Package 'r3dmol'

October 26, 2020

Title Create Interactive 3D Visualizations of Molecular Data

Version 0.1.0

Maintainer Wei Su <swsoyee@gmail.com>

Description Create rich and fully interactive 3D visualizations of molecular data. Visualizations can be included in Shiny apps and R markdown documents, or viewed from the R console and 'RStudio' Viewer. 'r3dmol' includes an extensive API to manipulate the visualization after creation, and supports getting data out of the visualization into R. Based on the '3dmol.js' and the 'htmlwidgets' R package.

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports htmlwidgets, magrittr, methods

Suggests knitr, rmarkdown, shiny, colourpicker, covr, testthat

VignetteBuilder knitr

Depends R (>= 2.10)

URL https://github.com/swsoyee/r3dmol

BugReports https://github.com/swsoyee/r3dmol/issues

NeedsCompilation no

Author Wei Su [aut, cre]

Repository CRAN

Date/Publication 2020-10-26 09:20:06 UTC

R topics documented:

cif_254385 3
cube_benzene_homo 3
init 4
m_add_arrow 5

m_add_as_one_molecule . . . 7
m_add_custom . . . 8
m_add_isosurface . . . 9
m_add_label . . . 10
m_add_model . . . 11
m_add_models_as_frames . . . 12
m_add_property_labels . . . 12
m_add_res_labels . . . 13
m_add_shape . . . 14
m_add_style . . . 15
m_add_surface . . . 16
m_add_unit_cell . . . 16
m_animate . . . 18
m_center . . . 19
m_clear . . . 19
m_create_model_from . . . 20
m_enable_fog . . . 20
m_get_model . . . 21
m_is_animated . . . 21
m_remove_all_labels . . . 22
m_remove_all_models . . . 23
m_remove_all_shapes . . . 23
m_remove_all_surfaces . . . 24
m_remove_label . . . 25
m_render . . . 25
m_rotate . . . 26
m_set_color_by_element . . . 26
m_set_default_cartoon_quality . . . 27
m_set_hover_duration . . . 28
m_set_preceived_distance . . . 28
m_set_projection . . . 29
m_set_slab . . . 30
m_set_view . . . 30
m_set_viewer . . . 31
m_set_zoom_limits . . . 32
m_shiny_demo . . . 32
m_spin . . . 33
m_stop_animate . . . 33
m_translate . . . 34
m_vector3 . . . 35
m_vibrate . . . 36
m_zoom . . . 37
m_zoom_to . . . 37
pdb_1j72 . . . 38
pdb_6zsl . . . 39
r3dmol-shiny . . . 39
sdf_multiple . . . 40
xyz_multiple . . . 40

Index **41**

`cif_254385` *Cif file example*

Description

Cif file example

Usage

```
cif_254385
```

Format

cif format

Source

https://github.com/3dmol/3Dmol.js/blob/master/tests/auto/data/254385.cif

---

`cube_benzene_homo` *Gaussian cube file example*

Description

Gaussian cube file example

Usage

```
cube_benzene_homo
```

Format

Gaussian cube format

Source

https://github.com/3dmol/3Dmol.js/blob/master/tests/test_structs/benzene-homo.cube

`init` *Initialise a WebGL-based viewer*

Description

Create and initialize an appropriate viewer at supplied HTML element using specification in config

Usage

```
r3dmol(id = NULL, ..., width = NULL, height = NULL, elementId = NULL)
```

Arguments

`id`	HTML element id of viewer.
`...`	Viewer input specification, see http://3dmol.csb.pitt.edu/doc/types.html#ViewerSpec for more details.
`width`	Fixed width for viewer (in css units). Ignored when used in a Shiny app – use the `width` parameter in `r3dmolOutput`. It is not recommended to use this parameter because the widget knows how to adjust its width automatically.
`height`	Fixed height for viewer (in css units). It is recommended to not use this parameter since the widget knows how to adjust its height automatically.
`elementId`	Use an explicit element ID for the widget (rather than an automatically generated one). Ignored when used in a Shiny app.

