A Appendix

Please refer to https://github.com/cavalab/srbench/ for the most up-to-date guide to SRBench.

A.1 Running the Benchmark

The README in our Github repository includes the full set of commands to reproduce the benchmark experiment, which are summarized here. Experiments are launched from the `experiments/` folder via the script `analyze.py`. The script can be configured to run the experiment in parallel locally, on an LSF job scheduler, or on a SLURM job scheduler. To see the full set of options, run `python analyze.py -h`.

After installing and configuring the conda environment, the complete black-box experiment can be started via the command:

```
python analyze.py /path/to/pmlb/datasets -n_trials 10 -results
../results -time_limit 48:00
```

Similarly, the ground-truth regression experiment for Strogatz datasets and a target noise of 0.0 are run by the command:

```
python analyze.py -results ../results_sym_data -target_noise
0.0 "/path/to/pmlb/datasets/strogatz*" -sym_data -n_trials 10
-time_limit 9:00 -tuned
```

A.2 Contributing a Method

A living version of the method contribution instructions are described in the Contribution Guide. To illustrate the simplicity of contributing a method, Figure 4 shows the script submitted for Bayesian Symbolic Regression [16]. In addition to the code snippet, authors may either add their code package to the conda/pip environment, or provide an install script. When a pull request is issued by a contributor, new methods and installs are automatically tested on a minimal version of the benchmark. Once the tests pass and the method is approved by the benchmark maintainers, the contribution becomes part of the resource and can be tested via the commands above.

A.3 Additional Background and Motivation

Eureqa Eureqa is a commercial GP-based SR software that was acquired by DataRobot in 2017[8]. Due to its closed-source nature and incorporation into the DataRobot platform, it is impossible to benchmark its performance while controlling for important experimental variables such as number of evaluations, space and time limits, population size, and so forth. However, the novel algorithmic aspects of Eureqa are rooted in a number of ablation studies [38, 51, 77] that we summarize here. First is its use of directed acyclic graphs for representing equations in lieu of trees, which resulted in more space-efficient model encoding relative to trees, without a significant difference in accuracy [77]. The most significant improvement over traditional tournament-based selection is Eureqa's use of age-fitness Pareto optimization (AFP), a method in which random restarts are incorporated each generation as new offspring, and are protected from competing with older, more fit equations by including age as an objective to be minimized [38]. Eureqa also includes the co-evolution of fitness

[8] https://www.datarobot.com/nutonian/

```python
# method: Bayesian Symbolic Regression
# contributor: Ying Jin
# source: https://github.com/ying531/MCMC-SymReg
from bsr.bsr_class import BSR

hyper_params = []
for val, itrNum in zip([100,500,1000],[5000,1000,500]):
    for treeNum in [3,6]:
        hyper_params.append(
                    {'treeNum': [treeNum],
                     'itrNum': [itrNum],
                     'val': [val],
                    })
# default estimator
est = BSR(val=100, itrNum=5000, treeNum=3, alpha1=0.4, alpha2=0.4,
          beta=-1, disp=False, max_time=2*60*60)

def complexity(est):
"""returns final model complexity"""
    return est.complexity()


def model(est):
"""returns final model as string"""
    return est.model()
```

Figure 4: An example code contribution, defining the estimator, its hyperparameters, and functions to return the complexity and symbolic model.

predictors, in which fitness assignment is sped up by optimizing a second population of training sample indices that best distinguish between equations in the population [51]. Unfortunately we cannot guarantee that Eureqa currently uses any of these reported algorithms for SR, due to its closed-source nature. We chose instead to benchmark known algorithms (AFP, AFP_FE) with open-source implementations, hoping that the resulting study's conclusions may better inform future methods development. We note that AFP has been outperformed by a number of other optimization methods in controlled studies since its release (e.g., [27, 28]).

Constant optimization in Genetic Programming One of the clearest improvements over Koza-style GP has been the adoption of local search methods to handle constant optimization distinctly from evolutionary learning. Regarding the optimization of constants in GP, several reasons can explain why backpropagation and gradient descent can be considered to be relatively under-used in GP (compared to, e.g., evolutionary neural architecture search). For example, early works often ignored the use of feature standardization (e.g., by z-scoring), the lack of which can harm gradient propagation [78]. Next to this, GP relies on crafting compositions out of a multitude of operations, some of which are prone to cause vanishing or exploding gradients. Last but not least, to the best of our knowledge, the field lacks a comprehensive study that provides guidelines for the appropriate hyperparameters for constant optimization (learning rate schedule, iterations, batch size, etc.), and how to effectively balance parameter learning with the evolutionary process.

