

# Monte Carlo, Revisited


Derrick W. Turk

Director's cut! With narration  
by the author.



# Great expectations...

“We’ve run a Monte Carlo simulation – modeling **each independent block of development** – and we have good news. If we run an extensive pilot program to **de-risk the asset**, the **P50 outcome** for the Implausible Codename project has an IRR over 20%!”



How many times have you heard – or said – something like this about a probabilistic modeling project? How did that turn out? A few **red** flags?

# How often do you hit that P50?

- Is it about half the time?
- Less than a quarter?
- I have no idea because management always ignores our models?
- Does the P50 even matter?

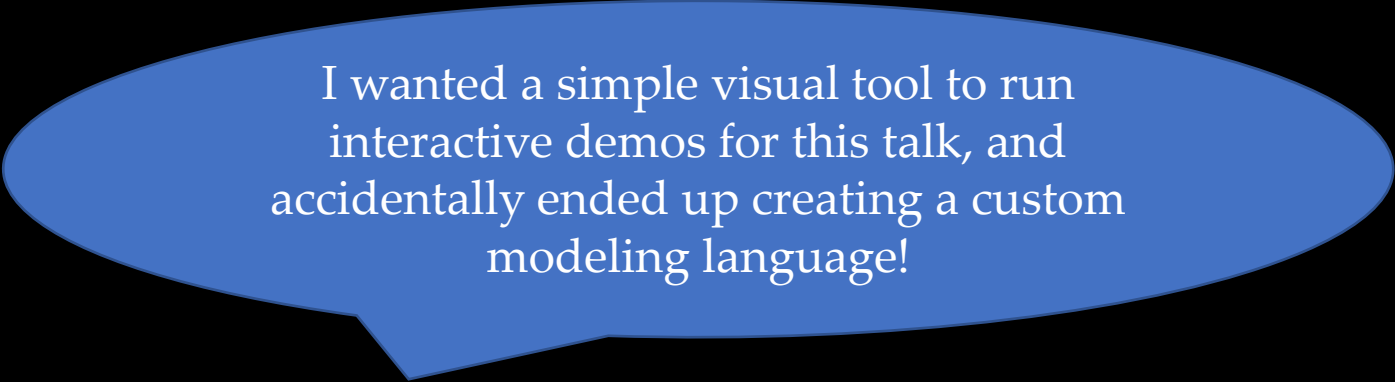
Things that make you say  
“hmmmm....”



That's my son Alex,  
modeling the concept of  
“hmmmm”. He's a  
Monte Carlo skeptic  
like your boss.

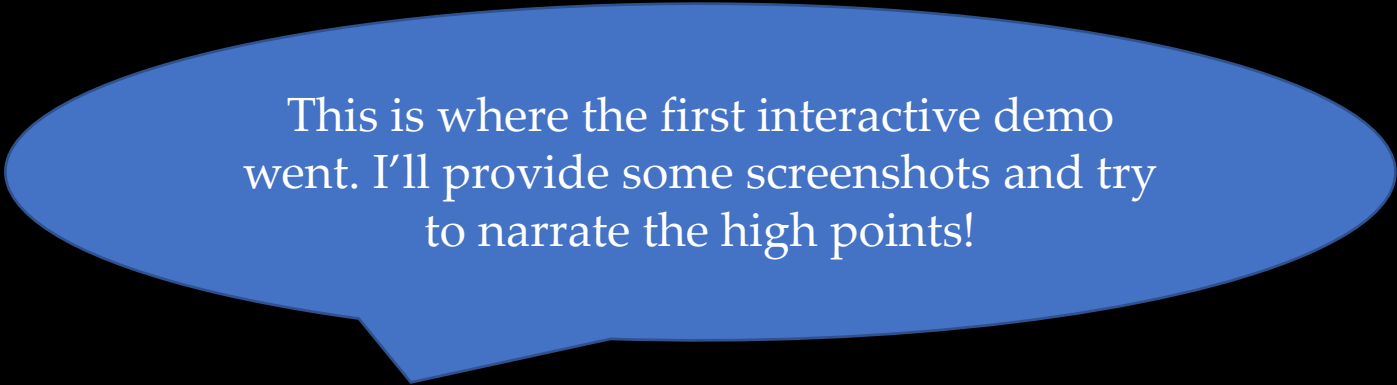
# Let's play

- SPML = “simple” probabilistic modeling language
- We list named variables and define their distributions
  - Constants, like 500.0
  - (uniform lo hi)
  - (normal mean sd)
  - (lognormal mean p10/p90-ratio)
  - Operations on other variables or distributions, like (+ x y) or (- z 17.3)
- We'll see some more later!

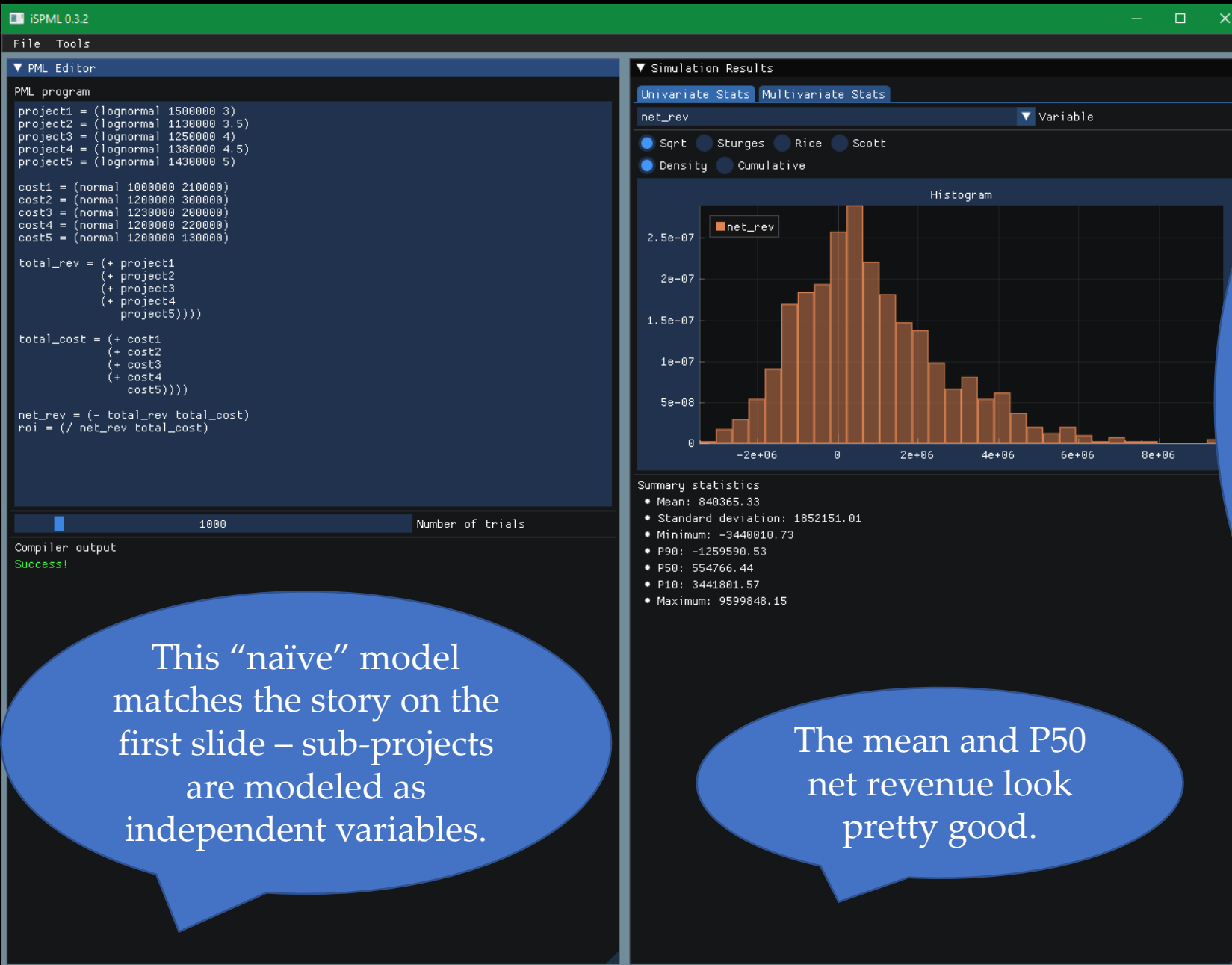


I wanted a simple visual tool to run interactive demos for this talk, and accidentally ended up creating a custom modeling language!

