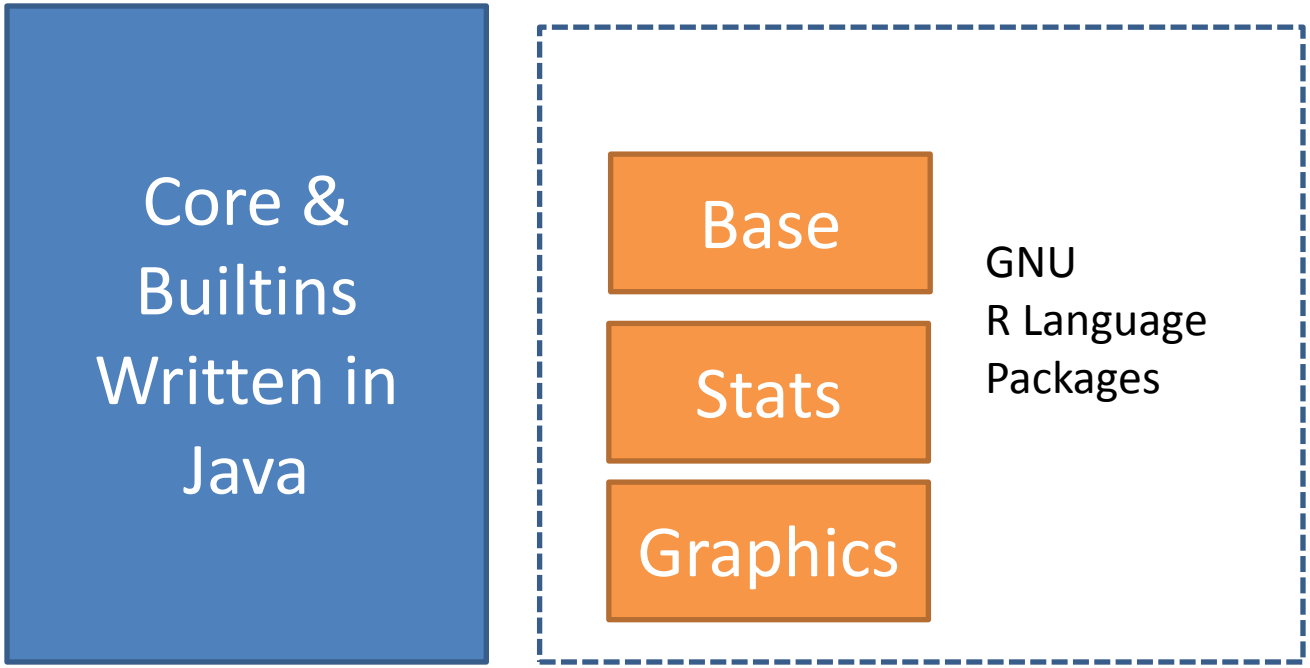

Alexander Bertram

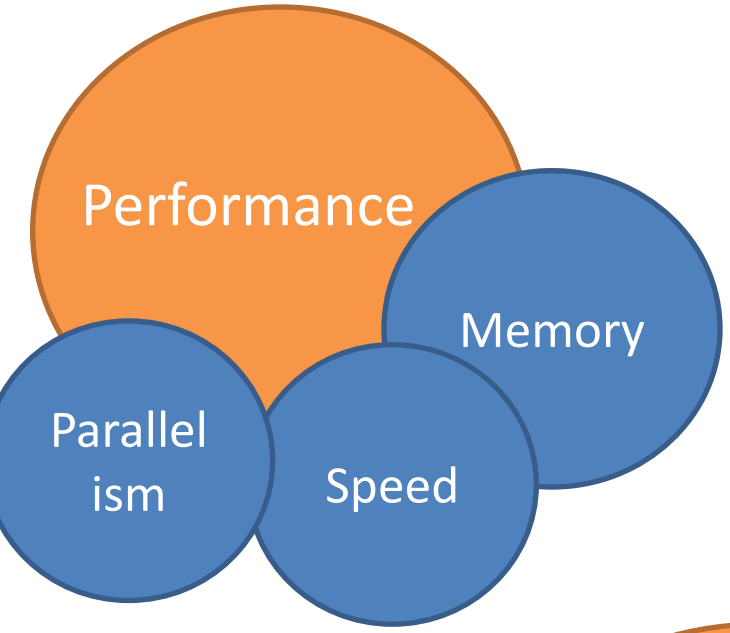
BeDataDriven

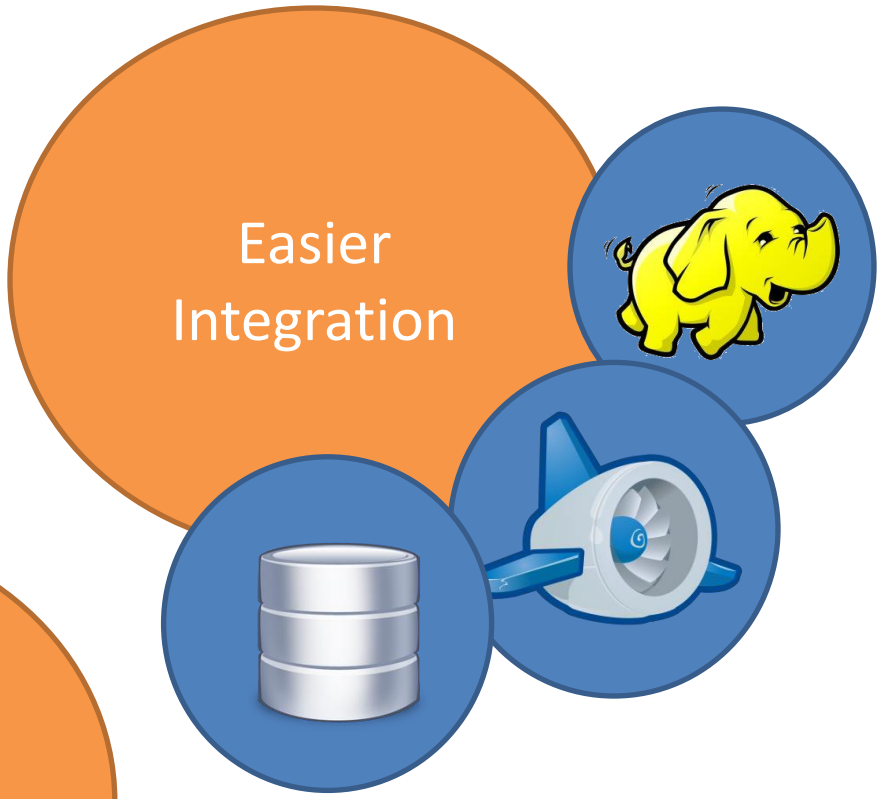
# What?

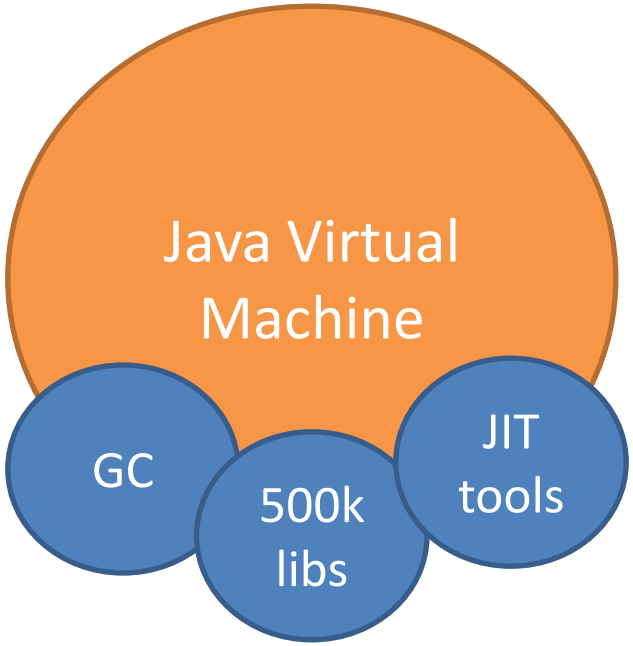
Renjin is a new interpreter for the R language.



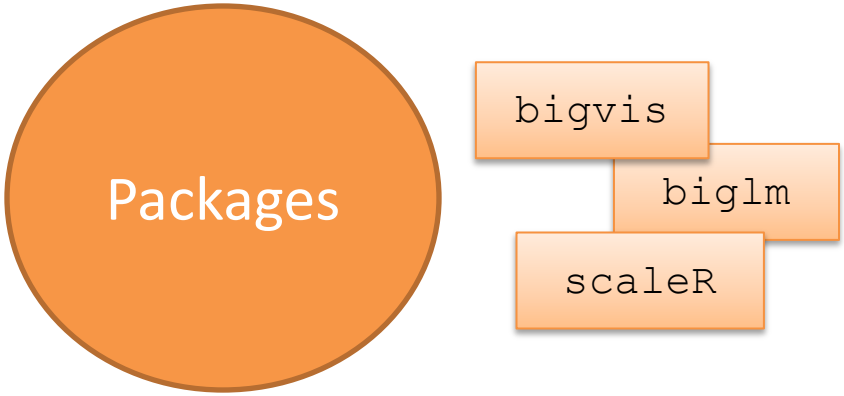
# Why?


Performance
Memory
Parallel ism
Speed

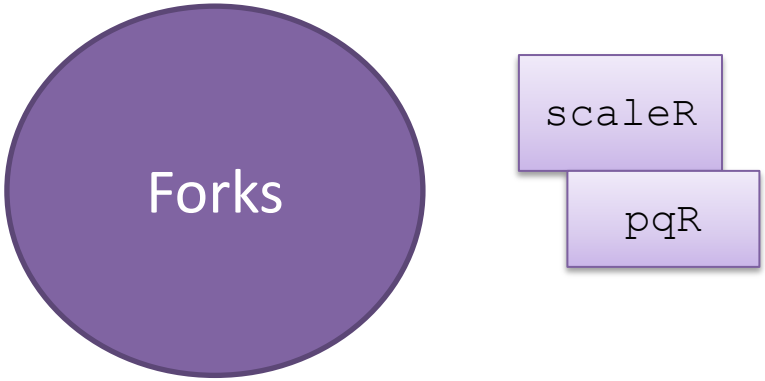

Easier Integration


Java Virtual Machine
GC
500k libs
JIT tools

# Sure, but why Renjin?




+ **High performance for specific applications**
- Require rewriting existing code
- Limited applicability




+ **Marginal improvements for all code**
- **Unable to address underlying limitations of the GNU R interpreter**

What do I get, like, today?

