

Repository and Mining of Temporal Data

# RepoMining

User Manual

**Team Members:**

Jessica Nguy:	jnguy2014@my.fit.edu
Siomara Nieves:	snieves2014@my.fit.edu

**Faculty Sponsor:**

Dr. Philip Chan:	pkc@fit.edu
------------------	-------------

**Project Website:**

<https://jlnguy.github.io/RepoMining/>

# Table of Contents

Introduction	2
Glossary	3
Getting Started	4
Setting Up the System	4
Uploading a file	5
Analyzing a Target	6
Moving points in graph	7
Zooming in points in graph	7
Reset graph	8
Interpreting Results	8
Target Variable Search	9
Navigating through the site	9
Example for Data Providers	10
Example for Data Consumers	10
For Developers	11
Code Overview	11
Python Code	11
Django	12
Imports	13
Future Improvements	13
GitHub Structure	13

## Introduction

Repository and Mining of Temporal Data is an application designed to help users understand more about their data. It focuses on answering three main questions:

- 1.) Is there a significant change in the target variable?
- 2.) Why was there a significant change, and what are the top-k variables that have affected the target variable?
- 3.) What is the value in the next timestamp?

RepoMining aims to simplify the visualization and analysis of information (in the form of a .csv file with timestamps and data) by answering the previous questions, and by creating a system that is simple, easy to navigate, and easy to use.

Eight (8) visualizations will be returned to the user: Four (4) in Question 1, three (3) in Question 2, and one (1) in Question 3. The program analyzes data selected against all the data in the program provided that it has the same granularity and timestamp thresholds. The application is hosted on the web using Localhost.

The application has two types of users, Data Providers and Data Consumers. The first type of user contributes to the system by uploading their files, and the second type of user uploads the files (target variables) they want the program to analyze and obtain results on it.

The first question is answered by 4 graphs: time vs values, time vs change in values, time vs z-score, and time vs change in z-score. The visualizations include a zoom option for viewing the points and color coding in order to detect the most change. For the purposes of the program, Green indicates Low/No Change, Yellow indicates Some change, and Red indicates areas of High change and areas of interest.

The second question shows the top-k variables which could be affecting the target variable (of the Data Consumer) which is set to a default of three top variables. These top-k variables are picked depending if they have the same granularity (daily/monthly/yearly timestamps); if they fall within the same period of time; if they have data before the starting timestamp of the target; if they have data after the ending timestamp (for answering the third question using Linear Regression); and if they have correlation at all.

The first graph shows time vs the values of all 4 files, the target and other three variables in order to detect some type of pattern between them (ex. As the target variable goes up, the other goes down). The second graph shows the target vs the correlation values generated of the

three variables, with the scale being from 0.0 to 1.0. The closer to 1.0 a score is, the more correlated to the target variable it is. In only one case should the correlation be exactly 1.0: the target variable calculating correlation to itself. The correlation values are presented in absolute values. The last graph shows the top-1 correlation values calculated for each lag time.

Currently the program assumes a lag time of 5, with whatever the granularity indicates. For example, a granularity type of Yearly would assume that the lag time would go as far back as 5 Years. With a granularity type of Daily the lag time would go as far back as 5 Days.

The last question consists on only 1 graph which aims to ‘predict’ or forecast the next possible value for the target (whether it will go up or down in the next time period). It shows the time vs values and adds the newly generated point in red.

## Glossary

Data Consumers - Users that are looking to use RepoMining to analyse their data. May or may not consist of uploading their own data to analyze, or searching through the repository for data to analyze.

Data Provider - Users looking to help populate the repository by uploading their own files.

Granularity - Refers to the timescale of a dataset (daily, monthly, yearly).

Metadata - Refers to the additional information that users put in when uploading files. Contains the file name, tags, type of granularity, first timestamp, last timestamp, and ID number in the repository.

Q1 - Shorthand for Question 1.

Q2 - Shorthand for Question 2.

Q3 - Shorthand for Question 3.

