# Package: ptcompare (via r-universe)

November 18, 2026

**Type** Package

**Title** Bayesian Multiscale Two-Sample Comparison

**Version** 0.1.0

**Description** Bayesian comparison of two multivariate distributions using hidden Markov Polya trees with adaptive partitions, following Awaya and Ma (2024) <doi:10.1080/01621459.2022.2105223>. Provides posterior evidence for overall differences, interpretable partition trees, local effect sizes and computational diagnostics. Posterior computation combines sequential Monte Carlo and recursive message passing.

**URL** https://github.com/nawaya040/ptcompare

**BugReports** https://github.com/nawaya040/ptcompare/issues

**License** GPL (>= 2)

**Encoding** UTF-8

**Imports** Rcpp, graphics, grDevices, stats, utils

**LinkingTo** Rcpp

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, mclust

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**RoxygenNote** 7.3.3

**NeedsCompilation** yes

**Author** Naoki Awaya [aut, cre]

**Maintainer** Naoki Awaya <nawaya@waseda.jp>

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

**Date/Publication** 2026-11-17 18:20:02 UTC

**RemoteUrl** https://github.com/cran/ptcompare

**RemoteRef** HEAD

**RemoteSha** 8ff5b15bb89dda1f33db8fedb91d8457f50c2de1

# Contents

ptcompare-package . . . . . 2
compare_pt . . . . . 3
diagnostics . . . . . 5
plot.ptcompare_fit . . . . . 6
pt_control . . . . . 8
pt_prior . . . . . 9
region_rows . . . . . 10
regions . . . . . 11
summary.ptcompare_fit . . . . . 13

**Index** **14**

---

`ptcompare-package` *ptcompare: compare the distributions of two groups*

---

## Description

Compares the distributions of two groups of observations on numeric variables and finds the regions where they differ, using the hidden Markov Pólya tree method of Awaya and Ma (2024).

## Details

A typical analysis:

1. `fit <- compare_pt(x, y)` compares group 1 (`x`) with group 2 (`y`).
2. `fit` or `summary(fit)` shows the posterior probability that the distributions differ.
3. `regions(fit, prob = 0.9)` lists the regions where the groups differ.
4. `plot(fit)` and `plot(fit, type = "regions")` show the results.

Settings are changed with `pt_prior()` (statistical model) and `pt_control()` (computation).

## Author(s)

**Maintainer**: Naoki Awaya <nawaya@waseda.jp>

## References

Awaya, N. and Ma, L. (2024). Hidden Markov Pólya Trees for High-Dimensional Distributions. *Journal of the American Statistical Association*, 119(545), 189–201. doi:10.1080/01621459.2022.2105223

compare_pt *Compare the distributions of two groups*

## Description

Compares the distributions of two groups of observations on one or more numeric variables, using the hidden Markov Pólya tree method of Awaya and Ma (2024). It gives the posterior probability that the two distributions differ, and finds regions of the data space where they differ. Each region is described by conditions on the variables, such as `age <= 40`.

## Usage

```
compare_pt(
  x,
  y = NULL,
  group = NULL,
  support = NULL,
  prior = pt_prior(),
  control = pt_control(),
  transform = if (is.null(support)) "rank" else "range"
)
```

## Arguments

| | |
|---|---|
| `x` | Data for group 1: a numeric matrix or data frame with one row per observation and one column per variable, or a numeric vector for a single variable. When `group` is used, `x` contains the data of both groups. |
| `y` | Data for group 2, with the same columns as `x`. Give either `y` or `group`. |
| `group` | Group labels for the rows of `x`, with exactly two different values. For a factor, the first level is group 1. Otherwise, the label that appears first is group 1. |
| `support` | Optional known range of the variables: a matrix with one row per variable and two columns (lower and upper limit), or a vector `c(lower, upper)` for a single variable. Used with `transform = "range"`. |
| `prior` | Model settings from `pt_prior()`. |
| `control` | Computational settings from `pt_control()`. |
| `transform` | How the variables are put on a common scale before dividing.<br>• `"rank"` (default when `support` is not given): each variable is replaced by the rank of its values in both groups combined. Results do not change if a variable is transformed monotonically (for example by `log()`).<br>• `"range"` (default when `support` is given): each variable is rescaled linearly to the interval from 0 to 1, as in Awaya and Ma (2024). Without `support`, the observed range is extended by 1% on each side.<br>In both cases, thresholds and regions are reported in the original units. |

## Details

The method divides the data space into regions step by step, building a tree of divisions, and compares how the two groups fall on the two sides of each division. Many candidate trees are explored; the one that best fits the data (the representative tree) is used to describe where the groups differ.

