# Package 'ComPrAn'

April 2, 2026

**Type** Package

**Title** Complexome Profiling Analysis package

**Version** 1.18.0

**Description** This package is for analysis of SILAC labeled complexome profiling data. It uses peptide table in tab-delimited format as an input and produces ready-to-use tables and plots.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** data.table, dplyr, forcats, ggplot2, magrittr, purrr, tidyr, rlang, stringr, shiny, DT, RColorBrewer, VennDiagram, rio, scales, shinydashboard, shinyjs, stats, tibble, grid

**RoxygenNote** 7.1.1

**Suggests** testthat (>= 2.1.0), knitr, rmarkdown

**VignetteBuilder** knitr

**biocViews** MassSpectrometry, Proteomics, Visualization

**git_url** https://git.bioconductor.org/packages/ComPrAn

**git_branch** RELEASE_3_22

**git_last_commit** 16928c0

**git_last_commit_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2026-04-02

**Author** Rick Scavetta [aut], Petra Palenikova [aut, cre] (ORCID: <https://orcid.org/0000-0002-2465-4370>)

**Maintainer** Petra Palenikova <palenikova3@gmail.com>

## Contents

---

`allPeptidesPlot` *Create scatter plot*

---

## Description

This function creates a plot of all peptides that belong to a single protein

## Usage

```
allPeptidesPlot(
  .listDF,
  protein,
  max_frac,
  meanLine = FALSE,
  repPepLine = FALSE,
  separateLabStates = FALSE,
  grid = TRUE,
  titleLabel = "all",
  titleAlign = "left",
  ylabel = "Precursor Area",
  xlabel = "Fraction",
  legendLabel = "Condition",
  labelled = "Labeled",
  unlabelled = "Unlabeled",
  controlSample = "",
  textSize = 12,
  axisTextSize = 8
)
```

## Arguments

| | |
|---|---|
| .listDF | list, list containing data frames of peptides for each protein indexed by 'Protein Group Accessions' |
| protein | character, 'Protein Group Accession' to show in the plot |
| max_frac | numeric, total number of fractions |
| meanLine | logical, specifies whether to plot a mean line |
| repPepLine | logical, specifies whether to plot a representative peptide line |
| separateLabStates | logical, specifies whether label states will be separated into facets |
| grid | logical, specifies presence/absence of gridline in the plot |
| titleLabel | character, what to call the plot |
| titleAlign | character, one of the 'left', 'center'/'centre', 'right', specifies alignment of the title in plot |
| ylabel | character |
| xlabel | character |
| legendLabel | character |
| labelled | character, label to be used for isLabel == TRUE |
| unlabelled | character, label to be used for isLabel == FALSE |
| controlSample | character, either labelled or unlabelled, this setting will adjust coloring based on which sample is a control |
| textSize | numeric, size of text in the plot |
| axisTextSize | numeric, size of axis labels in the plot |

## Value

plot

## Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
## separate chemical modifications and labelling into separate columns
peptides <- splitModLab(peptides)
## remove unneccessary columns, simplify rows
peptides <- simplifyProteins(peptides)
## Pick representative peptide for each protein for both scenarios
peptide_index <- pickPeptide(peptides)

##create a plot showing all peptides of selected protein
protein <- "P52815"
max_frac <- 23
#default plot
allPeptidesPlot(peptide_index,protein, max_frac = max_frac)
#other plot version
allPeptidesPlot(peptide_index,protein, max_frac = max_frac,
repPepLine = TRUE, grid = FALSE, titleAlign = "center")
#other plot version
```

```
allPeptidesPlot(peptide_index,protein, max_frac = max_frac,
repPepLine = TRUE, meanLine = TRUE, separateLabStates =TRUE,
titleLabel = "GN")
```

---

assignClusters *Create a data frames with cluster assignment*

---

## Description

This function creates a data frame with column specifying clusters assigned ot each protein using the table and distance matrix produced by clusterComp() function.

## Usage

