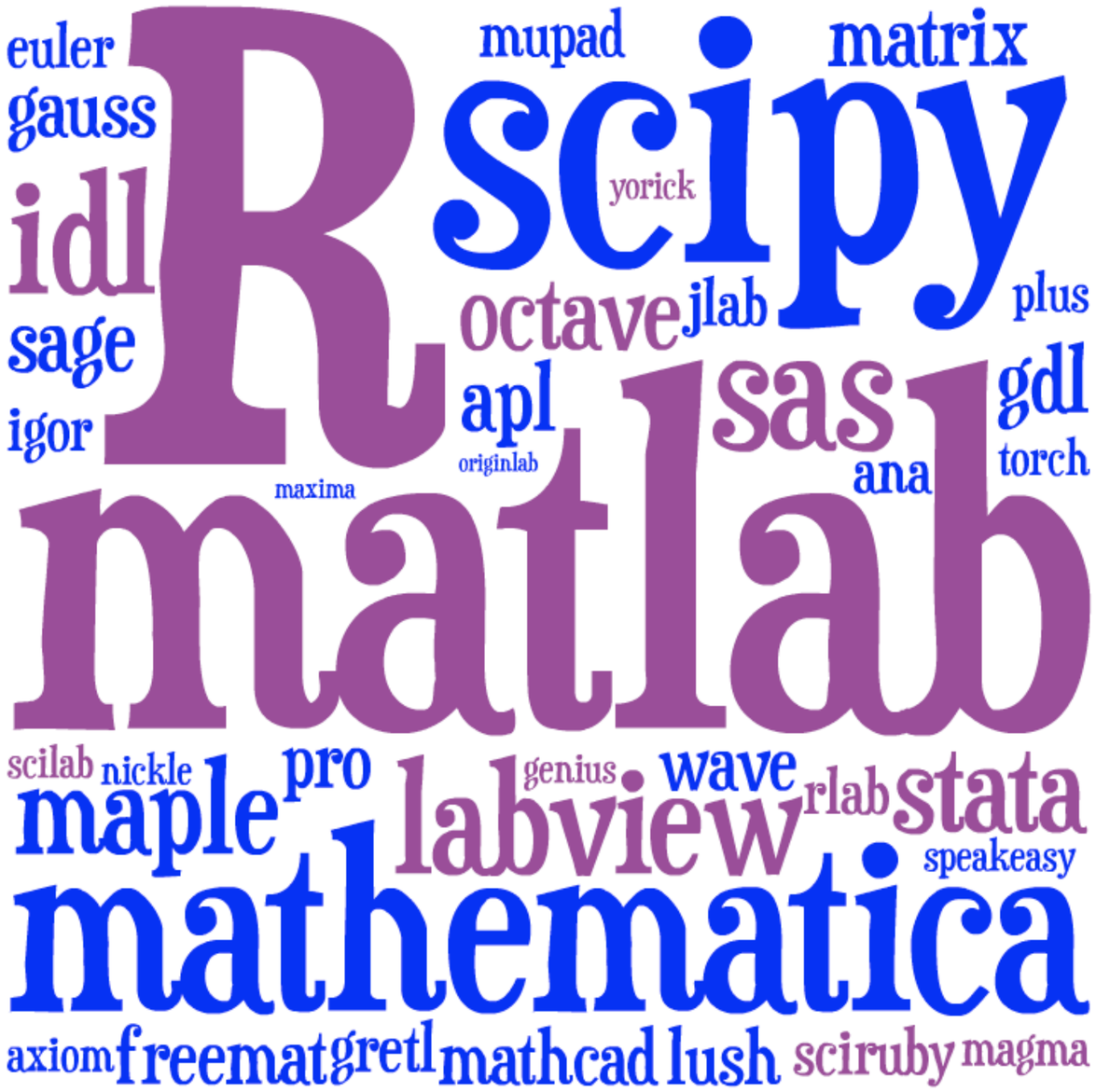
A Fast Dynamic Language for Technical Computing

Viral B. Shah

Jeff Bezanson, Stefan Karpinski, Alan Edelman,
and many others!