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to()

# Viewer configs setting
r3dmol(
  backgroundColor = "black",
  lowerZoomLimit = 1,
  upperZoomLimit = 350
) %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to()
```

m_add_arrow *Create and add shape*

Description

Create and add shape

Usage

```
m_add_arrow(id, spec = list())

m_add_box(id, spec = list())

m_add_curve(id, spec = list())

m_add_cylinder(id, spec = list())

m_add_line(id, spec = list())

m_add_sphere(id, spec = list())
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
spec	Shape style specification.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

# Add arrow
r3dmol() %>%
  m_add_arrow(
    spec = list(
      start = m_vector3(-10, 0, 0),
      end = m_vector3(0, -10, 0),
      radius = 1,
      radiusRadio = 1,
      mid = 1,
      clickable = TRUE,
      callback =
        "function() {
          this.color.setHex(0xFF0000FF);
          viewer.render()
```

```
        }"
    )
  )

# Add curve
r3dmol() %>%
  m_add_curve(
    spec = list(
      points = list(
        m_vector3(0, 0, 0),
        m_vector3(5, 3, 0),
        m_vector3(5, 7, 0),
        m_vector3(0, 10, 0)
      ),
      radius = 0.5,
      smooth = 10,
      fromArrow = FALSE,
      toArrow = TRUE,
      color = "orange"
    )
  )

# Add cylinder
r3dmol() %>%
  m_add_cylinder(
    spec = list(
      start = list(x = 0.0, y = 0.0, z = 0.0),
      end = list(x = 10.0, y = 0.0, z = 0.0),
      radius = 1.0,
      fromCap = 1,
      toCap = 2,
      color = "red",
      hoverable = TRUE,
      clickable = TRUE,
      callback = "
      function() {
        this.color.setHex(0x00FFFF00);
        viewer.render();
      }",
      hover_callback = "
      function() {
        viewer.render();
      }",
      unhover_callback = "
      function() {
        this.color.setHex(0xFF000000);
        viewer.render();
      }"
    )
  )

# Add line
r3dmol() %>%
```

```
  m_add_line(spec = list(
    dashed = TRUE,
    start = m_vector3(0, 0, 0),
    end = m_vector3(30, 30, 30)
  ))

# Add box
r3dmol() %>%
  m_add_box(spec = list(
    center = m_vector3(0, 5, 0),
    demensions = list(w = 3, h = 4, d = 2),
    color = "magenta"
  ))

# Add sphere
r3dmol() %>%
  m_add_sphere(spec = list(
    center = m_vector3(0, 0, 0),
    radius = 10,
    color = "red"
  ))
```

`m_add_as_one_molecule` *Create and add model to viewer*

Description

Given multimodel file and its format, all atoms are added to one model

Usage

```
m_add_as_one_molecule(id, data, format)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`data`	Input data
`format`	Input format

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

`m_add_custom` *Add custom shape component from user supplied function*

Description

Add custom shape component from user supplied function

Usage

```
m_add_custom(id, spec)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`spec`	Style specification (see: http://3dmol.csb.pitt.edu/doc/types.html#CustomShapeSpec).

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

r <- 20

vertices <- list(
  m_vector3(0, 0, 0),
  m_vector3(r, 0, 0),
  m_vector3(0, r, 0)
)

normals <- list(
  m_vector3(0, 0, 1),
  m_vector3(0, 0, 1),
  m_vector3(0, 0, 1)
)

colors <- list(
  list(r = 1, g = 0, b = 0),
  list(r = 0, g = 1, b = 0),
  list(r = 0, g = 0, b = 1)
)

faces <- 0:2

r3dmol() %>%
  m_add_custom(spec = list(
    vertexArr = vertices,
```

```
    normalArr = normals,
    faceArr = faces,
    color = colors
  ))
```

`m_add_isosurface` *Construct isosurface from volumetric data in gaussian cube format*

Description

Construct isosurface from volumetric data in gaussian cube format

Usage

```
m_add_isosurface(id, data, isoSpec)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`data`	Path of input data path or a vector of data.
`isoSpec`	Volumetric data shape specification

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_isosurface(
    data = cube_benzene_homo,
    isoSpec = list(
      isoval = -0.01,
      color = "red",
      opacity = 0.95
    )
  ) %>%
  m_set_style(
    sel = list(cartoon = list()),
    style = list(stick = list())
  ) %>%
  m_zoom_to()
```

`m_add_label` *Add label to viewer*

Description

Add label to viewer

Usage

```
m_add_label(id, text, options = list(), sel = list(), noshow = TRUE)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`text`	Label text
`options`	Label style specification
`sel`	Set position of label to center of this selection
`noshow`	if `TRUE`, do not immediately display label - when adding multiple labels this is more efficient

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_add_label(
    "Label",
    options = list(
      position = m_vector3(-6.89, 0.75, 0.35),
      backgroundColor = "#666666",
      backgroundOpacity = 0.9
    )
  ) %>%
  m_zoom_to()
```

`m_add_model` *Create and add model to viewer*

Description

Create and add model to viewer, given molecular data and its format. If multi-model file is provided, use `m_add_models` adding atom data to the viewer as separate models.

