Package 'shiny.gosling'

November 26, 2024

Title A Grammar-based Toolkit for Scalable and Interactive Genomics Data Visualization for R and Shiny

Version 1.2.0

Language en-US

Description A Grammar-based Toolkit for Scalable and Interactive Genomics Data Visualization. http://gosling-lang.org/. This R package is based on gosling.js. It uses R functions to create gosling plots that could be embedded onto R Shiny apps.

License LGPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

biocViews ShinyApps, Genetics, Visualization

Imports htmltools, jsonlite, rlang, shiny, shiny.react, fs, digest, rjson

Suggests config, covr, knitr, lintr, mockery (>= 0.4.3), rcmdcheck, rmarkdown, sessioninfo, spelling, testthat (>= 3.0.0), GenomicRanges, VariantAnnotation, StructuralVariantAnnotation, biovizBase, ggbio

VignetteBuilder knitr

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/shiny.gosling

git_branch RELEASE_3_20

git_last_commit 892dca2

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-25

Author Appsilon [aut, cre], Anirban Shaw [aut] (<https://orcid.org/0000-0003-4021-513X>), Federico Rivadeneira [aut] (<https://orcid.org/0000-0001-7818-1225>), Vedha Viyash [aut]

Maintainer Appsilon <opensource@appsilon.com>

Contents

add_file_to_resource_path *Track data object builder for local csv files*

Description

Get an object for using local csv to build plots

Usage

```
add_file_to_resource_path(file_path = NULL, object = NULL)
```

Arguments

file_path	A character. Specify the file_path to the local csv file.
object	A gr ranges object.

Value

list of data specs for a local csv file

add_mark *Visual marks*

Description

Visual marks (e.g., points, lines, and bars) are the basic graphical elements of a visualization.

Usage

```
add_mark(
  x = NULL,
  xe = NULL,
  x1 = NULL,
  x1e = NULL,
  y = NULL,
  strokeWidth = NULL,
  opacity = NULL,
  row = NULL,
  size = NULL,
  color = NULL,
  stroke = NULL
)
```

Arguments

x	An object returned by visual_channel_x().
xe	An object returned by visual_channel_x().
x1	An object returned by visual_channel_x().
x1e	An object returned by visual_channel_x().
y	An object returned by visual_channel_y().
strokeWidth	A number or an object returned by visual_channel_stroke_width().
opacity	A number or an object returned by visual_channel_opacity().
row	A factor data column Channel row is used with channel y to stratify a visualization with categorical values.
size	A number or an object returned by visual_channel_size().
color	A character or an object returned by visual_channel_color().
stroke	A number or an object returned by visual_channel_stroke().

Details

For info visit http://gosling-lang.org/docs/mark

Value

list of mark specifications

---

add_multi_tracks *Combine single tracks.*

Description

Combine single tracks.

Usage

```
add_multi_tracks(...)
```

Arguments

...	Multiple tracks from add_single_track() function.

Value

json list.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
      json_list(
        chromosomeField = "chr2",
        genomicFields = c("start2", "end2")
      )
    )
  )
  track5_tracks <- add_multi_tracks(
    add_single_track(
      mark = "rect"
    ),
    add_single_track(
      mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
      strokeWidth = 0
    )
  )
  track5_color <- visual_channel_color(
    field = "svclass",
    type = "nominal",
    legend = TRUE,
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
  track5_stroke <- visual_channel_stroke(
    field = "svclass",
    type = "nominal",
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
  track5_x <- visual_channel_x(field = "start1", type = "genomic")
  track5_xe <- visual_channel_x(field = "end2", type = "genomic")
  track5 <- add_single_track(
    id = "track5", title = "Structural Variant",
```

```
  data = track5_data, mark = "withinLink",
  x = track5_x, xe = track5_xe,
  color = track5_color, width = 500, height = 80, stroke = track5_stroke,
  strokeWidth = 1, opacity = 0.6, style = track5_styles
)

composed_track <- compose_view(
  multi = TRUE,
  tracks = add_multi_tracks(
    track5
  ),
  xOffset = 190, layout = "circular", spacing = 1
)

composed_views <- arrange_views(
  views = composed_track,
  arrangement = "vertical"
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot"))
  )
)


server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
      component_id = "component_2",
      composed_views, clean_braces = FALSE
    )
  })
}

shinyApp(ui, server)

}
```

`add_single_track` *Add a single track*

Description

Add a single track to the plot of a mark type (plot type). This function constructs a single track from the inputs. The inputs can be id, data, mark etc. Please check gosling.js documentation for usage.

Usage

```
add_single_track(
  id = NULL,
  data = NULL,
```

```
  mark = NULL,
  assembly = NULL,
  row = NULL,
  size = NULL,
  color = NULL,
  strokeWidth = NULL,
  opacity = NULL,
  x = NULL,
  xe = NULL,
  x1 = NULL,
  x1e = NULL,
  y = NULL,
  stroke = NULL,
  width = NULL,
  height = NULL,
  dataTransform = NULL,
  ...
)
```

Arguments

id	Optional argument to assign an id to the track.
data	An object of from track_data() function.
mark	Type of plot. One of c("point", "line", "rect", "bar", "area", "link", "triangle", "text"). Each mark type has some supported visual channel. Different marks support different visual channels: • point: x, y, row, size, color, strokeWidth, opacity • line: x, y, row, color, strokeWidth • rect: x, xe, row, color, strokeWidth, opacity • bar: x, y, row, color, strokeWidth, opacity • area: x, y, row, color, strokeWidth • link: x, xe, x1, x1e, color, opacity • triangle: x, xe, row, size, color, opacity • text: x, xe, row, color, opacity For more info visit http://gosling-lang.org/tutorials/
assembly	Currently support "hg38", "hg19", "hg18", "hg17", "hg16", "mm10", "mm9". Defaults to "hg38".
row	An object of from visual_channel_row().
size	An object of from visual_channel_size() OR an atomic number.
color	An object of from visual_channel_color() OR and atomic character hex code of the form "#123456".
strokeWidth	An object of from visual_channel_stroke_width() OR an atomic number.
opacity	An object of from visual_channel_opacity() OR and atomic ratio from 0 to 1.
x	An object of from visual_channel_x() OR an atomic value.
xe	An object of from visual_channel_x() OR an atomic value.
x1	An object of from visual_channel_x() OR an atomic value.
x1e	An object of from visual_channel_x() OR an atomic value.

y	An object of from visual_channel_y() OR an atomic value.
stroke	An object of from visual_channel_stroke() function OR a character of hex color code like "#123456".
width	A number interpreted in units of pixel.
height	A number interpreted in units of pixel.
dataTransform	An object of from track_data_transform() function.
...	Any other arguments to be passed onto gosling.js.

Value

list object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
      binSize = 4,
    ),
    mark = "rect",
    x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
    xe = visual_channel_x(field = "end", type = "genomic"),
    row = visual_channel_row(
      field = "sample",
      type = "nominal",
      legend = TRUE
    ),
    color = visual_channel_color(
      field = "peak",
      type = "quantitative",
      legend = TRUE
    ),
    tooltip = visual_channel_tooltips(
      visual_channel_tooltip(field = "start", type = "genomic",
                             alt = "Start Position"),
      visual_channel_tooltip(field = "end", type = "genomic",
                             alt = "End Position"),
      visual_channel_tooltip(
        field = "peak",
        type = "quantitative",
        alt = "Value",
        format = "0.2"
```

```
      )
    ),
    width = 600,
    height = 130
  )

  single_composed_track <- compose_view(
    tracks = single_track
  )

  single_composed_views <- arrange_views(
    title = "Single Track",
    subtitle = "This is the simplest single track visualization with a linear layout",
    layout = "circular", #"linear"
    views = single_composed_track,
    xDomain = list(
      chromosome = "chr1",
      interval = c(1, 3000500)
    )
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot")),
      column(
        1, br(), actionButton(
          "download_png",
          "PNG",
          icon = icon("cloud-arrow-down")
        )
      )
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_1",
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$download_png, {
      export_png(component_id = "component_1")
    })
  }

  shinyApp(ui, server)

}
```

---

arrange_views *Arrange views*

Description

Arrange views from compose_view() function.

Usage

```
arrange_views(layout = NULL, views = NULL, listify = TRUE, ...)
```

Arguments

layout	One of "linear" or "circular".
views	An object from compose_view() function.
listify	A Boolean. Convert views to list..
...	More options passed to gosling.js.