A.4 Additional Dataset Information

All datasets, including metadata, are available from PMLB. Each dataset is stored using Git Large File Storage and PMLB is planned for long-term maintenance. PMLB is available under an MIT

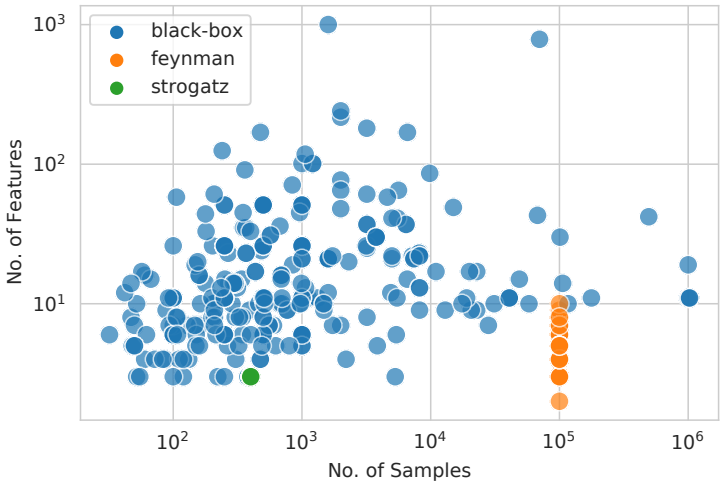

Figure 5: Distribution of dataset sizes in PMLB.

license, and is described in detail in Romano et al. [36]. The authors bear all responsibility in case of violation of rights.

Dataset Properties The distribution of dataset sizes by samples and features are shown in Fig. 5. Datasets vary in size from tens to millions of samples, and up to thousands of features. The datasets can be navigated and inspected in the repository documentation.

Ethical Considerations and Intended Uses PMLB is intended to be used as a framework for benchmarking ML and SR algorithms and as a resource for investigating the structure of datasets. This paper does not contribute new datasets, but rather collates and standardizes datasets that were already publicly available. In that regard, we do not foresee SRBench as creating additional ethical issues around their use. Nevertheless, it is worth noting that PMLB contains well-known, real-world datasets from UCI and OpenML for which ethical considerations are important, such as the USCrime dataset. Whereas we would view the risk of harm arising specifically from this dataset to be low (the data is from 1960), it is exemplary of a task for which algorithmic decision making could exacerbate existing biases in the criminal justice system. As such it is used as a benchmark in a number of papers in the ML fairness literature (e.g. [79, 80]). None of the datasets herein contain personally identifiable information.

Feynman datasets The Feynman benchmarks were sourced from the Feynman Symbolic Regression Database. We standardized the Feynman and Bonus equations to PMLB format and included metadata detailing the model form and the units for each variable. We used the version of the equations that were not simplified by dimensional analysis. Udrescu and Tegmark [18] describe each dataset as containing 10^5 rows, but each actually contains 10^6. Given this discrepancy and after noting that sub-sampling did not significantly change the correlation structure of any of the problems, each dataset was down-sampled from 1 million samples to 100,000 to lower the computational burden. We also observed that Eqn. II.11.17 was missing from the database. Finally, we excluded three datasets from our analysis that contained $\arcsin$ and $\arccos$ functions, as these were not implemented in the majority of SR algorithms we tested.

Strogatz datasets The Strogatz datasets were sourced from the ODE-Strogatz repository [67]. Each dataset is one state of a 2-state system of first-order, ordinary differential equations (ODEs). The goal of each problem is to predict rate of change of the state given the current two states on which it depends. Each represents natural processes that exhibit chaos and non-linear dynamics. The problems were originally adapted from [66] by Schmidt [25]. In order to simulate their behavior, initial conditions were chosen within stable basins of attraction. Each system was simulated using Simulink, and the simulation code is available in the repository above. The equations for each of these datasets are shown in Table 3.

Table 3: The Strogatz ODE problems.

Name	Target
Bacterial Respiration	$\dot{x} = 20 - x - \frac{x \cdot y}{1+0.5 \cdot x^2}$ $\dot{y} = 10 - \frac{x \cdot y}{1+0.5 \cdot x^2}$
Bar Magnets	$\dot{\theta} = 0.5 \cdot \sin(\theta - \phi) - \sin(\theta)$ $\dot{\phi} = 0.5 \cdot \sin(\phi - \theta) - \sin(\phi)$
Glider	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$ $\dot{\theta} = v - \cos(\theta)/v$
Lotka-Volterra interspecies dynamics	$\dot{x} = 3 \cdot x - 2 \cdot x \cdot y - x^2$ $\dot{y} = 2 \cdot y - x \cdot y - y^2$
Predator Prey	$\dot{x} = x \cdot \left(4 - x - \frac{y}{1+x}\right)$ $\dot{y} = y \cdot \left(\frac{x}{1+x} - 0.075 \cdot y\right)$
Shear Flow	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$ $\dot{\phi} = \left(\cos^2(\phi) + 0.1 \cdot \sin^2(\phi)\right) \cdot sin(\theta)$
van der Pol oscillator	$\dot{x} = 10 \cdot \left(y - \frac{1}{3} \cdot (x^3 - x)\right)$ $\dot{y} = -\frac{1}{10} \cdot x$