## Getting Started

### *Setting Up the System*

Download the repository

Open cmd (or terminal)

Go in the 'repo' folder

`cd repo`

Type `python manage.py runserver`

The server will be available at 127. .0.0.1:8000 (LocalHost)



## Uploading a file

Click the Upload button on the navigation bar or on the homepage

Select a file (.csv file)

Add metadata (target variable, description, tags, granularity)

Upload

If the file is accepted, it will be saved on the system's database and redirected to a thank-you page

The image shows two screenshots of the RepoMining website. The top screenshot is the 'Contribute' page, which has a dark navigation bar with links: Home, About, Upload, Analyze, Contact, and a search bar with a 'GO' button. The main header area features a background image of a smartphone and a laptop with the text 'Contribute' and 'add your data to our system'. Below this, the 'Requirements:' section lists: 'Comma-separated values (.csv) Format', 'Two columns: Time and Data', and '1 tag required', with a 'Download Sample' link. To the right is the upload form with fields for 'Upload file:' (with a 'Choose File' button and 'No file chosen' text), 'Target:', 'Description:', 'Time:' (a dropdown menu), 'Tag 1:', and 'Tag 2:'. A blue 'Upload' button is at the bottom right of the form. The bottom screenshot is the 'Thank You!' page, which has the same navigation bar. The main content area says 'Thank You!' and 'We appreciate you helping our system.' It includes a link 'Having trouble? Contact us' and a blue 'Continue to homepage' button. At the very bottom, there is a light gray box with the text 'Click below to upload more files' and a blue 'Upload' button.

RepoMining

Home About Upload Analyze Contact Search... GO

# Contribute

add your data to our system

**Requirements:**

- ★ Comma-separated values (.csv) Format
- ★ Two columns: Time and Data
- ★ 1 tag required

[Download Sample](#)

Upload file:  No file chosen

Target:

Description:

Time:

Tag 1:

Tag 2:

[Upload](#)

Activate Windows  
Go to Settings to activate Windows.

RepoMining

Home About Upload Analyze Contact Search... GO

# Thank You!

We appreciate you helping our system.

Having trouble? [Contact us](#)

[Continue to homepage](#)

Click below to upload more files

[Upload](#)

## Analyzing a Target

Click the Analyze button on the navigation bar or on the homepage

Select the target file to analyze

Add metadata (target variable, description, tags, granularity)

Click Upload

Wait (less than 1 minute, depending on the file size)

Results will be displayed on the new page (Q1, Q2, Q3)

RepoMining

Home About Upload Analyze Contact Search... GO

# Analyze

change - factors - forecast

**Requirements:**

- \* Comma-separated values (.csv) Format
- \* Two columns: Time and Data
- \* 1 tag required

[Download Sample](#)

Upload file:  No file chosen

Target:

Description:

Time:

Tag 1:

Tag 2:

[Upload](#)

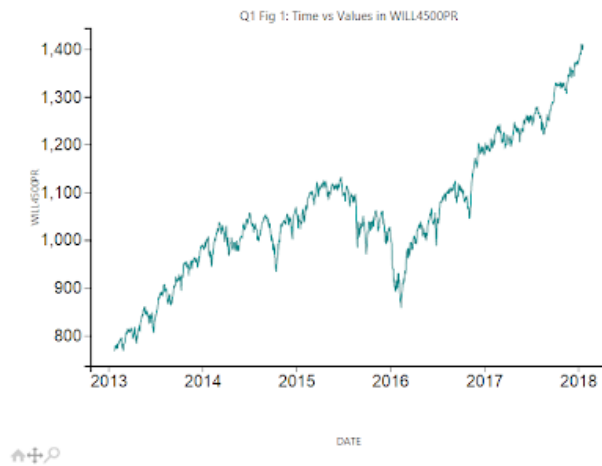
Activate Windows  
Go to Settings to activate Windows.

## *Moving points in graph*

Hover over graph (Q1, Q2, or Q3)

Click the pan button

Move around the graph

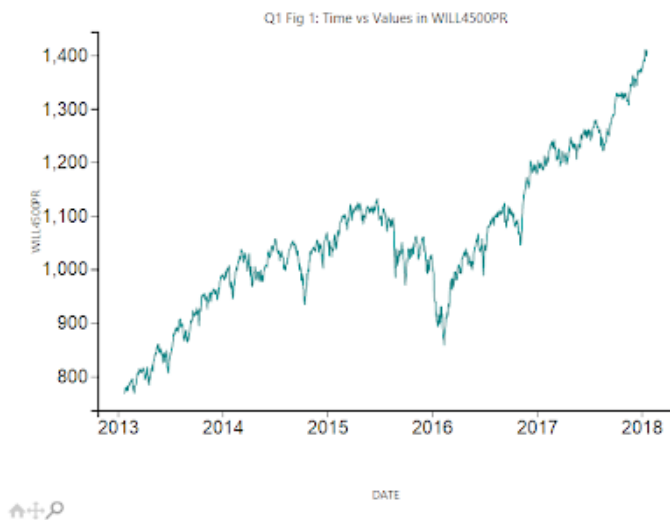


## *Zooming in points in graph*

Hover over graph (Q1, Q2, or Q3)

