# README

This file provides the relevant code for the ***Functional Volatility Forecasting*** (*Tan Y, Tan Z, Tang Y, et al. Functional Volatility Forecasting [J]. Available at SSRN 4543626, 2023*). Due to confidentiality requirements, empirical data from the project cannot be disclosed. Therefore, only a portion of simulated test data is provided here for testing purposes. This document is divided into three sections: **Data**, **Code**, and **Result**. Each section will be introduced separately below.

1 Data

The dataset comprises three types of simulated data: the "**Smooth**" folder, as well as "**Jump-Price**" and "**Price**" within the "**CIR**" folder. The "**IV**" folder within each folder reflects the corresponding setting of volatility. These three types of data correspond to the three scenarios (a), (b), and (c) of Section 4.1 "*The Simulation Setting*" in the ***Functional Volatility Forecasting***. Each type of data provides 10 trajectories.

2 Code

The Code folder includes the main code file, **main.R**, and a **PreRequisiteCode** subfolder containing prerequisite code files.

2.1 main.R

The main code mainly references files from the PreRequisiteCode subfolder. Users simply need to download the code package, adjust the **first two lines** of the main code file to set the download path and select the type of test dataset (designated as `Smooth`, `CIRnojump`, or `CIRjump` for scenarios (a), (b), and (c) respectively), and the program will run automatically.

If readers want to customize the code to achieve personalized goals, they need to make changes to the files in the '**PreRequisiteCode**' folder and the data in the '**Data**' folder. The data should be provided in the format specified in the Data dataset.

The main content comprises three parts:

- **Settings**:
  - Readers only need to modify the download path in the code package, and the '**AutoSetting.R**' file will automatically configure other settings. If errors occur, please check if any dependent packages are not installed.
- **Volatility Forecasting** :
  - There are two code files to help us complete the volatility forecasting part.
  - The first code file is **'BaselineModel.R'**, which will provide the prediction results of GARCH, RGARCH, MEM, HAR, and HEAVY models, and save the prediction results to the Result folder.
  - The second code file is **'fVP.R'**, which provides the proposed fVP method and fVP-kr method mentioned in the paper. The fVP-kr method is commented out in the code file (i.e., it will not be executed automatically).
- **Result Analysis** :
  - This part calculates various loss function results for different prediction methods under different sample trajectories using the **'ResultAnalysis.R'** code file. The loss functions include: MAE, RMSE,

HMAE, HRMSE, and AMAPE. Finally, it returns a result array `lossfun_Result`, where each dimension represents the loss function, prediction method, and sample trajectory respectively.
    - Users can choose different dimensions for analysis. In the example code, analysis based on MAE is provided, along with density estimation for the loss function. However, it should be noted that reliable density estimation is only possible when there is a sufficient amount of sample trajectory data. **The example dataset is small and may not fully reflect the situation.** Readers can generate simulated data trajectories for verification.

2.2 PreRequisiteCode

This folder contains various prerequisite codes, which users can read or modify as needed. In addition to the aforementioned code files, there are three dependency files: **'Sample_functions.R'**, **'kernel_predict_functions.R'**, and **'Likelihood_functions.R'**. These files are prerequisites required for the fVP, fVP-kr, and baseline models methods respectively.

3 Result

This folder serves as the location to store prediction results. Depending on the `type` specified, the prediction results are saved in the corresponding folder. The folders already contain some prediction results generated from forecasting simulated data. Readers can update and overwrite the existing results by running the main code.