⚙ Switching Gears ⚙

- Now we get to talk about "how" to do this!
- Code is provided through Google Collaborate.
- Video walk-throughs are available and contain more details than this talk.
- You can also run the code locally if you prefer.

My background is... not Sheaf Theory!

Many here come from different backgrounds. I hope to "represent" those who are new to Sheaf Theory & Topology, but not necessarily new coding/mathematics generally.

My background:

- Computer-Aided Geometric Design
- Recreational Mathematics (puzzle analysis)
- Teaching and Education
  - Taught at 7 colleges/universities (multiple departments)

Examples, Examples, Examples

These were created through ongoing Q&A sessions with MR, with each example highlighting one or more important areas of theory.

- In this set of slides
  - 0: Linear Regression
  - 1: Tracking an explosion through sound
  - 2: “Fox Hunting” (Radio hide-&-go-seek game)

Linear Regression

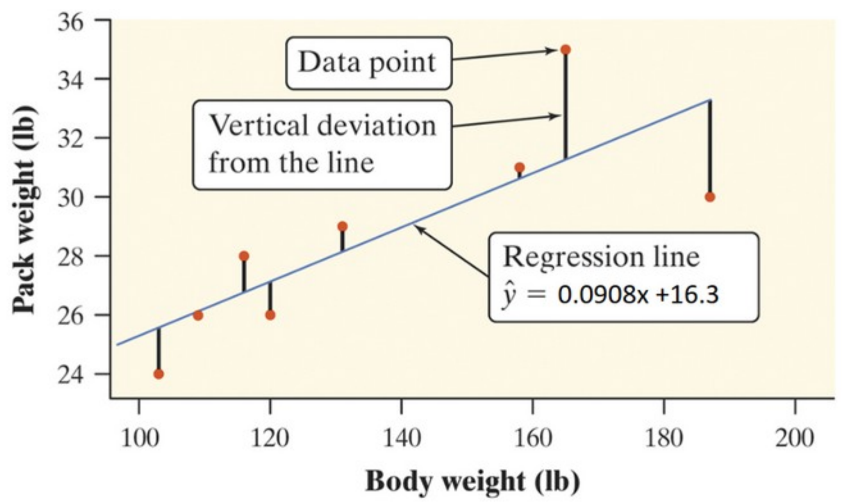
- Starting with something familiar

https://www.floydmath.com/uploads/7/5/0/5/75057801/ap_stats_3.2_least_square_regressions.pdf

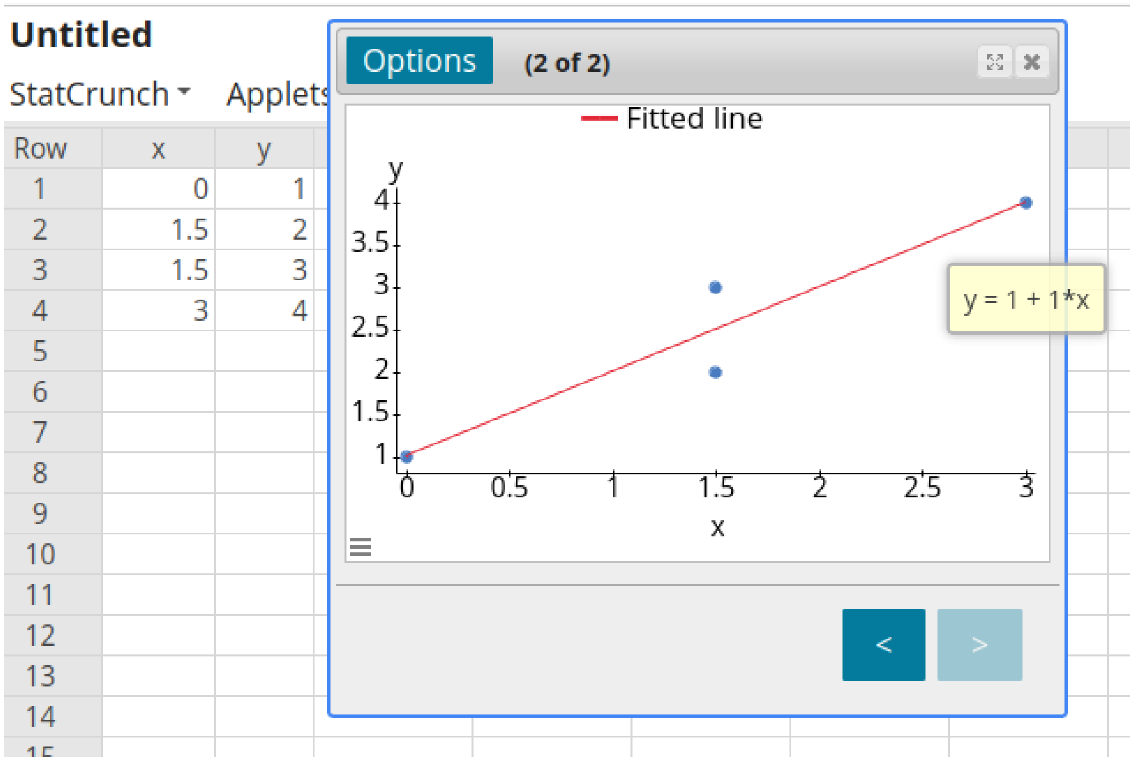
Automatic Zoom

Regression Line Equation

Body weight (lb):	120	187	109	103	131	165	158	116
Backpack weight (lb):	26	30	26	24	29	35	31	28



Start with an overly-simple problem:



