Scientific Benefits of a *Shared Experimental Framework*: On Developing Public Keystroke-Dynamics Resources

Kevin Killourhy
Carnegie Mellon University
ksk@cs.cmu.edu

Giving instructions

- Teaching exercise:

How do I make a peanut-butter sandwich?

- Typical instruction:

Put the peanut-butter on the bread with the knife

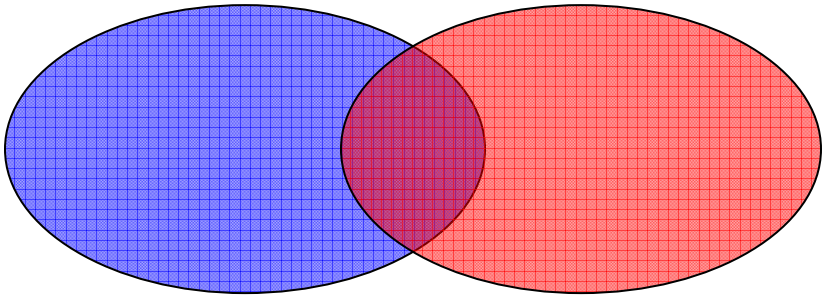
Giving instructions

Put the peanut-butter on the bread with the knife

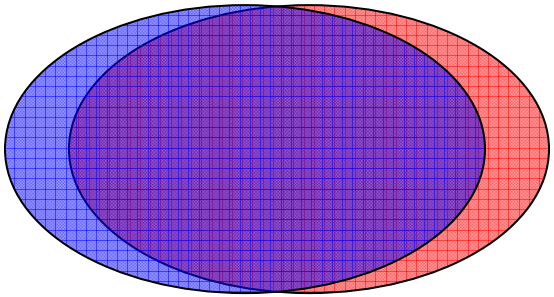
Giving instructions

- Clarity is both important and hard
- Be aware of potential misinterpretations
- Instructions assume a shared context

Little Sharing



Lots of Sharing



- Problems arise when there is little actual sharing

Key ideas

- **Methods are instructions**
  - for reproducing an experiment
  - that explain what was done and why
  - that *must* be understood by other researchers

- **A shared experimental framework provides context**
  - comprised of data, tools, and algorithms
  - publicly available for researchers' use
  - provide context necessary for understanding
  - make methods clearer and more accurate
  - promote scientific principles and accelerating progress

Outline

1. **Problem:**
   Ambiguous methods in keystroke dynamics

2. **Solution:**
   A shared repository of data, tools, and algorithms

3. **Tradeoffs:**
   Discussion about shared experimental frameworks

Typical methods section

- The methods are ambiguous:
  - Subject recruitment and task
  - Data-collection methods
  - Data-analysis procedure
- The problems:
  - Uncertainty threatens reproducibility.
  - Hides (alternative) explanatory factors.
- Reproducibility, explanation, and prediction are hallmarks of science.
  - No one knows how, why, or if my detector works.
  - It looks promising but would you deploy it?

Typical methods section

We recruited 10 subjects from the university community. Each one chose a 10-character password and typed it 200 times over the course of two weeks. We had 5 additional subjects act as impostors and type each of the other subjects' passwords 10 times each.

A keystroke logger recorded each keystroke along with a timestamp (10ms accuracy). Each correctly-typed password was converted into a 19-element vector of 10 hold times and 9 keydown-keydown times.

For each subject, an auto-associative neural network was trained using the first 100 passwords from the subject. The network was used to generate anomaly scores for the subject's remaining passwords, and for the impostor passwords. We generated ROC curves and recorded the network's average equal-error rate across all subjects.

Subject recruitment and task

We recruited 10 subjects from the university community. Each one chose a 10-character password and typed it 200 times over the course of two weeks. We had 5 additional subjects act as impostors and type each of the other subjects' passwords 10 times each.

Subject recruitment and task

We recruited 10 subjects from the university community. Each one chose a 10-character password and typed it 200 times over the course of two weeks. We had 5 additional subjects act as impostors and type each of the other subjects' passwords 10 times each.

- Subject demographics are ambiguous (staff vs. students)
  - It makes a significant difference: 15.7% vs 21.9% EER

Subject recruitment and task

We recruited 10 subjects from the university community. Each one chose a 10-character password and typed it 200 times over the course of two weeks. We had 5 additional subjects act as impostors and type each of the other subjects' passwords 10 times each.

