PIVOT SCIENCES

High performance R

Integrating cpp into your workflow

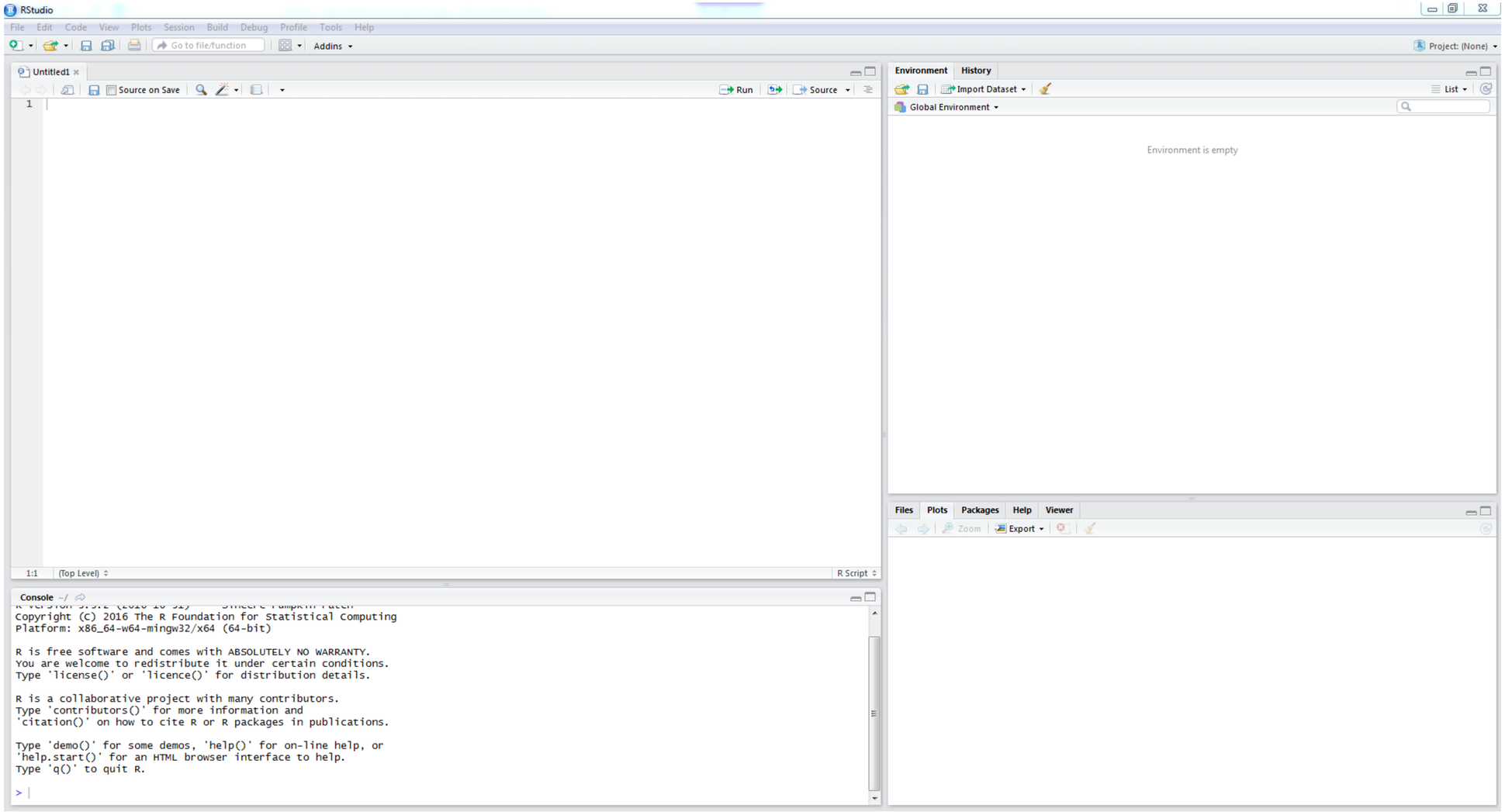
SatRday - February 2017

Robert Bennetto

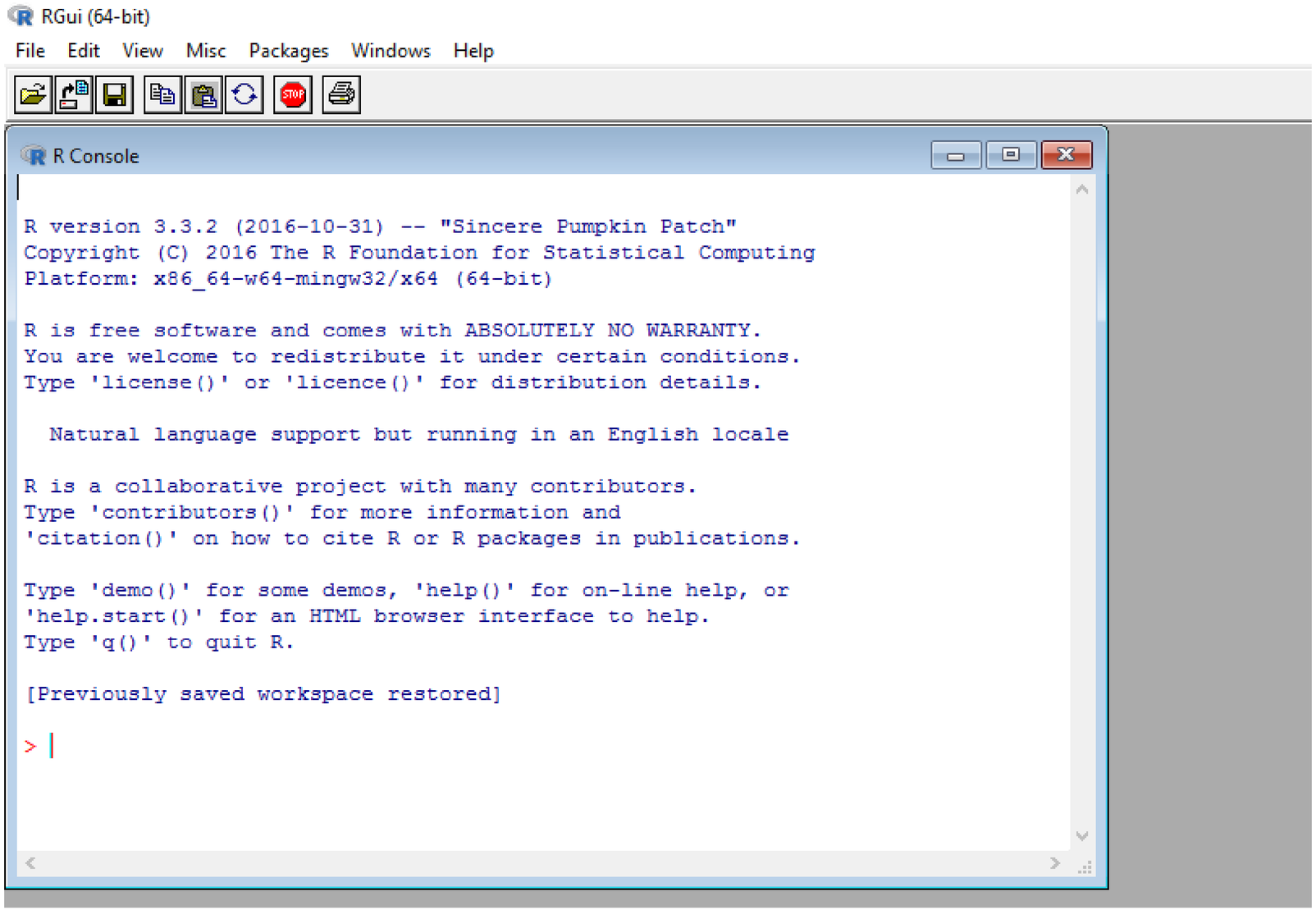
Outline

- The design of R
- Motivation
- Some examples
- Discussion

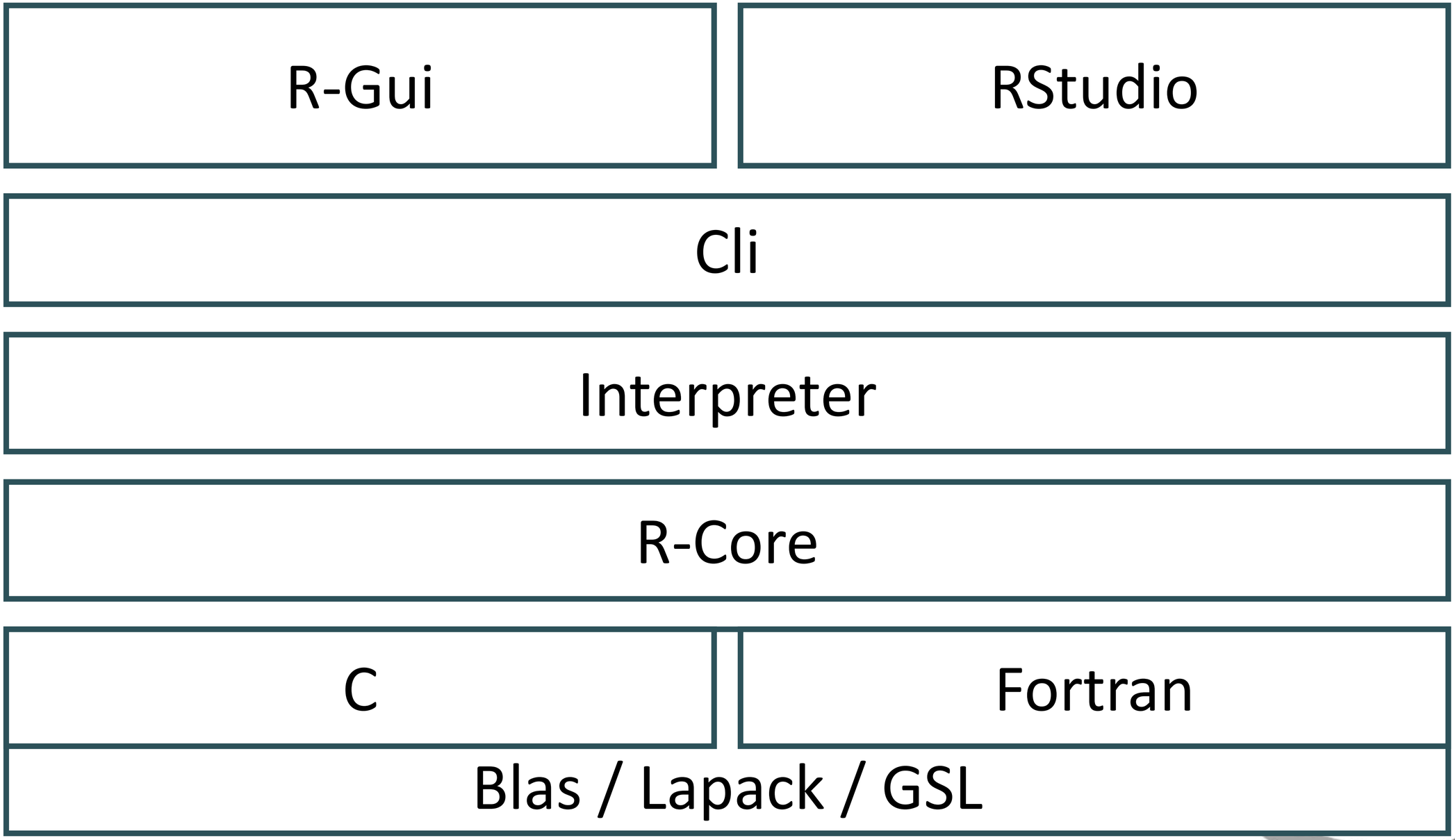
The design of R



The design of R



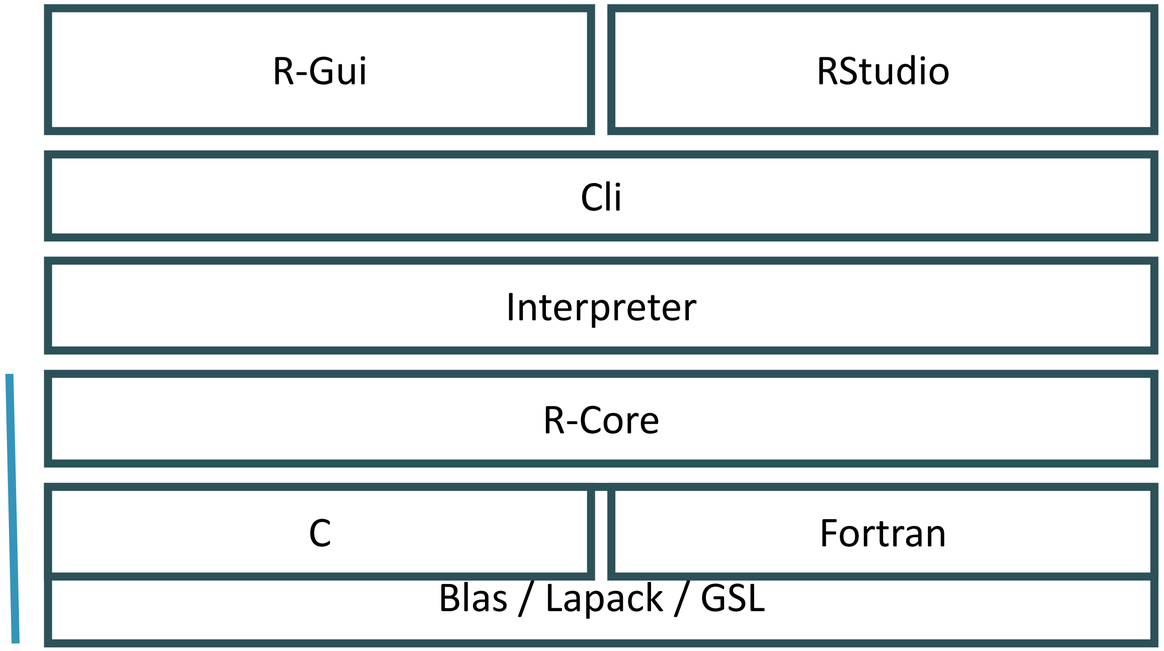
The design of R



How you use R matters

- You can embed R-methods in an application
- Or use the interpreter
- R is hugely optimised here

- It's optimised in other places
  - But those operations are scary



Example: counting

```
> sumlots<-function(){
+   j<-0
+   for(i in 1:1000000){
+     j <- j + 1
+   }
+ }
> system.time(sumlots())
   user  system elapsed
   0.36    0.00    0.36
>
```

- We would not normally code like this in R
- Not much to this function
- We've used a loop
- Seem quick, 360ms.
- It is quick, considering it's done 1,000,000 interpretations of: `j <- j + 1`

Example: counting

- If we insist on making a vector, and adding 1m components, this does the same:

```
> system.time(sum(rep(1,1000000)))
   user  system elapsed
      0       0       0
```

- It was trivial, but we phrased it wrong the first time.

