Interacting with Remote Systems

Advanced Statistical Programming Camp
Jonathan Olmsted (Q-APS)

Day 2: May 28th, 2014
AM Session
Yesterday . . .

**Topics**

1. monitoring performance
2. vectorization, built-in functions
3. convenience wrappers
4. a new looping construct
5. parallel `foreach`
6. parallel RNG

**Examples**

1. pairwise distance calculation
2. non-parameteric bootstrap confidence intervals (voter turnout)
3. parametric bootstrap confidence intervals (voter turnout)
4. cross-validation (civil war onset)
This session . . .

1. Overview of TIGRESS Systems
2. Interacting with Remote Systems
3. Running Single-Node Jobs on Adroit
4. Group Exercise
TIGRESS

- TIGRESS is a collection of computational resources for Princeton Researchers
  - Computational clusters
  - Visualization hardware
  - Visualization laboratory with visualization wall
    - 4096 x 2160 pixels
    - 150 sq. ft.
## Computational Clusters

<table>
<thead>
<tr>
<th></th>
<th>Processor Speed</th>
<th>Memory per Core</th>
<th>Interconnect</th>
<th>Total Cores</th>
<th>Total RAM</th>
<th>Local Disk</th>
<th>Performance: Theoretical (LINPACK)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dell</strong></td>
<td>2.67 GHz</td>
<td>4 GB / 8 GB</td>
<td>QDR IB</td>
<td>1536</td>
<td>9 TB</td>
<td>137 TB</td>
<td>16 TFLOPS ()</td>
</tr>
<tr>
<td>Dell Linux Cluster</td>
<td>Westmere</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hecate</strong></td>
<td>2.67 GHz</td>
<td>5.3 GB / 10.7 GB</td>
<td>NUMA-link 5</td>
<td>1536</td>
<td>12 TB</td>
<td>180 TB</td>
<td>16 TFLOPS ()</td>
</tr>
<tr>
<td>SGI UV 1000</td>
<td>Westmere</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Orbital</strong></td>
<td>3.47 GHz</td>
<td>4 GB</td>
<td>QDR IB</td>
<td>3696</td>
<td>14 TB</td>
<td>335 TB</td>
<td>50.1 TFLOPS (41.4 TFLOPS)</td>
</tr>
<tr>
<td>HP Linux Cluster</td>
<td>Westmere</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tiger</strong></td>
<td>2.6 GHz Sandybr.</td>
<td>4 GB / 2 MB</td>
<td>FDR IB</td>
<td>10,304</td>
<td>41 TB</td>
<td>64 GB</td>
<td>451 TFLOPS ()</td>
</tr>
<tr>
<td>Dell/SGI Linux Cluster</td>
<td>705 MHZ K20</td>
<td>135 MB</td>
<td></td>
<td>499,200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.053 GHz 5110P</td>
<td></td>
<td></td>
<td>480</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Adroit</strong></td>
<td>2.33 GHz</td>
<td>2 GB</td>
<td>Gig-E</td>
<td>64</td>
<td>128 GB</td>
<td>1 TB</td>
<td>596 GFLOPS ()</td>
</tr>
<tr>
<td>Dell Linux Cluster</td>
<td>Xeon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tigressdata</strong></td>
<td>2.93 GHz</td>
<td>8 GB</td>
<td>N/A</td>
<td>16</td>
<td>128 GB</td>
<td>1.5 TB</td>
<td>188 GFLOPS ()</td>
</tr>
<tr>
<td>Dell Linux Server</td>
<td>X7350 Xeon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nobel</strong></td>
<td>3.33 GHz</td>
<td>8 GB</td>
<td>N/A</td>
<td>24</td>
<td>192 GB</td>
<td>260 GB</td>
<td>320 GFLOPS ()</td>
</tr>
<tr>
<td>Dell Linux Server</td>
<td>X5680 Xeon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Logging In to **Adroit**

- Connect to cluster through the SSH protocol.
  - On Windows: use PuTTY
  - On Mac/Linux: use the terminal and `ssh`
    - `ssh usersystem.princeton.edu`
    - If using a wireless connection (with a laptop), use the SRA client.
Logging In to Adroit

