Package 'admetshiny'

October 25, 2026

Type Package

Title Interactive ADMET and Drug-Likeness Analysis of Small Molecules

Version 1.0.1

Date 2026-10-24

Maintainer Xavier Clemente Garcia Cevallos <xgarcia@unicauca.edu.co>

Description Provides an interactive Shiny application and a toolbox of R functions for the management, calculation, filtering, visualization and exploratory analysis of molecular descriptors and ADMET (Absorption, Distribution, Metabolism, Excretion and Toxicity) properties of small molecules. Computes descriptors locally via the Chemistry Development Kit (CDK), and offers drug-likeness filters (Lipinski, Veber, Ghose, Egan, Muegge), the BOILED-Egg model for gastrointestinal absorption and blood-brain barrier permeability, a P-glycoprotein (P-gp, also known as ATP-binding cassette sub-family B member 1, ABCB1) substrate Random Forest classifier, Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP), radar plots and Tanimoto / AGglomerative NESting (AGNES) clustering to support compound prioritization in early-stage drug discovery.

License MIT + file LICENSE

URL https://xavierclementegarcia.github.io/ADMETShiny/

BugReports https://github.com/xavierclementegarcia/admetshiny/issues

Depends R (>= 3.5.0)

Imports cluster, dplyr, DT, fingerprint, fmsb, GGally, ggplot2, ggrepel, graphics, grDevices, grid, magrittr, openxlsx, rcdk, rmarkdown, Rtsne, shiny, stats, tools, utils, uwot, vegan, viridisLite, webchem

Suggests knitr, spelling, testthat

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Encoding UTF-8

SystemRequirements Java (>= 8); only required for the optional CDK-based descriptor calculation (rcdk) module.

Language en-US

NeedsCompilation no

Author Xavier Clemente Garcia Cevallos [aut, cre] (ORCID: <https://orcid.org/0000-0002-3284-2363>)

Repository CRAN

Date/Publication 2026-10-25 10:00:09 UTC

Contents

admetshiny-package . . . 3
applyFilters . . . 5
apply_palette . . . 6
calcCDKDescriptors . . . 6
computeADMETProperties . . . 7
computeViolationColumns . . . 8
eganFilter . . . 9
generateReport . . . 10
getSmilesFromIdentifiers . . . 11
ghoseFilter . . . 12
lipinskiFilter . . . 13
mapADMETColumns . . . 14
mapCDKDescriptors . . . 15
mueggeFilter . . . 16
palette_selector_ui . . . 17
plotBoiledEgg . . . 18
plotClusterHeatmap . . . 19
plotCorrHeatmap . . . 20
plotHistogramCustom . . . 21
plotLogP . . . 22
plotMW . . . 22
plotParallel . . . 23
plotPCA . . . 23
plotRadar . . . 24
plotTanimoto . . . 25
plotTPSA . . . 26
plotTSNE . . . 27
plotUMAP . . . 28
plotViolin . . . 29
run_app . . . 30
veberFilter . . . 30

Index **32**

admetshiny-package *admetshiny: Interactive ADMET and Drug-Likeness Analysis*

Description

The **admetshiny** package provides an interactive Shiny application and a collection of R functions for the management, calculation, filtering, visualization and exploratory analysis of molecular descriptors and ADMET properties of small molecules. It computes descriptors locally via the Chemistry Development Kit (CDK) through **rcdk**, and offers drug-likeness filters (Lipinski, Veber, Ghose, Egan, Muegge), the BOILED-Egg model, a P-glycoprotein substrate Random Forest classifier, PCA, t-SNE, UMAP, parallel coordinates, radar plots and Tanimoto/AGNES structural similarity clustering.

Details

To launch the interactive application simply run:

```
admetshiny::run_app()
```

The exported functions (filters, plots, data normalisation and CDK helpers) can also be used programmatically in plain R scripts. By Xavier Clemente Garcia Cevallos :3 "Hecho con amor para la ciencia"

Drug-likeness filters

- `lipinskiFilter` - Lipinski Rule-of-Five
- `veberFilter` - Veber oral bioavailability
- `ghoseFilter` - Ghose qualifying range
- `eganFilter` - Egan PSA-LogP rule
- `mueggeFilter` - Muegge pharmacophore filter
- `applyFilters` - Apply multiple filters in sequence
- `computeViolationColumns` - Compute violation columns

Data normalization

- `mapADMETColumns` - Map any user dataset to standard schema
- `mapCDKDescriptors` - Map CDK descriptors to standard schema

CDK & webchem

- `getSmilesFromIdentifiers` - Retrieve SMILES from PubChem
- `calcCDKDescriptors` - Calculate descriptors with CDK
- `computeADMETProperties` - Compute BOILED-Egg ADMET properties

Visualizations

- plotBoiledEgg - BOILED-Egg model
- plotMW, plotTPSA, plotLogP - Distributions
- plotPCA - PCA chemical space
- plotTSNE - t-SNE chemical space
- plotUMAP - UMAP chemical space
- plotParallel - Parallel coordinates
- plotViolin - Violin plot
- plotRadar - Radar chart
- plotTanimoto - Tanimoto/AGNES clustering
- plotCorrHeatmap - Correlation heatmap
- plotClusterHeatmap - Cluster heatmap with dendrogram
- plotHistogramCustom - Customizable histogram of any numeric column
- palette_selector_ui, apply_palette - Colour customization

