# High Performance Machine Learning in R with H2O

## ISM HPC on R Workshop

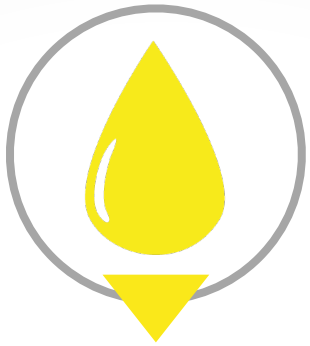
Tokyo, Japan
October 2015

**Erin LeDell Ph.D.**

H2O.ai

# Introduction

- Statistician & Machine Learning Scientist at H2O.ai in Mountain View, California, USA
- Ph.D. in Biostatistics with Designated Emphasis in Computational Science and Engineering from UC Berkeley (focus on Machine Learning)
- Worked as a data scientist at several startups
- Written a handful of machine learning R packages

# Agenda

- What/who is H2O.ai?
- H2O Machine Learning Software
- H2O Architecture
- H2O in R & Demo
- Sparking Water: H2O on Spark
- Ensembles in H2O & Demo

# H2O.ai

## H2O Company

- Team: 35. Founded in 2012, Mountain View, CA
- Stanford Math & Systems Engineers

## H2O Software

- Open Source Software
- Ease of Use via Web Interface
- R, Python, Scala, Spark & Hadoop Interfaces
- Distributed Algorithms Scale to Big Data

# H2O.ai Founders

## SriSatish Ambati

- CEO and Co-founder at H2O.ai
- Past:  Platfora, Cassandra, DataStax, Azul Systems, UC Berkeley

---

## Dr. Cliff Click

- CTO and Co-founder at H2O.ai
- Past:  Azul Systems, Sun Microsystems
- Developed the Java HotSpot Server Compiler at Sun
- PhD in CS from Rice University

# Scientific Advisory Council

## Dr. Trevor Hastie

- John A. Overdeck Professor of Mathematics, Stanford University
- PhD in Statistics, Stanford University
- Co-author, *The Elements of Statistical Learning: Prediction, Inference and Data Mining*
- Co-author with John Chambers, *Statistical Models in S*
- Co-author, *Generalized Additive Models*
- 108,404 citations (via Google Scholar)

## Dr. Rob Tibshirani

- Professor of Statistics and Health Research and Policy, Stanford University
- PhD in Statistics, Stanford University
- COPPS Presidents' Award recipient
- Co-author, *The Elements of Statistical Learning: Prediction, Inference and Data Mining*
- Author, *Regression Shrinkage and Selection via the Lasso*
- Co-author, *An Introduction to the Bootstrap*

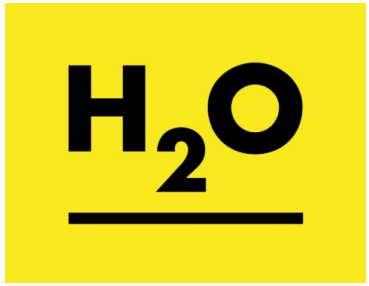
## Dr. Stephen Boyd

- Professor of Electrical Engineering and Computer Science, Stanford University
- PhD in Electrical Engineering and Computer Science, UC Berkeley
- Co-author, Convex Optimization
- Co-author, Linear Matrix Inequalities in System and Control Theory
- Co-author, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*

# H2O Platform

## Part 1 of 7
## High Performance ML in R with H2O

# H2O Software




H2O is an open source, distributed, Java machine learning library.


APIs are available for:
R, Python, Scala & JSON

# H2O Overview

## Speed Matters!

- Time is valuable
- In-memory is faster
- Distributed is faster
- High speed AND accuracy

## No Sampling

- Scale to big data
- Access data links
- Use all data without sampling

## Interactive UI

- Web-based modeling with H2O Flow
- Model comparison

## Cutting-Edge Algorithms

- Suite of cutting-edge machine learning algorithms
- Deep Learning & Ensembles
- NanoFast Scoring Engine

# Current Algorithm Overview

## Statistical Analysis

- Linear Models (GLM)
- Cox Proportional Hazards
- Naïve Bayes

## Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- R Package - Super Learner Ensembles

## Deep Neural Networks

- Multi-layer Feed-Forward Neural Network
- Auto-encoder
- Anomaly Detection
- Deep Features

