When R could not impress you, then?

----Some tools may help you in statistical genetics related computational problems

Chatterjee Lab Meeting

Ruzhang Zhao

Octorber 1st, 2021

R is excellent

- A language that was developed for data analysis, statistical modeling, simulation and graphics

- Packages
- Interactive code
- R studio

My favorite two features of R

- 1. R’s unique equation: BMI ~ SNP+Height+Age

- 2. R’s vector-oriented: mean_age<-mean(age)

R is not good enough!

- Slow!
- Not good at tasks other than data analysis, statistical modeling, simulation and graphics

Why is R so slow?

- It is designed to be slow!

- Computer vs Human

Why C is so fast?

- Trust the programmer

  C does not check the array
- Don’t stop programmer doing anything
- Keep language simple and small
- Even not convenient or not safe, keep it fast

Python is excellent!

- Readable
- So many developers
- Heavily used in programmers for artificial intelligence

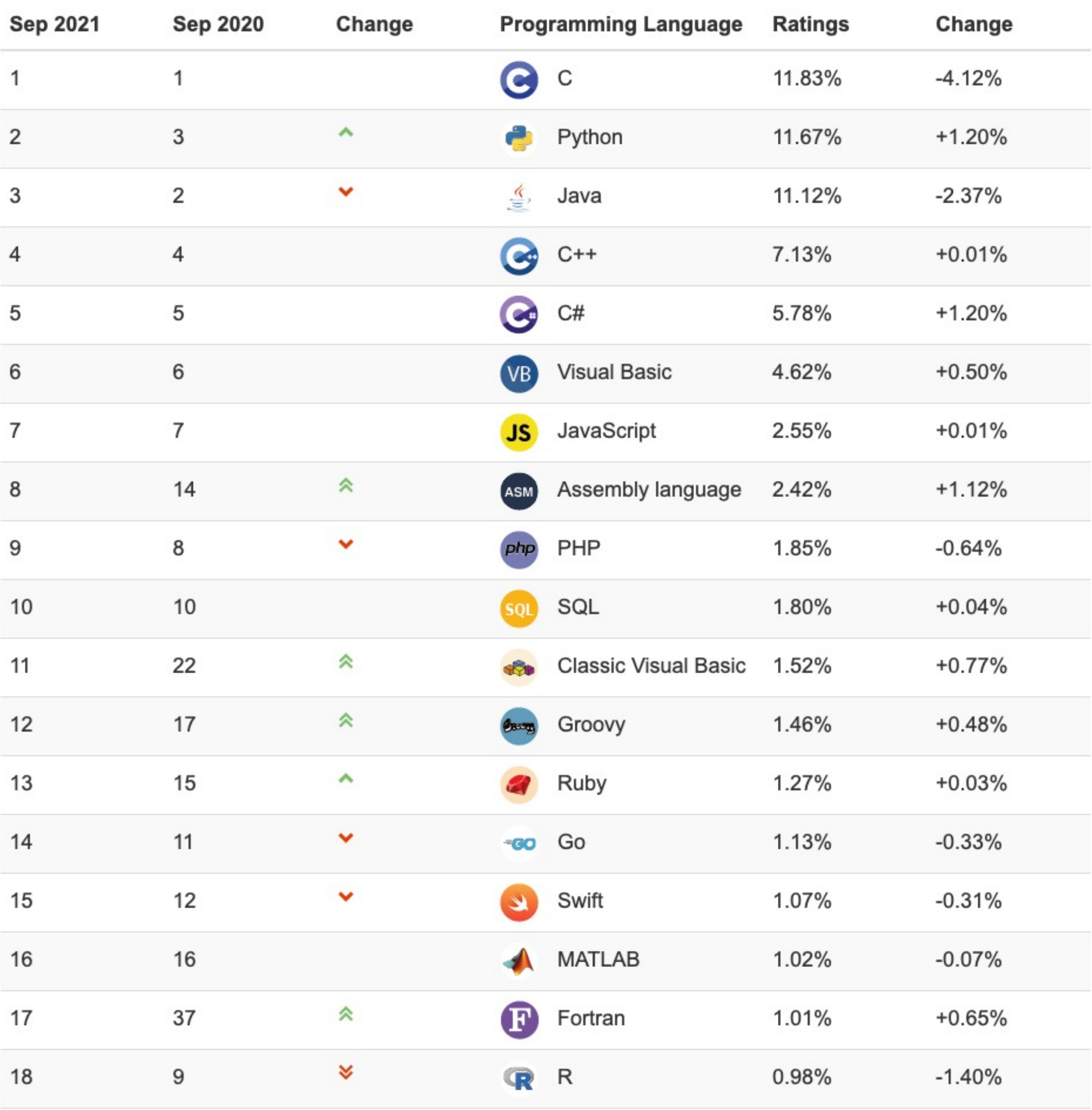
Python is not good enough!

- Much worse at plotting compared with R
- Not heavily used by statistical community compared with R

Why more people use Python than R?

- Programming languages grows faster when a corporate sponsor backs it. For example, PHP is backed by Facebook, Java by Oracle and Sun, Visual Basic & C# by Microsoft. Python Programming language is heavily backed by Facebook, Amazon Web Services, and especially Google.

TIOBE Index: Programming Language Rank September 2021



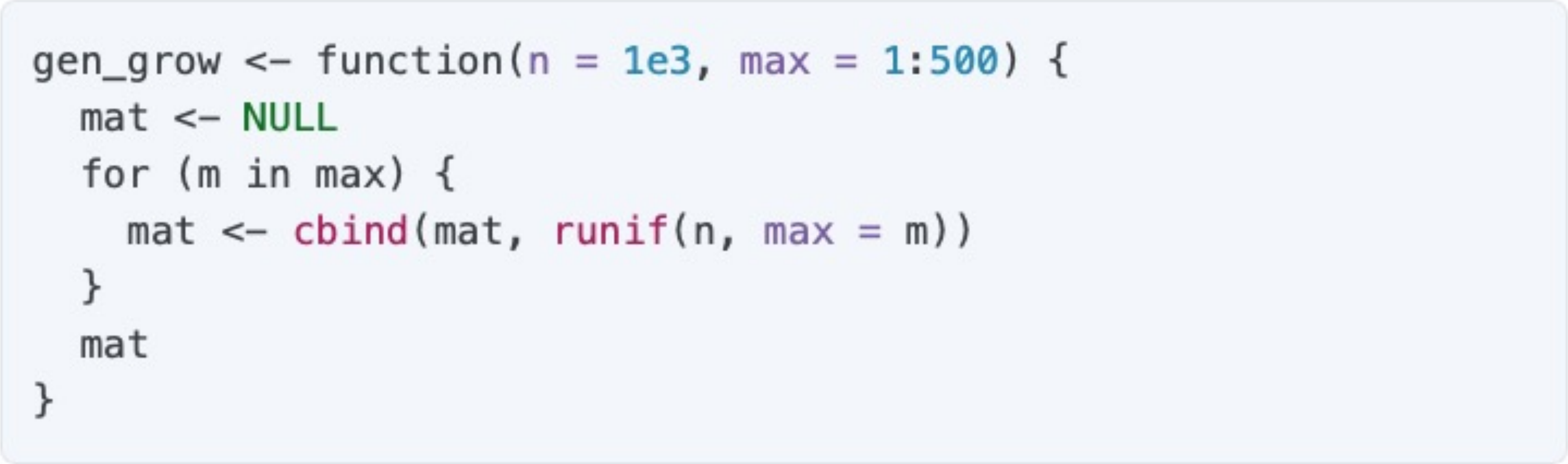
Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	↑	Python	11.67%	+1.20%
3	2	↓	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	↓	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⇈	Classic Visual Basic	1.52%	+0.77%
12	17	⇈	Groovy	1.46%	+0.48%
13	15	↑	Ruby	1.27%	+0.03%
14	11	↓	Go	1.13%	-0.33%
15	12	↓	Swift	1.07%	-0.31%
16	16		MATLAB	1.02%	-0.07%
17	37	⇈	Fortran	1.01%	+0.65%
18	9	⇊	R	0.98%	-1.40%

Share Some Tricks in R

- Loop ? Extremely Slow, may be stuck. TRUE? FALSE?
- sapply, lapply works much better than loop! TRUE? FALSE?

