

Package ‘mpwR’

June 22, 2022

Title Standardized Comparison of Workflows in Mass Spectrometry-Based Bottom-Up Proteomics

Version 0.1.0

Description Useful functions to analyze proteomic workflows including number of identifications, data completeness, missed cleavages, quantitative and retention time precision etc. Various software outputs are supported such as 'ProteomeDiscoverer', 'Spectronaut', 'DIA-NN' and 'MaxQuant'.

License MIT + file LICENSE

Imports comprehenr, data.table, dplyr, flowTraceR, forcats, ggplot2, magrittr, plotly, purrr, stats, stringr, tibble, tidyr, UpSetR

Suggests flextable, knitr, rmarkdown, testthat (>= 3.0.0), utils

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Author Oliver Kardell [aut, cre]

Maintainer Oliver Kardell <okd11@gmx.net>

Repository CRAN

Date/Publication 2022-06-22 07:30:02 UTC

R topics documented:

create_example	2
get_CV_LFQ_pep	3
get_CV_LFQ_pg	4
get_CV_RT	5
get_DC_Report	6
get_ID_Report	8
get_MC_Report	9

get_summary_Report	10
get_Upset_list	13
plot_CV_density	14
plot_DC_barplot	16
plot_DC_stacked_barplot	17
plot_ID_barplot	19
plot_ID_boxplot	20
plot_MC_barplot	21
plot_MC_stacked_barplot	23
plot_radarchart	24
plot_Upset	25
prepare_mpwR	26

Index	28
--------------	-----------

create_example	<i>Create example data</i>
----------------	----------------------------

Description

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

Usage

```
create_example()
```

Details

Example data is generated for each software for testing functions of mpwR. Each column is created in a randomized fashion. Connections between columns are not necessarily valid. E.g. column of Precursor Charges might not reflect charges of Precursor.IDs column.

Value

This function returns list with example data. Each list entry has filename and software information as well as a corresponding data set.

Author(s)

Oliver Kardell

Examples

```
data <- create_example()
```

get_CV_LFQ_pep	<i>Peptide-level: Quantitative precision</i>
----------------	--

Description

Calculate quantitative precision on peptide-level

Usage

```
get_CV_LFQ_pep(input_list)
```

Arguments

`input_list` A list with data frames and respective quantitative peptide information.

Details

For each submitted data the coefficient of variation is calculated on peptide-level for LFQ intensities. Only full profiles are included.

Value

This function returns the original submitted data of the `input_list` including a new output column:

- `CV_Peptide_LFQ_mpwR` - coefficient of variation in percentage.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(stringr)
library(magrittr)
library(tibble)

# Example data
set.seed(123)
data <- list(
  Spectronaut = list(
    filename = "C",
    software = "Spectronaut",
    data = list(
      "Spectronaut" = tibble::tibble(
        Run_mpwR = rep(c("A","B"), times = 5),
        Precursor.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 2),
        Stripped.Sequence_mpwR = rep(c("A", "B", "C", "D", "E"), each = 2),
        Peptide.IDs_mpwR = rep(c("A", "B", "C", "D", "E"), each = 2),
        ProteinGroup.IDs_mpwR = rep(c("A", "B", "C", "D", "E"), each = 2),
```

```
        Retention.time_mpwR = sample(1:20, 10),
        Peptide_LFQ_mpwR = sample(1:30, 10),
        ProteinGroup_LFQ_mpwR = sample(1:30, 10))
    )
)

# Result
output <- get_CV_LFQ_pep(
  input_list = data
)
```

get_CV_LFQ_pg

Proteingroup-level: Quantitative precision

Description

Calculate quantitative precision on proteingroup-level

Usage

