Making Massive Computational Experiments Painless

Hatef Monajemi[†], David Donoho[†] and Victoria Stodden[‡]

[†]Department of Statistics, Stanford University
[‡]School of Information Sciences, University of Illinois at Urbana-Champaign

2016 IEEE Open Science in Big Data Workshop
December 5, 2016

Paths to knowledge

Traditionally:

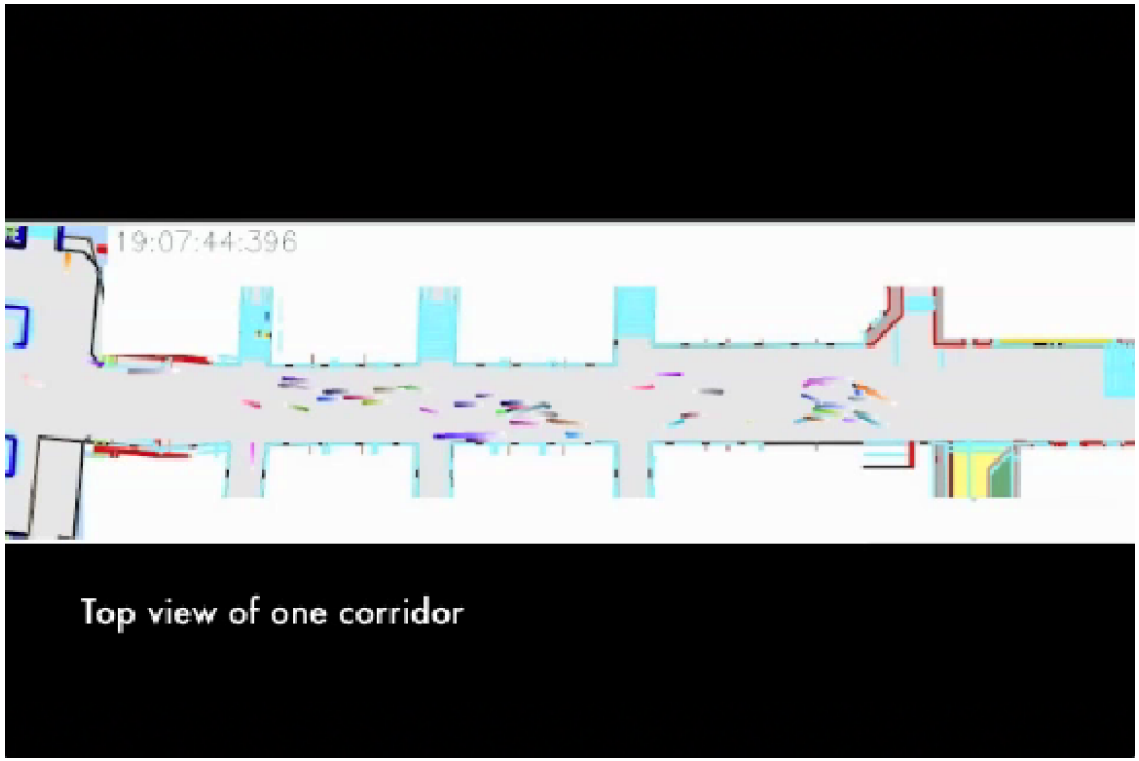
- Deduction
  - derivations and proofs in mathematical sciences
- Induction
  - measurements and hypothesis driven experimentation in physical and biological sciences

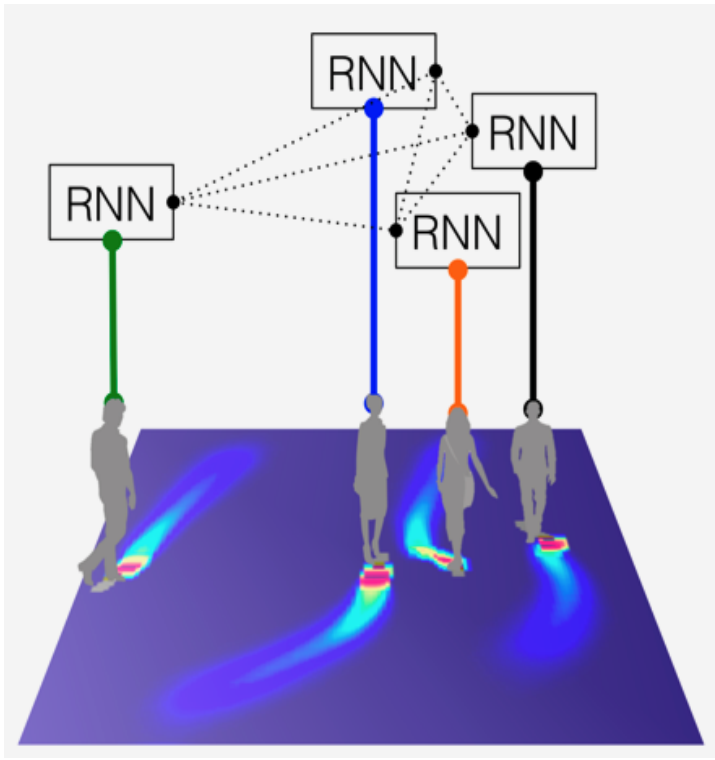
Today:

- Massive scientific computations
  - computational experiments involving several million CPU hours

Examples abound...

