Reproducible Results:
A Workflow for Data Analysis

Scott Long

wfclass2017 lecture parts08-21 2017-06-20.docx

Table of Contents

Part 8: Using do-files

WFDAUS pages 47-82.

Robust and *legible* do-files are critical for reproducibility, accuracy, & efficiency.

- Robust: they run the next time, no matter where you run them.
- Legible: you can easily understand what is being done.

Robust do-files

Robust files produce *exactly* the same results later with *no modifications*.

Example: fragile do-files

1. **`wfx-step1-fragile.do`**: creates variables

```
log using wfx-step1-fragile, replace text
use wf-lfp, clear

gen hask5 = (k5>0) & (k5<.)
label var hask5 "Has children less than 5 yrs old?"

gen hask618 = (k618>0) & (k618<.)
label var hask618 "Has children between 6 and 18 yrs old?"

log close
```

2. **`wfex-step2-fragile.do`**: estimate a model

```
log using wfx-step2-fragile, replace
logit lfp hask5 hask618 age wc hc lwg inc, nolog
log close
```

3. If **`wfx-step1-fragile.do`** is run followed by **`wfx-step2-fragile.do`**, all is fine.

4. If I run **`wfx-step1-fragile.do`**, then run other do-files, and then run **`wfx-step2-fragile.do`**, I get an error since variables from **`wfx-step1-fragile.do`** are not in memory:

```
. logit lfp hask5 hask618 age wc hc lwg inc, nolog
no variables defined
r(111);
```

5. If I load the data in step 2,

```
log using wfx-step2-fragile, replace
use wf-lfp, clear
logit lfp hask5 hask618 age wc hc lwg inc, nolog
log close
```

The error is:

```
. logit lfp hask5 hask618 age wc hc lwg inc, nolog
variable hask5 not found
r(111);
```

6. **`hask5`** is not in **`wf-lfp.dta`**; it was created by **`wfx-step1-fragile.do`**.
7. To avoid this problem, make the programs *self-contained*.

wfex-step1V2.do: robust version

```
log using wfx-step1V2-robust, replace

use wf-lfp, clear

gen hask5 = (k5>0) & (k5<.)
label var hask5 "Has children less than 5 yrs old?"

gen hask618 = (k618>0) & (k618<.)
label var hask618 "Has children between 6 and 18 yrs old?"

save wf-lfp2, replace

log close
```

wfx-step2V2.do: robust version

```
log using wfx-step2V2-robust, replace

use wf-lfp2, clear
logit lfp hask5 hask618 age wc hc lwg inc, nolog

log close
```

Making do-files robust

Exclude anything specific to your computing environment

Exclude directory information

If you hard code directory names your program won't run on a computer with a different directory structure. I suggest:

```
use mydata, clear
```

Not:

```
use d:\work\mydata, clear
```

Files provided to journals

Michael Frisby, Directory of ISCC, was contacted in Nov 2015 by a client:

> I just heard from the journal editors I submitted my article to, and they told me that the R script includes a command to:
>
> setwd("I:\\Clients\\Client_Name\\August 2015")
>
> They would like to have a code that does not rely on an absolute file path.

What if data must be in a specific location?

1. Sometimes you cannot put a dataset in your working directory.
2. You do *not* want your do-file to include:

```
cd q:\TXRDC\census2000\PUMS
```

   Or:

```
use d:\data\binlfp3, clear
```

3. A solution is discussed after we consider global macros.

Results must not depend on information “left” in memory

1. If a script file depends on information not created by the script file, the program is fragile.
2. Results that could be in memory:
   - Datasets: never assume data is in memory.
   - Returns from prior commands
   - Variables created by another command
   - System settings

Version control

1. Different versions of Stata can give different results.
2. The **`version`** command tells Stata which results you want.
3. If a do-file containing **`version 6`** is run in Stata 10, you *usually* get the same answer that you would get in Stata 6.
4. If your include **`version 10`** and run it in Stata 8:

```
. version 10
this is version 8.2 of Stata; it cannot run version 10.0 programs
You can purchase the latest version of Stata by visiting
http://www.stata.com.
r(9);
```

Include seeds for random numbers

1. Random numbers are generated by a formula.
2. If you start with the same *seed* you get the same RNs.
3. To replicate your results, you need to start with the same seed:

```
set seed <integer>
```

Summary of robust do-files

1. Robust do-files are *necessary* for reproducible results.
2. They make debugging easier.
3. The rules above help make your scripts robust, but are not perfect.

What you can’t control

1. Software can change in ways that you cannot control.
   - Ideally, backup the software you used, especially user written commands.
2. The flavor of Stata can affect results in minor ways.
3. Collaborators might write fragile do-files.

Legible do-files

What makes a script file legible

1. Clearly and uniformly formatted.
2. Effective internal comments.
3. Output that is easy to read.
4. Do-files have the style and same structure.

Procedures for legible script files

1. Review do-files a day or two after you write them
   - They usually look good right after you run them
2. Use a template so you don’t forget things and the files are uniform
   - This also save time
3. Collaborators should agree on a template.

Comments

1. Comments explain what you are doing.
2. When I start a do-file, comments outline what I need to do.
3. I write brief comments as I write code.
4. Before posting, I check the work and add, revise, or delete comments.

Four types of for comments

```
* hc & hc do not assume respondent graduated

// select sample based on age and gender
logit lfp wc hc education income edhs edcol /// no HS is exclude
          i.race female

/*
  Analyses are preliminary based on
  countries with complete data on 2005-01-17.
*/
```

Comment indicating location

```
// #1 check data
:::
// #2 desc stats
:::
// #3 baseline models
```

Used for provenance

Location comments are used when documenting the provenance of a result.

Used for debugging

It is easy to “fix” the a command at the wrong location creating a new error instead of fixing old error.

Used with collaborators

John, In swb-02b #3, I'm concerned that the minimum age is 23, not 25. Did something change? Let me know before I proceed. Scott

---

Obscure comments make things more confusing

```
* check this. wrong variable?
* see ekp's comment on model specification
```

Worthless comments are distracting and waste time

```
* create age squared
gen agesq = age*age if age<=.
```

Good names avoid the need for comments

1. If you chose names wisely, many comments are not needed.
2. The comment

   `* gender: 1 is female`

   is not needed if the variable is called `female` and you use this rules:
3. Rule for naming binary variables:

   Name binary variables as the category that is equal to 1.

Alignment and indentation

1. Formatting similar commands

Option 1

```
rename dev origin
rename major jobchoice
rename HE parented
rename interest goals
rename score testscore
rename sgbt sgstd
```

Option 2

```
rename  dev       origin
rename  major     jobchoice
rename  HE        parented
rename  interest  goals
rename  score     testscore
rename  sgbt      sgstd
rename  restrict  restrictions
```

---

2. Formatting commands that take multiple lines

Option 1

```
logit y var01 var02 var03 var04 var05 var06 ///
var07 var08 var09 var10 var11 var12 var13 var14 var15
```

Option 2

```
logit y var01 var02 var03 var04 var05 var06 ///
        var07 var08 var09 var10 var11 var12 ///
        var13 var14 var15
```

NEVER let lines wrap

1. Mistakes are easy if you can't see the complete command.
2. Someone sent me a problem with `listcoef`. Their `mlogit` command was 182 characters long:

```
mlogit jobchoice income origin prestpar aptitude siblings fr...
listcoef
```

3. Here I show the error with only 3 outcomes (they had 8):

This looks OK

```
Variable: income (sd=1.1324678)
Odds comparing    |
Alternative 1     |
to Alternative 2  |       b         z     P>|z|     e^b   e^bStdX
------------------+---------------------------------------------
2       -3        |   0.49569    0.825   0.409   1.6416   1.7530
2       -1        |   0.68435    2.483   0.013   1.9825   2.1706
3       -2        |  -0.49569   -0.825   0.409   0.6092   0.5704
3       -1        |   0.18866    0.377   0.706   1.2076   1.2382
1       -2        |  -0.68435   -2.483   0.013   0.5044   0.4607
1       -3        |  -0.18866   -0.377   0.706   0.8281   0.8076
----------------------------------------------------------------
```

This must be wrong

```
Variable: female (sd=.50129175)

Odds comparing    |
Alternative 1     |
to Alternative 2  |       b         z     P>|z|     e^b   e^bStdX
------------------+---------------------------------------------
2       -1        |   1.25085    1.758   0.079   3.4933   1.8721
1       -2        |  -1.25085   -1.758   0.079   0.2863   0.5342
----------------------------------------------------------------
```

4. I found no errors in my program `listcoef.ado`.
5. Eventually, I reformatted their do-file:

```
mlogit jobchoice income origin prestigepar aptitude siblings friends ///
    scale1_std demands interestlvl jobgoal scale3 scale2_std motivation ///
    parented city female, noconstant basecat(1)
```

6. The problem was caused by an invalid option.
7. When consulting or debugging your own code, start by reformatting the script file to make it legible.

lim abb: limit abbreviations

Variable abbreviations

1. The *shortest unique abbreviation* is valid in Stata.

   a. If only **`age_at_1st_survey`** begins with **`a`**

   ```
   sum a
   :::
   ```

   b. If you have another variable starting with **`a`**, say **`agesq`**

   ```
   sum a
   a ambiguous abbreviation
   r(111);
   ```

2. Abbreviations lead to perplexing problems.

   a. I routinely used the names for categories of BMI

   ```
   bmi1      bmi2      bmi3      bmi4
   ```

   b. I forgot that the full names were

   ```
   bmi1_1019 bmi2_2024 bmi3_2530 bmi4_31up
   ```

---

   c. I got this error

   ```
   . svy: mean bmi1, over(black)
   test [bmi1]black = [bmi1]white
   equation [bmi1] not found
   r(303);
   ```

   d. I knew the names were right so **`test`** must not work with **`svy: mean`**.

   e. Eventually I discovered that **`svy: mean`** allows abbreviations in names of variables but **`test`** does not.

   f. This works:

   ```
   test[bmi1_1019]black = [bmi1_1019]white
   ```

3. Use variable abbreviations with care.

4. Choose names that are not too long to type -- details later.

5. To enter long names:

- Click the names in the Variables Window
- Then copy the names from the Command Window to your do-file

---

Command abbreviations

1. Instead of this: **`summarize education`**

2. You can use: **`sum e`**

3. A compromise is: **`sum educ`**

Try to use at least 3 letters

Full command name	→	*Abbreviation*
`generate`	→	`gen`
`label define`	→	`lab def`
`label values`	→	`lab val`
`label variable`	→	`lab var`
`quietly`	→	`qui`
`summarize`	→	`sum`
`tabulate`	→	`tab`
`display`	→	`di`

Legible log files

Mistake 1: truncate right

```
           |                          Years of education
Occupation |          3          6          7          8          9         10
-----------+-------------------------------------------------------------------...
    Menial |          0          2          0          0          3          1
           |       0.00       6.45       0.00       0.00       9.68       3.23
-----------+-------------------------------------------------------------------...
   BlueCol |          1          3          1          7          4          6
           |       1.45       4.35       1.45      10.14       5.80       8.70
-----------+-------------------------------------------------------------------...
     Craft |          0          3          2          3          2          2
           |       0.00       3.57       2.38       3.57       2.38       2.38
-----------+-------------------------------------------------------------------...
  WhiteCol |          0          0          0          1          0          1
           |       0.00       0.00       0.00       2.44       0.00       2.44
-----------+-------------------------------------------------------------------...
      Prof |          0          0          1          1          0          0
           |       0.00       0.00       0.89       0.89       0.00       0.00
-----------+-------------------------------------------------------------------...
     Total |          1          8          4         12          9         10
           |       0.30       2.37       1.19       3.56       2.67       2.97
```

Mistake 2: wrap

```
           |                                    Years of education
Occupation |          3          6          7          8          9         10
        11         12         13 |     Total
-----------+-------------------------------------------------------------------
----------------------------------+----------
    Menial |          0          2          0          0          3          1
         3         12          2 |        31
           |       0.00       6.45       0.00       0.00       9.68       3.23
      9.68      38.71       6.45 |    100.00
-----------+-------------------------------------------------------------------
----------------------------------+----------
   BlueCol |          1          3          1          7          4          6
         5         26          7 |        69
           |       1.45       4.35       1.45      10.14       5.80       8.70
      7.25      37.68      10.14 |    100.00
-----------+-------------------------------------------------------------------
----------------------------------+----------
     Craft |          0          3          2          3          2          2
         7         39          7 |        84
           |       0.00       3.57       2.38       3.57       2.38       2.38
      8.33      46.43       8.33 |    100.00
-----------+-------------------------------------------------------------------
----------------------------------+----------
```

Mistake 3: smcl printed as plain ascii

```
           {txt}{c |}                  Years of education
Occupation {c |}         3          6          7          8          9 {c |}
Total
{hline 11}{c +}{hline 55}{c +}{hline 10}
    Menial {c |}{res}        0          2          0          0          3
{txt}{c |}{res}       31
{txt}   BlueCol {c |}{res}        1          3          1          7
4 {txt}{c |}{res}       69
{txt}     Craft {c |}{res}        0          3          2          3
2 {txt}{c |}{res}       84
{txt}  WhiteCol {c |}{res}        0          0          0          1
0 {txt}{c |}{res}       41
{txt}      Prof {c |}{res}        0          0          1          1
0 {txt}{c |}{res}      112
{txt}{hline 11}{c +}{hline 55}{c +}{hline 10}
     Total {c |}{res}        1          8          4         12          9
{txt}{c |}{res}      337

           {txt}{c |}                  Years of education
Occupation {c |}        10         11         12         13         14 {c |}
Total
{hline 11}{c +}{hline 55}{c +}{hline 10}
    Menial {c |}{res}        1          3         12          2          7
{txt}{c |}{res}       31
{txt}   BlueCol {c |}{res}        6          5         26          7
3 {txt}{c |}{res}       69
```

Solution: limit line size

```
log using <filename>.log, replace text
set linesize 80
```

Do you need to specify text?

1. You can make text the default:

   `set logtype text, permanently`

2. This makes your output fragile since other users might not have this default
3. Solution: Include the `text` option in your `log` command

Is a longer linesize every appropriate?

1. If you always print or view output in wider formats that do not wrap, a larger linesize is appropriate.
2. Make sure others can read/print the files.

Templates for script files

1. The more uniform your do-files, the less likely you are to make errors and the easier it is to read the script and the output.
2. Standards for what to include in every do-file prevents errors and saves time.
3. I recommend using a *do-file template*.

Workflows for using templates

Copying templates

1. Your templates is called `wfx-template-dofile.do` in `\Templates`
2. Copy `wfx-template-dofile.do` to your working directory with the name of the do-file you want to create.

Automation

1. Macros can be used to insert the template into your editor.

What belongs in every do-files

1. Commands to clear memory to make the file robust
2. Commands to control formatting of output
3. Version control
4. Provenance of *who* ran *what* do-file *when* and *why*.
5. Comments that explain what is not “obvious” and highlight key findings.
6. Location comments to make it easier to document provenance.

Do-file wfx-template-dofile.do (details follow)

```
capture log close
log using name, replace text
set linesize 80
version 14.1
clear all
macro drop _all
set scheme s1manual

// keyword: task description
local pgm name
local dte 2017-03-16
local who Scott Long
local tag "`pgm'.do `who' `dte'"

// #1

// #2

log close
exit
```

Details follow

Introduction to locals macros

display

1. `display` displays things.

```
. display "Workflow, not slow. --Bruce Frasier"
Workflow, not slow. --Bruce Frasier
```

Local macros

1. Macros are *abbreviations* representing characters or number.

   local *local-name* "*string*"

2. For example,

   `local dte 2017-03-16`

3. To display a local

```
. di "date is: `dte'"
date is: 2017-03-16
```

4. **Note**: opening quote ` and closing quote ' are different.

Details on do-file template

```
log using name, replace text
set linesize 80
```

1. Open the log file with the same name as the do-file.
2. `replace` replaces the log if it exists.
3. `text` creates plain ASCII, not SMCL.
4. `linesize` prevents wrapping

```
version 14.1
```

Run the program to emulate Stata 14.1 when using later versions of Stata.

```
clear all
macro drop _all
```

Reset Stata to how it was when it was started (with a few exceptions).

```
// CWH: explore missing
local pgm cwh-data03-misschk
local dte 2016-01-28
local who Scott Long
local tag "`pgm'.do `who' `dte'"
```

1. `// CWH: explore missing` documents what the file is doing.
2. Local `pgm` is metadata that allows you to search for the file if it is accidentally renamed.
3. Local `tag` is used for *provenance*. Details later.

```
// #1
```

Sections of code are numbered for easy reference.

```
log close
```

Stop recording information to the log file opened in line 2.

```
exit
```

1. Stata executes a command only after it encounters with Enter.
2. Without enter, `log close` is not run.
3. `exit` makes sure there is an enter after `log close`.
4. `exit` also stops the do-file from executing anything later in the do-file.

```
capture log close
```

1. `capture` tells Stata to ignore errors caused by a command.
2. Accordingly, if you try to close a log that isn't open, the do-file still runs.
3. If a log is open, it is closed, preventing the error that the log is open.

Do-files and the project diary

1. A project diary and effective do-files are critical for reproducible results.

2. Your project diary should record:
   a. The date you ran the do-file
   b. The name of the do-file
   c. Sometimes the name of the dataset is useful
   d. Anything that isn't obvious by looking at the do-file (e.g., why this step is critical; why the do-file is correct, even though it seems to be a mistake)

3. The do-file documents exactly what you did.

4. This work only if do-files are:
   a. Clearly and uniquely named.
   b. Fully documented.
   c. Posted and preserved.

Summary of do-files

1. Create a template that incorporates principles of robustness and legibility.
   o Collaborators should agree on the template.

2. Before posting a do-file, revise for legibility and refine comments.
   o Then re-run it to make sure it still works.

3. Debugging do-files is considered later.

Part 9: Stata macros & returns

WFDAUS pages 83-92; *Guide to Automation*.

1. Automation saves time and prevent errors.

2. **Macros**: abbreviations for a string of characters or a number.

3. **Returns**: retrieved values left in memory by Stata commands.

4. Macros and returns improve efficiency, accuracy, and reproducibility.

5. We start with `display` so we can see what these tools do.

---

display

1. `display` displays things.

```
. display "Workflow, not slow. --Bruce Frasier"
Workflow, not slow. --Bruce Frasier
```

2. It is also a calculator:

```
. display 2*3
6
```

3. A very powerful calculator, where `display` is abbreviated `di`:

```
. di exp(cos(2^4))
.3837901
```

4. We use `display` to show the content of macros and returns.

Local macros

1. Macros are *abbreviations* representing characters or numbers.
2. Syntax:

   local *local-name* "*string*"

   local *local-name* = *expression*

3. For example,

```
local rhs "var1 var2 var3 var4"
local ncases = 198
```

4. To display a local:

```
local moptions "msym(square circle) mcol(red blue) jitter(5)"

. di "moptions: `moptions'"
moptions: msym(square circle) mcol(red blue) jitter(5)
```

5. **Note**: the opening quote ` and closing quote ' are different.

Why is it called local?

1. Local macros exist only when a do-file is running.
   - When that program ends, the macro *disappears*.
2. This makes do-files robust since everything is defined in the do-file.
3. If you use locals, *run the entire file, not parts*.

A tag for provenance

1. My program includes:

```
local pgm wflec01
local dte 2017-06-16
local who Scott Long
local tag "`pgm'.do `who' `dte'"
```

2. I can display the tag:

```
. di "The tag is: `tag'"
The tag is: wflec01.do Scott Long 2017-06-16
```

3. It is *essential* to understand how to create a tag in your do-files since tags are used extensively to document provenance.

Locals for tabulate options

1. I set my options and enter two `tabulate` commands:

```
local opt_tab "cell miss chi2"
tabulate wc hc,  `opt_tab'
tabulate wc lfp, `opt_tab'
```

   Which is equivalent to:

```
tabulate wc hc,  cell miss chi2
tabulate wc lfp, cell miss chi2
```

2. To change the options for multiple tabulates, I only change the local:

```
local opt_tab "row miss chi2"
tabulate wc hc, `opt_tab'
tabulate wc lfp, `opt_tab'
```

   Which is equivalent to:

```
tabulate wc hc,  row miss chi2
tabulate wc lfp, row miss chi2
```

Locals for using the same variables

1. I start with:

```
summarize lfp k5 k618 age wc hc lwg inc
logit     lfp k5 k618 age wc hc lwg inc
```

2. To change the variables I can edit both lines:

```
summarize lfp k5 k618 age wc lwg inc
logit     lfp k5 k618 age wc lwg inc
```

3. Alternatively, define the local `myvars`:

```
local myvars "lfp k5 k618 age wc hc lwg inc"
summarize `myvars'
logit     `myvars'
```

4. To change variables and rerun things I only need to change the local:

```
local myvars "lfp k5 k618 age wc lwg inc "
summarize `myvars'
logit     `myvars'
```

Locals for nested models

1. I create locals with sets of variables:

```
. local set1_age  "age agesq"
. local set2_educ "wc hc"
. local set3_kids "k5 k618"
```

2. Specify four nested models:

```
. local model1 "`set1_age'"
. display "model1: `model1'"
model1: age agesq

. local model2 "`model1' `set2_educ'"
. display "model2: `model2'"
model2: age agesq wc hc

. local model3 "`model2' `set3_kids'"
. display "model3: `model3'"
model3: age agesq wc hc k5 k618
```

3. Using these locals, estimate nested logits:

```
logit lfp `model1'
logit lfp `model2'
logit lfp `model3'
```

4. If I decide I do not want a squared term, I only need one change:

```
local set1_age "age"
```

5. If you do not use locals, you need to find errors like these...

```
logit y black
logit y black age10 age10sq edhs edcollege edpost incdollars childsqrt
logit y black age10 age10sq edhs edcollege edpost incdollars ///
    childsqrt bmi1 bmi3 bmi4 menoperi menopost mcs_12 pcs_12
logit y black age10 age10sq edhs edcollege edpost incdollars ///
    childsqrt bmi1 bmi3 bmi4 menoperi menopost mcs_12 ///
    pcs_12 sexactsqrt phys8_imp2 subj8_imp2
logit y black age10 age10sq edhs edcollege edpost incdollars ///
    childsqrt bmi1 bmi3 bmi4 menoperi menopost mcs_12 ///
    pcs_12 sexactsqrt phys8_imp2 subj8_imp2 selfattr partattr
```

Creating long strings with macros

1. Keep your command lines less than 80 columns.

2. Avoid things like this:

```
local demvars "fem black hisp age agesq edHS edc......
```

Using /// for long locals

1. The `///` terminator continues the command on the next line:

```
local myvars female black age agesq ///
  edHS edcol edpost incdollars "
```

2. Everything after `///` before the return is a comment:

```
local myvars female black /// demographics
  age agesq               /// quadratic in age
  edHS edcol edpost       /// education
  incdollars
```

3. What happens if you added `///` after `incdollars`?

Building long macros

1. First part of the macro

```
. local demvars "female black hispanic age agesq"
. di "demvars: `demvars'"
demvars: female black hispanic age agesq
```

2. Add new information to existing macro

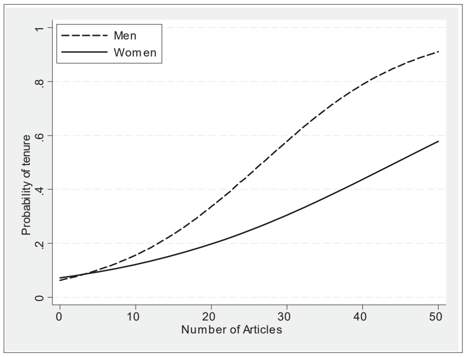
```
. local demvars "` demvars edHS edcol edpost"
. di "demvars: `demvars'"
demvars: female black hispanic age agesq edHS edcol
>  edpost
```

3. And again...

```
. local demvars "`demvars' incdollars childsqrt"
. di "demvars:`demvars'"
demvars: female black hispanic age agesq edHS edcol
>  edpost incdollars childsqrt
```

Using locals with `graph`

1. Options for `graph` are complicated even for a simple graph:



2. Here are the commands:

```
graph twoway ///
  (connected pr_women articles, lpat(solid) lwid(medthick) ///
      lcol(black) msym(i)) ///
  (connected pr_men articles,   lpat(dash)  lwid(medthick) ///
      lcol(black) msym(i)) ///
  , ylab(0(.2)1., grid glwid(medium) glpat(dash)) ///
    xlab(0(10)50) ytitle("Probability of tenure") ///
    legend(pos(11) order(2 1) ring(0) cols(1))
```

3. I create locals with sets of options:

```
* Female line options
local opt_linF "lpat(solid) lwid(medthick) lcol(black) msym(i)"

* Male line options
local opt_linM "lpat(dash)  lwid(medthick) lcol(black) msym(i)"

* Options for y grid marks
local opt_ygrid "grid glwid(medium) glpat(dash)"

* Options for legend in upper left
local opt_legend "pos(11) order(2 1) ring(0) cols(1)"
```

4. Now my graph command is:

```
graph twoway ///
  (connected pr_women articles, `opt_linF')  ///
  (connected pr_men   articles, `opt_linM')  ///
  , xlab(0(10)50) ylab(0(.2)1., `opt_ygrid') ///
    ytitle("Probability of tenure") legend(`opt_legend')
```

5. If I want colored lines, I change the locals:

```
local opt_linF "lpat(solid) lwid(medthick) lcol(red)  msym(i)"
local opt_linM "lpat(dash)  lwid(medthick) lcol(blue) msym(i)"
```

Global macros

1. Global macros are created as:

```
global vars "x1 x2 x3"
```

2. Content is retrieved using $*globalname*

```
display "$vars"
```

Globals can make do-files fragile

1. Globals stay in memory until you delete them or leave Stata.
2. Do-files are fragile when they require a global created *outside* of the do-file.
3. `macro drop _all` in your do-file makes the do-file robust.

