

# Package ‘DiNAMIC.Duo’

May 7, 2026

**Title** Finding Recurrent DNA Copy Number Alterations and Differences

**Version** 1.0.4

**Description** In tumor tissue, underlying genomic instability can lead to DNA copy number alterations, e.g., copy number gains or losses. Sporadic copy number alterations occur randomly throughout the genome, whereas recurrent alterations are observed in the same genomic region across multiple independent samples, perhaps because they provide a selective growth advantage. Here we use cyclic shift permutations to identify recurrent copy number alterations in a single cohort or recurrent copy number differences in two cohorts based on a common set of genomic markers. Additional functionality is provided to perform downstream analyses, including the creation of summary files and graphics. DiNAMIC.Duo builds upon the original DiNAMIC package of Walter et al. (2011) <[doi:10.1093/bioinformatics/btq717](https://doi.org/10.1093/bioinformatics/btq717)> and leverages the theory developed in Walter et al. (2015) <[doi:10.1093/biomet/asv046](https://doi.org/10.1093/biomet/asv046)>. An article describing DiNAMIC.Duo by Walter et al. (2022) can be found at <[doi:10.1093/bioinformatics/btac542](https://doi.org/10.1093/bioinformatics/btac542)>.

**Depends** R (>= 3.5.0)

**VignetteBuilder** R.rsp

**Suggests** R.rsp

**biocViews** biomaRt

**Imports** biomaRt, dinamic, plyr, httr

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**SystemRequirements** Python, NumPy Python library

**NeedsCompilation** no

**Author** Vonn Walter [aut, cre] (ORCID: <<https://orcid.org/0000-0001-6114-6714>>),  
Hyo Young Choi [aut] (ORCID: <<https://orcid.org/0000-0002-7627-8493>>),  
Xiaobei Zhao [aut] (ORCID: <<https://orcid.org/0000-0002-5277-0846>>),  
Yan Gao [aut] (ORCID: <<https://orcid.org/0000-0002-6852-2038>>),  
David Neil Hayes [aut] (ORCID: <<https://orcid.org/0000-0001-6203-7771>>)

**Maintainer** Vonn Walter <vwalter1@pennstatehealth.psu.edu>

**Repository** CRAN

**Date/Publication** 2025-10-15 05:10:16 UTC

## Contents

|                             |    |
|-----------------------------|----|
| cyclicNullR . . . . .       | 2  |
| cyclicShiftColR . . . . .   | 3  |
| dataPrep . . . . .          | 4  |
| findNull . . . . .          | 5  |
| genomeChrPlot . . . . .     | 6  |
| genomePlot . . . . .        | 8  |
| luadSubset . . . . .        | 10 |
| luscSubset . . . . .        | 11 |
| pD . . . . .                | 11 |
| peelingOne . . . . .        | 12 |
| peelingOneIterate . . . . . | 13 |
| peelingTwo . . . . .        | 14 |
| peelingTwoIterate . . . . . | 16 |
| resultsProcess . . . . .    | 17 |

**Index** **19**

---

|             |  |
|-------------|--|
| cyclicNullR | <i>Create a cyclic shift-based null distribution for one or two copy number matrices</i> |
|-------------|--|

---

## Description

Create a cyclic shift-based null distribution for one or two copy number matrices

## Usage

```
cyclicNullR(X, Y = NULL, numPerms = 100, randomSeed = NULL)
```

## Arguments

|            |   |
|------------|---|
| X          | a matrix or a data frame of copy number data. The rows and columns of X correspond to genes and subjects, respectively.   |
| Y          | a matrix or a data frame of copy number data. The rows and columns of X correspond to genes and subjects, respectively. It is assumed that the rows of X and Y are indexed by the same set of genes that appear in genomic order. |
| numPerms   | the number of cyclic shifts used to create the null distribution. Default = 1e2.  |
| randomSeed | a random seed. Default = NULL.  |

**Details**

This function iteratively calls `cyclicShiftColR` to create an empirical permutation-based null distribution to assess the statistical significance of either (i) the maximum and minimum difference row means of X - row means of Y, or (ii) the maximum and minimum row means of X, depending on whether two or one copy number matrices are being analyzed. The application of cyclic shift permutations to DNA copy number matrices was originally described by Walter et al. (Bioinformatics, 2011;27(5):678–685).

