Package 'r3dmol'

March 14, 2021

Title Create Interactive 3D Visualizations of Molecular Data

Version 0.1.2

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, bio3d

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] (<https://orcid.org/0000-0002-9302-5332>), Brady Johnston [aut] (<https://orcid.org/0000-0001-6301-2269>)

Repository CRAN

Date/Publication 2021-03-14 15:10:02 UTC

R topics documented:

`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,
  viewer_spec = m_viewer_spec(),
  ...,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

id	HTML element id of viewer.
viewer_spec	Some useful viewer input specifications. Additional options pass in via ... will override options set in viewer_spec.
...	Additional, more niche 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 *Add arrow shape*

Description

Add an arrow from start to end, additional customisation through m_shape_spec().

Usage

```
m_add_arrow(
  id,
  start,
  end,
  radius = 0.2,
  radiusRatio = 1.62,
  mid = 0.62,
  spec = m_shape_spec(),
  hidden = FALSE
)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
start	Start location of arrow Can be either m_sel() or m_vector3().
end	End location of arrow. Can be either m_sel() or m_vector3().
radius	Radius of base cylinder for arrow.
radiusRatio	Ratio of arrow point to the base cylinder. Default 1.618034.
mid	Relative position of the arrow point base, along the length of arrow object. Default to 0.618034.
spec	Additional shape specifications defined with m_shape_spec().
hidden	Hide object if TRUE.

Examples

```
## Not run:
r3dmol() %>%
  m_add_model(data = m_fetch_pdb("1bna")) %>%
  m_zoom_to(sel = m_sel(resi = 1)) %>%
  m_add_arrow(
    start = m_sel(resi = 1),
    end = m_sel(resi = 3),
    spec = m_shape_spec(color = "green")
  )

## End(Not run)
```

---

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_box *Create and add shape*

Description

Create and add shape

Usage

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

m_add_curve(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(
    start = m_vector3(-10, 0, 0),
    end = m_vector3(0, -10, 0),
    radius = 1,
    radiusRatio = 1,
    mid = 1,
    spec = m_shape_spec(
      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(
    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,
    spec = m_shape_spec(
      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(
    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(
    center = m_vector3(0, 0, 0),
    radius = 10,
    spec = m_shape_spec(color = "red")
  )
```

`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_cylinder` *Add Cylinder Between Points*

Description

Add cynliders between the given points. Will match starting point/s with ending point/s to create a line between each point. Styling options can be supplied as one option, or a vector of length equal to the number of lines.

Usage

```
m_add_cylinder(
  id,
  start,
  end,
  radius = 0.1,
  fromCap = 1,
  toCap = 1,
  dashed = FALSE,
  color = "black",
  alpha = FALSE,
  wireframe = FALSE,
  hidden = FALSE,
  spec = m_shape_spec()
)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`).
`start`	Starting position (or `list()` of positions) of line. Can be a single position or `list()` of positions. Format either `m_sel()` or `m_vector3()`.

end	Ending position (or list() of positions) of line. Can be a single position or list() of positions. Format either m_sel() or m_vector3().
radius	Radius of cylinder.
fromCap	Cap at start of cylinder. 0 for none, 1 for flat, 2 for rounded.
toCap	Cap at end of cylinder. 0 for none, 1 for flat, 2 for rounded.
dashed	Boolean, dashed style cylinder instead of solid.
color	Color value for cylinders. Either 1 or vector of colors equal in length to start.
alpha	Alpha value for transparency.
wireframe	Logical, display as wireframe.
hidden	Logical, whether or not to hide the cylinder.
spec	Additional shape specifications defined with m_shape_spec().

Examples

```
## Add a cylinder between residue 1 & 2 of Chain "A"
r3dmol() %>%
  m_add_model(pdb_6zsl) %>%
  m_zoom_to(sel = m_sel(resi = 1)) %>%
  m_add_cylinder(
    start = m_sel(resi = 1, chain = "A"),
    end = m_sel(resi = 2, chain = "A"),
    dashed = TRUE,
    radius = 0.1
  )

