ESR & Katz

Exhaustive Symbolic Regression and Priors for SR

https://github.com/DeaglanBartlett/ESR
https://github.com/DeaglanBartlett/katz

Deaglan Bartlett

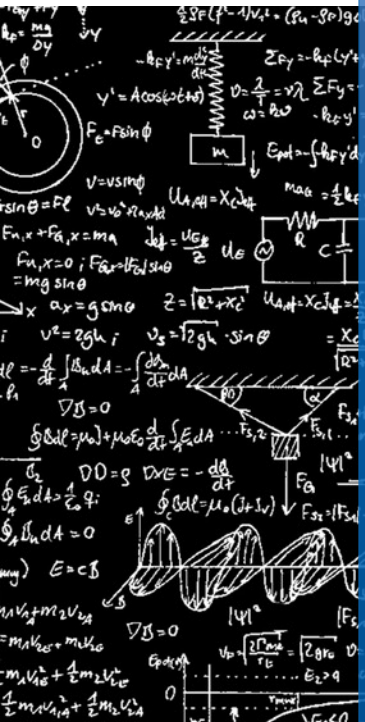
deaglan.bartlett@iap.fr

Institut d'Astrophysique de Paris
CNRS & Sorbonne Université

Astrophysics, University of Oxford

Harry Desmond & Pedro Ferreira

6 June 2023



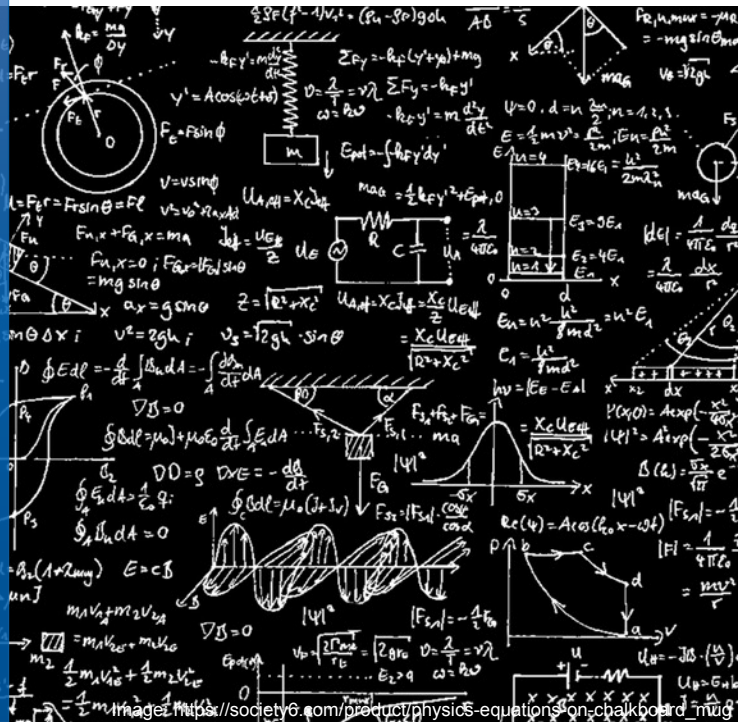


Image: https://society6.com/product/physics-equations-on-chalkboard_mug

Symbolic Regression - Overview

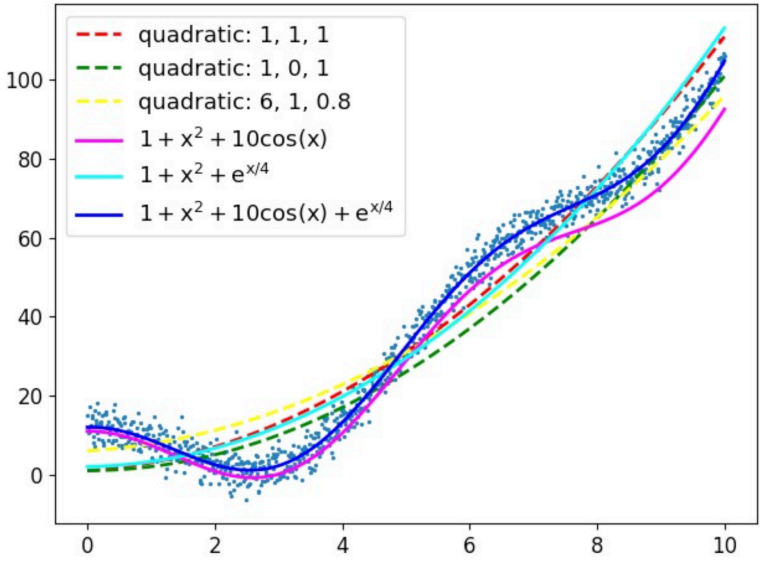
Find the best function rather than just the best parameters of a given function