Click the magnifier button on the lower left

Select the points

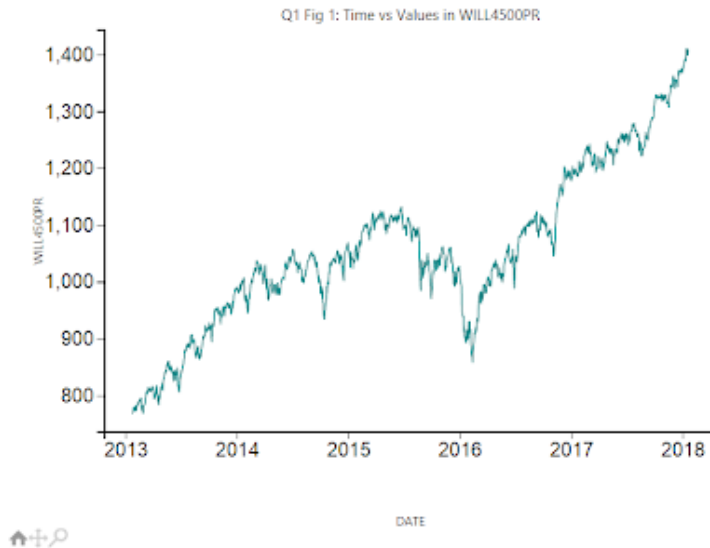




## Reset graph

Hover over graph (Q1, Q2, or Q3)

Click the house button on the lower left



## Interpreting Results

At the moment of writing this document, the website does not handle showing the predicted vs actual results of the analysis, but can be viewed on the Command-Prompt/Terminal.

```
Analysis:
R^2 prediction: 0.8582406760014895
---
Mean Absolute Error: 41.57615061070526
Mean Squared Error: 2775.5948483795796
Root Mean Squared Error: 52.68391451268195
Percent Error: 4.051213614638982
Percent Difference: 8.102427229277964
---

Coefficient
0 0.110758
1 0.034280
2 0.034280

Last timestamp on graph: 2018-01-18 00:00:00
Last value on graph: 1404.46
X-Value: 2018-01-19 00:00:00
Y-Value: 1322.0403898712543
```

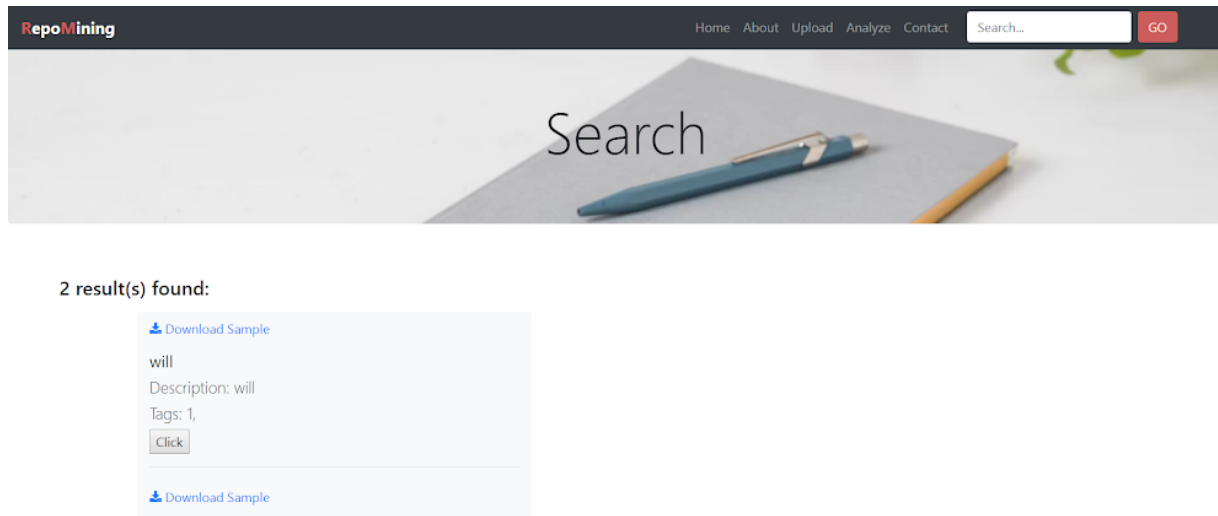
## *Target Variable Search*

Go on the navigation bar (while on any page of the website)