# Add two cylinders.
# Blue cylinder is between residues 1 & 2
# Green cylinder is between residues 3 & 4
r3dmol() %>%
  m_add_model(pdb_6zsl) %>%
  m_zoom_to(sel = m_sel(resi = 1:4, chain = "A")) %>%
  m_add_cylinder(
    start = list(
      m_sel(resi = 1, chain = "A"),
      m_sel(resi = 3, chain = "A")
    ),
    end = list(
      m_sel(resi = 2, chain = "A"),
      m_sel(resi = 4, chain = "A")
    ),
    dashed = TRUE,
    radius = 0.1,
    color = c("blue", "green")
  ) %>%
  m_add_res_labels(m_sel(resi = 1:4, chain = "A"))

# The same scene achieved with m_multi_resi_sel()
r3dmol() %>%
  m_add_model(pdb_6zsl) %>%
```

```
  m_zoom_to(sel = m_sel(resi = 1:4, chain = "A")) %>%
  m_add_cylinder(
    start = m_multi_resi_sel(resi = c(1, 3), chain = "A"),
    end = list(
      m_sel(resi = 2, chain = "A"),
      m_sel(resi = 4, chain = "A")
    ),
    dashed = TRUE,
    radius = 0.1,
    color = c("blue", "green")
  ) %>%
  m_add_res_labels(m_sel(resi = 1:4, chain = "A"))
```

`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_zoom_to()
```

`m_add_label` *Add label to viewer*

Description

Add label to viewer

Usage

```
m_add_label(id, text, style = m_style_label(), sel = m_sel(), noshow = TRUE)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`text`	Label text
`style`	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(
    text = "Label",
    sel = m_vector3(-6.89, 0.75, 0.35),
    style = m_style_label(
      backgroundColor = "#666666",
      backgroundOpacity = 0.9
    )
  ) %>%
  m_zoom_to()
```

m_add_line *Add Lines Between Points*

Description

Add lines between the given points. Will match starting point/s with ending point/s to create a line between each point. Styling options can be supplied as one option, or a vector of length equal to the number of lines.

Usage

```
m_add_line(
  id,
  start,
  end,
  dashed = TRUE,
  color = "black",
  opacity = 1,
  hidden = FALSE
)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol()).
start	Starting position (or list() of positions) of line. Can be a single position or list() of positions. Format either m_sel() or m_vector3().
end	Ending position (or list() of positions) of line. Can be a single position or list() of positions. Format either m_sel() or m_vector3().
dashed	Logical whether the lines are dashed.
color	Either single or list of color values equal to number of lines.
opacity	Either single or list of opacity values equal to number of lines.
hidden	Either single or list of hidden values equal to number of lines.

Value

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

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl) %>%
  m_set_style(style = m_style_cartoon()) %>%
  m_zoom_to() %>%
  m_add_style(
```

```
    sel = m_sel(resi = 1:10),
    style = c(
      m_style_stick(),
      m_style_sphere(scale = 0.3)
    )
  ) %>%
  m_add_line(
    start = list(
      m_sel(resi = 1, chain = "A"),
      m_sel(resi = 1, chain = "A")
    ),
    end = list(
      m_sel(resi = 10, chain = "A"),
      m_sel(resi = 10, chain = "B")
    ),
    dashed = TRUE
  )
```

`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"),
  keepH = FALSE,
  options = list()
)

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'`).
`keepH`	Default to FALSE, whether to keep or strip hydrogens from imported model.
`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()

# Add model and keep hydrogens.
## Not run:
r3dmol() %>%
  m_add_model(m_fetch_pdb("5D8V"), keepH = TRUE) %>%
  m_set_style(m_style_sphere()) %>%
  m_zoom_to() %>%
  m_spin()

## End(Not run)
```

`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 = m_style_stick(colorScheme = "magentaCarbon")) %>%
  m_zoom_to()
```

---

m_add_outline *Add colored outline to all objects in scene.*

Description

Adds a colored outline to all objects in the scene, helping the viewer to distinguish depth in often complex molecular scenes.

