

Lecture 9: Cluster computing

Brian Williamson

BIOST 561: Computational Skills For Biostatistics I

29 May 2019

Motivation

We are often interested in **large-scale** computing:

Motivation

We are often interested in **large-scale** computing:

- simulation studies (e.g., lecture 4)

Motivation

We are often interested in **large-scale** computing:

- simulation studies (e.g., lecture 4)
- intensive data analyses (e.g., **air pollution and mortality**)

Motivation

We are often interested in **large-scale** computing:

- simulation studies (e.g., lecture 4)
- intensive data analyses (e.g., **air pollution and mortality**)

In Lecture 8, you learned how to compute without R.

Today, you'll learn how to transfer those skills to computing on a **cluster**.

Examples from my research

Large-scale simulation studies:

Examples from my research

Large-scale simulation studies:

- proof-of-concept examples

Examples from my research

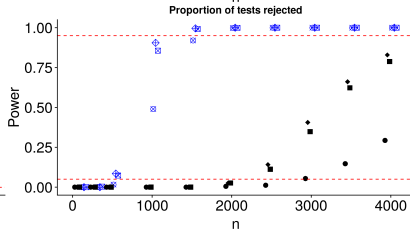
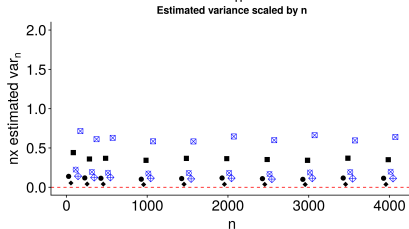
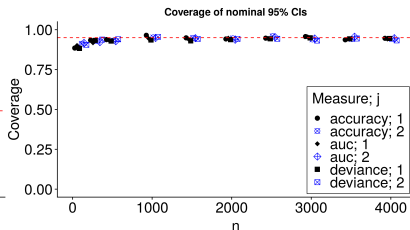
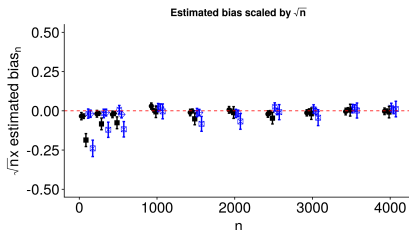
Large-scale simulation studies:

- proof-of-concept examples
- showcase **operating characteristics** of proposed method

Examples from my research

Large-scale simulation studies:

- proof-of-concept examples
- showcase **operating characteristics** of proposed method



Examples from my research

Data analysis:

Examples from my research

Data analysis:

- fit a time-consuming estimator

Examples from my research

Data analysis:

- fit a time-consuming estimator
- cross-validation?

Examples from my research

Data analysis:

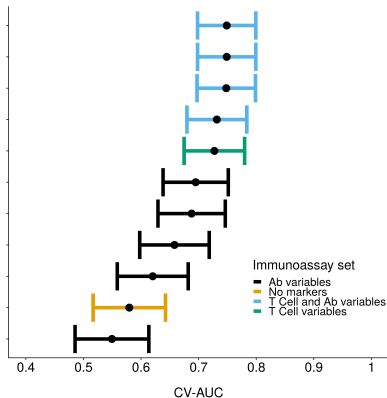
- fit a time-consuming estimator
- cross-validation?
- memory-intensive computations?

Examples from my research

Data analysis:

- fit a time-consuming estimator
- cross-validation?
- memory-intensive computations?

Assay combination	CV-AUC [95% CI]
IgG + IgA + IgG3 + T Cells	0.749 [0.698, 0.799]
IgG + IgA + T Cells	0.749 [0.698, 0.799]
All markers	0.748 [0.697, 0.799]
T Cells + Fx Ab	0.732 [0.680, 0.784]
T Cells	0.727 [0.675, 0.780]
IgG3	0.695 [0.638, 0.751]
IgG + IgA + IgG3	0.688 [0.629, 0.746]
IgG + IgA + IgG3 + Fx Ab	0.658 [0.598, 0.718]
IgG + IgA	0.620 [0.559, 0.682]
No markers	0.580 [0.517, 0.643]
Fx Ab	0.550 [0.485, 0.614]



The problem

How do I run

The problem

How do I run

- many

The problem

How do I run

- many
- potentially time-consuming

The problem

How do I run

- many
- potentially time-consuming
- or memory-intensive

The problem

How do I run

- many
- potentially time-consuming
- or memory-intensive

analyses without

The problem

How do I run

- many
- potentially time-consuming
- or memory-intensive

analyses without keeping an R session open (for the duration!)

The problem

How do I run

- many
- potentially time-consuming
- or memory-intensive

analyses without keeping an R session open (for the duration!)
on my personal machine?