```
get_CV_LFQ_pg(input_list)
```

Arguments

`input_list` A list with data frames and respective quantitative proteingroup information.

Details

For each submitted data the coefficient of variation is calculated on proteingroup-level for LFQ intensities. Only full profiles are included.

Value

This function returns the original submitted data of the `input_list` including a new output column:

- `CV_ProteinGroup_LFQ_mpwR` - coefficient of variation in percentage.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(stringr)
library(magrittr)
library(tibble)

# Example data
set.seed(123)
data <- list(
  Spectronaut = list(
    filename = "C",
    software = "Spectronaut",
    data = list(
      "Spectronaut" = tibble::tibble(
        Run_mpwR = rep(c("A","B"), times = 5),
        Precursor.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 2),
        Peptide.IDs_mpwR = rep(c("A", "B", "C", "D", "E"), each = 2),
        ProteinGroup.IDs_mpwR = rep(c("A", "B", "C", "D", "E"), each = 2),
        Retention.time_mpwR = sample(1:20, 10),
        Peptide_LFQ_mpwR = sample(1:30, 10),
        ProteinGroup_LFQ_mpwR = sample(1:30, 10))
      )
    )
)

# Result
output <- get_CV_LFQ_pg(
  input_list = data
)
```

get_CV_RT

Retention time precision

Description

Calculate retention time precision

Usage

```
get_CV_RT(input_list)
```

Arguments

`input_list` A list with data frames and respective retention time information.

Details

For each submitted data the coefficient of variation is calculated on precursor-level for retention time. Only full profiles are included.

Value

This function returns the original submitted data of the `input_list` including a new output column:

- `CV_Retention.time_mpwr` - coefficient of variation in percentage.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(tibble)

# Example data
set.seed(123)
data <- list(
  Spectronaut = list(
    filename = "C",
    software = "Spectronaut",
    data = list(
      "Spectronaut" = tibble::tibble(
        Run_mpwr = rep(c("A","B"), times = 5),
        Precursor.IDs_mpwr = rep(c("A2", "A3", "B2", "B3", "C1"), each = 2),
        Peptide.IDs_mpwr = rep(c("A", "B", "C", "D", "E"), each = 2),
        ProteinGroup.IDs_mpwr = rep(c("A", "B", "C", "D", "E"), each = 2),
        Retention.time_mpwr = sample(1:20, 10),
        Peptide_LFQ_mpwr = sample(1:30, 10),
        ProteinGroup_LFQ_mpwr = sample(1:30, 10))
    )
  )
)

# Result
output <- get_DC_Report(
  input_list = data
)
```

get_DC_Report

Data Completeness Report

Description

Generates a data completeness report from precursor to proteingroup-level

Usage

```
get_DC_Report(input_list, metric = c("absolute", "percentage"))
```

Arguments

input_list	A list with data frames and respective level information.
metric	"absolute" for absolute numbers or "percentage" for displaying percentages. Default is absolute.

Details

For each submitted data a data completeness report is generated highlighting missing values on precursor-, peptide-, protein- and proteingroup-level.

Value

This function returns a list. For each analysis a respective data frame including missing value information per level is stored in the generated list.

- Analysis - analysis name.
- Nr.Missing.Values - number of missing values.
- Precursor.IDs - number of precursor identification per missing value entry - absolute or in percentage.
- Peptide.IDs - number of peptide identification per missing value entry - absolute or in percentage.
- Protein.IDs - number of protein identification per missing value entry - absolute or in percentage.
- ProteinGroup.IDs - number of proteingroup identification per missing value entry - absolute or in percentage.
- Profile - categorical entries: "unique", "sparse", "shared with at least 50%" or "complete".

Author(s)

Oliver Kardell

Examples

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

# Example data
data <- list(
  DIANN = list(
    filename = "B",
    software = "DIA-NN",
    data = list(
      "DIA-NN" = tibble::tibble(
        Run_mpwR = rep(c("A","B"), times = 10),
        Precursor.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4),
        Protein.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4),
        Peptide.IDs_mpwR = rep(c("A", "A", "B", "B", "C"), each = 4),
```

```
        ProteinGroup.IDs_mpWR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4)
      )
    )
  )

# Result
output <- get_DC_Report(
  input_list = data,
  metric = "absolute"
)
```

get_ID_Report	<i>Report for identifications</i>
---------------	-----------------------------------

Description

Generates a report for identifications

Usage

```
get_ID_Report(input_list)
```

Arguments

`input_list` A list with data frames and respective level information.