Adding Noise White gaussian noise was added to the target as a fraction of the signal root mean square value. In other words, for target noise level γ,

$$y_{noise} = y + \epsilon, \ \epsilon \sim \mathcal{N}\left(0, \gamma\sqrt{\frac{1}{N}\sum y_i^2}\right)$$

A.5 Additional Experiment Details

Experiments were run in a heterogeneous cluster computing environment composed of hosts with 24-28 core Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz processors and 250 GB of RAM. Jobs consisted of the training of each method on a single dataset for a fixed random seed. Each job received one CPU core and up to 16GB of RAM, and was time-limited as shown in Table 2. For the ground-truth problems, the final models from each method were given an additional hour of computing time with 8GB of RAM to be simplified with sympy and assessed by the solution criteria (see Def. 4.1). For the black-box problems, if a job was killed due to the time limit, we re-ran the experiment without hyperparameter tuning, thereby only requiring a single training iteration to complete within 48 hours. To ease the computational burden for large datasets, training data exceeding 10,000 samples was randomly subset to 10,000 rows; test set predictions were still evaluated over the entire test fold.

The hyperparameter settings for each method are shown in Tables 4-6. Each SR method was tuned from a set of six hyperparameter combinations. The most common parameter setting chosen during the black-box regression experiments was then used as the “tuned" version of each algorithm for

Table 4: ML methods and the hyperparameter spaces used in tuning.

Method	Hyperparameters
AdaBoost	{'learning_rate': (0.01, 0.1, 1.0, 10.0), 'n_estimators': (10, 100, 1000)}
KernelRidge	{'kernel': ('linear', 'poly', 'rbf', 'sigmoid'), 'alpha': (0.0001, 0.01, 0.1, 1), 'gamma': (0.01, 0.1, 1, 10)}
LassoLars	{'alpha': (0.0001, 0.001, 0.01, 0.1, 1)}
LGBM	{'n_estimators': (10, 50, 100, 250, 500, 1000), 'learning_rate': (0.0001, 0.01, 0.05, 0.1, 0.2), 'subsample': (0.5, 0.75, 1), 'boosting_type': ('gbdt', 'dart', 'goss')}
LinearRegression	{'fit_intercept': (True,)}
MLP	{'activation': ('logistic', 'tanh', 'relu'), 'solver': ('lbfgs', 'adam', 'sgd'), 'learning_rate': ('constant', 'invscaling', 'adaptive')}
RandomForest	{'n_estimators': (10, 100, 1000), 'min_weight_fraction_leaf': (0.0, 0.25, 0.5), 'max_features': ('sqrt', 'log2', None)}
SGD	{'alpha': (1e-06, 0.0001, 0.01, 1), 'penalty': ('l2', 'l1', 'elasticnet')}
XGB	{'n_estimators': (10, 50, 100, 250, 500, 1000), 'learning_rate': (0.0001, 0.01, 0.05, 0.1, 0.2), 'gamma': (0, 0.1, 0.2, 0.3, 0.4), 'subsample': (0.5, 0.75, 1)}

the ground-truth problems, with updates to 1) include any mathematical operators needed for those problems and 2) double the evaluation budget.

A.6 Additional Results

A.6.1 Subgroup analysis of black-box regression results

Many of the black-box problems for regression in PMLB were originally sourced from OpenML. A few authors have noted that several of these datasets are sourced from Friedman [81]'s synthetic benchmarks. These datasets are generated by non-linear functions that vary in degree of noise, variable interactions, variable importance, and degree of non-linearity. Due to their number, they may have an out-sized effect on results reporting in PMLB. In Fig. 6, we separate out results on this set of problems relative to the rest of PMLB. We do find that, relative to the rest of PMLB, the results on the Friedman datasets distinguish top-ranked methods more strongly than among the rest of the benchmark, on which performance between top-performing methods is more similar. In general, although we do see methods rankings change somewhat when looking at specific data groupings, we do not observe large differences. An exception is Kernel ridge regression, which performs poorly on the Friedman datasets but very well on the rest of PMLB. We recommend that future revisions to PMLB expand the dataset collection to minimize the effect of any one source of data, and include subgroup analysis to identify which types of problems are best solved by specific methods.

