Package ‘flowTraceR’

June 3, 2022

**Title**  Tracing Information Flow for Inter-Software Comparisons in Mass Spectrometry-Based Bottom-Up Proteomics

**Version**  0.1.0

**Description**  Useful functions to standardize software outputs from ProteomeDiscoverer, Spectronaut, DIA-NN and MaxQuant on precursor, modified peptide and protein-group level and to trace software differences for identifications such as varying protein-group denotations for common precursor.

**License**  MIT + file LICENSE

**Depends**  R (>= 2.10)

**Imports**  comprphrenr, dplyr, ggplot2, magrittr, stringr, tibble, tidyr

**Suggests**  data.table, kableExtra, knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder**  knitr

**Config/testthat/edition**  3

**Encoding**  UTF-8

**URL**  https://github.com/OKdll/flowTraceR

**RoxygenNote**  7.1.2

**NeedsCompilation**  no

**Author**  Oliver Kardell [aut, cre]

**Maintainer**  Oliver Kardell <Okdll@gmx.net>

**Repository**  CRAN

**Date/Publication**  2022-06-03 08:00:07 UTC

**R topics documented:**

- analyze_connected_levels ........................................... 2
- analyze_unknown_mods .............................................. 3
- connect_traceR_levels ............................................. 5
- convert_all_levels .................................................. 7
- convert_modified_peptides ........................................ 8
- convert_precursor ................................................... 9
analyze_connected_levels

Analysis of connected levels

Description

Analysis of the traceR_connected_pg_prec or traceR_connected_mod.pep_prec column

Usage

analyze_connected_levels(
  input_df,
  connected_levels = c("proteinGroup_precursor", "mod.peptides_precursor"),
  count_level = c("upper", "lower"),
  plot = TRUE,
  plot_characteristic = c("absolute", "relative")
)

Arguments

input_df A tibble with flowTraceR’s connected level information e.g. traceR_connected_pg_prec.
connected_levels Choose either proteinGroup_precursor or mod.peptides_precursor for the corresponding traceR connection. Default is proteinGroup_precursor.
count_level Counts appearances per possible connections. Choose either upper or lower - lower is always precursor level; upper is either proteingroup or mod.peptide level depending on chosen connected_levels. Default is upper. Duplicate entries are removed.
plot Logical value, default is TRUE. If TRUE barplot is generated, if FALSE report as output.
plot_characteristic if absolute the absolute count is displayed in barplot, if relative the relative count is displayed in barplot. Default is absolute. plot_characteristic has no influence on report.
analyze_unknown_mods

Details

Shows the absolute and relative counts of possible connections - unique_unique/unique_common/common_unique/common_common - of the respective column - as report or plot.

Value

This function returns a plot - absolute/relative counts - or a data frame.

Author(s)

Oliver Kardell

Examples

```r
# Load libraries
library(dplyr)
library(stringr)
library(ggplot2)
library(tibble)

# DIA-NN example data
data <- tibble::tibble(
  "traceR_connected_pg_prec" = c("common_common", "common_unique", "unique_common"),
  "traceR_traced_proteinGroups" = c("common", "common", "unique"),
  "traceR_traced_mod.peptides" = c("common", "unique", "common"),
  "traceR_traced_precursor" = c("common", "unique", "common"),
  "traceR_proteinGroups" = c("P02768", "P02671", "Q92496"),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "RLEVDIDIK2", "EGIVEYPR2")
)

# Upper level - proteingroup level - how many proteingroups have a specific categorization
analyze_connected_levels(input_df = data,
                         connected_levels = "proteinGroup_precursor",
                         count_level = "upper",
                         plot = TRUE,
                         plot_characteristic = "relative")

# Report
analyze_connected_levels(input_df = data,
                         connected_levels = "proteinGroup_precursor",
                         count_level = "upper",
                         plot = FALSE)
```

Description

Analysis of the traceR_precursor_unknownMods or traceR_mod.peptides_unknownMods column
analyze_unknown_mods