Usage

```
m_add_outline(id, width = 0.1, color = "black")
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
width	Width of the outline, defaults to 0.1
color	Color of the outline, defaults to black.

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_1j72) %>%
  m_set_style(style = m_style_stick()) %>%
  m_add_outline()
```

---

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 = m_sel(), style = m_style_label())
```

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 = m_style_stick(radius = 2)) %>%
  m_zoom_to() %>%
  m_add_property_labels(
    prop = "index",
    sel = list(not = list(elem = "H")),
    style = m_style_label(
      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 = m_sel(), style = m_style_label(), 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 = c(
      m_style_stick(radius = 0.15),
      m_style_cartoon()
    )
  ) %>%
  m_add_res_labels(
    sel = m_sel(resn = "GLY"),
    style = m_style_label(
      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_sphere` *Add Sphere Shape*

Description

Adds sphere at given location, with given radius.

Usage

```
m_add_sphere(id, center, radius = 1, spec = m_shape_spec(), ...)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`)
`center`	center point of sphere. Can be `m_sel()`.
`radius`	radius of sphere.
`spec`	Additional shape specifications defined with `m_shape_spec()`.
`...`	Additional shape specifcations, that can be called outside of `m_shape_spec()` such as `color = 'blue'`

Examples

```
r3dmol() %>%
  m_add_model(data = m_fetch_pdb("1bna")) %>%
  m_add_sphere(
    center = m_sel(resi = 1),
    spec = m_shape_spec(color = "green", wireframe = TRUE)
  ) %>%
  m_zoom_to(sel = m_sel(resi = 1))
```

`m_add_style` *Overwrite Previous Style*

Description

Takes a selection and overwrites previous styling with given styles.

Usage

```
m_add_style(id, style = m_style_cartoon(), sel = m_sel())
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol())
style	Style spec to apply to specified atoms using m_style_*()
sel	Atom selection specification with m_sel()

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 = m_style_cartoon()) %>%
  m_zoom_to()

# Set style to model
r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = m_style_cartoon()) %>%
  m_set_style(
    sel = m_sel(chain = "A"),
    style = m_style_stick(
      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 = m_style_surface(),
  atomsel = m_sel(),
  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 = c(
    m_style_sphere(colorScheme = "Jmol", scale = 0.25),
    m_style_stick(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 = m_style_sphere(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 = m_style_stick()) %>%
  m_animate(list(loop = "backAndForth")) %>%
  m_zoom_to()
```

---

m_bio3d *Load structure from package bio3d*

Description

Function to take bio3d structure and use in the r3dmol app.

Usage

```
m_bio3d(pdb)
```

Arguments

pdb	bio3d object containing coordinates for desired structure

Examples

```
library(bio3d)
library(r3dmol)

# create bio3d object
pdb <- read.pdb("1bna")

# inspect bio3d object
pdb

# load bio3d object into r3dmol
r3dmol() %>%
  m_add_model(data = m_bio3d(pdb)) %>%
  m_zoom_to()
```

---

m_button *Add button into viewer*

Description

Add additional buttons to the viewer and pass in JavaScript functions to enable additional actions to be done when the button is clicked (such as styling changes to the model). You can also use css flex layout to control the layout of all added buttons.

Usage

```
m_button(
  id,
  name,
  label,
  func,
  align_items = "flex-start",
  justify_content = "flex-start"
)
```

Arguments

id	R3dmol id or a r3dmol object (the output from r3dmol()).
name	Name for button.
label	Label for button.
func	The function executed when the button is clicked.
align_items	The css align-items property specifies the default alignment for items inside the viewer.

justify_content
: The css `justify-content` property aligns the buttons when the items do not use all available space on the main-axis (horizontally).

Details

If more than one button is set, only the layout (`justify-content` and `align-items`) of the first button will be used.

Value

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

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_zoom_to() %>%
  m_button(
    name = "cartoon",
    label = "Cartoon",
    align_items = "flex-end",
    justify_content = "center",
    func = "
      function() {
        viewer.setStyle({cartoon:{}});
        viewer.render();
      }
    "
  ) %>%
  m_button(
    name = "stick",
    label = "Stick",
    func = "
      function() {
        viewer.setStyle({stick:{}});
        viewer.render();
      }
    "
  )
```

---

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 = m_style_cartoon()) %>%
  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 = m_style_cartoon()) %>%
  m_enable_fog(fog = FALSE)
```

`m_fetch_pdb` *Fetch Structure from PDB*

Description

Using specified pdb id, retrieved .pdb file using bio3d::get.pdb() function. Will always query the only PDB for structure, and not store on local drive. May take some time to fetch information, every time it is run.

