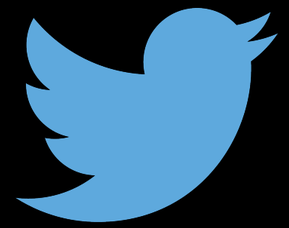
Data Workflows for Machine Learning:

Seattle, WA
2014-01-29



Paco Nathan
@pacoid
http://liber118.com/pxn/

meetup.com/Seattle-DAML/events/159043422/

Why is this talk here?

Machine Learning in production apps is less and less about algorithms (even though that work is quite fun and vital)

Performing real work is more about:

- socializing a problem within an organization
- feature engineering ("Beyond Product Managers")
- tournaments in CI/CD environments
- operationalizing high-ROI apps at scale
- etc.

So I'll just crawl out on a limb and state that leveraging great frameworks to build data workflows is more important than chasing after diminishing returns on highly nuanced algorithms.

Because Interwebs!

Data Workflows for Machine Learning

Middleware has been evolving for Big Data, and there are some great examples — we'll review several. Process has been evolving too, right along with the use cases.

Popular frameworks typically provide some Machine Learning capabilities within their core components, or at least among their major use cases.

Let's consider features from *Enterprise Data Workflows* as a basis for what's needed in Data Workflows for Machine Learning.

Their requirements for scale, robustness, cost trade-offs, interdisciplinary teams, etc., serve as guides in general.

Caveat Auditor

I won't claim to be expert with each of the frameworks and environments described in this talk. Expert with a few of them perhaps, but more to the point: embroiled in many use cases.

This talk attempts to define a "scorecard" for evaluating important ML data workflow features:
what's needed for use cases, compare and contrast of what's available, plus some indication of which frameworks are likely to be best for a given scenario.

Seriously, this is a work in progress.

Outline

- Definition: Machine Learning
- Definition: Data Workflows
- A whole bunch o' examples across several platforms
- Nine points to discuss, leading up to a scorecard
- A crazy little thing called PMML
- Questions, comments, flying tomatoes…

Data Workflows for Machine Learning:

Frame the question…

“A Basis for What’s Needed”

Definition: Machine Learning

"Machine learning algorithms can figure out how to perform important tasks by **generalizing** *from examples. This is often feasible and cost-effective where manual programming is not. As more data becomes available, more ambitious problems can be tackled. As a result, machine learning is widely used in computer science and other fields."*

Pedro Domingos, U Washington
A Few Useful Things to Know about Machine Learning

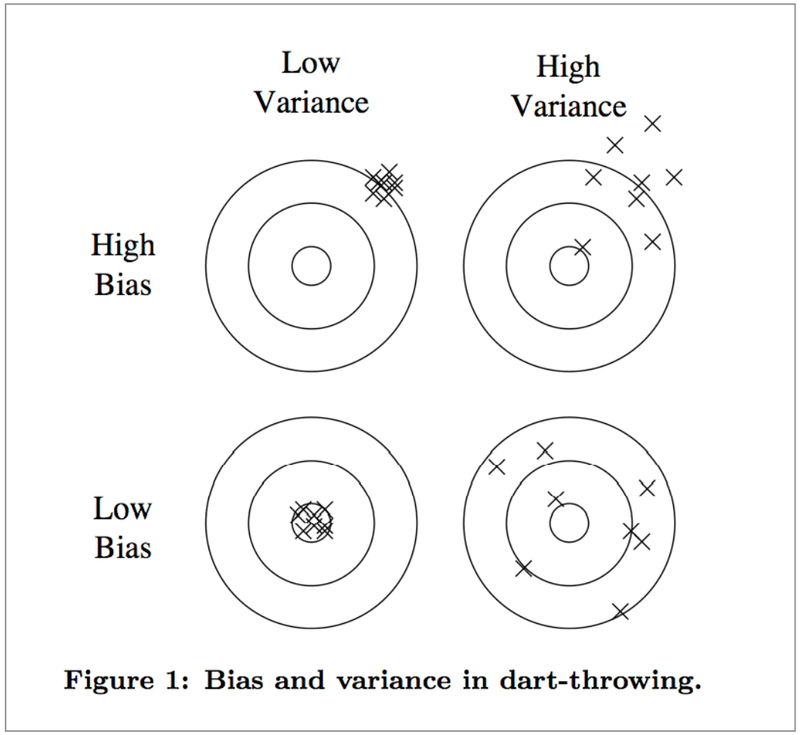


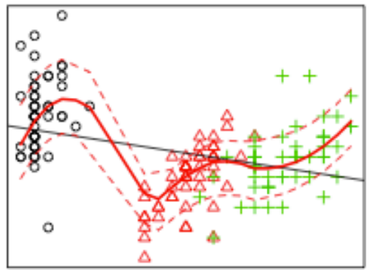
Figure 1: Bias and variance in dart-throwing.

[learning as generalization] =
[representation] + [evaluation] + [optimization]

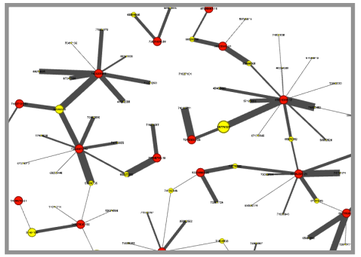
- overfitting (variance)
- underfitting (bias)
- "perfect classifier" (no free lunch)

Definition: Machine Learning … Conceptually

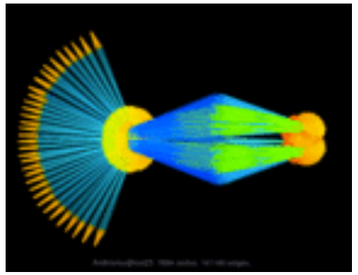
1. real-world data ⇒



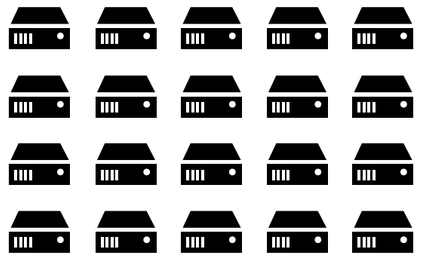
2. graph theory for representation ⇒



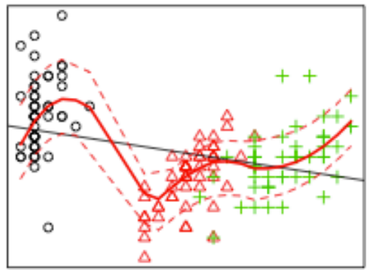
3. convert to sparse matrix for production work ⇒
leveraging abstract algebra + func programming

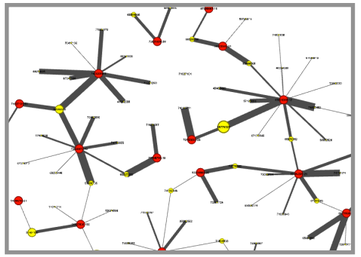


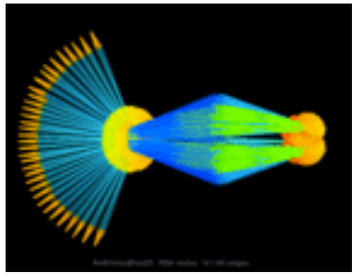
4. cost-effective parallel processing for ML apps
at scale, live use cases



Definition: Machine Learning … Conceptually







$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

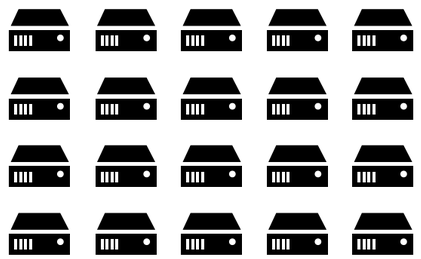
f(x): loss function
g(z): regularization term

1. real-world data ⇒
[generalization]

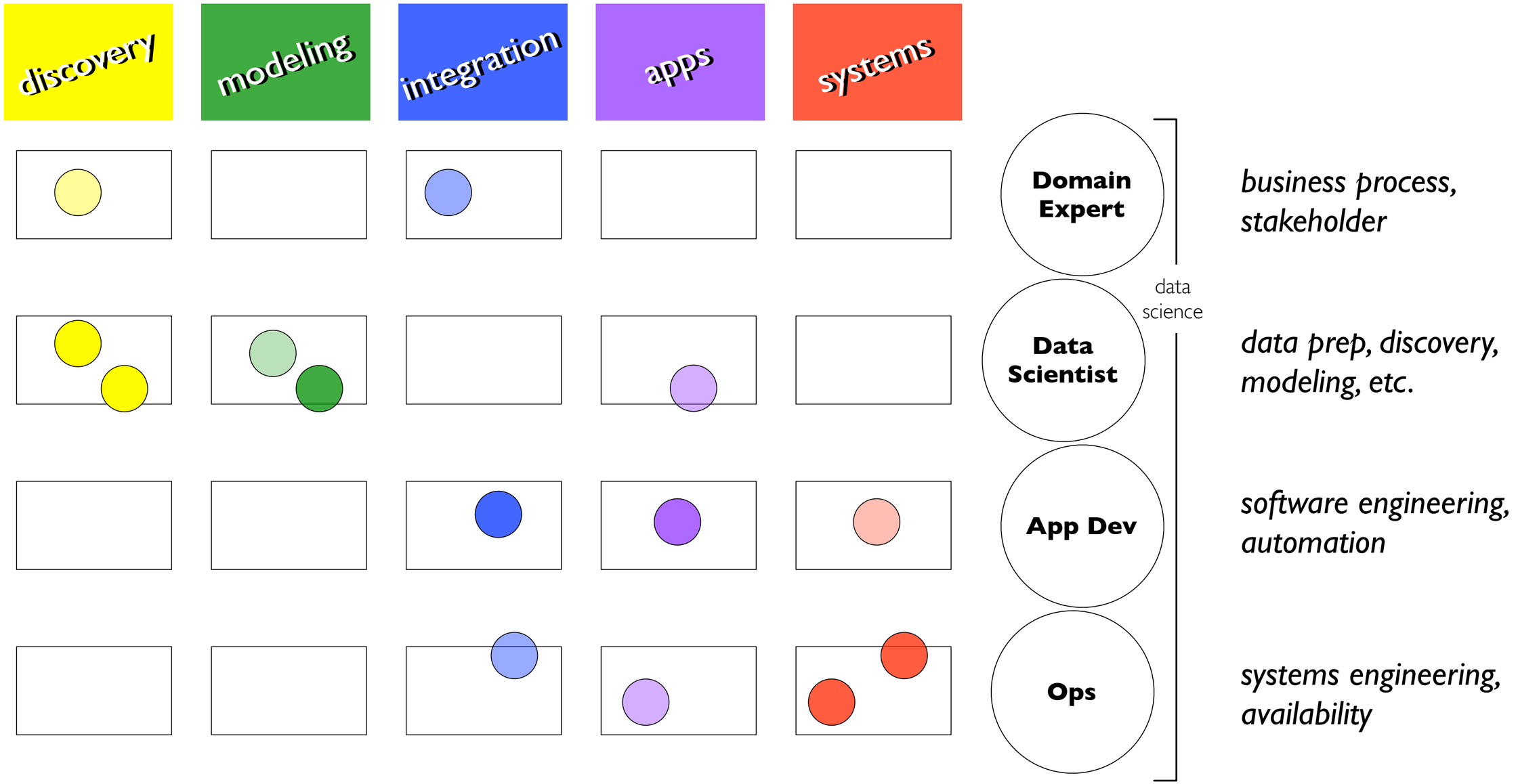
2. graph theory for representation ⇒
[representation]

3. convert to sparse matrix for production work ⇒
leveraging abstract algebra + func programming
[evaluation]

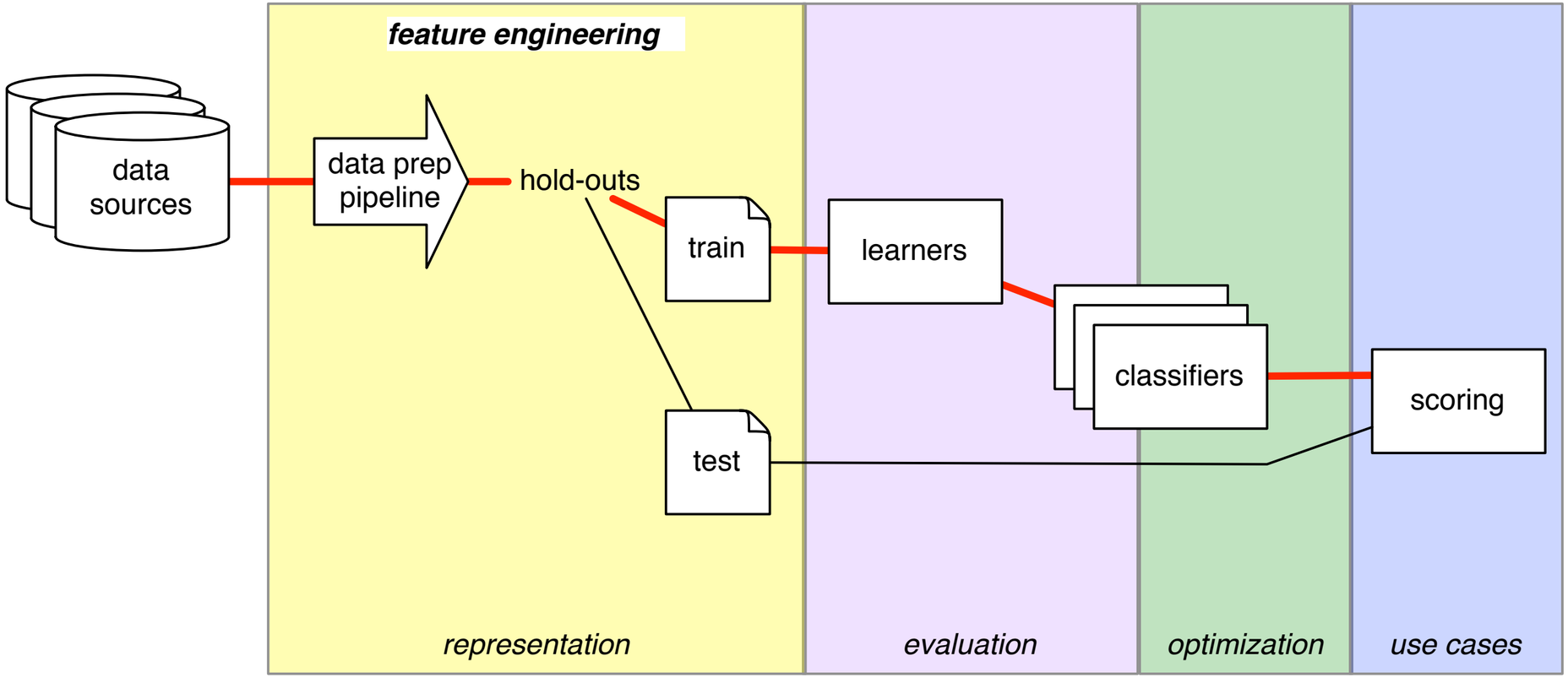
4. cost-effective parallel processing for ML apps
at scale, live use cases
[optimization]



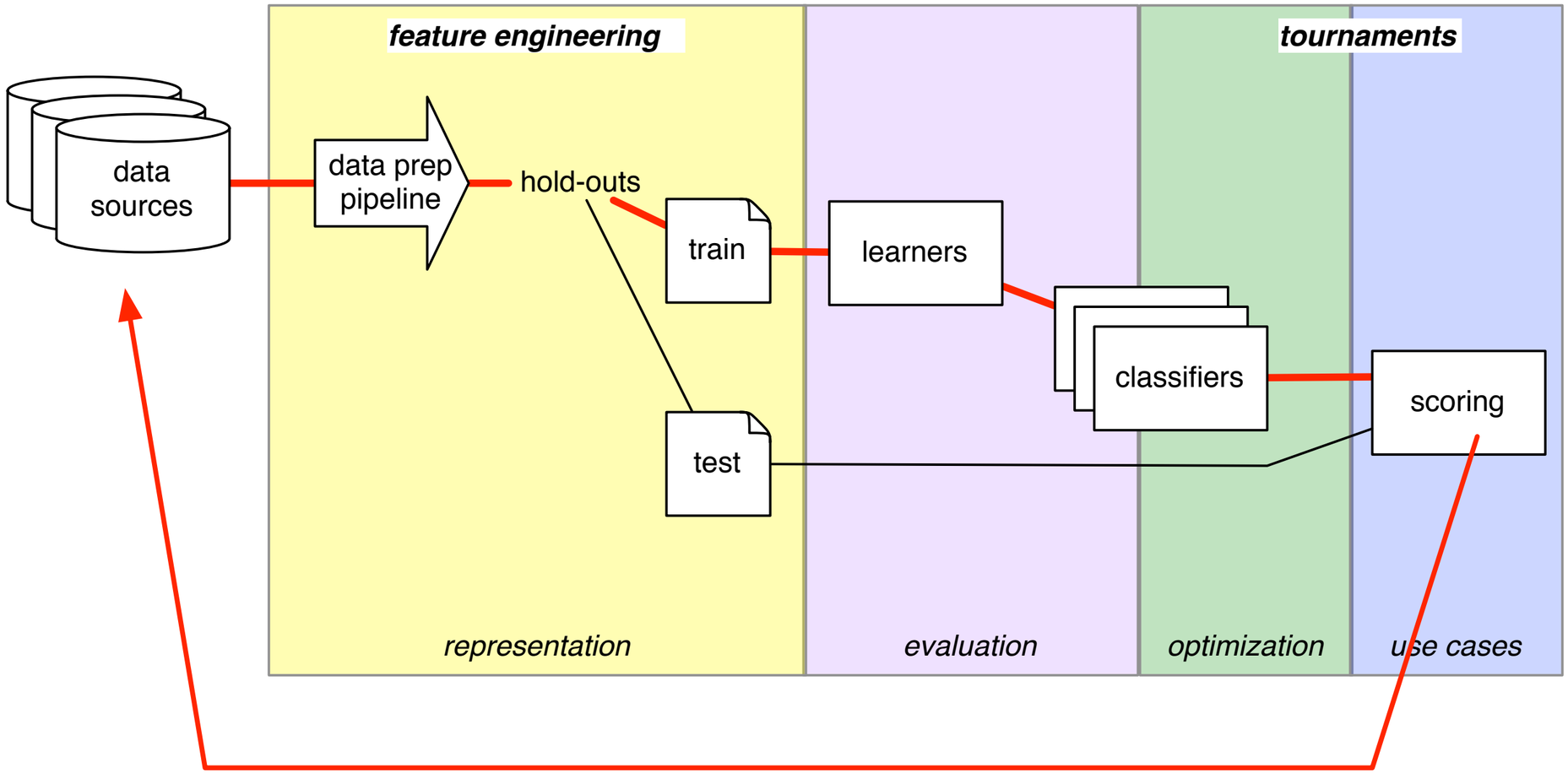
Definition: Machine Learning … Interdisciplinary Teams, Needs × Roles



Definition: Machine Learning … Process, Feature Engineering



Definition: Machine Learning ... Process, Tournaments



- obtain other data?
- improve metadata?
- refine representation?
- improve optimization?