Details

For each submitted data a report with achieved number of identifications is generated on precursor-, peptide-, protein- and proteingroup-level.

Value

This function returns a list. For each analysis a respective data frame including number of identifications per run is stored in the generated list.

- Analysis - analysis name.
- Run - run information.
- Precursor.IDs - number of precursor identification.
- Peptide.IDs - number of peptide identification.
- Protein.IDs - number of protein identification.
- ProteinGroup.IDs - number of proteingroup identification.

Author(s)

Oliver Kardell

Examples

```

# Load libraries
library(tibble)
library(stringr)

# Example data
data <- list(
  DIANN = list(
    filename = "B",
    software = "DIA-NN",
    data = list(
      "DIA-NN" = tibble::tibble(
        Run_mpwR = rep(c("A","B"), times = 10),
        Precursor.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4),
        Protein.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4),
        Peptide.IDs_mpwR = rep(c("A", "A", "B", "B", "C"), each = 4),
        ProteinGroup.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4)
      )
    )
  )
)

# Result
output <- get_ID_Report(
  input_list = data
)

```

get_MC_Report

Report about Missed Cleavages

Description

Generates report with information about number of missed cleavages

Usage

```
get_MC_Report(input_list, metric = c("absolute", "percentage"))
```

Arguments

input_list	A list with data frames and respective missed cleavage information.
metric	"absolute" for absolute numbers or "percentage" for displaying percentages. Default is absolute.

Details

For each submitted data a report is generated with information about the number of missed cleavages.

Value

This function returns a list. For each analysis a respective data frame including information of missed cleavages is stored in the generated list.

- Analysis - analysis name.
- Missed.Cleavage - categorical entry with number of missed cleavages.
- mc_count - number of missed cleavages per categorical missed cleavage entry - absolute or in percentage.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(tibble)
library(magrittr)
library(stringr)

# Example data
data <- list(
  Spectronaut = list(
    filename = "C",
    software = "Spectronaut",
    data = list(
      "Spectronaut" = tibble::tibble(
        Stripped.Sequence_mpwR = c("A", "B", "C", "D", "E"),
        Missed.Cleavage_mpwR = c(0, 1, 1, 2, 2)
      )
    )
  )
)

# Result
output <- get_MC_Report(
  input_list = data,
  metric = "absolute"
)
```

get_summary_Report *Summary report*

Description

Generates a summary report

Usage