Usage

analyze_unknown_mods(
  input_df,
  level = c("precursor", "modified_peptides"),
  plot = TRUE,
  plot_characteristic = c("absolute", "relative")
)

Arguments

- **input_df**: A tibble with the `traceR_precursor_unknownMods` or `traceR_mod.peptides_unknownMods` column.
- **level**: Choose either `precursor` for `traceR_precursor_unknownMods` or `modified_peptides` for `traceR_mod.peptides_unknownMods`. Default is `precursor`.
- **plot**: Logical value, default is `TRUE`. If `TRUE` barplot is generated, if `FALSE` report as output.
- **plot_characteristic**: If `absolute` the absolute count is displayed in barplot, if `relative` the relative count is displayed in barplot. Default is `absolute`. `plot_characteristic` has no influence on report.

Details

Shows the absolute and relative counts of TRUE/FALSE of the `traceR_precursor_unknownMods` or `traceR_mod.peptides_unknownMods` column - as data frame or plot. Duplicate `traceR_mod.peptides` entries or `traceR_precursor` entries are removed, respectively.

Value

This function returns a plot - absolute/relative counts - or a data frame.

Author(s)

Oliver Kardell

Examples

```r
# Load libraries
library(dplyr)
library(stringr)
library(ggplot2)
library(tibble)

# Generate data
data <- tibble::tibble(
  "traceR_mod.peptides" = c("AACLLPK",
    "ALTDM(UniMod:35)PQM(UniMod:35)R",
    "ALTDM(DummyModification)PQMK",
    "ALTDM(UniMod:35)PQM(UniMod:35)R",
    "ALTDM(DummyModification)PQMK"
)
connect_traceR_levels

`connect_traceR_levels` connects traced levels after categorization in unique and common entries.

**Description**
Connects two levels after categorization in unique and common entries.

**Usage**
```r
connect_traceR_levels(
  input_df, 
  level = c("proteinGroups", "modified_peptides")
)
```

**Arguments**
- `input_df`: A tibble with flowTraceR's traced level information e.g. `traceR_traced_proteinGroups`.
- `level`: Choose between `proteinGroups` or `modified_peptides`. Connection between `proteinGroups/modified_peptides` and precursor categorization. Default is `proteinGroups`.

**Details**
Based on flowTraceR's categorization in unique and common identifications, two levels are connected. Possible connections are `proteinGroup` or modified peptide with precursor categorization.
Value

This function returns a tibble with one of the following columns depending on chosen level:

- `traceR_connected_pg_prec` - connection between proteinGroup categorization and precursor categorization.
- `traceR_connected_mod.pep_prec` - connection between modified peptide categorization and precursor categorization.

Author(s)

Oliver Kardell

Examples

```r
# Load libraries
library(tidyr)
library(stringr)
library(tibble)

# DIA-NN example data
diann <- tibble::tibble(
  "traceR_traced_proteinGroups" = c("common", "common", "unique"),
  "traceR_traced_mod.peptides" = c("common", "unique", "common"),
  "traceR_traced_precursor" = c("common", "unique", "common"),
  "traceR_proteinGroups" = c("P02768", "P02671", "Q92496"),
  "traceR_mod.peptides" = c("AAC(UniMod:4)LLPK", "RLEVDIDIK", "EGIVEYPR"),
  "traceR_mod.peptides_unknownMods" = c(FALSE, FALSE, FALSE),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "RLEVDIDIK2", "EGIVEYPR2"),
  "traceR_precursor_unknownMods" = c(FALSE, FALSE, FALSE)
)
spectronaut <- tibble::tibble(
  "traceR_traced_proteinGroups" = c("common", "common", "unique"),
  "traceR_traced_mod.peptides" = c("common", "unique", "common"),
  "traceR_traced_precursor" = c("common", "unique", "common"),
  "traceR_proteinGroups" = c("P02768", "P02671", "Q02985"),
  "traceR_mod.peptides" = c("AAC(UniMod:4)LLPK", "M(UniMod:35)KPVPDLVPGNFK", "EGIVEYPR"),
  "traceR_mod.peptides_unknownMods" = c(FALSE, FALSE, FALSE),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "M(UniMod:35)KPVPDLVPGNFK2", "EGIVEYPR2"),
  "traceR_precursor_unknownMods" = c(FALSE, FALSE, FALSE)
)

