exMCnebula2

February 24, 2023

Title What the Package Does (One Line, Title Case)

Version 0.0.0.9000

Author LiChuang Huang <shaman.yellow@foxmail.com> [aut]
(<https://orcid.org/0000-0002-5445-1988>)Gang Cao [cre]

Maintainer LiChuang Huang <shaman.yellow@foxmail.com>

Description What the package does (one paragraph).

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.0

Depends ggplot2,
grid,
R (>= 2.10)

Imports aplot,
BiocParallel,
ChemmineOB,
classyfireR,
data.table,
dplyr,
FELLA,
ggraph,
ggsci,
ggtree,
grImport2,
Hmisc,
igraph,
magrittr,
MCnebula2,
pbapply,
pdftools,
png,
RCurl,
rlang,

rsvg,
scales,
stringr,
tibble,
tidygraph,
tidyr,
XML

R topics documented:

`align_merge` *Alignment merge of 'features' via m/z and RT*

Description

Merge two data.frame of 'features' according to m/z and RT. The formula is the same in MZmine2: Score = (1 - rt.difference / rt.tolerance) * rt.weight + (1 - mz.defference / mz.tolerance) * mz.weight

`align_merge`: ...

`tol_merge`: ...

Usage

```
align_merge(
  main,
  sub,
  id = ".features_id",
  mz.main = "mz",
  mz.sub = "mz",
  rt.main = "rt.min",
  rt.sub = "rt.min",
  mz.tol = 0.05,
  rt.tol = 0.3,
  mz.weight = 75,
  rt.weight = 25,
  unique = T
)

tol_merge(
  main,
  sub,
  main_col = "mz",
  sub_col = "mz",
  tol = 0.002,
  bin_size = 1
)
```

arrow *draw arrow*

Description

...

parrow: ...

garrow: ...

garrow_city: ...

garrow_snake: ...

garrow_city_args: ...

sagnage: ...

sagnage_shiny: ...

maparrow: ...

Usage

```
parrow(n = 5, col, type = "dashed", lwd = u(1, line))

garrow(p1, p2, args_line, args_arrow, fun_line, fun_arrow, city)

garrow_city(p1, p2, up, left, shift, gp_line)

garrow_snake(p1, p2, color, lwd = u(1, line), cur = 1)

garrow_city_args(
  shift = u(2, line),
  axis = "x",
  mid = 0.5,
  args_line = list(ncp = 1, curvature = 1, square = T),
  grob_anno = NULL,
  vp_anno = NULL,
  rev = F
)

sagnage(
  grob,
  left = T,
  l_gpar,
  borderF = 1.5,
  front_len = 0.1,
  vp_shift = u(1.5, line),
  ...
)

sagnage_shiny(label, left, color)

maparrow(
  obj,
  data,
  pattern = list(),
  round_cur = 0.3,
  pos = list(r = list(x = 1), l = list(x = 0), t = list(y = 1), b = list(y = 0))
)
```

as_grob *convert ’ggplot’ object to ’grobs’*

Description

...

Usage

```
as_grob(x)
```

draw-methods *draw class of 'graph' and 'grobs'*

Description

...

format_table *Format table via dplyr::**

Description

Format the data.frame via: `dplyr::filter`, `dplyr::arrange`, `dplyr::distinct`, `dplyr::mutate`, `dplyr::select`, `dplyr::rename`.

`rename_table`: ...

`format_table`: ...

`.filter_format`: ...

`.arrange_format`: ...

`.distinct_format`: ...

`.mutate_format`: ...

`.select_format`: ...

`.export_name`: ...

Usage

```
rename_table(data, export_name = .export_name)

format_table(
  data,
  filter = .filter_format,
  arrange = .arrange_format,
  distinct = .distinct_format,
  mutate = .mutate_format,
  select = .select_format,
  export_name = .export_name
)

.filter_format
```

```
.arrange_format

.distinct_format

.mutate_format

.select_format

.export_name
```

Arguments

`data`	data.frame. From `features_annotation(mcn)`.

Format

An object of class `list` of length 1.

An object of class `list` of length 3.

An object of class `list` of length 1.

An object of class `list` of length 4.

An object of class `character` of length 14.

An object of class `character` of length 17.

`frame` *draw in grid frame*

Description

...

`layout_row`: ...

`frame_row`: ...

`layout_col`: ...

`frame_col`: ...

`frame_place`: ...