Application

- run_app - Launch the Shiny application
- generateReport - Generate a report from the R console

Author(s)

Maintainer: Xavier Clemente Garcia Cevallos <xgarcia@unicauca.edu.co> (ORCID)

Authors:

- Xavier Clemente Garcia Cevallos <xgarcia@unicauca.edu.co> (ORCID)

See Also

Useful links:

- https://xavierclementegarcia.github.io/ADMETShiny/
- Report bugs at https://github.com/xavierclementegarcia/admetshiny/issues

applyFilters *Apply drug-likeness filters in sequence*

Description

Applies the selected drug-likeness filters to a normalized data.frame. The filters are applied in the order Lipinski, Veber, Ghose, Egan, Muegge; only those named in `filters` are executed.

Usage

```
applyFilters(
  data,
  filters,
  lipinski = list(),
  veber = list(),
  ghose = list(),
  egan = list(),
  muegge = list()
)
```

Arguments

`data`	A data.frame already normalized (e.g. by `mapADMETColumns`, `mapCDKDescriptors`, or `mapCDKDescriptors`).
`filters`	Character vector with the names of the filters to apply. Any subset of `c("Lipinski", "Veber", "Ghose", "Egan", "Muegge")`.
`lipinski, veber, ghose, egan, muegge`	Named lists of parameters forwarded to the corresponding filter function.

Value

A data.frame with the rows of `data` that pass every selected filter.

See Also

`lipinskiFilter`, `veberFilter`, `ghoseFilter`, `eganFilter`, `mueggeFilter`.

Examples

```
d <- data.frame(MW = 300, LogP = 2, TPSA = 80, MR = 90,
  "#Heavy atoms" = 22, "#Rotatable bonds" = 3,
  "#H-bond acceptors" = 4, "#H-bond donors" = 2,
  "Lipinski #violations" = 0, "Ghose #violations" = 0,
  "Veber #violations" = 0, "Egan #violations" = 0,
  "Muegge #violations" = 0, check.names = FALSE)
applyFilters(d, filters = c("Lipinski", "Veber", "Egan"))
```

apply_palette *Apply a colour palette to a ggplot object*

Description

Modifies an existing ggplot object by adding a colour/fill scale matching the selected palette, intelligently detecting whether the colour variable is continuous or discrete.

Usage

```
apply_palette(p, palette_name, data, color_col)
```

Arguments

p	A ggplot object.
palette_name	Character. Name of the palette (as returned by palette_selector_ui).
data	The data.frame used to build p.
color_col	Character. Name of the column used for colouring, or "None".

Value

A ggplot object (possibly modified).

Examples

```
library(ggplot2)
d <- data.frame(MW = c(300, 400, 500), group = c("A", "A", "B"))
p <- ggplot(d, aes(x = group, y = MW, color = group)) + geom_point()
p <- apply_palette(p, "Set1", d, "group")
```

calcCDKDescriptors *Calculate molecular descriptors with CDK*

Description

Parses SMILES strings and calculates the requested molecular descriptors locally using the Chemistry Development Kit (CDK) through the suggested package **rcdk**. Requires a working Java JDK (a JRE alone is not sufficient).

Usage

```
calcCDKDescriptors(
  smiles,
  which = c("mw", "alogp", "tpsa", "hbd", "hba", "rotb", "heavy", "aroma", "mr")
)
```

Arguments

smiles	Character vector of SMILES strings.
which	Character vector of descriptor short names. Any subset of c("mw", "alogp", "tpsa", "hbd", "hba", "rotb", "heavy", "aroma","mr"). By default all are calculated.

Value

A data.frame with one row per valid molecule and a SMILES column.

See Also

mapCDKDescriptors, getSmilesFromIdentifiers.

Examples

```
smiles <- c("CCO", "CC(=O)Oc1ccccc1C(=O)O")
desc <- calcCDKDescriptors(smiles)
```

---

computeADMETProperties

Compute BOILED-Egg ADMET properties

Description

Computes the gastrointestinal absorption (GI absorption), blood-brain barrier permeability (BBB permeant) and P-glycoprotein substrate (Pgp substrate) categorical columns from the LogP and TPSA values, using the official BOILED-Egg polygon coordinates (Daina & Zoete, 2016, Data S3) for GI/BBB classification via point-in-polygon testing, and a Random Forest classifier for P-gp substrate prediction.

Usage

```
computeADMETProperties(data)
```

Arguments

data	A data.frame with LogP, TPSA, MW, #H-bond donors and #H-bond acceptors columns.

Details

The BOILED-Egg model was originally calibrated with WLOGP; here the application's generic LogP column is used (which may be WLOGP, Consensus Log P, ALogP, or the source platform's own LogP). This is an acceptable approximation for exploratory visualization.

Value

A data.frame with the added ADMET property columns.

References

Daina, A., & Zoete, V. (2016). A boiled egg to predict gastrointestinal absorption and brain penetration of small molecules. *ChemMedChem*, 11(11), 1117-1121.

