# Package: scConform (via r-universe)

October 30, 2026

**Type** Package

**Title** Conformal Inference for Cell Type Annotation

**Version** 1.1.1

**Description** Builds prediction interval for cell type annotation using conformal inference and conformal risk control. It provides two main methods. The first one gives prediction intervals with coverage guarantees based on standard conformal inference. The second one instead gives hierarchical prediction intervals that are consistent with the cell ontology.

**License** Artistic-2.0

**Depends** R (>= 4.5.0)

**Suggests** knitr, Matrix, rmarkdown, BiocStyle, VGAM, ontoProc, MerfishData, doParallel, scuttle, SingleCellExperiment, scran, testthat (>= 3.0.0)

**Imports** igraph, stats, SummarizedExperiment, BiocParallel, Rgraphviz

**VignetteBuilder** knitr

**Roxygen** list(markdown = TRUE)

**Encoding** UTF-8

**biocViews** Software, Classification, SingleCell, Annotation

**URL** https://github.com/ccb-hms/scConform

**BugReports** https://github.com/ccb-hms/scConform/issues

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** libglpk-dev libxml2-dev zlib1g-dev

**Repository** https://bioc.r-universe.dev

**Date/Publication** 2026-07-02 08:47:28 UTC

**RemoteUrl** https://github.com/bioc/scConform

**RemoteRef** HEAD

**RemoteSha** b7b5a77ea3d16724726491390f8722c173c51b39

# Contents

getCommonAncestor . . . . . 2
getPredictionSets . . . . . 3
plotResult . . . . . 6
precomputeOnto . . . . . 8

**Index** **9**

---

`getCommonAncestor` *Return the common ancestor of the labels in the prediction set*

---

## Description

Given a prediction set and an ontology represented as a directed graph, this function returns the most specific common ancestor of the labels in the prediction set. It is mainly intended for hierarchical conformal prediction, where a set of predicted labels can be summarized by a single ontology term representing their common ancestor.

## Usage

```
getCommonAncestor(pred_set, onto, onto_cache = NULL)
```

## Arguments

| | |
|---|---|
| `pred_set` | character vector of labels included in the prediction set. These labels should correspond to node names in `onto`. |
| `onto` | an `igraph` object representing the ontology. |
| `onto_cache` | optional precomputed cache from `precomputeOnto()`. If `NULL`, the cache is computed automatically (and discarded after the call). |

## Value

A character string corresponding to the most specific common ancestor of the labels in `pred_set` according to the ontology `onto`.

## Examples

```
library(igraph)
# Let's build a random ontology
onto <- graph_from_literal(
    animal -+dog:cat, cat -+british:persian,
    dog -+cocker:retriever, retriever -+golden:labrador
)
# Let's consider this prediction set
pred_set <- c("golden", "labrador", "cocker")
com_anc <- getCommonAncestor(pred_set, onto)
```

getPredictionSets *Get conformal prediction sets*

## Description

This function returns conformal prediction sets for the cell type of cells in a query dataset. It implements two methods: standard split conformal inference and a hierarchical conformal risk-control approach that incorporates the cell ontology structure. Depending on the input and on the value of `return_sc`, the output is either a list of prediction sets or a `SingleCellExperiment`/`SpatialExperiment` object with prediction sets stored in the `colData`.

## Usage

```
getPredictionSets(
  x_query,
  x_cal,
  y_cal,
  onto = NULL,
  onto_cache = NULL,
  alpha = 0.1,
  lambdas = seq(0.001, 0.999, length.out = 100),
  follow_ontology = TRUE,
  resample = FALSE,
  labels = NULL,
  return_sc = NULL,
  pr_name = "pred.set",
  simplify = FALSE,
  method = "full",
  BPPARAM = SerialParam()
)
```

## Arguments

| | |
|---|---|
| `x_query` | query data for which we want to build prediction sets. This can be either a `SingleCellExperiment` (or `SpatialExperiment`) object with the estimated probabilities for each cell type in the `colData`, or a named numeric matrix with `n` rows and `K` columns, where `n` is the number of cells and `K` is the number of different labels. The colnames of the matrix have to correspond to the cell labels. |
| `x_cal` | calibration data. This can be either a `SingleCellExperiment` object with the estimated probabilities for each cell type in the `colData`, or a named matrix of dimension `m x K`, where `m` is the number of cells and `K` is the number of different labels. The colnames of the matrix have to correspond to the cell labels. |
| `y_cal` | a vector of length `m` with the true labels of the cells in the calibration data. |
| `onto` | An `igraph` object representing the considered section of the cell ontology. |
| `onto_cache` | Optional precomputed ontology cache from `precomputeOnto(onto)`. If `NULL`, the cache is computed automatically (and discarded after the call). |

`alpha` Numeric value between 0 and 1 that indicates the allowed miscoverage

`lambdas` a numeric vector of possible lambda values to be considered. Necessary only when `follow_ontology=TRUE`.

