RStudio®

Connecting R to the Good Stuff!

Joseph B. Rickert
RStudio R Community Ambassador

THE NATURE OF R

"One of the attractions of R has always been the ability to compute an interesting result quickly.

"A key motivation for the original S remains important now: to give easy access to the best computations for understanding data."

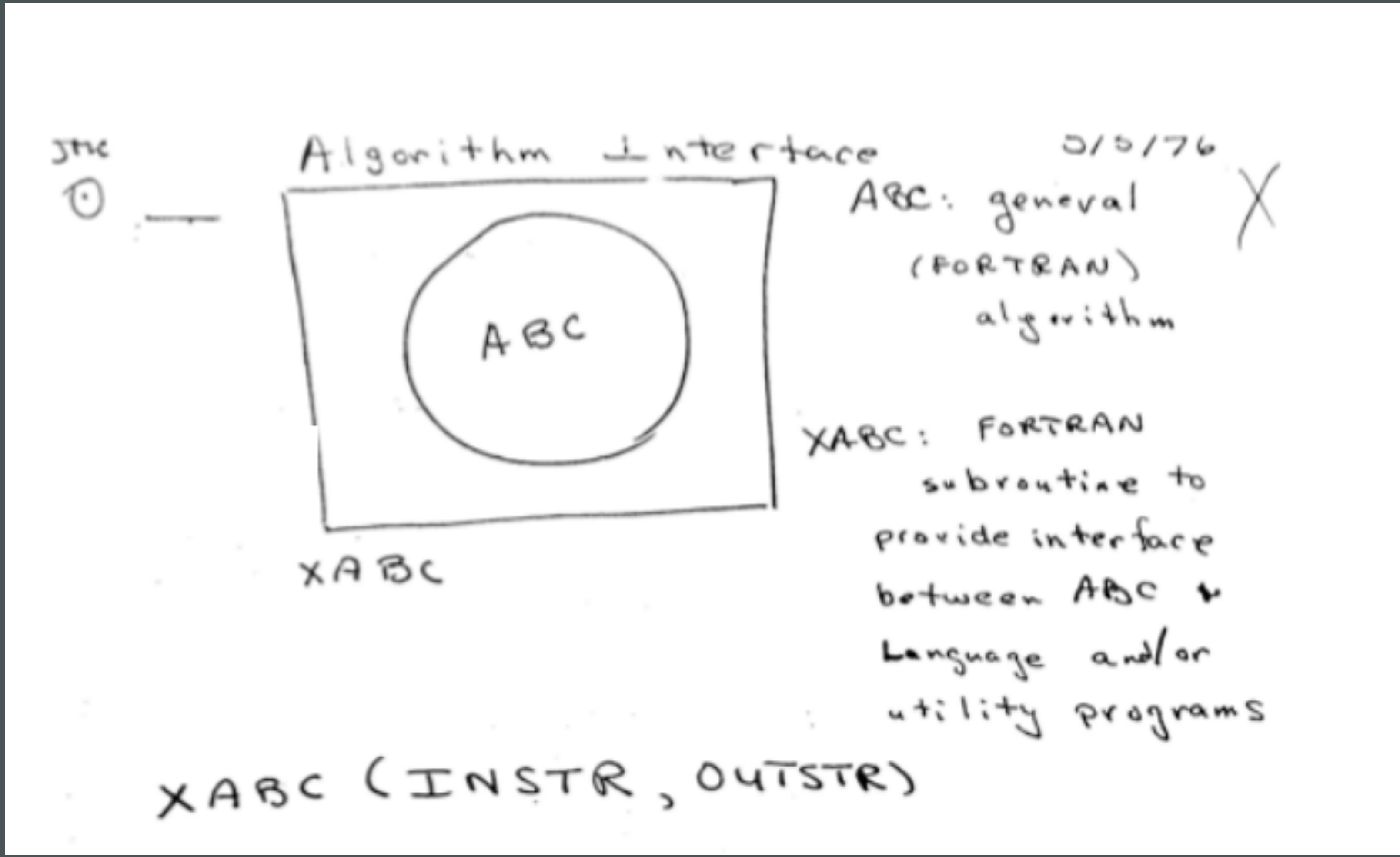
John Chambers: Extending R, CRC Press

S WAS CONCEIVED AS AN INTERFACE

The top half John Chambers' famous diagram from that afternoon in May 1976 indicates their intention to design a software interface so that one could call an arbitrary Fortran subroutine, ABC, by wrapping it in some simplified calling syntax: XABC( ).

The main idea was to bring the best computational facilities to the people doing the analysis. As John phrased it: "combine serious computational challenges with convenience".