- iterate with stakeholders?
- can models (inference) inform first principles approaches?

quantify and measure:
- benefit?
- risk?
- operational costs?

Definition: Machine Learning … Invasion of the Monoids

monoid (an alien life form, courtesy of abstract algebra):

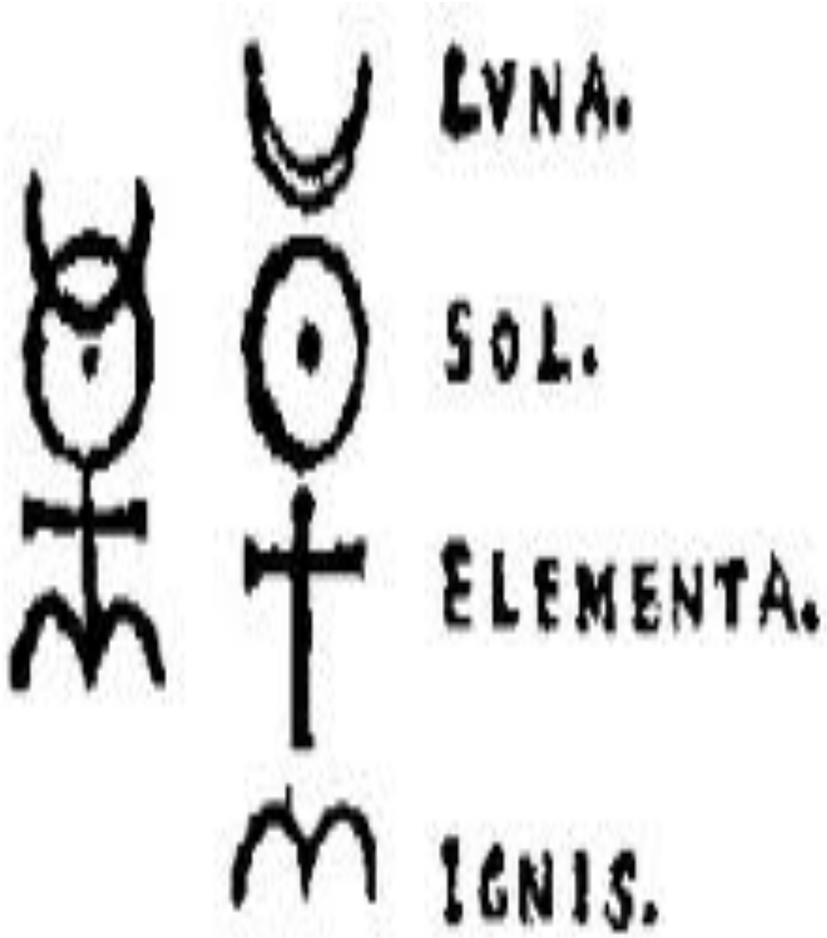
- binary associative operator
- closure within a set of objects
- unique identity element

what are those good for?

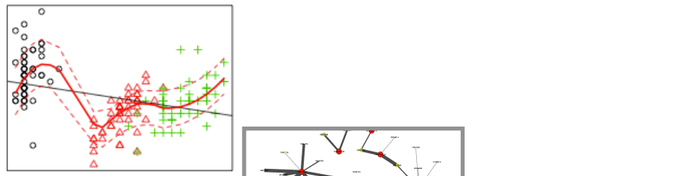
- composable functions in workflows
- compiler "hints" on steroids, to parallelize
- reassemble results minimizing bottlenecks at scale
- reusable components, like Lego blocks
- think: Docker for business logic

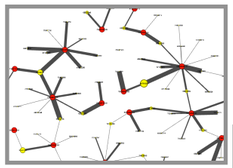
Monoidify! Monoids as a Design Principle for Efficient MapReduce Algorithms
Jimmy Lin, U Maryland/Twitter

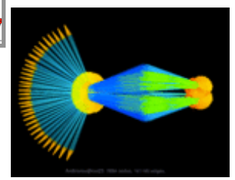
kudos to Oscar Boykin, Sam Ritchie, et al.

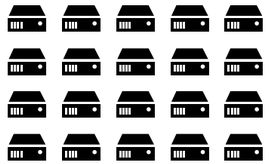


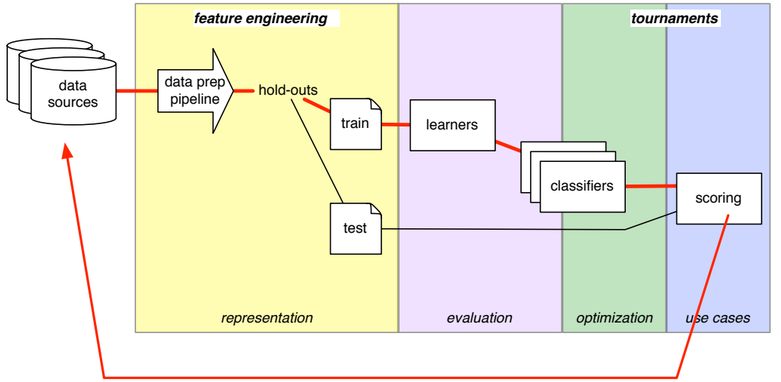
Definition: Machine Learning











- obtain other data?
- improve metadata?
- refine representation?
- improve optimization?

- iterate with stakeholders?
- can models (inference) inform first principles approaches?

quantify and measure:
- benefit?
- risk?
- operational costs?

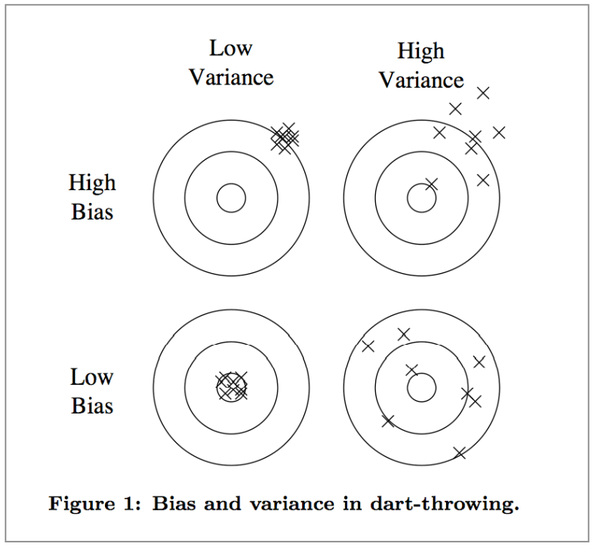
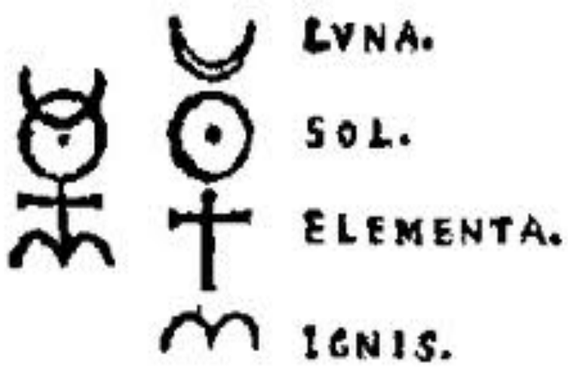
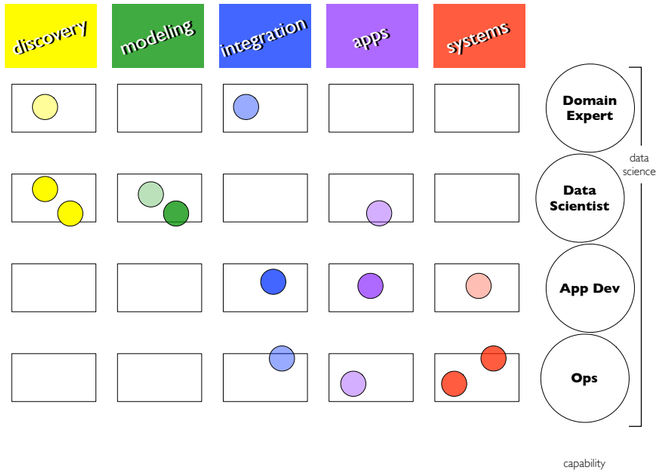
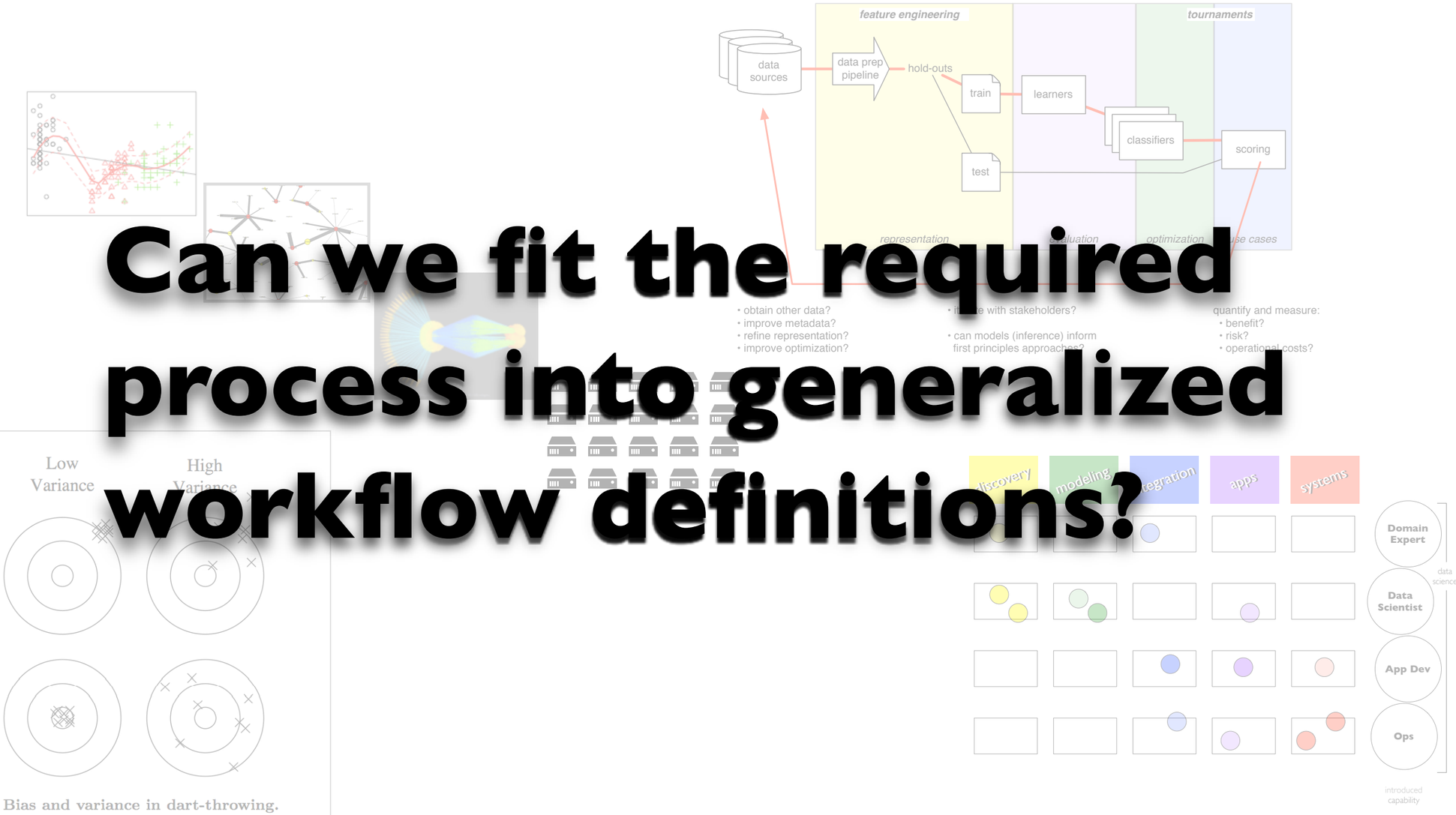


Figure 1: Bias and variance in dart-throwing.





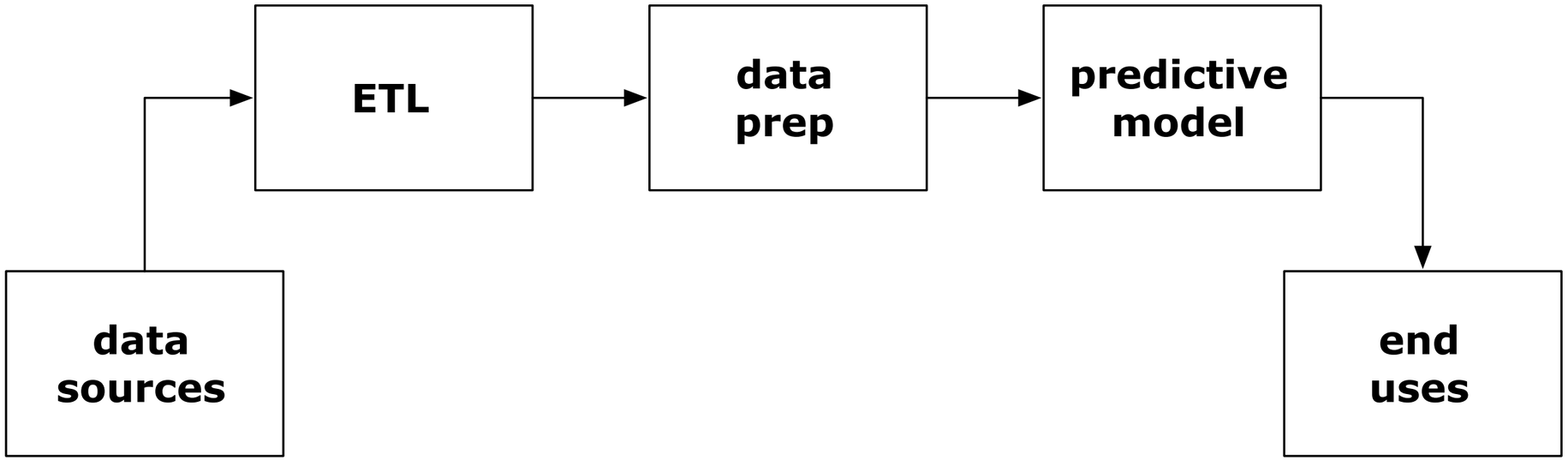
Definition



Can we fit the required process into generalized workflow definitions?