# Connect Precursor and ProteinGroup level
diann_connected <- connect_traceR_levels(input_df = diann, level = "proteinGroups")
spectronaut_connected <- connect_traceR_levels(input_df = spectronaut, level = "proteinGroups")
```
**convert_all_levels**

*Conversion of software specific levels*

**Description**
Conversion of precursor, modified peptide and proteinGroup entries to standardized format.

**Usage**

```r
closest_levels(input_df, input_MQ_pg, software = c("MaxQuant", "DIA-NN", "Spectronaut", "PD"))
```

**Arguments**

- **input_df**: A tibble with precursor, modified peptide and proteinGroup level information. For MaxQuant: evidence.txt and proteinGroups.txt, for PD: PSMs.txt with R-friendly headers enabled, for DIA-NN and Spectronaut default output reports.
- **input_MQ_pg**: For MaxQuant: A tibble with proteinGroup level information - proteinGroups.txt.
- **software**: The used analysis software - MaxQuant, PD, DIA-NN or Spectronaut. Default is MaxQuant.

**Details**
The input entries are converted to a software independent format. The generated entries are appended to the submitted dataframe.

**Value**
This function returns the original submitted tibble - `input_df` - including the following new columns:

- `traceR_precursor`: software-independent standardized text for precursor entries.
- `traceR_precursor_unknownMods`: logical value, if TRUE: a modification is detected, which is not converted to a standardized format.
- `traceR_mod.peptides`: software-independent standardized text for modified peptide entries.
- `traceR_mod.peptides_unknownMods`: logical value, if TRUE: a modification is detected, which is not converted to a standardized format.
- `traceR_proteinGroups`: software-independent standardized text for proteinGroups.

**Author(s)**
Oliver Kardell
convert_modified_peptides

Conversion of software specific modified peptide entries

Description

Modified peptide entries are converted to a common text representation

Usage

```r
convert_modified_peptides(
  input_df,
  software = c("MaxQuant", "PD", "DIA-NN", "Spectronaut")
)
```

Arguments

- **input_df**: A tibble with modified peptide level information. For MaxQuant: evidence.txt, for PD: PSMs.txt with R-friendly headers enabled, for DIA-NN and Spectronaut default output reports.
**convert_precursor**

The used analysis software for the input_df - MaxQuant, PD, DIA-NN or Spectrum. Default is MaxQuant.

**Details**

The input entries are converted to a software independent format. The generated entries are appended to the submitted dataframe. Conversion of modifications is currently only available for UniMod:35 and UniMod:4. Other modifications will not be converted to standardized format.

**Value**

This function returns the original submitted tibble - input_df - including two new columns:

- traceR_mod.peptides - software-independent standardized text for modified peptide entries.
- traceR_mod.peptides_unknownMods - logical value, if TRUE: a modification is detected, which is not converted to a standardized text.

**Author(s)**

Oliver Kardell

**Examples**

```r
# Load libraries
library(dplyr)
library(stringr)
library(tidyr)
library(tibble)

# MaxQuant example data
data <- tibble::tibble(
  "Modified sequence" = c("_AACLLPK_",
  "_ALTDM(Oxidation (M))PQM(Oxidation (M))R_",
  "ALTDM(Dummy_Modification)PQMK"),
  Charge = c(2,2,3)
)