To get a better sense of the performance variability across methods and datasets, method rankings on each dataset are bi-clustered and visualized in Fig. 7. Methods that perform most similarly across the benchmark are placed adjacent to each other, and likewise datasets that induce similar method rankings are grouped. We note some expected groupings first: AFP and AFP_FE, which differ only in fitness estimation, and FEAT and EPLEX, which use the same selection method, perform similarly. We also observe clustering among the Friedman datasets (names beginning with "fri_"), and again note stark differences between methods that perform well on these problems, e.g. Operon, SBP-GP, and FEAT, and those that do not, e.g. MLP. This view of the results also reveals a cluster of SR methods (AFP, AFP_FE, DSR, gplearn) that perform well on a subset of real-world problems (analcatdata_neavote_523 - vineyard_192) for which linear models also perform well. Interestingly, for that problem subset, Operon's performance is mediocre relative to its strong performance on other datasets. We also note with surprise that DSR and gplearn exhibit performance similarity on par with

Table 5: Part 1: SR methods and the hyperparameter spaces used in tuning on the black-box regression problems.

Method	Hyperparameters
AFP	{'popsize': 100, 'g': 2500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 100, 'g': 2500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt', 'sin', 'cos']} {'popsize': 500, 'g': 500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 500, 'g': 500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt', 'sin', 'cos']} {'popsize': 1000, 'g': 250, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 1000, 'g': 250, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt', 'sin', 'cos']}
AIFeynman	{'BF_try_time': 60, 'NN_epochs': 4000, 'BF_ops_file_type'="10ops.txt"} {'BF_try_time': 60, 'NN_epochs': 4000, 'BF_ops_file_type'="14ops.txt"} {'BF_try_time': 60, 'NN_epochs': 4000, 'BF_ops_file_type'="19ops.txt"} {'BF_try_time': 600, 'NN_epochs': 400, 'BF_ops_file_type'="10ops.txt"} {'BF_try_time': 600, 'NN_epochs': 400, 'BF_ops_file_type'="14ops.txt"} {'BF_try_time': 600, 'NN_epochs': 400, 'BF_ops_file_type'="19ops.txt"}
BSR	{'treeNum': 6, 'itrNum': 500, 'val': 1000} {'treeNum': 6, 'itrNum': 1000, 'val': 500} {'treeNum': 3, 'itrNum': 500, 'val': 1000} {'treeNum': 6, 'itrNum': 5000, 'val': 100} {'treeNum': 3, 'itrNum': 5000, 'val': 100} {'treeNum': 3, 'itrNum': 1000, 'val': 500}
DSR	{'batch_size': array([10, 100, 1000, 10000, 100000])}
EPLEX	{'popsize': 1000, 'g': 250, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 500, 'g': 500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 1000, 'g': 250, 'op_list': ['n', 'v', '+', '-', '*', '/', 'sin', 'cos', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 100, 'g': 2500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 100, 'g': 2500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'sin', 'cos', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 500, 'g': 500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'sin', 'cos', 'exp', 'log', '2', '3', 'sqrt']}
FEAT	{'pop_size': 100, 'gens': 2500, 'lr': 0.1} {'pop_size': 100, 'gens': 2500, 'lr': 0.3} {'pop_size': 500, 'gens': 500, 'lr': 0.1} {'pop_size': 500, 'gens': 500, 'lr': 0.3} {'pop_size': 1000, 'gens': 250, 'lr': 0.1} {'pop_size': 1000, 'gens': 250, 'lr': 0.3}
FE_AFP	{'popsize': 1000, 'g': 250, 'op_list': ['n', 'v', '+', '-', '*', '/', 'sin', 'cos', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 500, 'g': 500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 1000, 'g': 250, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 100, 'g': 2500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 100, 'g': 2500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'sin', 'cos', 'exp', 'log', '2', '3', 'sqrt']} {'popsize': 500, 'g': 500, 'op_list': ['n', 'v', '+', '-', '*', '/', 'sin', 'cos', 'exp', 'log', '2', '3', 'sqrt']}

AFP/AFP_FE, and are the next most similar-performing methods (note the dendrogram connecting these columns).

A.6.2 Extended analysis of ground-truth regression results

As noted in Sec. 6, despite Operon's good performance on black-box regression, it finds few models with symbolic equivalence. An alternative (and weaker) notion of solution is based on test set accuracy, which we show in Fig. 8; by this metric, the relative method performance corresponds more closely to that seen for black-box regression. We also note that methods that impose structural assumptions on the model (BSR, FEAT, ITEA, FFX) are worse at finding symbolic solutions, most of which do not match those assumptions (e.g. most processes in Table 3).

Table 6: Part 2: SR methods and the hyperparameter spaces used in tuning on the black-box regression problems.