- Forecasting human trajectory in crowded spaces using deep learning (Stanford Vision Lab - Alahi et al. 2016)





Examples abound…

- Several million CFD simulations to find optimal design strategies for oil field development (Shirangi's Ph.D. work at Stanford 2016).

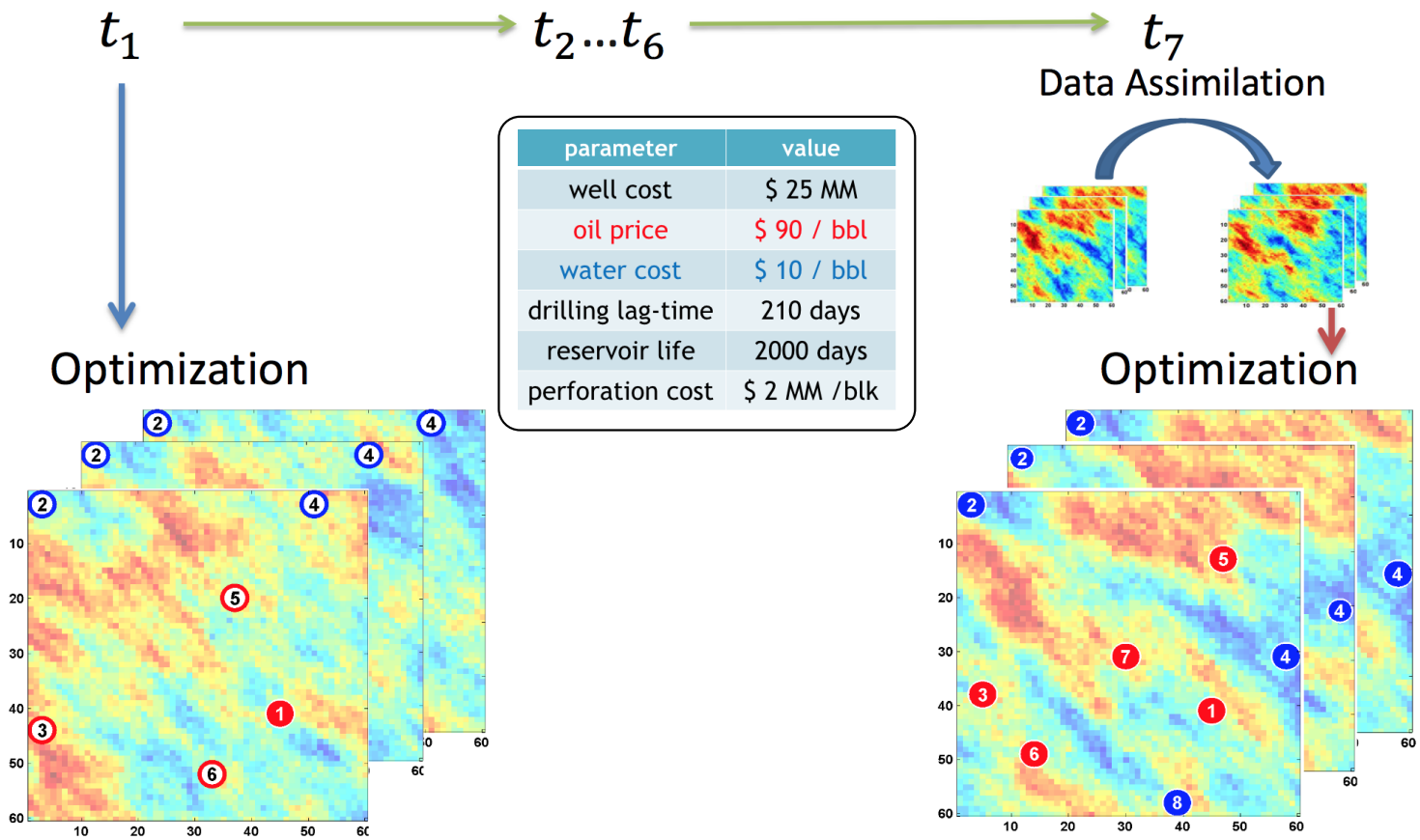
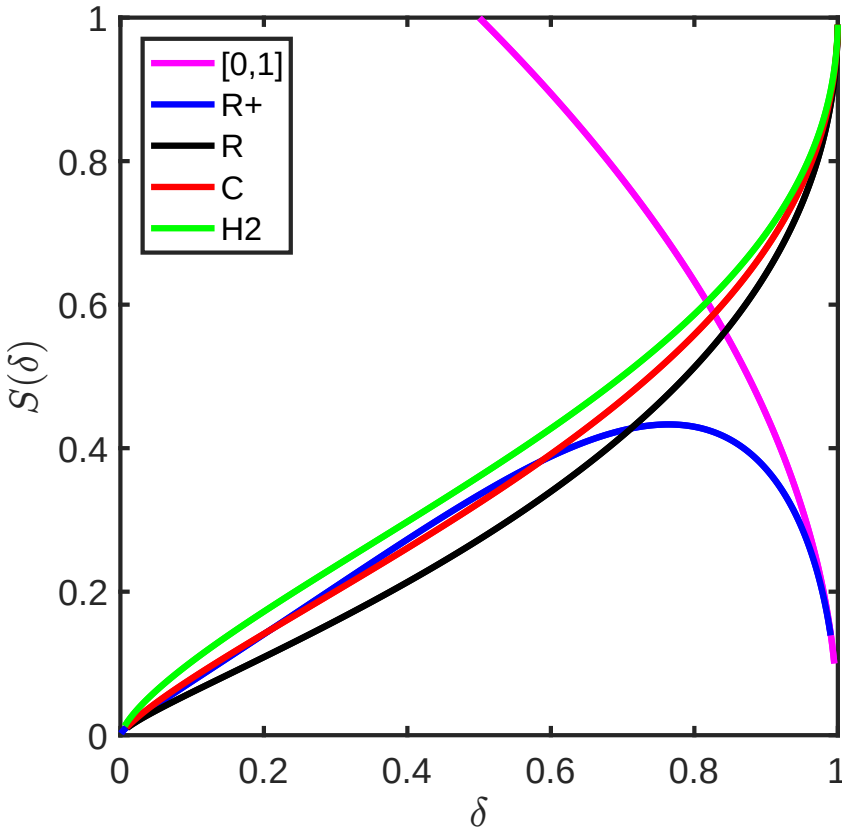
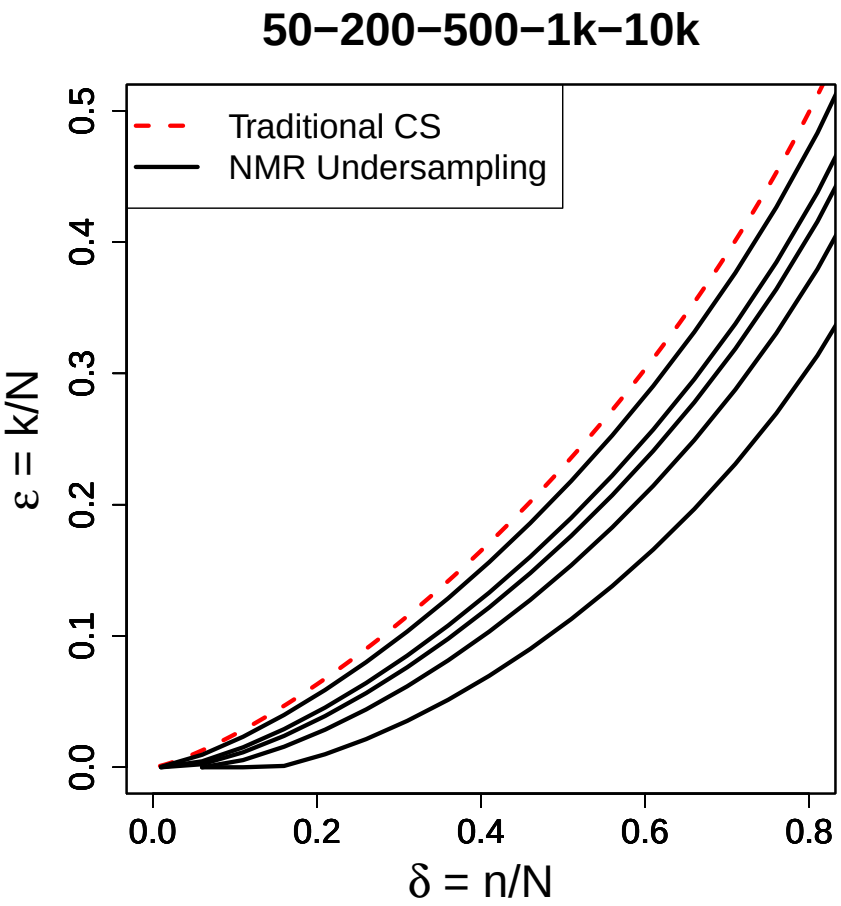