Solution: $y = x + 1$

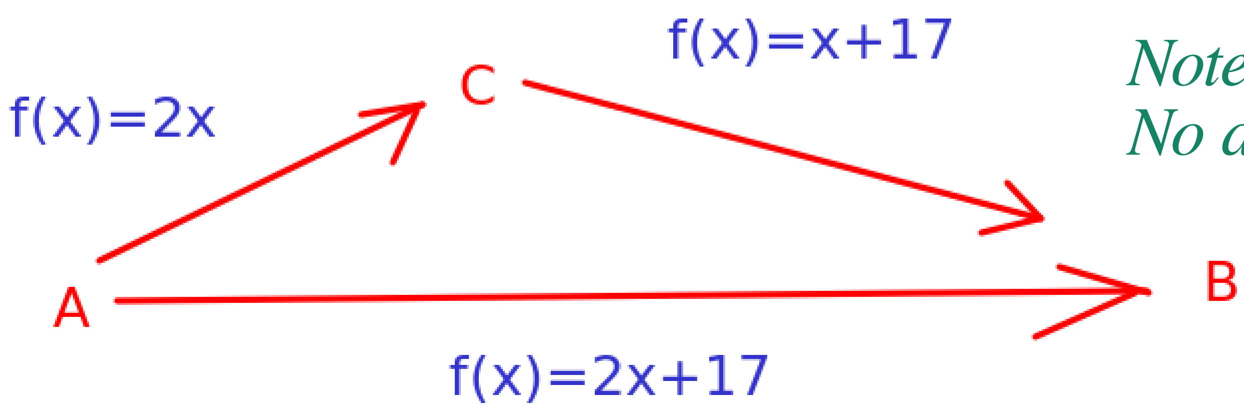
What we need for a Sheaf:

- Directed Acyclic Graph (nodes & arcs, no loops)
- Nodes are sets (stalks)
- Arcs/Edges are functions (restrictions)
- Path independence (kind-of, mostly*)

* *Warning: We'll violate this in a future example, somewhat.*

What we need for a Sheaf:

- Directed Acyclic Graph (nodes & arcs, no loops)
- Nodes are sets (ie: All Real Numbers)
- Edges are functions (ie: Some polynomial)
- Path independence (kind-of, mostly)

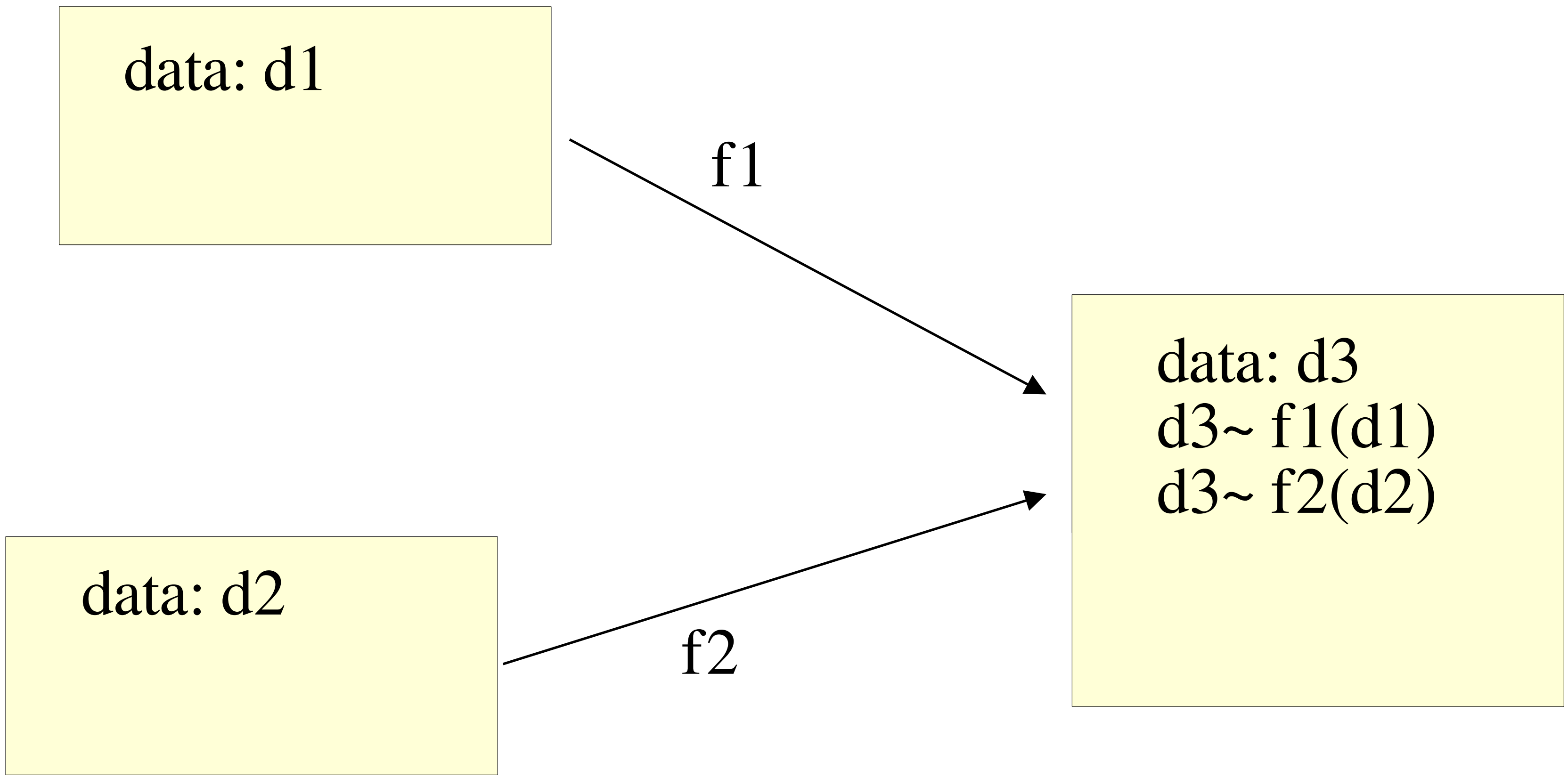


Note:
No data!

Caveats

- Some of the data will be "baked into" the sheaf
- Some data will be part of the "assignment"
- Best practices: Create sheaf, then add data
- We have to rethink what we know about the flow of the information! We will not follow the same logical process we are all used to!