Share Some Tricks in R

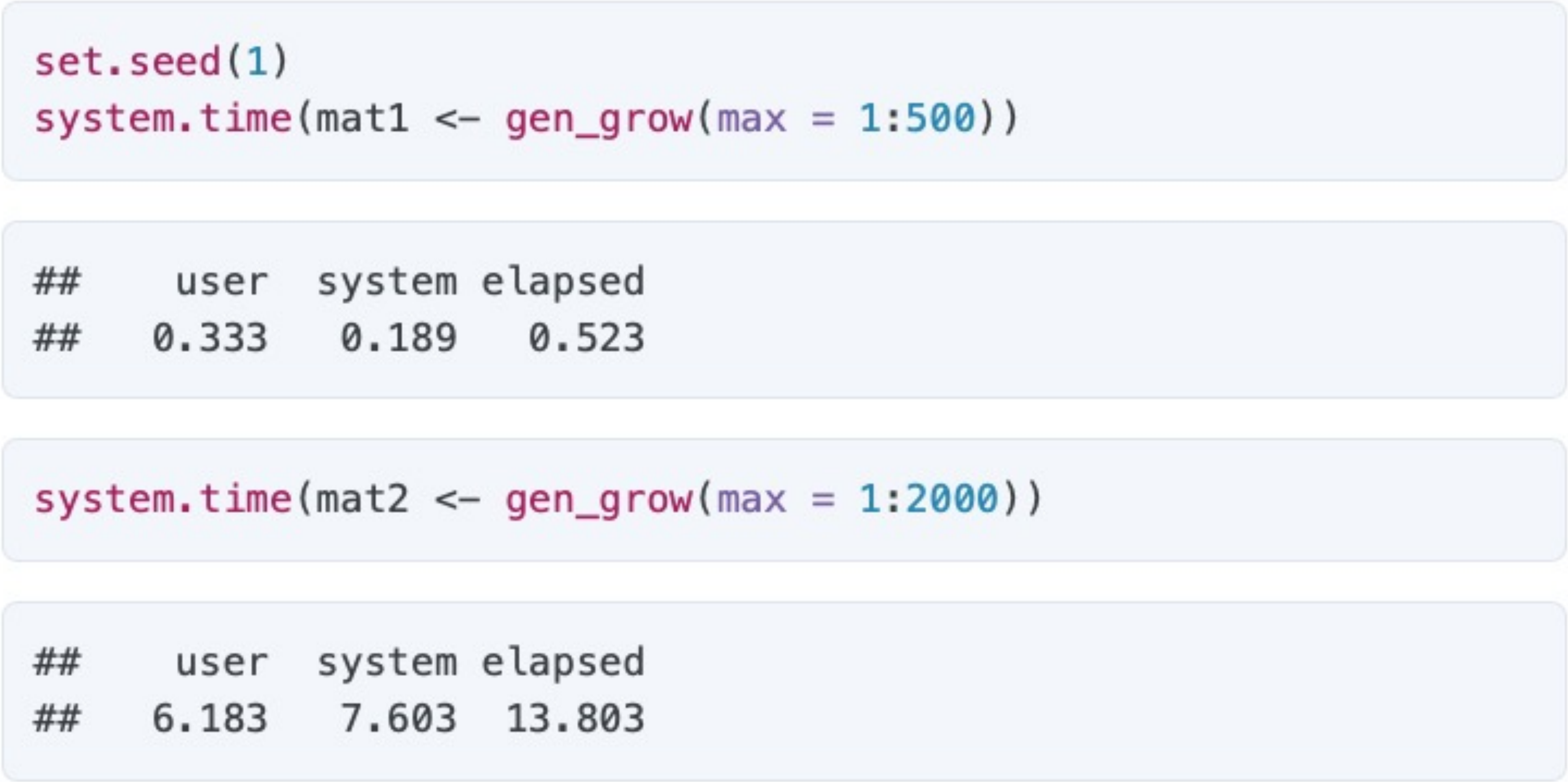
- Loop



```
gen_grow <- function(n = 1e3, max = 1:500) {
  mat <- NULL
  for (m in max) {
    mat <- cbind(mat, runif(n, max = m))
  }
  mat
}
```

Share Some Tricks in R

- Loop



```
set.seed(1)
system.time(mat1 <- gen_grow(max = 1:500))
```

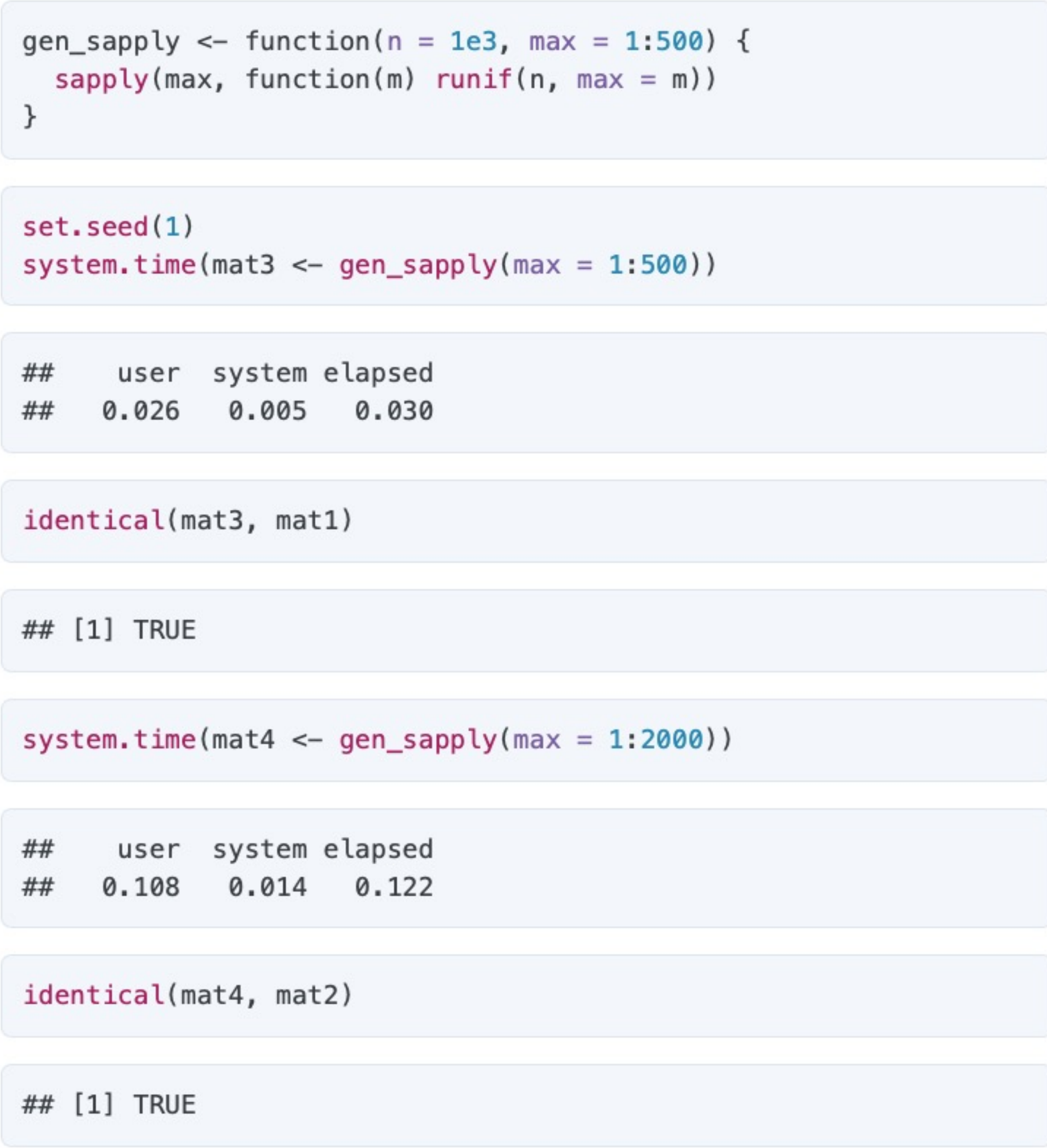
```
##    user  system elapsed 
##   0.333   0.189   0.523
```

```
system.time(mat2 <- gen_grow(max = 1:2000))
```

```
##    user  system elapsed 
##   6.183   7.603  13.803
```

