1 Aims

The medical field often uses data coming from cohorts. This kind of data is often sampled irregularly, at each patient's visit, which causes issues:

- different individuals are measured at different time
- the lapse between each measure changes
- the number of repeated measures is not the same for different individuals

Those information can be informative, (ex: dementia).
Other quirks of medical data:

- strong heterogeneity among individuals
- measurement error

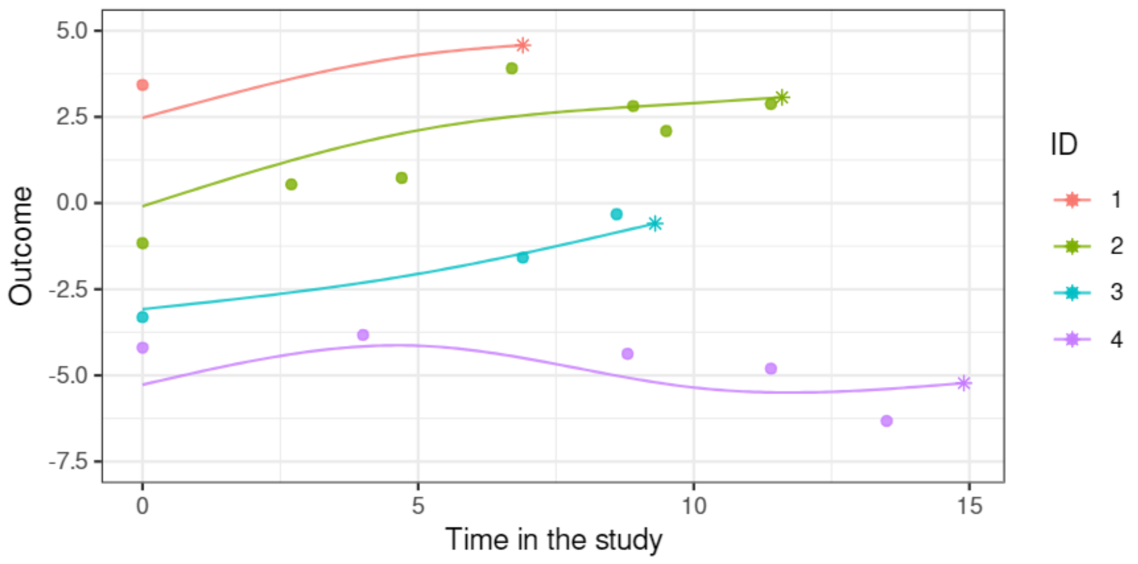

Figure 1: Irregular sampling (from C. Proust-Lima's RRI presentation)

2 Data-generating mechanism

We generate 100 training datasets and 1 test dataset with the following mechanism.

Our simulations are done for a population of $i \in [1, 500]$ individuals observed with repeated measures for $t \in [1, 25]$ periods.
We generate variables $X^*_{k,i}(t), k \in [1, 8]$ with individual-specific random effects $\alpha^k_{l,i}$ for each of them to model heterogeneity across individuals.
From those covariates we then generate 2 variables of interest $Y^*_{mixed,i}(t)$ and $Y^*_{fixed,i}(t)$, with random effects and fixed effects respectively. Finally we add Gaussian noises $\epsilon^k_i(t)$ and $\epsilon^y_i(t)$ to get our measures $X_{k,i}(t)$ and $Y_i(t)$.