Usage

```
m_fetch_pdb(pdb, save.pdb = FALSE, path = NULL)
```

Arguments

`pdb`	PDB ID string for structure.
`save.pdb`	Logical, whether or not to save the PDB to local drive. Will speed up subsequent load times. Defaults to `FALSE`
`path`	If `save.pdb = TRUE`, determines the location for file to be saved. Defaults to `getwd()`.

Examples

```
library(r3dmol)
## Not run:
r3dmol() %>%
  m_add_model(data = m_fetch_pdb("1bna")) %>%
  m_set_style(style = c(m_style_cartoon(), m_style_stick())) %>%
  m_zoom_to()

## End(Not run)
```

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_glimpse *Quickly View Given Structure*

Description

Creates a scene with a number of simple defaults in order to quickly view the structure without having to write multiple lines of code.

Usage

```
m_glimpse(
  model,
  highlight = m_sel(),
  zoom = TRUE,
  spin = FALSE,
  nomouse = FALSE,
  ribbon = FALSE,
  outline = TRUE,
  backgroundColor = "white"
)
```

Arguments

model	Model to add to scene. Can be {bio3d} pdb object or PDB id code string (i.e "4ozs").
highlight	Given selection will additionally have 'ball-n-stick' representation. View will also zoom to selection.
zoom	Logical. FALSE will not zoom onto highlighted selection.
spin	TRUE / FALSE will enable or disable spin. A numeric value will change spin speed and negative will reverse the direction.
nomouse	Logical. Enables / disables mouse input.
ribbon	Logical. Enables / disables ribbon representation.
outline	Logical. Enables / disables black outline.
backgroundColor	String of simple colour names or hex code to change background color of viewer.

Examples

```
library(r3dmol)

# write/read demo structure as {bio3d} object
tmp <- tempfile()
write(pdb_6zsl, tmp)
pdb <- bio3d::read.pdb(tmp)

# quickly preview structure
pdb %>%
  m_glimpse()

# preview structure, highlighting particular region.
pdb %>%
  m_glimpse(m_sel(resi = 1:10, chain = "A"), spin = 0.2)
## Not run:

# Fetch given PDB string and quickly preview structure
"4ozs" %>%
  m_glimpse(spin = TRUE)

## End(Not run)
```

---

m_grid *Create a grid of viewers that share a WebGL canvas*

Description

Create a grid of viewers that share a WebGL canvas

Usage

```
m_grid(
  viewer,
  element_id,
  rows = NULL,
  cols = NULL,
  control_all = TRUE,
  viewer_config = m_viewer_spec(),
  width = NULL,
  height = NULL
)
```

Arguments

viewer	A list contains sub-viewers.
element_id	HTML string identifier.
rows	Number of rows in viewer grid.
cols	Number of columns in viewer grid.
control_all	Logical, simaultaneous mouse control of all windows in the grid.
viewer_config	Viewer specification to apply to all subviewers.
width	Fixed width for combined 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 combined viewer (in css units). It is recommended to not use this parameter since the widget knows how to adjust its height automatically.

Value

An `r3dmol` object (the output from `r3dmol()`).

Examples

```
library(r3dmol)

m1 <- r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_zoom_to()

m2 <- m1 %>%
  m_set_style(style = m_style_cartoon(color = "spectrum"))

m3 <- m1 %>%
  m_set_style(style = m_style_stick())

m4 <- m1 %>%
  m_set_style(style = m_style_sphere())

m_grid(
```

```
  viewer = list(m1, m2, m3, m4),
  control_all = TRUE,
  viewer_config = m_viewer_spec(
    backgroundColor = "black"
  )
)
```

`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_multi_resi_sel` *Selection Across Multiple Residues*

Description

Behaves just like the `m_sel()`, but returns a new selection for each residue specified with `resi`.