Prepared for Fifth Elephant
July 13, 2013

Why do we need one more?

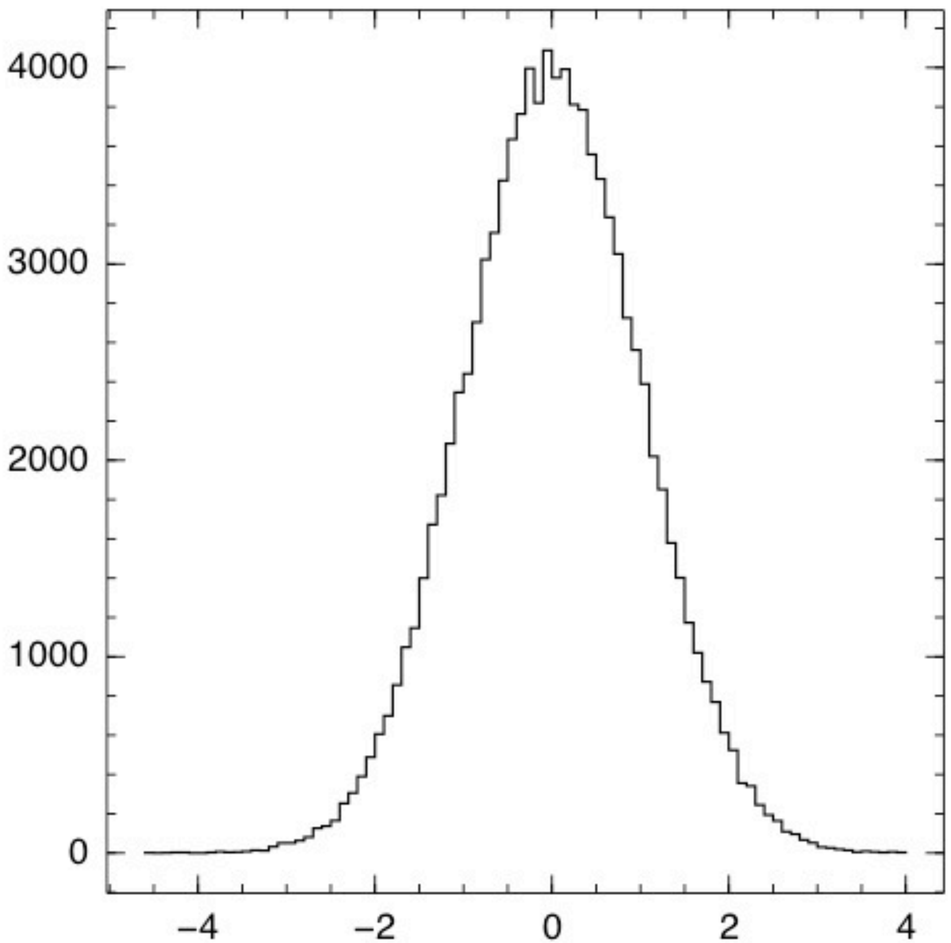


Some noteworthy features

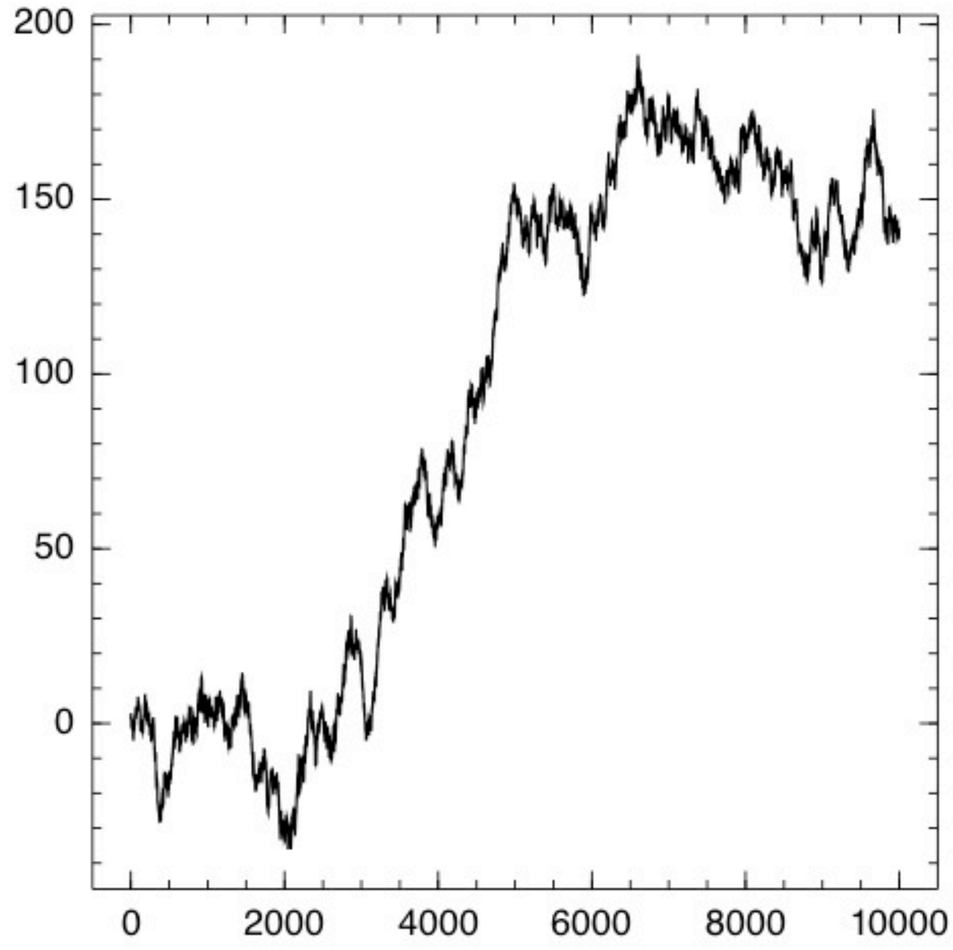
- Open source with an MIT licensed core
- Easy installation - Just download a precompiled binary and run
- Dynamically typed with fast user-defined types
- Multiple dispatch with a sophisticated parametric type system