Value

list object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  # View 2 Track 3----
  view2_track3_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=NC_045512_2-multivec",
    type = "multivec",
    row = "base",
    column = "position",
    value = "count",
    categories = c("A", "T", "G", "C"),
    start = "start",
    end = "end"
  )

  view2_track3a <- add_single_track(
    mark = "bar",
    y = visual_channel_y(
      field = "count", type = "quantitative", axis = "none"
    )
  )

  view2_track3b <- add_single_track(
    dataTransform = track_data_transform(
      type = "filter",
      field = "count",
      oneOf = list(0),
      not = TRUE
    ),
    mark = "text",
    x = visual_channel_x(
      field = "start", type = "genomic"
    ),
    xe = visual_channel_x(
```

```
    field = "end", type = "genomic"
  ),
  size = 24,
  color = "white",
  visibility = list(list(
    operation = "less-than",
    measure = "width",
    threshold = "|xe-x|",
    transitionPadding = 30,
    target = "mark"
  ),
  list(
    operation = "LT",
    measure = "zoomLevel",
    threshold = 40,
    target = "track"
  ))
)

view2_track3_x <- visual_channel_x(
  field = "position", type = "genomic"
)

view2_track3_color <- visual_channel_color(
  field = "base",
  type = "nominal",
  domain = c("A", "T", "G", "C"),
  legend = TRUE
)

view2_track3_text <- visual_channel_text(
  field = "base", type = "nominal"
)

view2_track3_style <- default_track_styles(
  inlineLegend = TRUE
)

view2_track3 <- add_single_track(
  title = "NC_045512.2 Sequence",
  alignment = "overlay",
  data = view2_track3_data,
  tracks = add_multi_tracks(
    view2_track3a, view2_track3b
  ),
  x = view2_track3_x,
  color = view2_track3_color,
  text = view2_track3_text,
  style = view2_track3_style,
  width = 800, height = 40
)

view2 <- compose_view(
  multi = TRUE,
  centerRadius = 0,
  xDomain = list(interval = c(1, 29903)),
  linkingId = "detail",
```

```
    alignment = "stack",
    tracks = add_multi_tracks(
      view2_track3
    )
  )

  combined_view <- arrange_views(
    title = "SARS-CoV-2",
    subtitle = "Data Source: WashU Virus Genome Browser, NCBI, GISAID",
    assembly = list(list("NC_045512.2", 29903)),
    layout = "linear",
    spacing = 50,
    views = list(view2),
    listify = FALSE
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "sars_cov2",
        combined_view
      )
    })
  }

  shinyApp(ui, server)

}
```

`atomic_values_to_list` *atomic_values_to_list*

Description

atomic_values_to_list

Usage

```
atomic_values_to_list(property_list)
```

Arguments

`property_list`	A character or number or another atomic value.

Value

List.

brush_styles *style of the brush mark*

Description

Customize the style of the brush mark in the `rangeSelect` mouse event.

Usage

```
brush_styles(
  strokeWidth = NULL,
  strokeOpacity = NULL,
  stroke = NULL,
  opacity = NULL,
  color = NULL
)
```

Arguments

`strokeWidth`	A number. stroke width of the marks when mouse events are triggered.
`strokeOpacity`	A number.
`stroke`	A character. Stroke color of the marks when mouse events are triggered.
`opacity`	A number. Opacity of the marks when mouse events are triggered.
`color`	A character. Color of the marks when mouse events are triggered.

Details

For more info visit http://gosling-lang.org/docs/visual-channel/#type-brush

Value

List object with brush styles.

build_json *Build gosling spec from R list*

Description

Build gosling spec from R list

Usage

```
build_json(r_list, clean_braces = TRUE, pretty = TRUE, auto_unbox = TRUE)
```

Arguments

`r_list`	R list object built with other gosling functions
`clean_braces`	Whether to remove extra square brackets from the json string.
`pretty`	Whether to get json with indentation, line breaks etc.
`auto_unbox`	If TRUE will automatically unbox() all atomic vectors of length 1.

Value

json spec for the gosling output

`component` *Create react component*

Description

Create react component

Usage

```
component(name)
```

Arguments

`name`	name of the react component

Value

function to create react element

`compose_view` *Compose views*

Description

Compose views from add_single_track() and add_multi_tracks() functions.

Usage

```
compose_view(
  multi = FALSE,
  layout = NULL,
  width = NULL,
  height = NULL,
  centerRadius = NULL,
  tracks,
  ...
)
```

Arguments

`multi`	Whether multiple tracks in the view.
`layout`	One of "linear" or "circular".
`width`	A number interpreted in units of pixel.
`height`	A number interpreted in units of pixel.
`centerRadius`	Specify the proportion of the radius of the center white space. A number between c(0,1), default=0.3
`tracks`	The tracks with add_multi_tracks() function.
`...`	More arguments passed along with view to gosling.js.

Value

list object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  # View 2 Track 3----
  view2_track3_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=NC_045512_2-multivec",
    type = "multivec",
    row = "base",
    column = "position",
    value = "count",
    categories = c("A", "T", "G", "C"),
    start = "start",
    end = "end"
  )

  view2_track3a <- add_single_track(
    mark = "bar",
    y = visual_channel_y(
      field = "count", type = "quantitative", axis = "none"
    )
  )

  view2_track3b <- add_single_track(
    dataTransform = track_data_transform(
      type = "filter",
      field = "count",
      oneOf = list(0),
      not = TRUE
    ),
    mark = "text",
    x = visual_channel_x(
      field = "start", type = "genomic"
    ),
    xe = visual_channel_x(
      field = "end", type = "genomic"
    ),
    size = 24,
    color = "white",
    visibility = list(list(
      operation = "less-than",
      measure = "width",
      threshold = "|xe-x|",
      transitionPadding = 30,
      target = "mark"
    ),
    list(
      operation = "LT",
      measure = "zoomLevel",
      threshold = 40,
      target = "track"
```

```
  ))
)

view2_track3_x <- visual_channel_x(
  field = "position", type = "genomic"
)

view2_track3_color <- visual_channel_color(
  field = "base",
  type = "nominal",
  domain = c("A", "T", "G", "C"),
  legend = TRUE
)

view2_track3_text <- visual_channel_text(
  field = "base", type = "nominal"
)

view2_track3_style <- default_track_styles(
  inlineLegend = TRUE
)

view2_track3 <- add_single_track(
  title = "NC_045512.2 Sequence",
  alignment = "overlay",
  data = view2_track3_data,
  tracks = add_multi_tracks(
    view2_track3a, view2_track3b
  ),
  x = view2_track3_x,
  color = view2_track3_color,
  text = view2_track3_text,
  style = view2_track3_style,
  width = 800, height = 40
)

view2 <- compose_view(
  multi = TRUE,
  centerRadius = 0,
  xDomain = list(interval = c(1, 29903)),
  linkingId = "detail",
  alignment = "stack",
  tracks = add_multi_tracks(
    view2_track3
  )
)

combined_view <- arrange_views(
  title = "SARS-CoV-2",
  subtitle = "Data Source: WashU Virus Genome Browser, NCBI, GISAID",
  assembly = list(list("NC_045512.2", 29903)),
  layout = "linear",
  spacing = 50,
  views = list(view2),
  listify = FALSE
)
```

```
  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "sars_cov2",
        combined_view
      )
    })
  }

  shinyApp(ui, server)

}
```

`default_track_styles` *Default styles for tracks*

Description

Default styles for tracks

Usage

```
default_track_styles(
  textStrokeWidth = NULL,
  textStroke = NULL,
  textFontWeight = NULL,
  textFontSize = NULL,
  textAnchor = NULL,
  select = NULL,
  outlineWidth = NULL,
  outline = NULL,
  mouseOver = NULL,
  matrixExtent = NULL,
  linkStyle = NULL,
  linkMinHeight = NULL,
  linkConnectionType = NULL,
  linePattern = NULL,
  legendTitle = NULL,
  inlineLegend = NULL,
  enableSmoothPath = NULL,
  dy = NULL,
  dx = NULL,
  dashed = NULL,
  curve = NULL,
```

```
  brush = NULL,
  backgroundOpacity = NULL,
  background = NULL,
  align = NULL,
  ...
)
```

Arguments

textStrokeWidth	A number. Specify the stroke width of text marks. Can also be specified using the strokeWidth channel option of text marks.
textStroke	A character. Specify the stroke of text marks. Can also be specified using the stroke channel option of text marks.
textFontWeight	A character. One of "bold", "normal". Specify the font weight of text marks.
textFontSize	A number. Specify the font size of text marks. Can also be specified using the size channel option of text marks.
textAnchor	A character. One of "start", "middle", "end". Specify the alignment of text marks to a given point.
select	An object returned by event_styles(). Customize visual effects of rangeSelect events on marks.
outlineWidth	A number.
outline	A character.
mouseOver	An object returned by event_styles(). Customize visual effects of mouseOver events on marks.
matrixExtent	A character.One of "full", "upper-right", "lower-left". Determine to show only one side of the diagonal in a HiGlass matrix. Default: "full".
linkStyle	A character. One of "elliptical", "circular", "straight", "experimentalEdgeBundling". The style of withinLink and betweenLink marks. Default: 'circular' 'elliptical' will be used as a default option.
linkMinHeight	A number. The minimum height of withinLink and betweenLink marks. Unit is a percentage Default: 0.5.
linkConnectionType	A character. One of "straight", "curve", "corner". Specify the connection type of betweenLink marks. Default: "corner".
linePattern	A list of the form list(size="number",type="string"). One of "triangleLeft", "triangleRight".) Specify the pattern of dashes and gaps for rule marks.
legendTitle	A character. If defined, show legend title on the top or left.
inlineLegend	A Boolean. Specify whether to show legend in a single horizontal line?
enableSmoothPath	A Boolean. Whether to enable smooth paths when drawing curves. Default: FALSE.
dy	A number. Offset the position of marks in y direction. This property is currently only supported for text marks.
dx	A number. Offset the position of marks in x direction. This property is currently only supported for text marks.
dashed	An vector of number like c(1, 2). Specify the pattern of dashes and gaps for rule marks.

curve	A character. One of "top", "bottom", "left", "right". Specify the curve of rule marks.
brush	An object returned by brush_styles(). Customize the style of the brush mark in the rangeSelect mouse event.
backgroundOpacity	A number.
background	A character.
align	A character. One of "left", "right". Specify the alignment of marks. This property is currently only supported for triangle marks.
...	Any other styles to be passed to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel/#style-related-properties

Value

List object with default styles.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
      json_list(
        chromosomeField = "chr2",
        genomicFields = c("start2", "end2")
      )
    )
  )
  track5_tracks <- add_multi_tracks(
    add_single_track(
      mark = "rect"
    ),
    add_single_track(
      mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
      strokeWidth = 0
    )
  )
  track5_color <- visual_channel_color(
    field = "svclass",
    type = "nominal",
    legend = TRUE,
```