Usage

```
layout_row(weight)

frame_row(weight, data, if.ex)

layout_col(weight)

frame_col(weight, data, if.ex)

frame_place(weight, data, type, if.ex)
```

`ggather` *a mutate of grid::gTree*

Description

...

`ggather`: ...

`zo`: ...

Usage

```
ggather(..., vp = NULL, gp = NULL)

zo(x, w = 0.9, h = 0.9)
```

`graph-class` *A class built on 'grobs'*

Description

...

Examples

```
## Not run:
new('graph', ...)

## End(Not run)
```

`into` *place 'grobs' into 'graph'*

Description

...

Usage

```
into(x, content)
```

network *Quickly draw random network diagrams*

Description

...

`fast_layout`: ...

`random_graph`: ...

Usage

```
fast_layout(edges, layout = "fr", nodes = NULL)

random_graph(ids, n = 5, e = 4, layout = "fr")
```

pathway_enrichment *Perform pathway enrichment via package of 'FELLA'*

Description

Pathway enrichment analysis was performed using KEGG ID via package of 'FELLA'. (Convert CID to KEGG ID using the 'MetaboAnalystR' package. See https://github.com/xia-lab/MetaboAnalystR for installation.)

`init_fella`: ...

`load_fella`: ...

`enrich_fella`: ...

`graph_fella`: ...

`plotGraph_fella`: Draw the graph via package of 'ggplot2'.

`cid.to.kegg`: ...

Usage

```
init_fella(dir, org = c("hsa", "mmu", "rno"), seed = 1, rebuild = F)

load_fella(dir)

enrich_fella(id.lst, data)

graph_fella(
  obj.lst,
  data,
  method = c("pagerank", "diffusion", "hypergeom"),
  threshold = 0.1
)

plotGraph_fella(
  graph,
  layout = "graphopt",
  seed = 1,
  shape = c(Input = 15, Others = 16),
  color = c(Pathway = "#E64B35FF", Module = "#E377C2", Enzyme = "#EFC000", Reaction =
    "#4DBBD5FF", Compound = "#00A087FF"),
  size = c(Pathway = 7, Module = 5, Enzyme = 6, Reaction = 5, Compound = 10)
)

cid.to.kegg(cids)
```

See Also

`FELLA::buildDataFromGraph()`, `FELLA::buildGraphFromKEGGREST()`

`pick_annotation` *Pick unique annotation for compounds*

Description

Pick unique chemical class or synonyms for 'features'.

`pick_class`: ...

`.filter_pick.class`: ...

`PickClass`: ...

`pick_synonym`: ...

`.filter_pick.general`: ...

`PickGeneral`: ...

Usage

```
pick_class(
  inchikey2d,
  class.rdata,
  filter = .filter_pick.class,
  fun = PickClass
)

.filter_pick.class

PickClass(class)
```

```
pick_synonym(
  inchikey2d = NULL,
  inchikey.rdata = NULL,
  synonym.rdata,
  iupac.rdata = NULL,
  filter = .filter_pick.general,
  fun = PickGeneral
)

.filter_pick.general

PickGeneral(
  syno,
  ps = c("^[a-zA-Z]*$", "^[a-zA-Z-]*$", "^[a-zA-Z0-9-]*$", "^[^:]*$")
)
```

Format

An object of class list of length 2.

An object of class list of length 6.

See Also

Other queries: query_classification(), query_inchikey(), query_iupac(), query_synonyms()

---

plot_EIC_stack *Draw extracted ions chromatography for 'features'*

Description

Use quantification table (with peak start time and end time) exported by MCmine to draw EIC plot.

plot_EIC_stack: ...

bioc.par: ...

Usage

```
plot_EIC_stack(
  idset,
  metadata,
  quant.path,
  mzml.path,
  palette = (ggsci::pal_npg())(10),
  mz.tol = 0.01,
  rt.tol = 0.1,
  cl = NULL,
  data = NULL
)

bioc.par(cl = 4)
```

plot_heatmap *Plot heat map with ggplot2*

Description

According to list of 'ID' to draw mutiple heatmap...

plot_heatmap: ...

handling_na: For each subset of data, the missing values will be filled with the average value; if the set is all missing values, they will be filled with zero.

log_trans: Convert wide data to long data; log transform the values; if there is a value 0, replace it with 1/10 of the minimum value of the value column.

add_tree.heatmap: ...

add_xgroup.heatmap: ...

add_xgroup.tile.heatmap: ...

add_ygroup.tile.heatmap: ...