Usage

```
m_add_model(
  id,
  data,
  format = c("pdb", "sdf", "xyz", "pqr", "mol2", "cif"),
  options
)

m_add_models(id, data, format = c("pdb", "sdf", "xyz", "pqr", "mol2", "cif"))
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`data`	Path of input data path or a vector of data.
`format`	Input format (`'pdb'`, `'sdf'`, `'xyz'`, `'pqr'`, or `'mol2'`).
`options`	Format dependent options. Attributes depend on the input file format.

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

# Single-model file with m_add_model() function
r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb")

# Multi-model file with m_add_models() function
r3dmol() %>%
  m_add_models(data = sdf_multiple, "sdf") %>%
  m_zoom_to()

# Multi-model file with m_add_model() function
r3dmol() %>%
  m_add_model(data = sdf_multiple, "sdf") %>%
  m_zoom_to()
```

m_add_models_as_frames *Create and add model to viewer*

Description

Create and add model to viewer. Given multimodel file and its format, different atomlists are stored in model's frame property and model's atoms are set to the 0th frame

Usage

```
m_add_models_as_frames(id, data, format)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
data	Path of input data path or a vector of data.
format	Input format (see http://3dmol.csb.pitt.edu/doc/types.html#FileFormats).

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_models_as_frames(data = xyz_multiple, format = "xyz") %>%
  m_animate(options = list(loop = "forward", reps = 1)) %>%
  m_set_style(style = list(stick = list(colorscheme = "magentaCarbon"))) %>%
  m_zoom_to()
```

m_add_property_labels *Add property labels*

Description

This will generate one label per a selected atom at the atom's coordinates with the property value as the label text.

Usage

```
m_add_property_labels(id, prop, sel, style)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
prop	Property name
sel	Atom selection specification
style	Style spec to add to specified atoms

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = "data-raw/Conformer3D_CID_5291.sdf", format = "sdf") %>%
  m_set_style(style = list(stick = list(radius = 2))) %>%
  m_zoom_to() %>%
  m_add_property_labels(
    prop = "index",
    sel = list(not = list(elem = "H")),
    style = list(
      fontColor = "black",
      font = "sans-serif",
      fontSize = 28,
      showBackground = FALSE,
      alignment = "center"
    )
  )
```

---

m_add_res_labels *Add residue labels*

Description

Add residue labels. This will generate one label per a residue within the selected atoms. The label will be at the centroid of the atoms and styled according to the passed style. The label text will be resnresi

Usage

```
m_add_res_labels(id, sel, style, byframe)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
sel	Atom selection specification
style	Style spec to add to specified atoms
byframe	if true, create labels for every individual frame, not just current

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(
    style = list(stick = list(radius = 0.15), cartoon = list())
  ) %>%
  m_add_res_labels(
    sel = list(resn = "GLY"),
    style = list(
      font = "Arial",
      fontColor = "white",
      backgroundColor = "black",
      showBackground = TRUE
    )
  ) %>%
  m_zoom_to()
```

---

m_add_shape *Add shape object to viewer*

Description

Add shape object to viewer

Usage

```
m_add_shape(id, shapeSpec = list())
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`shapeSpec`	Style specification for label

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

`m_add_style` *Set or set style properties to all selected atoms*

Description

Set or set style properties to all selected atoms

Usage

```
m_add_style(id, sel = list(), style = list())

m_set_style(id, sel = list(), style = list())
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`sel`	Atom selection specification
`style`	Style spec to apply to specified atoms

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

# Add style to model
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_add_style(style = list(cartoon = list())) %>%
  m_zoom_to()