- Subject demographics are ambiguous (staff vs. students)
  - It makes a significant difference: 15.7% vs 21.9% EER
- The actual task is ambiguous (ten sessions vs. two)

Subject recruitment and task

We recruited 10 subjects from the university community. Each one chose a 10-character password and typed it 200 times over the course of two weeks. We had 5 additional subjects act as impostors and type each of the other subjects' passwords 10 times each.

- Subject demographics are ambiguous (staff vs. students)
  - It makes a significant difference: 15.7% vs 21.9% EER
- The actual task is ambiguous (ten sessions vs. two)
- The passwords are unspecified.

Subject recruitment and task

We recruited 10 subjects from the university community. Each one chose a 10-character password and typed it 200 times over the course of two weeks. We had 5 additional subjects act as impostors and type each of the other subjects' passwords 10 times each.

- Subject demographics are ambiguous (staff vs. students)
  - It makes a significant difference: 15.7% vs 21.9% EER
- The actual task is ambiguous (ten sessions vs. two)
- The passwords are unspecified.
- The impostors practice and incentives are unspecified.

Data-collection methods

A keystroke logger recorded each keystroke along with a timestamp (10ms accuracy). Each correctly-typed password was converted into a 19-element vector of 10 hold times and 9 keydown-keydown times.

Data-collection methods

A keystroke logger recorded each keystroke along with a timestamp (10ms accuracy). Each correctly-typed password was converted into a 19-element vector of 10 hold times and 9 keydown-keydown times.

- The accuracy claim is inexact
  - Is it a real-time guarantee, statement of resolution, or implementation (i.e., 10ms polling)?

Data-collection methods

A keystroke logger recorded each keystroke along with a timestamp (10ms accuracy). Each correctly-typed password was converted into a 19-element vector of 10 hold times and 9 keydown-keydown times.

- The accuracy claim is inexact
  - Is it a real-time guarantee, statement of resolution, or implementation (i.e., 10ms polling)?
- Shift and number-key handling is unspecified

Data-collection methods

A keystroke logger recorded each keystroke along with a timestamp (10ms accuracy). Each correctly-typed password was converted into a 19-element vector of 10 hold times and 9 keydown-keydown times.

- The accuracy claim is inexact
  - Is it a real-time guarantee, statement of resolution, or implementation (i.e., 10ms polling)?
- Shift and number-key handling is unspecified
- Causes problems deciding what is “correctly typed”

Data-analysis procedure

For each subject, an auto-associative neural network was trained using the first 100 passwords from the subject. The network was used to generate anomaly scores for the subject's remaining passwords, and for the impostor passwords. We generated ROC curves and recorded the network's average equal-error rate across all subjects.

Data-analysis procedure

For each subject, an auto-associative neural network was trained using the first 100 passwords from the subject. The network was used to generate anomaly scores for the subject's remaining passwords, and for the impostor passwords. We generated ROC curves and recorded the network's average equal-error rate across all subjects.

- The training procedure lacks sufficient detail:
  - How many hidden nodes were used?
  - What learning rate and momentum parameters?
  - How long did it train and for how many iterations?

- It makes a difference in performance:
  - 4 nodes, 1e-4, 3e-4 parameters, 500 epochs best for students
  - 19 nodes, 1e-4, 3e-5 parameters, 10,000 epochs best for staff
  - Both tunings are unusually low (i.e., outliers) over tuning range

Typical methods section

- The methods are ambiguous:
  - Subject recruitment and task
  - Data-collection methods
  - Data-analysis procedure
- The problems:
  - Uncertainty threatens reproducibility.
  - Hides (alternative) explanatory factors.
- Reproducibility, explanation, and prediction are hallmarks of science.
  - No one knows how, why, or if my detector works.
  - It looks promising but would you deploy it?

Outline

1. **Problem:**
   Ambiguous methods in keystroke dynamics

2. **Solution:**
   A shared repository of data, tools, and algorithms

3. **Tradeoffs:**
   Discussion about shared experimental frameworks

A shared experimental framework

- Planned repository for keystroke-dynamics research
- Goals:
  - easy and reliable experimental replication and reproduction
  - identification and testing of explanatory factors
  - rapid acceleration of scientific progress
- Shared contents:
  - Data (password-timing tables and raw XML data)
  - Collection tools (prompter, config-files, processing scripts)
  - Algorithms (language, detector code, evaluation procedure)

Shared data (password timing tables)

