````
---
title: "Using MSDRCapstone"
author: "Your Name"
date: "`r Sys.Date()`"
output: rmarkdown::html_vignette
vignette: >
  %\VignetteIndexEntry{Using MSDRCapstone}
  %\VignetteEngine{knitr::rmarkdown}
  %\VignetteEncoding{UTF-8}
---

# Introduction

The `MSDRCapstone` package provides tools for data cleaning and visualization.

# Installation

To install the `MSDRCapstone` package, use:

```{r install, eval = FALSE}
install.packages("MSDRCapstone")

```


Getting Started

First, load the package:

```{r}

library(MSDRCapstone)

```

Functions Overview


---

## Function 1: `eq_clean_data()`

### Description
````

The `eq_clean_data` function is designed to clean and preprocess earthquake data. It converts date and location columns to appropriate formats, ensuring the data is ready for further analysis.

Usage

```r
eq_clean_data(data)
```

Arguments

- `data`: A data frame containing earthquake data with columns `time`, `latitude`, `longitude`, `deaths`, and `locationsource`.

Value

The function returns a cleaned data frame with the following transformations:
- The `time` column is converted to `DATE` in `Date` class.
- The `latitude` and `longitude` columns are converted to numeric columns `LATITUDE` and `LONGITUDE`.
- The `deaths` column is converted to an integer.
- A new column `LOCATION_NAME` is created by cleaning the `locationsource` column using `eq_location_clean`.
- The original columns `time`, `locationsource`, `latitude`, and `longitude` are optionally removed.

Example

```r
# Load necessary libraries
library(dplyr)

# Sample data frame
sample_data <- data.frame(
  time = c("2020-01-01", "2020-02-01"),
  latitude = c("34.05", "36.16"),
  longitude = c("-118.24", "-115.15"),
  deaths = c("10", "5"),
  locationsource = c("California", "Nevada")
)
```

```
# Clean the sample data using eq_clean_data
cleaned_data <- eq_clean_data(sample_data)
print(cleaned_data)
```

Details

The `eq_clean_data` function performs the following steps:

1. **Convert Time Column**: The `time` column is converted to a `Date` class by extracting the date part using the `as.Date` function.
2. **Convert Latitude and Longitude**: The `latitude` and `longitude` columns are converted to numeric types.
3. **Convert Deaths**: The `deaths` column is converted to an integer type.
4. **Clean Location Source**: The `locationsource` column is cleaned using a helper function `eq_location_clean` to create a new column `LOCATION_NAME`.
5. **Remove Original Columns**: The original columns (`time`, `locationsource`, `latitude`, `longitude`) are removed from the data frame using the `select` function.

This function is crucial for ensuring that the raw earthquake data is transformed into a consistent and analysis-ready format.

Function 2: `eq_location_clean()`

Description

The `eq_location_clean` function cleans and standardizes location names. It removes prefixes and formats the names to title case, making the location names more consistent and easier to analyze.

Usage

```r
eq_location_clean(location_name)
```

Arguments

- `location_name`: A character vector containing location names to be cleaned.

Value

The function returns a character vector with cleaned and standardized location names.

Example

```r
# Sample location names
sample_locations <- c("USA: California", "Canada: British Columbia")

# Clean the location names using eq_location_clean
cleaned_locations <- eq_location_clean(sample_locations)
print(cleaned_locations)
```

Details

The `eq_location_clean` function performs the following steps:

1. **Remove Prefixes**: Uses the `sub` function to remove any prefix before the colon (`:`) and the following space.
2. **Convert to Lowercase**: Converts the cleaned names to lowercase.
3. **Convert to Title Case**: Uses `tools::toTitleCase` to convert the names to title case, where each word starts with an uppercase letter and the rest of the letters are lowercase.

This function is useful for ensuring location names are in a standardized format, which can help in consistent data analysis and visualization.

Function 3: `geom_timeline()`

Description

The `geom_timeline` function creates a timeline plot of earthquake events. It uses ggplot2 to visualize the events with points whose size represents the magnitude (Richter scale value) and color represents the number of deaths.

Usage

```r
geom_timeline(data, x, y = NULL, color = "blue", size = 2, alpha = 0.5, ...)
```

Arguments

- `data`: A data frame containing the earthquake data.
- `x`: The name of the column in the data frame that represents the date.
- `y`: (Optional) The name of the column in the data frame that represents the y-axis variable. Defaults to `NULL`.
- `color`: The name of the column in the data frame that represents the color of the points. Defaults to `"blue"`.
- `size`: The name of the column in the data frame that represents the size of the points. Defaults to `2`.
- `alpha`: A numeric value representing the transparency level of the points. Defaults to `0.5`.
- `...`: Additional arguments to be passed to the ggplot2 functions.

Value

The function returns a ggplot object that represents the timeline of earthquake events.