## Clustering

- K-Means

## Dimension Reduction

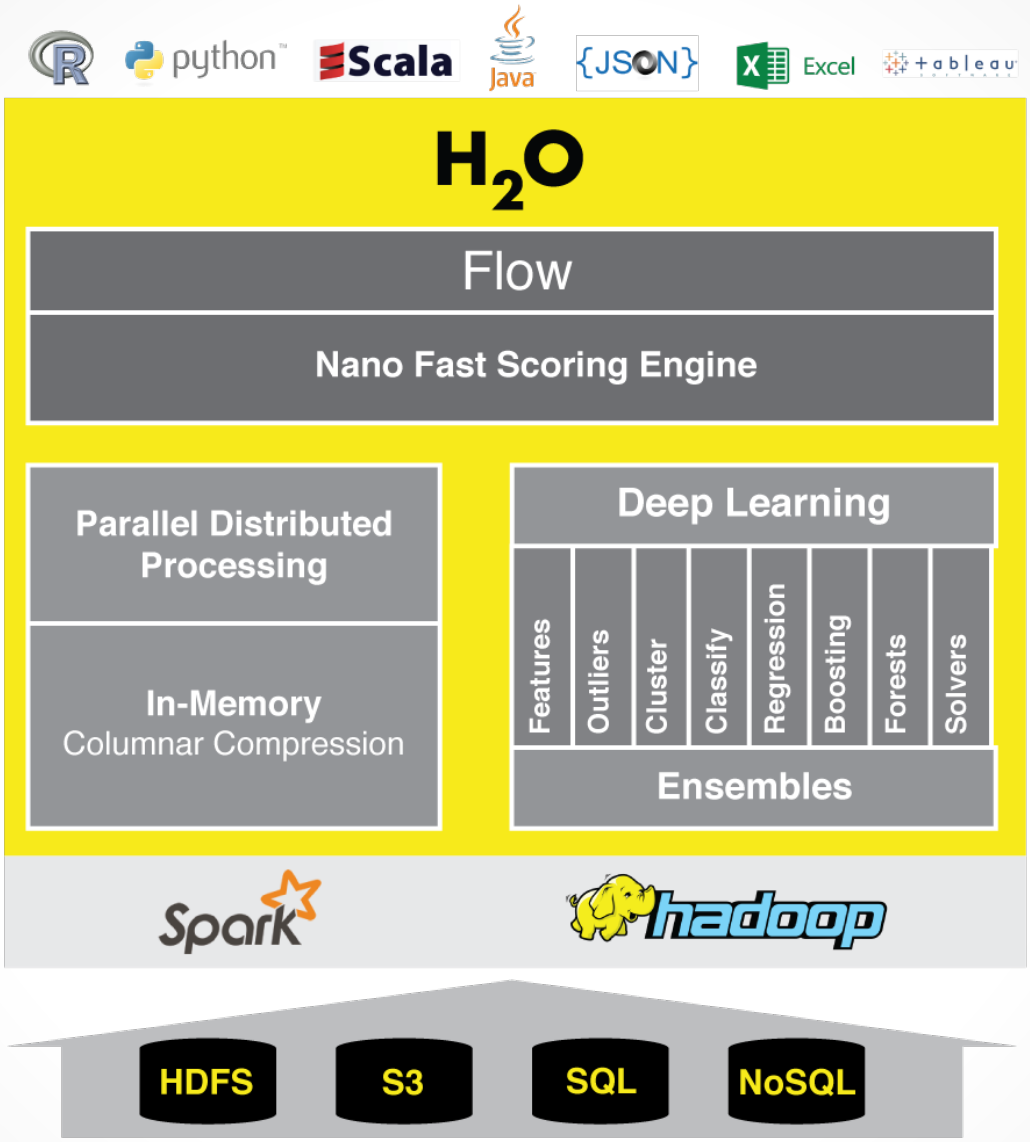
- Principal Component Analysis
- Generalized Low Rank Models

## Solvers & Optimization

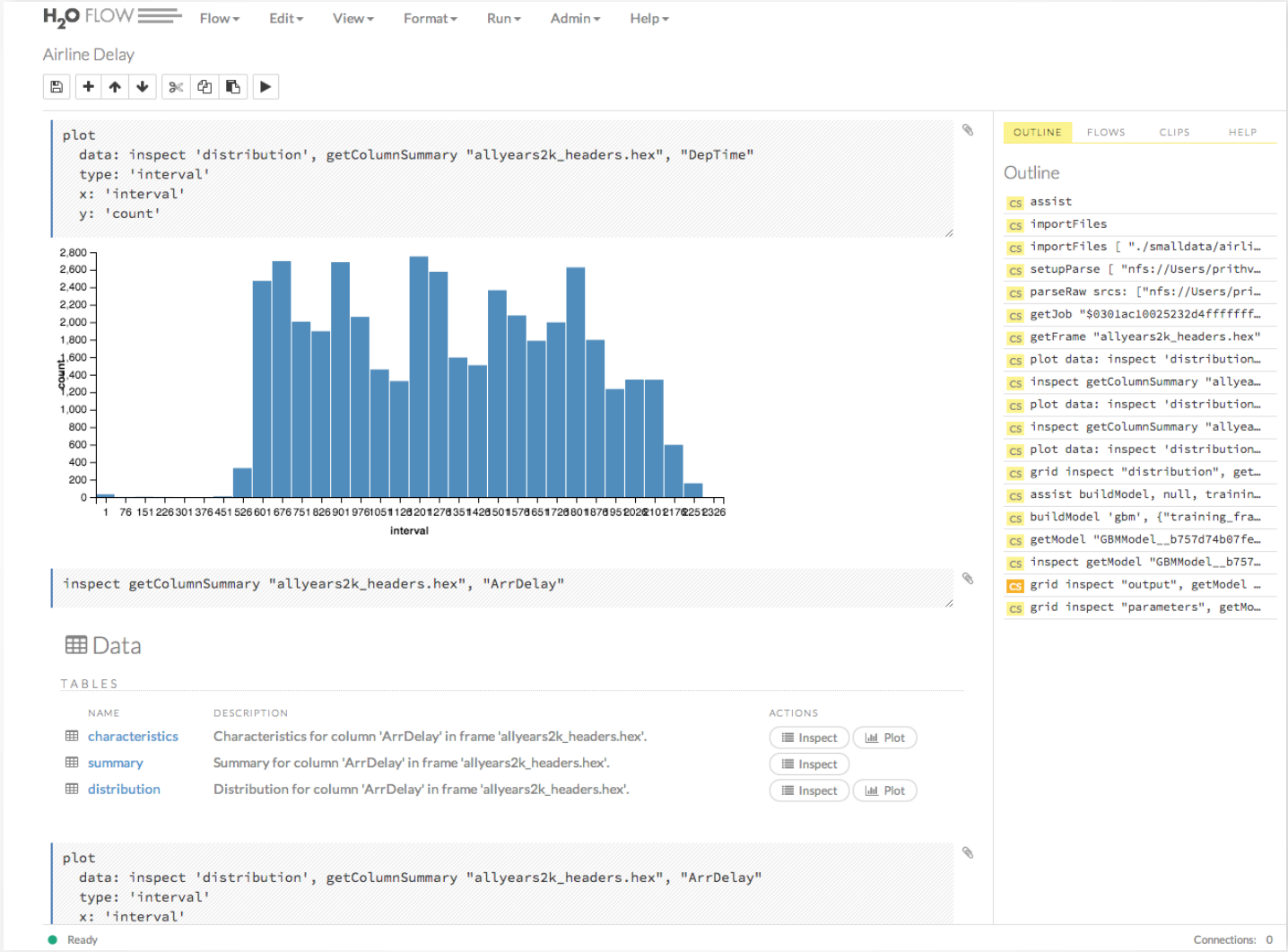
- Generalized ADMM Solver
- L-BFGS (Quasi Newton Method)
- Ordinary Least-Square Solver
- Stochastic Gradient Descent