**Value**

a matrix with two columns. The first column, `maxNull`, is an empirical permutation-based null distribution for the maximum difference of row means of X - row means of Y based on cyclic shift permutations of the columns of each matrix; the second column, `minNull`, is an empirical distribution of the minimum difference of the row means of X - the row means of Y based on the same permutations. If Y = NULL, the null distributions apply to the maximum and minimum row means of X.

**Examples**

```
data(DiNAMIC.Duo)
output = cyclicNullR(X = pD[["X"]], Y = pD[["Y"]], numPerms = 25, randomSeed = NULL)
```

---

|                              |  |
|------------------------------|--|
| <code>cyclicShiftColR</code> | <i>Perform the cyclic shift procedure on the columns of a matrix</i> |
|------------------------------|--|

---

**Description**

Perform the cyclic shift procedure on the columns of a matrix

**Usage**

```
cyclicShiftColR(X, randomSeed = NULL)
```

**Arguments**

|                         |   |
|-------------------------|---|
| <code>X</code>          | a matrix or a data frame of copy number data. The rows and columns of X correspond to genes and subjects, respectively. |
| <code>randomSeed</code> | a random seed. Default = NULL.  |

**Details**

Many algorithms for identifying recurrent DNA copy number alterations, e.g., amplifications and deletions, assess statistical significance using permutation-based null distributions. Like many genomic data types, DNA copy number data has an underlying spatial correlation structure. Randomly permuting the values within a given subject ignores this structure, and this can impact type I error rates. In contrast, cyclic permutation largely preserves the local correlation structure.

**Value**

a matrix  $Z$  whose dimensions are the same as  $X$ . Each column of  $Z$  is obtained by perform a cyclic shift of the corresponding column of  $X$ .

**Examples**

```
test = matrix(c(1:50), 10, 5)
cyclicShiftColR(test, randomSeed = NULL)
```

---

 dataPrep

---

*Prepare copy number data for downstream analysis*


---

**Description**

Prepare copy number data for downstream analysis

**Usage**

```
dataPrep(
  X,
  Y = NULL,
  species = c("human", "mouse"),
  ensemblHost = "https://aug2020.archive.ensembl.org"
)
```

**Arguments**

|                          |  |
|--------------------------|--|
| <code>X</code>           | a matrix or a data frame of copy number data. The rows and columns of <code>X</code> correspond to genes and subjects, respectively. <code>X</code> must have gene symbols as its row names.   |
| <code>Y</code>           | an optional matrix or data frame of copy number data to compare with <code>X</code> . As is the case for <code>X</code> , the rows and columns of <code>Y</code> correspond to genes and subjects, respectively. <code>Y</code> must have gene symbols as its row names. |
| <code>species</code>     | a value specifying species for ensembl database to be used; Default is "human".  |
| <code>ensemblHost</code> | specifies the Ensembl archive used; Default is "http://aug2020.archive.ensembl.org".   |

**Details**

This helper function is designed to prepare input matrices or data frames `X` and `Y` containing DNA copy number data for analysis. The downstream functions that `X` and `Y` are indexed by a common set of genes that appear in genomic order. In addition, the `peelingOne` and `peelingTwo` functions require information about the cytoband for each gene, and the `resultsProcess` function uses information about gene position. The `dataPrep` function queries biomaRt so users are not required to provide this information.

**Value**

a list of processed X and Y with a data frame containing gene annotation information.