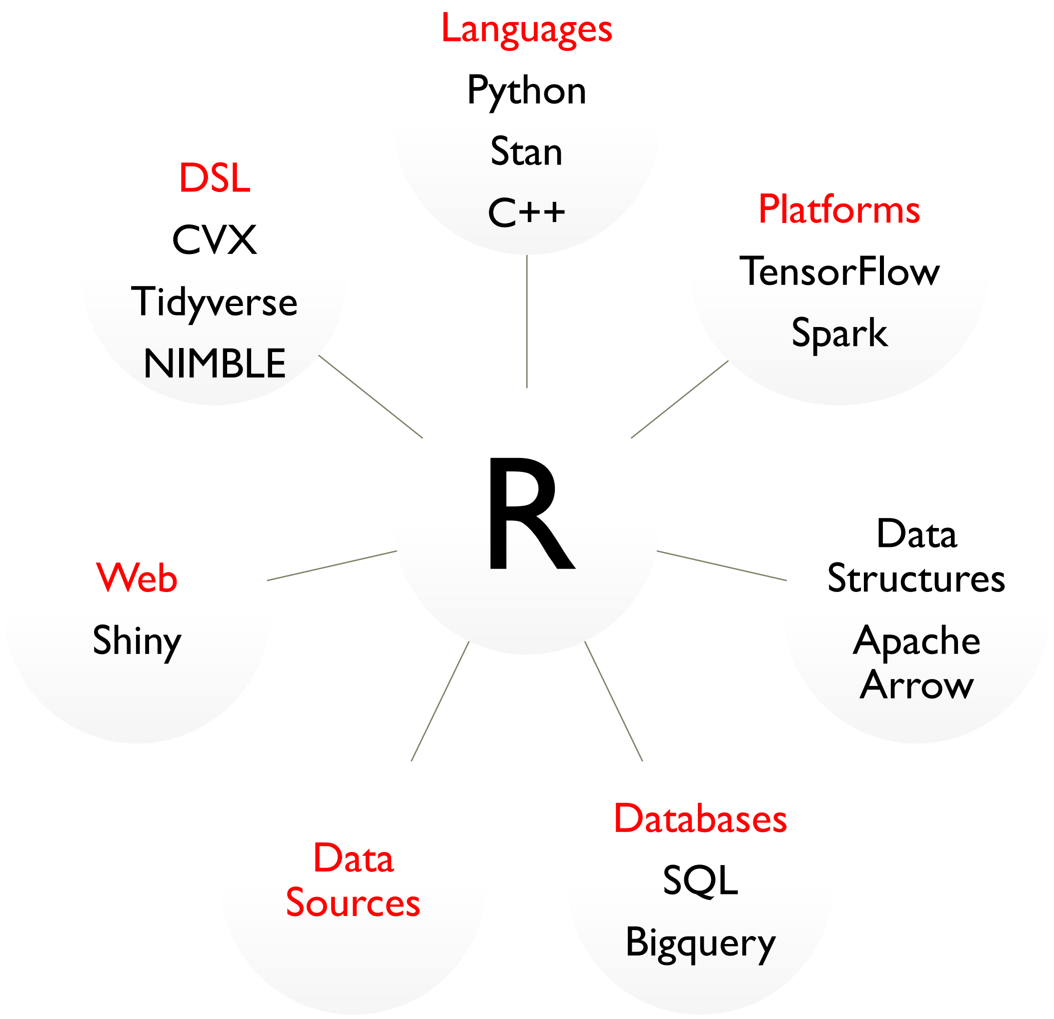
Source - http://bit.ly/1y4vwwL



CONNECTING TO THE GOOD STUFF

Methods of Connecting

- Simple Import / API
- Database connections
- DSL
- Language Interfaces
- Sharing Data structures

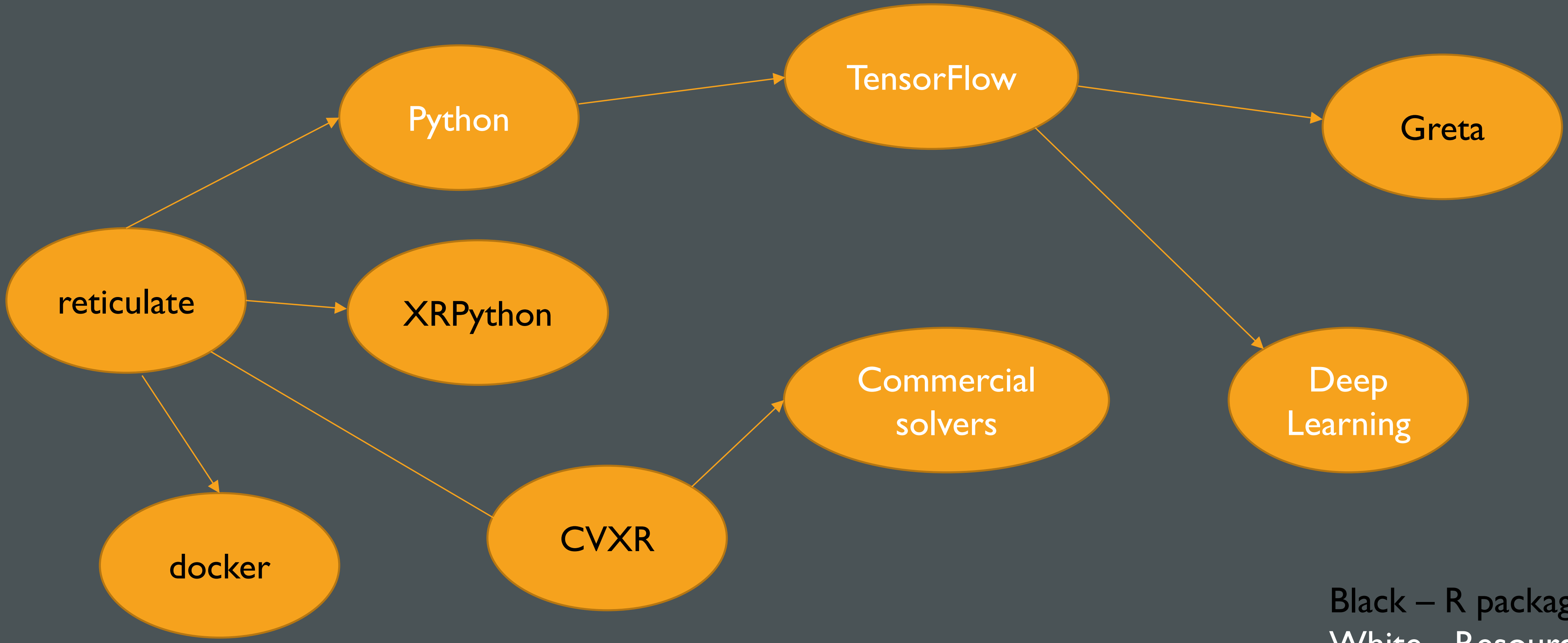


RETICULATE PACKAGE:
R INTERFACE TO PYTHON

reticulate provides a comprehensive set of tools for interoperability between Python and R:

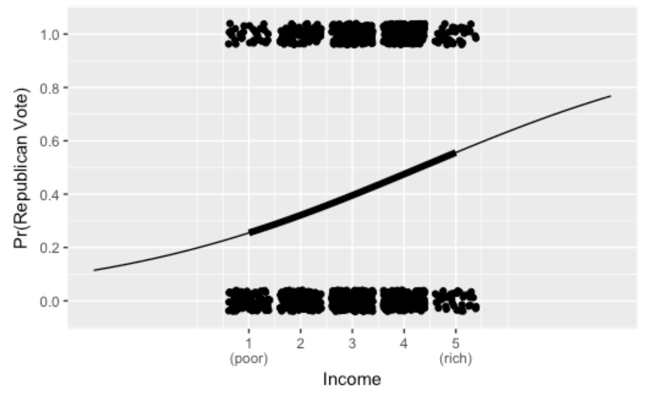
- Calling Python from R:
  using R Markdown
  by sourcing Python scripts
  by importing Python modules
  using Python interactively within an R session.
- Translation between R and Python objects (e.g., between R and Pandas data frames, or between R matrices and NumPy arrays).
- Bind to different versions of Python including virtual environments and Conda environments.

LEVERAGING RETICULATE



Black – R package
White - Resource

- A probabilistic programming language for Bayesian inference and optimization
- Stan programs compute the log-posterior density
- Code is compiled and run with data
- Result is a set of posterior simulations of the model
- Uses the no-U-turn sampler, an adaptive variant of Hamiltonian Monte Carlo
- Run from R with rstan package



Stan Logistic Regression

```
# The Stan model in the file nes_logit.stan
data {
  int<lower=0> N;
  vector[N] income;
  int<lower=0,upper=1> vote[N];
}
parameters {vector[2] beta;}
model {vote ~ bernoulli_logit(beta[1] + beta[2] * income);}
```

Set up and run the model.

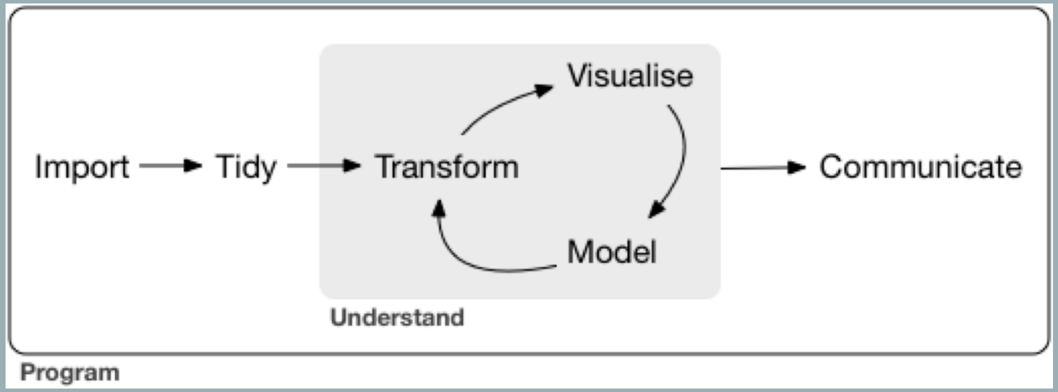
```
library(rstan)
library(ggplot2)

### Data
source("nes1992_vote.data.R", echo = TRUE)

### Logistic model: vote ~ income
data.list <- c("N", "vote", "income")
nes_logit.sf <- stan(file='nes_logit.stan', data=data.list,iter=1000, chains=2)
```