- JIT compiler - no need to vectorize for performance
- Co-routines
- Distributed memory parallelism
- Effortlessly call C, Fortran, and Python libraries
- Metaprogramming with Lisp-like macros
- Unicode support

A simulated stock market

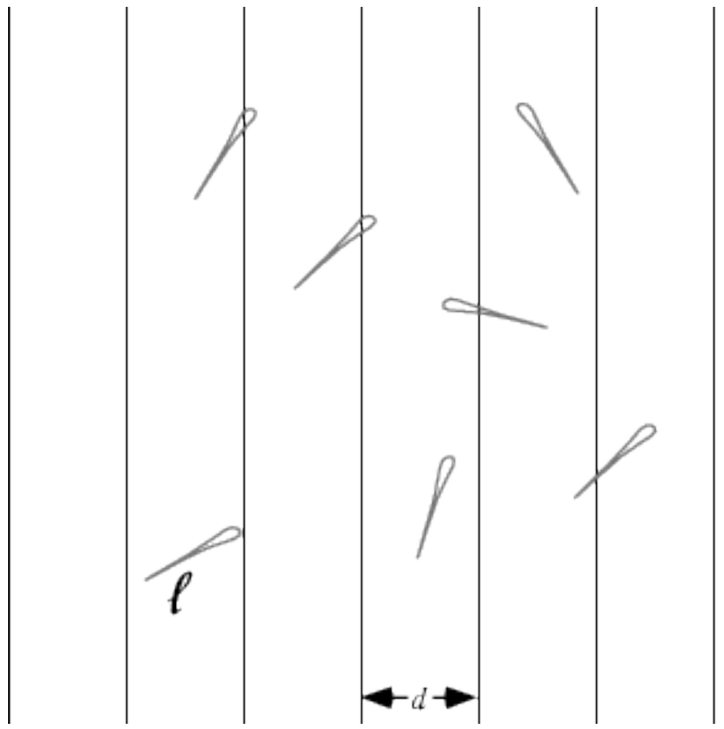
```
julia> plothist(randn(100000), 100)
<plot>
```




```
julia> plot(cumsum(randn(10000)))
<plot>
```

