



# **NANOHUB: HOW TO CREATE AND MAINTAIN A SUCCESSFUL NATIONAL HUB**

Michael Zentner, Gerhard Klimeck, Steve Snyder





# **NANOHUB: (LEARNING) HOW TO CREATE AND MAINTAIN A SUCCESSFUL (INTER)NATIONAL HUB**

Michael Zentner, Gerhard Klimeck  
Nathan Denny, Dwight McKay, Steve Snyder





# SITUATION

Execution



Innovation





# SITUATION ❖ TARGET

Execution



Innovation



# SITUATION ❖ TARGET ❖ PLAN

Execution



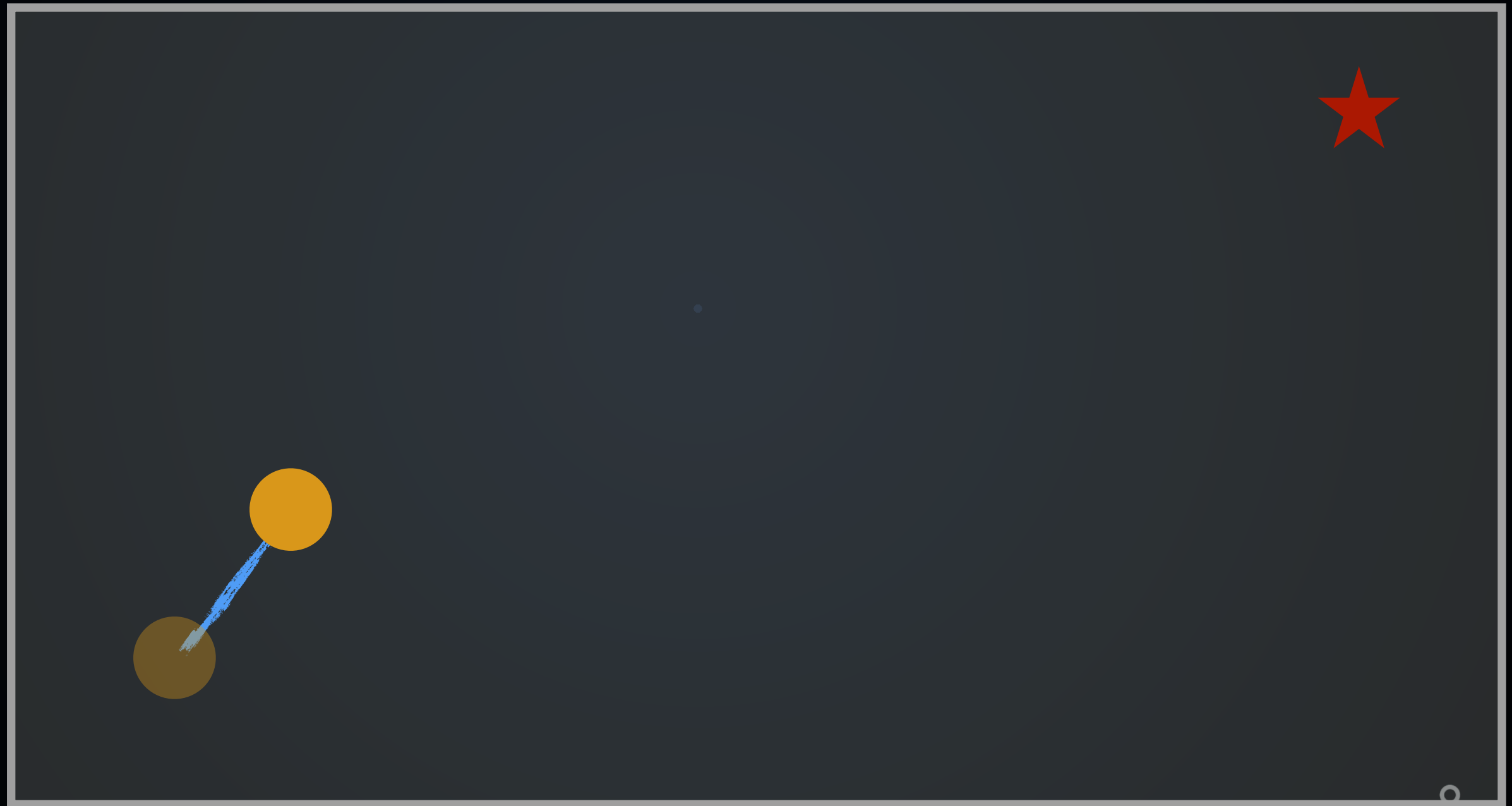
Innovation





# SITUATION ❖ TARGET ❖ RE-PLAN

Execution

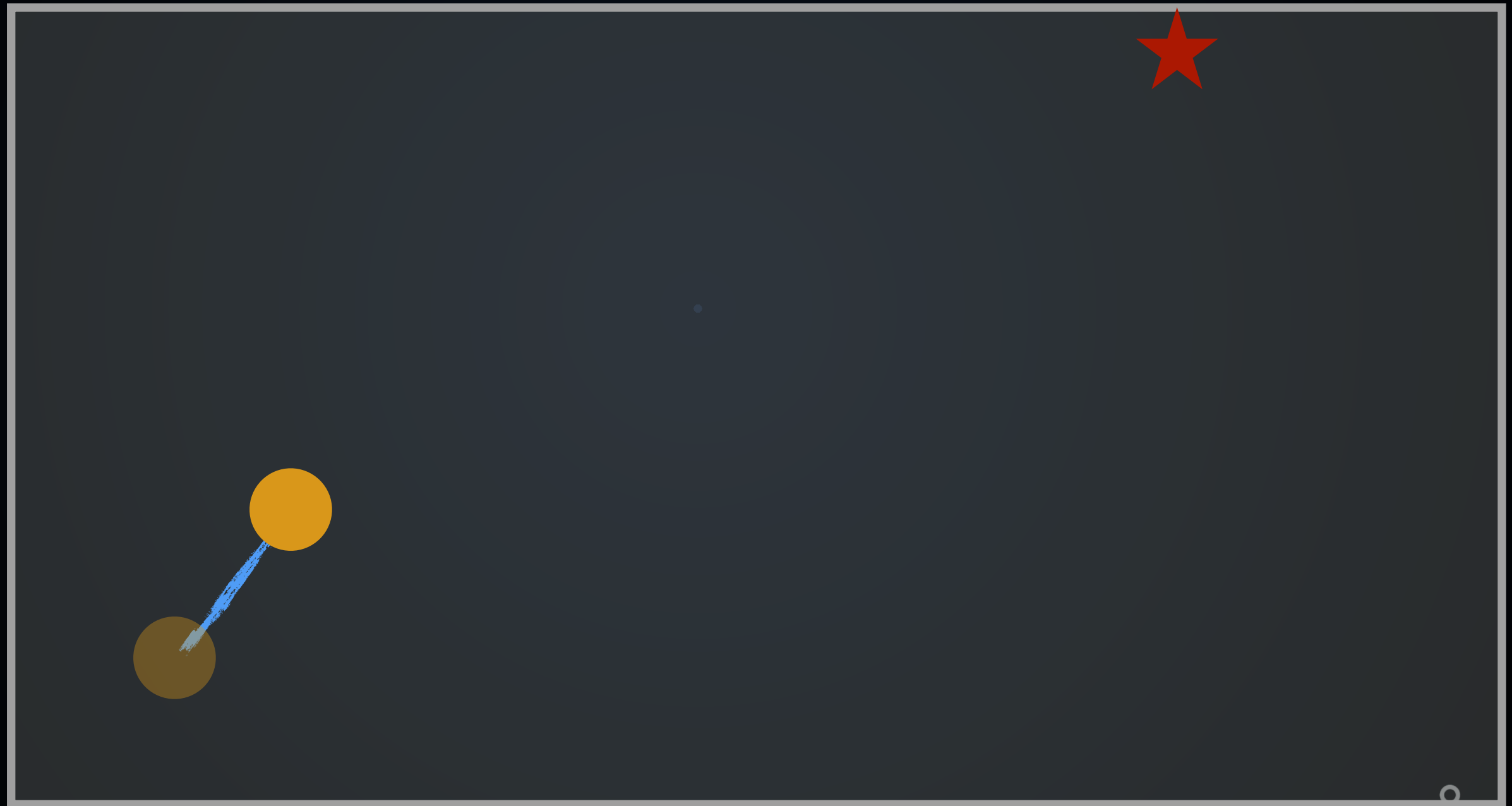


Innovation



# SITUATION ❖ TARGET ❖ RE-PLAN

Execution

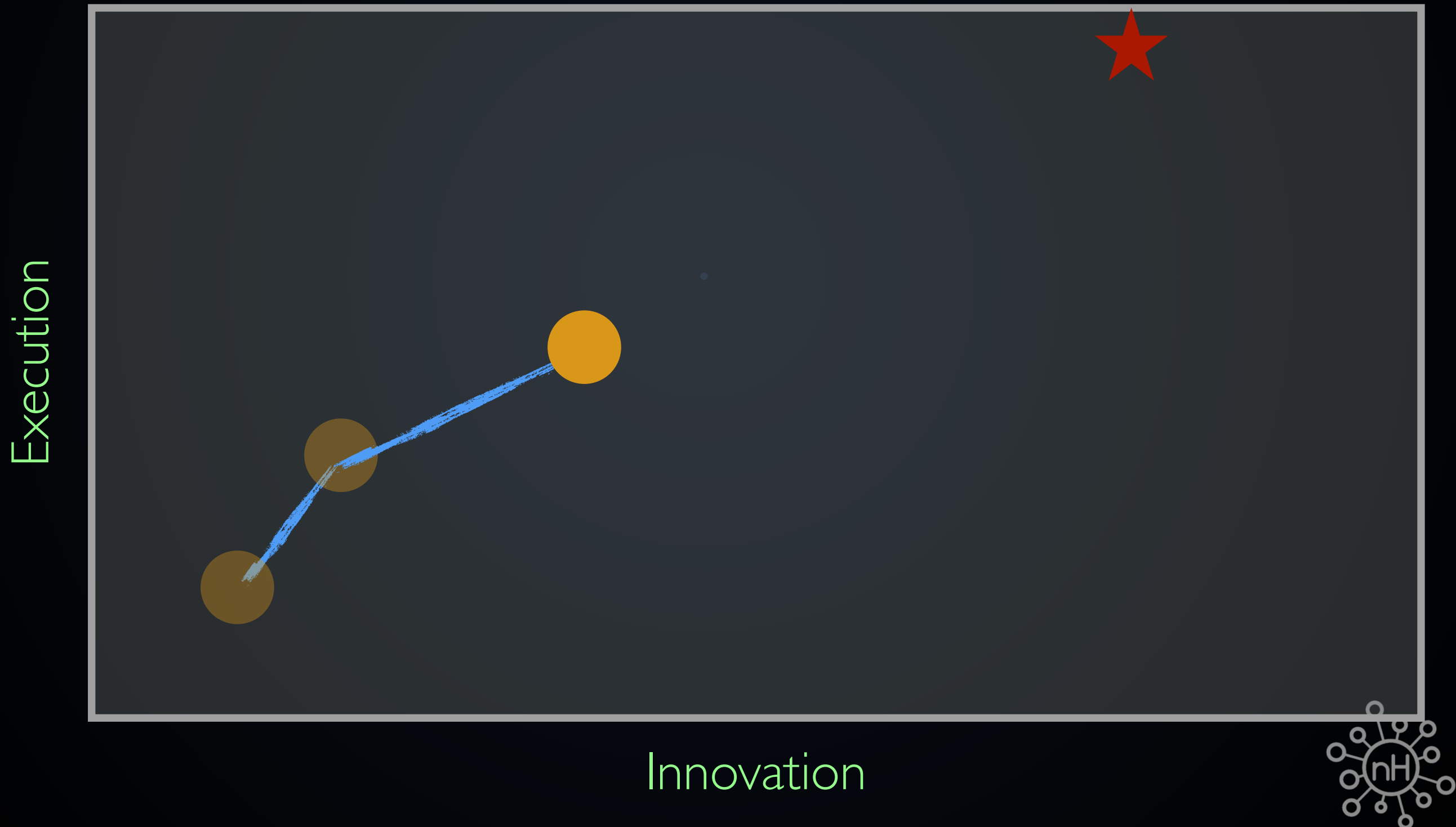


Innovation

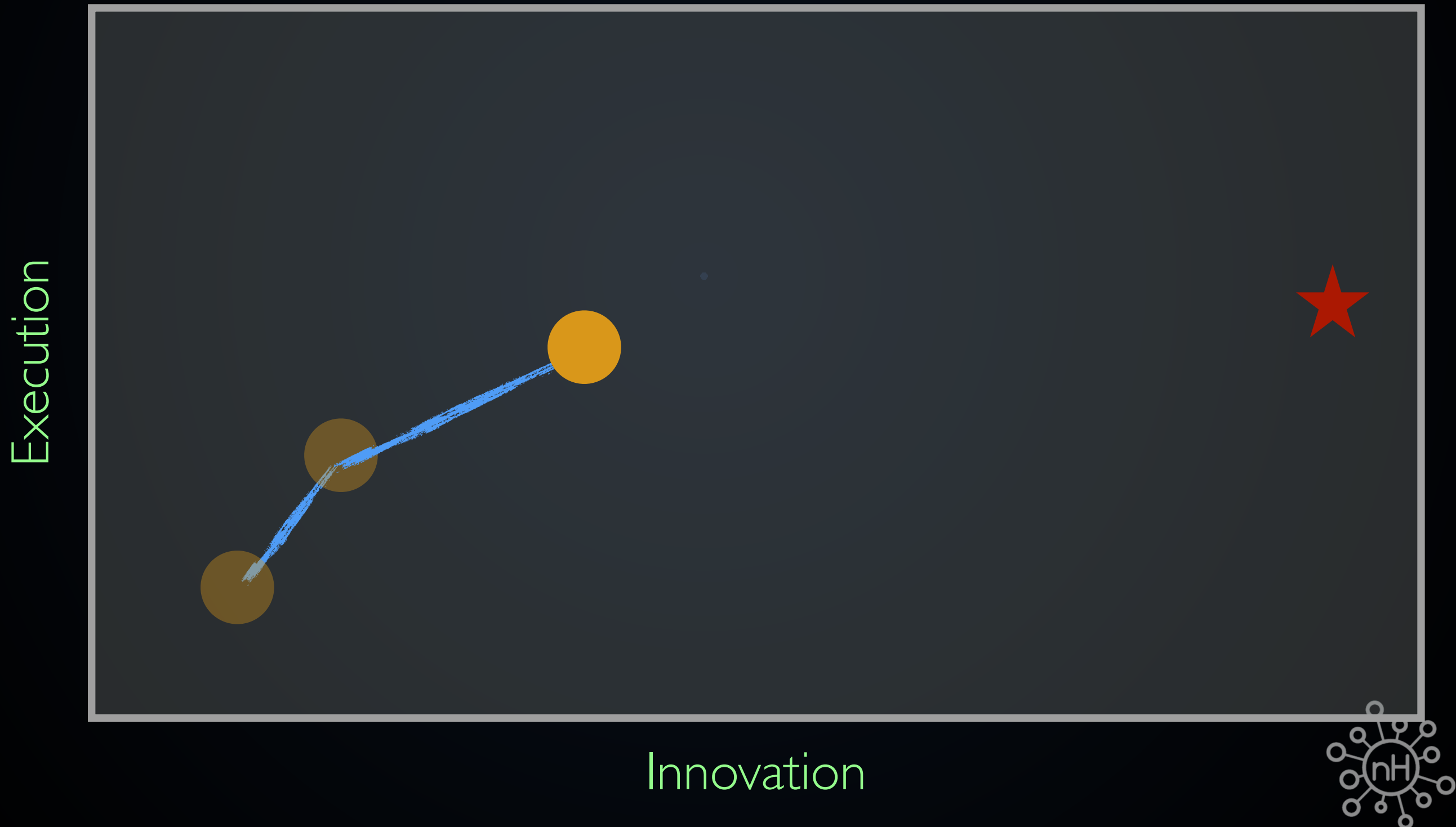




# SITUATION ❖ TARGET ❖ RE-PLAN

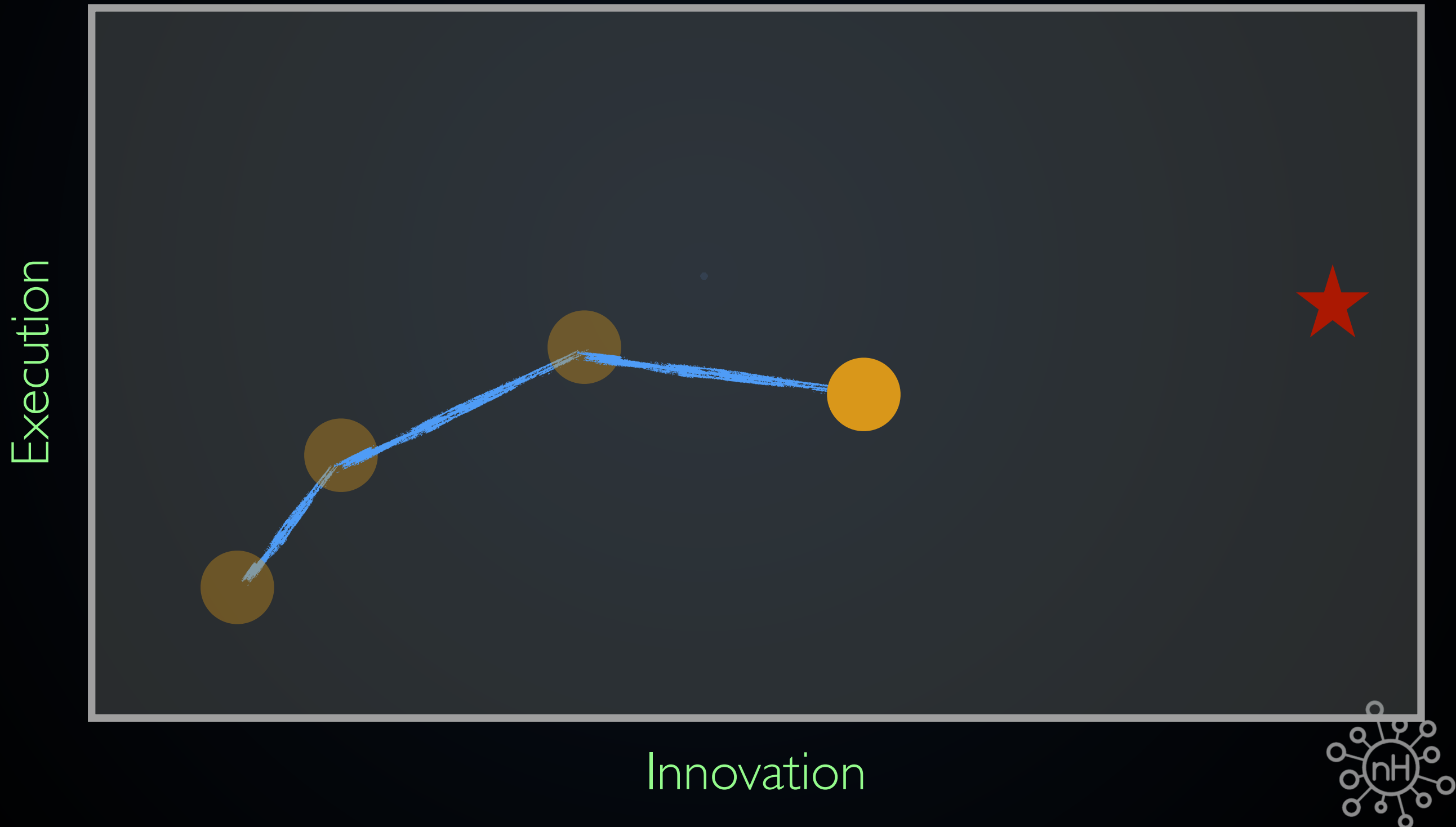


# SITUATION ❖ TARGET ❖ RE-PLAN

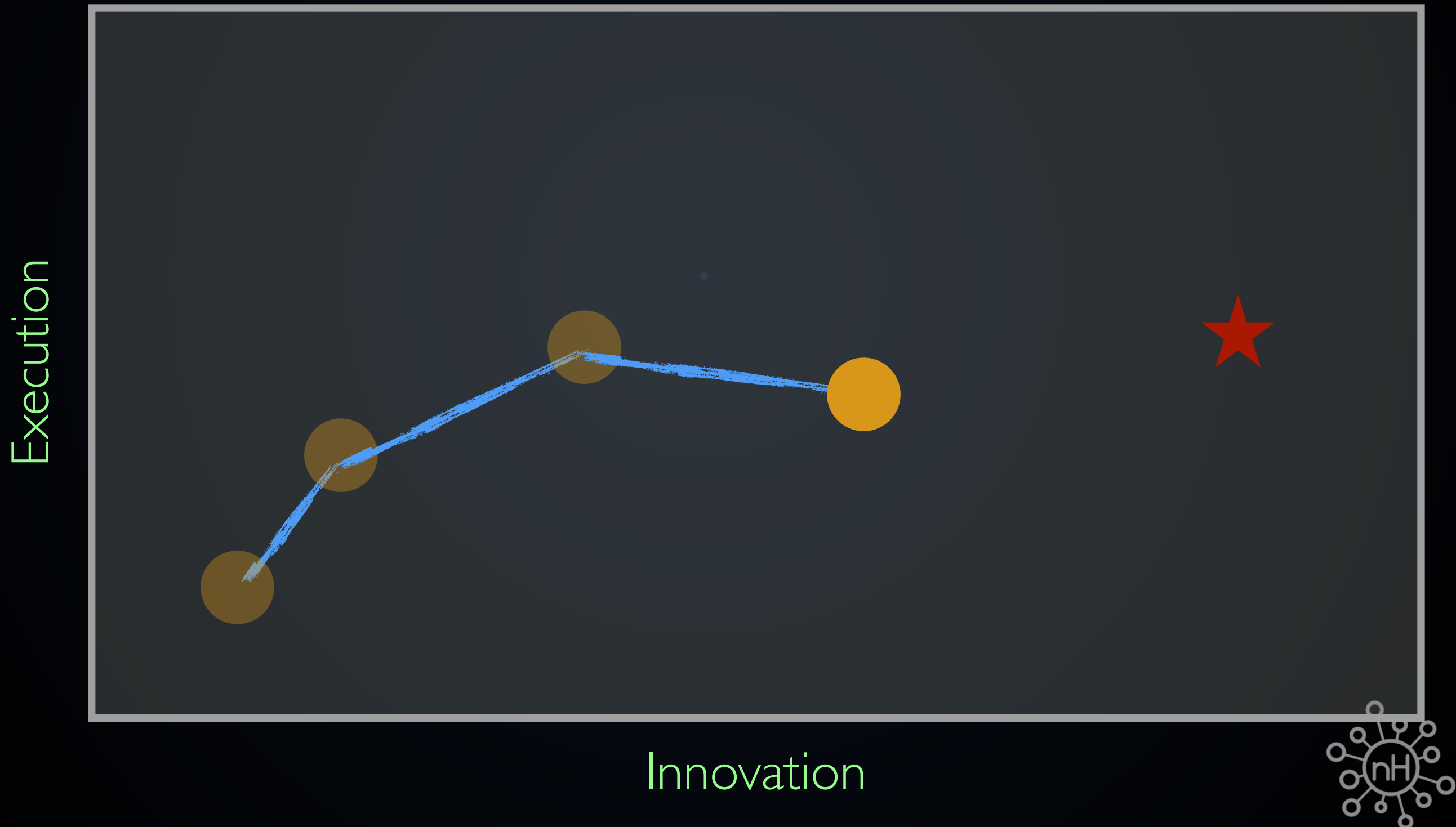




# SITUATION ❖ TARGET ❖ RE-PLAN

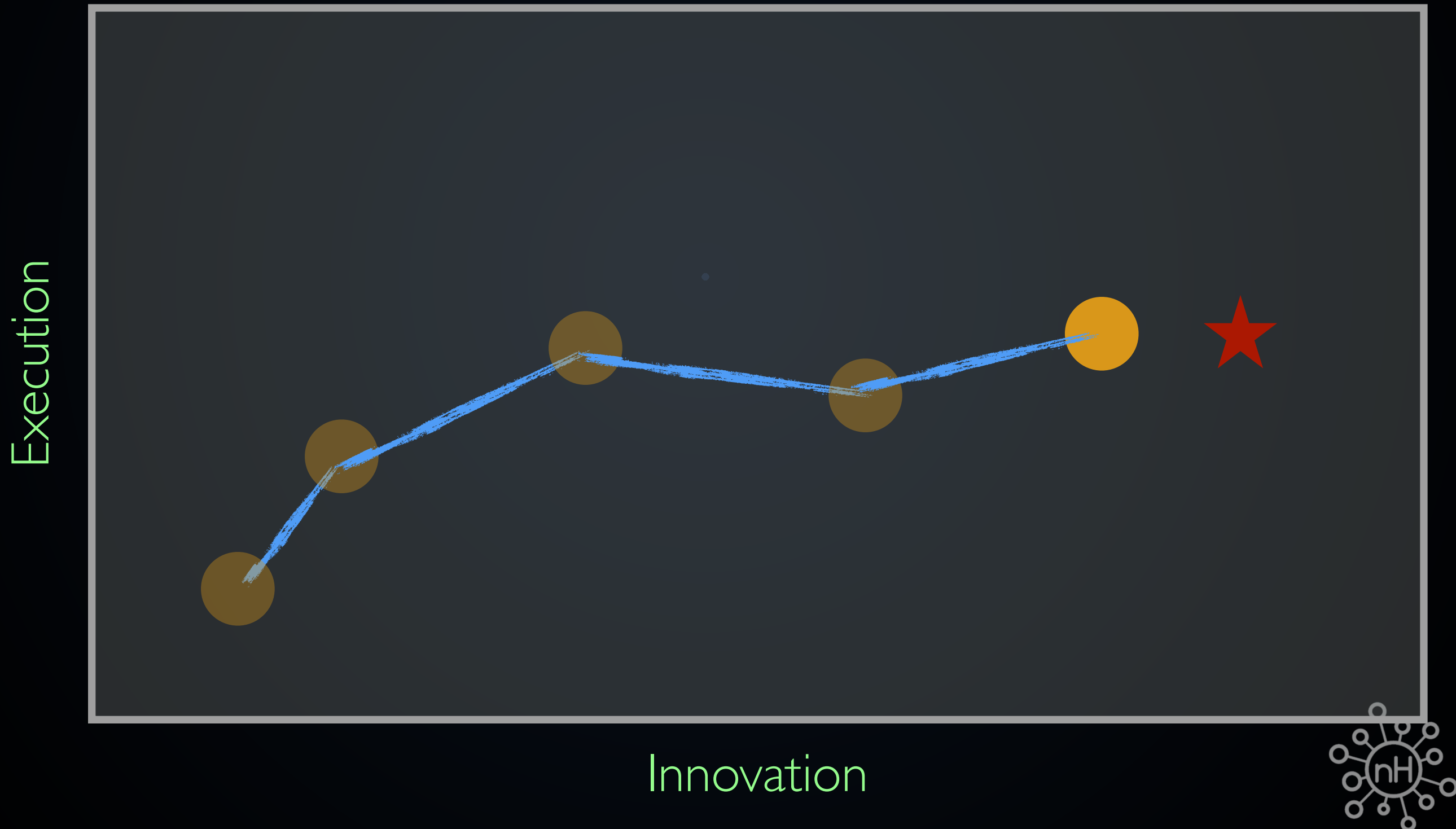


# SITUATION ❖ TARGET ❖ RE-PLAN

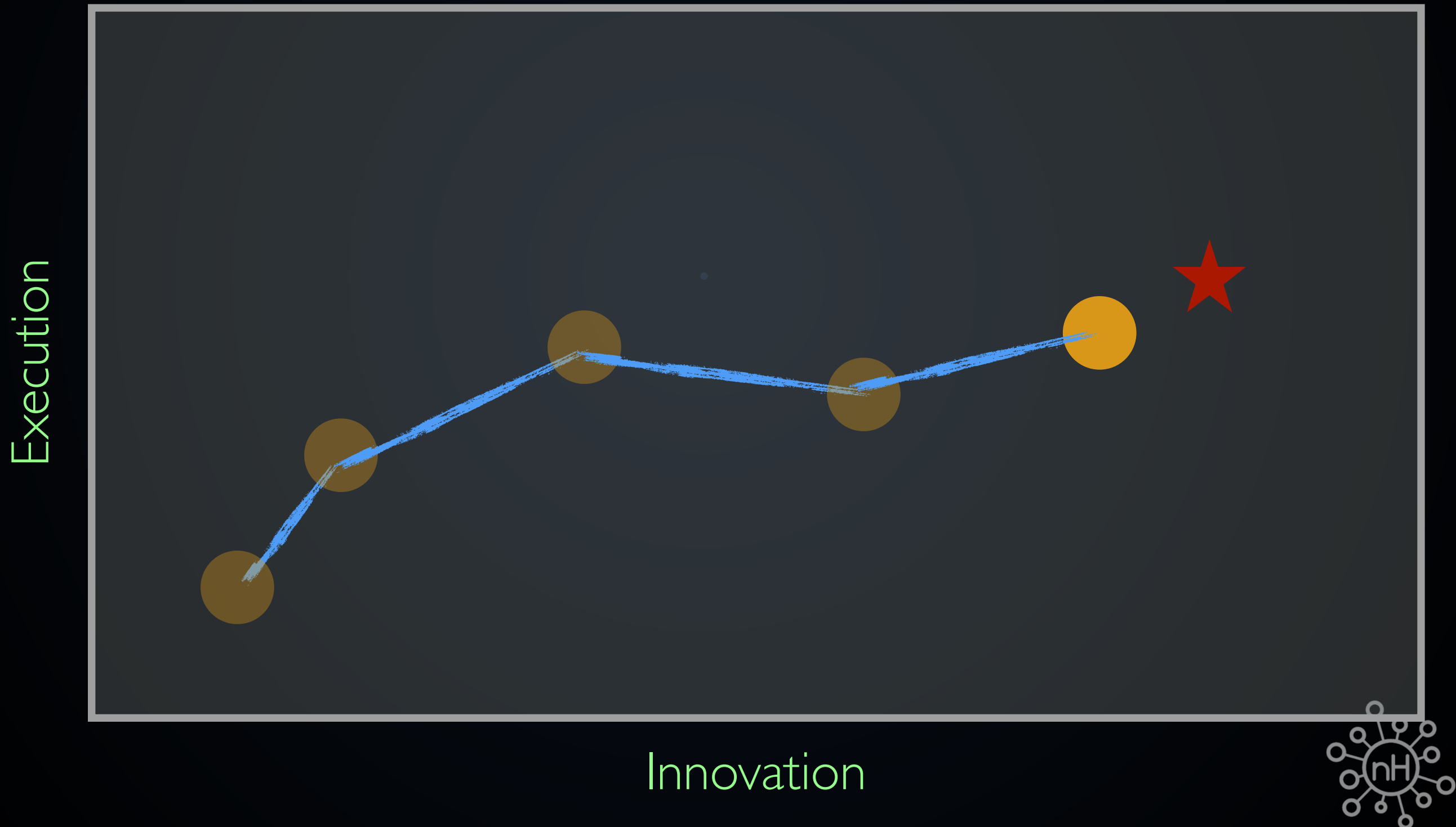




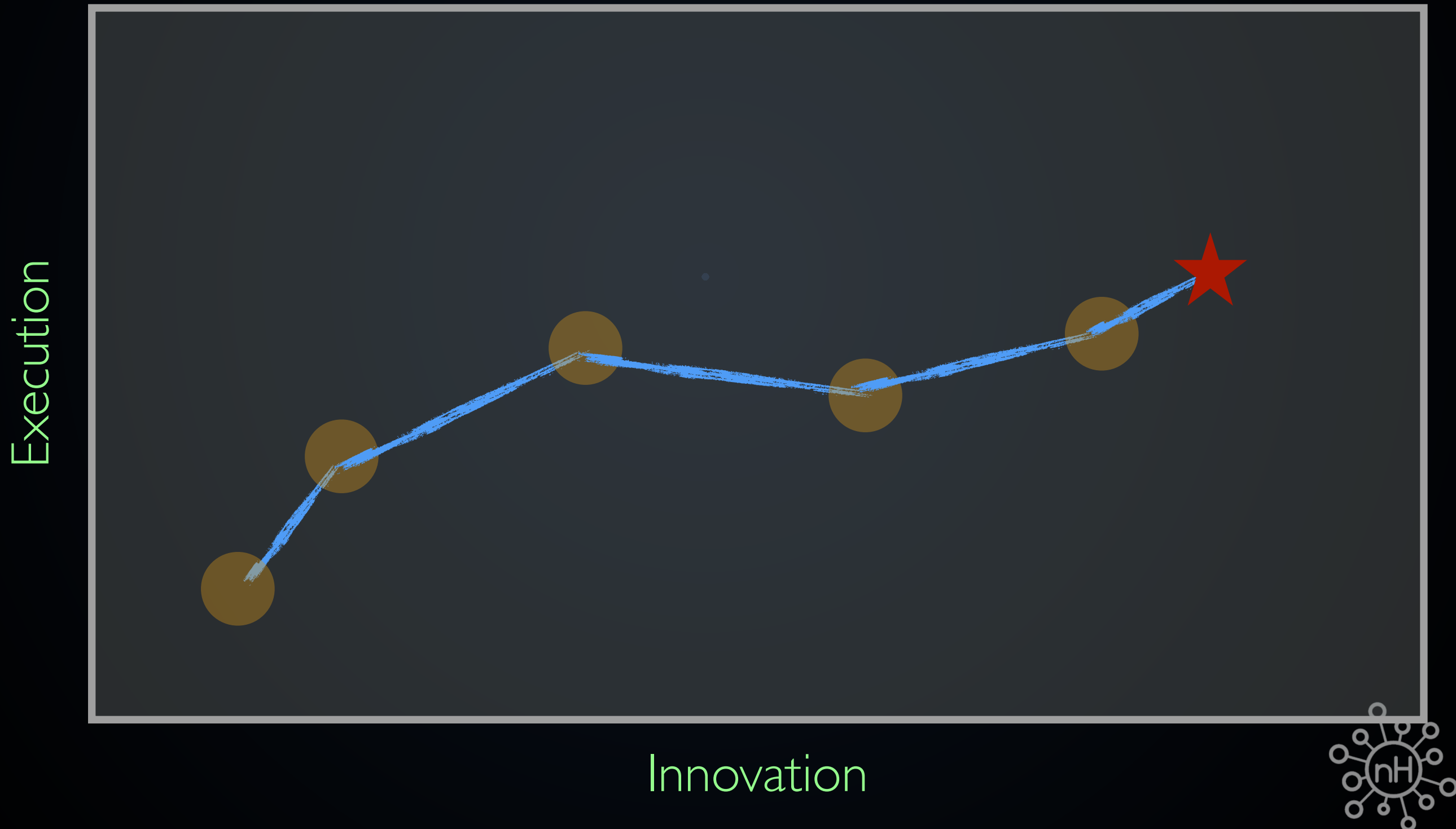
# SITUATION ❖ TARGET ❖ RE-PLAN



# SITUATION ❖ TARGET ❖ RE-PLAN

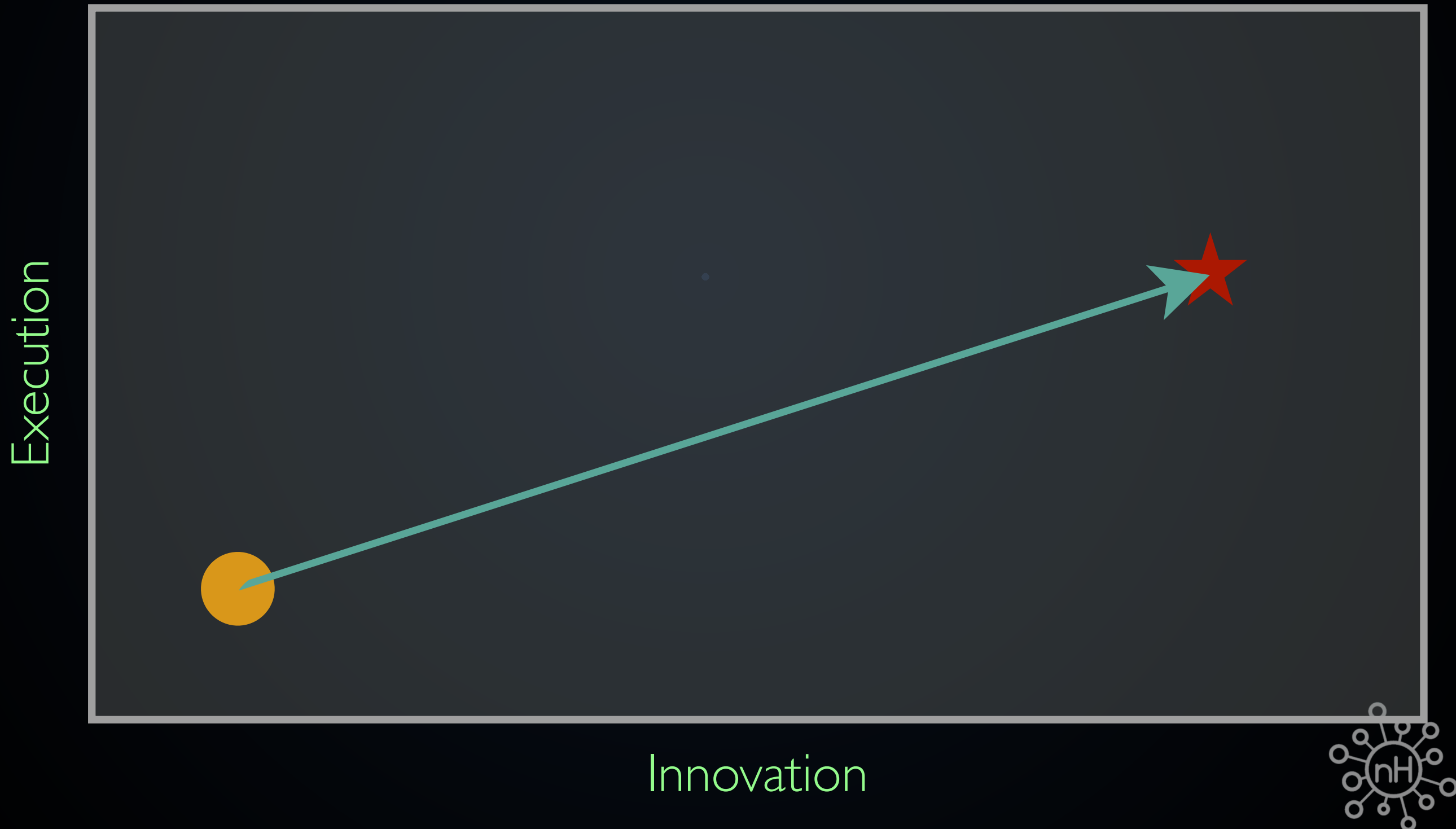


# SITUATION ❖ TARGET ❖ RE-PLAN





NOT A CALLED SHOT...  
BUT IT LOOKS LIKE ONE



# SITUATION

Execution



Innovation



```
117     }
118
119     Vector<MasterString> mskeys = new Vector<MasterString>(masters.keySet());
120     Collections.sort(mskeys);
121     for (int it = mskeys.size() - 1; it >= 0; it--) {
122         MasterString ms = mskeys.get(it);
123         Vector<SubsumedString> subs = masters.get(ms);
124         ol[0] = ms.count.toString();
125         ol[1] = ms.s;
126         ol[2] = subs.get(0).count.toString();
127         ol[3] = subs.get(0).s;
128         ol[4] = Double.toString(subs.get(0).levenshtein);
129         ol[5] = Double.toString(subs.get(0).jarovinkler);
130         w.writeNext(ol);
131         for (int i = 1; i < subs.size(); i++) {
132             SubsumedString ss = subs.get(i);
133             ol[0] = "";
134             ol[1] = "";
135             ol[2] = ss.count.toString();
136             ol[3] = ss.s;
137             ol[4] = Double.toString(ss.levenshtein);
138             ol[5] = Double.toString(ss.jarovinkler);