Show the results

```
print(nes_logit.sf, pars = c("beta", "lp__"))
```

```
Inference for Stan model: nes_logit.
2 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1000.

           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
beta[1]   -1.40    0.01 0.18   -1.74   -1.51   -1.39   -1.28   -1.04   298    1
beta[2]    0.32    0.00 0.05    0.23    0.29    0.32    0.36    0.44   291    1
lp__    -779.45    0.06 1.00 -782.24 -779.80 -779.15 -778.75 -778.47   271    1

Samples were drawn using NUTS(diag_e) at Fri Jun 22 15:12:27 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

TIDYVERSE: A DSL FOR DATA SCIENCE



The Tidyverse is

- A coherent system of packages for data manipulation, exploration and visualization
- Share a common design philosophy
- Intended to make statisticians and data scientists more productive by guiding them through workflows
- Facilitate communication
- Result in reproducible work products

CVXR IS A DSL FOR FORMULATING AND SOLVING CONVEX OPTIMIZATION PROBLEMS

CVXR provides an object-oriented modeling language for convex optimization. It allows the user to formulate convex optimization problems in a natural mathematical syntax rather than the restrictive form required by most solvers.

The general convex optimization problem is of the form

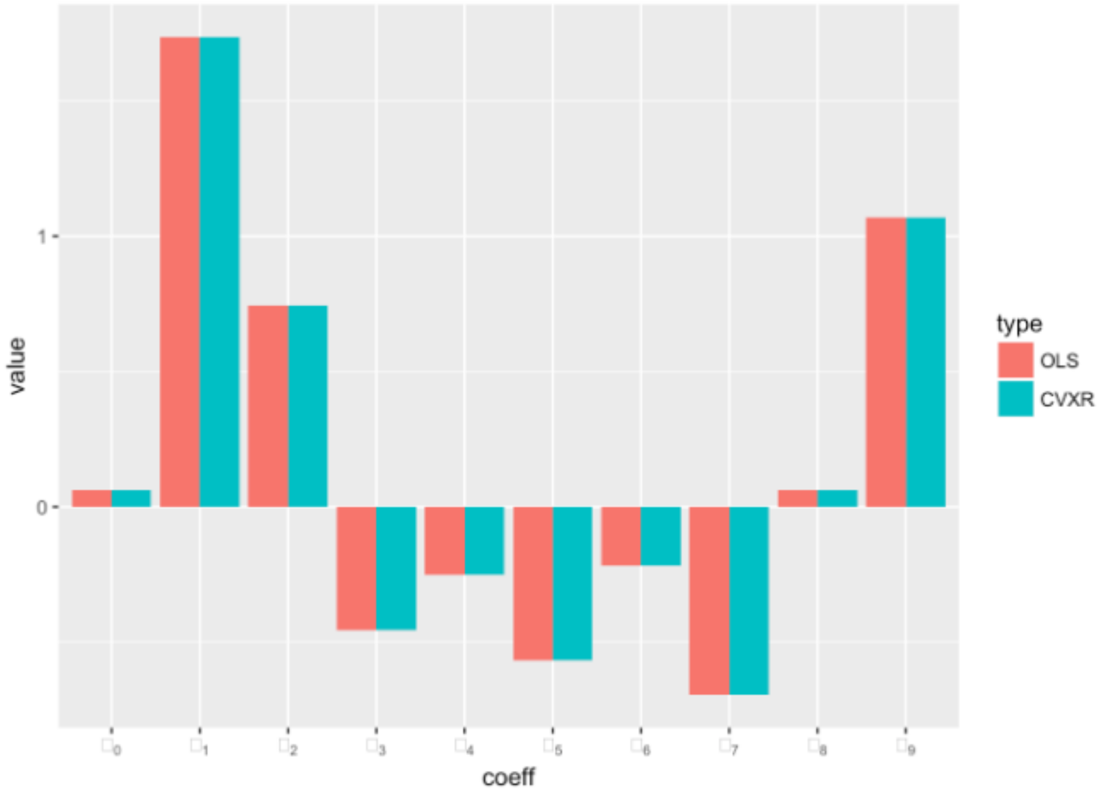
$$\begin{array}{ll} \underset{v}{\text{minimize}} & f_0(v) \\ \text{subject to} & f_i(v) \leq 0, \quad i = 1, \ldots, M \\ & Av = b, \end{array}$$

where $v \in \mathbf{R}^n$ is our variable of interest, and $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$ are constants describing our linear equality constraints. The objective and inequality constraint functions $f_0, \ldots, f_M$ are convex, *i.e.*, they are functions $f_i : \mathbf{R}^n \to \mathbf{R}$ that satisfy

$$f_i(\theta u + (1-\theta)v) \leq \theta f_i(u) + (1-\theta) f_i(v)$$

for all $u, v \in \mathbf{R}^n$ and $\theta \in [0, 1]$. This class of problems arises in a variety of fields, including machine learning and statistics.

