Workshop: High-performance computing for economists

Lars Vilhuber[1]   John M. Abowd[1]   Richard Mansfield[1]
Hautahi Kingi[1]   Flavio Stanchi[1]   Sida Peng[1]
Kevin L. McKinney

[1] Cornell University, Economics Department,

August 18-21, 2014: Day 2

What are VCS

Software development perspective

Version control systems (VCS) (or Source Configuration
Management (SCM) systems ∗) allow developers or authors to
keep track of the history of their projects? source code. [source]

What are VCS

Software development perspective

Version control systems (VCS) (or Source Configuration
Management (SCM) systems ∗) allow developers or authors to
keep track of the history of their projects? source code. [source]

Generic view

Detailed mechanism to manage different versions (historical,
parallel) of documents, files, programs, etc.

You already have it

Implicit uses

- ▶ Backup systems (Apple Time Machine, others)
- ▶ Word processors (Undo feature, Track changes in Word, finer-grained features in content management systems: Google Docs, Blog software, etc.)
- ▶ Versioning filesystems
- ▶ Paper books!

Modern Labor Economics: Theory and Public Policy

# Modern Labor Economics: Theory and Public Policy

Click to **LOOK INSIDE!**

MODERN LABOR ECONOMICS
THEORY AND PUBLIC POLICY   11TH EDITION
RONALD G. EHRENBERG   ROBERT S. SMITH
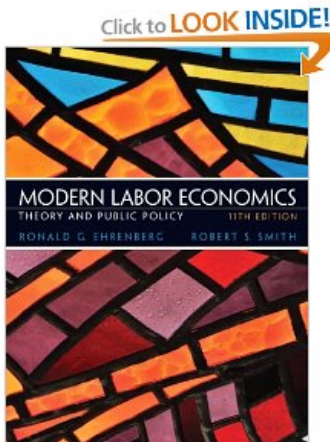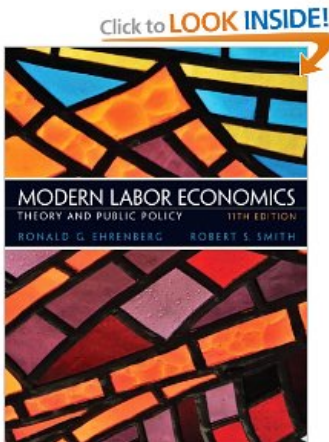
Preface

New to This Edition

- This eleventh edition of *Modern Labor Economics* has been thoroughly updated in terms of both tabular material and references to the latest literature. Our goal in these updates is to make our textbook a comprehensive reference, for both students and professors, to critical factual information about the labor market *and* to the professional literature in labor economics.
- In recognition of the growing need for rigorous and dispassionate analyses of American immigration policy, we have expanded our analysis of undocumented immigration in chapter 10 to include an enhanced analysis of both its theoretical and measured effects on society.
- We have also incorporated, in relevant chapters, discussions that include labor-market effects of the Great Recession of 2008, along with an examination of recent changes in such outcomes as earnings inequality, human-capital acquisition, and labor-force participation.
- In chapter 6, we added a discussion of the labor supply behavior of married women and a new boxed example on the labor supply of New York City taxi drivers.
- In chapter 11, we amplified the "Group Incentives and Executive Pay" section and added a new boxed example on the "rat race" in law firms.
- In addition to including new material on the recession, we added a new boxed example on earnings inequality in developed countries and a new section on earnings instability to chapter 15.
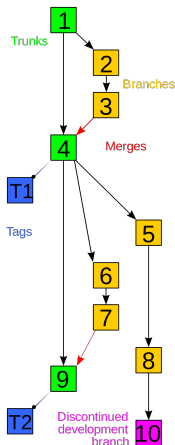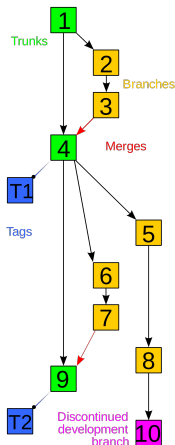
Principal idea

1 First edition
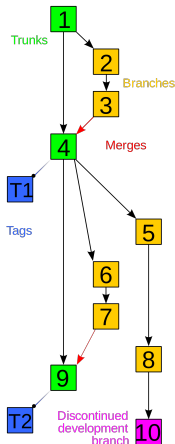
Principal idea
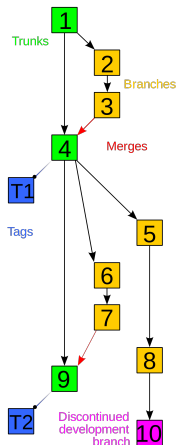
1 First edition

