Package 'linkeR'

May 8, 2026

Title Link Interactive Plots and Tables in 'shiny' Applications

Version 0.1.3

Description Build powerful, linked-view dashboards in 'shiny' applications. With a declarative, one-line setup, you can create bidirectional links between interactive components. When a user interacts with one element (e.g., clicking a map marker), all linked components (such as 'DT' tables or other charts) instantly update. Supports 'leaflet' maps, 'DT' tables, 'plotly' charts, and spatial data via 'sf' objects out-of-the-box, with an extensible API for custom components.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports shiny (>= 1.5.0), magrittr (>= 2.0.0), later (>= 1.0.0)

Suggests leaflet, DT, sf, testthat (>= 3.0.0), knitr, plotly, bslib, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://epiforesite.github.io/linkeR/, https://github.com/EpiForeSITE/linkeR/

BugReports https://github.com/EpiForeSITE/linkeR/issues/

NeedsCompilation no

Author Jake Wagoner [aut, cre] (ORCID: <https://orcid.org/0009-0000-5053-2281>), Centers for Disease Control and Prevention's Center for Forecasting and Outbreak Analytics [fnd] (Cooperative agreement CDC-RFA-FT-23-0069)

Maintainer Jake Wagoner <jakew@sci.utah.edu>

Repository CRAN

Date/Publication 2025-10-07 17:10:02 UTC

Contents

`apply_default_leaflet_behavior` *Apply Default Leaflet Behavior for Selection Events*

Description

`apply_default_leaflet_behavior` is a helper function that provides consistent default behavior for leaflet maps when handling selection events. It manages popup display and map navigation based on the selection state.

Usage

```
apply_default_leaflet_behavior(map_proxy, selected_data, component_info)
```

Arguments

`map_proxy`	A leaflet map proxy object used to update the map
`selected_data`	A data frame or list containing the selected row/item data. If NULL, indicates deselection occurred.
`component_info`	A list containing component configuration information:

shared_id_column Character. Name of the column containing unique identifiers

config List containing:

lng_col Character. Name of the longitude column

lat_col Character. Name of the latitude column

highlight_zoom Numeric. Zoom level to use when highlighting selection

Details

When `selected_data` is provided:

- Creates a popup with "Selected" header and ID information
- Sets map view to the selected location coordinates
- Applies the configured highlight zoom level

When `selected_data` is NULL (deselection):

- Removes all existing popups from the map

Value

Returns the modified map proxy object with updated view and popups

---

`create_link_registry` *Create a Link Registry for 'shiny' Component Coordination*

Description

`create_link_registry` creates a registry system that manages linked interactions between multiple 'shiny' components, allowing them to share selection state and coordinate their behavior.

Usage

```
create_link_registry(session, on_selection_change = NULL)
```

Arguments

`session`	A 'shiny' session object, required for server-side reactivity
`on_selection_change`	Optional callback function that gets called when selection changes. Should accept parameters: selected_id, selected_data, source_component_id, and session

Details

The registry maintains a shared state across all registered components, automatically setting up observers to synchronize selections. When a selection changes in one component, all other registered components are updated to reflect the same selection.

Components are automatically cleaned up when re-registered to prevent memory leaks from orphaned observers.

Value

A link_registry object with the following methods:

register_component(session, component_id, type, data_reactive, shared_id_column, config) Register a new component with the registry. Parameters:

- session: 'shiny' session object for namespacing. Can be global session in non-modular apps.
- component_id: Unique string identifier for the component
- type: Component type (e.g., "table", "plot")
- data_reactive: Reactive expression returning the component's data
- shared_id_column: Name of the column used for linking selections
- config: Optional list of component-specific configuration

clear_all() Remove all registered components and reset shared state

set_selection(selected_id, source_component_id) Programmatically update the selection state

get_selection() Get current selection as list with selected_id and source

get_on_selection_change() Return the on_selection_change callback function

get_components() Get registry components info (for debugging)

get_shared_state() Get current shared state (for debugging)

See Also

`setup_component_observers()` for component observer setup

Examples

```
# Create a mock session for the example
session <- shiny::MockShinySession$new()

# Create registry with optional callback
registry <- create_link_registry(
  session = session,
  on_selection_change = function(id, data, source, session) {
    message("Selection changed to ID: ", id, " from: ", source)
  }
)

# In a real app, you would register components like this:
# my_data <- reactive({ data.frame(id = 1:3, name = c("A", "B", "C")) })
# registry$register_component("table1", "table", my_data, "id")
# registry$register_component("plot1", "plot", my_data, "id")
```

linkeR-imports *Package imports*

Description

Package imports

link_plots *Simple Plot Linking Function for Non-Modular 'shiny' Apps*

Description

`link_plots` provides a simple, one-line interface to link interactive components in a **single-file or non-modular 'shiny' application**. It automatically detects component types and sets up bidirectional linking.