```
assignClusters(.listDf, sample, method = "complete", cutoff = 0.5)
```

## Arguments

| | |
|---|---|
| .listDf | list of data frames produced by clusterComp() function |
| sample | which of the two samples you want to apply the function to (labeled/unlabeled). |
| method | character, One of 'average', 'single' or 'complete' (default), specifies the linkage method to be used inside R hclust() function |
| cutoff | numeric, specifies the h value in R cutree() function, height at which to 'cut the tree', everything with distance below this value is assigned into same cluster everything with larger distance is in a different cluster extreme possible values are 0 to 2 (might not be reached for all data sets) |

## Value

dataframe

## See Also

clusterComp

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
# create components necessary for clustering
clusteringDF <- clusterComp(forAnalysis,scenar = "A", PearsCor = "centered")
#assign clusters
labTab_clust <- assignClusters(.listDf = clusteringDF,sample = "labeled",
method = 'complete', cutoff = 0.5)
unlabTab_clust <- assignClusters(.listDf = clusteringDF,sample = "unlabeled",
                                 method = 'complete', cutoff = 0.5)
```

`cleanData` *Clean raw peptide complexomics data*

## Description

Perform initial, mandatory, cleaning of data Function to process raw input data into format required for subsequent analysis. .data is a data frame containing raw input data. This function checks (not neccessarily in this order):

- renames Sequence ID column to Fraction and converts values in this column from letters to numbers
- reorders Protein Group Accessions containing multiple proteins
- removes rows in which PSM Ambiguity == 'Rejected'
- removes rows in which # Protein Groups == 0
- removes rows in which Precursor Area is NA
- removes cols that are not used at all

## Usage

```
cleanData(.data, fCol = "Search ID")
```

## Arguments

| | |
|---|---|
| `.data` | dataframe |
| `fCol` | character The column containing the fractions, e.g. "Search ID" (default) |

## Value

dataframe

## Author(s)

Petra Palenikova <pp451@cam.ac.uk>

Rick Scavetta <office@scavetta.academy>

## Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
```

clusterComp *Create components necessary for clustering*

## Description

Reformat the table for the one neccessary for assignClusters function. Calculate the distance matirx using selected variant of correlation.

## Usage

```
clusterComp(.df, scenar = "A", PearsCor = "centered")
```

## Arguments

| | |
|---|---|
| .df | data frame, table of normalised protein values |
| scenar | character, scenario intended for clustering, either "A" or "B" |
| PearsCor | character, pearsons correlation variant (centered/uncentered) |

## Value

list of data frames

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
# create components necessary for clustering
clusteringDF <- clusterComp(forAnalysis,scenar = "A", PearsCor = "centered")
```

compranApp *Execute the complexomics Shiny app*

## Description

Execute the complexomics Shiny app

## Usage

```
compranApp()
```

## Value

Shiny app

### Examples

```
#' @examples
##to start the shiny app associated with ComPrAn package run
if(interactive()){
    compranApp()
}
```

---

`exportClusterAssignments` *Covert clustered tables into format for export*

---

### Description

Covert clustered tables into format for export

### Usage

```
exportClusterAssignments(labClustTable, unlabClustTable)
```

### Arguments

| | |
|---|---|
| `labClustTable` | output: data frame containing columns: 'Protein Group Accessions' character 'Protein Descriptions' character 'Cluster number - unlabeled' integer 'Cluster number - labeled' integer |
| `unlabClustTable` | labClustTable, unlabClustTable: data frames, contain columns: 'Protein Group Accessions' character 'Protein Descriptions' character isLabel character ('TRUE'/'FALSE') - here in one data frame all are TRUE in second one all are FALSE columns 1 to n, numeric, n is the total number of fractions/slices, each of this columns contains 'Precursor Area' values in a given fraction(columns) for a protein(rows) cluster integer |