# Set style to model
r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_set_style(
    sel = list(chain = "A"),
    style = list(stick = list(
      radius = 0.5,
      colorscheme = "magentaCarbon"
    ))
  ) %>%
  m_zoom_to()
```

m_add_surface *Add surface representation to atoms*

Description

Add surface representation to atoms

Usage

```
m_add_surface(id, type, style, atomsel, allsel, focus, surfacecallback)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
type	Surface type (VDW, MS, SAS, or SES)
style	Optional style specification for surface material (e.g. for different coloring scheme, etc).
atomsel	Show surface for atoms in this selection.
allsel	Use atoms in this selection to calculate surface; may be larger group than atomsel.
focus	Optionally begin rendering surface specified atoms.
surfacecallback	function to be called after setting the surface.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

m_add_unit_cell *Unit cell visualization*

Description

Use m_add_unit_cell to create and add unit cell visualization, and m_remove_unit_cell to remove it from model. Use m_replicate_unit_cell to replicate atoms in model to form a super cell of the specified dimensions. Original cell will be centered as much as possible.

Usage

```
m_add_unit_cell(id, model, spec)

m_replicate_unit_cell(id, a, b, c, model)

m_remove_unit_cell(id, model)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
model	Model with unit cell information (e.g., pdb derived). If omitted uses most recently added model.
spec	Visualization style.
a	number of times to replicate cell in X dimension.
b	number of times to replicate cell in Y dimension. If absent, X value is used.
c	number of times to replicate cell in Z dimension. If absent, Y value is used.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

# Create model
mol <- r3dmol() %>%
  m_add_model(
    data = cif_254385,
    "cif",
    options = list(doAssembly = TRUE, normalizeAssembly = TRUE)
  ) %>%
  m_set_style(style = list(
    sphere = list(colorscheme = "Jmol", scale = 0.25),
    stick = list(colorscheme = "Jmol")
  )) %>%
  m_add_unit_cell(spec = list(
    alabel = "x",
    blabel = "y",
    clabel = "z",
    box = list(hidden = TRUE)
  )) %>%
  m_zoom_to()

# Render model
mol

# Remove unit cell
mol %>%
  m_remove_unit_cell()

# Replicate atoms in model to form a super cell
r3dmol() %>%
  m_add_model(data = cif_254385, format = "cif") %>%
  m_set_style(style = list(sphere = list(scale = 0.25))) %>%
  m_add_unit_cell() %>%
  m_zoom_to() %>%
  m_replicate_unit_cell(a = 3, b = 2, c = 1)
```

`m_animate` *Animate all models in viewer from their respective frames*

Description

Animate all models in viewer from their respective frames

Usage

```
m_animate(id, options)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`options`	can specify `interval` (speed of animation), `loop` (direction of looping, `'backward'`, `'forward'` or `'backAndForth'`), step interval between frames (`'step'`), and `reps` (number of repetitions, `0` indicates infinite loop)

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

xyz <- "4
* (null), Energy   -1000.0000000
N     0.000005    0.019779   -0.000003   -0.157114    0.000052   -0.012746
H     0.931955   -0.364989    0.000003    1.507100   -0.601158   -0.004108
H    -0.465975   -0.364992    0.807088    0.283368    0.257996   -0.583024
H    -0.465979   -0.364991   -0.807088    0.392764    0.342436    0.764260
"

r3dmol(
  width = 400,
  height = 400,
  backgroundColor = "0xeeeeee"
) %>%
  m_add_model(
    data = xyz,
    format = "xyz",
    options = list(vibrate = list(frames = 10, amplitude = 1))
  ) %>%
  m_set_style(style = list(stick = list())) %>%
  m_animate(list(loop = "backAndForth")) %>%
  m_zoom_to()
```

m_center *Re-center the viewer around the provided selection*

Description

Re-center the viewer around the provided selection (unlike zoomTo, does not zoom).