Cell-to-Cell Data Flow



Cell-to-Cell Data Flow

- We need a distance metric to measure how far off these calculated estimates are.
- For LSR example, if we use the squares of the differences (L2) we get back to our usual least-squares regression model.

data: d3
d3~ f1(d1)
d3~ f2(d2)

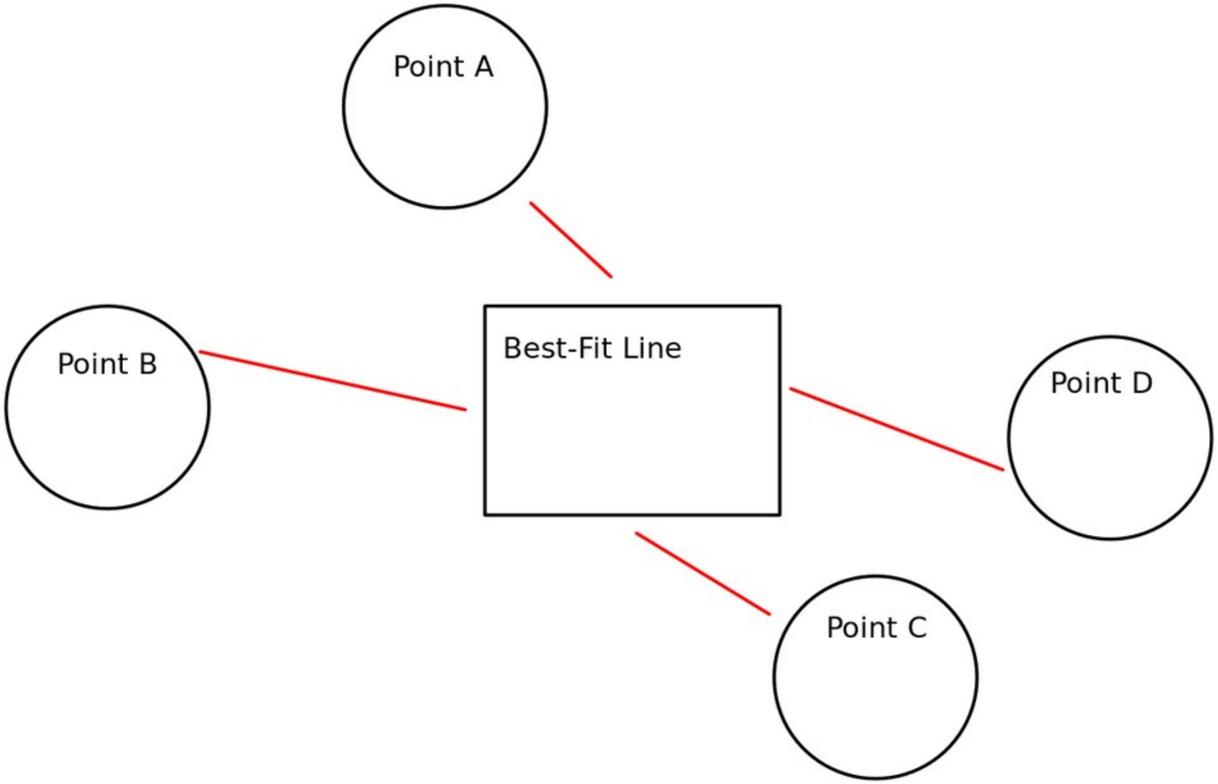
$$\sum (d_i - f_n(d_n))^2$$

- Sum over all paths * from cell(s) i to n.
- This is the CR (Consistency Radius)

** More later on this!*

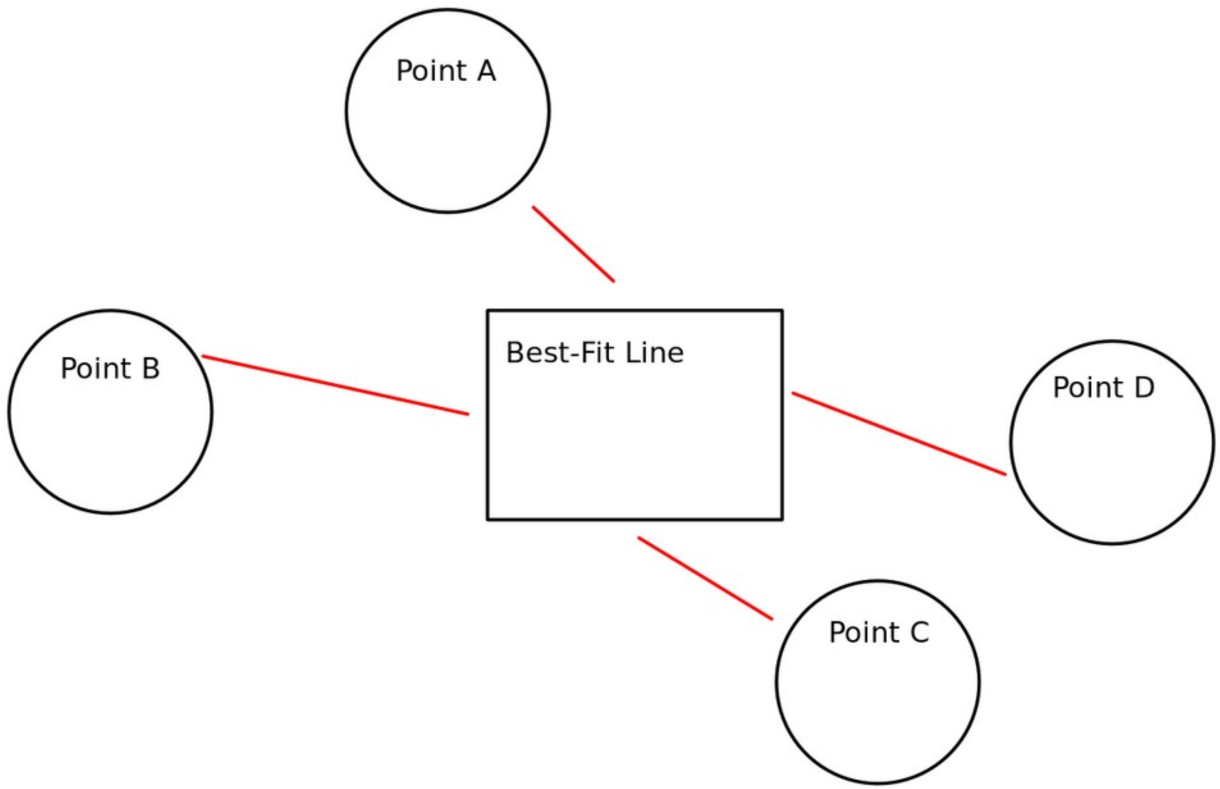
Let's Build a Sheaf!