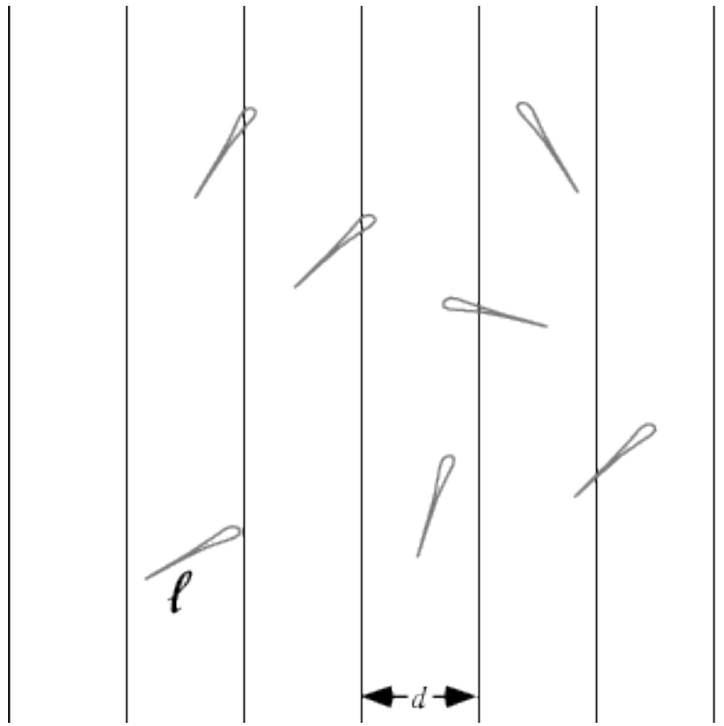


Let’s compute π: Buffon needle problem




```
function buffon(m)
    hit = 0
    for l = 1:m
        mp = rand()
        phi = (rand() * pi) - pi / 2
        xrechts = mp + cos(phi)/2
        xlinks = mp - cos(phi)/2
        if xrechts >= 1 || xlinks <= 0
            hit += 1
        end
    end
    miss = m - hit
    piapprox = m / hit * 2
end
```

Let's compute π in parallel




```
function buffon_par(m)
    hit = @parallel (+) for l = 1:m
        mp = rand()
        phi = (rand() * pi) - pi / 2
        xrechts = mp + cos(phi)/2
        xlinks = mp - cos(phi)/2
        (xrechts>=1||xlinks<=0) ? 1 : 0
    end
    miss = m - hit
    piapprox = m / hit * 2
end
```

Familiar syntax for Matlab / Octave users

```
function randmatstat(t; n=10)
    v = zeros(t)
    w = zeros(t)
    for i = 1:t
        a = randn(n,n)
        b = randn(n,n)
        c = randn(n,n)
        d = randn(n,n)
        P = [a b c d]
        Q = [a b; c d]
        v[i] = trace((P'*P)^4)
        w[i] = trace((Q'*Q)^4)
    end
    std(v)/mean(v), std(w)/mean(w)
end
```

Keyword arguments

Familiar array syntax

Common matrix operations

Common statistics

Last expression is return value

Yes, you can also write low-level code!

```
function qsort!(a,lo,hi)
    i, j = lo, hi
    while i < hi
        pivot = a[(lo+hi)>>>1]
        while i <= j
            while a[i] < pivot; i = i+1; end
            while a[j] > pivot; j = j-1; end
            if i <= j
                a[i], a[j] = a[j], a[i]
                i, j = i+1, j-1
            end
        end
        if lo < j; qsort!(a,lo,j); end
        lo, j = i, hi
    end
    return a
end
```