Usage

```
m_multi_resi_sel(
  resi = NULL,
  resn = NULL,
  chain = NULL,
  model = NULL,
  elem = NULL,
  atom = NULL,
  invert = NULL,
  byres = NULL,
  b = NULL,
  expand = NULL,
```

```
  bonds = NULL,
  ss = NULL,
  clickable = NULL,
  callback = NULL
)
```

Arguments

resi	Residue number/s. (vector)
resn	Parent residue name as 3-letter code (e.g. "ALA", "GLY", "CYS"...)
chain	String, chain this atom belongs to (e.g. 'A' for chain A)
model	a single model or list of models from which atoms should be selected. Can also specify by numerical creation order. Reverse indexing is allowed (-1 specifies last added model).
elem	element abbreviation (e.g 'H', 'Ca', etc)
atom	Atom name, may be more specific than 'elem' (e.g. 'CA' for alpha carbon)
invert	Logical, if `invert = TRUE`, Inverts the selection criteria.
byres	Logical, if `byres = TRUE`, expands the selection to entire residues that include any selected atoms.
b	Atom b factor data
expand	Expand selection to include atoms within a specified distance from current selection. all atoms of any residue that has any atom already selected.
bonds	overloaded to select number of bonds, e.g. `bonds = 0` will select all non-bonded atoms
ss	Secondary structure identifier. 'h' for helix, 's' for beta-sheet.
clickable	Set this flag to true to enable click selection handling for this atom
callback	Callback click handler function to be executed on this atom and its parent viewer.

Details

The `m_sel(resi = 1:10)` returns a selection of all 10 residues. The `m_multi_resi_sel(resi = 1:10)` returns 10 individual selections, each containing only 1 of the residues.

Value

sel `list()` for selecting atoms.

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_6zsl) %>%
  m_set_style(style = m_style_cartoon()) %>%
  m_zoom_to() %>%
  m_add_style(
```

```
    sel = m_sel(resi = 1:10),
    style = c(
      m_style_stick(),
      m_style_sphere(scale = 0.3)
    )
  ) %>%
  m_add_line(
    start = m_multi_resi_sel(resi = rep(1, 9), chain = "A"),
    end = m_multi_resi_sel(
      resi = 2:10,
      chain = "B"
    )
  )
```

`m_png` *Convert widgets to PNG image*

Description

Convert widgets to PNG image

Usage

```
m_png(id, width, height)
```

Arguments

`id`	R3dmol `id` or a `r3dmol` object (the output from `r3dmol()`).
`width, height`	image width and height.

Value

Base64 encoded png image wrapped by `<img>` tag.

Examples

```
library(r3dmol)

r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_cartoon()) %>%
  m_zoom_to() %>%
  m_png(width = 600)
```

`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 = m_style_stick(radius = 2)) %>%
  m_zoom_to() %>%
  m_add_property_labels(
    prop = "index",
    sel = list(not = list(elem = "H")),
    style = m_style_label(
      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(
    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_sel *Selection Function for r3dmol*

Description

Provides documentation for some basic useful selection criteria. For more advanced selection options, see the Official Documenation

Usage

```
m_sel(
  model = NULL,
  resi = NULL,
  resn = NULL,
  invert = NULL,
  chain = NULL,
  elem = NULL,
  atom = NULL,
  byres = NULL,
  b = NULL,
  expand = NULL,
  bonds = NULL,
  ss = NULL,
  clickable = NULL,
  callback = NULL
)
```

Arguments

model	a single model or list of models from which atoms should be selected. Can also specify by numerical creation order. Reverse indexing is allowed (-1 specifies last added model).
resi	Residue number/s. (vector)
resn	Parent residue name as 3-letter code (e.g. "ALA", "GLY", "CYS"...)
invert	Logical, if `invert = TRUE`, Inverts the selection criteria.
chain	String, chain this atom belongs to (e.g. 'A' for chain A)
elem	element abbreviation (e.g 'H', 'Ca', etc)
atom	Atom name, may be more specific than 'elem' (e.g. 'CA' for alpha carbon)
byres	Logical, if `byres = TRUE`, expands the selection to entire residues that include any selected atoms.
b	Atom b factor data
expand	Expand selection to include atoms within a specified distance from current selection. all atoms of any residue that has any atom already selected.
bonds	overloaded to select number of bonds, e.g. `bonds = 0` will select all non-bonded atoms
ss	Secondary structure identifier. 'h' for helix, 's' for beta-sheet.
clickable	Set this flag to true to enable click selection handling for this atom
callback	Callback click handler function to be executed on this atom and its parent viewer.

Value

sel `list()` for selecting atoms.

Examples