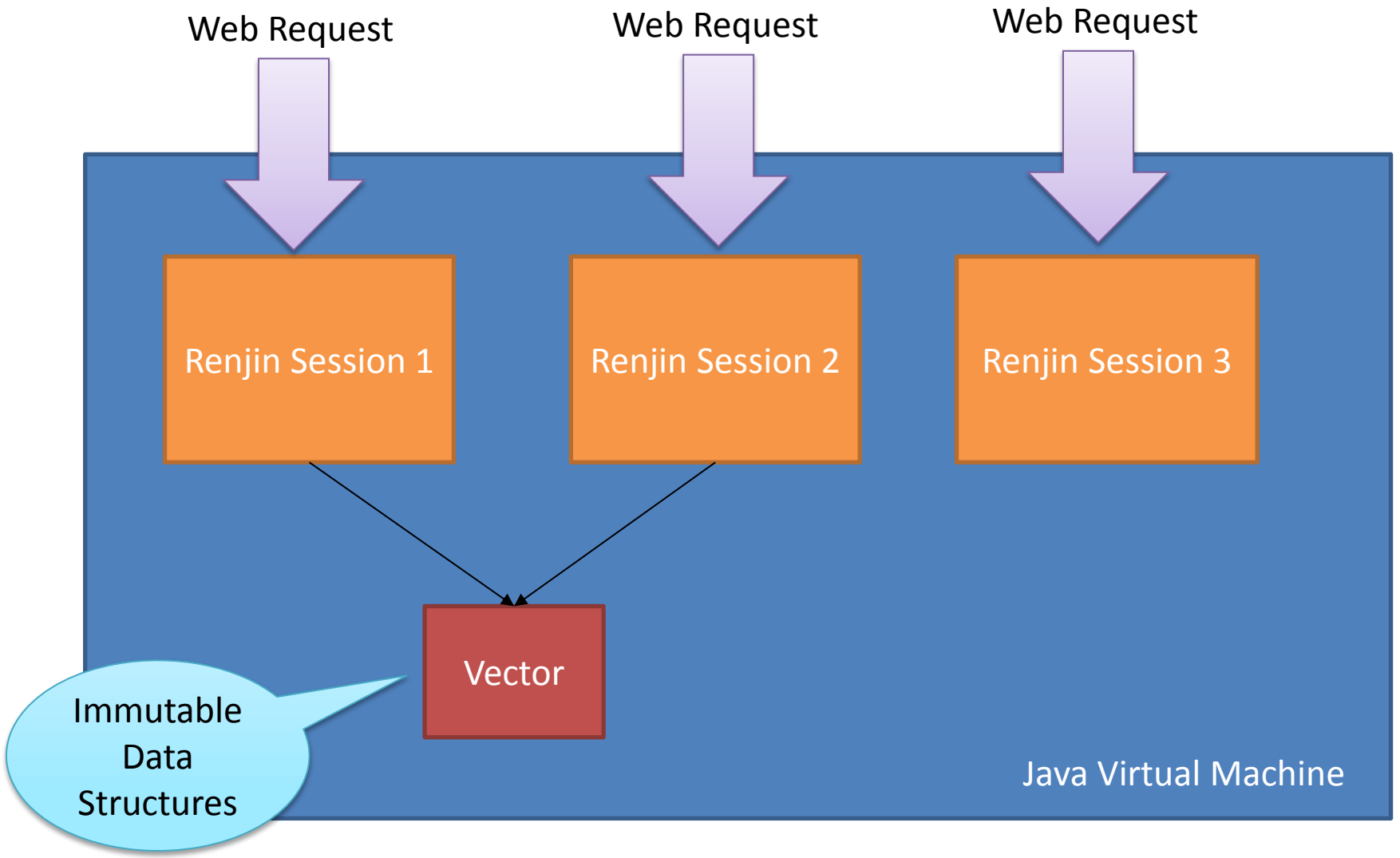
# Flexible

Command-Line Interpreter

```
> renjin –f myscript.R
```

Embeddable Java Library

# Multiple In-process sessions, Shared Data



# Memory Efficiency

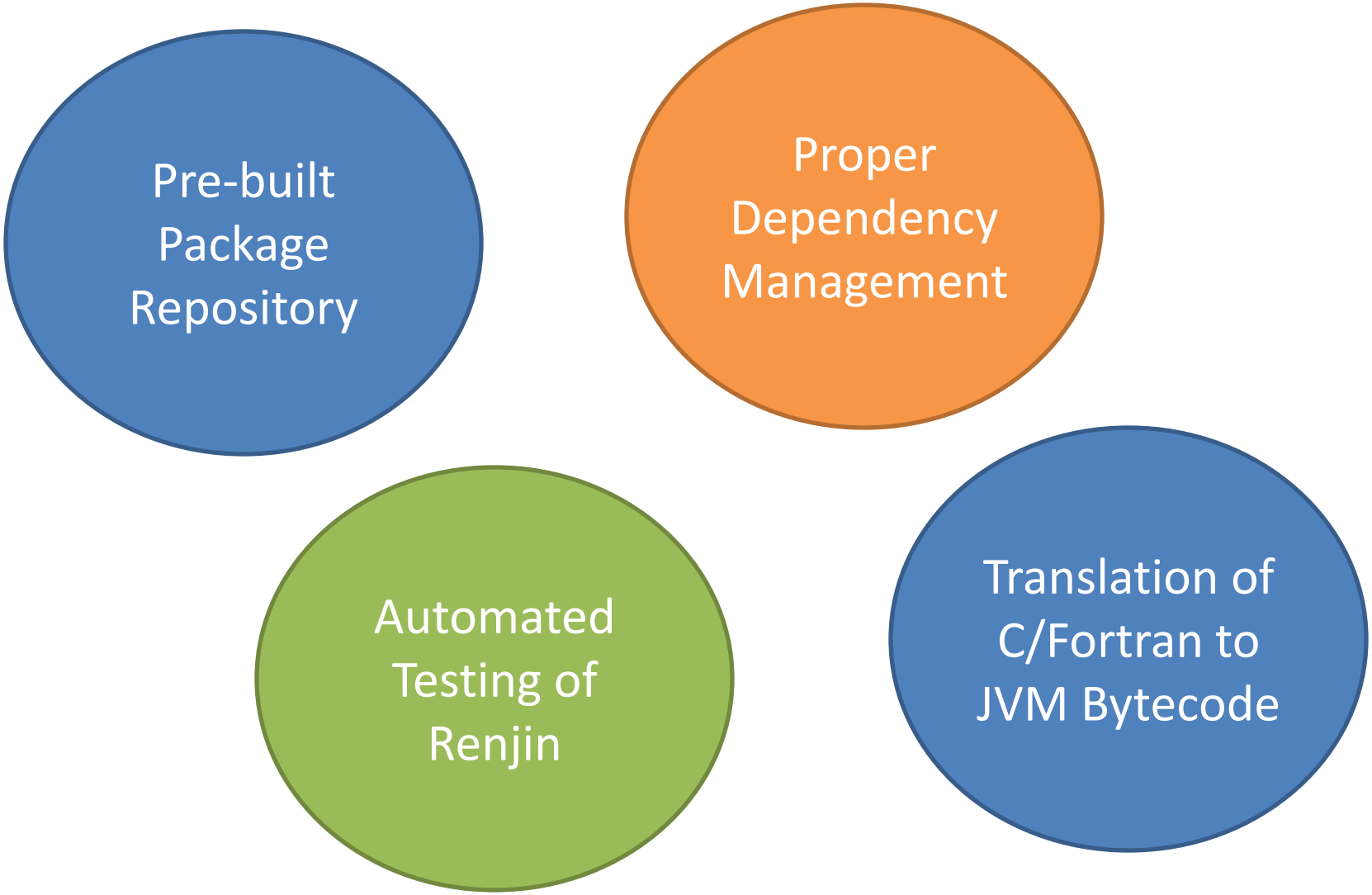
```
                                  # GNU R          Renjin
x <- runif(1e8)                   # +721 MB        + 721 MB
y <- x + 1                        # +761 MB
comment(y) <- "important!"        # +763 MB
```

- getAttributes()

Vector Interface

- length()
- getElement(int index)

# packages.renjin.org



# Seamless Access to Java/Scala Classes

```
import(com.acme.Customer)

bob <- Customer$new(name='Bob', age=36)
carol <- Customer$new(name='Carole', age=41)

bob$name <- "Bob II"
cat(c("Name: ", bob$name, "; Age: ", bob$age))
```

# Simple to embed in larger systems

```
// create a script engine manager
ScriptEngineManager factory = new ScriptEngineManager();

// create an R engine
ScriptEngine engine = factory.getEngineByName("Renjin");

// load package from classpath
engine.eval("library(survey)");

// evaluate R code from String
engine.eval("print('Hello, World')");

// evaluate R script on disk
engine.eval(new FileReader("myscript.R"));

// evaluate R script from classpath
engine.eval(new InputStreamReader(
   getClass().getResourceAsStream("myScript.R")));
```

# Package Development in Java

```
@DataParallel
@Deferrable
public static String chartr(
      String oldChars,
      String newChars,
      @Recycle String x)
{
  StringBuilder translation = new StringBuilder(x.length());
  for(int i=0;i!=x.length();++i) {
    int codePoint = x.codePointAt(i);
    int charIndex = oldChars.indexOf(codePoint);
    if(charIndex == -1) {
        translation.appendCodePoint(codePoint);
    } else {
        translation.appendCodePoint(
            newChars.codePointAt(charIndex));
    }
  }
    return translation.toString();
}
```

