Code Guide

The **ipynb** files contained in this folder are used to run combined R and JULIA code.

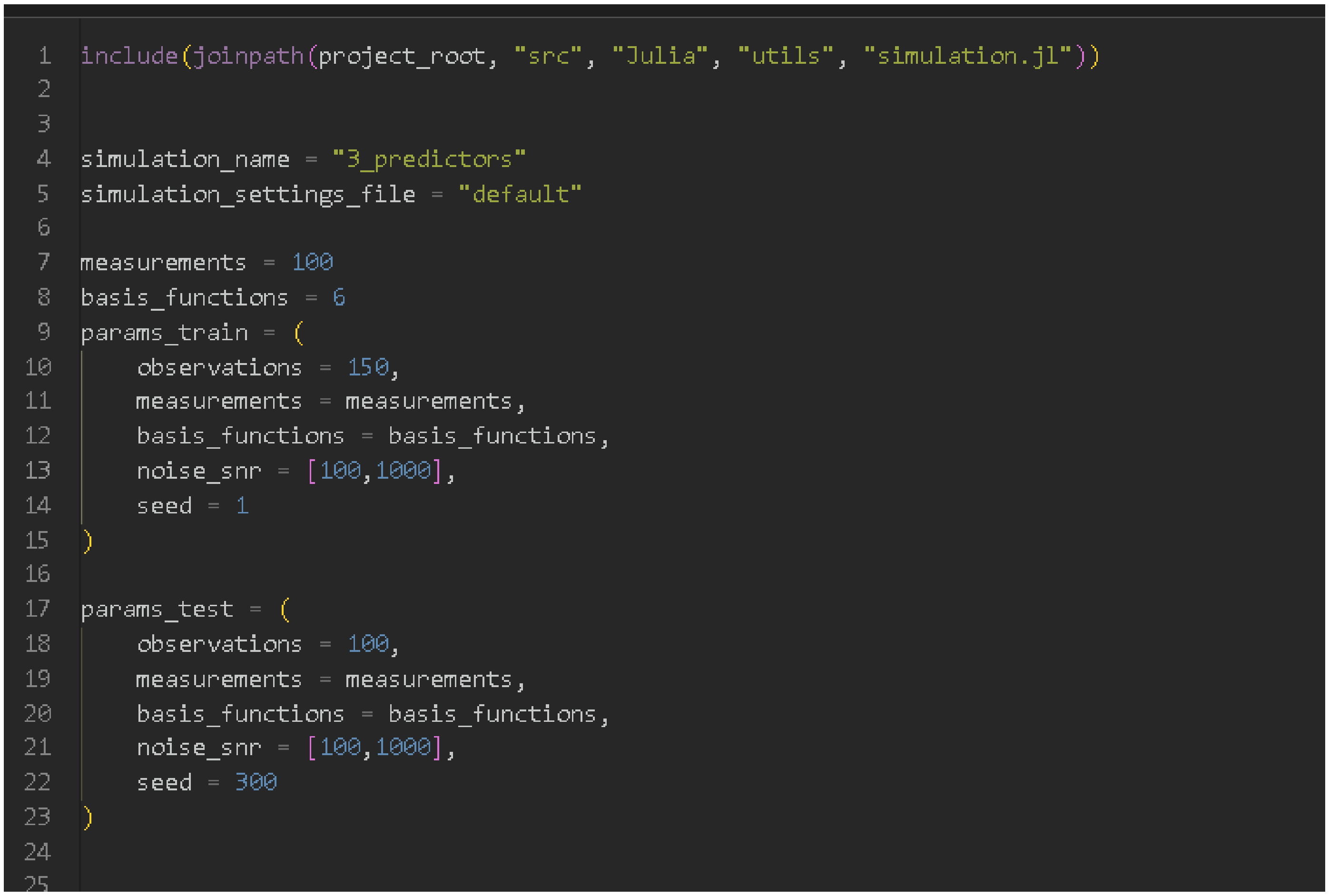
1. Setting the simulation params

Cell #1 is used to setup the environment and load the libraries.
Edit the **functional-data-regression-mip\setup\init_env.jl** with the correct path to your R user_libs in case Jupyter has some problems.