```
get_summary_Report(  
  input_list,  
  CV_RT_th_hold = 5,  
  CV_LFQ_Pep_th_hold = 20,  
  CV_LFQ_PG_th_hold = 20  
)
```

Arguments

input_list A list with data frames including ID, DC, MC, LFQ and RT information.

CV_RT_th_hold Numeric. User-specified threshold for CV value of retention time precision. Default is 5.

CV_LFQ_Pep_th_hold Numeric. User-specified threshold for CV value of quantitative precision. Default is 20.

CV_LFQ_PG_th_hold Numeric. User-specified threshold for CV value of quantitative precision. Default is 20.

Details

For each submitted data a summary report including information about achieved identifications (ID), data completeness (DC), missed cleavages (MC), and both quantitative (LFQ) and retention time (RT) precision is generated.

Value

This function returns a list. For each analysis a respective data frame is stored in the list with the following information:

- Analysis - analysis name.
- "Median ProteinGroup.IDs abs." - median number of proteingroup identifications.
- "Median Protein.IDs abs." - median number of protein identifications.
- "Median Peptide.IDs abs." - median number of peptide identifications.
- "Median Precursor.IDs abs." - median number of precursor identifications.
- "Full profile - Precursor.IDs abs." - number of precursor identifications for full profiles.
- "Full profile - Peptide.IDs abs." - number of peptide identifications for full profiles.
- "Full profile - Protein.IDs abs." - number of protein identifications for full profiles.
- "Full profile - ProteinGroup.IDs abs." - number of proteingroup identifications for full profiles.
- "Full profile - Precursor.IDs %" - number of precursor identifications for full profiles in percentage.
- "Full profile - Peptide.IDs %" - number of peptide identifications for full profiles in percentage.
- "Full profile - Protein.IDs %" - number of protein identifications for full profiles in percentage.

- "Full profile - ProteinGroup.IDs %" - number of proteinGroup identifications for full profiles in percentage.
- "Precursor.IDs abs. with a CV Retention time < X %" - number of precursor identifications with a CV value for retention time precision under user-specified threshold X.
- "Proteingroup.IDs abs. with a CV LFQ < X %" - number of proteingroup identifications with a CV value for quantitative precision under user-specified threshold X.
- "Peptide.IDs abs. with a CV LFQ < X %" - number of peptide identifications with a CV value for quantitative precision under user-specified threshold X.
- "Peptide IDs with zero missed cleavages abs." - number of peptide identifications with zero missed cleavages.
- "Peptide IDs with zero missed cleavages %" - number of peptide identifications with zero missed cleavages in percentage.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(tibble)

# Example data
data <- list(
  DIANN = list(
    filename = "B",
    software = "DIA-NN",
    data = list(
      "DIA-NN" = tibble::tibble(
        "Run_mpwR" = c("R01", "R01", "R02", "R03", "R01"),
        "Precursor.IDs_mpwR" = c("A1", "A1", "A1", "A1", "B2"),
        "Retention.time_mpwR" = c(3, 3.5, 4, 5, 4),
        "ProteinGroup_LFQ_mpwR" = c(3, 4, 5, 4, 4),
        "Peptide.IDs_mpwR" = c("A", "A", "A", "A", "B"),
        "Protein.IDs_mpwR" = c("A", "A", "A", "A", "B"),
        "ProteinGroup.IDs_mpwR" = c("A", "A", "A", "A", "B"),
        "Stripped.Sequence_mpwR" = c("ABCR", "AKCR", "ABKCK", "ARKAR", "ABCDR")
      )
    )
)

# Result
output <- get_summary_Report(
  input_list = data
)
```

get_Upset_list	<i>Generate Upset list</i>
----------------	----------------------------

Description

Generate a list as input for Upset plot

Usage

```
get_Upset_list(  
  input_list,  
  level = c("Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs"),  
  percentage_runs = 100,  
  flowTraceR = FALSE,  
  remove_traceR_unknownMods = FALSE  
)
```

Arguments

input_list	A list with data frames and respective level information.
level	Character string. Choose between "Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs". Default is "Precursor.IDs".
percentage_runs	Number. Percentage of appearance in runs. 100 means: Identification is present in 100% of runs. Default is 100.
flowTraceR	Logical. If FALSE no level conversion is applied. Useful for inter-software comparisons. Default is FALSE.
remove_traceR_unknownMods	Logical. If FALSE no unknown Modifications are filtered out. Only applies if flowTraceR is set to TRUE. Default is FALSE.

Details

An input is generated for Upset plotting for either precursor-, peptide-, protein- or proteingroup-level. For inter-software comparisons flowTraceR is integrated.

Value

This function returns a list for each analysis with respective level information.

Author(s)

Oliver Kardell

Examples

```

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

# Example data
data <- list(
  DIANN = list(
    filename = "B",
    software = "DIA-NN",
    data = list(
      "DIA-NN" = tibble::tibble(
        Run_mpwR = rep(c("A","B"), times = 10),
        Precursor.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4),
        Protein.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4),
        Peptide.IDs_mpwR = rep(c("A", "A", "B", "B", "C"), each = 4),
        ProteinGroup.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 4)
      )
    )
  ),
  Spectronaut = list(
    filename = "C",
    software = "Spectronaut",
    data = list(
      "Spectronaut" = tibble::tibble(
        Run_mpwR = rep(c("A","B"), times = 15),
        Precursor.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 6),
        Peptide.IDs_mpwR = rep(c("A", "A", "B", "B", "C"), each = 6),
        ProteinGroup.IDs_mpwR = rep(c("A2", "A3", "B2", "B3", "C1"), each = 6)
      )
    )
  )
)

# Result
output <- get_Upset_list(
  input_list = data,
  level = "Precursor.IDs"
)