Type on the search bar

Click enter

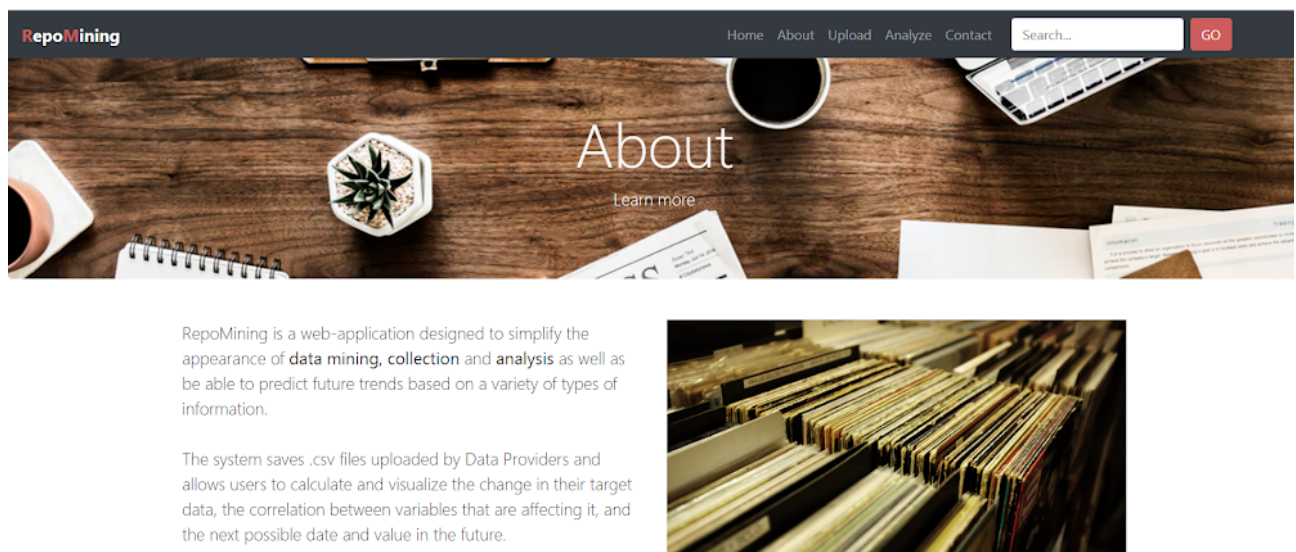
Visualize the search result



## *Navigating through the site*

Go around the site as any other webpage

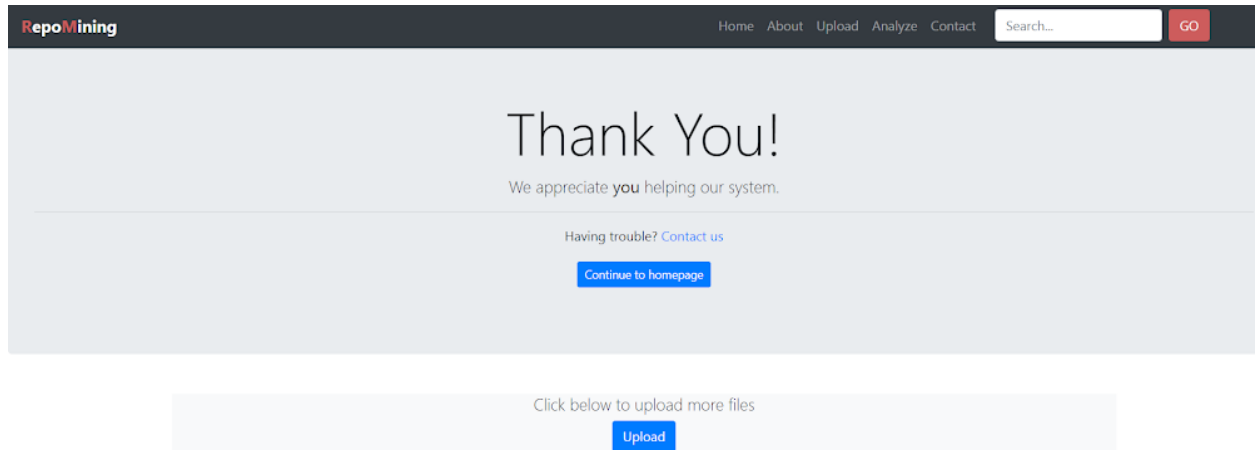
All buttons work as desired redirecting to the correct pages