Common workarounds

1: Avoid loops

- Remember to also avoid deep recursions

2: Use an (*)apply

3: Only use Vectors

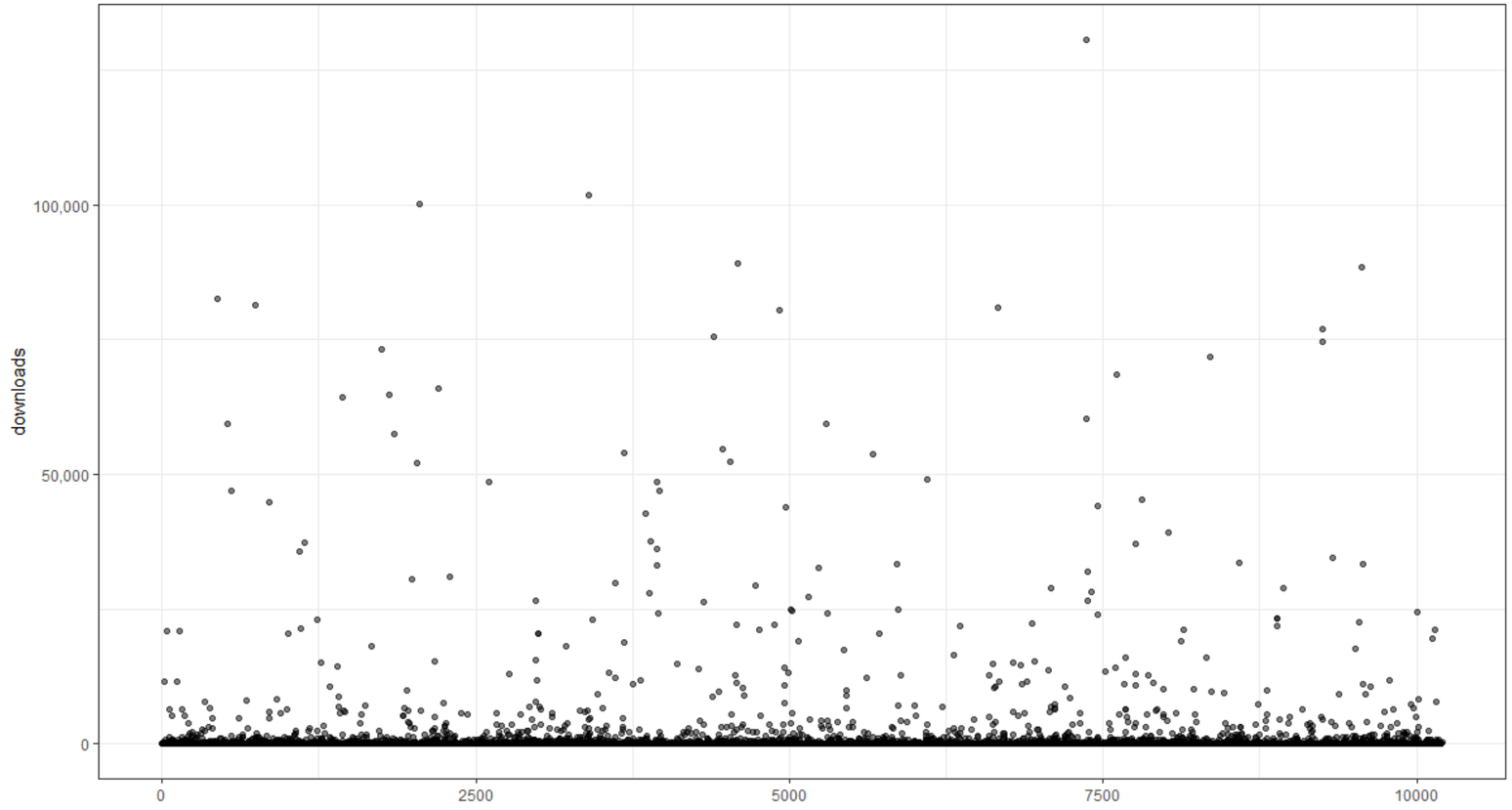
- kd-trees, priority queues, any useful data structures are out.