The covariates are generated following the formulas:
$X^*_{1,i}(t) = \alpha^1_{0i} + \alpha^1_{1,i}t$
$X^*_{5,i}(t) = \frac{\alpha^5_{0,i}}{1+e^{-\alpha^5_{1,i}t}}$
$X^*_{2,i}(t) = \alpha^2_{0i} + \alpha^2_{1,i}log(t+1)$
$X^*_{6,i}(t) = max(0, \alpha^6_{0,i} + 0.1 * \alpha^6_{1,i} + t^2)$
$X^*_{3,i}(t) = \alpha^3_{0i} + \alpha^3_{1,i}t^2$
$X^*_{7,i}(t) = \frac{\alpha^7_{0,i}}{1+e^{-\alpha^7_{1,i}t}}$
$X^*_{4,i}(t) = \alpha^4_{0i} + \alpha^4_{1,i}e^{-0.1t}$
$X^*_{8,i}(t) = \{1 \text{ if i pair, else } 0\}$

The random effects distributions were manually selected for the variables to make sense with the time scale $t \in [1, 25]$:
$\alpha^k_{0i} \sim \mathcal{N}(\mu^k_0, \sigma^k_0), \mu^k_0 \sim \mathcal{U}(-1, 1), \sigma^k_0 = 0.5$
$\alpha^k_{1i} \sim \mathcal{N}(\mu^k_1, \sigma^k_1), \mu^k_1 \sim \mathcal{U}(-1, 1), \sigma^{k\in\{1,2,4,7\}}_1 = 0.5, \sigma^{3,6}_1 = 0.1, \sigma^{k\in\{5\}}_1 = 1$

We then add noise to those variables to mimic observation:
$\forall k \in [1, 7], X_{k,i}(t) = X^*_{k,i}(t) + \epsilon^k_i(t)$

Noises were calibrated to match corresponding variables.
$\epsilon^k_i(t) \sim \mathcal{N}(0, \sigma^k_\epsilon), \sigma^{k\in\{2,3,4\}}_\epsilon = 0.1, \sigma^{1,5,7}_\epsilon = 0.5, \sigma^6_\epsilon = 1$

Variables of interest:
$Y^*_{mixed,i} = \gamma_{0,i} + \gamma_{1,i} X^*_{2,i}(t) \cdot X^*_{5,i}(t) + \gamma_{2,i} X^*_{4,i}(t) \cdot X^*_{7,i}(t)$
$Y^*_{fixed,i} = \gamma_0 + \gamma_1 X^*_{2,i}(t) \cdot X^*_{5,i}(t) + \gamma_2 X^*_{4,i}(t) \cdot X^*_{7,i}(t)$

Observed variable: $Y_i = Y^*_i + \epsilon^y_i(t)$

Once again, we select the distribution parameters of the random effects and noise regarding Y.

$\gamma_{ki} \sim \mathcal{N}(0, \sigma^k_y), \sigma^0_y = \sigma^1_y = 0.5, \sigma^2_y = 0.05$

$\epsilon^y_i(t) \sim \mathcal{N}(0, 1)$

3 Target

This study aims to solve a regression task, using the co-variates X_k until time t to predict Y at time t, and so for each time $t \in [1, 25]$. We use two metrics to measure our models' performances: the Mean Absolute Error (MAE) and the Mean Squared Error (MSE).

For one individual i, those metrics are computed as such:
$MAE_i = \frac{1}{T} \sum_{t=1}^{T} |Y_i - \hat{Y}_i|$
$MSE_i = \frac{1}{T} \sum_{t=1}^{T} (Y_i - \hat{Y}_i)^2$

For a batch of individual, we use the mean of those metrics to evaluate performances:
$MAE = \frac{1}{N} \sum_{i=1}^{N} MAE_i$
$MSE = \frac{1}{N} \sum_{i=1}^{N} MSE_i^2$

This MSE is used as the loss function we want to minimize during the training of neural networks.

4 Methods

4.1 Oracle

The model said "Oracle" is a statistical model built knowing the exact mechanism of data generation, it represents the upper ceiling we try to get close to.