Numerical Regression:

$$y = 6 + 1x + 0.8x^2$$

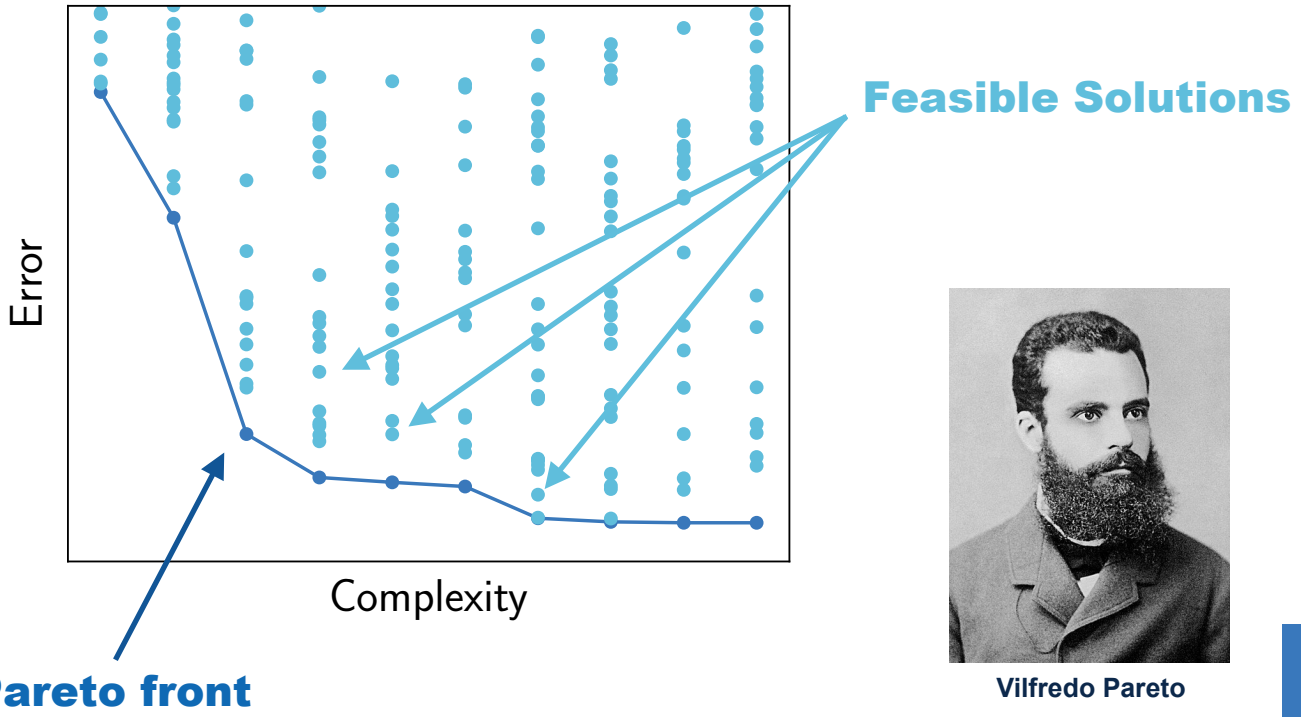
Symbolic Regression:

$$y = 1 + x^2 + 10\cos(x)$$



The Pareto Front - Balance of Simplicity and Accuracy

- **Pareto front**: most accurate solution at a given complexity
- Problem: Can make function arbitrarily complex to get zero error
- Which function on the Pareto front is the “best”?