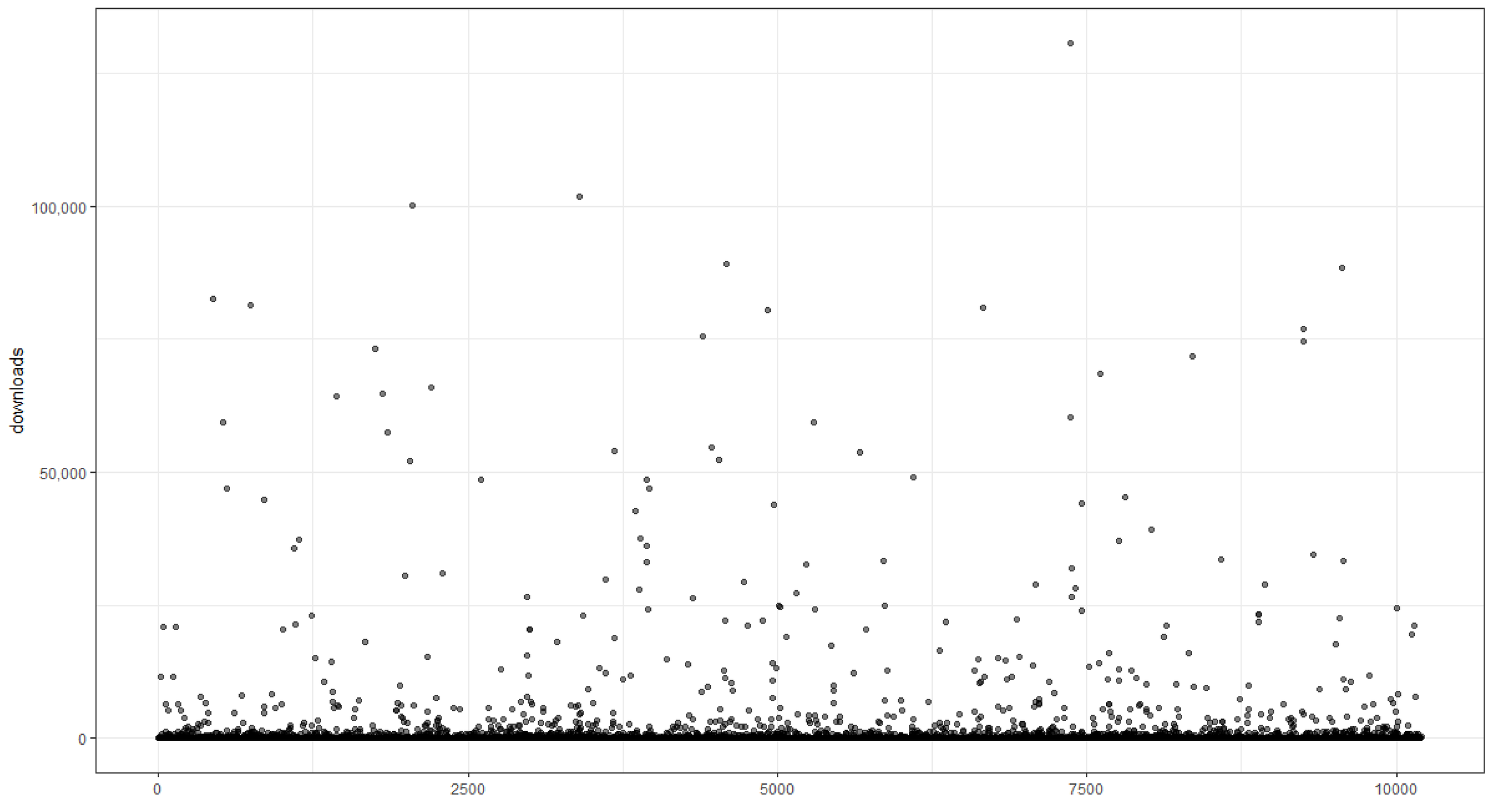
4: Work in python

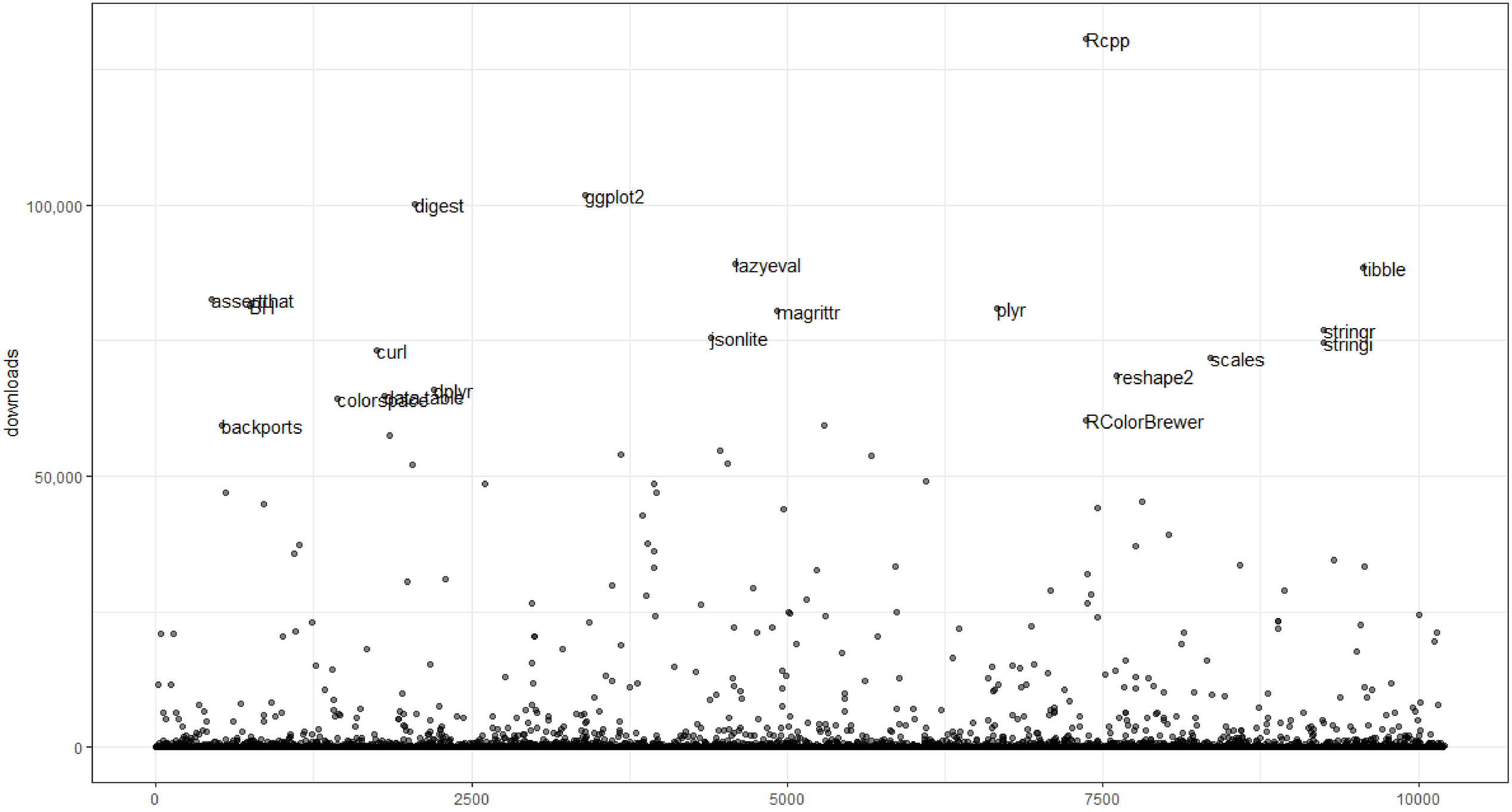
- lol

Motivation - at 16/02/2017

- There are 10093 packages on CRAN
- Not all these packages are equally popular downloads.




downloads
100,000
50,000
0
0
2500
5000
7500
10000


Rcpp
ggplot2
digest
lazyeval
tibble
assertthat
BH
magrittr
plyr
jsonlite
stringr
stringi
curl
scales
reshape2
colorspace
data.table
dplyr
backports
RColorBrewer
downloads
100,000
50,000
0
0
2500
5000
7500
10000

Rccp !

- Rcpp allows packages which leverage c/c++/fortran to be compiled for the system you're using.

- dplyr is a good example of this.
  - Install on windows – 10 seconds (prebuilt binaries – windows is lame)
  - Install on Unix – 5 minutes (compiles from source)

- How hard can it be?

Rcpp example:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

stance_matrix_gen.R | server.R | boost_testing.R | Untitled2* | Untitled3* | Untitled4 | Untitled

Source on Save Source

```
#include <Rcpp.h>
using namespace Rcpp;

// This is a simple example of exporting a C++ function to R. You can
// source this function into an R session using the Rcpp::sourceCpp
// function (or via the Source button on the editor toolbar). Learn
// more about Rcpp at:
//
//   http://www.rcpp.org/
//   http://adv-r.had.co.nz/Rcpp.html
//   http://gallery.rcpp.org/
//

// [[Rcpp::export]]
NumericVector timesTwo(NumericVector x) {
  return x * 2;
}


// You can include R code blocks in C++ files processed with sourceCpp