### Value

dataframe

### Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
# create components necessary for clustering
clusteringDF <- clusterComp(forAnalysis,scenar = "A", PearsCor = "centered")
#assign clusters
labTab_clust <- assignClusters(.listDf = clusteringDF,sample = "labeled",
method = 'complete', cutoff = 0.5)
unlabTab_clust <- assignClusters(.listDf = clusteringDF,sample = "unlabeled",
                                 method = 'complete', cutoff = 0.5)
#make table of cluster assginment
tableClusterExport <- exportClusterAssignments(labTab_clust,unlabTab_clust)
```

## extractRepPeps *Extract Only Data Belonging to Representative Peptide*

### Description

Incomplete labelling - there are cases when in peptides containing multiple Lys/Arg not all of them are heavy in labelled samples. As in SILAC we assume that addition of label does not affect peptide properties, we are taking a mean ‘Precursor Area‘ value as the representative ‘Precursor Area‘ in such cases.

### Usage

```
extractRepPeps(.data, scenario, label = "Label neccessary for scenario A")
```

### Arguments

| | |
|---|---|
| .data | dataframe containing all peptides of one protein |
| scenario | character "A", or "B" |
| label | character, selects for which label state the representative peptides will be exported, can have value of "TRUE" or "FALSE", required only for scenario "A" |

### Value

dataframe containing only representative peptide

## getNormTable *Get normalised table for all proteins*

### Description

Extracts values for representative peptides for each protein, for both scenario A and scenario B. Results are combined into one data frame in a format either indended for further analysis or for export.

### Usage

```
getNormTable(.listDf, purpose = "analysis")
```

### Arguments

| | |
|---|---|
| .listDf | list of data frames |
| purpose | character, purpose of use of function output, values either "analysis" of "export" |

### Value

dataframe

## Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
## separate chemical modifications and labelling into separate columns
peptides <- splitModLab(peptides)
## remove unneccessary columns, simplify rows
peptides <- simplifyProteins(peptides)
## Pick representative peptide for each protein for both scenarios
peptide_index <- pickPeptide(peptides)
## extract table with normalised protein values for both scenarios
forAnalysis <- getNormTable(peptide_index,purpose = "analysis")
```

---

# groupHeatMap *Make heatmap*

---

## Description

This function creates a heatmap for a subset of proteins in dataFrame specified in groupData, heatmap is divided into facets according to isLabel

## Usage

```
groupHeatMap(
  dataFrame,
  groupData,
  groupName,
  titleAlign = "left",
  newNamesCol = NULL,
  colNumber = 2,
  ylabel = "Protein",
  xlabel = "Fraction",
  legendLabel = "Relative Protein Abundance",
  legendPosition = "right",
  grid = TRUE,
  labelled = "labeled",
  unlabelled = "unlabeled",
  orderColumn = NULL
)
```

## Arguments

| | |
|---|---|
| dataFrame | data frame, contains columns: 'Protein Group Accessions' character 'Protein Descriptions' character Fraction integer isLabel character ('TRUE'/'FALSE' values) 'Precursor Area' double scenario character |
| groupData | data frame, mandatory column: 'Protein Group Accessions' character - this column is used for filtering optional columns: any other column of type character that should be used for renaming |
| groupName | character, name that should be used for the group specified in groupData |

| | |
|---|---|
| titleAlign | character, one of the 'left', 'center'/'centre', 'right', specifies alignment of the title in plot |
| newNamesCol | character, if groupData contains column for re-naming and you want to use it, specify the column name in here |
| colNumber | numeric, values of 1 or 2, specifies whether facets will be shown side-by-side or above each other |
| ylabel | character |
| xlabel | character |
| legendLabel | character |
| legendPosition | character, one of "right" or "bottom" |
| grid | logical, specifies presence/absence of gridline in the plot |
| labelled | character, label to be used for isLabel == TRUE |
| unlabelled | character, label to be used for isLabel == FALSE |
| orderColumn | character, if groupData contains column for re-ordering and you want to use it, specify the column name in here |

## Value

plot

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
##example plot:
groupDfn <- system.file("extData", "exampleGroup.txt", package = "ComPrAn")
groupName <- 'group1'
groupData <- data.table::fread(groupDfn)
groupHeatMap(forAnalysis[forAnalysis$scenario == "B",], groupData, groupName)
```

---

# makeBarPlotClusterSummary

*Title*

---

## Description

Title

## Usage