# Model #1 – “Great expectations”



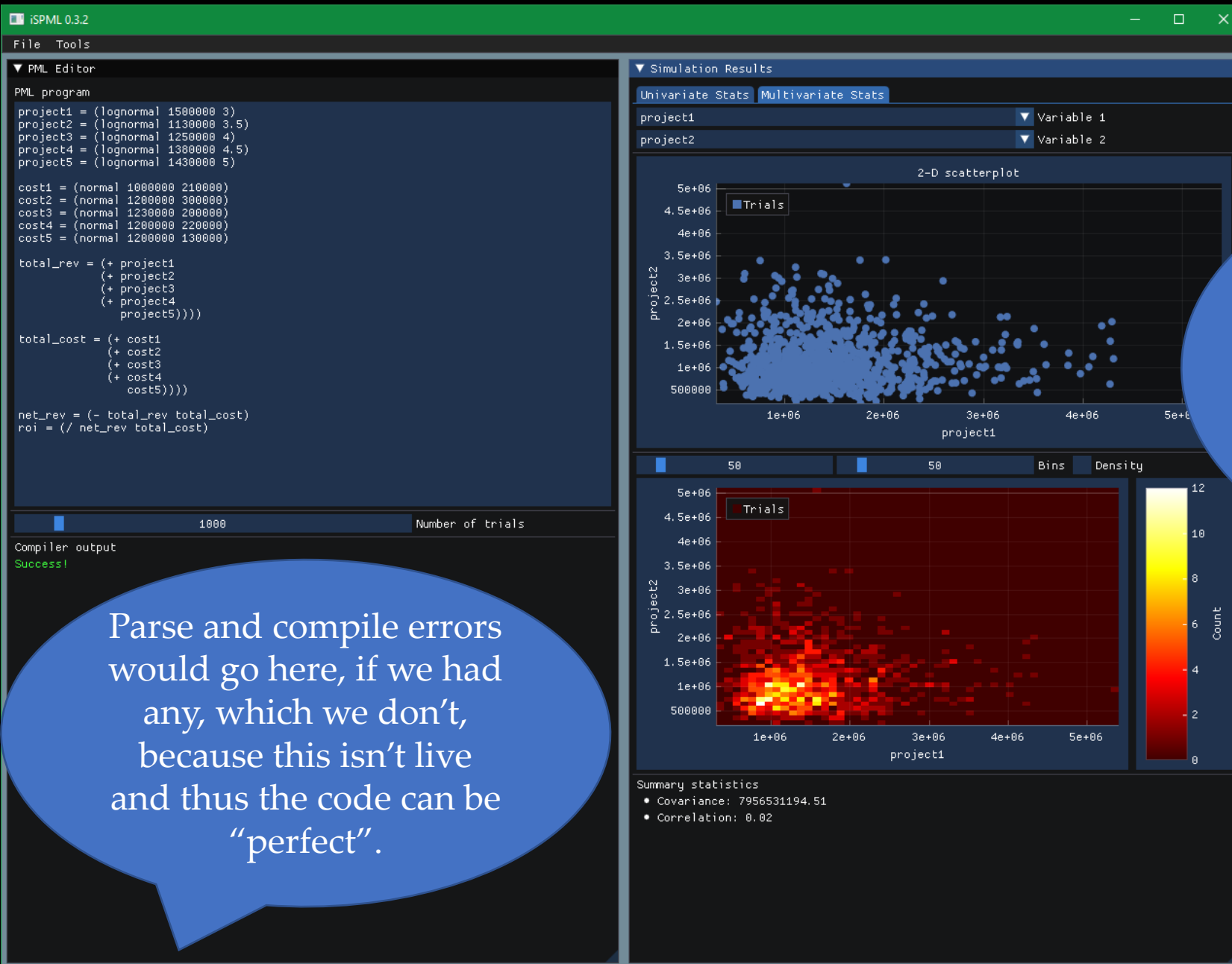
This is where the first interactive demo went. I'll provide some screenshots and try to narrate the high points!



This “naïve” model matches the story on the first slide – sub-projects are modeled as independent variables.

The mean and P50 net revenue look pretty good.

Here we’re visualizing some univariate stats for the net\_rev variable. In our simplified world, the project1-project5 variables stand in for revenue less variable costs, and cost1-cost5 stand in for capex. Sort of – it’s a toy world! We’re trying to keep things simple so we can focus on some key issues.



Here's a view of multivariate stats for project1 and project2. Remember, these are independent, and it shows – no correlation (more on this in a moment).

Parse and compile errors would go here, if we had any, which we don't, because this isn't live and thus the code can be "perfect".



This “ROI” variable is my hacky attempt to quantify return on investment in this toy model – net revenue over capex.

The “ROI” for this model looks great – the P50 is about 10%, and the average is 15%. That’d get us funded in a lot of organizations.



# How can this go wrong?

- Let's assume these are exploration & development projects in a new basin
- If project #1 is a bust, what does that mean for our chances of project #2 succeeding?
- What about for our chances of attempting project #2?

If we're blindfolded and grabbing bits of cereal out of a box, how many berries before we ask if this is "oops, all berries"?



For that matter, if we hate berries, how long do we keep trying this stupid cereal?

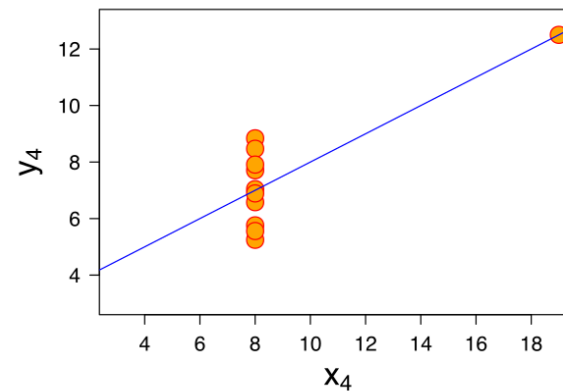
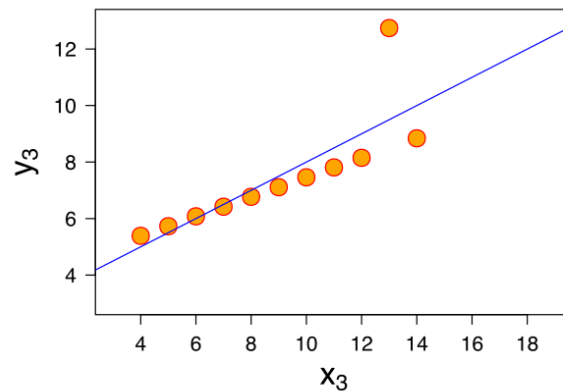
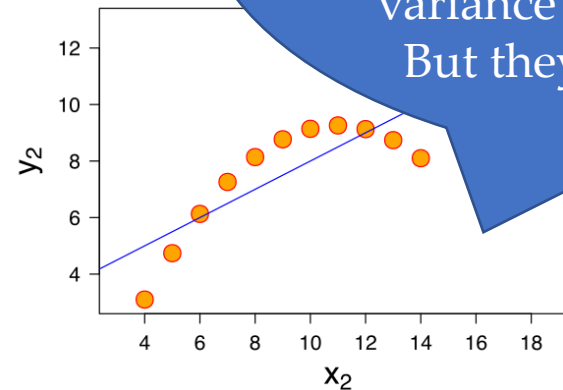
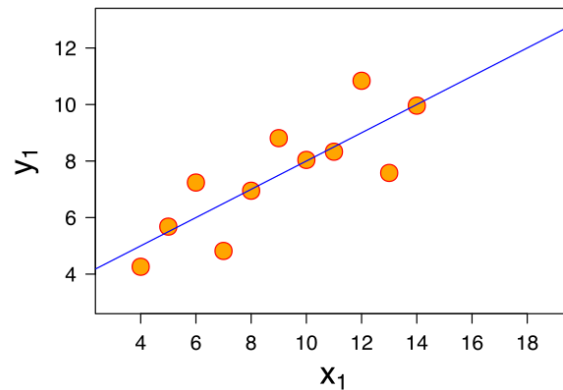
# Covariance and correlation

Nothing to say here!  
Hopefully this is all just  
review material. These  
concepts are very important  
to statistics.