```
CVXR_solution3.1 <- solve(prob3.1)
lm_solution3.1 <- lm(y ~ 0 + X)
```

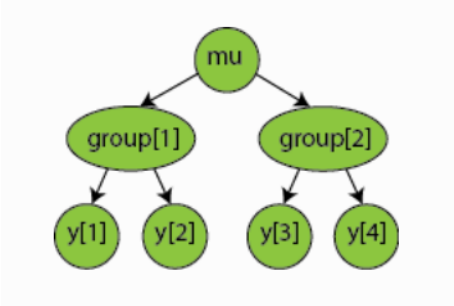


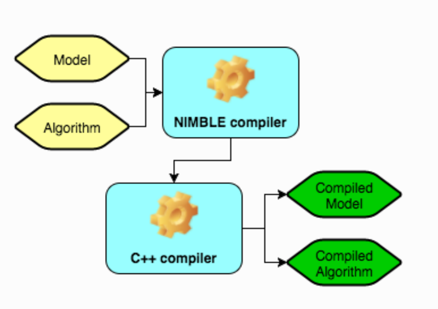
https://cvxr.rbind.io/

https://web.stanford.edu/~boyd/papers/pdf/cvxr_paper.pdf

NIMBLE
NUMERICAL INFERENCE FOR STATISTICAL MODELS USING BAYESIAN AND LIKELIHOOD ESTIMATION

- A DSL for Hierarchical Models
- Built on R
- Uses BUGs Language
- Separates Model from Algorithms
- Separates Setup from Model Execution
- Compiles R Code

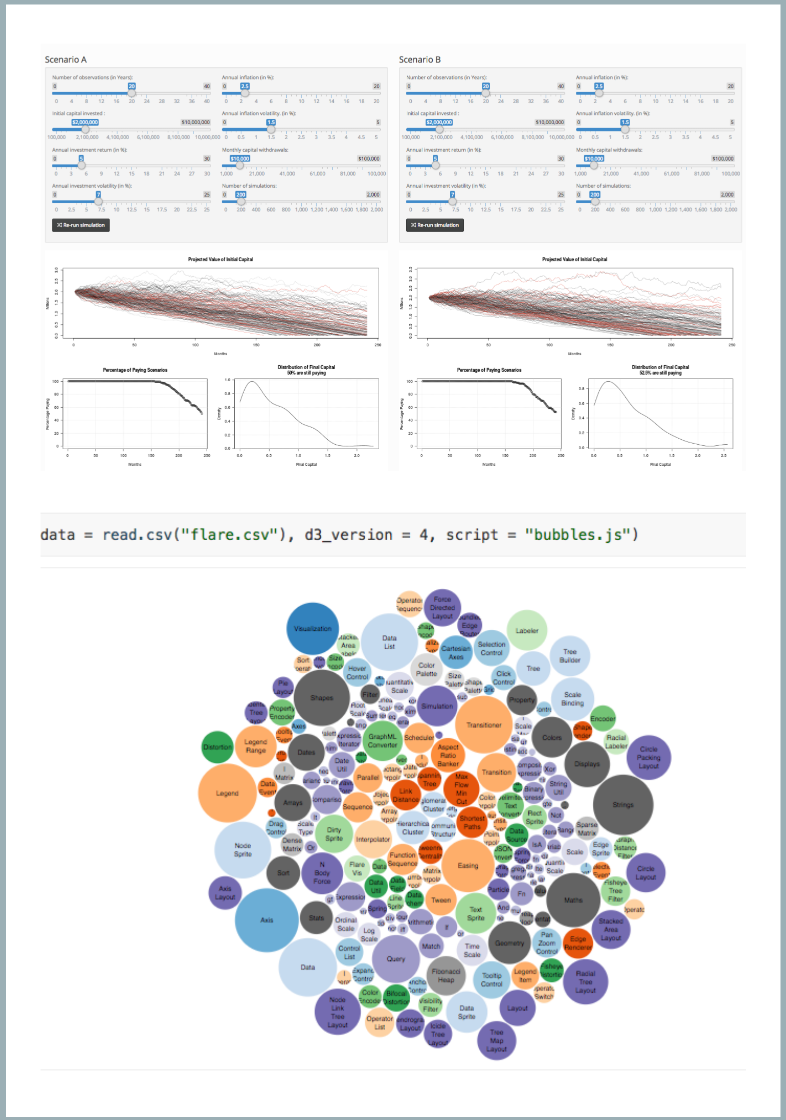





```r
## MCMC for logistic regression with random effects
## load the NIMBLE library
library(nimble)

## define the model
code <- nimbleCode({
    beta0 ~ dnorm(0, sd = 10000)
    beta1 ~ dnorm(0, sd = 10000)
    sigma_RE ~ dunif(0, 1000)
    for(i in 1:N) {
        beta2[i] ~ dnorm(0, sd = sigma_RE)
        logit(p[i]) <- beta0 + beta1 * x[i] + beta2[i]
        r[i] ~ dbin(p[i], n[i])
    }
})

## constants, data, and initial values
constants <- list(N = 10)

data <- list(
    r = c(10, 23, 23, 26, 17, 5, 53, 55, 32, 46),
    n = c(39, 62, 81, 51, 39, 6, 74, 72, 51, 79),
    x = c(0,  0,  0,  0,  0,  1, 1,  1,  1,  1)
)

inits <- list(beta0 = 0, beta1 = 0, sigma_RE = 1)