Sedykh, A., Fourches, D., Duan, J., et al. (2013). Human intestinal transporter database: QSAR modeling and virtual excretion experiments. *J. Cheminformatics* 2015, 7:21 (Metrabase P-gp data).

Examples

```
d <- data.frame(LogP = 2, TPSA = 80, MW = 300,
  "#H-bond donors" = 2, "#H-bond acceptors" = 4, check.names = FALSE)
d <- computeADMETProperties(d)
```

---

computeViolationColumns

Compute drug-likeness violation columns

Description

Computes the five drug-likeness `"#violations"` columns (Lipinski, Ghose, Veber, Egan, Muegge) from the physicochemical properties, using the thresholds from the original publications. Missing columns are safely filled with `NA`.

Usage

```
computeViolationColumns(data)
```

Arguments

`data`	A data.frame with physicochemical property columns.

Details

The generic `LogP` column is used for all LogP-dependent rules. The original publication thresholds are used:

- Lipinski: LogP > 5 (original Rule-of-Five)
- Ghose: LogP < -0.4 or LogP > 5.6
- Egan: LogP > 5.88
- Muegge: LogP < -2 or LogP > 5

For the Ghose atom-count criterion (20-70 atoms), the function uses `"#Total atoms"` (heavy + H, matching the original Ghose 1999 definition) when available, and falls back to `"#Heavy atoms"` for backwards compatibility.

The Veber violation is binary: a compound violates if it has more than 10 rotatable bonds OR if both polarity conditions fail (TPSA > 140 AND HBA + HBD > 12).

Value

A data.frame with the added violation columns.

See Also

`lipinskiFilter`, `ghoseFilter`, `veberFilter`, `eganFilter`, `mueggeFilter`.

Examples

```
d <- data.frame(MW = 300, LogP = 2, TPSA = 80, MR = 90,
  "#H-bond acceptors" = 4, "#H-bond donors" = 2,
  "#Rotatable bonds" = 3, "#Heavy atoms" = 22, check.names = FALSE)
d <- computeViolationColumns(d)
```

---

eganFilter *Egan drug-likeness filter*

Description

Filters compounds according to the Egan rule (Egan et al., 2000) based on TPSA and LogP.

Usage

```
eganFilter(data, e_tpsa = 131.6, e_logp = 5.88, violations = 0)
```

Arguments

`data`	A data.frame with the standard column schema.
`e_tpsa`	Numeric. Maximum TPSA. Default 131.6.
`e_logp`	Numeric. Maximum LogP. Default 5.88.
`violations`	Integer. Maximum tolerated Egan violations. Default 0.

Value

A data.frame with the rows of `data` that satisfy all thresholds.

References

Egan, W. J., Merz, K. M., & Baldwin, J. J. (2000). Prediction of drug absorption using multivariate statistics. *Journal of Medicinal Chemistry*, 43(21), 3867-3877.

Examples

```
d <- data.frame(TPSA = 80, LogP = 2, "Egan #violations" = 0,
  check.names = FALSE)
eganFilter(d)
```

`generateReport` *Generate an ADMETShiny report from the R console*

Description

Generates a comprehensive ADMET and drug-likeness analysis report from one or more datasets, directly from the R console (without launching the Shiny app). The report includes per-dataset statistics, drug-likeness filter results, BOILED-Egg ADMET classification, additional literature-supported metrics, a composite drug-likeness score, cross-dataset comparison, visualizations and references.

Usage

```
generateReport(
  data,
  filters = character(0),
  format = "html",
  output_file = NULL,
  source_name = "Dataset"
)
```

Arguments

`data`	A data.frame, or a named list of data.frames. If a single data.frame is provided, it is treated as one dataset named `"Dataset 1"`. If a list is provided, each element is a separate dataset and the list names are used as dataset labels.
`filters`	Character vector of drug-likeness filter names applied to the data (e.g. `c("Lipinski", "Veber")`). Use `character(0)` if no filters were applied. Default `character(0)`.
`format`	Character; one of `"html"`, `"pdf"`, `"doc"`. Default `"html"`.
`output_file`	Character; path where the output file will be written. If `NULL`, a default name is used in the current working directory.
`source_name`	Character; human-readable name for the dataset. Only used when `data` is a single data.frame. Default `"Dataset"`.

Value

Invisible `NULL`; called for the side-effect of writing the rendered report to `output_file`.

Examples

```
## Build a small synthetic ADMET dataset in the standard schema
d <- data.frame(
  Name = c("druglike", "violator"),
  SMILES = c("CCO", "CCCCCCCCCCCCCCCCCCCCCCCCCC"),
  MW = c(300, 650),
  LogP = c(2, 7),
  TPSA = c(40, 160),
  MR = c(70, 150),
  "#H-bond acceptors" = c(4, 12),
  "#H-bond donors" = c(2, 7),
  "#Rotatable bonds" = c(3, 14),
  "#Heavy atoms" = c(20, 80),
  "#Aromatic heavy atoms" = c(6, 9),
  check.names = FALSE
)
d <- computeViolationColumns(d)

## Generate an HTML report (written to a temp file)
out <- tempfile(fileext = ".html")
generateReport(d, filters = c("Lipinski", "Veber"),
               format = "html", output_file = out,
               source_name = "Example")
```

`getSmilesFromIdentifiers` *Retrieve canonical SMILES from PubChem*

Description

Queries PubChem (via the suggested package **webchem**) to obtain CIDs and canonical SMILES for a vector of chemical identifiers (names, CAS numbers, InChIKeys or PubChem CIDs).