- *Covariance* is exactly what the name suggests – a measure of how two variables tend to vary together (or not)
- Covariance = 0 when variables are *independent* (but how often does that really happen)?
- Covariance  $> 0$  when they tend to vary in the same direction
- Covariance  $< 0$  when they tend to vary in opposite directions
- *Correlation* (well, Pearson's  $\rho$ ) is covariance normalized to fall on an interval from -1 to 1, which makes it a little nicer to work with for modeling

# Danger, danger!

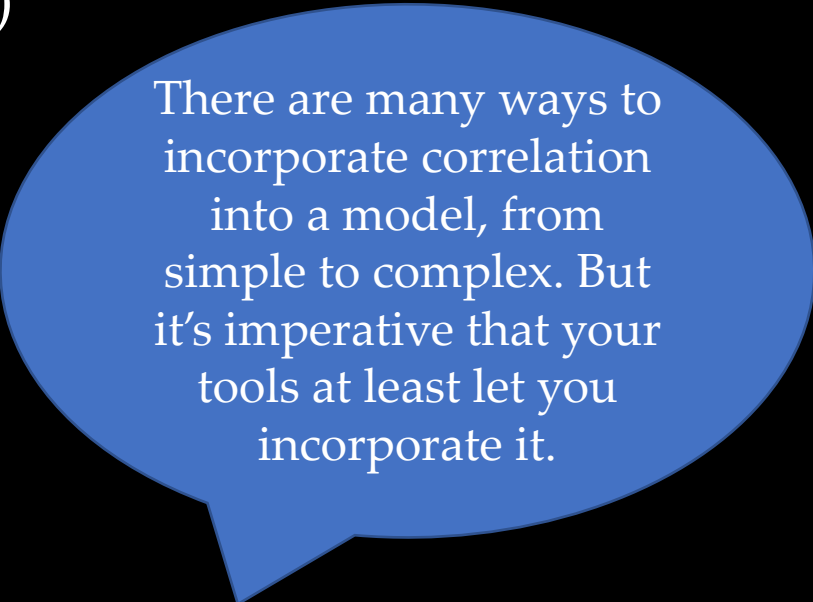
It's outside this scope, but read about the Gaussian copula model and the 2008 financial crisis for a great case study.



We still have to be careful even when we quantify correlation. This is Anscombe's quartet: each data set has the same mean  $X$ , mean  $Y$ , variance of  $X$ , variance of  $Y$  and correlation! But they are very different.

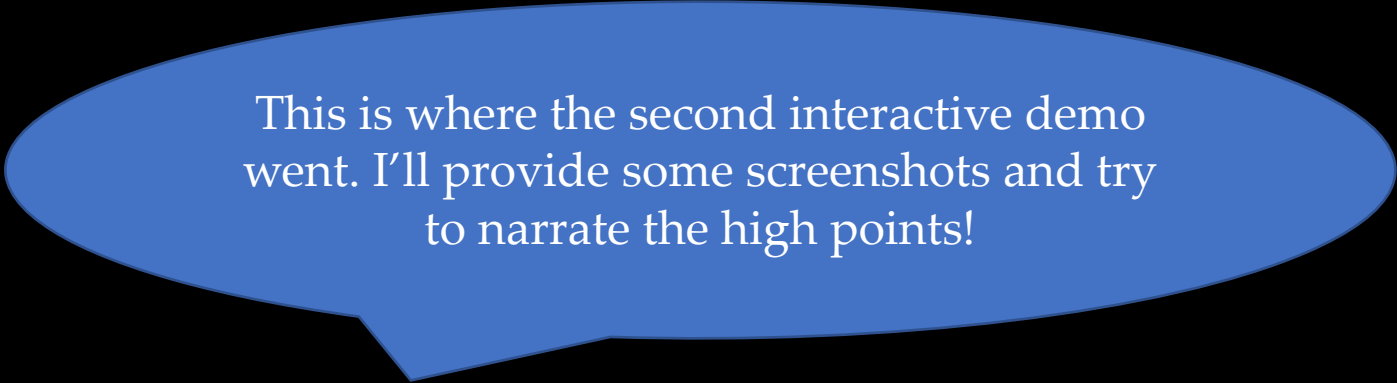
# Simulating with covariance

- Monte Carlo simulation can include variables coupled by a specified covariance (usually in the form of a correlation coefficient)
- In SPML, we can write:
  - (correlate coef some-distribution some-variable)
  - with correlation coef ... end
- If your tools can't do this, get better tools!

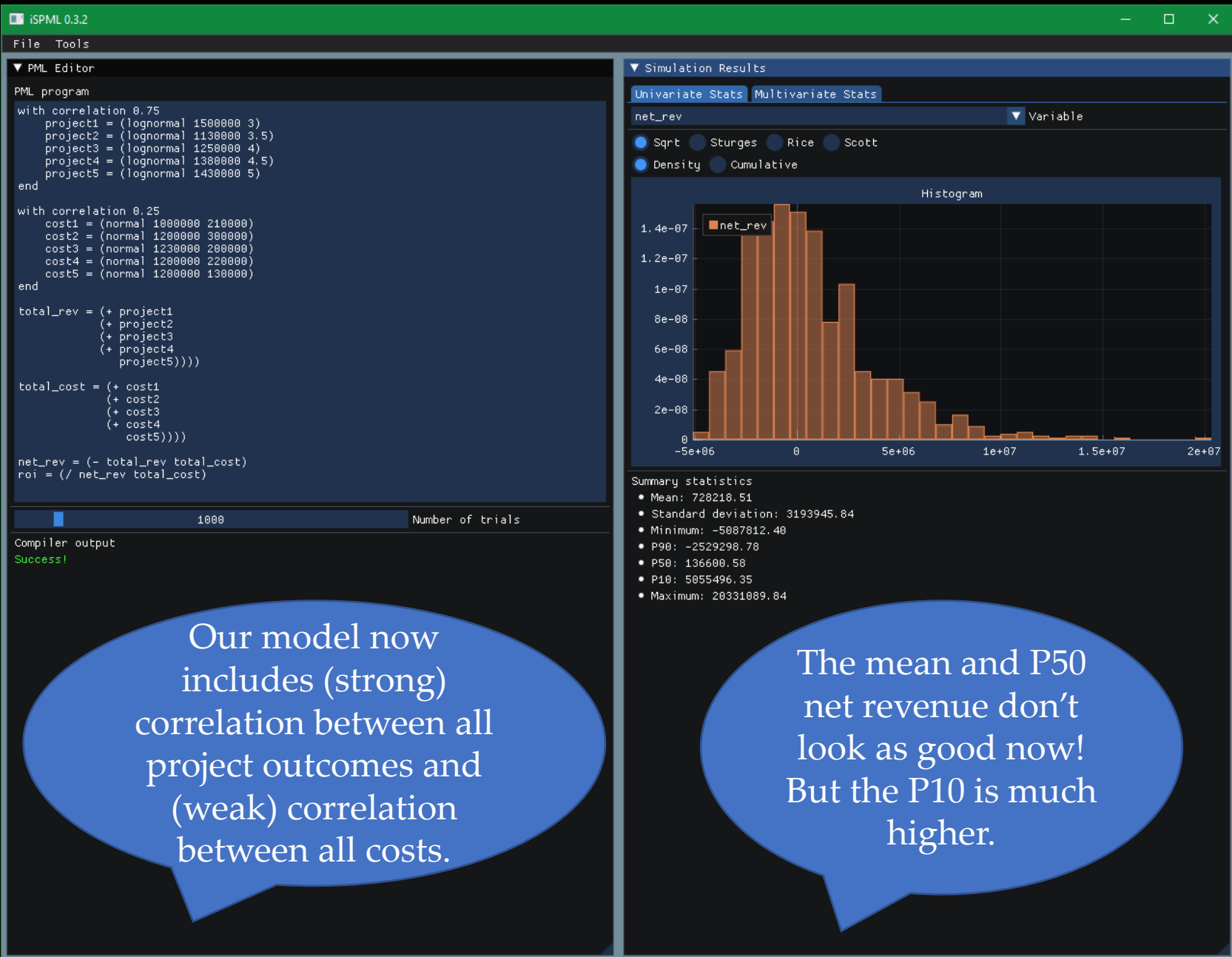


There are many ways to incorporate correlation into a model, from simple to complex. But it's imperative that your tools at least let you incorporate it.