## Example for Data Providers

Contributing with their data to the system:

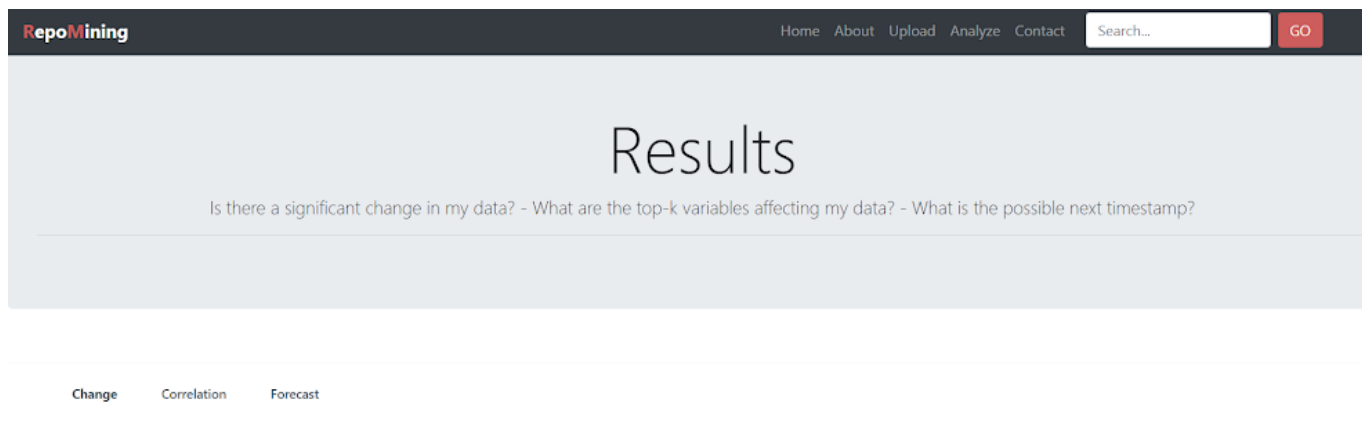
1. Click the 'Upload' button on the home page or navigation bar
2. Choose a file (.csv format, two columns)
3. Fill out metadata (target, description, granularity, tags)
4. Click 'Upload'
5. If performed correctly, a thank you page should load.



## Example for Data Consumers

Analyzing their data:

1. Click the 'Analyze' button on the homepage or navigation bar
2. Choose the file to be analyzed (.csv format, two columns)
3. Fill out the metadata (target, description, granularity, tags)
4. Click 'Upload'
5. Next page will show the results.



## For Developers

The following pages details structure, code, and API used in creating RepoMining. This section should be used in the case that another group picks up this project in subsequent Senior Design classes.

### *Code Overview*

RepoMining is written with Python 3, Django, HTML, and CSS. The program uses a sqlite3 database to store metadata. All the .csv files are saved locally on the disk.

### *Python Code*

A summary of each class being used is listed. For reference, these files are stored in the *src* folder in the GitHub Repository.

Initialize.py - Code that creates a database with the current schema set. It creates an sqlite3 database called data.db and creates empty tables for the programmer to fill. It should be used to create a dummy (testing) database. For error fixing, a flush should be called on the database instead.

NarrowData.py - Main analysis of equation for the project. Is called during Q2.py.

NarrowData.py's main area of analysis and retrieval is called in getID. The other classes are used for reference in Q2.py and used to prepare the arguments to be used in getID. All functions in NarrowData.py open a connection to the sqlite3 database and close it. This should be noted when one wants to change the database used or is using a dummy database for testing purposes.