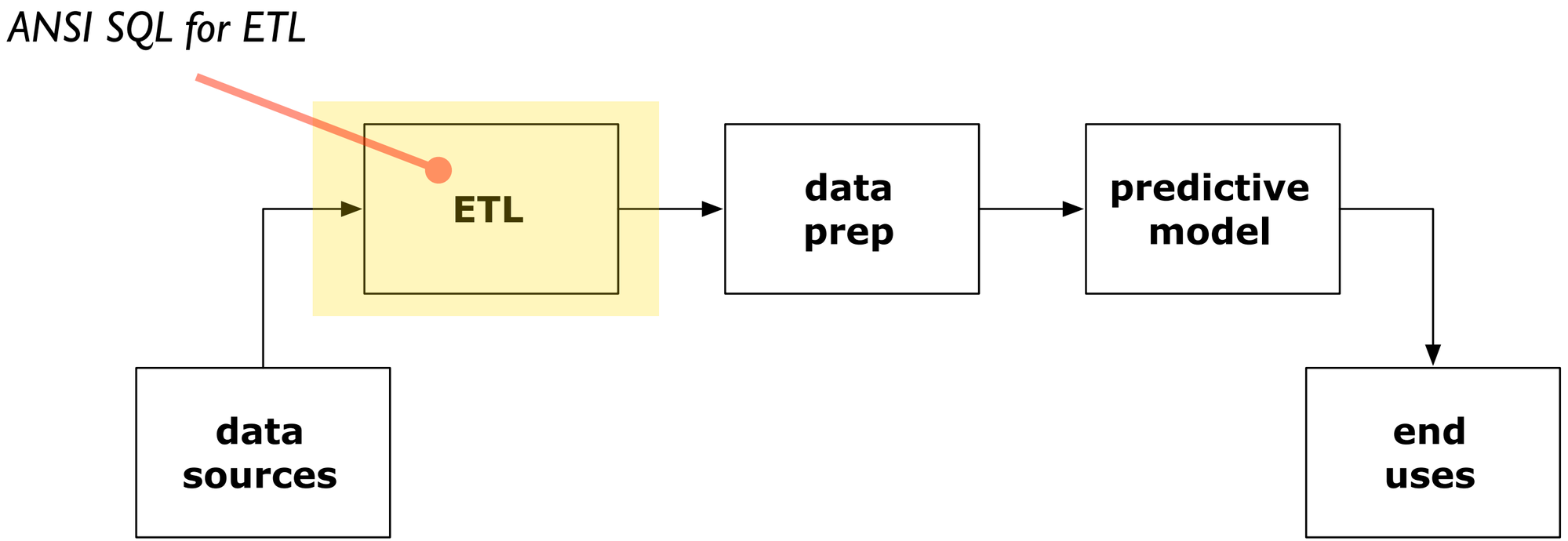
Definition: Data Workflows

Middleware, effectively, is evolving for Big Data and Machine Learning…
The following design pattern — a DAG — shows up in many places,
via many frameworks:



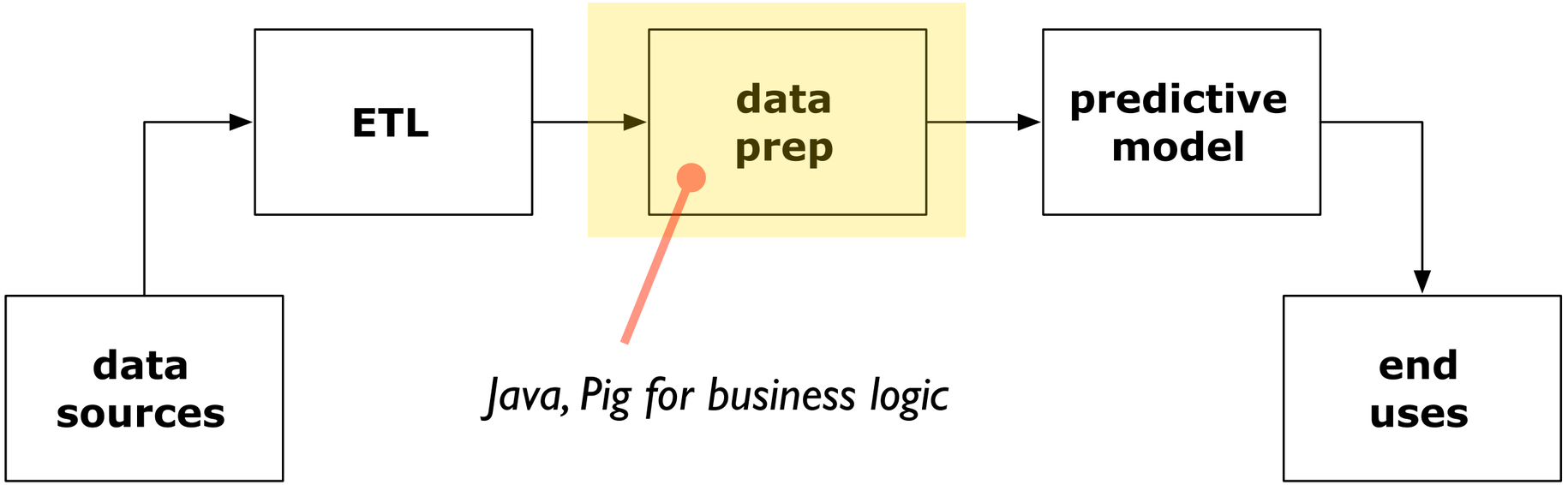
Definition: Data Workflows

definition of a typical Enterprise workflow which crosses through multiple departments, languages, and technologies…



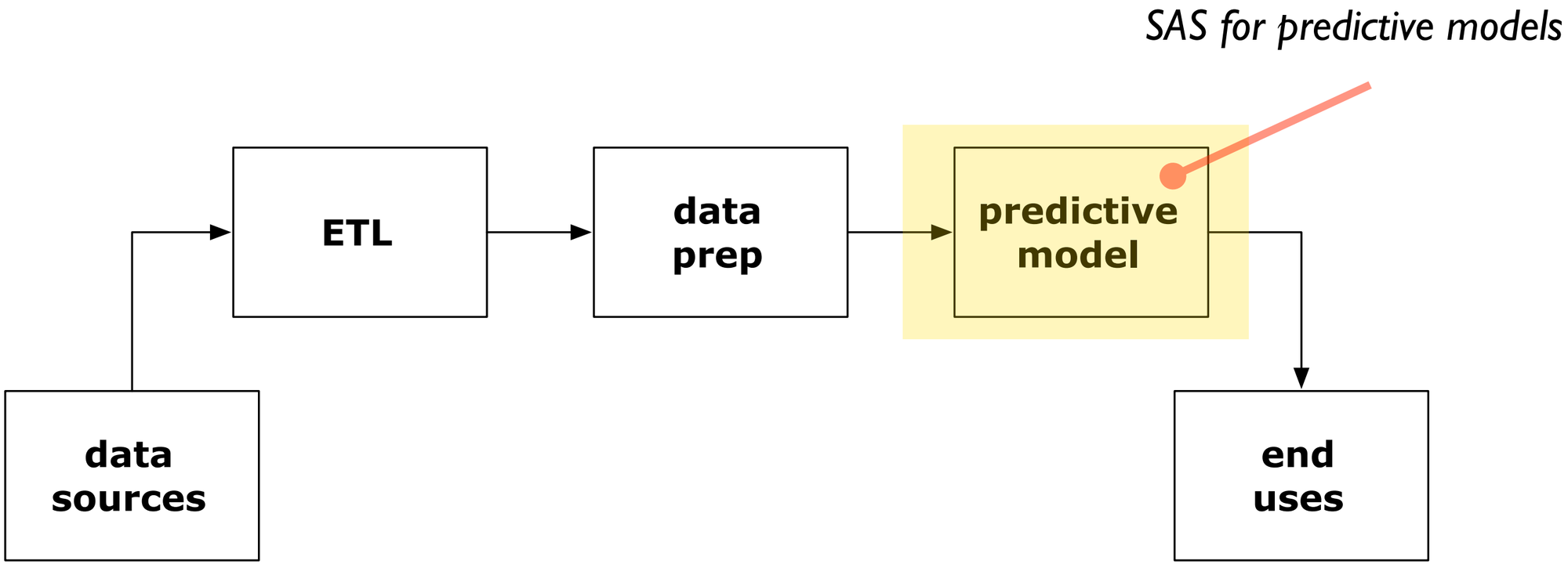
Definition: Data Workflows

definition of a typical Enterprise workflow which crosses through multiple departments, languages, and technologies…



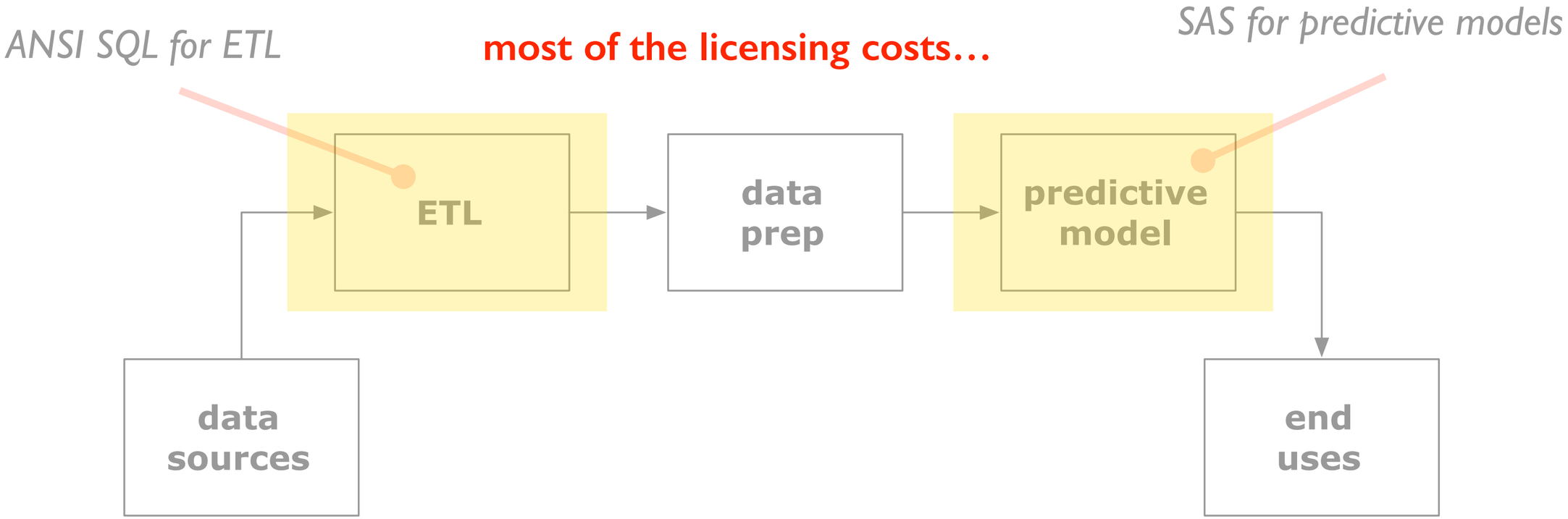
Definition: Data Workflows

definition of a typical Enterprise workflow which crosses through multiple departments, languages, and technologies…



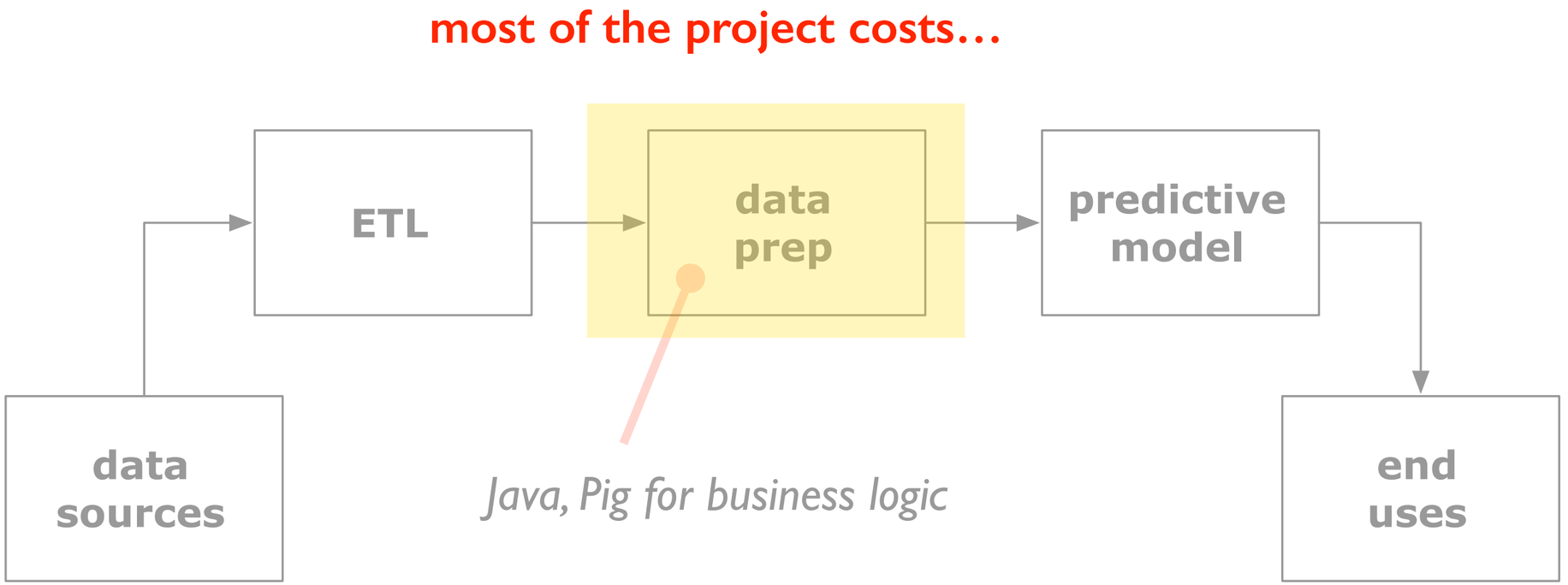
Definition: Data Workflows

definition of a typical Enterprise workflow which crosses through multiple departments, languages, and technologies…



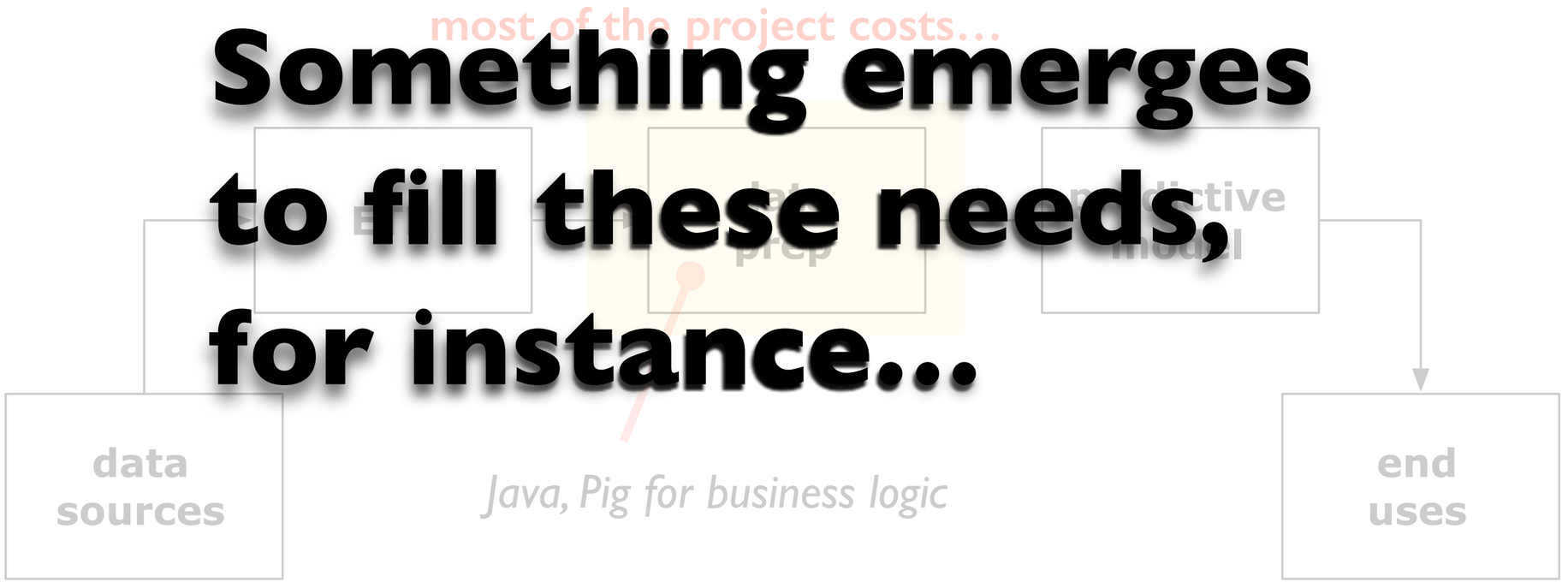
Definition: Data Workflows

definition of a typical Enterprise workflow which crosses through multiple departments, languages, and technologies…



Definition: Data Workflows

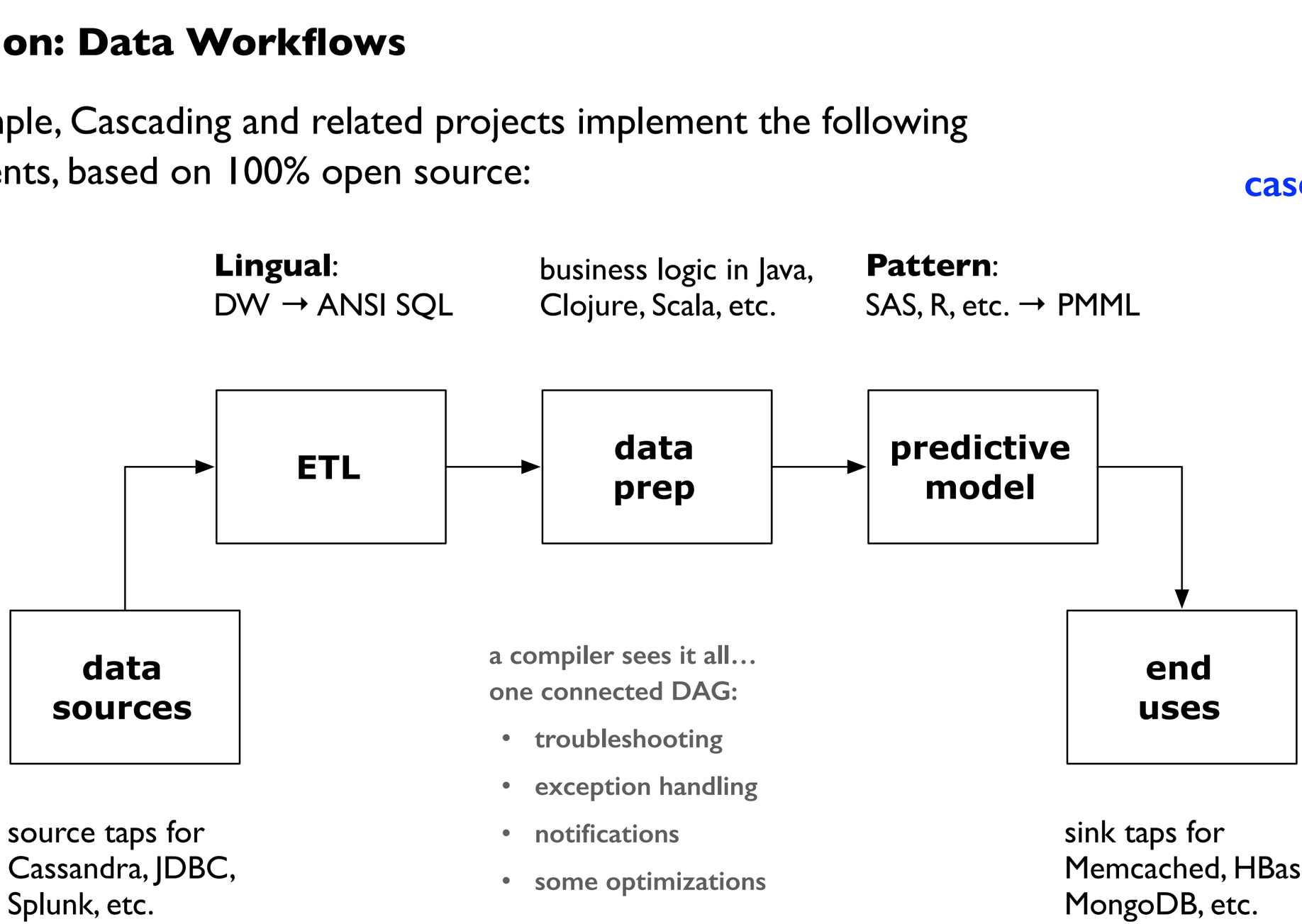
definition of a typical Enterprise workflow which crosses through multiple departments, languages, and technologies…