Pass by reference

Functions ending in ! modify the inputs

Swap elements

Recursion

Call C, Fortran, Python libraries

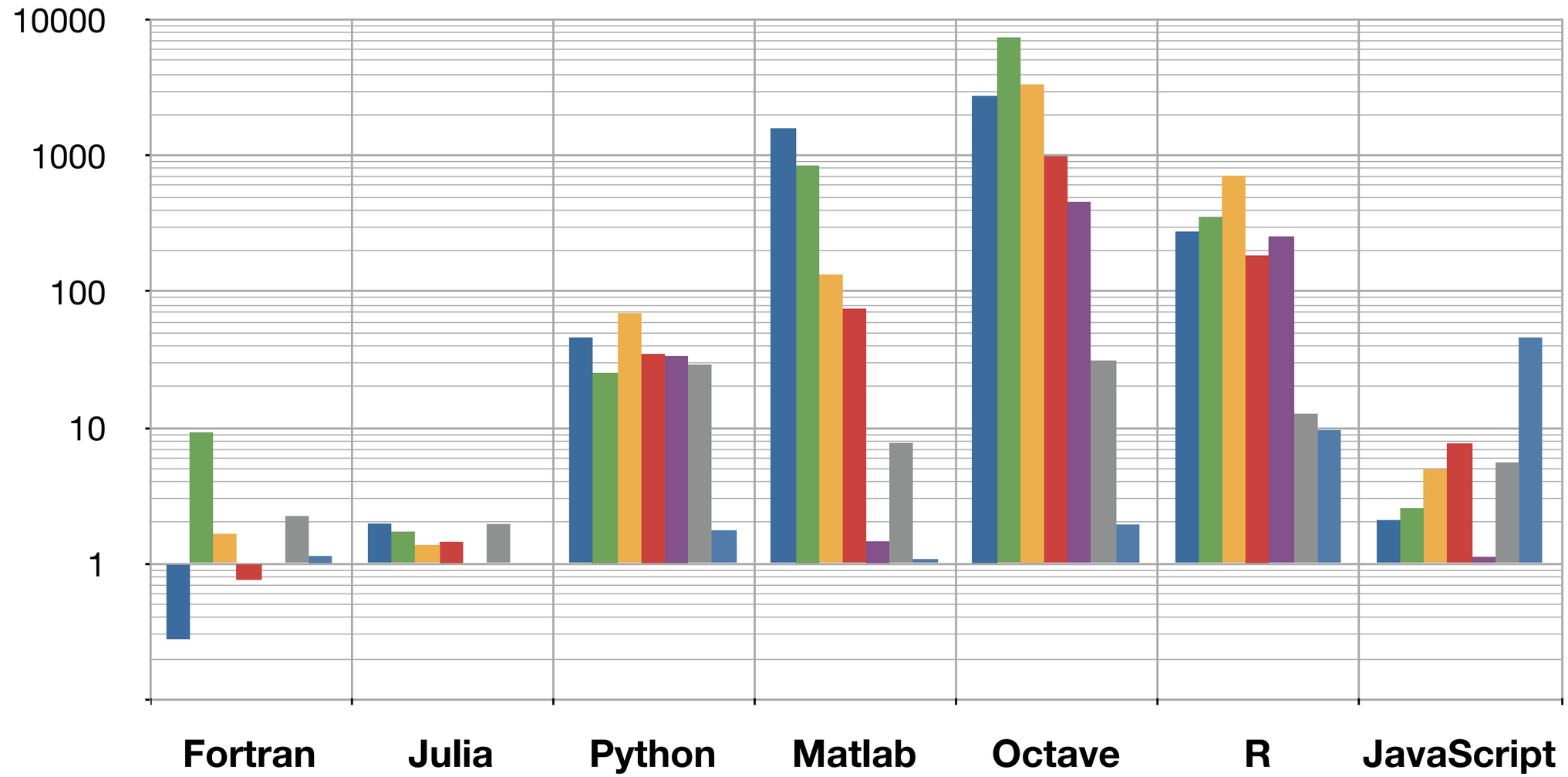
```
julia> ccall(:clock, Int32, ())
2292761

julia> ccall(:getenv, Ptr{Uint8int8}, (Ptr{Uint8},), "SHELL")
Ptr{Uint8} @0x00007fff5fbffc45
julia> bytestring(ans)
"/bin/bash"
```

```
julia> using PyCall     # Installed with Pkg.add(“PyCall”)
julia> @pyimport math
julia> math.sin(math.pi / 4) - sin(pi / 4)
0.0

julia> @pyimport pylab
julia> x = linspace(0,2*pi,1000); y = sin(3*x + 4*cos(2*x));
julia> pylab.plot(x, y; color="red", linewidth=2.0, linestyle="--")
julia> pylab.show()
```

