
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))
}
```

$= \begin{cases} 1 & \text{if } |T_n| > |t_0| \\ 0 & \text{otherwise.} \end{cases}$

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 $|t(x)| > |t_0|$.

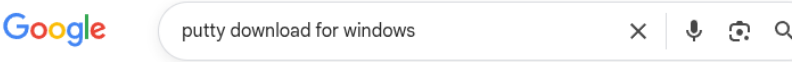
```
> source("psim.R"); source("pexact.R")
> data=rnorm(30,0,1)
> psim(data,0)
[1] 0.4383
> pexact(data,0)
[1] 0.4402226
> psim(data,0.5)
[1] 0.04237
> pexact(data,0.5)
[1] 0.0433096
```

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.



AI Mode | **All** | Videos | Shopping | Images | Forums | Short videos | More | Tools

PuTTY
https://putty.org

Download PuTTY - a free SSH and telnet client for Windows

Download PuTTY. PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is ... Read more

Microsoft Store
https://apps.microsoft.com › detail

PuTTY - Download and install on Windows

PuTTY is a communications tool for running interactive command-line sessions on other computers, usually via the SSH protocol. It can also communicate over ... Read more

Greenend
https://www.chiark.greenend.org.uk › ~sgtatham › latest

Download PuTTY: latest release (0.83)

Jan 11, 2026 — This page contains **download links for the latest released version of PuTTY**. Currently this is 0.83, released on 2025-02-08. Read more

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

Windows

```
$ 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 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`.

```
$ wget -q http://fractal.math.unr.edu/~okapi/2026/psim.R
$ wget -q http://fractal.math.unr.edu/~okapi/2026/pexact.R
```

To quickly check what's in a file type

```
$ cat filename
```

or

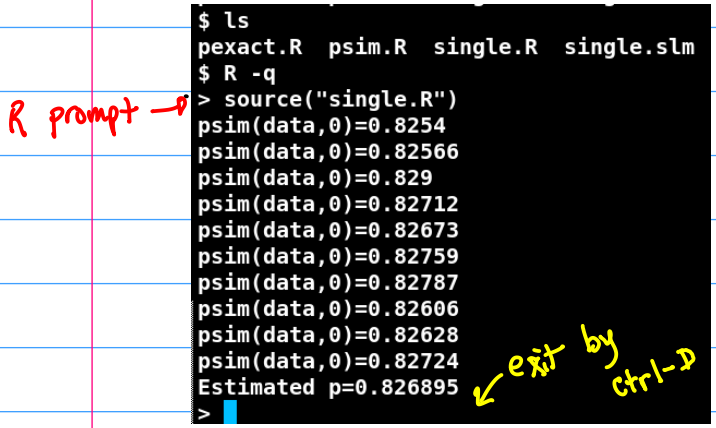
```
$ less filename
```

To edit a file try

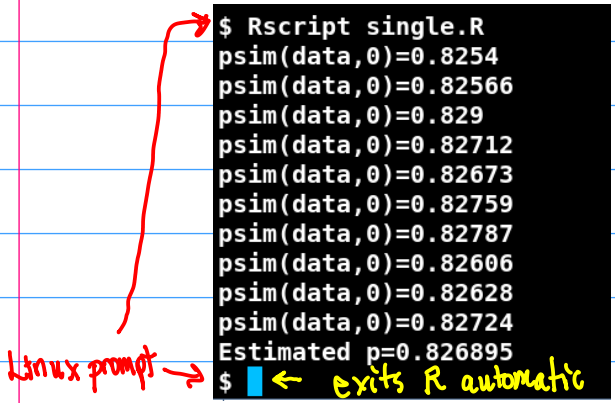
```
$ nano filename
```

Other editors vi and emacs are installed. Some people use the remote editing feature of vscode.

- Run interactively by starting R and then source the file into the environment.



Run as a script from the operating system prompt



Run non-interactively using slurm:

```
$ ls
pexact.R  psim.R  single.R  single.slm
$ sbatch single.slm
Submitted batch job 276631
$ squeue
    JOBID          NAME      USER ST        TIME MIN CPU   REASON PARTIT
N
 276631    single.slm   ejolson  R        0:02  2G   1     None fast
 276611 run_main_gpu.  cwingard  R    12:57:29  2G 128     None slow
$ ls
pexact.R  psim.R  single.R  single.slm  slurm-276631.out
$
```

our JOBID

output from job. To see output type `cat slurm-276631.out`

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.

For this one, make sure to create the output director before lauching the script:

```
$ mkdir output
$ sbatch array.slm
```

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)))
```