Package ‘rdkitpyr’

March 5, 2026

Type Package

Title Task-Oriented Cheminformatics in R Using 'RDKit' via 'Python'

Version 0.2.1

Description A task-oriented R interface to the 'RDKit' <https://www.rdkit.org> library through its 'Python' API via 'reticulate'. The package offers high-level cheminformatics functionality, including molecule parsing, descriptor calculation, and fingerprint generation without replicating the native structure of 'RDKit'.

License MIT + file LICENSE

URL https://mass-spec.ru/projects/cheminformatics/rdkitpyr/eng/

BugReports https://github.com/AndreySamokhin/rdkitpyr/issues

Depends R (>= 3.5.0)

Imports reticulate, utils

Suggests testthat (>= 3.0.0)

SystemRequirements Python, NumPy, RDKit (Python)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Andrey Samokhin [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-0223-6087>)

Maintainer Andrey Samokhin <andrey.s.samokhin@gmail.com>

Repository CRAN

Date/Publication 2026-03-05 19:30:02 UTC

Contents

`.IsRdkitAvailable` *Check whether RDKit is available*

Description

Used in examples and tests to determine whether the Python module `rdkit` is available via **reticulate**.

This function is exported to support examples and tests. It is not part of the stable user-facing API and may change without notice.

The result is cached for the duration of the R session to avoid repeated calls to `reticulate::py_module_available()`.

Usage

```
.IsRdkitAvailable(initialize = TRUE)
```

Arguments

`initialize`	A logical value. If `FALSE` and Python is not initialized, RDKit is assumed to be unavailable and the corresponding tests and examples are skipped.

Value

A logical value indicating whether the `rdkit` Python module is available.

CalculateAllDescriptors

Calculate all RDKit molecular descriptors

Description

Calculate all molecular descriptors available in RDKit for a set of molecules.

The descriptors are calculated using the `CalcMolDescriptors()` function from the `rdkit.Chem.Descriptors` module in RDKit.

The set of returned descriptors may depend on the installed RDKit version.

Each molecule is represented by a full set of descriptor values returned as a data frame. Invalid molecules are represented by rows containing `NA` values. Row order is preserved so that the output aligns with the input.

Usage

```
CalculateAllDescriptors(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A data frame with one row per molecule and one column per descriptor. Elements corresponding to invalid molecules are returned as `NA`. Original RDKit descriptor names are used.

Additionally, the `"valid"` attribute is attached to indicate which molecules were successfully processed.

Examples

```
# Calculate all RDKit descriptors
smiles <- c("CCO", "c1ccccc1", "invalid_molecule")
desc <- CalculateAllDescriptors(smiles)

# Inspect names of first three descriptors
names(desc)[1:3]
#> "MaxAbsEStateIndex" "MaxEStateIndex" "MinAbsEStateIndex"

# Display Molecular weight, LogP, TPSA, and molar refractivity
desc[c("MolWt", "MolLogP", "TPSA", "MolMR")]
#>    MolWt MolLogP  TPSA   MolMR
#> 1 46.069 -0.0014 20.23 12.7598
#> 2 78.114  1.6866  0.00 26.4420
#> 3     NA      NA    NA      NA
```

```
# Check which molecules were successfully processed
attr(desc, "valid")
#> TRUE TRUE FALSE
```

`CalculateExactMass` *Calculate the exact mass of molecules*

Description

Calculate the exact mass (monoisotopic mass) for a set of molecules.

The calculation is performed using the `ExactMolWt()` function from the `rdkit.Chem.Descriptors` module in RDKit.

Usage

```
CalculateExactMass(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A numeric vector containing the exact mass for each molecule. Elements corresponding to invalid molecules are returned as `NA`.

Examples

```
# Calculate exact mass for a set of molecules
smiles <- c("CCO", "c1ccccc1", "invalid_molecule")
CalculateExactMass(smiles)
#> 46.04186 78.04695 NA
```

`CalculateMaccsFingerprints` *Calculate MACCS fingerprints for a set of molecules*

Description

Calculate MACCS (Molecular ACCess System) fingerprints for a set of molecules. Each fingerprint is a fixed-length binary vector representing the presence or absence of predefined structural features.