Method	Hyperparameters
GPGOMEA	{'initmaxtreeheight': (4,), 'functions': ('+_-_*_p/_plog_sqrt_sin_cos',), 'popsize': (1000,), 'linearscaling': (True,)} {'initmaxtreeheight': (6,), 'functions': ('+_-_*_p/_plog_sqrt_sin_cos',), 'popsize': (1000,), 'linearscaling': (True,)} {'initmaxtreeheight': (4,), 'functions': ('+_-_*_p/',), 'popsize': (1000,), 'linearscaling': (True,)} {'initmaxtreeheight': (6,), 'functions': ('+_-_*_p/',), 'popsize': (1000,), 'linearscaling': (True,)} {'initmaxtreeheight': (4,), 'functions': ('+_-_*_p/_plog_sqrt_sin_cos',), 'popsize': (1000,), 'linearscaling': (False,)} {'initmaxtreeheight': (6,), 'functions': ('+_-_*_p/_plog_sqrt_sin_cos',), 'popsize': (1000,), 'linearscaling': (False,)}
ITEA	{'exponents': ((-5, 5),), 'termlimit': ((2, 15),), 'transfunctions': ('[Id, Tanh, Sin, Cos, Log, Exp, SqrtAbs]',)} {'exponents': ((-5, 5),), 'termlimit': ((2, 5),), 'transfunctions': ('[Id, Tanh, Sin, Cos, Log, Exp, SqrtAbs]',)} {'exponents': ((-5, 5),), 'termlimit': ((2, 15),), 'transfunctions': ('[Id, Sin]',)} {'exponents': ((0, 5),), 'termlimit': ((2, 15),), 'transfunctions': ('[Id, Sin]',)} {'exponents': ((0, 5),), 'termlimit': ((2, 5),), 'transfunctions': ('[Id, Sin]',)} {'exponents': ((0, 5),), 'termlimit': ((2, 15),), 'transfunctions': ('[Id, Tanh, Sin, Cos, Log, Exp, SqrtAbs]',)}
MRGP	{'popsize': 1000, 'g': 250, 'rt_cross': 0.8, 'rt_mut': 0.2} {'popsize': 100, 'g': 2500, 'rt_cross': 0.2, 'rt_mut': 0.8} {'popsize': 100, 'g': 2500, 'rt_cross': 0.8, 'rt_mut': 0.2} {'popsize': 500, 'g': 500, 'rt_cross': 0.2, 'rt_mut': 0.8} {'popsize': 500, 'g': 500, 'rt_cross': 0.8, 'rt_mut': 0.2} {'popsize': 1000, 'g': 250, 'rt_cross': 0.2, 'rt_mut': 0.8}
Operon	{'population_size': (500,), 'pool_size': (500,), 'max_length': (50,), 'allowed_symbols': ('add,mul,aq,constant,variable',), 'local_iterations': (5,), 'offspring_generator': ('basic',), 'tournament_size': (5,), 'reinserter': ('keep-best',), 'max_evaluations': (500000,)} {'population_size': (500,), 'pool_size': (500,), 'max_length': (25,), 'allowed_symbols': ('add,mul,aq,exp,log,sin,tanh,constant,variable',), 'local_iterations': (5,), 'offspring_generator': ('basic',), 'tournament_size': (5,), 'reinserter': ('keep-best',), 'max_evaluations': (500000,)} {'population_size': (500,), 'pool_size': (500,), 'max_length': (25,), 'allowed_symbols': ('add,mul,aq,constant,variable',), 'local_iterations': (5,), 'offspring_generator': ('basic',), 'tournament_size': (5,), 'reinserter': ('keep-best',), 'max_evaluations': (500000,)} {'population_size': (100,), 'pool_size': (100,), 'max_length': (50,), 'allowed_symbols': ('add,mul,aq,constant,variable',), 'local_iterations': (5,), 'offspring_generator': ('basic',), 'tournament_size': (3,), 'reinserter': ('keep-best',), 'max_evaluations': (500000,)} {'population_size': (100,), 'pool_size': (100,), 'max_length': (25,), 'allowed_symbols': ('add,mul,aq,exp,log,sin,tanh,constant,variable',), 'local_iterations': (5,), 'offspring_generator': ('basic',), 'tournament_size': (3,), 'reinserter': ('keep-best',), 'max_evaluations': (500000,)} {'population_size': (100,), 'pool_size': (100,), 'max_length': (25,), 'allowed_symbols': ('add,mul,aq,constant,variable',), 'local_iterations': (5,), 'offspring_generator': ('basic',), 'tournament_size': (3,), 'reinserter': ('keep-best',), 'max_evaluations': (500000,)}
gplearn	{'population_size': 100, 'generations': 5000, 'function_set': ('add', 'sub', 'mul', 'div', 'log', 'sqrt')} {'population_size': 1000, 'generations': 500, 'function_set': ('add', 'sub', 'mul', 'div', 'log', 'sqrt')} {'population_size': 1000, 'generations': 500, 'function_set': ('add', 'sub', 'mul', 'div', 'log', 'sqrt', 'sin', 'cos')} {'population_size': 500, 'generations': 1000, 'function_set': ('add', 'sub', 'mul', 'div', 'log', 'sqrt')} {'population_size': 500, 'generations': 1000, 'function_set': ('add', 'sub', 'mul', 'div', 'log', 'sqrt', 'sin', 'cos')} {'population_size': 100, 'generations': 5000, 'function_set': ('add', 'sub', 'mul', 'div', 'log', 'sqrt', 'sin', 'cos')}
sembackpropgp	{'popsize': (1000,), 'functions': ('+_-_*_aq_plog_sin_cos',), 'linearscaling': (False,), 'sbrdo': (0.9,), 'submut': (0.1,), 'tournament': (4,), 'maxsize': (250,), 'sblibtype': ('p_6_9999',)} {'popsize': (1000,), 'functions': ('+_-_*_aq_plog_sin_cos',), 'linearscaling': (True,), 'sbrdo': (0.9,), 'submut': (0.1,), 'tournament': (4,), 'maxsize': (1000,)} {'popsize': (1000,), 'functions': ('+_-_*_aq_plog_sin_cos',), 'linearscaling': (True,), 'sbrdo': (0.9,), 'submut': (0.1,), 'tournament': (8,), 'maxsize': (1000,)} {'popsize': (1000,), 'functions': ('+_-_*_aq_plog_sin_cos',), 'linearscaling': (True,), 'sbrdo': (0.9,), 'submut': (0.1,), 'tournament': (4,), 'maxsize': (5000,)} {'popsize': (1000,), 'functions': ('+_-_*_aq_plog_sin_cos',), 'linearscaling': (True,), 'sbrdo': (0.9,), 'submut': (0.1,), 'tournament': (8,), 'maxsize': (5000,)} {'popsize': (10000,), 'functions': ('+_-_*_aq_plog_sin_cos',), 'linearscaling': (False,), 'sbrdo': (0.9,), 'submut': (0.1,), 'tournament': (8,), 'maxsize': (250,), 'sblibtype': ('p_6_9999',)}