```
library(r3dmol)
## Not run:
r3dmol() %>%
  m_add_model(data = m_fetch_pdb("1bna")) %>%
  m_add_style(
    style = m_style_stick(),
    sel = m_sel(resi = 1:2)
  ) %>%
  m_zoom_to(sel = m_sel(resi = 1))

# Expand example
r3dmol() %>%
  m_add_model(data = m_fetch_pdb("1bna")) %>%
  m_add_style(
    style = m_style_stick(),
    sel = m_sel(
      resi = 1,
      expand = 10,
      byres = TRUE
    )
  ) %>%
  m_zoom_to(sel = m_sel(resi = 1))

## End(Not run)
```

`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 = m_style_cartoon()) %>%
  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 = m_style_cartoon()) %>%
  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 = m_style_cartoon()) %>%
  m_zoom_to() %>%
  m_set_slab(near = -90, far = 0)
```

`m_set_style` *Add Style to Selection*

Description

Takes a selection and adds additional styling to selection.

Usage

```
m_set_style(id, style = m_style_cartoon(), sel = m_sel())
```

Arguments

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

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 = m_style_cartoon()) %>%
  m_zoom_to()
```

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, 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 = m_style_cartoon()) %>%
  m_set_view(arg = c(20, -20, 10, -200, 0, 1, 0, 0)) %>%
  m_add_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_shape_spec *Specify Styling for Generic Shapes*

Description

Styling options for the various shapes. Used inside `m_add_sphere()`, `m_add_arrow()`, `m_add_cylinder()` etc.

Usage

```
m_shape_spec(
  color = NULL,
  opacity = 1,
  wireframe = FALSE,
  hidden = FALSE,
  frame = NULL,
  clickable = FALSE,
  callback = NULL,
  hoverable = FALSE,
  hover_callback = NULL,
  unhover_callback = NULL
)
```

Arguments

`color`	Solid color values.
`opacity`	Transparency value. 1 for opaque, 0 for invisible.
`wireframe`	Draw as wireframe, not solid surface.
`hidden`	If true, do not display object.
`frame`	If set, only display in this frame of an animation.
`clickable`	If true, user can click on object to trigger callback.
`callback`	Function to call on click.
`hoverable`	Logical, enabling hover_callback and unhover_callback functions to be called. Set hoverDuration in the viewer_spec() of r3dmol().
`hover_callback`	Function to be called upon hover.
`unhover_callback`	Function to be called upon hover stopping.

Examples

```
library(r3dmol)
## Not run:
r3dmol() %>%
  m_add_model(data = m_fetch_pdb("1bna")) %>%
  m_add_sphere(
```

```
    center = m_sel(resi = 1),
    spec = m_shape_spec(color = "green", wireframe = TRUE)
  ) %>%
  m_zoom_to(sel = m_sel(resi = 1))

## End(Not run)
```

`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", speed = 1)
```

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"`.
`speed`	Speed multiplier for spin animation. Defaults to 1. Negative value reverses the direction of spin.

Value

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

Examples

```
library(r3dmol)
model <- r3dmol() %>%
  m_add_model(data = pdb_6zsl, format = "pdb") %>%
  m_set_style(style = m_style_cartoon(color = "spectrum")) %>%
  m_zoom_to()

# spin the model
model %>% m_spin()

# reverses the direction of spin
model %>% m_spin(speed = -0.5)
```

---

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_style_cartoon *Specify Styling for Cartoon*

Description

Styling options for the cartoon representation. Used inside m_add_style() and m_set_style().