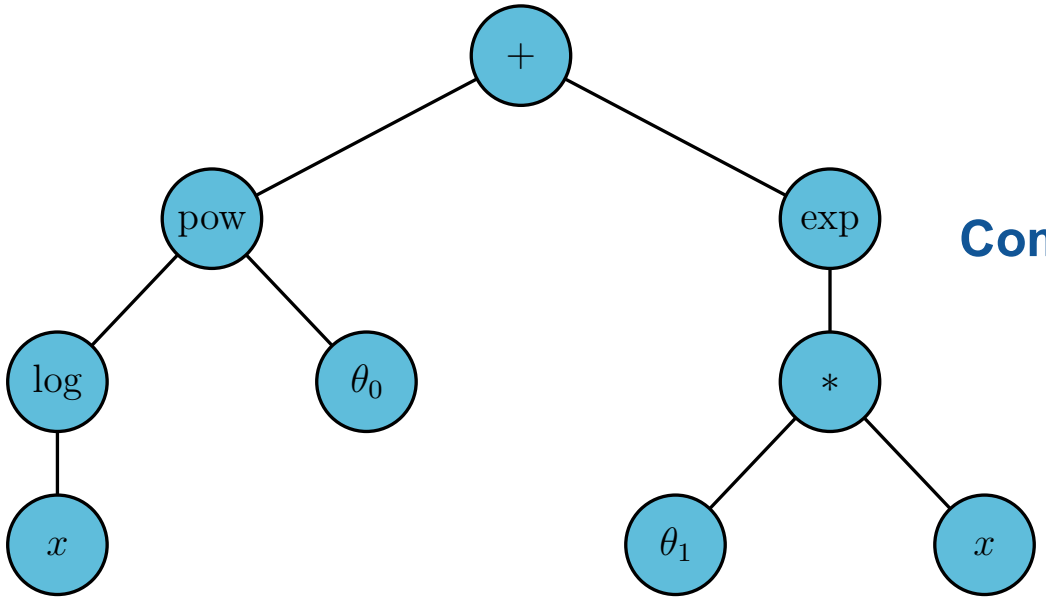
Vilfredo Pareto

Three Big Problems

1. Traditional methods do not necessarily find the best function
2. How do you balance accuracy with simplicity?
3. How do I automate human preference for certain types of function?

Typically in SR - Represent Functions as Trees or Lists

$$(\log(x))^{\theta_0} + \exp(\theta_1 x) = [+, \text{pow}, \log, x, \theta_0, \exp, *, \theta_1, x]$$

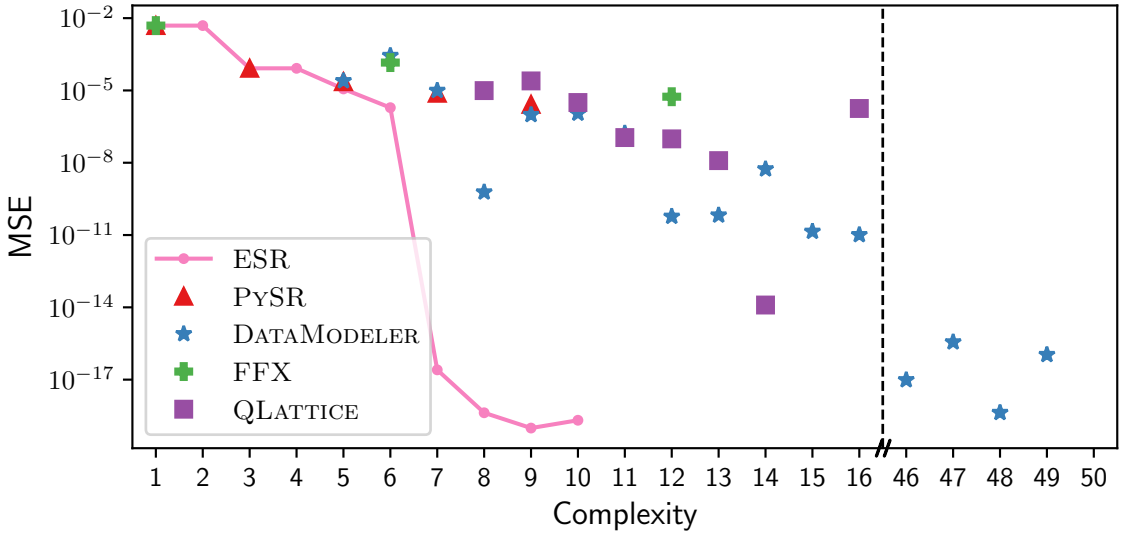


Complexity = Number of Nodes

Three Big Problems

1. Traditional methods do not necessarily find the best function
2. How do you balance accuracy with simplicity?
3. How do I automate human preference for certain types of function?