Share Some Tricks in R

- sapply



```
gen_sapply <- function(n = 1e3, max = 1:500) {
  sapply(max, function(m) runif(n, max = m))
}
```

```
set.seed(1)
system.time(mat3 <- gen_sapply(max = 1:500))
```

```
##    user  system elapsed
##   0.026   0.005   0.030
```

```
identical(mat3, mat1)
```

```
## [1] TRUE
```

```
system.time(mat4 <- gen_sapply(max = 1:2000))
```

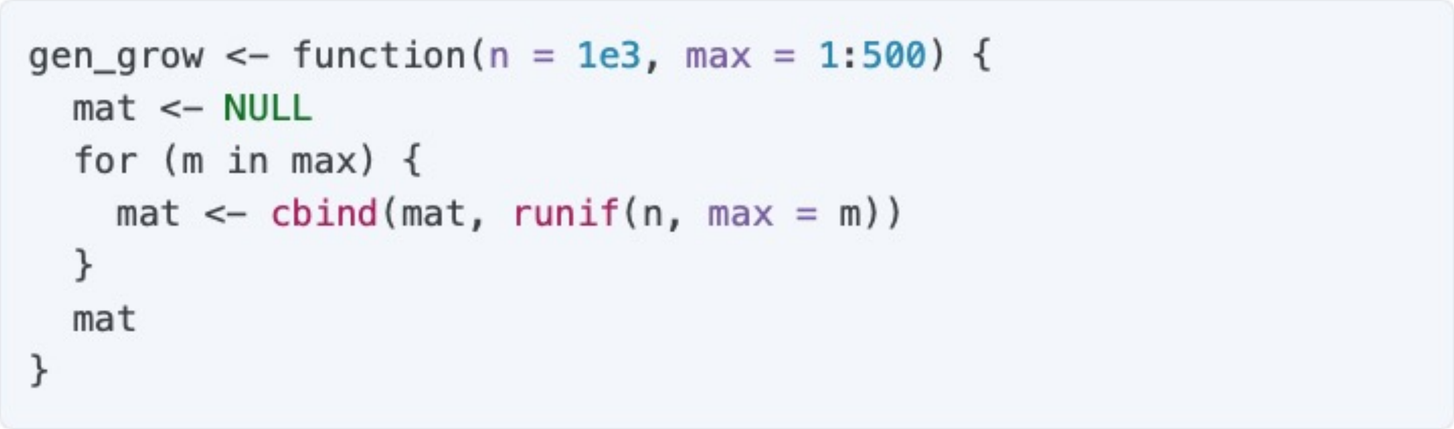
```
##    user  system elapsed
##   0.108   0.014   0.122
```

```
identical(mat4, mat2)
```

```
## [1] TRUE
```

Share Some Tricks in R

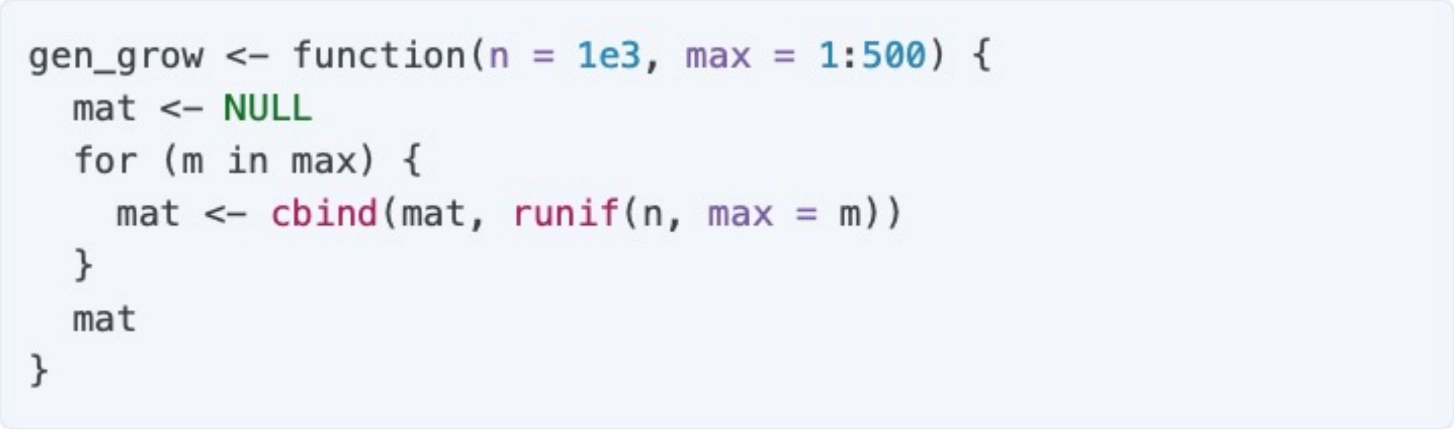
- Loop ? Extremely Slow, may be stuck. TRUE? FALSE?
- sapply, lapply works much better than loop! TRUE? FALSE?



```
gen_grow <- function(n = 1e3, max = 1:500) {
  mat <- NULL
  for (m in max) {
    mat <- cbind(mat, runif(n, max = m))
  }
  mat
}
```

