STAR methods tables

Methods S1. Definition of terms, related to Table 1, Figure 1(b), and Figure 1(c)	
Richness	The count of species or KOs detected in a sample
Mean absolute error	$$\frac{\sum_{i=1}^{n} \left\lvert y_i - x_i \right\rvert}{n}$$
R^2	The proportion of the variance in the dependent variable that is predictable from the independent variables. $$1 - \frac{RSS}{TSS}$$ Where RSS is the sum of squares of residuals, and TSS is the total sum of squares.
Shannon's diversity	$$-\sum_{i=1}^{R} p_i \, ln(p_i)$$ Where p_ii is the empirical probability of detecting species *i* in the dataset.
Pielou's **Evenness index**	$$\frac{H}{H_{Max}}$$ Where H is shannon diversity and $H_{MAX}$is the maximum possible achievable shannon entropy.

Methods S2 **related to Table 1, Figure 1(b), and Figure 1(c)**		
Library	**Relevant functions/modules**	**Description**
https://scikit-learn.org/ Version 0.22.0	1. `sklearn.linear_model.ElasticNet` 2. `sklearn.model_selection.GridSearchCV` 3. `sklearn.preprocessing.PowerTransformer` 4. `sklearn.pipeline.Pipeline` 5. `sklearn.metrics.r2_score` 6. `sklearn.metrics.mean_absolute_error` 7. `sklearn.metrics.train_test_split` 8. `sklearn.model_selection.RepeatedStratifiedKfold` 9. `sklearn.ensemble.RandomForestRegressor`	1. Elastic net implementation 2. Used for hyperparameter optimization 3. Power transformation implementation 4. Utility to chain commands that need to applied during training 5. Coefficient of determination 6. MAE 7. Used to create train/test/validation data 8. Used for nested cross validation 9. Random forest implementation
http://scikit-bio.org/ Version 0.5.6	1. `skbio.stats.composition.multiplicative_replacement` 2. `skbio.stats.composition.clr`	1. Implementation of imputation using multiplicative replacement 2. CLR transformation implementation
Xgboost Version 1.0.2	1. `XGBRegressor`	1. Gradient boosting machine implementation
PyTorch Version 1.1.0	1. `torch.utils.data: DataLoader, TensorDataset` 2. `torch.autograd.Variable` 3. `3.torch.optim.*` 4. `4. torch.nn.functional` 5. `5. torch.nn`	1. Dataset utilities 2. Autodiff utilities 3. Optimization 4. Functional neural net API 5. Neural network functions

Methods S3 **related to Table 1, Figure 1(b), and Figure 1(c)**		
Model class	**Hyperparameter Space**	**Model selection method**
Neural net	`1 and 2 hidden layer NNs varying hidden dimension from 50 to 1500.`	Grid search using held out validation set.

Random forest	`n_estimators = [50, 100, 200, 300, 500]`
	`max_depth = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None]`
	`min_samples_split = [2, 5, 10]`
	`min_samples_leaf = [1, 2, 4]`
		Random search + CV

Gradient boosting machine	`learning_rate = [0.05, 0.10, 0.15, 0.20, 0.25, 0.30] ,`
	`max_depth = [ 3, 4, 5, 6, 8, 10, 12, 15]`
	`min_child_weight=[ 1, 3,5,7],`
	`gamma = [ 0.0, 0.1, 0.2, 0.3, 0.4]`
	`colsample_bytree = [0.3, 0.4, 0.5, 0.7 ]`
	`n_estimators =[50, 100, 200, 300, 500, 600]`
		Random search +CV

Methods S4 related to Table 1, Figure 1(b), and Figure 1(c)		
Model class	**Feature type**	**Selected model**
Neural net	Blood	2 hidden layers, hidden dim =1000
	Stool	1 hidden layer, hidden dim=500
Random forest	Blood	n_estimators = 200 max_depth = None min_samples_split = 10 min_samples_leaf = 2
	Stool	n_estimators = 100 max_depth = 5 min_samples_split = 2 min_samples_leaf = 1
Gradient boosting machine	Blood	max_depth = 5 min_child_weight= 1 gamma =0.3 colsample_bytree = 0.5 n_estimators = 500
	Stool	max_depth = 4 min_child_weight= 5 gamma = 0.0 colsample_bytree = 0.3 n_estimators = 500