RStudio Project to Zenodo exercise

In this exercise you will setup of a basic RStudio project following good practices for organizing data, writing clean code and decomposing your workflow introduced in Rproject section. We'll also initialize `renv` to manage package dependencies, ensuring reproducibility. Once the project is set up, we'll go through the process of how to publish the project to Zenodo as described in the Zenodo workflow section.

Step 1: Download the exercise resources

- Click here to download the resources for the exercise: Download resources for exercise
- Unzip the downloaded file and move the folder to a location on your computer where you can easily find it.

Step 2: Create a new RStudio Project

Start by creating a new RStudio project in the root of the `exercise_1_data` directory. you have just downloaded. You can name the project what you like but in this example, we have named it `Rice_farm_analysis.proj`.

- Open RStudio.
- Create the project using: *File > New Project > Existing Directory.*
- Select the `exercise_1_data` folder as the location and give the project a name, for example, `Rice_farm_analysis.proj`.

This creates a `.Rproj` file in the root of your project to help manage the workspace and project-specific settings.

Step 3: Organize Your Data

It's good practice to organize raw and processed data in separate folders. Let's start by organizing the data:

- Create a directory `Data/Raw` inside your project folder.
- Move the provided CSV file into this `Data/Raw` directory.

Step 4: Organize and Split Your Scripts

We'll now organize the project's scripts by splitting the original script into separate analysis and visualization scripts.

- Create a `Scripts` folder inside your project directory.
- Move the original `RiceFarm_project.R` script into the `Scripts` folder.

We'll now organize the project's scripts by splitting the original script into separate *analysis* and *visualization* scripts.

- Create a `scripts` folder inside your project directory.
- Move the original `RiceFarm_project.R` script into the `scripts` folder.
- Create two new scripts named `01_data_analysis.R` and `02_data_visualisation.R`.

For `01_data_analysis.R`

- Copy the following code from `RiceFarm_project.R`:
  - The call to the relevant library `library(stringr)`
  - Everything before the call to the `ggplot()` function
- In addition to this, replace the `setwd()` function with this code to set up relative paths, create a directory to save the processed data in and save `rice_data_summary` to disk after processing:

```
# vector and create processed data save dir
save_dir <- "Data/Processed"
dir.create(save_dir,
           showWarnings = FALSE,
           recursive = TRUE)

# Load csv file of data
rice_data <- read.csv(file.path(raw_dir, "RiceFarms.csv"))

# Save the summarized data
write.csv(rice_data_summary,
          file.path(save_dir, "RiceFarms_summary.csv"),
          row.names = FALSE)
```

For `02_data_visualisation.R`

- Copy the following code from `RiceFarm_project.R`:
  - The call to the relevant library `library(ggplot2)`
  - The call to the functions `ggplot()` and `ggsave()`
- Add the following code after the `library()` call to create an output directory for the plots and load the summarized data from the processed data folder:

```
# Directory for saving plots
plot_dir <- "Output/Visualisations"
dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

# load the summarized data
rice_data_summary <- read.csv("Data/Processed/RiceFarms_summary.csv")
```

Step 5: Add Script Headers

- Add headers to both new scripts. You can use this template:

```
# ---------------------------------------------
# Script Name: [01_data_analysis.R / 02_data_visualisation.R]
# Project: Rice Farm Analysis
# Purpose: [Data analysis / Data visualization]
# Author: [Your Name]
# Date: [YYYY-MM-DD]
# ---------------------------------------------
```

Step 6: Create a master script

As a next step you will create a master script that runs both the data analysis and visualization scripts.

- In the root of your project, create a new file named `RiceFarm_master.R`
- Add a header as in Step 5.
- Add the following code snippet to the script to source `01_data_analysis.R` and `02_data_visualisation.R`:

```
###============================================================================
### 01- Data analysis
### ============================================================================

# Source the data analysis script
source("Scripts/01_data_analysis.R")

### ============================================================================
### 02- Data visualization
### ============================================================================

# Source the data visualization script
source("Scripts/02_data_visualisation.R")
```

Running this master script will execute both analysis and visualization steps.

Step 7: Initialize `renv` to manage package dependencies

We will use `renv` to make your projects package environment reproducible.

- install renv package

```
install.packages("renv")
```

- Run the following command in your master script to set up the project-specific environment

```
renv::init()
```

This creates a local library for your project and captures the required packages.

- Once the initialization is complete, run:

Step 8: Automate Opening the Master Script

For convenience, we can configure RStudio to automatically open the master script when the project is loaded.

- install the rstudioapi package:

```
install.packages("rstudioapi")
```

- Open the `.Rprofile` file in the root of your project directory. The file might be hidden. On Windows click "View" > "Show" > "Hidden items" in the explorer and on MacOS click Press Command+Shift+Dot within the root directory to see the file.

```
renv::snapshot()
```

This records the project's environment in a `renv.lock` file, which is essential for reproducibility.

Step 8: Automate opening of the master script

For convenience, we will configure RStudio to automatically open the master script when the project is loaded.

- Open the `.Rprofile` file in the root of your project directory. The file might be hidden. On Windows click *View* > *Show* > *Hidden items* in the explorer and on MacOS click Press Command+Shift+Dot within the root directory to see the file.
- Add the following R code to the `.Rprofile` file:

```
rsetHook("rstudio.sessionInit", function(newSession) {
  if (newSession)
    rstudioapi::navigateToFile('RiceFarm_master.R', line = -1L, column = -1L)
}, action = "append")
```

Step 9: Re-snapshot the Project

After modifying the `.Rprofile` file, it's important to capture these changes in the `renv.lock` file.