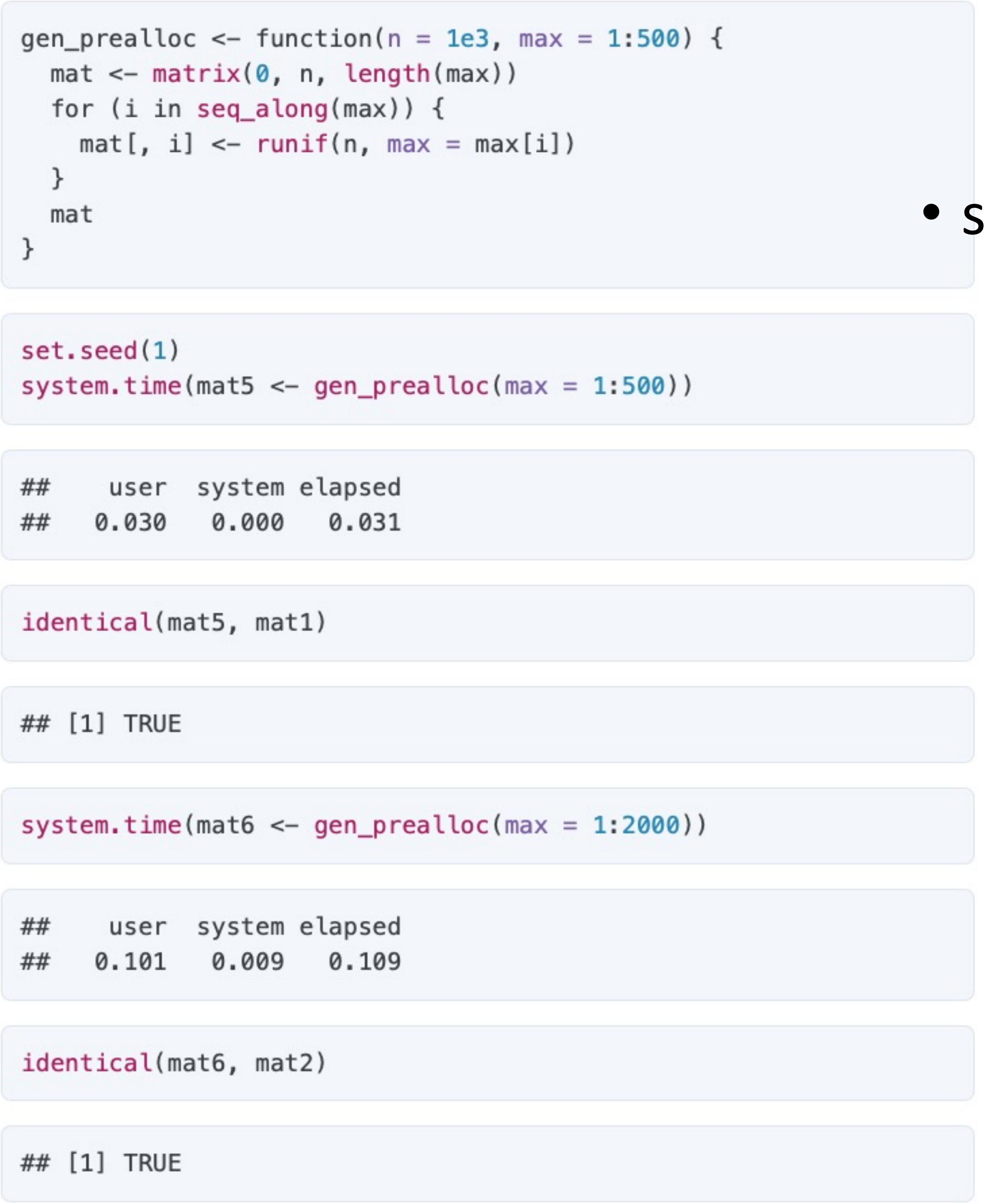
Share Some Tricks in R

- Loop ? Extremely Slow, may be stuck. TRUE
- sapply, lapply works much better than loop!



```
gen_grow <- function(n = 1e3, max = 1:500) {
  mat <- NULL
  for (m in max) {
    mat <- cbind(mat, runif(n, max = m))
  }
  mat
}
```

- Loop



```r
gen_prealloc <- function(n = 1e3, max = 1:500) {
  mat <- matrix(0, n, length(max))
  for (i in seq_along(max)) {
    mat[, i] <- runif(n, max = max[i])
  }
  mat
}
```

```r
set.seed(1)
system.time(mat5 <- gen_prealloc(max = 1:500))
```

```
##    user  system elapsed
##   0.030   0.000   0.031
```

```r
identical(mat5, mat1)
```

```
## [1] TRUE
```

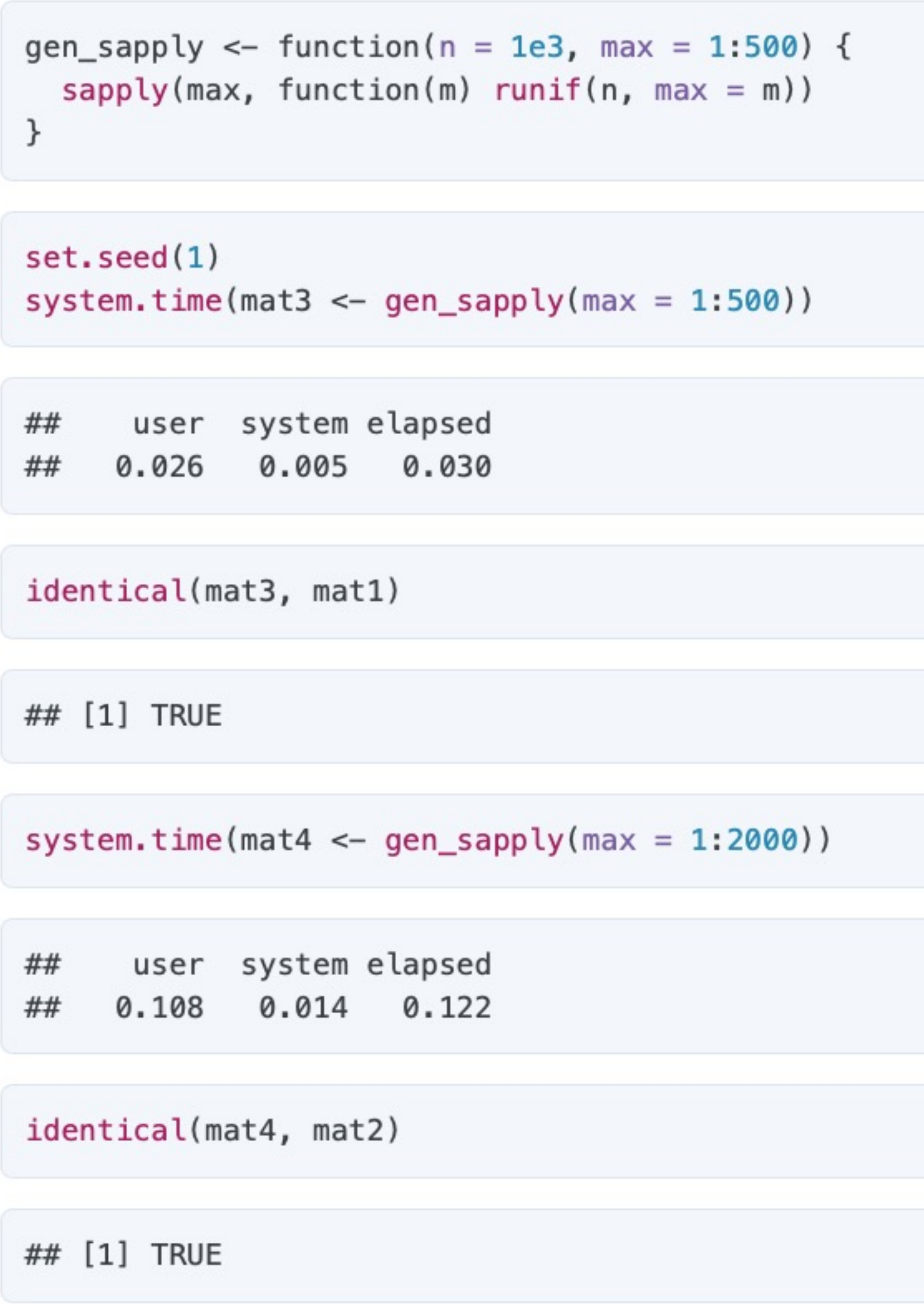
```r
system.time(mat6 <- gen_prealloc(max = 1:2000))
```

```
##    user  system elapsed
##   0.101   0.009   0.109
```

```r
identical(mat6, mat2)
```

```
## [1] TRUE
```