Usage

```
getSmilesFromIdentifiers(ids, from = "name")
```

Arguments

`ids`	Character vector of identifiers.
`from`	Character. Type of identifier: one of `"name"`, `"cas"`, `"inchikey"`, `"cid"`. Default `"name"`.

Value

A data.frame with columns `query`, `cid`, `CanonicalSMILES`, `IsomericSMILES`, `MolecularFormula`, `IUPACName`.

See Also

calcCDKDescriptors.

Examples

```
## Not run:
# This service requires a constant internet connection and may fail if the
# server goes down. It may also involve long wait times. That's the main
# reason for using dontrun in this case.
ids <- c("aspirin", "ibuprofen")
smiles <- getSmilesFromIdentifiers(ids, from = "name")

## End(Not run)
```

---

ghoseFilter *Ghose drug-likeness filter*

Description

Filters compounds according to the Ghose qualifying range (Ghose et al., 1999): molecular weight, molar refractivity, LogP and number of atoms.

Usage

```
ghoseFilter(
  data,
  g_mw_min = 160,
  g_mw_max = 480,
  g_mr_min = 40,
  g_mr_max = 130,
  g_logp_min = -0.4,
  g_logp_max = 5.6,
  g_ha_min = 20,
  g_ha_max = 70,
  violations = 0
)
```

Arguments

data	A data.frame with the standard column schema.
g_mw_min	Numeric. Minimum molecular weight. Default 160.
g_mw_max	Numeric. Maximum molecular weight. Default 480.
g_mr_min	Numeric. Minimum molar refractivity. Default 40.
g_mr_max	Numeric. Maximum molar refractivity. Default 130.
g_logp_min	Numeric. Minimum LogP. Default -0.4.
g_logp_max	Numeric. Maximum LogP. Default 5.6.

g_ha_min	Numeric. Minimum number of atoms. Default 20.
g_ha_max	Numeric. Maximum number of atoms. Default 70.
violations	Integer. Maximum tolerated Ghose violations. Default 0.

Details

The "number of atoms" criterion (range 20-70) refers to TOTAL atoms (heavy + hydrogens), following the original Ghose (1999) definition.

Value

A data.frame with the rows of data that satisfy all thresholds.

References

Ghose, A. K., Viswanadhan, V. N., & Wendoloski, J. J. (1999). A knowledge-based approach in designing combinatorial or medicinal chemistry libraries for drug discovery. 1. A qualitative and quantitative characterization of known drug databases. *Journal of Combinatorial Chemistry*, 1(1), 55-68.

Examples

```
d <- data.frame(MW = 300, MR = 90, LogP = 2, "#Total atoms" = 30,
  "Ghose #violations" = 0, check.names = FALSE)
ghoseFilter(d)
```

---

lipinskiFilter *Lipinski Rule-of-Five filter*

Description

Filters compounds according to the Rule of Five (Lipinski et al., 1997): molecular weight, LogP, number of H-bond acceptors and donors, plus the pre-computed number of Lipinski violations.

Usage

```
lipinskiFilter(data, mw = 500, logp = 5, hba = 10, hbd = 5, violations = 0)
```

Arguments

data	A data.frame with the standard column schema (processed by mapADMETColumns, mapCDKDescriptors).
mw	Numeric. Maximum molecular weight. Default 500.
logp	Numeric. Maximum LogP. Default 5.
hba	Numeric. Maximum number of H-bond acceptors. Default 10.
hbd	Numeric. Maximum number of H-bond donors. Default 5.
violations	Integer. Maximum tolerated Lipinski violations. Default 0.

Details

Uses the generic `LogP` column with the original threshold of 5.

Value

A data.frame with the rows of `data` that satisfy all thresholds.

References

Lipinski, C. A., Lombardo, F., Dominy, B. W., & Feeney, P. J. (1997). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23(1-3), 3-25.

Examples

```
d <- data.frame(MW = 300, LogP = 2, "#H-bond acceptors" = 4,
  "#H-bond donors" = 2, "Lipinski #violations" = 0, check.names = FALSE)
lipinskiFilter(d)
```

`mapADMETColumns` *Map user columns to standard ADMET schema*

Description

Takes a raw data.frame and a user-specified column mapping, renames columns to the application's standard schema, converts types, optionally calculates missing descriptors with CDK, and computes violation columns and ADMET properties.

Usage

```
mapADMETColumns(data, mapping, calculate_cdk = TRUE)
```

Arguments

`data`	A data.frame as uploaded by the user (CSV or Excel).
`mapping`	A named character vector where names are user column names and values are standard field names. Use "None" for columns to skip. Example: c("mol_weight" = "MW", "logp" = "LogP", "smiles" = "SMILES")
`calculate_cdk`	Logical. If TRUE and SMILES column is available, calculate missing descriptors (MW, LogP, TPSA, HBD, HBA, RB, MR, HeavyAtoms, AromAtoms) using CDK. Default TRUE.