## Data Munging

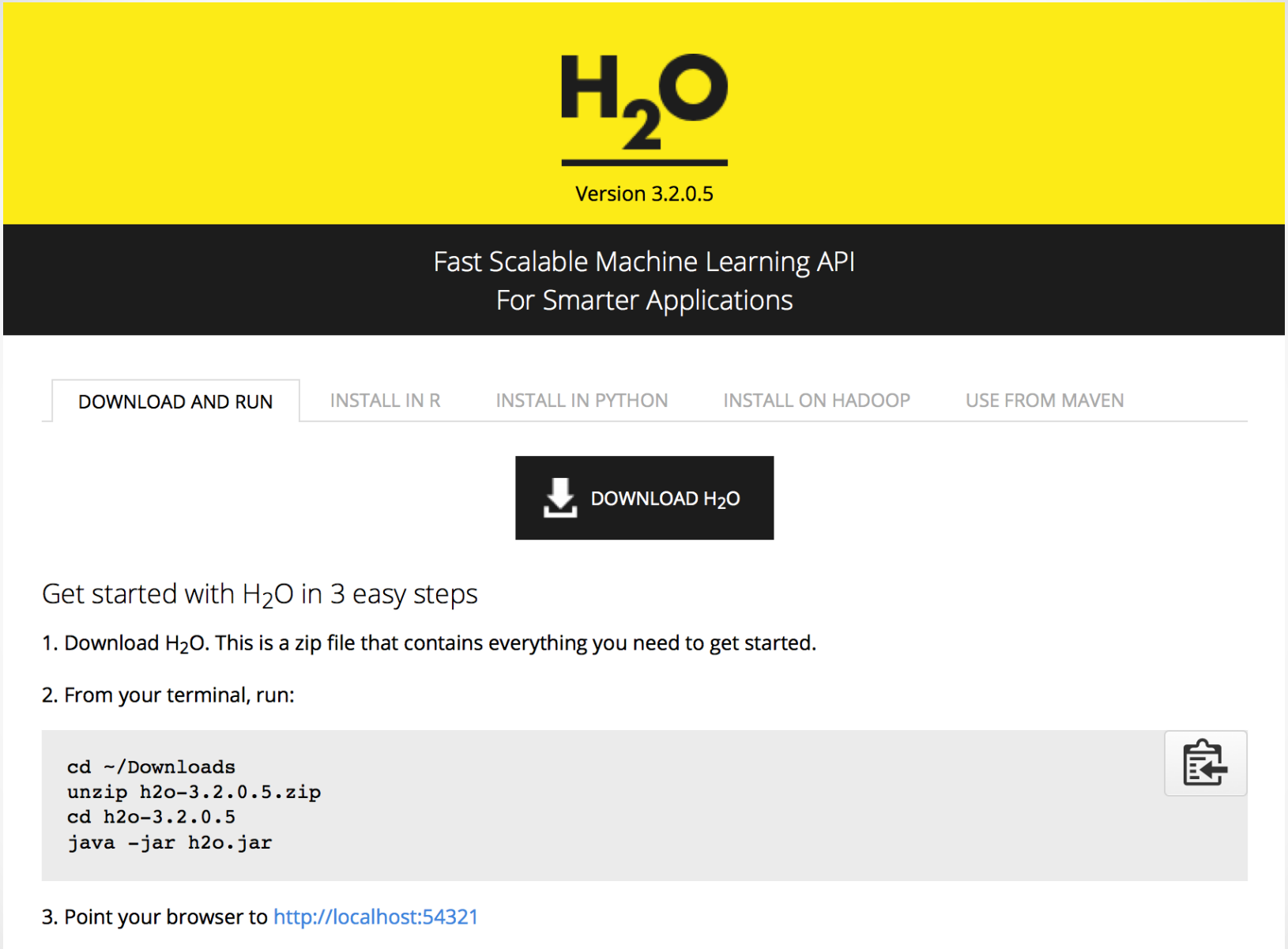
- Integrated R-Environment
- Slice, Log Transform


python
Scala
Java
{JSON}
Excel
tableau
H2O
Flow
Nano Fast Scoring Engine
Parallel Distributed Processing
In-Memory
Columnar Compression
Deep Learning
Features
Outliers
Cluster
Classify
Regression
Boosting
Forests
Solvers
Ensembles
Spark
hadoop
HDFS
S3
SQL
NoSQL

# H2O Flow Interface




H2O.ai
Machine Intelligence




H2O.ai
Machine Intelligence

# http://h2o.ai/download

# H2O

GITTER JOIN CHAT →

H2O scales statistics, machine learning, and math over Big Data.

H2O uses familiar interfaces like R, Python, Scala, the Flow notebook graphical interface, Excel, & JSON so that Big Data enthusiasts & experts can explore, munge, model, and score datasets using a range of algorithms including advanced ones like Deep Learning. H2O is extensible so that developers can add data transformations and model algorithms of their choice and access them through all of those clients.