Globals when running parts of a do-file

1. This code works when the entire do-file is run:

```
local vars "x1 x2 x3"
regress `vars’
```

2. If you only run the **`regress`** command by highlighting it in the Stata editor, you get an error since the local **`vars`** is not in memory:

```
regress `vars’
```

3. This would not be a problem with a global:

```
global vars "x1 x2 x3"
regress $vars
```

Data on a network

1. If you hard code the directory in your do-file, the do-file is fragile.
2. Sometimes data must reside on a location that is not your working directory.
3. One solution is to create a global *outside* of the do-file with the path.
4. Use that global inside the do-file to use the data.
5. If the global is not defined, the data is loaded from the working directory.
6. You must not include **`macro drop _all`** in the do-file.
7. For example…

Directory locations using a global

1. The data must be located here:

```
q:\TXRDC\census2000\PUMS
```

2. This makes the do-file fragile:

```
use q:\TXRDC\census2000\PUMS\file9112, clear
```

3. *Outside* of the do-file, run:

```
global path "q:\TXRDC\census2000\PUMS\"
```

4. *Inside* the do-file, do not drop macros:

```
* macro drop _all
```

5. *Inside* the do-file, use the data with:

```
use ${path}file9112, clear
```

6. We use **`${ }`** since:
   - **`$pathfile9112`** look for a global named **`pathfile9912`**
   - **`$path file9112`** decodes with a space that causes an error:

```
q:\Stat612\S612Resources\Datasets\◦file9912
```

Spaces in directory names

1. If you have a space in the path name, you cannot:

```
use g:\Box Sync\Datasets\binlfp4, clear
```

2. You must:

```
use "g:\Box Sync\Datasets\binlfp4", clear
```

3. We create the global for the path as:

```
global path g:\Box Sync\Datasets\
```

4. Then:

```
use "${path}binlfp4", clear
```

Returns

1. Stata commands leave results in memory called *returns*.

```
. sum age

  Variable |    Obs        Mean    Std. Dev.       Min        Max
-----------+---------------------------------------------------------
       age |    753    42.53785    8.072574         30         60

. return list

scalars:
                  r(N) =  753
              r(sum_w) =  753
               r(mean) =  42.53784860557769
                <snip>
```

2. I can put returns into locals.

```
. local agemean = r(mean)

. di "Mean: `agemean'"
Mean: 42.53784860557769
```

ereturns from estimation commands

```
. logit lfp k5 k618 age wc hc lwg inc
<snip>

. ereturn list
scalars:
               e(rank) =  8
                  e(N) =  753
                 e(ic) =  4
                  e(k) =  8
               e(k_eq) =  1
               e(k_dv) =  1
          e(converged) =  1
                 e(rc) =  0
                 e(ll) =  -452.632957530274
         e(k_eq_model) =  1
               e(ll_0) =  -514.8732045671461
               e(df_m) =  7
               e(chi2) =  124.4804940737441
                  e(p) =  8.92311460517e-24
              e(N_cdf) =  0
              e(N_cds) =  0
               e(r2_p) =  .1208846109775658
```

```
macros:
            e(cmdline) : "logit lfp k5 k618 age wc hc lwg inc"
                e(cmd) : "logit"
          e(estat_cmd) : "logit_estat"
            e(predict) : "logit_p"
     e(marginsnotok) : "stdp DBeta DEvi <snip> Number Resid.."
              e(title) : "Logistic regression"
           e(chi2type) : "LR"
                e(opt) : "moptimize"
                e(vce) : "oim"
               e(user) : "mopt__logit_d2()"
          e(ml_method) : "d2"
          e(technique) : "nr"
              e(which) : "max"
             e(depvar) : "lfp"
         e(properties) : "b V"
matrices:
                  e(b) :  1 x 8
                  e(V) :  8 x 8
                e(mns) :  1 x 8
              e(rules) :  1 x 4
               e(ilog) :  1 x 20
           e(gradient) :  1 x 8
functions:
             e(sample)
```

Example: Centering a variable

1. I want to center **age** (i.e., subtract the mean):

```
. summarize age

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |       753    42.53785    8.072574         30         60
```

2. I type the mean from **summarize** with **gen**:

```
. gen age_mean = age - 42.53785
. summarize age_mean

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    age_mean |       753   -1.49e-06    8.072574  -12.53785   17.46215
```

- **e-#** means "move the decimal to the left # digits"

```
-1.49e-06 = -000001.49e-06 = .00000149
               ^^^^^^          ^^^^^^
               123456          123456
```

---

3. Now I use the returned mean:

```
. summarize age

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |       753    42.53785    8.072574         30         60

. return list

scalars:
                  r(N) =  753
              r(sum_w) =  753
               r(mean) =  42.53784860557769
                r(Var) =  65.16645121641095
                 r(sd) =  8.072574014303674
                r(min) =  30
                r(max) =  60
                r(sum) =  32031

. gen age_meanV2 = age - r(mean)
. summarize age_mean age_meanV2

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    age_mean |       753   -1.49e-06    8.072574  -12.53785   17.46215
  age_meanV2 |       753    6.29e-08    8.072574  -12.53785   17.46215
```

---

4. I could get more precise using double precision:

```
. summarize age

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |       753    42.53785    8.072574         30         60

. gen double age_meanV3 = age - r(mean)
. label var  age_meanV3 "age - mean(age) using double precision"
. summarize  age_mean age_meanV2 age_meanV3

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    age_mean |       753   -1.49e-06    8.072574  -12.53785   17.46215
  age_meanV2 |       753    6.29e-08    8.072574  -12.53785   17.46215
  age_meanV3 |       753    3.14e-15    8.072574  -12.53785   17.46215
```

Aside: Numerical precision

See the Stata blog (blog.stata.com/2011/06/17/precision-yet-again-part-i/) for a *very* detailed discussion.

Part 10: Datasets

WFDAUS pages 136-141, 260-271.

Overview

Before we turn to variables, we look at datasets which are containers to hold variables and metadata about the dataset. We consider:

1. Naming
2. Metadata
3. Saving
4. Tracing the provenance of a dataset
5. Merging datasets

Naming datasets

When you change a dataset, save it with a new name.

1. If you don't follow the rule, you violate the posting principle.
2. The template I use for naming datasets is:

   ***dataset-name##-description*.dta**

   where **##** changes with every change to the contents.
3. Example:

Source of dataset	*Name of dataset*
Original dataset	`mydata01-source.dta`
`data01.do`: add variable labels	`mydata02-labels.dta`
`data02.do`: adds scales	`mydata3-scales.dta`

Never name it final!

1. Naming something final, doesn't make it final.
   - o It can be submitted, published, first draft, but not final.
2. I received `final14.dta`! Is there a `final15.dta`?
3. Someone showed this text:

   "**Urgent**: don't analyze `final.dta`, use `lastversion.dta` for our presentation tomorrow."

   Which makes me wonder about `alldonewithproject.dta`.
4. A file's date stamp is not reliable for determining the most recent dataset.

Metadata for datasets

1. Metadata is data about data.
   - o It is stored within the dataset so it travels with the data.
   - o It is critical for an effective workflow.
2. When you save a dataset, *always* use these commands to add metadata:
   - a. **label data**: to identify the data when it is used.
   - b. **note**: to document the provenance of the dataset.

---

label data

1. Example using our local `dte`:

```
label data "cwh01.dta | CWH analysis file | `dte'"
save cwh01, replace
```

   Recall that `local dte` was run earlier.

2. The data label is echoed when you use the data:

```
. use cwh01, clear
(cwh01.dta | CWH analysis file | 2006-12-07)
```

3. I like to include the dataset name in the label, but you might prefer:

```
. label data "CWH analysis file | `dte'"
. save cwh01, replace
. use cwh01, clear
(CWH analysis file | 2006-12-07)
```

Use note: for provenance

1. The syntax for adding a note to the dataset is:

   note: *note*
2. Before saving a dataset, add a note indicating:
   - a. The name of the dataset
   - b. A brief description
   - c. Who created the dataset, when, and with what script file
3. Our tag local simplifies this:

```
note: base01.dta | `tag' | base vars birthyr & cohort
```

   Since `local tag` is in every do file, adding provenance is very easy.
4. Notes help you fix errors since it tells you which do-file to fix.

Example of notes in a dataset that took years to create

```
. notes _dta

_dta:
1.  base01.dta | base01a.do jsl 2001-05-31 | base vars birthyr & cohort
2.  base02.dta | base01b.do jsl 2001-06-29 | add attrition info
    :::
    :::
    :::
38. ageism04.dta | age07b.do jsl 2006-06-27 | add analysis variables
```

1. It took 38 steps to create `ageism04.dta`.
2. Suppose I find a problem in the attrition variable:
   - a. Note 2 points to `base01b.do` from 2001-06-29.
   - b. I copy the posted `base01b.do` to `base01bV2.do` and fix the problem, creating `base02V2.dta`.
   - c. I make similar changes in other do-files needed to create `ageism04V2.dta`.

Workflow for saving datasets

1: Select observations.

2: Select variables.

3: Rearrange variables.

4: Add metadata.

5: Minimize file size.

6: Run diagnostics.

7: Add a data signature.

8: Save file with a new name.

Commands for saving data: details below

`0 :`	`local tag "`*do-file-name*`.do` *your-name date*`"`
`1a:`	`keep if` *exp*
`1b:`	`drop if` *exp*
`1c:`	`keep in` *numeric-list*
`1d:`	`drop in` *numeric-list*
`2a:`	`keep` *variable-list*
`2b:`	`drop` *variable-list*
`3a:`	`aorder` [*varlist*]
`3b:`	`order` *varlist*
`4a:`	`label data "`*name*`.dta |` *description* `|` *date*`"`
`4b:`	`note:` *name*`.dta	`tag' |` *describe what do-file did*
`5 :`	`quietly compress`
`6 :`	`<diagnostics:`
`7 :`	`datasignature`
`8 :`	`save` *mydata*`, version(#) replace`

0: Program tag

1. The local **`tag`** makes it easy to add provenance to notes and labels.

```
// wfclass: dataset example
local pgm myprogram
local dte 2017-06-17
local who Scott Long
local tag "`pgm'.do `who' `dte'"
```

2. The tag looks like this:

```
di "`tag'"
. myprogram.do Scott Long 2017-06-17
```

1: Select observations

1. Syntax

keep if *exp*

drop if *exp*

2. Examples:

```
keep if female==1
keep if age>40 & !missing(age)
keep if age>40
```

3. You can use **`in`** to select observations based on their rows:

keep in *start-row***/***end-row*

drop in *start-row***/***end-row*

---

* Deleting cases versus creating selection variables

1. Should you create different datasets for different groups you plan to analyze?

- One dataset with men and a second dataset with women?

2. If there are observations you will *never* need, you drop them.

- You are studing Russia and can drop cases from other countries.

3. If you plan to analyze subsets of observations, create **sample selection variables** that define the samples:

```
gen SAMPfem = (female==1)
label var SAMPfem "Sample: females only"

gen SAMPmale = (female==0)
label var SAMPmale "Sample: males only"

label def Lsamp 0 0_Drop 1 1_InSample
label val SAMPfem Lsamp
label val SAMPmale Lsamp
```

4. My sample selection variables are all named to being with **SAMP**. Why?

---

5. I load the dataset and select cases.

```
* analysis of women
use mydata, clear
keep if SAMPfem // women only
sum age inc

* analysis of men
use mydata, clear
keep if SAMPmale // men only
sum age inc
```

6. Or I can use if conditions in the analysis commands:

```
use mydata, clear
sum age inc if SAMPfem // women only
sum age inc if SAMPmale // men only
```

7. Creating selection variables is particularly useful with complex criteria:

```
gen SAMPf2040urb = (female==1) ///
  & (age>19 & age<41) & (urban==1)
label var SAMPf2040urb "Female, 20-40, and urban?"
label val SAMPf2040urb Lsamp
```

8. Then:

```
logit lfp income age wage kid5 if SAMPf2040urb
```

9. Or:

```
keep if SAMPf2040urb
```

2: Select variables

1. You can drop variables you are *certain* you will not need.

- Questions not asked in the country you are studying.

2. Do *not* drop source variables used to create new variables since you may want to later verify a variable later if you encounter a problem.

3. Commands:

drop *varlist*

keep *varlist*

* Selecting variables for the ISSP 2002 Russian data

1. The dataset has 234 variables, some of which are meaningless for the Russian sample.

2. I load the data (*wf6-save.do*):

```
. use wf-isspru01, clear
(Workflow data from Russian ISSP 2002 | 2008-04-02)
```

3. To find variables to drop:

```
. codebook, problems

Potential problems in dataset   wf-isspru01.dta

           potential problem   variables
------------------------------------------------------------------------------
constant (or all missing) vars   v1 v3 v206 v207 v208 v209 v210 v211 v212 v213
                               v214 v215 v216 v217 v218 v219 v220 v221 v222
                               v223 v224 v225 v226 v227 v228 v229 v230 v231
                               v233 v234 v235 v236 v237 v238 v248 v280 v287
                               v290 v291 v337 v358 v359 v360 v362
   incompletely labeled vars   v36 v37 v69 v71 v201 v204 v240 v243 v249 v250
                               v361
------------------------------------------------------------------------------
```

4. I use a return and a local to quickly drop all variables that have no variation:

```
* assign constant variable names to a local
local dropvars = r(cons)
drop `dropvars'
```

3: Rearrange variables

1. The order is shown in the *Variables Window* and in the *Data Editor*.

2. Order affects the output from some commands and which variables are selected by `xxxx-yyyy`.

3. Variables at the front are easy to select from the *Variables Window*.

a. You might want variables you are unlikely to use at the end of the dataset.

b. You might simply prefer to have variables arranged alphabetically.

4. To alphabetize names, I use:

```
order, alphabetic // alphabetize all variables
order id-variables // place these at the front
```

4: Internal documentation

1. Use `label data` to add a label shown when you load the data:

label data "*label*"

2. For example,

```
label data ///
    "wf-isspru01.dta | WF Russian ISSP 2002 | `dte'"
```

3. Use `notes:` for metadata about the dataset:

notes: *text*

4. For example,

```
note: wf-isspru02.dta | `tag' | Revise sample
```

5. To see the notes:

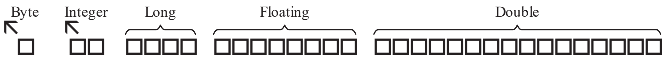
```
notes _dta
```

5: Compressing the file

1. By default variables are stored as **floating point** which holds values from

(-1.70141173319*10^38, 1.70141173319*10^36)

2. Stata has five storage types:



3. `compress` converts each variable to the most compact type that does not lose information.

4. This file was converted from SPSS to Stata without compression. Variables are double precision:

```
. use 04106-0001-data, clear
. dir 04106-0001-data.dta
83.4M 3/11/06 9:42 04106-0001-data.dta
```

5. The file is compressed in Stata:

```
. compress
v1 was double now int
v2 was double now long
(output omitted)
v361 was double now long
v362 was double now byte
```

6. Then,

```
. save mydata1, replace
file mydata1.dta saved
. dir mydata1.dta
11.3M 4/14/07 11:02 mydata1.dta
```

7. To suppress the list of compressed variables:

```
quietly compress
```

---

* 6: Diagnostics

`codebook, problems` looks for three things:

- Variables that have no variation which is not necessarily a problem.
- Variables with nonexistent value labels.
- Variables where only some values are labeled.

Example of codebook, problems

```
. use wf-diagnostics, clear
(Workflow data to illustrate data diagnostics | 2008-04-05)
. codebook, problems

   Potential problems in dataset   wf-diagnostics.dta

               potential problem   variables
--------------------------------------------------------------------
  constant (or all missing) vars   v3 v256 v265 v274 v283 v294 v303
     vars with nonexisting label   v7
       incompletely labeled vars   v36 v37
--------------------------------------------------------------------
```

---

1. Eight variables are constant. Since our sample includes only those from Russia, we do not want any variation in the country variable:

```
. tab1 v3, miss

-> tabulation of v3

    Country |      Freq.     Percent        Cum.
------------+-----------------------------------
        RUS |        100      100.00      100.00
------------+-----------------------------------
      Total |        100      100.00
```

2. This variable is only useful for the Bulgarian sample:

```
. tab1 v256, miss

-> tabulation of v256

 R: Party affiliation: |
              Bulgaria |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
                   NAV |        100      100.00      100.00
-----------------------+-----------------------------------
                 Total |        100      100.00
```

---

3. The second error indicates that the **`v7`** label does not exist:

```
. describe v7

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------
v7              byte   %10.0g      labv7      What women really want is
                                                home & kids
```

4. I mistakenly assigned **`labv7`** to variable **`v7`** instead of label **`v7`**.

5. The third error message points to gaps in a value label:

```
. tab1 v37, miss

 How many hrs spouse,partner |
                 works on hh |      Freq.     Percent        Cum.
-----------------------------+-----------------------------------
               NAP,no partner |         50       50.00       50.00
     1 hour or less than 1 hr |          1        1.00       51.00
                        2 hrs |          1        1.00       52.00
                              |          2        2.00       54.00
                            7 |          6        6.00       60.00
                            8 |          1        1.00       61.00
                            9 |          1        1.00       62.00
(output omitted)
```

6. To fix this I need to revise the value label definition assigned to this variable.

---

* 7: datasignature

1. **`datasignature set`** creates a string of **checksums** that describe the data.

- A checksum is a number summarizing some property of a file.

2. It is used to detect errors by computing the checksum with the current file and verifying that it matches the checksums saved as metadata in the file.

3. For example,

```
datasignature
753:8(54146):1899015902:1680634677
```

4. The signature is a function of:

- The number of observations and number of variables in the data
- The values of the variables
- The names of the variables
- The order in which the variables occur in the dataset
- The storage types of the individual variables

5. The signature is *not* a function of variable labels, value labels, notes, etc.

Why would a signature fail?

1. Fred used **`wf-datasig02.dta`** that I created on March 9, 2008.

2. He renamed a variable, changed the data label, and saved the changed data with the same name ("It is exactly the same—I only changed one variable."):

```
. use wf-datasig02, clear
(Workflow dataset for illustrating datasignature | 2008-04-03)
. rename k5 kids5
. save wf-datasig02, replace
file wf-datasig02.dta saved
```

3. When I load the dataset:

```
. use wf-datasig02, clear
(Workflow data for illustrating datasignature | 2008-04-03)
. datasignature confirm
 data have changed since 03apr2008 09:58
 r(9)
```

Creating a signature

1. To create a signature:

```
. datasignature set
753:8(54146):1899015902:1680634677  (data signature set)
. save wf-datasig02, replace
```

2. To verify the signature when loading the dataset:

```
. use wf-datasig02, clear
(WF dataset for illustrating datasignature | 2008-03-09)
. datasignature confirm
 (data unchanged since 09mar2008 12:40)
```

3. If you try to set a signature when one already exists:

```
. datasignature set
  data signature already set -- specify option -reset-
r(110);
```

You need to reset it:

```
. datasignature set, reset
753:9(85320):1280133433:4173826113 (data signature reset)
```

8: Save the file

Syntax

save *filename* [**, replace version(#)**]

- o *filename* is in quotes if there are spaces.
- o **`replace`** overwrites the dataset if it exists.
- o **`version()`** saves the file in versions 11-14 of Stata; you lose some features, but usually it will be fine.

After a file is saved

1. Check your documentation.

2. Decide if you are ready to post the file.

3. Backup your files.

---

* Combining datasets

1. There are many ways to combine datasets and reshape an existing dataset. Here we consider only the most basic.

2. For details, see Stata Corp [D] Stata Data Management Reference Manual.

Merging

1. You can merge two datasets to add variables or observations.

2. Often both files are for the same individuals (countries, firms, etc.).

Example of merging

1. A study of health and aging used the National Longitudinal Survey (NLS).

2. It took years to construct the variables we needed.

3. Since three people were working on the data, we divided variables into 7 topical areas and construct each dataset independently.

4. For analysis, variables from multiple files were combined into a single dataset.

Terminology for merging

1. Two files can be combined with the **`merge`** command.

a. The *master dataset* is in memory.

b. The *using dataset* is in a file stored on disk (i.e., it is used).