2. Setting the simulation params

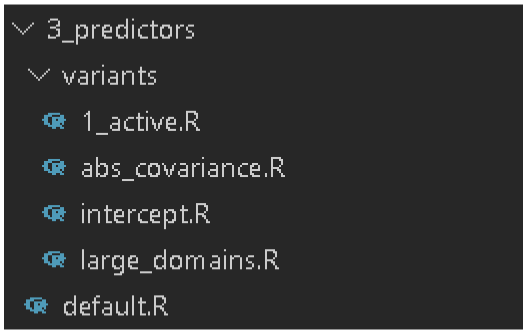
Cell Block #2 is used to set:

- the simulation setting: i.e. simulation name and simulation settings file
- the simulation parameters: i.e. #observations, basis functions, measurements..



```
include(joinpath(project_root, "src", "Julia", "utils", "simulation.jl"))


simulation_name = "3_predictors"
simulation_settings_file = "default"

measurements = 100
basis_functions = 6
params_train = (
    observations = 150,
    measurements = measurements,
    basis_functions = basis_functions,
    noise_snr = [100,1000],
    seed = 1
)

params_test = (
    observations = 100,
    measurements = measurements,
    basis_functions = basis_functions,
    noise_snr = [100,1000],
    seed = 300
)

```

The simulation setting:

- **simulation-name**: refers to the that contains the simulation file. It is found inside "functional-data-regression-mip/simulations/settings/
- **simulation-setting-file**: the name of the proper data simulation file. It could either be the default setting file (which contains all the basic informations about that simulation) or a variant (which can override certain fields of that specific configuration).
  i.e."1_active" overrides the active predictors of the 3_predictors simulation setting
  

```
3_predictors
  variants
    1_active.R
    abs_covariance.R
    intercept.R
    large_domains.R
  default.R
```

3. Setting the simulation params

Cell Block #3 is used to run the generation of the simulation data and to load the generated data in the notebook environment.
The functions used are inside the path ""functional-data-regression-mip/src/R/generic-simulation/""

To specify **how** to to generate data, use the methods:

- **Data manipolation** functions are inside "functional-data-regression-mip\src\R\generic_simulator\utils"
  > - **basis_utilities**, utilities that use the fda package to expand X and Betas given a specified time domain (which could be different between the X predictors but it is implied to be the same between each pair of predictor-beta func)
  > - **covariance_utilities**, covariance functions defined to generate variability over the X predictors. Check the README in the root folder for detailed informations.
  > - **model_utilities**, functions to compute W,J,Z using the basis expansion approach of both the X and B
  > - **simulation_utilities**, common functions to compute Y values using trapezoidal approx

- **load_simulation_data**
  > - runs "functional-data-regression-mip\src\R\generic_simulator\simulate_main.R"
  > - uses "functional-data-regression-mip\src\R\generic_simulator\simulation\cov paper and paper2 files"
  > - inside the **simulation-setting-file** is defined the **simulation-type** variable that defines how X and Y should be computed