## create the model object
Rmodel <- nimbleModel(code=code, constants=constants, data=data, inits=inits, check = FALSE)
```

WEB CONNECTIVITY AND JAVASCRIPT VISUALIZATIONS

- Interactive Web Integration:
  - Shiny
- Web Scraping:
  - RCrawler
  - rvest
- JavaScript Visualizations
  - r2d3 - R Interface to D3 Visualizations
  - Htmlwidgets



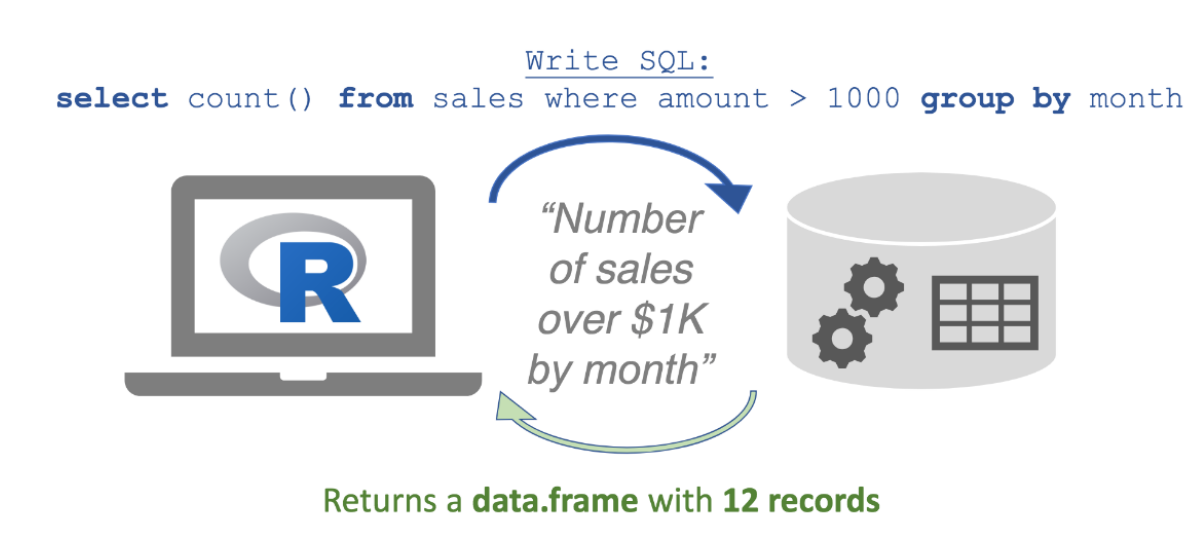
DATABASE CONNECTIVITY

The DBI package provides a common interface that allows dplyr to work with many different databases using the same code.

The R Consortium is is funding work to improve DBI (https://www.r-dbi.org/).

DBI is automatically installed with dbplyr.

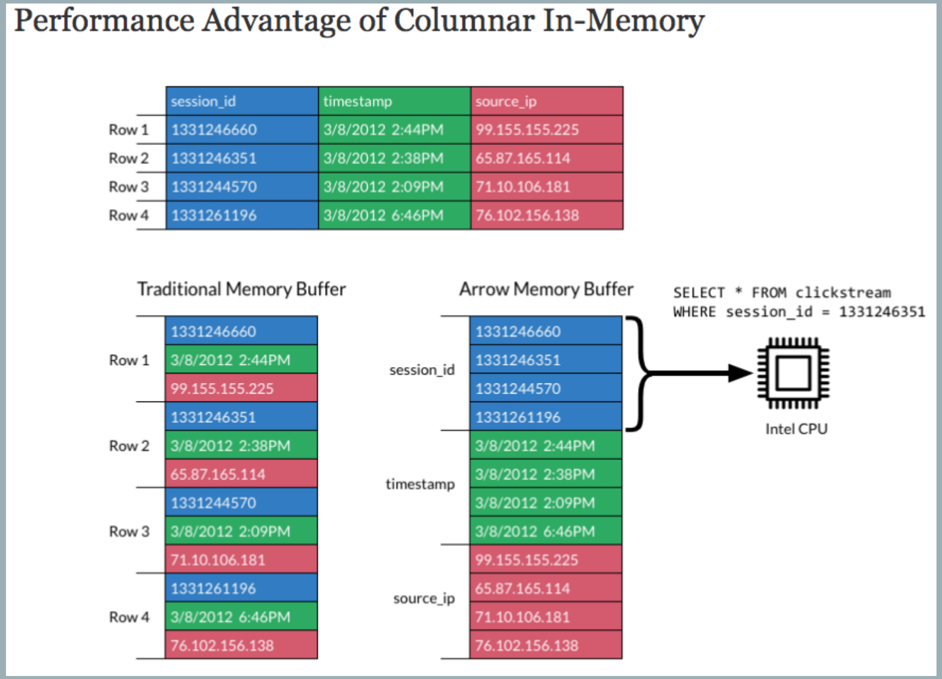
Analyze in database with dplyr

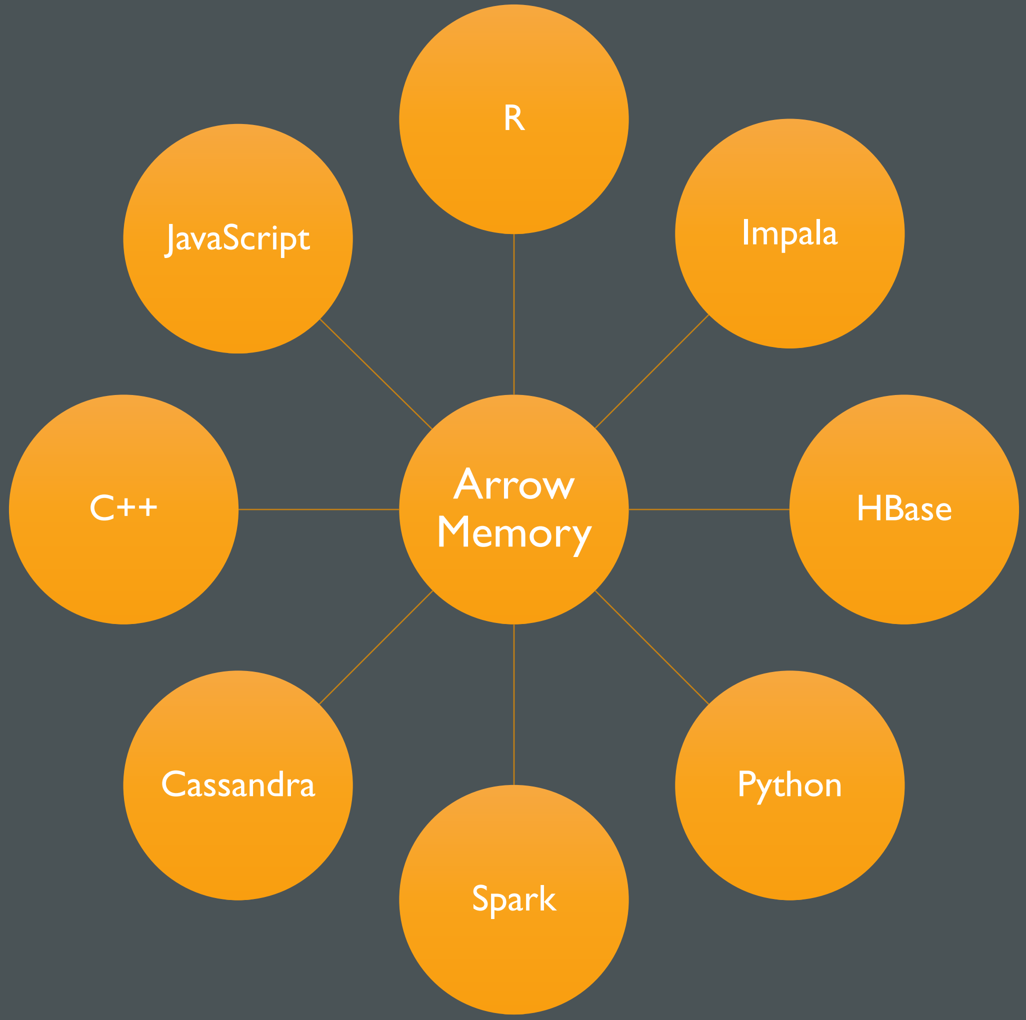


- MS SQL Server and Access
- Oracle
- Apache Hive and Impala
- Postgress SQL and MariaDB (MySql)
- Amazon Redshift
- Teradata
- Google Biqquery

URSA LABS
&
THE APACHE ARROW PROJECT

- Language-independent memory structures
- Columnar format for flat and hierarchical data
- Better performance on CPUs and GPUs





SPARKLYR: R INTERFACE FOR APACHE SPARK