```
  domain = json_list(
    "tandem-duplication", "translocation", "deletion", "inversion"
  ),
  range = json_list(
    "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
  )
)
track5_stroke <- visual_channel_stroke(
  field = "svclass",
  type = "nominal",
  domain = json_list(
    "tandem-duplication", "translocation", "deletion", "inversion"
  ),
  range = json_list(
    "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
  )
)
track5_x <- visual_channel_x(field = "start1", type = "genomic")
track5_xe <- visual_channel_x(field = "end2", type = "genomic")
track5 <- add_single_track(
  id = "track5", title = "Structural Variant",
  data = track5_data, mark = "withinLink",
  x = track5_x, xe = track5_xe,
  color = track5_color, width = 500, height = 80, stroke = track5_stroke,
  strokeWidth = 1, opacity = 0.6, style = track5_styles
)

composed_track <- compose_view(
  multi = TRUE,
  tracks = add_multi_tracks(
    track5
  ),
  xOffset = 190, layout = "circular", spacing = 1
)

composed_views <- arrange_views(
  views = composed_track,
  arrangement = "vertical"
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot"))
  )
)


server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
      component_id = "component_2",
      composed_views, clean_braces = FALSE
    )
  })
}
```

```
  shinyApp(ui, server)

}
```

`event_styles` *Mouse event styles*

Description

The styles defined here will be applied to the targets of mouse events, such as a point mark after user click mouse.

Usage

```
event_styles(
  strokeWidth = NULL,
  strokeOpacity = NULL,
  stroke = NULL,
  opacity = NULL,
  color = NULL,
  arrange = NULL
)
```

Arguments

`strokeWidth`	A number. stroke width of the marks when mouse events are triggered.
`strokeOpacity`	A number.
`stroke`	A character. Stroke color of the marks when mouse events are triggered.
`opacity`	A number. Opacity of the marks when mouse events are triggered.
`color`	A character. Color of the marks when mouse events are triggered.
`arrange`	A character. One of "behind", "front". Show event effects behind or in front of marks.

Details

For more info visit http://gosling-lang.org/docs/visual-channel/#type-eventstyle

Value

List object with event styles.

export_pdf *Export PDF*

Description

Exports PDF

Usage

```
export_pdf(
  component_id,
  transparent_background = FALSE,
  session = getDefaultReactiveDomain()
)
```

Arguments

component_id	A character. The id of the component_id prop passed to the GoslingComponent function.
transparent_background	A Boolean. Determine if the background should be transparent or not (Default: false).
session	A shiny session object.

Value

None.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
      binSize = 4,
    ),
    mark = "rect",
    x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
    xe = visual_channel_x(field = "end", type = "genomic"),
    row = visual_channel_row(
      field = "sample",
```

```
    type = "nominal",
    legend = TRUE
  ),
  color = visual_channel_color(
    field = "peak",
    type = "quantitative",
    legend = TRUE
  ),
  tooltip = visual_channel_tooltips(
    visual_channel_tooltip(field = "start", type = "genomic",
                           alt = "Start Position"),
    visual_channel_tooltip(field = "end", type = "genomic",
                           alt = "End Position"),
    visual_channel_tooltip(
      field = "peak",
      type = "quantitative",
      alt = "Value",
      format = "0.2"
    )
  ),
  width = 600,
  height = 130
)

single_composed_track <- compose_view(
  tracks = single_track
)

single_composed_views <- arrange_views(
  title = "Single Track",
  subtitle = "This is the simplest single track visualization with a linear layout",
  layout = "circular", #"linear"
  views = single_composed_track,
  xDomain = list(
    chromosome = "chr1",
    interval = c(1, 3000500)
  )
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot")),
    column(
      1, br(), actionButton(
        "download_pdf",
        "PDF",
        icon = icon("cloud-arrow-down")
      )
    )
  )
)


server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
```

```
        component_id = "component_1",
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$download_pdf, {
      export_pdf(component_id = "component_1")
    })
  }

  shinyApp(ui, server)

}
```

`export_png` *Export PNG*

Description

Exports PNG

Usage

```
export_png(
  component_id,
  transparent_background = FALSE,
  session = getDefaultReactiveDomain()
)
```

Arguments

`component_id`	A character. The id of the component_id prop passed to the GoslingComponent function.
`transparent_background`	A Boolean. Determine if the background should be transparent or not (Default: false).
`session`	A shiny session object.

Value

None.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"
```

```
single_track <- add_single_track(
  id = "track1",
  data = track_data(
    url = cistrome_data,
    type = "multivec",
    row = "sample",
    column = "position",
    value = "peak",
    categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
    binSize = 4,
  ),
  mark = "rect",
  x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
  xe = visual_channel_x(field = "end", type = "genomic"),
  row = visual_channel_row(
    field = "sample",
    type = "nominal",
    legend = TRUE
  ),
  color = visual_channel_color(
    field = "peak",
    type = "quantitative",
    legend = TRUE
  ),
  tooltip = visual_channel_tooltips(
    visual_channel_tooltip(field = "start", type = "genomic",
                           alt = "Start Position"),
    visual_channel_tooltip(field = "end", type = "genomic",
                           alt = "End Position"),
    visual_channel_tooltip(
      field = "peak",
      type = "quantitative",
      alt = "Value",
      format = "0.2"
    )
  ),
  width = 600,
  height = 130
)

single_composed_track <- compose_view(
  tracks = single_track
)

single_composed_views <- arrange_views(
  title = "Single Track",
  subtitle = "This is the simplest single track visualization with a linear layout",
  layout = "circular", #"linear"
  views = single_composed_track,
  xDomain = list(
    chromosome = "chr1",
    interval = c(1, 3000500)
  )
)

ui <- fluidPage(
  use_gosling(),
```

```
    fluidRow(
      column(6, goslingOutput("gosling_plot")),
      column(
        1, br(), actionButton(
          "download_png",
          "PNG",
          icon = icon("cloud-arrow-down")
        )
      )
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_1",
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$download_png, {
      export_png(component_id = "component_1")
    })
  }

  shinyApp(ui, server)

}
```

`get_file_track_data` *Track data object builder for local csv files*

Description

Get an object for using local csv to build plots

Usage

```
get_file_track_data(
  file_name,
  chromosomeField = NULL,
  genomicFields = NULL,
  separator = ",",
  sampleLength = 1000,
  headerNames = NULL,
  ...
)
```

Arguments

file_name	A character. Specify the file_name.
chromosomeField	A character. Specify the name of chromosome data fields.
genomicFields	A character vector. Specify the name of genomic data fields.
separator	A character. Specify file separator, Default: ','
sampleLength	A number. Specify the number of rows loaded from the URL. Default: 1000
headerNames	A character vector. Specify the names of data fields if a CSV file does not have header row.
...	Any other parameters passed to json data object.

Value

list of data specs for a local csv file

---

gosling *Build gosling plot object*

Description

Build gosling plot object

Usage

```
gosling(component_id, composed_views, clean_braces = TRUE)
```

Arguments

component_id	Assign a component id to use other api like zoom.
composed_views	The views composed with arrange_views.
clean_braces	Whether to remove extra square brackets from the json string.

Value

Gosling component for rendering on R shiny apps

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.gosling)

  # Circular track 1 ----
  circular_track1_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec",
    type = "multivec",
    row = "sample",
    column = "position",
    value = "peak",
    categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
```

```
  binSize = 4
)

circular_track1_x <- visual_channel_x(field = "start", type = "genomic")
circular_track1_xe <- visual_channel_x(field = "end", type = "genomic")

circular_track1_y <- visual_channel_y(field = "peak", type = "quantitative")

circular_track1_row <- visual_channel_row(
  field = "sample", type = "nominal"
)

circular_track1_color <- visual_channel_color(
  field = "sample", type = "nominal"
)

circular_track1_tracks <- add_multi_tracks(
  add_single_track(
    mark = "bar"
  ),
  add_single_track(
    mark = "brush",
    x = visual_channel_x(linkingId = "detail-1"),
    color = "blue"
  ),
  add_single_track(
    mark = "brush",
    x = visual_channel_x(linkingId = "detail-2"),
    color = "red"
  )
)

circular_track1_styles <- default_track_styles(
  outlineWidth = 0
)

circular_track1 <- add_single_track(
  id = "circular_track1", alignment = "overlay", data = circular_track1_data,
  x = circular_track1_x, xe = circular_track1_xe,
  y = circular_track1_y, row = circular_track1_row,
  color = circular_track1_color,
  stroke = "black", strokeWidth = 0.3,
  tracks = circular_track1_tracks,
  style = circular_track1_styles,
  width = 500, height = 100
)

# Compose Circular track ----
circular_composed_view <- compose_view(
  multi = TRUE,
  tracks = add_multi_tracks(
    circular_track1
  ),
  static = TRUE, layout = "circular", alignment = "stack"
)

# Arrange final view
```

```
  circular_linear_view <- arrange_views(
    arrangement = "horizontal",
    views = list(circular_composed_view)
  )




  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "circular_component",
        circular_linear_view, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)
}
```

`GoslingComponent` *Create Gosling component*

Description

Create Gosling component

Usage

```
GoslingComponent(...)
```

Arguments

`...`	Name of component.

Value

A function to create the gosling component.

goslingDependency *Setup gosling dependencies*

Description

Setup gosling dependencies

Usage

```
goslingDependency()
```

Value

list of dependencies for Gosling

goslingOutput *gosling output function*

Description

gosling output function for shiny use. Must use this function instead of shiny output functions.