For Y generated with fixed effect, it is a simple Linear Regression on the products $X^*_{2,i}(t) \cdot X^*_{5,i}(t)$ and $X^*_{4,i}(t) \cdot X^*_{7,i}(t)$:
$\hat{Y}_{fixed,i} = \hat{\gamma}_0 + \hat{\gamma}_1 X^*_{2,i}(t) \cdot X^*_{5,i}(t) + \hat{\gamma}_2 X^*_{4,i}(t) \cdot X^*_{7,i}(t)$

For Y generated with mixed effect, we fit a Linear Mixed Effect model by giving it the exact dynamics. The predictions are subject specific so that we have:
$\hat{Y}_{mixed,i} = \hat{\gamma}_{0,i} + \hat{\gamma}_{1,i} X^*_{2,i}(t) \cdot X^*_{5,i}(t) + \hat{\gamma}_{2,i} X^*_{4,i}(t) \cdot X^*_{7,i}(t)$

4.2 Naive

The "naive" model should be a model that could be made without much information on the dynamics involved in the simulation, and that the other models should be able to beat easily.

For Y generated with fixed effects, it is a linear regression on all variables, without taking into account any interaction:
$\hat{Y}_{fixed,i} = \hat{\gamma}_0 + \sum_{k=1}^{8} \hat{\gamma}_k \cdot X^*_{k,i}$

For Y generated with random effects, it is a Linear Mixed Effect model taking powers of time as covariates:
$\hat{Y}_{mixed,i} = \sum_{k=0}^{4} \hat{\gamma}_{k,i} \cdot t^4$

4.3 RNN

Our RNN model is a classic RNN with 25 neurons per hidden layer, two hidden layers stacked, and a linear layer with 25 input neurons and one output neurons layer the predictions. The input are preprocessed with the robust scaler method, which substracts the median and divides by the interquartile range for each variable (besides X_8).

To train, we split the training dataset between training and validation subsets with a 80/20 split. This RNN is trained until convergence, ie until the loss on validation subset does not change more than 0.0005 for at least 100 epochs; or until 20000 epochs (never reached in practice). The training algorithm is the Adam method, with 0.001 for the initial learning rate and defaults values for other parameters.

For Y generated with fixed effects, we train and test the RNN for the 100 training datasets, and take the average metrics as result.

For Y generated with mixed effect, we only train and test for 10 training datasets, as the convergence is much longer.

4.4 ODE-GRU

The encoder is based on standard GRU-RNN. The update gate is a feed forward with 28 inputs neurons, one hidden layer of 25 neurons and 10 output neurons. The reset gate is also a feed-forward with the same dimensions. The new state gate is again a 3 parts feed forward, but the output layer have 10 neurons instead. Once again, inputs are preprocessed with the robust scaling method.

Instead of taking directly the output of the last gate for the next time point, the model solves an ODE the time lapse between two observations and a trained function. This function is approximated by a feed-forward network with 10 input neurons, 2 hidden layers of 25 neurons and 10 output neurons.

The decoder is a feed-forward network with 10 input neurons, two hidden layers with 25 neurons each, and 1 output neuron.

The training algorithm is the Adamax method with 0.01 as the initial learning rate and the default value for all other parameters. The training is done on one dataset for 5000 epochs for the fixed effects as well as for the random effects.

5 Performances

Performances avec Y_{fixed}								
Modèle	MAE moyenne sur le train	MSE moyenne sur le train	MAE moyenne sur le train bruité	MSE moyenne sur le train bruité	MAE moyenne sur le test	MSE moyenne sur le test	MAE moyenne sur le test bruité	MSE moyenne sur le test bruit
ODE RNN	0.190	0.066	0.809	1.021	0.229	0.135	0.836	1.109
LSTM	0.306	0.237	0.869	1.203	0.278	0.41	0.885	1.399
RNN	0.309	0.324	0.878	1.322	0.274	0.512	0.886	1.495
Oracle	0.011	0.0002	0.011	0.0002	0.798	1.000	0.801	1.002
Naif	2.340	12.022	2.494	13.791	2.519	13.021	2.670	14.812