- We'll need our four points and our model, but... which way will our arrows point!?



Let's Build a Sheaf!

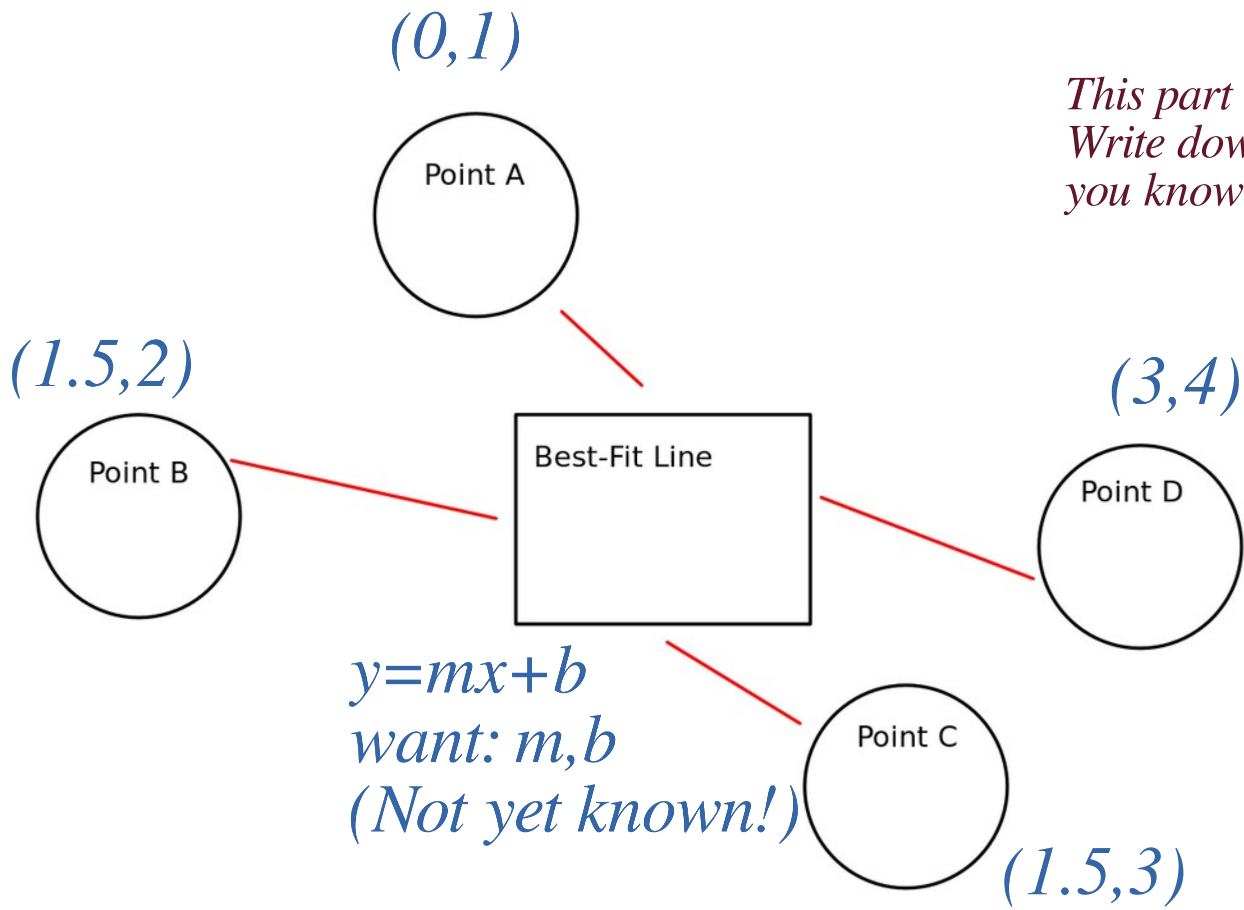
We have a directed acyclic graph regardless, also path independence. Nodes must be sets. Arcs/Edges must be functions.



Let's Build a Sheaf!

- Nodes must be sets. Edges must be functions.
- functions must push "predictions" & compare with data.