Something emerges to fill these needs, for instance…

Definition: Data Workflows

For example, Cascading and related projects implement the following components, based on 100% open source:

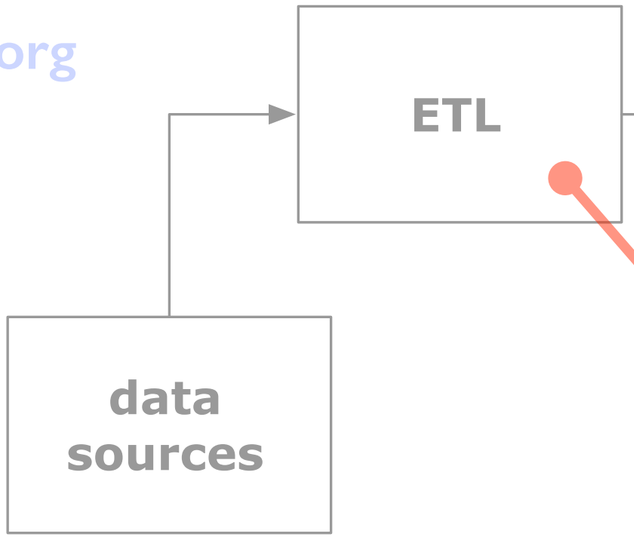


Definition: Data Workflows

Cascading allows multiple departments to combine their workflow components into an integrated app – one among many, typically – based on 100% open source

cascading.org

Lingual:
DW → ANSI SQL



source taps for Cassandra, JDBC, Splunk, etc.

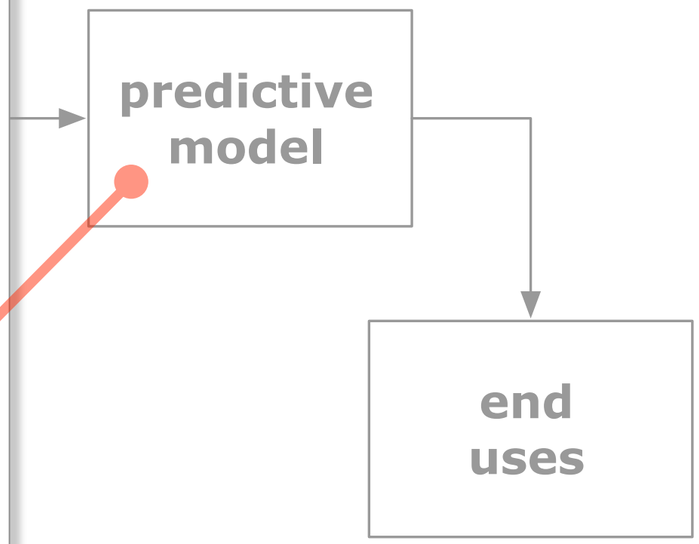
```
FlowDef flowDef = FlowDef.flowDef()
  .setName( "etl" )
  .addSource( "example.employee", emplTap )
  .addSource( "example.sales", salesTap )
  .addSink( "results", resultsTap );

SQLPlanner sqlPlanner = new SQLPlanner()
  .setSql( sqlStatement );

flowDef.addAssemblyPlanner( sqlPlanner );
```

Definition: Data Workflows

Cascading allows multiple departments to combine their workflow components into an integrated app – one among many, typically – based on 100% open source

```
FlowDef flowDef = FlowDef.flowDef()
  .setName( "classifier" )
  .addSource( "input", inputTap )
  .addSink( "classify", classifyTap );

PMMLPlanner pmmlPlanner = new PMMLPlanner()
  .setPMMLInput( new File( pmmlModel ) )
  .retainOnlyActiveIncomingFields();

flowDef.addAssemblyPlanner( pmmlPlanner );
```

Pattern:
SAS, R, etc. → PMML

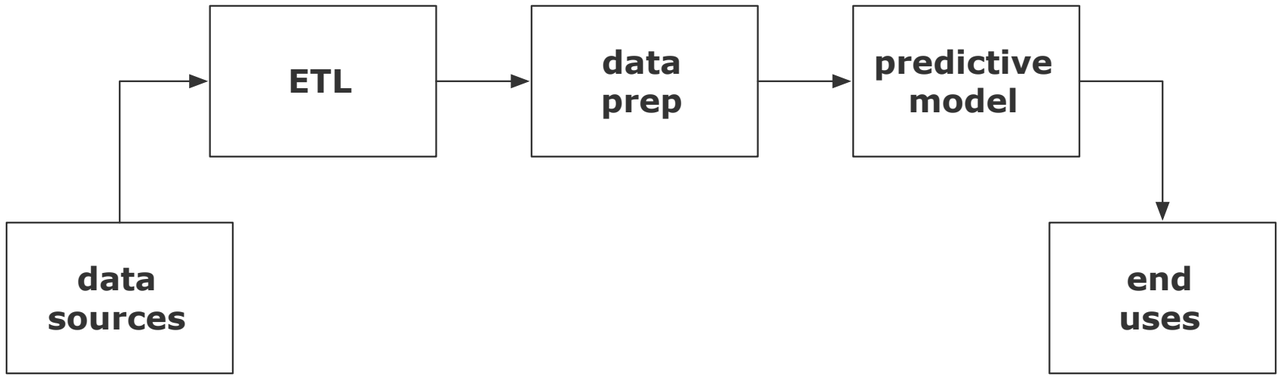


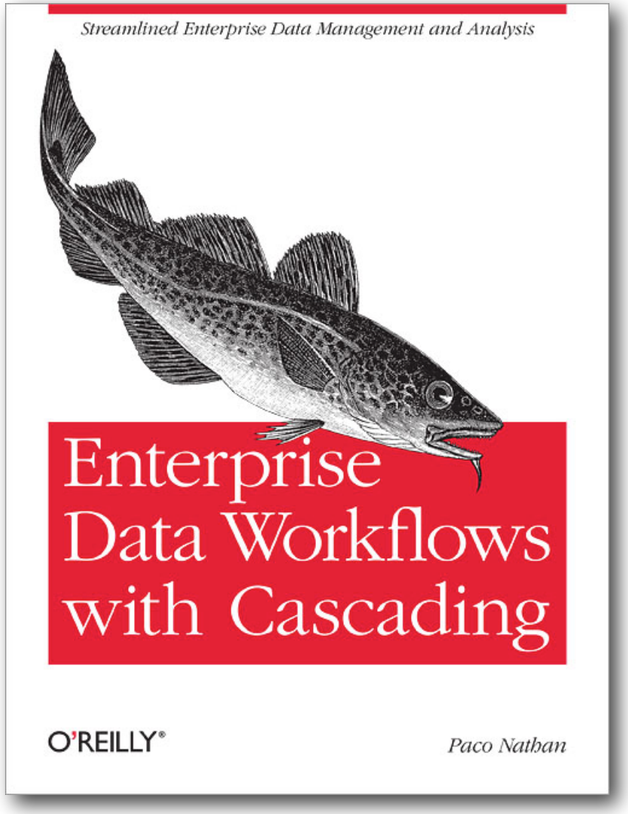
sink taps for
Memcached, HBase,
MongoDB, etc.

Enterprise Data Workflows with Cascading

O'Reilly, 2013

shop.oreilly.com/product/0636920028536.do

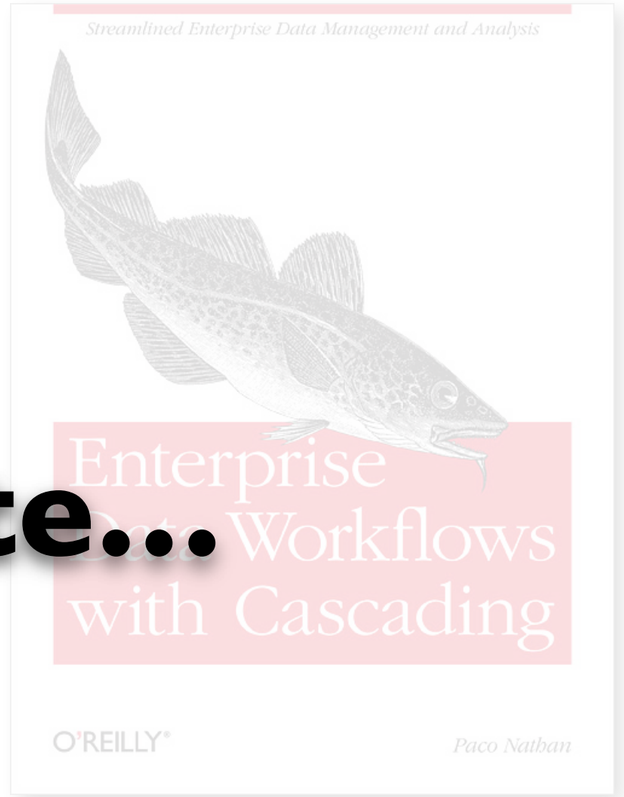




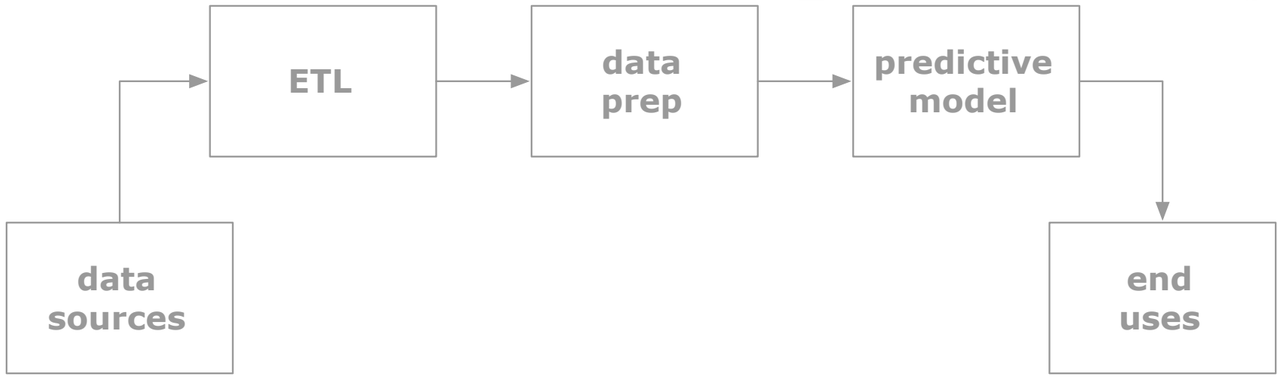
Enterprise Data Workflows with Cascading

O'Reilly, 2013

shop.oreilly.com/product/0636920028536.do



This begs an update...



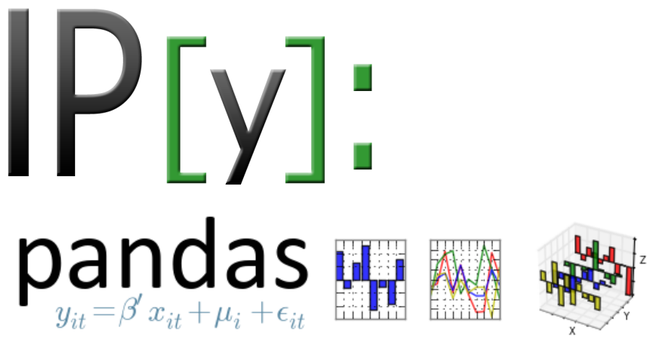
Enterprise Data Workflows with Cascading

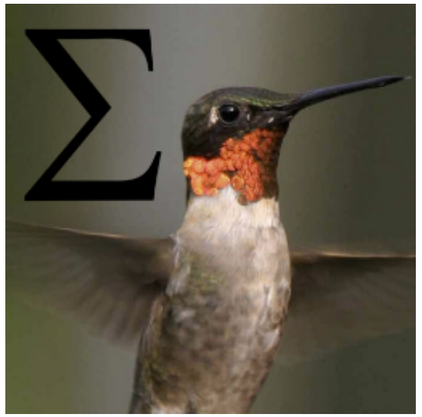
O'Reilly, 2013

shop.oreilly.com/product/0636920028536.do

Because...


KNIME


IP[y]:
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$


Σ


actian


open data
Augustus

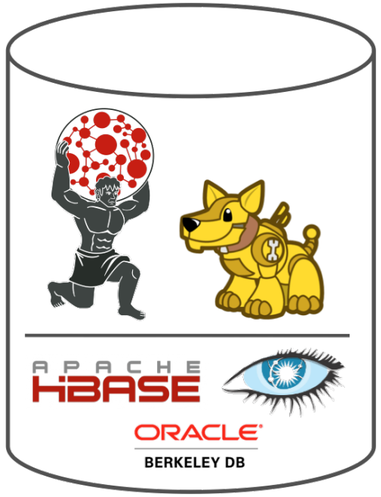

Scalding


RStudio


{m}


Cascalog


Spark


APACHE
HBASE
ORACLE
BERKELEY DB


julia


cascading


ML base


ADAPA®


mahout

Data Workflows for Machine Learning:

Examples…

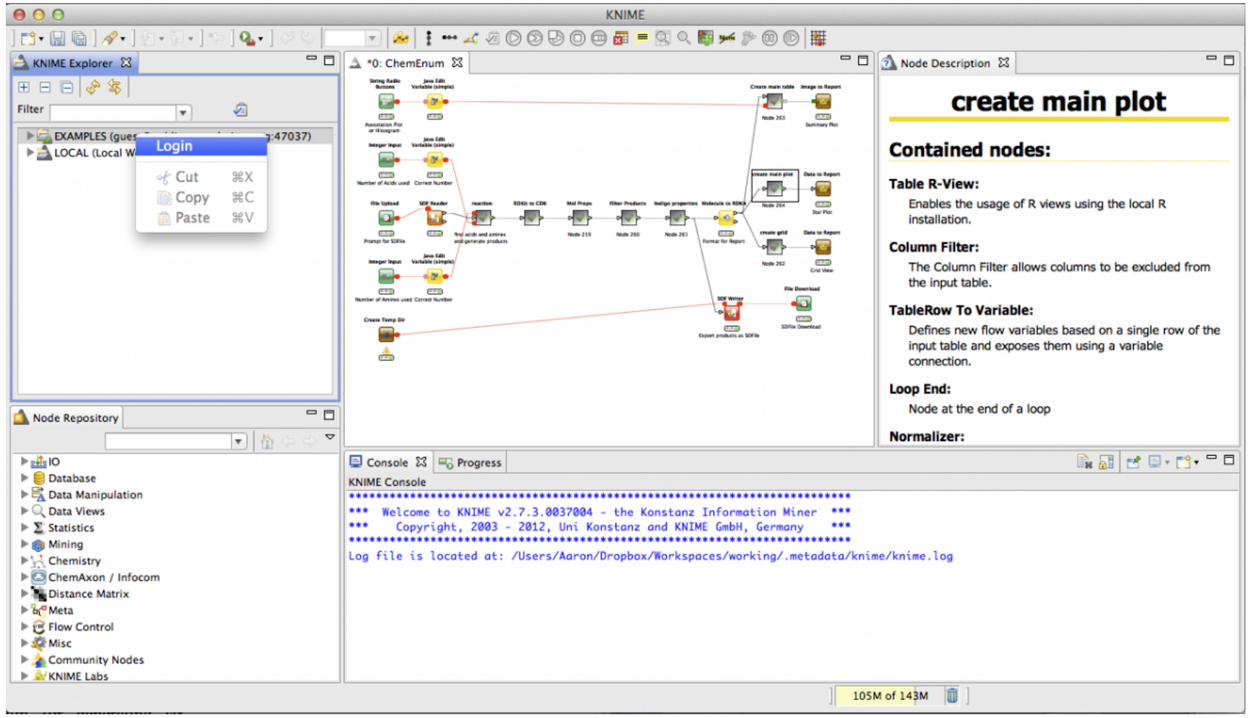
“Neque per exemplum”

Example: KNIME

knime.org

"a user-friendly graphical workbench for the entire analysis process: data access, data transformation, initial investigation, powerful predictive analytics, visualisation and reporting."