Traditional methods do not always find the true optimum



Task:

Univariate function with 100,000 noiseless data points

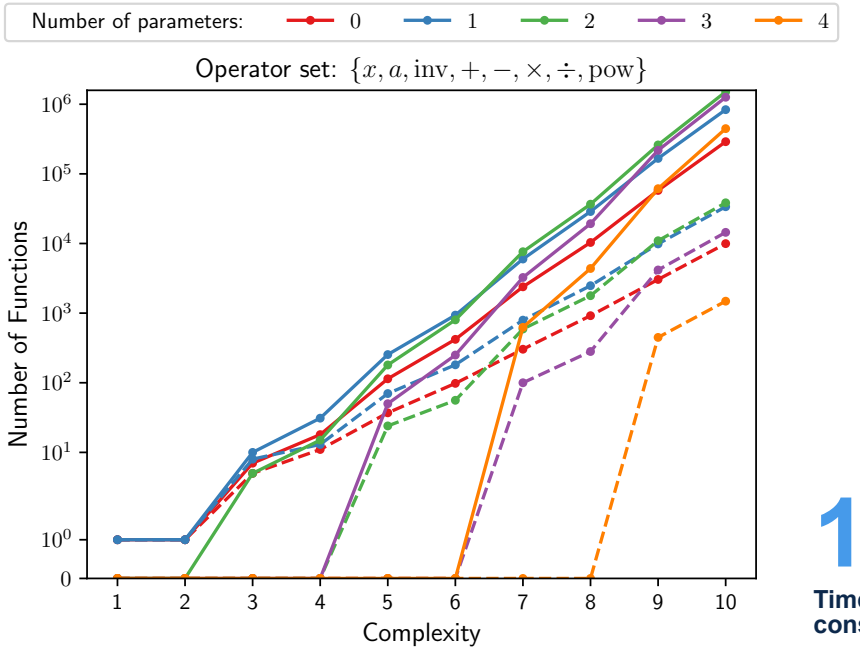
Best equation found:

$$y = \theta_1 \theta_0^{x^2}$$

$$\theta_0 \approx 1/\sqrt{e}, \quad \theta_1 \approx 1/\sqrt{2\pi}$$

None of these stochastic methods found this!

Simplifications mean that an exhaustive search is feasible



168 million

Naïve estimate: $\sum_{j=1}^{n} j^k = H_n^{(-k)}$

5.2 million

Total number of decorated trees

134,234

Number of unique equations

1400

Times fewer equations to consider than our naïve guess

119,861

Number of equations containing at least one parameter

Three Big Problems

1. Traditional methods do not necessarily find the best function
2. How do you balance accuracy with simplicity?
3. How do I automate human preference for certain types of function?

Minimum Description Length

Why do we want to fit a function anyway?

- Compress the data
- If I have a perfect function, I don't need to tell someone the y values, but just the function

How much information do I need to transmit?

$$L(D) = L(H) + L(D \mid H)$$

Description length — Hypothesis — Residuals

The best function compresses the data the most → Minimise $L(D)$

The Description Length of a Function

$$L(D) = -\log(\mathscr{L}(\hat{\theta})) + k\log(n) - \frac{p}{2}\log(3) + \sum_{i}^{p}\left(\frac{1}{2}\log(\mathrm{I}_{ii}) + \log(|\hat{\theta}_i|)\right)$$

This is the thing to optimise

Combined accuracy and simplicity into a single number

Favours functions which

- Fit the data accurately
- Contain only a few parameters
- Do not contain many operators
- Do not need finely tuned parameters