The solution

Cluster computers provide a solution!

The solution

Cluster computers provide a solution!

Cluster computers:

The solution

Cluster computers provide a solution!

Cluster computers:

- allow you to submit multiple **jobs*** at once

The solution

Cluster computers provide a solution!

Cluster computers:

- allow you to submit multiple **jobs*** at once
- can schedule jobs for you

The solution

Cluster computers provide a solution!

Cluster computers:

- allow you to submit multiple **jobs*** at once
- can schedule jobs for you
- are optimized for high-performance computing (HPC)

The solution

Cluster computers provide a solution!

Cluster computers:

- allow you to submit multiple **jobs*** at once
- can schedule jobs for you
- are optimized for high-performance computing (HPC)

*: we will discuss this further!

Running example: robust standard errors (SEs)

Consider a sample of n observations generated iid according to

$$X \sim N(0, 1),$$

$$u \sim N(0, 1), \text{ independent of } X;$$

$$Y \mid X, u = \beta_0 + \beta_1 X + \epsilon, \text{ where}$$

$$\epsilon = |X|u.$$

Questions: can we use

1. linear regression to estimate β_1 ?

Running example: robust standard errors (SEs)

Consider a sample of n observations generated iid according to

$$X \sim N(0, 1),$$

$$u \sim N(0, 1), \text{ independent of } X;$$

$$Y \mid X, u = \beta_0 + \beta_1 X + \epsilon, \text{ where}$$

$$\epsilon = |X|u.$$

Questions: can we use

1. linear regression to estimate β_1 ? **Yes!**

Running example: robust standard errors (SEs)

Consider a sample of n observations generated iid according to

$$X \sim N(0, 1),$$

$$u \sim N(0, 1), \text{ independent of } X;$$

$$Y \mid X, u = \beta_0 + \beta_1 X + \epsilon, \text{ where}$$

$$\epsilon = |X|u.$$

Questions: can we use

1. linear regression to estimate β_1 ? **Yes!**
2. model-based standard errors to estimate $sd(\beta_1)$?

Running example: robust standard errors (SEs)

Consider a sample of n observations generated iid according to

$$X \sim N(0, 1),$$

$$u \sim N(0, 1), \text{ independent of } X;$$

$$Y \mid X, u = \beta_0 + \beta_1 X + \epsilon, \text{ where}$$

$$\epsilon = |X|u.$$

Questions: can we use

1. linear regression to estimate β_1 ? **Yes!**
2. model-based standard errors to estimate $sd(\beta_1)$? **No!**

Running example: robust standard errors (SEs)

Consider a sample of n observations generated iid according to

$$X \sim N(0, 1),$$

$$u \sim N(0, 1), \text{ independent of } X;$$

$$Y \mid X, u = \beta_0 + \beta_1 X + \epsilon, \text{ where}$$

$$\epsilon = |X|u.$$

Questions: can we use

1. linear regression to estimate β_1 ? **Yes!**
2. model-based standard errors to estimate $sd(\beta_1)$? **No!**
3. robust standard errors to estimate $sd(\beta_1)$?

Running example: robust standard errors (SEs)

Consider a sample of n observations generated iid according to

$$X \sim N(0, 1),$$

$$u \sim N(0, 1), \text{ independent of } X;$$

$$Y | X, u = \beta_0 + \beta_1 X + \epsilon, \text{ where}$$

$$\epsilon = |X|u.$$

Questions: can we use

1. linear regression to estimate β_1 ? **Yes!**
2. model-based standard errors to estimate $sd(\beta_1)$? **No!**
3. robust standard errors to estimate $sd(\beta_1)$? **Yes!**

Running example: robust SEs

Goals:

Running example: robust SEs

Goals:

- compare model-based to robust SEs

Running example: robust SEs

Goals:

- compare model-based to robust SEs
- do this without using our own computers

Running example: robust SEs

Goals:

- compare model-based to robust SEs
- do this without using our own computers

We will use the cluster to do this!

Part I: coding for the cluster

Coding for the cluster

Before moving to the cluster, we need to code differently:

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (**easy debugging**)

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (**easy debugging**)
- setting seeds (**reproducible results**)

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (**easy debugging**)
- setting seeds (**reproducible results**)
- saving output (**for results!**)

The simulator (Lecture 4) makes this easy:

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly
- forces you to set a seed

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly
- forces you to set a seed
- forces you to save lots of output

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly
- forces you to set a seed
- forces you to save lots of output

A couple of drawbacks:

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly
- forces you to set a seed
- forces you to save lots of output

A couple of drawbacks:

- can cause memory leaks

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly
- forces you to set a seed
- forces you to save lots of output

A couple of drawbacks:

- can cause memory leaks
- sometimes difficult to link to cluster nodes

Coding for the cluster

Before moving to the cluster, we need to code differently:

- modular code (easy debugging)
- setting seeds (reproducible results)
- saving output (for results!)