**Examples**

```
#This runtime for this code slightly exceeds the limits imposed by CRAN.  
data(DiNAMIC.Duo)  
output = dataPrep(X=luadSubset,Y=NULL)
```

---

|          |   |
|----------|---|
| findNull | <i>Find DiNAMIC's null distribution</i> |
|----------|---|

---

**Description**

Find DiNAMIC's null distribution

**Usage**

```
findNull(x, num.perms, random.seed = NULL)
```

**Arguments**

|             |   |
|-------------|---|
| x           | An n by m numeric matrix containing DNA copy number data from n subjects at m markers.                    |
| num.perms   | A positive integer that represents the number of cyclic shifts used to create the empirical distribution. |
| random.seed | An optional random seed (default = NULL).   |

**Details**

The cyclic shift procedure is detailed in *Bioinformatics* (2011) 27(5) 678 - 685. Briefly, cyclic shift is a permutation procedure for DNA copy number data that largely preserves the underlying correlation of the markers. This function uses num.perms cyclic shifts of the copy number matrix x to create an approximate null distribution for  $\max(\text{colSums}(x))$  or  $\min(\text{colSums}(x))$ . The statistical significance of the observed value of  $\max(\text{colSums}(x))$  or  $\min(\text{colSums}(x))$  is assessed by the functions [quickLook](#) and [detailedLook](#).

**Value**

A numerical vector of length num.perms

## Examples

```
random.seed = 12345
set.seed(random.seed)
x = matrix(rnorm(50), 5, 10)
num.perms = 10
findNull(x, num.perms, random.seed)
```

---

|               |  |
|---------------|--|
| genomeChrPlot | <i>A Function for Plotting Mean Copy Number Values and Differences Across Multiple Chromosomes</i> |
|---------------|--|

---

## Description

This function plots mean copy number values from one or two cohorts at a common set of markers across multiple chromosomes.

## Usage

```
genomeChrPlot(
  inputList,
  plottingChrs = NULL,
  lwdVec = rep(1, 3),
  ltyVec = c(1:3),
  lineColorVec = c("red", "blue", "black"),
  ylimLow = -1,
  ylimHigh = 1,
  chrLabel = TRUE,
  xaxisLabel = "Chromosome",
  yaxisLabel = NULL,
  mainLabel = NULL,
  axisCex = 1,
  labelCex = 1,
  xaxisLine = 2.5,
  yaxisLine = 2.5,
  mainLine = 0,
  marginVec = c(4, 4, 3, 3),
  legendText = NULL,
  highThreshold = NULL,
  lowThreshold = NULL,
  showLegend = FALSE,
  legendXQuantile = 0.55,
  legendYCoord = 1
)
```