# Under the hood

# Specialized Execution Modes

"Slow" AST Interpreter

- Supports full dynamism of R
- Compute on the language

Vector Pipeliner

- Acts like a query planner
- Batches, auto-parallelizes vector workflows

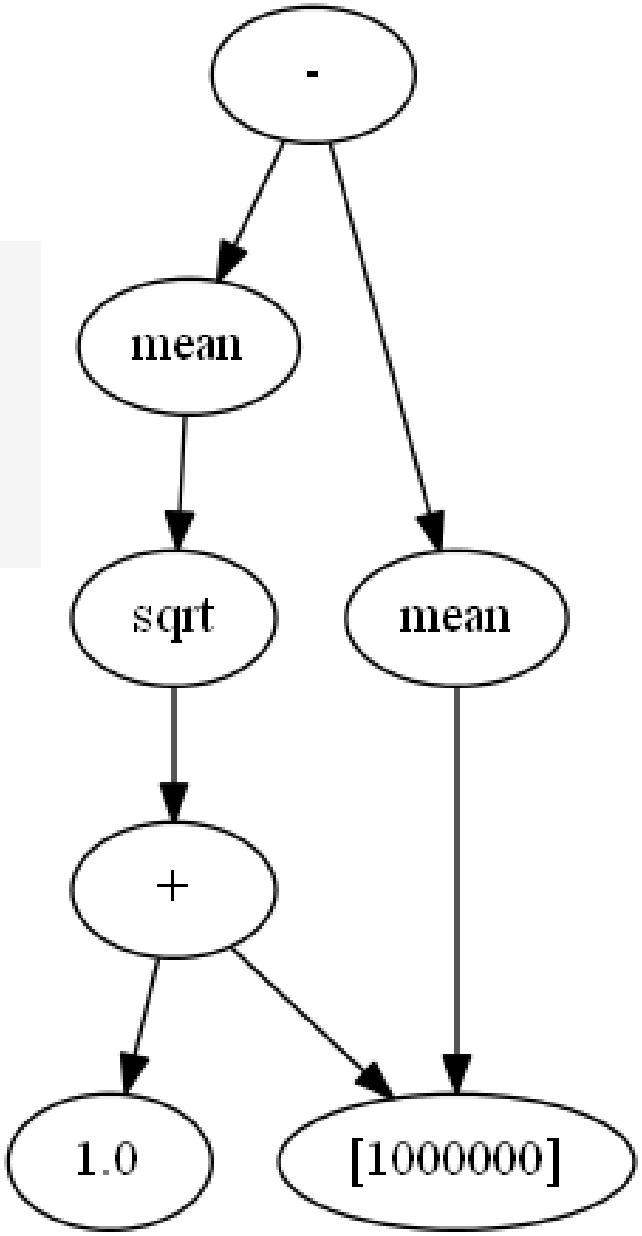
Scalar Compiler

- Partially evaluates & compiles loop bodies, apply functions to JVM byte code

# Queuing up work for the Vector Pipeliner

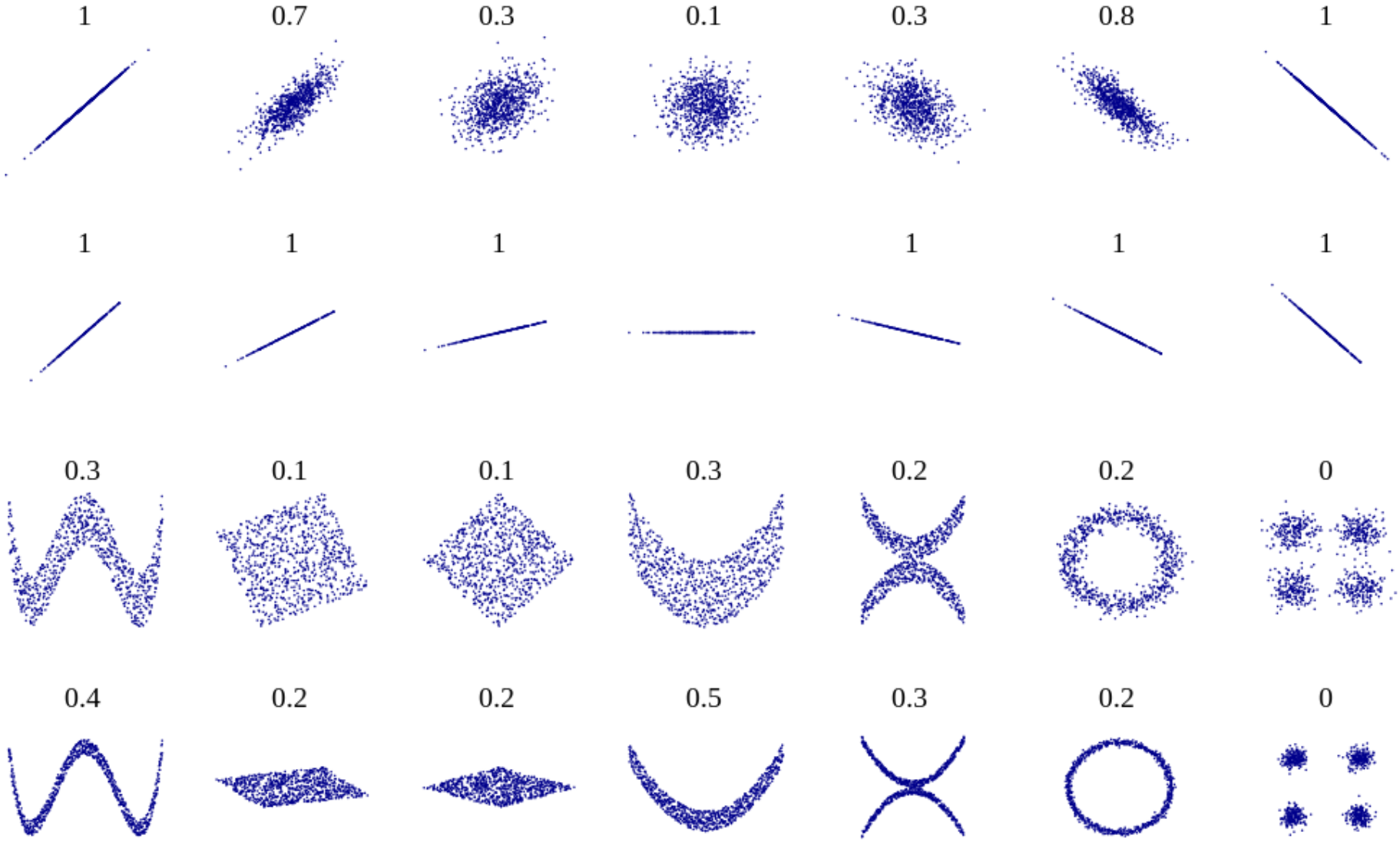
```
x <- runif(1e6)
y <- sqrt(x + 1)
z <- mean(y) - mean(x)
attr(z, 'comments') <- 'still not computed'
print(length(z)) # prints "1"
                 # but doesn't
                 #evaluate the mean
print(z) # triggers computation
```

```
x <- runif(1e6)
y <- sqrt(x + 1)
z <- mean(y) - mean(x)
```



