jrSpatial - Practical 4

Jumping Rivers

We'll start by loading the necessary packages.

```
library("sf")
library("dplyr")
library("tmap")
library("jrSpatial")
```

Question 1

1. Create a dataframe representing our current location and it's longitude and latitude. `loc = data.frame(lon = , lat = )`. You can use Google Maps to find our current longitude and latitude.

2. In practical 1, we learned how to create a **sf** data frame from scratch using `st_sf()`. Because our coordinates are just numeric columns instead of a single geometry column, we can't use `st_sf()`. Instead we have to use `st_as_sf()` and specify the coordinate columns using the `coords` argument, like so

```
loc = st_as_sf(loc, coords = c("lon", "lat"))
```

3. Inspect the dataframe. Use the function `st_is_longlat()` to check whether the data is recognised as using longlat degrees. What about `st_crs()`?

4. Set the CRS to have epsg 4326 using `st_set_crs()` and then check `st_is_longlat()` again.

   Question 2.

1. Load the shape files for New Zealand using `data(nz_missing, package = "jrSpatial")`. Plot the data using `tm_shape()` with `tm_borders()`. What do you notice?

2. The Canterbury region of New Zealand is missing from the data. Load in the Canterbury data using

```
data(canterbury, package = "jrSpatial")
```

Say we have two spatial data frame, `x` and `y`, we can row bind them by using

```
rbind(x, y)
```

Try to use `rbind()` to add the Canterbury row to the original New Zealand data. What does the output tell you?

3. The two data sets have different CRS. Find out the CRS of the New Zealand data using `st_crs()`. Then use `st_transform()` to transform the Canterbury data to use the same CRS as the rest of New Zealand. Combine the data with `rbind()` and plot the output.

Solutions

```
vignette("solutions4", package = "jrSpatial")
```