Data collection is easy. Decision making is hard. H2O makes it fast and easy to derive insights from your data through faster and better predictive modeling. H2O allows online scoring and modeling in a single platform.





https://github.com/h2oai/h2o-3

# H2O Architecture

## Part 2 of 7
## High Performance ML in R with H2O

# H2O Components

**H2O Cluster**

- Multi-node cluster with shared memory model.
- All computations in memory.
- Each node sees only some rows of the data.
- No limit on cluster size.

**Distributed Key Value Store**

- Objects in the H2O cluster such as data frames, models and results are all referenced by key.
- Any node in the cluster can access any object in the cluster by key.

**H2O Frame**

- Distributed data frames (collection of vectors).
- Columns are distributed (across nodes) arrays.
- Each node must be able to see the entire dataset (achieved using HDFS, S3, or multiple copies of the data if it is a CSV file).

# Distributed K/V Store

**Peer-to-Peer**

- The H2O K/V Store is a classic peer-to-peer distributed hash table.
- There is no “name-node” nor central key dictionary.

**Pseudo-Random Hash**

- Each key has a home-node, but the homes are picked pseudo-randomly per-key.
- This allows us to force keys to “home” to different nodes (usually for load-balance reasons).

**Key’s Home Node**

- A key's “home” is solely responsible for breaking ties in racing writes and is the “source of truth.”
- Keys can be cached anywhere, and both reads & writes can be cached (although a write is not complete until it reaches “home”.)

# Data in H2O

**Highly Compressed**

- We read data fully parallelized from: HDFS, NFS, Amazon S3, URLs, URIs, CSV, SVMLight.
- Data is highly compressed (about 2-4 times smaller than gzip).

**Speed**

- Memory bound, not CPU bound.
- If data accessed linearly, as fast as C or Fortran.
- Speed = data volume / memory bandwidth
- ~50GB / sec (varies by hardware).

**Data Shape**

- Table width: <1k fast, <10k works, <100k slow
- Table length: Limited only by memory
- We have tested 10's of billions of rows (TBs)

# Distributed H2O Frame

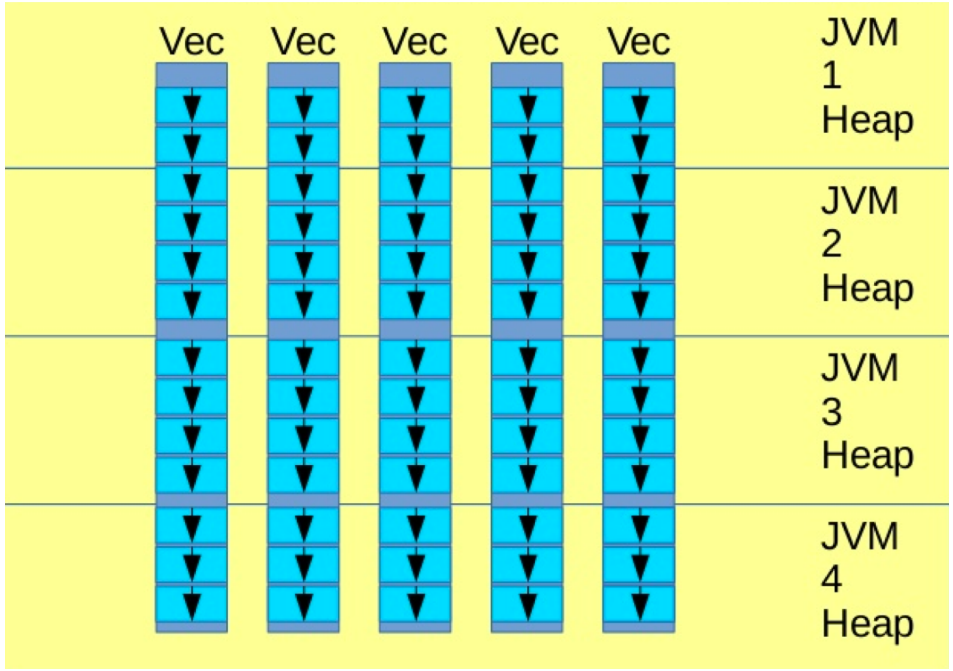



Diagram of distributed arrays. An “H2O Frame” is a collection of distributed arrays.

# Communication in H2O

**Network Communication**

- H2O requires network communication to JVMs in unrelated process or machine memory spaces.
- That network communication can be fast or slow, or may drop packets & sockets (even TCP can silently fail), and may need to be retried.

**Reliable RPC**

- H2O implements a reliable RPC which retries failed communications at the RPC level.
- We can pull cables from a running cluster, and plug them back in, and the cluster will recover.

**Optimizations**

- Message data is compressed in a variety of ways (because CPU is cheaper than network).
- Short messages are sent via 1 or 2 UDP packets; larger message use TCP for congestion control.

# Data Processing in H2O

**Map Reduce**

- Map/Reduce is a nice way to write blatantly parallel code (although not the only way), and we support a particularly fast and efficient flavor.
- Distributed fork/join and parallel map: within each node, classic fork / join

**Group By**

- We have a GroupBy operator running at scale (called ddply in the R community).
- GroupBy can handle millions of groups on billions of rows, and runs Map/Reduce tasks on the group members.