Usage

```
m_center(id, sel, animationDuration, fixedPath)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
sel	Selection specification specifying model and atom properties to select. Default: all atoms in viewer
animationDuration	an optional parameter of milliseconds numeric that denotes the duration of a zoom animation
fixedPath	if true animation is constrained to requested motion, overriding updates that happen during the animation

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_center(animationDuration = 1000)
```

m_clear *Clear scene of all objects*

Description

Clear scene of all objects

Usage

```
m_clear(id)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())

Value

R3dmol id or a r3dmol object (the output from r3dmol())

---

m_create_model_from *Create a new model from atoms specified by sel*

Description

Create a new model from atoms specified by sel. If extract, removes selected atoms from existing models.

Usage

```
m_create_model_from(id, sel, extract)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
sel	Atom selection specification.
extract	If true, remove selected atoms from existing models

Value

R3dmol id or a r3dmol object (the output from r3dmol())

---

m_enable_fog *Enable/disable fog for content far from the camera*

Description

Enable/disable fog for content far from the camera

Usage

```
m_enable_fog(id, fog = TRUE)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
fog	whether to enable or disable the fog, default is TRUE.

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_enable_fog(fog = FALSE)
```

---

m_get_model *Return specified model*

Description

Return specified model

Usage

```
m_get_model(id, modelId)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`modelId`	Retrieve model with specified id

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

---

m_is_animated *Get viewer animate status*

Description

Return true if viewer is currently being animated, false otherwise

Usage

```
m_is_animated(id)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Value

logical

`m_remove_all_labels` *Remove all labels from viewer*

Description

Remove all labels from viewer

Usage

```
m_remove_all_labels(id)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())

Value

id R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

mol <- r3dmol() %>%
  m_add_model(data = "data-raw/Conformer3D_CID_5291.sdf", format = "sdf") %>%
  m_set_style(style = list(stick = list(radius = 2))) %>%
  m_zoom_to() %>%
  m_add_property_labels(
    prop = "index",
    sel = list(not = list(elem = "H")),
    style = list(
      fontColor = "black",
      font = "sans-serif",
      fontSize = 28,
      showBackground = FALSE,
      alignment = "center"
    )
  )

# Render model with labels
mol

# Remove all labels
mol %>%
  m_remove_all_labels()
```

m_remove_all_models *Delete all existing models*

Description

Delete all existing models

Usage

```
m_remove_all_models(id)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())

Value

id R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

mol <- r3dmol() %>%
  m_add_model(data = "data-raw/Conformer3D_CID_5291.sdf", format = "sdf")

# Render model
mol

# Remove all labels
mol %>%
  m_remove_all_models()
```

m_remove_all_shapes *Remove all shape objects from viewer*

Description

Remove all shape objects from viewer

Usage

```
m_remove_all_shapes(id)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())

Value

id R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

mol <- r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_add_sphere(spec = list(
    center = list(x = 0, y = 0, z = 0),
    radius = 10.0,
    color = "red"
  ))

# Render model with shape
mol

# Remove shape
mol %>%
  m_remove_all_shapes()
```

`m_remove_all_surfaces` *Remove all labels from viewer*

Description

Remove all labels from viewer

Usage

```
m_remove_all_surfaces(id)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Value

id R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

m_remove_label *Remove label from viewer*

Description

Remove label from viewer

Usage

```
m_remove_label(id, label)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
label	R3dmol object label

Value

id R3dmol id or a r3dmol object (the output from r3dmol())

m_render *Render current state of viewer*

Description

Render current state of viewer, after adding/removing models, applying styles, etc. In most cases, the model will render automatically, only call it when manual rendering is required.

Usage

```
m_render(id)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_render()
```

m_rotate *Rotate scene by angle degrees around axis*

Description

Rotate scene by angle degrees around axis

Usage

```
m_rotate(id, angle, axis = "v", animationDuration = 0, fixedPath)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
angle	Angle, in degrees numeric, to rotate by.
axis	Axis ("x", "y", "z", "vx", "vy", "vz") to rotate around. Default "y". View relative (rather than model relative) axes are prefixed with "v". Axis can also be specified as a vector.
animationDuration	an optional parameter of milliseconds numeric that denotes the duration of the rotation animation. Default 0 (no animation)
fixedPath	if true animation is constrained to requested motion, overriding updates that happen during the animation

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)
r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_rotate(angle = 90, axis = "y", animationDuration = 1000)
```

m_set_color_by_element *Set color by element*

Description

Set color by element

Usage

```
m_set_color_by_element(id, sel, colors)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
sel	Atom selection.
colors	Color hex code or name.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

---

m_set_default_cartoon_quality

Set the default cartoon quality for newly created models

Description

Set the default cartoon quality for newly created models. Default is 5. Current models are not affected.

Usage

```
m_set_default_cartoon_quality(id, quality)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
quality	Default cartoon quality.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_set_default_cartoon_quality(20) %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_zoom_to()
```

m_set_hover_duration *Set the duration of the hover delay*

Description

Set the duration of the hover delay

Usage

```
m_set_hover_duration(id, hoverDuration)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
hoverDuration	an optional parameter that denotes the duration of the hover delay (in milliseconds) before the hover action is called

Value

R3dmol id or a r3dmol object (the output from r3dmol())

m_set_preceived_distance *Set the distance between the model and the camera*

Description

Essentially zooming. Useful while stereo rendering.