Real-world case study:

# Distance Correlation in the Energy Package



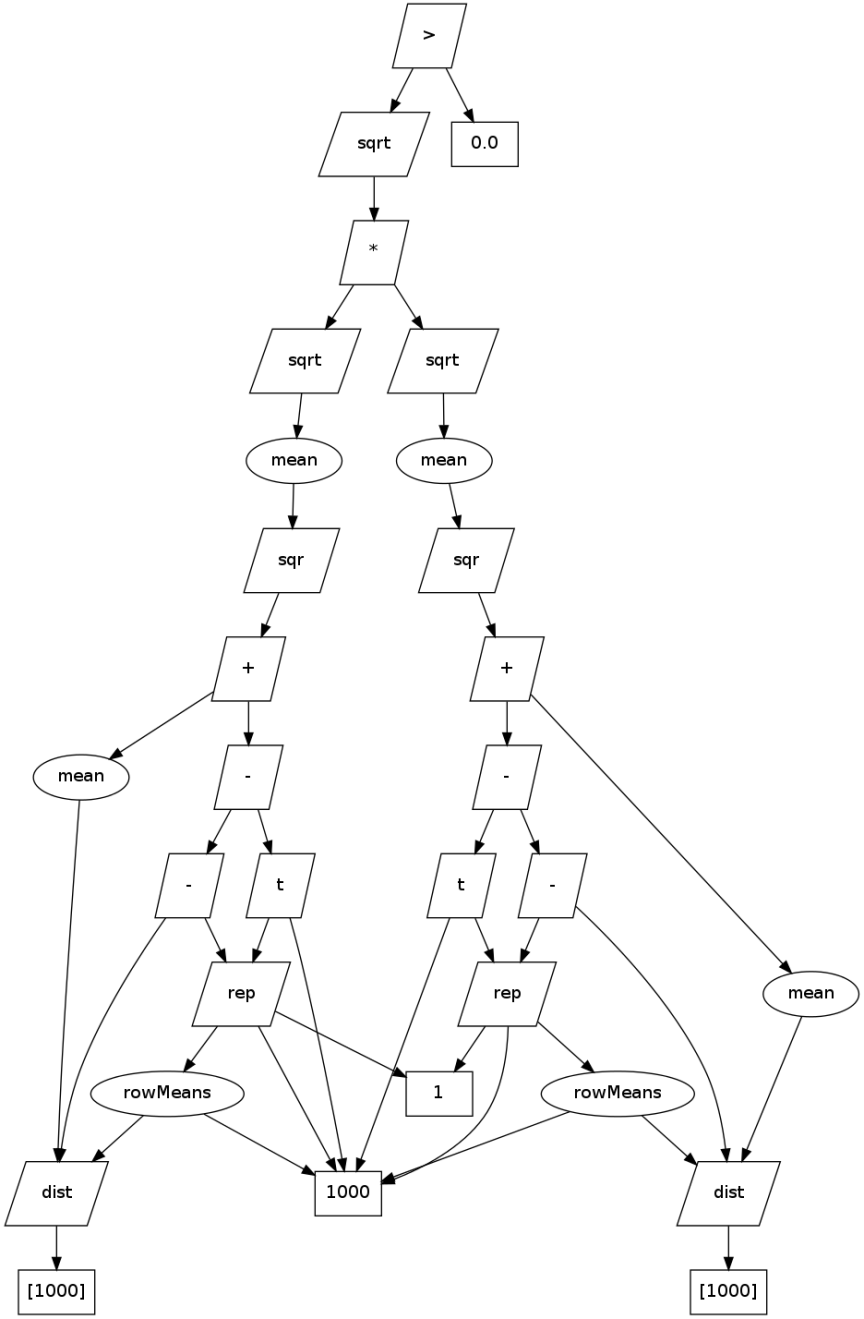

**Distance correlation**: robust measure of association.
Zero <u>if and only if</u> variables are independent.

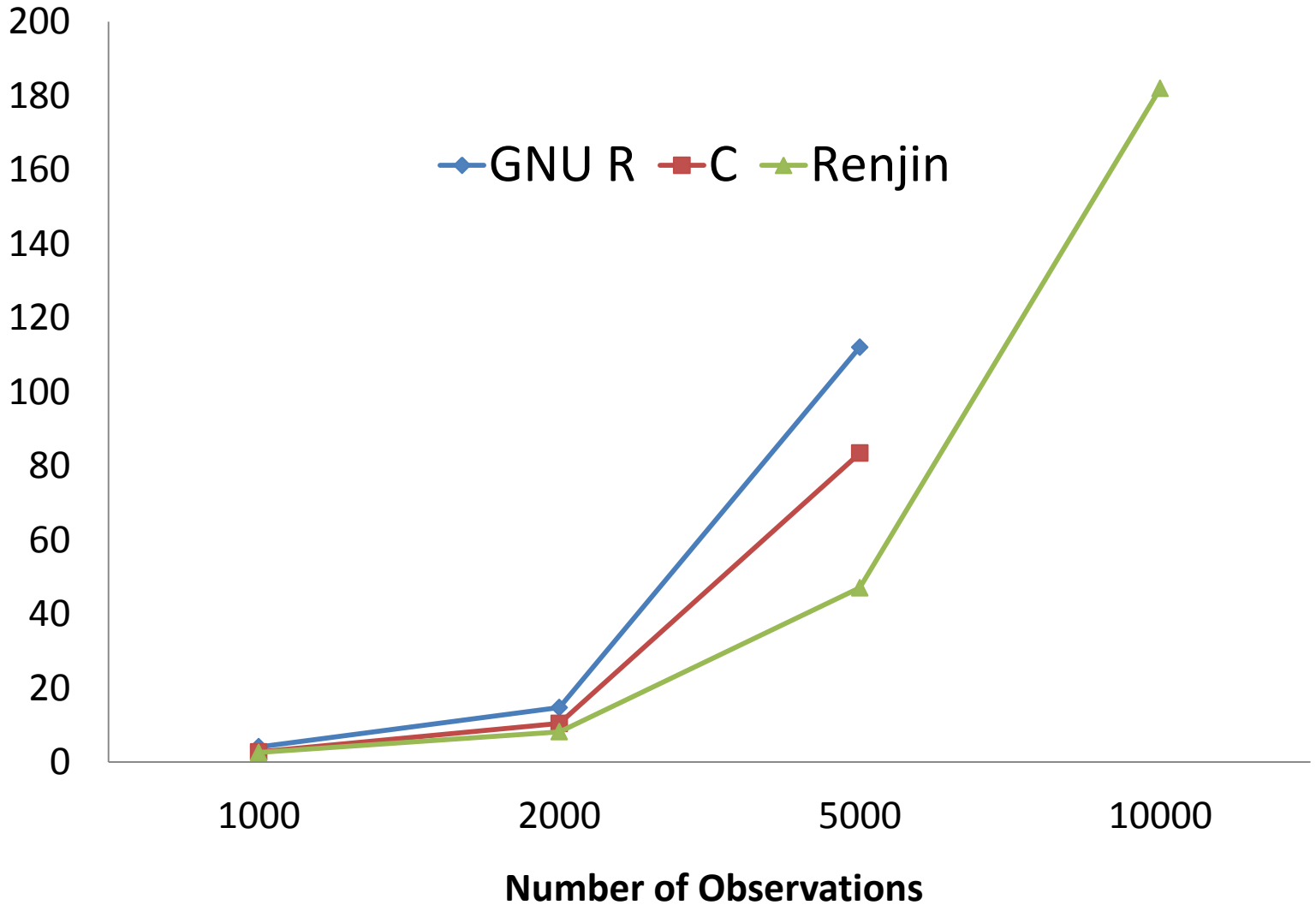
```
dcor <- function (x, y, index = 1) {
  x <- as.matrix(dist(x))
  y <- as.matrix(dist(y))
  n <- nrow(x)
  m <- nrow(y)
  dims <- c(n, ncol(x), ncol(y))
  Akl <- function(x) {
    d <- as.matrix(x)^index
    m <- rowMeans(d)
    M <- mean(d)
    a <- sweep(d, 1, m)
    b <- sweep(a, 2, m)
    return(b + M)
  }
  A <- Akl(x)
  B <- Akl(y)
  dCov <- sqrt(mean(A * B))
  dVarX <- sqrt(mean(A * A))
  dVarY <- sqrt(mean(B * B))
  V <- sqrt(dVarX * dVarY)
  if (V > 0)
    dCor <- dCov/V
  else dCor <- 0
  return(list(dCov = dCov, dCor = dCor, dVarX =
dVarX, dVarY = dVarY))
}
```