- sapply



```r
gen_sapply <- function(n = 1e3, max = 1:500) {
  sapply(max, function(m) runif(n, max = m))
}
```

```r
set.seed(1)
system.time(mat3 <- gen_sapply(max = 1:500))
```

```
##    user  system elapsed
##   0.026   0.005   0.030
```

```r
identical(mat3, mat1)
```

```
## [1] TRUE
```

```r
system.time(mat4 <- gen_sapply(max = 1:2000))
```

```
##    user  system elapsed
##   0.108   0.014   0.122
```

```r
identical(mat4, mat2)
```

```
## [1] TRUE
```

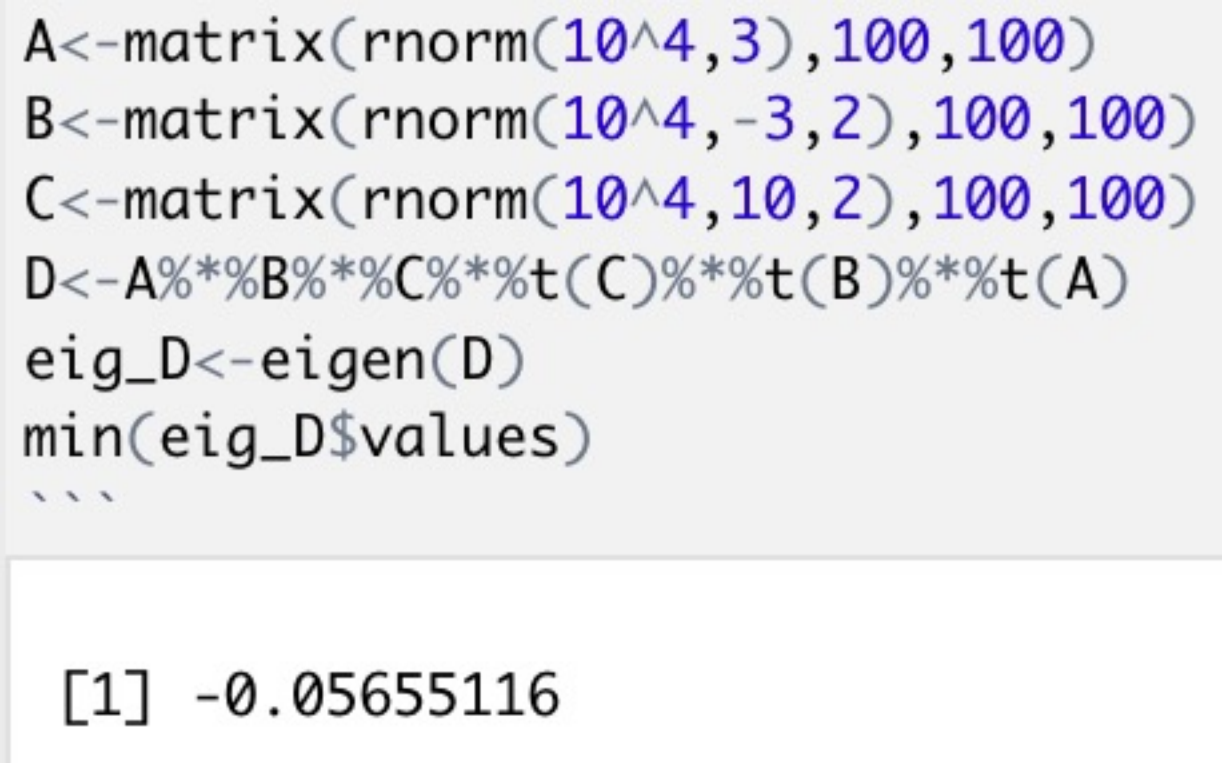
Share Some Tricks in R

- Loop ? Extremely Slow, may be stuck. FALSE
- sapply, lapply works much better than loop!

- The thing we do in loop make it slow!

Share Some Tricks in R (2)

- Matrix Multiplication
- A, B, C are full rank matrices
- Will A%*%B%*%C%*%t(C)%*%t(B)%*%t(A) be positive definite?

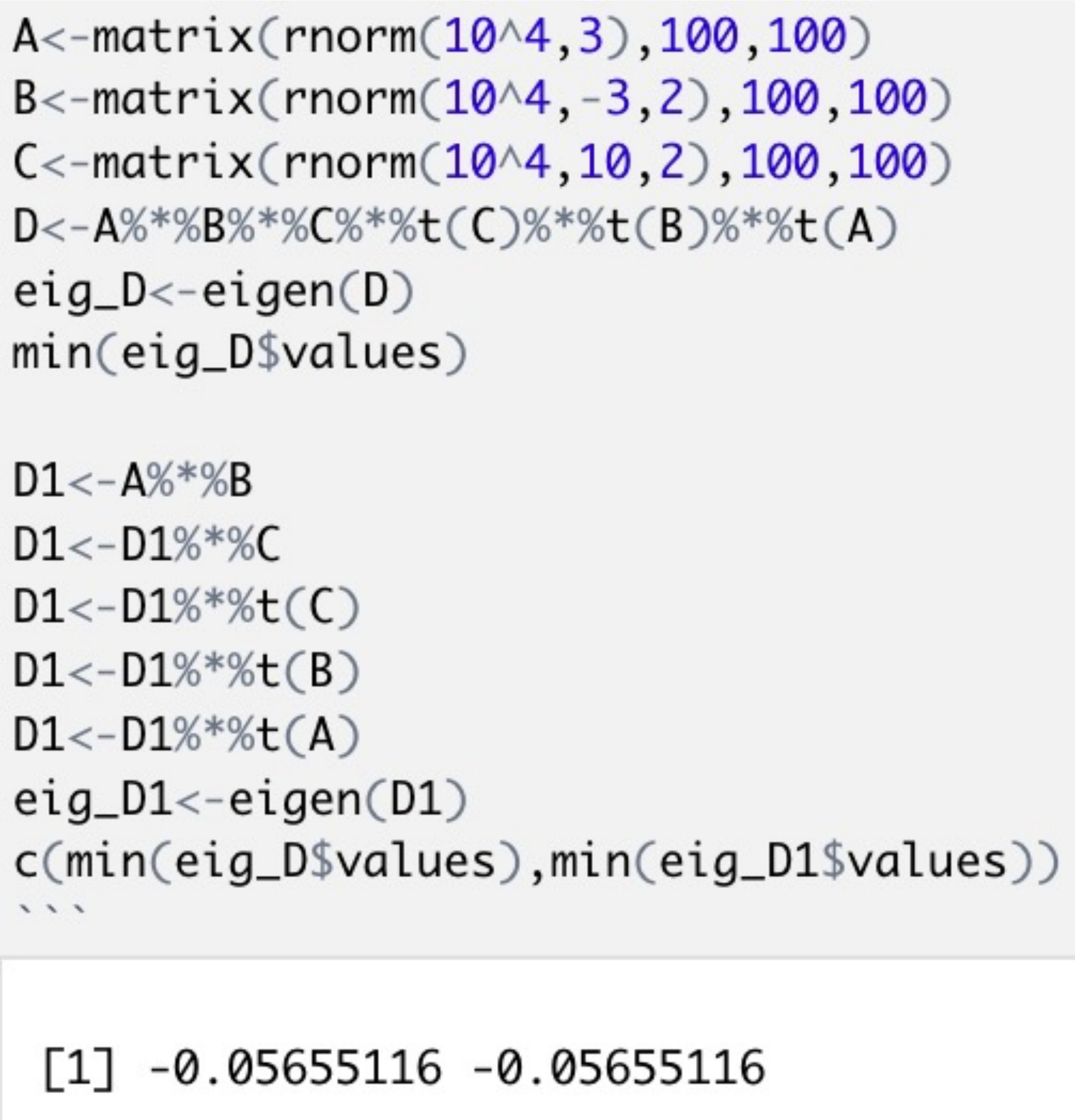


```
A<-matrix(rnorm(10^4,3),100,100)
B<-matrix(rnorm(10^4,-3,2),100,100)
C<-matrix(rnorm(10^4,10,2),100,100)
D<-A%*%B%*%C%*%t(C)%*%t(B)%*%t(A)
eig_D<-eigen(D)
min(eig_D$values)
```
```