Usage

```
goslingOutput(outputId)
```

Arguments

outputId	ID of the output element

Value

reactOutput HTML for UI render

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
```

```
    json_list(
      chromosomeField = "chr2",
      genomicFields = c("start2", "end2")
    )
  )
)
track5_tracks <- add_multi_tracks(
  add_single_track(
    mark = "rect"
  ),
  add_single_track(
    mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
    strokeWidth = 0
  )
)
track5_color <- visual_channel_color(
  field = "svclass",
  type = "nominal",
  legend = TRUE,
  domain = json_list(
    "tandem-duplication", "translocation", "deletion", "inversion"
  ),
  range = json_list(
    "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
  )
)
track5_stroke <- visual_channel_stroke(
  field = "svclass",
  type = "nominal",
  domain = json_list(
    "tandem-duplication", "translocation", "deletion", "inversion"
  ),
  range = json_list(
    "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
  )
)
track5_x <- visual_channel_x(field = "start1", type = "genomic")
track5_xe <- visual_channel_x(field = "end2", type = "genomic")
track5 <- add_single_track(
  id = "track5", title = "Structural Variant",
  data = track5_data, mark = "withinLink",
  x = track5_x, xe = track5_xe,
  color = track5_color, width = 500, height = 80, stroke = track5_stroke,
  strokeWidth = 1, opacity = 0.6, style = track5_styles
)

composed_track <- compose_view(
  multi = TRUE,
  tracks = add_multi_tracks(
    track5
  ),
  xOffset = 190, layout = "circular", spacing = 1
)

composed_views <- arrange_views(
  views = composed_track,
  arrangement = "vertical"
```

```
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_2",
        composed_views, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)

}
```

`is_atomic_field` *is_atomic_field*

Description

is_atomic_field

Usage

```
is_atomic_field(field_name)
```

Arguments

`field_name`	A character or number or another atomic value.

Value

List.

`json_list` *Create list*

Description

Create list

Usage

```
json_list(...)
```

Arguments

...	Items to be put in a list

Value

list of items

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
      json_list(
        chromosomeField = "chr2",
        genomicFields = c("start2", "end2")
      )
    )
  )
  track5_tracks <- add_multi_tracks(
    add_single_track(
      mark = "rect"
    ),
    add_single_track(
      mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
      strokeWidth = 0
    )
  )
  track5_color <- visual_channel_color(
    field = "svclass",
    type = "nominal",
    legend = TRUE,
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
  track5_stroke <- visual_channel_stroke(
    field = "svclass",
    type = "nominal",
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
```

```
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
  track5_x <- visual_channel_x(field = "start1", type = "genomic")
  track5_xe <- visual_channel_x(field = "end2", type = "genomic")
  track5 <- add_single_track(
    id = "track5", title = "Structural Variant",
    data = track5_data, mark = "withinLink",
    x = track5_x, xe = track5_xe,
    color = track5_color, width = 500, height = 80, stroke = track5_stroke,
    strokeWidth = 1, opacity = 0.6, style = track5_styles
  )

  composed_track <- compose_view(
    multi = TRUE,
    tracks = add_multi_tracks(
      track5
    ),
    xOffset = 190, layout = "circular", spacing = 1
  )

  composed_views <- arrange_views(
    views = composed_track,
    arrangement = "vertical"
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_2",
        composed_views, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)
}
```

`list_rm_null` *Remove null from list*

Description

Remove null from list

Usage

```
list_rm_null(r_list)
```

Arguments

r_list	An r list with NULL values

Value

r list without NULL values

---

print.gosling *Print method for the gosling component*

Description

Print method for the gosling component

Usage

```
## S3 method for class 'gosling'
print(x, ...)
```

Arguments

x	A gosling object
...	further arguments passed to or from other methods.

Value

r list without NULL values

---

renderGosling *gosling render function*

Description

gosling render function for shiny use

Usage

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

Arguments

expr	Expression returning the HTML / 'React' to render.
env	Environment in which to evaluate expr.
quoted	Is expr a quoted expression?

Value

A function which can be assigned to an output in a Shiny server function.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
      json_list(
        chromosomeField = "chr2",
        genomicFields = c("start2", "end2")
      )
    )
  )
  track5_tracks <- add_multi_tracks(
    add_single_track(
      mark = "rect"
    ),
    add_single_track(
      mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
      strokeWidth = 0
    )
  )
  track5_color <- visual_channel_color(
    field = "svclass",
    type = "nominal",
    legend = TRUE,
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
  track5_stroke <- visual_channel_stroke(
    field = "svclass",
    type = "nominal",
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
```

```
  track5_x <- visual_channel_x(field = "start1", type = "genomic")
  track5_xe <- visual_channel_x(field = "end2", type = "genomic")
  track5 <- add_single_track(
    id = "track5", title = "Structural Variant",
    data = track5_data, mark = "withinLink",
    x = track5_x, xe = track5_xe,
    color = track5_color, width = 500, height = 80, stroke = track5_stroke,
    strokeWidth = 1, opacity = 0.6, style = track5_styles
  )

  composed_track <- compose_view(
    multi = TRUE,
    tracks = add_multi_tracks(
      track5
    ),
    xOffset = 190, layout = "circular", spacing = 1
  )

  composed_views <- arrange_views(
    views = composed_track,
    arrangement = "vertical"
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_2",
        composed_views, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)

}
```

`run_example` *Runs a shiny.gosling example*

Description

Runs a shiny.gosling example

Usage

```
run_example(example = NA)
```

Arguments

example	A character indicating a valid example.

Value

A Shiny App is launched.

Examples

```
if (interactive()) {
  run_example("circularLinearWithBrush")
}
```

---

track_data *Data object builder*

Description

Build the data object for gosling plots

Usage

```
track_data(
  url = NULL,
  type,
  separator = NULL,
  sampleLength = NULL,
  headerNames = NULL,
  genomicFields = NULL,
  chromosomeField = NULL,
  genomicFieldsToConvert = NULL,
  ...
)
```

Arguments

url	A character. Specify the URL address of the data file.
type	A character. Type of data. One of "csv", "json", "bigwig", "bam", "vcf", "vector", "multivec" and "beddb". For usage refer to http://gosling-lang.org/docs/data#supported-data-formats.
separator	A character. Specify file separator, Default: ','
sampleLength	A number. Specify the number of rows loaded from the URL. Default: 1000
headerNames	A character vector. Specify the names of data fields if a CSV file does not have header row.
genomicFields	A character vector. Specify the name of genomic data fields.
chromosomeField	A character. Specify the name of chromosome data fields.
genomicFieldsToConvert	Define the genomic fields from the data in list format. Experimental Property. Each object follows the format "chromosomeField":"string","genomicFields":"string[]" ( )
...	Any other parameters passed to json data object.

Details

For info visit http://gosling-lang.org/docs/data. Check the various supported data formats and their parameters. All of them can be constructed using this function.

Value

list of data specs

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.gosling)

  # View 2 Track 3----
  view2_track3_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=NC_045512_2-multivec",
    type = "multivec",
    row = "base",
    column = "position",
    value = "count",
    categories = c("A", "T", "G", "C"),
    start = "start",
    end = "end"
  )

  view2_track3a <- add_single_track(
    mark = "bar",
    y = visual_channel_y(
      field = "count", type = "quantitative", axis = "none"
    )
  )

  view2_track3b <- add_single_track(
    dataTransform = track_data_transform(
      type = "filter",
      field = "count",
      oneOf = list(0),
      not = TRUE
    ),
    mark = "text",
    x = visual_channel_x(
      field = "start", type = "genomic"
    ),
    xe = visual_channel_x(
      field = "end", type = "genomic"
    ),
    size = 24,
    color = "white",
    visibility = list(
      list(
        operation = "less-than",
        measure = "width",
        threshold = "|xe-x|",
        transitionPadding = 30,
        target = "mark"
```

```
    ),
    list(
      operation = "LT",
      measure = "zoomLevel",
      threshold = 40,
      target = "track"
    )
  )
)

view2_track3_x <- visual_channel_x(
  field = "position", type = "genomic"
)

view2_track3_color <- visual_channel_color(
  field = "base",
  type = "nominal",
  domain = c("A", "T", "G", "C"),
  legend = TRUE
)

view2_track3_text <- visual_channel_text(
  field = "base", type = "nominal"
)

view2_track3_style <- default_track_styles(
  inlineLegend = TRUE
)

view2_track3 <- add_single_track(
  title = "NC_045512.2 Sequence",
  alignment = "overlay",
  data = view2_track3_data,
  tracks = add_multi_tracks(
    view2_track3a, view2_track3b
  ),
  x = view2_track3_x,
  color = view2_track3_color,
  text = view2_track3_text,
  style = view2_track3_style,
  width = 800, height = 40
)

view2 <- compose_view(
  multi = TRUE,
  centerRadius = 0,
  xDomain = list(interval = c(1, 29903)),
  linkingId = "detail",
  alignment = "stack",
  tracks = add_multi_tracks(
    view2_track3
  )
)

combined_view <- arrange_views(
  title = "SARS-CoV-2",
  subtitle = "Data Source: WashU Virus Genome Browser, NCBI, GISAID",
```

```
    assembly = list(list("NC_045512.2", 29903)),
    layout = "linear",
    spacing = 50,
    views = list(view2),
    listify = FALSE
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "sars_cov2",
        combined_view
      )
    })
  }

  shinyApp(ui, server)
}
```

`track_data_csv` *Data object builder for a csv file*

Description

Build the data object for gosling plots

Usage

```
track_data_csv(
  file,
  genomicFields = NULL,
  chromosomeField = NULL,
  separator = ",",
  sampleLength = 1000,
  headerNames = NULL,
  ...
)
```

Arguments

`file`	A character. Specify the URL address or local file name in the www directory of the data file.
`genomicFields`	A character vector. Specify the name of genomic data fields.
`chromosomeField`	A character. Specify the name of chromosome data fields.

separator	A character. Specify file separator, Default: ','
sampleLength	A number. Specify the number of rows loaded from the URL. Default: 1000
headerNames	A character vector. Specify the names of data fields if a CSV file does not have header row.
...	Any other parameters passed to json data object.

