

# Workshop: High-performance computing for economists

Lars Vilhuber<sup>1</sup> John M. Abowd<sup>1</sup> Richard Mansfield<sup>1</sup>  
Kevin L. McKinney

<sup>1</sup>Cornell University, Economics Department,

August 20-22, 2013: Day 1

What do you learn in a Ph.D. program?

What do you learn in a Ph.D. program?

How to learn...

## Goal of this class

## Goal of this class

To open new doors, to be able to conceive of problems that you didn't think had a feasible solution.

## Goal of this class

To open new doors, to be able to conceive of problems that you didn't think had a feasible solution.

To broaden your knowledge about what you do NOT know

# Overview

## Day 1

- ▶ Programming basics (Lars)

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor



# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting
- ▶ Basics of version control (Lars)

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting
- ▶ Basics of version control (Lars)
  - ▶ File-system based version control

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting
- ▶ Basics of version control (Lars)
  - ▶ File-system based version control
  - ▶ More formal version control (Subversion, Git)

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting
- ▶ Basics of version control (Lars)
  - ▶ File-system based version control
  - ▶ More formal version control (Subversion, Git)
  - ▶ Working with servers



# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting
- ▶ Basics of version control (Lars)
  - ▶ File-system based version control
  - ▶ More formal version control (Subversion, Git)
  - ▶ Working with servers
  - ▶ Setting up infrastructure at Cornell

# Overview

## Day 1

- ▶ Programming basics (Lars)
  - ▶ Choosing an editor
  - ▶ How to structure programs, texts, etc.
  - ▶ A clean sequence of programs
  - ▶ NX, SSH, Linux, request an account on cluster
  - ▶ Basic scripting
- ▶ Basics of version control (Lars)
  - ▶ File-system based version control
  - ▶ More formal version control (Subversion, Git)
  - ▶ Working with servers
  - ▶ Setting up infrastructure at Cornell
- ▶ Subroutines: the example of function programming in R (Lars)

# Overview

Day 2

# Overview

Day 3

# Structure of the class

Teaching...

We'll take you on a 4,000 m flight through topics...

# Structure of the class

## Teaching...

We'll take you on a 4,000 m flight through topics...

## ... and practice

... and then swoop in on some examples, leaving ample time to practice it.

# Choosing editors

## Why does choosing editors matter?

The (applied) research process iterates through writing papers and doing estimation. You want to use the appropriate tools for each task.

## Integrated or separate

- ▶ You can use native tools that come with each word processing facility/programming language/etc.
- ▶ Not all of them will have one.
- ▶ Not all of them will work on all platforms.
- ▶ You will likely use multiple tools

# Choosing an editor

... or system

## Separate editors and systems

- ▶ MS Word and math editor (Windows/OSX but compatibility issues)
- ▶ LibreOffice (Windows/OSX/Linux but not as good)
- ▶ NotePad++ (Windows)
- ▶ Gedit, (X)Emacs, Kate (Linux)
- ▶ ?? (OSX)

**L<sup>A</sup>T<sub>E</sub>X**: all platforms, but some GUIs are not cross-platform, ease of use varies:

- ▶ TeXstudio (all platforms)
- ▶ TeXMaker (all platforms)
- ▶ Scientific Workplace (Windows, mythical Linux)
- ▶ TeXWorks+Miktex
- ▶ T<sub>E</sub>XnicCenter
- ▶ and (many more)



# Choosing an editor

... or system

## Integrating programming and running

- ▶ IDE ( Eclipse, ActiveState Komodo, etc.)
- ▶ Native programming GUIs (SAS, Matlab, Stata)
- ▶ Gedit, (X)Emacs (with add-on functionality)

## Integrating programs and text/results

- ▶ SWeave (integrates  $\text{\LaTeX}$  and R)
- ▶ RStudio (GUI to R and SWeave)
- ▶ StatRep (Integrated SAS and  $\text{\LaTeX}$ , Source 1, Source 2)

# Structuring programs

Easy...

## Listing 1: mystuff.sas

```
1 data "C:\Users\Me\CensusChina.sas7bdat";  
2 set "C:\Users\Me\CensusChina.sas7bdat";  
3 earn=log(earn);  
4 run;  
5 proc reg data="C:\Users\Me\CensusChina.sas7bdat";  
6 model earn = sex education experience;  
7 run;
```

What can possibly be wrong about that?

# Structuring programs 2

Easier...