4 Second edition

Principal idea
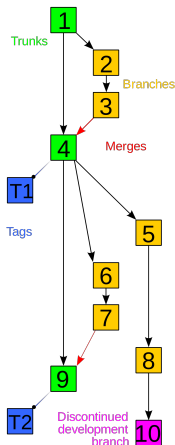

1 First edition

4 Second edition

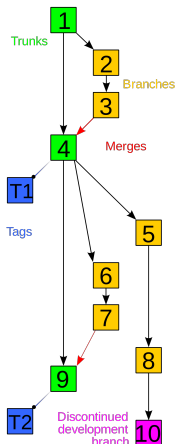5 Start of work on a
   Canadian edition

Principal idea

1 First edition

4 Second edition

5 Start of work on a Canadian edition

6 Start of work on the next US edition

Principal idea

1 First edition

4 Second edition

5 Start of work on a Canadian edition

6 Start of work on the next US edition

9 Third edition

Principal idea

1 First edition

4 Second edition

5 Start of work on a Canadian edition

6 Start of work on the next US edition

9 Third edition

10 First Canadian edition

File-system based versioning

The most common method... Also used in email versioning...

Remember this?

01_01_readBLS.R

02_01_readCensus.R

02_02_prepareCensus.R

03_01_create_analysis_data.R

04_01_runOLS.sas

README.txt

File-system based versioning

The most common method... Also used in email versioning...

What if I make changes?

01_01_readBLS.R

02_01_readCensus.R

02_02_prepareCensus.R

03_01_create_analysis_data.R

04_01_runOLS.sas

README.txt

File-system based versioning

The most common method... Also used in email versioning...

What if I make changes?

01_01_readBLS.R

02_01_readCensus.R

02_02_prepareCensus.R

03_01_create_analysis_data.R

04_01_runOLS.sas

README.txt

02_01_readCensus.R.bak

File-system based versioning

The most common method... Also used in email versioning...

What if I make changes?

```
01_01_readBLS.R
02_01_readCensus.R
02_02_prepareCensus.R
03_01_create_analysis_data.R
04_01_runOLS.sas
README.txt
02_01_readCensus.R.bak
02_01_readCensus_V2.R
```

File-system based versioning

The most common method... Also used in email versioning...

What if I make changes?

01_01_readBLS.R

02_01_readCensus.R

02_02_prepareCensus.R

03_01_create_analysis_data.R

04_01_runOLS.sas

README.txt

02_01_readCensus.R.bak

02_01_readCensus_V2.R

02_01_readCensus_V3.R

File-system based versioning

The most common method... Also used in email versioning...

What if I make changes?

```
01_01_readBLS.R
02_01_readCensus.R
02_02_prepareCensus.R
03_01_create_analysis_data.R
04_01_runOLS.sas
README.txt
02_01_readCensus.R.bak
02_01_readCensus_V2.R
02_01_readCensus_V3.R
02_01_readCensus_V3-jma.R
```

File-system based versioning

The most common method... Also used in email versioning...

What if I make changes?

```
01_01_readBLS.R
02_01_readCensus.R
02_02_prepareCensus.R
03_01_create_analysis_data.R
04_01_runOLS.sas
README.txt
02_01_readCensus.R.bak
02_01_readCensus_V2.R
02_01_readCensus_V3.R
02_01_readCensus_V3-jma.R
02_01_readCensus_V3-jma-rm.R
```

Better way

Is there a better way?

History

Two major types of version-management

Centralized model

Server-client approach, editors check out a copy, modify it, and check it back in. Multiple editors:

- *File locking*: only one person can check out any given file
- *Version merging*: discrepancies are handled upon checkin

Two major types of version-management

Centralized model

Server-client approach, editors check out a copy, modify it, and check it back in. Multiple editors:

- ► *File locking*: only one person can check out any given file
- ► *Version merging*: discrepancies are handled upon checkin

Distributed model

There is no central server (prescribed by software). Every editor has a full copy of all version, synchronisation occurs by exchanging patches.