- large number of integrations (over 1000 modules)
- ranked #1 in customer satisfaction among open source analytics frameworks
- visual editing of reusable modules
- leverage prior work in R, Perl, etc.
- Eclipse integration
- easily extended for new integrations



Example: Python stack

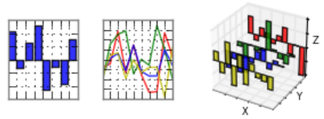
Python has much to offer – ranging across an organization, no just for the analytics staff

- ipython.org
- pandas.pydata.org
- scikit-learn.org
- numpy.org
- scipy.org
- code.google.com/p/augustus
- continuum.io
- nltk.org
- matplotlib.org

IP[y]:

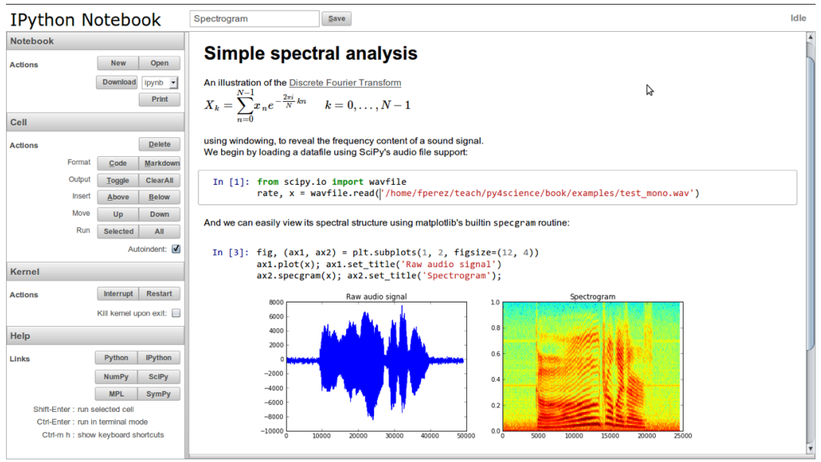
pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Augustus

open data

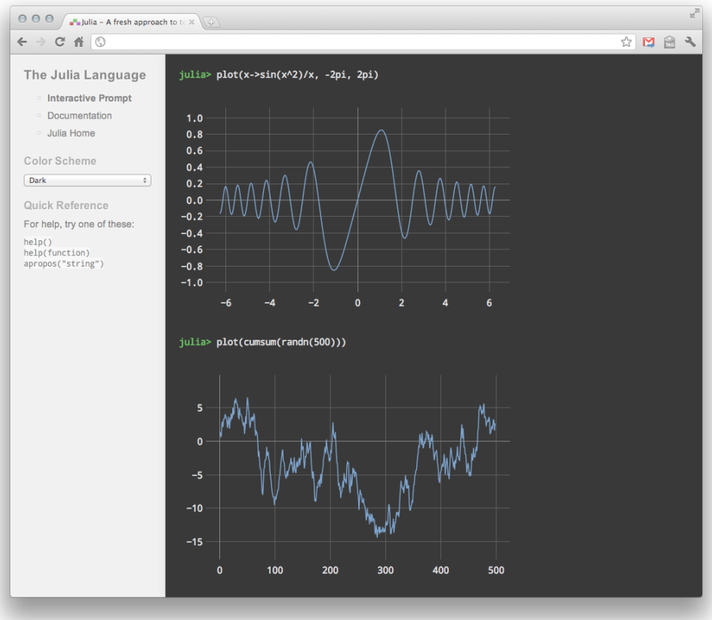


Example: Julia

julialang.org

"a high-level, high-performance dynamic programming language for technical computing, with syntax that is familiar to users of other technical computing environments"

- significantly faster than most alternatives
- built to leverage parallelism, cloud computing
- still relatively new — one to watch!




```
importall Base

type BubbleSort <: Sort.Algorithm end

function sort!(v::AbstractVector, lo::Int, hi::Int, ::BubbleSort, o::Sort.Ordering)
    while true
        clean = true
        for i = lo:hi-1
            if Sort.lt(o, v[i+1], v[i])
                v[i+1], v[i] = v[i], v[i+1]
                clean = false
            end
        end
        clean && break
    end
    return v
end
```

Example: Summingbird

github.com/twitter/summingbird

"a library that lets you write streaming MapReduce programs that look like native Scala or Java collection transformations and execute them on a number of well-known distributed MapReduce platforms like Storm and Scalding."

- switch between Storm, Scalding (Hadoop)
- Spark support is in progress
- leverage Algebird, Storehaus, Matrix API, etc.


```
def wordCount[P <: Platform[P]]
  (source: Producer[P, String], store: P#Store[String, Long]) =
    source.flatMap { sentence =>
      toWords(sentence).map(_ -> 1L)
    }.sumByKey(store)
```

Example: Scalding

github.com/twitter/scalding


```
import com.twitter.scalding._

class WordCount(args : Args) extends Job(args) {
  Tsv(args("doc"),
      ('doc_id, 'text),
      skipHeader = true)
    .read
    .flatMap('text -> 'token) {
      text : String => text.split("[ \\[\\]\\(\\),.]")
     }
    .groupBy('token) { _.size('count) }
    .write(Tsv(args("wc"), writeHeader = true))
}
```

Example: Scalding

github.com/twitter/scalding

- extends the Scala collections API so that distributed lists become "pipes" backed by Cascading
- code is compact, easy to understand
- nearly 1:1 between elements of conceptual flow diagram and function calls
- extensive libraries are available for linear algebra, abstract algebra, machine learning – e.g., *Matrix API*, *Algebird*, etc.
- significant investments by Twitter, Etsy, eBay, etc.
- less learning curve than Cascalog
- build scripts… those take a while to run :\

Example: Cascalog

cascalog.org

Cascalog

```
(ns impatient.core
  (:use [cascalog.api]
        [cascalog.more-taps :only (hfs-delimited)])
  (:require [clojure.string :as s]
            [cascalog.ops :as c])
  (:gen-class))

(defmapcatop split [line]
  "reads in a line of string and splits it by regex"
  (s/split line #"[\[\]\\\(\),.)\s]+"))

(defn -main [in out & args]
  (?<- (hfs-delimited out)
       [?word ?count]
       ((hfs-delimited in :skip-header? true) _ ?line)
       (split ?line :> ?word)
       (c/count ?count)))

; Paul Lam
; github.com/Quantisan/Impatient
```

Example: Cascalog

cascalog.org

- implements **Datalog** in Clojure, with predicates backed by Cascading – for a highly declarative language
- run ad-hoc queries from the Clojure REPL – approx. 10:1 code reduction compared with SQL
- composable subqueries, used for test-driven development (TDD) practices at scale
- **Leiningen** build: simple, no surprises, in Clojure itself
- more new deployments than other Cascading DSLs – Climate Corp is largest use case: 90% Clojure/Cascalog
- has a learning curve, limited number of Clojure developers
- aggregators are the magic, and those take effort to learn

Example: Apache Spark

spark-project.org

in-memory cluster computing,
by Matei Zaharia, Ram Venkataraman, et al.

- intended to make data analytics fast to write and to run
- load data into memory and query it repeatedly, much more quickly than with Hadoop
- APIs in Scala, Java and Python, shells in Scala and Python
- Shark (Hive on Spark), Spark Streaming (like Storm), etc.
- integrations with MLbase, Summingbird (in progress)

```
// word count
val file = spark.textFile("hdfs://…")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://…")
```

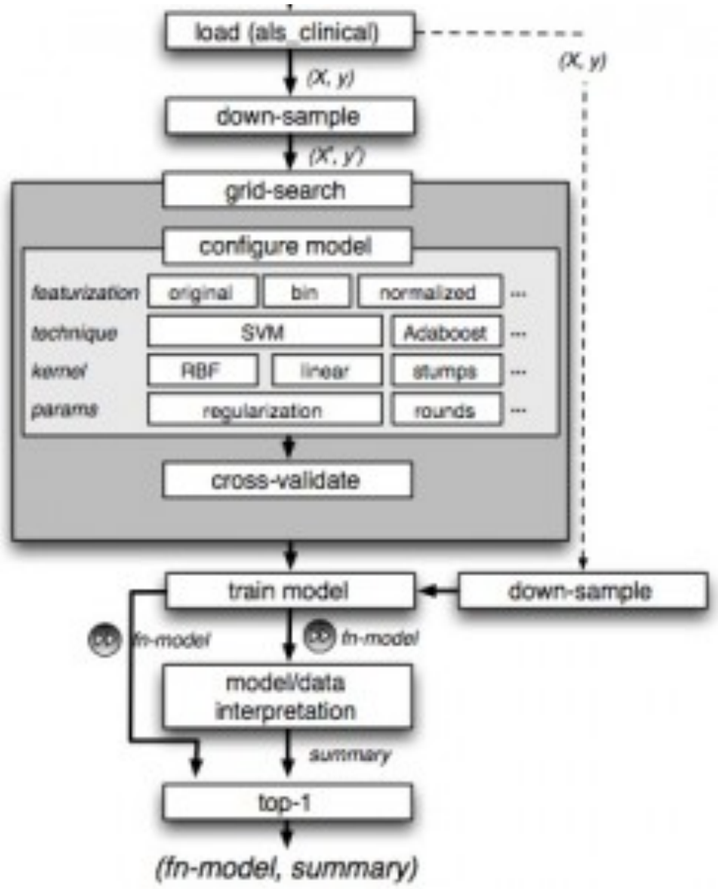
Example: MLbase

"distributed machine learning made easy"

- sponsored by UC Berkeley EECS AMP Lab
- MLlib – common algorithms, low-level, written atop Spark
- MLI – feature extraction, algorithm dev atop MLlib
- ML Optimizer – automates model selection, compiler/optimizer
- see article: http://strata.oreilly.com/2013/02/mlbase-scalable-machine-learning-made-accessible.html

```
data = load("hdfs://path/to/als_clinical")
// the features are stored in columns 2-10
X = data[, 2 to 10]
y = data[, 1]
model = do_classify(y, X)
```

mlbase.org



Example: Titan

thinkaurelius.github.io/titan

distributed graph database,
by Matthias Broecheler, Marko Rodriguez, et al.

- scalable graph database optimized for storing and querying graphs
- supports hundreds of billions of vertices and edges
- transactional database that can support thousands of concurrent users executing complex graph traversals
- supports search through Lucene, ElasticSearch
- can be backed by HBase, Cassandra, BerkeleyDB
- TinkerPop native graph stack with Gremlin, etc.

```
// who is hercules' paternal grandfather?
g.V('name','hercules').out('father').out('father').name
```

Example: MBrace

m-brace.net

"a .NET based software stack that enables easy large-scale distributed computation and big data analysis."

- declarative and concise distributed algorithms in F# asynchronous workflows
- scalable for apps in private datacenters or public clouds, e.g., Windows Azure or Amazon EC2
- tooling for interactive, REPL-style deployment, monitoring, management, debugging in Visual Studio
- leverages monads (similar to Summingbird)
- MapReduce as a library, but many patterns beyond

```
let rec mapReduce (map: 'T -> Cloud<'R>)
(reduce: 'R -> 'R -> Cloud<'R>)
(identity: 'R)
(input: 'T list) =
cloud {
match input with
| [] -> return identity
| [value] -> return! map value
| _ ->
let left, right = List.split input
let! r1, r2 =
(mapReduce map reduce identity left)
<||>
(mapReduce map reduce identity right)
return! reduce r1 r2
}
```

Data Workflows for Machine Learning:

Update the definitions…

“Nine Points to Discuss”

Data Workflows

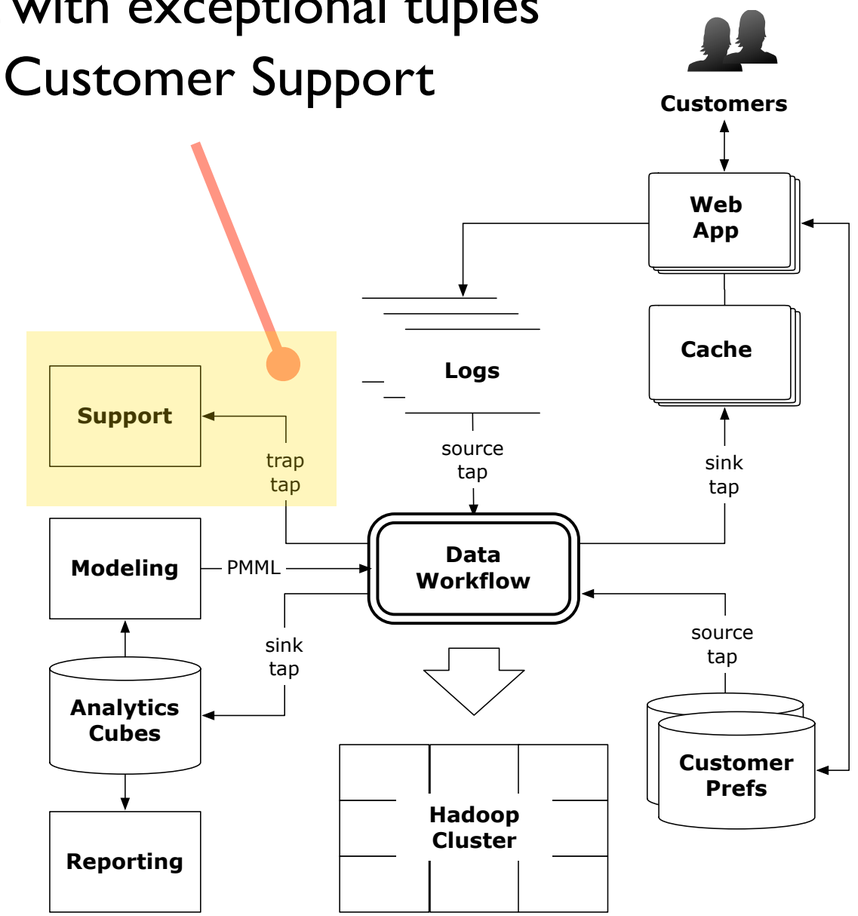
1

includes people, defines oversight for exceptional data

Formally speaking, workflows *include people* (cross-dept) and define oversight for exceptional data

…otherwise, *data flow* or *pipeline* would be more apt

…examples include **Cascading** *traps*, with exceptional tuples routed to a review organization, e.g., Customer Support



Data Workflows

Workflows impose a *separation of concerns*, allowing for multiple *abstraction layers* in the tech stack

…specify *what* is required, not *how* to accomplish it

…articulating the business logic

… **Cascading** leverages *pattern language*

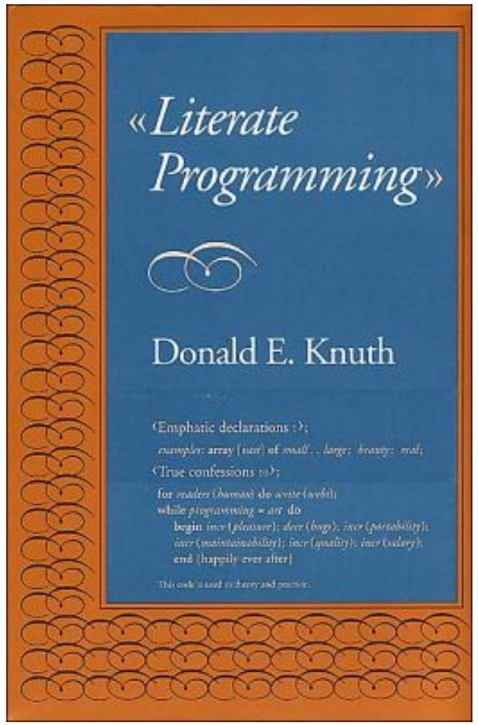
…related notions from Knuth, **literate programming**

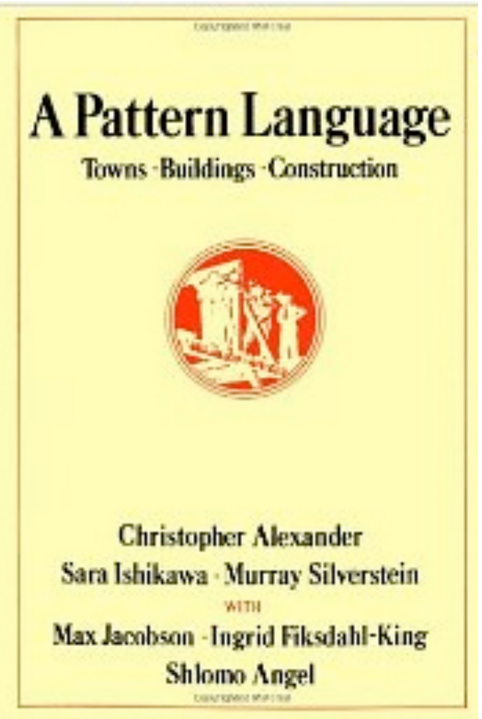
…not unlike BPM/BPEL for Big Data

…examples: **IPython**, **R Markdown**, etc.