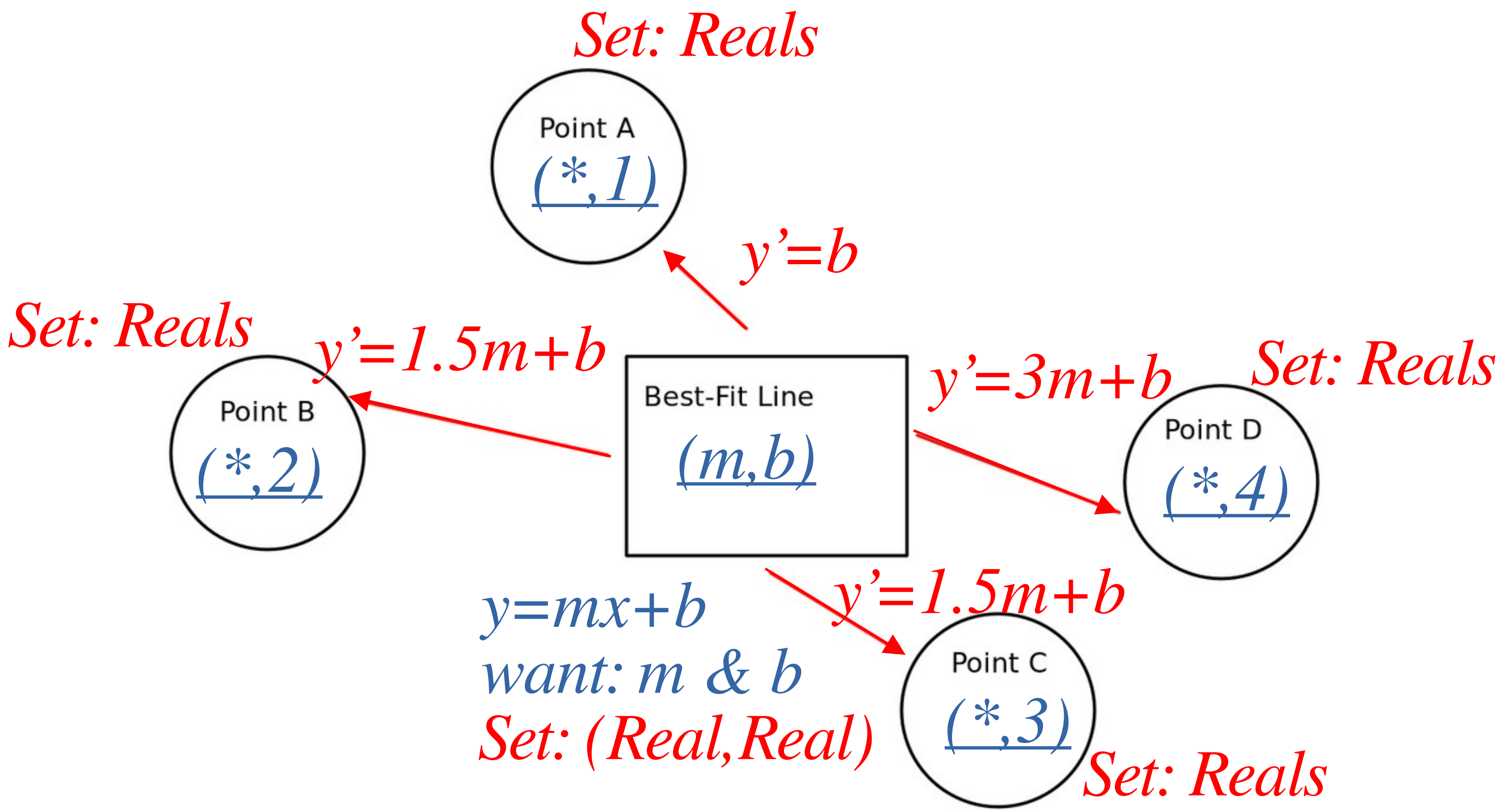
This part is messy! Write down what you know first!



a Sheaf plus an Assignment

- *Sheaf is Red*
- *Assignment is Blue*

Note: x-values are “baked-in”



How to code it in PySheaf

1) Load in PySheaf and required dependencies
2) shf=ps.Sheaf() *#Calls a sheaf into existence*
3) Select (or use default) norm type for distances
4) Add all cells (nodes), specifying their dimensions
5) Add cofaces (arcs) between cells, with functions
6) (Optional: Create sketch to check geometry)
7) Place (remaining) data on cells (all cells!)
8) Maximally extend relevant cells (or all cells)

(Continues on next page...)

How to code in PySheaf

9) Check the starting consistency radius (CR)
   - CR plays same role as Residuals in LSR models
   - Using the 2-norm, it's in fact the Sum of Squares

10) Set optimization cells (containing initial guesses)

11) FuseAssignment *#This is where the magic happens*

12) Report on what just happened, especially new CR

This code is provided in Google Collaborate.

Note: *pip install ./pysheaf* and other install hacks!

Solution Test Case Results:

- 0.7071067811865476 *(When we start at optimal!)*
- Value at A is 1 *(A-D were not allowed to change.)*
- Value at B is 2
- Value at C is 3
- Value at D is 4
- Value at L is [1.   1.00000001] *(Stays at optimal.)*
- Consistency Radius: 0.7071067811865481

Now iterate to optimize:

- 4.743416490252569 *CR when L=[-1,3]*
- Value at A is 1
- Value at B is 2
- Value at C is 3
- Value at D is 4
- Value at L is [0.99999961 1.00000253]
- Consistency Radius: 0.7071067811977653