```

plot_CV_density

Density plot

Description

Plot cumulative density for precision results

Usage

```
plot_CV_density(  
  input_list,  
  xaxes_limit = 50,  
  cv_col = c("RT", "Pep_quant", "PG_quant")  
)
```

Arguments

<code>input_list</code>	A list with data frames and respective information on quantitative or retention time precision.
<code>xaxes_limit</code>	Numeric. Limit of x-axes in plot.
<code>cv_col</code>	Character string. Choose between "RT", "Pep_quant", "PG_quant" for corresponding precision category. Default is RT for retention time precision. Pep_quant equals quantitative precision on peptide-level. PG_quant equals quantitative precision on proteingroup-level.

Details

Quantitative or retention time precision are plotted as cumulative density.

Value

This function returns a density plot.

Author(s)

Oliver Kardell

Examples

```
# Load libraries  
library(dplyr)  
library(comprehenr)  
library(tibble)  
  
# Example data  
set.seed(123)  
data <- list(  
  "A" = tibble::tibble(  
    Analysis_mpwR = rep("A", times = 10),  
    CV_Retention.time_mpwR = sample(1:20, 10),  
    CV_Peptide_LFQ_mpwR = sample(1:30, 10),  
    CV_ProteinGroup_LFQ_mpwR = sample(1:30, 10)),  
  "B" = tibble::tibble(  
    Analysis_mpwR = rep("B", times = 10),  
    CV_Retention.time_mpwR = sample(1:20, 10),  
    CV_Peptide_LFQ_mpwR = sample(1:30, 10),  
    CV_ProteinGroup_LFQ_mpwR = sample(1:30, 10))  
)
```

```
# Plot
plot_CV_density(
  input_list = data,
  cv_col = "Pep_quant"
)
```

plot_DC_barplot *Individual Barplots - Data Completeness*

Description

Plot number of identifications per missing values for each analysis.

Usage

```
plot_DC_barplot(
  input_list,
  level = c("Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs"),
  label = c("absolute", "percentage")
)
```

Arguments

input_list	A list with data frames and respective level information.
level	Character string. Choose between "Precursor.IDs", "Peptide.IDs", "Protein.IDs" or "ProteinGroup.IDs" for corresponding level. Default is "Precursor.IDs".
label	Character string. Choose between "absolute" or "percentage". Default is "absolute".

Details

For each submitted individual analysis a detailed barplot is generated with information about the number of achieved identifications per missing values.

Value

This function returns a list with a barplot for each analysis.

Author(s)

Oliver Kardell

Examples

```

# Load libraries
library(magrittr)
library(comprehenr)
library(tibble)

# Example data
data <- list(
  "A" = tibble::tibble(
    Analysis = c("A", "A", "A"),
    Nr.Missing.Values = c(2, 1, 0),
    Precursor.IDs = c(50, 200, 4500),
    Peptide.IDs = c(30, 190, 3000),
    Protein.IDs = c(20, 40, 600),
    ProteinGroup.IDs = c(15, 30, 450),
    Profile = c("unique", "shared with at least 50%", "complete")
  ),
  "B" = tibble::tibble(
    Analysis = c("B", "B", "B"),
    Nr.Missing.Values = c(2, 1, 0),
    Precursor.IDs = c(50, 180, 4600),
    Peptide.IDs = c(50, 170, 3200),
    Protein.IDs = c(20, 40, 500),
    ProteinGroup.IDs = c(15, 30, 400),
    Profile = c("unique", "shared with at least 50%", "complete")
  )
)

# Plot
plot_DC_barplot(
  input_list = data,
  level = "Precursor.IDs",
  label = "absolute"
)

```

plot_DC_stacked_barplot

Summary Barplot - Data Completeness

Description

Plot number of identifications per missing values as stacked barplot.

Usage

```

plot_DC_stacked_barplot(
  input_list,
  level = c("Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs"),
  label = c("absolute", "percentage")
)

```

Arguments

input_list	A list with data frames and respective level information.
level	Character string. Choose between "Precursor.IDs", "Peptide.IDs", "Protein.IDs" or "ProteinGroup.IDs" for corresponding level. Default is "Precursor.IDs".
label	Character string. Choose between "absolute" or "percentage". Default is "absolute".