Usage

```
m_set_preceived_distance(id, dist)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
dist	Numeric distance.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_preceived_distance(dist = 200)
```

`m_set_projection` *Set view projection scheme*

Description

Set view projection scheme

Usage

```
m_set_projection(id, scheme = c("perspective", "orthographic"))
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`scheme`	Either `orthographic` or `perspective`. Default is `perspective`. Orthographic can also be enabled on viewer creation by setting `orthographic` to true in the config object.

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_set_projection(scheme = "orthographic")
```

m_set_slab *Set slab of view*

Description

Set slab of view (contents outside of slab are clipped).

Usage

```
m_set_slab(id, near, far)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
near	near clipping plane distance
far	far clipping plane distance

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_zoom_to() %>%
  m_set_slab(near = -90, far = 0)
```

m_set_view *Sets the view to the specified translation, zoom, rotation and style*

Description

Sets the view to the specified translation, zoom, rotation and style

Usage

```
m_set_view(id, arg)

m_set_view_style(id, style)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
arg	Vector formatted view setting, c(pos.x,pos.y,pos.z,rotationGroup.position.z,q.x,q.y,q.z,q.w Requires any one of q.x, q.y, q.z, q.w to be set to 1 to enable mouse control, otherwise only static image is rendered.
style	css style object in list.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = list(cartoon = list())) %>%
  m_set_view(arg = c(20, -20, 10, -200, 0, 1, 0, 0)) %>%
  m_set_view_style(style = list(style = "outline", color = "blue"))
```

---

m_set_viewer *Set viewer properties*

Description

Functions of setting viewer properties, such as width, height, background color, etc. The viewer size can be adjusted automatically under normal circumstances.

Usage

```
m_set_width(id, width)

m_set_height(id, height)

m_set_background_color(id, hex, alpha)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
width, height	Weight and height numeric in pixels
hex	Hex code specified background color, or standard color spec character
alpha	Alpha level numeric (default 1.0)

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to() %>%
  m_set_width(300) %>%
  m_set_background_color("#666666", alpha = 0.9)
```

---

m_set_zoom_limits *Set lower and upper limit stops for zoom*

Description

Set lower and upper limit stops for zoom

Usage

```
m_set_zoom_limits(id, lower = 0, upper = Inf)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
lower	limit on zoom in (positive numeric number). Default 0.
upper	limit on zoom out (positive numeric number). Default Inf.

Value

R3dmol id or a r3dmol object (the output from r3dmol())

---

m_shiny_demo *Run examples of using* r3dmol *in a Shiny app*

Description

Run examples of using r3dmol in a Shiny app

Usage

```
m_shiny_demo()
```

Examples

```
if (interactive()) {
  m_shiny_demo()
}
```

m_spin *Continuously rotate a scene around the specified axis*

Description

Continuously rotate a scene around the specified axis

Usage

```
m_spin(id, axis = "y")
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
axis	Axis ("x", "y", "z", "vx", "vy", "vz") to rotate around. Default "y". View relative (rather than model relative) axes are prefixed with "v".

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)
r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to() %>%
  m_spin()
```

m_stop_animate *Stop animation of all models in viewer*

Description

Stop animation of all models in viewer

Usage

```
m_stop_animate(id)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())

`m_translate` *Translate current view or models by x,y screen coordinates*

Description

`m_translate()` pans the camera rather than translating the model. `m_translate_scene()` translates the models relative to the current view. It does not change the center of rotation.