dist(x)
Evaluates as a view

Defer rowMeans(x) until later

Need to evaluate


>
sqrt
0.0
*
sqrt
sqrt
mean
mean
sqr
sqr
+
+
mean
-
-
-
t
t
-
rep
rep
mean
rowMeans
1
rowMeans
dist
1000
dist
[1000]
[1000]

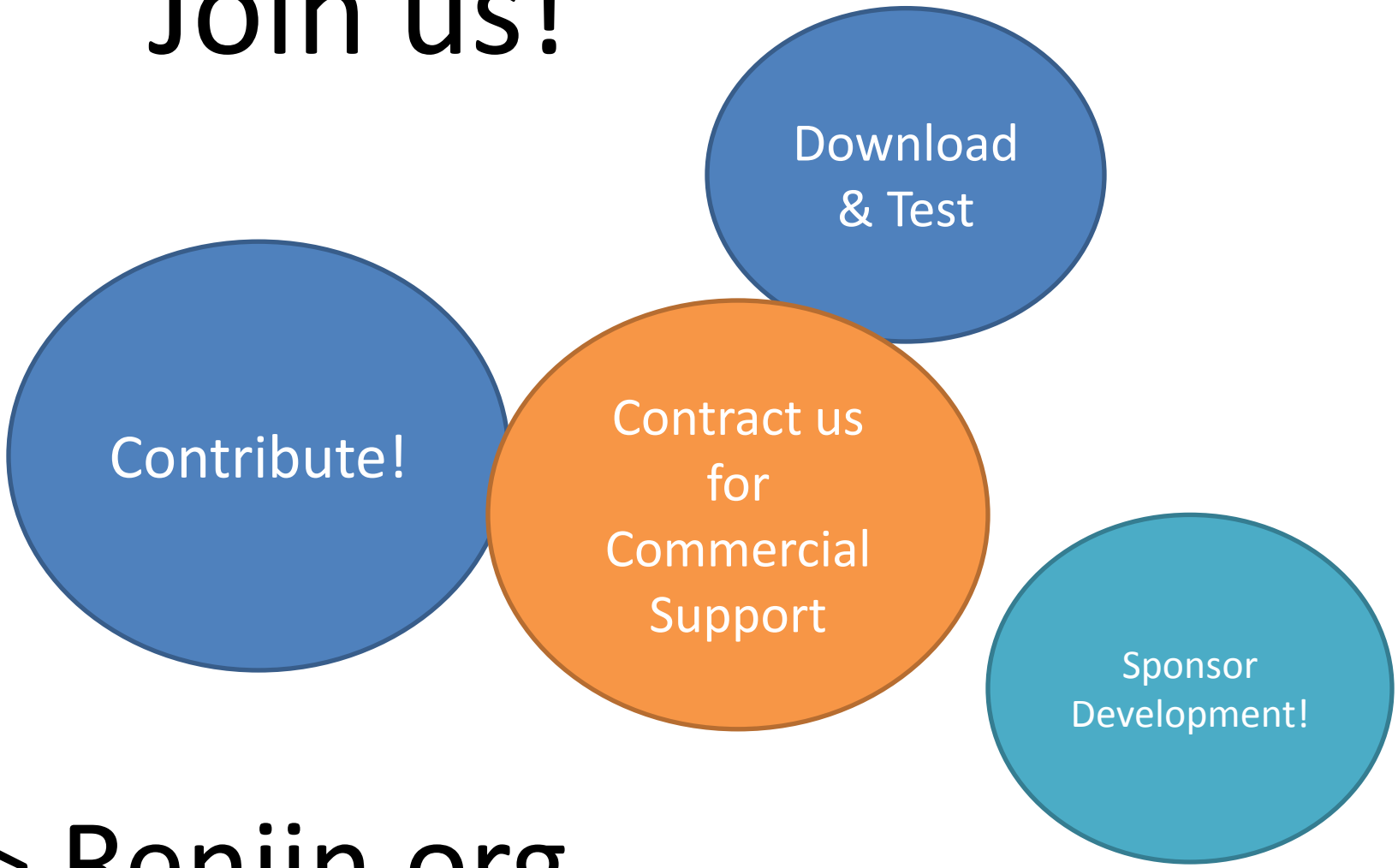

Run time of distance correlation of 10 pairs of variables
200
180
160
140
120
100
80
60
40
20
0
GNU R
C
Renjin
1000
2000
5000
10000
Number of Observations

# Where do we go from here?

# Inspired by…


PyPy

# Join us!




> Renjin.org