Usage

```
m_style_cartoon(
  color = NULL,
  style = "rectangle",
  ribbon = FALSE,
  arrows = TRUE,
  tubes = FALSE,
  thickness = 0.4,
  width = NULL,
  opacity = 1,
  colorfunc = NULL
)
```

Arguments

color	Block color values. Strand color, may specify as 'spectrum' which will apply reversed gradient based on residue number.
style	style of cartoon rendering ("trace", "oval", "rectangle" (default), "parabola", "edged").
ribbon	whether to use constant strand width, disregarding secondary structure; use thickness to adjust radius.
arrows	whether to add arrows showing beta-sheet directionality; does not apply to trace or ribbon.
tubes	whether to display alpha helices as simple cylinders; does not apply to trace.
thickness	cartoon strand thickness, default is 0.4.
width	cartoon strand width, default is secondary structure-dependent; does not apply to trace or ribbon.
opacity	set opacity from 0-1; transparency is set per-chain with a warning outputted in the event of ambiguity.
colorfunc	Allows the user to provide a function for setting the colorSchemes, written in javascript. Official Documentation

Examples

```
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_cartoon(color = "spectrum")) %>%
  m_zoom_to()
```

---

m_style_label *Specify Styling for Labels*

Description

Styling options for the labels. Used inside `m_add_label()`, `m_add_res_labels()` and `m_add_property_labels()`.

Usage

```
m_style_label(
  font = "sans-serif",
  fontSize = 18,
  fontColor = "white",
  fontOpacity = 1,
  backgroundColor = "black",
  backgroundOpacity = 1,
  borderOpacity = 1,
  borderThickness = 0,
  borderColor = backgroundColor,
```

```
  inFront = TRUE,
  showBackground = TRUE,
  fixed = FALSE,
  alignment = c("topLeft", "topCenter", "topRight", "centerLeft", "center",
    "centerRight", "bottomLeft", "bottomCenter", "bottomRight"),
  position = NULL,
  frame = NULL
)
```

Arguments

font	Font name, default sans-serif.
fontSize	Height of text, default 18.
fontColor	Font color, default white.
fontOpacity	Font opacity, default 1.
backgroundColor	Color of background, default black.
backgroundOpacity	Opacity of background, default 1.
borderOpacity	Opacity of border, default 1.
borderThickness	Line width of border around label, default 0.
borderColor	Color of border, default backgroundColor.
inFront	Logical, if TRUE always put in front of model.
showBackground	Logical, show background rounded rectangle, default TRUE.
fixed	Logical, setes the label to change with the model when zooming.
alignment	String, how to orient the label with respect to position: 'topLeft' (default), 'topCenter', 'topRight', 'centerLeft', 'center', 'centerRight', 'bottomLeft', 'bottomCenter', 'bottomRight'.
position	x,y,z coordinates for label (for custom positioning).
frame	If set, only display in this frame of an animation.

Examples

```
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_stick()) %>%
  m_add_res_labels(style = m_style_label(
    fontSize = 14,
    backgroundColor = "green"
  )) %>%
  m_zoom_to()
```

m_style_line *Specify Styling for Lines*

Description

Styling options for the line representation. Used inside `m_add_style()` and `m_set_style()`. Can also be used for styling when adding individual lines with `m_add_line()`.

Usage

```
m_style_line(
  colorScheme = "default",
  color = NULL,
  opacity = 1,
  hidden = FALSE
)
```

Arguments

`colorScheme`	Specify scheme to color the atoms by. Default is "default". Other choies are "Carbon", ssPyMOL", "ssJmol", "Jmol", "default", "amino", "shapely", "nucleic", "chain", "chainHetatm", "prop".
`color`	Fixed coloring, overrides `colorScheme`.
`opacity`	Opacity, must be the same for all atoms in the model.
`hidden`	Logical, do not show line.

Examples

```
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_line(color = "blue")) %>%
  m_zoom_to()
```

m_style_sphere *Specify Styling for Sphere*

Description

Styling options for the sphere representation. Used inside `m_add_style()` and `m_set_style()`.