139             w.writeNext(ol);
140         }
141     }
142
143     w.close();
144 } catch (Exception e) {
145     System.err.println("Exception" + e + " at line " + linesRead);
146     e.printStackTrace();
147 }
```







```
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
}
Vector<MasterString> askeys = new Vector<MasterString>(masters.keySet());
Collections.sort(askeys);
for (int it = askeys.size() - 1; it >= 0; it--) {
    MasterString as = askeys.get(it);
    Vector<SubusedString> subs = masters.get(as);
    ol[0] = as.count.toString();
    ol[1] = as.s;
    ol[2] = subs.get(0).count.toString();
    ol[3] = subs.get(0).s;
    ol[4] = Double.toString(subs.get(0).levenshtein);
    ol[5] = Double.toString(subs.get(0).jarowinkler);
    w.writeNext(ol);
    for (int i = 1; i < subs.size(); i++) {
        SubusedString ss = subs.get(i);
        ol[0] = "";
        ol[1] = "";
        ol[2] = ss.count.toString();
        ol[3] = ss.s;
        ol[4] = Double.toString(ss.levenshtein);
        ol[5] = Double.toString(ss.jarowinkler);
        w.writeNext(ol);
    }
}
w.close();
} catch (Exception e) {
    System.err.println("Exception" + e + " at line " + linesRead);
    e.printStackTrace();
}
```

```
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
}
Vector<MasterString> askeys = new Vector<MasterString>(masters.keySet());
Collections.sort(askeys);
for (int it = askeys.size() - 1; it >= 0; it--) {
    MasterString as = askeys.get(it);
    Vector<SubusedString> subs = masters.get(as);
    ol[0] = as.count.toString();
    ol[1] = as.s;
    ol[2] = subs.get(0).count.toString();
    ol[3] = subs.get(0).s;
    ol[4] = Double.toString(subs.get(0).levenshtein);
    ol[5] = Double.toString(subs.get(0).jarowinkler);
    w.writeNext(ol);
    for (int i = 1; i < subs.size(); i++) {
        SubusedString ss = subs.get(i);
        ol[0] = "";
        ol[1] = "";
        ol[2] = ss.count.toString();
        ol[3] = ss.s;
        ol[4] = Double.toString(ss.levenshtein);
        ol[5] = Double.toString(ss.jarowinkler);
        w.writeNext(ol);
    }
}
w.close();
} catch (Exception e) {
    System.err.println("Exception" + e + " at line " + linesRead);
    e.printStackTrace();
}
```