2. Match merging matches on values of an ID variable.

id-variable==1 in master dataset <-> *id-variable*==1 in using dataset

id-variable==2 in master dataset <-> *id-variable*==2 in using dataset

::::

If the *id-variable* is not a datasets, variables from that dataset are made missing.

3. One-to-one merging matches observations by position in the datasets.

Observation 1 in master dataset <-> *Observation* 1 in using dataset

Observation 2 in master dataset <-> *Observation* 2 in using dataset

::::

Match merging

Syntax: merge 1:1 *id-variable* **using** *filename*

1. *id-variable* is common to both datasets and contains unique IDs.

2. Datasets are sorted on *id-variable*.

3. Variable _**`merge`** is created to indicate where an observation came from.

Merging wf-nls-cntrl07.dta and wf-nls-flim05.dta

1. Check the datasets

```
. // #1 check signatures and load the master dataset

. use wf-nls-flim05, clear
(Workflow example with NLS FLIM variables \ 2008-04-02)

. datasignature confirm
  (data unchanged since 02apr2008 13:29)

. use wf-nls-cntrl07, clear // leave it in memory
(Workflow example with NLS control variables \ 2008-04-02)

. datasignature confirm
  (data unchanged since 02apr2008 13:29)
```

---

2. With master **`wf-nls-cntrl07.dta`** in memory, I merge the datasets:

```
. // #2 merge flim and control and check _merge

. merge 1:1 id using wf-nls-flim05

    Result                           # of obs.
    -----------------------------------------
    not matched                            21
        from master                        21  (_merge==1)
        from using                          0  (_merge==2)

    matched                                79  (_merge==3)
    -----------------------------------------
   . merge 1:1 id using wf-nls-flim05
```

3. Assuming that there was no problem with the 21 cases only in master:

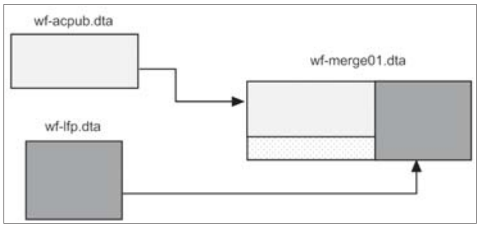
```
local dsn "wfx-data-merge01"
drop _merge
quietly compress
label data "`dsn'.dta | WF merged NLS flim & control variables |
`dte'"
note: `dsn'.dta / `tag' / example of one to one merging
datasignature set, reset
save `dsn', replace
```

One-to-one merging

1. Combine
    a. The 1st row of master dataset with the 1st row of using dataset
    b. The 2nd with the 2nd
    b. And so on.

2. There is no ID variable.

3. You can combine datasets that have nothing to do with one another.



Combing unrelated datasets without matching

1. I combine **`wf-lfp.dta`** on labor force participation and **`wf-acpub.dta`** on research productivity of biochemists.

2. The number of observations equals that of the larger **`wf-lfp.dta`**.

3. Extra observations for the **`wf-acpub.dta`** variables will be missing values.

```
. use wf-lfp, clear
(Workflow data on labor force participation | 2008-04-02)

. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         lfp |       753    .5683931    .4956295          0          1
          k5 |       753    .2377158     .523959          0          3
        k618 |       753    1.353254    1.319874          0          8
         age |       753    42.53785    8.072574         30         60
          wc |       753    .2815405    .4500494          0          1
          hc |       753    .3917663    .4884694          0          1
         lwg |       753    1.097115    .5875564  -2.054124   3.218876
         inc |       753    20.12897     11.6348  -.0290001         96
-------------+--------------------------------------------------------
```

```
. use wf-acpub, clear
(Workflow data on scientific productivity | 2008-04-04)

. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          id |       308    58654.49    2283.465      57001      62420
       enrol |       278     5.92446     2.92346          3         25
      female |       308    .3474026    .4769198          0          1
         phd |       308    3.177987    1.012738          1       4.77
         pub |       308    3.185065    3.908752          0         31
 enrol_fixed |       278    5.564748    1.467253          3         14
```

4. One-to-one merge the datasets:

```
. use wf-lfp, clear
(Workflow data on labor force participation \ 2008-04-02)
. merge 1:1 _n using wf-acpub

    Result                           # of obs.
    -----------------------------------------
    not matched                           445
        from master                       445  (_merge==1)
        from using                          0  (_merge==2)

    matched                               308  (_merge==3)
    -----------------------------------------
```

---

5. Some housekeeping:

```
. local dsn "wfx-data-merge02"
. drop _merge
. quietly compress
. label data "`dsn'.dta | match merge by observation | `dte'"
. note: `dsn'.dta / `tag' / example of match merging
. datasignature set, reset
  753:14(117189):528693629:719271906       (data signature reset)
. save `dsn', replace
(note: file wfx-data-merge02.dta not found)
file wfx-data-merge02.dta saved
```

6. Originally I had to load two datasets:

```
   use binlfp2, clear
   logit lfp k5 k618 age wc hc lwg inc
   use couart2.dta, clear
   nbreg art fem mar kid5 phd ment
```

7. Now I can use only one:

```
   use wfx-merge02.dta, clear
   logit lfp k5 k618 age wc hc lwg inc
   nbreg art fem mar kid5 phd ment
```

8. This might not be too useful, but it is the only time I can think of when you would do this type of merge *by choice*.

Forgetting to match merge

1. I want to combine a scientist's **biographical** data with her **bibliographic** data. After confirming data signatures, I run the commands:

```
. use wf-mergebio, clear
(Workflow biographical data to illustrate merging \ 2008-04-05)

. merge 1:1 _n using wf-mergebib

    Result                           # of obs.
    -----------------------------------------
    not matched                             0
    matched                               408  (_merge==3)
    -----------------------------------------
```

2. The descriptive statistics are *correct*

```
. codebook, compact

Variable   Obs Unique      Mean  Min       Max  Label
------------------------------------------------------------------------------
job        408     80  2.233431    1       4.8  Prestige of first job
fem        408      2  .3897059    0         1  Gender: 1=female 0=male
phd        408     89  3.200564    1       4.8  PhD prestige
ment       408    123  45.47058    0  531.9999  Citations received by mentor
id         408    408     204.5    1       408  ID number
art        408     14  2.276961    0        18  # of articles published
cit        408     87  21.71569    0       203  # of citations received
------------------------------------------------------------------------------
```

---

3. The correlations are *wrong*

```
. pwcorr job fem phd ment art cit

             |      job      fem      phd     ment      art      cit
-------------+------------------------------------------------------
         job |   1.0000
         fem |  -0.1076   1.0000
         phd |   0.3636  -0.0550   1.0000
        ment |   0.2129  -0.0100   0.3253   1.0000
         art |  -0.3471   0.0691  -0.9115  -0.2835   1.0000
         cit |  -0.2314  -0.0278  -0.6607  -0.2046   0.7340   1.0000
```

4. To fix the problem, I must match by **`id`**:

```
    use wf-mergebio, clear
    merge 1:1 id using wf-mergebib, sort
```

Useful data commands

append	Append datasets
cf	Compare two datasets
collapse	Make dataset of summary statistics
compress	Compress data in memory
contract	Make dataset of frequencies and percentages
cross	Form every pairwise combination of two datasets
expand	Duplicate observations
export	Overview of exporting data from Stata in various formats
import	Overview of importing data into Stata from various formats
joinby	Form all pairwise combinations within groups
merge	Merge datasets
outfile	Export dataset in text format

---

reshape	Convert data from wide to long form and vice versa
stack	Stack data
sysuse	Use shipped dataset
xmlsave	Export or import dataset in XML format
xpose	Interchange observations and variables
zipfile	Compress/uncompress files and directories

Before I start major data management work, I often review

STATA DATA MANAGEMENT REFERENCE MANUAL

to see if there is a command that will make the job easier.

One time datasets

1. For a dataset used once, give it the name of the do-file that created it.
2. **`demogcheck01.do`** merges datasets to verify that the demographic data from two sources are consistent.
3. I don't anticipate further analyses using this dataset, but I want to keep it in case I have questions later, so I name it **`demogcheck01.dta`**.

Temporary datasets

1. If I don't need to keep a dataset I create, I start the name with **`x-`**.
2. For example,
   a. I am merging **`demog05.dta`** and **`flim06.dta`**.
   b. **`fl-mrg01.do`** extracts variables to **`x-fl-mrg01.dta.`**
   c. **`fl-mrg02.do`** extracts variables to **`x-fl-mrg02.dta`**.
   d. **`fl-mrg03.do`** merges these to **`fl-paper01.dta`**.
   e. Later I delete the **`x-*.dta`** files.

Part 11: Variable names & labels

WFDAUS pages 143-195.

1. We consider commands to change names and add labels and notes.
2. The decisions you make will *help* or *haunt* you throughout the project.
3. *Plan* before you start.
   - *Get feedback* from others
   - *Let things age before finalizing decisions*
4. It is painstaking work that is worth the time.

New content requires a new variable

Never change an existing variable; create a new variable.

1. Never have two datasets with **`var27`** where the content of **`var27`** differs.
2. Suppose you need to recode **`var27`** to truncate values above 100.
   a. Do *not*:
   ```
   replace var27 = 100 if var27>100 // Wrong!
   ```
   b. Instead,
   ```
   gen var27trun = var27
   replace var27trun = 100 if var27>100
   ```
   c. Save the new variable in the next version of your dataset.
3. Suppose you find an error in **`var27`**.
   a. Create **`var27V2`** with the correct values.
   b. Save the new variable in the next version of your dataset.

Naming variables is hard

New variables need names!

General guidelines

1. Names should be informative and easy to use.
2. Plan names before analysis begins with analysis in mind.
   - Ad hoc names tend to be inconsistent and confusing.
3. If you use data collected by others, you can change the names.
   - You don't have to use R0033287 for the age of the respondent!

Systems for naming variables

1. Sequential naming
2. Source naming
3. Mnemonic naming
4. A combination of these systems

Sequential naming systems

1. A stub followed by digits:
   - ISSP uses names like v1, v2, v3,..., v362.
   - National Longitudinal Survey uses names like R0000100 and R0002203.
2. It is easy to use the wrong variable and difficult to interpret output.

   Is this the model I want?

```
logit R0051400 R0000100 R0002203 R0081000
```

   Or is this?

```
logit R0054140 R1000100 R0002208 R0081000
```

Source naming systems

1. Source names use information about a variable's origin as part of the name.
   - a. `q1`, `q2`, etc. or `q4c`. `q4a`, `q4b`, and `q4c`.
2. Datasets from cards use card number and column such as `c1c15`.
3. Source names do not convey content.

```
logit q54 q4 q5 q15
```

Mnemonic naming systems

1. Mnemonic names convey content.
2. I prefer:

```
logit lfp age educ kids
```

   to:

```
logit R0051400 R0000100 R0002203 R0081000
```

   or:

```
logit q17 q31 q19 q02
```

3. You want names that are short, unambiguous, and informative.
   - Finding good mnemonic names for 1000 variables is **hard**.
4. Mnemonic names created "on the fly" can be very confusing.
   - `WWCHWRK` is not much better than `R000134`.

Changing variable names

1. Source data has names chosen by others.
   - Names created when collecting data are rarely ideal for analysis.
2. It is often worth the time to rename variables.
3. Document the original name with `notes` or with `label var`.

```
rename R0013871 socialdist
label var socialdist "R0013871 social dist from person with MI"
note socialdist: renamed R0013871 / `tag’
```

4. *Does changing names violate the posting principle?*

Principles for selecting names

Anticipate looking for variables

1. `lookfor` searches for a string in the name or variable label.
2. There is a trade-off between short names and being able to find things.
3. If I name indicators `raceblck`, `racewhite`, and `raceasian`, then:

```
. lookfor race
                   storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------
racewhite       byte   %9.0g       Lyn        Is white?
raceblack       byte   %9.0g       Lyn        Is black?
raceasian       byte   %9.0g       Lyn        Is Asian?
```

4. If I use `black`, `white` and `asian`, then `lookfor race` does not find them.

Alphabetizing names

1. You can sort variables in memory with `order, alpha`.
2. Several variables measure social distance from someone with mental illness.
   - a. Questions ask about types of contact: having the person as a friend, having the person marry a relative, working with the person, and so on.
   - b. `friendsd`, `marrysd`, and `worksd` are not adjacent when sorted.
   - c. `sdfriend`, `sdmarry`, and `sdwork` are adjacent when alphabetized.
3. Which sets work better?

Set 1:	`raceblck`	`racewhite`	`raceasian`
Set 2:	`blckrace`	`whiterace`	`asianrace`
Set 3:	`black`	`white`	`Asian`

Set 1:	`edhs`	`edcol`	`edphd`
Set 2:	`hsed`	`coled`	`phded`
Set 3:	`highschool`	`college`	`phd`

Simple, unambiguous names

1. There is a trade-off between *length* and *clarity*.
   - a. `Q_23v` is short, but hard to remember and hard to type.
   - b. `socialdistancescale2` is descriptive, hard to type, truncated in output, and won't convert to some data formats.
2. In a large dataset, it is hard to find “perfect” names.
3. With planning you can find better names.

How long should a name be?

1. Names can be 32 characters long.
2. In output, names are often truncated to 12 characters.
3. I suggest

   Use names that are at most 12 characters long.

   Original names should be 10 or shorter to allow Vx.

Names too long

```
publication~9 |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  phdprestige |   .2143538   .0552267     3.88   0.000     .1061115    .3225961
workactivit~n |  -.1746687   .2235618    -0.78   0.435    -.6128418    .2635044
workactivit~g |   .1646048   .1266275     1.30   0.194    -.0835806    .4127902
publication~1 |    .138701   .0181383     7.65   0.000     .1031506    .1742515
        _cons |   .2482469   .2111628     1.18   0.240    -.1656246    .6621184
```

Names too short

```
          p9 |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
------------+----------------------------------------------------------------
         phd |   .2143538   .0552267     3.88   0.000     .1061115    .3225961
         wad |  -.1746687   .2235618    -0.78   0.435    -.6128418    .2635044
        wtch |   .1646048   .1266275     1.30   0.194    -.0835806    .4127902
          p1 |    .138701   .0181383     7.65   0.000     .1031506    .1742515
       _cons |   .2482469   .2111628     1.18   0.240    -.1656246    .6621184
```

Names too short

```
      pubyr9 |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
------------+----------------------------------------------------------------
     phdprst |   .2143538   .0552267     3.88   0.000     .1061115    .3225961
    adminjob |  -.1746687   .2235618    -0.78   0.435    -.6128418    .2635044
    teachjob |   .1646048   .1266275     1.30   0.194    -.0835806    .4127902
      pubyr1 |    .138701   .0181383     7.65   0.000     .1031506    .1742515
       _cons |   .2482469   .2111628     1.18   0.240    -.1656246    .6621184
```

Older software

1. Some packages only allow 8 characters.
2. What should you do?

Use clear & consistent abbreviations

1. Abbreviations make names shorter and sometimes clearer.
2. For example:

Set 1	Set 2	Set 3
educlths	**educationlths**	**ed_lths**
educhs	**educationhs**	**ed_hs**

3. Abbreviations can be ambiguous, so plan them and use them consistently.
4. Add a note to document abbreviations:

```
note _dta: ed=education in names / `tag’
note _dta: sd=social distance in names / `tag’
```

Use names that convey content

1. All else being equal, names with content are easier.
2. For binary variables, use names that indicate the category coded as 1.
3. For scales, names can indicate direction (P=positive, N=negative):

`dist1P`, **`sdist2N`**, and **`sdist3N`**

4. How do you indicate panel?

Set 1:	**p1wages**	**p2wages**	**p3wages**
Set 2:	**wagesp1**	**wagesp2**	**wagesp3**
Set 3:	**wages_1**	**wages_2**	**wages_3**
Set 4:	**wages_panel1**	**wages_panel2**	**wages_panel3**

Be careful with capitalization

1. Stata distinguishes case in names.
   - **educ**, **Educ** and **EDUC** are different variables
2. I avoid names that differ only by capitalization.
3. I use capitalizations to highlight features of variables, such as:

Letter	Meaning	Example
B	Binary variable.	`highschlB`
I	Indicator variable.	`edIhs`, `edIgths`, `edIcol`
L	Value labels used by multiple variables.	`Lyesno`
M	Indicator of data being missing.	`educM`
N	A negatively coded scale.	`sdworkN`
O	Too close to the number 0, so I don’t use it!	
P	A positively coded scale.	`sdkidsP`
S	The unchanged, source variable.	`educS`; `Seduc`
V	Version number for modified variables.	`marstatV2`
X	A temporary variable.	`Xtemp`

Try names before you decide

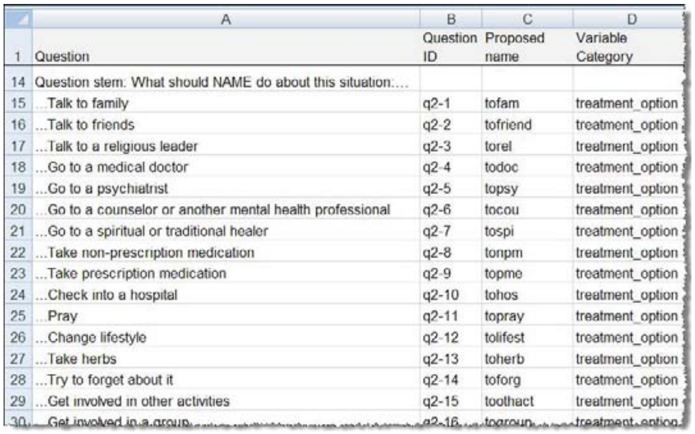
1. Selecting effective names and labels is an iterative process.
2. Check how the names work in the commands you plan to use.
3. Let others critique them.
4. Here’s an example…

Planning names

1. Before extracting variables, plan which variables you need and how to rename them. Try to extract all the variables at the same time.
2. The more complex the project, the more detailed your plan needs to be.

Planning names for the SGC

1. In a study of stigma in 17 countries, each collection center used source names for most variables and *most of the time* they used the same names.
2. We listed the original names and questions.
3. We classified variables into categories (e.g., questions about treatment; demographics; measures of social distance).
4. One person proposed mnemonic names that were circulated for comments.
5. After several iterations we had acceptable names.
6. Here is a portion of the spreadsheet:



	A	B	C	D
1	Question	Question ID	Proposed name	Variable Category
14	Question stem: What should NAME do about this situation:...			
15	...Talk to family	q2-1	tofam	treatment_option
16	...Talk to friends	q2-2	tofriend	treatment_option
17	...Talk to a religious leader	q2-3	torel	treatment_option
18	...Go to a medical doctor	q2-4	todoc	treatment_option
19	...Go to a psychiatrist	q2-5	topsy	treatment_option
20	...Go to a counselor or another mental health professional	q2-6	tocou	treatment_option
21	...Go to a spiritual or traditional healer	q2-7	tospi	treatment_option
22	...Take non-prescription medication	q2-8	tonpm	treatment_option
23	...Take prescription medication	q2-9	topme	treatment_option
24	...Check into a hospital	q2-10	tohos	treatment_option
25	...Pray	q2-11	topray	treatment_option
26	...Change lifestyle	q2-12	tolifest	treatment_option
27	...Take herbs	q2-13	toherb	treatment_option
28	...Try to forget about it	q2-14	toforg	treatment_option
29	...Get involved in other activities	q2-15	toothact	treatment_option
30	...Get involved in a group	q2-16	togroup	treatment_option

Variable labels

1. Variable labels are strings of up to 80 characters.

2. Stata commands use labels to document output.

3. Long labels are often truncated in output.

General principles

1. *Every* variable should have a variable label.

2. If a variable does not have a label, add one.

3. When you create a variable, *immediately* add a variable label.

- A quick label avoids the accumulation of stray variables.

```
label var checkvar "Scott's temp var; can be dropped"
```

Listing variable labels

You can prevent mistakes by starting analysis do-files by listing the names, labels, and descriptive statistics for the variables you are going to analyze.

Commands to examine variable labels

```
. use binlfp4,clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)

. local lhsvar lfp
. local rhsvars k5 k618 age wc hc lwg inc
```

codebook, compact

Syntax: **codebook** [*varlist*] [*if*] [*in*] **, compact**

```
. codebook `lhsvar' `rhsvars', compact

Variable   Obs Unique      Mean       Min       Max  Label
--------------------------------------------------------------------------------
lfp        753      2  .5683931         0         1  In paid labor force?
k5         753      4  .2377158         0         3  # kids < 6
k618       753      9  1.353254         0         8  # kids 6-18
age        753     31  42.53785        30        60  Wife's age in years
wc         753      2  .2815405         0         1  Wife attended college?
hc         753      2  .3917663         0         1  Husband attended college?
lwg        753    676  1.097115 -2.054124  3.218876  Log of wife's estimated...
inc        753    621  20.12897 -.0290001        96  Family income excluding...
```

---

nmlab and summarize (I wrote nmlab)

```
. nmlab `lhsvar' `rhsvars'

lfp   In paid labor force?
k5    # kids < 6
k618  # kids 6-18
age   Wife's age in years
wc    Wife attended college?
hc    Husband attended college?
lwg   Log of wife's estimated wages
inc   Family income excluding wife's

. sum   `lhsvar' `rhsvars'

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         lfp |        753    .5683931    .4956295          0          1
          k5 |        753    .2377158     .523959          0          3
        k618 |        753    1.353254    1.319874          0          8
         age |        753    42.53785    8.072574         30         60
          wc |        753    .2815405    .4500494          0          1
-------------+---------------------------------------------------------
          hc |        753    .3917663    .4884694          0          1
         lwg |        753    1.097115    .5875564  -2.054124   3.218876
         inc |        753    20.12897     11.6348  -.0290001         96
```

---

tabulate: to check variable and value labels

1. You see the variable and value labels but not the variable name:

```
. tabulate tcfam, missing

   Q43 How Impt: |
  Turn to family |
        for help |      Freq.     Percent        Cum.
-----------------+-----------------------------------
1Not at all Impt |          9        0.83        0.83
               2 |          4        0.37        1.20
```

(output omitted)

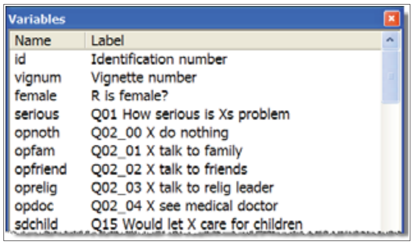
2. `tabulate` does not truncate long label:

```
. tabulate tcfam, missing

Question 43: How |
 important is it |
   to you to turn |
    to the family |
     for support? |      Freq.     Percent        Cum.
-----------------+-----------------------------------
1Not at all Impt |          9        0.83        0.83
               2 |          4        0.37        1.20
```

(output omitted)

The Variables Window

1. Variable labels are also shown here:



Name	Label
id	Identification number
vignum	Vignette number
female	R is female?
serious	Q01 How serious is Xs problem
opnoth	Q02_00 X do nothing
opfam	Q02_01 X talk to family
opfriend	Q02_02 X talk to friends
oprelig	Q02_03 X talk to relig leader
opdoc	Q02_04 X see medical doctor
sdchild	Q15 Would let X care for children

Syntax for label variable

1. Variable labels can be up to 80 characters:

label variable *varname* **"***label***"**

I often abbreviate this:

```
label var artsqrt "Square root of # of articles"
```

2. To remove a label do not specify the label:

```
label var artsqrt
```

Beware of truncation

1. A variable label should provide the essential information.

2. Put important things in the front of a label in case it is truncated.

Examples follow...