The simulator (Lecture 4) makes this easy:

- forces you to code modularly
- forces you to set a seed
- forces you to save lots of output

A couple of drawbacks:

- can cause memory leaks
- sometimes difficult to link to cluster nodes

Today, I'll provide an alternative method (this should not always replace the simulator!).

Coding for the cluster: modular code

Coding for the cluster: setting the seed

Coding for the cluster: #! and executables

Coding for the cluster: saving output

Coding for the cluster: debugging

Coding for the cluster: compiling output

Part II: using the cluster

Using the cluster

Department resources for HPC*:

Using the cluster

Department resources for HPC*:

- cox: 12-core computer

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

- a head node

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

- a head node (where you are)

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

- a head node (where you are)
- compute nodes

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

- a head node (where you are)
- compute nodes (where your code gets run)

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

- a head node (where you are)
- compute nodes (where your code gets run)
- submission system

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

A cluster consists of:

- a head node (where you are)
- compute nodes (where your code gets run)
- submission system (how your code gets run)

Using the cluster

Department resources for HPC*:

- cox: 12-core computer (not a cluster)
- bayes: compute cluster

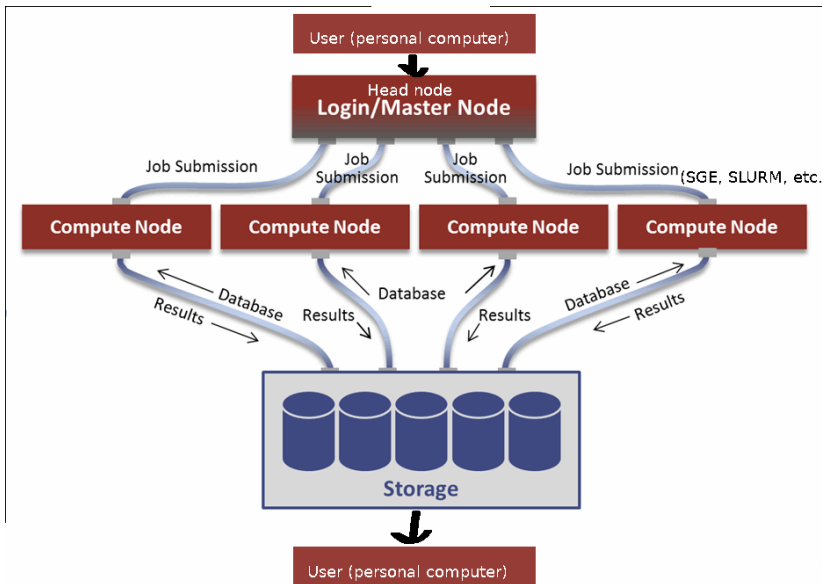
A cluster consists of:

- a head node (where you are)
- compute nodes (where your code gets run)
- submission system (how your code gets run)

*other groups have similar resources, e.g.,

- hyak (managed by UW-IT)
- pearson (statistical genetics group only)
- gizmo (Fred Hutchinson Cancer Research Center only)
- Microsoft Azure, Amazon Web Services (AWS)

Using the cluster



Using the cluster: bayes

More specifically, bayes

Using the cluster: bayes

More specifically, bayes

- has 4 department-wide compute nodes, each with 12 cores

Using the cluster: bayes

More specifically, bayes

- has 4 department-wide compute nodes, each with 12 cores
- uses Sun Grid Engine (SGE) for submissions

Using the cluster: bayes

More specifically, bayes

- has 4 department-wide compute nodes, each with 12 cores
- uses Sun Grid Engine (SGE) for submissions

Since this is a shared resource, being nice is important:

Using the cluster: bayes

More specifically, bayes

- has 4 department-wide compute nodes, each with 12 cores
- uses Sun Grid Engine (SGE) for submissions

Since this is a shared resource, being nice is important:

- don't run simulations on the head node

Using the cluster: bayes

More specifically, bayes

- has 4 department-wide compute nodes, each with 12 cores
- uses Sun Grid Engine (SGE) for submissions

Since this is a shared resource, being nice is important:

- don't run simulations on the head node
- don't flood the cluster

Using the cluster: bayes

More specifically, bayes

- has 4 department-wide compute nodes, each with 12 cores
- uses Sun Grid Engine (SGE) for submissions

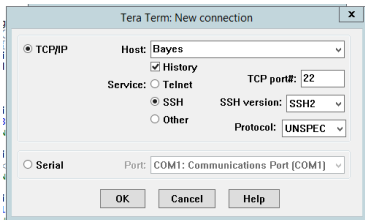
Since this is a shared resource, being nice is important:

- don't run simulations on the head node
- don't flood the cluster

We'll practice good habits for being nice as we go along.