Theoretical Electron Density Visualizer

Input → Simulate

WFN: [Coordinate complex, H3N.BH3]

Function

Choice: [Electron Density]

Grid dimensions and coarseness

Size: [16.00]  
Increment: [0.2]

Center

X: [0.0]  
Y: [0.0]  
Z: [0.0]

Wavefunction

Atom	Type	Charge	Coordinates (x, y, z)
H 1	(CENTRE 1)	1.0	1.10637655 -1.91630108 -2.19048878
H 2	(CENTRE 2)	1.0	1.10637695 1.91630108 -2.19048878
H 3	(CENTRE 3)	1.0	-2.21755889 0.00000000 -2.19048878
B 4	(CENTRE 4)	5.0	0.00000000 0.00000000 -1.61653643
N 5	(CENTRE 5)	7.0	0.00000000 0.00000000 1.52755535
H 6	(CENTRE 6)	1.0	-0.89935284 1.55772481 2.22014914

Simulate >

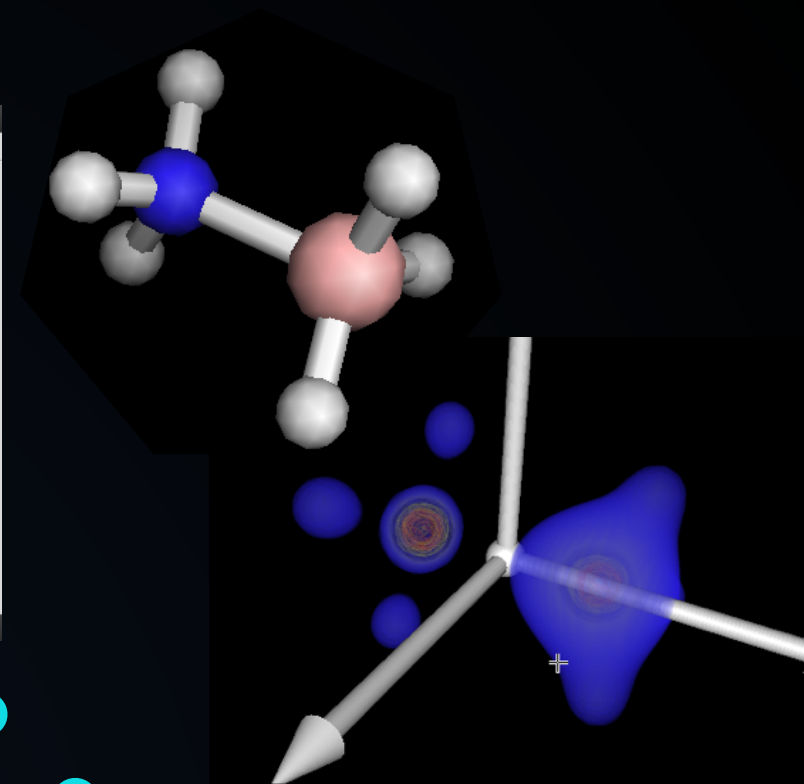
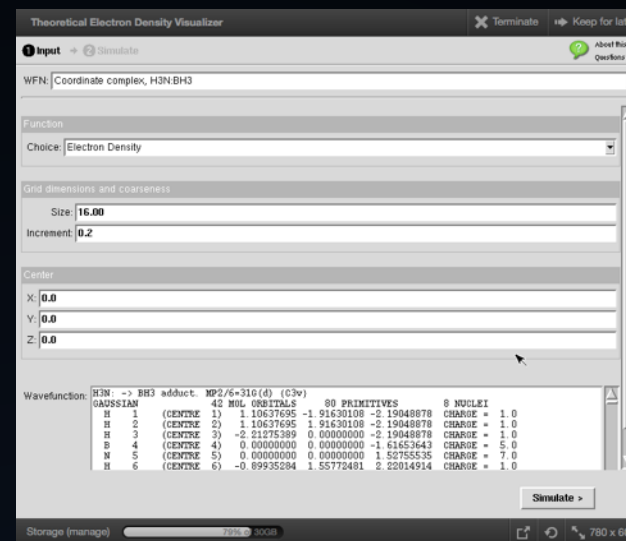
Storage (manage) 7296 3008

e.printStackTrace();



```
Vector<MasterString> askeys = new Vector<MasterString>(masters.keySet().size());
for (int i = 0; i < askeys.size(); i++) {
    MasterString as = askeys.get(i);
    Vector<SubwordString> subs = masters.get(as);
    ol[0] = as.count.toString();
    ol[1] = as.get(0).count.toString();
    ol[2] = as.get(0).count.toString();
    ol[3] = as.get(0).count.toString();
    ol[4] = Double.toString(subs.get(0).levenshtein());
    ol[5] = Double.toString(subs.get(0).jarowinkler());
    w.writeNext(ol);
    for (int i = 1; i < subs.size(); i++) {
        SubwordString ss = subs.get(i);
        ol[0] = ss.count.toString();
        ol[1] = ss.get(0).count.toString();
        ol[2] = ss.get(0).count.toString();
        ol[3] = ss.get(0).count.toString();
        ol[4] = Double.toString(ss.levenshtein());
        ol[5] = Double.toString(ss.jarowinkler());
        w.writeNext(ol);
    }
}
close();
} catch (Exception e) {
    System.err.println("Exception" + e + " at line " + linesRead);
    e.printStackTrace();
}
```





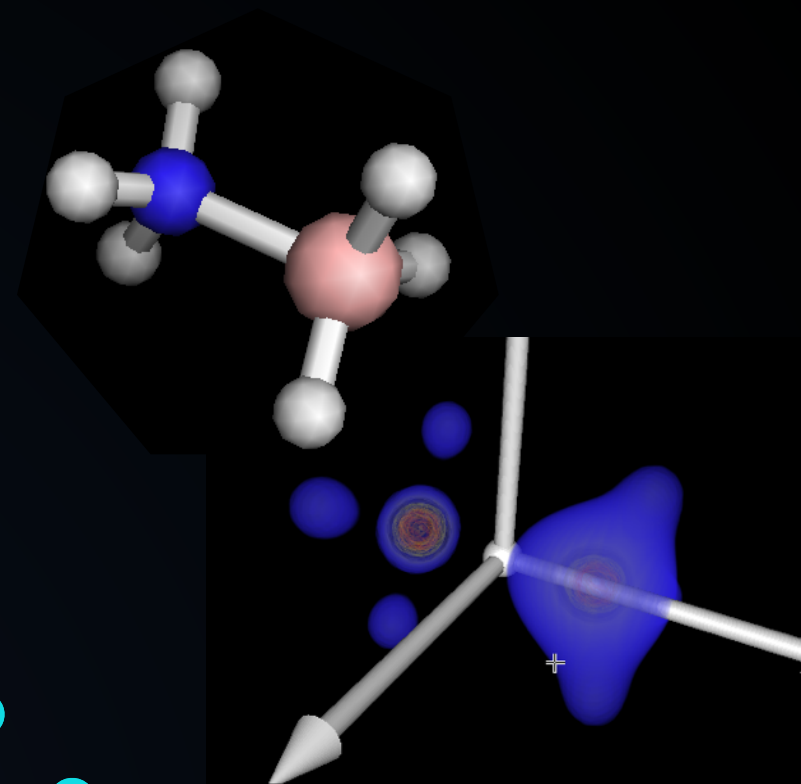
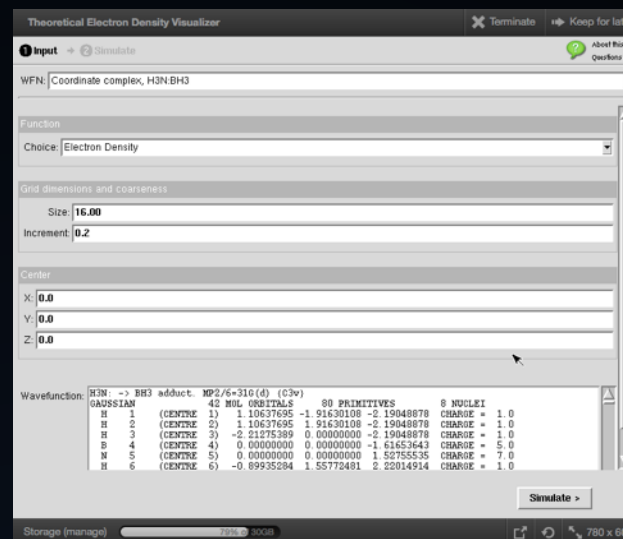
```

}
Vector<MasterString> askeys = new Vector<MasterString>(masters.keySet());
Collections.sort(askeys);
for (int i = askeys.size() - 1; i >= 0; i--) {
    MasterString as = askeys.get(i);
    Vector<SubwordString> subs = masters.get(as);
    ol[0] = as.count.toString();
    ol[1] = as.z;
    ol[2] = subs.get(0).count.toString();
    ol[3] = subs.get(0).z;
    ol[4] = Double.toString(subs.get(0).levenshtein);
    ol[5] = Double.toString(subs.get(0).jarovinkier);
    w.writeNext(ol);
    for (int s = 1; s < subs.size(); s++) {
        SubwordString ss = subs.get(s);
        ol[0] = "";
        ol[1] = "";
        ol[2] = ss.count.toString();
        ol[3] = ss.z;
        ol[4] = Double.toString(ss.levenshtein);
        ol[5] = Double.toString(ss.jarovinkier);
        w.writeNext(ol);
    }
}
close();
} catch (Exception e) {
    System.err.println("Exception" + e + " at line " + linesRead);
    e.printStackTrace();
}

```







```

}
Vector<MasterString> askeys = new Vector<MasterString>(masters.keySet());
Collections.sort(askeys);
for (int i = askeys.size() - 1; i >= 0; i--) {
    MasterString as = askeys.get(i);
    Vector<SubswedString> subs = masters.get(as);
    o[i] = as.count.toString();
    o[i+1] = as.get(0).count.toString();
    o[i+2] = as.get(0).count.toString();
    o[i+3] = as.get(0).count.toString();
    o[i+4] = Double.toString(subs.get(0).levenshtein());
    o[i+5] = Double.toString(subs.get(0).jarovinkier());
    w.writeText(o[i]);
    for (int k = 1; k < subs.size(); k++) {
        SubswedString ss = subs.get(k);
        o[i] = "";
        o[i+1] = "";
        o[i+2] = ss.count.toString();
        o[i+3] = ss.count.toString();
        o[i+4] = Double.toString(ss.levenshtein());
        o[i+5] = Double.toString(ss.jarovinkier());
        w.writeText(o[i]);
    }
}
}
close();
}
catch (Exception e) {
    System.err.println("Exception" + e + " at line " + linesRead);
    e.printStackTrace();
}
}

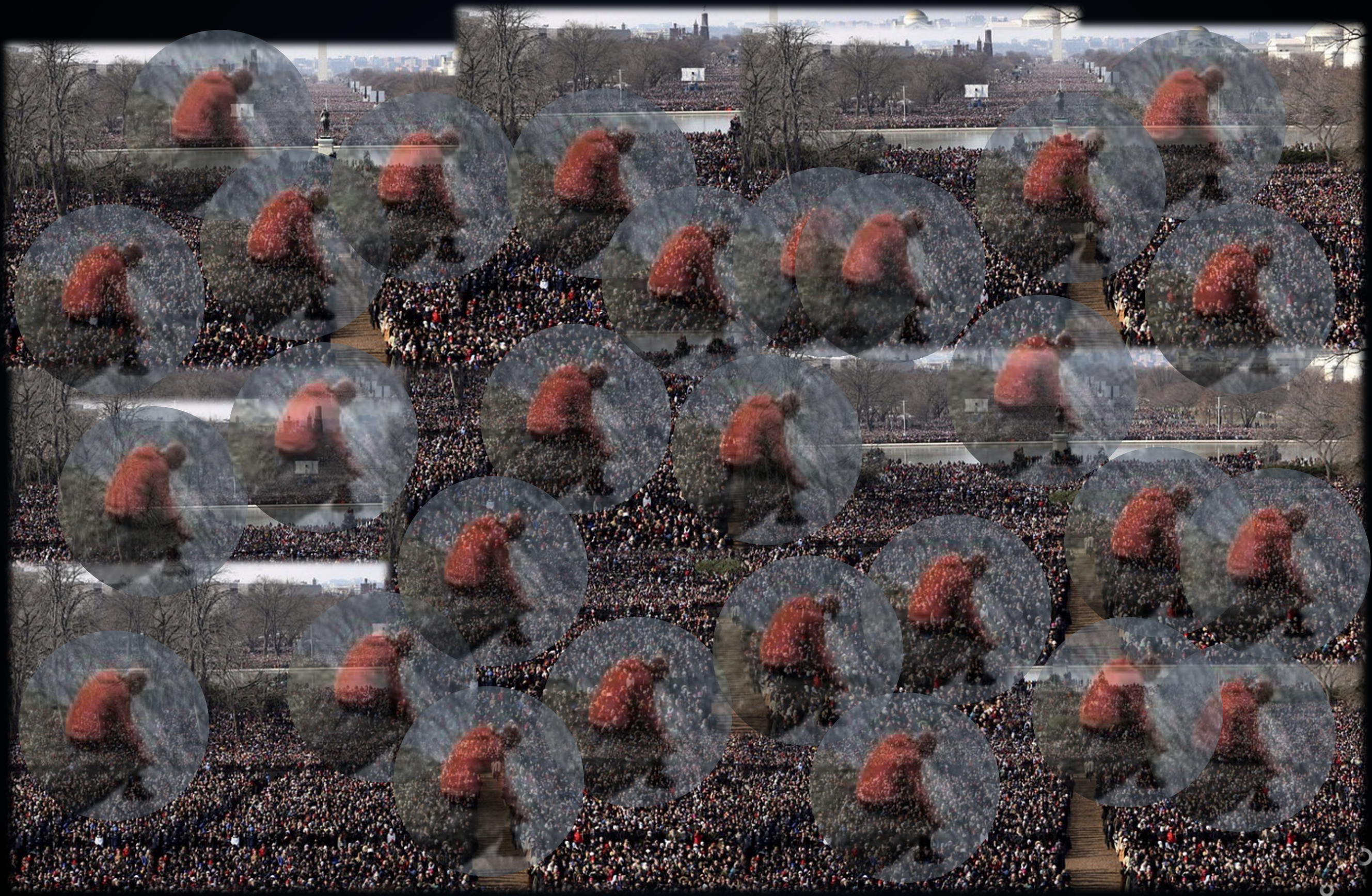
```















# NANOHUB TODAY

370+ Simulation Tools

4700+ Other Resources

3000+ Online Presentations

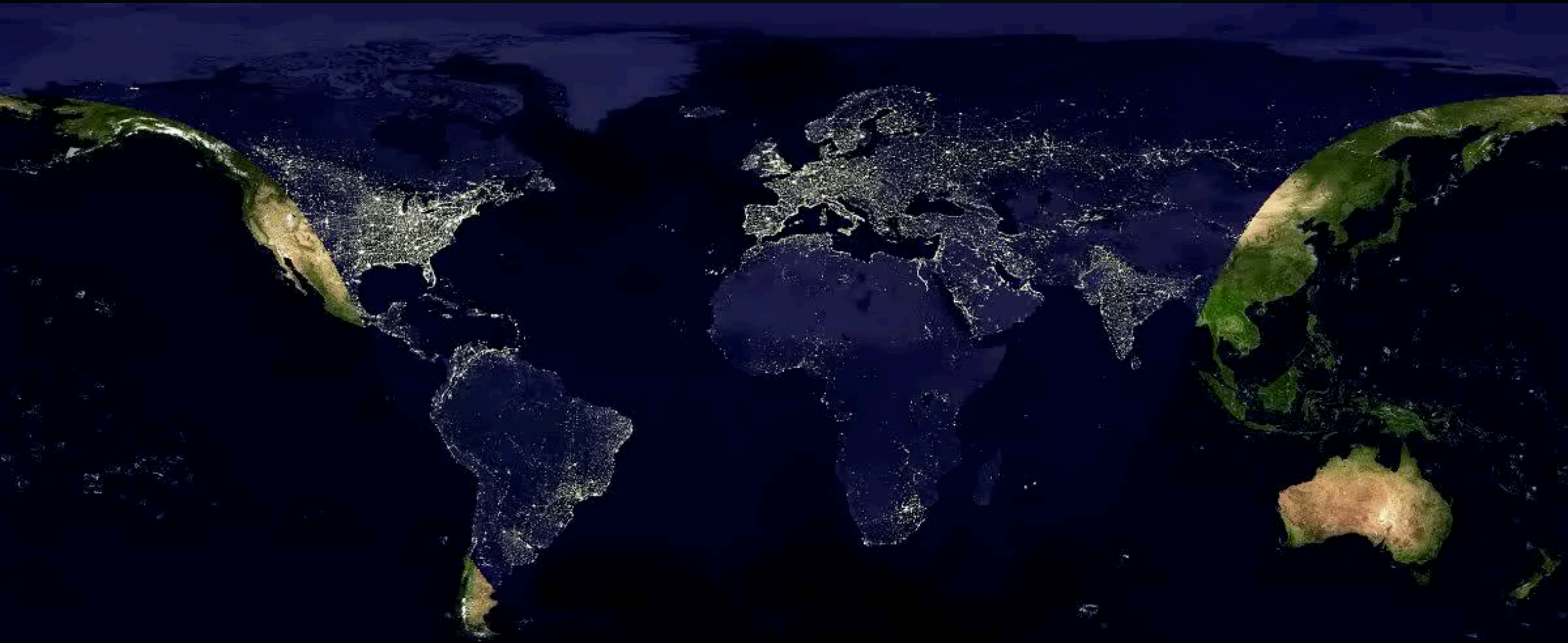
540+ Teaching Materials



At No Cost: [nanohub.org](https://nanohub.org)