Usage

```
link_plots(
  session,
  ...,
  shared_id_column,
  leaflet_lng_col = "longitude",
  leaflet_lat_col = "latitude",
  leaflet_click_handler = NULL,
  dt_click_handler = NULL,
  plotly_click_handler = NULL,
  on_selection_change = NULL
)
```

Arguments

`session`	The 'shiny' session object
`...`	Named arguments where names are component output IDs and values are reactive data frames. Each data frame must contain the shared_id_column. For leaflet maps: can be sf objects (coordinates auto-extracted) or regular data frames with longitude/latitude columns.
`shared_id_column`	Character string naming the column that contains unique identifiers present in all linked components.
`leaflet_lng_col`	Character string naming the longitude column for leaflet maps. Defaults to "longitude". For sf objects, this will be the name of the created column.

`leaflet_lat_col`
: Character string naming the latitude column for leaflet maps. Defaults to "latitude". For sf objects, this will be the name of the created column.

`leaflet_click_handler`
: Optional function that handles leaflet marker clicks. This will be used for both direct clicks and when other components select this marker. Function should accept (map_proxy, selected_data, session).

`dt_click_handler`
: Optional function that handles DT row selections. This will be used for both direct clicks and when other components select this row. Function should accept (dt_proxy, selected_data, session).

`plotly_click_handler`
: Optional function that handles plotly point clicks. This will be used for both direct clicks and when other components select this point. Function should accept (plot_proxy, selected_data, session).

`on_selection_change`
: Optional callback function that gets called when selection changes. Function should accept parameters: (selected_id, selected_data, source_component_id, session)

Details

This function is the fastest way to get started with `linkeR` and is ideal for straightforward dashboards.

For more complex applications that use **'shiny' Modules**, you should use the more robust pattern of creating a central registry with `create_link_registry()` and passing it to your modules, where you will call `register_leaflet()` or `register_dt()` directly. This preserves module encapsulation and leads to more maintainable code. See the `modularized_example` for a complete example of this pattern.

Value

Invisibly returns the created registry object

Examples

```
# This example is for a single-file app.
# For modular apps, please see the "Using linkeR with Modules" vignette.
if (interactive()) {
  library(shiny)
  library(leaflet)
  library(DT)

  # Sample data
  sample_data <- data.frame(
    id = 1:10,
    name = paste("Location", 1:10),
    latitude = runif(10, 40.7, 40.8),
    longitude = runif(10, -111.95, -111.85),
```

```
  value = round(runif(10, 100, 1000))
)

ui <- fluidPage(
  titlePanel("linkeR Example"),
  fluidRow(
    column(6, leafletOutput("my_map")),
    column(6, DTOutput("my_table"))
  )
)

server <- function(input, output, session) {
  my_data <- reactive({
    sample_data
  })

  output$my_map <- renderLeaflet({
    leaflet(my_data()) %>%
      addTiles() %>%
      addMarkers(
        lng = ~longitude,
        lat = ~latitude,
        layerId = ~id,
        popup = ~name
      )
  })

  output$my_table <- renderDT({
    datatable(my_data()[, c("name", "value")], selection = "single")
  })

  link_plots(
    session,
    my_map = my_data,
    my_table = my_data,
    shared_id_column = "id"
  )
}

shinyApp(ui, server)
}
```

`prepare_plotly_linking` *Prepare Plotly for Linking*

Description

Utility function to automatically add required parameters to a plotly object for reliable linking, regardless of plot structure (single/multiple traces).

Usage

```
prepare_plotly_linking(plotly_obj, id_column, source)
```

Arguments

plotly_obj	A plotly object created with plot_ly()
id_column	Character string: name of the ID column in the data
source	Character string: plotly source identifier

Value

Modified plotly object with linking parameters added

Examples

```
 library(plotly)

# Sample data
df <- data.frame(
  id = 1:5,
  value = c(10, 20, 15, 25, 30),
  group = c("A", "A", "B", "B", "C")
)

# Create a plotly scatter plot
p <- plot_ly(
  data = df,
  x = ~value,
  y = ~id,
  color = ~group
)

# Prepare for linking (adds customdata and source)
p <- prepare_plotly_linking(p, "id", "my_plot")

# Print the plot object (for demonstration)
print(p)
```

---

register_dt *Register a DT DataTable Component*

Description

register_dt registers a DT datatable for linking with other components.