**Arguments**

|                 |   |
|-----------------|---|
| inputList       | A list produced by dataPrep.  |
| plottingChrs    | A numeric list of chromosomes to be plotted. A separate plot is produced for each chromosome.   |
| lwdVec          | A vector of line widths. Default = rep(1, 3). See <a href="#">par</a> .   |
| ltyVec          | A vector of line types. Default = c(1:3). See <a href="#">par</a> .   |
| lineColorVec    | A vector of line colors. Default = c("red", "blue", "black").   |
| ylimLow         | The lower limit of the y-values in the plot. Default = -1. See <a href="#">plot</a> .   |
| ylimHigh        | The upper limit of the y-values in the plot. Default = 1. See <a href="#">plot</a> .  |
| chrLabel        | Binary value determining whether or not chromosomes are labeled. Default = TRUE.  |
| xaxisLabel      | Label for the x-axis in the plot. Default = "Chromosome". See <a href="#">plot</a> .  |
| yaxisLabel      | Label for the y-axis in the plot. Default = NULL. See <a href="#">plot</a> .  |
| mainLabel       | Main label in the plot. Default = NULL. See <a href="#">plot</a> .  |
| axisCex         | Point size for the scale on the axis. Default = 1. See <a href="#">par</a> .  |
| labelCex        | Point size for the axis label. Default = 1. See <a href="#">par</a> .   |
| xaxisLine       | Numerical value used to specify the location (line) of the x-axis label. Default = 2.5. See <a href="#">mtext</a> .                               |
| yaxisLine       | Numerical value used to specify the location (line) of the y-axis label. Default = 2.5. See <a href="#">mtext</a> .                               |
| mainLine        | Numerical value used to specify the location (line) of the main.label. Default = 0. See <a href="#">mtext</a> .                                   |
| marginVec       | Numerical vector specifying margin sizes. Default = c(4, 4, 3, 3). See <a href="#">par</a> .  |
| legendText      | Character vector used in the legend. Only shown if showLegend = TRUE. Default = NULL. See <a href="#">legend</a> .                                |
| highThreshold   | Numerical value representing the position of the upper horizontal line, e.g., a threshold for assessing statistical significance. Default = NULL. |
| lowThreshold    | Numerical value representing the position of the lower horizontal line, e.g., a threshold for assessing statistical significance. Default = NULL. |
| showLegend      | Binary value determining whether or not the legend is shown. Default = FALSE. See <a href="#">legend</a> .  |
| legendXQuantile | Quantile to specify the "x" location of the legend. Only relevant if showLegend = TRUE. Default = 0.55. See <a href="#">legend</a> .              |
| legendYCoord    | Numerical value to specify the "y" location of the legend. Only relevant if showLegend = TRUE. Default = 1. See <a href="#">legend</a> .          |

## Details

Although `genomePlot` can be used to visualize copy number values and copy number alterations across the genome, the scale makes it difficult to see events that affect small genomic regions. These events are easier to see if the viewing window is restricted to individual chromosomes, as is done here. If `Y = NULL` in the input list, then the plot shows a single line corresponding to the mean DNA copy number values based on the entries in `X`. If both `X` and `Y` are specified, the plot shows three lines corresponding to the mean DNA copy number values in `X`, the mean DNA copy number values in `Y`, and the difference of the mean DNA copy number values.

## Value

Creates a multi-page plot of mean copy number values and differences by chromosome.

## Examples

```
genomeChrPlot(inputList = pD, ylimLow = -1.4, ylimHigh = 1.4)
```

---

genomePlot

*A Function for Plotting Mean Copy Number Values and Differences Across the Genome*

---

## Description

This function plots mean copy number values from one or two cohorts at a common set of markers across the genome.

## Usage

```
genomePlot(
  inputList,
  lwdVec = rep(1, 3),
  ltyVec = c(1:3),
  lineColorVec = c("red", "blue", "black"),
  ylimLow = -1,
  ylimHigh = 1,
  chrLabel = TRUE,
  xaxisLabel = "Chromosome",
  yaxisLabel = NULL,
  mainLabel = NULL,
  rectColors = c("light gray", "gray"),
  axisCex = 1,
  labelCex = 1,
  xaxisLine = 2.5,
  yaxisLine = 2.5,
  mainLine = 0,
  marginVec = c(4, 4, 3, 3),
```

```

    legendText = NULL,
    highThreshold = NULL,
    lowThreshold = NULL,
    showLegend = FALSE,
    legendXQuantile = 0.55,
    legendYCoord = 1
)