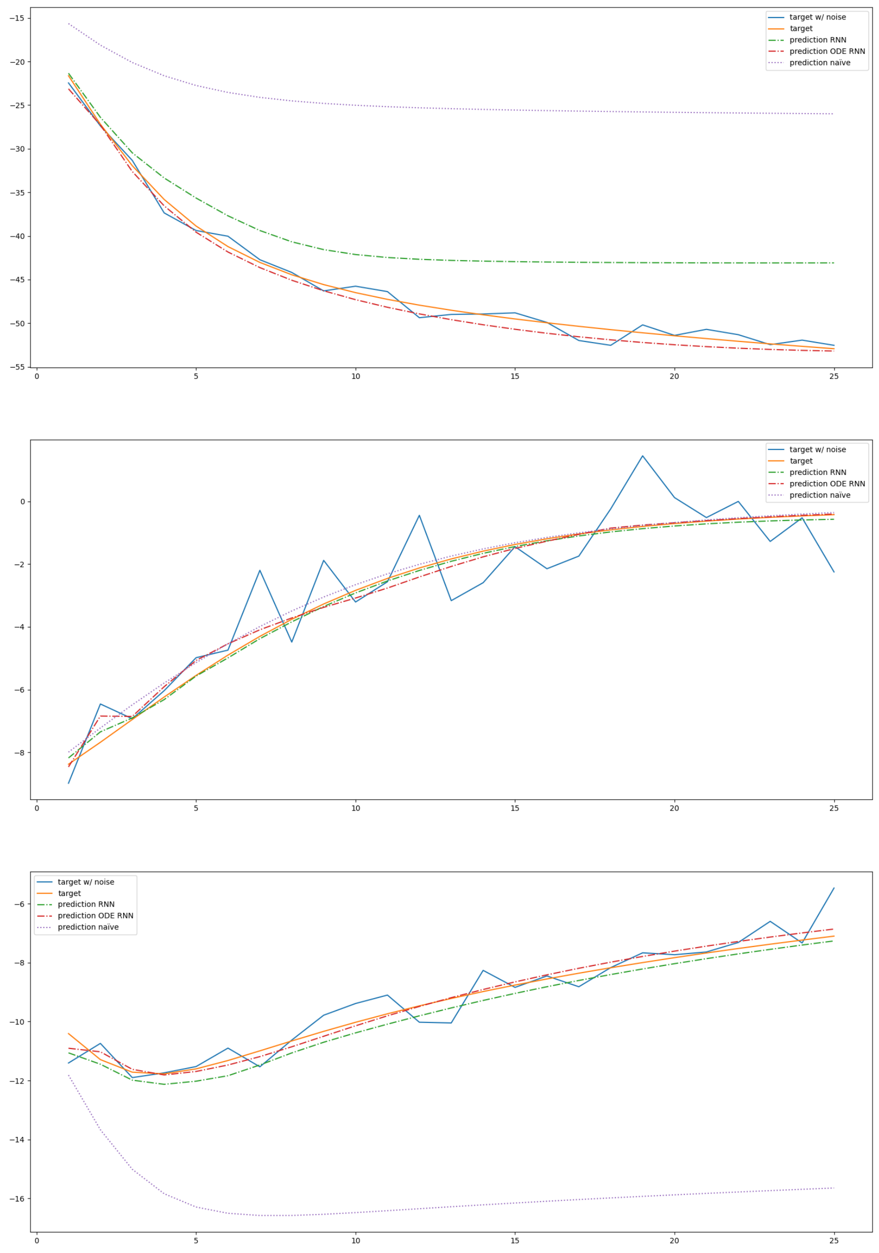

Figure 2: RNN and ODE-GRU performances on training data for Y generated with fixed effects