# Model #2 – “Everything is connected”



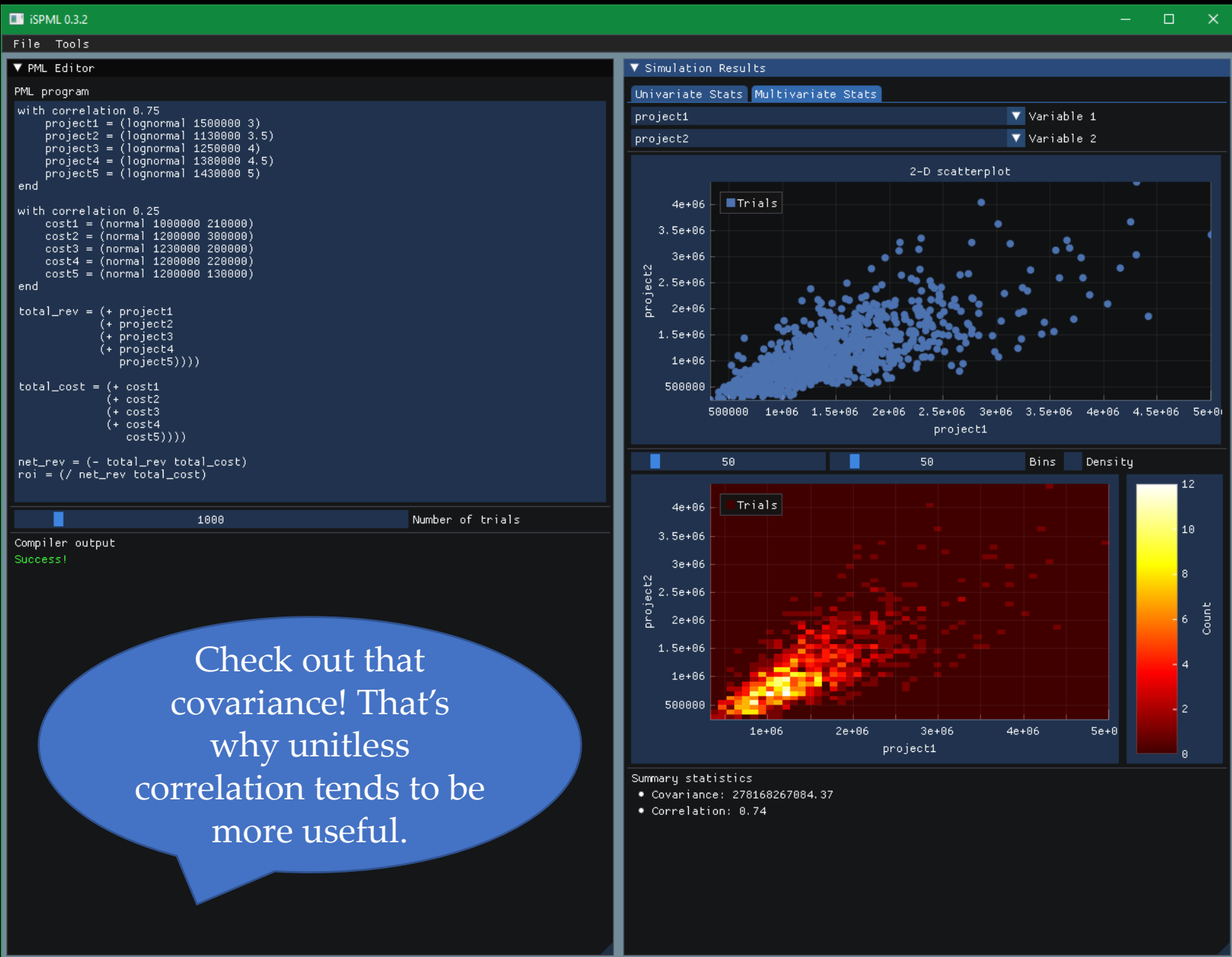
This is where the second interactive demo went. I'll provide some screenshots and try to narrate the high points!



Our model now includes (strong) correlation between all project outcomes and (weak) correlation between all costs.

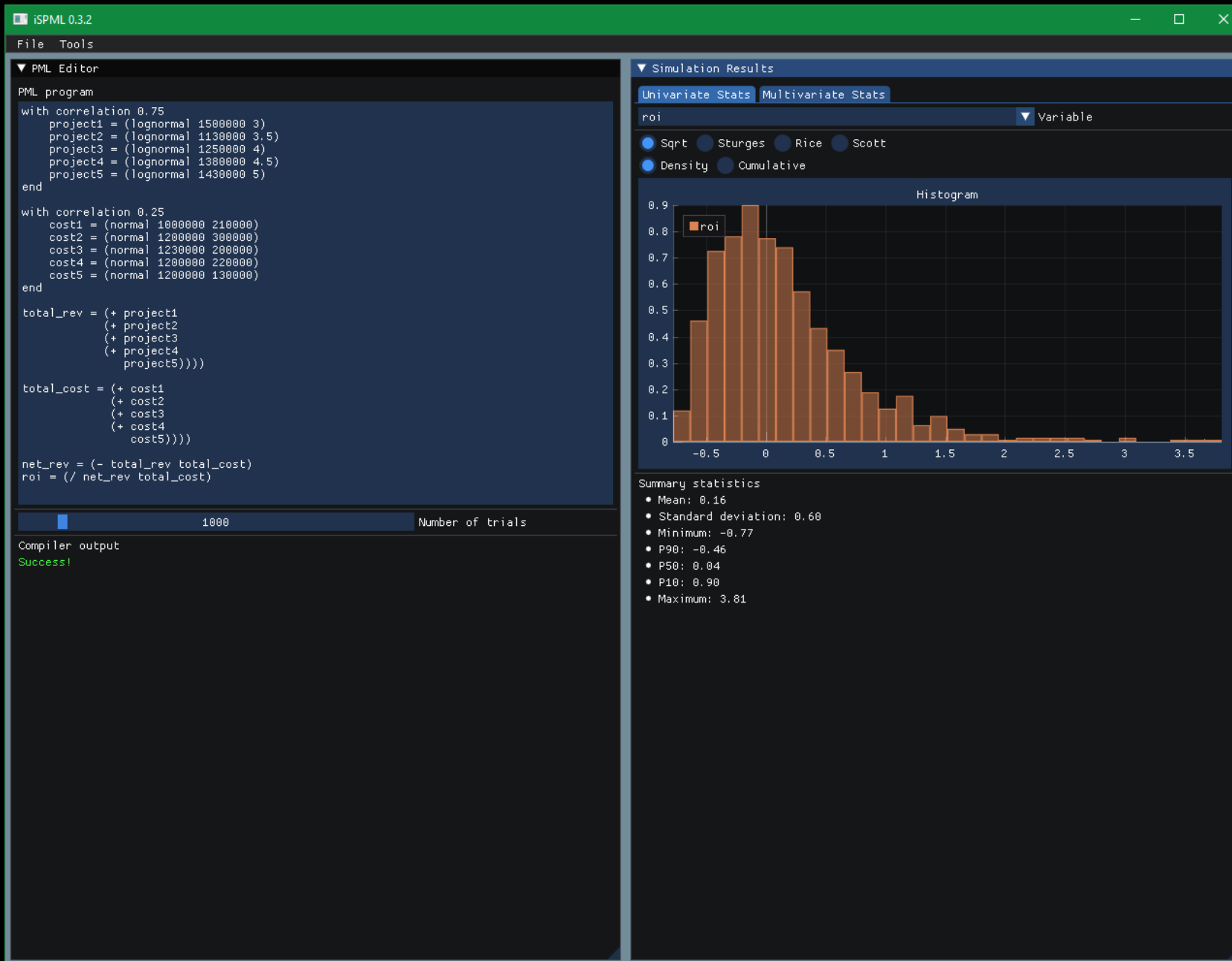
The mean and P50 net revenue don't look as good now! But the P10 is much higher.

This is what (positive) covariance does – it increases the overall variance. The highs are higher and the lows are lower, because they tend to come in flocks now. If we neglect covariance/correlation, our models result in overconfidence around the mean due to understated variance.



Check out that covariance! That's why unitless correlation tends to be more useful.

This is the “nothing up my sleeve” shot – we can see that pairs of projects have correlated outcomes as requested.



Here's our "ROI" –  
much worse than in  
our naïve model.

I sure hope we  
didn't make any  
multi-million dollar  
decisions based on  
that first model...



# Taking the off-ramp

- In any big enough project, we make *decisions* along the way
- Based on what we've found so far, we decide how (or how not) to proceed
- Realistic models can't assume fixed development paths, but need to *model the decisions*!
- The simple case of this is *dependency* between projects

Not much to say here!  
Exploration projects especially usually come with “off ramps” – and sometimes “on ramps”.



