Package 'groupwalk'

May 8, 2026

Title Implement the Group Walk Algorithm

Version 0.1.2

Description A procedure that uses target-decoy competition (or knockoffs) to reject multiple hypotheses in the presence of group structure. The procedure controls the false discovery rate (FDR) at a user-specified threshold.

URL https://www.biorxiv.org/content/10.1101/2022.01.30.478144v1,
https://github.com/freejstone/groupwalk

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Jack Freestone [aut, cre, cph],
Uri Keich [aut, cph]

Maintainer Jack Freestone <jfre0619@uni.sydney.edu.au>

Repository CRAN

Date/Publication 2022-06-18 06:30:02 UTC

Contents

group_walk *Implements group-walk algorithm*

Description

This function returns a list of q-values corresponding to hypotheses that have been partitioned into groups. For FDR control, users should report the target-hypotheses with q-values less than or equal to their choice of threshold, alpha. For further details about how group-walk works, see: https://www.biorxiv.org/content/10.1101/2022.01.30.478144v1

Usage

```
group_walk(
  winning_scores,
  labels,
  all_group_ids,
  K = 40,
  return_frontier = FALSE,
  correction = 1
)
```

Arguments

winning_scores	A numerical vector of winning scores generated from the target-decoy competitions for each hypothesis.
labels	A vector of winning labels indicating whether it was a target (= 1) or a decoy (!= 1) for each hypothesis.
all_group_ids	A vector of group IDs associated to each hypothesis (can be recorded as integers, factors, characters).
K	A window size parameter (integer).
return_frontier	A boolean indicating whether the function should return the complete sequence of frontiers.
correction	A correction factor used to in the numerator of the estimated false discovery rate (FDR) (Use 1 for FDR control).

Value

A sequence of q-values for each hypothesis. If return_frontier = T, additionally the sequence of frontiers will be returned.

Examples

```
create_uncalibrated_hypotheses <- function(m_vec, pi_0_vec, mus, sds) {
  total <- sum(m_vec)
  g_total <- length(m_vec)
```

```
  data <- matrix(0, ncol = 4, nrow = total)
  for (g in 1:length(m_vec)){
    m <- m_vec[g]
    pi_0 <- pi_0_vec[g]
    mu <- mus[g]
    sd <- sds[g]
    if (g == 1) {
      start <- 0
    } else {
      start <- sum(m_vec[1:(g - 1)])
    }
    targets_nonnull <- rnorm(floor(m*pi_0), mean = mu, sd = sd)
    targets_null <- rnorm(m - floor(m*pi_0), mean = 0, sd = 1)
    decoys <- rnorm(m, mean = 0, sd = 1)
    targets <- c(targets_nonnull, targets_null)
    W <- pmax(targets, decoys)
    data[(start + 1):(start + m), 1] <- W
    data[(start + 1):(start + m), 2] <- g
    decoy_inds <- which(decoys > targets)
   inc_native_inds <- (which(targets_null > decoys[(floor(m*pi_0) + 1):m])) + floor(m*pi_0)
    X <- rep(0, m)
    X[decoy_inds] <- -1
    X[inc_native_inds] <- 1
    Y <- X
    X[X == 0] <- 1
    data[(start + 1):(start + m), 3] <- Y
    data[(start + 1):(start + m), 4] <- X
  }
  return(data)
}

data <- create_uncalibrated_hypotheses(m_vec = rep(1000, 3),
           pi_0_vec = rep(0.6, 3), mus = c(2.5, 3, 3.5), sds = rep(1, 3))

winning_scores <- data[, 1]
all_group_ids <- data[, 2]
labels <- data[, 4]
q_vals <- group_walk(winning_scores, labels, all_group_ids)
```

Index