## Listing 2: mystuff.do

```
1 use "C:\Users\Me\CensusChina.dta"  
2 replace earn=log(earn)  
3 regress earn sex education experience  
4 save, replace
```

What can possibly be wrong about that?

# Structuring programs 3

Actually...

Everything!

- ▶ Name of program: uninformative
- ▶ Destruction of original data: program cannot be re-run for same results
- ▶ No portability: cannot be run anywhere else
- ▶ No explanation: why are we doing this?

But of course, nobody does that, right?

# Structuring programs 4

Better...?

## Listing 3: china-regression.sas

```
1 data logCensusChina;  
2     set "C:\Users\Me\CensusChina.sas7bdat";  
3     earn=log(earn);  
4 run;  
5 proc reg data=logCensusChina;  
6 model earn = sex education experience;  
7 run;
```

# Structuring programs 4

Better...?

## Listing 4: china-regression.sas

```
1 data logCensusChina;  
2     set "C:\Users\Me\CensusChina.sas7bdat";  
3     earn=log(earn);  
4 run;  
5 proc reg data=logCensusChina;  
6     model earn = sex education experience;  
7 run;
```

Somewhat...

# Structuring programs 5

## Addressing these issues

- ▶ Naming of programs: here
- ▶ Commenting: here
- ▶ Versioning: up next
- ▶ Portability and Data management: tomorrow

# Key notions about naming

Think of yourself as highly amnesiac...

- ▶ The research paper you are writing now will be submitted, rejected, worked on, questioned...



# Key notions about naming

Think of yourself as highly amnesiac...

- ▶ The research paper you are writing now will be submitted, rejected, worked on, questioned...
- ▶ ... by others and yourself

# Key notions about naming

Think of yourself as highly amnesiac...

- ▶ The research paper you are writing now will be submitted, rejected, worked on, questioned...
- ▶ ... by others and yourself
- ▶ ... in intervals of weeks, months, years...

# Key notions about naming

## Think of yourself as highly amnesiac...

- ▶ The research paper you are writing now will be submitted, rejected, worked on, questioned...
- ▶ ... by others and yourself
- ▶ ... in intervals of weeks, months, years...
- ▶ Your future research assistant and the future YOU will need to understand how to go through it.

# Naming

## The really bad

mystuff.R

**read**.R

version2.R

ols.sas

# Naming

## The really bad

mystuff.R

**read**.R

version2.R

ols.sas

## The bad

readCensus.R

readBLS.R

prepareCensus.R

runOLS.sas

# Naming

## Better

```
01_readBLS.R  
02_readCensus.R  
03_prepareCensus.R  
04_create_analysis_data.R  
05_runOLS.sas
```

# Naming

## Better

```
01_readBLS.R  
02_readCensus.R  
03_prepareCensus.R  
04_create_analysis_data.R  
05_runOLS.sas
```

## Even better

```
01_01_readBLS.R  
02_01_readCensus.R  
02_02_prepareCensus.R  
03_01_create_analysis_data.R  
04_01_runOLS.sas  
README.txt
```

# Naming

## Going overboard?

```
icf/ctrlprogs/control_icf.sas  
icf/ctrlprogs/parameters_icf.sas  
icf/library/macros/icf_cleanup.sas  
icf/library/macros/icf_impute_county_res.sas  
icf/library/macros/licf_findnum.sas  
icf/library/macros/licf_proxy.sas  
icf/library/macros/licf_stars1.sas  
icf/library/macros/licf_tgrlatlongs.sas  
icf/library/sasprogs/01_icfqa.sas  
icf/library/sasprogs/01_icf.sas  
icf/library/sasprogs/02_icfqa.sas  
icf/library/sasprogs/02_icf.sas  
icf/library/sasprogs/03_icfqa.sas  
icf/library/sasprogs/03_icf.sas  
[snip]  
icf/library/sasprogs/19_icf.sas
```



# Naming

## Going overboard?

```
icf / ctrlprogs / control_icf .sas  
icf / ctrlprogs / parameters_icf .sas  
icf / library / macros / icf_cleanup .sas  
icf / library / macros / icf_impute_county_res .sas  
icf / library / macros / licf_findnum .sas  
icf / library / macros / licf_proxy .sas  
icf / library / macros / licf_stars1 .sas  
icf / library / macros / licf_tgrlatlongs .sas  
icf / library / sasprogs / 01_icfqa .sas  
icf / library / sasprogs / 01_icf .sas  
icf / library / sasprogs / 02_icfqa .sas  
icf / library / sasprogs / 02_icf .sas  
icf / library / sasprogs / 03_icfqa .sas  
icf / library / sasprogs / 03_icf .sas  
[snip]  
icf / library / sasprogs / 19_icf .sas
```

```
ehf / ctrlprogs / control_ehf .sas  
ehf / library / macros / read_bls .sas  
ehf / library / sasprogs / 01_ehf .sas  
[snip]
```