Value

list of data specs for a csv file

Examples

```
if (interactive()) {
library(shiny.gosling)
library(shiny)
library(GenomicRanges)

url <- "https://rb.gy/7y3fx"
temp_file <- file.path(tempdir(), "GSM1295076_CBX6_BF_ChipSeq_mergedReps_peaks.bed.gz")
download.file(url, destfile = temp_file)
df <- read.delim(
  temp_file,
  header = FALSE,
  comment.char = "#"
)
gr <- GRanges(
  seqnames = df$V1,
  ranges = IRanges(df$V2, df$V3)
)

if (!dir.exists("data")) {
  dir.create("data")
}
utils::write.csv(gr, "data/ChipSeqPeaks.csv", row.names = FALSE)

ui <- fluidPage(
  use_gosling(clear_files = FALSE),
  goslingOutput("gosling_plot")
)

track_1 <- add_single_track(
  width = 800,
  height = 180,
  data = track_data_csv(
    "data/ChipSeqPeaks.csv", chromosomeField = "seqnames",
    genomicFields = c("start", "end")
  ),
  mark = "bar",
  x = visual_channel_x(
    field = "start", type = "genomic", axis = "bottom"
  ),
  xe = visual_channel_x(field = "end", type = "genomic"),
  y = visual_channel_y(
    field = "width", type = "quantitative", axis = "right"
  ),
  size = list(value = 5)
```

```
)

composed_view <- compose_view(
  layout = "linear",
  tracks = track_1
)

arranged_view <- arrange_views(
  title = "Basic Marks: bar",
  subtitle = "Tutorial Examples",
  views = composed_view
)

server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
      component_id = "component_1",
      arranged_view
    )
  })
}

shiny::shinyApp(ui, server)
}
```

`track_data_gr` *Data object builder for a GRanges object by locally saving it*

Description

Build the data object for gosling plots

Usage

```
track_data_gr(
  granges,
  chromosomeField = NULL,
  genomicFields = NULL,
  separator = ",",
  sampleLength = 1000,
  headerNames = NULL,
  ...
)
```

Arguments

`granges`	A GRanges object from the GenomicRanges package with seqnames and ranges
`chromosomeField`	A character. Specify the name of chromosome data fields.
`genomicFields`	A character vector. Specify the name of genomic data fields.
`separator`	A character. Specify file separator, Default: ','

sampleLength	A number. Specify the number of rows loaded from the URL. Default: 1000
headerNames	A character vector. Specify the names of data fields if a CSV file does not have header row.
...	Any other parameters passed to json data object.

Value

list of data specs for a csv file

Examples

```
if (interactive()) {
library(shiny.gosling)
library(shiny)
library(GenomicRanges)

url <- "https://rb.gy/7y3fx"
temp_file <- file.path(tempdir(), "GSM1295076_CBX6_BF_ChipSeq_mergedReps_peaks.bed.gz")
download.file(url, destfile = temp_file)
df <- read.delim(
  temp_file,
  header = FALSE,
  comment.char = "#"
)
gr <- GRanges(
  seqnames = df$V1,
  ranges = IRanges(df$V2, df$V3)
)

ui <- fluidPage(
  use_gosling(clear_files = FALSE),
  goslingOutput("gosling_plot")
)

track_1 <- add_single_track(
  width = 800,
  height = 180,
  data = track_data_gr(
    gr, chromosomeField = "seqnames",
    genomicFields = c("start", "end")
  ),
  mark = "bar",
  x = visual_channel_x(
    field = "start", type = "genomic", axis = "bottom"
  ),
  xe = visual_channel_x(field = "end", type = "genomic"),
  y = visual_channel_y(
    field = "width", type = "quantitative", axis = "right"
  ),
  size = list(value = 5)
)

composed_view <- compose_view(
  layout = "linear",
  tracks = track_1
)
```

```
arranged_view <- arrange_views(
  title = "Basic Marks: bar",
  subtitle = "Tutorial Examples",
  views = composed_view
)

server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
      component_id = "component_1",
      arranged_view
    )
  })
}

shiny::shinyApp(ui, server)
}
```

`track_data_transform` *Data transformer*

Description

Do data transformations

Usage

```
track_data_transform(type = NULL, field = NULL, oneOf = NULL, not = NULL, ...)
```

Arguments

`type`	A character. One of "filter", "concat", "replace", "log", "displace", "exonSplit", "coverage", "genomicLength", "svType" and "subjson". Check usage details at http://gosling-lang.org/docs/data/#data-transform.
`field`	A character. filter is applied based on the values of the specified data field.
`oneOf`	A vector of characters or numbers. Check whether the value is an element in the provided list.
`not`	A Boolean. When "not": true, apply a NOT logical operation to the filter. Default: false.
`...`	Any other parameters to pass to gosling.js.

Details

For info visit http://gosling-lang.org/docs/data#data-transform There are multiple ways to transform data. Check documentation for details of usage.

Value

list of data transformations specs

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.gosling)

  # View 2 Track 3----
  view2_track3_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=NC_045512_2-multivec",
    type = "multivec",
    row = "base",
    column = "position",
    value = "count",
    categories = c("A", "T", "G", "C"),
    start = "start",
    end = "end"
  )

  view2_track3a <- add_single_track(
    mark = "bar",
    y = visual_channel_y(
      field = "count", type = "quantitative", axis = "none"
    )
  )

  view2_track3b <- add_single_track(
    dataTransform = track_data_transform(
      type = "filter",
      field = "count",
      oneOf = list(0),
      not = TRUE
    ),
    mark = "text",
    x = visual_channel_x(
      field = "start", type = "genomic"
    ),
    xe = visual_channel_x(
      field = "end", type = "genomic"
    ),
    size = 24,
    color = "white",
    visibility = list(
      list(
        operation = "less-than",
        measure = "width",
        threshold = "|xe-x|",
        transitionPadding = 30,
        target = "mark"
      ),
      list(
        operation = "LT",
        measure = "zoomLevel",
        threshold = 40,
        target = "track"
      )
    )
  )
```

```
view2_track3_x <- visual_channel_x(
  field = "position", type = "genomic"
)

view2_track3_color <- visual_channel_color(
  field = "base",
  type = "nominal",
  domain = c("A", "T", "G", "C"),
  legend = TRUE
)

view2_track3_text <- visual_channel_text(
  field = "base", type = "nominal"
)

view2_track3_style <- default_track_styles(
  inlineLegend = TRUE
)

view2_track3 <- add_single_track(
  title = "NC_045512.2 Sequence",
  alignment = "overlay",
  data = view2_track3_data,
  tracks = add_multi_tracks(
    view2_track3a, view2_track3b
  ),
  x = view2_track3_x,
  color = view2_track3_color,
  text = view2_track3_text,
  style = view2_track3_style,
  width = 800, height = 40
)

view2 <- compose_view(
  multi = TRUE,
  centerRadius = 0,
  xDomain = list(interval = c(1, 29903)),
  linkingId = "detail",
  alignment = "stack",
  tracks = add_multi_tracks(
    view2_track3
  )
)

combined_view <- arrange_views(
  title = "SARS-CoV-2",
  subtitle = "Data Source: WashU Virus Genome Browser, NCBI, GISAID",
  assembly = list(list("NC_045512.2", 29903)),
  layout = "linear",
  spacing = 50,
  views = list(view2),
  listify = FALSE
)

ui <- fluidPage(
  use_gosling(),
```

```
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "sars_cov2",
        combined_view
      )
    })
  }

  shinyApp(ui, server)
}
```

`track_data_transforms` *Combine multiple data transforms*

Description

Combine multiple data transforms

Usage

```
track_data_transforms(...)
```

Arguments

`...`	Multiple data transform specs separated by comma.

Value

list of multiple data transform specs

`use_gosling` *Initiate gosling*

Description

Add this function at the beginning of ui. This is needed for gosling to work in shiny plots.

Usage

```
use_gosling(clear_files = TRUE)
```

Arguments

`clear_files`	default FALSE. To clear the locally stored csv files created by gosling or not.

Value

Gosling initiator HTML.

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
      json_list(
        chromosomeField = "chr2",
        genomicFields = c("start2", "end2")
      )
    )
  )
  track5_tracks <- add_multi_tracks(
    add_single_track(
      mark = "rect"
    ),
    add_single_track(
      mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
      strokeWidth = 0
    )
  )
  track5_color <- visual_channel_color(
    field = "svclass",
    type = "nominal",
    legend = TRUE,
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
  track5_stroke <- visual_channel_stroke(
    field = "svclass",
    type = "nominal",
    domain = json_list(
      "tandem-duplication", "translocation", "deletion", "inversion"
    ),
    range = json_list(
      "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
    )
  )
```

```
  track5_x <- visual_channel_x(field = "start1", type = "genomic")
  track5_xe <- visual_channel_x(field = "end2", type = "genomic")
  track5 <- add_single_track(
    id = "track5", title = "Structural Variant",
    data = track5_data, mark = "withinLink",
    x = track5_x, xe = track5_xe,
    color = track5_color, width = 500, height = 80, stroke = track5_stroke,
    strokeWidth = 1, opacity = 0.6, style = track5_styles
  )

  composed_track <- compose_view(
    multi = TRUE,
    tracks = add_multi_tracks(
      track5
    ),
    xOffset = 190, layout = "circular", spacing = 1
  )

  composed_views <- arrange_views(
    views = composed_track,
    arrangement = "vertical"
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_2",
        composed_views, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)
}
```

`visual_channel` *Generic visual channel builder*

Description

Generic visual channel builder

Usage

```
visual_channel(field = NULL, type = NULL, range = NULL, domain = NULL, ...)
```

Arguments

field	A character. Name of the data field.
type	A character. Must be "genomic". Specify the data type.
range	A vector of characters or numbers. Values of the visual channel.
domain	A vector of characters or numbers. Values of the data.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#encode-a-visual-channel

Value

List object.