Bad labels

```
tc1fam     Q43 How important is it to turn to family for help
tc1friend  Q44 How important is it to turn to friends for help
tc1relig   Q45 How important is it to turn to a minister, priest, rabbi or
                 other religious
tc1doc     Q46 How important is it to go to a general medical doctor for help
tc1psy     Q47 How important is it to go to a psychiatrist for help
```

The truncated labels are not helpful.

```
. codebook tc1*, compact

Variable    Obs Unique     Mean Min  Max  Label
------------------------------------------------------------------------------
tc1doc     1074     10 8.714153   1   10  Q46 How important is it to go to ...
tc1fam     1074     10 8.755121   1   10  Q43 How important is it to turn t...
tc1friend  1073     10 7.799627   1   10  Q44 How important is it to turn t...
tc1psy     1050     10 7.567619   1   10  Q47 How important is it to go to ...
tc1relig   1039     10  5.66025   1   10  Q45 How important is it to turn t...
------------------------------------------------------------------------------
```

Better labels

```
. codebook tc2*, compact

Variable    Obs Unique     Mean Min  Max  Label
----------------------------------------------------------------------------
tc2doc     1074     10 8.714153   1   10  Q46 How Impt: Go to a gen med doc...
tc2fam     1074     10 8.755121   1   10  Q43 How Impt: Turn to family for ...
tc2friend  1073     10 7.799627   1   10  Q44 How Impt: Turn to friends for...
tc2mhprof  1045     10  7.58756   1   10  Q48 How Impt: Go to a mental heal...
tc2psy     1050     10 7.567619   1   10  Q47 How Impt: Go to a psych for Help
tc2relig   1039     10  5.66025   1   10  Q45 How Impt: Turn to a religious...
----------------------------------------------------------------------------
```

Even better labels

```
. codebook tc3*, compact

Variable    Obs Unique      Mean  Min  Max  Label
------------------------------------------------------------------------------
tc3doc     1074     10  8.714153    1   10  Q46 Med doctor help important
tc3fam     1074     10  8.755121    1   10  Q43 Family help important
tc3friend  1073     10  7.799627    1   10  Q44 Friends help important
tc3mhprof  1045     10   7.58756    1   10  Q48 MH prof help important
tc3psy     1050     10  7.567619    1   10  Q47 Psychiatric help important
tc3relig   1039     10   5.66025    1   10  Q45 Relig leader help important
------------------------------------------------------------------------------
```

Test labels before you post the data

1. After creating labels, check them in various commands.

2. Revise and repeat till you are satisfied.

3. This can take hours.

4. When you are satisfied, let it sit for a week. Check again.

Short and long labels with language

1. You can have multiple sets of labels saved in a dataset.

2. These can be used to create:

- o Short variable labels for analysis
- o Longer labels for more details
- o Original labels for documentation

3. This can be done with Stata's `language` command explained in the WF book.

Temporary labels

You can *temporarily* change a variable label to improve output in tables created by commands like `esttab` or for labeling axes in graphs.

```
. use wf-acjob
(Workflow data on academic biochemists \ 2008-04-02)

. nmlab job phd

job  Prestige of 1st job on 1 to 5 scale
phd  PhD prestige on 1 to 5 scale

. label var job "Prestige of job"
. label var phd "Prestige of PhD"
. scatter job phd
```

Variable notes

1. *Always* use `notes` to document the provenance of new variables.

notes [*varname*]**:** *text*

2. For example:

```
. use wf-acpub, clear
(Workflow data on scientific productivity \ 2008-04-04)

. gen pubtrunc = pub
. replace pubtrunc = 20 if pubtrunc>20 & !missing(pubtrunc)
(3 real changes made)
. label var pubtrunc "pub truncated at 20"

. note pubtrunc: pubs>20 to 20 | `tag'
. note pubtrunc

pubtrunc:
  1.  pubs>20 to 20 | wflec-var-commands.do Scott Long 2017-06-18
```

3. Notes can be 8,681 characters in Small Stata and 67,784 characters in other versions. For example,

```
. note pubtrunc: Earlier analyses (pubreg04a.do 2006-09-20) ///
> showed that cases with a  large number of articles were ///
> outliers. pubreg04b.do 2006-09-21 examined different ///
> transformations of pub9 and found that truncation at 20 ///
> was most effective at removing the outliers. / `tag'

. note pubtrunc

pubtrunc:
  1.  pubs>20 to 20 | wflec-var-commands.do Scott Long 2017-06-18
  2.  Earlier analyses (pubreg04a.do 2006-09-20) showed that cases
      with a large number of articles were outliers. pubreg04b.do
      2006-09-21 examined different transformations of pub9 and found
      that truncation at 20 was most effective at removing the
      outliers. / wflec-var-commands.do Scott Long 2017-06-18
```

---

4. When you rename a variable:

```
. rename R012213 socialdist
. note socialdist: renamed R012213 / `tag’
```

Listing notes

1. To list all notes in a dataset

    notes

2. To list the notes for selected variables,

    notes *variable-list*

3. Using **`codebook, notes`**:

Results follow...

```
. codebook pubtrunc, notes

-------------------------------------------------------------------------------
pubtrunc                                                   pub truncated at 20
-------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [0,20]                       units:  1
         unique values:  17                       missing .:  0/308

                  mean:   3.10065
             std. dev:     3.4148

           percentiles:        10%       25%       50%       75%       90%
                                 0         1         2       4.5         7

pubtrunc:
  1.  pubs>20 to 20 / wflec-var-commands.do Scott Long 2017-06-18
  2.  Earlier analyses (pubreg04a.do 2006-09-20) showed that cases
      with a large number of articles were outliers. pubreg04b.do
      2006-09-21 examined different transformations of pub9 and found
      that truncation at 20 was most effective at removing the
      outliers. / wflec-var-commands.do Scott Long 2017-06-18
```

Add citations and svy information

```
. notes _dta

_dta:
  1.  For svy estimation in Stata use: svyset secu [pweight=kwgtr],
      strata(stratum) vce(linearized) singleunit(centered)
  2.  Data extracted from (1) h06f2b.dta and (2) rndhrs_n.dta;
      Health and Retirement Study public use dataset. Produced and
      distributed by the Univ of Michigan with funding from the
      National Institute on Aging (grant number NIA U01AG009740).
      Ann Arbor, MI. (2) RAND HRS Data, Version N. Produced by the
      RAND Center for the Study of Aging, with funding from the
      National Institute on Aging and the Social Security
      Administration. Santa Monica, CA (September 2014).
  3.  groups-hrs3.dta \ HRS data for studying group differences \
      groups-hrs-supportV8.do Scott Long & Sarah Mustillo 2017-06-06
      run on 6 Jun 2017 11:36
```

Removing notes

notes drop *variable-name*

See **`help notes`** for more options

Searching notes in Stata 11+

notes search *text*

```
. notes search outliers
pubtrunc:
  2.  Earlier analyses (pubreg04a.do 2006-09-20) showed that cases
      with a large number of articles were outliers. pubreg04b.do
      2006-09-21 examined different transformations of pub9 and
      found that truncation at 20 was most effective at removing the
      outliers. / wflec-var-commands.do Scott Long 2017-06-18
```

Value labels

1. Value labels assign text to numeric values of a variable.
2. Categorical variables should have value labels unless the variable has an inherent metric.
3. Without value labels:

```
. tabulate wc_v1 k5

  Did wife |
    attend |       # of children younger than 6
  college? |         0          1          2          3 |     Total
-----------+--------------------------------------------+----------
         0 |       444         85         12          0 |       541
         1 |       162         33         14          3 |       212
-----------+--------------------------------------------+----------
     Total |       606        118         26          3 |       753
```

Can you assume that 1 stands for yes and 0 stands for no?

4. Is 1 still yes?

```
. tabulate wc_v2 k5

  Did wife |
    attend |       # of children younger than 6
  college? |         0          1          2          3 |     Total
-----------+--------------------------------------------+----------
         1 |       444         85         12          0 |       541
         2 |       162         33         14          3 |       212
-----------+--------------------------------------------+----------
     Total |       606        118         26          3 |       753
```

5. You can use labels that include the value and text:

```
. tabulate wc_v3 k5

  Did wife |
    attend |       # of children younger than 6
  college? |         0          1          2          3 |     Total
-----------+--------------------------------------------+----------
       0No |       444         85         12          0 |       541
      1Yes |       162         33         14          3 |       212
-----------+--------------------------------------------+----------
     Total |       606        118         26          3 |       753
```

Example of why value labels are important

```
mlogit (N=337): Factor change in the odds of occ

Variable: 1.white (sd=0.276)
------------------------------------------------------------------------
                             |        b         z     P>|z|       e^b   e^bStdX
-----------------------------+------------------------------------------
BlueCol      vs Menial       |   1.2365     1.707     0.088     3.444     1.407
Craft        vs Menial       |   0.4723     0.782     0.434     1.604     1.139
Craft        vs BlueCol      |  -0.7642    -1.208     0.227     0.466     0.810
WhiteCol     vs Menial       |   1.5714     1.741     0.082     4.813     1.544
WhiteCol     vs BlueCol      |   0.3349     0.359     0.720     1.398     1.097
WhiteCol     vs Craft        |   1.0990     1.343     0.179     3.001     1.355
Prof         vs Menial       |   1.7743     2.350     0.019     5.896     1.633
Prof         vs BlueCol      |   0.5378     0.673     0.501     1.712     1.160
Prof         vs Craft        |   1.3020     2.011     0.044     3.677     1.433
Prof         vs WhiteCol     |   0.2029     0.233     0.815     1.225     1.058

Variable: 1.white (sd=0.276)
------------------------------------------------------------------------
                             |        b         z     P>|z|       e^b   e^bStdX
-----------------------------+------------------------------------------
2            vs 1            |   1.2365     1.707     0.088     3.444     1.407
3            vs 1            |   0.4723     0.782     0.434     1.604     1.139
3            vs 2            |  -0.7642    -1.208     0.227     0.466     0.810
4            vs 1            |   1.5714     1.741     0.082     4.813     1.544
4            vs 2            |   0.3349     0.359     0.720     1.398     1.097
4            vs 3            |   1.0990     1.343     0.179     3.001     1.355
5            vs 1            |   1.7743     2.350     0.019     5.896     1.633
5            vs 2            |   0.5378     0.673     0.501     1.712     1.160
5            vs 3            |   1.3020     2.011     0.044     3.677     1.433
5            vs 4            |   0.2029     0.233     0.815     1.225     1.058
```

Labeling values is a two-step process

Step 1: Define labels

1. Syntax: **label define** label-name *value1* label1 *value2* label2 ...

2. For example:

`label define Lyesno 1 Yes 0 No`

3. `label define` does *not* indicate which variables use these labels.

4. To improve output with factor variables, I often start labels with a capital or a number (e.g., `1yes` or `Yes`, not `yes`).

Step 2: Assign labels to variables

1. Syntax: **label value** variable label-name

2. I want `wc` to have the label defined as `Lyesno`:

`label value wc Lyesno`

3. I want the same labels for `hc`:

`label value hc Lyesno`

Why two-steps?

1. A two-steps system leads to more consistent labels.

2. Surveys have many similar types of answers. For example:

```
label define Lyesno 0 No 1 Yes
label define Lneg5  1 StDisagree 2 Disagree 3 Neutral 4 Agree 5 StAgree
label define Lpos5  1 StAgree 2 Agree 3 Neutral 4 Disagree 5 StDisagree
```

3. I can assign these labels to 1000's of variables and they will be consistent.

4. It also helps when changing labels. Suppose that I want to change labels:

```
label define Lyesno 0 0_No 1 1_Yes, modify
label define Lneg5  1 1_SD 2 2_D 3 3_N 4 4_A 5 5_SA, modify
label define Lpos5  1 1_SA 2 2_A 3 3_N 4 4_D 5 5_SD, modify
```

5. The revised labels are applied to all variables using these definitions.

- I do not need new `label value` statements.

Planning value labels

1. Plan value labels *before* you create them.

2. Determine which variables can *share labels*.

3. Decide how to label missing values.

Keep labels short

1. Value labels get truncated: keep critical information in the first 8 characters.

2. To see the problem with long labels, consider two labels:

```
. labelbook sd_v1 sd_v2
-----------------------------------------------------------------
value label sd_v1

definition
       1   Definitely Willing
       2   Probably Willing
       3   Probably Unwilling
       4   Definitely Unwilling
```

Continued...

```
value label sd_v2

definition
       1   1Definite
       2   2Probably
       3   3ProbNot
       4   4DefNot
```

3. `sdchild_v1` uses label `sd_v1`:

```
. tabulate female sdchild_v1

     R is |       Q15 Would let X care for children
  female? | Definitel   Probably   Probably  Definitel |     Total
----------+--------------------------------------------+----------
    0Male |        41         99        155        197 |       492
  1Female |        73         98        156        215 |       542
```

4. `sd_v2` labels are shorter with category numbers:

```
. tabulate female sdchild_v2

     R is |       Q15 Would let X care for children
  female? | 1Definite  2Probably   3ProbNot    4DefNot |     Total
----------+--------------------------------------------+----------
    0Male |        41         99        155        197 |       492
  1Female |        73         98        156        215 |       542
----------+--------------------------------------------+----------
    Total |       114        197        311        412 |     1,034
```

Include the category number

1. I often need to know the numeric value assigned to a category. For example:

```
keep if sdchild==?
```

2. You can suppress showing the labels with the `nolabel` option:

```
. tabulate sdchild_v1, nolabel

  Q15 Would |
 let X care |
        for |
   children |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        114       11.03       11.03
          2 |        197       19.05       30.08
          3 |        311       30.08       60.15
          4 |        412       39.85      100.00
------------+-----------------------------------
      Total |      1,034      100.00
```

3. You can explicitly include numeric values in your labels. For example:

```
label define defnot 1 1Definite 2 2Probably 3 3ProbNot 4 4DefNot
```

Using numlabel

1. `numlabel` adds numbers to existing value labels.

Syntax: **numlabel** *vallabel*, **mask(**..**) add**

2. Example:

```
. label define defnot 1 Definite 2 Probably 3 ProbNot 4 DefNot
. numlabel defnot, mask(#_) add
. label val sdchild defnot
. tabulate sdchild

  Q15 Would |
 let X care |
   for kids |      Freq.     Percent        Cum.
------------+-----------------------------------
 1_Definite |        114       11.03       11.03
 2_Probably |        197       19.05       30.08
  3_ProbNot |        311       30.08       60.15
   4_DefNot |        412       39.85      100.00
------------+-----------------------------------
      Total |      1,034      100.00
```

Copy labels before changing them

1. Since `numlabel` destroys the original label, use `label copy` to create an identical value definition with a new name.

2. Revise the copy, leaving the original label intact:

```
. label copy defnot defnotNum
. numlabel defnotNum, mask(#_) add
. label val sdchild defnotNum
. tabulate sdchild
  Q15 Would |
 let X care |
        for |
   children |      Freq.     Percent        Cum.
------------+-----------------------------------
 1_Definite |        114       11.03       11.03
 2_Probably |        197       19.05       30.08
  3_ProbNot |        311       30.08       60.15
   4_DefNot |        412       39.85      100.00
------------+-----------------------------------
      Total |      1,034      100.00
```

Avoid special characters

1. Spaces and characters like ., :, =, /, %, and @ can cause problems.

2. It is safest to use _, -, numbers, and letters. Maybe, spaces.

3. If you use spaces (shown as ◦), use quotes around labels:

```
label define yesno_v2 1 "1◦yes" 0 "0◦no"
```

4. You don't need quotes if there are no spaces:

```
label define yesno_v3 1 1_yes 0 0_no
```

5. Sometimes spaces cause problems with badly behaved commands.

- o Some commands object to valid labels that contain special characters

---

Danger of shared definitions

1. Suppose `female` is 1 for female and 0 for male; `lfp` is 1 for in the labor force and 0 for not.

2. They use the *same label definition*:

```
label define twocat 0 No 1 Yes
label value lfp female twocat  // I assign 2 vars at once
```

3. This is fine:

```
. tabulate female lfp

     R is |   Paid labor force?
  female? |        No        Yes |     Total
-----------+----------------------+----------
        No |       154        198 |       352
       Yes |       171        230 |       401
-----------+----------------------+----------
     Total |       325        428 |       753
```

4. I decide to change the label for `female`:

```
label define twocat 0 Male 1 Female, modify
```

---

5. Now I have a problem:

```
. tabulate female lfp

      R is |   Paid labor force?
   female? |      Male     Female |     Total
-----------+----------------------+----------
      Male |       154        198 |       352
    Female |       171        230 |       401
-----------+----------------------+----------
     Total |       325        428 |       753
```

Workflow to keep track of value labels

1. If a value label is assigned to only *one variable*, give the label definition the name of the variable.

2. If a value label is assigned to *multiple variables*, start the label definition with an **L**.

- o Always be careful about changing definitions that start with **L**!

3. Example, ...

```
. label define   lfp 0 NoInLF 1 InLF
. label value    lfp lfp
. label define   Lcollege 1 College 0 None
. label value wc Lcollege
. label value hc Lcollege

. tabulate wc lfp

     Wife |
 attended | In paid labor force?
 college? |    NoInLF       InLF |     Total
----------+----------------------+----------
     None |       257        284 |       541
  College |        68        144 |       212
----------+----------------------+----------
    Total |       325        428 |       753

. tabulate hc lfp

  Husband |
 attended | In paid labor force?
 college? |    NoInLF       InLF |     Total
----------+----------------------+----------
     None |       207        251 |       458
  College |       118        177 |       295
----------+----------------------+----------
    Total |       325        428 |       753
```

Checking value labels

1. **`describe`** and **`nmlab`**..., **`vl`** list variables with their label definitions.
2. **`codebook, problems`** looks for problems with labels and other things.
3. **`labelbook`** lists labels (edited):

```
. labelbook Ltenpt

value label Ltenpt
-------------------------------------------------------------------------------
      values                                    labels
       range:  [1,10]                     string length:  [6,16]
           N:  5              unique at full length:  yes
        gaps:  yes              unique at length 12:  yes
  missing .*:  3                        null string:  no
                            leading/trailing blanks:  no
  definition
           1   1Not at all Impt
          10   10Vry Impt
          .a   .a_NAP
          .c   .c_Dont know
          .d   .d_No ansr, ref

 variables:  tcfam tc1fam tc2fam tc3fam tc1friend tc2friend tc3friend tc1relig
             tc2relig tc3relig tc1doc tc2doc tc3doc tc1psy tc2psy tc3psy
             tc1mhprof tc2mhprof tc3mhprof
```

Value labels for missing values

1. Stata has the sysmiss . and extended missing .a through .z.
2. Missing values are:
    - Excluded from numerical computations
    - Treated as a category in a tables and similar commands
    - Treated as the largest number in logical comparisons!
3. Multiple missing codes can distinguish reasons for being missing:
    - Respondent did not know the answer.
    - Respondent refused to answer.
    - Respondent did not answer the current question since the lead-in question was refused.
    - Question was not appropriate for the respondent (e.g., asking children how many cars they own).
    - Respondent was not asked the question (e.g., random assignment of who gets asked which questions).

---

4. Use consistent missing value codes for all variables. For example

Letter	Meaning	Example
`.`	Unspecified missing value.	Missing data without the reason being made explicit.
`.d`	Don't know.	Respondent did not know the answer.
`.l`	– Do not use this code –	l (lower-case L) is too close to 1 (one) so avoid it.
`.n`	Not applicable.	Only adults were asked this question.
`.p`	Preliminary question refused.	Question 5 wasn't asked since respondent did not answer the lead-in question.
`.r`	Refused.	Respondent refused to answer question.
`.s`	Skipped due to skip pattern.	Given answer to Question 5, Question 6 was not asked.
`.t`	Technical problem.	Read error on mark sense forms.

Conclusions on variables

1. Never change a variable unless you give it a new name.
2. Every variable needs a variable label.
3. Naming and labeling variables is hard but worth the effort.
    - Try before you decide
    - Anticipate that names and labels might be truncated.
4. Variables you add need a note to document provenance.
5. If you let someone else make the decisions – ask to be involved even if you don't really want to!
6. The decisions you make will *help* or *haunt* you throughout the project.

Part 12: Stata loops

WFDAUS pages 92-105. Guide to Automation.

Loops simply repeat things, are easy to use, and very powerful

1. Data management and analysis involve doing similar operations to many variables.
2. **Loops** repeat the same set of command.
3. Combined with macros and returns, loops are powerful yet easy to use.
    - **`foreach`** creates locals with a sequence of names or numbers.
    - **`forvalues`** creates locals with a sequence of numbers.
4. See *Lab Guide for Automation*.

foreach

foreach *loop-local* **in** *loop-list* **{**

commands using `loop-local'

}

Example 1: displaying names

```
. foreach ivarnm in var1 var2 var3 {
  2.         display "ivarnm is `ivarnm'"
  3. }

ivarnm is var1
ivarnm is var2
ivarnm is var3
```

Example 2: displaying values

```
. foreach ivalue in 1 72 3 {
  2.         display "ivalue is `ivalue'"
  3. }

ivalue is 1
ivalue is 72
ivalue is 3
```

Example 3: series of summary commands

1. Load a dataset and list the variables we are interested in:

```
. use wf-lfp, clear
(Workflow data on labor force participation \ 2008-04-02)

. nmlab lfp k5 k618 age wc hc lwg inc

lfp   In paid labor force?
k5    # kids < 6
k618  # kids 6-18
age   Wife's age in years
wc    Wife attended college?
hc    Husband attended college?
lwg   Log of wife's estimated wages
inc   Family income excluding wife's
```

2. I want to run the same command on each variable:

```
sum lfp
sum k5
sum k618
sum age
sum wc
sum hc
sum lwg
sum inc
```

---

3. I can use a loop:

```
. foreach varnm in lfp k5 k618 age wc hc lwg inc {
  2.         sum `varnm'
  3. }

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         lfp |       753    .5683931    .4956295          0          1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          k5 |       753    .2377158     .523959          0          3

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        k618 |       753    1.353254    1.319874          0          8

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |       753    42.53785    8.072574         30         60

::: and so on :::
```

Example 4: echoing the commands

1. Commands within a loop are not echoed.

2. I simulate the echoing using `display`:

```
. foreach varnm in lfp k5 k618 age wc hc lwg inc {
  2.     display _new ". sum `varnm'"
  3.     sum `varnm'
  4. }

. sum lfp

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         lfp |       753    .5683931    .4956295          0          1

. sum k5

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          k5 |       753    .2377158     .523959          0          3

::: and so on :::
```

Example 5: a series of binary logits

1. I want binary logits on three outcomes:

```
local rhsvars "yr89 male white age ed prst"
local lhsvars "warm_lt2 warm_lt3 warm_lt4"
foreach lhsvar in `lhsvars' {
    di _new ". tab `lhsvar'"
               tab `lhsvar'
    di _new ". logit `lhsvar' `rhsvars'"
               logit `lhsvar' `rhsvars'
}
```

2. The output is:

```
. tab warm_lt2
:::
. logit warm_lt2 yr89 male white age ed prst
:::
. tab warm_lt3
:::
```

Example 6: constructing the variable name

1. If `local cutpt = 2`, then

```
local lhs "warm_lt`cutpt'"
```

equals `warm_lt2`.