Focus in this class

We will focus on two modern systems:
- ▶ Subversion (centralized) - Flavio

Windows   TortoiseSVN (free)
OSX   Versions ($), Xcode (free)

- ▶ Git (decentralized) - Hautahi

Quickstart on SSG/ECCO

The following really quick tutorial can be run on SSG or on your own computer, if you already have the software installed.

```
ssh NETID@ssg.vrdc.cornell.edu
Last login: Mon Aug 19 15:43:33 2013 from lv39-ws.ilr.cornell.edu
-------------------------------------------------------------------
Welcome to the Social Science Gateway (SSG),
for help, send email to ssg-help@cac.cornell.edu
-------------------------------------------------------------------
ssg:~> svn help
usage: svn <subcommand> [options] [args]
Subversion command-line client, version 1.6.6.
Type 'svn help <subcommand>' for help on a specific subcommand.
Type 'svn --version' to see the program version and RA modules
  or 'svn --version --quiet' to see just the version number.

Most subcommands take file and/or directory arguments, recursing
on the directories.  If no arguments are supplied to such a
command, it recurses on the current directory (inclusive) by default.

Available subcommands:
    add
    blame (praise, annotate, ann)
    cat
[snip]
```

Quickstart Step 1

Let's create a place to work

```
mkdir Workspace
cd Workspace
```

Quickstart Step 1

Let's create a place to work

```
mkdir Workspace
cd Workspace
```

Check out a version of the code:

```
> svn co http://repository.vrdc.cornell.edu/public/test
A     test/test.3
A     test/test.4
A     test/releases
A     test/trunk
A     test/trunk/bls
[snip]
A     test/test.1
Checked out revision 23.
>
```

Checkout   To *check out* (or *co*) is to create a local working copy
          from the repository. A user may specify a specific
          revision or obtain the latest. The term 'checkout' can
          also be used as a noun to describe the working copy.

Quickstart 2

Using your favorite editor, modify a file:

```
> cd test
> vi test.1
```

Quickstart 2

Using your favorite editor, modify a file:

```
> cd test
> vi test.1
```

Verify the state of the repsository

```
> svn status
M        test.1
```

Quickstart 2

Using your favorite editor, modify a file:

```
> cd test
> vi test.1
```

Verify the state of the repsository

```
> svn status
M       test.1
```

Let's also add a file:

```
> vi test.today
> svn status
?       test.today
M       test.1
```

The question mark indicates that the file is not (yet) versioned.

Quickstart 3

Add the unversioned file to Subversion tracking

```
> svn add  test.today
A         test.today
> svn status
M         test.1
A         test.today
```

Quickstart 3

Add the unversioned file to Subversion tracking

```
> svn add   test.today
A            test.today
> svn status
M        test.1
A        test.today
```

Now we can commit our changes

```
> svn commit
[editor pops up]
Sending         test.1
Adding          test.today
Transmitting file data ..
Committed revision 24.
```

> Commit To *commit* (*check in*, *ci*) is to write or merge the
> changes made in the working copy back to the
> repository. The terms 'commit' and 'checkin' can also
> be used as nouns to describe the new revision that is
> created as a result of committing.

Additional things you will want to learn

Manipulating files

- ▶ `svn mv` (vs regular Linux mv or cut-and-paste)
- ▶ Reverting changes (`svn revert` before a commit, more complicated after a commit)
- ▶ Merging changes (`svn merge`)
- ▶ Inspecting logs (`svn log`)
- ▶ Identifying changes relative to previous revisions (`svn diff`)

THE reference for Subversion

svnbook.red-bean.com

Learning more

A longer tutorial

A longer tutorial (focussing on a complex project setup) is available at the course website.

Tracking history

What happened? And who did what?

One of the key advantages of using version control systems is
... to control versions.

- ► Straightforward to view multiple versions of a file
  (assuming proper usage)
- ► Possibility to view who changed what ("blame" or
  "annotate")

Tracking using web interfaces: SVN

Tracking using web interfaces: SVN

Tracking using web interfaces: SVN

Tracking using web interfaces: SVN

Computing for Economists

Tracking using web interfaces: SVN

Including our demo change

VRDC repository viewer

Tracking using web interfaces: git

Tracking using web interfaces: git

Tracking using web interfaces: git

Subversion vs. Git

incomplete

Checking out a subdirectory

SVN

```
svn co ${URL}/sub/directory (name)
```

GIT

```
mkdir <repo> && cd <repo>
git init
git remote add −f <name> <url>
git config core.sparsecheckout   true
echo some/dir/ >> .git/info/sparse−checkout
echo another/sub/tree >>
.git/info/sparse−checkout
git pull <remote> <branch>
```

Source: here

VCS infrastructure

At Cornell
Cornell Sourceforge (see instructions on how to create)

Elsewhere

- ▶ CloudForge (free for personal use)
- ▶ GitHub (Git, subversion, free for open-source)
- ▶ BitBucket (Git, no subversion, free for academic users)
- ▶ Roll your own: very simple webserver setup for Subversion, no server required for Git on your PC

Next section