```
[1] -0.05655116
```
```

Share Some Tricks in R

- Matrix Multiplication



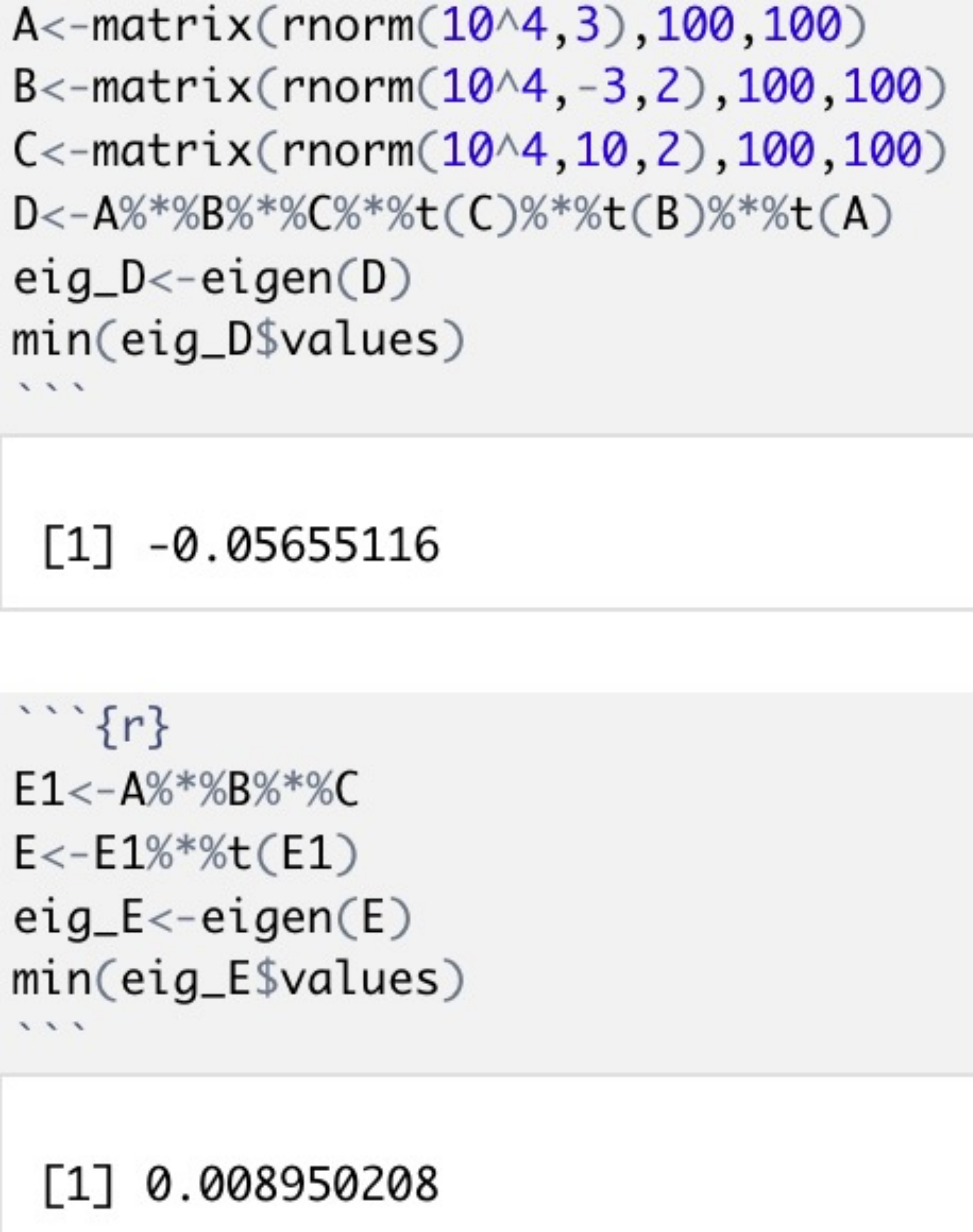
~~~
A<-matrix(rnorm(10^4,3),100,100)
B<-matrix(rnorm(10^4,-3,2),100,100)
C<-matrix(rnorm(10^4,10,2),100,100)
D<-A%*%B%*%C%*%t(C)%*%t(B)%*%t(A)
eig_D<-eigen(D)
min(eig_D$values)

D1<-A%*%B
D1<-D1%*%C
D1<-D1%*%t(C)
D1<-D1%*%t(B)
D1<-D1%*%t(A)
eig_D1<-eigen(D1)
c(min(eig_D$values),min(eig_D1$values))
```
~~~

~~~
[1] -0.05655116 -0.05655116
~~~

Share Some Tricks in R

- Matrix Multiplication



```
A<-matrix(rnorm(10^4,3),100,100)
B<-matrix(rnorm(10^4,-3,2),100,100)
C<-matrix(rnorm(10^4,10,2),100,100)
D<-A%*%B%*%C%*%t(C)%*%t(B)%*%t(A)
eig_D<-eigen(D)
min(eig_D$values)
```

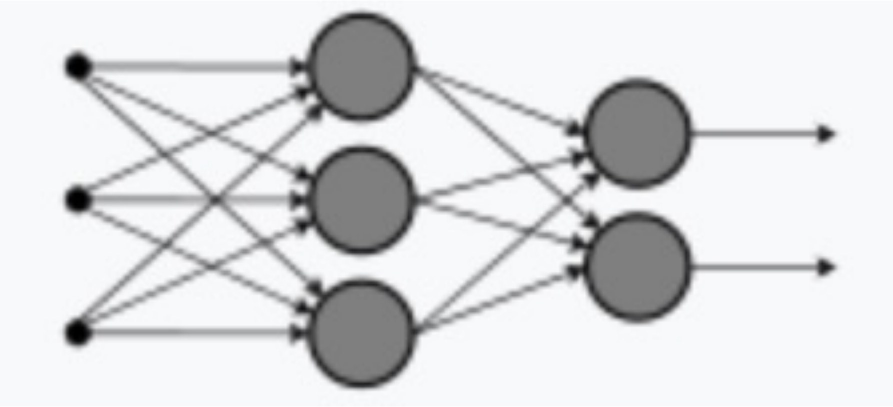
```
 [1] -0.05655116
```

~~~
```{r}
E1<-A%*%B%*%C
E<-E1%*%t(E1)
eig_E<-eigen(E)
min(eig_E$values)
~~~

```
 [1] 0.008950208
```

Deep Learning

- A kind of Machine Learning.
- A deep neural network (DNN) is an <u>artificial neural network</u> (ANN) with multiple layers between the input and output layers.



Deep Learning Toolbox

- Python

Deep Learning Toolbox

- Python
- TensorFlow

Deep Learning Toolbox

- Python
- TensorFlow
- PyTorch

Deep Learning Toolbox

- Optimization in R: “optim”: write in C
- Researcher can be hard to beat Big Tech Company in this aspect
- CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing
- Easy to do parallel computation.