Usage

```
register_dt(
  session,
  registry,
  dt_output_id,
  data_reactive,
  shared_id_column,
  click_handler = NULL
)
```

Arguments

session	'shiny' session object. The session from the module where the DT is used. This could be global session in non-modular apps.
registry	A link registry created by `create_link_registry()`
dt_output_id	Character string: the outputId of your DT::DTOutput
data_reactive	Reactive expression returning the data frame for the table
shared_id_column	Character string: name of the ID column
click_handler	Optional function: custom click handler for row selection, must have args (map_proxy, selected_data, session), overrides all default behavior

Value

NULL (invisible). This function is called for its side effects of registering the component.

Examples

```
# Create a mock session for the example
session <- shiny::MockShinySession$new()

# Create a registry
registry <- create_link_registry(session)

# Sample reactive data
my_data <- shiny::reactive({
  data.frame(
    id = 1:5,
    name = c("A", "B", "C", "D", "E"),
    value = 11:15
  )
})

# Register a DT component
register_dt(session, registry, "my_table", my_data, "id")

# Verify registration
print(registry$get_components())
```

`register_leaflet` *Register a Leaflet Component*

Description

`register_leaflet` registers a Leaflet map for linking with other components.

Usage

```
register_leaflet(
  session,
  registry,
  leaflet_output_id,
  data_reactive,
  shared_id_column,
  lng_col = "longitude",
  lat_col = "latitude",
  highlight_zoom = 12,
  click_handler = NULL
)
```

Arguments

`session`	'shiny' session object. The session from the module where the DT is used. This could be global session in non-modular apps.
`registry`	A link registry created by `create_link_registry()`
`leaflet_output_id`	Character string: the outputId of your leafletOutput
`data_reactive`	Reactive expression returning the data frame for the map
`shared_id_column`	Character string: name of the ID column
`lng_col`	Character string: name of longitude column (default: "longitude")
`lat_col`	Character string: name of latitude column (default: "latitude")
`highlight_zoom`	Numeric: zoom level when highlighting (default: 12)
`click_handler`	Optional function: custom click handler for row selection, must have args (map_proxy, selected_data, session), overrides all default behavior

Value

No return value, called for side effects.

Examples

```
# Create a mock session for the example
session <- shiny::MockShinySession$new()

# Create a registry
registry <- create_link_registry(session)

# Sample reactive data
my_data <- shiny::reactive({
  data.frame(
    id = 1:5,
    name = c("A", "B", "C", "D", "E"),
    longitude = -111.9 + runif(5, -0.1, 0.1),
    latitude = 40.7 + runif(5, -0.1, 0.1)
  )
})

# Register a leaflet component
register_leaflet(session, registry, "my_map", my_data, "id")

# Verify registration
print(registry$get_components())
```

`register_plotly` *Register a Plotly Component*

Description

`register_plotly` registers a Plotly component for linking with other components. The default behavior uses plotly's built-in point selection highlighting, which is simple and works reliably across all plot types.

Usage

```
register_plotly(
  session,
  registry,
  plotly_output_id,
  data_reactive,
  shared_id_column,
  event_types = c("plotly_click"),
  source = NULL,
  click_handler = NULL
)
```

Arguments

session	Shiny session object
registry	A link registry created by create_link_registry()
plotly_output_id	Character string: the outputId of your plotlyOutput
data_reactive	Reactive expression returning the data frame for the plot
shared_id_column	Character string: name of the ID column
event_types	Character vector: plotly event types to listen for
source	Character string: plotly source identifier for event tracking
click_handler	Optional function: custom selection update handler. Function signature: function(plot_proxy, selected_data, session) where selected_data is the row from data_reactive() or NULL to clear selection.

Value

NULL (invisible). This function is called for its side effects.

Examples

```
# Create a mock session for the example
session <- shiny::MockShinySession$new()

# Create a registry
registry <- create_link_registry(session)

# Sample reactive data
my_data <- shiny::reactive({
  data.frame(
    id = 1:5,
    name = c("A", "B", "C", "D", "E"),
    value = 11:15
  )
})

# Register a plotly component
register_plotly(
  session,
  registry,
  plotly_output_id = "my_plot",
  data_reactive = my_data,
  shared_id_column = "id"
)

# Verify registration
print(registry$get_components())
```

setup_datatable_observers *Setup DataTable Observers*

Description

setup_datatable_observers Sets up reactive observers for a DataTable component to handle user interactions and state changes. This function establishes the necessary event handlers for selection changes and synchronizes the component with the shared application state.