// (useful for testing and development). The R code will be automatically
// run after the compilation.
```

13:1 (Top Level) C/C++

Rcpp example:

```
// [[Rcpp::export]]
int sumlots_quick(){
  int j = 0;
  for(int i = 0; i < 1000000; i++){
    j += 1;
  }
  return (j);
}
```

```
> Rcpp::sourceCpp('~/cpp_example.cpp')

> sumlots_quick()
[1] 1000000

> system.time(sumlots_quick())
   user  system elapsed
      0       0       0

> system.time(Rcpp::sourceCpp('~/cpp_example.cpp'))
   user  system elapsed
   0.00    0.03    5.47
```

Trade-off between once off compile time and multiple calls to the same function

Some *gotacha's* to watch out for

- There be dragons
  - Types matter
  - The size of things matter

```
// [[Rcpp::export]]
int sumlots_quick(){
  int j = 0;
  for(int i = 0; i < 10000000000; i++){
    j += 1;
  }
  return (j);
}
```

∞

Fortunately you can avoid this and code like you would in R.

Rcpp Sugar

What you get for your efforts

- Flexibility of R with maximum speed
- Can identify bottleneck processes and optimise those while still being ‘experimental’
- The cpp STL is well tested and *just* works.

Example: priority queues

Very useful data structure, supports two functions:

• Insert with priority	O(log(n))	cheap
• Get min	O(1)	≈free

No reasonable way of replicating this performance in native-R.

A vector-based implementation gives:

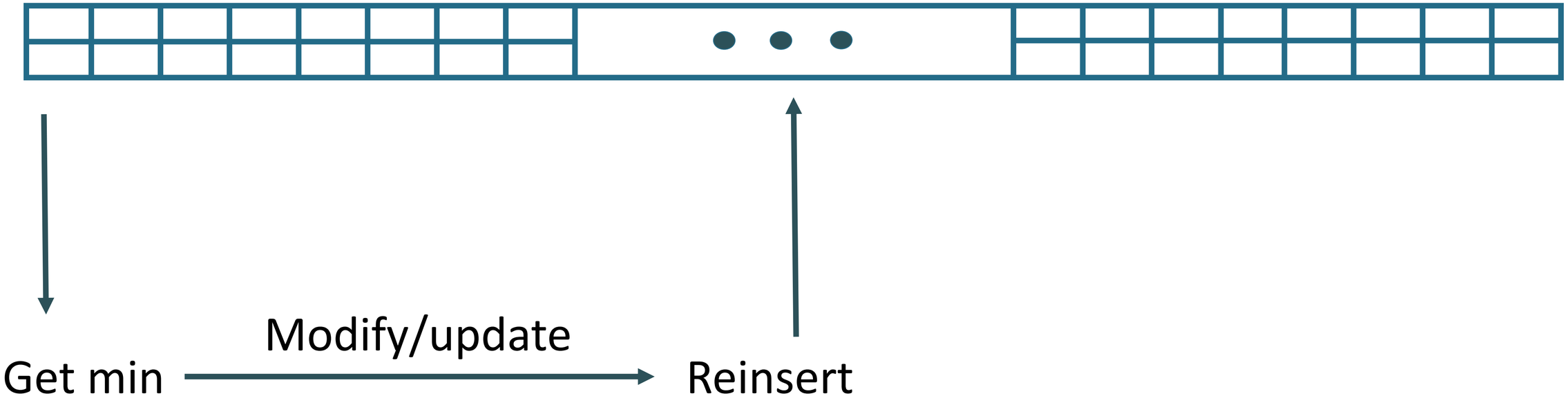
• Insert with priority	O(n log(n))*	not cheap
• Get min	O(1)	≈free

