Riesz Estimator: Motivation and Implementation

Slawomir Tur

September 2024

Riesz Estimators

- **Intuitive definition**: Riesz estimators model relationships that are linear but change across segments or thresholds - **continuous piecewise linear functions**
- **Why not MARS?** (Most) economists are still apprehensive about machine learning (ML as 'black box' inference), we know less about splines than we do about piecewise linear functions, and MARS do come with some bias
- **Why not threshold regression?** Riesz algorithms work just as well with thresholds on many predictors as they do with thresholds on one

Why are they popular as objects of inquiry?

- They capture structural breaks, or 'kinks', in data where typical linear regression fails
- They naturally occur in many theoretical and practical problems:
  - The neural net activation function **ReLU** is a piecewise linear function – $ReLU(ax + b) = 0 \vee (ax + b)$
  - In finance, option payoff curves are represented by piecewise linear functions
  - In microeconomics, such functions can be found in the **revealed preference** literature

Just enough maths to help you understand

- You might have noticed that I used $\vee$ in the definition of ReLU, instead of max
- This is counterintuitive notation (blame the order theorists):
  - The down arrow, $\vee$, is the *supremum* (maximum)
  - The up arrow, $\wedge$, is the *infimum* (minimum)
- We will be working with pointwise maxima and minima, e.g. for two 4-element vectors X, Y:

X	Y	$X \wedge Y$	$X \vee Y$
1	2	1	2
3	5	3	5
5	2	2	5
6	5	5	6

Just enough maths to help you understand

- Ovchinnikov (2002); Aliprantis, Harris, and Tourky (2006): Every piecewise linear function can be represented with the right combination of maxima and minima of its linear **components**
- If a piecewise linear function has a simple enough form (more on that later), there is only a finite number of such combinations
- If we manage to extract the linear functions from each region, we can search over those combinations and get a good estimate of the piecewise function

Worked example

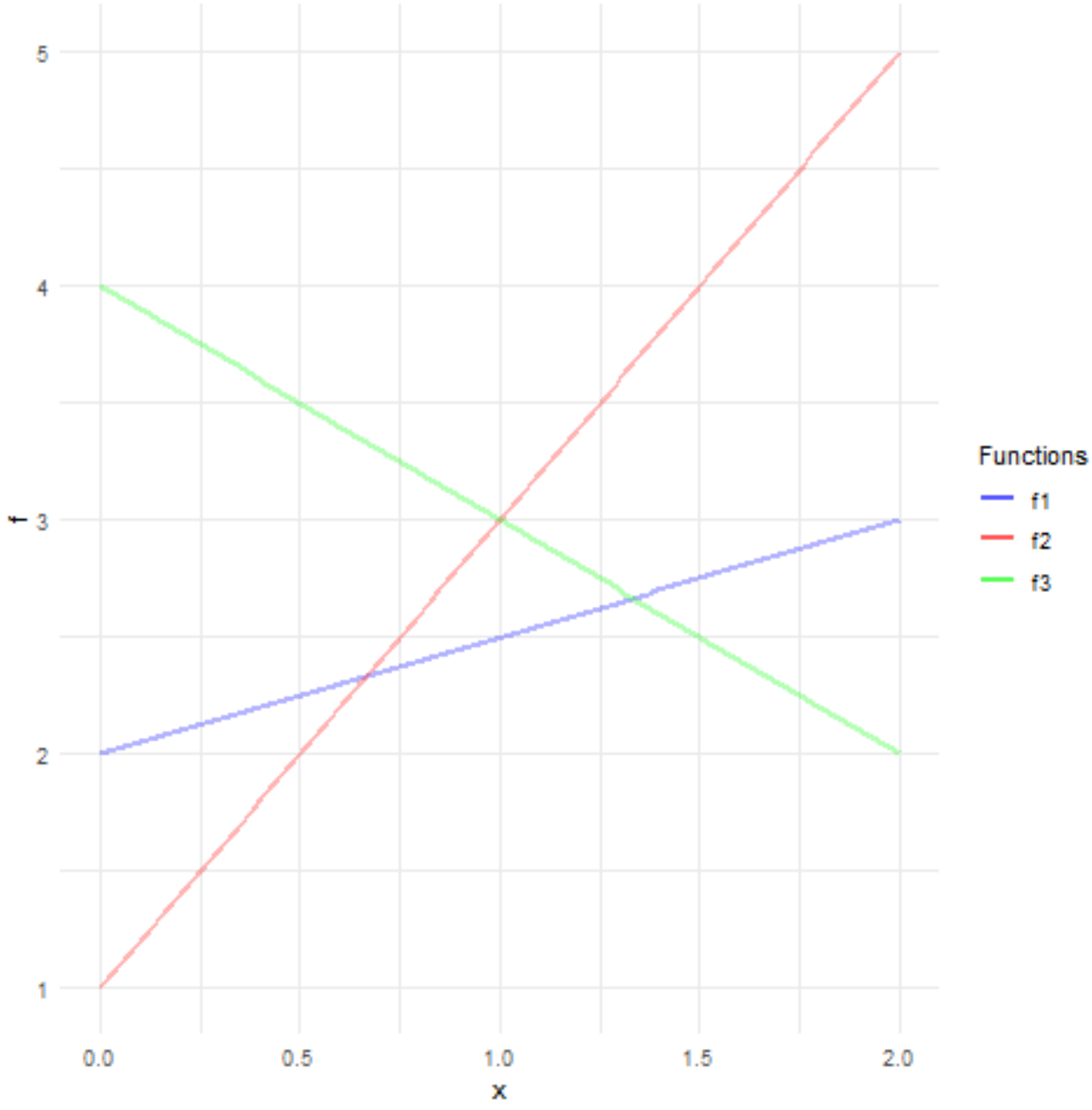
Let $f : [0, 2] \to \mathbb{R}$, where:

$$f(x) = \begin{cases} 2 + 0.5x & x \in [0, \frac{2}{3}] \cup [\frac{4}{3}, 2] \\ 1 + 2x & x \in [\frac{2}{3}, 1] \\ 4 - x & x \in [1, \frac{4}{3}] \end{cases}$$

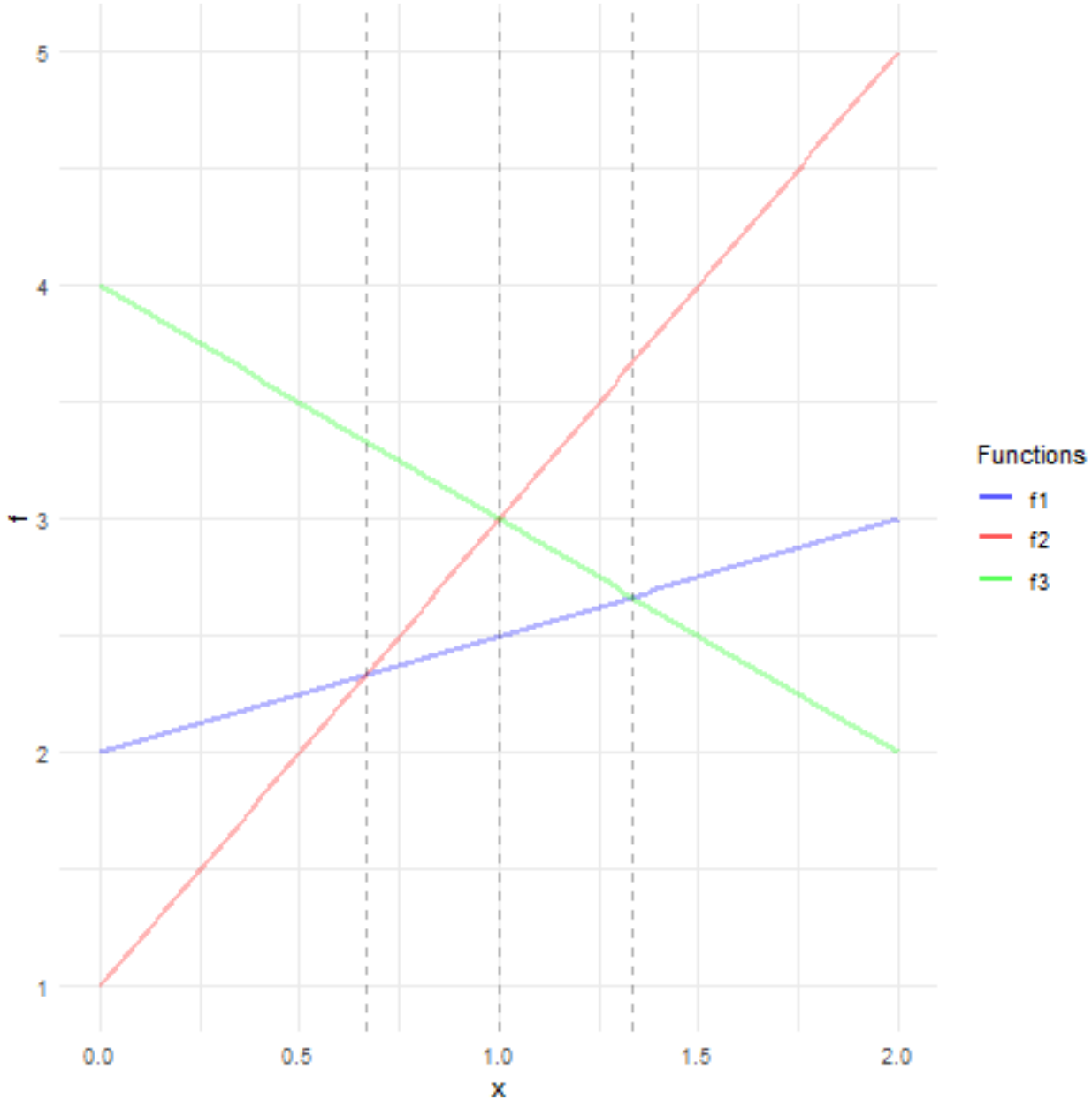
Worked example

1. Enumerate the linear components of the function, and determine the set of intersecting functions E:
   - $f_1 = 2 + 0.5x, f_2 = 1 + 2x, f_3 = 4 - x$
   - All of the functions intersect, so $E = \{(1,2),(1,3),(2,3)\}$
2. Solve for the points at which each pair intersects:
   - $H_{(1,2)} = [f_1 = f_2] = [2 + 0.5x = 1 + 2x] = [x = \frac{2}{3}] = \{x = \frac{2}{3}\}$
   - $H_{(1,3)} = [f_1 = f_3] = [2 + 0.5x = 4 - x] = [x = \frac{4}{3}] = \{x = \frac{4}{3}\}$
   - $H_{(2,3)} = [f_2 = f_3] = [1 + 2x = 4 - x] = [x = 1] = \{x = 1\}$

Worked example



Worked example



Worked example

These points split $[0, 2]$ into J cells:

- $K_1 = [0, \frac{2}{3}], K_2 = [\frac{2}{3}, 1], K_3 = [1, \frac{4}{3}], K_4 = [\frac{4}{3}, 2]$