`follow_ontology`
Logical. If `TRUE`, then the function returns hierarchical prediction sets that follow the cell ontology structure. If `FALSE`, it returns classical conformal prediction sets. See Details.

`resample` Logical. If `TRUE`, the calibration dataset is resampled according to the estimated relative frequencies of cell types in the query data.

`labels` Character vector of labels of different considered cell types. Necessary if `onto=NULL`, otherwise they are set equal to the leaf nodes of the provided graph.

`return_sc` Logical. Parameter the controls the output type. If `TRUE`, the function returns a `SingleCellExperiment`. If `FALSE`, the function returns a list. By default, it is set to `TRUE` when `x_query` is a `SingleCellExperiment` (or `SpatialExperiment`) object and to `FALSE` when `x_query` is a matrix.

`pr_name` Character string giving the name of the `colData` variable in the returned `SingleCellExperiment` object that will contain the prediction sets. The default name is `pred.set`.

`simplify` Logical. If `TRUE`, the output will be the common ancestor of the labels inserted into the prediction set. If `FALSE` (default), the output will be the set of the leaf labels.

`method` character string or function specifying how hierarchical prediction sets are constructed when `follow_ontology=TRUE`. If a character string, it must be one of:

- `"full"` the default hierarchical construction described in the Details section, which guarantees non-empty prediction sets;
- `"step"` a construction that includes all ancestors up to a fixed number of steps above the predicted class;
- `"rank"` a construction that ranks leaf nodes by predicted probability, accumulates probability until a threshold is reached, and then expands the lowest common ancestor of the selected leaves.

Alternatively, `method` can be a user-defined function with signature `function(lambda, pred, onto)`, returning a character vector of leaf labels defining the prediction set.

`BPPARAM` BiocParallel instance for parallel computing. Default is `SerialParam()`.

## Details

**Split conformal sets:** Conformal inference is a statistical framework that allows to build prediction sets for any probabilistic or machine learning model. Suppose we have a classification task with $K$ classes. We fit a classification model $\hat{f}$ that outputs estimated probabilities for each class: $\hat{f}(x) \in [0,1]^K$. Split conformal inference requires to reserve a portion of the labelled training data, $(X_1, Y_1), \ldots, (X_n, Y_n)$, to be used as calibration data. Given $\hat{f}$ and the calibration data, the objective of conformal inference is to build, for a new observation $X_{n+1}$, a prediction set $C(X_{n+1}) \subseteq \{1, \ldots, K\}$ that satisfies

$$P\left(Y_{n+1} \in C(X_{n+1})\right) \geq 1 - \alpha$$

for a user-chosen error rate $\alpha$. Note that conformal inference is distribution-free and the sets provided have finite-samples validity. The only assumption is that the test data and the calibration data are exchangeable. The algorithm of split conformal inference is the following:

1. For the data in the calibration set, $(X_1, Y_1), \ldots, (X_n, Y_n)$ , obtain the *conformal scores*, $s_i = 1 - \hat{f}(X_i)_{Y_i},\ i = 1, \ldots, n$. These scores will be high when the model is assigning a small probability to the true class, and low otherwise.
2. Obtain $\hat{q}$, the $\lceil (1-\alpha)(n+1) \rceil / n$ empirical quantile of the conformal scores.
3. Finally, for a new observation $X_{n+1}$, construct a prediction set by including all the classes for which the estimated probability is higher than $1 - \hat{q}$:

$$C(X_{n+1}) = \{y : \hat{f}(X_{n+1})_y \geq 1 - \hat{q}\}.$$

**Hierarchical conformal sets:** Let $\hat{y}(x)$ be the class with maximum estimated probability. Moreover, given a directed graph let $\mathcal{P}(v)$ and $\mathcal{A}(v)$ be the set on children nodes and ancestor nodes of $v$, respectively. Finally, for each node $v$ define a score $g(v, x)$ as the sum of the predicted probabilities of the leaf nodes that are children of $v$. To build the sets we propose the following algorithm:

$$\mathcal{P}(v) \cup \{\mathcal{P}(a) : a \in \mathcal{A}(\hat{y}(x)) : g(a, x) \leq \lambda\},$$

where $v : v \in \mathcal{A}(\hat{y}(x)),\ g(v, x) \geq \lambda,\ v = \arg\min_{u: g(u,x) \geq \lambda} g(u, x)$. In words, we start from the predicted class and we go up in the graph until we find an ancestor of $\hat{y}(x)$ that has a score that is at least $\lambda$ and include in the prediction sets all its children. For theoretical reasons, to this subgraph we add all the other ones that contain $\hat{y}(x)$ for which the score is less than $\lambda$. To choose $\lambda$, we follow eq. (4) in Angelopoulos et al. (2023), considering the miscoverage as loss function. In this way, it is still guaranteed that

$$P(Y_{n+1} \notin C_\lambda(X_{n+1})) \leq \alpha.$$

The construction described above corresponds to the default choice `method = "full"`. Other values of `method` implement alternative nested prediction-set constructions that incorporate the ontology structure in different ways. All methods are calibrated using the same conformal risk-control procedure to select the threshold parameter $\lambda$.