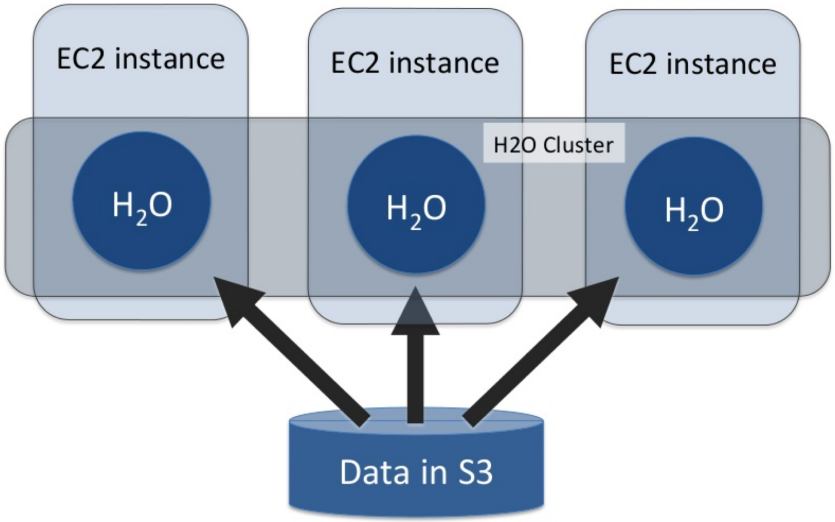
**Ease of Use**

- H2O has overloaded all the basic data frame manipulation functions in R and Python.
- Tasks such as imputation and one-hot encoding of categoricals is performed inside the algorithms.

# H2O on Amazon

## Part 3 of 7
## High Performance ML in R with H2O

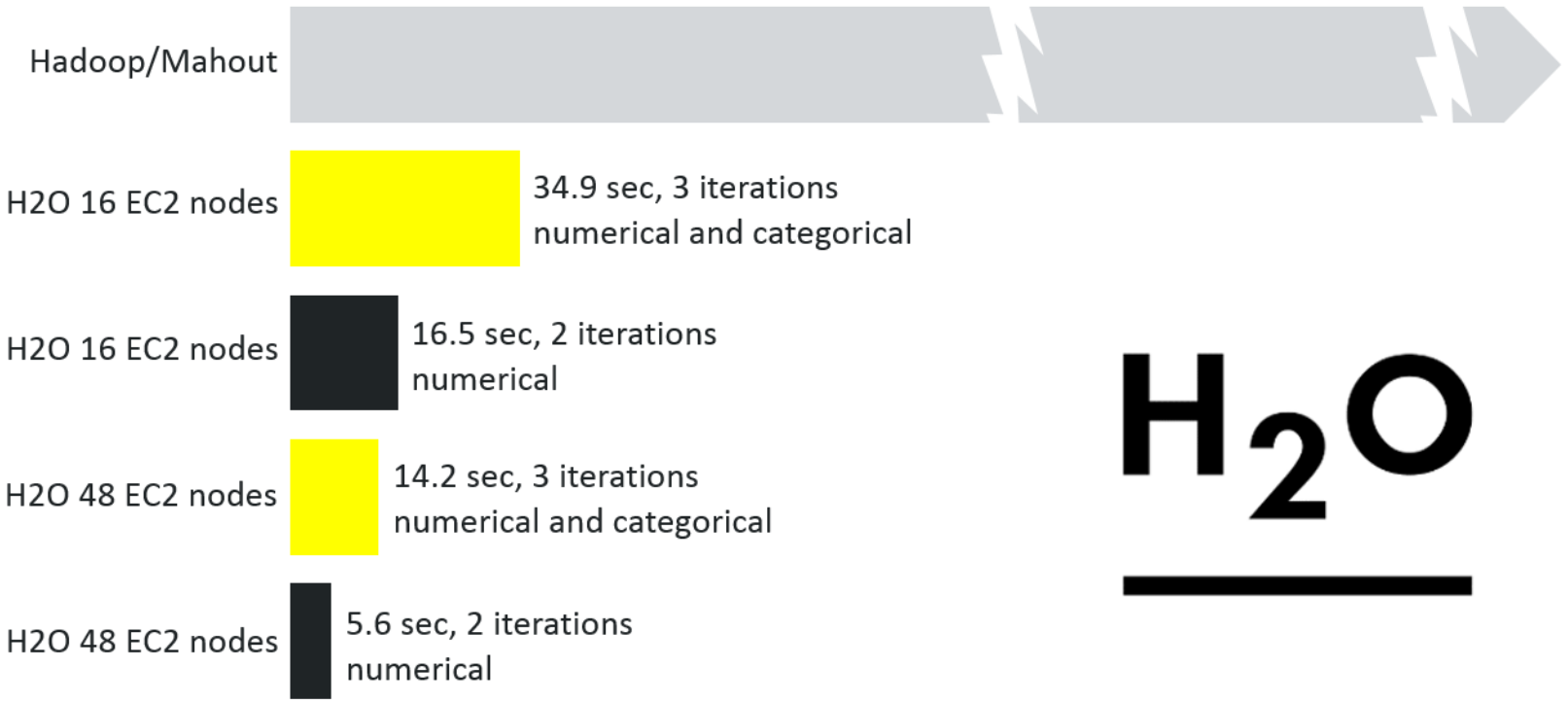
# H2O on Amazon EC2




H2O can easily be deployed on an Amazon EC2 cluster.
The GitHub repository contains example scripts that
help to automate the cluster deployment.

# H2O Billion Row Machine Learning Benchmark

## GLM Logistic Regression




$H_2O$

Compute Hardware: AWS EC2 c3.2xlarge - 8 cores and 15 GB per node, 1 GbE interconnect

Airline Dataset 1987-2013, 42 GB CSV, 1 billion rows, 12 input columns, 1 outcome column
9 numerical features, 3 categorical features with cardinalities 30, 376 and 380