2. Using this trick:

```
foreach cutp in 2 3 4 {
    local lhsvar "warm_lt`cutpt'"
    di _new ". tab   `lhsvar'"
               tab   `lhsvar'
    di _new ". logit `lhsvar' `rhsvars'"
               logit `lhsvar' `rhsvars'
}
```

3. I can easily make it run for more variables:

```
foreach cutpt in 2 3 4 5 6 7 8 9 {
```

Example 7: create binary indicators

1. `warm` has values 1 to 4. I want binary variables:

```
gen warm_lt2 = warm<2 if !missing(warm)
gen warm_lt3 = warm<3 if !missing(warm)
gen warm_lt4 = warm<4 if !missing(warm)
```

2. A loop does the same thing:

```
foreach cutpt in 2 3 4 {
    gen warm_lt`cutpt' = warm<`cutpt' if !missing(warm)
}
```

3. **1st time** through the loop `` `cutpt’ `` is 2:

```
gen warm_lt2 = warm<2 if !missing(warm)
```

4. **2nd time** through loop `` `cutpt’ `` is 3:

```
gen warm_lt3 = warm<3 if !missing(warm)
```

Example 8: list variable name and labels

1. Enter `help local` and click `extended_fcn` for details on retrieving information about your data. For example:
2. I can place the *variable label* for `warm` in a local:

```
local varlabel : variable label warm
```

3. To see the contents of `varlabel`:

```
. display "label var for warm: `varlabel'"
label var for warm: Mom can have warm relations w child
```

4. I can place the label in column 12:

```
. local varnm "warm"
. di "`varnm'" _col(12) "`varlabel'"
warm       Mom can have warm relations with child
```

5. Next try this in a loop.

6. Start by putting the variable names in a local:

```
local varnames "warm yr89 male white age ed prst"
```

7. Loop through the names:

```
foreach varnm in `varnames' {
    local varlabel : variable label `varnm'
    di "`varnm'" _col(12) "`varlabel'"
}
```

8. This produces:

```
warm    Working mom can have warm relations w child?
yr89    Survey year: 1=1989 0=1977
male    Gender: 1=male 0=female
white   Race: 1=white 0=not white
age     Age in years
ed      Years of education
prst    Occupational prestige
```

Example 8R1: adding a counter

1. I want to number my list of variables.
2. Create a counter:

```
local counter = 0
```

3. To increase the counter by 1, I could:

```
local counter = `counter’ + 1
```

4. Using this idea in the loop:

```
local counter = 0
foreach varnm in `lhsvars' `rhsvars' {
    local counter = `counter' + 1
    local varlabel : variable label `varnm'
    di "`counter'. `varnm'" _col(12) "`varlabel'"
}
```

5. Results in:

```
1. yr89    Survey year: 1=1989 0=1977
2. male    Gender: 1=male 0=female
3. white   Race: 1=white 0=not white
<snip>
```

6. Incrementing a counter is so useful, there is a shortcut:

```
local counter = 0
foreach varnm in `lhsvars' `rhsvars' {
    local ++counter
    local varlabel : variable label `varnm'
    di "`counter'. `varnm'" _col(12) "`varlabel'"
}
```

Example 9: add variable name to variable labels

1. I want to add the name of the variable to the front of the variable label.

2. For one variable:

```
. local varname  "warm"
. local varlabel : variable label `varname'
. label var `varname' "`varname': `varlabel'"
. tab `varname'

      warm: |
Working mom |
   can have |
       warm |
relations w |
     child? |      Freq.     Percent        Cum.
------------+-----------------------------------
        1SD |        297       12.95       12.95
         2D |        723       31.53       44.48
         3A |        856       37.33       81.81
        4SA |        417       18.19      100.00
------------+-----------------------------------
      Total |      2,293      100.00
```

---

3. In a loop:

```
. local varnames "warm yr89 male white age ed prst"

. foreach varname in `varnames' {
  2.     local varlabel : variable label `varname'
  3.     label var `varname' "`varname': `varlabel'"
  4. }

. nmlab `varnames'

warm   warm: Working mom can have warm relations w child?
yr89   yr89: Survey year: 1=1989 0=1977
male   male: Gender: 1=male 0=female
white  white: Race: 1=white 0=not white
age    age: Age in years
ed     ed: Years of education
prst   prst: Occupational prestige
```

4. Having the variable name in the variable label is useful if you use commands that show the label, but not the variable's name.

- I do this when plotting variables duing data cleaning (discussed later)

Example 10: create interaction variables and labels

1. Loop over variables:

```
foreach varname in yr89 white age ed prst {
    gen       maleX`varname' = male*`varname'
    label var maleX`varname'  "male*`varname'"
    note maleX`varname': `tag'
}
```

so that:

```
. codebook maleX*, compact

Variable     Obs Unique      Mean  Min  Max  Label
-------------------------------------------------------------
maleXyr89   2293      2  .1766245    0    1  male*yr89
maleXwhite  2293      2  .4147405    0    1  male*white
maleXage    2293     71  20.50807    0   89  male*age
maleXed     2293     21  5.735717    0   20  male*ed
maleXprst   2293     59  18.76625    0   82  male*prst
-------------------------------------------------------------
```

---

2. I could add the original label to the label for the interaction:

```
foreach varname in yr89 white age ed prst {
    local varlabel : variable label `varname'
    gen       maleX`varname' = male*`varname'
    label var maleX`varname'  "male*`varlabel'"
}
```

so that:

```
. codebook maleX*, compact

Variable     Obs Unique      Mean  Min  Max  Label
---------------------------------------------------------------------------
maleXyr89   2293      2  .1766245    0    1  male*Survey year: 1=1989 0=1977
maleXwhite  2293      2  .4147405    0    1  male*Race: 1=white 0=not white
maleXage    2293     71  20.50807    0   89  male*Age in years
maleXed     2293     21  5.735717    0   20  male*Years of education
maleXprst   2293     59  18.76625    0   82  male*Occupational prestige
---------------------------------------------------------------------------
```

Example 11: recoding variables

1. I want to dichotomize these variables:

```
local sdvars "sdneighb sdsocial sdchild sdfriend sdwork sdmarry"

foreach varname in `sdvars' {
    gen       B`varname' = `varname'
    label var B`varname'  "`varname': (1,2)=0 (3,4)=1"
    replace   B`varname' = 0 if `varname'==1 | `varname'==2
    replace   B`varname' = 1 if `varname'==3 | `varname'==4
}
```

2. To take the log of multiple variables:

```
foreach varname in incp1 incp2 incp3 incp4 incp5 {
    gen       ln`varname' = ln(`varname'+.5) if `varname'<.
    label var ln`varname' "Log(`varname'+.5)"
}
```

---

forvalues

forvalues *loop-local* **=** *loop-range* **{**

commands referring to `loop-local'

}

range is specified as a numlist (essential for **`margins`** and SPost13):

Syntax	*Meaning*	*Example*	*Generates*
#1(#d)#2	From #1 to #2 in steps of #d.	`1(2)10`	1, 3, 5, 7, 9
#1/#2	From #1 to #2 in steps of 1.	`1/10`	1, 2, 3, ..., 10
#1 #t `to` #2	From #1 to #2 in steps of (#t-#1)	`1 4 to 15`	1, 4, 7, 10, 13

1. To loop from 40 to 80 by 5's:

```
forvalues i = 40(5)80 {
```

2. To loop from 0 to 100 by .1:

```
forvalues i = 1(.1)100 {
```

Example 12: listing 1 to 10

```
. forvalue ivalue = 1/10 {
  2.          display "ivalue = `ivalue'"
  3. }
ivalue = 1
ivalue = 2
ivalue = 3
ivalue = 4
ivalue = 5
ivalue = 6
ivalue = 7
ivalue = 8
ivalue = 9
ivalue = 10
```

Example 12: compute 1st through 4th root of inc

1. I want to take the roots of income:

```
use wf-lfp, clear
gen       inc_root1 = inc^(1/1)
label var inc_root1 "income^(1/1)
gen       inc_root2 = inc^(1/2)
label var inc_root2 "income^(1/2)
:::
```

2. Using a loop is easier...

```
. forvalues iroot = 1(1)5 {
  2.    gen       inc_root`iroot' =   inc^(1/`iroot')
  3.    label var inc_root`iroot' "income^(1/`iroot')
  4.    note      inc_root`iroot': `tag'.
  5.}
(1 missing value generated)
(1 missing value generated)
(1 missing value generated)
(1 missing value generated)

. nmlab inc*

inc        Family income excluding wife's
inc_root1  income^(1/1)
inc_root2  income^(1/2)
:::

. note inc*

inc_root1:
  1.  wflec-auto-loops.do Scott Long 2017-06-18
inc_root2:
  1.  wflec-auto-loops.do Scott Long 2017-06-18
:::
```

Debugging loops

1. **`display`** is helpful for debugging

2. For example, this loop looks fine:

```
foreach varname in "sdneighb sdsocial sdchild sdfriend sdwork sdmarry" {
    gen      B`varname' = `varname'
    replace  B`varname' = 0 if `varname'==1 | `varname'==2
    replace  B`varname' = 1 if `varname'==3 | `varname'==4
}
```

But creates an error:

```
sdsocial already defined
r(110)
```

3. I remove **`sdsocial`** to see if something is peculiar about this variable.

- Now an error is reported for **`sdchild`**.
- The problem is probably not with the variables.

---

4. I add a **`display`** immediately after the **`foreach`**:

```
di "==> varname is: |`varname'|"
```

The **|**'s makes it possible to see blanks in the local.

```
==> varname is: |sdneighb sdsocial sdchild sdfriend sdwork sdmarry|
sdsocial already defined
r(110)
```

which shows the problem.

5. The first time through the loop, I want:

```
local varname sdneighb
```

But everything in quotes is a single string.

6. The solution is deleting quote marks:

```
foreach varname in ///
    sdneighb sdsocial sdchild sdfriend sdwork sdmarry {
```

Using trace to debug loops

1. Turn on trace: **`set trace on`**

2. Lines with **-** show the command before expanding macros.

3. Lines with = show the command after expanding macros.

4. Then,

```
. foreach varname in "sdneighb sdsocial sdchild sdfriend sdwork sdmarry" {
  2.     gen B`varname' = `varname'
  3.     replace  B`varname' = 0 if `varname'==1 | `varname'==2
  4.     replace  B`varname' = 1 if `varname'==3 | `varname'==4
  5. }
- foreach varname in "sdneighb sdsocial sdchild sdfriend sdwork sdmarry" {
- gen B`varname' = `varname'
= gen Bsdneighb sdsocial sdchild sdfriend sdwork sdmarry = sdneighb sdsocial
  > sdchild sdfriend sdwork sdmarry
sdsocial already defined
  replace B`varname' = 0 if `varname'==1 | `varname'==2
  replace B`varname' = 1 if `varname'==3 | `varname'==4
  }
r(110)
```

5. You can easily see what/where the problem is.

To turn trace off: **`set trace off`**.

Part 13: Extended WF for names & labels

WFDAUS pages 176-195.

1. This is an advanced example of how to use automation tools to revise the names and labels in a large dataset.
2. If you are collecting data or creating a dataset by combining multiple source files, these tools might be a good investment of time.
3. If you are extracting a small number of variables from a single data source (e.g., 100 variables from the GSS), these tools might be inefficient.

The data management challenge

1. This example is from a 17 country survey of stigma and mental health with Bernice Pescosolido and Jack Martin.
2. The data came with non-mnemonic names and labels from the questionnaire.
   - Names were inconsistent and sometimes misleading.
   - Labels were often truncated.
3. We spent a *month planning* how to revise names & labels.
4. With 17 datasets to revise, we had to automate. Still, it took over a year.
5. This example shows the flexibility of Stata for this type of work.

Concept: dummy commands

1. This is what I need for 100s of variables:

```
rename atdis   atdisease
rename atgenes atgenet
rename ctxfdoc clawdoc
rename ctxfhos clawhosp
rename ctxfmed clawpmed
```

2. I use automation to create **dummy commands** in a text file:

```
rename atdis   atdis
rename atgenes atgenes
rename ctxfdoc ctxfdoc
rename ctxfhos ctfxhos
rename ctxfmed ctxfmed
```

3. The commands are edited to create the commands I use to rename variables.
   - This is less error prone typing the commands from scratch.

Planning the work

Step 1: Plan the changes

wf5-sgc1a-list.do: List names & labels from `wf-sgc-source.dta`. Export information to spreadsheet to plan changes.

wf5-sgc1b-try.do: Try the names and labels with `tabulate`.

Backup everything twice before proceeding.

Step 2: Clone and rename

wf5-sgc2a-clone.do: Make cloned variables to create `wf-sgc01.dta`.

wf5-sgc2b-rename-dump.do: Create file with dummy `rename` commands.

wf5-sgc2c-rename.do: Rename using edited commands; create `wf-sgc02.dta`.

Step 3: Revise original variable labels

wf5-sgc3a-varlab-dump.do: Use loop with **extended** functions to create a file with dummy `label variable` commands.

---

wf5-sgc3b-varlab-revise.do: Create `original` language for original variable labels; save the revised labels as `default` language in `wf-sgc03.dta`.

Step 4: Revise value labels

wf5-sgc4a-vallab-check.do: List current value labels to review.

wf5-sgc4b-vallab-dump.do: Create a file with dummy `label define` and `label value` commands.

wf5-sgc4c-vallab-revise.do: Add new value labels to `default` language and save `wf5-sgc04.dta`.

Step 5: Entire team looks at new names & labels

wf5-sgc5a-check.do: Check names and labels by trying them with Stata commands.

Steps 1 - 5 are iterated till everyone is satisfied

Step 1: Check the source data

Step 1a: List the current names and labels *(wf5-sgc1a-list.do)*

```
. use wf-sgc-source, clear
(Workflow data for SGC renaming example | 2008-04-03)
. datasignature confirm
  (data unchanged since 03apr2008 13:25)

. notes _dta

_dta:
  1.  wf-sgc-source.dta | wf-sgc-support.do jsl 2008-04-03.
```

1. List current information:

```
0>. unab varlist : _all // local varlist contains all var names
1>  local counter = 1
2>  foreach varname in `varlist' {
3>      local varlabel : variable label `varname'
4>      local vallabel : value label `varname'
5>      di "`counter'." _col(6)  "`varname'" _col(19) ///
 >         "`vallabel'" _col(32) "`varlabel'"
6>      local ++counter
7>  }
```

2. The output is running off the page...

```
1.    id_iu                     Respondent Number
2.    cntry_iu     cntry_iu     IU Country Number
3.    vignum       vignum       Vignette
4.    serious      serious      Q1 How serious would you consider Xs situa...
5.    opfam        Ldummy       Q2_1 What X should do:Talk to family
6.    opfriend     Ldummy       Q2_2 What X should do:Talk to friends
7.    tospi        Ldummy       Q2_7 What X should do:Go to spiritual or t...
8.    tonpm        Ldummy       Q2_8 What X should do:Take non-prescriptio...
```

(output omitted)

3. I could use this list to plan my changes, but prefer a spreadsheet.

4. I need a text file like this for input to Excel. The 1st five lines are:

```
Number;Name;Value label;Variable labels
1;id_iu;;Respondent Number
2;cntry_iu;cntry_iu;IU Country Number
3;vignum;vignum;Vignette
4;serious;serious;Q1 How serious would you consider Xs situation to be?
```

5. To create this file, I start by opening the file:

```
capture file close myfile
file open myfile using wf5-sgc1a-list.txt, write replace
```

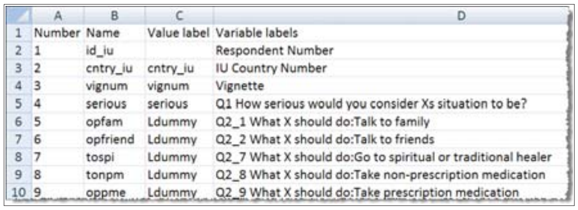
NOTE: **`putexcel`** is a newer way to do this!

---

6. Now I write the file:

```
file write myfile "Number;Name;Value label;Variable labels" _newline
local counter = 1
foreach varname in `varlist' {
     local varlabel : variable label `varname'
     local vallabel : value label `varname'
     file write myfile "`counter';`varname';`vallabel';`varlabel'"
               _newline
     local ++counter
}
file close myfile
```

7. I import this to **`wf5-sgc1a-list.xls`** and plan the changes:



	A	B	C	D
1	Number	Name	Value label	Variable labels
2	1	id_iu		Respondent Number
3	2	cntry_iu	cntry_iu	IU Country Number
4	3	vignum	vignum	Vignette
5	4	serious	serious	Q1 How serious would you consider Xs situation to be?
6	5	opfam	Ldummy	Q2_1 What X should do:Talk to family
7	6	opfriend	Ldummy	Q2_2 What X should do:Talk to friends
8	7	tospi	Ldummy	Q2_7 What X should do:Go to spiritual or traditional healer
9	8	tonpm	Ldummy	Q2_8 What X should do:Take non-prescription medication
10	9	oppme	Ldummy	Q2_9 What X should do:Take prescription medication

---

Step 1b: Try the current names and labels (wf5-sgc1b-try.do)

```
. codebook, compact

Variable   Obs Unique      Mean      Min      Max  Label
--------------------------------------------------------------------------------
id_iu      200    200   1772875  1100107  2601091  Respondent Number
cntry_iu   200      8    17.495       11       26  IU Country Number
vignum     200     12     6.305        1       12  Vignette
serious    196      4  1.709184        1        4  Q1 How serious would you  !
opfam      199      2  1.693467        1        2  Q2_1 What X should do:Talk!
opfriend   198      2  1.833333        1        2  Q2_2 What X should do:Talk!
```

(output omitted)

1. I look at a lot of tabulates:

```
drop id_iu cntry_iu age
unab varlist : _all
foreach varname in `varlist' {
    di  "`varname':" // since tab does not list the name
    tabulate gender `varname', miss
}
```

---

2. Here are some problems

```
vignum:

           |                         Vignette
    Gender | Depressiv  Depressiv  Depressiv  Depressiv  Schizophr |     Total
-----------+-------------------------------------------------------+----------
      Male |        15         11          3          4          7 |        90
    Female |         8         12          9          5         13 |       110
-----------+-------------------------------------------------------+----------
     Total |        23         23         12          9         20 |       200
:::

sdlive:

           |              Q13 To have X as a neighbor?
    Gender | Definitel  Probably   Probably   Definitel         .c |     Total
-----------+-------------------------------------------------------+----------
      Male |        39         32         10          4          4 |        90
    Female |        45         51          9          5          0 |       110
-----------+-------------------------------------------------------+----------
     Total |        84         83         19          9          4 |       200
:::
```

---

3. Less serious problems

```
serious:

           | Q1 How serious would you consider Xs situation to be?
    Gender | Very seri  Moderatel  Not very   Not at al         .c |     Total
-----------+-------------------------------------------------------+----------
      Male |        42         37          8          2          1 |        90
    Female |        49         38         18          2          3 |       110
-----------+-------------------------------------------------------+----------
     Total |        91         75         26          4          4 |       200

trust:

           | Q75 Would you say people can be trusted or need to be
           |               careful dealing w/people?
    Gender | Most peop  Need to b         .a         .c         .d |     Total
-----------+-------------------------------------------------------+----------
      Male |        14         47         29          0          0 |        90
    Female |        13         71         24          1          1 |       110
-----------+-------------------------------------------------------+----------
     Total |        27        118         53          1          1 |       200
```

Step 2: Create clones & rename variables

Step 2a: Create clones (wf5-sgc2a-clone.do)

```
local tag "wf5-sgc2a.do jsl 2008-04-09."
use wf-sgc-source, clear
datasignature confirm
unab varlist : _all
foreach varname in `varlist' {
    clonevar S`varname' = `varname' // S==source
    note S`varname': Source variable for `varname' | `tag'
    note `varname': Clone of source var S`varname' | `tag'
}
note: wf-sgc01.dta | `tag' | create clones of source vars
label data "WF data for SGC renaming example | 2008-04-09"
datasignature set, reset
save wf-sgc01, replace
```

Step 2b: creating `rename` *statements* *(wf5-sgc2b-rename-dump.do)*

```
  1> use wf-sgc01, clear
  2> datasignature confirm
  3> notes _dta

    _dta:
      1.  wf-sgc-source.dta | wf-sgc-support.do jsl 2008-04-03.
      2.  wf-sgc01.dta | wf5-sgc2a.do jsl 2008-04-09 | create clones of
          source variables

  4> drop S*
  5> aorder
  6> unab varlist : _all
  7> file open myfile using wf5-sgc2b-rename-dummy.doi, write replace
  8> foreach varname in `varlist' {
  9>    file write myfile "*rename  `varname'" _col(22) "`varname'" _newline
 10> }
 11> file close myfile
```

1. The `write` command begins with `*` so that the commands are comments:

```
*rename  age          age
*rename  atdisease    atdisease
*rename  atgenes      atgenes
```

(output omitted)

2. Copy `wf5-sgc2b-rename.doi` to `wf5-sgc2b-rename-revised.doi` and edit the dummy commands.

Step 2c: rename variables *(wf5-sgc2c-rename.do)*

```
1>  local tag "wf5-sgc2c.do jsl 2008-04-09."
2>  use wf-sgc01, clear
3>  datasignature confirm
4>  notes _dta
5>  include wf5-sgc2b-rename-revised.doi
```

1. The edited commands look like this:

```
*rename  age          age
*rename  atdisease    atdisease
 rename  atgenes      atgenet
*rename  atraised     atraised
*rename  cause        cause
*rename  cntry_iu     cntry_iu
 rename  ctxfdoc      clawdoc
 rename  ctxfhos      clawhosp
 rename  ctxfmed      clawpmed
```

(remaining commands deleted)

---

2. Here are the changes:

Original		Revised
`atgenes`	⇒	`atgenet`
`ctxfdoc`	⇒	`clawdoc`
`ctxfhos`	⇒	`clawhosp`
`ctxfmed`	⇒	`clawpmed`
`gvdisben`	⇒	`gvdisab`
`gvhous`	⇒	`gvhouse`
`opforg`	⇒	`opforget`
`oppme`	⇒	`oppremed`
`pubfright`	⇒	`pubfrght`
`sdlive`	⇒	`sdneighb`
`stuncom`	⇒	`stuncmft`
`tonpm`	⇒	`opnomed`
`tospi`	⇒	`opspirit`

1: **genet** is the abbrev used in other names.