```

## Arguments

|                 |   |
|-----------------|---|
| inputList       | A list produced by dataPrep.  |
| lwdVec          | A vector of line widths. Default = rep(1, 3). See <a href="#">par</a> .   |
| ltyVec          | A vector of line types. Default = c(1:3). See <a href="#">par</a> .   |
| lineColorVec    | A vector of line colors. Default = c("red", "blue", "black"). See <a href="#">par</a> .   |
| ylimLow         | The lower limit of the y-values in the plot. Default = -1. See <a href="#">plot</a> .   |
| ylimHigh        | The upper limit of the y-values in the plot. Default = 1. See <a href="#">plot</a> .  |
| chrLabel        | Binary value determining whether or not chromosomes are labeled. Default = TRUE.  |
| xaxisLabel      | Label for the x-axis in the plot. Default = "Chromosome". See <a href="#">plot</a> .  |
| yaxisLabel      | Label for the y-axis in the plot. Default = NULL. See <a href="#">plot</a> .  |
| mainLabel       | Main label in the plot. Default = NULL. See <a href="#">plot</a> .  |
| rectColors      | Background colors for different chromosomes. Default = c("light gray", "gray").   |
| axisCex         | Point size for the scale on the axis. Default = 1. See <a href="#">par</a> .  |
| labelCex        | Point size for the axis label. Default = 1. See <a href="#">par</a> .   |
| xaxisLine       | Numerical value used to specify the location (line) of the x-axis label. Default = 2.5. See <a href="#">mtext</a> .                               |
| yaxisLine       | Numerical value used to specify the location (line) of the y-axis label. Default = 2.5. See <a href="#">mtext</a> .                               |
| mainLine        | Numerical value used to specify the location (line) of the main.label. Default = 0. See <a href="#">mtext</a> .                                   |
| marginVec       | Numerical vector specifying margin sizes. Default = c(4, 4, 3, 3). See <a href="#">par</a> .  |
| legendText      | Character vector used to legend. Only shown if showLegend = TRUE. Default = NULL. See <a href="#">legend</a> .                                    |
| highThreshold   | Numerical value representing the position of the upper horizontal line, e.g., a threshold for assessing statistical significance. Default = NULL. |
| lowThreshold    | Numerical value representing the position of the lower horizontal line, e.g., a threshold for assessing statistical significance. Default = NULL. |
| showLegend      | Binary value determining whether or not the legend is shown. Default = FALSE. See <a href="#">legend</a> .  |
| legendXQuantile | Quantile to specify the "x" location of the legend. Only relevant if showLegend = TRUE. Default = 0.55. See <a href="#">legend</a> .              |
| legendYCoord    | Numerical value to specify the "y" location of the legend. Only relevant if showLegend = TRUE. Default = 1. See <a href="#">legend</a> .          |

**Details**

This function is used to visualize copy number values and copy number alterations across the genome. If `Y = NULL` in the input list, then the plot shows a single line corresponding to the mean DNA copy number values based on the entries in `X`. If both `X` and `Y` are specified, the plot shows three lines corresponding to the mean DNA copy number values in `X`, the mean DNA copy number values in `Y`, and the difference of the mean DNA copy number values.

**Value**

Creates a genomewide plot of mean copy number values and differences.

**Examples**

```
genomeChrPlot(inputList = pD, ylimLow = -1.4, ylimHigh = 1.4)
```

---

|            |  |
|------------|--|
| luadSubset | <i>DNA copy number data for lung adenocarcinoma.</i> |
|------------|--|

---

**Description**

A subset of the DNA copy number data from the TCGA lung lung adenocarcinoma cohort.

**Usage**

```
luadSubset
```

**Format**

A numeric matrix of quantitative DNA copy number values with 3475 rows and 65 columns. Rows are indexed by genes, and columns are indexed by samples. The entries are on the log ratio scale and were produced by the GISTIC pipeline.

**Details**

The original data set downloaded from <https://gdac.broadinstitute.org/> contained gene-level segmented DNA copy number values for 24776 genes and 516 samples that were produced by the GISTIC pipeline. Because of the size limitations on data in R packages, a random subset of genes and samples was selected. By construction, the number of genes and samples in the lung adenocarcinoma data is different than the number of genes and samples in the lung squamous cell carcinoma data.

**Source**

<https://gdac.broadinstitute.org/>

---

|            |   |
|------------|---|
| luscSubset | <i>DNA copy number data for lung squamous cell carcinoma.</i> |
|------------|---|

---

**Description**

A subset of the DNA copy number data from the TCGA lung lung squamous cell carcinoma cohort.