Micro-benchmarks (log-scale)



execution time relative to C++ (lower is better)

Benchmarks: fib, parse_int, quicksort, mandel, pi_sum, rand_mat_stat, and rand_mat_mul

Let's look at some real data

```
julia> Pkg.add("DataFrames")
Installing DataFrames: v0.3.6

julia> using DataFrames

julia> vl = readtable("2013_BS_VL.csv", allowquotes=false)

julia> size(vl)
(1797590,12)

julia> colnames(vl)

julia> describe(vl)
```

What does this data look like?

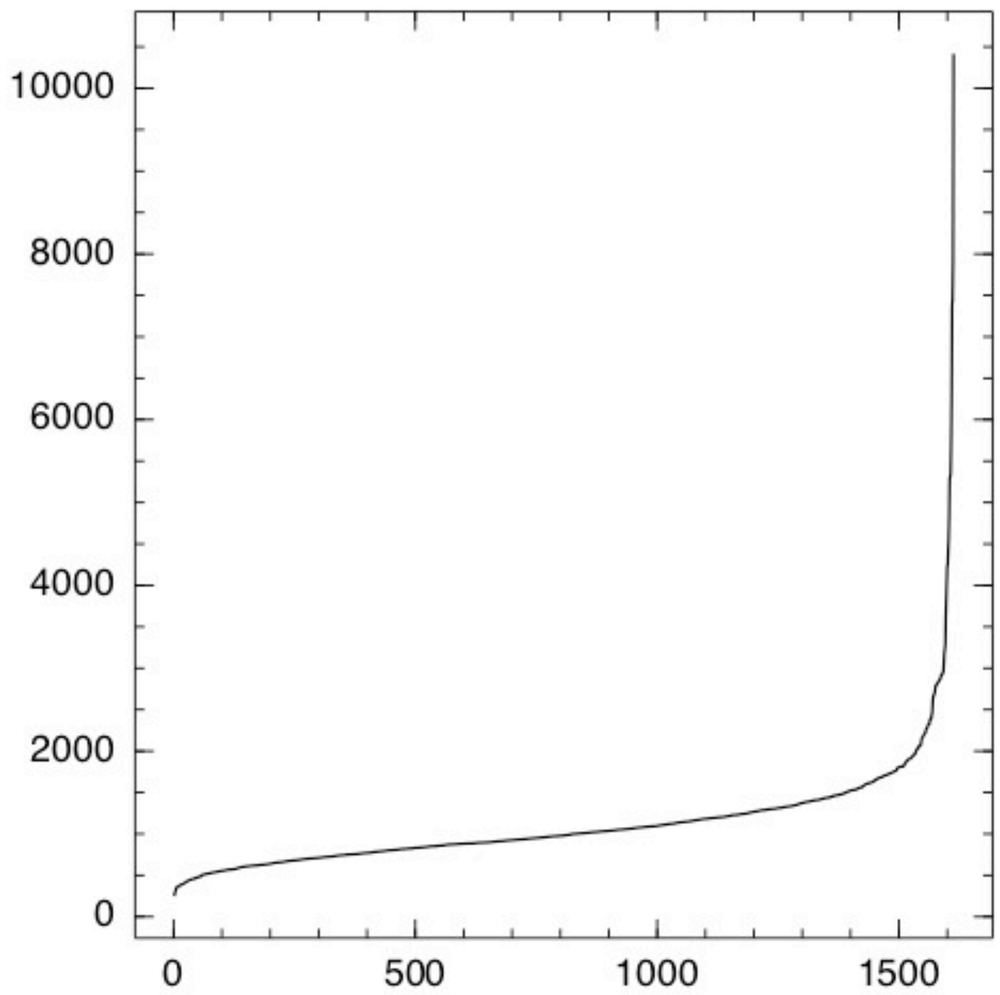
```
julia> colnames(vl)
12-element Union(ASCIIString,UTF8String) Array:
 "AC"
 "ACNAME"
 "PS"
 "PSNAME"
 "PSADDR"
 "PSPART"
 "VoterID"
 "Name"
 "FatherHusband"
 "House"
 "Age"
 "Gender"
```