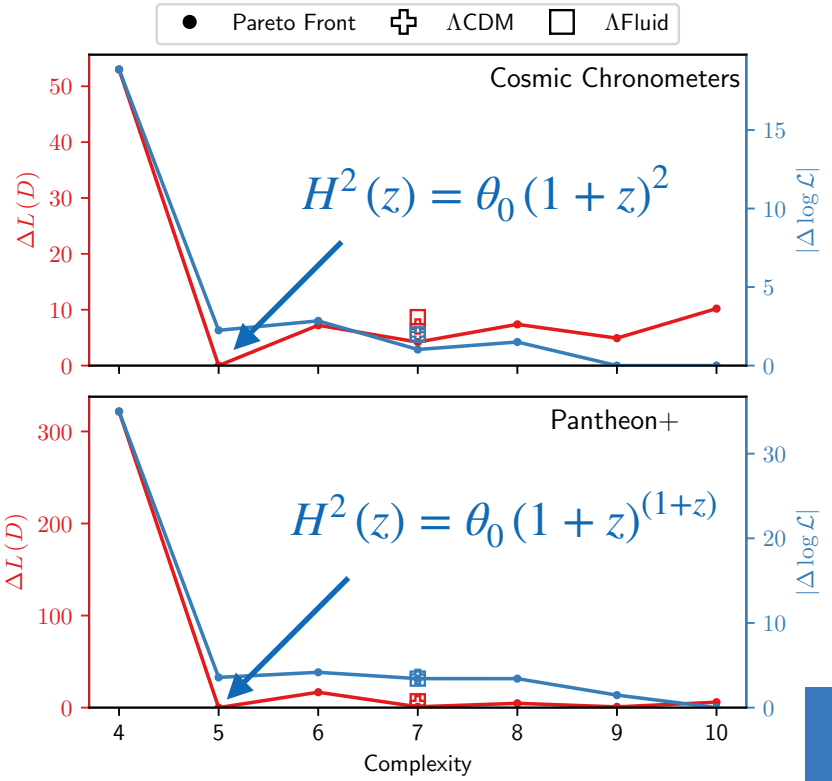
Example: Expansion Rate of the Universe

Generate all equations

```python
import esr.generation.duplicate_checker

runname = 'core_maths'
comp = 5
esr.generation.duplicate_checker.main(runname, comp)
```

Fit equations to data and find best

```python
import esr.fitting.test_all
import esr.fitting.test_all_Fisher
import esr.fitting.match
import esr.fitting.combine_DL
import esr.fitting.plot
from esr.fitting.likelihood import CCLikelihood

comp = 5
likelihood = CCLikelihood()

esr.fitting.test_all.main(comp, likelihood)
esr.fitting.test_all_Fisher.main(comp, likelihood)
esr.fitting.match.main(comp, likelihood)
esr.fitting.combine_DL.main(comp, likelihood)
esr.fitting.plot.main(comp, likelihood)
```