2

separation of concerns, allows for literate programming





Data Workflows

Multiple *abstraction layers* in the tech stack are needed, but still emerging

…feedback loops based on *machine data*

…optimizers feed on abstraction

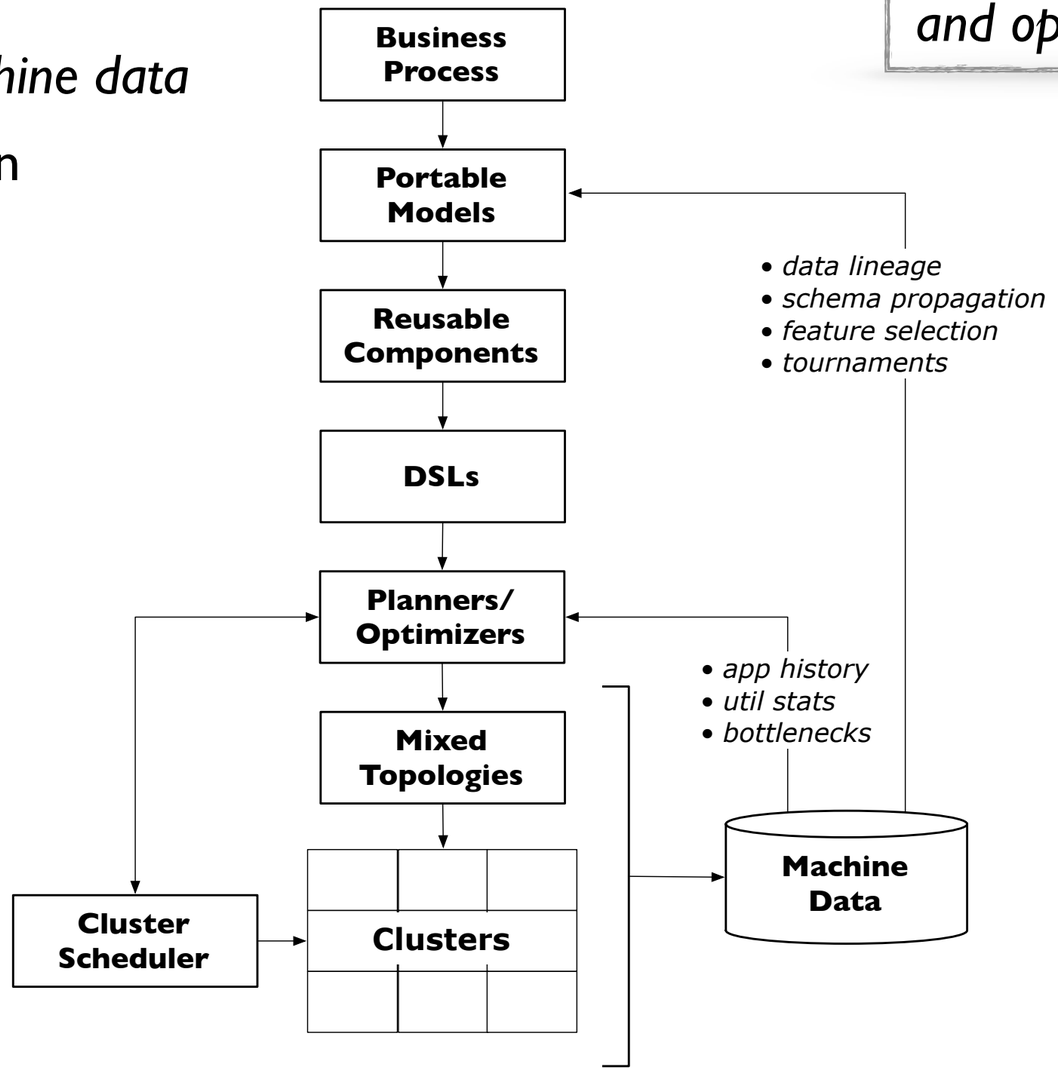
…metadata accounting:

- track data lineage
- propagate schema
- model / feature usage
- ensemble performance

…app history accounting:

- util stats, mixing workloads
- heavy-hitters, bottlenecks
- throughput, latency

3 *multiple abstraction layers for metadata, feedback, and optimization*



Data Workflows

Multiple
needed, but still emerging

…feedback loops based on

…optimizers feed on abstraction

…metadata accounting:

- track data lineage
- propagate
- model / feature usage
- ensemble performance

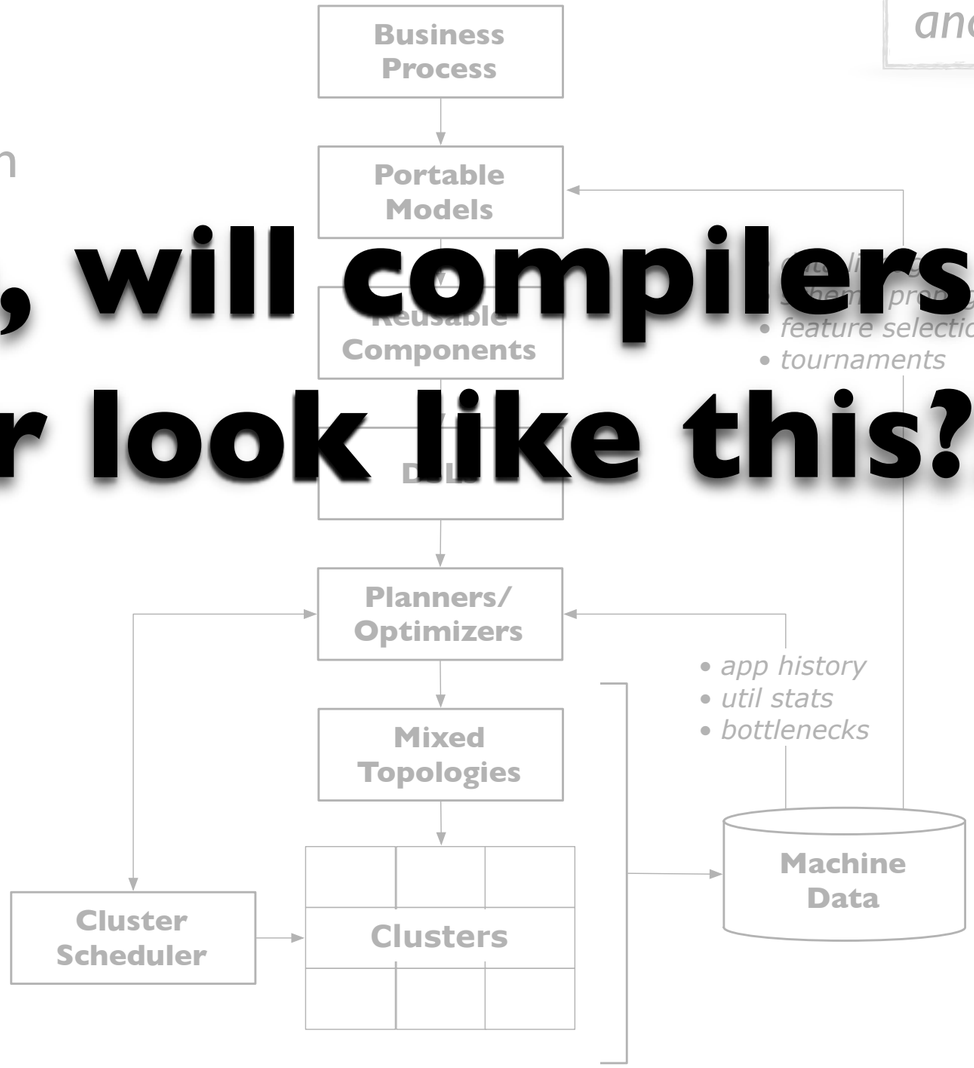
…app history accounting:

- util stats, mixing workloads
- heavy-hitters, bottlenecks
- throughput, latency

3

multiple abstraction layers for metadata, feedback, and optimization

Um, will compilers ever look like this??



Data Workflows

4 *testing: model evaluation, TDD, app deployment*

Workflows must provide for *test*, which is no simple matter

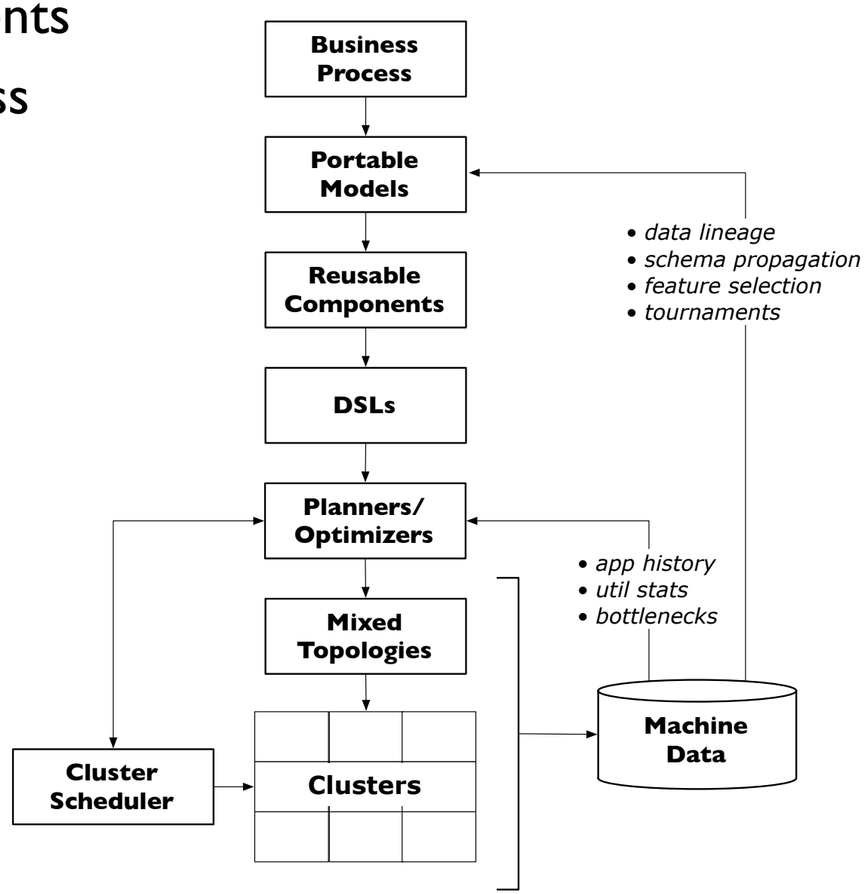
…testing is required on three abstraction layers:

- model portability/evaluation, ensembles, tournaments
- TDD in reusable components and business process
- continuous integration / continuous deployment

…examples: **Cascalog** for TDD, **PMML** for model eval

…still so much more to improve

…keep in mind that workflows involve people, too



Data Workflows

5 *future-proof system integration, scale-out, ops*

Workflows enable *system integration*

…future-proof integrations and scale-out

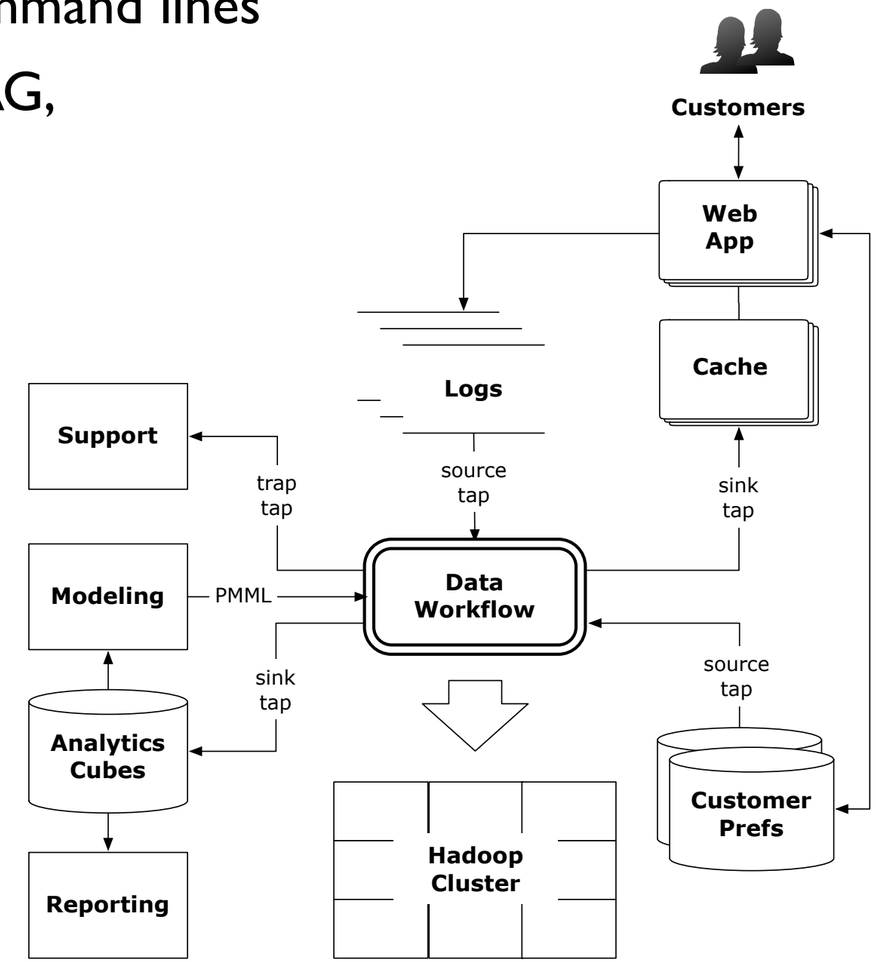
…build *components* and *apps*, not jobs and command lines

…allow compiler optimizations across the DAG,
i.e., cross-dept contributions

…minimize operationalization costs:

- troubleshooting, debugging at scale
- exception handling
- notifications, instrumentation

…examples: **KNIME**, **Cascading**, etc.



Data Workflows

6

visualizing allows people to collaborate through code

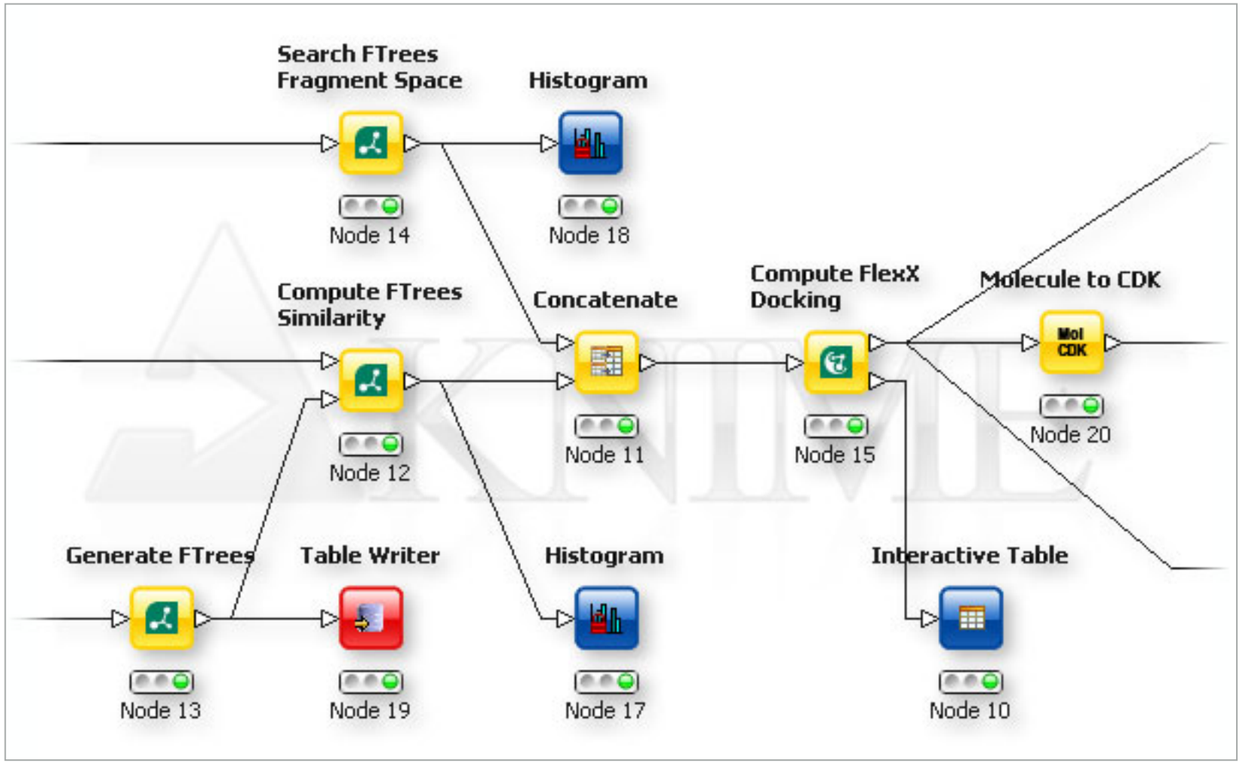
Visualizing workflows, what a great idea.

…the practical basis for:

- collaboration
- rapid prototyping
- component reuse

…examples: KNIME wins best in category

…Cascading generates flow diagrams, which are a nice start



Data Workflows

7

abstract algebra and functional programming containerize business process

Abstract algebra, containerizing workflow metadata

…monoids, semigroups, etc., allow for reusable components with well-defined properties for running in parallel at scale

…let business process be agnostic about underlying topologies, with analogy to Linux containers (Docker, etc.)

…compose functions, take advantage of sparsity (monoids)

…because "data is fully functional" – FP for s/w eng benefits

…aggregators are almost always "magic", now we can solve for common use cases

…read **Monoidify! Monoids as a Design Principle for Efficient MapReduce Algorithms** by Jimmy Lin

…**Cascading** introduced some notions of this circa 2007, but didn't make it explicit

…examples: **Algebird** in **Summingbird**, **Simmer**, **Spark**, etc.