Usage

```
setup_datatable_observers(
  component_id,
  session,
  components,
  shared_state,
  on_selection_change,
  registry = NULL
)
```

Arguments

component_id	Character string. Unique identifier for the DataTable component.
session	'shiny' session object. The current 'shiny' session for reactive context.
components	List. Collection of UI components in the application.
shared_state	Reactive values object. Shared state container for cross-component communication.
on_selection_change	Function. Callback function to execute when table selection changes.
registry	List or NULL. Optional registry for component management. Defaults to NULL.

Details

This function creates reactive observers that monitor DataTable interactions and update the shared state accordingly. It handles selection events and ensures proper synchronization between the DataTable component and other application components.

Value

NULL. This function is called for its side effects of setting up observers.

`setup_leaflet_observers` *Setup Leaflet Map Observers*

Description

`setup_leaflet_observers` creates two observers for handling Leaflet map interactions in a linked component system. The first observer handles direct marker clicks on the map, while the second observer responds to selection changes from other linked components.

Usage

```
setup_leaflet_observers(
  component_id,
  session,
  components,
  shared_state,
  on_selection_change,
  registry = NULL
)
```

Arguments

`component_id`	Character string. The unique identifier for the Leaflet component.
`session`	'shiny' session object for the current user session.
`components`	List containing component configuration data including data reactives and shared ID columns.
`shared_state`	Reactive values object containing selected_id and selection_source for coordinating selections across components.
`on_selection_change`	Function to call when selection changes (currently unused).
`registry`	Optional registry object with set_selection method for managing selections. If NULL, falls back to direct shared_state updates.

Details

The marker click observer:

- Extracts clicked marker ID from the click event
- Retrieves corresponding data row from the component's data
- Clears existing popups and applies click behavior (custom or default)
- Updates selection state through registry or direct shared_state modification

The selection response observer:

- Only responds to selections from other components (not self-selections)
- Updates the map visualization to reflect the new selection

Value

List containing two observer objects:

observer1	Handles marker click events on the map
observer2	Responds to selection changes from other components

---

update_dt_selection *Update DT Selection Based on Shared ID*

Description

update_dt_selection Updates the selection state of a DataTable (DT) component when a shared ID is selected or deselected from another linked component. This function handles both custom click handlers and default selection behavior.

Usage

```
update_dt_selection(component_id, selected_id, session, components)
```

Arguments

component_id	Character string. The ID of the DT component to update.
selected_id	The shared ID value to select. If NULL, deselects all rows.
session	'shiny' session object for the current user session.
components	List containing component configuration information, including data reactives, shared ID columns, and optional custom click handlers.

Details

The function performs the following steps:

- Validates that the DT package is available
- Retrieves current data from the component's reactive data source
- Validates that the shared ID column exists in the data
- Creates a DT proxy for programmatic table manipulation
- Finds the matching row based on the shared ID
- Executes either custom click handler or default selection behavior

Value

NULL (invisible). Function is called for side effects only.

Custom Click Handlers

If a custom click handler is provided in the component configuration (component_info$config$click_handler), it will be called with the DT proxy, selected data (or NULL for deselection), and session. Otherwise, default row selection/deselection is performed.

update_leaflet_selection *Update Leaflet Map Selection*

Description

update_leaflet_selection updates a Leaflet map component to reflect a new selection state. This function handles both selection and deselection events, applying either custom user-defined click handlers or default behaviors.

Usage

```
update_leaflet_selection(component_id, selected_id, session, components)
```

Arguments

component_id	Character string. The ID of the Leaflet map component to update.
selected_id	Character string or NULL. The ID of the selected item. If NULL, indicates deselection.
session	'shiny' session object. The current 'shiny' session.
components	List. A named list containing component information, where each element contains component configuration including data_reactive, shared_id_column, and config settings.

Details

The function performs the following operations:

- Validates that the leaflet package is available
- Checks that required columns (shared_id_column, lng_col, lat_col) exist in the data
- Clears existing popups on the map
- For selections: finds the selected data row and applies either custom click handler or default behavior
- For deselections: delegates to custom handler or performs default cleanup

Required columns in the component data:

- shared_id_column: Column containing unique identifiers for map features
- lng_col: Column containing longitude coordinates
- lat_col: Column containing latitude coordinates

Value

NULL (invisibly). The function is called for its side effects on the Leaflet map.

Note

If the leaflet package is not available, the function returns early without error. Missing required columns will generate a warning and cause early return.

Index