Usage

```
m_translate(id, x, y, animationDuration, fixedPath)

m_translate_scene(id, x, y, animationDuration, fixedPath)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`x`	Relative change `numeric` in view coordinates of camera
`y`	Relative change `numeric` in view coordinates of camera
`animationDuration`	an optional parameter of milliseconds `numeric` that denotes the duration of a zoom animation
`fixedPath`	if `true` animation is constrained to requested motion, overriding updates that happen during the animation

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

# Translate current view by x,y screen coordinates
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = list(cartoon = list(), stick = list())) %>%
  m_translate(
    x = 200,
    y = 50,
    animationDuration = 1000
  ) %>%
  m_rotate(
    angle = 90,
    axis = "z",
    animationDuration = 1000
  ) %>%
  m_zoom_to()
```

```
# Translate current models by x,y screen coordinates
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = list(cartoon = list(), stick = list())) %>%
  m_translate_scene(
    x = 200,
    y = 50,
    animationDuration = 1000
  ) %>%
  m_rotate(
    angle = 90,
    axis = "z",
    animationDuration = 1000
  ) %>%
  m_zoom_to()
```

`m_vector3` *Create a 3 dimensional vector*

Description

Create a 3 dimensional vector

Usage

```
m_vector3(x = 0, y = 0, z = 0)
```

Arguments

x	x coordinate, `character` and `numeric` are both accepted.
y	y coordinate, `character` and `numeric` are both accepted.
z	z coordinate, `character` and `numeric` are both accepted.

Value

3 dimensional list object

Examples

```
library(r3dmol)
m_vector3(1, 2, 3)
```

`m_vibrate` *Add model's vibration*

Description

If atoms have dx, dy, dz properties (in some xyz files), vibrate populates each model's frame property based on parameters. Models can then be animated.

Usage

```
m_vibrate(id, numFrames, amplitude, bothWays, arrowSpec)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`numFrames`	Number of frames to be created, default to 10
`amplitude`	Amplitude of distortion, default to 1 (full)
`bothWays`	If true, extend both in positive and negative directions by numFrames
`arrowSpec`	Specification for drawing animated arrows. If color isn't specified, atom color (sphere, stick, line preference) is used.

Value

R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)

Examples

```
library(r3dmol)

xyz <- "4
* (null), Energy   -1000.0000000
N     0.000005    0.019779   -0.000003   -0.157114    0.000052   -0.012746
H     0.931955   -0.364989    0.000003    1.507100   -0.601158   -0.004108
H    -0.465975   -0.364992    0.807088    0.283368    0.257996   -0.583024
H    -0.465979   -0.364991   -0.807088    0.392764    0.342436    0.764260
"

r3dmol() %>%
  m_add_model(data = xyz, format = "xyz") %>%
  m_set_style(style = list(stick = list())) %>%
  m_vibrate(numFrames = 10, amplitude = 1) %>%
  m_animate(options = list(loop = "backAndForth", reps = 0)) %>%
  m_zoom_to()
```

m_zoom *Zoom current view by a constant factor*

Description

Zoom current view by a constant factor

Usage

```
m_zoom(id, factor = 2, animationDuration, fixedPath)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
factor	Magnification numeric factor. Values greater than 1 will zoom in, less than one will zoom out. Default 2.
animationDuration	an optional parameter of milliseconds numeric that denotes the duration of a zoom animation
fixedPath	if true animation is constrained to requested motion, overriding updates that happen during the animation

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to() %>%
  m_zoom(factor = 2, animationDuration = 1000)
```

m_zoom_to *Zoom to center of atom selection*

Description

Zoom to center of atom selection. The slab will be set appropriately for the selection, unless an empty selection is provided, in which case there will be no slab.

Usage

```
m_zoom_to(id, sel, animationDuration, fixedPath)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
sel	Selection specification specifying model and atom properties to select. Default: all atoms in viewer.
animationDuration	an optional parameter of milliseconds numeric that denotes the duration of a zoom animation
fixedPath	if true animation is constrained to requested motion, overriding updates that happen during the animation

Value

R3dmol id or a r3dmol object (the output from r3dmol())

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to()
```

---

pdb_1j72 *Crystal Structure of Mutant Macrophage Capping Protein (Cap G) with Actin-severing Activity in the Ca2+-Free Form in PDB format*

Description

Crystal Structure of Mutant Macrophage Capping Protein (Cap G) with Actin-severing Activity in the Ca2+-Free Form in PDB format

Usage