Data Workflows

Workflow needs vary in time, and need to blend time

…something something *batch*, something something *low-latency*

…scheduling batch isn't hard; scheduling low-latency is computationally brutal (see the **Omega** paper)

…because people like saying "Lambda Architecture", it gives them goosebumps, or something

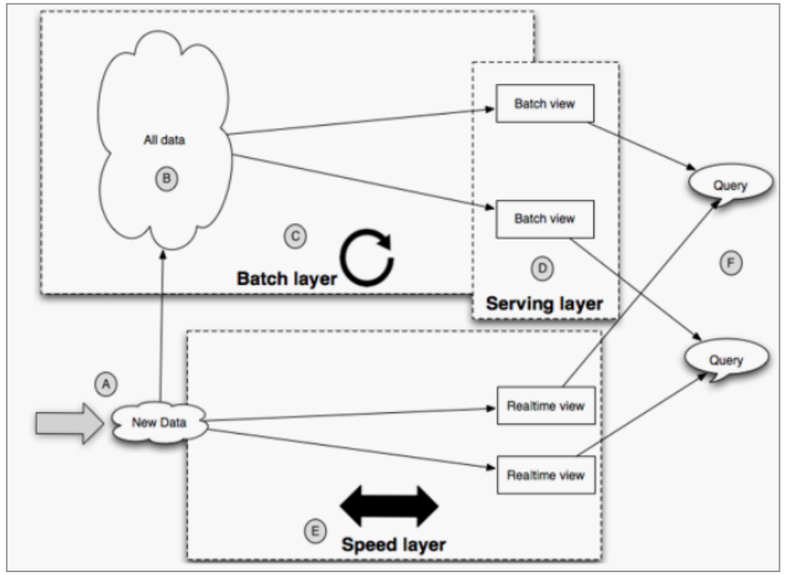
…because *real-time* means so many different things

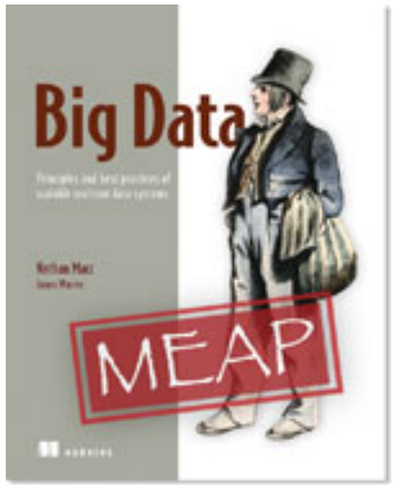
…because batch windows are so 1964

…examples: **Summingbird**, **Oryx**

8

blend results from different time scales: batch plus low-latency





Big Data
Nathan Marz, James Warren
manning.com/marz

Data Workflows

Workflows may define contexts in which model selection possibly becomes a compiler problem

…for example, see **Boyd, Parikh, et al.**

…ultimately optimizing for *loss function* + *regularization term*

…probably not ready for prime-time tomorrow

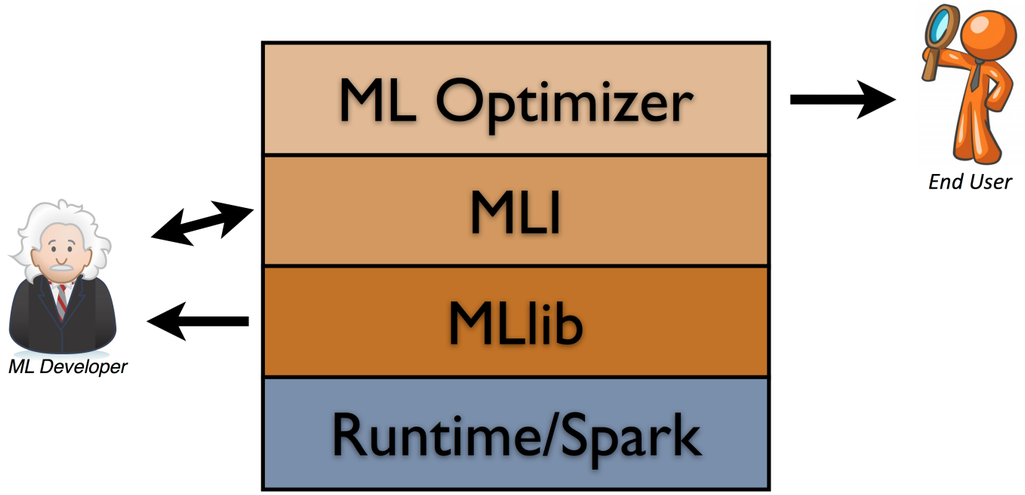
…examples: **MLbase**

9

optimize learners in context, to make model selection potentially a compiler problem

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

f(x): loss function
g(z): regularization term



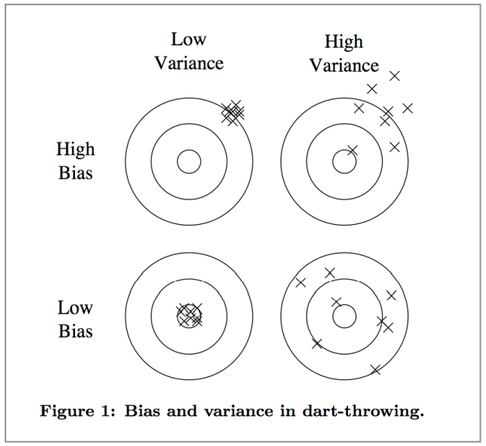


Figure 1: Bias and variance in dart-throwing.

Data Workflows

bonus

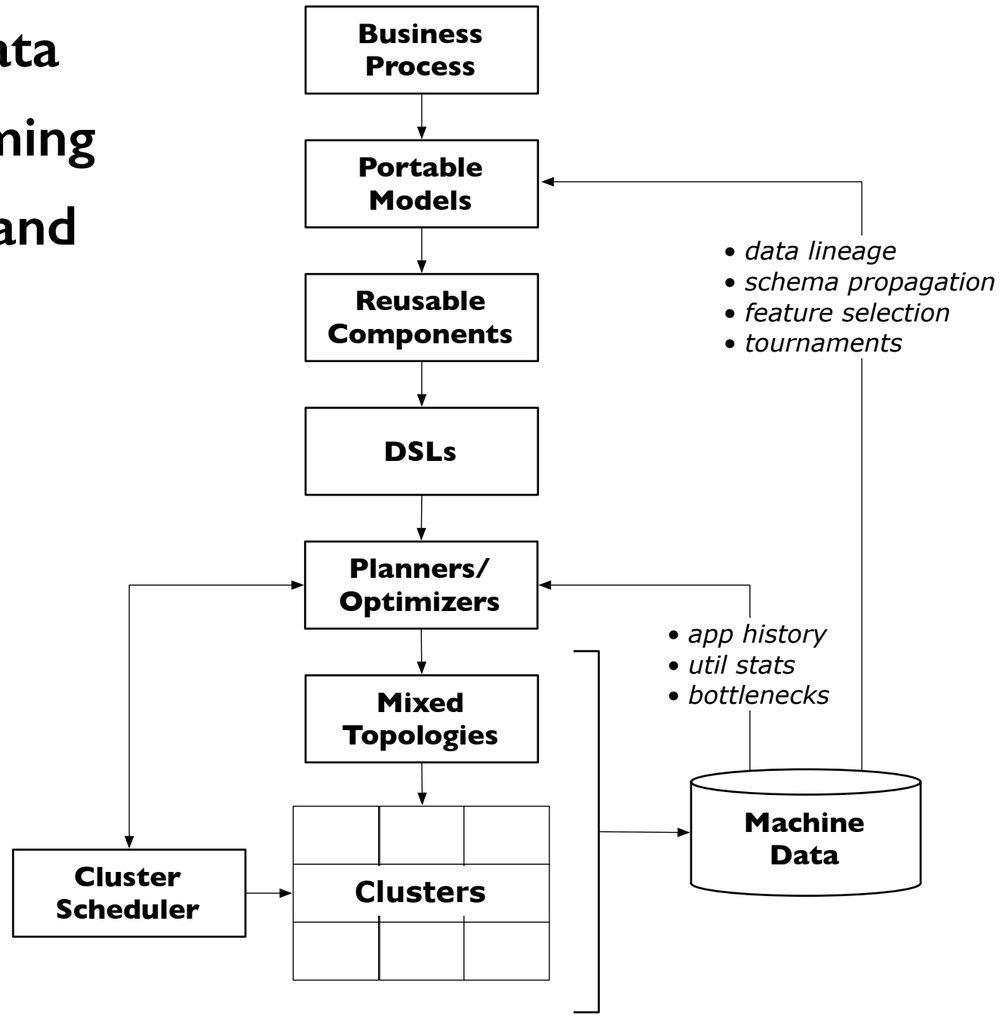
For one of the best papers about what workflows really truly require, see **Out of the Tar Pit** by Moseley and Marks

Citations

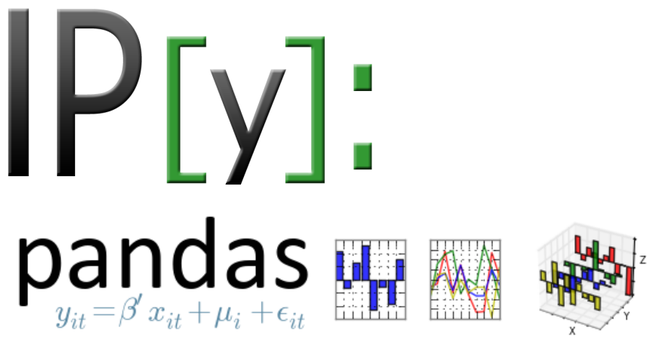
1299	Monads for functional programming - Wadler - 1995
801	A relational model of data for large shared data banks - Codd - 1970
614	The Art of Prolog - Sterling, Shapiro - 1994
535	No Silver Bullet: Essence and Accidents of Software Engineering - Brooks - 1987
502	An Introduction to Database Systems - DATE - 1982
435	Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs - Backus - 1978
237	Extending the database relational model to capture more meaning - CODD - 1979
197	Functional reactive animation - Elliott, Hudak - 1997
164	Object Oriented Design with Applications. The Benjamin/Cummings - Booch - 1991
148	The Craft of Prolog - O'Keefe - 1990
121	Algorithm = Logic + Control - Kowalski - 1979
119	Software Engineering: Report of a conference sponsored by - Peter, Randell - 1968
96	The humble programmer - DIJKSTRA - 1972
87	The Relational Model for Database Management": Version 2. Addison-Wesley - Codd - 2000
86	Specifications are not (necessarily) executable - Hayes, Jones - 1989
53	The Emperor's Old Clothes - Hoare - 1981

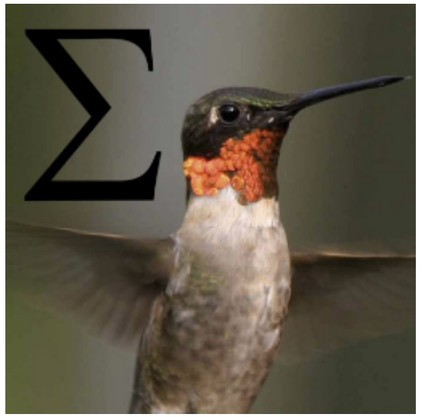
Nine Points for Data Workflows — a wish list

1. includes people, defines oversight for exceptional data
2. separation of concerns, allows for literate programming
3. multiple abstraction layers for metadata, feedback, and optimization
4. testing: model evaluation, TDD, app deployment
5. future-proof system integration, scale-out, ops
6. visualizing workflows allows people to collaborate through code
7. abstract algebra and functional programming containerize business process
8. blend results from different time scales: batch plus low-latency
9. optimize learners in context, to make model selection potentially a compiler problem




KNIME


IP[y]:
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$


Σ


actian


open data
Augustus

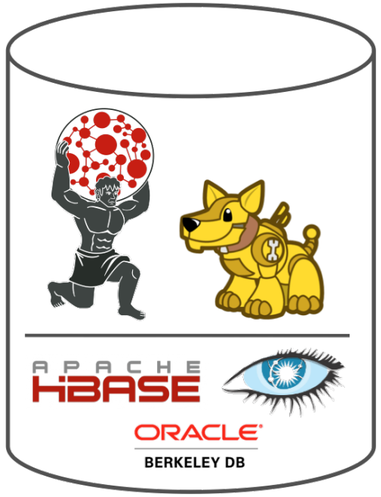

Scalding


RStudio


{m}


Cascalog


Spark


APACHE
HBASE
ORACLE
BERKELEY DB


julia


cascading


ML base


ADAPA®


mahout

Nine Points for Data Workflows — a scorecard

	Spark, MLbase	Oryx	Summing bird	Cascalog	Cascading	KNIME	Py Data	R Markdown	MBrace
includes people, exceptional data									
separation of concerns									
multiple abstraction layers									
testing in depth									
future-proof system									
visualize to collab									
can haz monoids									
blends batch + “real-time”									
optimize learners in context									
can haz PMML		✔			✔	✔	✔	✔	

Data Workflows for Machine Learning:

PMML...

“Oh, by the way...”

PMML – an industry standard

- established XML standard for predictive model markup
- organized by Data Mining Group (DMG), since 1997
  http://dmg.org/
- members: *IBM*, *SAS*, *Visa*, *FICO*, *Equifax*, *NASA*, *Microstrategy*, *Microsoft*, etc.
- PMML concepts for *metadata*, *ensembles*, etc., translate directly into workflow abstractions, e.g., Cascading

"PMML is the leading standard for statistical and data mining models and supported by over 20 vendors and organizations. With PMML, it is easy to develop a model on one system using one application and deploy the model on another system using another application."

wikipedia.org/wiki/Predictive_Model_Markup_Language

PMML – vendor coverage

PMML – model coverage

- Association Rules: *AssociationModel* element
- Cluster Models: *ClusteringModel* element
- Decision Trees: *TreeModel* element
- Naïve Bayes Classifiers: *NaiveBayesModel* element
- Neural Networks: *NeuralNetwork* element
- Regression: *RegressionModel* and *GeneralRegressionModel* elements
- Rulesets: *RuleSetModel* element
- Sequences: *SequenceModel* element
- Support Vector Machines: *SupportVectorMachineModel* element
- Text Models: *TextModel* element
- Time Series: *TimeSeriesModel* element

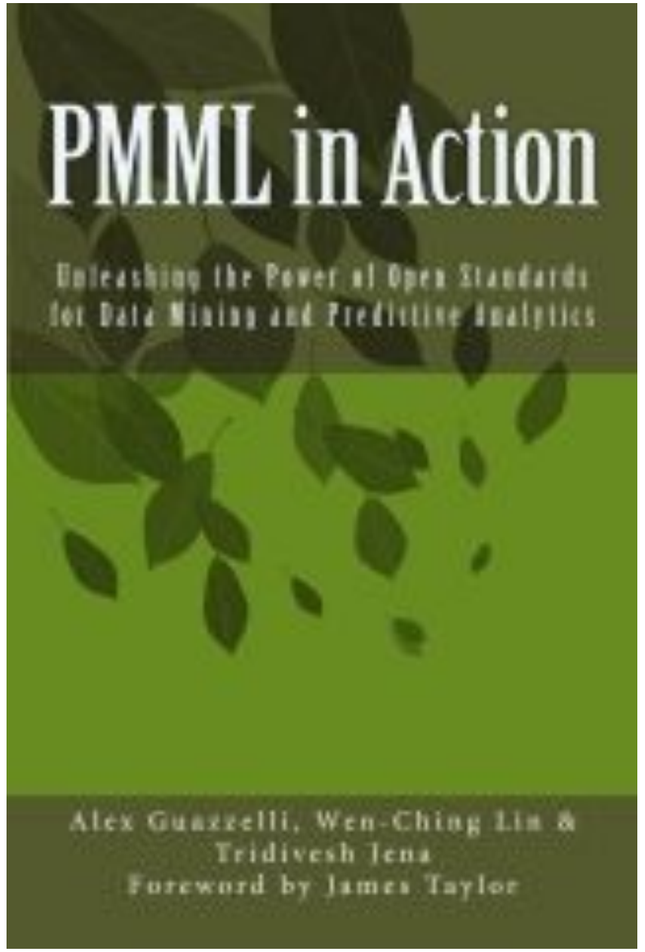
ibm.com/developerworks/industry/library/ind-PMML2/

PMML – further study

PMML in Action
Alex Guazzelli, Wen-Ching Lin, Tridivesh Jena
amazon.com/dp/1470003244

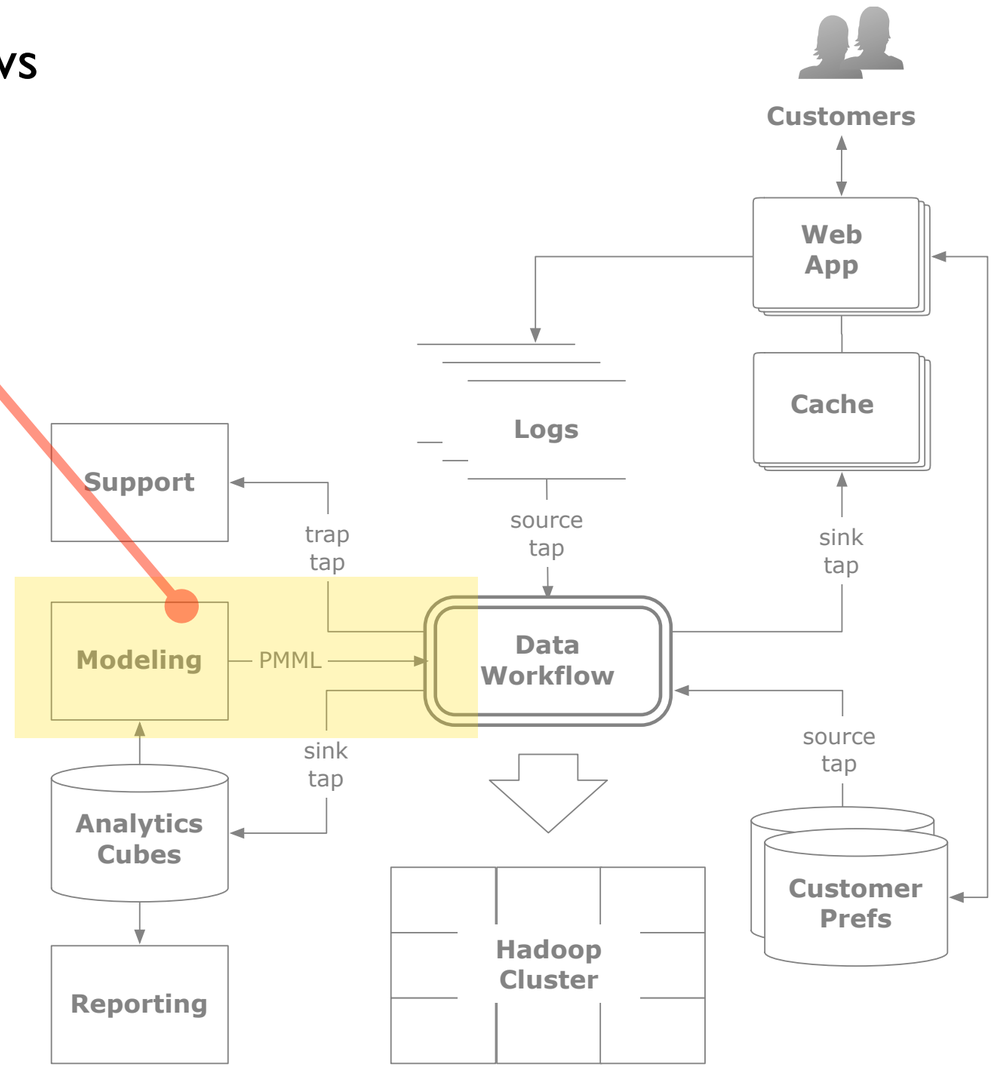
See also excellent resources at:

zementis.com/pmml.htm



Pattern – model scoring

- a PMML library for Cascading workflows
- migrate workloads: *SAS*, *Teradata*, etc., exporting predictive models as PMML
- great open source tools – *R*, *Weka*, *KNIME*, *Matlab*, *RapidMiner*, etc.
- leverage PMML as another kind of DSL
- only for scoring models, not training