Details

The analyses are summarized in a stacked barplot displaying information about the number of achieved identifications per missing values.

Value

This function returns a stacked barplot.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(magrittr)
library(dplyr)
library(tibble)

# Example data
data <- list(
  "A" = tibble::tibble(
    Analysis = c("A", "A", "A"),
    Nr.Missing.Values = c(2, 1, 0),
    Precursor.IDs = c(50, 200, 4500),
    Peptide.IDs = c(30, 190, 3000),
    Protein.IDs = c(20, 40, 600),
    ProteinGroup.IDs = c(15, 30, 450),
    Profile = c("unique", "shared with at least 50%", "complete")
  ),
  "B" = tibble::tibble(
    Analysis = c("B", "B", "B"),
    Nr.Missing.Values = c(2, 1, 0),
    Precursor.IDs = c(50, 180, 4600),
    Peptide.IDs = c(50, 170, 3200),
    Protein.IDs = c(20, 40, 500),
    ProteinGroup.IDs = c(15, 30, 400),
    Profile = c("unique", "shared with at least 50%", "complete")
  )
)

# Plot
plot_DC_stacked_barplot(
```

```
    input_list = data,  
    level = "Precursor.IDs",  
    label = "absolute"  
  )
```

plot_ID_barplot

Individual Barplots - Identifications

Description

Plot number of achieved identifications per analysis.

Usage

```
plot_ID_barplot(  
  input_list,  
  level = c("Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs")  
)
```

Arguments

input_list	A list with data frames and respective level information.
level	Character string. Choose between "Precursor.IDs", "Peptide.IDs", "Protein.IDs" or "ProteinGroup.IDs" for corresponding level. Default is "Precursor.IDs".

Details

For each submitted individual analysis a detailed barplot is generated with information about the number of achieved identifications per run.

Value

This function returns a list with a barplot for each analysis.

Author(s)

Oliver Kardell

Examples

```
# Load libraries  
library(magrittr)  
library(comprehenr)  
library(tibble)  
  
# Example data  
data <- list(  
  "A" = tibble::tibble(  
    Analysis = c("A", "A", "A"),
```

```

Run = c("R01", "R02", "R03"),
Precursor.IDs = c(4800, 4799, 4809),
Peptide.IDs = c(3194, 3200, 3185),
Protein.IDs = c(538, 542, 538),
ProteinGroup.IDs = c(487, 490, 486)
),
"B" = tibble::tibble(
  Analysis = c("B", "B", "B"),
  Run = c("R01", "R02", "R03"),
  Precursor.IDs = c(4597, 4602, 4585),
  Peptide.IDs = c(3194, 3200, 3185),
  Protein.IDs = c(538, 542, 538),
  ProteinGroup.IDs = c(487, 490, 486)
)
)

# Plot
plot_ID_barplot(
  input_list = data,
  level = "Precursor.IDs"
)

```

plot_ID_boxplot

Summary Boxplot - Identifications

Description

Plot summary of number of identifications in boxplot.

Usage

```

plot_ID_boxplot(
  input_list,
  level = c("Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs")
)

```

Arguments

input_list	A list with data frames and respective level information.
level	Character string. Choose between "Precursor.IDs", "Peptide.IDs", "Protein.IDs" or "ProteinGroup.IDs" for corresponding level. Default is "Precursor.IDs".

Details

The analyses are summarized in a boxplot displaying information about the number of achieved identifications.

Value

This function returns a boxplot.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(magrittr)
library(dplyr)
library(tibble)

# Example data
data <- list(
  "A" = tibble::tibble(
    Analysis = c("A", "A", "A"),
    Run = c("R01", "R02", "R03"),
    Precursor.IDs = c(7000, 6100, 4809),
    Peptide.IDs = c(3194, 3200, 3185),
    Protein.IDs = c(538, 542, 538),
    ProteinGroup.IDs = c(487, 490, 486)
  ),
  "B" = tibble::tibble(
    Analysis = c("B", "B", "B"),
    Run = c("R01", "R02", "R03"),
    Precursor.IDs = c(3000, 3500, 4585),
    Peptide.IDs = c(3194, 3200, 3185),
    Protein.IDs = c(538, 542, 538),
    ProteinGroup.IDs = c(487, 490, 486)
  )
)

# Plot
plot_ID_boxplot(
  input_list = data,
  level = "Precursor.IDs"
)
```

plot_MC_barplot*Individual Barplots - Missed Cleavages*

Description

Plot number of missed cleavages for each analysis.