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t
1	s017	1	1	0.0679	0.2043	0.1364
2	s017	1	2	0.0781	0.2187	0.1406
3	s017	1	3	0.0729	0.2380	0.1651
4	s017	1	4	0.0686	0.2746	0.2060
5	s017	1	5	0.0665	0.3195	0.2530
6	s017	1	6	0.0742	0.4586	0.3844
7	s017	1	7	0.0694	0.3198	0.2504
8	s017	1	8	0.0652	0.3343	0.2691
9	s017	1	9	0.0681	0.2380	0.1699

- Each row encodes the time deltas of a correctly-typed password
- Tabular format is easily read by most data-analysis programs

Shared data (XML log files)

```
<event type="keystroke">
  <timestamp source="ticks"> 39.1090 </timestamp>
  <timestamp source="qpc"> 39.1135 </timestamp>
  <key> t </key>
  <key_event> make </key_event>
</event>

<event type="keystroke">
  <timestamp source="ticks"> 39.2030 </timestamp>
  <timestamp source="qpc"> 39.2033 </timestamp>
  <key> t </key>
  <key_event> break </key_event>
</event>
```

- Raw keydown (make) and keyup (break) times
- Readable by humans while still able to be parsed and verified
- Can be extended (e.g., multiple time sources, other data sources)

Shared data (XML log files)

```
<event type="custom">
  <timestamp source="qpc"> 38.0508 </timestamp>
  <name> ScreenDisplayed </name>
  <value> .tie5Roanl </value>
</event>

<event type="keystroke">
  <timestamp source="ticks"> 38.9530 </timestamp>
  <timestamp source="qpc"> 38.9538 </timestamp>
  <key> t </key>
  <key_event> make </key_event>
</event>

<event type="custom">
  <timestamp source="qpc"> 38.9581 </timestamp>
  <name> EachKeyError </name>
  <value> </value>
</event>
```

- Auxilliary events:
  - Prompts
  - Errors
  - Context
- Diagnose outliers
  - Long pauses
  - Error sequences
  - Correct sequences

Shared data (XML log files)

```
<instrumentation>
  <processor
     type="Genuine Intel(R) CPU L2400“
     clock_freq="1662MHz“
     number="2" />
  <memory total="1526 MB“
     free="1110 MB" />
</instrumentation>

<user id="s055">
   <session id="Strong Password - Jul 07“
     datetime_start="10/23/07 09:30:13“
     timestamp_unit="sec”
     timestamp_multiple="1">
```

- Meta-data provides details:
  - Machine specifications
  - Date and time information
  - Time-stamping precision
  - Resource availability (e.g., I/O load)
- Extensible to include new factors

Shared data

- Addresses ambiguity in subject recruitment and task
- Enables experimental replication:
  - The exact data are available for other researchers' use
- Facilitates further explanation and exploration:
  - Meta-data and auxiliary events may help diagnose outliers
  - Summary statistics may help explain per-subject variations
  - Additional demographic information can help as well
- Promotes more efficient research:
  - New algorithms can be tested and compared on the same data
  - Collecting data is expensive but amortization helps

Shared data-collection tool (prompter)

- Prompter displays stimuli (e.g., passwords)
- Verifies that they are typed correctly
- Produces XML log files

Shared data-collection tool (prompter)

On the next screen, you will see a paragraph of text from a civil service exam.

Please type in the text as-is.

Most people want to make sure their own welfare is being taken care of properly. This need will embrace such items as job security, fringe benefits, and final retirement. There are dozens of other items that could be grouped under personal welfare, but, people can cope with these other items if the basic three are covered.