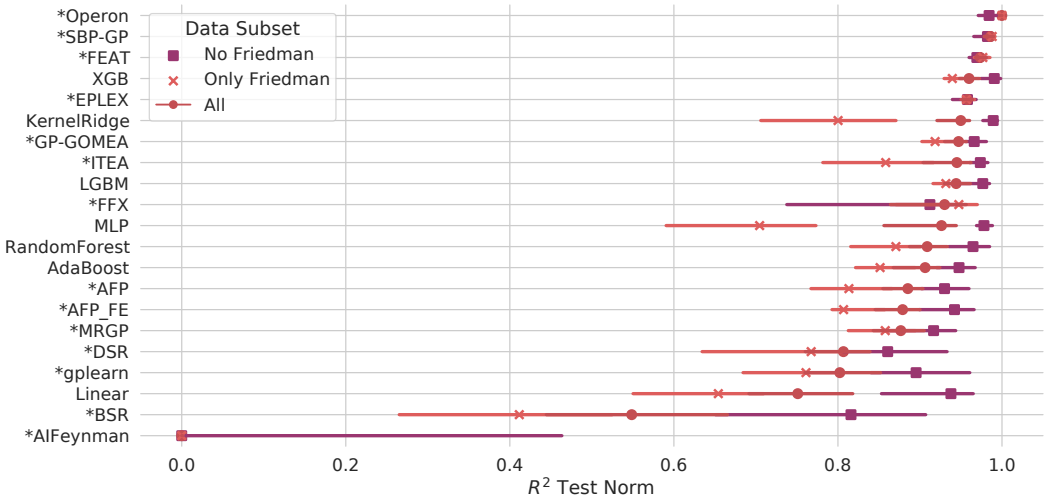

Figure 6: Comparison of normalized R^2 test scores on all black-box datasets, just the Friedman datatasets, and just the non-Friedman datasets.

A.7 Statistical Tests

Figures 9-11 give summary significance levels of pairwise tests of significance between estimators on the black-box and ground-truth problems. All pair-wise statistical tests are Wilcoxon signed-rank tests. A Bonferroni correction was applied, yielding the α levels given in each. This methodology for assessing statistical significance is based on the recommendations of Demšar [82] for comparing multiple estimators over many datasets. These figures are intended to complement Figures 1-3 in which effect sizes are shown.