# A CONTINUOUS USER BASE

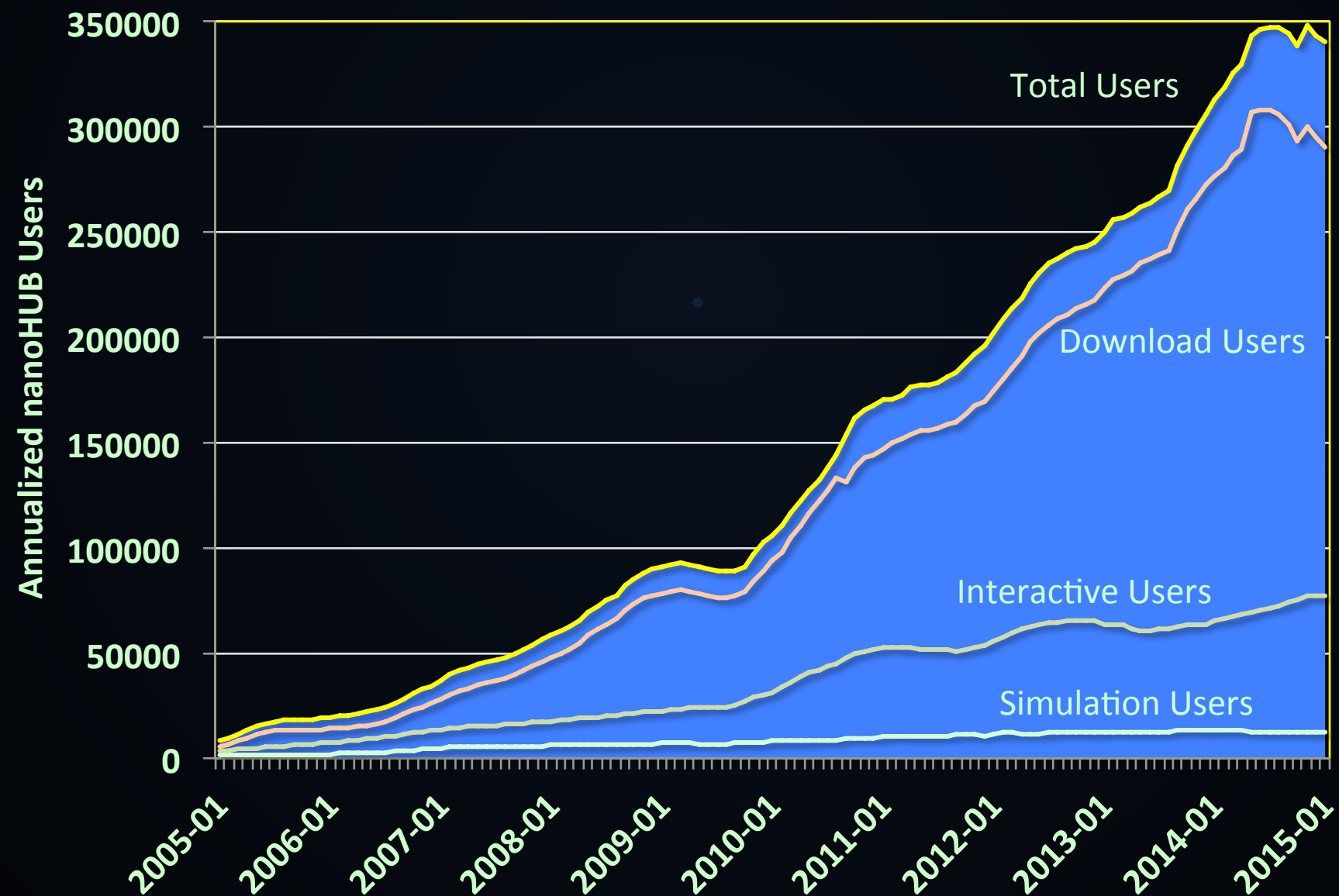


nanoHUB.org usage 2015-01-12 00:00:00

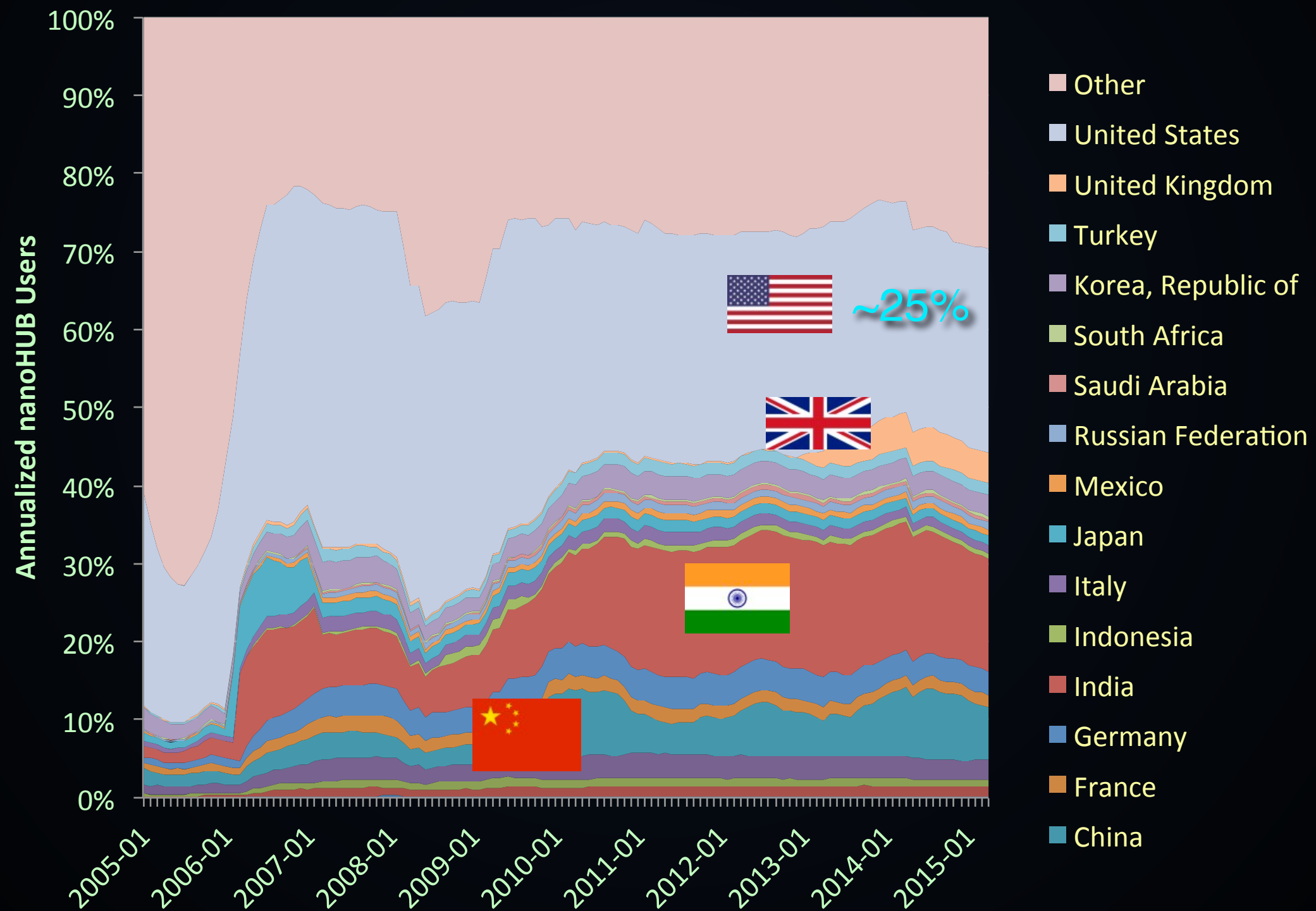




# A LARGE USER BASE



# AN INTERNATIONAL USER BASE







# LESSON 1

Expand Vertically:

“The needs of the one outweigh the needs of the many”

More capabilities

More users

As the “market” informs you of its needs



# EXPERTISE NEEDED

---

- Domain Scientists
- HPC & Middleware
- Security
- Web Design / Programming
- Analytics
- Hardware
- Production Operations
- Databases



# EXPERTISE NEEDED

---

- Domain Scientists
- HPC & Middleware
- Security
- Web Design / Programming
- Analytics
- Hardware
- Production Operations
- Databases





# OH...AND MORE EXPERTISE NEEDED

---

- Domain Scientists
- HPC & Middleware
- Security
- Web Design / Programming
- Analytics
- Hardware
- Production Operations
- Databases
- Financial Management
- Project Management
- Outreach
- Administrative
- Educational Specialists







## LESSON 2

Expand horizontally

You cannot afford the resources you need to do something like this.

Generalize and leverage across projects.















60+ HUBs  
28 HUBzero Staff  
10 NCN staff (7 FTE)  
~2,000,000 Visitors Annually