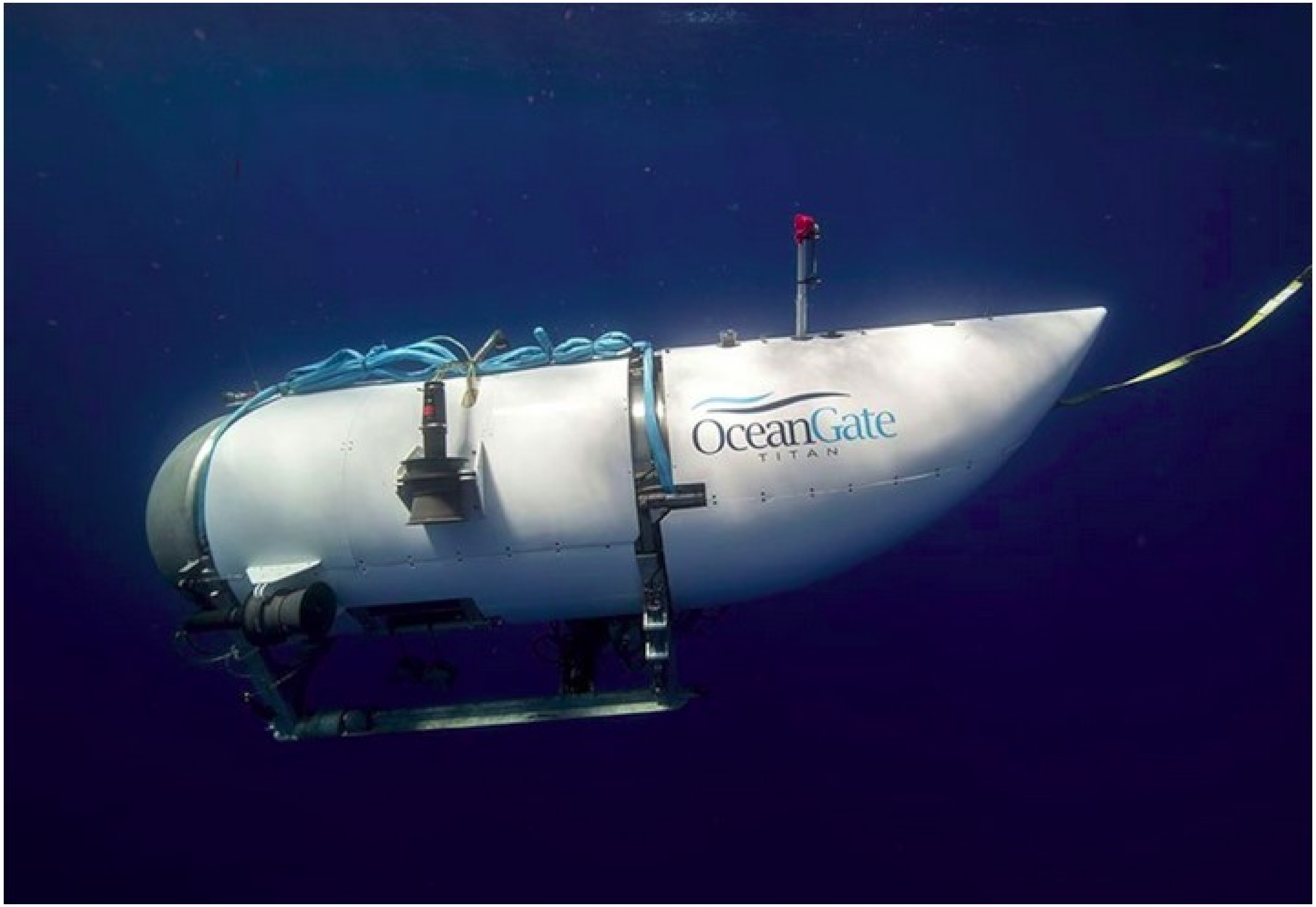
We returned to the optimal solution.

CR gives sum of squares of deviations.

For Practice (Later)

- Rework code provided for this 4-point case to solve for the LSR line given on slide 3, which has 8 points.