# Conversion
collate_modified_peptides(
  input_df = data,
  software = "MaxQuant"
)
```

---

**convert_precursor**  
*Conversion of software specific precursor entries*

**Description**

Precursor entries are converted to a common text representation
convert_precursor

Usage

convert_precursor(
  input_df,
  software = c("MaxQuant", "PD", "DIA-NN", "Spectronaut")
)

Arguments

input_df       A tibble with precursor level information. For MaxQuant: evidence.txt, for PD: PSMs.txt with R-friendly headers enabled, for DIA-NN and Spectronaut default output reports.
software       The used analysis software for the input_df - MaxQuant, PD, DIA-NN or Spectronaut. Default is MaxQuant.

Details

The input entries are converted to a software independent format. The generated entries are appended to the submitted dataframe. Conversion of modifications is currently only available for UniMod:35 and UniMod:4. Other modifications will not be converted to standardized format.

Value

This function returns the original submitted tibble - input_df - including two new columns:

- traceR_precursor - software-independent standardized text for precursor entries.
- traceR_precursor_unknownMods - logical value, if TRUE: a modification is detected, which is not converted to a standardized text.

Author(s)

Oliver Kardell

Examples

# Load libraries
library(dplyr)
library(stringr)
library(tidyr)
library(tibble)

# MaxQuant example data
data <- tibble::tibble(
  "Modified sequence" = c("_AACLLPK_",
  "_ALTDM(Oxidation (M))PQM(Oxidation (M))R_",
  "ALTDM(Dummy_Modification)PQMK"),
  Charge = c(2,2,3)
)

# Conversion
convert_precursor(
convert_proteingroups

```
  input_df = data,
  software = "MaxQuant"
)
```

---

**convert_proteingroups**  
*Conversion of software specific proteinGroups*

---

**Description**

ProteinGroups are converted to a common text representation

**Usage**

```r
convert_proteingroups(
  input_df,
  software = c("MaxQuant", "DIA-NN", "Spectronaut", "PD")
)
```

**Arguments**

- **input_df**: A tibble with proteinGroup level information. For MaxQuant: proteinGroups.txt, for PD: PSMs.txt with R-friendly headers enabled, for DIA-NN and Spectronaut default output reports.
- **software**: The used analysis software for the input_df - MaxQuant, PD, DIA-NN or Spectronaut. Default is MaxQuant.

**Details**

The input entries are converted to a software independent format. The generated entries are appended to the submitted dataframe.

**Value**

This function returns the original submitted tibble - input_df - including one new column:

- `traceR_proteinGroups` - software-independent standardized text for proteinGroups.

**Author(s)**

Oliver Kardell
Examples

```r
# Load libraries
library(dplyr)
library(stringr)
library(comprehenr)
library(tibble)

# MaxQuant example data
data <- tibble::tibble(
  "Protein IDs" = c("A0A075B6P5;P01615;A0A087WW87;P01614;A0A075B6S6", "P02671", "P02672"),
  id = c(26, 86, 17)
)