# LESSON 3

Justify Your Existence

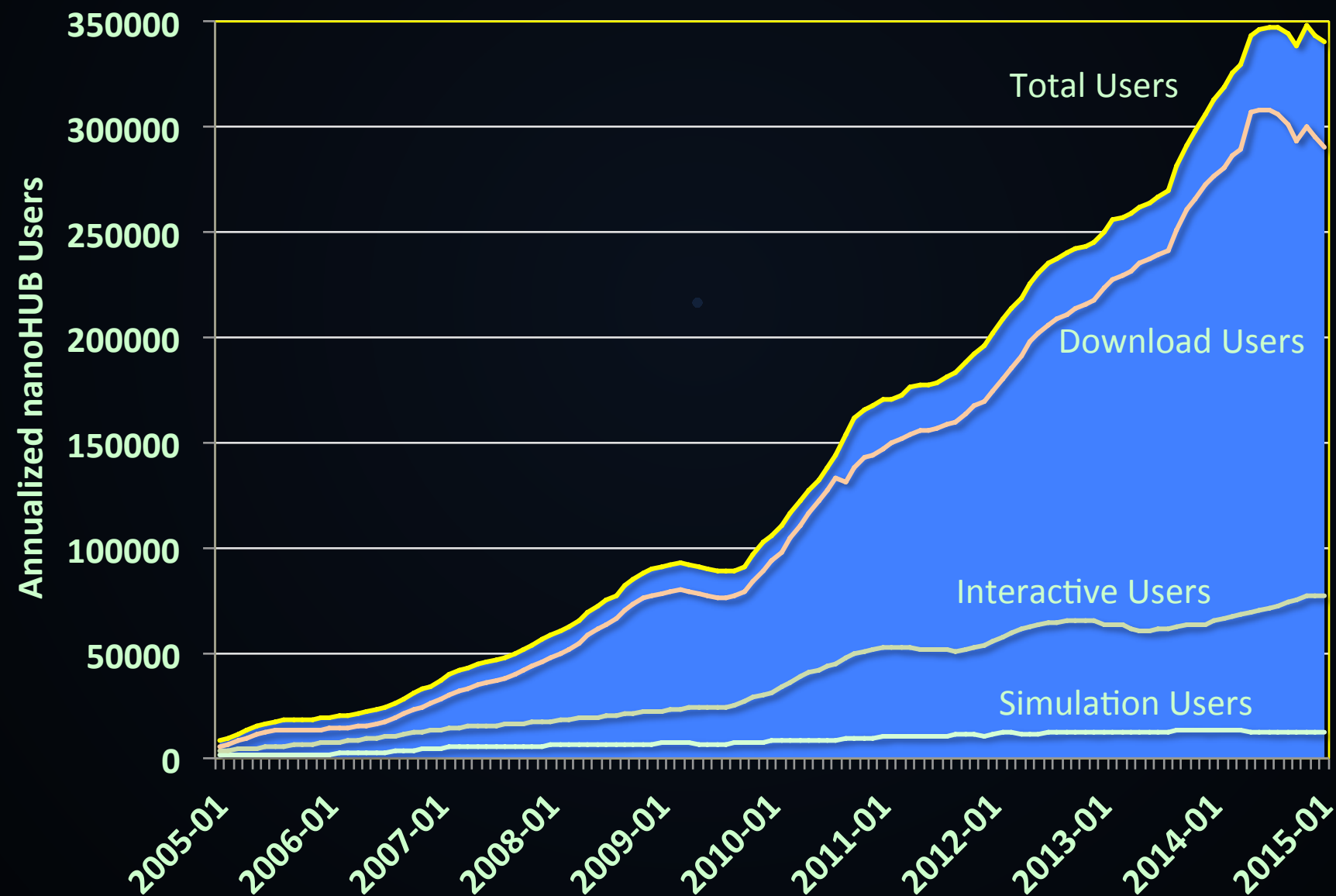
Measure Impact, Not Just Users

Make Your Contributors Want to Contribute

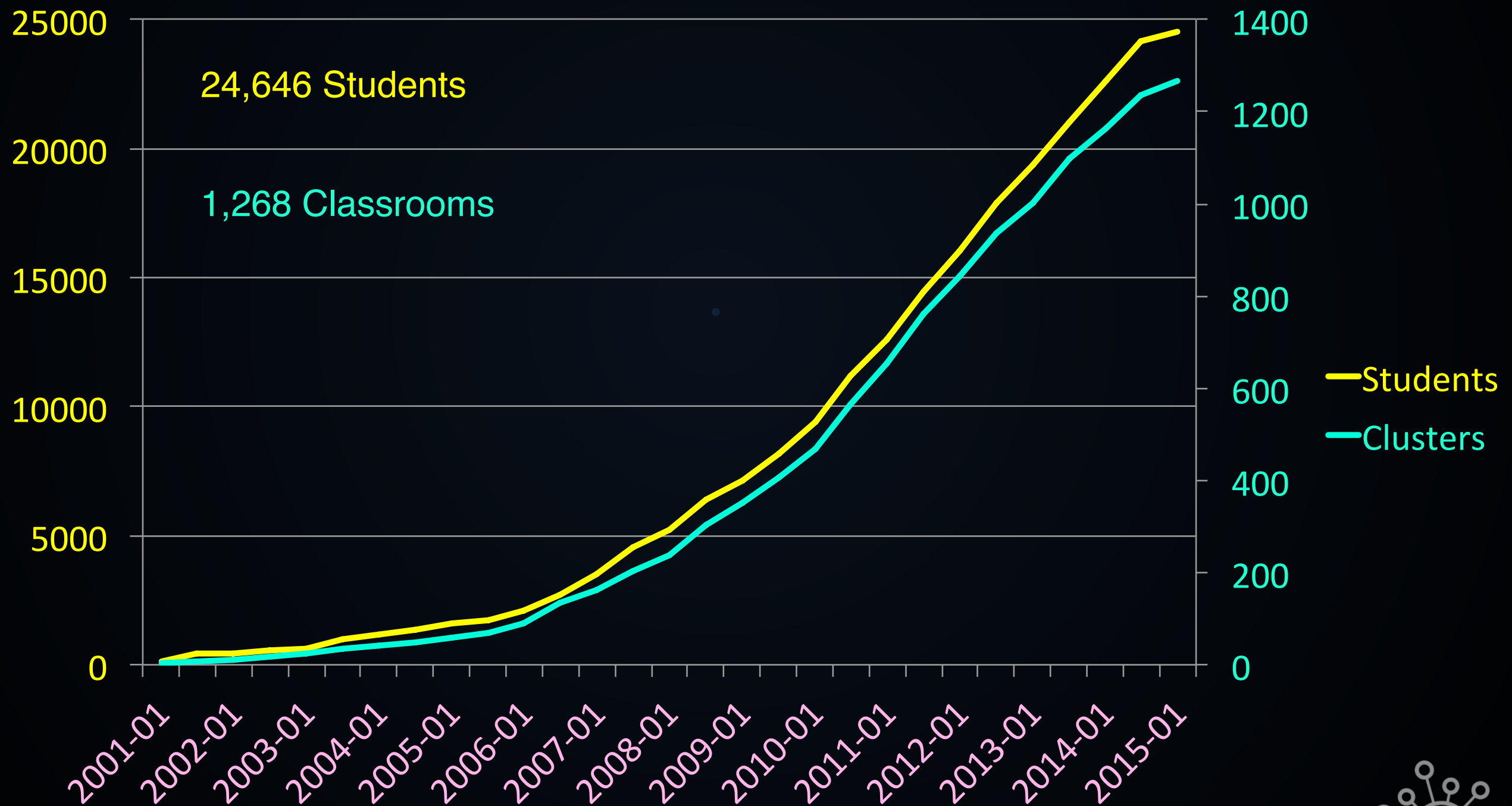
Learn from Users



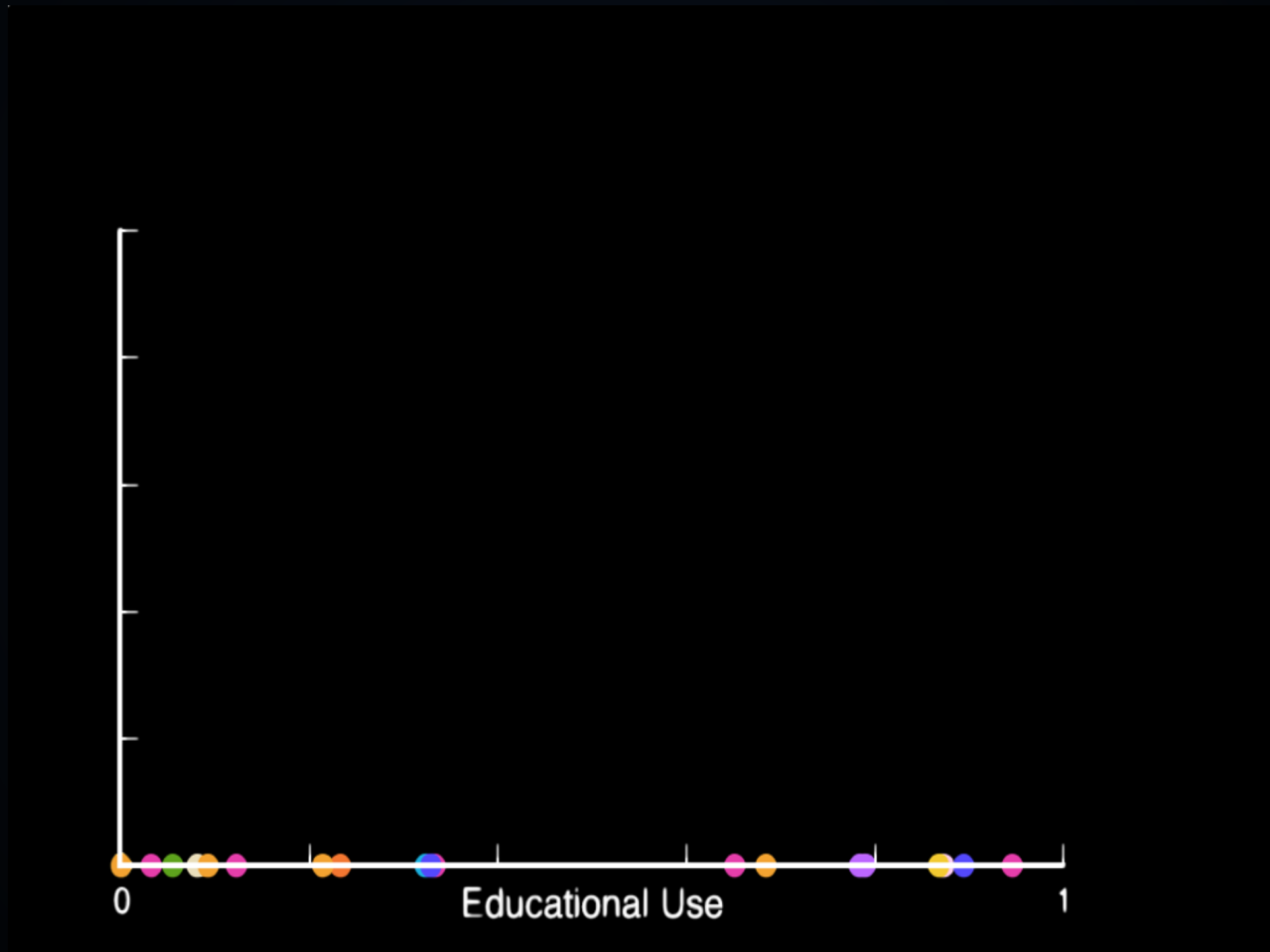
# A LARGE USER BASE



# CLASSROOM IMPACT

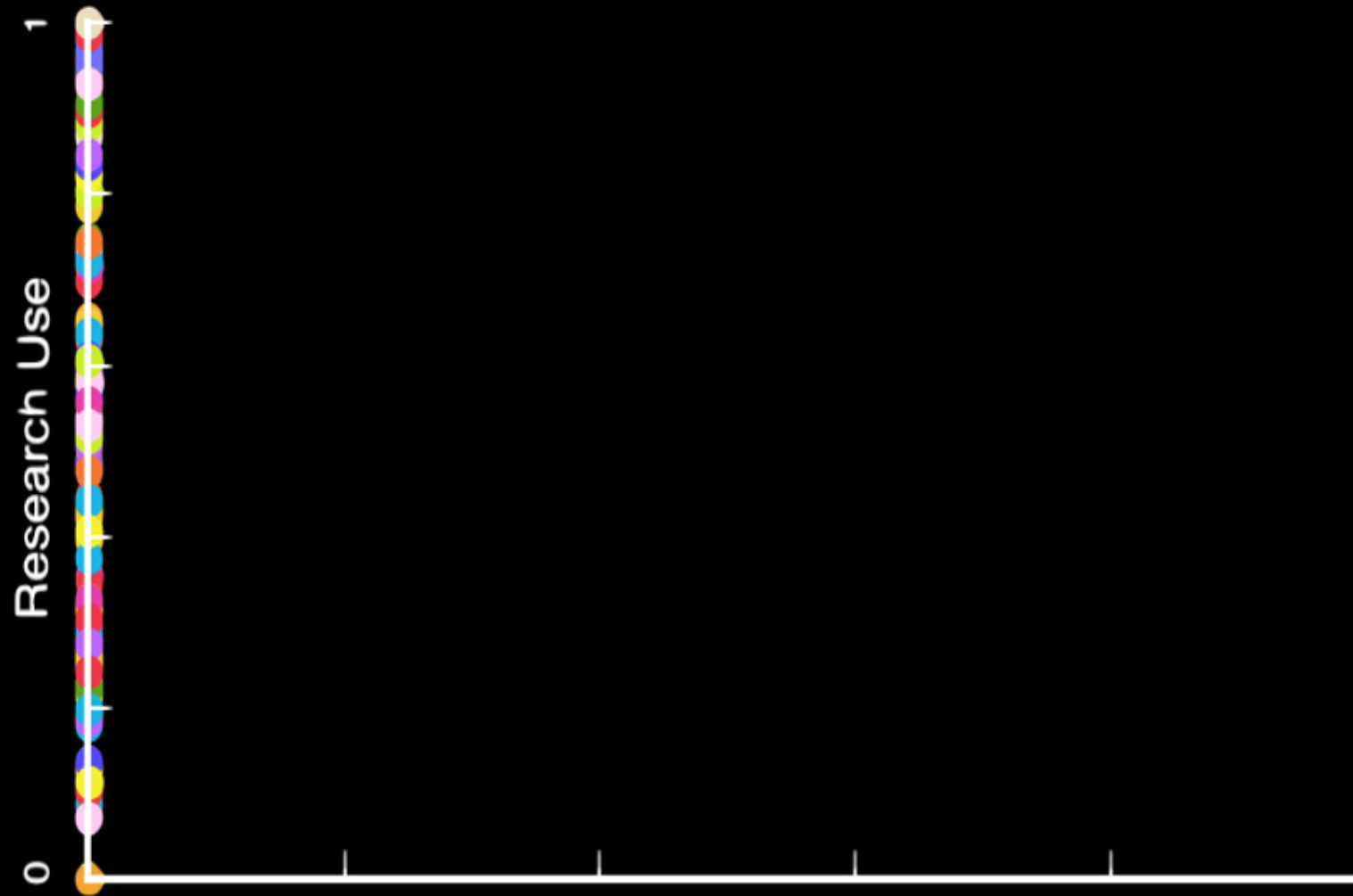
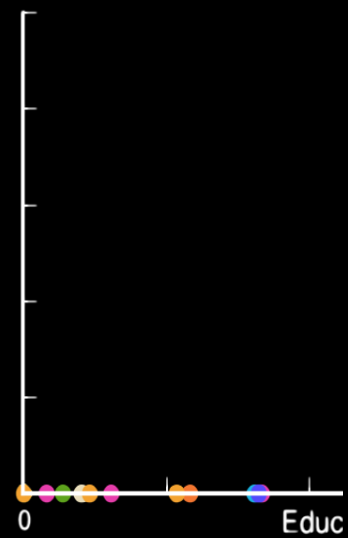


# EDUCATIONAL MIGRATION

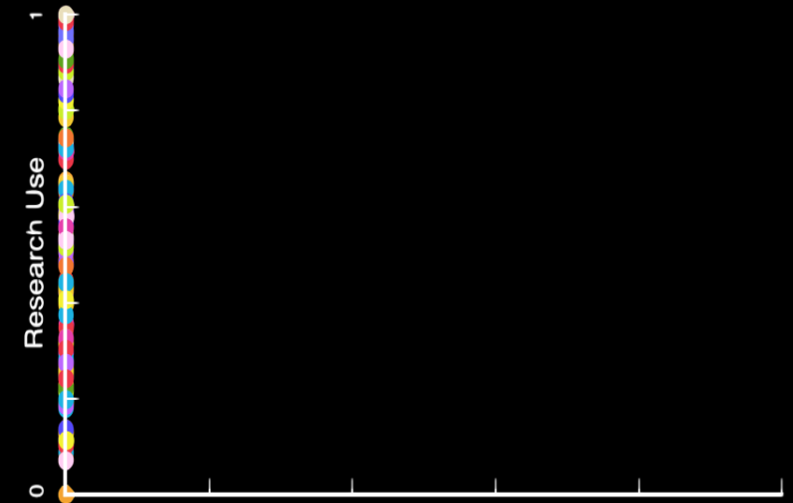
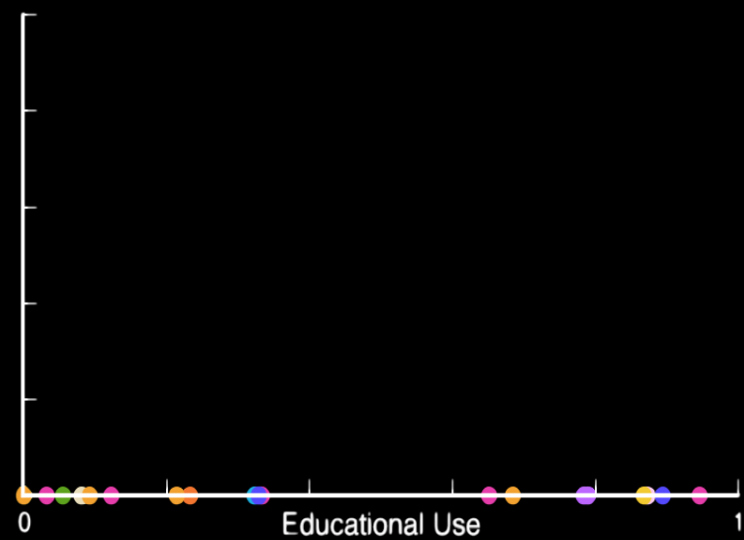




