
okapi.math.unr.edu
24 cores
2 GPUs
384 GB RAM
20 TB HD
TESLA
TESLA


caprine.math.unr.edu
128 cores
2 GPUs
512 GB RAM
20 TB HD

Okapi and Caprine are department servers that

- are available to all graduate students and faculty.
- can be used for small computational runs.
- provide a Linux software environment.
- help learn about HPC and statistical simulation.

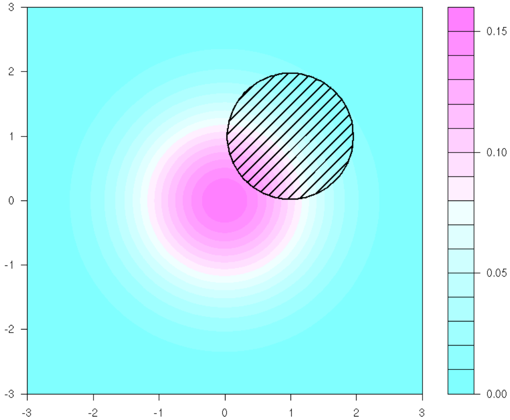
Goal:

- Learn how to use okapi.

How?

- Attend the graduate student seminar.
- Consider a simple computation.
- Watch someone run it.
- Try it yourself.

Problem: Let P be the standard normal probability measure on $\mathbf{R}^2$ and A be the circle of radius r with center v. Find $P(A)$.



Problem: Let P be the standard normal probability measure on $\mathbf{R}^2$ and A be the circle of radius r with center v. Find $P(A)$.

Thus

$$A = \{\, x \in \mathbf{R}^2 : \|x - v\| < r \,\}$$

and

$$P(A) = \int_A e^{-\frac{1}{2}\|x\|^2}\, dx.$$

Computation: Independently sample a bunch of points in $\mathbf{R}^2$ and then count how many of those points lie in A.

```
r <- 1
v <- c(1,1)
inA <- function(x) sum((x-v)^2)<r^2
N <- 100000
X <- matrix(rnorm(2*N),N)
C <- sum(as.integer(apply(X,1,inA)))
cat(sprintf("P(A)=%g\n",C/N))
```

Computation: Independently sample a bunch of points in $\mathbf{R}^2$ and then count how many of those points lie in A.

```
$ Rscript sim.R
P(A)=0.18022
$ Rscript sim.R
P(A)=0.18151
$ Rscript sim.R
P(A)=0.17964
$ Rscript sim.R
P(A)=0.18023
```

Each simulation took 1/2 second on my notebook, but the approximations only agree to a couple digits.

- Use a better method to approximate $P(A)$.

Sometimes the best method still takes a long time.

- Scale up a simulation using a server.
- Many cores are available.
- Can run for days without problem.
- The laptop doesn't overheat.

Computation: Sample a bunch of points in $\mathbf{R}^2$, count how many lie in A and do this a whole bunch of times.

```
r <- 1
v <- c(1,1)
inA <- function(x) sum((x-v)^2)<r^2
N <- 100000; K <- 1000; T <- 0
for (k in 1:K){
    X <- matrix(rnorm(2*N),N)
    C <- sum(as.integer(apply(X,1,inA)))
    T <- T+C
}
cat(sprintf("P(A)=%g\n",T/N/K))
```

Computation: Sample a bunch of points in $\mathbf{R}^2$, count how many lie in A and do this a whole bunch of times in parallel.

```r
r <- 1
v <- c(1,1)
inA <- function(x) sum((x-v)^2)<r^2
N <- 100000; K <- 1000
C <- foreach(k=1:K,.combine='c') %dopar% {
    X <- matrix(rnorm(2*N),N)
    sum(as.integer(apply(X,1,inA)))
}
cat(sprintf("P(A)=%g\n",sum(C)/N/K))
```

Programming Details: Parallel processing in R requires some setup at the beginning and teardown at the end.

Parallel setup:

```
library("doParallel")
library("foreach")
cluster <- makeCluster(7)
registerDoParallel(cluster)
```

Parallel teardown:

```
stopCluster(cluster)
```

You might need `install.packages` to install the libraries.

Logging in to Okapi: Let's start with something simple and avoid parallel processing and those extra libraries.

Connect with `ssh` or Remote Desktop. For example

```
$ slogin okapi.math.unr.edu
ejolson@okapi.math.unr.edu's password:
------------------------------------------------------------------------
 Dual Intel Xeon 6126 Gold 2.60GHz/384GB             okapi.math.unr.edu
                                                      @__@
 Welcome to the UNR Mathematics and                   (oo)\
 Statistics Department server!                        |_/\ \________
                                                          \      ===)
 This system based on Void Linux.                          |=-----==|\
 Unauthorized use prohibited.                              ||      ||
                                                           ||      ||
------------------------------------------------------------------------
 Please read the documentation at https://fractal.math.unr.edu/~okapi/
------------------------------------------------------------------------
$ 
```

Submitting a Job on Okapi: Let's start with something simple and avoid parallel processing and those extra libraries.

The batch submission file looks like

```
#!/bin/bash
time Rscript scaled.R
```

Download the files

- `scaled.R` — The non-parallel Monte Carlo code.
- `scaled.slm` — The batch submission file.

from

`https://fractal.math.unr.edu/~okapi/2023/`

Running the Script: Use the `sbatch` command to launch the R script. Then use `squeue` to check if it's running.

```
$ mkdir demo2023
$ cd demo2023
$ wget -q https://fractal.math.unr.edu/~okapi/2023/scaled.R
$ wget -q https://fractal.math.unr.edu/~okapi/2023/scaled.slm
$ ls
scaled.R  scaled.slm
$ sbatch scaled.slm
Submitted batch job 263301
$ squeue
 JOBID             NAME      USER ST        TIME MIN CPU   REASON PARTITION
263301       scaled.slm   ejolson  R        0:02  2G   1     None fast
$ 
```

The script will run for about 6 minutes.

To cancel it type `scancel` n where n is the `JobID`.

Submitting a Parallel Job on Okapi: If there's time we'll try parallel processing and installing those extra libraries.

To install the libraries start R interactively and type

```
$ R
R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing

> install.packages("doParallel")
--- Please select a CRAN mirror for use in this session ---
Selection: 72

> install.packages("foreach")
> quit()
Save workspace image? [y/n/c]: n
$
```

The rest is similar to running the non-parallel code.

Submitting a Parallel Job on Okapi: If there's time we'll try parallel processing and installing those extra libraries.

The batch file looks like

```
#!/bin/bash
#SBATCH -n8
time Rscript parallel.R
```

Note the `-n8` corresponds to `makeCluster(7)` in the R parallel setup as follows.

- For luck the number 8 is one more than 7.

The batch file reserves 8 cores for the job; the R script uses 7 for parallel processing and reserves 1 for everything else.

Running the Parallel Script: Use the `sbatch` command to launch the R script. Then use `squeue` to check if it's running.

```
$ wget -q https://fractal.math.unr.edu/~okapi/2023/parallel.R
$ wget -q https://fractal.math.unr.edu/~okapi/2023/parallel.slm
$ ls
parallel.R  parallel.slm  scaled.R  scaled.slm
$ sbatch parallel.slm
Submitted batch job 263303
$ squeue
 JOBID             NAME      USER ST        TIME MIN CPU   REASON PARTITION
263303     parallel.slm   ejolson  R        0:40  2G   8     None fast
$ 
```

The script will finish in less than a minute. Check the output:

```
$ cat slurm-263303.out
Loading required package: foreach
Loading required package: iterators
Loading required package: parallel
P(A)=0.180611
```