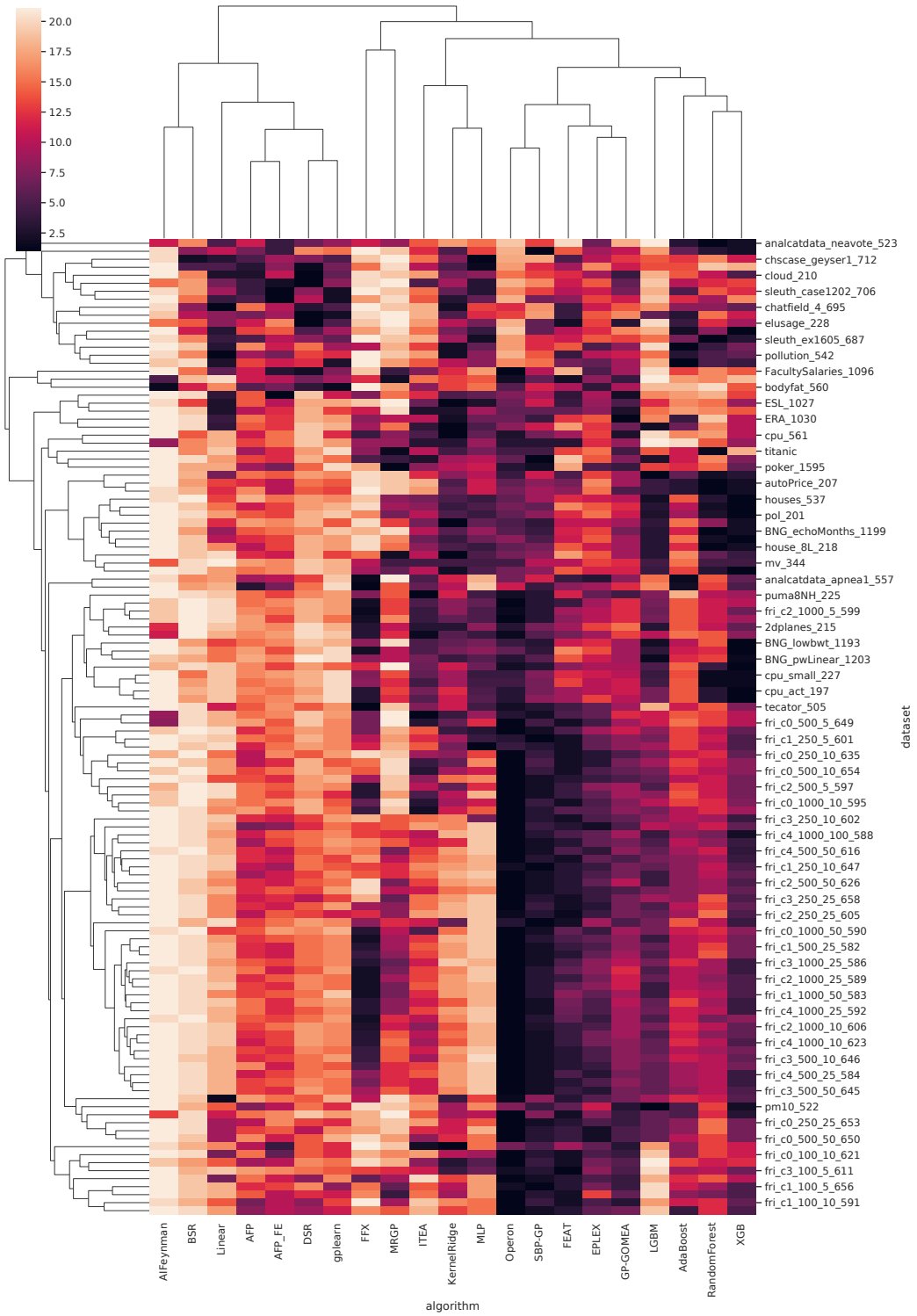

Figure 7: Rankings of methods by R^2 test score on the black-box problems (lower/darker is better). Results are bi-clustered by SR method (columns) and dataset (rows). Darker cells indicate that a method performs well on that dataset relative to its competitors. Note only a subset of the datasets are labelled due to space constraints.

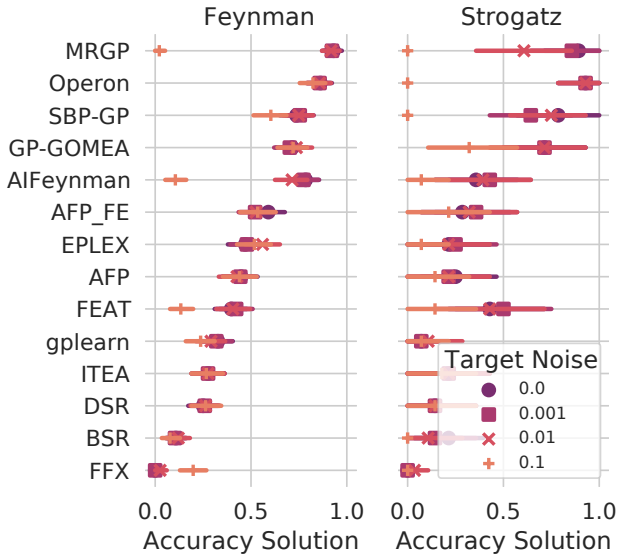

Figure 8: Subset comparison of "Accuracy Solutions", i.e. models with R^2>0.999 on the Feynman and Strogatz problems, differentiated by noise level.