*can do a O(n) vector implementation in R - which would be quicker (but still terrible)

Example: priority queue

```
typedef std::pair<int,double> pqpair;

class Compare
{
  public:
    bool operator() (pqpair a, pqpair b)
    {
      return (a.second < b.second);
    }
};

std::priority_queue< pqpair, std::vector<pqpair>, Compare >  q;

// [[Rcpp::export]]
void add_to_queue(int index, double value){
  std::pair<int, double> v(index, value);
  q.push(v);
}

// [[Rcpp::export]]
int pop_queue(){
  std::pair<int, double> res = q.top();
  int v = res.first;
  q.pop();
  return (v);
}
```

Example: priority queue

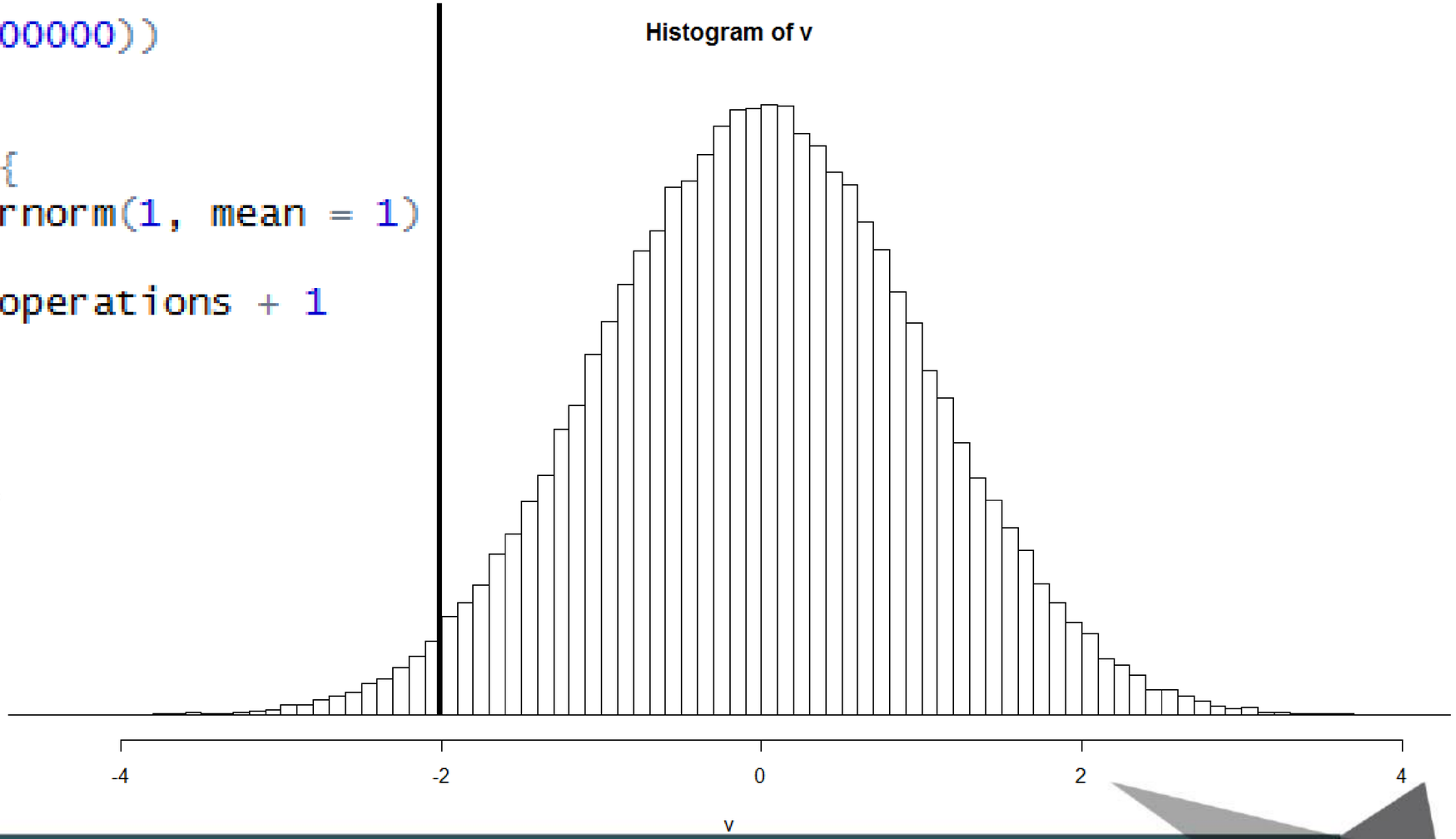


Example: priority queue

```
v<- sort(rnorm(100000))

operations<- 0
while(v[1] < -2){
  v[1]<- v[1] + rnorm(1, mean = 1)
  v<- sort(v)
  operations <- operations + 1
}
```