---

visual_channel_color *color visual channel*

Description

color visual channel

Usage

```
visual_channel_color(
  field = NULL,
  title = NULL,
  type = NULL,
  scaleOffset = NULL,
  scale = NULL,
  legend = NULL,
  grid = NULL,
  axis = NULL,
  aggregate = NULL,
  ...
)
```

Arguments

field	A character. Name of the data field.
title	A character. Title of the legend. Default: undefined.
type	A character. Must be "genomic". Specify the data type.
scaleOffset	A number vector of the form c(1, 2). Whether to use offset of the domain proportionally. This is bound to brushes on the color legend. Default: c(0, 1).
scale	A character. One of "linear", "log".
legend	A Boolean. Whether to display legend. Default: FALSE.
grid	A Boolean. Whether to display grid. Default: FALSE.

axis	A character. One of "none", "top", "bottom", "left", "right". Specify where should the axis be put.
aggregate	A character. One of "max", "min", "mean", "bin", "count". Specify how to aggregate data. Default: undefined.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#color

Value

List object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  # View 2 Track 3----
  view2_track3_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=NC_045512_2-multivec",
    type = "multivec",
    row = "base",
    column = "position",
    value = "count",
    categories = c("A", "T", "G", "C"),
    start = "start",
    end = "end"
  )

  view2_track3a <- add_single_track(
    mark = "bar",
    y = visual_channel_y(
      field = "count", type = "quantitative", axis = "none"
    )
  )

  view2_track3b <- add_single_track(
    dataTransform = track_data_transform(
      type = "filter",
      field = "count",
      oneOf = list(0),
      not = TRUE
    ),
    mark = "text",
    x = visual_channel_x(
      field = "start", type = "genomic"
    ),
    xe = visual_channel_x(
      field = "end", type = "genomic"
    ),
    size = 24,
    color = "white",
    visibility = list(list(
```

```
    operation = "less-than",
    measure = "width",
    threshold = "|xe-x|",
    transitionPadding = 30,
    target = "mark"
  ),
  list(
    operation = "LT",
    measure = "zoomLevel",
    threshold = 40,
    target = "track"
  ))
)

view2_track3_x <- visual_channel_x(
  field = "position", type = "genomic"
)

view2_track3_color <- visual_channel_color(
  field = "base",
  type = "nominal",
  domain = c("A", "T", "G", "C"),
  legend = TRUE
)

view2_track3_text <- visual_channel_text(
  field = "base", type = "nominal"
)

view2_track3_style <- default_track_styles(
  inlineLegend = TRUE
)

view2_track3 <- add_single_track(
  title = "NC_045512.2 Sequence",
  alignment = "overlay",
  data = view2_track3_data,
  tracks = add_multi_tracks(
    view2_track3a, view2_track3b
  ),
  x = view2_track3_x,
  color = view2_track3_color,
  text = view2_track3_text,
  style = view2_track3_style,
  width = 800, height = 40
)

view2 <- compose_view(
  multi = TRUE,
  centerRadius = 0,
  xDomain = list(interval = c(1, 29903)),
  linkingId = "detail",
  alignment = "stack",
  tracks = add_multi_tracks(
    view2_track3
  )
)
```

```
  combined_view <- arrange_views(
    title = "SARS-CoV-2",
    subtitle = "Data Source: WashU Virus Genome Browser, NCBI, GISAID",
    assembly = list(list("NC_045512.2", 29903)),
    layout = "linear",
    spacing = 50,
    views = list(view2),
    listify = FALSE
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "sars_cov2",
        combined_view
      )
    })
  }

  shinyApp(ui, server)

}
```

`visual_channel_opacity` *opacity visual channel*

Description

opacity visual channel

Usage

```
visual_channel_opacity(
  field = NULL,
  type = NULL,
  range = NULL,
  domain = NULL,
  ...
)
```

Arguments

field	A character. Name of the data field.
type	A character. Must be "genomic". Specify the data type.
range	A vector of characters or numbers. Values of the visual channel.
domain	A vector of characters or numbers. Values of the data.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#opacity

Value

List object.

---

visual_channel_row *row visual channel*

Description

row visual channel

Usage

```
visual_channel_row(
  field = NULL,
  type = NULL,
  padding = NULL,
  legend = NULL,
  grid = NULL,
  clip = NULL,
  axis = NULL,
  aggregate = NULL,
  ...
)
```

Arguments

field	A character. Name of the data field.
type	A character. Must be "genomic". Specify the data type.
padding	A number. Determines the size of inner white spaces on the top and bottom of individual rows. Default: 0.
legend	A Boolean. Whether to display legend. Default: FALSE.
grid	A Boolean. Whether to display grid. Default: FALSE.
clip	A Boolean. Clip row when the actual y value exceeds the max value of the y scale. Used only for bar marks at the moment. Default: TRUE.
axis	A character. One of "none", "top", "bottom", "left", "right". Specify where should the axis be put.
aggregate	A character. One of "max", "min", "mean", "bin", "count". Specify how to aggregate data. Default: undefined.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#row

Value

List object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  # Circular track 1 ----
  circular_track1_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec",
    type = "multivec",
    row = "sample",
    column = "position",
    value = "peak",
    categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
    binSize = 4
  )

  circular_track1_x <- visual_channel_x(field = "start", type = "genomic")
  circular_track1_xe <- visual_channel_x(field = "end", type = "genomic")

  circular_track1_y <- visual_channel_y(field = "peak", type = "quantitative")

  circular_track1_row <- visual_channel_row(
    field = "sample", type = "nominal"
  )

  circular_track1_color <- visual_channel_color(
    field = "sample", type = "nominal"
  )

  circular_track1_tracks <- add_multi_tracks(
    add_single_track(
      mark = "bar"
    ),
    add_single_track(
      mark = "brush",
      x = visual_channel_x(linkingId = "detail-1"),
      color = "blue"
    ),
    add_single_track(
      mark = "brush",
      x = visual_channel_x(linkingId = "detail-2"),
      color = "red"
    )
  )

  circular_track1_styles <- default_track_styles(
    outlineWidth = 0
  )
```

```
  circular_track1 <- add_single_track(
    id = "circular_track1", alignment = "overlay", data = circular_track1_data,
    x = circular_track1_x, xe = circular_track1_xe,
    y = circular_track1_y, row = circular_track1_row,
    color = circular_track1_color,
    stroke =  "black", strokeWidth = 0.3,
    tracks = circular_track1_tracks,
    style = circular_track1_styles,
    width = 500, height = 100
  )

  # Compose Circular track ----
  circular_composed_view <- compose_view(
    multi = TRUE,
    tracks = add_multi_tracks(
      circular_track1
    ),
    static = TRUE, layout = "circular", alignment = "stack"
  )

  # Arrange final view
  circular_linear_view <- arrange_views(
    arrangement = "horizontal",
    views = list(circular_composed_view)
  )




  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "circular_component",
        circular_linear_view, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)

}
```

`visual_channel_size` *size visual channel*

Description

size visual channel

Usage

```
visual_channel_size(
  field = NULL,
  type = NULL,
  range = NULL,
  domain = NULL,
  ...
)
```

Arguments

field	A character. Name of the data field.
type	A character. Must be "genomic". Specify the data type.
range	A vector of characters or numbers. Values of the visual channel. Range to be specified like range = c(min_size, max_size)
domain	A vector of characters or numbers. Values of the data.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#size

Value

List object.

---

visual_channel_stroke *stroke visual channel*

Description

stroke visual channel

Usage

```
visual_channel_stroke(
  field = NULL,
  title = NULL,
  type = NULL,
  scaleOffset = NULL,
  legend = NULL,
  grid = NULL,
  axis = NULL,
  aggregate = NULL,
  ...
)
```

Arguments

field	A character. Name of the data field.
title	A character. Title of the legend. Default: undefined.
type	A character. Must be "genomic". Specify the data type.
scaleOffset	A number vector of the form c(1, 2). Whether to use offset of the domain proportionally. This is bound to brushes on the color legend. Default: c(0, 1).
legend	A Boolean. Whether to display legend. Default: FALSE.
grid	A Boolean. Whether to display grid. Default: FALSE.
axis	A character. One of "none", "top", "bottom", "left", "right". Specify where should the axis be put.
aggregate	A character. One of "max", "min", "mean", "bin", "count". Specify how to aggregate data. Default: undefined.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#stroke

Value

List object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  track5_styles <- default_track_styles(
    legendTitle = "SV Class"
  )
  track5_data <- track_data(
   url = "https://s3.amazonaws.com/gosling-lang.org/data/cancer/rearrangement.PD35930a.csv",
    type = "csv",
    genomicFieldsToConvert = json_list(
      json_list(
        chromosomeField = "chr1",
        genomicFields = c("start1", "end1")
      ),
      json_list(
        chromosomeField = "chr2",
        genomicFields = c("start2", "end2")
      )
    )
  )
  track5_tracks <- add_multi_tracks(
    add_single_track(
      mark = "rect"
    ),
    add_single_track(
      mark = "withinLink", x = visual_channel_x(linkingId = "mid-scale"),
      strokeWidth = 0
    )
```