Details

The mapping argument is a named character vector where each name is the name of a column in data and each value is one of the standard short field codes used by the ADMET Master Manager: "None", "SMILES", "Name", "MW", "LogP", "WLOGP", "TPSA", "HBD", "HBA", "Rotatable Bonds", "Molar Refractivity", "Heavy Atoms", "Aromatic Heavy Atoms", "GI Absorption", "GI Absorption_num", "BBB Permeant", "BBB Permeant_num", "Pgp Substrate", "Pgp Substrate_num", "LogS", "LogD".

Value

A data.frame with standardized column names, violation columns, and ADMET properties.

See Also

computeViolationColumns, computeADMETProperties, calcCDKDescriptors.

---

mapCDKDescriptors *Map CDK descriptors to the application's standard schema*

Description

Translates the raw CDK descriptor names into the application's canonical column names and computes the drug-likeness violation columns and the BOILED-Egg ADMET properties (GI absorption, BBB permeability, P-gp substrate).

Usage

```
mapCDKDescriptors(cdk_df)
```

Arguments

cdk_df	A data.frame as returned by calcCDKDescriptors.

Details

The CDK ALogP descriptor is mapped to the generic LogP column. No artificial MLOGP/WLOGP/XLOGP3 columns are created; the application uses a single LogP column for all drug-likeness filters, with thresholds from the original publications.

Value

A data.frame with renamed columns, violation columns and ADMET properties.

See Also

calcCDKDescriptors, computeViolationColumns, computeADMETProperties.

Examples

```
smiles <- c("CCO", "CC(=O)Oc1ccccc1C(=O)O")
desc <- calcCDKDescriptors(smiles)
mapped <- mapCDKDescriptors(desc)
```

`mueggeFilter` *Muegge drug-likeness filter*

Description

Filters compounds according to the Muegge pharmacophore-point filter (Muegge et al., 2001): MW, LogP, HBA, HBD, TPSA and rotatable bonds.

Usage

```
mueggeFilter(
  data,
  m_mw_min = 200,
  m_mw_max = 600,
  m_logp_min = -2,
  m_logp_max = 5,
  m_hba = 10,
  m_hbd = 5,
  m_rb = 15,
  m_tpsa = 150,
  violations = 0
)
```

Arguments

`data`	A data.frame with the standard column schema.
`m_mw_min`	Numeric. Minimum molecular weight. Default 200.
`m_mw_max`	Numeric. Maximum molecular weight. Default 600.
`m_logp_min`	Numeric. Minimum LogP. Default -2.
`m_logp_max`	Numeric. Maximum LogP. Default 5.
`m_hba`	Numeric. Maximum H-bond acceptors. Default 10.
`m_hbd`	Numeric. Maximum H-bond donors. Default 5.
`m_rb`	Numeric. Maximum rotatable bonds. Default 15.
`m_tpsa`	Numeric. Maximum TPSA. Default 150.
`violations`	Integer. Maximum tolerated Muegge violations. Default 0.

Value

A data.frame with the rows of `data` that satisfy all thresholds.

References

Muegge, I., Heald, S. L., & Brittelli, D. (2001). Simple selection criteria for drug-like chemical matter. *Journal of Medicinal Chemistry*, 44(12), 1841-1846.

Examples

```
d <- data.frame(MW = 300, LogP = 2, TPSA = 80,
  "#H-bond acceptors" = 4, "#H-bond donors" = 2,
  "#Rotatable bonds" = 3, "Muegge #violations" = 0, check.names = FALSE)
mueggeFilter(d)
```

---

palette_selector_ui *Colour palette selector UI element*

Description

A Shiny `selectInput` that lets the user choose a colour palette for the plots produced by the application.

Usage

```
palette_selector_ui(id)
```

Arguments

`id`	Character. Input id to use for the select input.

Value

A `shiny.tag` (select input).

Examples

```
library(shiny)
palette_selector_ui("my_palette")
```

`plotBoiledEgg` *BOILED-Egg plot*

Description

Produces the BOILED-Egg model plot (Daina & Zoete, 2016) using the official polygon coordinates from the supplementary data (Data S3). Points are coloured by P-gp substrate status when a `"Pgp substrate"` column is available.

Usage

```
plotBoiledEgg(data, logp_source = NULL)
```

Arguments

`data`	A data.frame that must contain the columns `LogP` and `TPSA`. A `"Pgp substrate"` column (with values "Yes"/"No") is optional and used for point colouring.
`logp_source`	Character. Which LogP variant is being used for the BOILED-Egg. If `"WLOGP"`, use the official WLOGP polygons from Daina & Zoete (2016). If any other value (or `NULL`), use the ALogP-trained polygons. Defaults to auto-detection: if the data has a `WLOGP` column, WLOGP polygons are used; otherwise ALogP.

Details

The BOILED-Egg model was originally calibrated with WLOGP; the application uses the generic `LogP` column as an approximation. The axes are TPSA on the x-axis and LogP on the y-axis.

Value

A **ggplot2** object.

References

Daina, A., & Zoete, V. (2016). A boiled egg to predict gastrointestinal absorption and brain penetration of small molecules. *ChemMedChem*, 11(11), 1117-1121.

