# Package 'raymolecule'

February 22, 2024

**Type** Package

**Title** Parse and Render Molecular Structures in 3D

**Version** 0.5.3

**Date** 2024-2-18

**Maintainer** Tyler Morgan-Wall <tylermw@gmail.com>

**Description** Downloads and parses 'SDF' (Structural Description Format) and 'PDB' (Protein Database) files for 3D rendering.

**License** GPL-3

**Encoding** UTF-8

**Imports** rayrender (>= 0.31.2), magrittr, PeriodicTable, httr, rayvertex (>= 0.10.4)

**RoxygenNote** 7.3.0

**URL** http://www.raymolecule.com/, https://github.com/tylermorganwall/raymolecule

**BugReports** https://github.com/tylermorganwall/raymolecule/issues

**NeedsCompilation** no

**Author** Tyler Morgan-Wall [aut, cph, cre] (<https://orcid.org/0000-0002-3131-3814>)

**Repository** CRAN

**Date/Publication** 2024-02-22 00:10:02 UTC

## R topics documented:

---

`generate_atom_scene` *Build Scene (atoms only)*

---

## Description

Reads an SDF file and extracts the 3D molecule model

## Usage

```
generate_atom_scene(
  model,
  x = 0,
  y = 0,
  z = 0,
  scale = 1,
  center = TRUE,
  pathtrace = TRUE,
  material = rayrender::glossy,
  material_vertex = material_list(type = "phong")
)
```

## Arguments

| | |
|---|---|
| `model` | Model extracted from a PDB or SDF file. |
| `x` | Default ‘0‘. X offset, applied after centering. |
| `y` | Default ‘0‘. Y offset, applied after centering. |
| `z` | Default ‘0‘. Z offset, applied after centering. |
| `scale` | Default ‘1‘. Amount to scale the inter-atom spacing. |
| `center` | Default ‘TRUE‘. Centers the bounding box of the model. |
| `pathtrace` | Default ‘TRUE‘. If ‘FALSE‘, the ‘rayvertex‘ package will be used to render the scene. |
| `material` | Default ‘rayrender::glossy‘. Rayrender material to use when ‘pathtrace = TRUE‘. Must be either ‘glossy‘, ‘diffuse‘, or ‘dielectric‘. |
| `material_vertex` | Default ‘rayvertex::material_list()‘. Material to use when ‘pathtrace = FALSE‘. ‘diffuse‘/‘ambient‘ colors and ‘ambient_intensity‘ are determined automatically, but all other material properties can be changed. |

## Value

Rayrender/rayvertex scene containing only the atoms in a molecule/protein.

## Examples

```
#Generate a scene with caffeine molecule with just the atoms

get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_atom_scene() %>%
  render_model(samples=256,sample_method="sobol_blue")

#Generate a rayvertex scene, using toon shading
shiny_toon_material = rayvertex::material_list(type="toon_phong",
                                               toon_levels=3,
                                               toon_outline_width=0.1)
get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_atom_scene(pathtrace=FALSE, material_vertex = shiny_toon_material) %>%
  render_model(background="white")

#Generate a scene with caffeine, reducing the inter-atom spacing
get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_atom_scene(scale=0.5) %>%
  render_model(samples=256,sample_method="sobol_blue")
```

---

# generate_bond_scene *Build Scene (bonds only)*

---

## Description

Reads an SDF file and extracts the 3D molecule model

## Usage

```
generate_bond_scene(
  model,
  x = 0,
  y = 0,
  z = 0,
  scale = 1,
  center = TRUE,
  force_single_bonds = FALSE,
  pathtrace = TRUE,
  material = rayrender::glossy,
 material_vertex = material_list(diffuse = "grey33", ambient = "grey33", type = "phong",
    ambient_intensity = 0.3)
)
```