The fingerprints are calculated using the `GetMACCSKeysFingerprint()` function from the `rdkit.Chem.rdMolDescriptors` module in RDKit.

Invalid molecules are represented by `NA` vectors. Row order is preserved so that the output aligns with the input.

Usage

```
CalculateMaccsFingerprints(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A matrix of integers (0 or 1) with one row per molecule and 167 columns (MACCS keys). Rows corresponding to invalid molecules contain `NA`.

Additionally, the `"valid"` attribute is attached to indicate which molecules were successfully processed.

Examples

```
# Calculate MACCS fingerprints
smiles <- c("CCO", "c1ccccc1", "invalid_molecule")
fps <- CalculateMaccsFingerprints(smiles)

# Get the number of fingerprints (columns)
ncol(fps)
#> 167

# Check which molecules were successfully processed
attr(fps, "valid")
#> TRUE TRUE FALSE
```

CalculateMolecularWeight

Calculate the average molecular weight of molecules

Description

Calculate the molecular weight (average mass) for a set of molecules.

The calculation is performed using the `MolWt()` function from the `rdkit.Chem.Descriptors` module in RDKit.

Usage

```
CalculateMolecularWeight(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A numeric vector containing the molecular weight for each molecule. Elements corresponding to invalid molecules are returned as `NA`.

Examples

```
# Calculate average molecular weight for a set of molecules
smiles <- c("CCO", "c1ccccc1", "invalid_molecule")
CalculateMolecularWeight(smiles)
#> 46.069 78.114 NA
```

CalculateMorganFingerprints

Calculate Morgan fingerprints for a set of molecules

Description

Calculate Morgan (circular) fingerprints for a set of molecules.

The fingerprints are calculated using the `GetMorganGenerator()` function from the `rdkit.Chem.rdFingerprintGenerator` module in RDKit.

Invalid molecules are represented by `NA` vectors. Row order is preserved so that the output aligns with the input.

Usage

```
CalculateMorganFingerprints(
  mols,
  radius = 3L,
  fp_size = 2048L,
  count_simulation = FALSE,
  include_chirality = FALSE,
  use_bond_types = TRUE,
  include_ring_membership = TRUE,
  verbose = FALSE
)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`radius`	An integer value. Bond radius defining the size of circular substructures.
`fp_size`	An integer value. Number of bits in the fingerprint.
`count_simulation`	A logical value. If set, use count simulation while generating the fingerprint.
`include_chirality`	A logical value. If set, chirality information will be added to the generated fingerprint.
`use_bond_types`	A logical value. If set, bond types will be included as a part of the default bond invariants.
`include_ring_membership`	A logical value. If set, whether or not the atom is in a ring will be used in the invariant list.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A matrix of integers (0 or 1) with one row per molecule and `fp_size` columns. Rows corresponding to invalid molecules contain `NA`.

Additionally, the `"valid"` attribute is attached to indicate which molecules were successfully processed.

Examples

```
# Calculate Morgan fingerprints
smiles <- c("CCO", "c1ccccc1", "invalid_molecule")
fps <- CalculateMorganFingerprints(smiles)

# Get the number of fingerprints (columns)
ncol(fps)
#> 2048
```

```
# Check which molecules were successfully processed
attr(fps, "valid")
#> TRUE TRUE FALSE
```

`CalculateRdkitFingerprints` *Calculate RDKit topological fingerprints for a set of molecules*

Description

Calculate RDKit topological (path-based) fingerprints for a set of molecules.

The fingerprints are calculated using the `RDKFingerprint()` function from the `rdkit.Chem.rdmolops` module in RDKit.

Invalid molecules are represented by `NA` vectors. Row order is preserved so that the output aligns with the input.

Usage

```
CalculateRdkitFingerprints(
  mols,
  min_path = 1L,
  max_path = 7L,
  fp_size = 2048L,
  n_bits_per_hash = 2L,
  use_hydrogens = TRUE,
  target_density = 0,
  min_size = 128L,
  branched_paths = TRUE,
  use_bond_order = TRUE,
  verbose = FALSE
)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`min_path`	An integer value. Minimum number of bonds to include in the subgraphs.
`max_path`	An integer value. Maximum number of bonds to include in the subgraphs.
`fp_size`	An integer value. Number of bits in the fingerprint.
`n_bits_per_hash`	An integer value. Number of bits to set per path.
`use_hydrogens`	A logical value. Include paths involving hydrogens in the fingerprint if the molecule has explicit hydrogens.
`target_density`	A numeric value. Fold the fingerprint until this minimum density has been reached.