# EDUCATIONAL MIGRATION

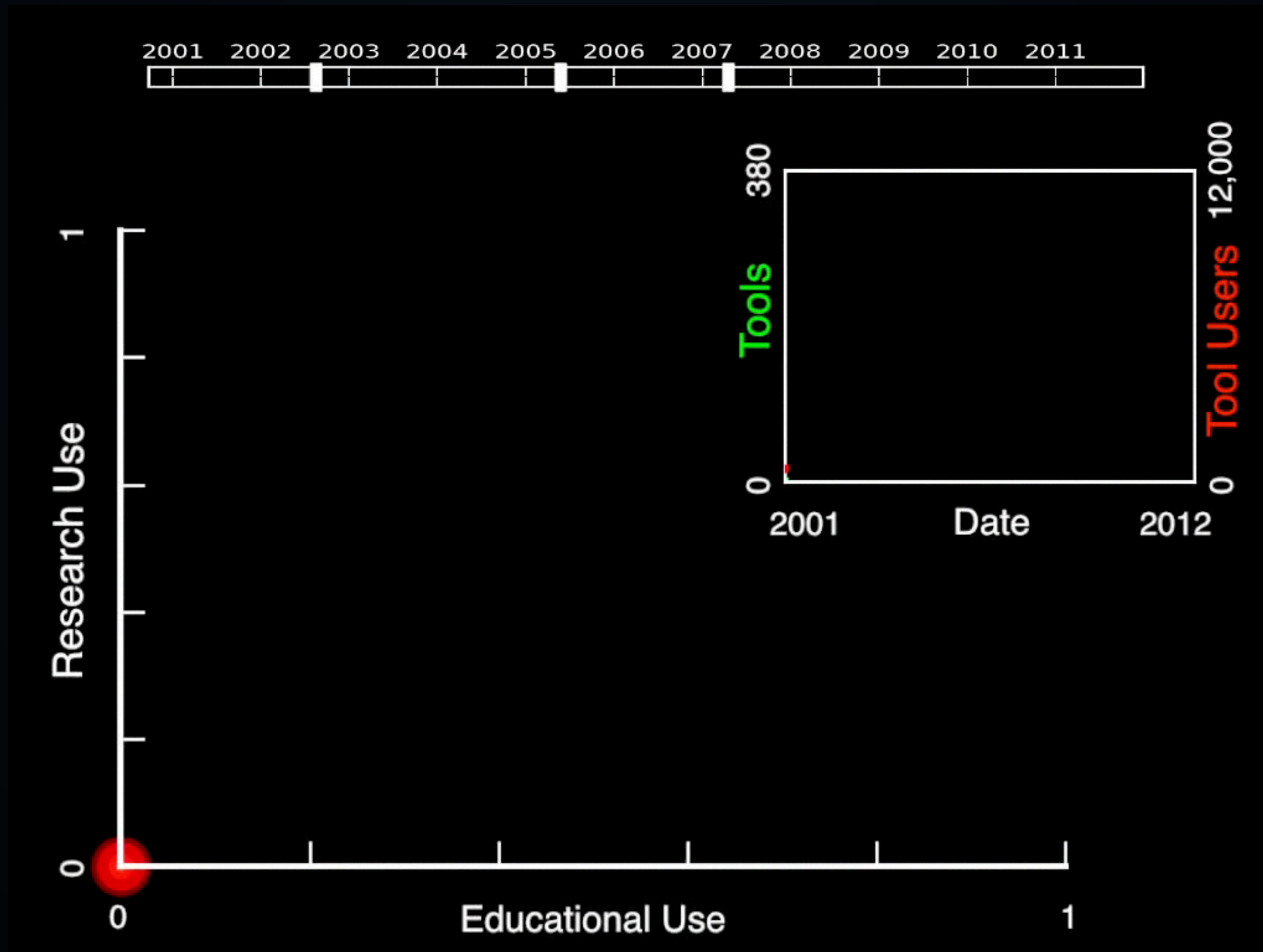


# EDUCATIONAL MIGRATION

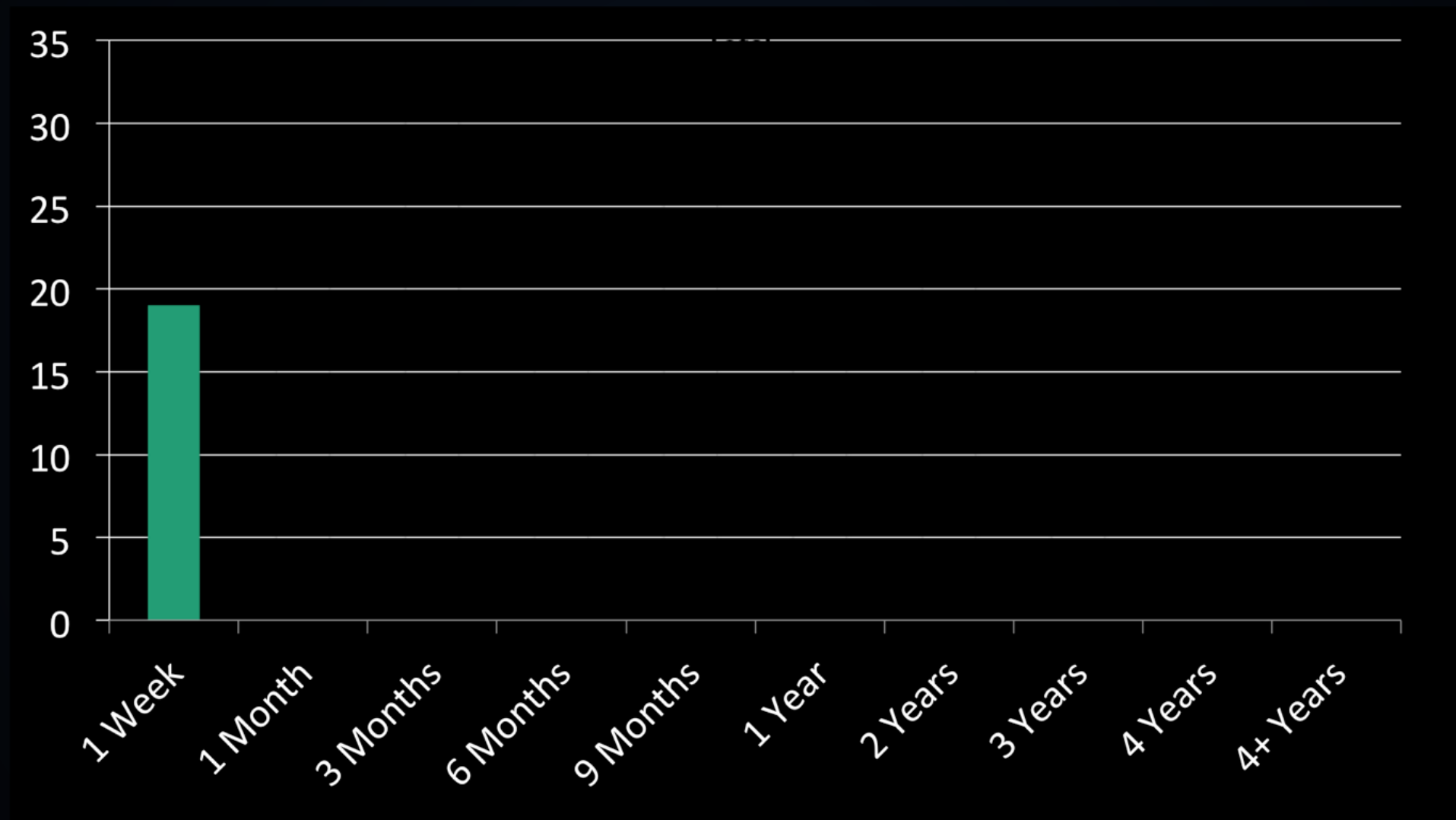




# EDUCATIONAL MIGRATION

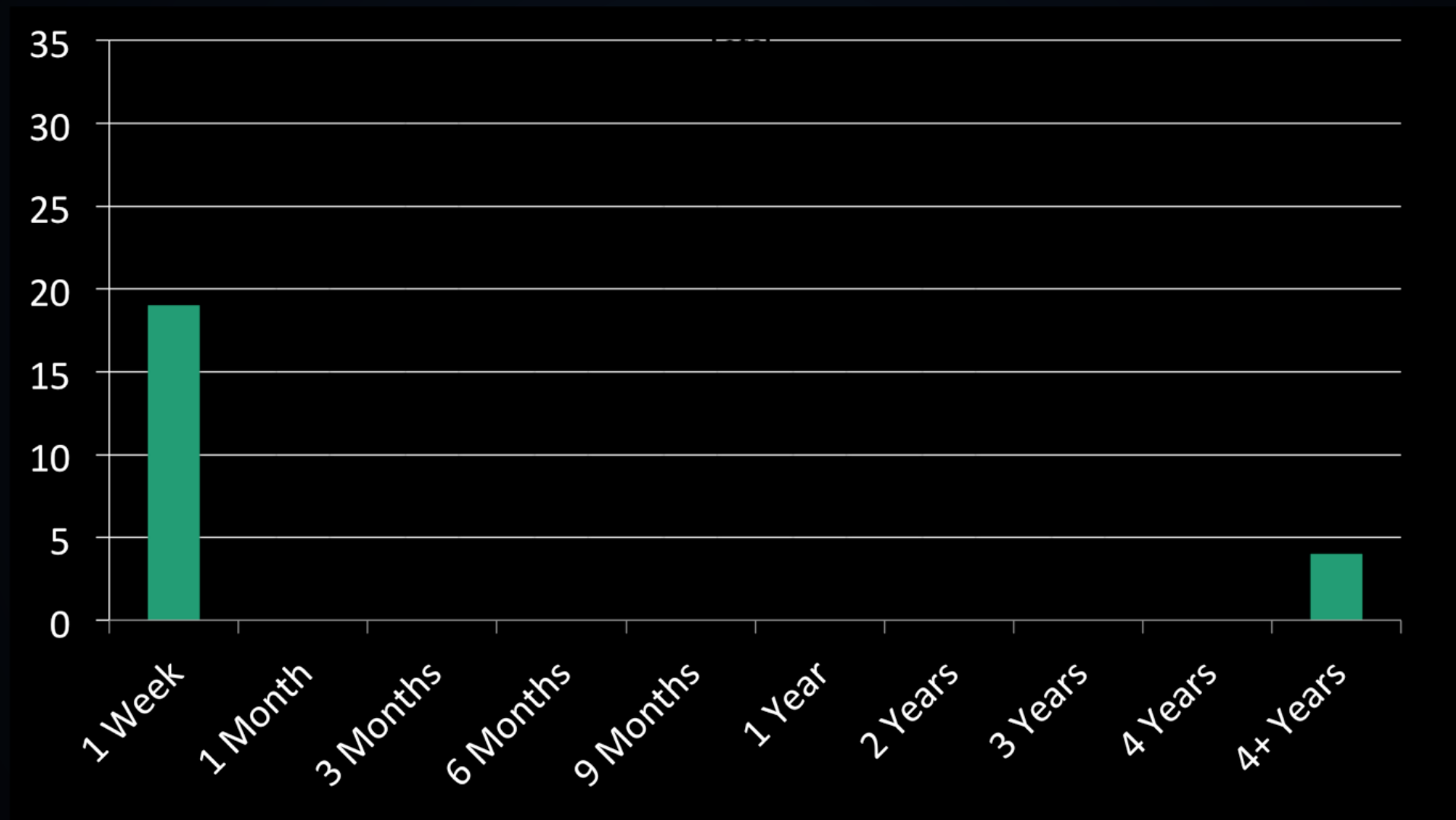


# EDUCATIONAL ADOPTION

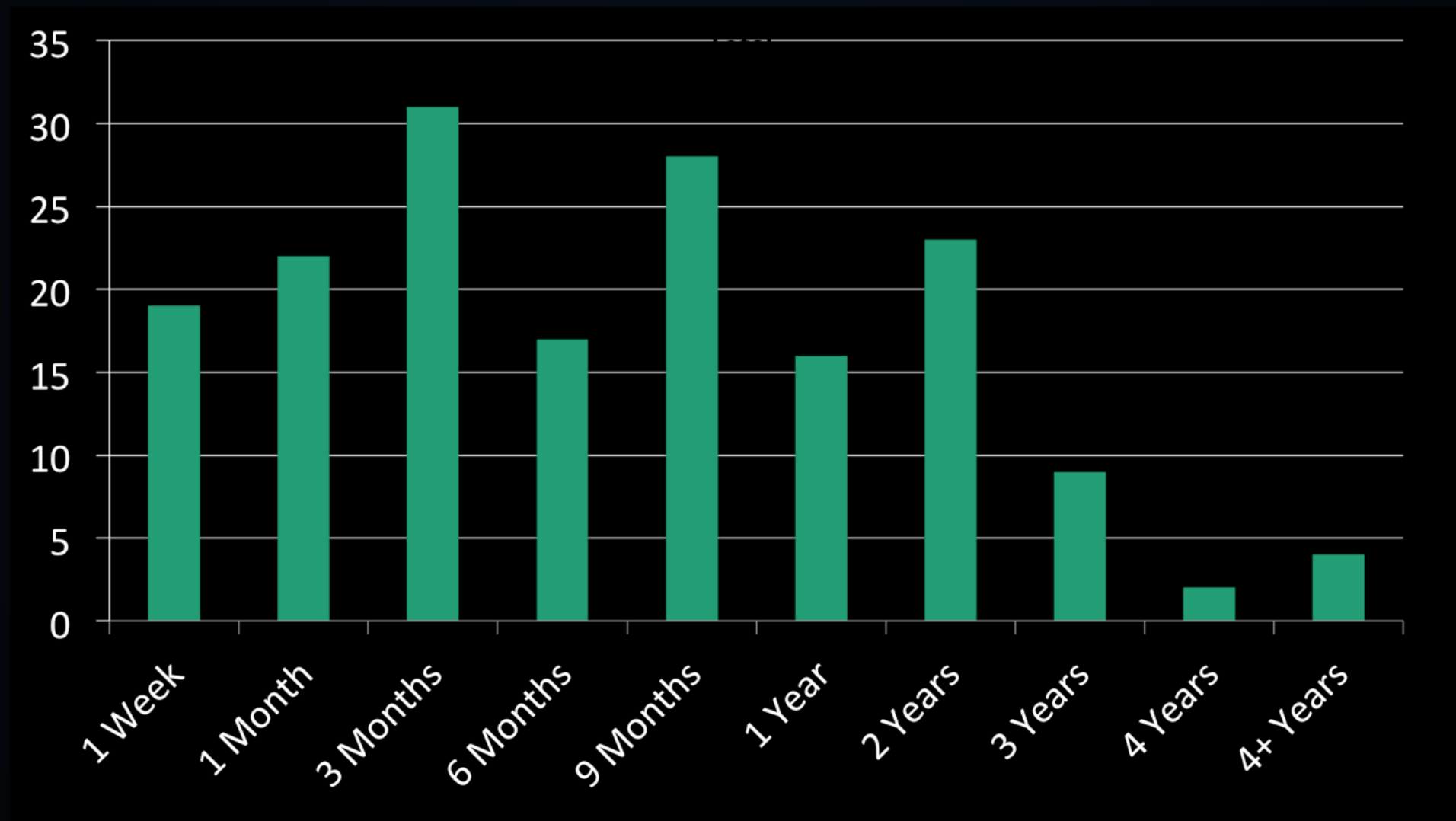




# EDUCATIONAL ADOPTION

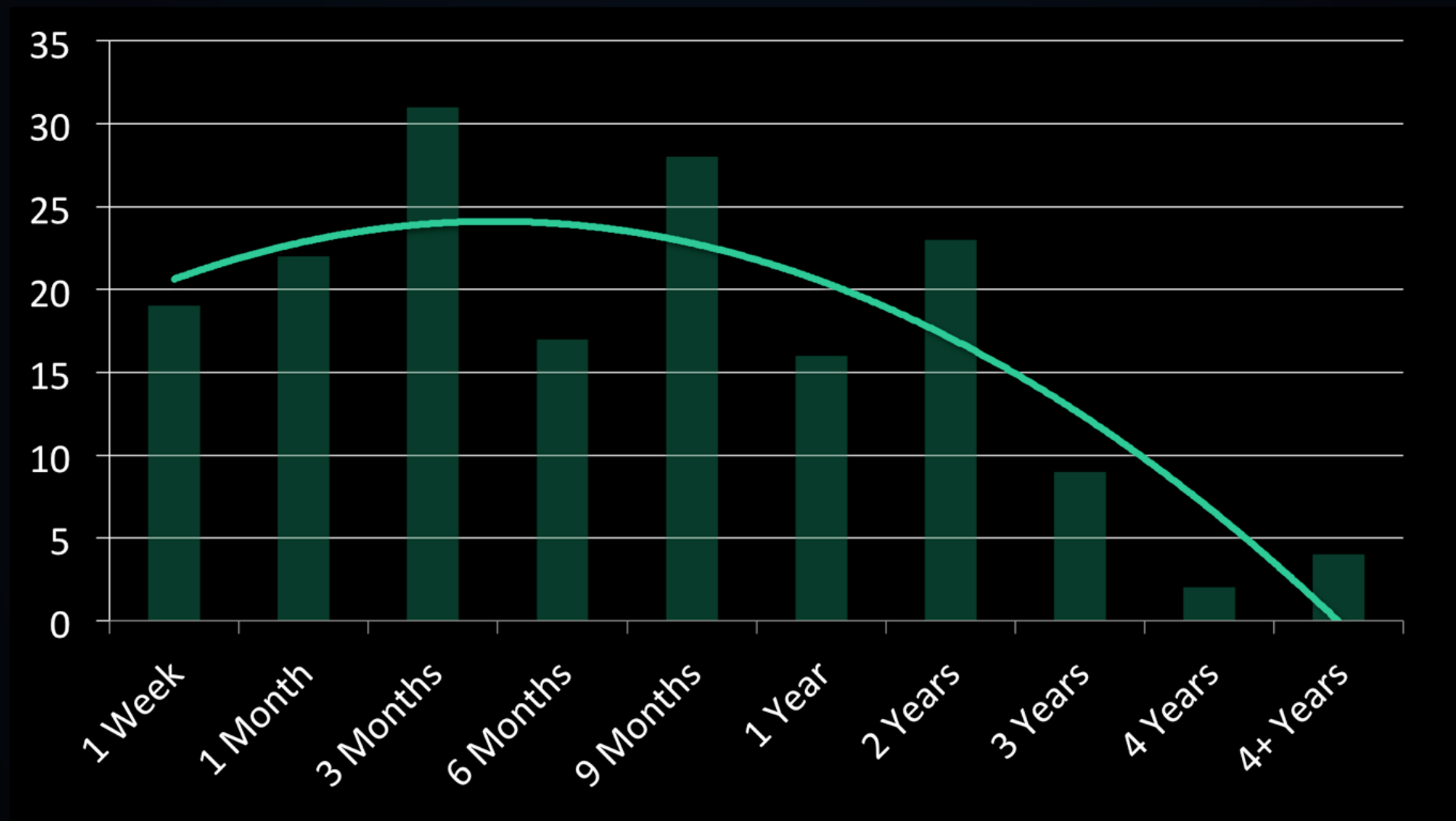


# EDUCATIONAL ADOPTION

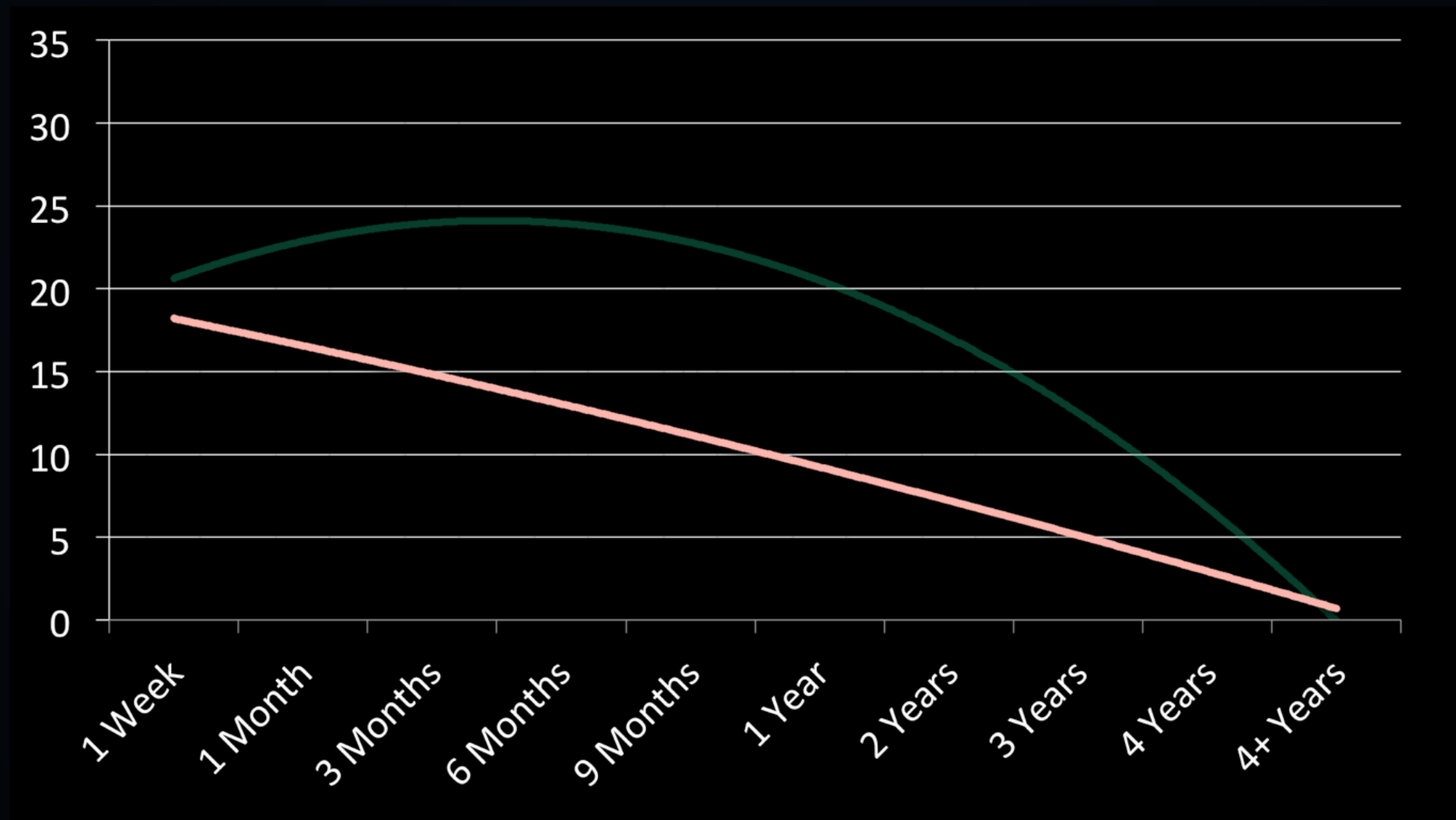




# EDUCATIONAL ADOPTION

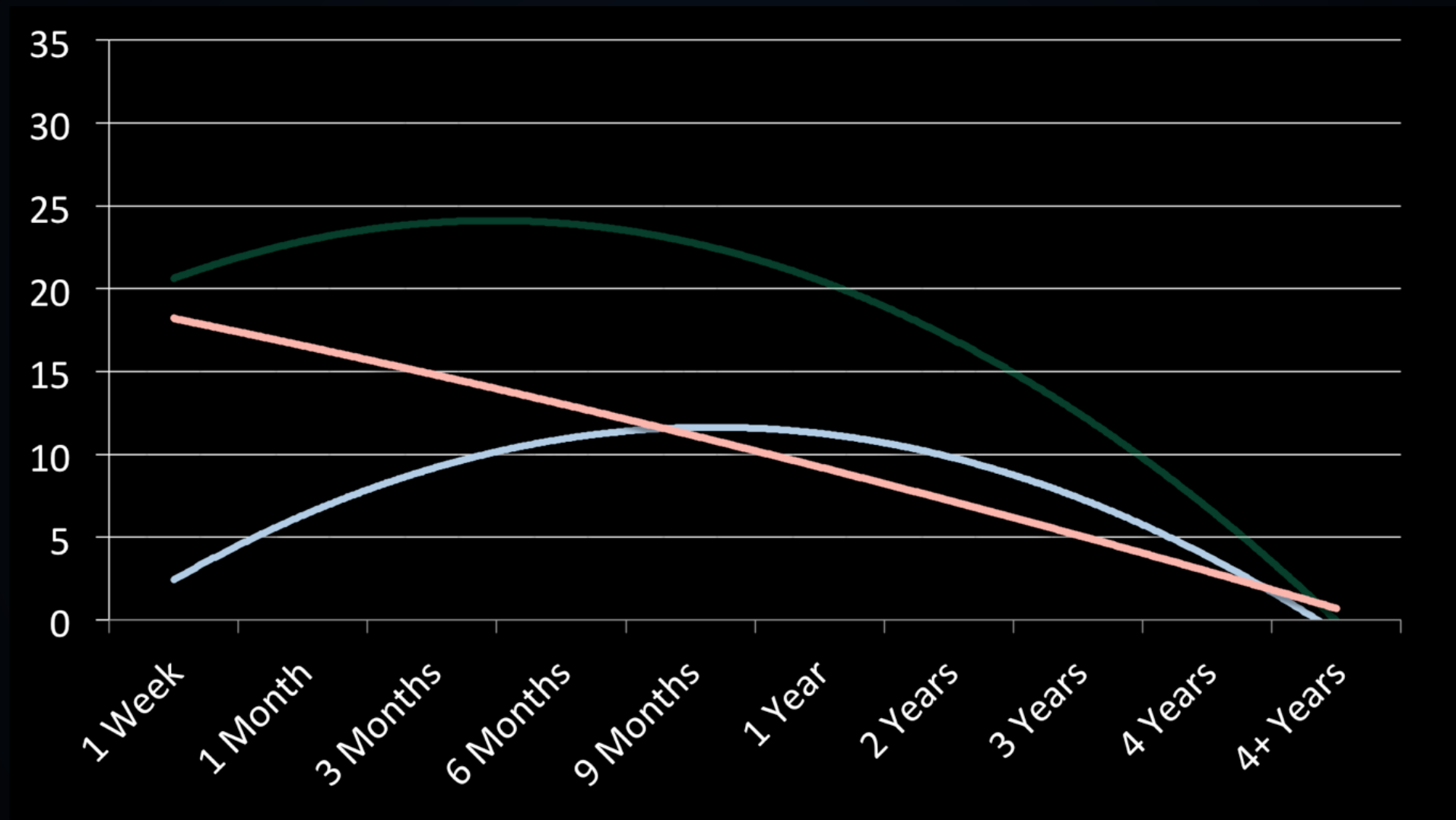


# EDUCATIONAL ADOPTION





# EDUCATIONAL ADOPTION





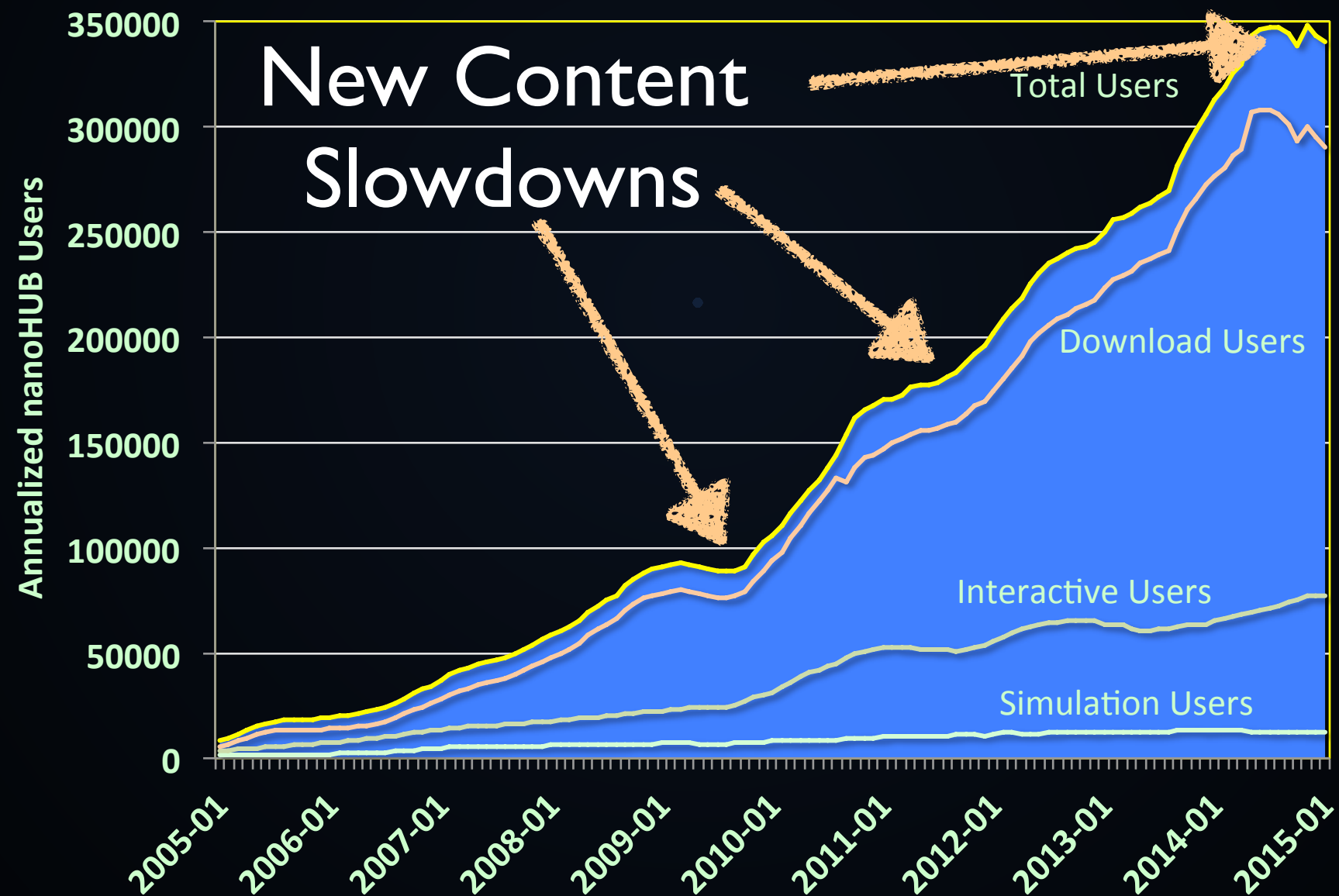


# LESSON 4

Fresh and excellent content



# CONTENT - GROWTH RELATIONSHIP







# LESSON 5

Assemble the Right Team





Mark Lundstrom,  
Core Science,  
Visionary







Gerhard Klimeck,  
Core Science,  
Scale-up





Gerhard Klimeck,  
Core Science,  
Scale-up



Content, Users





Michael McLennan,  
Scale-out  
(HUBzero)



Content, Users

Applications





# Google





Google







# LESSON 6

You're never done!

How can we have the same impact for  
people who work with data that  
we did for people who simulate?