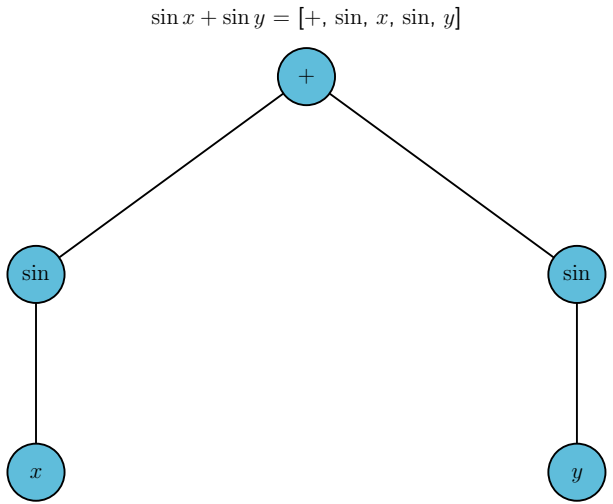


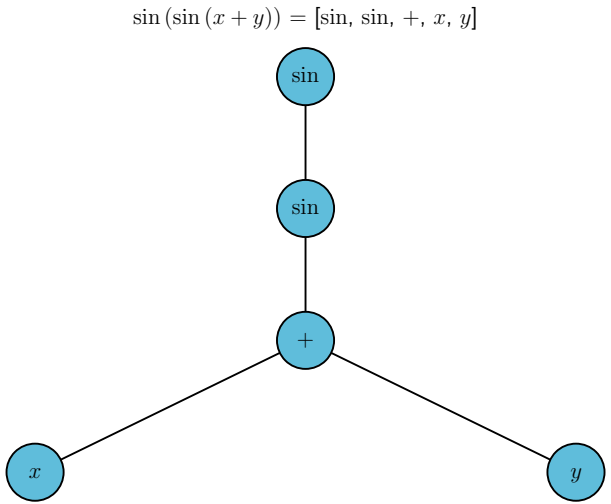
Three Big Problems

1. Traditional methods do not necessarily find the best function
2. How do you balance accuracy with simplicity?
3. How do I automate human preference for certain types of function?

Not all functions should be preferred equally a priori

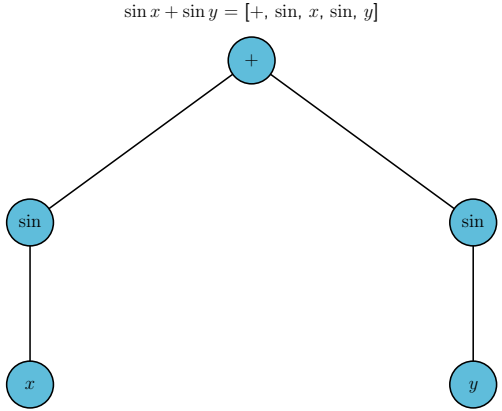
$$L(D) = -\log(\mathscr{L}(\hat{\theta})) + k\log(n) - \frac{p}{2}\log(3) + \sum_i^p \left(\frac{1}{2}\log(\mathrm{I}_{ii}) + \log(|\hat{\theta}_i|)\right)$$