Usage

```
plot_MC_barplot(input_list, label = c("absolute", "percentage"))
```

Arguments

input_list	A list with data frames and respective information about missed cleavages.
label	Character string. Choose between "absolute" or "percentage". Default is "absolute".

Details

For each submitted individual analysis a detailed barplot is generated with information about the number of missed cleavages.

Value

This function returns a list with a barplot for each analysis.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(comprehenr)
library(tibble)

# Example data
data <- list(
  "A" = tibble::tibble(
    Analysis = c("A", "A", "A", "A", "A"),
    Missed.Cleavage = c("0", "1", "2", "3", "No R/K cleavage site"),
    mc_count = c("2513", "368", "23", "38", "10")
  ),
  "B" = tibble::tibble(
    Analysis = c("B", "B", "B", "B", "B"),
    Missed.Cleavage = c("0", "1", "2", "3", "No R/K cleavage site"),
    mc_count = c("2300", "368", "23", "38", "10")
  )
)

# Plot
plot_MC_barplot(
  input_list = data,
  label = "absolute"
)
```

`plot_MC_stacked_barplot`*Summary Barplot - Missed Cleavages*

Description

Plot number of missed cleavages as stacked barplot.

Usage

```
plot_MC_stacked_barplot(input_list, label = c("absolute", "percentage"))
```

Arguments

<code>input_list</code>	A list with data frames and respective information about missed cleavages.
<code>label</code>	Character string. Choose between "absolute" or "percentage". Default is "absolute".

Details

The analyses are summarized in a stacked barplot displaying information about the number of missed cleavages.

Value

This function returns a stacked barplot.

Author(s)

Oliver Kardell

Examples

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

# Example data
data <- list(
  "A" = tibble::tibble(
    Analysis = c("A", "A", "A", "A", "A"),
    Missed.Cleavage = c("0", "1", "2", "3", "No R/K cleavage site"),
    mc_count = c("2513", "368", "23", "38", "10")
  ),
  "B" = tibble::tibble(
    Analysis = c("B", "B", "B", "B", "B"),
    Missed.Cleavage = c("0", "1", "2", "3", "No R/K cleavage site"),
    mc_count = c("2300", "368", "23", "38", "10")
  )
)
```

```
)  
  
# Plot  
plot_MC_stacked_barplot(  
  input_list = data,  
  label = "absolute"  
)
```

plot_radarchart	<i>Radar chart</i>
-----------------	--------------------

Description

Plot radar chart of summary statistics.

Usage

```
plot_radarchart(input_df)
```

Arguments

input_df	Data frame with summary information. Analysis column and at least one category column is required.
----------	--

Details

Summary results are displayed via radar chart. Each analysis has its own trace.

Value

This function returns a radar chart as htmlwidget.

Author(s)