# Simulating decisions

“cond” is like an “if statement”:

$v = (\text{cond } x \ y \ z)$

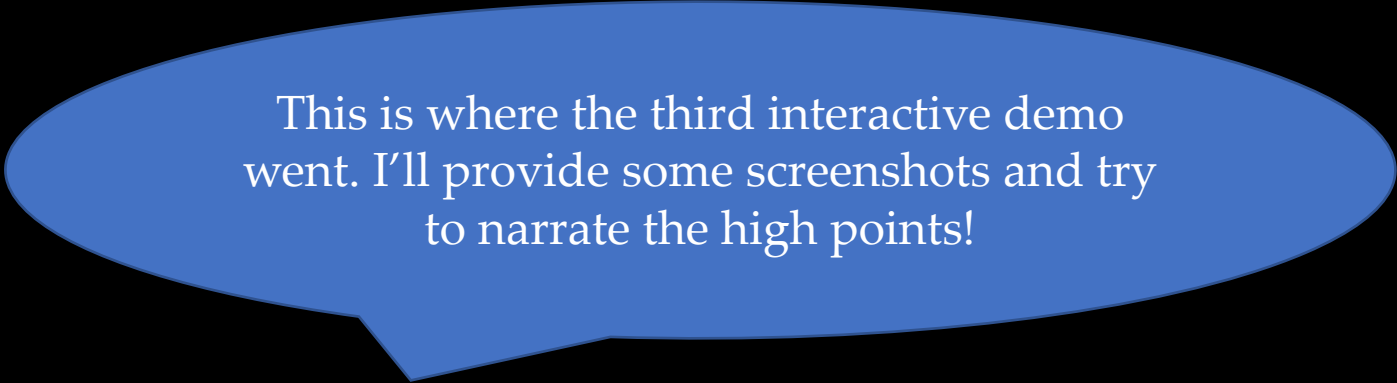
means

if  $(x \neq 0)$  {  $v = y$  } else {  $v = z$  }

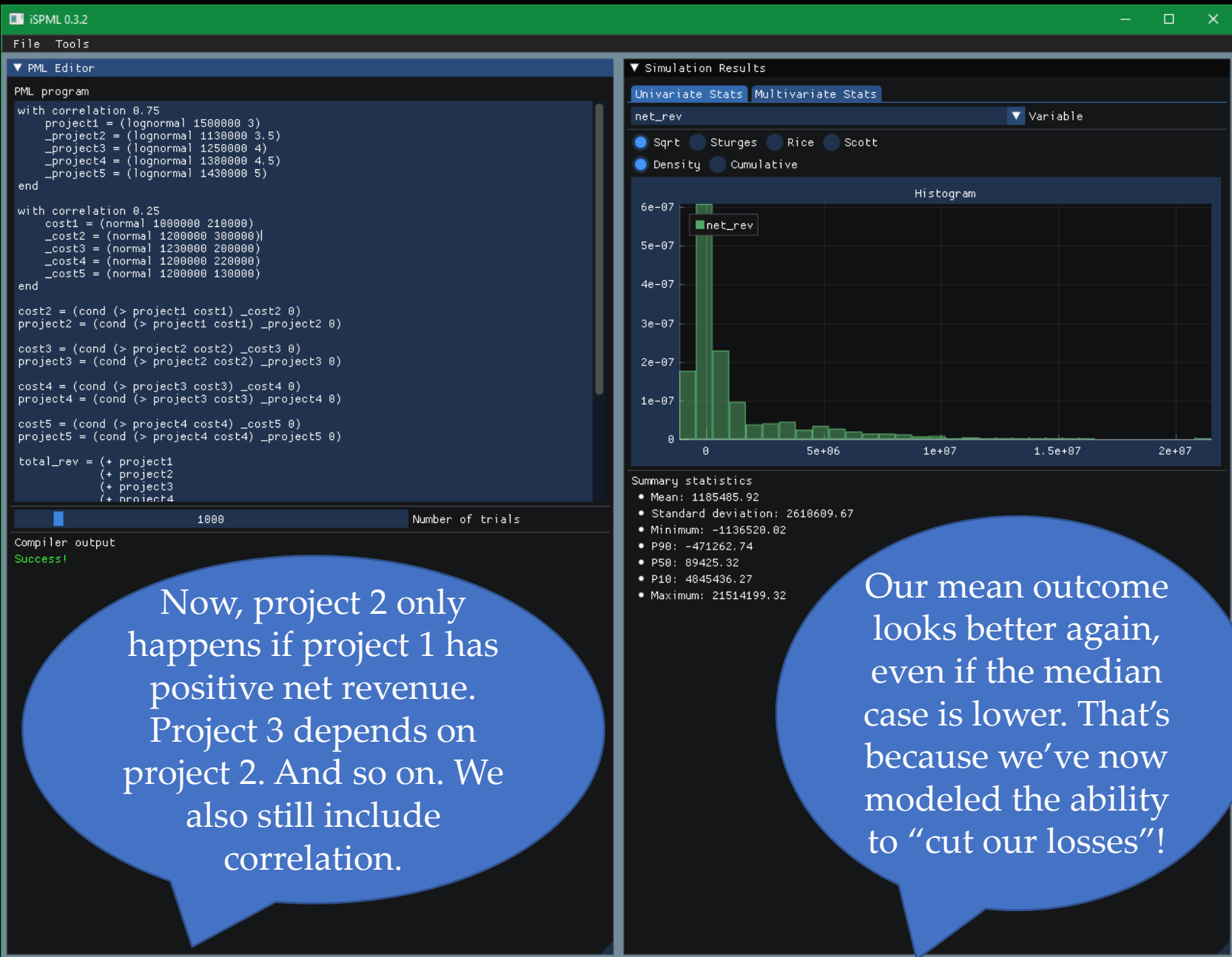
- Some specialized tools can model this directly
- You can probably hack something together in other tools
- In SPML, we can write:
  - $(< \text{some-variable some-other-variable})$  [or  $>$ ,  $\leq$ ,  $\geq$ ], which will evaluate to 0 or 1 in each realization
  - $(\text{cond some-0-or-1-variable some-variable some-variable})$

The fundamental constraint on SPML's design was “what one guy with a three-month old baby can write in a weekend”. For that reason, the syntax is easier for a computer to parse than for a human to write, and the type system is trivial – everything's a floating point number.

# Model #3 – “Everything is connected”



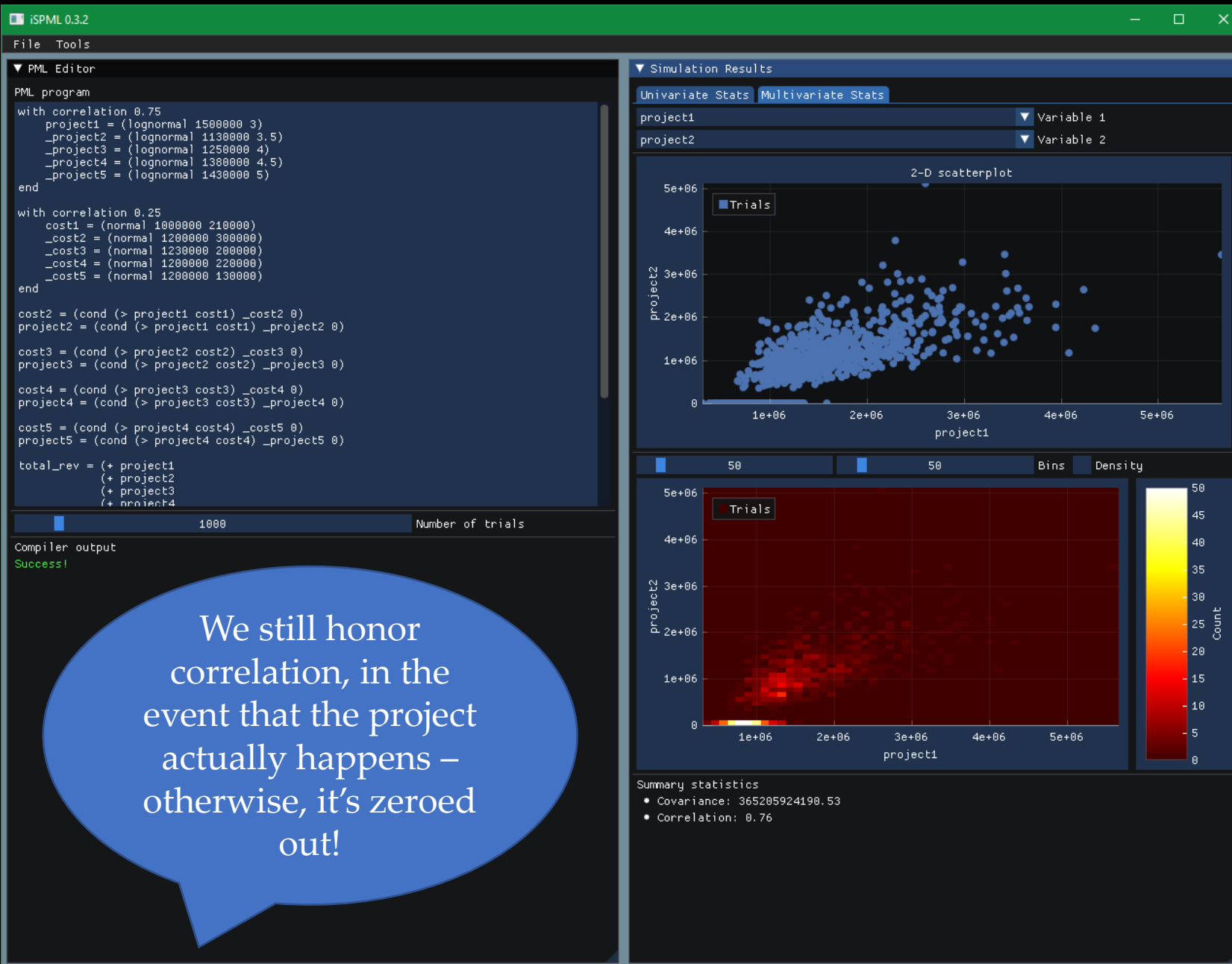
This is where the third interactive demo went. I'll provide some screenshots and try to narrate the high points!