## Value

**If** `return_sc = TRUE`**:** A `SingleCellExperiment` or `SpatialExperiment` object with the prediction sets stored in the `colData`. The name of the corresponding variable is given by `pr_name`.

**If** `return_sc = FALSE`**:** A list of length equal to the number of cells in the query data. Each element contains the prediction set for one cell.

## References

Corbetta, D. et al. *Conformal inference for cell type annotation with graph-structured constraints*. arXiv preprint arXiv:2410.23786.

Angelopoulos, A. N. and Bates, S. *A gentle introduction to conformal prediction and distribution-free uncertainty quantification*. arXiv preprint arXiv:2107.07511.

Angelopoulos, A. N. et al. *Conformal risk control*. arXiv preprint arXiv:2208.02814.

## Examples

```
# random p matrix
set.seed(1040)
p <- matrix(rnorm(2000 * 4), ncol = 4)
# Normalize the matrix p to have all numbers between 0 and 1 that sum to 1
# by row
p <- exp(p - apply(p, 1, max))
p <- p / rowSums(p)
cell_types <- c("T (CD4+)", "T (CD8+)", "B", "NK")
colnames(p) <- cell_types

# Take 1000 rows as calibration and 1000 as test
p_cal <- p[1:1000, ]
p_test <- p[1001:2000, ]

# Randomly create the vector of real cell types for p_cal and p_test
y_cal <- sample(cell_types, 1000, replace = TRUE)
y_test <- sample(cell_types, 1000, replace = TRUE)

# Obtain conformal prediction sets
conf_sets <- getPredictionSets(
    x_query = p_test,
    x_cal = p_cal,
    y_cal = y_cal,
    onto = NULL,
    alpha = 0.1,
    follow_ontology = FALSE,
    resample = FALSE,
    labels = cell_types,
    return_sc = FALSE
)
```

---

# plotResult *Plot prediction sets*

---

## Description

This function takes as input a prediction set and an ontology and plots the ontology, highlighting the labels included in the set.

## Usage

```
plotResult(
  pred_set,
  onto,
  onto_cache = NULL,
  probs = NULL,
  col_grad = c("lemonchiffon", "orange", "darkred"),
```

```
  attrs = NULL,
  k = 4,
  title = NULL,
  add_scores = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| pred_set | character vector containing the labels in the prediction set |
| onto | an igraph object representing the ontology |
| onto_cache | optional precomputed cache from precomputeOnto(). If NULL, the cache is computed automatically (and discarded after the call). |
| probs | numeric vector of estimated probabilities for the classes. The names of probs should correspond to node names in onto |
| col_grad | character vector of colors used to highlight the classes. If probs is provided, this should contain a color gradient; otherwise, a single color can be supplied |
| attrs | attrs list of additional graphical attributes passed to plot() |
| k | integer number of decimal digits to consider in probs |
| title | title of the plot |
| add_scores | Logical. If TRUE, estimated probabilities are added to the name of the classes |
| ... | additional graphical parameters passed to plot() |

## Value

A plot of the ontology with the labels in the prediction set highlighted.

## Examples

```
library(igraph)
# Let's build a random ontology
onto <- graph_from_literal(
    animal -+dog:cat, cat -+british:persian,
    dog -+cocker:retriever, retriever -+golden:labrador
)
# Let's consider this prediction set
pred_set <- c("golden", "labrador", "cocker")
plotResult(pred_set, onto,
    col_grad = "pink", add_scores = FALSE,
    title = "Prediction set"
)
```

precomputeOnto *Precompute ontology lookup structures for fast hierarchical inference*

## Description

Call this once after building your igraph ontology object. The result can be passed to getPrediction-Sets() via the onto_cache argument.

## Usage

```
precomputeOnto(onto)
```

## Arguments

| | |
|---|---|
| onto | An igraph object representing the cell ontology. |

## Value

A named list with precomputed structures:

| | |
|---|---|
| dist_mat | Full node x node distance matrix (rows = from, cols = to) |
| leaf_children | Named list: for each node, character vector of descendant leaf nodes |
| ancestors_of | Named list: for each node, character vector of ancestor nodes (including self) |
| leaves | Character vector of leaf node names |
| root | Character string: name of the root node |
| node_names | Character vector of all node names |

## Examples

```
library(igraph)
# Let's build a random ontology
onto <- graph_from_literal(
    animal -+dog:cat, cat -+british:persian,
    dog -+cocker:retriever, retriever -+golden:labrador
)
# Precompute the ontology cache
onto_cache <- precomputeOnto(onto)
```

# Index

getCommonAncestor, 2
getPredictionSets, 3

plotResult, 6
precomputeOnto, 8