        - **simulation-type= paper1**, requires Zambom paper data generation process which envolves adding an amplitude noise over the observed X data and response Y.
        - **simulation-type= paper2**, uses the same data generation function provided by Gerthais paper. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4131701/bin/NIHMS535202-supplement-SUPP_FUNT_CODE.r so only the "simulation parameters" will be used since Gerthais defines the specific generator function.
        - **simulation-type= cov**, check the section "Strategy 1: Consistent Mean Across Observations" of the README.md file under ""functional-data-regression-mip/" folder.
    - In either case, the Z design matrix is computed using the **basis expansion approach of both the X predictors and Beta basis** (using the same basis) with the approximation of **Y ~ Z*B** with **Z = W*J**
    - **Why**? Evaluatin the impact of different data generation strategies.
- **load_simulation_reiman**
    - runs "functional-data-regression-mip\src\R\generic_simulator\simulate_main_reiman.R"
    - uses "functional-data-regression-mip\src\R\generic_simulator\simulation\ cov and paper files"
    - the data generation is process it the same as load_simulation_data but in the post processing, **only the Betas are expanded** as Betas(t) = sum(y* b(t)) using the basis expansion approach while the X predictors are not. So Z is computed as Z ~ integral(X(t),b(t)). Trapezoidal approximation was used.
    - **Why**? Evaluating the impact of the basis expansion approximation of both predictors and betas over a same basis vs expansion of only the betas

- **load_simulation_robust**,
    - runs "functional-data-regression-mip\src\R\generic_simulator\simulate_main_robust.R"
    - uses "functional-data-regression-mip\src\R\generic_simulator\simulation\robust.R
    - the library library(robflreg) is used to generate scalar on function data. So only the "simulation parameters" will be used since the library generates data in a specific way.
    - **Why**? I wanted to compare the MIP model performance using a data simulation process that was well defined and which code was available and documented.

- **load_simulation_gertheiss**,
    - runs "functional-data-regression-mip\src\R\generic_simulator\simulate_main_gertheiss.R"
    - uses "functional-data-regression-mip\src\R\generic_simulator\simulation\ cov and paper files"
    - The data generation part is the same as in **load_simulation_data** but the post processing used to compute the Z matrix is the ones used by Gertheiss (**grplFlinear** function,

> https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4131701/bin/NIHMS535202-supplement-SUPP_FUNT_CODE.r).
> - **Why**? I wanted to compare the MIP model performance over the same simulation using the same data transformation pipeline used by Gertheiss

4. Setting the MIP model params

Cell Block #5 is used to define the MIP model settings.
The models are defined inside the folder **src\Julia\models** but for simplicity, I have moved only the important models inside the folder *src\Julia\ols_vs_mip_models*

- **model_name**, it's the name of the MIP model.
  - **simple_regressor.jl**, It should behave just as a linear regressor and has no constraints
  - **regressor_with_group_constraint.jl**. It adds the "group" constraint that **limits the number of active predictors** to "group_limit" (which is received as an input to the function).
  - **WHY**? I wanted to test the impact of the BIG M constraint when group_limit is set to a specific value. I tested the case when all predictors are active (and I set group_limit = #predictors) and I compared it to the OLS solution.
- **model_file_path**
- **BigM values** (they will be used only by the regressor_with_group_constraint model)
- **to_predict**, is the "group_limit" value so the number of true predictors

5. OLS solution computation

Computes the OLS solution

6. Comparing results

Comparing the real betas (beta_matrix) to the estimated ones (beta_star) and to the OLS solution (Beta_ols).
NOTICE: beta_matrix is computed from the real beta curves in the data generation process(fda package).

7. Visualize beta curves

Used to plot the curves defined by these basis coefficients.
Each plot, from left to right, references a specific predictor beta curve.
BLUE CURVE -> the real BETA CURVE of that predictor
RED CURVE -> the estimanted BETA CURVE
The second param of the plot_combined_predicted_curve function can be changed with either

beta_matrix/beta_star/beta_ols to visualize how these coefficients are fitting the expected solutions.
Obviously if we provide "beta_matrix" the red and blue curves will perfectly overlap.

8. Compare performances

(**Y,Z) and (Y_test,Z_test)** were generated in codce block #2.
At the path "src\Julia\utils\data_analysis.jl" are defined some performance metrics that are computed over the expected and real solutions (beta_matrix) and the estimated ones (beta_star).
The method **compute_metrics(Y, Z, EXPECTED_COEFF, ESTIMATED_COEFF,...)**
can be edited to accomodate the test samples or to check if the model solution is OVERFITTING the training set.