Data is never clean

```
julia> f = [ try int(vl["Age"][i]); catch -1; end for i
in 1:size(vl,1) ]
julia> vl = vl[f.!=-1, :]
julia> vl[:Age] = PooledDataArray(int(vl[:Age]))
julia> vl[:ACNAME] = PooledDataArray(int(vl[:ACNAME]))
julia> vl[:PSNAME] = PooledDataArray(vl[:PSNAME])


julia> by(vl, :ACNAME, nrow)
julia> by(vl, :ACNAME, x->mean(x[:Age]))
julia> by(vl, :ACNAME, x->sum(DataArray(x[:Age] .<=
40)))
```

Draw your own insights




```
julia> describe(by(vl, :PSNAME, nrow))

PSNAME
Length: 1613
Type  : Pooled UTF8String
NAs   : 0

x1
Min       250.0
1st Qu.   772.0
Median    981.0
Mean      1113.8326100433974
3rd Qu.   1276.0
Max       10416.0

julia> plot(1:1613, sort(pspop[:x1]))
```

A great community

100+ contributors, 1000+ mailing list subscribers, 175+ packages

AWS, ArgParse, BSplines, Benchmark, BinDeps, BioSeq, BloomFilters, Cairo, Calculus, Calendar, Cartesian, Catalan, ChainedVectors, ChemicalKinetics, Clang, Clp, ClusterManagers, Clustering, Codecs, CoinMP, Color, Compose, ContinuedFractions, Cpp, Cubature, Curl, DICOM, DWARF, DataFrames, DataStructures, Datetime, Debug, DecisionTree, Devectorize, DictUtils, DictViews, DimensionalityReduction, DiscreteFactor, Distance, Distributions, DualNumbers, ELF, Elliptic, Example, ExpressionUtils, FITSIO, FactCheck, FastaIO, FastaRead, FileFind, FunctionalCollections, FunctionalUtils, GLFW, GLM, GLPK, GLPKMathProgInterface, GLUT, GSL, GZip, Gadfly, Gaston, GeoIP, GeometricMCMC, GetC, GoogleCharts, Graphs, Grid, Gtk, Gurobi, HDF5, HDFS, HTTP, HTTPClient, Hadamard, HttpCommon, HttpParser, HttpServer, HypothesisTests, ICU, ImageView, Images, ImmutableArrays, IniFile, Iterators, Ito, JSON, JudyDicts, JuliaWebRepl, KLDivergence, LIBSVM, Languages, LazySequences, LibCURL, LibExpat, LinProgGLPK, Loss, MAT, MATLAB, MCMC, MDCT, MLBase, MNIST, MarketTechnicals, MathProg, MathProgBase, Meddle, Memoize, Meshes, Metis, MixedModels, Monads, Mongo, Mongrel2, Morsel, Mustache, NHST, NIfTI, NLopt, Named, NetCDF, NumericExtensions, NumericFunctors, ODBC, ODE, OpenGL, OpenSSL, Optim, Options, PLX, PTools, PatternDispatch, Phylo, Phylogenetics, Polynomial, Profile, ProgressMeter, ProjectTemplate, PyCall, PyPlot, PySide, Quandl, QuickCheck, RDatasets, REPL, RNGTest, RPMmd, RandomMatrices, Readline, Regression, Resampling, Rif, Rmath, RobustStats, Roots, SDE, SDL, SVM, SemidefiniteProgramming, SimJulia, SimpleMCMC, Sims, Sodium, Soundex, Sqlite, Stats, StrPack, Sundials, SymPy, TOML, Terminals, TextAnalysis, TextWrap, TimeModels, TimeSeries, Tk, TopicModels, TradingInstrument, Trie, URLParse, UTF16, Units, ValueDispatch, WAV, WebSockets, Winston, YAML, ZMQ, Zlib, kNN