# NERSC Supercomputers

## COMPUTATIONAL SYSTEMS

### NERSC Computational Systems

| System Name | System Type | CPU | | | | Computational Pool | | | | | Node Interconnect | Scratch Disk | Avg. Power (KW) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Type | Speed (GHz) | Nodes | SMP Size | Total Cores | Flops per Core (Gflops/sec) | Peak Performance (Tflops/sec) | Aggregate Memory | Avg. Memory/core | | | |
| **Edison** | Cray XC30 | Intel Ivy Bridge | 2.4 | 5,576 | 24 | 133,824 | 19.2 | 2569.4 | 357 TB | 2.67 | Aries | 7.56 PB (local) + 3.9 PB (global) | 1,600 |
| **Hopper** | Cray XE6 | Opteron | 2.1 | 6,384 | 24 | 153,216 | 8.4 | 1287.0 | 211.5 TB | 1.41 GB | Gemini | 2.2 PB (local) + 3.9 PB (global) | 2,200 |
| **PDSF**[1)] | Linux Cluster | Opteron, Xeon | 2.0, 2.27, 2.33, 2.67 | 232 | 8, 12, 16 | 2,632 | 8.0, 9.08, 9.32, 10.68 | 17.6 | 9.5 TB | 4 GB | Ethernet / InfiniBand | 34.9 TB for batch nodes and 184 GB for interactive nodes | - |
| **Genpool**[2)] | Various vendor systems | Nehalem, Opteron | 2.27, 2.67 | 547 | 8, 24, 32, 80 | 4,680 | 9.08, 10.68 | 42.8 | 33.7 TB | 7.36 GB | Ethernet | 3.9 PB (global) | - |

1) PDSF is a special-use system hosted by NERSC for the High Energy Physics and Nuclear Science community.

Edison is ranked 34 and Hopper is 62 on TOP500.
We hope to be running H2O at NERSC soon… :-)

# H2O in R

## Part 4 of 7
## High Performance ML in R with H2O

H2O.ai
Machine Intelligence

# “h2o” R package on CRAN

**Requirements**

- The only requirement to run the “h2o” R package is R >=3.1.0 and Java 7 or later.
- Tested on many versions of Linux, OS X and Windows.

**Installation**

- The easiest way to install the “h2o” R package is to install directly from CRAN.
- Latest version: `http://h2o.ai/download`

**Design**

- No computation is ever performed in R.
- All computations are performed (in highly optimized Java code) in the H2O cluster and initiated by REST calls from R.

# Start H2O Cluster from R

```
> library(h2o)
> localH2O <- h2o.init(nthreads = -1, max_mem_size = "8G")

H2O is not running yet, starting it now...

Note:  In case of errors look at the following log files:
    /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T//RtmpAXY9gj/h2o_me_started_from_r.out
    /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T//RtmpAXY9gj/h2o_me_started_from_r.err

java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)

.Successfully connected to http://127.0.0.1:54321/

R is connected to the H2O cluster:
    H2O cluster uptime:         1 seconds 96 milliseconds
    H2O cluster version:        3.3.0.99999
    H2O cluster name:           H2O_started_from_R_me_kfo618
    H2O cluster total nodes:    1
    H2O cluster total memory:   7.11 GB
    H2O cluster total cores:    8
    H2O cluster allowed cores:  8
    H2O cluster healthy:        TRUE

>
```

# H2O in R: Load Data

**Example**

```
library(h2o)  # First install from CRAN
localH2O <- h2o.init()  # Initialize the H2O cluster

# Data directly into H2O cluster (avoids R)
train <- h2o.importFile(path = "train.csv")

# Data into H2O from R data.frame
train <- as.h2o(my_df)
```

R code example: Load data

# H2O in R: Train & Test

**Example**

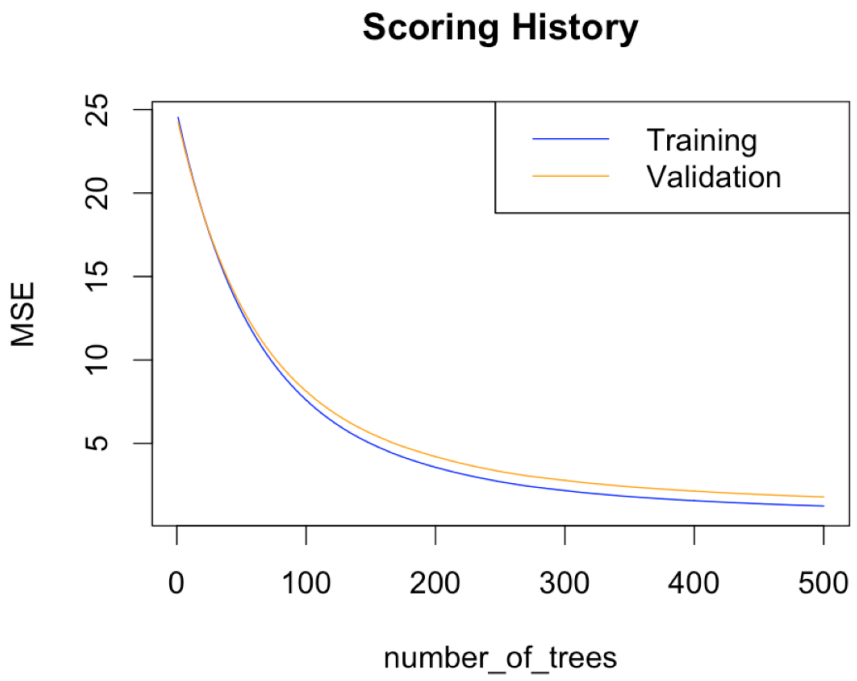
```
y <- "Class"
x <- setdiff(names(train), y)

fit <- h2o.gbm(x = x, y = y, training_frame = train)

pred <- h2o.predict(fit, test)
```

R code example: Train and Test a GBM

# H2O in R: Plotting




`plot(fit)` plots scoring history over time.