## Arguments

| | |
|---|---|
| model | Model extracted from a PDB or SDF file. |
| x | Default '0'. X offset, applied after centering. |
| y | Default '0'. Y offset, applied after centering. |
| z | Default '0'. Z offset, applied after centering. |
| scale | Default '1'. Amount to scale the interatom spacing. |
| center | Default 'TRUE'. Centers the bounding box of the model. |
| force_single_bonds | Default 'FALSE'. Whether to force all bonds to show as a single connection. |
| pathtrace | Default 'TRUE'. If 'FALSE', the 'rayvertex' package will be used to render the scene. |
| material | Default 'rayrender::glossy'. Rayrender material to use when 'pathtrace = TRUE'. Must be either 'glossy', 'diffuse', or 'dielectric'. |
| material_vertex | Default 'material_list(diffuse="grey33",ambient="grey33",type="phong", ambient_intensity=0.3)'. Material to use for the bonds when 'pathtrace = FALSE'. |

## Value

Rayrender/rayvertex scene containing only the connections between atoms in a molecule/protein.

## Examples

```
#Generate a scene with benzene molecule with just the atoms

get_example_molecule("benzene") %>%
  read_sdf() %>%
  generate_bond_scene() %>%
  render_model(lights = "both", samples=256,sample_method="sobol_blue")

#Force single bonds to just show the shape of the molecule
get_example_molecule("benzene") %>%
  read_sdf() %>%
  generate_bond_scene(force_single_bonds = TRUE) %>%
  render_model(lights = "both", samples=256,sample_method="sobol_blue")

#Generate a scene with PFOA, reducing the inter-atom spacing
get_example_molecule("pfoa") %>%
  read_sdf() %>%
  generate_bond_scene(scale=0.3,force_single_bonds = TRUE) %>%
  render_model(lights = "both", samples=256,sample_method="sobol_blue")
```

`generate_full_scene` *Build Scene (bonds + atoms)*

## Description

Reads an SDF file and extracts the 3D molecule model

## Usage

```
generate_full_scene(
  model,
  x = 0,
  y = 0,
  z = 0,
  scale = 1,
  center = TRUE,
  pathtrace = TRUE,
  force_single_bonds = FALSE,
  material = rayrender::glossy,
  material_vertex = material_list(type = "phong")
)
```

## Arguments

| | |
|---|---|
| `model` | Model extracted from a PDB or SDF file. |
| `x` | Default ‘0‘. X offset, applied after centering. |
| `y` | Default ‘0‘. Y offset, applied after centering. |
| `z` | Default ‘0‘. Z offset, applied after centering. |
| `scale` | Default ‘1‘. Amount to scale the interatom spacing. |
| `center` | Default ‘TRUE‘. Centers the bounding box of the model. |
| `pathtrace` | Default ‘TRUE‘. If ‘FALSE‘, the ‘rayvertex‘ package will be used to render the scene. |
| `force_single_bonds` | Default ‘FALSE‘. Whether to force all bonds to show as a single connection. |
| `material` | Default ‘rayrender::glossy‘. Rayrender material to use when ‘pathtrace = TRUE‘. Must be either ‘glossy‘, ‘diffuse‘, or ‘dielectric‘. |
| `material_vertex` | Default ‘rayvertex::material_list()‘. Material to use when ‘pathtrace = FALSE‘. ‘diffuse‘/‘ambient‘ colors and ‘ambient_intensity‘ are determined automatically, but all other material properties can be changed. |