```
makeBarPlotClusterSummary(df, name = "sample 1")
```

### Arguments

| | |
|---|---|
| df | data frame, contains columns: 'Protein Group Accessions' character 'Protein Descriptions' character isLabel character ('TRUE'/'FALSE') columns 1 to n, numeric, n is the total number of fractions/slices, each of this columns contains 'Precursor Area' values in a given fraction(columns) for a protein(rows) cluster integer |
| name | character, specifies the name of the sample |

### Value

plot

### Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
# create components necessary for clustering
clusteringDF <- clusterComp(forAnalysis,scenar = "A", PearsCor = "centered")
#assign clusters
labTab_clust <- assignClusters(.listDf = clusteringDF,sample = "labeled",
method = 'complete', cutoff = 0.5)
unlabTab_clust <- assignClusters(.listDf = clusteringDF,sample = "unlabeled",
                                 method = 'complete', cutoff = 0.5)
#Make bar plots for labeled and unlabeled samples
makeBarPlotClusterSummary(labTab_clust, name = 'labeled')
makeBarPlotClusterSummary(unlabTab_clust, name = 'unlabeled')
```

---

## makeDist *Make disstance matrix*

---

### Description

This function calculates distance matrix for a data frame, column by column requires uncenteredCor function to work

### Usage

```
makeDist(df, centered = FALSE)
```

### Arguments

| | |
|---|---|
| df | data frame, contains columns: 'Protein Group Accessions' character 'Protein Descriptions' character isLabel character ('TRUE'/'FALSE') columns 1 to n, numeric, n is the total number of fractions/slices, each of this columns contains 'Precursor Area' values in a given fraction(columns) for a protein(rows) |
| centered | centered: logical,if TRUE return dist matrix based on centered Pearson correlation (uses R cor() function, fast) ,if FALSE return dist matrix based on uncentered Pearson correlation (uses custom uncenteredCor() function, slow) |

### Value

matrix

---

`normalizeTable` *Convert extractRepPeps output to a Matrix*

---

### Description

Convert the dataframe as output from extractRepPeps() to matrix-like table return normalized or raw values of Precursor Area, by default return normalized values

### Usage

```
normalizeTable(.data, applyNormalization = TRUE)
```

### Arguments

| | |
|---|---|
| `.data` | a dataframe |
| `applyNormalization` | logical apply normalization or not |

### Value

a matrix

---

`normTableForExport` *Convert Normalized Dataframe to Export format*

---

### Description

This is a convenient function for plotting

### Usage

```
normTableForExport(labTab, unlabTab, comboTab)
```

### Arguments

| | |
|---|---|
| `labTab` | a dataframe |
| `unlabTab` | a dataframe |
| `comboTab` | a dataframe |

### Value

a dataframe

normTableWideToLong *Convert Normalized Dataframe To Long format*

## Description

This is a convenient function for plotting

## Usage

```
normTableWideToLong(labTab, unlabTab, comboTab)
```

## Arguments

| | |
|---|---|
| labTab | a dataframe |
| unlabTab | a dataframe |
| comboTab | a dataframe |

## Value

a dataframe

oneGroupTwoLabelsCoMigration
*Compare a Single Group of Proteins Between Two Label States*

## Description

This function creates a ?scatter plot? for a subset of proteins in dataFrame specified in groupData. Intended use of the function - using scenario A data, compare shape of the migration profile for a SINGLE GROUP of proteins BETWEEN the two LABEL STATES.

## Usage

```
oneGroupTwoLabelsCoMigration(
  dataFrame,
  max_frac,
  groupData = NULL,
  groupName = "group1",
  meanLine = FALSE,
  medianLine = FALSE,
  ylabel = "Relative Protein Abundance",
  xlabel = "Fraction",
  legendLabel = "Condition",
  labelled = "Labeled",
  unlabelled = "Unlabeled",
  jitterPoints = 0.3,
  pointSize = 2.5,
  grid = FALSE,
  titleAlign = "left",
```

```
  alphaValue = 1,
  controlSample = "",
  textSize = 12,
  axisTextSize = 8
)
```