```
SPARK_HOME=/opt/spark/spark-2.0.0-bin-hadoop2.6
```

To connect, pass the address of the master node to spark_connect, for example:

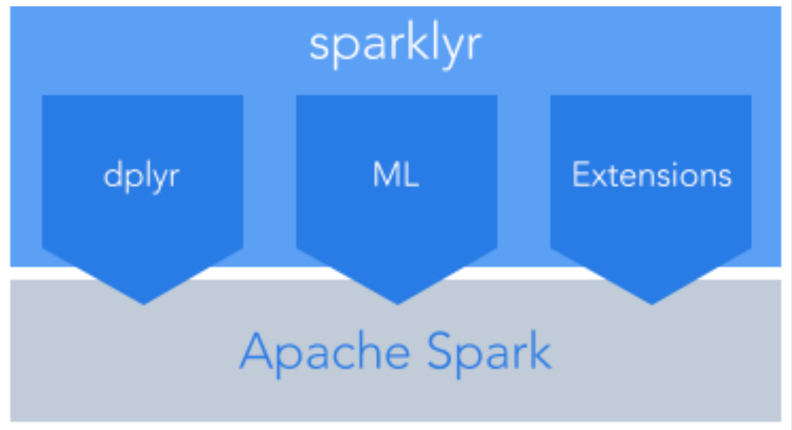
```
library(sparklyr)
sc <- spark_connect(master = "spark://local:7077")
```

For a Hadoop YARN cluster, you can connect using the YARN master, for example:

```
library(sparklyr)
sc <- spark_connect(master = "yarn-client")
```

If you are running on EC2 using the Spark EC2 deployment scripts then you can read the master from `/root/spark-ec2/cluster-url`, for example:

```
library(sparklyr)
cluster_url <- system('cat /root/spark-ec2/cluster-url', intern=TRUE)
sc <- spark_connect(master = cluster_url)
```



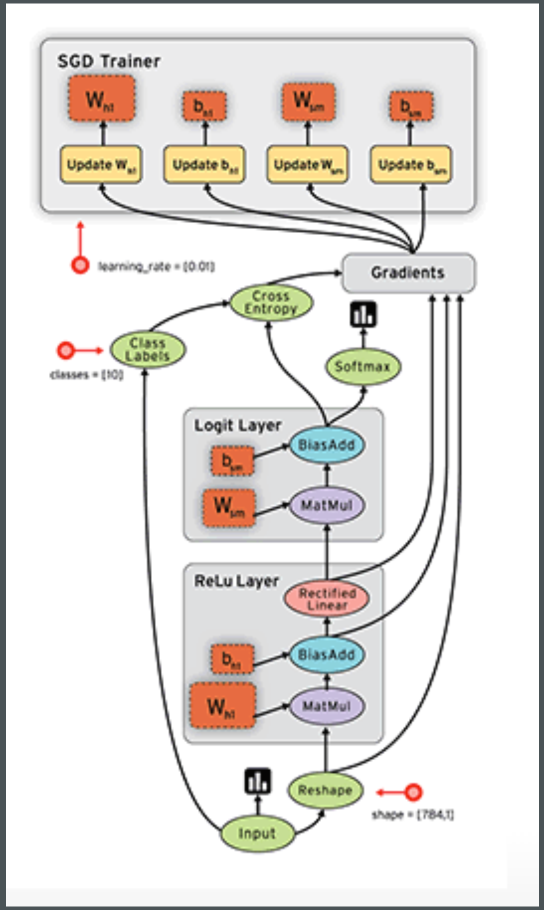
- Connect to Spark from R
- Dplyr backend
- Filter and aggregate Spark datasets
- Run Spark's MLlib models
- Create extensions the call the full Spark API

TENSORFLOW

A general purpose numerical computing library

- Developed by the Google Brain Team
- Open Source (Apache v2.0 license)
- Hardware independent:
  - CPU (via Eigen and BLAS)
  - GPU (via CUDA and cuDNN)
  - TPU (Tensor Processing Unit)
- Supports automatic differentiation
- Distributed Execution for large datasets

Tensors (arrays) flow through a dataflow graph where nodes represent units of computation.



Applications:

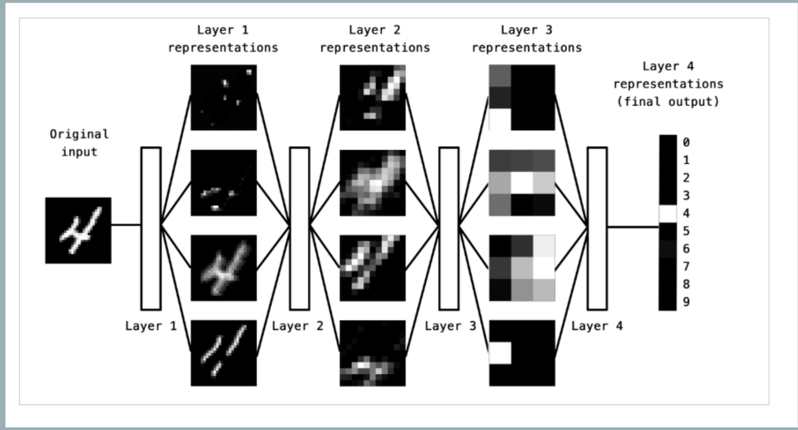
- Deep Learning
- Bayesian Modeling
- And More

R PACKAGES FOR TENSORFLOW

TensorFlow APIs

- keras—Interface for neural networks, with a focus on enabling fast experimentation.
- tfestimators— Implementations of common model types such as regressors and classifiers.
- tensorflow—Low-level interface to the TensorFlow computational graph.
- tfdatasets—Scalable input pipelines for TensorFlow models

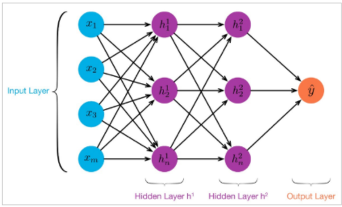
DEEP LEARNING

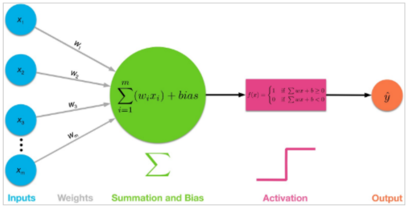


A machine learning technique for classification and prediction

"Learns" by transforming data through many layers of representations

Data transformation functions are parameterized by weights





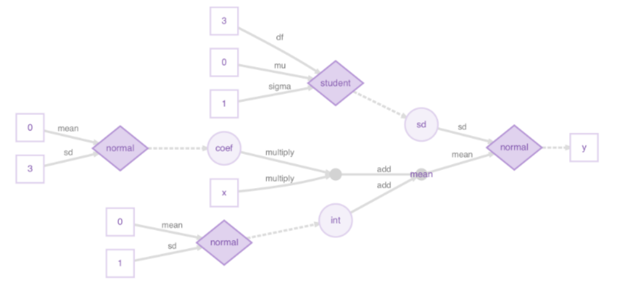
greta

simple and scalable statistical modelling in R

BAYESIAN MODELING

Write statistical models in R and fit them by MCMC on CPUs and GPUs, using Google TensorFlow

```r
library(greta)

# data
x <- as_data(iris$Petal.Length)
y <- as_data(iris$Sepal.Length)

# variables and priors
int = normal(0, 1)
coef = normal(0, 3)
sd = student(3, 0, 1, truncation = c(0, Inf))

# operations
mean <- int + coef * x

# likelihood
distribution(y) = normal(mean, sd)

# defining the model
m <- model(int, coef, sd)

# plotting
plot(m)

# sampling
draws <- mcmc(m, n_samples = 1000)
```



SOME LINKS

- Apache Arrow - https://arrow.apache.org/
- CVXR - https://cvxr.rbind.io/
- Database connectivity - https://db.rstudio.com/databases/
- Greta - https://greta-dev.github.io/greta/
- NIMBLE - https://r-nimble.org/
- Reticulate - https://rstudio.github.io/reticulate/
- R2d3 - https://github.com/rstudio/r2d3
- Shiny - https://shiny.rstudio.com/
- Sparklyr - http://spark.rstudio.com/
- TensorFlow - https://tensorflow.rstudio.com/
- Ursa Labs - https://ursalabs.org/

For code examples look here:
https://github.com/joseph-rickert/useR_2018

R CONTINUES TO REINVENT ITSELF

Thank you

Joseph.rickert@RStudio.com

@RStudioJoe

Connecting R to the Good Stuff

In his book, Extending R, John Chambers writes:

> One of the attractions of R has always been the ability to compute an interesting result quickly. A key motivation for the original S remains as important now: to give easy access to the best computations for understanding data.

R developers have taken the challenge implied in John's statement to heart, and have integrated R with some really "good stuff' while providing easy access that conforms to natural R workflows. Rcpp and Shiny, for example, are both spectacularly successful projects in which R developers expanded the reach of R by connecting to external resources.

In this talk, I will survey the ongoing work to connect R to "good stuff" such as the CVX optimization software, the Stan Bayesian engine, Spark, Keras and TensorFlow; and provide some code examples including using the sparlkyr package to run machine learning models on Spark and the keras package to run deep learning and other models on TensorFlow.