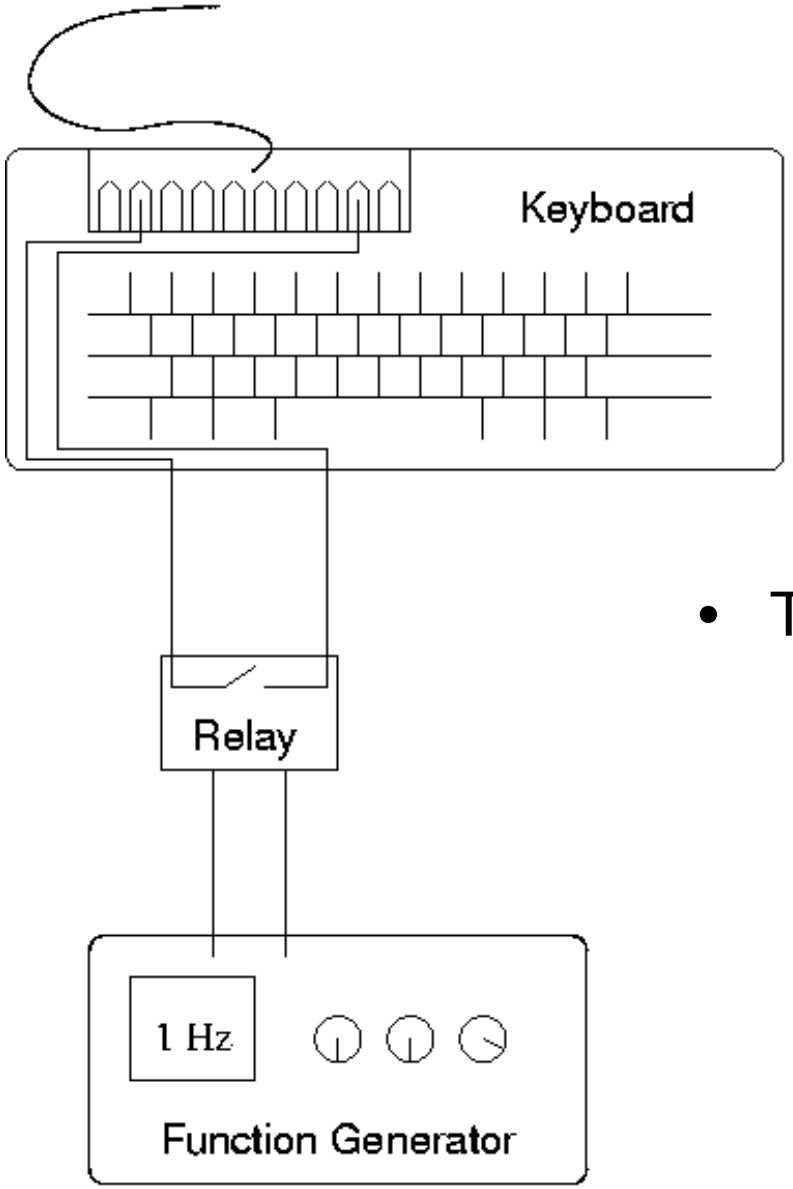
- Can be extended to present other typing tasks

Shared data-collection tool (config file)

```
# Next, type the strong password (.tie5Roanl) fifty times.
# Check for errors after each key, and do not cap the number
# of possible errors.

[Set ErrorCheck = EachKey]
[Set ErrorCap = 0]

.tie5Roanl
.tie5Roanl
.tie5Roanl
.tie5Roanl
.tie5Roanl
```

- Prompter is driven by a configuration file
- Stimulus and task criteria are encoded

Shared data-collection tool (calibration)



- Timestamp accuracy is calibrated
  - `qpc`: ± 8 ms
  - `ticks`: ±16 ms

Shared data-processing tools (perl scripts)

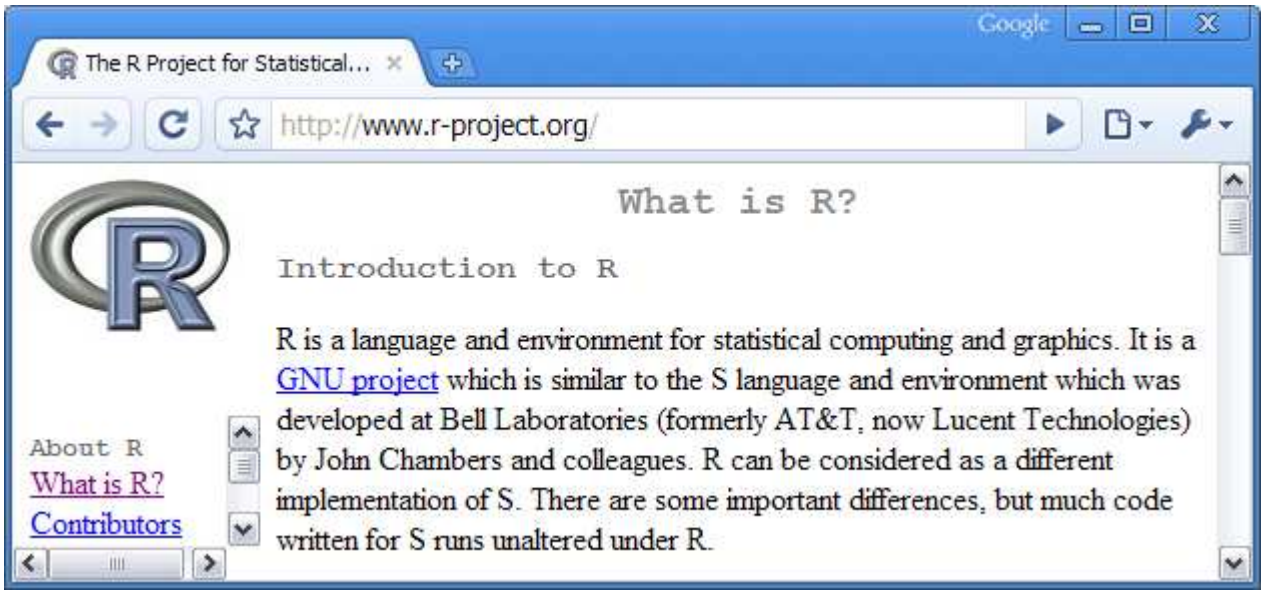
```
$ MakePasswordTable.pl -help

MakePasswordTable.pl [opts] <xmlfiles> <wordfile>
                     <outfile>

Read in one or more XML keystroke log files in
which the subject is asked to type one or more
passwords in repetition.  For each of the
passwords specified in the wordfile, output a
password-timing table.
```