Pattern – model scoring

- originally intended as a general purpose, tuple-oriented model scoring engine for JVM-based frameworks
- library plugs into Cascading (Hadoop), and ostensibly Cascalog, Scalding, Storm, Spark, Summingbird, etc.
- dev forum: https://groups.google.com/forum/#!forum/pattern-user
- original GitHub repo: https://github.com/ceteri/pattern/
- somehow, the *other* fork became deeply intertwined with Cascading dependencies…
- work in progress to integrate with other frameworks, add models, and also train models at scale using Spark

Pattern – create a model in R

```
## train a RandomForest model

f <- as.formula("as.factor(label) ~ .")
fit <- randomForest(f, data_train, ntree=50)

## test the model on the holdout test set

print(fit$importance)
print(fit)

predicted <- predict(fit, data)
data$predicted <- predicted
confuse <- table(pred = predicted, true = data[,1])
print(confuse)

## export predicted labels to TSV

write.table(data, file=paste(dat_folder, "sample.tsv", sep="/"),
  quote=FALSE, sep="\t", row.names=FALSE)

## export RF model to PMML

saveXML(pmml(fit), file=paste(dat_folder, "sample.rf.xml", sep="/"))
```

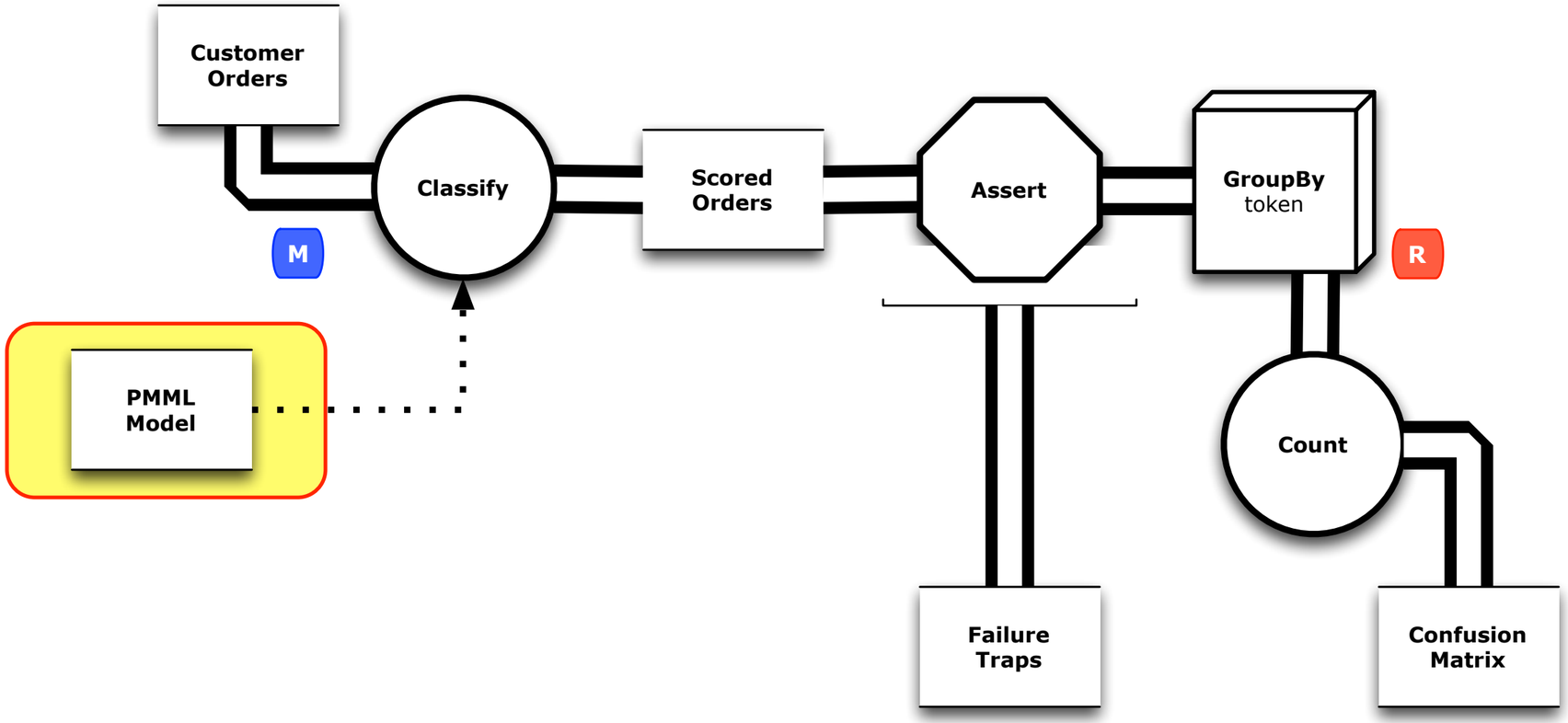
Pattern – capture model parameters as PMML

```
<?xml version="1.0"?>
<PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.dmg.org/PMML-4_0
 http://www.dmg.org/v4-0/pmml-4-0.xsd">
 <Header copyright="Copyright (c)2012 Concurrent, Inc." description="Random Forest Tree Model">
  <Extension name="user" value="ceteri" extender="Rattle/PMML"/>
  <Application name="Rattle/PMML" version="1.2.30"/>
  <Timestamp>2012-10-22 19:39:28</Timestamp>
 </Header>
 <DataDictionary numberOfFields="4">
  <DataField name="label" optype="categorical" dataType="string">
   <Value value="0"/>
   <Value value="1"/>
  </DataField>
  <DataField name="var0" optype="continuous" dataType="double"/>
  <DataField name="var1" optype="continuous" dataType="double"/>
  <DataField name="var2" optype="continuous" dataType="double"/>
 </DataDictionary>
 <MiningModel modelName="randomForest_Model" functionName="classification">
  <MiningSchema>
   <MiningField name="label" usageType="predicted"/>
   <MiningField name="var0" usageType="active"/>
   <MiningField name="var1" usageType="active"/>
   <MiningField name="var2" usageType="active"/>
  </MiningSchema>
  <Segmentation multipleModelMethod="majorityVote">
   <Segment id="1">
    <True/>
    <TreeModel modelName="randomForest_Model" functionName="classification" algorithmName="randomForest" splitCharacteristic="binarySplit">
     <MiningSchema>
      <MiningField name="label" usageType="predicted"/>
      <MiningField name="var0" usageType="active"/>
      <MiningField name="var1" usageType="active"/>
      <MiningField name="var2" usageType="active"/>
     </MiningSchema>
...
```

Pattern – score a model, within an app

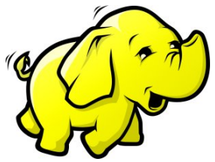
```
public static void main( String[] args ) throws RuntimeException {
  String inputPath = args[ 0 ];
  String classifyPath = args[ 1 ];
  // set up the config properties
  Properties properties = new Properties();
  AppProps.setApplicationJarClass( properties, Main.class );
  HadoopFlowConnector flowConnector = new HadoopFlowConnector( properties );
   // create source and sink taps
  Tap inputTap = new Hfs( new TextDelimited( true, "\t" ), inputPath );
  Tap classifyTap = new Hfs( new TextDelimited( true, "\t" ), classifyPath );
   // handle command line options
  OptionParser optParser = new OptionParser();
  optParser.accepts( "pmml" ).withRequiredArg();
  OptionSet options = optParser.parse( args );

  // connect the taps, pipes, etc., into a flow
  FlowDef flowDef = FlowDef.flowDef().setName( "classify" )
    .addSource( "input", inputTap )
    .addSink( "classify", classifyTap );

  if( options.hasArgument( "pmml" ) ) {
    String pmmlPath = (String) options.valuesOf( "pmml" ).get( 0 );
    PMMLPlanner pmmlPlanner = new PMMLPlanner()
      .setPMMLInput( new File( pmmlPath ) )
      .retainOnlyActiveIncomingFields()
      .setDefaultPredictedField( new Fields( "predict", Double.class ) ); // default value if missing from the model
    flowDef.addAssemblyPlanner( pmmlPlanner );
    }

  // write a DOT file and run the flow
  Flow classifyFlow = flowConnector.connect( flowDef );
  classifyFlow.writeDOT( "dot/classify.dot" );
  classifyFlow.complete();
  }
```

Pattern – score a model, using pre-defined Cascading app



Pattern – score a model, using pre-defined Cascading app

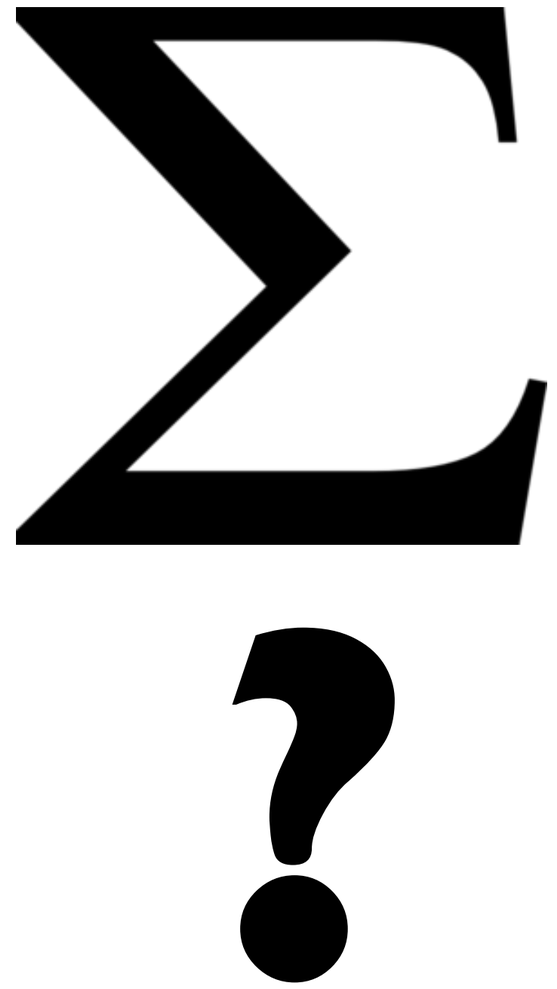
```
## run an RF classifier at scale

hadoop jar build/libs/pattern.jar \
 data/sample.tsv out/classify out/trap \
 --pmml data/sample.rf.xml
```



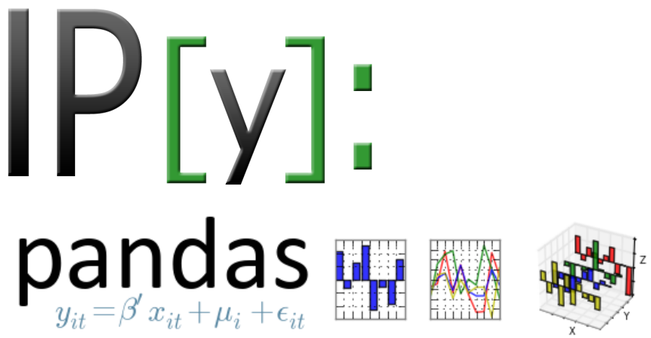
Data Workflows for Machine Learning:

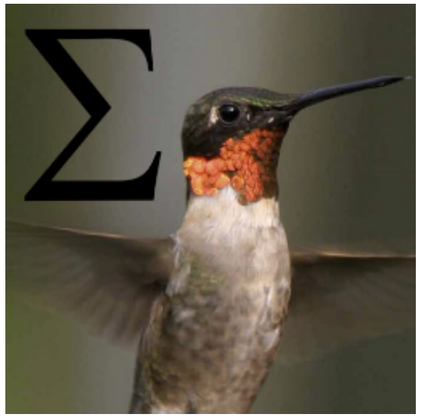
Summary

“feedback and discussion”




KNIME


IP[y]:
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$


Σ


actian


open data
Augustus

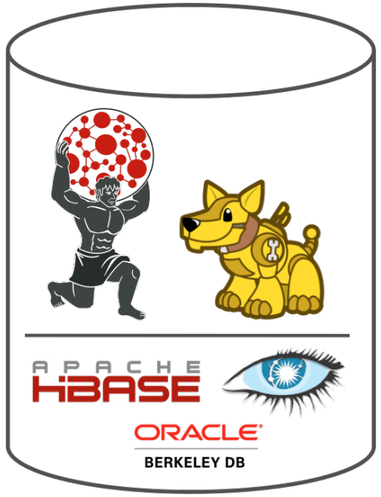

Scalding


RStudio


{m}


Cascalog


Spark


APACHE
HBASE
ORACLE
BERKELEY DB


julia


cascading


ML base


ADAPA®


mahout

Nine Points for Data Workflows — a scorecard

	Spark, MLbase	Oryx	Summing bird	Cascalog	Cascading	KNIME	Py Data	R Markdown	MBrace
includes people, exceptional data									
separation of concerns									
multiple abstraction layers									
testing in depth									
future-proof system									
visualize to collab									
can haz monoids									
blends batch + “real-time”									
optimize learners in context									
can haz PMML		✔			✔	✔	✔	✔	

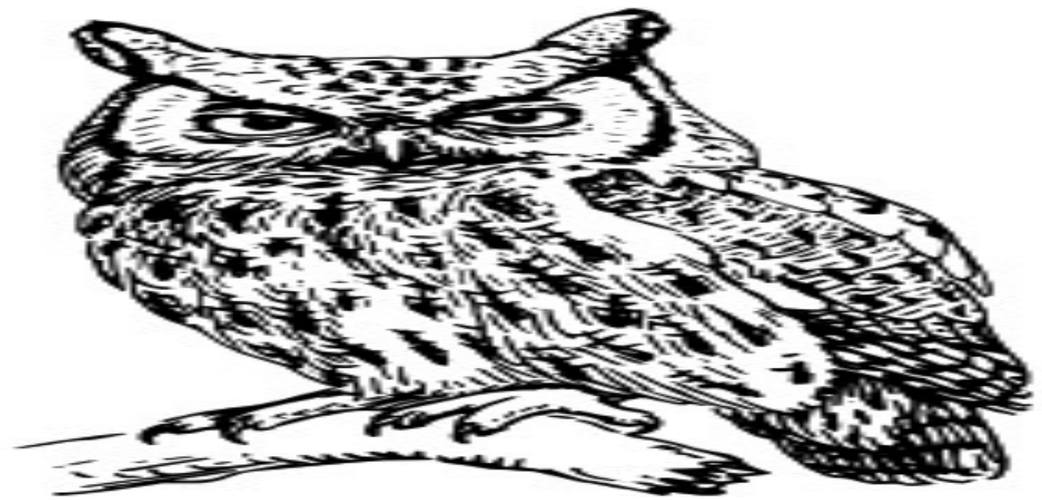
PMML – what's needed?

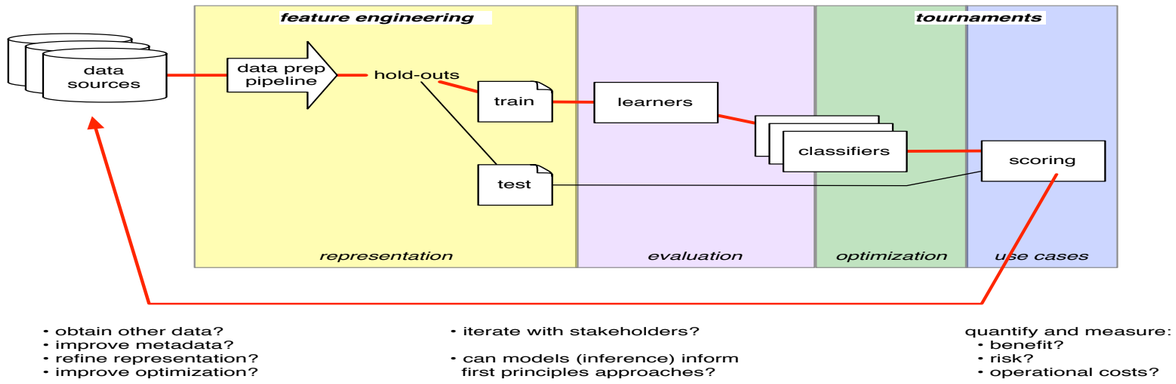
in the *language standard*:

- data preparation
- data sources/sinks as parameters
- track data lineage
- more monoid, less XML
- handle data exceptions => TDD
- updates/active learning?
- tournaments?

in the *workflow frameworks*:

- include feature engineering w/ model?
- support evaluation better
- build ensembles





- obtain other data?
- improve metadata?
- refine representation?
- improve optimization?

- iterate with stakeholders?
- can models (inference) inform first principles approaches?

quantify and measure:
- benefit?
- risk?
- operational costs?

Enterprise Data Workflows with Cascading

O'Reilly, 2013

shop.oreilly.com/product/0636920028536.do

monthly newsletter for updates, events, conference summaries, etc.:

liber118.com/pxn/