Figure: Closed-loop Field Development (Shirangi & Durlofsky 2015)

Examples abound

- Breakthroughs in applying compressed sensing - more than 50 million separate convex optimizations (Monajemi & Donoho 2016)





Why are massive computational experiments emerging?

- Increasing **complexity** of research questions
  - Mathematical analysis provide answer to *idealized (simplified)* situation
- Questions concern systems described **in silico** or historical databases
  - Which algorithm is better
  - Which configuration is optimal
  - What would be the trade-off between goals A and B
- Explosion of **computational resources** globally
  - open-source software, ubiquitous internet connection, massive distributed databases, cloud computing (AWS, Google, Microsoft)
- Ability to do **breakthrough science**

Amazing capacity, not many involved scientists

There aren't many scientists involved as there ought to be! What is holding them back?

- Complexity of using cloud computing environments
- Problematic habit of using *Interactive Computing* paradigm

Amazing capacity, not many involved scientists

There aren't many scientists involved as there ought to be! What is holding them back?

- Complexity of using cloud computing environments
- Problematic habit of using *Interactive Computing* paradigm
  - Cannot track massive volume of work
  - Cannot track and fix inevitable errors in large-scale computations
  - Often ends in unfinished projects or incorrect results in massive studies

A new era of scientific computing is upon us

In field after field, the modern research community will develop new expectations about the role of computing

1. Most central contribution is to **dream up new experiments**
2. **Reputation** by doing massive experiments with impeccable technique
3. The "royal road" to scientific reputation passes through **assembling and/or building tools** enabling impeccable technique and outstanding productivity in doing massive computational experiments.

A new computing paradigm is emerging

The new emerging paradigm will:

- make it **painless** to do massive computational experiments
- make it **easy to understand what took place** in a large experiment
  - + trust in computational results and depend on them as reliable science
- drastically **improve the quality and productivity** of computational science - once it crystallizes and spreads widely.

Computational experiments, painlessly

In our telling, a computational experiment involves:

1. **Precise specification**: defining a performance metric and a grid of different systems to be compared.
2. **Distribution and management** of all the jobs
3. **Harvesting** of all the data generated by all the jobs
4. **Analysis** of the data produced
5. **Reporting** and dissemination of results.

The new paradigm will seamlessly integrate and automates all these 5 tasks

Disederata of the sought-after paradigm

An *experiment-definition system* (EDS) where the conduct of the experiment follows inexorably and automatically from the mere definition of the experiment itself.

1. **Legibility**: human-readable
2. **Simplicity**: pushing a single button.
3. **Productivity**: all the tedium of large experiments obviated.
4. **Durability:** easily retrievable at a later time.
5. **Transparency**: easily be understood post facto
6. **Reproducibility**: all the tasks happen in a reproducible way
7. **Scalability**: massive scaling-up of experiments

Our effort in building painless computing platform

- In 2012, we began building **ClusterJob (CJ)** to facilitate massive computational experiments
- In 2015, we made CJ open-source
- CJ has been used by us and other researchers to conduct million-CPU-hour computational experiments on Stanford campus
- People who used it, love it!
  - "Your software has made my life much easier" - Charles Chang (Stanford Social Science Postdoc)
  - "This is how it [computation] should be done" - Veniamin Morgenshtern (Stanford Statistics Postdoc)

Conclusion

- Changes in science is forcing us inescapably to do massive computational experiments