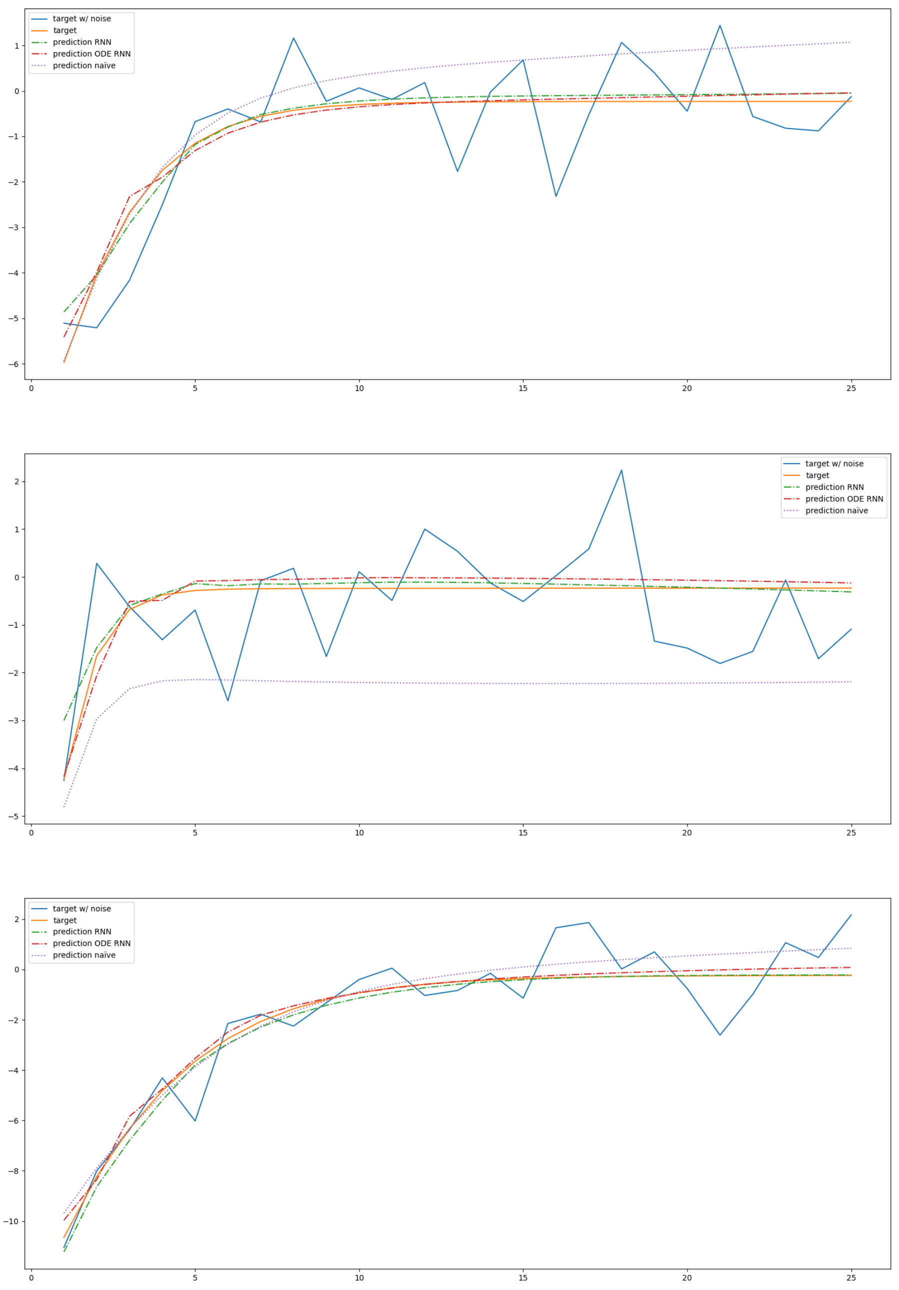

Figure 3: RNN and ODE-GRU performances on test data for Y generated with fixed effects