Publication with DOI & Journal Partners  
Reputation with Thompson Reuters (Web of Science)  
Uncertainty Quantification



# DENSE DATA

Within Your Experiment, Your Knowledge is Somewhat Complete  
(in your home you know what you own)

Sphere:	Gold				
Medium:	Water				
Graph type:	Cext/Csca/Cabs v. wavelength				
Scattering angle = 0					
Wavelength	Ref Index (real)	Ref Index (imaginary)	Cext	Csca	Cabs
400	1.087871431	1.446879408	2.04E-15	1.23E-16	1.92E-15
401	1.087514072	1.447303652	2.09E-15	1.33E-16	1.96E-15
402	1.087156632	1.447727993	2.09E-15	1.31E-16	1.95E-15
403	1.08679913	1.448152407	2.08E-15	1.30E-16	1.95E-15
404	1.086441546	1.448576918	2.07E-15	1.29E-16	1.95E-15
405	1.086083131	1.449000474	2.07E-15	1.28E-16	1.94E-15
406	1.085723491	1.449422573	2.06E-15	1.27E-16	1.94E-15
407	1.085363769	1.449844767	2.06E-15	1.26E-16	1.93E-15
408	1.085003987	1.450267031	2.05E-15	1.24E-16	1.93E-15
409	1.084644123	1.450689391	2.05E-15	1.23E-16	1.93E-15

Dense





# (SOMEWHAT) SPARSE DATA

In Your Immediate Community, Knowledge is Less Complete  
(you know many of the things your neighbors own)

Liver Uptake

Brain Uptake

Gold



Sphere:	Gold				
Medium:	Water				
Graph type:	Cext/Csca/Cabs v. wavelength				
Scattering angle = 0					
Wavelength	Ref Index (real)	Ref Index (imaginary)	Cext	Csca	Cabs
400	1.087871431	1.446879408	2.04E-15	1.23E-16	1.92E-15
401	1.087514072	1.447303652	2.09E-15	1.33E-16	1.96E-15
402	1.087156632	1.447727993	2.09E-15	1.31E-16	1.95E-15
403	1.08679913	1.448152407	2.08E-15	1.30E-16	1.95E-15
404	1.086441546	1.448576918	2.07E-15	1.29E-16	1.95E-15
405	1.086083131	1.449000474	2.07E-15	1.28E-16	1.94E-15
406	1.085723491	1.449422573	2.06E-15	1.27E-16	1.94E-15
407	1.085363769	1.449844767	2.06E-15	1.26E-16	1.93E-15
408	1.085003987	1.450267031	2.05E-15	1.24E-16	1.93E-15
409	1.084644123	1.450689391	2.05E-15	1.23E-16	1.93E-15

Silver

Sphere:	Gold				
Medium:	Water				
Graph type:	Cext/Csca/Cabs v. wavelength				
Scattering angle = 0					
Wavelength	Ref Index (real)	Ref Index (imaginary)	Cext	Csca	Cabs
400	1.087871431	1.446879408	2.04E-15	1.23E-16	1.92E-15
401	1.087514072	1.447303652	2.09E-15	1.33E-16	1.96E-15
402	1.087156632	1.447727993	2.09E-15	1.31E-16	1.95E-15
403	1.08679913	1.448152407	2.08E-15	1.30E-16	1.95E-15
404	1.086441546	1.448576918	2.07E-15	1.29E-16	1.95E-15
405	1.086083131	1.449000474	2.07E-15	1.28E-16	1.94E-15
406	1.085723491	1.449422573	2.06E-15	1.27E-16	1.94E-15
407	1.085363769	1.449844767	2.06E-15	1.26E-16	1.93E-15
408	1.085003987	1.450267031	2.05E-15	1.24E-16	1.93E-15
409	1.084644123	1.450689391	2.05E-15	1.23E-16	1.93E-15



# (MORE) SPARSE DATA

In Your Extended Community, Knowledge is Less Complete  
(you know many of the things your neighbors own)





# (VERY) SPARSE DATA

In A Wide Community, Knowledge is Very Incomplete  
(you don't know what most people on earth own)

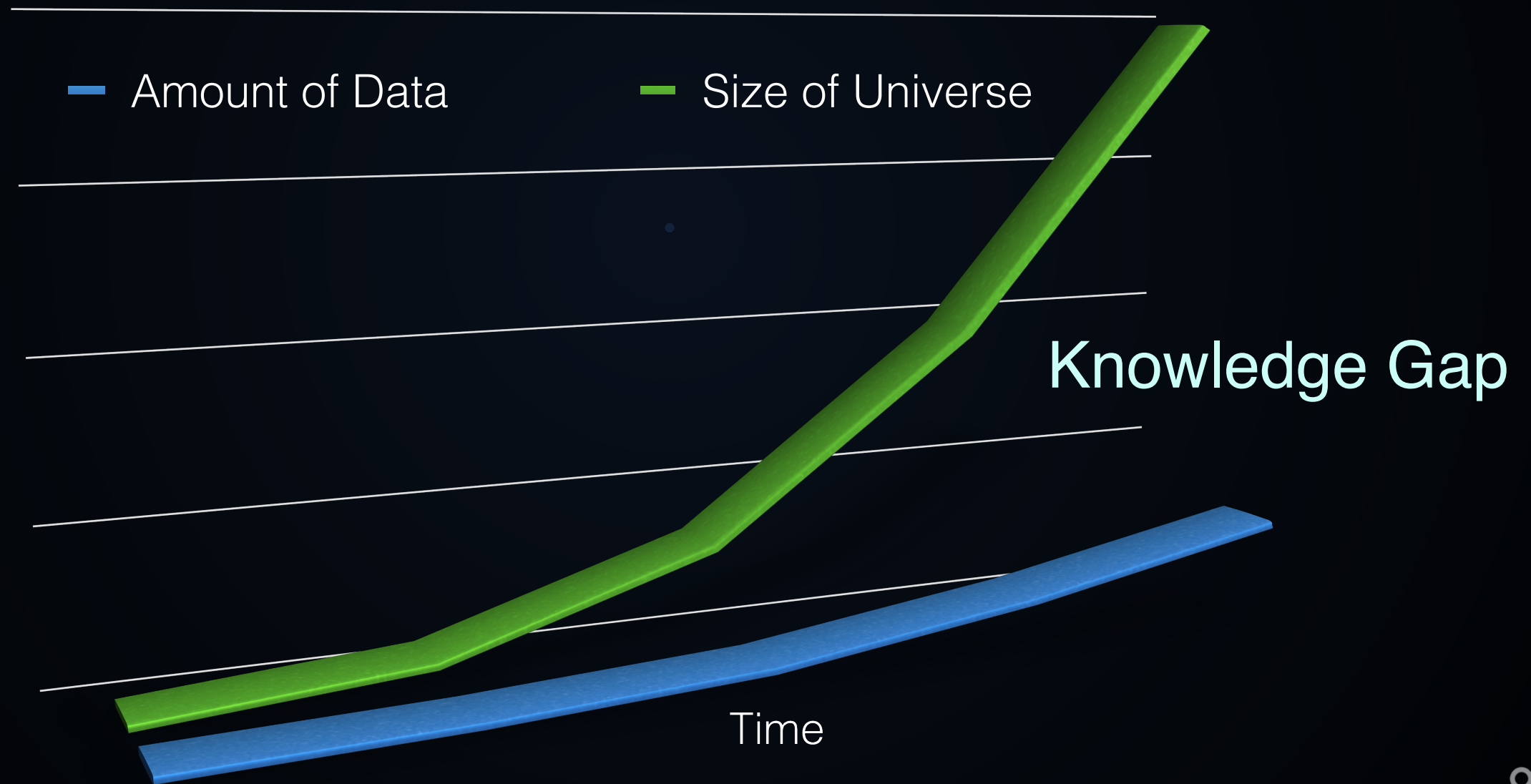
Electrical - Environmental - Medical - Physical - Financial - Interfacial - etc...

Wide  
Variety  
of  
Materials



# KNOWLEDGE VS UNIVERSE

The more we have, the less we know...





# THE TROUBLE WITH TABLES...


Dense tables are wonderful for sort, search, and filtration

Run Meta Data				Inputs				Outputs			
User	Time	Tool	etc.	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	etc	O <sub>1</sub>	O <sub>2</sub>	O <sub>3</sub>	etc

Sparse tables are terrible for sort, search, and filtration



# EXPLORATION

Run Meta Data				Inputs				Outputs			
User	Time	Tool	etc.	$I_1$	$I_2$	$I_3$	etc	$O_1$	$O_2$	$O_3$	etc
											





# EXPLORATION

How do we effectively explore a sparse data landscape?





# IDENT: PARTNERSHIP WITH NANOMATERIAL REGISTRY

Interactive Data Exploration & Navigation Tool  
for Nano Technology

Nothing selected Use columns Custom column 288 total rows Edit filters

ID	Name	Description
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

NMRI data +

**Giving Data  
The Spark of Life**

Visit our evolving project for  
Nanomaterial Registry data  
exploration

**NANOMATERIALREGISTRY**





# IDENT

## COLUMN POPULATION LOOKAHEAD

Nothing selected Use columns Custom column

Search

1 Absorbance:as a function of Solvent

1 Absorbance:as a function of Time

2 pH

3 Size:as a function of Light Exposure

4 Size:as a function of Temperature

5

6

7 Surface Area

8 Bulk Density (g/cm^3)

9 Specific Surface Area

1

Surface Charge

Electrokinetic Mobility

Zeta Potential (mV)

**Bar Length Indicates Amount of Non-Null Data**




# IDENT

## COLUMN OF INTEREST SELECTION

Nothing selected ▼ Use columns Custom column

288 total rows Edit filters

### Table of chosen data

ID	Mean Diameter (nm)	Mean Hydrodynamic Diameter (nm)	DisplayName	
6			Ag	
7	7.2		Ag NP	
8	20.8	32	Ag NP	
9	29	39.6	Ag NP	
10	43.4	47.6	Ag NP	
11	41.9	43.1	Ag NP	
12	49.1	59.2	Ag NP	
13	53.5	57.1	Ag NP	
14	57.7	68.3	Ag NP	
15	8.2		Ag NP	
16	19.2		Ag NP	

Nanomaterials Registry Dataset

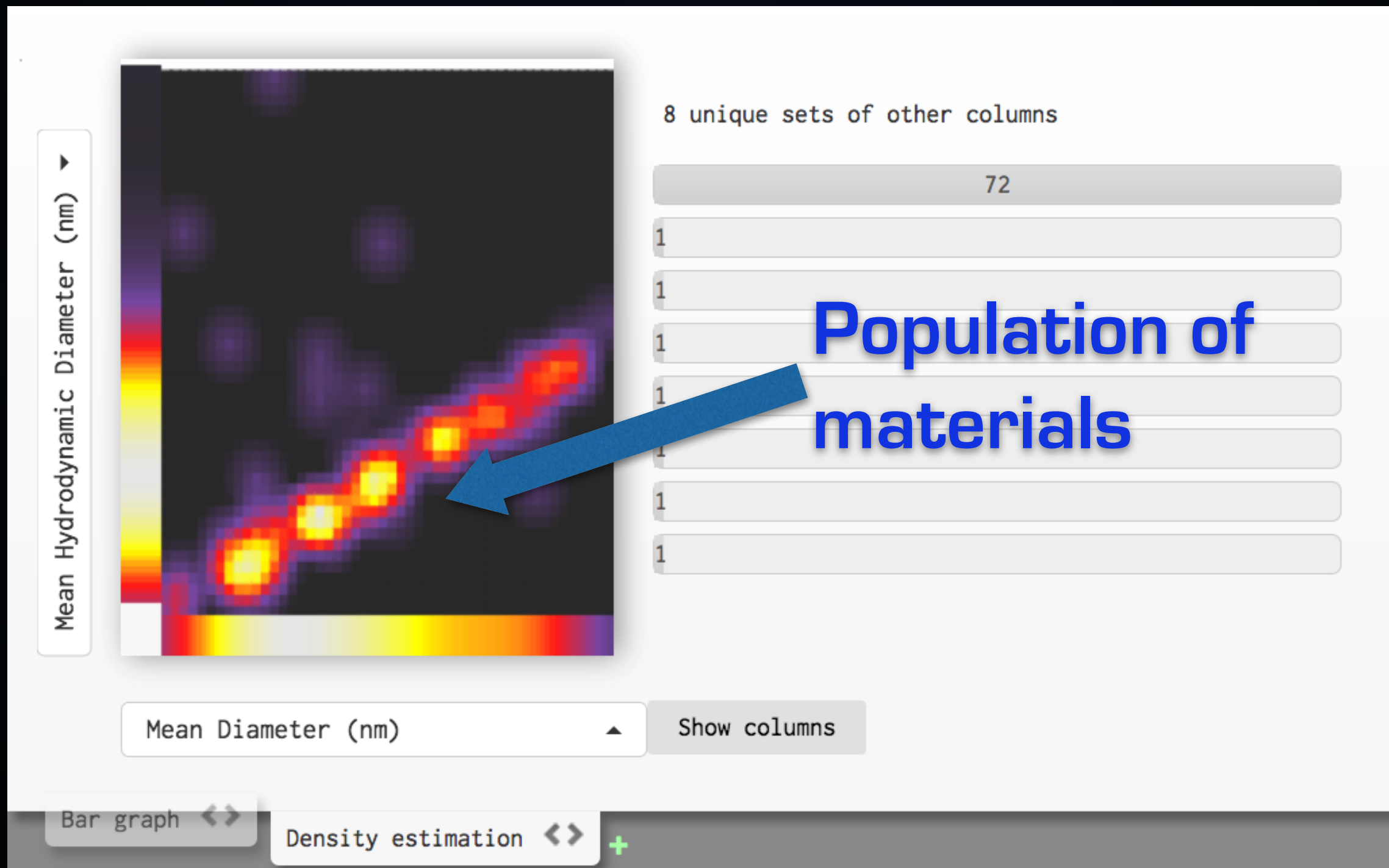
### Visualize Content

Add a visualization +



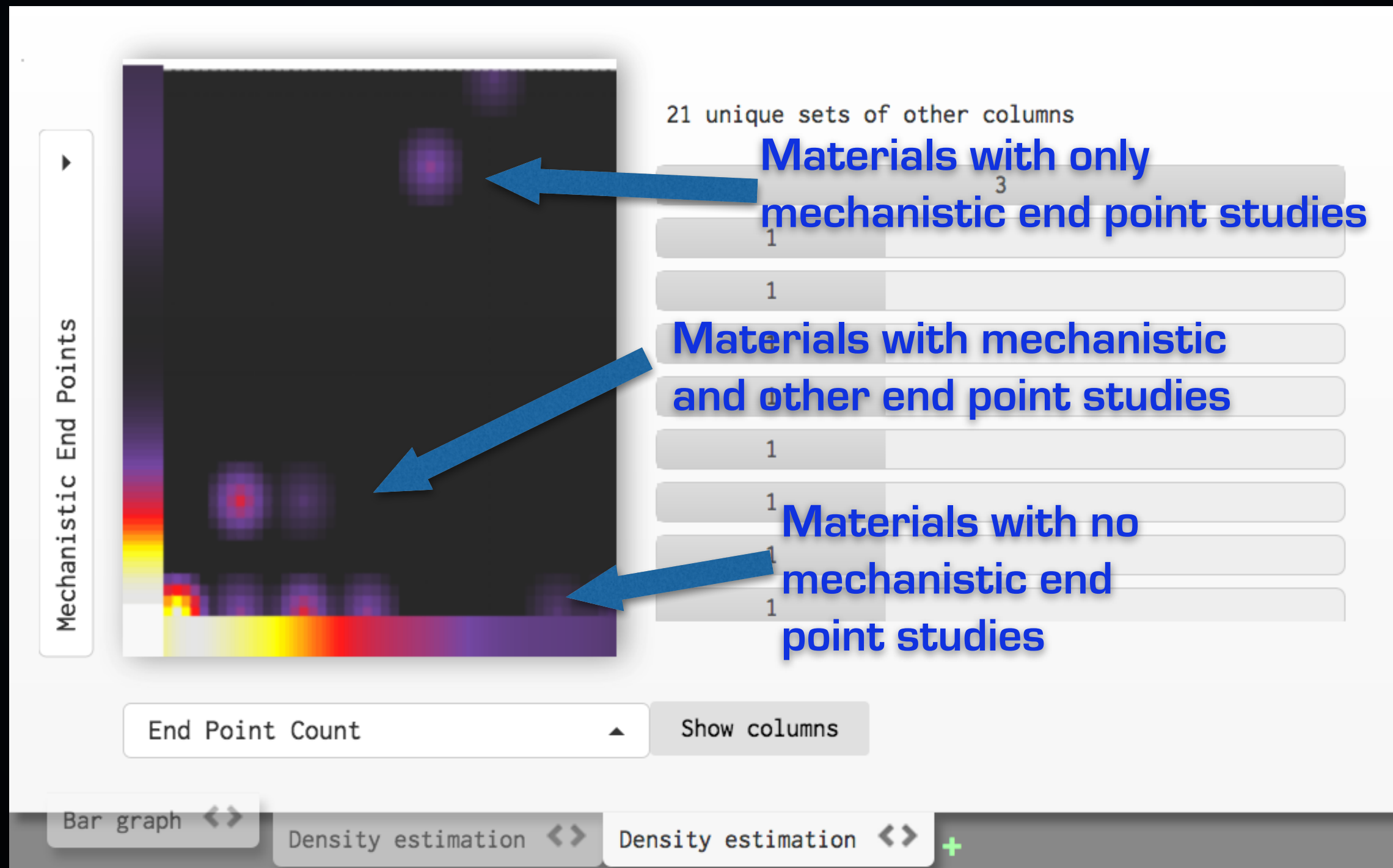


# IDENT VISUALIZATION OF CONTENT



# IDENT

## VISUALIZATION OF KNOWLEDGE








# LESSON DIGRESSION

Ongoing Project:

Use Your Users to Help Each Other



# DATA ABOUT HOW PEOPLE USE DATA




All ▾ grill












Shop by Department ▾

mgzentner's Amazon.com Today's Deals Gift Cards Sell Help ▶

Patio, Lawn & Garden Best Sellers Deals Outdoor Décor ▾ Gardening ▾ Grilling ▾ Patio Furniture ▾ Wedding Registry ▾ Mowers & Landscaping Tools ▾ Pools & Spa Supplies ▾

◀ Back to search results for "grill"





## Char-Broil Classic 40,000 BTU 4-Burner Gas Grill w

by [Char-Broil](#)

★★★★★ ▾ [470 customer reviews](#) | [234 answered questions](#)

**#1 Best Seller** in [Freestanding Grills](#)

---

List Price: \$199.99  
Price: **\$169.99** ✓ **Prime**  
You Save: **\$30.00 (15%)**

**In Stock.**


Ships from and sold by Amazon.com **exclusively for Prime members.**

**Want it Saturday, Aug. 22?** Order within **19 hrs 6 mins** and choose **Saturday Del**

- Char-Broil four burner, 40,000 BTU gas grill with 10,000 BTU lidded side burner
- 480 sq. in. of primary cooking on porcelain-coated cast iron grates plus 180 sq. i
- Large, painted metal side shelves offer lots of workspace
- Stainless steel lid, handle, control panel, and fascia add style and durability to th
- Electronic ignition system offers a reliable spark with every push

**30 new** from **\$169.99**

This item's packaging will be visible when delivered and cannot be gift-wrapped.

**Free One-Day Shipping on**  
**Purdue Textbooks** [Learn more](#) 





# DATA ABOUT HOW PEOPLE USE DATA



All ▾ grill

Shop by Department ▾ mgzentner's Amazon.com Today's Deals Gift Cards Sell Help ▶

Patio, Lawn & Garden Best Sellers Deals Outdoor Décor ▾ Gardening ▾ Grilling ▾ Patio Furniture ▾ Wedding Registry ▾ Mowers & Landscaping Tools ▾ Pools & Spa Supplies ▾

◀ Back to search results for "grill"





All ▾ coffee coals

Shop by Department ▾ mgzentner's Amazon.com Today's Deals Gift Cards Sell Help ▶

Patio, Lawn & Garden Best Sellers Deals Outdoor Décor ▾ Gardening ▾ Grilling ▾ Patio Furniture ▾ Wedding Registry ▾ Mowers & Landscaping Tools ▾ Pools & Spa Supplies ▾

◀ Back to search results for "coffee coals"



## Coffee Coals Charcoal

by [Coffee Coals](#)

★★★★★ ▾ 5 customer reviews

**#1 New Release** in [Charcoal Starters](#)

Price: **\$6.15** ✓ **Prime**

**In Stock.**

Sold by [Coffee Coals](#) and [Fulfilled by Amazon](#). Gift-wrap available.