The machine gives you a prompt, waiting for input (like R)
Everything is a file or directory and organized in a hierarchy. Get your current location with `pwd`.  

```
jolmsted@adroit3:~$ pwd
/home/jolmsted
[jolmsted@adroit3 ~]$
```
Interacting with **Adroit**

```bash
[jolmsted@adroit3 ~]$ ls
ASPC          matlab_metadata.mat  ml       old  R  tigress-scripts
ASPC-2014    matlab.mat          mpi_config  pwd  src
[jolmsted@adroit3 ~]$`
```

**ls** lists the contents of the current (working) directory.
Interacting with Adroit

```
jolmsted@adroit3:~ $ mkdir newdir
[jolmsted@adroit3 ~]$  
```

**mkdir** creates a directory.
Interacting with **Adroit**

*rm* removes/deletes a file. *rmdir* removes/deletes a directory.
cd changes the directory to the specified location. cd .. changes the directory up one level.
You can make simple edits to files with Emacs. Start it with `emacs -nw filename`.
Once in Emacs, you can type like you would in a GUI text editor.
Interacting with Adroit

To save, type Ctrl+x Ctrl+s. To quit, type Ctrl+x Ctrl+c.
Interacting with **Adroit**

```bash
[holmsted@adroit3 ~]$ ls
newfile.txt
[holmsted@adroit3 ~]$ cp newfile.txt acopy.txt
[holmsted@adroit3 ~]$ ls
acopy.txt  newfile.txt
[holmsted@adroit3 ~]$ |
```

$cp$ copies files from one location (or file name) to another.
Interacting with **Adroit**

```bash
[jolmsted@adroit3 ASPC]$ ls
acopy.txt  newfile.txt
[jolmsted@adroit3 ASPC]$ mv acopy.txt newname.txt
[jolmsted@adroit3 ASPC]$ ls
newfile.txt  newname.txt
[jolmsted@adroit3 ASPC]$ |
```

`mv` moves/renames files or directories.
Interacting with **Adroit**

```
[jolmsted@adroit3 ASPC-2014]$ cat first.cmd
#!/bin/bash
#SBATCH --nodes 1 # << number of compute nodes
#SBATCH --ntasks-per-node=1 # << number of processors per compute node
#SBATCH -t 00:02:00 # << max time required (hh:mm:ss)
#SBATCH -J cat_ex # << brief, descriptive job label
#SBATCH -o log.%j # << file to save output/error information

cat ~/ASPC-2014/outline.txt # display contents of file
hostname # name of current machine
[jolmsted@adroit3 ASPC-2014]$
```

cat displays the contents of a file.
Interacting with **Adroit**

```
[holmsted@adroit3 ASPC-2014]$ head ovb-logit.R
## OVB Example
## Binomial Data

## ##################
## Function Def
## ##################

genLogitData <- function(.n,
                        .scale = 1,
                        .rho = .5

[holmsted@adroit3 ASPC-2014]$ head displays the beginning of a file.
```
Interacting with Adroit

```
[jolmsted@adroit3 ASPC-2014]$ tail ovb-logit.R
closeCluster(cl)
mpi.quit()
```

tail displays the end of a file.
Submitting Jobs

• With these pieces, we can submit jobs.

• The resources on Adroit (and other systems) use a scheduler (SLURM).

• It’s like taking a ticket at the deli counter.

• Once the machine is ready to run what you’ve requested, your job comes out of the queue.

• Requesting resources requires specific commands and a specific format of the request.
Submitting Jobs

Listing 1: Example SLURM Job Script

```bash
#!/bin/sh
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -t 10:00
#SBATCH -J Ex1
#SBATCH -o log.%j
#SBATCH --mail-type=begin
#SBATCH --mail-type=end

Rscript -e 'a <- rnorm(1e3)'
```
### Submitting Jobs

`sbatch` submits a SLURM script to the queue.
Submitting Jobs

`squeue` displays information about the jobs in the queue.
Submitting Jobs

A busy queue will have a lot of information.
Omitted Variable Bias

In the handout, we demonstrate the effect of omitted variable bias in a linear model.