ParseComma.py - Opens a csv file and reads it through. Converts the data to be a datetime type for ease of reading, analysis, and display.

ParseData.py - Prepares data to be used in Q2. Is called after NarrowData. This code shuffles the data to match up with the target data using the lag times as reference. Has two types of shuffles: one for the values in the data (shuffleData) and one for the y-axis (shuffleY). Also has a section named trimData that makes sure that all data sets are the same length so matplotlib does not have issues.

Q1.py - Answers the following question: Is there a significant change?. All visualizations are indicated in the code and should be easy to follow.

Q2.py - Answers the following question: What are the top-k values, and what is the correlation value of it?. Also calls NarrowData, ParseData, and several other classes. Imports several APIs. The analysis section happens in most of the code with the visualization section happening near the last 50 lines of code. Also calls Q3.py, passing through a matrix of correlation values to the next class.

Q3.py - Answers the following question: What is the value in the next timestamp?. Uses a matrix passed through in Q2.py to calculate correlation weights. The equation,  $\text{Target} = w_0 + w_1x_0 + w_2x_1 + \dots + w_nx_{n-1}$  is used. Target being the value in the next timestamp,  $w_0$  being the y-intercept,  $x_n$  being the value of correlation, and  $w_1$  to  $w_n$  being the weights calculated using the linear regression. The timestamp is simply incremented by one based off the type of granularity used.

Upload.py - Code used to upload a file to the database. Requests that the user input several values: The file name, a description of the file, the target variable of the file (To be used for searching later on), granularity type, and 4 tags. If using on the Django site, it only requires 1 tag with 2 being visible.

csvReader.py - Reads and converts the file to two lists in Python so that Q1.py can use it to answer the question.

data.db - The database of files uploaded already.

parseCSV.py - Reads and converts the file to two lists in Python. Also converts the x-axis files into dateTime items.

Unused classes: Input.py, Pcorr.py.

## *Django*

Used to create the GUI and website interface. Also used to host the website on LocalHost. Porting over to an actual hosting site, such as PythonAnywhere or Amazon Web Services, is suggested should this project continue another year.

## *Imports*

APIs (or in our definition, items we imported to use in RepoMining) used in our program include but are not exclusive to:

csv, numpy, pandas, datetime, matplotlib, math, scipy, sqlite3, and sklearn's LinearRegression.

## *Future Improvements*

A list of possible future improvements are as follows. Some are suggested by our faculty sponsor, others by the programmers themselves.

Fixing the display to show Q2 Fig.3.

Changing the visualization software from matplotlib to something else, such as Google Charts, or something else. Currently there are many errors with translating matplotlib to Django. Many API's used with matplotlib do not work or have unknown errors. The current visualization software, mpl3, is deprecated and is no longer being updated.

Color-coding the correlation values for Q2 Fig.2.

Optional download of graphs and/or .csv files once analysis is done. Add an option to download .csv files from the repository.

Implement tag search in NarrowData.

Depending if NarrowData is successful or returns a number less than 3 (currently set as top-k = 3), show the other analysis rather than showing nothing.

Optimize the program. The program runs extremely slow for large values in the repository.

## *GitHub Structure*

The structure of the GitHub repository, and items saved in the folders, is summarized as follows:

CSV Files - Contains the .csv files used for analysis. This folder also contains a data.db, which is a sqlite3 database that has all the current files uploaded, and a data.sql, which contains the schema of data.db. The file data.sql can be opened up in a text editor for ease of reading.

RepoMining/RepoMining - Contains Django version I code. (Very old)

mysite - Contains initial tutorial django code.

repo - Contains the Django code.

seniorDesignDocs - Contains the documentation of the project. Includes Project Plans, Milestone Evaluations, Milestone Presentation, Showcase Poster, and the User Manual. Most of these files can be accessed via the ReadMe file, the description file on the GitHub site, or the Project website.

src - Contains the raw Python code for the program. The Python code can be run by itself, although many of the sections do not call each other. The main classes, Q1.py, Q2.py, and Q3.py do not run one after the other. Q2.py and Q3.py run one after the other. For more information, see above.

README.md - Contains the description and links to create the Project website.

Scribbles - Small note of what RepoMining is..

\_config.yml - Creation and setting of the Project website theme.