## Value

Rayrender/rayvertex scene

## Examples

```
# Generate a scene with caffeine molecule

get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_full_scene() %>%
  render_model(samples=256,sample_method="sobol_blue")

#Generate a rayvertex scene with a custom material
get_example_molecule("caffeine") %>%
  read_sdf() %>%
 generate_full_scene(pathtrace=FALSE, material_vertex=rayvertex::material_list(type="phong")) %>%
  render_model(background="grey33")

#Generate a rayvertex scene, using toon shading
shiny_toon_material = rayvertex::material_list(type="toon_phong",
                                               toon_levels=3,
                                               toon_outline_width=0.1)
get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_full_scene(pathtrace=FALSE, material_vertex=shiny_toon_material) %>%
  render_model(background="grey66")

# Generate a scene with morphine, increasing the inter-atom spacing
get_example_molecule("tubocurarine_chloride") %>%
  read_sdf() %>%
  generate_full_scene(scale=1.5) %>%
  render_model(samples=256,sample_method="sobol_blue")


# Force bonds to appear as a single link (to focus purely on the shape of the molecule)
get_example_molecule("tubocurarine_chloride") %>%
  read_sdf() %>%
  generate_full_scene(force_single_bonds = TRUE) %>%
  render_model(samples=256,sample_method="sobol_blue")
```

---

# get_example_molecule *Get Example Molecule*

---

## Description

Loads the structure of the built-in molecules. All SDF files obtained from Pubchem. txt extension only included to pass R CHECK.

## Usage

```
get_example_molecule(molecule)
```

### Arguments

| | |
|---|---|
| molecule | One of the built-in SDF files. These are "benzene", "buckyball", "caffeine", "capsaicin", "cinnemaldehyde", "geraniol", "luciferin", "morphine", "penicillin", "pfoa", "skatole", "tubocurarine_chloride". |

### Value

List giving the atom locations and the connections between atoms.

### Examples

```
get_example_molecule("benzene")
get_example_molecule("cinnemaldehyde")
get_example_molecule("geraniol")
```

---

## get_molecule *Get Molecule*

---

### Description

Loads the structure of a molecule by fetching an SDF file from Pubchem, which can be piped to generate_full_scene

### Usage

```
get_molecule(molecule)
```

### Arguments

| | |
|---|---|
| molecule | A character variable of a compound name or a numeric variable of an official compound ID |

### Value

List giving the atom locations and the connections between atoms.

### Examples

```
if(run_documentation()) {
get_molecule("caffeine") %>%
  generate_full_scene() %>%
  render_model()
}
if(run_documentation()) {
#estradiol (aka estrogen)
get_molecule(5757) %>%
  generate_full_scene() %>%
  render_model()
}
```

```
if(run_documentation()) {
get_molecule("testosterone") %>%
  generate_full_scene() %>%
  render_model()
}
if(run_documentation()) {
get_molecule("aspirin") %>%
  generate_full_scene() %>%
  render_model()
}
if(run_documentation()) {
get_molecule("rutoside") %>%
  generate_full_scene() %>%
  render_model()
}
if(run_documentation()) {
#If the 3D SDF doesn't exist, this function will pull the 2D SDF and inform the user
get_molecule("cyanocobalamin") %>%
  generate_full_scene() %>%
  render_model()
}
```

---

`read_pdb` *Read PDB File*

---

## Description

Reads an PDB file and extracts the atom locations and bonds (does not include any other structual information currently). This pulls out ATOM and HETAHM records by default, along with available connections.

## Usage

```
read_pdb(filename, atom = TRUE, nsr = TRUE)
```

## Arguments

| | |
|---|---|
| `filename` | Path to the PDB file. |
| `atom` | Default 'TRUE'. Whether to pull out standard residue (ATOM) records. |
| `nsr` | Default 'TRUE'. Whether to pull out nonstandard residue (HETAHM) records. |

## Value

List giving the atom locations.

## Examples

```
#This assumes a hypothetical PDB file in your working directory:
if(file.exists("3nir.pdb")) {
  read_pdb("3nir.pdb") %>%
    generate_full_scene() %>%
    render_model()
}
```

---

`read_sdf` *Read SDF File*

---

## Description

Reads an SDF file and extracts the 3D molecule model

## Usage

```
read_sdf(filename)
```

## Arguments

| | |
|---|---|
| `filename` | Filename to the sdf file. |

## Value

List giving the atom locations and the connections between atoms.