Variables that take the same value in all observations are removed with a warning. Missing (`NA`) and infinite values are not allowed; remove those rows first. If there are fewer than `min_node` observations in total (see `pt_prior()`), no division is possible and `prob_difference` is 0.

## Value

A `ptcompare_fit` object. Use summary(), `regions()` and plot() to look at the results. Its main components are:

`prob_difference` Posterior probability that the two distributions differ.

`prob_null` Posterior probability that they are the same (`1 - prob_difference`).

`tree` The representative tree, one row per region. `regions()` gives an easier-to-read table of its divisions.

`diagnostics` Information about the computation; see `diagnostics()`.

`group_names`, `sample_sizes`, `variables` Group names, number of observations per group and names of the variables used.

`prior`, `control`, `transform` The settings used.

`data`, `group` The observations in the original units and their group (0 or 1), in the order they were given; `NULL` when `pt_control(keep_data = FALSE)` was used. `region_rows()` and `region_data()` use them to return the observations of a region.

## References

Awaya, N. and Ma, L. (2024). Hidden Markov Pólya Trees for High-Dimensional Distributions. *Journal of the American Statistical Association*, 119(545), 189–201. doi:10.1080/01621459.2022.2105223

## See Also

`regions()`, `region_data()`, summary(), plot(), `pt_prior()`, `pt_control()`

## Examples

```
set.seed(1)
x <- matrix(rnorm(200), ncol = 2)
y <- matrix(rnorm(200, mean = 0.8), ncol = 2)
fit <- compare_pt(x, y, control = pt_control(particles = 32))
fit
summary(fit)

# Data in one data frame with a group column
two <- droplevels(subset(iris, Species != "setosa"))
fit2 <- compare_pt(two[, 1:4], group = two$Species,
                   control = pt_control(particles = 32))
regions(fit2, prob = 0.9)
```

`diagnostics` *Computational diagnostics*

## Description

Information about the computation of `compare_pt()`, such as the number of effective particles and the running time. `plot(fit, type = "diagnostics")` shows the effective number of particles as a graph.

## Usage

```
diagnostics(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Result of `compare_pt()`. |
| `...` | Not used. |

## Value

A list with components:

`final_ess` Effective number of particles at the end (at most `particles`).

`resamples` Number of resampling steps.

`elapsed_seconds` Running time of `compare_pt()` in seconds.

`trace` Data frame with one row per step of the computation: effective number of particles before resampling (`ess`), number of different trees (`unique_trees`), number of distinct initial particles that the current particles descend from (`root_ancestors`), and whether resampling took place (`resampled`).

`weights`, `particle_null`, `particle_scores` For each final particle: its weight, its probability that the groups are the same, and the fit score used to choose the representative tree.

`log_evidence` Estimated log marginal likelihood of the data.

`depth_limited_leaves`, `depth_limited_fraction` Number of regions of the representative tree that reached `max_depth` with at least `min_node` observations, and the fraction of observations in them.

`numerical_stop_leaves` Number of regions too narrow to divide further.

`unique_proposals`, `peak_particle_payload_bytes` Number of distinct division steps computed, and the approximate memory used by the particles (bytes).

## Examples

