Summary of Jupyter Notebooks in Forward-Equivalent Simulation

1 Overview

The Jupyter notebooks serve as the execution and visualization tools for the Forward-Equivalent (FE) simulation framework. These notebooks:

- **Run simulations** and store results for analysis.
- **Compare tree statistics** between FullSim and FE models.
- **Analyze performance** by benchmarking execution times.
- **Visualize phylogenetic trees** from the simulation.

2 Notebook Summaries

2.1 `efficiency.ipynb` - Running Simulations

Purpose:

- Runs **all FE simulations** and stores output in the `data/` directory.
- Ensures simulations are executed with predefined settings.
- Saves data for further analysis in other notebooks.

Key Features:

- Reads configuration settings to determine simulation parameters.
- Calls `FEModulator` from `modulators.py` to retrieve FE-adjusted rates.
- Uses `DiscreteProcess` and `CustomProcess` to execute the simulation.

Why is this notebook important?

- It is the **starting point** for running all FE simulations.
- Without it, no simulation data would be available for analysis.

2.2 `distribution_tests.ipynb` - Comparing Tree Statistics

Purpose:

- Compares **phylogenetic tree statistics** between FullSim and FE.
- Generates **histograms** to check if FE and FullSim produce statistically similar results.
- Uses **statistical tests** to validate that FE maintains accuracy.

Key Features:

- Loads tree simulation data from `efficiency.ipynb`.
- Computes **branch length distributions** and **coalescent times**.
- Uses `compute_tree_stats()` from `utils.py` for statistical comparison.

Why is this notebook important?

- Ensures that FE simulation **statistically matches** FullSim.
- Demonstrates that **FE maintains the same phylogenetic properties** while being more efficient.

2.3 `timing_plots.ipynb` - Analyzing Performance

Purpose:

- Compares **execution times** of FullSim vs. FE.
- Shows how FE achieves a **2-6× speed-up**.
- Evaluates performance across different simulation settings.

Key Features:

- Runs multiple timing tests with different parameters.
- Plots graphs showing time complexity improvements.
- Uses built-in Python timing functions and benchmarking tools.

Why is this notebook important?

- Proves that FE **reduces computational cost**.
- Validates the efficiency claims made in the paper.

2.4 `draw_tree.ipynb` - Visualizing Phylogenetic Trees

Purpose:

- Plots **phylogenetic trees** generated by the simulation.
- Provides a **graphical representation** of the evolutionary process.
- Helps in visualizing differences between FullSim and FE outputs.

Key Features:

- Loads tree data from `efficiency.ipynb`.
- Uses the `ete3` library for visualization.
- Customizes tree structure, branch lengths, and annotations.

Why is this notebook important?

- Provides an intuitive way to **understand simulation outputs**.
- Helps in validating that FE trees look similar to FullSim trees.

3 How These Notebooks Work Together

1. `efficiency.ipynb` runs the simulations and saves data.
2. `distribution_tests.ipynb` analyzes and compares tree statistics.
3. `timing_plots.ipynb` benchmarks performance improvements.
4. `draw_tree.ipynb` provides visual validation of the results.

4 Key Takeaways

- The notebooks collectively validate that **FE maintains statistical equivalence to FullSim**.
- They show that **FE is computationally efficient** (2-6× faster).
- The tree visualization confirms that **FE produces realistic phylogenetic trees**.

5 Potential Questions from the Professor

1. What is the most important notebook? *Answer:* `efficiency.ipynb` because it runs all simulations.

2. How do these notebooks validate FE's accuracy? *Answer:* `distribution_tests.ipynb` compares tree statistics to ensure FE and FullSim are equivalent.

3. How does `timing_plots.ipynb` demonstrate efficiency? *Answer:* It shows that FE simulation is 2-6× faster than FullSim through execution time analysis.