`min_size`	An integer value. The minimum size the fingerprint will be folded to when trying to reach `target_density`.
`branched_paths`	A logical value. If set, both branched and unbranched paths will be used in the fingerprint.
`use_bond_order`	A logical value. If set, both bond orders will be used in the path hashes.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A matrix of integers (0 or 1) with one row per molecule and `fp_size` columns. Rows corresponding to invalid molecules contain `NA`.

Additionally, the `"valid"` attribute is attached to indicate which molecules were successfully processed.

Examples

```
# Calculate RDKit fingerprints
smiles <- c("CCO", "c1ccccc1", "invalid_molecule")
fps <- CalculateRdkitFingerprints(smiles)

# Get the number of fingerprints (columns)
ncol(fps)
#> 2048

# Check which molecules were successfully processed
attr(fps, "valid")
#> TRUE TRUE FALSE
```

`ConvertToInchi` *Convert molecules to InChI strings*

Description

Convert molecules to InChI strings.

InChI identifiers can technically be provided as input. In this case, the output is expected to be identical to the input. This can be useful only to test RDKit's consistency with challenging molecules.

Usage

```
ConvertToInchi(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A character vector. InChI strings. Elements that cannot be converted are returned as `NA`.

Examples

```
# Convert a vector of SMILES to InChI identifiers
smiles <- c("CC", "CCC")
ConvertToInchi(smiles)
#> "InChI=1S/C2H6/c1-2/h1-2H3"
#> "InChI=1S/C3H8/c1-3-2/h3H2,1-2H3"

# Providing InChI as input returns identical output
ConvertToInchi("InChI=1S/CH4/h1H4")
#> "InChI=1S/CH4/h1H4"
```

`ConvertToInchikey` *Convert molecules to InChIKey strings*

Description

Convert molecules to InChIKey strings.

Conversion of an InChI string to an InChIKey relies on the IUPAC library, allowing conversion without creating intermediate RDKit Mol objects.

Usage

```
ConvertToInchikey(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A character vector. InChIKey strings. Elements that cannot be converted are returned as `NA`.

Examples

```
# Convert a vector of InChI to InChIKey identifiers
inchi <- c("InChI=1S/C2H6/c1-2/h1-2H3",
           "InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H")
ConvertToInchikey(inchi)
#> "OTMSDBZUPAUEDD-UHFFFAOYSA-N"
#> "UHOVQNZJYSORNB-UHFFFAOYSA-N"
```

```
# Convert a vector of SMILES to InChIKey identifiers
smiles <- c("CC", "c1ccccc1")
ConvertToInchikey(smiles)
#> "OTMSDBZUPAUEDD-UHFFFAOYSA-N"
#> "UHOVQNZJYSORNB-UHFFFAOYSA-N"
```

`ConvertToSmiles` *Convert molecules to SMILES strings*

Description

Convert molecules to SMILES strings.

SMILES strings can be provided as input to obtain their canonical form or to remove stereochemistry.

Usage

```
ConvertToSmiles(
  mols,
  isomeric = TRUE,
  kekule = FALSE,
  canonical = TRUE,
  explicit_bonds = FALSE,
  explicit_hydrogens = FALSE,
  verbose = FALSE
)
```

Arguments

`mols`	A character vector of SMILES or InChI strings, or a list of RDKit Mol objects.
`isomeric`	A logical value. If `TRUE`, include stereochemistry in the output.
`kekule`	A logical value. If `TRUE`, kekulize aromatic bonds.
`canonical`	A logical value. If `TRUE`, generate canonical SMILES.
`explicit_bonds`	A logical value. If `TRUE`, make all bonds explicit.
`explicit_hydrogens`	A logical value. If `TRUE`, make all hydrogens explicit.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A character vector. SMILES strings. Elements that cannot be converted are returned as `NA`.