Usage

```
m_style_sphere(
  scale = 1,
  colorScheme = "default",
  color = NULL,
  radius = NULL,
  hidden = FALSE,
  opacity = 1
)
```

Arguments

scale	Scale radius by specified amount.
colorScheme	Specify scheme to color the atoms by. Default is "default". Other choies are "Carbon", ssPyMOL", "ssJmol", "Jmol", "default", "amino", "shapely", "nucleic", "chain", "chainHetatm", "prop".
color	Discrete, fixed coloring, overrides any colorScheme.
radius	Override van der waals radius.
hidden	Boolean - do not show atom. Default FALSE.
opacity	Opacity of spheres, 0 being invisible. Must be the same for all atoms in the model.

Examples

```
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_sphere(radius = 0.5)) %>%
  m_zoom_to()
```

---

m_style_stick *Specify Styling for Stick*

Description

Styling options for the stick representation. Used inside `m_add_style()` and `m_set_style()`.

Usage

```
m_style_stick(
  radius = 0.3,
  singleBonds = FALSE,
  colorScheme = "default",
  color = NULL,
  opacity = 1,
  hidden = FALSE
)
```

Arguments

radius	Radius of sticks.
singleBonds	Draw all bonds as single bonds if TRUE.
colorScheme	Specify scheme to color the atoms by. Default is "default". Other choies are "Carbon", ssPyMOL", "ssJmol", "Jmol", "default", "amino", "shapely", "nucleic", "chain", "chainHetatm", "prop".
color	Fixed coloring, overrides colorScheme.
opacity	Opacity, must be the same for all atoms in the model.
hidden	Do not show.

Examples

```
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_stick(opacity = 0.4)) %>%
  m_zoom_to()
```

---

m_style_surface *Specify Styling for Surface*

Description

Styling options for the surface representation. Used inside m_add_surface().

Usage

```
m_style_surface(opacity = 1, colorScheme = "default", color = NULL)
```

Arguments

opacity	Opacity, 0 for transparent, 1 for opaque.
colorScheme	Specify scheme to color the atoms by. Default is "default". Other choies are "Carbon", ssPyMOL", "ssJmol", "Jmol", "default", "amino", "shapely", "nucleic", "chain", "chainHetatm", "prop".
color	Fixed coloring, overrides colorScheme.

Examples

```
r3dmol() %>%
  m_add_model(data = pdb_1j72, format = "pdb") %>%
  m_set_style(style = m_style_stick()) %>%
  m_add_surface(style = m_style_surface(opacity = 0.4)) %>%
  m_zoom_to()
```

`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 = c(m_style_cartoon(), m_style_stick())) %>%
  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 = c(m_style_cartoon(), m_style_stick())) %>%
  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 = m_style_stick()) %>%
  m_vibrate(numFrames = 10, amplitude = 1) %>%
  m_animate(options = list(loop = "backAndForth", reps = 0)) %>%
  m_zoom_to()
```

`m_viewer_spec` *Specifying setup options for viewer*

Description

Returns a list for the setup `r3dmol()` function, to set overall settings for the viewer going forward.

Usage

```
m_viewer_spec(
  id = NULL,
  defaultcolors = NULL,
  cartoonQuality = 5,
  antialias = TRUE,
  nomouse = FALSE,
  backgroundColor = "white",
  lowerZoomLimit = 5,
  upperZoomLimit = 400,
  orthographic = FALSE,
  disableFog = FALSE
)
```

Arguments

`id`	id of the canvas.
`defaultcolors`	Object defining default atom colors as atom => color property value pairs for all models within this viewer.
`cartoonQuality`	Defaults to 5.
`antialias`	Logical, disable to decrease quality but improve performance.
`nomouse`	Whether to disable handling of mouse events. Disabled will prevent user interaction.
`backgroundColor`	color of the canvas's background.
`lowerZoomLimit`	Specify how far the user can zoom in.
`upperZoomLimit`	Specify how far the user can zoom out.
`orthographic`	Logical. Setting orthographic instead of perspective representation.
`disableFog`	Logical, disable fog, defaults to `FALSE`

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