## Arguments

| | |
|---|---|
| dataFrame | dataFrame: data frame, data frame of normalised values for proteins from SCENARIO A, contains columns: 'Protein Group Accessions' character 'Protein Descriptions' character Fraction integer isLabel character ('TRUE'/'FALSE' values) 'Precursor Area' double scenario character |
| max_frac | numeric, total number of fractions |
| groupData | character vector, contins list of Protein Group Accessions that belong to the group we want to plot |
| groupName | character, name that should be used for the group specified in groupData |
| meanLine | logical, specifies whether to plot a mean line for all values in the group |
| medianLine | logical, specifies whether to plot a median line for all values in the group |
| ylabel | character |
| xlabel | character |
| legendLabel | character |
| labelled | character, label to be used for isLabel == TRUE |
| unlabelled | character, label to be used for isLabel == FALSE |
| jitterPoints | numeric |
| pointSize | numeric, size of the point in the plot |
| grid | logical, specifies presence/absence of gridline in the plot |
| titleAlign | character, one of the 'left', 'center'/'centre', 'right', specifies alignment of the title in plot |
| alphaValue | numeric, transparency of the point, values 0 to 1 |
| controlSample | character, either labelled or unlabelled, this setting will adjust plot coloring based on which sample is a control |
| textSize | numeric, size of text in the plot |
| axisTextSize | numeric, size of axis labels in the plot |

## Value

plot

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
##example plot:
groupDV <- c("Q16540","P52815","P09001","Q13405","Q9H2W6")
groupName <- 'group1'
max_frac <- 23
oneGroupTwoLabelsCoMigration(forAnalysis, max_frac, groupDV,groupName)
```

## onlyInOneLabelState *Report Proteins Present In Only One Label State*

### Description

This function returns NAMES of proteins present in only labelled/only unlabelld or both label states

### Usage

```
onlyInOneLabelState(.data)
```

### Arguments

| | |
|---|---|
| .data | An environment containing dataframes |

### Value

a list with 3 items, each item is a vector containing names belonging to one of 3 groups

### Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
## separate chemical modifications and labelling into separate columns
peptides <- splitModLab(peptides)
## remove unneccessary columns, simplify rows
peptides <- simplifyProteins(peptides)
## Pick representative peptide for each protein for both scenarios
peptide_index <- pickPeptide(peptides)
## extract list of names of proteins present in one/both samples
oneStateList <- onlyInOneLabelState(peptide_index)
```

## peptideImport *Import raw peptide complexomics data*

### Description

Check presence of required columns inputFile is a character vector containing the location of peptide file This function checks:

- are all required columns present
- are these columns in correct format

### Usage

```
peptideImport(inputFile)
```

### Arguments

| | |
|---|---|
| inputFile | character |

## Value

dataframe

## Author(s)

Petra Palenikova <pp451@cam.ac.uk>

Rick Scavetta <office@scavetta.academy>

## Examples

```
##Use example peptide data set, read in data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
```

---

# pickPeptide *Select Top Peptide For Various Scenarios*

---

## Description

This function selects a single unique peptide to represent each 'Protein Group Accession' There are 2 ways of selecting peptides, both are perform as they are needed for different tasks later on.

1. Scenario A: select peptide occuring in most fractions, do this individually for labelled/unlabelled (max value for any peptide is equal to number of fractions) in case of ties, pick peptide whith highest 'Precursor Area' in any fraction.
2. Scenario B: select peptide occuring in most fractions counting both label states together (max value for any peptide is equal to twice the number of fractions) in case of ties, pick peptide with highest 'Precursor Area' in any fraction. Representative peptide in Scenario B is picked only for proteins that have shared peptide between label states.