**Usage**

```
luscSubset
```

**Format**

A numeric matrix of quantitative DNA copy number values with 3480 rows and 60 columns. Rows are indexed by genes, and columns are indexed by samples. The entries are on the log ratio scale and were produced by the GISTIC pipeline.

**Details**

The original data set downloaded from <https://gdac.broadinstitute.org/> contained gene-level segmented DNA copy number values for 24776 genes and 501 samples that were produced by the GISTIC pipeline. Because of the size limitations on data in R packages, a random subset of genes and samples was selected. By construction, the number of genes and samples in the lung squamous cell carcinoma data is different than the number of genes and samples in the lung adenocarcinoma data.

**Source**

<https://gdac.broadinstitute.org/>

---

|    |                                      |
|----|--------------------------------------|
| pD | <i>Prepped data for DiNAMIC.Duo.</i> |
|----|--------------------------------------|

---

**Description**

A list produced by the dataPrep() function.

**Usage**

```
pD
```

**Format**

A list containing three components for a common set of genes.

**Details**

The components of the list are named X, Y, and posDT. They contain the DNA copy number data for lung adenocarcinoma, the DNA copy number data for lung squamous cell carcinoma, and gene position data, respectively. This data object was produced by applying [dataPrep](#) using the [luadSubset](#) and [luscSubset](#) data sets. This reduces run time when the package is compiled by CRAN, thus eliminating run time errors.

---

|            |   |
|------------|---|
| peelingOne | <i>A Function to Apply the Peeling Algorithm in a Single Copy Number Matrix</i> |
|------------|---|

---

**Description**

This function applies the peeling algorithm described in Walter et al. (Bioinformatics, 2011;27(5):678–685)

**Usage**

```
peelingOne(X, posDT, k, threshold = NULL)
```

**Arguments**

|           |   |
|-----------|---|
| X         | A matrix of normalized gene-level copy number data (rows = genes, columns = subjects).  |
| posDT     | A data frame containing genomic position information for the genes in X.  |
| k         | The location (row of X) containing the peak that will be peeled.  |
| threshold | A tuning parameter that controls the size of the peeled region. Rows of X with mean copy number less than threshold will not be peeled. |

**Details**

to remove a peak from a copy number data set and define a genomic interval of interest around the peak.

Tumor genomes often contain multiple DNA copy number alterations, e.g., amplifications or deletions. The locus that harbors the most extreme alteration, k, as evidenced by the maximum or minimum column mean, provides a point estimate for the location of an underlying driver gene. Also, loci near k may be affected by the same underlying genomic event. The peeling procedure is applied to "nullify" entries in X that contribute to the alteration at k, thus making it possible to identify altered regions elsewhere in the genome. This function is called by [peelingOneIterate](#).

**Value**

A list containing two elements: X and interval. X is an updated version of the input copy number matrix in which the peak at k has been removed, and interval is genomic region containing k. By construction, interval cannot extend beyond the chromosome arm containing k.

**Examples**

```

lusc=pD[["X"]]

posDT=pD[["posDT"]]

kLusc=which.max(rowMeans(lusc))

peeledLusc=peelingOne(X=lusc,posDT=posDT,k=kLusc,threshold=NULL)

```

---

|                   |   |
|-------------------|---|
| peelingOneIterate | <i>A Function to Apply the Peeling Algorithm in a Single Copy Number Matrix</i> |
|-------------------|---|

---

**Description**

This function iteratively applies the peelingOne function, thereby identifying multiple

**Usage**

```

peelingOneIterate(
  X,
  posDT,
  gain = TRUE,
  nullDist = NULL,
  threshold = NULL,
  numIters = 5
)

```

**Arguments**

|           |   |
|-----------|---|
| X         | A matrix of normalized gene-level copy number data (rows = genes, columns = subjects).  |
| posDT     | A data frame containing genomic position information for the genes in X.  |
| gain      | A logical value indicating whether gains (TRUE) or losses (FALSE) will be peeled.<br>Default = TRUE.  |
| nullDist  | An empirical null distribution produced by the cyclic shift algorithm.<br>Default = NULL.   |
| threshold | A tuning parameter that controls the size of the peeled region. Rows of X with mean copy number less than threshold will not be peeled. Default = NULL. |
| numIters  | The number of times peelingOne will be iterated. Default = 5.   |

## Details

peaks across the genome in a single cohort. Gains and losses should be analyzed separately.