Examples

```
d <- data.frame(LogP = c(2, 5), TPSA = c(50, 150))
plotBoiledEgg(d)
```

`plotClusterHeatmap` *Cluster heatmap with dendrogram*

Description

Produces a heatmap of physicochemical properties with a dendrogram coupled to the left side (compounds) showing hierarchical clustering. The properties (columns) are also clustered and the column dendrogram appears on top. Values are z-score standardized per property.

Usage

```
plotClusterHeatmap(
  data,
  variables = NULL,
  id_col = NULL,
  method = "ward.D2",
  scale_data = TRUE,
  palette = "default"
)
```

Arguments

`data`	A data.frame with physicochemical property columns.
`variables`	Character vector of numeric column names to use. If `NULL`, a default set of common descriptors is used.
`id_col`	Character. Name of the column to use as row labels. If `NULL` or `"None"`, row numbers are used.
`method`	Character. Agglomeration method for hierarchical clustering: one of `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"` (= UPGMA), `"mcquitty"`, `"median"`, `"centroid"`. Default `"ward.D2"`.
`scale_data`	Logical. Whether to z-score standardize each property before clustering. Default `TRUE`.
`palette`	Character. Colour palette name: one of `"default"`, `"viridis"`, `"magma"`, `"inferno"`, `"RdBu"`, `"Set1"`, `"Set2"`. Default `"default"` (red-blue).

Details

The plot is rendered using pheatmap package graphics so it works without additional dependencies beyond what the package already requires.

Value

Invisible `NULL`; called for the side-effect of drawing the plot on the current graphics device.

References

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1), 86-97.

Examples

```
d <- data.frame(
  Name = c("A", "B", "C", "D"),
  MW = c(300, 350, 600, 320),
  LogP = c(2, 3, 5, 1),
  TPSA = c(50, 60, 120, 40),
  check.names = FALSE
)
plotClusterHeatmap(d, id_col = "Name")
```

---

plotCorrHeatmap *Correlation heatmap of physicochemical properties*

Description

Produces a heatmap of the Pearson correlation between key physicochemical properties.

Usage

```
plotCorrHeatmap(
  data,
  props = c("MW", "LogP", "TPSA", "MR", "#H-bond acceptors", "#H-bond donors",
    "#Rotatable bonds", "#Heavy atoms")
)
```

Arguments

data	A data.frame with numeric property columns.
props	Character vector of property column names. Defaults to a set of common descriptors.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500), LogP = c(2, 3, 4),
  TPSA = c(60, 80, 100))
plotCorrHeatmap(d)
```

plotHistogramCustom *Custom histogram*

Description

Produces a highly customizable histogram of any numeric column in the dataset, with optional grouping (overlaid), kernel density overlay, rug plot, and configurable number of bins. This complements the pre-defined `plotMW`, `plotTPSA` and `plotLogP` histograms by allowing the user to pick any numeric variable (including user-supplied columns from CSV uploads, ADMET probabilities, etc.).

Usage

```
plotHistogramCustom(
  data,
  variable,
  bins = 30,
  group_by = "None",
  show_density = TRUE,
  show_rug = FALSE
)
```

Arguments

`data`	A data.frame.
`variable`	Character. Name of the numeric variable to histogram.
`bins`	Integer. Number of bins. Default 30.
`group_by`	Character. Optional categorical column for overlaid groups, or `"None"`.
`show_density`	Logical. Overlay a kernel density estimate. Default `TRUE`.
`show_rug`	Logical. Show a rug plot at the bottom. Default `FALSE`.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500, 350, 450, 380),
  group = c("A", "A", "B", "B", "C", "C"))
plotHistogramCustom(d, variable = "MW", group_by = "group")
```

plotLogP *LogP distribution histogram*

Description

LogP distribution histogram

Usage

```
plotLogP(data)
```

Arguments

data	A data.frame containing a numeric LogP column.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(LogP = c(1, 2, 3))
plotLogP(d)
```

plotMW *Molecular weight distribution histogram*

Description

Molecular weight distribution histogram

Usage

```
plotMW(data)
```

Arguments

data	A data.frame containing a numeric MW column.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500))
plotMW(d)
```

plotParallel *Parallel coordinates plot*

Description

Produces a parallel coordinates plot of the selected numeric variables, optionally coloured by a grouping variable. Requires the suggested package **GGally**.

Usage

```
plotParallel(data, variables, color_by = NULL, scale_data = TRUE)
```

Arguments

data	A data.frame with numeric property columns.
variables	Character vector of numeric column names to display.
color_by	Character. Column to colour lines by, or NULL/ "None" for a single colour.
scale_data	Logical. Whether to standardize variables. Default TRUE.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500), LogP = c(2, 3, 4),
  TPSA = c(60, 80, 100), group = c("A", "A", "B"))
plotParallel(d, variables = c("MW", "LogP", "TPSA"), color_by = "group")
```

plotPCA *Principal Component Analysis (PCA) chemical space plot*

Description

Computes a PCA on the selected numeric physicochemical variables and produces a scatterplot of the first two principal components, optionally coloured by a grouping variable, with 95% confidence ellipses, observation labels and variable loading arrows. Labels require the suggested package **ggrepel**.

Usage