## Usage

```
pickPeptide(.data)
```

## Arguments

| | |
|---|---|
| .data | a dataframe |

## Value

list of data frames

## Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
## separate chemical modifications and labelling into separate columns
peptides <- splitModLab(peptides)
## remove unneccessary columns, simplify rows
peptides <- simplifyProteins(peptides)
## Pick representative peptide for each protein for both scenarios
peptide_index <- pickPeptide(peptides)
```

proteinPlot *Create Line Plots*

## Description

This function creates a line plot for a proteins in dataFrame specified by protein

## Usage

```
proteinPlot(
  dataFrame,
  protein,
  max_frac,
  grid = TRUE,
  titleLabel = "all",
  titleAlign = "left",
  ylabel = "Relative Protein Abundance",
  xlabel = "Fraction",
  legendLabel = "Condition",
  labelled = "Labeled",
  unlabelled = "Unlabeled",
  controlSample = "",
  textSize = 12,
  axisTextSize = 8
)
```

## Arguments

| | |
|---|---|
| dataFrame | data frame, contains columns: 'Protein Group Accessions' character; 'Protein Descriptions' character;bFraction integer; isLabel character ("TRUE"/"FALSE" values);'Precursor Area' double; scenario character |
| protein | character the protein of interest |
| max_frac | integer total number of fractions |
| grid | logical specifies presence/absence of gridline in the plot |
| titleLabel | character, if it is 'all' or 'GN', it specifies whether to show full label or just the gene name, if any other character is used, the value of titleLabel will be used as plot title |
| titleAlign | character one of the 'left', 'center'/'centre', 'right', specifies alignment of the title in plot |
| ylabel | character |
| xlabel | character |
| legendLabel | character |
| labelled | character label to be used for isLabel == TRUE |
| unlabelled | character label to be used for isLabel == FALSE |
| controlSample | character, either labelled or unlabelled, this setting will adjust plot coloring based on which sample is a control |
| textSize | numeric, size of text in the plot |
| axisTextSize | numeric, size of axis labels in the plot |

## Value

a plot

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
##example plot:
protein <- "P52815"
max_frac <- 23
proteinPlot(forAnalysis, protein, max_frac)
```

---

`protImportForAnalysis` *Modify import protein data*

---

## Description

This function converts imported protien table into format compatible with downstream analysis Imported file needs to contain following columns:

- "Protein Group Accessions" - character/factor
- "Protein Descriptions" - character
- "scenario" - character/factor
- "label" - logical
- columns "1" to "n" - numeric

## Usage

```
protImportForAnalysis(inputFile)
```

## Arguments

| | |
|---|---|
| `inputFile` | - character vector containing the location of protein file |

## Value

data frame

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package = "ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
```

## simplifyProteins *Simplify Raw Proteins file*

### Description

- For rows: Keep only one row with highest Precursor Area in cases where for a single Protein Group Accession in a single fraction there are multiple rows with the same combination of Sequence, Mods and Charge
- For cols: remove columns that are not neccessary any more

### Usage

```
simplifyProteins(.data, direction = c("rows", "cols"))
```

### Arguments

| | |
|---|---|
| .data | a dataframe |
| direction | character, rows, cols or both |

### Value

a dataframe

### Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
## separate chemical modifications and labelling into separate columns
peptides <- splitModLab(peptides)
## remove unneccessary columns, simplify rows
peptides <- simplifyProteins(peptides)
```

## splitModLab *Split Modification and Label tags*

### Description

Splits up the Modifications column into lists of vectors for modifications(Mods) and labels(Labels) It adds two more columns to the data frame:

- UniqueCombinedID_A: Unique combinations of Sequence, Mods and Charge for "scenario A".
- UniqueCombinedID_B: Unique combinations of Sequence, Mods, Charge and Labels for "scenario B"

### Usage

```
splitModLab(.data)
```

### Arguments

| | |
|---|---|
| .data | dataframe |

### Value

dataframe

### Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
## separate chemical modifications and labelling into separate columns
peptides <- splitModLab(peptides)
```

---

## toFilter *Optional Filtering For Raw Data*

---

### Description

Filters only rows with specified values in columns Rank and Confidence Level , specified as cl

### Usage

```
toFilter(.data, rank = 1, cl = c("Low", "Middle", "High"))
```

### Arguments

| | |
|---|---|
| .data | dataframe |
| rank | integer |
| cl | charater any combination of one or more of 'Low', 'Middle', or 'High' |

### Value

a dataframe

### Examples