Performances avec Y$_{mixed}$								
Modèle	MAE moyenne sur le train	MSE moyenne sur le train	MAE moyenne sur le train bruité	MSE moyenne sur le train bruité	MAE moyenne sur le test	MSE moyenne sur le test	MAE moyenne sur le test bruité	MSE moyenne sur le test bruit
ODE RNN	1.857	10.330	1.459	3.763	9.287	425.197	9.487	425.854
RNN	3.798	95.366	3.780	98.810	9.979	472.178	10.133	472.951
LSTM	4.361	194.849	4.158	224.475	10.244	521.161	10.405	521.063
Oracle	0.201	0.078	0.200	0.081	0.764	0.925	0.758	0.914
Naif	0.279	0.167	0.282	0.182	0.769	0.947	0.765	0.942

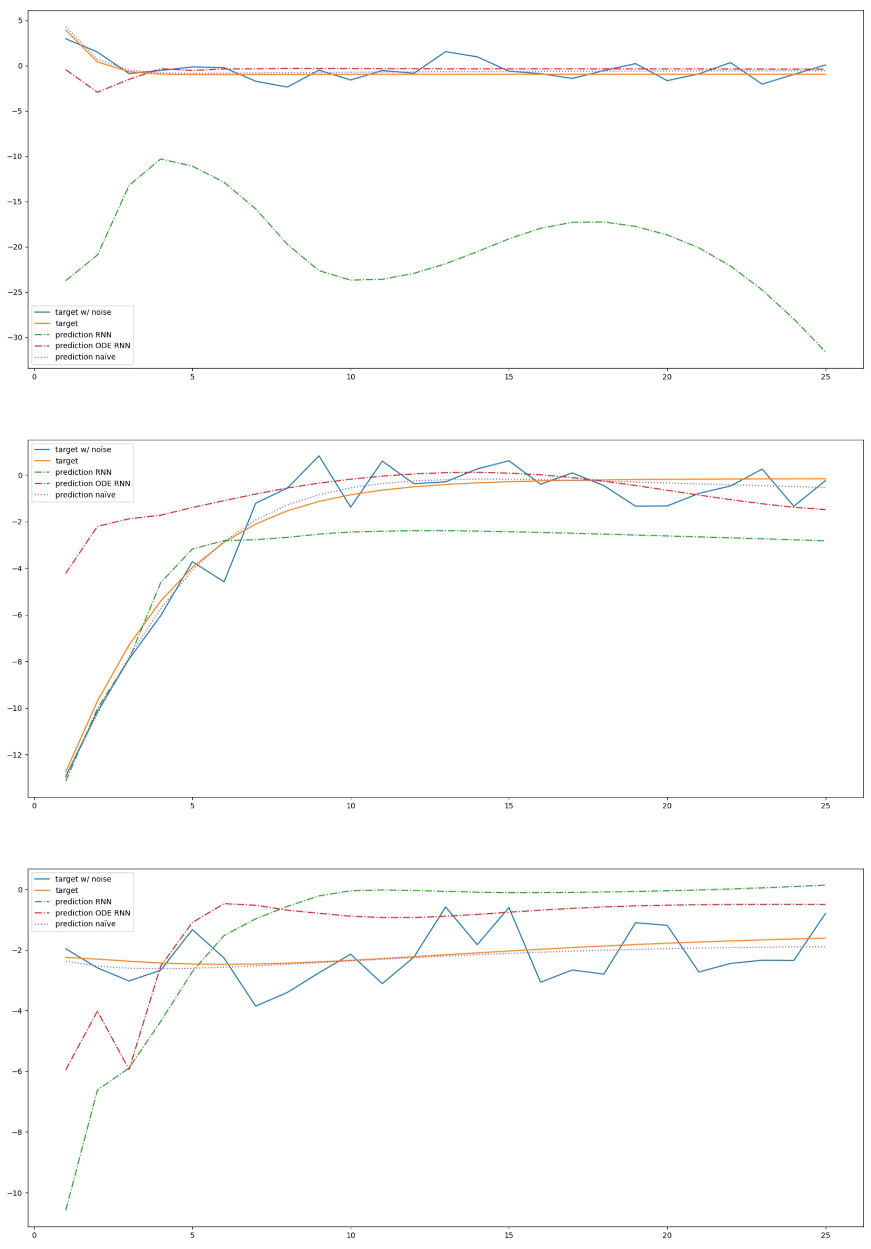

Figure 4: RNN and ODE-GRU performances on training data for Y generated with mixed effects