2: **ctxf** refers to coerced treatment forced which is awkward; **claw** for coerced by law.

3: **hos** was changed to **hosp** for hospital.

4: **med** was changed to **pmed** to indicate psychopharmacological medications.

---

3. Then:

```
. note: wf-sgc02.dta | `tag' | rename source variables
. label data "Workflow data for SGC renaming example | 2008-04-09"
. datasignature set, reset
  200:90(109624):981823927:1981917236       (data signature reset)
. save wf-sgc02, replace
file wf-sgc02.dta saved
```

4. I check new names with `nmlab`, `summarize`, or `codebook, compact`.

Step 3: Revising variable labels

Step 3a: creating variable labels *(wf5-sgc3a-varlab-dump.do)*

1. Load the data:

```
use wf5-sgc02.dta
datasignature confirm
drop S*
aorder
unab varlist : _all
```

---

2. `file write` sends information to the file:

```
file open myfile using wf5-sgc3a-varlab-dummy.doi, write replace
foreach varname in `varlist' {
   local varlabel : variable label `varname'
   file write myfile "label var  `varname' " _col(24) `"""`varlabel'"""' ///
      _newline
}
file close myfile
```

3. The trick is the double quotes around the variable labels. Why does the above code work? Do you really want to know?

4. The resulting file looks like this with lines running off the page:

```
label var  age          "Age"
label var  atdisease    "Q4 Xs situation is caused by: A brain disease o...
label var  atgenet      "Q7 Xs situation is caused by: A genetic or inhe...
label var  atraised     "Q5 Xs situation is caused by: the way X was raised"
label var  cause        "Q62 Is Xs situation caused by depression, asthma...
```

(output omitted)

5. Copy `wf5-sgc3a-varlab-dump.doi` to `wf5-sgc3a-varlab-revised.doi` and edit the dummy commands to be used in Step 3b.

Step 3b: revising variable labels *(wf5-sgc3b-varlab-revise.do)*

1. Load the data:

```
local tag "wf5-sgc3b.do jsl 2008-04-09."
use wf-sgc02, clear
datasignature confirm
notes _dta
```

2. To keep the original labels, a copy is made in the `original` language:

```
label language original, new copy
```

3. I return to `default` language where I change the labels:

```
label language default
note: language original uses the original, unrevised ///
   labels; language default uses revised labels | `tag'
include wf5-sgc3a-varlab-revised.doi
```

4. The commands in the include file look like this:

```
label var age          "Age in years"
label var atdiseas     "Q04 Cause is brain disorder"
label var atgenet      "Q07 Cause is genetic"
label var atraised     "Q05 Cause is way X was raised"
label var cause        "Q62 Xs situation caused by what?"
```

(output omitted)

5. With the changes made, I save the data:

```
note: wf-sgc03.dta | `tag' | revised var labels for source
     & default languages
label data "Workflow data for SGC renaming example | 2008-04-09"
datasignature set, reset
save wf-sgc03, replace
```

6. I check the new labels in the default language:

```
. nmlab tcfam tcfriend vignum

tcfam     Q43 Family help important?
tcfriend  Q44 Friends help important?
vignum    Vignette number
```

7. To see the original labels:

```
. label language original
. nmlab tcfam tcfriend vignum

tcfam     Q43 How Important: Turn to family for help
tcfriend  Q44 How Important: Turn to friends for help
vignum    Vignette
```

Step 4: Revising value labels

This is more complicated. For now, just try to get the basic idea.

Step 4a: List the current labels *(wf5-sgc4a-vallab-check.do)*

1. Load the data

```
use wf-sgc03, clear
datasignature confirm
notes _dta
```

2. `length(#)` option checks if value labels are unique to length #.

```
labelbook, length(10)
```

---

3. Here is output for the `Ldist` label definition:

```
---------------------------------------------------------------------------
value label Ldist
---------------------------------------------------------------------------

      values                                    labels
       range:  [1,4]                    string length:  [16,20]
           N:  4                 unique at full length:  yes
        gaps:  no                  unique at length 10:  no
  missing .*:  0                           null string:  no
                               leading/trailing blanks:  no
                                    numeric -> numeric:  no
  definition
           1   Definitely Willing
           2   Probably Willing
           3   Probably Unwilling
           4   Definitely Unwilling

   in default attached to sdneighb sdsocial sdchild sdfriend sdwork sdmarry
                    Ssdlive Ssdsocial Ssdchild Ssdfriend Ssdwork Ssdmarry

   in original attached to sdneighb sdsocial sdchild sdfriend sdwork sdmarry
                    Ssdlive Ssdsocial Ssdchild Ssdfriend Ssdwork Ssdmarry
```

Step 4b: Dummy `label def` commands *(wf5-sgc4b-vallab-dump.do)**:***

1. Load the data

```
use wf-sgc03, clear
datasignature confirm
notes _dta
drop S*
```

2. I need the names of the label definitions but not the output:

```
quietly labelbook
local valdeflist = r(names)
```

3. This can be done two ways.

Approach 1: Create label define statements with label save

```
label save `valdeflist' using ///
     wf5-sgc4b-vallab-labelsave-dummy.doi, replace
```

1. The `label save` creates a file with:

```
label define Ldist 1 `"Definitely Willing"', modify
label define Ldist 2 `"Probably Willing"', modify
label define Ldist 3 `"Probably Unwilling"', modify
label define Ldist 4 `"Definitely Unwilling"', modify
```

2. Copy wf5-sgc4b-vallab-labelsave-dummy.doi to wf5-sgc4b-vallab-labelsave-revised.doi

   a. Revise the labels.

   b. Change name of `NLdist` to keep the original `Ldist` labels unchanged.

3. The edited definitions look like this and are used in Step 4c:

```
label define NLdist 1 `"1DefWillng"', modify
label define NLdist 2 `"2ProbWill"', modify
label define NLdist 3 `"3ProbUnwil"', modify
label define NLdist 4 `"4DefUnwill"', modify
```

4. After revising all definitions, I use the edited file as an include file in Step 4c.

Approach 2: create customized `label define` statements

1. Load data and write dummy commands:

```
use wf-sgc03, clear
drop S*
aorder
unab varlist : _all
file open myfile using wf5-sgc4b-vallab-labval-dummy.doi, write replace
foreach varname in `varlist' {
    local lblnm : value label `varname'
    if "`lblnm'"!="" {
        file write myfile ///
            "label value  `varname'" _col(27) "N`lblnm'" _newline
    }
}
file close myfile
```

2. The output looks like this:

```
label value  age          Nage
label value  atdisease    NLlikely
label value  atgenet      NLlikely
label value  atraised     NLlikely
label value  cause        Ncause
label value  clawdoc      NLrespons
label value  clawhosp     NLrespons
```

(output omitted)

Step 4c: Revise labels *(wf5-sgc4c-vallab-revise.do)*

1. Copy `wf5-sgc4b-vallab-labdef-dummy.doi` to `wf5-sgc4b-vallab-labdef-revised.doi` & revise the definitions. For example:

```
//                                  1234567890
label define NLdist     1    "1Definite", modify
label define NLdist     2    "2Probably", modify
label define NLdist     3    "3ProbNot", modify
label define NLdist     4    "4DefNot", modify
```

The **guide numbers** verify that the new labels are not too long.

2. Copy `wf5-sgc4b-vallab-labval-dummy.doi` to `wf5-sgc4b-vallab-labval-revised.doi` and revise:

```
label value  age          Nage
label value  atdiseas     NLlikely
label value  atgenet      NLlikely
label value  atraised     NLlikely
label value  cause        Ncause
label value  clawdoc      NLrespons
```

(output omitted)

3. I prefer labels 8 or shorter, but they aren't working. So we use 10 max.

---

4. I am ready to change the labels:

```
. use wf-sgc03, clear
(Workflow data for SGC renaming example | 2008-04-09)
. datasignature confirm
  (data unchanged since 09apr2008 17:59)
```

5. I include the files with the changes to the labels:

```
include wf5-sgc4b-vallab-labdef-revised.doi
include wf5-sgc4b-vallab-labval-revised.doi
```

6. Next, load the temporary dataset:

```
save x-temp, replace
drop S*
quietly labelbook
local valdeflist = r(names)
use x-temp, clear
```

7. Loop through the value definitions and assign labels for missing values:

```
foreach valdef in `valdeflist' {
    label define `valdef' .a  `".a_NAP"'      , modify
    label define `valdef' .b  `".b_Refuse"'   , modify
    label define `valdef' .c  `".c_DK"'       , modify
    label define `valdef' .d  `".d_NA_ref"'   , modify
    label define `valdef' .e  `".e_DK_var"'   , modify
}
```

---

8. Close up:

```
note: wf-sgc04.dta | `tag' | revise val labels with source & default
>  languages label data "Workflow data for SGC renaming example"
datasignature set, reset
save wf-sgc04, replace
```

9. To check the labels,

```
. tabulate marital, missing

    Marital |
     status |      Freq.     Percent        Cum.
------------+-----------------------------------
   1Married |        112       56.00       56.00
   2Widowed |         16        8.00       64.00
  3Divorced |         10        5.00       69.00
  4Separatd |          6        3.00       72.00
   5Cohabit |         21       10.50       82.50
    6Single |         34       17.00       99.50
  .d_NA_ref |          1        0.50      100.00
------------+-----------------------------------
      Total |        200      100.00
```

---

10. By changing languages, I compare these labels to the originals:

```
. label language original
. tabulate marital, missing

               Marital status |      Freq.     Percent        Cum.
------------------------------+-----------------------------------
                      Married |        112       56.00       56.00
                      Widowed |         16        8.00       64.00
                     Divorced |         10        5.00       69.00
        Separated, but married |          6        3.00       72.00
Living as a couple/Cohabiting |         21       10.50       82.50
         Single, never married |         34       17.00       99.50
                           .d |          1        0.50      100.00
------------------------------+-----------------------------------
                        Total |        200      100.00
```

11. Which are better and why?

Step 5: Check new names and labels

1. I let things age
   - After working on a project like this, *any* name or label looks better than the prospect of making more changes.
2. After a break, I systematically review what I have done.
3. If necessary, I revise programs in Steps 2 through 4.

Summary

1. Automation requires time to learn
2. If you use automation regularly, it will save you time and make your work more accurate.
3. "Stealing" my code or examples in the manuals is a great way to use automation without mastering everything.
   - If it works, you don't always need to know why.

Part 14: Debugging do-files

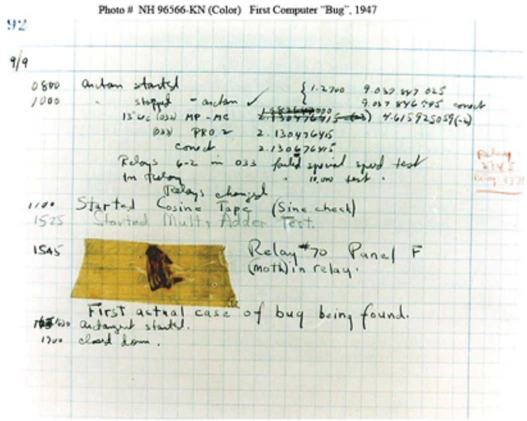


Photo # NH 96566-KN (Color) First Computer "Bug", 1947

92

9/9

0800 Arctan started

1000 " stopped - arctan ✓

Relays 6-2 in 033 failed special speed test

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult+ Adder Test.

1545 Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

1630 Arctangent started.

1700 closed down.

Simple errors and how to fix them

Log file is open

```
. log using example1, replace
log file already open
r(604);
```

Solution: place `capture log close` at the top of your do-file.

Log file already exists

To log the output:

```
log using example2, text
```

If the log file already exists, you get the error:

```
file e:\workflow\work\example2.log already exists
r(602);
```

Solution: use the replace option

```
log using example2, text replace
```

Incorrect command name

1. You try to estimate a logit:

```
loget lfp k5 k618 age wc hc lwg inc
unrecognized command:  loget
r(199)
```

2. If you don't see the problem, click the blue `r(199)`:

```
[P] error . . . . . . . . . . . . . . . . . .  Return code 199
unrecognized command;
Stata failed to recognize command, program, or ado-file name,
probably because of a typographical or abbreviation error.
```

3. Sometimes, unrecognized commands won't be easy to see. For example,

```
. tabl lfp k5
unrecognized command:  tabl
r(199)
```

4. Syntax highlighting in your editor is a good way to catch this type of error.

Incorrect variable name

The name of one of the variables is incorrect.

```
. logit lfp kO5 k618 age wc hc lwg inc
variable kO5 not found
r(111)
```

Solution: `k05` (kay-zero-five) was typed as `kO5` (kay-capital-oh-five).

If you think the name is right and you still get errors

1. Run `describe k*` to list all variables that begin with `k`.

2. Click on the variable name in the Variables Window to paste the name into the Command Window. Copy from there to your do-file.

Incorrect option

You used an incorrect option and get an error message:

```
. logit lfp k5 k618 age wc hc lwg inc, logoff
option logoff not allowed
r(198)
```

Solutions

1. Build the command with a dialog box.

2. Read the manual or `help logit`.

Missing comma before options

This error confuses people learning Stata:

```
. logit lfp wc hc k5 k618 age lwg inc nocon
variable nocon not found
r(111)
```

Solution:

```
logit lfp wc hc k5 k618 age lwg inc, nocon
```

Steps for resolving errors

Step 1: Prevent with robust and legible do-files

1. Robust files eliminate some problems.

2. Legible files prevent mistakes and make problems easier to see.

- I often resolve errors when I "clean up" my do-file to make it legible.

3. Templates prevent typing errors and forgetting standard commands.

Step 2: Update Stata and user written programs

1. THIS IS ESSENTIAL!

2. Official Stata: `update all`

3. User Stata (e.g., SPost package): `adoupate, update`

- For some user programs, this does not work. You have to `search` *command-or-package* and follow the instructions you receive.

Step 3: Clear memory: Increasingly extreme steps

1. In your do-file add

```
clear all
macro drop _all
```

2. Restart Stata

Exit Stata and re-launch Stata.

3. Reboot your computer

Reboot your computer and try the program again.

4. Disable any programs running in the background

I spent a week discovering that this can be the problem.

5. Use another computer

The problem might be due to how Stata was installed on your system.

- If this is convenient, you can start by doing this.

Step 4: Try other data

1. Some errors are caused by **diseases of the data**.

2. Specific names or labels can cause problems.

- o SPost's **`mlogview`** generates errors when certain valid characters are included in the value labels. Stata's **`hausman`** does the same thing.
- o Some SPost commands generates an error when a variable name is 32 characters long.

3. If the error does not occur in other data, focus on characteristics of your data.

Step 5: Assume everything is wrong

1. Do not ignore commands that you are *sure* are right.

- o People who do a lot of programming learn this the hard way.

2. Verify all assumptions about your data.

- o Is yes/no coded 0/1? Or is it 1/2?

Step 6: Run the program in steps

1. Test your program one line or one section at a time.

- o Start with a do-file that loads the data and runs descriptive statistics.
- o If that works, add the next set of commands.
- o If that works, add the next part, and so on.

To exclude parts of the do-file

1. Add * in front of a line you do not want to run.

```
* logit lfp wc hc
```

2. To exclude a series of commands, use /* and */.

- o Everything in between is ignored.

3. Add **`exit`** in the do-file and nothing following it is run.

Step 7: Explain the problem to the rubber duck

1. This really works!

2. Writing an explanation of the problem often helps as well.

Step 8: Start over

Throw out all of the original code

1. It is tempting to keep what you *know* is correct.

- o I spent a week debugging a program. The error was that **`scalar b=-3`** that should have been **`scalar lb=-4`**.

Use a new file with a new name

1. A problem can be caused by hidden characters your editor cannot delete.

2. Sometimes names just won't work on a given day.

- o Impossible, but true.

Try alternative approaches

1. If you think the command is **`tabl`** and not **`tab1`**, you will enter the same incorrect command again.

2. Use a series of **`tabulate`** commands instead.

Step 9: Rarely, it isn't your mistake

1. Try the Stata forum (www.statalist.org/forums/)

- o Search before you ask!

2. Check the Stata FAQ (www.stata.com/support/faqs/)

3. Contacting technical support at StataCorp (www.stata.com/support/)

How to ask for help

1. Try to debug your program. Google the error. Read the manual or the help file.

2. Update Stata and the ado files.

3. *Write* a brief description of the problem. List your steps to solve the problem.

4. Create a *small, self-contained do-file* that generates the error using a *small* dataset. Do not send a huge dataset as an attachment.

5. Send your question, the do-file, log file in text format, and dataset to the person you are asking for help.

6. How to successfully ask a question on Statalist, 14 December 2010William Gould, President (blog.stata.com/2010/12/).

Debugging a syntax error

1. The graph command

```
use wf-acjob, clear
twoway (scatter job phd, msym(smcircle_hollow) msiz(small)), ///
ytitle(Where do you work?) yscale(range(1 5.)) ylab(1(1)5,
angle(ninety)) xtitle(Where did you graduate?) ///
xscale(range(1 5)) xlab(1,5) caption(wfex-debug-graph1.do 2006-
03-17, size(small)) scheme(s2manual) aspectratio(1) by(fem)
```

2. The error is

```
option 5 not allowed
r(198)
```

The message is confusing, so I click on r(198) which is more confusing.

```
 [P]     error . . .. . . . . . . . . . . . . . . . . .   Return code 198
invalid syntax;
__________ invalid;
range invalid;
__________ invalid obs no;
invalid filename;
__________ invalid varname;
__________ invalid name;
```

```
multiple by's not allowed;
__________ found where number expected;
on or off required;
All items in this list indicate invalid syntax.  These errors are
often, but not always, due to typographical errors.  Stata attempts
to provide you with as much information as it can.  Review the
syntax diagram for the designated command.
```

3. Why error messages can be misleading

Stata knows how to parse a correct command,
but does not know how to parse incorrect commands!

4. Reformat the code

```
twoway (scatter job phd, msym(smcircle_hollow) msiz(small)),   ///
    ytitle(Where do you work?) yscale(range(1 5.)) ///
    ylab(1(1)5, angle(ninety)) ///
    xtitle(Where did you graduate?) xscale(range(1 5)) xlab(1,5) ///
    caption(wfex-debug-graph2.do 2006-03-17, size(small)) ///
    scheme(s2manual) aspectratio(1) by(fem)
```

You might see the error now. I don't.

5. Check variables

```
    scatter job phd
```

This works, so I know that the problem is not the data.

6. Run only parts of the code

```
    twoway (scatter job phd, msym(smcircle_hollow) msiz(small)), /* ///
       ytitle(Where do you work?) yscale(range(1 5.)) ///
       ylab(1(1)5, angle(ninety)) ///
       xtitle(Where did you graduate?) xscale(range(1 5)) xlab(1,5) ///
       caption(wfex-debug-graph4.do 2008-04-03, size(small)) ///
       scheme(s2manual) aspectratio(1) by(fem) */
```

This works.

7. Try more code:

```
    twoway (scatter job phd, msym(smcircle_hollow) msiz(small)), ///
       ytitle(Where do you work?) yscale(range(1 5.)) ///
       ylab(1(1)5, angle(ninety)) /* ///
       xtitle(Where did you graduate?) xscale(range(1 5)) xlab(1,5) ///
       caption(wfex-debug-graph5.do 2008-04-03, size(small)) ///
       scheme(s2manual) aspectratio(1) by(fem) */
```

This works.

---

8. Add commands for the x-axis

```
    twoway (scatter job phd, msym(smcircle_hollow) msiz(small)), ///
       ytitle(Where do you work?) yscale(range(1 5.)) ///
       ylab(1(1)5, angle(ninety)) ///
       xtitle(Where did you graduate?) xscale(range(1 5)) xlab(1,5) /* ///
       caption(wfex-debug-graph6.do 2008-04-03, size(small)) ///
       scheme(s2manual) aspectratio(1) by(fem) */
```

The original error is produced, so the problem is probably here:

```
    xtitle(Where did you graduate?) xscale(range(1 5)) xlab(1,5)
```

Now I see that `xlab(1,5)` should be `xlab(1(1)5)`.

Debugging a loop

This looks right but the labels are wrong!

```
use wf-acjob, clear
local varlist "job phd ment art cit fem fel"

foreach y_var in `varlist' { // y axis variable

   local y_lbl : variable label `y_var' // get y label
   label var `y_var' "`y_var': `y_lbl'" // add name to var lab

   foreach x_var in `varlist' { // x axis variable

      if "`y_var'"!="`x_var'" {
         local x_lbl : variable label `x_var'
         label var `x_var' "`x_var': `x_lbl'"
         scatter `y_var' `x_var', msym(circle_hollow) jitter(8) ///
            ylab(, grid) xlab(, grid) aspectratio(1)
         graph export wf6-review-`y_var'-`x_var'.png, replace
      } // if not same variable

   } // x axis loop

} // y axis loop
```

Debugging code in red

```
local varlist "job phd ment"
foreach y_var in `varlist' { // y axis variable

    local y_lbl : variable label `y_var' // get y label
    label var `y_var' "`y_var': `y_lbl'" // add name to var lab
    foreach x_var in `varlist' { // x axis variable
        if "`y_var'"!="`x_var'" {
            local x_lbl : variable label `x_var'
            label var `x_var' "`x_var': `x_lbl'"

local x_lbl : variable label `x_var'
local y_lbl : variable label `y_var'
di "xvar label=> `x_lbl'"
di "yvar label=> `y_lbl'"

/*        scatter `y_var' `x_var', msym(circle_hollow) jitter(8) ///
            ylab(, grid) xlab(, grid) aspectratio(1)
        graph export wf6-review-`y_var'-`x_var'.png, replace
*/
      } // if not same variable
   } // x axis loop
} // y axis loop
```

Output show the problem

```
xvar label=> phd: PhD prestige
yvar label=> job: Prestige of first job
xvar label=> ment: Citations received by mentor
yvar label=> job: Prestige of first job
xvar label=> job: job: Prestige of first job
yvar label=> phd: PhD prestige
xvar label=> ment: ment: Citations received by mentor
yvar label=> phd: phd: PhD prestige
xvar label=> job: job: job: Prestige of first job
yvar label=> ment: ment: ment: Citations received by mentor
xvar label=> phd: phd: phd: PhD prestige
yvar label=> ment: ment: ment: Citations received by mentor
```

Solution

```
spex wf-acjob, clear
local varlist "job phd ment"
// create temporary labels
foreach var in `varlist' {
    local lbl : variable label `var'
    label var `var' "`var': `lbl'"
}
// create graphs
foreach y_var in `varlist' { // y axis variable

    foreach x_var in `varlist' { // x axis variable

        if "`y_var'"!="`x_var'" {
          scatter `y_var' `x_var', msym(circle_hollow) jitter(8) ///
              ylab(, grid) xlab(, grid) aspectratio(1)
          graph export wf6-review-`y_var'-`x_var'.png, replace
        } // if not same variable

   } // x axis loop
} // y axis loop
```

This program is so handy I will make this a template.