```
set.seed(1)
x <- matrix(rnorm(200), ncol = 2)
y <- matrix(rnorm(200, mean = 0.8), ncol = 2)
fit <- compare_pt(x, y, control = pt_control(particles = 32))
d <- diagnostics(fit)
d$final_ess
d$elapsed_seconds
```

---

`plot.ptcompare_fit` *Plot the results of a comparison*

---

## Description

Four kinds of plots:

- `type = "tree"` (default): the representative tree. Each node is a region, labeled with its division (`variable <= threshold`) and `prob_difference`. Larger nodes contain more observations. The color shows the direction of the difference: blue when group 1 is more likely to be in the left part, brown when group 2 is.
- `type = "regions"`: the regions where the two groups differ most, one panel per region (similar to Figure 14 of the supplement of Awaya and Ma, 2024). The regions are those of `regions()` with `prob_difference` at least `prob` and at least `min_n` observations of each group, ordered by `rank_by`; at most `top` are shown. With two or more variables, each panel is a scatter plot: the horizontal axis is the variable that divides the region (dashed line: threshold), the vertical axis another variable that bounds the region. Observations inside the region are shown as blue squares (group 1) and brown triangles (group 2), the others in gray. The box is the region seen in the two variables of the panel only. With three or more variables the region may also be bounded by variables that the panel does not show, so an observation can lie inside the box and still be gray; the symbols, not the box, show which observations are in the region. With two variables the box is the region itself. With one variable, each panel shows the distributions of the two groups inside the region.
- `type = "levels"`: for two variables only. One scatter plot per depth of the tree; each region at that depth is shaded by its `prob_difference` (as in Figure 7 of Soriano and Ma, 2017). Dotted lines show how each region is divided at the next depth.
- `type = "diagnostics"`: the effective number of particles at each step of the computation, with the resampling threshold as a dashed line and the steps where resampling took place marked. The other recorded quantities are in `diagnostics()`.

## Usage

```
## S3 method for class 'ptcompare_fit'
plot(
  x,
  type = c("tree", "regions", "levels", "diagnostics"),
  node = NULL,
```

```
  max_depth = 3L,
  top = 9L,
  prob = 0.5,
  rank_by = c("probability", "effect"),
  min_n = 50L,
  ...
)
```

## Arguments

| | |
|---|---|
| x | Result of `compare_pt()`. |
| type | `"tree"`, `"regions"`, `"levels"` or `"diagnostics"`. |
| node | For `type = "regions"`: `node_ids` of regions to show (see `regions()`), instead of choosing them by `prob`, `min_n` and `rank_by`. A single region is drawn as one large panel. |
| max_depth | For `type = "tree"`: show the tree down to this depth (0 to 6; default 3). A `+` marks nodes whose subregions are not shown. For `type = "levels"`: the deepest level shown (default: all levels, at most 8). |
| top | For `type = "regions"`: maximum number of regions to show. |
| prob | For `type = "regions"`: show only regions with `prob_difference` at least this value. |
| rank_by | For `type = "regions"`: which regions to show first. `"probability"` (default): largest `prob_difference`, the posterior probability of a difference. `"effect"`: largest `effect_abs`, the size of the difference (see `regions()`). |
| min_n | For `type = "regions"`: show only regions with at least this many observations of each group. |
| ... | Graphical parameters such as `main` (for `"tree"`, and for `"regions"` with a single region). |

## Details

`"regions"` and `"levels"` need the data, which are kept unless `pt_control(keep_data = FALSE)` was used.

## Value

The data shown, invisibly.

## References

Soriano, J. and Ma, L. (2017). Probabilistic multi-resolution scanning for two-sample differences. *Journal of the Royal Statistical Society, Series B*, 79(2), 547–572. doi:10.1111/rssb.12180

## Examples

```
set.seed(1)
x <- matrix(rnorm(400), ncol = 2)
y <- matrix(rnorm(400, mean = 0.8), ncol = 2)
fit <- compare_pt(x, y, control = pt_control(particles = 32))
plot(fit)
plot(fit, type = "regions")
plot(fit, type = "regions", top = 4, rank_by = "effect")
plot(fit, type = "regions", node = 2)
plot(fit, type = "levels", max_depth = 3)
plot(fit, type = "diagnostics")
```

---

`pt_control` *Computational settings*

---

## Description

Settings that control how `compare_pt()` carries out its computation. They affect accuracy and running time. The defaults are suitable for most analyses.

## Usage

```
pt_control(
  particles = 256L,
  ess_threshold = 0.5,
  keep_particles = FALSE,
  keep_data = TRUE,
  effect_draws = 4096L,
  resample_power = 0.5
)
```

## Arguments

| | |
|---|---|
| `particles` | Number of candidate trees computed in parallel (the particles of sequential Monte Carlo). More particles give more accurate results but take longer. Try `particles = 1024` or more for a final analysis. |
| `ess_threshold` | Fraction of `particles` (between 0 and 1). When the effective number of particles falls below this fraction, the particles are resampled. |
| `keep_particles` | If `TRUE`, all final candidate trees are kept in the result (component `particles`). This can use a lot of memory. |
| `keep_data` | If `TRUE`, the data are kept in the result. They are needed for `plot(fit, type = "regions")` and `plot(fit, type = "levels")`. |
| `effect_draws` | Number of points (256 to 1048576) used to compute `effect_abs`, `lower`, `upper` and `prob_positive` in `regions()`. Larger values are more accurate and slower. |
| `resample_power` | Number in (0, 1] used when resampling. Particles are drawn with probabilities proportional to their weights raised to this power. The default 0.5 follows Awaya and Ma (2024); 1 gives standard resampling. |

## Value

A `pt_control` object to pass to `compare_pt()`.

## See Also

`pt_prior()` for the statistical model settings.

## Examples