3600 iterations

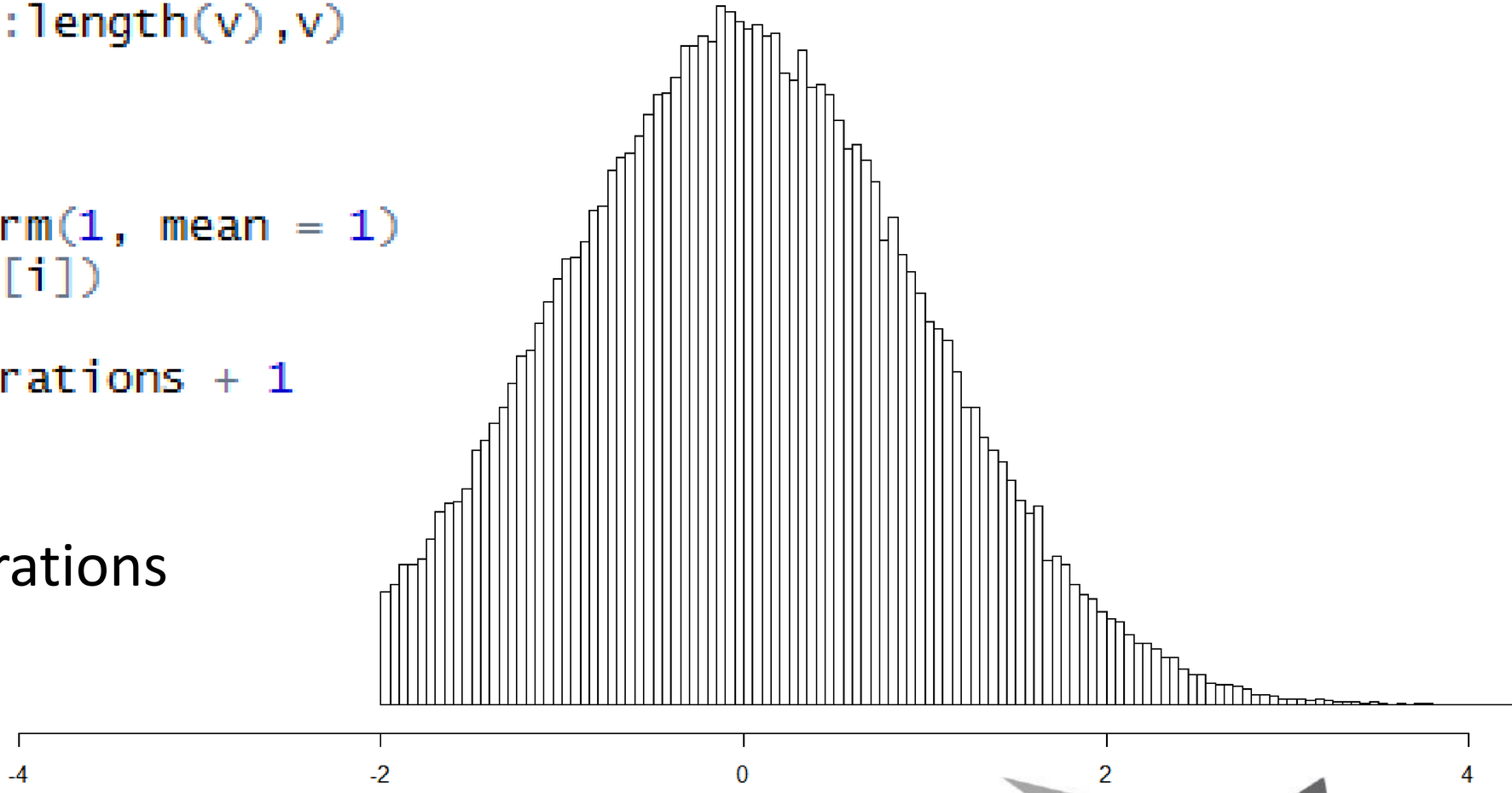
7.8 seconds ☹



Example: priority queue

```
v<- rnorm(100000)
add_to_queue_many(1:length(v),v)
i<- pop_queue()
operations<- 0
while(v[i] < -2){
  v[i]<- v[i] + rnorm(1, mean = 1)
  add_to_queue(i, v[i])
  i<- pop_queue()
  operations <- operations + 1
}
```

Same number of iterations



Example: priority queue

```
> system.time(pq_example())
   user  system elapsed
   6.75    1.53    8.37
```

```
> system.time(pq_example_fast())
   user  system elapsed
   0.05    0.00    0.04
```

Priority Queue Applied - Kaggle

150,000 points

n^2 operations should be avoided

Some alternatives

Farthest first, cheapest insertion, nearest neighbour etc.

- O$(n^2 \log(n))$
- 280,000,000,000 operations.

Priority queue to the rescue

- Start at a point on the hull (easy to find)
- Build your way out from there, maintain a frontier of the next cheapest point to insert

- Can test using both native-R and using priority queue

Discussion

- Easier than you would expect.
- Good enough for Hadley – Good enough for you!
- Deployable within SparkR, RServer, Cluster processes

PIVOT SCIENCES

robert.bennetto@pivotsciences.com