Now, project 2 only happens if project 1 has positive net revenue. Project 3 depends on project 2. And so on. We also still include correlation.

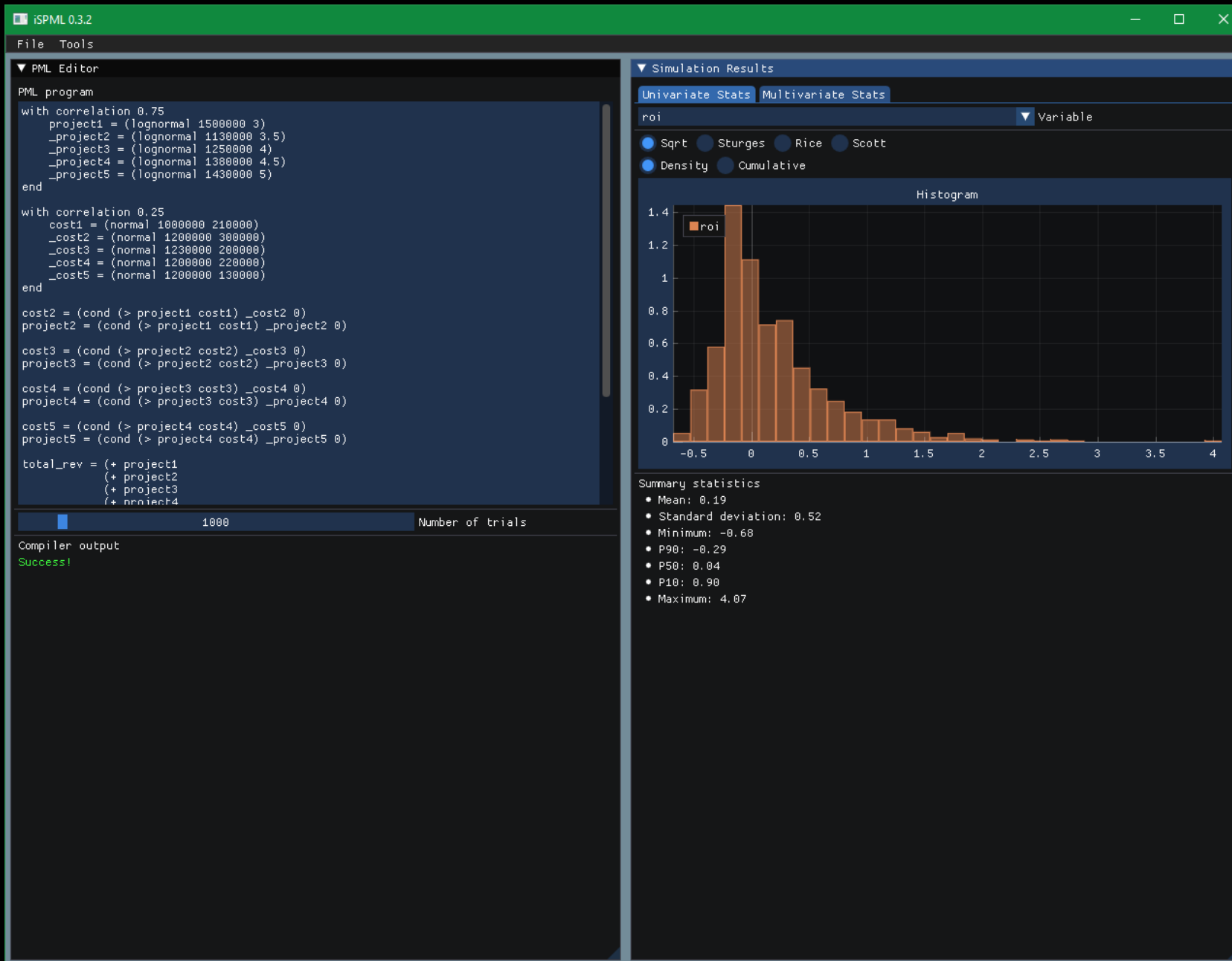
Our mean outcome looks better again, even if the median case is lower. That's because we've now modeled the ability to "cut our losses"!

Our net revenue distribution is now multi-modal, although it's hard to see here. We've got one mode where only project 1 happens, one where we stop after project 2, and so on.



We still honor correlation, in the event that the project actually happens – otherwise, it's zeroed out!

In the heatmap, we can see that the majority of trials don't execute project 2. (That's the "hot streak" at zero on the y axis.)



Obligatory “ROI” shot! We’ve really limited our downside and improved the mean, even if the median didn’t move much.

We’re now approaching a respectable model for guiding decision making.

# Would you like to play a game?

You can try this out now, even on your phone (probably, I don't speak Android)



BRIDGE  
CHECKERS  
CHESS  
POKER  
FIGHTER COMBAT  
GUERRILLA ENGAGEMENT  
DESERT WARFARE  
AIR-TO-GROUND ACTIONS  
THEATERWIDE TACTICAL WARFARE  
THEATERWIDE BIOTOXIC AND CHEMICAL WARFARE  
GLOBAL THERMONUCLEAR WAR

XR

I'm not going to dump screenshots here – go play the game! Try to pay attention to the value (difference in mean payout) of information as we pay for more “science”.

<https://apps.terminusdatascience.com/voigame/game>



# Asking useful questions

- “That’s nice”, your boss says
- “I’m really glad your pee fifty is so... probable... or whatever”
- “How does this help me make a decision?”

“Where is the wisdom we have  
lost in knowledge?  
Where is the knowledge we have  
lost in information?”  
- T. S. Eliot, “The Rock”



Alex is a natural in  
the role of  
“unimpressed boss”.

To me, this is the  
key: how does our  
model help us make  
*better decisions?*



# Useful questions

Nothing to add here!

- Some useful questions are about *probability of achievement*: “what is the chance that we achieve consistent year-over-year BOE growth over the next 5 years?”
- Some useful questions are about *risk*, which can sort of be probability of achievement’s evil twin: “what is the chance that our net loss exceeds 30 million dollars?”
- Even when useful questions can be answered by simple summary statistics, they’re not often on any variable which directly appears in the model: “How many years until we break even, on average? Best case? Worst case?”

# Tricking computers into answering them

- Direct support for these kinds of queries is rare
- Sometimes we can trick them, though
- Remember SPML's "Boolean" operators (<, <=, >, >=)?
- Consider: `pays_out = (> gross_revenue expenses)`
  - `pays_out = 1.0` if `gross_revenue > expenses`, `0.0` otherwise
  - So the mean of `pays_out` is the probability of achievement!
  - The P90, P50, P10 can be interpreted as "does the P90 [etc.] outcome pay out?"

It's fun and useful to find ways to  
smuggle more interesting computation  
into seemingly simple systems.