```
pt_control()
pt_control(particles = 1024)
```

---

`pt_prior` *Model settings*

---

## Description

Settings of the statistical model used by `compare_pt()`: how finely the data space may be divided into regions, where the divisions may be placed, and how likely the two groups are to differ before seeing the data.

## Usage

```
pt_prior(
  max_depth = 8L,
  split_grid = 15L,
  min_node = 20L,
  gamma = 0.3,
  beta = 1,
  prune = 0.3,
  precision = 1,
  eta = 0.1,
  midpoint_spike = TRUE
)
```

## Arguments

| | |
|---|---|
| `max_depth` | Maximum number of successive divisions (1 to 20). Larger values allow smaller regions and take longer. |
| `split_grid` | Number of candidate division points for a variable within a region (an odd number from 1 to 127). The candidates are equally spaced and include the middle of the region. |
| `min_node` | Regions with fewer observations than this (both groups together) are not divided further. |
| `gamma` | Prior tendency of the two groups to differ, between 0 and 1. Larger values make differences more likely before seeing the data. |

| | |
|---|---|
| beta | Number from 1 to 100. Values above 1 make differences in smaller (deeper) regions less likely before seeing the data. The default 1 follows Awaya and Ma (2024). |
| prune | Prior probability, between 0 and 1, that a region and all its subregions are the same in the two groups. |
| precision | Positive number. Larger values make the proportion of observations falling on each side of a division closer, before seeing the data, to the proportion of the region's width on that side. |
| eta | Non-negative number. Larger values favor division points closer to the middle of a region, more strongly in regions with many observations. `eta = 0` treats all candidates equally. |
| midpoint_spike | If `TRUE` (default), once a region is divided at its middle, all its subregions are also divided at their middles, as in Awaya and Ma (2024, Appendix A). If `FALSE`, every candidate division point is allowed in every region. |

## Value

A `pt_prior` object to pass to `compare_pt()`.

## See Also

`pt_control()` for the computational settings.

## Examples

```
pt_prior()
pt_prior(max_depth = 6, split_grid = 7)
```

---

# region_rows *Observations in a region*

---

## Description

`region_rows()` says which observations fall in a region of the representative tree, and `region_data()` returns those observations. Use them to look at the subsets that `regions()` reports as the most different.

## Usage

```
region_rows(x, node, ...)

region_data(x, node, ...)
```

## Arguments

| | |
|---|---|
| x | Result of `compare_pt()`. |
| node | `node_id` of one region. |
| ... | Not used. |

## Details

A region is any node of the tree, not only a divided one: node 1 is the whole data space, and the `node_id` column of `regions()` or the numbers in `plot(fit)` identify the others.

## Value

`region_rows()`: a logical vector with one entry per observation, `TRUE` for those in the region. The entries follow the order in which the data were given to `compare_pt()`: the rows of `x` and then the rows of `y`, or the rows of the single data set when `group` was used. `region_data()`: a data frame of the observations in the region, with one column per variable in the original units and a last column holding the group, named `group` unless a variable already has that name. Its row names are the positions in that same order.

Both need the data, which are kept unless `pt_control(keep_data = FALSE)` was used.

## See Also

`regions()` for which regions differ most; `plot(fit, type = "regions")` to draw them.

## Examples

```
set.seed(1)
x <- matrix(rnorm(200), ncol = 2, dimnames = list(NULL, c("a", "b")))
y <- matrix(rnorm(200, mean = 0.8), ncol = 2, dimnames = list(NULL, c("a", "b")))
fit <- compare_pt(x, y, control = pt_control(particles = 32))
top <- regions(fit)$node_id[1]
sum(region_rows(fit, top))
head(region_data(fit, top))
```

---

# regions *Regions where the two groups differ*

---

## Description

Lists the divisions of the representative tree and how much the two groups differ at each one. Use it to find where, and in which direction, the distributions differ.

## Usage

```
regions(x, prob = 0.5, min_n = 0L, rank_by = c("probability", "effect"), ...)