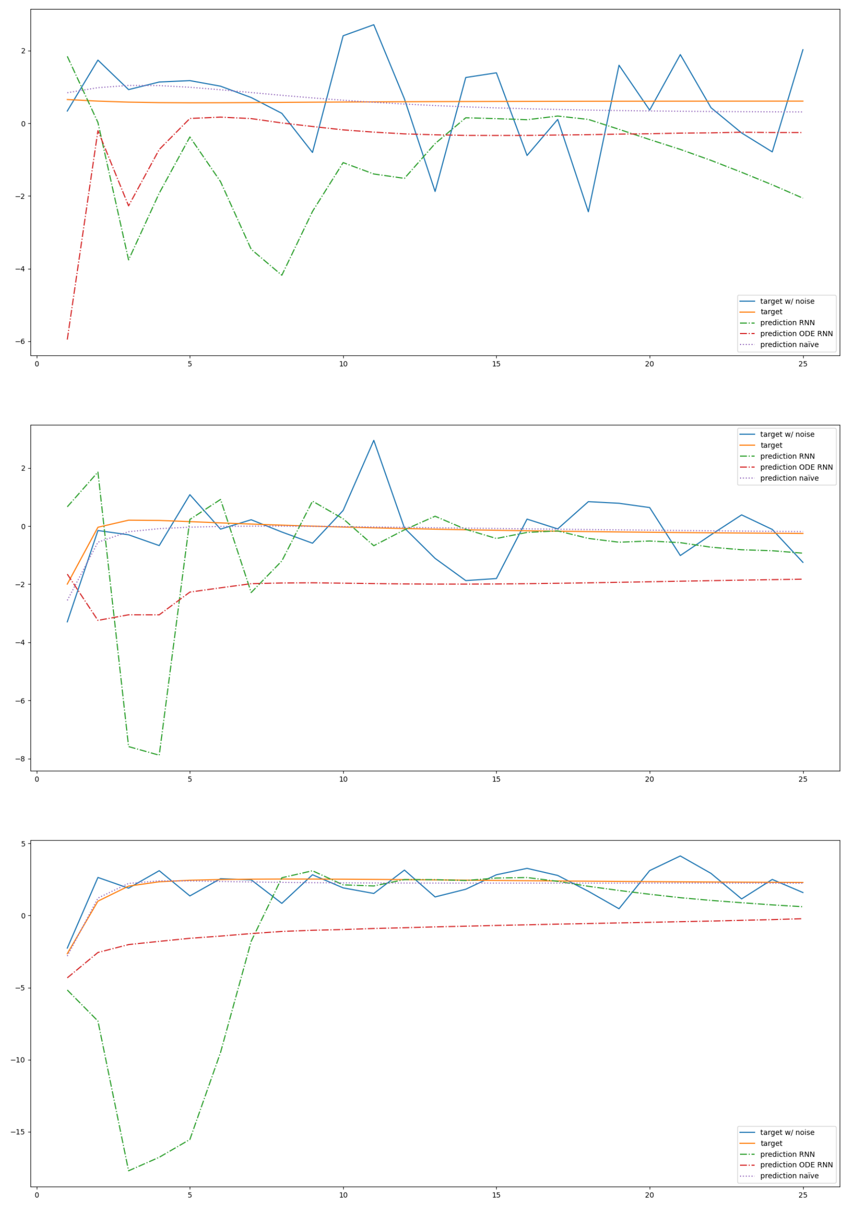

Figure 5: RNN and ODE-GRU performances on test data for Y generated with mixed effects