• Simulate data according to

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon \]  
\[ y \sim x_1 + x_2 + x_3 \]  
\[ \beta_0 = \beta_1 = \beta_2 = \beta_3 = 1 \]

• Estimate the (mis-specified) linear model according to

\[ E[y|x] = \hat{\alpha}_0 + \hat{\alpha}_1 x_1 + \hat{\alpha}_2 x_2. \]  
\[ y \sim x_1 + x_2 \]

• Estimate the correct linear model according to

\[ E[y|x] = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 \]  
\[ y \sim x_1 + x_2 + x_3 \]
Omitted Variable Bias

## Error: object 'form' not found

Canonical result: estimates from the mis-specified model are biased.
Omitted Variable Bias

Computational strategy:

1. Define a function to represent the data-generating process.

2. Define a `data.frame` of test conditions indicating how to generate data and estimate a model with `expand.grid`:
   - Monte Carlo iterations
   - sample size
   - specification

3. Run the simulation with `foreach` in parallel.

4. Collect results in a `data.frame` of output.
Omitted Variable Bias

Practical steps:

1. Write the R code locally.

2. Debug locally using a small scale simulation.

3. Copy files up to Adroit.

4. Submit the job to the scheduler.

5. Copy the output back to the local machine.

6. Summarize and visualize results.
Omitted Variable (Un?)Bias

• It turns out the degree of bias is not actually linked to the “correctness” of the specification in an intuitive way.
• Sometimes, fewer variables are “better”.
• What really matters is the correlation between the included regressors and the omitted regressors.

• **Goal**: Demonstrate this (perhaps) counter-intuitive result using a similar framework with a partner.
Omitted Variable (Un?)Bias

• Simulate data according to

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon \]  \hspace{1cm} (7)

• We already have this function in R.

• Estimate three different specifications:
  1. \( y \sim x_1 + x_2 + x_3 \)
  2. \( y \sim x_1 + x_2 \)
  3. \( y \sim x_1 \)

• You can and should use the framework from handout as a template.
Omitted Variable (Un?)Bias

The data-generating function (from the handout):

```r
genNormData <- function(.n, .scale = 1, .rho = .5) {
  cN <- .n ; cK <- 3
  cRho <- .rho
  cSigma <- diag(cK)
  cSigma[2, 1] <- cRho ; cSigma[1, 2] <- cRho
  cSigma[3, 1] <- -cRho ; cSigma[1, 3] <- -cRho
  mL <- chol(cSigma)
  mZ <- matrix(rnorm(cN * cK, sd = .scale), nrow = cN)
  mX <- cbind(1, mZ %*% mL)
  mE <- matrix(rnorm(cN), nrow = cN)
  vB <- c(1, 1, 1, 1) ; mMu <- mX %*% vB
  vY <- rnorm(cN,
              mean = as.vector(mMu)
            )
  ret <- data.frame(y = vY, mu = mMu,
                     x1 = mX[, 2], x2 = mX[, 3],
                     x3 = mX[, 4]
            )
  return(ret)
}
```

Omitted Variable (Un?)Bias

A Monte Carlo dataset:

```r
head(genNormData(50))
```

```plaintext
##             y     mu     x1     x2     x3
## #1  2.3508 0.1520 -0.5605 -0.0609 -0.2306
## #2  2.2600 0.9466 -0.2302 -0.1398  0.3166
## #3  2.0426 2.3078  1.5687  0.7422 -0.9932
## #4  2.9103 2.3671  0.0705  1.2205  0.0761
## #5 -0.3227 0.0916  0.1302 -0.1309 -0.9068
## #6  3.9531 4.4294  1.7151  2.1708 -0.4565
```
Simulation Conditions:

- Monte Carlo simulations:
  
  ```r
  vMC <- 1:5000  # remotely
  vMC <- 1:20    # locally
  ```

- Sample size:

- Specification:
  