```
plotPCA(
  data,
  variables = NULL,
  color_by = "None",
  label_by = "None",
  scale_data = TRUE,
  ellipse = TRUE
)
```

Arguments

data	A data.frame with numeric property columns.
variables	Character vector of numeric column names to use in the PCA. If NULL, a default set of common descriptors is used.
color_by	Character. Name of the column to colour points by, or "None" for a single colour.
label_by	Character. Name of the column to use as point labels, or "None" for no labels.
scale_data	Logical. Whether to scale variables to unit variance before the PCA. Default TRUE.
ellipse	Logical. Whether to draw 95% confidence ellipses per group. Default TRUE.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500), LogP = c(2, 3, 4),
  TPSA = c(60, 80, 100), group = c("A", "A", "B"))
plotPCA(d, variables = c("MW", "LogP", "TPSA"), color_by = "group")
```

---

plotRadar *Radar plot of physicochemical profile*

Description

Compares up to 15 molecules simultaneously on a radar chart using 6 key physicochemical properties, normalized 0-1 against the range observed in the full dataset. Requires the suggested package **fmsb**.

Usage

```
plotRadar(
  data,
  id_col,
  ids,
  props = c("MW", "LogP", "TPSA", "#H-bond acceptors", "#H-bond donors",
    "#Rotatable bonds")
)
```

Arguments

data	A data.frame with the physicochemical properties.
id_col	Character. Name of the identifier column.
ids	Character vector of molecule identifiers to compare (max 5 recommended).
props	Character vector of property column names to display. Defaults to MW, LogP, TPSA, #H-bond acceptors, #H-bond donors,#Rotatable bonds.

Value

Invisibly NULL; called for the side-effect of drawing the radar chart on the current graphics device.

References

Nakazawa, M. (2019). *fmsb: Functions for Medical Statistics Book with some Demographic Data*. R package.

Examples

```
d <- data.frame(
  Name = c("mol1", "mol2"),
  MW = c(300, 400), LogP = c(2, 3), TPSA = c(60, 90),
  "#H-bond acceptors" = c(4, 5), "#H-bond donors" = c(2, 1),
  "#Rotatable bonds" = c(3, 5), check.names = FALSE)
plotRadar(d, id_col = "Name", ids = c("mol1", "mol2"))
```

---

plotTanimoto *Tanimoto / AGNES structural similarity dendrogram*

Description

Parses SMILES strings, computes extended fingerprints, builds a Tanimoto similarity matrix and clusters the molecules with AGNES. Requires the suggested packages **rcdk**, **fingerprint** and **cluster**, plus a working Java JDK (for **rcdk**).

Usage

```
plotTanimoto(
  data,
  smiles_col,
  label_col = NULL,
  max_n = 40,
  method = "average"
)
```

Arguments

data	A data.frame containing a SMILES column.
smiles_col	Character. Name of the SMILES column.
label_col	Character. Name of the column to use as dendrogram labels, or NULL to use the SMILES themselves.
max_n	Integer. Maximum number of molecules to include. Default 40.
method	Character. AGNES linking method: one of "average", "complete", "single", "ward". Default "average".

Value

Invisibly the Tanimoto similarity matrix.

References

Willett, P., Barnard, J. M., & Downs, G. M. (1998). Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, 38(6), 983-996.

Examples

```
d <- data.frame(
  Name = c("ethanol", "aspirin", "benzene", "toluene", "phenol"),
  SMILES = c("CCO", "CC(=O)Oc1ccccc1C(=O)O", "c1ccccc1",
             "Cc1ccccc1", "Oc1ccccc1"),
  stringsAsFactors = FALSE)
plotTanimoto(d, smiles_col = "SMILES", label_col = "Name")
```

---

plotTPSA *TPSA distribution histogram*

Description

TPSA distribution histogram

Usage

```
plotTPSA(data)
```

Arguments

data	A data.frame containing a numeric TPSA column.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(TPSA = c(40, 80, 120))
plotTPSA(d)
```

---

plotTSNE *t-SNE chemical space plot*

Description

Computes a 2-dimensional t-SNE embedding of the selected numeric physicochemical variables and produces a scatterplot, optionally coloured and labelled. Requires the suggested packages **Rtsne** and **ggrepel**.

Usage

```
plotTSNE(
  data,
  variables,
  color_by = "None",
  label_by = "None",
  perplexity = 30,
  max_iter = 1000,
  scale_data = TRUE
)
```

Arguments

data	A data.frame with numeric property columns.
variables	Character vector of numeric column names to use.
color_by	Character. Column to colour points by, or "None".
label_by	Character. Column to use as labels, or "None".
perplexity	Numeric. t-SNE perplexity. Default 30.
max_iter	Integer. Number of iterations. Default 1000.
scale_data	Logical. Whether to scale variables. Default TRUE.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(
  MW = rnorm(50, 400, 100),
  LogP = rnorm(50, 3, 1.5),
  TPSA = rnorm(50, 80, 40),
  group = rep(c("A", "B"), each = 25)
)
plotTSNE(d, variables = c("MW", "LogP", "TPSA"),
         color_by = "group", perplexity = 5)
```

`plotUMAP` *UMAP chemical space plot*

Description

Produces a 2D UMAP projection of the selected numeric variables for exploratory analysis of the chemical space. Points can be coloured and labelled. Requires the suggested packages **uwot** and **ggrepel**.