## S3 method for class 'ptcompare_regions'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | Result of compare_pt(), or the table returned by regions() when printing it. |
| prob | Show only rows with prob_difference at least this value (between 0 and 1). Use prob = 0 to show all divisions. |
| min_n | Show only regions with at least this many observations of each group. |
| rank_by | Which divisions to put first. "probability" (default): largest prob_difference, the posterior probability of a difference. "effect": largest effect_abs, the size of the difference. The other quantity breaks ties. The same choice is available in plot(fit, type = "regions"). |
| ... | Not used. |

## Details

Each row is one division of a region into two parts. Inside the region described by condition, the observations are divided into those with variable <= threshold (the left part) and the rest (the right part). The row tells whether the two groups fall into the left part in different proportions.

## Value

A data frame with one row per division, sorted by rank_by (largest first). Columns:

node_id, parent_id, depth Position of the region in the tree (as in plot(fit)); depth 0 is the whole data space.

condition The region, as conditions on the variables.

variable, threshold The division: the left part is variable <= threshold.

n1, n2 Number of observations of group 1 and group 2 in the region.

prob_difference Posterior probability that the two groups fall into the left part in different proportions.

effect Size and direction of the difference: the log odds ratio of falling into the left part, group 1 versus group 2 (posterior mean). Positive values mean group 1 is more likely to be in the left part; negative values mean group 2 is.

effect_abs Posterior mean of the absolute log odds ratio: the size of the difference regardless of direction.

lower, upper 95% credible interval for the log odds ratio. A limit is exactly 0 when the groups may not differ at this division.

prob_positive Posterior probability that group 1 is more likely than group 2 to be in the left part.

## See Also

compare_pt(); plot(fit, type = "regions") draws these regions with the data; region_data() returns the observations of one region.

## Examples

```
set.seed(1)
x <- matrix(rnorm(200), ncol = 2)
y <- matrix(rnorm(200, mean = 0.8), ncol = 2)
fit <- compare_pt(x, y, control = pt_control(particles = 32))
regions(fit)
regions(fit, prob = 0.9, min_n = 20)
```

---

`summary.ptcompare_fit` *Summarize a comparison*

---

## Description

`print(fit)` shows the posterior probability that the two distributions differ. `summary(fit)` also shows the settings, a short report of the computation and the divisions with the largest `prob_difference` (see `regions()` for the meaning of the columns).

## Usage

```
## S3 method for class 'ptcompare_fit'
print(x, ...)

## S3 method for class 'ptcompare_fit'
summary(object, ..., top = 6L)

## S3 method for class 'summary.ptcompare_fit'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x, object` | Result of `compare_pt()` (or of `summary()`). |
| `...` | Not used. |
| `top` | Number of divisions to show. |

## Value

`summary()` returns a `summary.ptcompare_fit` object, which is printed. Its component `regions` holds the divisions shown. `print()` returns its argument invisibly.

## Examples

```
set.seed(1)
x <- matrix(rnorm(200), ncol = 2)
y <- matrix(rnorm(200, mean = 0.8), ncol = 2)
fit <- compare_pt(x, y, control = pt_control(particles = 32))
fit
summary(fit)
summary(fit, top = 3)
```

# Index


compare_pt, 3
compare_pt(), *5*, *7–13*

diagnostics, 5
diagnostics(), *4*, *6*

plot(), *4*
plot.ptcompare_fit, 6
print.ptcompare_fit
        *(*summary.ptcompare_fit*)*, 13
print.ptcompare_regions *(*regions*)*, 11
print.summary.ptcompare_fit
        *(*summary.ptcompare_fit*)*, 13
pt_control, 8
pt_control(), *2–4*, *10*
pt_prior, 9
pt_prior(), *2–4*, *9*
ptcompare *(*ptcompare-package*)*, 2
ptcompare-package, 2

region_data *(*region_rows*)*, 10
region_data(), *4*, *12*
region_rows, 10
region_rows(), *4*
regions, 11
regions(), *4*, *6–8*, *10, 11*, *13*

summary(), *4*
summary.ptcompare_fit, 13