The `peelingOne` function applies the peeling algorithm described by Walter et al. (Bioinformatics, 2011;27(5):678–685) at a given marker  $k$ . Because tumor genomes may contain multiple regions of copy number gain or loss, it is important to apply the peeling process iteratively, thereby identifying multiple markers and surrounding genomic regions.

## Value

A list containing two elements:  $X$  and interval.  $X$  is an updated version of the input copy number matrix in which the peak at  $k$  has been removed, and interval is genomic region containing  $k$ . By construction, interval cannot extend beyond the chromosome arm containing  $k$ .

## Examples

```
lusc=pD[["X"]]
posDT=pD[["posDT"]]
gain = TRUE
nullDist = NULL
threshold = NULL
numIters = 3
peeledLusc=peelingOneIterate(X=lusc,posDT=posDT,gain=TRUE,nullDist=NULL,threshold=NULL,numIters=3)
```

---

peelingTwo

*A Function to Apply the Peeling Algorithm in a Two Copy Number Matrices*

---

## Description

This function applies a modified version of the peeling algorithm originally described in Walter et al.,

## Usage

```
peelingTwo(X, Y, posDT, k, threshold = NULL)
```

**Arguments**

|           |   |
|-----------|---|
| X         | A matrix of normalized gene-level copy number data (rows = genes, columns = subjects).  |
| Y         | A matrix of normalized gene-level copy number data (rows = genes, columns = subjects).  |
| posDT     | A data frame containing genomic position information for the genes in X.  |
| k         | The location (row of X and Y) containing the peak that will be peeled.  |
| threshold | A tuning parameter that controls the size of the peeled region. Rows in which $\text{rowMeans}(X) - \text{rowMeans}(Y)$ are less than threshold will not be peeled. |

**Details**

(PMID 21183584) to remove a peak from the copy number differences and define a genomic interval of interest

around the peak.

When tumor genomes from two cohorts are compared, there may be multiple regions that harbor copy number differences. For example, gains or losses may be present in only one of the two cohorts, and this could give rise to copy number differences. Alternatively, the same region of the genome may exhibit gain or loss in both cohorts. If the magnitudes of the common gain or loss are distinct, then this also gives rise to copy number differences. The locus that harbors the most extreme difference, k, provides a point estimate for the underlying driver gene that gives rise to the difference. Loci near k may also be affected by the underlying difference in copy number. The peeling procedure for two cohorts is applied to "nullify" entries of both X and Y that contribute to the alteration at k, thus making it possible to identify other regions of the genome that harbor copy number differences. This function is called by [peelingTwoIterate](#).

**Value**

A list containing three elements: X, Y, and interval. X and Y are updated versions of the input copy number matrices X and Y in which the peak at k has been removed, and interval is genomic region containing k. By construction, interval cannot extend beyond the chromosome arm containing k.