Debugging unexpected results

1. I have 9 binary measures to use for a FLIM scale/

2. I start with the percent of 1's for each variable:

```
. use wf-flims, clear
(Workflow data on functional limitations \ 2008-04-02)
. summarize hnd hvy lft rch sit std stp str wlk

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         hnd |      1644     .169708    .3754903          0          1
         hvy |      1644    .4288321    .4950598          0          1
         lft |      1644    .2475669    .4317301          0          1
         rch |      1644    .1703163    .3760248          0          1
         sit |      1644    .2104623     .407761          0          1
-------------+--------------------------------------------------------
         std |      1644    .3607056    .4803514          0          1
         stp |      1644    .3643552    .4813953          0          1
         str |      1644    .2974453    .4572732          0          1
         wlk |      1644    .2706813    .4444469          0          1
```

3. The univariates are fine, but I want to see combinations of variables.

---

4. For example, for two variables:

```
. gen strwlk = 10*str + wlk
. tabulate strwlk, missing

     strwlk |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |      1,091       66.36       66.36
          1 |         64        3.89       70.26
         10 |        108        6.57       76.82
         11 |        381       23.18      100.00
------------+-----------------------------------
      Total |      1,644      100.00
```

5. I extend the idea to nine variables:

```
gen flimall = hnd*100000000 + hvy*10000000 + lft*1000000 ///
    + rch*100000 + sit*10000 + std*1000 + stp*100 + str*10 + wlk
```

Is this better?

```
gen flimall =      hnd*100000000 ///
                 + hvy*10000000 ///
                 + lft*1000000 ///
                 + rch*100000 ///
                 + sit*10000 ///
                 + std*1000 ///
                 + stp*100 ///
                 + str*10 ///
                 + wlk
label var flimall "hnd-hvy-lft-rch-sit-stp-stp-str-wlk"
```

---

6. The value 0 indicates no limitations; 111,111,111 indicates all 9 limitations:

```
. tabulate flimall, missing

hnd-hvy-lft |
-rch-sit-st |
d-stp-str-w |
         lk |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        715       43.49       43.49
          1 |          5        0.30       43.80
         10 |          8        0.49       44.28
         11 |          2        0.12       44.40
(output omitted)
    1100111 |          1        0.06       54.08
    1101100 |          1        0.06       54.14
   1.00e+07 |         86        5.23       59.37
(output omitted)
   1.10e+08 |          7        0.43       88.56
   1.11e+08 |         15        0.91       91.42
(output omitted)
------------+-----------------------------------
      Total |      1,644      100.00
```

---

7. To avoid scientific notation, I create a string variable and get a *weird* result:

```
. gen sflimall=string(flimall, "%16.0f")
. label var sflimall "hnd-hvy-lft-rch-sit-std-stp-str-wlk"
. tabulate sflimall, missing

hnd-hvy-lft |
-rch-sit-st |
d-stp-str-w |
         lk |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        715       43.49       43.49
          1 |          5        0.30       43.80
         10 |          8        0.49       44.28
        100 |         28        1.70       45.99
(output omitted)
   10000000 |         86        5.23       53.83
  100000000 |         15        0.91       54.74
   10000001 |          4        0.24       54.99
  100000096 |          4        0.24       55.23
(output omitted)
    1000001 |          1        0.06       55.29
   10000010 |          5        0.30       55.60
   10000011 |          5        0.30       55.90
(output omitted)
------------+-----------------------------------
```

---

8. I try with four variables.

- To debug, a good strategy is to get a simpler but similar program to work:

```
. gen flimall = std*1000 ///
>                + stp*100 ///
>                + str*10 ///
>                + wlk
. gen  sflimall=string(flimall,"%9.0f")
. label var sflimall "std-stp-str-wlk"
. tabulate  sflimall, missing

std-stp-str |
       -wlk |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        866       52.68       52.68
          1 |         16        0.97       53.65
         10 |         24        1.46       55.11
:::snip:::
       1101 |         27        1.64       76.40
        111 |         20        1.22       77.62
       1110 |         45        2.74       80.35
       1111 |        323       19.65      100.00
------------+-----------------------------------
      Total |      1,644      100.00
```

2. Looks fine so I continue adding variables and things still work with eight variables. It does not matter which eight I choose.

---

3. The problem is that a nine digit number is too large for single precision.

- 100,000,096 ≈ 100,000,100 (off by 1/25000000).
- The solution is to use double precision:

```
. gen double flimall = hnd*100000000 ///
>                        + hvy*10000000 ///
>                        + lft*1000000 ///
>                        + rch*100000 ///
>                        + sit*10000 ///
>                        + std*1000 ///
>                        + stp*100 ///
>                        + str*10 ///
>                      + wlk
```

4. Everything works.

Advanced debugging by tracing

1. If things are still not working, you can trace the error.
    - `set trace on`
    - `set tracedepth #`
2. Stata echoes each line of code it runs.
3. To turn tracing off, `set trace off`.
4. This is best understood by trying it yourself.

Summary

1. More time is often spent debugging programs than writing them.
    - Write programs more slowly and carefully.
2. Learning to debug is a good investment in working efficiently.

IBMs Deep Blue beat Spasky because of a bug!

Part 15: Cleaning data

WFDAUS pages 197-241.

Assume your variables are wrong. Prove they aren't.

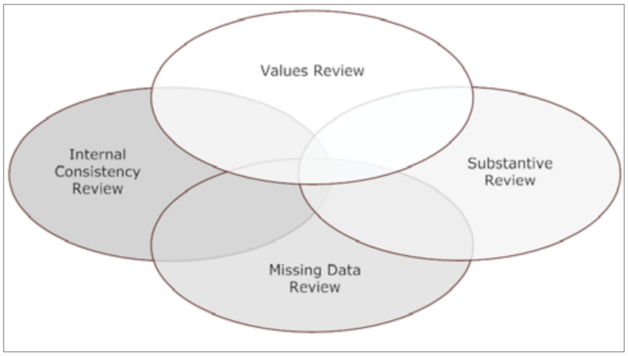
1. Verify that variables are correct *before* you start statistical analysis.
2. This demanding work requires the same care as statistical analyses.
    - Verify a small set of variables. Take a break. *Check the variables again*.
    - Repeat for the next the next set of variables.
3. As you clean variables, plan your analysis.
    - Take notes on ideas, expected findings, and limitations of the data.
    - Cleaning the attic is not fun. Exploring boxes in the attic is fun.
4. Cleaning prevents incorrect, *reproducible* results and retractions.
    - Results can be reproducible but wrong.

Terminology

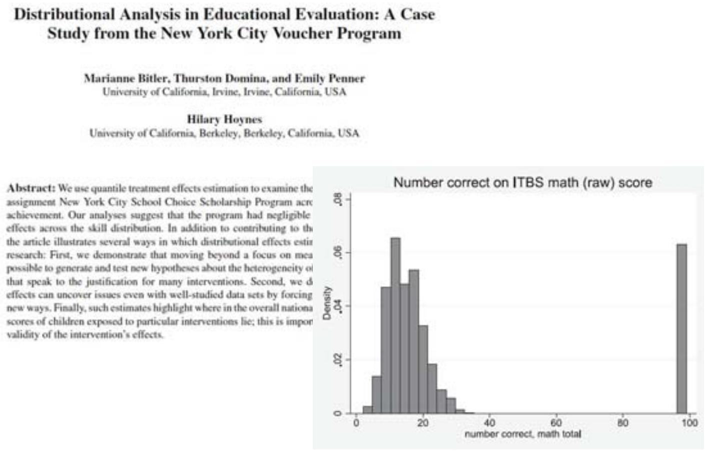
1. Two types of variables
    - *Source variables* are from your ingested, source dataset.
    - *Constructed variables* are created from source variables.
2. Both types need to be verified, but procedures sometimes differ

---

Types of data cleaning



Values Review: earlier work treated 99's as valid



Distributional Analysis in Educational Evaluation: A Case Study from the New York City Voucher Program

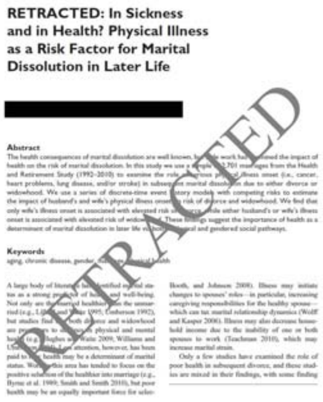
Marianne Bitler, Thurston Domina, and Emily Penner
University of California, Irvine, Irvine, California, USA

Hilary Hoynes
University of California, Berkeley, Berkeley, California, USA

Abstract: We use quantile treatment effects estimation to examine the
assignment New York City School Choice Scholarship Program acr
achievement. Our analyses suggest that the program had negligible
effects across the skill distribution. In addition to contributing to th
the article illustrates several ways in which distributional effects esti
research: First, we demonstrate that moving beyond a focus on mea
possible to generate and test new hypotheses about the heterogeneity o
that speak to the justification for many interventions. Second, we d
effects can uncover issues even with well-studied data sets by forcing
new ways. Finally, such estimates highlight where in the overall nationa
scores of children exposed to particular interventions lie; this is impor
validity of the intervention's effects.

Number correct on ITBS math (raw) score
Density
0
.02
.04
.06
.08
0
20
40
60
80
100
number correct, math total

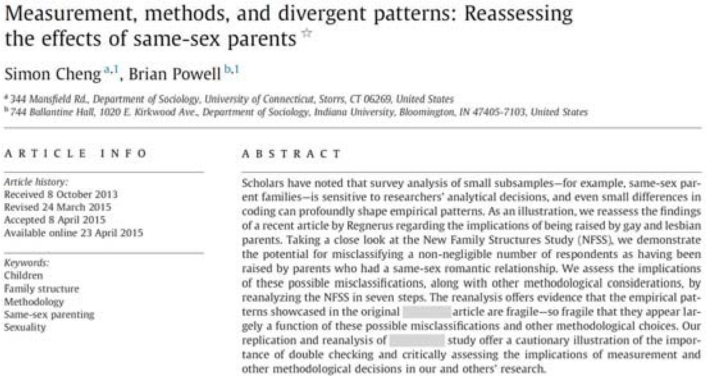
Missing Review: missing recoded to divorced



RETRACTED: In Sickness and in Health? Physical Illness as a Risk Factor for Marital Dissolution in Later Life

[redacted]

Substantive Review: thoughtful coding decisions



Measurement, methods, and divergent patterns: Reassessing the effects of same-sex parents ☆

Simon Cheng [a,1], Brian Powell [b,1]

[a] 344 Mansfield Rd., Department of Sociology, University of Connecticut, Storrs, CT 06269, United States
[b] 744 Ballantine Hall, 1020 E. Kirkwood Ave., Department of Sociology, Indiana University, Bloomington, IN 47405-7103, United States

ARTICLE INFO

Article history:
Received 8 October 2013
Revised 24 March 2015
Accepted 8 April 2015
Available online 23 April 2015

Keywords:
Children
Family structure
Methodology
Same-sex parenting
Sexuality

ABSTRACT

Scholars have noted that survey analysis of small subsamples—for example, same-sex parent families—is sensitive to researchers' analytical decisions, and even small differences in coding can profoundly shape empirical patterns. As an illustration, we reassess the findings of a recent article by Regnerus regarding the implications of being raised by gay and lesbian parents. Taking a close look at the New Family Structures Study (NFSS), we demonstrate the potential for misclassifying a non-negligible number of respondents as having been raised by parents who had a same-sex romantic relationship. We assess the implications of these possible misclassifications, along with other methodological considerations, by reanalyzing the NFSS in seven steps. The reanalysis offers evidence that the empirical patterns showcased in the original [redacted] article are fragile—so fragile that they appear largely a function of these possible misclassifications and other methodological choices. Our replication and reanalysis of [redacted] study offer a cautionary illustration of the importance of double checking and critically assessing the implications of measurement and other methodological decisions in our and others' research.

Values review

Tools

1. **`codebook, compact`** shows # of non-missing observations, # of unique values, mean, minimum, maximum, and variable label.
2. **`summarize`** includes descriptive statistics, but not variable labels.
3. If descriptive statistics look fine, check values with **`tab1`**, **`dotplot`**, or **`stem`**.
4. Use **`tabulate`** or **`scatter`** to examine pairs of variables.

Values review of data about the scientific career

```
. use wf-acjob, clear
(Workflow data on academic biochemists | 2008-04-02)

. codebook, compact

Variable     Obs Unique      Mean  Min       Max  Label
-------------------------------------------------------------------------------
job          408     80  2.233431    1       4.8  Prestige of first job
fem          408      2  .3897059    0         1  Gender: 1=female 0=male
phd          408     89  3.200564    1       4.8  PhD prestige
ment         408    123  45.47058    0  531.9999  Citations received by mentor
fel          408      2  .6176471    0         1  Fellow: 1=yes 0=no
art          408     14  2.276961    0        18  # of articles published
cit          408     87  21.71569    0       203  # of citations received
-------------------------------------------------------------------------------
```

1. 18 seems large, but I know collaboration is common in biochemistry.
2. Ranges for **`job`** and **`phd`** match; **`job`** has a lower mean as expected.
3. Binary variables have the right range.

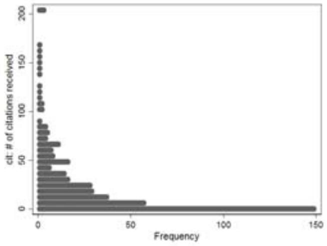
4. The distribution of articles is reasonable.

```
. tab1 art, missing

-> tabulation of art

       # of |
   articles |
  published |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         85       20.83       20.83
          1 |        102       25.00       45.83
          2 |         72       17.65       63.48
          3 |         49       12.01       75.49
          4 |         45       11.03       86.52
          5 |         25        6.13       92.65
          6 |         13        3.19       95.83
          7 |          9        2.21       98.04
          8 |          2        0.49       98.53
          9 |          1        0.25       98.77
         10 |          2        0.49       99.26
         12 |          1        0.25       99.51
         15 |          1        0.25       99.75
         18 |          1        0.25      100.00
------------+-----------------------------------
      Total |        408      100.00.
```

5. **`job`**, **`phd`**, **`ment`**, and **`cit`** have many values, so use a histogram:

```
. stem cit

Stem-and-leaf plot for cit (# of citations received)

   0* | 0000000000000000000000000000000000000000000000000000000000000 ...
(212)
   1* | 00000011122222223333333334444445555555677888888888899999
   2* | 0000011111222233333344444455666667777789
   3* | 0000122222333444556667789
   4* | 122335566777788888889
   5* | 1144556677789
   6* | 0034555556667
   7* | 01145778
   8* | 012368
   9* |
  10* | 0057
  11* | 3
  12* | 03
  13* |
  14* | 069
  15* | 4
  16* | 39
  17* |
  18* |
  19* |
  20* | 113
```

6. Or use **`dotplot`** to create graph files:



To create plots for all non-binary variables:

```
foreach varname in art cit phd job ment {
    local varlbl : variable label `varname'
    label var `varname' "`varname': `varlbl'" // add name label
    dotplot `varname'
    graph export `pgm'-wf-acjob-`varname'.png, replace
}
```

Lots of graphs

1. You can examine 100's of graphs per minute and detect problems.
2. Put them in the same folder and use a graph viewer.
   - I use IrfanView in Windows (www.irfanview.com)
3. Demonstration

Values review of data on family values

1. 2002 GSS: To what extent do you agree or disagree that a working mother can establish just as warm and secure a relationship with her children as a mother who does not work?
2. To compare responses in 2002 to those in 1977 and 1989, I start with:

```
. use wf-gsswarm, clear
(Workflow data from 2002 GSS on women and work | 2008-04-02)
. tabulate v4, miss

  Workg mom: warm relation |
                  child ok |      Freq.     Percent        Cum.
---------------------------+-----------------------------------
            Strongly agree |        468       39.97       39.97
                     Agree |        383       32.71       72.67
Neither agree nor disagree |        124       10.59       83.26
         Strongly disagree |        184       15.71       98.98
               Cant choose |         11        0.94       99.91
               Na, refused |          1        0.09      100.00
---------------------------+-----------------------------------
                     Total |      1,171      100.00
```

3. Where is disagree?

---

4. I check values associated with the value labels:

```
. tabulate v4, nolab

 Workg mom: |
       warm |
   relation |
   child ok |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        468       39.97       39.97
          2 |        383       32.71       72.67
          3 |        124       10.59       83.26
          5 |        184       15.71       98.98
          8 |         11        0.94       99.91
          9 |          1        0.09      100.00
------------+-----------------------------------
      Total |      1,171      100.00
```

5. Source data was SPSS portable file, so I used SPSS to check if problem was caused by data conversion. Nope.
6. In the GSS 2002 codebook I find category *disagree* is omitted.
   - GSS confirmed there was an error in data collection.

Substantive review

When reviewing 1000s of lines of output or 100s of graphs, do not forget:

- *What does the data tell you about the world?*
- *Does the distribution make subtantive sense?*

What does time to degree measure?

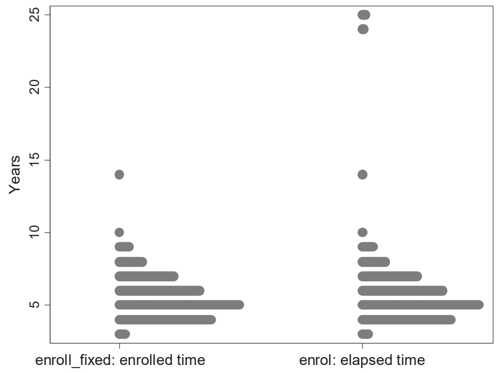
1. When I was writing a report for the National Academy of Sciences, I learned how critical a substantive review is and why “hired help” can’t do it.
2. Years *enrolled* in graduate school is a standard predictor of productivity.
   - Past studies found a negative effect.
3. The analyst found time to degree had a positive effect:

```
. use wf-acpub, replace
(Workflow data on scientific productivity | 2008-04-04)

. nbreg pub enrol phd female, nolog irr

Negative binomial regression                      Number of obs   =        278
                                                  LR chi2(3)      =      23.54
Dispersion     = mean                             Prob > chi2     =     0.0000
Log likelihood = -606.28466                       Pseudo R2       =     0.0190

------------------------------------------------------------------------------
         pub |        IRR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       enrol |   1.056071   .0156467     3.68   0.000     1.025845    1.087188
         phd |   1.103679   .0654233     1.66   0.096     .9826206    1.239652
      female |      .7533   .0968775    -2.20   0.028     .5854637    .9692504
-------------+----------------------------------------------------------------
    /lnalpha |  -.4592692   .1471825                     -.7477416   -.1707969
-------------+----------------------------------------------------------------
       alpha |   .6317451   .0929818                      .4734346    .8429928
------------------------------------------------------------------------------
Likelihood-ratio test of alpha=0:  chibar2(01) =  172.26 Prob>=chibar2 = 0.000
```

4. Many fruitless exchanges followed.

---

5. Eventually, I determined the analyst used *elapsed* time, not *enrolled* time.



6. Using the enrolled time, results were as expected:

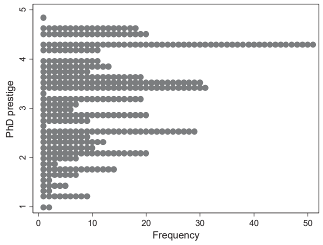
```
. nbreg pub enrol_fixed phd female, nolog irr

Negative binomial regression                      Number of obs   =        278
                                                  LR chi2(3)      =      26.97
Dispersion     = mean                             Prob > chi2     =     0.0000
Log likelihood = -604.5674                        Pseudo R2       =     0.0218

------------------------------------------------------------------------------
         pub |        IRR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
 enrol_fixed |     .82013    .037127    -4.38   0.000     .7504973    .8962233
         phd |   1.112075   .0666021     1.77   0.076     .9889072    1.250582
      female |   .7450266   .0964034    -2.27   0.023     .5781357     .960094
-------------+----------------------------------------------------------------
    /lnalpha |  -.4493616   .1428516                     -.7293456   -.1693777
-------------+----------------------------------------------------------------
       alpha |   .6380353   .0911444                      .4822244      .84419
------------------------------------------------------------------------------
Likelihood-ratio test of alpha=0:  chibar2(01) =  211.39 Prob>=chibar2 = 0.000
```

Examine high frequency values

1. The descriptive statistics look fine (*wf6-review-phdspike.do*):

```
. use wf-acjob, clear
(Workflow data on academic biochemists | 2008-04-02)
. summarize phd

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         phd |       408    3.200564    .9537509          1        4.8
```

2. The histogram shows spikes



---

3. When I tabulated **phd** for cases between 4 and 4.5:

```
. tabulate phd if phd>4 & phd<4.5

        PhD |
   prestige |      Freq.     Percent        Cum.
------------+-----------------------------------
       4.14 |          3        4.35        4.35
       4.16 |          8       11.59       15.94
       4.25 |          1        1.45       17.39
       4.29 |         37       53.62       71.01
       4.32 |          9       13.04       84.06
       4.34 |          4        5.80       89.86
       4.48 |          7       10.14      100.00
------------+-----------------------------------
      Total |         69      100.00
```

4. 10% of the degrees came from Wisconsin-Madison which had three large departments that awarded biochemistry degrees. Later analyses looked at these graduates more carefully.

Links among variables

1. Compare prestige of doctoral program and the prestige of first academic job:

```
. use wf-acjob, clear
(Workflow data on academic biochemists | 2008-04-02)
. codebook phd job, compact

Variable   Obs Unique      Mean  Min  Max  Label
-------------------------------------------------------------------
phd        408     89  3.200564    1  4.8  PhD prestige
job        408     80  2.233431    1  4.8  Prestige of first job
-------------------------------------------------------------------
```

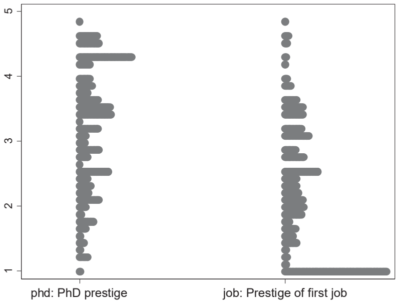
2. These statistics look reasonable.

- Ranges are the same.
- Mean is smaller for jobs, as expected.

---

3. To compare the distributions, I use **`dotplot`**: I change the variable labels to include the variable names and increase the font size for legibility

```
. label var phd "phd: PhD prestige"
. label var job "job: Prestige of first job"
. dotplot phd job, xlab(,labsize(medium))
```

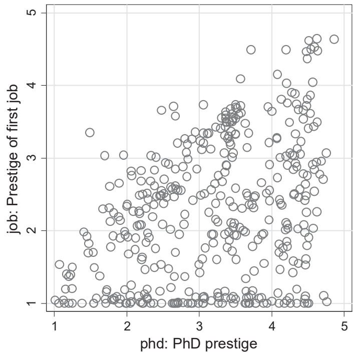


4. What is the spike at 1? Is something wrong?

---

5. Next I examine how they are related:

```
. scatter job phd, msym(circle_hollow) jitter(8) ///
>     ylab(, grid) xlab(, grid) aspectratio(1)
```



Mobility in science is largely downward. -- *Caplow and McGee, 1957*

Template to create pairs of graphs

Checking if variable are equal: don't plot x against x?