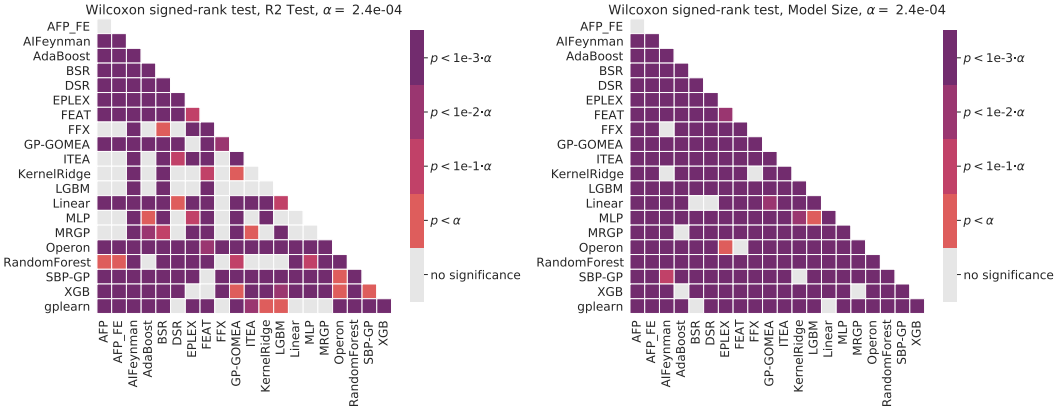

Figure 9: Pairwise statistical comparisons on the black-box regression problems. Wilcoxon signed-rank tests are used with a Bonferonni correction on α for multiple comparisons. (Left) R^2 test scores, (Right) model size.

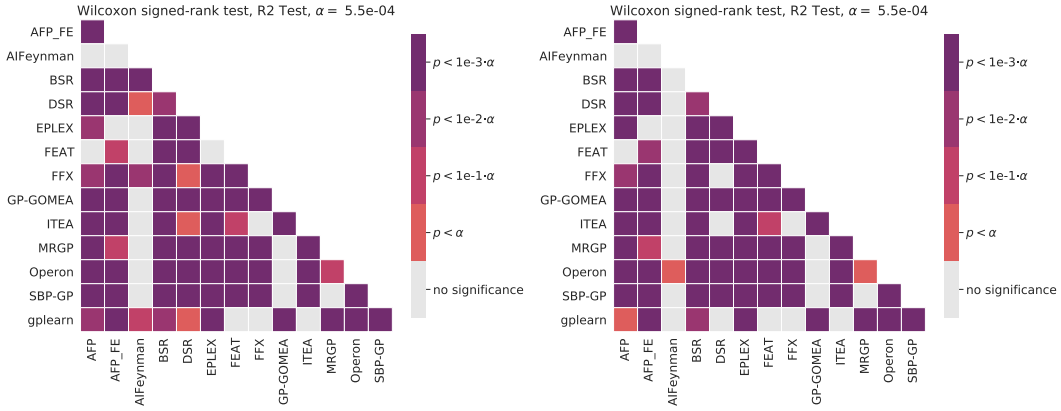

Figure 10: Pairwise statistical comparisons of R^2 test scores on the ground-truth regression problems. We report Wilcoxon signed-rank tests with a Bonferonni correction on α for multiple comparisons. (Left) target noise of 0, (Right) target noise of 0.01.

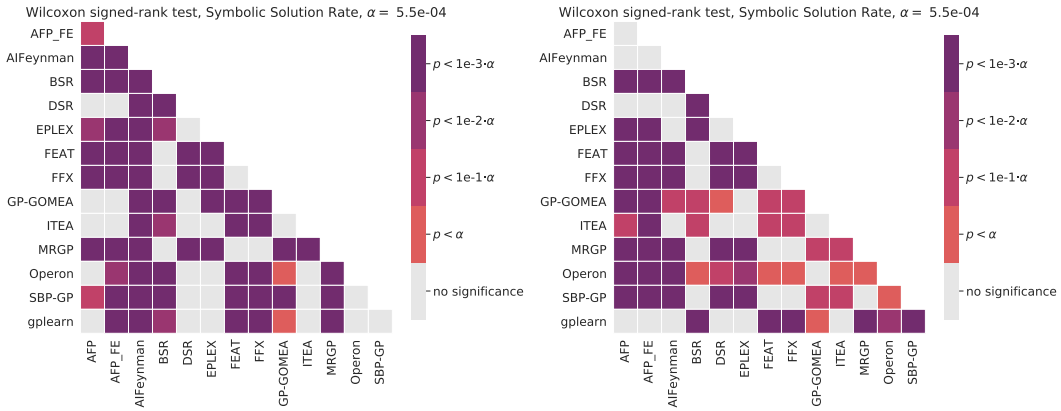

Figure 11: Pairwise statistical comparisons of solution rates on the ground-truth regression problems. We report Wilcoxon signed-rank tests with a Bonferonni correction on α for multiple comparisons. (Left) target noise of 0, (Right) target noise of 0.01.