```
)
track5_color <- visual_channel_color(
  field = "svclass",
  type = "nominal",
  legend = TRUE,
  domain = json_list(
    "tandem-duplication", "translocation", "deletion", "inversion"
  ),
  range = json_list(
    "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
  )
)
track5_stroke <- visual_channel_stroke(
  field = "svclass",
  type = "nominal",
  domain = json_list(
    "tandem-duplication", "translocation", "deletion", "inversion"
  ),
  range = json_list(
    "#569C4D", "#4C75A2", "#DA5456", "#EA8A2A"
  )
)
track5_x <- visual_channel_x(field = "start1", type = "genomic")
track5_xe <- visual_channel_x(field = "end2", type = "genomic")
track5 <- add_single_track(
  id = "track5", title = "Structural Variant",
  data = track5_data, mark = "withinLink",
  x = track5_x, xe = track5_xe,
  color = track5_color, width = 500, height = 80, stroke = track5_stroke,
  strokeWidth = 1, opacity = 0.6, style = track5_styles
)

composed_track <- compose_view(
  multi = TRUE,
  tracks = add_multi_tracks(
    track5
  ),
  xOffset = 190, layout = "circular", spacing = 1
)

composed_views <- arrange_views(
  views = composed_track,
  arrangement = "vertical"
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot"))
  )
)


server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
      component_id = "component_2",
```

```
        composed_views, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)

}
```

`visual_channel_stroke_width` *stroke width visual channel*

Description

stroke width visual channel

Usage

```
visual_channel_stroke_width(
  field = NULL,
  type = NULL,
  range = NULL,
  domain = NULL,
  ...
)
```

Arguments

`field`	A character. Name of the data field.
`type`	A character. Must be "genomic". Specify the data type.
`range`	A vector of characters or numbers. Values of the visual channel.
`domain`	A vector of characters or numbers. Values of the data.
`...`	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#strokewidth

Value

List object.

`visual_channel_text` *text visual channel*

Description

text visual channel

Usage

```
visual_channel_text(
  field = NULL,
  type = NULL,
  range = NULL,
  domain = NULL,
  ...
)
```

Arguments

`field`	A character. Name of the data field.
`type`	A character. Must be "genomic". Specify the data type.
`range`	A vector of characters or numbers. Values of the visual channel.
`domain`	A vector of characters or numbers. Values of the data.
`...`	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#text

Value

List object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  # View 2 Track 3----
  view2_track3_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=NC_045512_2-multivec",
    type = "multivec",
    row = "base",
    column = "position",
    value = "count",
    categories = c("A", "T", "G", "C"),
    start = "start",
    end = "end"
  )

  view2_track3a <- add_single_track(
```

```
  mark = "bar",
  y = visual_channel_y(
    field = "count", type = "quantitative", axis = "none"
  )
)

view2_track3b <- add_single_track(
  dataTransform = track_data_transform(
    type = "filter",
    field = "count",
    oneOf = list(0),
    not = TRUE
  ),
  mark = "text",
  x = visual_channel_x(
    field = "start", type = "genomic"
  ),
  xe = visual_channel_x(
    field = "end", type = "genomic"
  ),
  size = 24,
  color = "white",
  visibility = list(list(
    operation = "less-than",
    measure = "width",
    threshold = "|xe-x|",
    transitionPadding = 30,
    target = "mark"
  ),
  list(
    operation = "LT",
    measure = "zoomLevel",
    threshold = 40,
    target = "track"
  ))
)

view2_track3_x <- visual_channel_x(
  field = "position", type = "genomic"
)

view2_track3_color <- visual_channel_color(
  field = "base",
  type = "nominal",
  domain = c("A", "T", "G", "C"),
  legend = TRUE
)

view2_track3_text <- visual_channel_text(
  field = "base", type = "nominal"
)

view2_track3_style <- default_track_styles(
  inlineLegend = TRUE
)

view2_track3 <- add_single_track(
```

```
    title = "NC_045512.2 Sequence",
    alignment = "overlay",
    data = view2_track3_data,
    tracks = add_multi_tracks(
      view2_track3a, view2_track3b
    ),
    x = view2_track3_x,
    color = view2_track3_color,
    text = view2_track3_text,
    style = view2_track3_style,
    width = 800, height = 40
  )

  view2 <- compose_view(
    multi = TRUE,
    centerRadius = 0,
    xDomain = list(interval = c(1, 29903)),
    linkingId = "detail",
    alignment = "stack",
    tracks = add_multi_tracks(
      view2_track3
    )
  )

  combined_view <- arrange_views(
    title = "SARS-CoV-2",
    subtitle = "Data Source: WashU Virus Genome Browser, NCBI, GISAID",
    assembly = list(list("NC_045512.2", 29903)),
    layout = "linear",
    spacing = 50,
    views = list(view2),
    listify = FALSE
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "sars_cov2",
        combined_view
      )
    })
  }

  shinyApp(ui, server)

}
```

`visual_channel_tooltip` *tooltip visual channel*

Description

tooltip visual channel

Usage

```
visual_channel_tooltip(field = NULL, type = NULL, alt = NULL, ...)
```

Arguments

`field`	A character. Name of the data field.
`type`	A character. Must be "genomic". Specify the data type.
`alt`	A character.
`...`	Any other parameters to pass to gosling.js.

Details

For more info visit https://gosling.js.org/ and check for tooltip implementation

Value

List object. list object with tooltip list object

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
      binSize = 4,
    ),
    mark = "rect",
    x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
    xe = visual_channel_x(field = "end", type = "genomic"),
    row = visual_channel_row(
      field = "sample",
```

```
    type = "nominal",
    legend = TRUE
  ),
  color = visual_channel_color(
    field = "peak",
    type = "quantitative",
    legend = TRUE
  ),
  tooltip = visual_channel_tooltips(
    visual_channel_tooltip(field = "start", type = "genomic",
                           alt = "Start Position"),
    visual_channel_tooltip(field = "end", type = "genomic",
                           alt = "End Position"),
    visual_channel_tooltip(
      field = "peak",
      type = "quantitative",
      alt = "Value",
      format = "0.2"
    )
  ),
  width = 600,
  height = 130
)

single_composed_track <- compose_view(
  tracks = single_track
)

single_composed_views <- arrange_views(
  title = "Single Track",
  subtitle = "This is the simplest single track visualization with a linear layout",
  layout = "circular", #"linear"
  views = single_composed_track,
  xDomain = list(
    chromosome = "chr1",
    interval = c(1, 3000500)
  )
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot")),
    column(
      1, br(), actionButton(
        "download_pdf",
        "PDF",
        icon = icon("cloud-arrow-down")
      )
    )
  )
)


server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
```

```
      component_id = "component_1",
      single_composed_views,
      clean_braces = TRUE
    )
  })

  observeEvent(input$download_pdf, {
    export_pdf(component_id = "component_1")
  })
}

shinyApp(ui, server)

}
```

`visual_channel_tooltips` *Combine tooltips into a list*

Description

Combine tooltips into a list

Usage

```
visual_channel_tooltips(...)
```

Arguments

`...`	Any other parameters to pass to gosling.js.

Value

List object. json list with tooltips combined into a single spec

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
```

```
    binSize = 4,
  ),
  mark = "rect",
  x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
  xe = visual_channel_x(field = "end", type = "genomic"),
  row = visual_channel_row(
    field = "sample",
    type = "nominal",
    legend = TRUE
  ),
  color = visual_channel_color(
    field = "peak",
    type = "quantitative",
    legend = TRUE
  ),
  tooltip = visual_channel_tooltips(
    visual_channel_tooltip(field = "start", type = "genomic",
                           alt = "Start Position"),
    visual_channel_tooltip(field = "end", type = "genomic",
                           alt = "End Position"),
    visual_channel_tooltip(
      field = "peak",
      type = "quantitative",
      alt = "Value",
      format = "0.2"
    )
  ),
  width = 600,
  height = 130
)

single_composed_track <- compose_view(
  tracks = single_track
)

single_composed_views <- arrange_views(
  title = "Single Track",
  subtitle = "This is the simplest single track visualization with a linear layout",
  layout = "circular", #"linear"
  views = single_composed_track,
  xDomain = list(
    chromosome = "chr1",
    interval = c(1, 3000500)
  )
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot")),
    column(
      1, br(), actionButton(
        "download_pdf",
        "PDF",
        icon = icon("cloud-arrow-down")
      )
    )
```

```
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_1",
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$download_pdf, {
      export_pdf(component_id = "component_1")
    })
  }

  shinyApp(ui, server)

}
```

`visual_channel_x` *x and xe axis visual channel*

Description

x and xe axis visual channel

Usage

```
visual_channel_x(
  field = NULL,
  type = NULL,
  legend = NULL,
  grid = NULL,
  axis = NULL,
  aggregate = NULL,
  ...
)
```

Arguments

`field`	A character. Name of the data field.
`type`	A character. Must be "genomic". Specify the data type.
`legend`	A Boolean. Whether to display legend. Default: FALSE.
`grid`	A Boolean. Whether to display grid. Default: FALSE.
`axis`	A character. One of "none", "top", "bottom", "left", "right". Specify where should the axis be put.
`aggregate`	A character. One of "max", "min", "mean", "bin", "count". Specify how to aggregate data. Default: undefined.
`...`	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#x–xe

Value

List object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
      binSize = 4,
    ),
    mark = "rect",
    x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
    xe = visual_channel_x(field = "end", type = "genomic"),
    row = visual_channel_row(
      field = "sample",
      type = "nominal",
      legend = TRUE
    ),
    color = visual_channel_color(
      field = "peak",
      type = "quantitative",
      legend = TRUE
    ),
    tooltip = visual_channel_tooltips(
      visual_channel_tooltip(field = "start", type = "genomic",
                             alt = "Start Position"),
      visual_channel_tooltip(field = "end", type = "genomic",
                             alt = "End Position"),
      visual_channel_tooltip(
        field = "peak",
        type = "quantitative",
        alt = "Value",
        format = "0.2"
      )
    ),
    width = 600,
    height = 130
  )
```