# H2O in R: Grid Search

**Example**

```
hidden_opt <- list(c(200,200), c(100,300,100), c(500,500))
l1_opt <- c(1e-5,1e-7)
hyper_params <- list(hidden = hidden_opt, l1 = l1_opt)

model_grid <- h2o.grid("deeplearning",
        hyper_params = hyper_params,
        x = x, y = y,
        training_frame = train,
        validation_frame = test)
```

R code example: Execute a DL Grid Search

# Live H2O Demo!

https://gist.github.com/ledell

Install H2O (stable): install_h2o_slater.R
Demo: h2o_higgs_simple_demo.R

# H2O Ensemble

## Part 5 of 7
## High Performance ML in R with H2O

# What is Ensemble Learning?

## What it is:

- "Ensemble methods use multiple learning algorithms to obtain better predictive performance that could be obtained from any of the constituent learning algorithms." (Wikipedia, 2015)
- Random Forests and Gradient Boosting Machines (GBM) are both ensembles of decision trees.
- Stacking, or Super Learning, is technique for combining various learners into a single, powerful learner using a second-level metalearning algorithm.

## What it's not:

Ensembles typically achieve superior model performance over singular methods. However, this comes at a price — computation time.

# H2O Ensemble Overview

**ML Tasks**

- Regression
- Binary Classification / Ranking
- Coming soon: Support for multi-class

**Super Learner**

- H2O Ensemble implements the Super Learner algorithm.
- The Super Learner algorithm finds the optimal (based on defined loss) combination of a collection of base learning algorithms.

**Why ensembles?**

- When a single algorithm does not approximate the true prediction function well.
- When model performance is the most important factor (over training speed and interpretability).

# Super Learner: The setup

1. Define a base learner library of $L$ learners, $\Psi^1, ..., \Psi^L$.
2. Specify a metalearning method, $\Phi$.
3. Partition the training observations into $V$ folds.

# Super Learner: The Algorithm

1. Generate a matrix $Z$, of dimension $n \times L$, of cross-validated predictions as follows: During cross-validation, we obtain fits, $\hat{\Psi}^l_{-v}$, defined as fitting $\Psi^l$ on the observations that are not in fold $v$. Predictions are then generated for the observations in the $v^{th}$ fold.
2. Find the optimal combination of subset-specific fits according to a user-specified metalearner algorithm, $\hat{\Phi}$, with a new design matrix, $Z$.
3. Fit $L$ models (one for each base learner) on the original training set, $X$, and save the $L$ individual model fit objects along with $\hat{\Phi}$. This ensemble model can be used to generate predictions on new data.

# H2O Ensemble R Interface

**Example**

```
library(h2oEnsemble)  #Install from GitHub

learner <- c("h2o.randomForest.1",
             "h2o.deeplearning.1",
             "h2o.deeplearning.2")

metalearner <- "h2o.glm.wrapper"

family <- "binomial"
```

R code example: Set up an H2O Ensemble

# H2O Ensemble R Interface

**Example**

```
fit <- h2o.ensemble(x = x, y = y, training_frame = train,
                    family = family,
                    learner = learner,
                    metalearner = metalearner)

pred <- predict(fit, test)
```

R code example: Train and Test an Ensemble

# Live H2O Demo!

https://gist.github.com/ledell

Install H2O Ensemble: install_h2oEnsemble.R
Demo: lending_club_bad_loans_ensemble.R

# Sparkling Water

## Part 6 of 7
## High Performance ML in R with H2O

# Apache Spark and SparkR

**Apache Spark**

- Apache Spark is an open source in-memory processing engine built around speed.
- It was originally developed at UC Berkeley in 2009.

**Spark for HPC**

- Spark is commonly used on commodity clusters (such as Amazon EC2).
- CRAY has been working with Spark community to optimize Spark for CRAY supercomputers.

**SparkR**

- Spark is written in Scala, but APIs exist for Python and R.
- “SparkR” is the R API and has been part of Spark since Spark 1.4 (June, 2015).

# H2O vs SparkR

| | |
|---|---|
| Architecture | • Major difference is that SparkR creates a collection of slave R instances.<br>• H2O uses a single R session and communicates to the H2O Java cluster via REST calls. |
| Machine Learning Algorithms | • In Spark 1.5 (latest release), only GLM is accessible in R via SparkR.<br>• All H2O algorithms are available via R. |
| Distributed Frames | • Both H2O and Spark use distributed data frames.<br>• SparkR is most useful for data processing on distributed data frames. |

# H2O Sparkling Water

**Spark Integration**

- Sparkling Water is transparent integration of H2O into the Spark ecosystem.
- H2O runs inside the Spark Executor JVM.

**Benefits**

- Provides advanced machine learning algorithms to Spark workflows.
- Sophisticated alternative to the default MLlib library in Spark.

**Sparkling Shell**

- Sparkling Shell is just a standard Spark shell with additional Sparkling Water classes
- `export MASTER="local-cluster[3,2,1024]"`
- `spark-shell –jars sparkling-water.jar`

# H2O in Action

## Part 7 of 7
## High Performance ML in R with H2O

# Actual Customer Use Cases

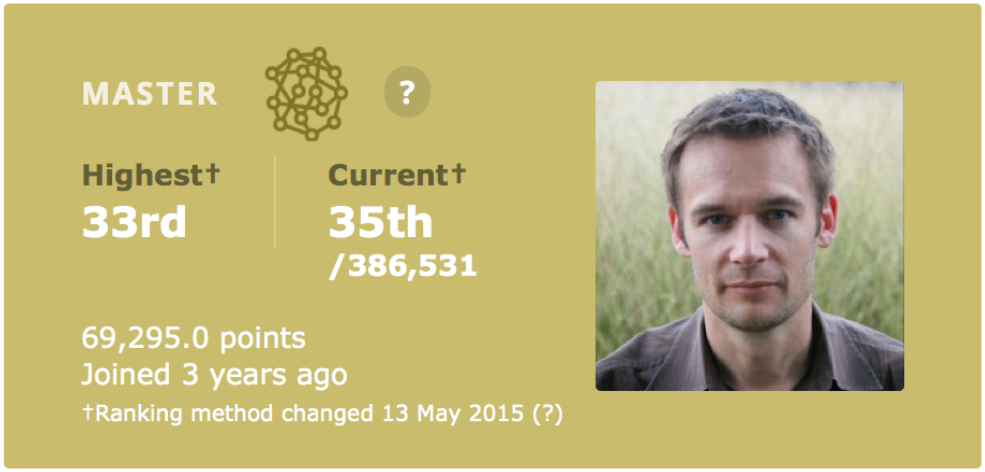

Ad optimization (200% CPA lift with H2O)