- Converts XML log files to password-timing tables
- Specifies exactly which keys are equivalent:
  - Shift vs. Caps-Lock
  - Top-row digits vs. number pad digits
- Ensures clarity of extraction procedure

Shared data collection/processing tools

- Ensures exactness in collection and processing methods
- Enables experimental reproducibility:
  - The exact stimuli are described in the configuration files
  - The exact data-processing procedure are laid out in scripts
- Facilitates further explanation and exploration:
  - Extensive calibration eliminates confounding timing effects
- Promotes more efficient research:
  - Code to present stimuli and manage tasks can be written once
  - Corrections and improvements can be shared across the field

Shared data-analysis environment (R)



- R is free (open-source) with a large development community
- Language optimized for statistical analysis and data visualisation
- Rapid prototyping and testing of pattern-recognition algorithms

Shared anomaly-detector code (neural net)

```
library( AMORE );

aanet.train <- function( Ytrain ) {
   d <- ncol(Y);
   net <- newff( n.neurons = c( d, d, d ),
                 learning.rate.global = 0.1 / 1000,
                 momentum.global = 0.3 / 1000 );
   net <- ADAPTgd.MLPnet( net, Ytrain, Ytrain, 500 );
   return( net );
}

aanet.score <- function( net, Yscore ) {
   Yout <- sim.MLPnet( net, Yscore );
   scores <- colSums( (Yscore - Yout)^2 );
   return( scores );
}
```

- Concise expression of common statistical procedures
- Utilize existing libraries for complex procedures
- Exact invocation parameters are specified (or defaults)

Shared detector-evaluation procedure

```
trainfunc <- aanet.train;
scorefunc <- aanet.score;
for( subj in subjects ) {
   Ytrain <- subset( Y, subject == subj & sessionIndex <= 2 );
   Yscore1 <- subset( Y, subject == subj & sessionIndex > 2 );
   Yscore2 <- subset( Y, subject != subj & sessionIndex == 1 &
                      rep <= 5 );
   trainobj <- trainfunc( Ytrain );
   scores1 <- scorefunc( trainobj, Yscore1 );
   scores2 <- scorefunc( trainobj, Yscore2 );
   eers[[ thesubject ]] <- geteer( scores1, scores2 );
}
avgeer <- mean( eers );
```

- Concise routine to manipulate data and invoke detector
- Enforces a standard interface to the detector
- Provides a framework for “dropping in” new detectors

Shared data-analysis algorithms

- Adds complete specificity to the data-analysis procedure
- Enables experimental reproducibility:
  - The exact data-analysis procedure is codified and executable
  - Permits sensitivity testing for detector-parameter choices
- Facilitates further explanation and exploration:
  - Evaluation-design choices can be parameterized and measured
  - New detectors can be “dropped in” and tested immediately
- Promotes more efficient research:
  - R is amenable to rapid prototyping of new statistical algorithms
  - Ensures sound comparisons between detection algorithms

A shared experimental framework

- Planned repository for keystroke-dynamics research:

- Goals:
  - easy and reliable experimental replication and reproduction
  - identification and testing of explanatory factors
  - rapid acceleration of scientific progress