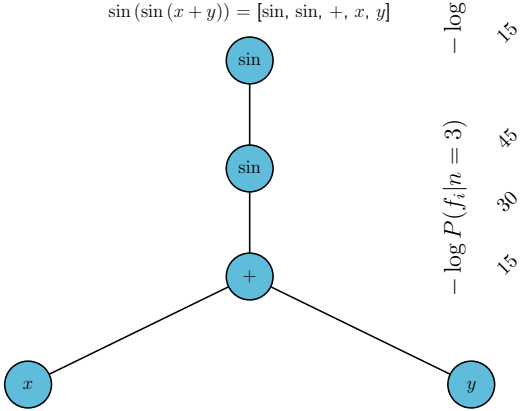


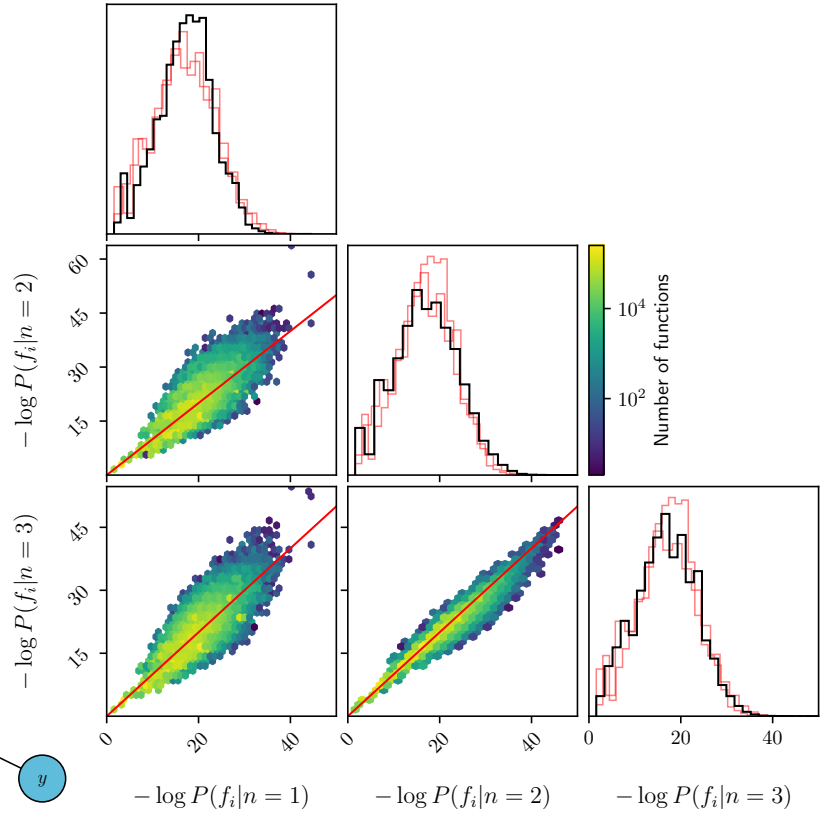


Priors on functions

- **Idea:** Weight functions based on given corpus of "good" equations based on how operators are combined
- Do this through a language model







Katz: Example

	Filename	Number	Output	Formula	# variables	v1_name	v1_low	v1_high	v2_name	v2_low	v2_high	v3_name	v3_low	v3_high
2	I.6.2a	1	f	exp(-theta**2/2)/sqrt(2*pi)	1	theta	1	3						
3	I.6.2	2	f	exp(-(theta/sigma)**2/2)/(sqrt(2*pi)*sigma)	2	sigma	1	3	theta	1	3			
4	I.6.2b	3	f	exp(-((theta-theta1)/sigma)**2/2)/(sqrt(2*pi)*sigma)	3	sigma	1	3	theta	1	3	theta1	1	3
5	I.8.14	4	d	sqrt((x2-x1)**2+(y2-y1)**2)	4	x1	1	5	x2	1	5	y1	1	5
6	I.9.18	5	F	G*m1*m2/((x2-x1)**2+(y2-y1)**2+(z2-z1)**2)	9	m1	1	2	m2	1	2	G	1	2
7	I.10.7	6	m	m_0/sqrt(1-v**2/c**2)	3	m_0	1	5	v	1	2	c	3	10
8	I.11.19	7	A	x1*y1+x2*y2+x3*y3	6	x1	1	5	x2	1	5	x3	1	5
9	I.12.1	8	F	mu*Nn	2	mu	1	5	Nn	1	5			
10	I.12.2	10	F	q1*q2*r/(4*pi*epsilon*r**3)	4	q1	1	5	q2	1	5	epsilon	1	5
11	I.12.4	11	Ef	q1*r/(4*pi*epsilon*r**3)	3	q1	1	5	epsilon	1	5	r	1	5
12	I.12.5	12	F	q2*Ef	2	q2	1	5	Ef	1	5			
13	I.12.11	13	F	q*(Ef+B*v*sin(theta))	5	q	1	5	Ef	1	5	B	1	5

Provide table of functions which language model can use

```python
from katz.prior import KatzPrior

n = 2
basis_functions = [["a", "x"],
        ["sqrt", "exp", "log", "sin", "cos", "arcsin", "tanh"],
        ["+", "-", "*", "/", "pow"]]

kp = KatzPrior(n, basis_functions, '../katz/data/FeynmanEquations.csv', '../katz/data/NewFeynman.csv')

for eq in ['x0**2', 'sin(x0) + sin(x1)', 'sin(sin(x0+x1))']:
    p = kp.logprior(eq)
    print(eq, p)
```

```
x0**2 -3.319379255053895
sin(x0) + sin(x1) -18.76483595964858
sin(sin(x0+x1)) -21.34888944026105
```

