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


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.
- provide a Linux software environment.
- can be used for numerical computation.
- can be used for statistical simulation.

Goal:

- Learn how to use okapi and caprine.

How?

- Consider a simple problem.
- Create a computation to solve it.
- Run the computation on the server.

Problem: Compute $P(|T| > |t_0|)$ where T follows a t-distribution with hypothesized mean μ_0 and t_0 is the t-statistic for the sample. Specifically, we want to compute the integral

$$p = \iint \cdots \int_{|t(x)| \geq |t_0|} f(x, \mu_0, s_0^2) dx_n \cdots dx_2 dx_1$$

where

$$t(x) = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}, \quad \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \quad s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

and $f(x, \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\big(-(x_i - \mu)^2/(2\sigma^2) \big)$.

Problem: Computing the p-value

$$p = \iint \cdots \int_{|t(x)| \geq |t_0|} f(x, \mu_0, s_0^2) dx_n \cdots dx_2 dx_1$$

can be done in the R language using the `pt` function as

$$p = \texttt{2*(1-pt(abs(t0),n-1))}$$

For many of hypothesis tests an explicit formula for p may not be available. In such cases, one can resort to a direct approximation of the integral using Monte Carlo integration.

Computation: Given t_0 and μ_0 independently sample points in $x \in \mathbf{R}^n$ and count how many of those satisfy $|t(x)| > |t_0|$.

```
psim <- function(x,mu0,B=100000) {
    n <- length(x); s0 <- sd(x)
    t0 <- (mean(x)-mu0)/(s0/sqrt(n))
    vx <- matrix(rnorm(B*n,mu0,s0),B,n)
    vxbar <- rowMeans(vx)
    vs <- apply(vx,1,sd)
    vt <- (vxbar-mu0)/(vs/sqrt(n))
    mean(abs(vt)>abs(t0))
}
```

Computation: Given t_0 and μ_0 use the explicit formula to compute p for comparison.

```
pexact <- function(x,mu0) {
    n <- length(x); s0 <- sd(x)
    t0 <- (mean(x)-mu0)/(s0/sqrt(n))
    2*(1-pt(abs(t0),n-1))
}
```

Computation: Given t_0 and μ_0 independently sample points in $x \in \mathbf{R}^n$ and count how many of those satisfy $|t(x)| > |t_0|$.

```
> source("psim.R"); source("pexact.R")
> data=rnorm(30,0,1)
> psim(data,0)
[1] 0.4383
> pexact(data,0)
[1] 0.4402226
> psim(data,0.5)
[1] 0.04237
> pexact(data,0.5)
[1] 0.0433096
```

Computation: Each simulation took 2 seconds, but the approximations only agree with the exact p to a couple digits.

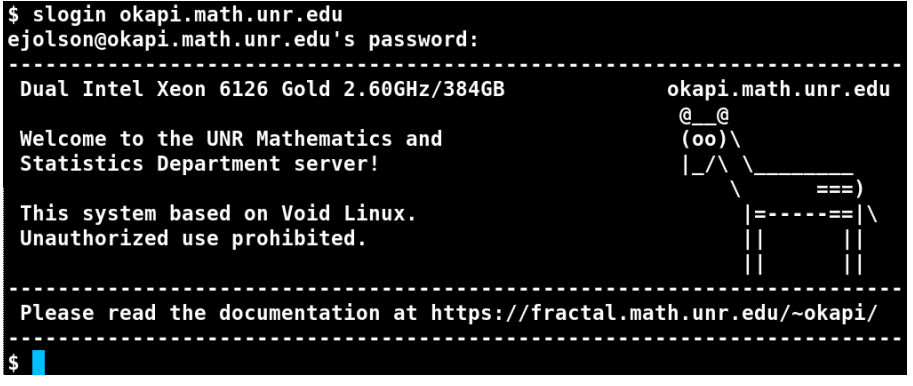
- Use a better method to approximate p.

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.

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

Connect with `ssh` or `putty`. For example



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 single.R
```

Download the files

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

from

`https://fractal.math.unr.edu/~okapi/2026/`

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

```
$ cd demo2026
$ wget -q http://fractal.math.unr.edu/~okapi/2026/single.R
$ wget -q http://fractal.math.unr.edu/~okapi/2026/single.slm
$ ls
single.R  single.slm
$ sbatch single.slm
Submitted batch job 276612
$ squeue
    JOBID          NAME      USER ST        TIME MIN CPU   REASON PARTITION
   276611 run_main_gpu.  cwingard  R     9:00:02  2G 128     None slow
$
```

The script will run for about 6 minutes.

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

Submitting an Array Job on Okapi: If there's time we'll try parallel array processing.

The batch file `array.slm` looks like

```
#!/bin/bash
#SBATCH -a 1-10
Rscript array.R
```

This batch file submits 10 single processor jobs.

The corresponding `array.R` file looks like

```
source("psim.R")
set.seed(11333)
data <- rnorm(30,0,1)

task_id <- as.numeric(Sys.getenv("SLURM_ARRAY_TASK_ID","1"))
cat(sprintf("task_id=%d\n",task_id))
set.seed(33111+task_id*12345)

p <- psim(data,0)
cat(sprintf("psim(data,0)=%g\n",p))
saveRDS(p,sprintf("output/p%04d.rds",task_id))
```

Gathering the Data: After the all the array jobs have finished gather the data and process it.

The batch file `gather.R` looks like

```
setwd("output")
files <- list.files()
pv <- unlist(lapply(files,readRDS))
cat(sprintf(
    "Gathered estimate p=%g\n",mean(pv)))
```