  ```r
  f1 <- formula(y ~ x1)
  f2 <- formula(y ~ x1 + x2)
  f3 <- formula(y ~ x1 + x2 + x3)
  lForms <- list(f1, f2, f3)
  ```
Omitted Variable (Un?)Bias

Construct the grid for test conditions:

```r
dfCases <- expand.grid(mc = vMC,
                        n = vSize,
                        form = c(1, 2, 3))
```
Omitted Variable (Un?)Bias

Set up the parallel backend:

```r
library("doParallel")

## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel

cl <- makeCluster(2, "PSOCK")
registerDoParallel(cl)
```
Run the simulation:

dfOut <- foreach(case = 1:nrow(dfCases),
        .combine = rbind,
        .export = c("lForms", "dfCases", "genNormData")
   ) %dopar% {
    cN <- dfCases$n[case]
    cForm <- dfCases$form[case]
    cMC <- dfCases$mc[case]
    dfTmpData <- genNormData(cN,.scale = .75)
    mod <- lm(formula = lForms[[cForm]],
              data = dfTmpData
       )
    mOut <- matrix(c(cMC, cForm, cN,
                      coef(mod), rep(NA, times = 3 - cForm)
               ),
                   nrow = 1
    )
    ret <- as.data.frame(mOut)
    return(ret)
}
Add useful column names:

```r
names(dfOut) <- c("mc", "form", "n", "x0", "x1", "x2", "x3")
```
Omitted Variable (Un?)Bias

Does the simulation output make sense?

```r
head(dfOut)
```

```
## mc form n x0 x1 x2 x3
## 1 1 1 100 1.1061 1.1070 NA NA
## 2 2 1 100 1.1488 1.2008 NA NA
## 3 3 1 100 0.9752 1.0529 NA NA
## 4 4 1 100 1.2431 1.3084 NA NA
## 5 5 1 100 1.1132 0.4977 NA NA
## 6 6 1 100 1.2134 1.0090 NA NA
```

```r
tail(dfOut)
```

```
## mc form n x0 x1 x2 x3
## 355 15 3 2000 0.9576 1.0422 0.9793 1.0178
## 356 16 3 2000 0.9465 0.9551 1.0409 1.0061
## 357 17 3 2000 1.0203 1.0248 1.0081 1.0118
## 358 18 3 2000 1.0139 1.0031 1.0210 1.0347
## 359 19 3 2000 0.9934 0.9279 1.0038 0.9614
## 360 20 3 2000 0.9950 0.9798 0.9995 0.9870
```
Omitted Variable (Un?)Bias

Close down the parallel backend:

```
stopCluster(cl)
```

Save the results:

```
write.csv(dfOut, "ov_unbias.csv")
```
Omitted Variable (Un?)Bias

1. Save your R code as `ov_unbias.R`.
2. Write a SLURM Job Script:

```bash
#!/bin/sh
#SBATCH --nodes 1 # << number of compute nodes
#SBATCH --ntasks-per-node=4 # << number of processors per node
#SBATCH -t 00:10:00 # << max time required (hh:mm:ss)
#SBATCH -J ovunbias # << brief, descriptive job label
#SBATCH -o log.%j # << file to save output/error

Rscript ov_unbias.R
```

Listing 2: `ov_unbias.slurm`
Omitted Variable (Un?)Bias

1. Copy your code to **Adroit**.
2. Log in to **Adroit**.
3. Navigate to your SLURM script.
4. Submit your job: `sbatch ov_unbias.slurm`.
5. Check the queue: `squeue -u netid`.
6. Check your log file: `cat log.452`.
7. Check that your output seems sensible: `head ov_unbias.csv`.
8. Copy your output file back to your local machine.
Omitted Variable (Un?)Bias

Visualize!

## Error: object 'variable' not found
Wrap-Up

• The entire development process using parallel computing on Adroit

• With practice, you will become more efficient.
• Countless tools to make each of these steps a little easier.
  • But, don’t want to learn too many new things at once.

• Developing and debugging locally is dramatically easier
• Set up your workflow so that this is possible.

• **Drawback**: The current approach only allows us to parallelize across processors within a single computational node on Adroit
  • This afternoon we discuss a more general approach.