**Examples**

```
luad=pD[["X"]]
lusc=pD[["Y"]]
posDT=pD[["posDT"]]
kDiff=which.max(rowMeans(luad)-rowMeans(lusc))
peeledDiff=peelingTwo(X=luad,Y=lusc,posDT=posDT,k=kDiff,threshold=NULL)
```

---

peelingTwoIterate      *A Function to Apply the Peeling Algorithm for Two Copy Number Matrices*

---

### Description

This function iteratively applies the peelingTwo function, thereby identifying multiple

### Usage

```
peelingTwoIterate(
  X,
  Y,
  posDT,
  gain = TRUE,
  nullDist = NULL,
  threshold = NULL,
  numIters = 5
)
```

### Arguments

|           |  |
|-----------|--|
| X         | A matrix of normalized gene-level copy number data (rows = genes, columns = subjects).   |
| Y         | A matrix of normalized gene-level copy number data (rows = genes, columns = subjects).   |
| posDT     | A data frame containing genomic position information for the genes in X.   |
| gain      | A logical value indicating whether gains (TRUE) or losses (FALSE) will be peeled.<br>Default = TRUE.   |
| nullDist  | An empirical null distribution produced by the cyclic shift algorithm.<br>Default = NULL.  |
| threshold | A tuning parameter that controls the size of the peeled region. Rows of X and Y with mean copy number differences less than threshold will not be peeled.<br>Default = NULL. |
| numIters  | The number of times peelingTwo will be iterated. Default = 5.  |

### Details

differences across the genome between a two cohorts. Gains and losses should be analyzed separately.

The `peelingTwo` function applies the peeling procedure for two cohorts to "nullify" entries in two copy number matrices X and Y that give rise to the most significant copy number difference. Because tumor genomes may contain multiple regions that harbor copy number differences, it is important to apply the peeling procedure for two cohorts iteratively, thereby identifying multiple markers and surrounding genomic regions.

**Value**

A list containing two elements: X, Y, and interval. X and Y are updated versions of the input copy number matrices in which the peak difference at k has been removed, and interval is genomic region containing k. By construction, interval cannot extend beyond the chromosome arm containing k.

**Examples**

```
l1ad=pD[["X"]]
l1usc=pD[["Y"]]
posDT=pD[["posDT"]]
gain = TRUE
nullDist = NULL
threshold = NULL
numIters = 3
out=peelingTwoIterate(X=l1ad,Y=l1usc,posDT=posDT,gain=TRUE,nullDist=NULL,threshold=NULL,numIters=3)
```

---

|                |                                   |
|----------------|-----------------------------------|
| resultsProcess | <i>Processing peeling results</i> |
|----------------|-----------------------------------|

---

**Description**

Processing peeling results

**Usage**

```
resultsProcess(peel.results, posDT)
```

**Arguments**

|              |  |
|--------------|--|
| peel.results | peeling results  |
| posDT        | a data frame containing gene annotation information; a list component created by dataPrep. |

**Details**

The [peelingOneIterate](#) function identifies (i) multiple loci across the genome where copy number gains are losses, and (ii) regions around those loci that also exhibit copy number changes. Similarly, [peelingTwoIterate](#) identifies loci and surrounding regions that harbor copy number differences. The output of either [peelingOneIterate](#) or [peelingTwoIterate](#) is processed to produce a tab-delimited text file that provides a summary of the peeling results. Each column corresponds to a

peeled locus, and the column contains the genomic location of the locus, the start and end positions of the surrounding peeled region, the mean copy number value (one matrix X) or difference of mean copy number values (two matrices X and Y), the cyclic shift-based p-value, and the names of the genes in the peeled region (in alphabetical order).

**Value**

processed peeling results with a list of genes corresponding to each peeled region

**See Also**

[dataPrep](#)

# Index

## \* datasets

luadSubset, 10

luscSubset, 11

pD, 11

cyclicNullR, 2

cyclicShiftColR, 3, 3

dataPrep, 4, 12, 18

detailedLook, 5

findNull, 5

genomeChrPlot, 6

genomePlot, 8, 8

legend, 7, 9

luadSubset, 10, 12

luscSubset, 11, 12

mtext, 7, 9

par, 7, 9

pD, 11

peelingOne, 12, 14

peelingOneIterate, 12, 13, 17

peelingTwo, 14, 16

peelingTwoIterate, 15, 16, 17

plot, 7, 9

quickLook, 5

resultsProcess, 17