Fraud detection — 11% higher accuracy with H2O Deep Learning (saves millions of dollars)

Propensity to Buy model factory (60,000 models, 15x faster with H2O)

# H2O on kaggle

- H2O starter scripts available on Kaggle
- H2O is used in many competitions on Kaggle
- Mark Landry, H2O Data Scientist and Competitive Kaggler

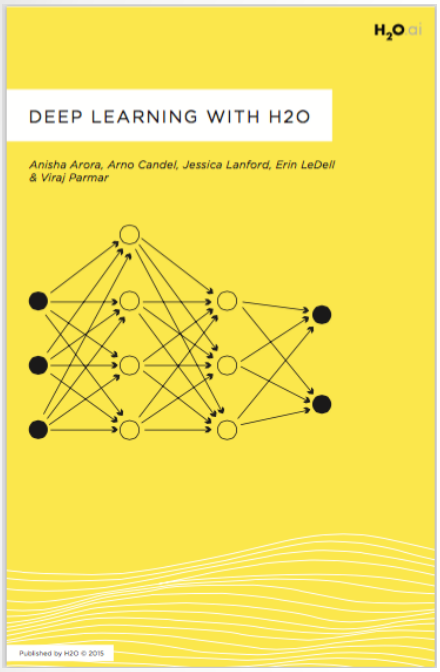


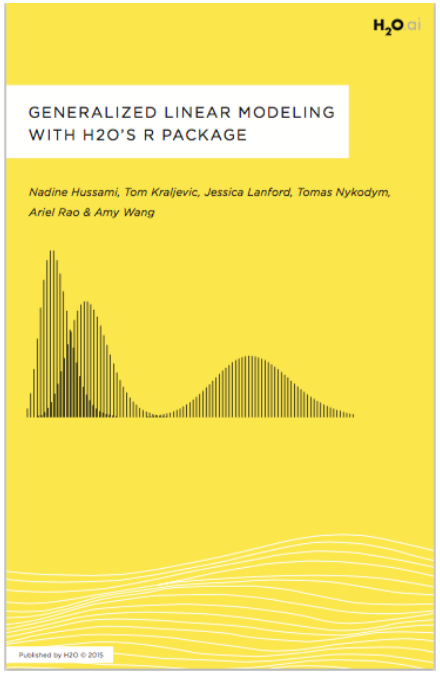

https://www.kaggle.com/mlandry

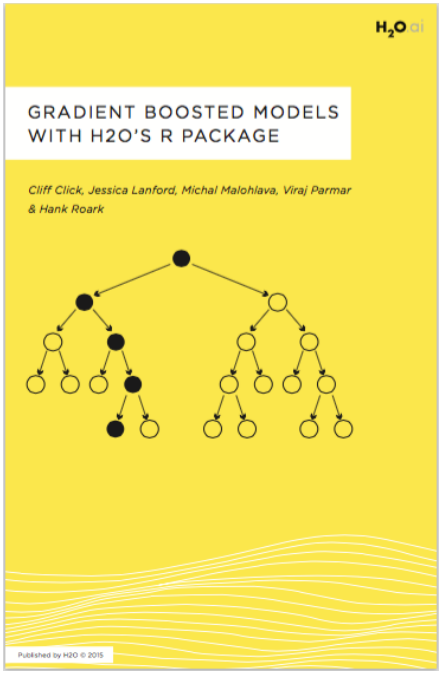
# Where to learn more?

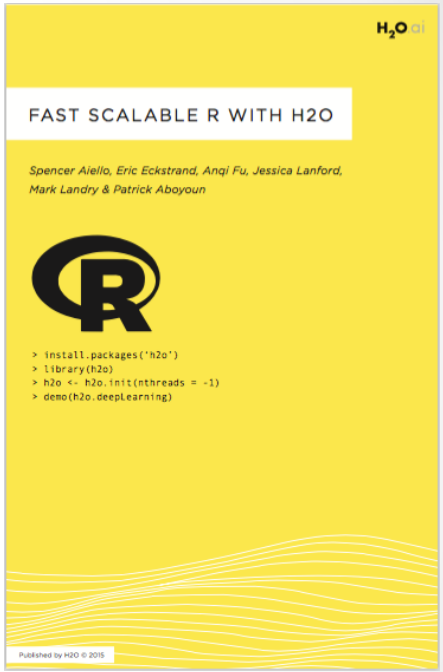
- H2O Online Training (free): http://learn.h2o.ai
- H2O Slidedecks: http://www.slideshare.net/0xdata
- H2O Video Presentations: https://www.youtube.com/user/0xdata
- H2O Community Events & Meetups: http://h2o.ai/events
- Machine Learning & Data Science courses: http://coursebuffet.com

# H2O Booklets




https://github.com/h2oai/h2o-3/tree/master/h2o-docs/src/booklets/v2_2015/PDFs/online











**H2O WORLD**

**35 Speakers**
**Training**
**2-Full Days**

**Nov. 9 - 11**

**http://world.h2o.ai**

20% Discount code:
h2ocommunity

**Customers • Community • Evangelists**

# ありがとう

@ledell on Twitter, GitHub
erin@h2o.ai

http://www.stat.berkeley.edu/~ledell