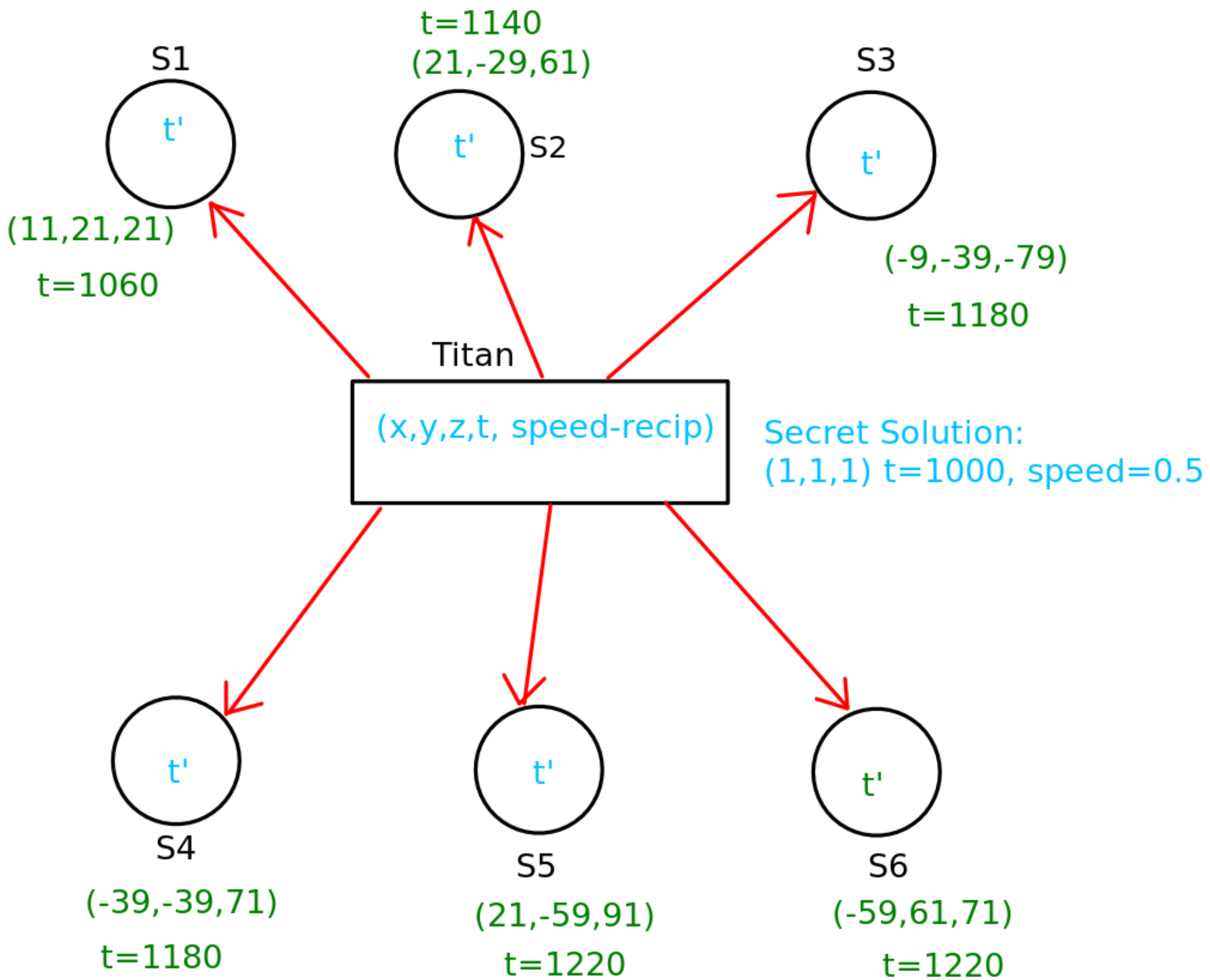
Underwater Explosion



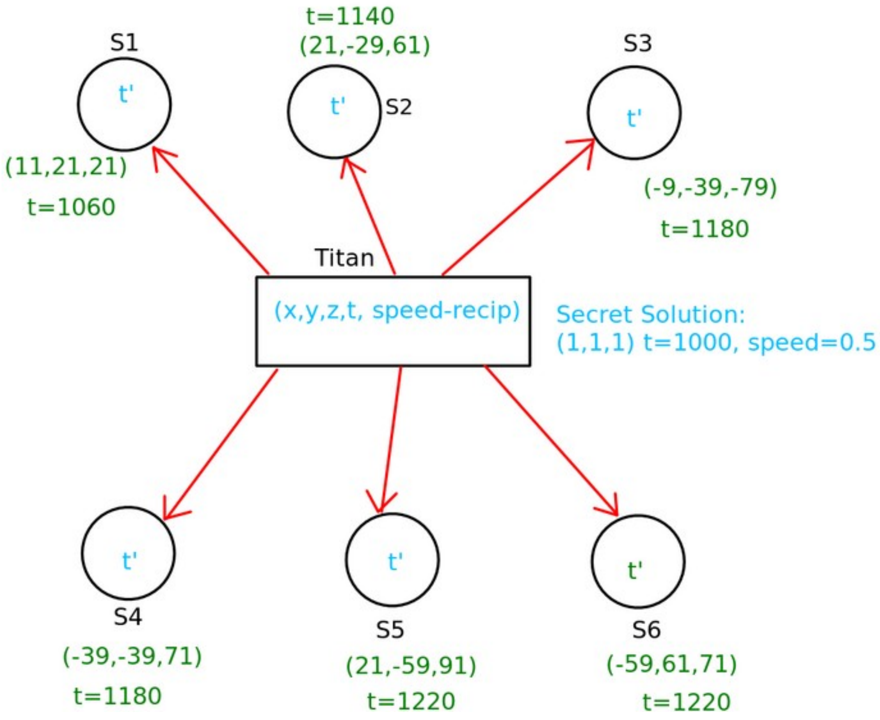
In this undated file photo, the Titan submersible, operated by OceanGate Expeditions to explore the wr... **Show more**

EyePress News via Shutterstock, FILE

Our simplistic model & solution



Our model



- Known: six sensor locations, pulse arrival times
- Unknown: Titan's location, explosion time, and speed of sound
- Functions predict pulse arrival times (t') given specified unknowns

First: check known solution stability

- 0.0
- Value at S1 is 1060
- Value at S2 is 1140
- Value at S3 is 1180
- Value at S4 is 1180
- Value at S5 is 1220
- Value at S6 is 1220
- Value at Titan is [1.00002008e+00 1.00001791e+00 9.99995551e-01 1.00000003e+03
- 1.99999950e+00]
- Consistency Radius: 3.749372915861026e-05

How far can we stray from solution?

- shf.GetCell('Titan').SetDataAssignment(ps.Assignment('real5',np.array([102,102,112,800,3]))) *# initial guess*
- 549.9544350316344 *# Original CR*
- Value at Titan is [1.00002697 1.00000825 1.00000081 1000.00003684 1.99999955]
- Consistency Radius: 3.404628416991676e-05

We got back to the solution when we started at [102,102,112,800,3].

For Practice (Later)

- Move the initial guess around to see how far away you can get from the true answer and still land on it.
- You can control the number of iterations as you explore. The default is 100.
- Example:

  shf.mMaximumOptimizationIterations=5
  Where "shf" is the name you gave your sheaf