Usage

```
plotUMAP(
  data,
  variables,
  color_by = "None",
  label_by = "None",
  n_neighbors = 15,
  min_dist = 0.1,
  scale_data = TRUE
)
```

Arguments

`data`	A data.frame with numeric property columns.
`variables`	Character vector of numeric column names to use.
`color_by`	Character. Column to colour points by, or `"None"`.
`label_by`	Character. Column to use as labels, or `"None"`.
`n_neighbors`	Integer. Number of nearest neighbours used by UMAP. Default 15.
`min_dist`	Numeric. Minimum distance between points in the embedding. Default 0.1.
`scale_data`	Logical. Whether to scale variables. Default `TRUE`.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500,600), LogP = c(2, 3, 4,1),
  TPSA = c(60, 80, 100,40), group = c("A", "A", "B", "B"))
plotUMAP(d, variables = c("MW", "LogP", "TPSA"), color_by = "group")
```

---

plotViolin *Violin plot*

Description

Produces a violin plot of a numeric variable grouped by a categorical variable, with an optional inner boxplot and jittered points.

Usage

```
plotViolin(data, variable, group_by, show_box = TRUE, show_points = FALSE)
```

Arguments

data	A data.frame.
variable	Character. Name of the numeric variable to plot.
group_by	Character. Name of the categorical grouping variable.
show_box	Logical. Whether to overlay a boxplot. Default TRUE.
show_points	Logical. Whether to overlay jittered points. Default FALSE.

Value

A **ggplot2** object.

Examples

```
d <- data.frame(MW = c(300, 400, 500),
  group = c("A", "A", "B"))
plotViolin(d, variable = "MW", group_by = "group")
```

run_app *Launch the ADMETShiny application*

Description

Starts the interactive ADMETShiny Shiny application. The function registers the package's static web assets (the hex sticker) and returns a Shiny app object that can be printed or run directly.

Usage

```
run_app(onStart = NULL, ...)
```

Arguments

onStart	An optional function called once when the app starts (forwarded to shiny::shinyApp).
...	Additional arguments forwarded to shiny::shinyApp (e.g. options).

Value

A shiny.appobj object (invisibly); called for the side-effect of launching a Shiny application when printed.

Examples

```
if (interactive()) {
  admetshiny::run_app()
}
```

veberFilter *Veber drug-likeness filter*

Description

Filters compounds according to the Veber oral-bioavailability rule (Veber et al., 2002): rotatable bonds and a polarity condition (TPSA or HBA+HBD).

Usage

```
veberFilter(data, v_rb = 10, v_tpsa = 140, v_hb_sum = 12, violations = 0)
```

Arguments

`data`	A data.frame with the standard column schema.
`v_rb`	Numeric. Maximum rotatable bonds. Default 10.
`v_tpsa`	Numeric. Maximum TPSA. Default 140.
`v_hb_sum`	Numeric. Maximum H-bond acceptors + donors. Default 12.
`violations`	Integer. Maximum tolerated Veber violations. Default 0.

Value

A data.frame with the rows of `data` that satisfy all thresholds.

References

Veber, D. F., Johnson, S. R., Cheng, H. Y., Smith, B. R., Ward, K. W., & Kopple, K. D. (2002). Molecular properties that influence the oral bioavailability of drug candidates. *Journal of Medicinal Chemistry*, 45(12), 2615-2623.

Examples

```
d <- data.frame(TPSA = 80, "#Rotatable bonds" = 3,
  "#H-bond acceptors" = 4, "#H-bond donors" = 2,
  "Veber #violations" = 0, check.names = FALSE)
veberFilter(d)
```

Index

admetshiny *(admetshiny-package)*, 3
admetshiny-package, 3
apply_palette, *4*, 6
applyFilters, *3*, 5

calcCDKDescriptors, *3*, 6, *12*, *15*
computeADMETProperties, *3*, 7, *15*
computeViolationColumns, *3*, 8, *15*

eganFilter, *3*, *5*, *9*, 9

generateReport, *4*, 10
getSmilesFromIdentifiers, *3*, *7*, 11
ghoseFilter, *3*, *5*, *9*, 12

lipinskiFilter, *3*, *5*, *9*, 13

mapADMETColumns, *3*, *13*, 14
mapCDKDescriptors, *3*, *7*, *13*, 15
mueggeFilter, *3*, *5*, *9*, 16

palette_selector_ui, *4*, *6*, 17
plotBoiledEgg, *4*, 18
plotClusterHeatmap, *4*, 19
plotCorrHeatmap, *4*, 20
plotHistogramCustom, *4*, 21
plotLogP, *4*, 22
plotMW, *4*, 22
plotParallel, *4*, 23
plotPCA, *4*, 23
plotRadar, *4*, 24
plotTanimoto, *4*, 25
plotTPSA, *4*, 26
plotTSNE, *4*, 27
plotUMAP, *4*, 28
plotViolin, *4*, 29

run_app, *4*, 30

shiny::shinyApp, *30*

veberFilter, *3*, *5*, *9*, 30