Using the cluster: logging in (Windows, incl. box)

1. Open TeraTerm [or your favorite secure shell (SSH) client]
2. Enter the address of your favorite cluster, e.g.,
bayes.biostat.washington.edu
3. Make sure that the “New connection” window is filled out as in the figure, except for “Host” (screenshot from box)
4. Click OK, enter your UW BIOST username and password when prompted (user and pass you use for box)



Using the cluster: logging in (Mac/Linux)

1. Open a Terminal window
2. Type `ssh mynetid@cluster.washington.edu`
 - replace `mynetid` with Your UW NetID, and
 - replace `cluster` with Your cluster, e.g., `bayes`
3. Enter password (same password as box) when prompted (the field will remain blank but your password will be received)

Using the cluster: logging in

Your turn!

1. Log into the cluster
2. What is the name of the directory that you are when you log in?
3. Create a directory called `robust_ses` in your home directory
4. Create a directory called `robust_ses` on **your** computer, under `biost561/lecture9`

Using the cluster: moving around

The cluster runs on Linux: in particular, the tools you learned in lecture 8, including

Using the cluster: moving around

The cluster runs on Linux: in particular, the tools you learned in lecture 8, including

- navigation,

Using the cluster: moving around

The cluster runs on Linux: in particular, the tools you learned in lecture 8, including

- navigation,
- vim,

Using the cluster: moving around

The cluster runs on Linux: in particular, the tools you learned in lecture 8, including

- navigation,
- vim,
- commands,

Using the cluster: moving around

The cluster runs on Linux: in particular, the tools you learned in lecture 8, including

- navigation,
- vim,
- commands, and
- shell scripts

are all used **in exactly the same way!**

Using the cluster: jobs

The basic unit of cluster computing is a **job**.

Using the cluster: jobs

The basic unit of cluster computing is a **job**.

Jobs:

Using the cluster: jobs

The basic unit of cluster computing is a **job**.

Jobs:

- perform a specified task

Using the cluster: jobs

The basic unit of cluster computing is a **job**.

Jobs:

- perform a specified task
- can be submitted to the cluster compute nodes

Using the cluster: jobs

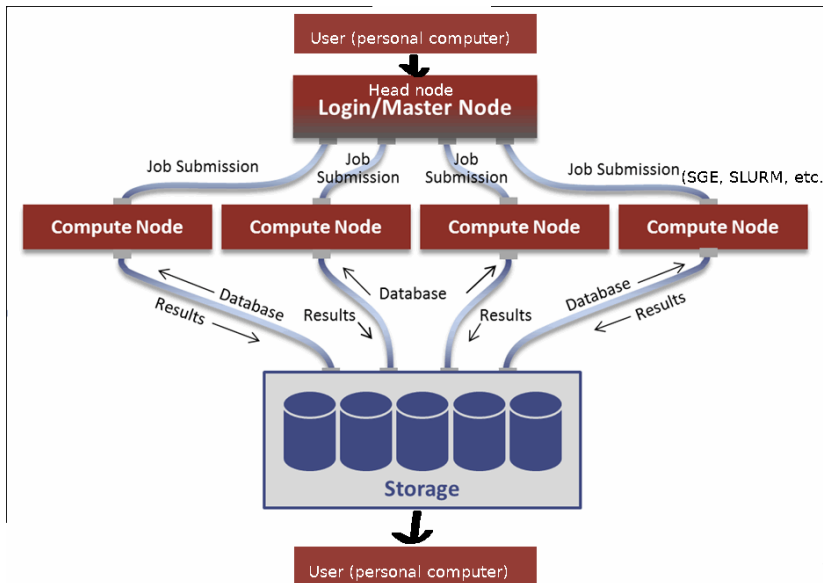
The basic unit of cluster computing is a **job**.

Jobs:

- perform a specified task
- can be submitted to the cluster compute nodes

Example job:

Using the cluster: jobs



Using the cluster: submitting jobs

Workhorse command on SGE: `qsub`

Many options, including:

Using the cluster: submitting jobs

Workhorse command on SGE: `qsub`

Many options, including:



Options are specified with a single `-`, as you saw in Lecture 8.

Using the cluster: checking jobs

Using the cluster: other helpful SGE commands

Appendix: other useful cluster things

Useful things: Windows file compatability