Optimizer

- Use the Optimizer in PyTorch

- Stochastic Gradient Descent(SGD), RMSprop, Adam, L-BFGS

Gradient Descent

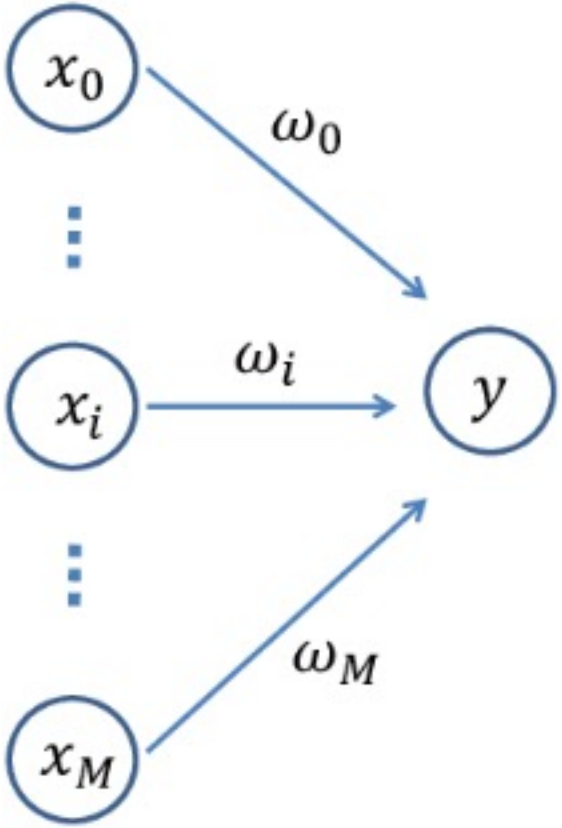
- Stochastic Gradient Descent(SGD) vs Gradient Descent
- Batch Version vs Non batch version