**Want it Saturday, Aug. 22?** Order within **19 hrs 35 mins** and choose **Saturday**

- Ready to cook in 10 minutes
- Rich and smoky charcoal aroma - does not smell like coffee
- Completely all natural - no petroleum distillates or synthetic binders
- Single serving burns at grilling temp for 50-60 minutes

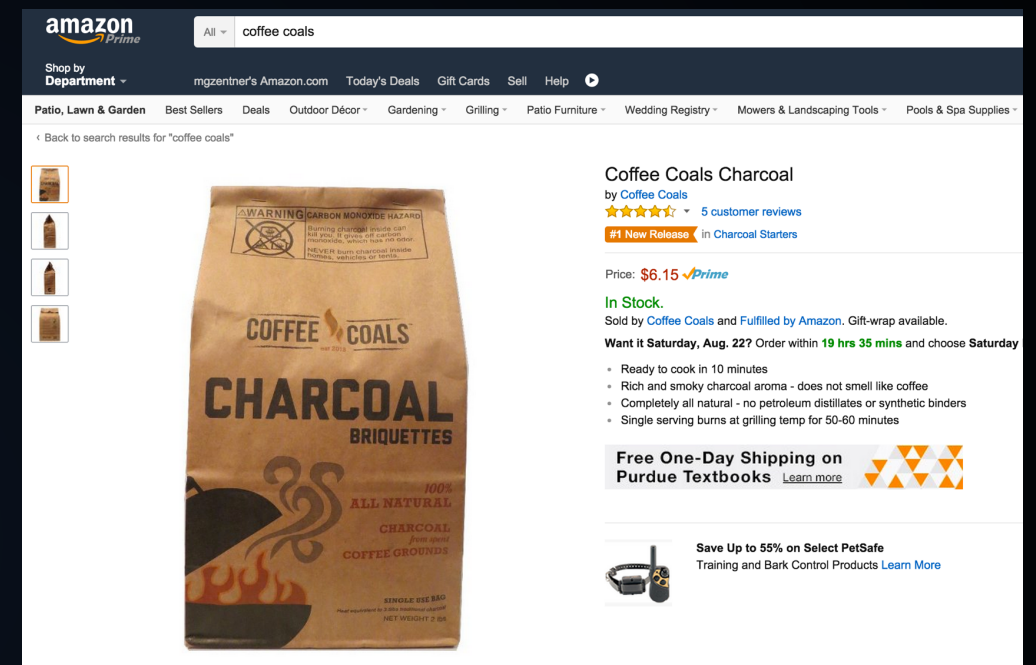
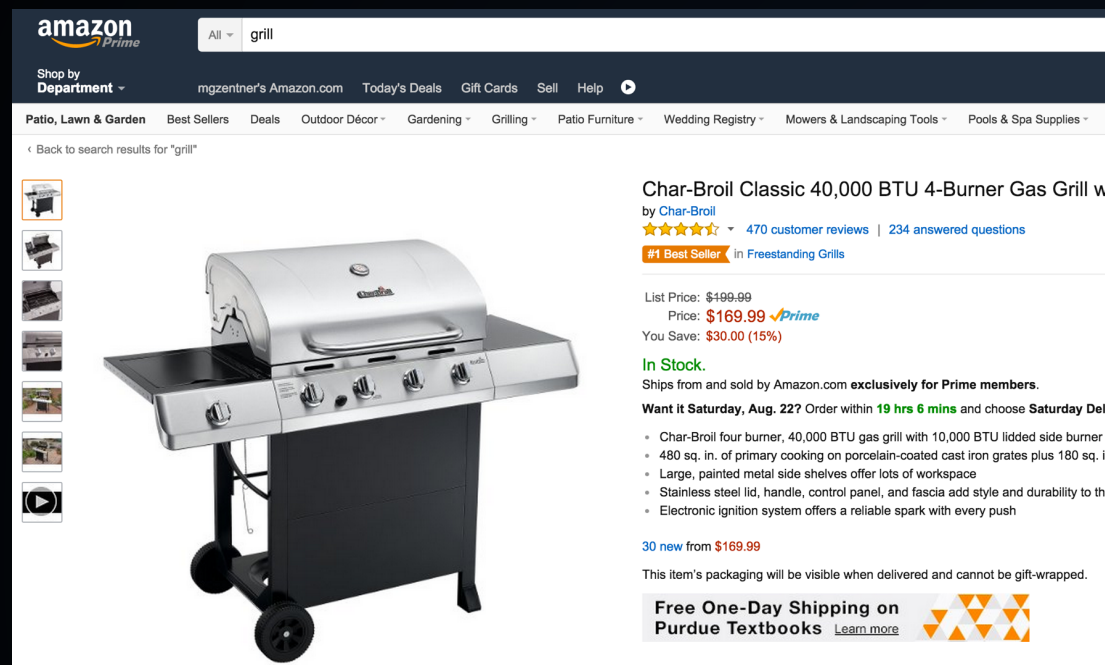
**Free One-Day Shipping on Purdue Textbooks** [Learn more](#)



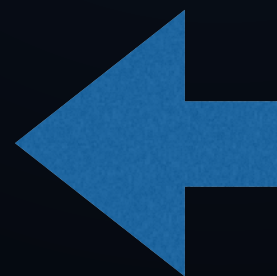
**Save Up to 55% on Select PetSafe**  
Training and Bark Control Products [Learn More](#)



# DATA ABOUT HOW PEOPLE USE DATA



Pat LaFrieda Aged Ribeye,  
500 F,  
15g seasoning,  
4 minutes on a side,  
10 minute rest...



This is exactly the kind of  
information watching our  
users explore can tell us!





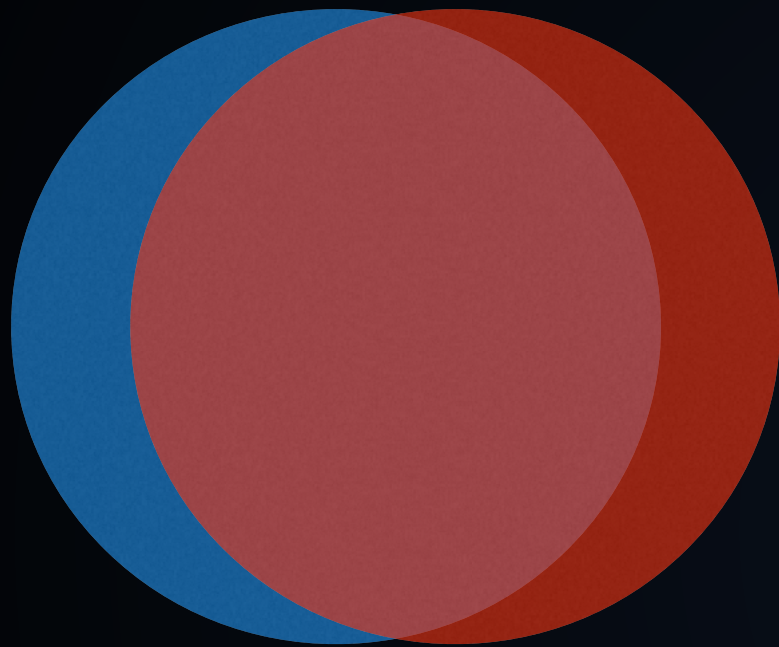
# THE STUDY OF HOW PEOPLE STUDY DATA

---



# THE STUDY OF HOW PEOPLE STUDY DATA

---



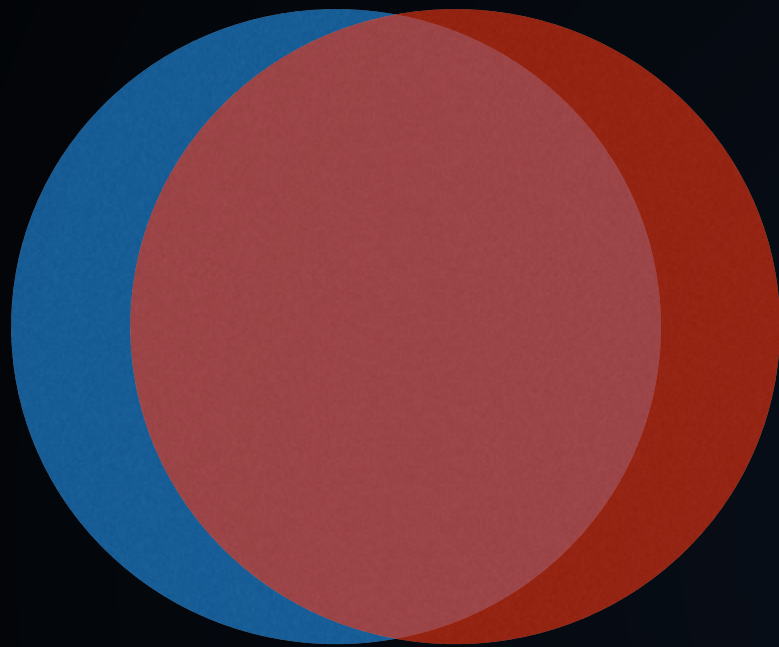
Small Digressions



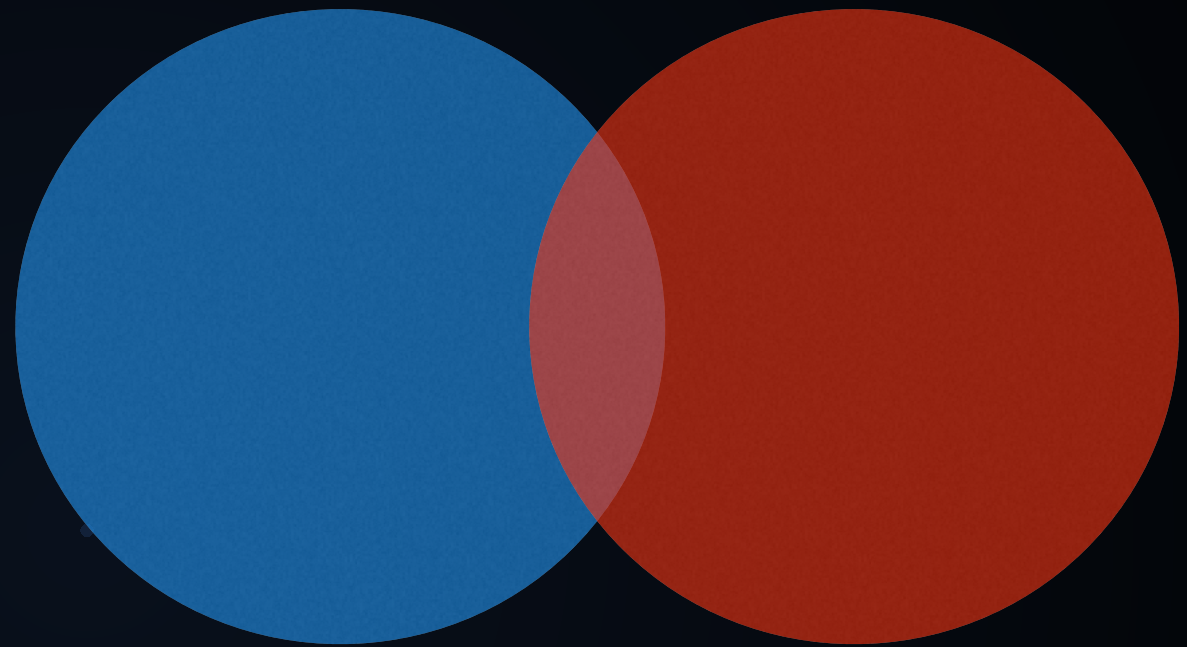


# THE STUDY OF HOW PEOPLE STUDY DATA

---



Small Digressions

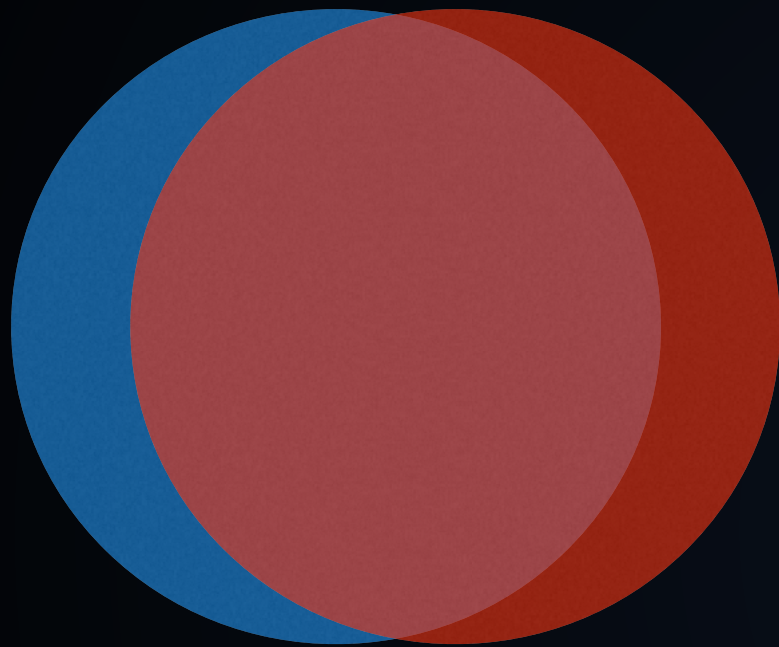


New Directions

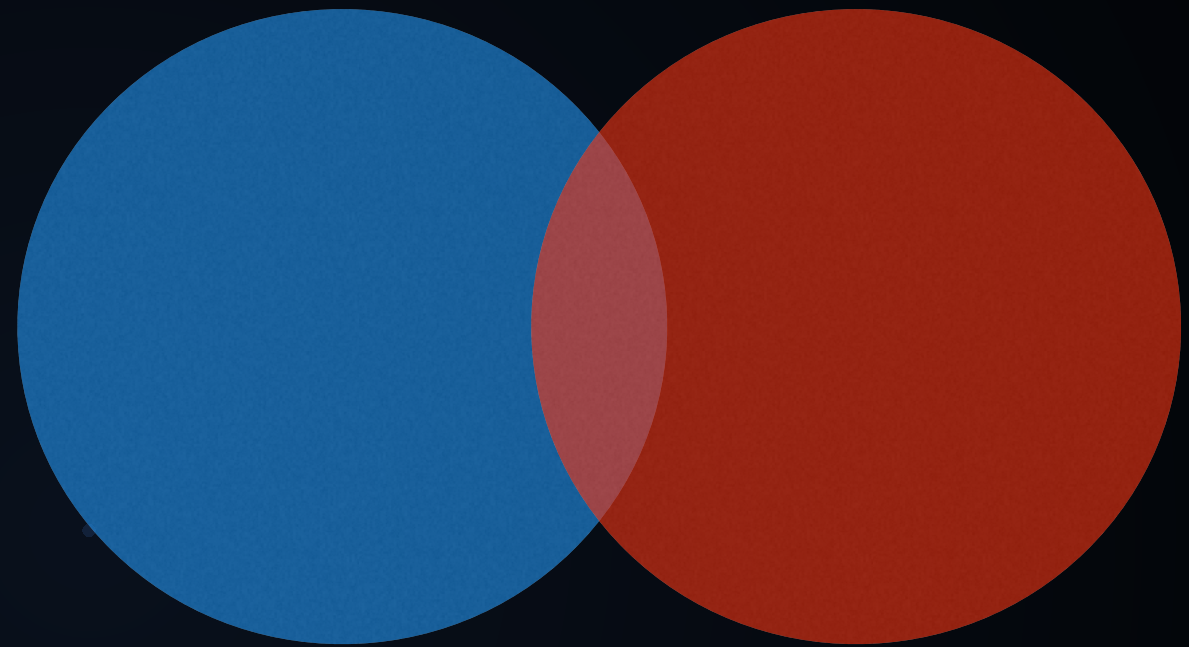


# THE STUDY OF HOW PEOPLE STUDY DATA

---



Small Digressions



New Directions



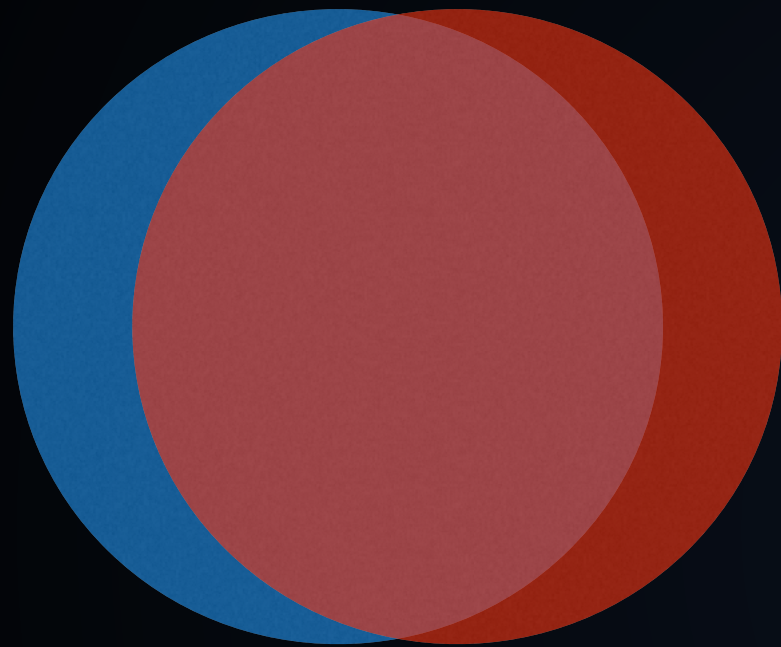
New Fields



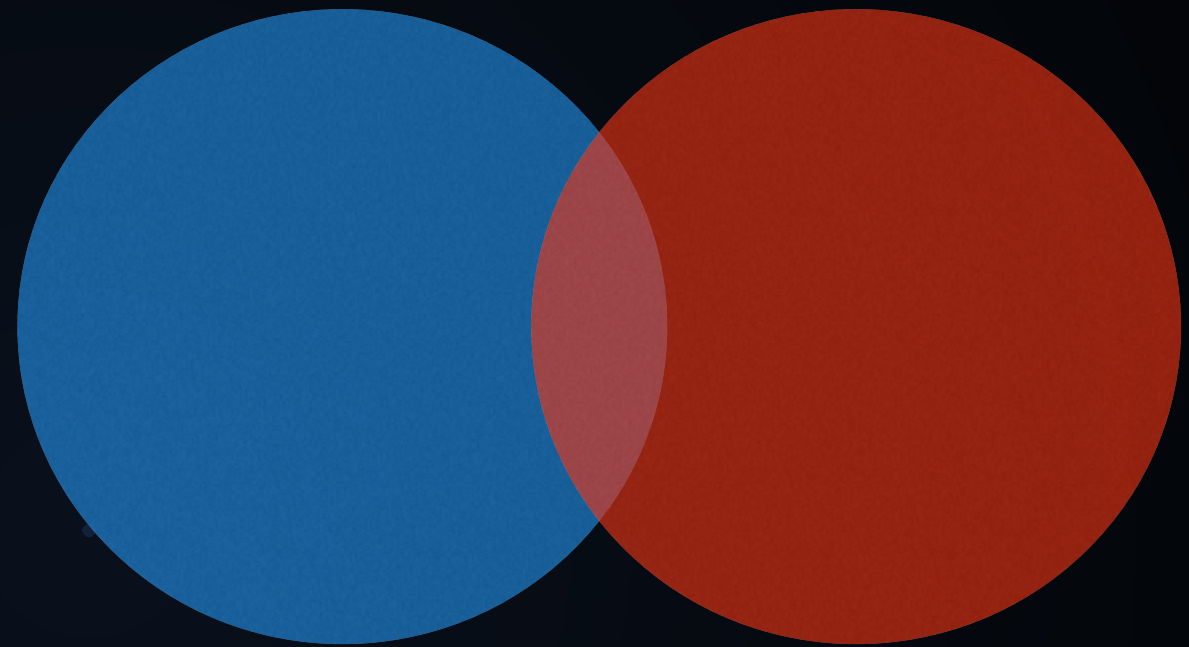


# THE STUDY OF HOW PEOPLE STUDY DATA

---



Small Digressions



New Directions



New Fields



?????







# LESSON 7

Begin Thinking About  
Sustainability Early



# THE DIFFERENCE A YEAR MAKES

---

Year<sub>i</sub>:

*“We are working on the following for sustainability...”*

*“Why are you telling us about sustainability when you have N years left?”*

*(Experience from startups tells us that you have 2 years to try a business model. That’s N/2 tries.)*

Year<sub>i+1</sub>:

*“In the next N-1 years...”*

*“Why do you assume you have N-1 years? You should look at how to become sustainable.”*







# LESSON 8

## Assemble the Right Team

NSF: Lynn Preston, Eduardo Misawa, Keith Roper

- Mark Lundstrom
- Gerhard Klimeck
- Michael McLennan
- Gerry McCartney
- George Howlett
- Alejandro Strachan
- Shawn Rice
- Betsy Hillery
- Steve Clark
- Derek Kearney
- Ben Haley
- Chris Smoke
- Alissa Nedossekina
- Nick Kisseberth
- Steve Snyder
- Nathan Denny
- Dwight McKay
- Sam Wilson
- Rick Kennell
- David Benham
- Pascal Meunier
- Kevin Wojkovich
- Martin Hunt
- Ilya Shunko
- Lynn Zentner
- Tanya Faltens
- John Wright
- Vicky Johnson
- Israa Bukhari
- Swaroop Samek
- Jeff Turkstra
- Leslie Schumacher
- Emily Kayser
- Erich Huebner
- Nikki Huang
- Vicky Farnsworth



# A FEW EVENTS



HUBbub 2015

14-16 Sept, 2015

Sheraton Indianapolis City Center Hotel

<http://hubzero.org/hubbub>

nanoHUB Users Conference

31 August - 1 September

Purdue University

In conjunction with International Workshop on Computational Electronics

<https://nanohub.org/groups/conference>

HUBzero - Sustainable Manufacturing Activities?