Example

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Sample data frame
sample_data <- data.frame(
```

```
  DATE = as.Date(c("2020-01-01", "2020-02-01", "2020-03-01")),
  RICHTER = c(5.5, 6.0, 7.2),
  DEATHS = c(10, 50, 200)
)

# Create a timeline plot using geom_timeline
timeline_plot <- geom_timeline(data = sample_data, x = DATE,
size = RICHTER, color = DEATHS)
print(timeline_plot)
```

Details

The `geom_timeline` function creates a scatter plot with the following features:

1. **Date on x-axis**: The x-axis represents the date of the earthquake events.
2. **Optional y-axis**: The y-axis can be used to represent another variable if provided; otherwise, it is not displayed.
3. **Point Size**: The size of the points is scaled to represent the Richter scale value, providing a visual cue for the magnitude of the earthquakes.
4. **Point Color**: The color of the points is scaled to represent the number of deaths, with a gradient from blue (low) to red (high).
5. **Transparency**: The transparency of the points can be adjusted using the `alpha` parameter.
6. **Minimal Theme**: The plot uses a minimal theme to reduce visual clutter and emphasize the data.

This function is useful for visualizing earthquake events over time, highlighting the severity and impact of each event through size and color coding.

Function 4: `geom_timeline_label()`

Description

The `geom_timeline_label` function creates a timeline plot of

earthquake events, similar to `geom_timeline`, but with additional labels for the top `n_max` earthquakes based on their magnitude. This function highlights significant events by displaying their labels.

Usage

```r
geom_timeline_label(data, x, label, n_max = 10, ...)
```

Arguments

- `data`: A data frame containing the earthquake data.
- `x`: The name of the column in the data frame that represents the date.
- `label`: The name of the column in the data frame that contains the labels for the points.
- `n_max`: The maximum number of top earthquakes (by magnitude) to label. Defaults to `10`.
- `...`: Additional arguments to be passed to the ggplot2 functions.

Value

The function returns a ggplot object that represents the timeline of earthquake events with labels for the top `n_max` earthquakes.

Example

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Sample data frame
sample_data <- data.frame(
  DATE = as.Date(c("2020-01-01", "2020-02-01", "2020-03-01",
"2020-04-01", "2020-05-01")),
  RICHTER = c(5.5, 6.0, 7.2, 6.8, 5.9),
  DEATHS = c(10, 50, 200, 150, 30),
  LOCATION = c("Location A", "Location B", "Location C",
"Location D", "Location E"),
```

```
  mag = c(5.5, 6.0, 7.2, 6.8, 5.9)
)

# Create a timeline plot with labels using geom_timeline_label
timeline_label_plot <- geom_timeline_label(data = sample_data, x
= DATE, label = LOCATION, n_max = 3)
print(timeline_label_plot)
```

Details

The `geom_timeline_label` function creates a scatter plot with the following features:

1. **Date on x-axis**: The x-axis represents the date of the earthquake events.
2. **Point Size**: The size of the points is scaled to represent the Richter scale value, providing a visual cue for the magnitude of the earthquakes.
3. **Point Color**: The color of the points is scaled to represent the number of deaths, with a gradient from blue (low) to red (high).
4. **Transparency**: The transparency of the points is set to 0.6.
5. **Labels for Top Earthquakes**: Labels are added for the top `n_max` earthquakes based on their magnitude. The labels are displayed at a 45-degree angle above dashed lines that connect the points to the labels.
6. **Minimal Theme**: The plot uses a minimal theme to reduce visual clutter and emphasize the data.
7. **Y-axis Customization**: The y-axis title, text, and ticks are removed for a cleaner look.

This function is useful for visualizing earthquake events over time and highlighting the most significant events with labels, making it easier to identify and analyze key events.

Function 5: `eq_map()`

Description

The `eq_map` function creates an interactive map of earthquake events using the `leaflet` package. Each earthquake is represented by a circle whose size is proportional to its magnitude, and a popup displays additional information.

Usage

```r
eq_map(data, annot_col)
```

Arguments

- `data`: A data frame containing the earthquake data with columns `LATITUDE`, `LONGITUDE`, `mag`, and the specified annotation column.
- `annot_col`: The name of the column in the data frame that contains the annotation to be displayed in the popup.

Value

The function returns a `leaflet` map object that visualizes the earthquake events.

Example

```r
# Load necessary libraries
library(leaflet)

# Sample data frame
sample_data <- data.frame(
  LATITUDE = c(34.05, 36.16, 40.73, 37.77, 34.05),
  LONGITUDE = c(-118.24, -115.15, -73.93, -122.42, -118.24),
  mag = c(5.5, 6.0, 7.2, 6.8, 5.9),
  LOCATION = c("Location A", "Location B", "Location C",
"Location D", "Location E")
)

# Create a map using eq_map
earthquake_map <- eq_map(data = sample_data, annot_col =
"LOCATION")
earthquake_map
```

```

### Details

The `eq_map` function creates an interactive map with the following features:

1. **Check Required Columns**: Ensures that the input data frame contains the required columns: `LATITUDE`, `LONGITUDE`, `mag`, and the specified annotation column. If any of these columns are missing, the function will stop and return an error message.
2. **Create Map with Leaflet**:
   - **Base Tiles**: Adds base map tiles using `addTiles()`.
   - **Circles for Earthquakes**: Adds circles to the map at the locations of the earthquakes. The radius of each circle is scaled by the magnitude (`mag`) of the earthquake for better visibility.
   - **Popups**: Adds popups to the circles that display information from the specified annotation column (`annot_col`).
   - **Circle Styling**: Sets the color of the circles to blue and the fill opacity to 0.5.
3. **Set Initial View**: Centers the map based on the mean latitude and longitude of the earthquakes, with an initial zoom level of 2.

This function is useful for creating a quick and interactive visualization of earthquake events, allowing users to explore the data geographically and see detailed information through popups.

---

## Function 6: `eq_create_label()`

### Description

The `eq_create_label` function generates HTML-formatted labels for earthquake data. These labels can be used in visualizations, such as popups in interactive maps, to display detailed information about each earthquake event.

### Usage
```

```r
eq_create_label(data)
```