Examples

```
# Convert a vector of InChI identifiers to canonical SMILES
inchi <- c("InChI=1S/C2H6/c1-2/h1-2H3",
           "InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H")
ConvertToSmiles(inchi)
#> "CC"
#> "c1ccccc1"

# Convert a vector of SMILES to SMILES with Kekulized aromatic bonds
smiles <- c("c1ccccc1", "c1ccc2ccccc2c1")
ConvertToSmiles(smiles, kekule = TRUE)
#> "C1=CC=CC=C1"
#> "C1=CC=C2C=CC=CC2=C1"
```

`GetPythonInfo` *Get information about the currently used Python environment*

Description

Get details about the Python interpreter, `numpy` and `rdkit` packages.

Usage

```
GetPythonInfo(verbose = TRUE)
```

Arguments

`verbose`	A logical value. If `TRUE` (default), function prints information about currently used Python environemt.

Value

Invisibly return a named list with the following components:

python_path Full path to the Python executable.

python_version Version of the Python interpreter.

numpy_version Installed version of the `numpy` package.

rdkit_version Installed version of the `rdkit` package.

forced_by If applicable, indicates whether the Python interpreter was forced via `RETICULATE_PYTHON`, `use_*()`, or `py_require()`.

Examples

```
# Print information about the Python environment
GetPythonInfo()

# Access programmatically
py_env <- GetPythonInfo(verbose = FALSE)
py_env$python_version
py_env$rdkit_version
```

`ParseMolecules` *Parse SMILES and InChI strings into RDKit Mol objects*

Description

Parse SMILES and InChI strings into RDKit Mol objects

This function converts a character vector of molecular representations (SMILES or InChI) into a list of RDKit Mol objects (Python-backed pointers via `reticulate`). The resulting objects can be reused in subsequent operations without repeated conversion from SMILES or InChI. This is particularly useful when multiple cheminformatics tasks are performed on the same set of molecules, improving efficiency by avoiding repeated parsing steps.

Usage

```
ParseMolecules(mols, verbose = FALSE)
```

Arguments

`mols`	A character vector of SMILES or InChI strings.
`verbose`	A logical value. If `TRUE`, display warning and error messages emitted by the RDKit backend. Default is `FALSE`.

Value

A list of RDKit Mol objects.

Examples

```
# Convert a vector of SMILES to RDKit Mol objects
mols <- ParseMolecules(c("CC", "CCC"))
print(mols[[1L]])
#> <rdkit.Chem.rdchem.Mol object at 0x000001CC4D60F4C0>

# Convert a list of RDKit Mol objects to InChI identifiers
ConvertToInchi(mols)
#> "InChI=1S/C2H6/c1-2/h1-2H3"
#> "InChI=1S/C3H8/c1-3-2/h3H2,1-2H3"
```

`test_compounds` *Test compounds*

Description

A small reference set of chemical compounds extracted from the PubChem database. Despite its limited size, the dataset covers a broad range of chemical properties, including:

- neutral, charged, and radical species;
- aromatic and aliphatic compounds;
- molecules containing heteroatoms;
- isotopically labeled compounds;
- stereochemistry;
- species with disconnected fragments.

Usage

```
test_compounds
```

Format

A data frame.

`pubchem_cid`	PubChem compound identifier (CID).
`name`	Compound name (as reported in the PubChem database).
`smiles`	SMILES identifier.
`inchi`	InChI identifier.
`inchikey`	InChIKey identifier.
`n_labeled_atoms`	Number of isotopically labeled atoms.
`charge`	The total charge of a molecule.
`exact_mass`	Exact monoisotopic mass.
`molecular_weight`	Average molecular weight.
`formula`	Molecular formula.
`formula_isotopes`	Molecular formula with explicit isotopes.

Index