# Conversion
custom_proteingroups(
  input_df = data,
  software = "MaxQuant"
)
```

---

flowTraceR: a package for standardization of level information and tracking inter-software differences in bottom-up label-free proteomics

Description

Useful functions to standardize software outputs from ProteomeDiscoverer, Spectronaut, DIA-NN and MaxQuant on precursor, modified peptide and proteingroup level and to trace software differences for identifications such as varying proteingroup denotations for common precursor.

Author(s)

Maintainer: Oliver Kardell <Okdll@gmx.net>

See Also

Useful links:

- https://github.com/OKdll/flowTraceR
get_example

Create example data

Description

Example data for ProteomeDiscoverer, Spectronaut, DIA-NN and MaxQuant.

Usage

get_example(
  example = c("MaxQuant", "DIA-NN", "Spectronaut", "PD", "RetentionTime")
)

Arguments

example Choose between "ProteomeDiscoverer", "Spectronaut", "DIA-NN" and "MaxQuant" or for an example for downstream analysis "RetentionTime". Default is MaxQuant.

Details

Data for each software for testing functions of flowTraceR. Additional example data for Spectronaut and DIA-NN for analyzing retention time distribution on precursor level.

Value

This function returns example data as dataframe for the respective chosen example. For "MaxQuant" a list with evidence/proteingroup dataframe. For "RetentionTime" a list with Spectronaut/DIA-NN data including retention time information.

Author(s)

Oliver Kardell

Examples

# Spectronaut example data
Spectronaut_data <- get_example(example = "Spectronaut")
get_unknown.mods  

Check of converted modifications

Description
Check if conversion to UniMod-format of identified modifications is successful.

Usage
get_unknown.mods(input_string, pattern_start, pattern_end)

Arguments
input_string character column traceR_precursor as string.
pattern_start character of software-dependent beginning of representation of modifications.
pattern_end character of software-dependent end of representation of modifications.

Details
After conversion to standardized format by convert_precursor or convert_modified_peptides, entries with modifications are checked for a successful conversion. Conversion of modifications is currently only available for UniMod:35 and UniMod:4. Other modifications will not be converted to standardized format.

Value
This function returns vector with logical values. This function is incorporated in the functions convert_precursor and convert_modified_peptides; used to generate the unknownMods column: if TRUE: a modification is detected, which is not converted to a standardized text.

Author(s)
Oliver Kardell

Examples
# Load libraries
library(dplyr)
library(stringr)
library(tibble)

# Generate data
data <- tibble::tibble(
  "traceR_precursor" = c("AACLLPK",
  "ALTDM(UniMod:35)PQM(UniMod:35)R2",
  "ALTDM(DummyModification)PQMK3")
)
trace_all_levels

Get unknown modifications present?
get_unknown_mods(input_string = data$traceR_precursor, pattern_start="(", pattern_end="")

Description

Identifications of two input data frames are compared and categorized in unique and common entries for each level.

Usage

trace_all_levels(
  input_df1,
  input_df2,
  analysis_name1 = "input_df1",
  analysis_name2 = "input_df2",
  filter_unknown_mods = TRUE
)

Arguments

input_df1 A tibble with flowTraceR’s standardized precursor, modified peptide and proteinGroup level information.
input_df2 A tibble with flowTraceR’s standardized precursor, modified peptide and proteinGroup level information.
analysis_name1 output tibble name for input_df1 - default is "input_df1".
analysis_name2 output tibble name for input_df2 - default is "input_df2".
filter_unknown_mods Logical value, default is TRUE. If TRUE, unknown modifications are filtered out - requires "traceR_precursor_unknownMods" or "traceR_mod.peptides_unknownMods" column.

Details

Based on flowTraceR’s standardized output format two software outputs can be compared and categorized into common and unique identifications - for precursor, modified peptide and proteinGroup level.
Value

This function returns a list with both original submitted tibbles - input_df1 and input_df2 - with the following new columns:

- `traceR_traced_precursor` - categorization on precursor level in common and unique entries.
- `traceR_traced_mod.peptides` - categorization on modified peptide level in common and unique entries.
- `traceR_traced_proteinGroups` - categorization on proteinGroups level in common and unique entries.

Author(s)

Oliver Kardell

Examples

```r
# Load libraries
library(dplyr)
library(stringr)
library(tibble)

# DIA-NN example data
diann <- tibble::tibble(
  "traceR_proteinGroups" = c("P02768", "P02671", "Q92496", "DummyProt"),
  "traceR_mod.peptides" = c("AAC(UniMod:4)LLPK", "RLEVDIDIK", 
                            "EGIVEYPR", "ALTDM(DummyModification)PQMK"),
  "traceR_mod.peptides_unknownMods" = c(FALSE, FALSE, FALSE, TRUE),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "RLEVDIDIK2", 
                         "EGIVEYPR2", "ALTDM(DummyModification)PQMK3" ),
  "traceR_precursor_unknownMods" = c(FALSE, FALSE, FALSE, TRUE)
)