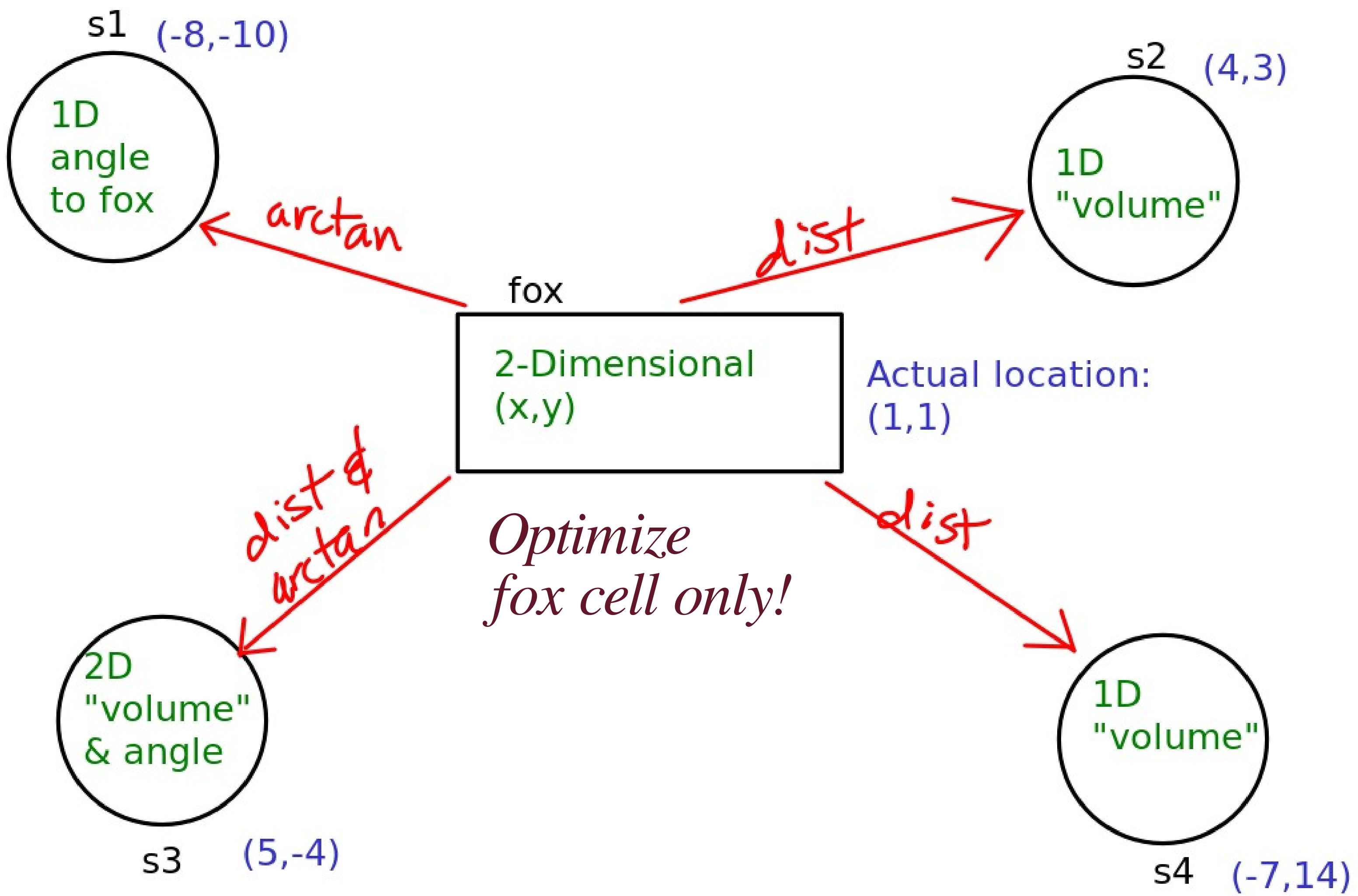
“Fox” Hunting

"Fox" Hunting

"Fox" Hunting

DE
KB1DDS

Our Model



Optimize fox cell only!

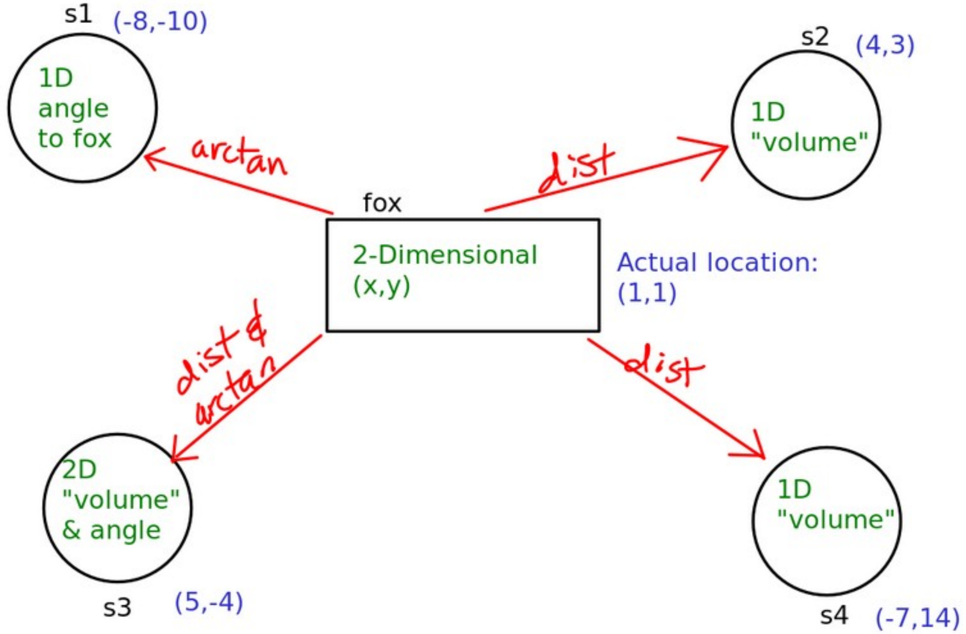
But we have mixed measurements!

- We have distance measurements
- We have angular measurements
- Multiply angular differences by "wt" so we can adjust the relative importance of the two kinds of measurements. (One full circle ~ 6.2 radians.)
- In this run, let's use wt=2.
- This means being off by 45 degrees is worth about the same as being off by 1.57 distance units.
- In practice, this means distances will optimize first and finally the angles will be adjusted once distances are fairly good.

Test it!

We started with initial guess of (20,30)

The actual solution should be (1,1)




```
Starting Consistency Radius: 2.187073633576996
Value at s1 is 1.7701336
Value at s2 is 0.076923076923
Value at s3 is [0.02439 4.49108]
Value at s4 is 0.00429184549
Value at fox is [0.9721613 0.9878475]
Consistency Radius: 0.07199175495712201
Recall: Expected fox solution is at (1,1).
```

Note: In this example, the values held by the cells s1,s2,s3,s4 are not free to change, but they hold angular and distance data, not (x,y) coordinates as shown in our diagram.

CR tells you when you fail!

- Change wt to 20, so angles go crazy on us!
- Still start at (20,30).

Far from solution!

```
Starting Consistency Radius: 21.85639859380892
Value at s1 is 17.701336
Value at s2 is 0.076923076923
Value at s3 is [2.43900e-02 4.49108e+01]
Value at s4 is 0.00429184549
Value at fox is [467089.07990715 256790.3712907 ]
Consistency Radius: 35.68650443002026
Recall: Expected fox solution is at (1,1).
```

Yes, these are correct, b/c they're weighted!

CR warns us not to trust this "optimal" solution!

For Practice (Later)

- Work with a partner if you can
- Invert the problem using the same code, but hide the fox somewhere else.
- (Change which cells are allowed to optimize!)
- Find the new “volume” and angular measurements
- Give these clues to your partner and let them discover where you have hidden the fox!

Main concepts thus far:

- Your sheaf is built on a directed acyclic graph.
- Nodes contain sets, and functions take sets to sets along your arcs/arrows.
- Your assignment is one way to satisfy your constraints; each cell has a set element on it as its data.
- Some data act as constraints, but other data can be optimized.
- Forward and inverse problems will use same sheaf.
- Some data are "baked-in" on the arcs.
- The sheaf holds the current data and functions, as well as the predictions over which distances are found for calculating the CR.