- Run the following command in your master script to ensure that the `rstudioapi` package (which enables automatic script opening) is included in the snapshot:

```
renv::snapshot()
```

Now we a have a nicely organised project structure with the workflow decomposed into seperate scripts and a master script that to run the whole project.

Step 10: Install `zen4R` to access Zenodo through R

The following steps are heavily based on [1]. We have extracted the most relevant parts to explain the workflow. If you are interested in more details, check out their user manual at: https://cran.r-project.org/web/packages/zen4R/vignettes/zen4R.html.

For this exercise we will not be using Zenodo directly but Zenodo Sandbox. The Zenodo Sandbox is a separate, secure testing environment where users can explore Zenodo's features without impacting the main platform's publicly accessible data. It allows you to test file uploads, generate test DOIs, and experiment with API integrations. DOIs created in the sandbox are only for testing and use a different prefix. You will need a separate account and access token for the sandbox, distinct from those used on Zenodo's main site.

- Create an account on https://sandbox.zenodo.org.

Zenodo can be accessed with the R package `zen4R` to upload, edit, publish and download data.

- Create a new R script outside of the project directory.
- Install zen4R library with the following code:

```
#install dependency "remotes"
install.packages("remotes")

#install zen4R
require("remotes")
install_github("eblondel/zen4R")
```

Step 11: Create a new Zenodo record

A Zenodo record includes metadata, data and a Digital Object Identifier (DOI) which is automatically generated by Zenodo for all uploads. But before you can add records to Zenodo, you need to get access to your account through R.

- Go to https://sandbox.zenodo.org/account/settings/applications/.
- Log into your account and then create a new "Personal access token" in the "Applications" section of your account.
- Then run the following code in your script to establish the access and create a new record.

```
library(zen4R)

#Create manager to access your Zenodo repository
zenodo <- ZenodoManager$new(
    token = "your_token",
    sandbox = TRUE,
    logger = "INFO"
    )

##Prepare a new record to be filled with metadata and uploaded to Zenodo
myrec <- ZenodoRecord$new()
```

If you want to connect to Zenodo and not Zenodo Sandbox, create the token in your Zenodo account and remove the line `sandbox = True` in the code above.

The types of metadata that can be included in a Zenodo record are vast. A full list can be found in the documentation.

- Copy and run the example below to add metadata to your record.

```
myrec$setTitle("RiceFarm") #title of the record
myrec$addAdditionalTitle("This is an alternative title", type = "alternative-title")
myrec$setDescription("Calculating statistics of RiceFarm dataset") #description
myrec$addAdditionalDescription("This is an abstract", type = "abstract")
myrec$setPublicationDate("2024-09-16") #Format YYYY-MM-DD
myrec$setResourceType("dataset")
myrec$addCreator(firstname = "Yourfirstname", lastname = "Yourlastname", role = "datamanager"
myrec$setKeywords(c("R","dataset")) #For filtering
myrec$addReference("Blondel E. et al., 2024 zen4R: R Interface to Zenodo REST API")
```

A record can be deposited on Zenodo before it is published. This will add the record to your account without making it public yet. A deposited record can still be edited or deleted. You can also upload data to a deposited record. If you prefer a graphical interface, you can also edit the record on the Zenodo website.

- Deposit the record on Zenodo:

```
#deposit record
myrec <- zenodo$depositRecord(myrec, publish = TRUE)
```

- View the deposited record at https://sandbox.zenodo.org/me/uploads?q=&l=list&p=1&s=10&sort=newe
- Compress your project directory to a .zip file.

- Upload the .zip file to your deposited record:

```
#add data to the record, adjust the path below
zenodo$uploadFile("path/to/your/file", record = myrec)
```

- Publish the record:

```
#make the record publicly available on Zenodo (Sandbox).
myrec <- zenodo$publishRecord(myrec$id)
```

Step 12: Edit a published Zenodo record

- It is also possible to edit or update the metadata of published records:

```
#get your record by metadata query, e.g. by title
myrec <- zenodo$getDepositions(q='title:zen4R')

#get depositions creates a list, access first element
myrec <- myrec[[1]]

#edit metadata
myrec <- zenodo$editRecord(myrec$id)
myrec$setTitle("zen4R 2.0")

#redeposit and publish the edited record
myrec <- zenodo$depositRecord(myrec, publish = TRUE)
```

- Once a record has been published, it is not possible to edit the data that has been attached to it. However, it is possible to upload an updated version of the data. The previous version of the data will remain accessible via Zenodo. The record will have one overall DOI, while each version will have its own DOI.
- Reconnect to your account:

```
zenodo <- ZenodoManager$new(
      token = "your_token",
      sandbox = TRUE,
      logger = "INFO")
```

- Access your record:

```
#get your record by querying the metadata, e.g. by title, this will give you a list of all re
myrec <- zenodo$getDepositions(q='title:RiceFarm Statistics')

#access the first item in the list, as there should only be one record with that particular t
myrec <- myrec[[1]]
```

- Rename your .zip file on your computer
- Upload the renamed .zip file:

```
#edit data, delete_latest_files = TRUE to not include data of previous version in newer versi
myrec <- zenodo$depositRecordVersion(myrec, delete_latest_files = TRUE, files = "path/to/your
```

- Again, go to https://sandbox.zenodo.org/me/uploads?q=&l=list&p=1&s=10&sort=newest
- Activate “View all versions” on the left hand side.
- Check if both versions show up

1. Blondel E, Barde J (2024) zen4R: R Interface to Zenodo REST API

<script type=”application/javascript” src=”light-dark.js”></script>