Evaluate prior on functions of interest

Prior on functions removes the physically unreasonable cases

Method	Rank 1	Rank 2		Rank 3		Rank 4	
	Function	Function	Value	Function	Value	Function	Value
Likelihood	$\left\lvert\theta_1 + \frac{1}{\theta_0 - x}\right\rvert^{\lvert\theta_2\rvert^x}$	$\left\lvert\theta_0 - x^{\theta_1}\right\rvert^{\theta_2 - x}$	1.48	$-\theta_2 + \lvert\theta_0\,(\theta_1 - x)\rvert^x$	3.15	$\frac{\lvert\theta_1 + x^{\theta_2}\rvert^x}{\theta_0}$	3.18
Score	$\theta_0 x$	$\theta_0\,\lvert\theta_1\rvert^{-x}$	11.54	$\left\lvert\theta_0 - x^{\theta_1}\right\rvert^{\theta_2 - x}$	14.32	$\left\lvert\theta_1 + \frac{1}{\theta_0 - x}\right\rvert^{\lvert\theta_2\rvert^x}$	14.59
MDL	$\theta_0 x^x$	$\lvert\theta_0\rvert^{x^{\theta_1}}$	0.78	$\theta_0\,\lvert\theta_1\rvert^{-x}$	0.83	$\theta_0 x^{x^{\theta_1}}$	0.91
MDL+LM	$\frac{x\left(2x + \frac{1}{x}\right)}{\theta_0}$	$\frac{2x^2 + \frac{1}{x}}{\theta_0}$	1.40	$\frac{x^2 + 2x}{\theta_0}$	1.63	$\theta_0 + \theta_1 x^3$	1.72
MDL+FBF+LM	$\lvert\theta_0\rvert^{\frac{1}{\theta_1 + 2x}}$	$\lvert\theta_0\rvert^{\lvert\theta_1\rvert^x}$	0.16	$\theta_0\,\lvert\theta_1\rvert^x$	1.77	$\frac{\theta_1 - x^2}{\theta_0}$	2.19
Bayes+FBF+LM	$\lvert\theta_0\rvert^{\frac{1}{\theta_1 + 2x}}$	$\lvert\theta_0\rvert^{\lvert\theta_1\rvert^x}$	0.16	$\lvert\theta_0\rvert^{-\theta_1 + x}$	1.03	$\frac{x^3 + \frac{1}{\theta_0}}{\theta_1}$	1.95

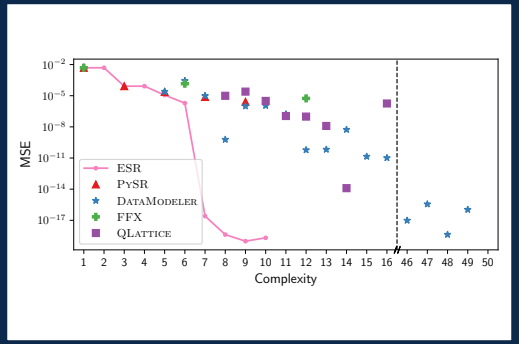
$y\,(x \equiv 1 + z) \equiv H^2\,(z)$

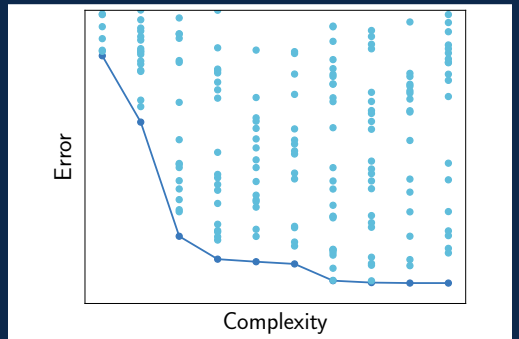
Basis: $\{x \equiv 1 + z, \theta, \text{inv}, +, -, \times, \div, \text{pow}\}$

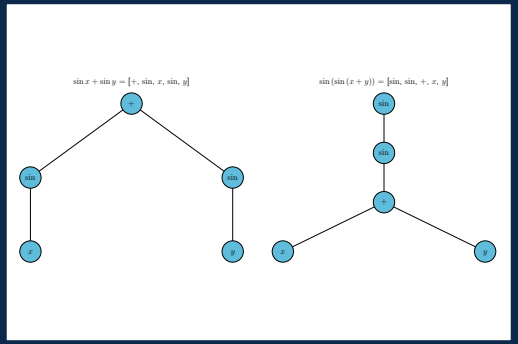
Conclusions: Three Big Problems Solved with ESR and Katz

arXiv:2211.11461
https://github.com/DeaglanBartlett/ESR

arXiv:2304.06333
https://github.com/DeaglanBartlett/katz







1. Traditional methods do not necessarily find the best function

2. How do you balance accuracy with simplicity?

3. How do I automate human preference for certain types of function?

deaglan.bartlett@iap.fr