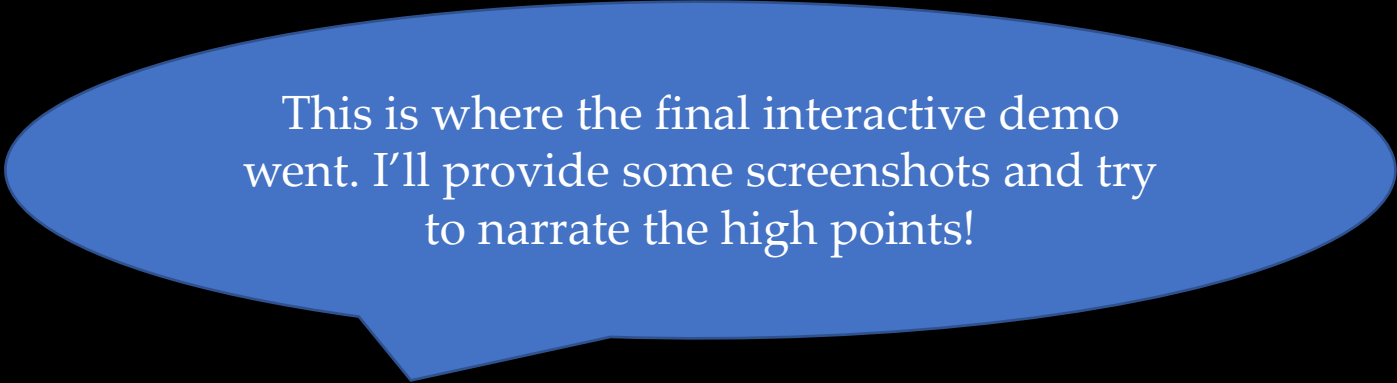
"The street finds its own  
uses for things"  
– William Gibson

# Tricking computers II: type derangement

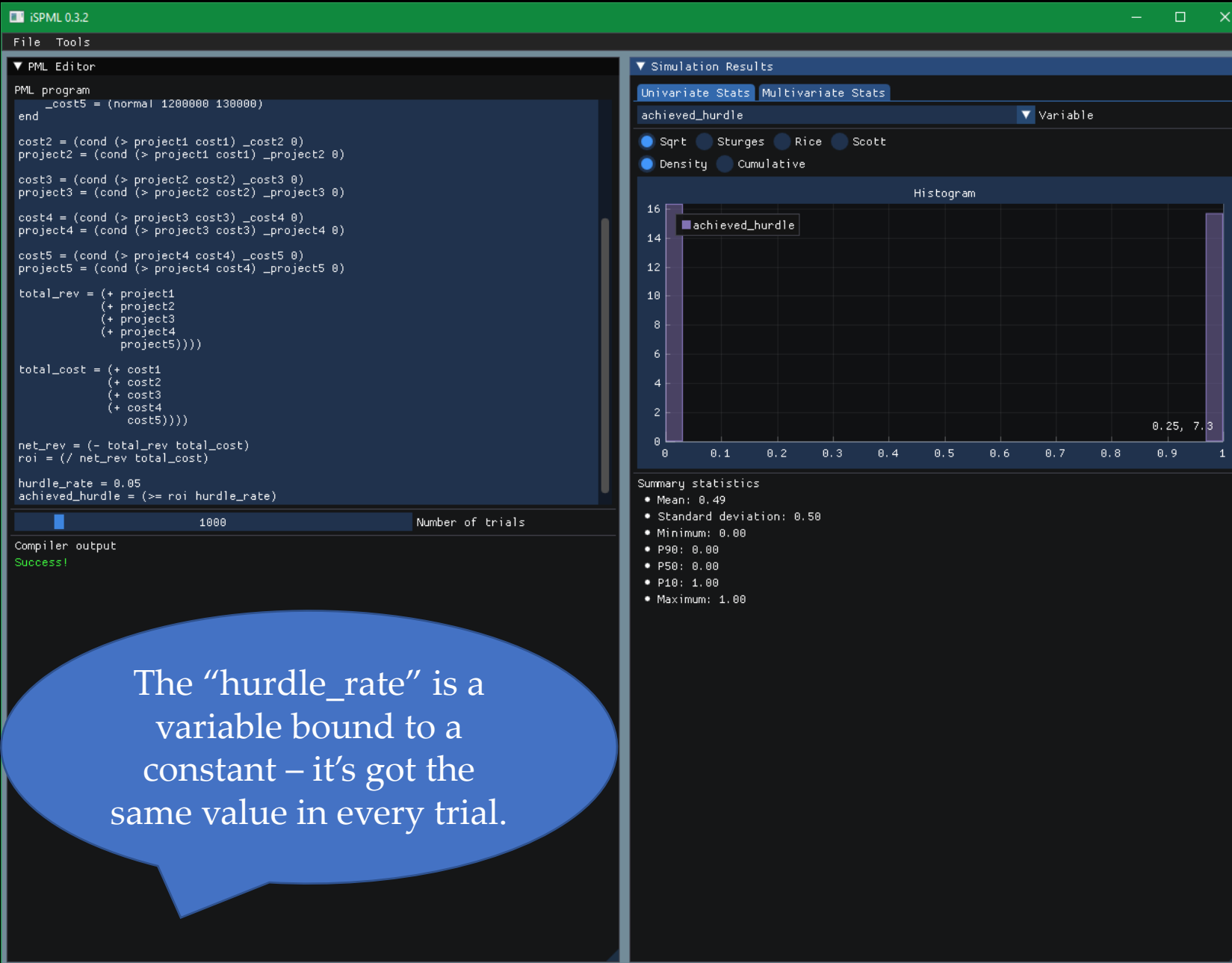
- In fact, we can “pun” all sorts of Boolean operations into floating-precision math, as long as we always have  $\text{true} = 1.0$  and  $\text{false} = 0.0$
- “and” =  $*$
- “or” =  $+$  [kind of...]
- “not” =  $x \rightarrow (1.0 - x)$  [unless you used  $+$  as or!]

To elaborate, “or” = “+” works as long as anything non-zero is “true”. But “not x” =  $(1.0 - x)$  only works if “true” is 1.0 exactly. These are cheap hacks – you get what you pay for!

# Model playtime – let's build some metrics



This is where the final interactive demo went. I'll provide some screenshots and try to narrate the high points!



The “hurdle\_rate” is a variable bound to a constant – it’s got the same value in every trial.

Really, there wasn’t much to say with a model this simple. We can compute metrics like “achieved\_hurdle”...

...and interpret the summary statistics. Looks like a 49% chance of achieving our target ROI.

That's as far as we got in the hour we had! If you'd like to keep reading, you'll find an abridged version of a case study about adding this "probability of achievement" capability to an existing, not very flexible, Monte Carlo tool. This was accomplished with a custom domain-specific programming language (notice a theme?).

I'm not going to annotate these slides, but if you're interested in more detail than is available here, please follow the link in the next slide to the original talk from 2017 from which this mini-study is excerpted.

Thanks to the Houston chapter of SPEE for hosting the original session of this talk, and thank you for reading! Please look us up at <https://terminusdatascience.com> and feel free to write [dwt@terminusdatascience.com](mailto:dwt@terminusdatascience.com) with any questions.

- Derrick

# But I already have a fancy simulation tool!

- (And it doesn't let me do that.)
- That's fine: where there's a will, there's a way\* (as long as you can get the raw trials out)
- Here's a short case study involving a “bolt-on” system for answering these kinds of questions
- This is super abridged! See:  
<https://github.com/derrickturk/dsl-talk>

\* For problems solvable\*\* by Turing machines

\*\* Although you may need to wait a few trillion years

# Domain-Specific Languages

- Let users express their problems in their vocabulary
  - Give them simple, composable parts (“words”)
  - And let them build up larger systems “in their own words”
- 
- It’s on us to figure out how to turn these domain-specific programs into efficient “solutions”
  - That is: parsing, interpreting or compiling, executing



# Case Study: (SC)PML

(Some Client's) Portfolio Model Language

- The client has built a sophisticated set of teams, processes, and technology for modeling, managing, and optimizing the corporate portfolio
- Fully probabilistic characterizations of available opportunities: timing, dependencies, cashflows
- Optimized for expected NPV under constraint and environment “scenarios”
- See SPE-187162-MS

# Case Study: (SC)PML

- We've also built a high-performance post-processor for Monte Carlo simulation
- Run the “optimized” selections under many probabilistic realizations (based on modeled uncertainty)
- Report mean, P90, P50, P10 for each cashflow item (e.g. total capital, gross oil...)

# Case Study: (SC)PML

This worked well, but interesting questions look less like

“what’s the P90 total oil production in 2018 for the Permian?”

and more like

“what’s the probability of achieving our cashflow targets, while also staying within the desired capital budget?”

In fact, there are an infinite number of these interesting questions... and the programmer is the worst person of all to try to guess them up front.