```
pdb_1j72
```

Format

PDB Format.

Source

DOI: 10.2210/pdb1J72/pdb. https://www.rcsb.org/structure/1J72

pdb_6zsl *Crystal structure of the SARS-CoV-2 helicase at 1.94 Angstrom resolution in PDB format*

Description

Crystal structure of the SARS-CoV-2 helicase at 1.94 Angstrom resolution in PDB format

Usage

```
pdb_6zsl
```

Format

PDB Format.

Source

DOI: 10.2210/pdb6ZSL/pdb. https://www.rcsb.org/structure/6zsl

r3dmol-shiny *Shiny bindings for r3dmol*

Description

Output and render functions for using r3dmol within Shiny applications and interactive Rmd documents.

Usage

```
r3dmolOutput(outputId, width = "100%", height = "400px")

renderR3dmol(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a r3dmol
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

sdf_multiple *Multiple sdf file example*

Description

Multiple sdf file example

Usage

```
sdf_multiple
```

Format

sdf format

Source

https://github.com/3dmol/3Dmol.js/blob/master/tests/test_structs/multiple.sdf

xyz_multiple *Multiple xyz file example*

Description

Multiple xyz file example

Usage

```
xyz_multiple
```

Format

xyz format

Source

https://github.com/3dmol/3Dmol.js/blob/master/tests/test_structs/multiple2.xyz

Index

∗ **datasets**
  cif_254385, 3
  cube_benzene_homo, 3
  pdb_1j72, 38
  pdb_6zsl, 39
  sdf_multiple, 40
  xyz_multiple, 40

add_model *(*m_add_model*)*, 11

cif_254385, 3
cube_benzene_homo, 3

init, 4

m_add_anyShape *(*m_add_arrow*)*, 5
m_add_arrow, 5
m_add_as_one_molecule, 7
m_add_box *(*m_add_arrow*)*, 5
m_add_curve *(*m_add_arrow*)*, 5
m_add_custom, 8
m_add_cylinder *(*m_add_arrow*)*, 5
m_add_isosurface, 9
m_add_label, 10
m_add_line *(*m_add_arrow*)*, 5
m_add_model, 11
m_add_models, *11*
m_add_models *(*m_add_model*)*, 11
m_add_models_as_frames, 12
m_add_property_labels, 12
m_add_res_labels, 13
m_add_shape, 14
m_add_sphere *(*m_add_arrow*)*, 5
m_add_style, 15
m_add_surface, 16
m_add_unit_cell, *16*, 16
m_animate, 18
m_center, 19
m_clear, 19
m_create_model_from, 20
m_enable_fog, 20
m_get_model, 21
m_is_animated, 21
m_remove_all_labels, 22
m_remove_all_models, 23
m_remove_all_shapes, 23
m_remove_all_surfaces, 24
m_remove_label, 25
m_remove_unit_cell, *16*
m_remove_unit_cell *(*m_add_unit_cell*)*, 16
m_render, 25
m_replicate_unit_cell, *16*
m_replicate_unit_cell
        *(*m_add_unit_cell*)*, 16
m_rotate, 26
m_set_background_color *(*m_set_viewer*)*, 31
m_set_color_by_element, 26
m_set_default_cartoon_quality, 27
m_set_height *(*m_set_viewer*)*, 31
m_set_hover_duration, 28
m_set_preceived_distance, 28
m_set_projection, 29
m_set_slab, 30
m_set_style *(*m_add_style*)*, 15
m_set_view, 30
m_set_view_style *(*m_set_view*)*, 30
m_set_viewer, 31
m_set_width *(*m_set_viewer*)*, 31
m_set_zoom_limits, 32
m_shiny_demo, 32
m_spin, 33
m_stop_animate, 33
m_style *(*m_add_style*)*, 15
m_translate, 34
m_translate_scene *(*m_translate*)*, 34
m_unit_cell *(*m_add_unit_cell*)*, 16
m_vector3, 35
m_vibrate, 36

m_zoom, 37
m_zoom_to, 37

pdb_1j72, 38
pdb_6zsl, 39

r3dmol *(*init*)*, 4
r3dmol-shiny, 39
r3dmolOutput, *4*
r3dmolOutput *(*r3dmol-shiny*)*, 39
renderR3dmol *(*r3dmol-shiny*)*, 39

sdf_multiple, 40

xyz_multiple, 40