Arguments

- `data`: A data frame containing the earthquake data with columns `LOCATION_NAME`, `mag`, and `deaths`.

Value

The function returns a character vector of HTML-formatted labels for each row in the data frame.

Example

```r
# Load necessary libraries
library(dplyr)

# Sample data frame
sample_data <- data.frame(
  LOCATION_NAME = c("California", "Nevada", "Alaska"),
  mag = c(5.5, 6.0, 7.2),
  deaths = c(10, NA, 200)
)

# Create labels using eq_create_label
labels <- eq_create_label(sample_data)
print(labels)
```

Details

The `eq_create_label` function performs the following steps:

1. **Row-wise Operation**: Uses `rowwise()` to ensure that operations are performed on each row individually.
2. **Generate Labels**: Creates a new column `label` with HTML-formatted strings. Each label includes:
   - **Location**: Displayed as "Location: [cleaned location name]
" if `LOCATION_NAME` is not missing.
   - **Magnitude**: Displayed as "Magnitude: [magnitude]

" if `mag` is not missing.
   - **Total Deaths**: Displayed as "Total deaths: [deaths]" if `deaths` is not missing.
3. **Clean Location Name**: Cleans the `LOCATION_NAME` using the `eq_location_clean` function.
4. **Return Labels**: Extracts and returns the `label` column as a character vector.

This function is useful for creating detailed and informative labels for earthquake data, which can be used in visualizations to provide additional context and information.

Comprehensive Example

In this section, we will demonstrate how to use the `MSDRCapstone` package to clean earthquake data, visualize it using timelines and maps, and create detailed labels for interactive exploration.

Step 1: Data Preparation

First, we need to load and clean the earthquake data using the `eq_clean_data` function.

```r
# Load necessary libraries
library(MSDRCapstone)
library(dplyr)

# Sample data frame
sample_data <- data.frame(
  time = c("2020-01-01", "2020-02-01", "2020-03-01", "2020-04-01", "2020-05-01"),
  latitude = c("34.05", "36.16", "40.73", "37.77", "34.05"),
  longitude = c("-118.24", "-115.15", "-73.93", "-122.42", "-118.24"),
  deaths = c("10", "50", "200", "150", "30"),
  locationsource = c("USA: California", "USA: Nevada", "USA: New York", "USA: California", "USA: California"),
  mag = c(5.5, 6.0, 7.2, 6.8, 5.9)
```

```
)

# Clean the data
cleaned_data <- eq_clean_data(sample_data)
print(cleaned_data)
```

Step 2: Create Timeline Plot

Next, we create a timeline plot of the earthquake events using the `geom_timeline` function.

```r
# Create a timeline plot
timeline_plot <- geom_timeline(data = cleaned_data, x = DATE,
size = mag, color = deaths)
print(timeline_plot)
```

Step 3: Create Timeline Plot with Labels

We can also create a timeline plot with labels for the top 3 earthquakes by magnitude using the `geom_timeline_label` function.

```r
# Create a timeline plot with labels
timeline_label_plot <- geom_timeline_label(data = cleaned_data,
x = DATE, label = LOCATION_NAME, n_max = 3)
print(timeline_label_plot)
```

Step 4: Create Interactive Map

We create an interactive map of the earthquake events using the `eq_map` function. The map will display detailed labels in popups.

```r
# Create an interactive map
earthquake_map <- eq_map(data = cleaned_data, annot_col =
"LOCATION_NAME")
earthquake_map
```

Step 5: Create Detailed Labels

Finally, we create detailed labels for the earthquake data using the `eq_create_label` function. These labels can be used in the interactive map.

```r
# Create detailed labels
detailed_labels <- eq_create_label(cleaned_data)
print(detailed_labels)

# Add detailed labels to the map
earthquake_map_with_labels <- eq_map(data = cleaned_data %>%
mutate(label = detailed_labels), annot_col = "label")
earthquake_map_with_labels
```

Conclusion

This comprehensive example demonstrates how to use the `MSDRCapstone` package to clean earthquake data, visualize it through timelines and interactive maps, and create detailed labels for deeper analysis. By combining these functions, you can effectively explore and present earthquake data in a meaningful way.

Conclusion

The MSDRCapstone package provides a set of versatile functions for data analysis and visualization. This vignette demonstrated how to use each function individually and in combination to achieve a comprehensive analysis.

For more detailed information on each function, refer to the package documentation and function-specific help pages.