# Case Study: (SC)PML

- My solution: a simple expression language for describing how to aggregate and summarize cashflows and collect statistics from them
- It allows arithmetic operations, discounting and rate-of-return calculations, cumulation over time, and simple logical comparisons (e.g. “does this value exceed this threshold?”)
- Minimalist syntax (s-expressions) based on Lisp
- Composable: operators nest in the expected way (“cumulative of total of NPV of difference of ...”)
- Trivially type-safe: everything’s a number
- Easy to parse: prefix notation and (lisp style forms) remove ambiguity from the grammar (no PEMDAS hacks)

# Case Study: (SC)PML

```
(statistics
  (summary (by-attribute [Region]) each-year
    (total [Wells Per Year])
    (total [Net BOE Volume Mboe])
    (total [Net Lifting LOE M$])
    (total (+ [Net Capital M$]
      [Net ITS M$]
      [Net Capitalized OH M$]
      [Net Leasehold M$]
      [Net Seismic M$]))
    as [Net Total Capital M$]
    (total [Net ATCF 0 M$])
    (cumulative (total (discount 0.10 [Net ATCF 0 M$]))))
    as [Cumulative Net ATCF 10 M$]
  )

  mean
  (percentile 90)
  (percentile 50)
  (percentile 10)
)
```

First, we replicated existing  
(static reports) functionality in  
(SC)PML...

# Case Study: (SC)PML

```
(statistics
  (summary all-groups all-times
    (and
      (>= (total [Net ATCF 0 M$])
        (average (* { 5 7 9 10 15 7 5 } 1000000)))
      (>= (total [Net oil volume Mbbls])
        1000000))
    )

  mean
  standard-deviation
  (percentile 90)
  (percentile 50)
  (percentile 10)
)
```

... but then we started building tools for ad-hoc investigations and “probability of achievement” queries.

# Example: year/year production growth

- We're curious about the chance to achieve consistent year-over-year production (Net BOE) growth, say over the next 5 years for the Permian Region.
- Some metrics we may want:
  - % change in total Net BOE year-to-year for first five years
  - probability of positive % change in each year
  - probability of positive % change every year over five years

# Posing the questions

```
(statistics
(summary (by-attribute [Region]) all-times

(* (/ (- (total (in-year 1 [Net BOE Volume Mboe]))
            (total (in-year 0 [Net BOE Volume Mboe])))
      (total (in-year 0 [Net BOE Volume Mboe])))
  100)
as [0->1 YOY growth %]

(* (/ (- (total (in-year 2 [Net BOE Volume Mboe]))
            (total (in-year 1 [Net BOE Volume Mboe])))
      (total (in-year 1 [Net BOE Volume Mboe])))
  100)
as [1->2 YOY growth %]

(* (/ (- (total (in-year 3 [Net BOE Volume Mboe]))
            (total (in-year 2 [Net BOE Volume Mboe])))
      (total (in-year 2 [Net BOE Volume Mboe])))
  100)
as [2->3 YOY growth %]

(* (/ (- (total (in-year 4 [Net BOE Volume Mboe]))
            (total (in-year 3 [Net BOE Volume Mboe])))
      (total (in-year 3 [Net BOE Volume Mboe])))
  100)
as [3->4 YOY growth %]

(* (/ (- (total (in-year 5 [Net BOE Volume Mboe]))
            (total (in-year 4 [Net BOE Volume Mboe])))
      (total (in-year 4 [Net BOE Volume Mboe])))
  100)
as [4->5 YOY growth %]
```

- Let's write a simple (SC)PML “statistics” program:
- First, we'll specify aggregation by region and over all times
- Then, we'll start defining summary metrics: for each year 1 through 5, find the % change in total Net BOE volume from the previous year to that year.



# Posing the questions

```
(>= (total (in-year 1 [Net BOE Volume Mboe]))
      (total (in-year 0 [Net BOE Volume Mboe])))
  as [0->1 YOY growth achieved]

(>= (total (in-year 2 [Net BOE Volume Mboe]))
      (total (in-year 1 [Net BOE Volume Mboe])))
  as [1->2 YOY growth achieved]

(>= (total (in-year 3 [Net BOE Volume Mboe]))
      (total (in-year 2 [Net BOE Volume Mboe])))
  as [2->3 YOY growth achieved]

(>= (total (in-year 4 [Net BOE Volume Mboe]))
      (total (in-year 3 [Net BOE Volume Mboe])))
  as [3->4 YOY growth achieved]

(>= (total (in-year 5 [Net BOE Volume Mboe]))
      (total (in-year 4 [Net BOE Volume Mboe])))
  as [4->5 YOY growth achieved]

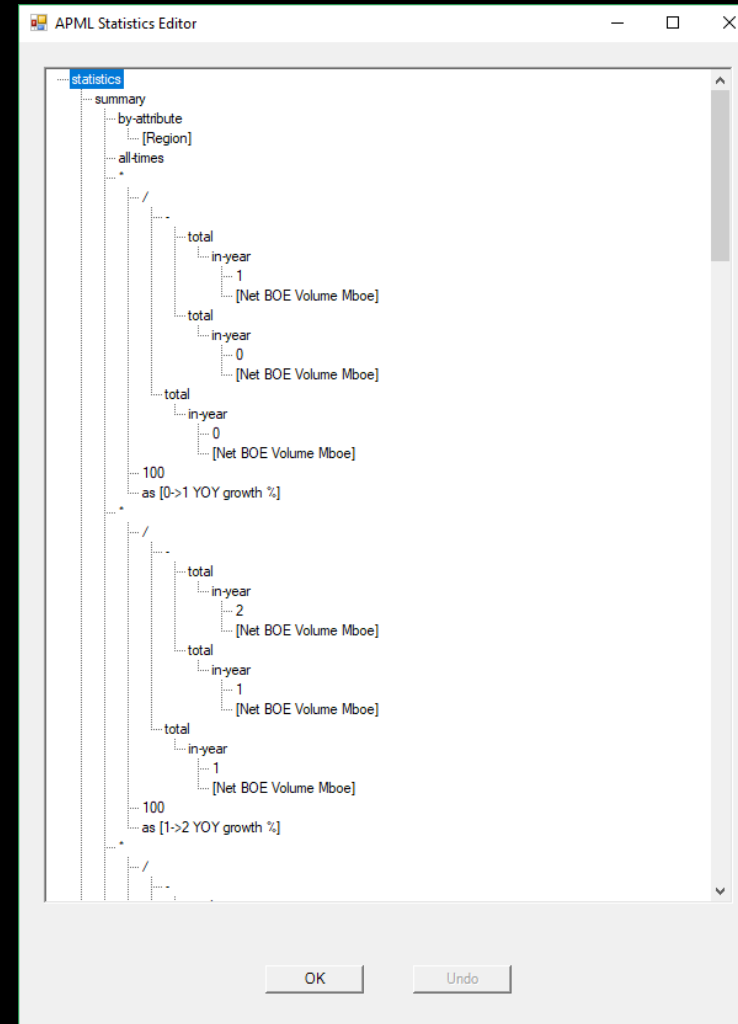
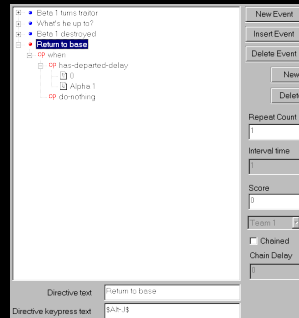
(and (>= (total (in-year 1 [Net BOE Volume Mboe]))
          (total (in-year 0 [Net BOE Volume Mboe])))
      (>= (total (in-year 2 [Net BOE Volume Mboe]))
          (total (in-year 1 [Net BOE Volume Mboe])))
      (>= (total (in-year 3 [Net BOE Volume Mboe]))
          (total (in-year 2 [Net BOE Volume Mboe])))
      (>= (total (in-year 4 [Net BOE Volume Mboe]))
          (total (in-year 3 [Net BOE Volume Mboe])))
      (>= (total (in-year 5 [Net BOE Volume Mboe]))
          (total (in-year 4 [Net BOE Volume Mboe]))))
  as [5-year YOY growth achieved])

mean
(percentile 90)
(percentile 50)
(percentile 10)
)
```

- We'll also capture some logical values (1 or 0) indicating whether the change year-over-year was positive.
- Our final summary metric combines these individual logical values with and to create an indicator for sustained growth in every year.
- Finally, we request the mean, P90, P50, and P10 stats metrics.

# Programs are trees

- We represent them as trees, so that we can manipulate them in our parsers, type checkers, compilers, and so on.
- But we can also let users edit them as trees, turning common GUI components into fancy “structural editors” (look ma, no syntax errors).
- I shamelessly stole this combination of Lispy syntax and a GUI tree editor from a videogame



# Thanks for attending!

