**x1** A matrix of size 100x20, each column has mean 1 and sd 1

**x2** A matrix of size 100x20, each column has mean 2 and sd 1

**x3** A matrix of size 100x20, each column has mean 3 and sd 1

**y1** A factor vector of 0's and 1's, created by beta and x1

**y2** A factor vector of 0's and 1's, created by beta and x2

**y3** A factor vector of 0's and 1's, created by beta and x3

**beta** A vector with first 10 entries drawn from random unif(-1, 1), otherwise 0's.

### Value

The example data.

---

|                |                       |
|----------------|-----------------------|
| TOP_lambdaPlot | <i>TOP_lambdaPlot</i> |
|----------------|-----------------------|

---

### Description

TOP\_lambdaPlot

### Usage

```
TOP_lambdaPlot(  
  TOP_model,  
  nFeatures = 20,  
  s = "lambda.min",  
  interactive = FALSE,  
  label = FALSE  
)
```

**Arguments**

|             |                                                                                              |
|-------------|----------------------------------------------------------------------------------------------|
| TOP_model   | A Transferable Omics Prediction model. The output from the TOP_model function.               |
| nFeatures   | The number of features to plot, features are ranked beta's for lambda.min. Default: 20       |
| s           | Lambda value for the lasso model. Default is "lambda.min"                                    |
| interactive | A boolean indicating whether the plot should be interactive. Defaults to FALSE .             |
| label       | A boolean indicating whether the features should be labeled on the plot. Defaults to FALSE . |

**Value**

A TOP lambda plot

**Examples**

```
data(TOP_data_binary, package = "TOP")

x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
TOP_lambdaPlot(model)
```

---

TOP\_model

TOP\_model

---

**Description**

The main function of the TOP package. This function returns a glmnet model .

**Usage**

```
TOP_model(
  x_list,
  y_list,
  covariates = NULL,
  dataset_weights = NULL,
  sample_weights = FALSE,
  optimiseExponent = FALSE,
  nCores = 1
)
```

**Arguments**

|                               |                                                                                                                                                                |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x_list</code>           | a list of data frames, each containing the data for a single batch or dataset. Columns should be features and rows should be observations.                     |
| <code>y_list</code>           | a list of factors, each containing the labels for a single batch or dataset. The length of this list should be the same as the length of <code>x_list</code> . |
| <code>covariates</code>       | a list of data frames with the covariates that should be included in the model, Default: NULL                                                                  |
| <code>dataset_weights</code>  | a list of data frames that refer to any grouping structure in the batches, Default: NULL                                                                       |
| <code>sample_weights</code>   | Should each batch be weighted equally? This is important in unequal sample sizes, Default: FALSE                                                               |
| <code>optimiseExponent</code> | Should the exponent used to modify the lasso weights be optimised using re-substitution?, Default: FALSE                                                       |
| <code>nCores</code>           | A numeric specifying the number of cores used if the user wants to use parallelisation, Default: 1                                                             |

**Value**

Returns a list with the following elements: `models`, which is a `glmnet` object and `features`, which is a list of the features used in each model.

**Examples**

```
data(TOP_data_binary, package = "TOP")

x1 <- TOP_data_binary$x1
x2 <- TOP_data_binary$x2
x3 <- TOP_data_binary$x3
y1 <- TOP_data_binary$y1
y2 <- TOP_data_binary$y2
y3 <- TOP_data_binary$y3

set.seed(23)
x_list <- list(x1, x2)
y_list <- list(factor(y1), factor(y2))

model <- TOP_model(x_list, y_list)
```

---

TOP\_survival

TOP\_survival

---

**Description**

FUNCTION\_DESCRIPTION

**Usage**

```
TOP_survival(
  x_list,
  y_list,
  nFeatures = 50,
  dataset_weights = NULL,
  sample_weights = FALSE,
  nCores = 1
)
```

**Arguments**

|                              |                                                                                                                                                                                                            |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x_list</code>          | A list of data frames, each containing the data for a single batch or dataset. Columns are features and rows are observations.                                                                             |
| <code>y_list</code>          | A list of data frames, where the first columns in each data frame is the time and the second column is the event status. The length of this list should be the same as the length of <code>x_list</code> . |
| <code>nFeatures</code>       | Number of features to return, Default: 50                                                                                                                                                                  |
| <code>dataset_weights</code> | a list of data frames that refer to any grouping structure in the batches, Default: NULL                                                                                                                   |
| <code>sample_weights</code>  | Should each batch be weighted equally? This is important in unequal sample sizes, Default: FALSE                                                                                                           |
| <code>nCores</code>          | A numeric specifying the number of cores used if the user wants to use parallelisation, Default: 1                                                                                                         |

**Details**

DETAILS

**Value**

A cox net model

**Examples**

```
data(TOP_data_binary, package = "TOP")
time <- rpois(300, c(600, 1000))
surv <- sample(c(0, 1), 300, replace = TRUE)
y <- data.frame(time, surv)

batch <- rep(paste0("y", 1:3), c(100, 100, 100))
y_list <- y |> split(batch)

x_list <- list(TOP_data_binary$x1, TOP_data_binary$x2, TOP_data_binary$x3)

TOP_survival(x_list[-3], y_list[-3], nFeatures = 10)
```

TOP\_survivalPrediction

*TOP\_survivalPrediction*

---

### Description

A prediction function for TOP\_survival

### Usage

```
TOP_survivalPrediction(TOP_survival, newx)
```

### Arguments

TOP\_survival     A TOP\_survival model. See [TOP\\_survival](#).  
newx             A new dataset to predict the survival time.

### Value

A vector of predicted survival time.

### Examples

```
data(TOP_data_binary, package = "TOP")
time <- rpois(300, c(600, 1000))
surv <- sample(c(0, 1), 300, replace = TRUE)
y <- data.frame(time, surv)

batch <- rep(paste0("y", 1:3), c(100, 100, 100))
y_list <- y |> split(batch)

x_list <- list(TOP_data_binary$x1, TOP_data_binary$x2, TOP_data_binary$x3)

surv_model <- TOP_survival(x_list[-3], y_list[-3], nFeatures = 10)
TOP_survivalPrediction(surv_model, newx = x_list[[3]])
```

# Index

## \* **datasets**

TOP\_data\_binary, [10](#)

coefNetworkPlot, [2](#)

expit, [3](#)

filterFeatures, [3](#)

pairwise\_col\_diff, [4](#)

performance\_TOP, [5](#)

predict\_TOP, [6](#)

ROC\_Plot, [7](#)

simplenetworkPlot, [7](#)

Surv\_TOP\_CI, [8](#)

TOP\_coefPlot, [9](#)

TOP\_data\_binary, [10](#)

TOP\_lambdaPlot, [10](#)

TOP\_model, [11](#)

TOP\_survival, [8](#), [12](#), [14](#)

TOP\_survivalPrediction, [14](#)