# Naming

## With minor modification

```
icf /ctrlprogs/control_icf.sas
icf /ctrlprogs/parameters_icf.sas
icf /library/macros/icf_cleanup.sas
icf /library/macros/icf_impute_county_res.sas
icf /library/macros/licf_findnum.sas
icf /library/macros/licf_proxy.sas
icf /library/macros/licf_stars1.sas
icf /library/macros/licf_tgrlatlongs.sas
icf /library/sasprogs/01_icf.sas
icf /library/sasprogs/02_icf.sas
icf /library/sasprogs/03_icf.sas
[snip]
icf /library/sasprogs/19_icf.sas
icf /library/sasprogs/01_icfqa.sas
icf /library/sasprogs/02_icfqa.sas
icf /library/sasprogs/03_icfqa.sas
```

Can you figure out in what sequence to run them?

## Basic Linux, basic scripting

## Linux

- ▶ used on most compute clusters
- ▶ used on very few desktop computers
- ▶ but...

## Linux

- ▶ used on most compute clusters
- ▶ used on very few desktop computers
- ▶ but...

## Bash

- ▶ bash is a “shell” - a text interface command interpreter
- ▶ bash or ksh (Korn shell) or csh (C-shell) are the most common
- ▶ bash is available on Linux and

## Linux

- ▶ used on most compute clusters
- ▶ used on very few desktop computers
- ▶ but...

## Bash

- ▶ bash is a “shell” - a text interface command interpreter
- ▶ bash or ksh (Korn shell) or csh (C-shell) are the most common
- ▶ bash is available on Linux and OSX
- ▶ you can also download Cygwin, getting bash for Windows

# Why worry?

You will end up doing something on the command line

- ▶ Launch a program from a compute-cluster job

# Why worry?

You will end up doing something on the command line

- ▶ Launch a program from a compute-cluster job
- ▶ Launch a job submission



# Why worry?

You will end up doing something on the command line

- ▶ Launch a program from a compute-cluster job
- ▶ Launch a job submission
- ▶ Basic scripting

# Linux in 2 minutes

- ▶ ls - will list the contents of a directory
- ▶ cd - will “change directory”
- ▶ cd .. (note the spaces) will go up a directory
- ▶ cd (name) will go into the directory (name)
- ▶ rm (name) will delete
- ▶ mkdir (name) will create a directory called (name)
- ▶ vi (name) will open a venerable command line editor for file (name)

# Linux in 2 minutes

- ▶ ls - will list the contents of a directory
- ▶ cd - will “change directory”
- ▶ cd .. (note the spaces) will go up a directory
- ▶ cd (name) will go into the directory (name)
- ▶ rm (name) will delete
- ▶ mkdir (name) will create a directory called (name)
- ▶ vi (name) will open a venerable command line editor for file (name) (CAUTION: to exit, hit ESC, then :q!)

# Basic scripting in Linux

## A basic loop on the command line

```
1  for i in 1 3 7 99
2  do
3    echo $i
4  done
5  for (( i ; i<10; i++ ))
6  do
7    echo $i
8  done
```

Source: [1]

# Capturing output

You can capture the output from a command

```
> seq 1 3  
1  
2  
3
```

Now let's use that:

```
for i in $(seq 1 3)  
do  
    echo $i  
done
```

# Basic scripting in Linux

## Use for practical things

Remember that ICF program sequence? How would we go about starting 19 programs in sequence?

```
for program in $(ls *_icf.sas)
do
    sas $program
done
```

# Advanced linux in 2 minutes

## The gateway to everything

man

or try <http://www.linuxmanpages.com> or <http://linux.die.net/man/>

## The toolkit

- ▶ sed
- ▶ grep
- ▶ awk
- ▶ regex (regular expressions)

# Advanced scripting in Linux

## Use for practical things

What if I'm running 100s of programs, and trying to figure out if any of them have errors?

```
for logfiles in $(ls *_icf.log)
do
    grep ERROR $logfiles
done
```



Now let's try it out

Next section

Next section