```
  single_composed_track <- compose_view(
    tracks = single_track
  )

  single_composed_views <- arrange_views(
    title = "Single Track",
    subtitle = "This is the simplest single track visualization with a linear layout",
    layout = "circular", #"linear"
    views = single_composed_track,
    xDomain = list(
      chromosome = "chr1",
      interval = c(1, 3000500)
    )
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot")),
      column(
        1, br(), actionButton(
          "download_pdf",
          "PDF",
          icon = icon("cloud-arrow-down")
        )
      )
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_1",
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$download_pdf, {
      export_pdf(component_id = "component_1")
    })
  }

  shinyApp(ui, server)

}
```

`visual_channel_y` *y and ye axis visual channel*

Description

y and ye axis visual channel

Usage

```
visual_channel_y(
  field = NULL,
  zeroBaseline = NULL,
  type = NULL,
  legend = NULL,
  grid = NULL,
  flip = NULL,
  baseline = NULL,
  axis = NULL,
  aggregate = NULL,
  ...
)
```

Arguments

field	A character. Name of the data field.
zeroBaseline	A Boolean. Specify whether to use zero baseline. Default: TRUE.
type	A character. Must be "genomic". Specify the data type.
legend	A Boolean. Whether to display legend. Default: FALSE.
grid	A Boolean. Whether to display grid. Default: FALSE.
flip	A Boolean. Whether to flip the y-axis. This is done by inverting the range property. Default: FALSE.
baseline	A character or number. Custom baseline of the y-axis. Default: 0.
axis	A character. One of "none", "top", "bottom", "left", "right". Specify where should the axis be put.
aggregate	A character. One of "max", "min", "mean", "bin", "count". Specify how to aggregate data. Default: undefined.
...	Any other parameters to pass to gosling.js.

Details

For more info visit http://gosling-lang.org/docs/visual-channel#y–ye

Value

List object.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  # Circular track 1 ----
  circular_track1_data <- track_data(
    url = "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec",
    type = "multivec",
    row = "sample",
    column = "position",
    value = "peak",
```

```
  categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
  binSize = 4
)

circular_track1_x <- visual_channel_x(field = "start", type = "genomic")
circular_track1_xe <- visual_channel_x(field = "end", type = "genomic")

circular_track1_y <- visual_channel_y(field = "peak", type = "quantitative")

circular_track1_row <- visual_channel_row(
  field = "sample", type = "nominal"
)

circular_track1_color <- visual_channel_color(
  field = "sample", type = "nominal"
)

circular_track1_tracks <- add_multi_tracks(
  add_single_track(
    mark = "bar"
  ),
  add_single_track(
    mark = "brush",
    x = visual_channel_x(linkingId = "detail-1"),
    color = "blue"
  ),
  add_single_track(
    mark = "brush",
    x = visual_channel_x(linkingId = "detail-2"),
    color = "red"
  )
)

circular_track1_styles <- default_track_styles(
  outlineWidth = 0
)

circular_track1 <- add_single_track(
  id = "circular_track1", alignment = "overlay", data = circular_track1_data,
  x = circular_track1_x, xe = circular_track1_xe,
  y = circular_track1_y, row = circular_track1_row,
  color = circular_track1_color,
  stroke =  "black", strokeWidth = 0.3,
  tracks = circular_track1_tracks,
  style = circular_track1_styles,
  width = 500, height = 100
)

# Compose Circular track ----
circular_composed_view <- compose_view(
  multi = TRUE,
  tracks = add_multi_tracks(
    circular_track1
  ),
  static = TRUE, layout = "circular", alignment = "stack"
)
```

```
  # Arrange final view
  circular_linear_view <- arrange_views(
    arrangement = "horizontal",
    views = list(circular_composed_view)
  )




  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot"))
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "circular_component",
        circular_linear_view, clean_braces = FALSE
      )
    })
  }

  shinyApp(ui, server)

}
```

---

zoom_to *Zoom to*

Description

Zooms to a specific genomic position with the animated transition.

Usage

```
zoom_to(
  component_id,
  view_id,
  position,
  padding = 0,
  duration = 1000,
  session = getDefaultReactiveDomain()
)
```

Arguments

component_id	A character. The id of the component_id prop passed to the GoslingComponent function.

view_id	A character. The ID of a view that you want to control. This ID is consistent to what you specify as track.id in your spec.
position	A character. The genomic position that your view should be navigated to. You can either specify chromosome (e.g., chr1) or a chromosome and range pair (e.g., chr1:1-10000).
padding	A numeric. This determines the padding around the specified position. The unit of this number is a base pair (Default: 0).
duration	A numeric. A duration of the animated transition in ms (Default: 1000).
session	A shiny session object.

Value

None.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  chromosome_options <- c(
    "Chr 1" = "chr1",
    "Chr 2" = "chr2",
    "Chr X" = "chrX",
    "Chr Y" = "chrY"
  )

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
      binSize = 4,
    ),
    mark = "rect",
    x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
    xe = visual_channel_x(field = "end", type = "genomic"),
    row = visual_channel_row(
      field = "sample",
      type = "nominal",
      legend = TRUE
    ),
    color = visual_channel_color(
      field = "peak",
      type = "quantitative",
      legend = TRUE
    ),
    tooltip = visual_channel_tooltips(
```

```
    visual_channel_tooltip(field = "start", type = "genomic",
                           alt = "Start Position"),
    visual_channel_tooltip(field = "end", type = "genomic",
                           alt = "End Position"),
    visual_channel_tooltip(
      field = "peak",
      type = "quantitative",
      alt = "Value",
      format = "0.2"
    )
  ),
  width = 600,
  height = 130
)

single_composed_track <- compose_view(
  tracks = single_track
)

single_composed_views <- arrange_views(
  title = "Single Track",
  subtitle = "This is the simplest single track visualization with a linear layout",
  layout = "circular", #"linear"
  views = single_composed_track,
  xDomain = list(
    chromosome = "chr1",
    interval = c(1, 3000500)
  )
)

ui <- fluidPage(
  use_gosling(),
  fluidRow(
    column(6, goslingOutput("gosling_plot")),
    column(
      1, br(), actionButton(
        "zoom_out",
        "Zoom To"
      )
    ),
    column(
      2,
      selectInput(
        "chromosomes",
        "Chromosome",
        selected = "chr1",
        choices = chromosome_options
      )
    )
  )
)


server <- function(input, output, session) {
  output$gosling_plot <- renderGosling({
    gosling(
      component_id = "component_1",
```

```
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$zoom_out, {
      zoom_to(
        component_id = "component_1",
        view_id = "track1",
        position = input$chromosomes
      )
    })
  }

  shinyApp(ui, server)

}
```

`zoom_to_extent` *Zoom to extent*

Description

Zooms out to see the entire view_id passed to this function.

Usage

```
zoom_to_extent(
  component_id,
  view_id,
  duration = 1000,
  session = getDefaultReactiveDomain()
)
```

Arguments

`component_id`	A character. The id of the component_id prop passed to the GoslingComponent function.
`view_id`	A character. The ID of a view that you want to control. This ID is consistent to what you specify as track.id in your spec.
`duration`	A numeric. A duration of the animated transition in ms (Default: 1000).
`session`	A shiny session object.

Value

None.

Examples

```
if(interactive()) {
  library(shiny)
  library(shiny.gosling)

  cistrome_data <-
    "https://server.gosling-lang.org/api/v1/tileset_info/?d=cistrome-multivec"

  single_track <- add_single_track(
    id = "track1",
    data = track_data(
      url = cistrome_data,
      type = "multivec",
      row = "sample",
      column = "position",
      value = "peak",
      categories = c("sample 1", "sample 2", "sample 3", "sample 4"),
      binSize = 4,
    ),
    mark = "rect",
    x = visual_channel_x(field = "start", type = "genomic", axis = "top"),
    xe = visual_channel_x(field = "end", type = "genomic"),
    row = visual_channel_row(
      field = "sample",
      type = "nominal",
      legend = TRUE
    ),
    color = visual_channel_color(
      field = "peak",
      type = "quantitative",
      legend = TRUE
    ),
    tooltip = visual_channel_tooltips(
      visual_channel_tooltip(field = "start", type = "genomic",
                             alt = "Start Position"),
      visual_channel_tooltip(field = "end", type = "genomic",
                             alt = "End Position"),
      visual_channel_tooltip(
        field = "peak",
        type = "quantitative",
        alt = "Value",
        format = "0.2"
      )
    ),
    width = 600,
    height = 130
  )

  single_composed_track <- compose_view(
    tracks = single_track
  )

  single_composed_views <- arrange_views(
    title = "Single Track",
    subtitle = "This is the simplest single track visualization with a linear layout",
    layout = "circular", #"linear"
```

```
    views = single_composed_track,
    xDomain = list(
      chromosome = "chr1",
      interval = c(1, 3000500)
    )
  )

  ui <- fluidPage(
    use_gosling(),
    fluidRow(
      column(6, goslingOutput("gosling_plot")),
      column(
        1, br(), actionButton(
          "zoom_out",
          "Zoom Out"
        )
      )
    )
  )


  server <- function(input, output, session) {
    output$gosling_plot <- renderGosling({
      gosling(
        component_id = "component_1",
        single_composed_views,
        clean_braces = TRUE
      )
    })

    observeEvent(input$zoom_out, {
      zoom_to_extent(
        component_id = "component_1",
        view_id = "track1"
      )
    })
  }

  shinyApp(ui, server)

}
```

Index