Usage

```
plot_heatmap(
  id.lst,
  data,
  metadata,
  pal_class = (ggsci::pal_futurama())(12),
  pal_group,
  clust_row = T,
  clust_col = T,
  method = "complete"
)

handling_na(
  data,
  id.cols = c(".features_id"),
  metadata,
  sample.col = "sample",
  group.col = "group"
)

log_trans(
  data,
  id.cols = c(".features_id"),
  key = "sample",
  value = "value",
  set_min = T,
  factor = 10,
  fun = log2,
  center = T
)

add_tree.heatmap(df, p, clust_row = T, clust_col = T, method = "complete")

add_xgroup.heatmap(df, p)

add_xgroup.tile.heatmap(df, p, pal = NA)

add_ygroup.tile.heatmap(df, p, pal = NA)
```

query_classification *Query classification of compounds via packgage of 'classifyerR'*

Description

The used function is: `classyfireR::get_classification(inchikey)`

`query_classification`: ...

`classyfire_get_classification`: ...

Usage

```
query_classification(
  inchikey2d,
  dir,
  inchikey.rdata = paste0(dir, "/inchikey.rdata"),
  rdata.name = "classification.rdata",
  classyfire_cl = NULL,
  gather_as_rdata = T,
  ...
)

classyfire_get_classification(
  sets,
  dir,
  classyfire_cl = NULL,
  log_file = paste0(dir, "/classyfire.log"),
  ...
)
```

See Also

Other queries: `pick_annotation`, `query_inchikey()`, `query_iupac()`, `query_synonyms()`

`query_inchikey` *Query InChIkey of compounds via 'InChIkey 2D'*

Description

The API: url_start = paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/", type, "/") url_end = paste0("/property/", paste(get, collapse = ","), "/CSV") url = paste0(url_start, "/", inchikey2d, "/", url_end)

`query_inchikey`: ...

`pubchem_get_inchikey`: ...

Usage

```
query_inchikey(
  inchikey2d,
  dir,
  rdata.name = "inchikey.rdata",
  curl_cl = NULL,
  gather_as_rdata = T,
  ...
)

pubchem_get_inchikey(inchikey2d, dir, type = "inchikey", get = "InChIkey", ...)
```

See Also

Other queries: `pick_annotation`, `query_classification()`, `query_iupac()`, `query_synonyms()`

`query_iupac` *Query IUPAC name of compounds via 'InChIkey 2D'*

Description

Similar to `query_inchikey()`, but get 'IUPACName'.

`query_iupac`: ...

Usage

```
query_iupac(
  inchikey2d,
  dir,
  rdata.name = "iupac.rdata",
  curl_cl = NULL,
  gather_as_rdata = T,
  ...
)
```

See Also

Other queries: pick_annotation, query_classification(), query_inchikey(), query_synonyms()

query_synonyms *Query synonyms of compounds via CID*

Description

Bulk search for compound synonyms via pubchem API. (https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/.../synony

query_synonyms: ...

pubchem_get_synonyms: ...

grouping_vec2list: ...

extract_rdata_list: extract results from .rdata

packing_as_rdata_list: gather table as .rdata

Usage

```
query_synonyms(
  cid,
  dir,
  rdata.name = "synonyms.rdata",
  curl_cl = NULL,
  gather_as_rdata = T,
  group_number = 50,
  ...
)

pubchem_get_synonyms(cid, dir, ...)

grouping_vec2list(vector, group_number, byrow = F)

extract_rdata_list(rdata, names = NA)

packing_as_rdata_list(
  path,
  pattern,
  rdata,
  extra = NULL,
  rm_files = T,
  dedup = T
)
```

See Also

Other queries: pick_annotation, query_classification(), query_inchikey(), query_iupac()

rect *a mutate of grid::rectGrob*

Description

...

grect: ...

grecti: ...

grecti2: ...

grecti3: ...

grecto: ...

grectn: ...

grectn_frame: ...

lst_grecti: ...

grectN: ...

grecta: ...

gshiny: ...

glayer: ...