```
##Use example peptide data set, read in and clean data
inputFile <- system.file("extData", "data.txt", package = "ComPrAn")
peptides <- peptideImport(inputFile)
peptides <- cleanData(peptides, fCol = "Search ID")
##optional filtering based on rank and confidence level
peptides <- toFilter(peptides, rank = 1)
```

`twoGroupsWithinLabelCoMigration` *Compare a Two Groups of Proteins Within One Label State*

## Description

This function creates a scatter plot for a subset of proteins in dataFrame specified in group1Data and group2Data, label states are always separated into facets

## Usage

```
twoGroupsWithinLabelCoMigration(
  dataFrame,
  max_frac,
  group1Data = NULL,
  group1Name = "group1",
  group2Data = NULL,
  group2Name = "group2",
  meanLine = FALSE,
  medianLine = FALSE,
  ylabel = "Relative Protein Abundance",
  xlabel = "Fraction",
  legendLabel = "Group",
  labelled = "Labeled",
  unlabelled = "Unlabeled",
  jitterPoints = 0.3,
  pointSize = 2.5,
  grid = FALSE,
  showTitle = FALSE,
  titleAlign = "left",
  alphaValue = 1,
  textSize = 12,
  axisTextSize = 8
)
```

## Arguments

| | |
|---|---|
| `dataFrame` | dataFrame: data frame, data frame of normalised values for proteins from SCENARIO A, contains columns: 'Protein Group Accessions' character 'Protein Descriptions' character Fraction integer isLabel character ('TRUE'/'FALSE' values) 'Precursor Area' double scenario character |
| `max_frac` | numeric, total number of fractions |
| `group1Data` | character vector, contins list of Protein Group Accessions that belong to the group we want to plot for group 1 |
| `group1Name` | character, name that should be used for the group specified in group1Data |
| `group2Data` | character vector, contins list of Protein Group Accessions that belong to the group we want to plot for group 2 |
| `group2Name` | character, name that should be used for the group specified in group2Data |
| `meanLine` | logical, specifies whether to plot a mean line for all values in the group |

| | |
|---|---|
| medianLine | logical, specifies whether to plot a median line for all values in the group |
| ylabel | character |
| xlabel | character |
| legendLabel | character |
| labelled | character, label to be used for isLabel == TRUE |
| unlabelled | character, label to be used for isLabel == FALSE |
| jitterPoints | numeric |
| pointSize | numeric, size of the point in the plot |
| grid | logical, specifies presence/absence of gridline in the plot |
| showTitle | logical |
| titleAlign | character, one of the 'left', 'center'/'centre', 'right', specifies alignment of the title in plot |
| alphaValue | numeric, transparency of the point, values 0 to 1 |
| textSize | numeric, size of text in the plot |
| axisTextSize | numeric, size of axis labels in the plot |

## Details

Intended use of the function - using scenario A data, compare shape of the migration profile between a TWO GROUPS of proteins WITHIN the ONE LABEL STATE

## Value

plot

## Examples

```
##Use example normalised proteins file
inputFile <- system.file("extData", "dataNormProts.txt", package ="ComPrAn")
#read file in and change structure of table to required format
forAnalysis <- protImportForAnalysis(inputFile)
##example plot:
g1D <- c("Q16540","P52815","P09001","Q13405","Q9H2W6") #group 1 data vector
g1N <- 'group1'                                        #group 1 name
g2D <- c("Q9NVS2","Q9NWU5","Q9NX20","Q9NYK5","Q9NZE8") #group 2 data vector
g2N <- 'group2'                                        #group 2 name
max_frac <- 23
twoGroupsWithinLabelCoMigration(forAnalysis, max_frac, g1D, g1N, g2D, g2N)
```

---

# uncenteredCor *Perform uncentered correlation*

---

## Description

Perform uncentered correlation

## Usage

```
uncenteredCor(xx, yy)
```

## Arguments

| | |
|---|---|
| xx | numeric vector |
| yy | numeric vector |

## Value

vector

# Index