# Spectronaut example data
spectronaut <- tibble::tibble(
  "traceR_proteinGroups" = c("P02768", "Q02985", "P02671"),
  "traceR_mod.peptides" = c("AAC(UniMod:4)LLPK", "EGIVEYPR", "M(UniMod:35)KPVPDLVPGNFK"),
  "traceR_mod.peptides_unknownMods" = c(FALSE, FALSE, FALSE),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "EGIVEYPR2", "M(UniMod:35)KPVPDLVPGNFK2"),
  "traceR_precursor_unknownMods" = c(FALSE, FALSE, FALSE)
)

# trace all levels in one step
traced_all <- trace_all_levels(
  input_df1 = diann,
  input_df2 = spectronaut,
  analysis_name1 = "DIA-NN",
  analysis_name2 = "Spectronaut",
  filter_unknown_mods = TRUE
)
```
trace_level

Trace common and unique identifications between different software outputs

Description

Identifications of two input data frames are compared and categorized in unique and common entries.

Usage

```r
trace_level(
  input_df1,  
  input_df2,  
  analysis_name1 = "input_df1",  
  analysis_name2 = "input_df2",  
  level = c("precursor", "modified_peptides", "proteinGroups"),  
  filter_unknown_mods = TRUE
)
```

Arguments

- `input_df1`: A tibble with flowTraceR’s standardized precursor, modified peptide, or proteinGroup level information - required column depends on chosen `level`.
- `input_df2`: A tibble with flowTraceR’s standardized precursor, modified peptide, or proteinGroup level information - required column depends on chosen `level`.
- `analysis_name1`: output tibble name for `input_df1` - default is "input_df1".
- `analysis_name2`: output tibble name for `input_df2` - default is "input_df2".
- `level`: "precursor", "modified_peptides", "proteinGroups" - respective level for tracing common vs. unique entries. Default is precursor.
- `filter_unknown_mods`: Logical value, default is TRUE. If TRUE, unknown modifications are filtered out - requires "traceR_precursor_unknownMods" or "traceR_mod.peptides_unknownMods" column; depends on chosen `level`.

Details

Based on flowTraceR’s standardized output format two software outputs can be compared and categorized into common and unique identifications for a chosen level: precursor, modified peptide or proteinGroup level.

Value

This function returns a list with both original submitted tibbles - `input_df1` and `input_df2` - including one of the following new columns depending on chosen `level`:
• traceR_traced_precursor - categorization on precursor level in common and unique entries.
• traceR_traced_mod.peptides - categorization on modified peptide level in common and unique entries.
• traceR_traced_proteinGroups - categorization on proteinGroups level in common and unique entries.

Author(s)
Oliver Kardell

Examples

# Load libraries
library(dplyr)
library(stringr)
library(tibble)

# DIA-NN example data
diann <- tibble::tibble(
  "traceR_proteinGroups" = c("P02768", "P02671", "Q02946", "DummyProt"),
  "traceR_mod.peptides" = c("AAC(UniMod:4)LLPK", "RLEVDI",
    "EGIVEYPR", "ALTDM(DummyModification)PQMK"),
  "traceR_mod.peptides_unknownMods" = c(FALSE, FALSE, FALSE, TRUE),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "RLEVDI2",
    "EGIVEYPR2", "ALTDM(DummyModification)PQMK3" ),
  "traceR_precursor_unknownMods" = c(FALSE, FALSE, FALSE, TRUE)
)

# Spectronaut example data
spectronaut <- tibble::tibble(
  "traceR_proteinGroups" = c("P02768", "Q02985", "P02671"),
  "traceR_mod.peptides" = c("AAC(UniMod:4)LLPK", "EGIVEYPR", "M(UniMod:35)KPVPDLVPGNFK"),
  "traceR_mod.peptides_unknownMods" = c(FALSE, FALSE, FALSE),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "EGIVEYPR2", "M(UniMod:35)KPVPDLVPGNFK2"),
  "traceR_precursor_unknownMods" = c(FALSE, FALSE, FALSE)
)

# trace proteinGroup level
traced_proteinGroups <- trace_level(
  input_df1 = diann,
  input_df2 = spectronaut,
  analysis_name1 = "DIA-NN",
  analysis_name2 = "Spectronaut",
  level = "proteinGroups",
  filter_unknown_mods = TRUE
)

# trace precursor level
traced_precursor <- trace_level(
  input_df1 = diann,
  input_df2 = spectronaut,
  analysis_name1 = "DIA-NN",
  analysis_name2 = "Spectronaut",
  level = "precursor",
  filter_unknown_mods = TRUE
)
trace_unique_common_pg

trace_unique_common_pg(
  input_df1, 
  input_df2, 
  analysis_name1 = "input_df1", 
  analysis_name2 = "input_df2", 
  string_analysis = FALSE
)

Arguments

input_df1 A tibble with flowTraceR’s unique_common categorization for the proteinGroup_precursor connection.

input_df2 A tibble which is the counter part for input_df1 - which was used to generate the unique_common categorization for the proteinGroup_precursor connection.

analysis_name1 String. Appended to input_df1’s traceR_proteinGroups column - default is “input_df1”.

analysis_name2 String. Appended to input_df1’s traceR_proteinGroups column - default is “input_df2”.

string_analysis Logical value, default is FALSE. If TRUE, only keeps proteinGroup identifications of input_df1 in which protein denotations are not present in the counterpart - the proteinGroups of input_df2 - and vice versa.

Details

For each submitted dataframe the unique_common proteinGroup_precursor connection is analyzed to highlight potential differences in proteinGroup denotations for common precursors.
Value

This function returns a tibble with the following columns:

- `traceR_proteinGroups_input_df1` - proteinGroup denotations of input_df1 for common precursor between input_df1 and input_df2
- `traceR_precursor` - common precursor between input_df1 and input_df2
- `traceR_proteinGroups_input_df2` - proteinGroup denotations of input_df2 for common precursor between input_df1 and input_df2

Author(s)

Oliver Kardell

Examples