Oliver Kardell

Examples

```
# Load libraries  
library(plotly)  
library(tibble)  
  
# Example data  
data <- tibble::tibble(  
  Analysis = c("A", "B"),  
  "Median ProteinGroup.IDs [abs.]" = c(5, 10),  
  "Median Protein.IDs [abs.]" = c(5, 10),  
  "Median Peptide.IDs [abs.]" = c(5, 10),  
  "Median Precursor.IDs [abs.]" = c(5, 10),  
  "Full profile - Precursor.IDs [abs.]" = c(5, 10),
```



```

"Full profile - Peptide.IDs [abs.]" = c(5, 10),
"Full profile - Protein.IDs [abs.]" = c(5, 10),
"Full profile - ProteinGroup.IDs [abs.]" = c(5, 10),
"Full profile - Precursor.IDs [%]" = c(5, 10),
"Full profile - Peptide.IDs [%]" = c(5, 10),
"Full profile - Protein.IDs [%]" = c(5, 10),
"Full profile - ProteinGroup.IDs [%]" = c(5, 10),
"Precursor.IDs [abs.] with a CV Retention time < 5 [%]" = c(5, 10),
"Proteingroup.IDs [abs.] with a CV LFQ < 20 [%]" = c(NA, 10),
"Peptide.IDs [abs.] with a CV LFQ < 20 [%]" = c(NA, 10),
"Peptide IDs with zero missed cleavages [abs.]" = c(5, 10),
"Peptide IDs with zero missed cleavages [%]" = c(5, 10)
)

# Plot
plot_radarchart(
  input_df = data
)

```

plot_Upset

Upset Plot

Description

Plot intersections of analyses for different levels.

Usage

```

plot_Upset(
  input_list,
  label = c("Precursor.IDs", "Peptide.IDs", "Protein.IDs", "ProteinGroup.IDs"),
  nr_intersections = 10,
  highlight_overlap = FALSE
)

```

Arguments

input_list	A list with data frames and respective level information.
label	Character string. Choose between "Precursor.IDs", "Peptide.IDs", "Protein.IDs" or "ProteinGroup.IDs" for corresponding level. Default is "Precursor.IDs".
nr_intersections	Numeric. Maximum number of intersections shown in plot. Default is 10.
highlight_overlap	Logical. If TRUE, overlapping intersections is highlighted in yellow. Default is FALSE. If TRUE, overlapping intersections need to be in plot!

Details

Identifications per level of each analysis are compared and possible intersections visualized.

Value

This function returns a Upset plot.

Author(s)

Oliver Kardell

Examples

```
# Load libraries
library(UpSetR)
library(tibble)

# Example data
data <- list(
  "A" = c("A", "B", "C", "D"),
  "B" = c("A", "B", "C", "F"),
  "C" = c("A", "B", "G", "E")
)

# Plot
plot_Upset(
  input_list = data,
  label = "Peptide.IDs"
)
```

prepare_mpwR

Load and Prepare the input data

Description

Input data will be imported, renamed and default filtering will be applied

Usage

```
prepare_mpwR(path)
```

Arguments

path	Path to folder where the input data is stored - only input data. No subfolders or other files. Analysis name as prefix + for MaxQuant: <code>_evidence</code> , <code>_peptides</code> , <code>_proteinGroups</code> ; for PD - R-friendly headers enabled: <code>_PSMs</code> , <code>_Proteins</code> , <code>_PeptideGroups</code> , <code>_ProteinGroups</code> ; for DIA-NN and Spectronaut: <code>_Report</code>
------	--

Details

Function for easily importing software imports and preparing for downstream analysis with mpwR. As default for MaxQuant "Reverse" and "Potential Contaminants" are filtered out. As default for PD only "High" confidence identifications are included. As default for Spectronaut only EG.Identified equals TRUE are included.

Value

A list - each list entry has filename and software info as well as stored data.

Author(s)

Oliver Kardell

Examples

```
## Not run:  
prepare_mpwR(path = "DIRECTORY_TO_FILES")  
  
## End(Not run)
```

Index

[create_example](#), 2

[get_CV_LFQ_pep](#), 3
[get_CV_LFQ_pg](#), 4
[get_CV_RT](#), 5
[get_DC_Report](#), 6
[get_ID_Report](#), 8
[get_MC_Report](#), 9
[get_summary_Report](#), 10
[get_Upset_list](#), 13

[plot_CV_density](#), 14
[plot_DC_barplot](#), 16
[plot_DC_stacked_barplot](#), 17
[plot_ID_barplot](#), 19
[plot_ID_boxplot](#), 20
[plot_MC_barplot](#), 21
[plot_MC_stacked_barplot](#), 23
[plot_radarchart](#), 24
[plot_Upset](#), 25
[prepare_mpwR](#), 26