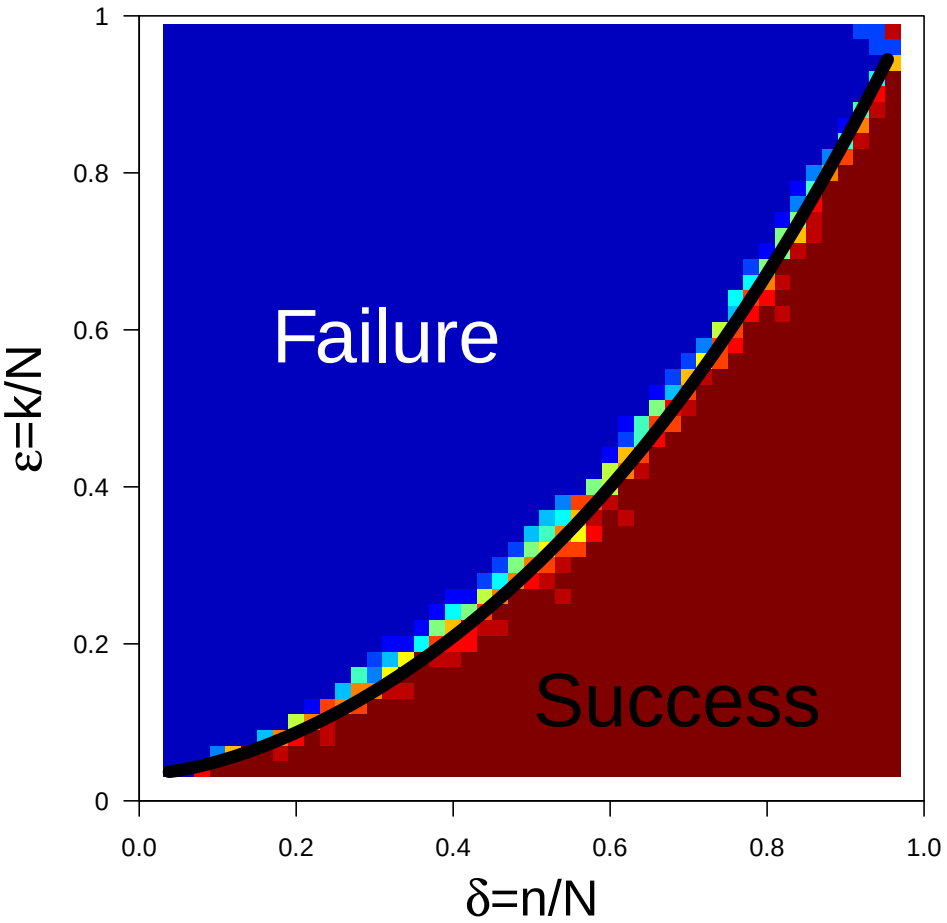
  > *"[In 2065] mathematical derivation and proof will not trump conclusions derived from state-of-the-art empiricism" - Prof. David Donoho (50 years of Data Science)*

- Using massive computational experiments, one can do *scientific breakthroughs and discoveries*
- We are confident that a new computing paradigm is emerging that makes it painless to do large amount of computations
- The contours of this new paradigm are clear, but details may vary

Extra Slides

How does CJ work, an example

Compressed Sensing Phase Transition Experiments:



How does CJ work, an example

Write a *simple and decipherable* MATLAB script:

```
% test.m
% This test code calculates the
% probability of successful
% reconstruction in compressed sensing.
% Author: Hatef Monajemi Nov 1 2016

file = 'results.txt';
delta   = 0.1:.1:.9;
epsilon = 0.02:0.02:0.98;
for i = 1:length(delta)
for j = 1:length(epsilon)
  pr = computeProb(delta, epsilon);
  fid = fopen(file,'at');
  fprintf(fid, '%3.2f,%3.2f,%3.2f\n', ...
                  delta,epsilon,pr);
  fclose(fid)
end
end
```

How does CJ work, an example

Let CJ handle the rest.

- Submit 441 separate jobs by a simple command

```
$ cj parrun test.m corn -dep bin -m "Test PT"
```

How does CJ work, an example

Let CJ handle the rest.

- Check status of jobs

```
$ cj state 8ab7a5aa

pid 8ab7a5aafa1b8232cc3da05a7814bed1d21dd0aa
remote_account: monajemi@sherlock.stanford.edu
1       10097772        COMPLETED
2       10097773        COMPLETED
3       10097774        COMPLETED
            .
            .
            .
441     10097786        RUNNING
```

How does CJ work, an example

Let CJ handle the rest.

- Retrieve information

```
$ cj log

pid 8ab7a5aafa1b8232cc3da05a7814bed1d21dd0aa
date: 2016-Oct-08  11:47:37  (GMT -07:00:00)
user: monajemi
agent: 2DCA5476-8197-11E6-B8C8-3A835C8A0BAC
account: monajemi@corn.stanford.edu
script: test.m
initial_flag: parrun

          Test PT
```

How does CJ work, an example

Let CJ handle the rest.

- Easily harvest results

```
$ cj reduce results.txt 8ab7a5aa
```

Other available systems

- CodaLab
- Image Harvest
- VisTrail
- Torch
- Pegasus

Problems:

- Many focus on “reproducibility” rather than to ease scalability
- Some that allow scalability pay little to no attention to reproducibility
- “Reproducibility” and “scalability” are equally important and intertwined