```r
# Load libraries
library(dplyr)
library(stringr)
library(tibble)

# DIA-NN example data
diann <- tibble::tibble(
  "traceR_connected_pg_prec" = c("common_common", "common_unique", "unique_common", "unique_common"),
  "traceR_proteinGroups" = c("P02768", "P02671", "Q02985", "P04433"),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "RLEVDIDIK2", "EGIVEYPR2", "ASQSVSSYAWYQQK2"),
)

# Spectronaut example data
spectronaut <- tibble::tibble(
  "traceR_connected_pg_prec" = c("common_common", "common_unique", "unique_common", "unique_common"),
  "traceR_proteinGroups" = c("P02768", "P02671", "Q02985", "A0A0A0MRZ8;P04433"),
  "traceR_precursor" = c("AAC(UniMod:4)LLPK1", "M(UniMod:35)KPVPDLVPQNK2", "EGIVEYPR2", "ASQSVSSYAWYQQK2"),
)

# Find difference in pg denotation
# string_analysis = TRUE
resultA <- trace_unique_common_pg(input_df1 = diann,
  input_df2 = spectronaut,
  analysis_name1 = "DIA-NN",
  analysis_name2 = "Spectronaut",
  string_analysis = TRUE)

# Find difference in pg denotation
# string_analysis = FALSE
# compare with resultA
resultB <- trace_unique_common_pg(input_df1 = diann,
  input_df2 = spectronaut,
  string_analysis = FALSE)
```
analysis_name1 = "DIA-NN",
analysis_name2 = "Spectronaut",
string_analysis = FALSE)
Index

analyze_connected_levels, 2
analyze_unknown.mods, 3
connect_traceR.levels, 5
convert_all_levels, 7
convert_modified_peptides, 8
convert_precursor, 9
convert_proteingroups, 11

flowTraceR, 12
flowTraceR-package (flowTraceR), 12

get_example, 13
get_unknown.mods, 14

trace_all_levels, 15
trace_level, 17
trace_unique_common_pg, 19