Converting optimization problem to be a neural network

- Example: minimize $[(x^\top \beta) - 100]^2$

  $x = (1 : 100)/10$

- x as an input of the neural network
- Y is the target function

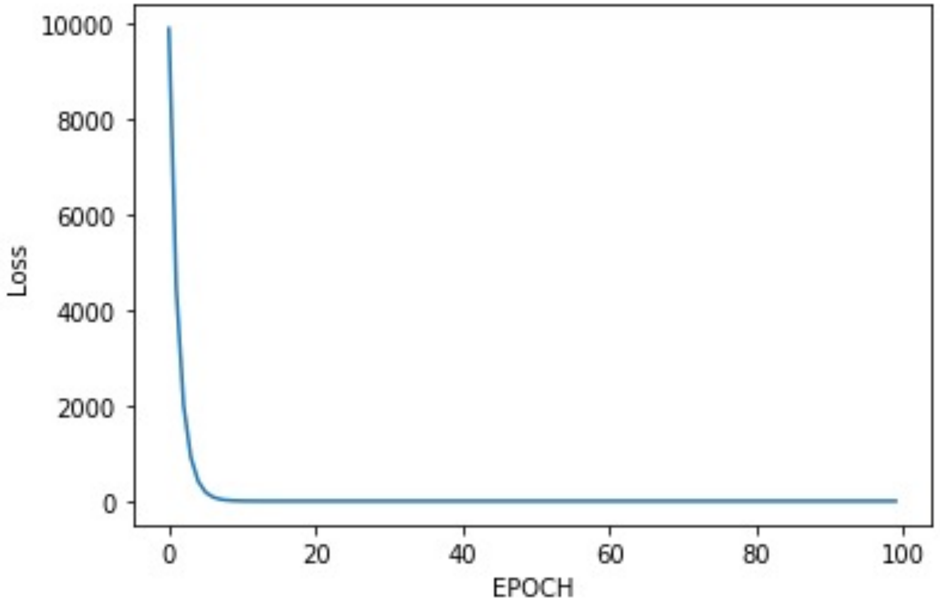


Converting optimization to be a neural network

- Example: minimize $[(x^\top \beta) - 100]^2$

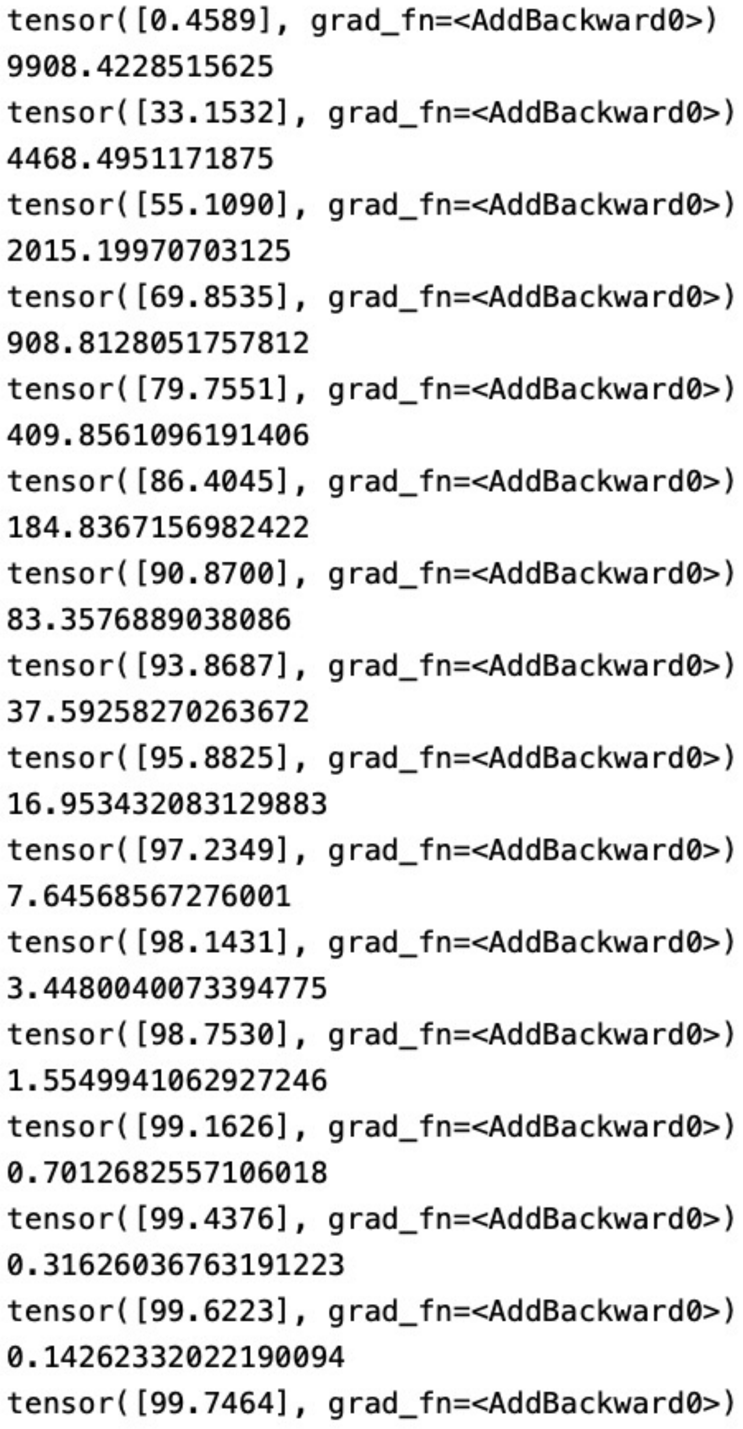
  $x = (1:100)/10$

- Loss Change with SGD optimizer



$(x^\top \beta)$

Loss



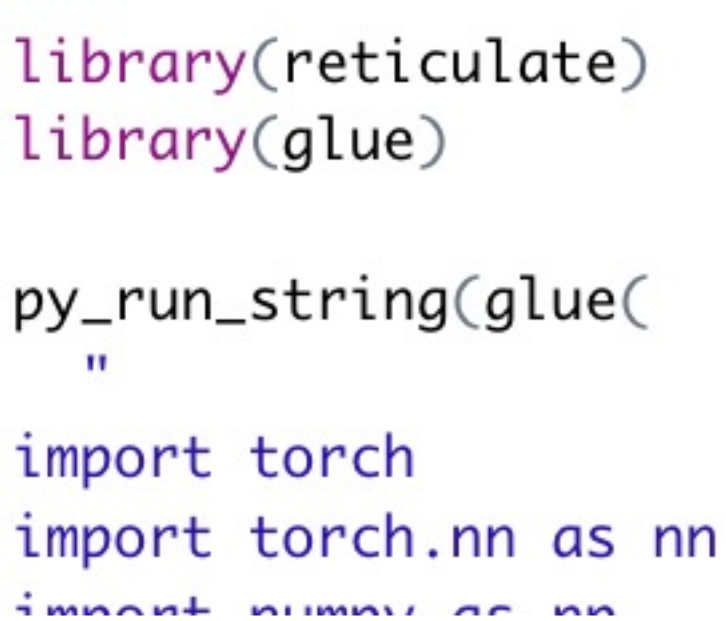
```
tensor([0.4589], grad_fn=<AddBackward0>)
9908.4228515625
tensor([33.1532], grad_fn=<AddBackward0>)
4468.4951171875
tensor([55.1090], grad_fn=<AddBackward0>)
2015.19970703125
tensor([69.8535], grad_fn=<AddBackward0>)
908.8128051757812
tensor([79.7551], grad_fn=<AddBackward0>)
409.8561096191406
tensor([86.4045], grad_fn=<AddBackward0>)
184.8367156982422
tensor([90.8700], grad_fn=<AddBackward0>)
83.3576889038086
tensor([93.8687], grad_fn=<AddBackward0>)
37.59258270263672
tensor([95.8825], grad_fn=<AddBackward0>)
16.953432083129883
tensor([97.2349], grad_fn=<AddBackward0>)
7.64568567276001
tensor([98.1431], grad_fn=<AddBackward0>)
3.4480040073394775
tensor([98.7530], grad_fn=<AddBackward0>)
1.5549941062927246
tensor([99.1626], grad_fn=<AddBackward0>)
0.7012682557106018
tensor([99.4376], grad_fn=<AddBackward0>)
0.31626036763191223
tensor([99.6223], grad_fn=<AddBackward0>)
0.14262332022190094
tensor([99.7464], grad_fn=<AddBackward0>)
```

Run Python in R

- Interface for different programming language

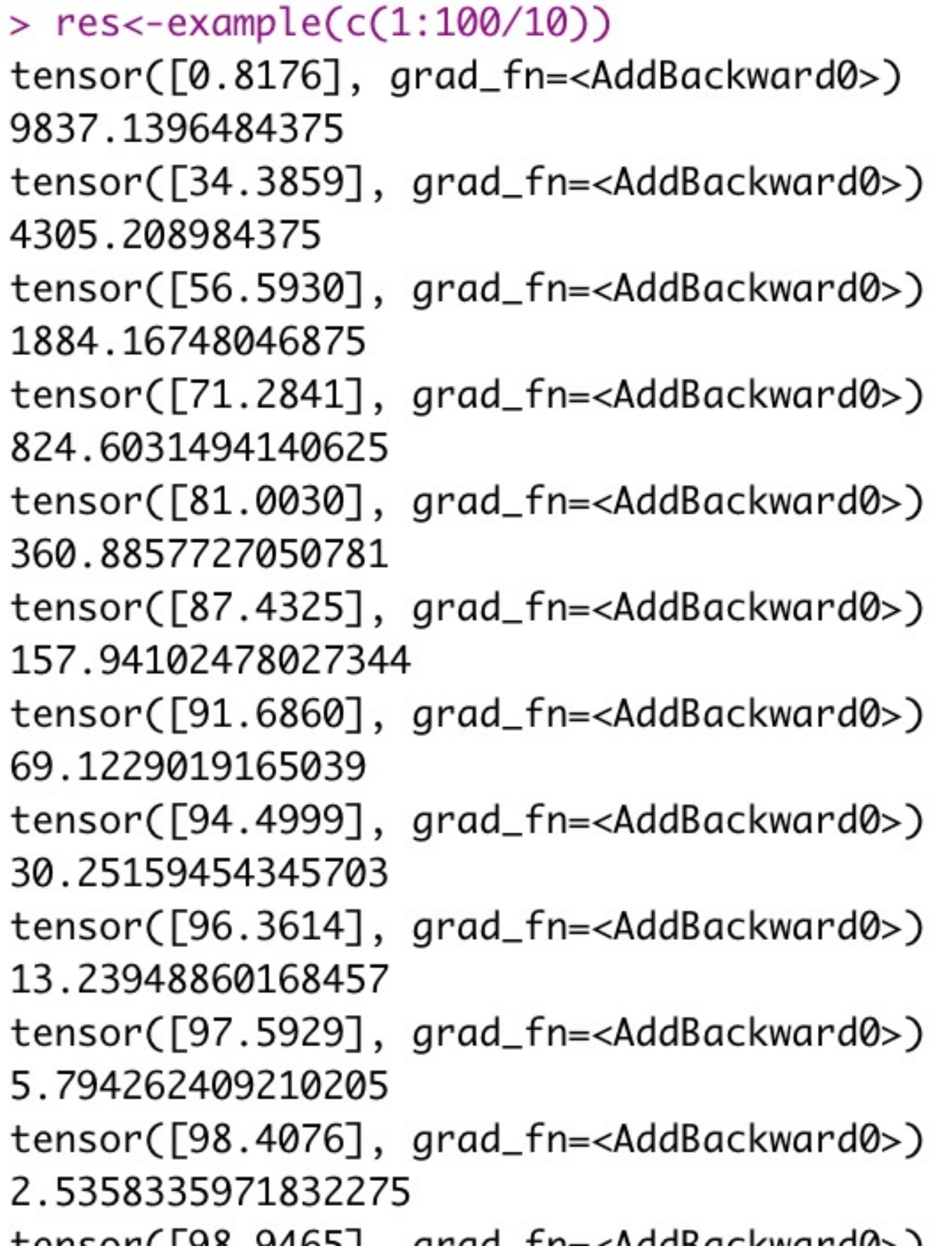
- R package reticulate

Get the function in R



```
library(reticulate)
library(glue)

py_run_string(glue(
  "
import torch
import torch.nn as nn
import numpy as np
```



```
> res<-example(c(1:100/10))
tensor([0.8176], grad_fn=<AddBackward0>)
9837.1396484375
tensor([34.3859], grad_fn=<AddBackward0>)
4305.208984375
tensor([56.5930], grad_fn=<AddBackward0>)
1884.16748046875
tensor([71.2841], grad_fn=<AddBackward0>)
824.6031494140625
tensor([81.0030], grad_fn=<AddBackward0>)
360.8857727050781
tensor([87.4325], grad_fn=<AddBackward0>)
157.94102478027344
tensor([91.6860], grad_fn=<AddBackward0>)
69.1229019165039
tensor([94.4999], grad_fn=<AddBackward0>)
30.25159454345703
tensor([96.3614], grad_fn=<AddBackward0>)
13.23948860168457
tensor([97.5929], grad_fn=<AddBackward0>)
5.794262409210205
tensor([98.4076], grad_fn=<AddBackward0>)
2.5358335971832275
tensor([98.9465], grad_fn=<AddBackward0>)
```

```
□ example    python… 1    18 KB  <function example a…
```