Usage

```
grect(
  name,
  tfill = "#E18727FF",
  bfill = "white",
  t_args,
  tgp_args,
  t = roundrectGrob,
  b_args,
  bgp_args,
  b = roundrectGrob,
  order = c(1, 2),
  vp = NULL
)

grecti(
  label,
  cex = 1,
  x = 0.5,
  y = 1.005,
  borderF = 2,
  just = c("center", "top"),
  tfill = "#E18727FF",
  vp = .grecti.vp,
  order = c(2, 1),
  cvp_clip = "on",
  cvp_fix = T,
  ...
)

grecti2(label, cex = 1, tfill = "#E18727FF", borderF = list(2), ...)

grecti3(label, cex = 1, tfill = "#E18727FF", borderF = list(2), ...)

grecto(
  label,
  cex = 1,
  x = 0.5,
  y = 0.995,
  borderF = 2,
  just = c("center", "bottom"),
  tfill = "#E18727FF",
  vp = .grecto.vp,
  order = c(1, 2),
  cvp_clip = "on",
  cvp_fix = F,
  ...
)

grectn(
  bfill = "white",
  b_args,
  bgp_args,
  b = roundrectGrob,
  cvp_clip = "inherit"
)

grectn_frame(content, title, zo = T)

lst_grecti(names, pal, tar = "slot", fun = grecti, ...)

grectN(lab.1, lab.2, gp = gpar(fontface = "plain"), bfill = "white")

grecta(label, cex = 4)

gshiny(
  xn = 4,
  yn = 3,
  xps = seq(0, 1, , xn),
  yps = seq(0, 1, , yn),
  size = c(0.15, 0.02),
  color = .default_color,
  rect = rectGrob(),
  vp = viewport(clip = "off"),
  cvp = viewport(, , 0.9, 0.9)
)

glayer(n = 5, to = 0.2, gp = gpar(fill = "white"), fun = rectGrob)
```

select_features *Select 'features' for MCnebula2*

Description

Select significant 'features' from MCnebula2 with statistic results for downstream analysis of metabolomics.

select_features: ...

cross_select: ...

Usage

```
select_features(
  mcn,
  classes = unique(nebula_index(mcn)$class.name),
  q.value = 0.05,
  logfc = 0.3,
  coef = NULL,
  tani.score_cutoff = NULL,
  order_by_coef = NULL,
  togather = F
)

cross_select(data.lst, filter.lst, target, split = NULL)
```

setnull *Set markers crossover viewports*

Description

...

setnull: ...

setnullvp: ...

grepPath: ...

sort_vpPaths: ...

Usage

```
setnull(target, args, name = "null")

setnullvp(pattern, args, x, name = NULL, fix = T, perl = F)

grepPath(pattern, x = NULL, grobs = T, viewports = T, perl = F)

sort_vpPaths(vpPaths)
```

setvp …

Description

…

Usage

```
setvp(x, ...)
```

text *a mutate of grid::textGrob*

Description

…

gtext: …

gtextp: …

gtext0: …

form: …

gltext: …

Usage

```
gtext(label, gp_arg, form = T, ...)

gtextp(label, gp_arg, form = T, ...)

gtext0(label, gp_arg, form = T, ...)

form(label)

gltext(
  label,
  gp_arg = list(),
  args = list(),
  l_gp = .gpar_dotted_line,
  flip = F,
  borderF = 1.2
)
```

utilites *utilites for programming*

Description

This is a combination of tools that are not always used.

tmp_pdf: ...

op: ...

.cairosvg_to_grob: Convert cairo svg to 'grob'.

.as_dic: ...

fill_list: ...

n: ...

namel: ...

repSuffix: ...

%>%: ...

%<>%: ...

.expath: ...

agroup: ...

write_tsv: ...

read_tsv: ...

mapply_rename_col: ...

turn_vector: ...

group_switch: ...

.find_and_sort_strings: ...

maps: ...

order_list: ...

molconvert_structure: ...

obj.size: ...

Usage

```
tmp_pdf()

op(file)

.cairosvg_to_grob(path)

.as_dic(vec, names, default, fill = T, as.list = T, na.rm = F)

fill_list(names, vec, default = vec[1])

n(name, n)

namel(...)

repSuffix(chs, anno = ".rep.")

lhs %>% rhs

lhs %<>% rhs

.expath

agroup(group, value, FUN.VALUE = character(1))

write_tsv(x, filename, col.names = T, row.names = F)

read_tsv(path)

mapply_rename_col(mutate_set, replace_set, names, fixed = F)

turn_vector(vec)

group_switch(data, meta.lst, by)

.find_and_sort_strings(strings, patterns)

maps(data, value, from, to)

order_list(list)

molconvert_structure(smile, path)

obj.size(x, ...)
```

Format

An object of class `character` of length 1.

`zoom_pdf` *Zoom in locally pdf to png*

Description

...

`zoom_pdf`: ...

Usage

```
zoom_pdf(
  file,
  position = c(0.5, 0.5),
  size = c(0.15, 0.1),
  page = 1,
  dpi = 2000,
  as.grob = T
)
```

Index