- Shared contents:
  - Data (password-timing tables and raw XML data)
  - Collection tools (prompter, config-files, processing scripts)
  - Algorithms (language, detector code, evaluation procedure)

Outline

1. **Problem:**
   Ambiguous methods in keystroke dynamics

2. **Solution:**
   A shared repository of data, tools, and algorithms

3. **Tradeoffs:**
   Discussion about shared experimental frameworks

Tradeoffs and questions

- Why should we use a shared framework?
- Do we need to use a shared framework?
- Do we need a methods section too?
- Is it going to take a lot of work?
- Why can’t we just ask the authors for more info?
- Why should others get to use my hard work?
- How do we make a shared framework?
- Doesn’t privacy get in the way of shared data?
- Should we reward sharing?

Why should we use a shared framework?

- Makes independent confirmation easier
- Increases chance that others will use and appreciate your work
- Greater efficiency – less duplication of effort
- Promotes better science:
  - Using observations to generate new hypotheses
  - Designing reproducible experiments to test them
  - Analyzing and publishing new observations

Do we need to use a shared framework?

- An experimental unit:

$$\exp(x_1, x_2, x_3, x_4, \ldots, x_n) \rightarrow \text{out}$$

  - Each parameter is an influential factor
    - subject demographics
    - stimuli (i.e., password)
    - time-stamping method
    - detector
  - The value is the outcome (e.g., equal-error rate)

- Isolating a factor's effect requires control

- A shared framework may be necessary if there are
  - many (possibly unknown) factors
  - subtle (possibly counterintuitive) effects

Do we need a good methods section too?

- Yes, they serve different purposes:
  - *Del rigor en la ciencia*, by Borges:

    And so, the College of Cartographers evolved a Map of the Empire that was of the same Scale as the Empire and that coincided with it point for point. (trans Di Giovanni)
  - A roadmap provides something other than the road

    Consider a “proof sketch” vs a “low-level proof” in hierarchical proofs (Lamport)

- A methods section provides:
  - Intuition, explanation, and rationale
  - Extensive treatment of subtle but important details

Is it going to take a lot of work?

- Sharing data and algorithms for one paper took two weeks
  - It should get easier with more experience
- Sharing data-collection tools will require more effort
  - Supporting a software project
  - Open-source development is a possibility (not without problems)
- Available-upon-request easier in the short term
  - Probably more of a hassle in the long term
- Yes, but the effort will be appreciated (and may be necessary)

Why can’t we just ask authors for more info?

- Experience requesting data has been erratic

- Experience reproducing an experiment was negative

Why should others get to use my hard work?

- Related questions:
  - “Why go through the trouble when there’s no incentive?”
  - “My algorithm / tool / data set is my intellectual property?”

- Possible answers:
  - “Good point” (or “I don’t know”)
  - “A shared framework is ‘community service’”
  - “Moreover, it’s *necessary* for scientific progress”

So how do we make a shared framework?

- Organically at first – one publication at a time.
  - Share the data and algorithms needed to replicate the results
- Organization should develop as it is needed:
  - Network and architecture simulators
  - Bioinformatics (bioconductor) repository
- Other options?

Doesn’t privacy get in the way?

- There are certainly tradeoffs:
  - Hypothesis:
    - Real names are good strings for keystroke-dynamics
  - Easy experiment to design and run
  - Difficult data to share

Should we reward data sharing?

- Some data must be necessarily restricted
- If people have no incentive to share, will they?
- Should we question unnecessary restrictions?
- Should we (i.e., reviewers) reward data sharing?

Key ideas (review)

- **Methods are instructions**
  - for reproducing an experiment
  - that explain what was done and why
  - that *must* be understood by other researchers

- **A shared experimental framework provides context**
  - comprised of data, tools, and algorithms
  - publicly available for researchers' use
  - provide context necessary for understanding
  - making methods clearer and more accurate
  - promoting scientific principles and accelerating progress

Final thoughts

*An article about computational science in a scientific publication is **not** the scholarship itself, it is merely **advertising** of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

(Clairbout's "slogan", via Buckheit and Donoho)

Questions for discussion:

- Why should we use a shared framework?
- Do we need to use a shared framework?
- Do we need a methods section too?
- Is it going to take a lot of work?
- Why can't we just ask the authors for more info?
- Why should others get to use my hard work?
- How do we make a shared framework?
- Doesn't privacy get in the way of shared data?
- Should we reward sharing?