## Examples

```
#This assumes a hypothetical SDF file in your working directory:
if(file.exists("molecule.sdf")) {
  read_pdb("molecule.sdf") %>%
    generate_full_scene() %>%
    render_model()
}
```

---

`render_model` *Render Molecule Model*

---

## Description

Automatically plots the molecule with a camera position and field of view that includes the full model. For more control over the scene, pass the scene to ‘rayrender::render_scene()‘ or ‘rayvertex::rasterize_scene()‘ and specify the camera position manually. Note: spheres and cylinders in the scene are used to automatically compute the field of view of the scene–if rendering with rayrender, adding additional sphere (e.g. with ‘rayrender::generate_ground()‘) will change this calculation. Use ‘rayrender::render_scene()‘ instead if this is a problem.

## Usage

```
render_model(
  scene,
  fov = NULL,
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  lights = "top",
  lightintensity = 80,
  ...
)
```

## Arguments

| | |
|---|---|
| scene | 'rayrender' scene of molecule model. |
| fov | Default 'NULL', automatically calculated. Camera field of view. |
| angle | Default 'c(0,0,0)'. Degrees to rotate the model around the X, Y, and Z axes. If this is a single number, it will be taken as the Y axis rotation. |
| order_rotation | Default 'c(1,2,3)'. What order to apply the rotations specified in 'angle'. |
| lights | Default 'top'. If 'none', removes all lights. If 'bottom', lights scene with light underneath model. If 'both', adds lights both above and below model. This can also be a matrix of light information generated with 'rayvertex'. |
| lightintensity | Default '80'. Light intensity for pathtraced scenes. |
| ... | Other arguments to pass to 'rayrender::render_scene()' or 'rayvertex::rasterize_scene()' |

## Value

Rendered image

## Examples

```
# Generate a scene with caffeine molecule with just the atoms

get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_full_scene() %>%
  render_model(samples=256,sample_method="sobol_blue")

#Light the example from below as well
get_example_molecule("caffeine") %>%
  read_sdf() %>%
  generate_full_scene() %>%
  render_model(lights = "both", samples=256,sample_method="sobol_blue")

#Generate a scene with penicillin, increasing the number of samples and the width/height
#for a higher quality render.
get_example_molecule("penicillin") %>%
  read_sdf() %>%
  generate_full_scene() %>%
```

```
 render_model(lights = "both", samples=256, width=800, height=800,sample_method="sobol_blue")

#Render the scene with rayvertex and custom lights
get_example_molecule("penicillin") %>%
  read_sdf() %>%
  generate_full_scene(pathtrace=FALSE) %>%
  render_model(width=800, height=800,background="grey66",
               lights = rayvertex::directional_light(c(0.2,1,1)))

#Rotate the molecule 30 degrees around the y axis, and the 30 degrees around the z axis
get_example_molecule("penicillin") %>%
  read_sdf() %>%
  generate_full_scene() %>%
  render_model(lights = "both", samples=256, width=800, height=800,
               angle=c(0,30,30),sample_method="sobol_blue")



#Add a checkered plane underneath, using rayrender::add_object and rayrender::xz_rect().
#We also pass a value to `clamp_value` to minimize fireflies (bright spots).
library(rayrender)
get_example_molecule("skatole") %>%
  read_sdf() %>%
  generate_full_scene() %>%
  add_object(xz_rect(xwidth=1000,zwidth=1000,y=-4,
                     material=diffuse(color="#330000",checkercolor="#770000"))) %>%
  render_model(samples=256, width=800, height=800, clamp_value=10,
               sample_method="sobol_blue")
```

---

`run_documentation` *Run Documentation*

---

## Description

This function determines if the examples are being run in pkgdown. It is not meant to be called by the user.

## Usage

```
run_documentation()
```

## Value

Boolean value.

## Examples

```
# See if the documentation should be run.
run_documentation()
```

# Index