3. For each cell h, pick a point x_h (arbitrary for maths reasons, I chose the midpoint of each cell)

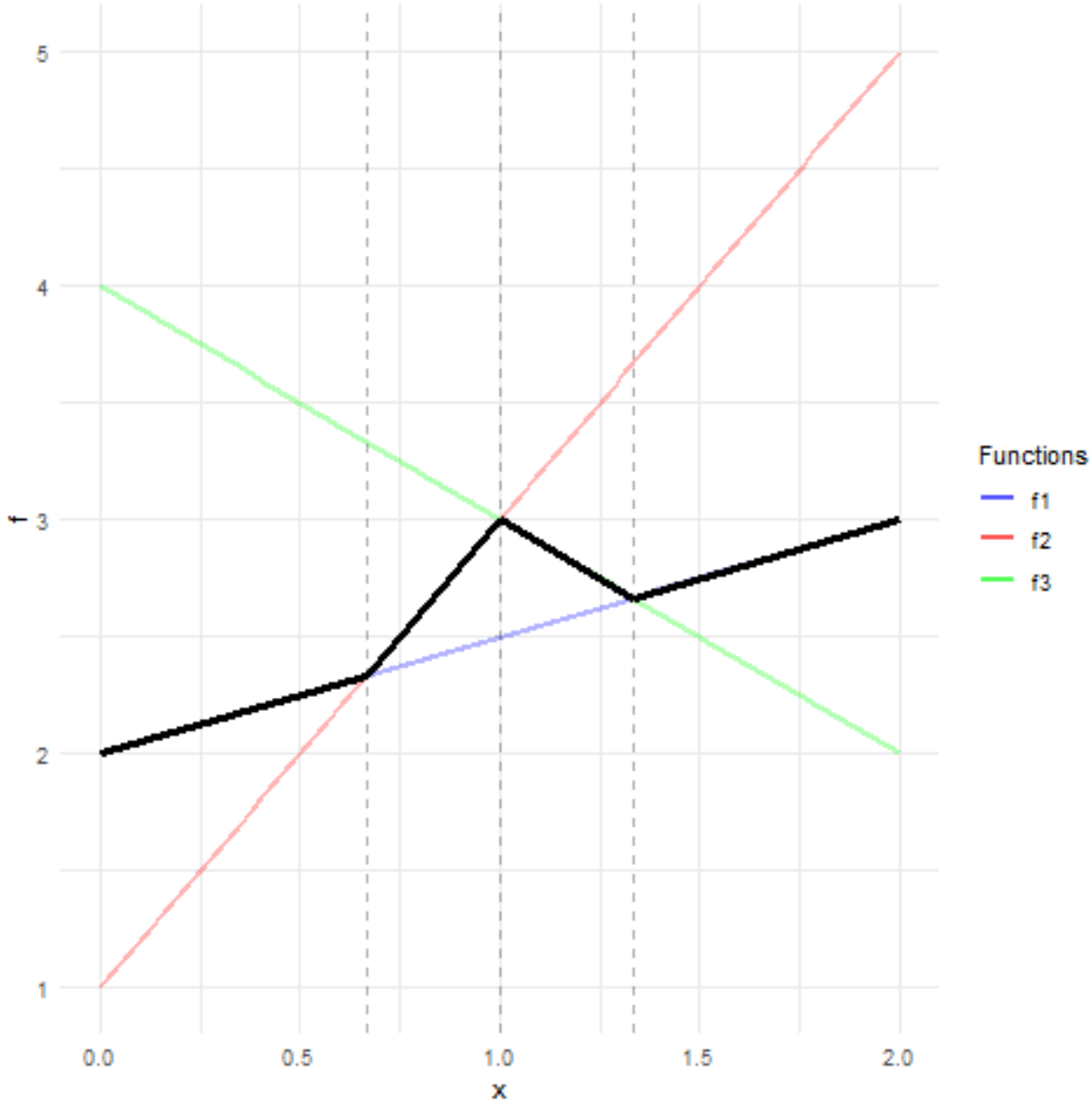
- $x_1 = 1/3$
- $x_2 = 5/6$
- $x_3 = 7/6$
- $x_4 = 5/3$

Worked example

4. For each x_h, determine the set E_h of linear components which have a greater or equal value to the piecewise $f(x)$ at x_h:
   - $E_1 = \{i \in \{1, ..., p\} : f_i(1/3) \geq f(1/3)\} = \{1, 3\}$
   - $E_2 = \{i \in \{1, ..., p\} : f_i(5/7) \geq f(5/6)\} = \{2, 3\}$
   - $E_3 = \{i \in \{1, ..., p\} : f_i(7/6) \geq f(7/6)\} = \{2, 3\}$
   - $E_4 = \{i \in \{1, ..., p\} : f_i(5/3) \geq f(5/3)\} = \{1, 2\}$
5. The function is:

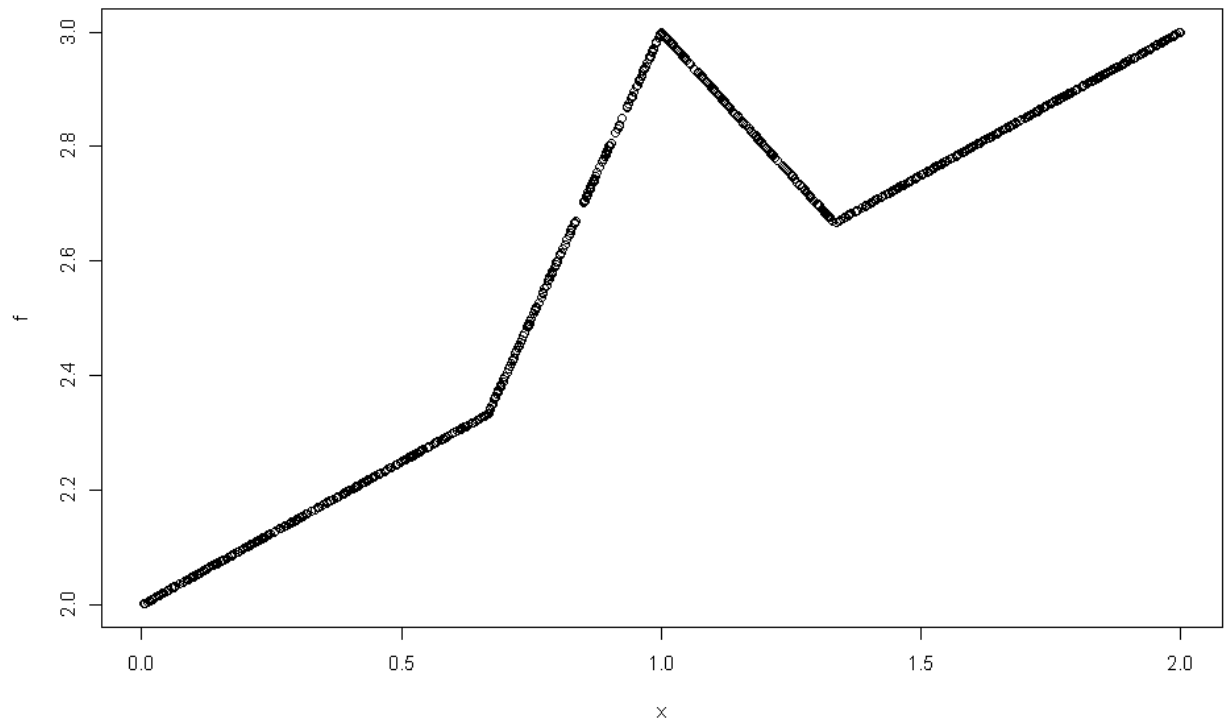
$$f = \bigvee_{j \in J} \bigwedge_{i \in E_j} f_i = (f_1 \wedge f_3) \vee (f_2 \wedge f_3) \vee (f_2 \wedge f_3) \vee (f_1 \wedge f_2)$$

Worked example



Worked example

```r
x <- runif(1000, min = 0, max = 2)
f1 <- 2 + 0.5*x
f2 <- 1 + 2*x
f3 <- 4 - x
f <- pmax(pmin(f1, f3), pmin(f2, f3), pmin(f1, f2))
plot(x, f)
```



Worked example

6. Whenever $E_i \subseteq E_j$, E_j is redundant since e.g. $(x \wedge y) \vee (x \wedge y \wedge z) = (x \wedge y)$. So, we take the minimal set J^* where for any two E_i, E_j in J^*, neither $E_i \subseteq E_j$ nor $E_j \subseteq E_i$

$$f = \bigvee_{j \in J^*} \bigwedge_{i \in E_j} f_i = (f_1 \wedge f_3) \vee (f_2 \wedge f_3) \vee (f_1 \wedge f_2)$$

Surprisingly, this process works just as well for functions on n dimensions (see Aliprantis, Harris, and Tourky's 'Continuous piecewise linear functions' for all the proofs, if you're so inclined).

Algorithm overview

- **Back to estimation:** We don't have access to the linear components, nor the precise points at which they intersect – we're working with finite data, after all
- **Two approaches**:
  - **RIESZVAR(i)**: the 'analytical' method
  - **RIESZVAR(ii)**: piecewise quadratic optimisation
- **Goal**: Find the optimal combination of linear segments to minimize overall error

RIESZVAR(i) – Estimation Procedure

Idea: if we split the predictor space into a fine enough grid of partitions, these partitions will be entirely inside their 'true' cells. We can retrieve the linear components on those cells with linear regression and search through the finite set of their max-min combinations.

- **Data Splitting**: Split the ordered predictor data into k regions of size k/n
- **Linear Regression**: Simple as that, run a regression on each region and extract the coefficients
- **Generate set of combinations**: Generate the combinations for the specified model
- **Searching**: Search across the generated combinations for k, and choose the combination that minimises the SSE

Sperner Families (last bit of maths, I promise)

- Two sets X, Y are **incomparable** whenever $X \not\subseteq Y$ and $Y \not\subseteq X$
- A family of incomparable subsets of a set is a **Sperner family**
- e.g. $\{f_1, f_2\}$ and $\{f_2, f_3\}$ are both contained in $\{f_1, f_2, f_3\}$, but neither is in the other; thus, they form a Sperner family
- Remember the last step of the worked example, where we removed every superfluous set? Doing so reduces the number of possible combinations from 2^{2^n} to something much more manageable

Sperner Families

- For each family, we take the minimum within the sets, and the maximum across the minima
- Limiting the number of elements any one family has to some q restricts the algorithm to an upper bound of q functions in the final output
- This can make the process more efficient computationally (we need to have a good idea of what the model might be)

Sperner Families – C subroutine

Base R doesn't (to my knowledge) have a 'set' type, unlike Python: we only have access to ordered and indexed data types
It's also a high-level statistical programming language, and doesn't do as well with complex maths as low-level languages do

Sperner Families – C subroutine

I instead used bitmasks to represent the sets (from the right, 1 if the element features in the set), and implemented a subroutine using C via R's API

$$\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\} \to 11111111$$
$$\dots$$
$$\{f_1\} \to 00000001$$
$$\{f_1, f_3\} \to 00000101$$
$$\{f_2, f_4, f_8\} \to 10001010$$

C also allows for easy parallelisation via OpenMP (one calculation per CPU core at any given time) and manual memory management

Example with simple threshold

```
> set.seed(123)
> x <- cbind(runif(1000, 0, 12), runif(1000, 0, 12))
> f1 <- 4 + 0.2*x[,1] + 0.3*x[,2]
> f2 <- x[,1] + 0.3*x[,2]
> y <- pmin(f1, f2) + rnorm(1000)
> f_hat <- rieszvar_i(y, x, 2, 2)
$f1
[1] 0.1307117 0.9621786 0.2962513

$f2
[1] 3.7769718 0.2294911 0.2991521

[1] "Best Sum of Squared Errors: 1031315599.98064"
[1] "Best Sperner Family: 4"
[1] "Corresponding Sets in the Best Sperner Family:"
[[1]]
[1] 1 2

[1] "The model that minimizes the error is: f = f1 ^ f2"
>
```

Problem 1 – Curse of dimensionality

- The number of Sperner families rises with n, but much less quickly than 2^{2^n}
- We know this number for n-sets up to 9 elements – it's the **Dedekind numbers** sequence (A007153 on OEIS):

  $1, 4, 18, 166, 7579, 7828352,$
  $2414682040996, 56130437228687557907786,$
  $286386577668298411128469151667598498812364$

- See the problem? (The program breaks for $k > 5$)

Problem 2 – Finite identification issues

- Suppose you misspecify the model, e.g. you choose $k = 4, q = 3$, where true $k = 3, q \geq 3$:
- The algorithm estimates one of the functions on two cells, leading to efficiency loss

RIESZVAR(ii) – Numerical Estimation

- **Idea:**

Appendix A – Potential RIESZVAR(i) fix

The space of Riesz estimators on the random variable **X** is a **vector lattice**, or **Riesz space**, on the set of linear functions on **X**. For any Riesz estimators Y, Z on **X**, $Y \vee Z$ and $Y \wedge Z$ are also Riesz estimators. The conceit of my current work is, if you can't estimate an 8-piece function, you might instead be able to estimate two 4-piece functions and combine *them*.
There are many recent articles using Ovchinnikov's result, owing to the popularity of ReLU in neural nets. One that hints at the result I'm thinking of is