Variables are different

```
. local y_var age
. local x_var income

. if ("`y_var'"!="`x_var'") display "y_var does not equal x_var"

    y_var does not equal x_var

. if ("`y_var'"=="`x_var'") display "y_var equals x_var"
```

Variables are the same

```
. local y_var income
. local x_var income

. if ("`y_var'"!="`x_var'") display "y_var does not equal x_var"

. if ("`y_var'"=="`x_var'") display "y_var equals x_var"

    y_var equals x_var
```

Loop to create scatterplots

```
use wf-acjob, clear
local varlist "job phd ment art cit fem fel"

foreach varname in `varlist' {
    local varlabel : variable label `varname'
    label var `varname' "`varname': `varlabel'"
}

foreach y_var in `varlist' { // y axis variable

    foreach x_var in `varlist' { // x axis variable

        if "`y_var'"!="`x_var'" {

          scatter `y_var' `x_var', msym(circle_hollow) jitter(8) ///
              ylab(, grid) xlab(, grid) aspectratio(1)
          graph export XXX-`y_var'-`x_var'.png, replace

        } // if not same variable

   } // x axis loop

} // y axis loop
```

Changes in survey questions

1. To study effects of religion on sexual activity Chavez (2007) used Add Health.
2. She found an dramatic decline in membership in Disciples of Christ:
    a. Wave 1: 10.7% are Disciples of Christ
    b. Wave 3: 00.5% are Disciples of Christ
3. Reviewing the survey instrument she found:
    a. Wave 1 referred to "Christian Church (Disciples of Christ)"
    b. Wave 3 referred to "Disciples of Christ"
4. She inferred that in wave 1 the respondents who were Christian, not necessarily Disciples of Christ, selected this category.

Missing data review

1. Missing data is commonly represented by `.`, `.a`, `.b`, ..., `.z`.
2. In comparisons:
    a. In comparisons `.z > .y >...> .b > .a > . >` other#s
    b. `999<.z` is false
    c. `.z>999` is true
3. Different missing values can be used to indicate different reasons for being missing:
    - **Refused**: The respondent refused to answer the question.
    - **Don't know**: The respondent did not know the answer.
    - **Not applicable**: The question was not appropriate for the respondent.
    - **Not asked**: In a split ballot design, respondent was not asked the question.
4. I recommend learning about Stata's `misstable` and SPost's `misschk`

Problems with missing values

1. Problem 1: In source data, missing values reflect how data are collected, *not* necessarily if the value is unknown.
    - You might be able to infer a value from another variable
    - Something not coded as missing might be
2. Problem 2: Missing values can be converted incorrectly when imported.
    - WFDAUS has an example of errors caused by converting formats.

Danger: Logical comparisons and missing values

1. Missing values are valid with comparisons.
2. In logical comparisons, missing values are larger than any number.
3. Consider the distribution of articles, with 19 missing values.

```
. tabulate art, missing

 # articles |
  published |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        102       41.98       41.98
          1 |         72       29.63       71.60
          2 |         25       10.29       81.89
          3 |         13        5.35       87.24
          4 |          9        3.70       90.95
          9 |          1        0.41       91.36
         12 |          1        0.41       91.77
         15 |          1        0.41       92.18
          . |         19        7.82      100.00
------------+-----------------------------------
      Total |        243      100.00
```

4. I want to truncate values above 5:

```
. gen art_tr5 = art
(19 missing values generated)
. label var art_tr5 "# of articles truncated at 5"
. note art_tr5: `tag’
. replace art_tr5 = 5 if art>5
(22 real changes made)
```

5. To see what happened:

```
. tabulate art art_tr5, missing

  # of |
 articles |                 trunc at 5 # of articles published
published |         0          1          2          3          4          5 |     Total
----------+------------------------------------------------------------------+----------
        0 |       102          0          0          0          0          0 |       102
        1 |         0         72          0          0          0          0 |        72
        2 |         0          0         25          0          0          0 |        25
        3 |         0          0          0         13          0          0 |        13
        4 |         0          0          0          0          9          0 |         9
        9 |         0          0          0          0          0          1 |         1
       12 |         0          0          0          0          0          1 |         1
       15 |         0          0          0          0          0          1 |         1
        . |         0          0          0          0          0         19 |        19
----------+------------------------------------------------------------------+----------
    Total |       102         72         25         13          9         22 |       243
```

This type of problem caused a recent retraction!

6. The solution

```
. gen art_tr5V2 = art
(19 missing values generated)
. replace art_tr5V2 = 5 if art>5 & art<.
(3 real changes made)
. label var art_tr5V2 "# of articles truncated at 5 "
. note art_tr5V2: `tag’

. tabulate art_tr5V2, missing

 trunc at 5 |
       # of |
   articles |
  published |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        102       41.98       41.98
          1 |         72       29.63       71.60
          2 |         25       10.29       81.89
          3 |         13        5.35       87.24
          4 |          9        3.70       90.95
          5 |          3        1.23       92.18
          . |         19        7.82      100.00
```

Indicators if cases are missing

To check patterns of missing, a binary indicator of missing is useful.

```
gen art_ismiss = missing(art)
label var art_ismiss "art is missing?"
note art_ismiss: `tag’
label def Lismiss 0 0_valid 1 1_missing
label val art_ismiss Lismiss

. tabulate art art_ismiss, missing

     # of |
 articles |    art is missing?
published |   0_valid  1_missing |     Total
-----------+----------------------+----------
         0 |       102          0 |       102
         1 |        72          0 |        72
         2 |        25          0 |        25
<snip>
        12 |         1          0 |         1
        15 |         1          0 |         1
         . |         0         19 |        19
-----------+----------------------+----------
     Total |       224         19 |       243
```

Examine the distribution of missing values

1. You can see the distribution of missing values with **`tabulate`** and **`tab1`** using option **`missing`**, but you also see valid values.

- o This makes it hard to see missing values for a variable like income.

2. To tabulate only missing values:

```
tab1 phd if missing(phd), miss
```

3. For all variables:

```
unab allvars : _all
foreach varname in `allvars’ {
    tab1 `varname' if missing(`varname'), missing
}

-> tabulation of phd if missing(phd)

        PhD |
   prestige |      Freq.     Percent        Cum.
------------+-----------------------------------
    a_NonUS |          7       36.84       36.84
 b_Unranked |         12       63.16      100.00
------------+-----------------------------------
      Total |         19      100.00

<snip>
```

Verifying and expanding missing data codes

1. You might be able to determine why something is missing by examining how the variable is related to other variables.

2. In a survey at The Kinsey Institute the source variables had two missing codes:

`.a` question was refused

`.b` question was not applicable

No further details were given on what these categories meant.

3. I used prior questions to determine the substantive reason why observations were missing, and sometimes determined they were not missing.

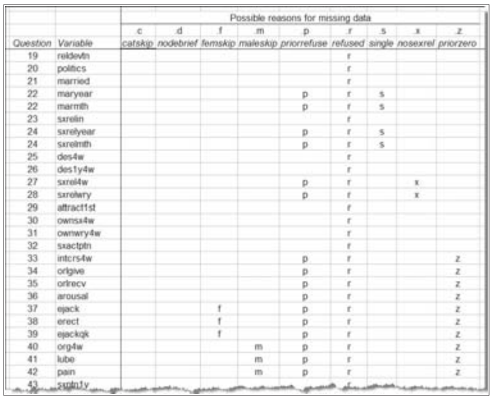
4. Reviewing the survey we determined 9 reasons for missing data:

Missing Code	Value Label	Meaning
.c	catskip	Categorical response was not needed.
.d	nodebrief	Refused to answer debriefing questions.
.f	femskip	Women were not asked this question.
.m	maleskip	Men were not asked this question.
.p	priorref	Question not asked since prior question refused.
.r	refused	Current question was refused.
.s	single	Not asked since respondent was single.
.x	nosxrel	Not asked since respondent was not in sexual relationship.
.z	prior_0	Not asked since answer to prior question was zero.

```
label def missdat .c "c_catskip"  .d "d_nodebrief" .f "f_femskip"  ///
                  .m "m_maleskip" .p "p_priorref"  .r "r_refused"  ///
                  .s "s_single"   .x "x_nosxrel"   .z "z_prior_0"
```

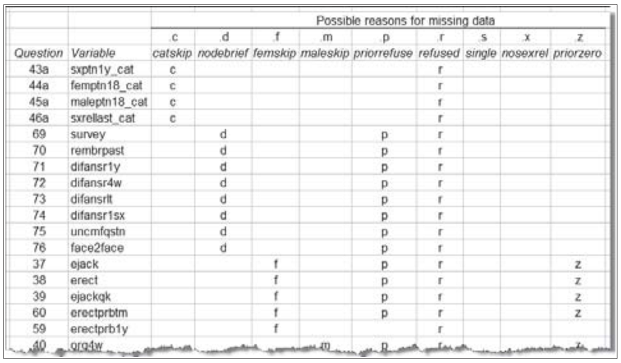
5. To recode the original **`.a`**'s and **`.b`**'s into the refined missing value types required days of careful review.

6. Review the survey to determine which types of missing values are possible for each question:



		Possible reasons for missing data								
		.c	.d	.f	.m	.p	.r	.s	.x	.z
Question	Variable	catskip	nodebrief	femskip	maleskip	priorrefuse	refused	single	nosexrel	priorzero
19	reldevtn						r			
20	politics						r			
21	married						r			
22	maryear					p	r	s		
22	marmth					p	r	s		
23	sxrelin						r			
24	sxrelyear					p	r	s		
24	sxrelmth					p	r	s		
25	des4w						r			
26	des1y4w						r			
27	sxrel4w					p	r		x	
28	sxrelwry					p	r		x	
29	attract1st						r			
30	ownsx4w						r			
31	ownwry4w						r			
32	sxactptn						r			
33	intcrs4w					p	r			z
34	orlgive					p	r			z
35	orlrecv					p	r			z
36	arousal					p	r			z
37	ejack			f		p	r			z
38	erect			f		p	r			z
39	ejackqk			f		p	r			z
40	org4w				m	p	r			z
41	lube				m	p	r			z
42	pain				m	p	r			z
43	sxptn1y						r			

7. Sort variables to find which variables require similar data processing using automation.



		Possible reasons for missing data								
		.c	.d	.f	.m	.p	.r	.s	.x	.z
Question	Variable	catskip	nodebrief	femskip	maleskip	priorrefuse	refused	single	nosexrel	priorzero
43a	sxptn1y_cat	c					r			
44a	femptn18_cat	c					r			
45a	maleptn18_cat	c					r			
46a	sxrellast_cat	c					r			
69	survey		d			p	r			
70	rembrpast		d			p	r			
71	difansr1y		d			p	r			
72	difansr4w		d			p	r			
73	difansrlt		d			p	r			
74	difansr1sx		d			p	r			
75	uncmfqstn		d			p	r			
76	face2face		d			p	r			
37	ejack			f		p	r			z
38	erect			f		p	r			z
39	ejackqk			f		p	r			z
60	erectprbtm			f		p	r			z
59	erectprb1y			f			r			
40	org4w				m	p	r			z

8. The process took days. See the WF book for details.

Missing values that are not missing

1. The time a person was married was determined by two questions:

How many *years* married?

How many *months* married?

2. Total months of marriage combined the questions

```
. gen martotal = (maryearV2*12) + marmthV2
(109 missing values generated)
. label var martotal "Total months married"
. note martotal: `tag’
```

3. Missing codes were created for those single or who refused the prior questions:

```
. replace martotal = .s if married==2
(86 real changes made, 86 to missing)
. replace martotal = .p if married==.a
(1 real change made, 1 to missing)
```

4. `martotal` was coded `.r` if either `maryearV2` or `marmthV2` were refused:

```
. replace martotal = .r if marmthV2==.r | maryearV2==.r
(22 real changes made, 22 to missing)
```

5. Looking at missing values:

```
. label val martotal missdat
. tab1 martotal if missing(martotal), missing

-> tabulation of martotal if missing(martotal)

      Total |
     months |
    married |      Freq.     Percent        Cum.
------------+-----------------------------------
 p_priorref |          1        0.92        0.92
  r_refused |         22       20.18       21.10
   s_single |         86       78.90      100.00
------------+-----------------------------------
      Total |        109      100.00
```

More respondents refused these questions than more sensitive questions.

6. Listing those with `.r`, *nobody refused both questions*

```
. list martotal maryearV2 marmthV2 if martotal==.r, clean
        martotal   maryearV2    marmthV2
 12.   r_refused          53   r_refused
 34.   r_refused          31   r_refused
 37.   r_refused   r_refused          11
 38.   r_refused          33   r_refused
 40.   r_refused   r_refused           8
 45.   r_refused          54   r_refused
```

(output omitted)

```
173.   r_refused          46   r_refused
190.   r_refused   r_refused           4
198.   r_refused          26   r_refused
206.   r_refused          24   r_refused
210.   r_refused          13   r_refused
214.   r_refused          28   r_refused
```

7. The most reasonable explanation is that some people rounded to the nearest year and didn't report months. Those married less than a year skipped the year question. Using these assumptions, I created a new version of `martotal`:

```
. gen martotalV2 = .
(218 missing values generated)
. label var martotalV2 "Total months married"
. note martotalV2: `tag’
. replace martotalV2 = (12*maryearV2) + marmthV2 ///
>     if !missing(maryearV2) & !missing(marmthV2)
(109 real changes made)
. replace martotalV2 = 12*maryearV2 if !missing(maryearV2) & marmthV2==.r
(19 real changes made)
```

These 19 cases had been coded `.r` originally.

8. Similarly, if year is refused, I only use the month:

```
. replace martotalV2 = marmthV2 if maryearV2==.r & !missing(marmthV2)
(3 real changes made)
```

Three cases that were originally missing now have valid values.

9. I added missing codes for those who are single and or who refused the prior question:

```
. replace martotalV2 = .s if married==2
(86 real changes made, 86 to missing)
. replace martotalV2 = .p if married==.a
(1 real change made, 1 to missing)
. label val martotalV2 missdat
```

10. The revised distribution of missing values is reasonable:

```
. tab1 martotalV2 if missing(martotalV2), miss

-> tabulation of martotalV2 if missing(martotalV2)

      Total |
     months |
    married |      Freq.     Percent        Cum.
------------+-----------------------------------
 p_priorref |          1        1.15        1.15
   s_single |         86       98.85      100.00
------------+-----------------------------------
      Total |         87      100.00
```

11. I checked my work by sorting on **`martotalV2`** and listing relevant variables:

```
. sort martotalV2
. list martotalV2 maryearV2 marmthV2, clean

       martotalV2    maryearV2     marmthV2
  1.            2            0            2
  2.            4    r_refused            4
  3.            8    r_refused            8
::snip::
  4.           11    r_refused           11
  5.           16            1            4
  9.           25            2            1
::snip::
127.          648           54    r_refused
128.          651           54            3
129.          715           59            7
::snip::131.          735           61            3
132.   p_priorref   p_priorref   p_priorref
133.     s_single     s_single     s_single
134.     s_single     s_single     s_single
::snip::
```

12. Further substantive analysis confirmed the decision was reasonable.

Internal consistency review

When there are logical links among variables, verify the data are consistent.

- If formal education begins at 5, years of education must be 5 less than age.
- If someone is not in the labor force, they should not report a wage.
- Responses to questions about attitudes on working mothers to be consistent in the sense that people who are positive on one question will be positive on related questions.
- I do not expect a scientist to be hired in a department that is substantially more prestigious than her doctoral origin.
- If you do not publish, you should not be cited.

Consistency in data on the scientific career

1. If a person does not publish, he cannot be cited.

```
. use wf-acjob, clear
(Workflow data on academic biochemists | 2008-04-02)
. tab cit if art==0, missing

      # of |
  citations |
   received |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         85      100.00      100.00
------------+-----------------------------------
      Total |         85      100.00
```

2. I expect a scientist's job prestige will be lower than her doctoral prestige although this is not logically required.

```
. compare job phd
                                        ---------- difference ----------
                            count       minimum      average     maximum
------------------------------------------------------------------------
job<phd                       288         -3.64    -1.462847        -.02
job=phd                        48
job>phd                        72           .01     .3709723        2.08
                       ----------
jointly defined               408         -3.64    -.9671323        2.08
                       ----------
total                         408
```

3. List cases where the job is substantially more prestigious than the Ph.D.:

```
. gen job_phd = job - phd
. label var job_phd "job-phd: >0 if better job"
. note job_phd: `tag’
. sort job phd, stable
. list job_phd art ment fem cit fel job phd if job_phd>.65, clean
```

```
        job_phd   art       ment        fem   cit          fel    job    phd
400.   .6500001     1         36   1_Female    18  0_NotFellow   3.52   2.87
401.        .74     2          6     0_Male    19     1_Fellow   2.49   1.75
402.   .8200002     0         20   1_Female     0  0_NotFellow   3.68   2.86
403.   .8899999     0          9     0_Male     0  0_NotFellow   3.04   2.15
404.   .8900001     0         20     0_Male     0     1_Fellow   4.48   3.59
405.       1.07     4        233     0_Male    22     1_Fellow   2.88   1.81
406.       1.13     0          0   1_Female     0  0_NotFellow   3.52   2.39
407.       1.17     4   69.99999     0_Male    41     1_Fellow   3.68   2.51
408.       2.08     1   3.999999   1_Female    32     1_Fellow   3.36   1.28
```

4. The only large difference is someone with 1 article with 32 citations. Perhaps important work from the dissertation lead to a prestigious job.

5. I look at those whose jobs are much less prestigious than their Ph.D.:

```
. list job_phd art ment fem cit fel job phd if job_phd<-2, clean

        job_phd   art       ment        fem   cit          fel    job    phd
  1.      -3.64     0          2   1_Female     0     1_Fellow      1   4.64
  2.      -3.64     3         16     0_Male    24     1_Fellow      1   4.64
  3.      -3.62     0   87.99999     0_Male     0     1_Fellow      1   4.62
  4.      -3.54     1         23   1_Female     5  0_NotFellow      1   4.54
  5.      -3.54     5   47.00001     0_Male    27  0_NotFellow      1   4.54
(output omitted)
```

Informal imputation

You might find inconsistencies in your data. Ideally, the solution is obvious. Sometimes you must decide what to do using imperfect information.

Duration of marriage

1. In the example of duration of marriage, I made a judgment call that some missing values were zeros. *Was that justified?*

Organizational resources

1. A survey of organizations asked:

Question 1: "Does your organization have any revenues?"

Question 2: "How much revenue does your organization received?"

2. What if question 1 is not answered, but question 2 has a positive response?

3. A survey ask 10 yes/no questions about availability of resources (e.g., computers). Suppose 5 resources are checked yes, but 5 others are blank. Is this a non-response or a short cut taken by the person filling out the survey?

Functional limitations

1. Two question:
    a. Do you have trouble when lifting heavy objects?
    b. Do you have trouble when lifting light objects?
2. What if someone has no trouble with heavy objects but not light objects?
    - Did the respondent make a mistake or answer exactly the question that was asked?
    - Since I don't lift heavy objects, I have no trouble lifting them.

Informal imputation

1. When cleaning data you make decisions to resolve issues where the data are ambiguous or inconsistent.
2. Simply coding these cases as missing can introduce distortion.
3. When you make *informal imputations*, document what you did and why.

Summary

The problem

1. If the data is not clean, everything that follows is affected.
2. If you are lucky, you will figure this out before publishing your work.
3. If you don't, hopefully someone will discover the error so the results can be corrected. Hard for you, but good for science.
4. Worst case, wrong results are accepted.

The solution

1. Make data cleaning an essential and informative part of your workflow.
2. Use it to clean the data and to plan your analyses.
3. Get to know your data. Don't delegate this important task.

Part 16: Adding variables

WFDAUS pages 241-259.

1. Variables in the source dataset are *source variables*.
2. Analysis usually requires new variables created from source variables
3. How does a dual workflow fit into this process?
    - During analysis you will decide to add new variables.
    - These variables should be added in the data workflow.
    - Part 18 shows a convenient way to do this by creating a "dual" dataset.

Adding new variables

Principles for adding variables

1: New variables get unique, new names--always.

2: Immediately verify new variables.

3: Add metadata for new variables (labels and notes).

4: Keep the source variables for later verification.

A new variable template

Step 1: *< create the variable >*

Step 2: **`label var`** *varname* **"***variable label***"**

Step 3: **`note`** *varname***:** *optional description* | **`` `tag' ``**

Step 4: **`label def … // if appropriate`**

Step 5: **`label val … // if appropriate`**

Principle 1: New variables get new names

New variables are always given new names. No exceptions. Never use the same of a variable that earlier was dropped from the dataset.

Danger of re-using a name

1. I estimate a model that includes log of wages **`lwg`**:

```
use wf-lfp, clear
logit lfp k5 k618 age wc hc lwg inc
estimates store model1
```

2. To explore the effect of wages without taking the log:

```
replace lwg = exp(lwg)
* where is the variable label and note?
logit lfp k5 k618 age wc hc lwg
estimates store model2
```

---

3. Comparing models:

```
. estimates table _all, stats(N bic) eform b(%9.3f) t(%6.2f)

-------------------------------------
    Variable |   model1      model2
-------------+-----------------------
          k5 |      0.232       0.233
             |      -7.43       -7.28
:::
          wc |      2.242       1.469
             |       3.51        1.65
          hc |      1.118       0.855
             |       0.54       -0.79
         lwg |      1.831       1.497
             |       4.01        6.48
         inc |      0.966
             |      -4.20
-------------+-----------------------
           N |    753.000     753.000
         bic |    958.258     924.418
-------------------------------------
                            legend: b/t
```

4. I mistakenly conclude that the change in the size and significance of the effect of **`lwg`** is due to excluding **`inc`** from the model.
5. The dual workflow prevents this type of mistake.

Principle 2: Verify that new variables are correct

1. Don’t be so involved in the analyses that you forget to verify the variables.

2. Use methods from chapter on cleaning and others explained below.

Example

1. I compute the log of income:

```
. gen inclog = log(inc)
 (1 missing value generated)
. label var inclog "log(inc)"
. note inclog: `tag’
```

2. Is the missing data came missing data in the source variable?

```
. list inc inclog if inc<0, clean

             inc    inclog
373.   -.0290001         .
```

3. I need to decide how to handle negative income.

Principle 3: Add metadata to new variables

1. When you create a variable, add:

- A variable label
- A note with provenance
- Value labels for categorical variables

2. For example:

```
gen inc_log5 = ln(inc+.5) if !missing(inc)
label var inc_log5 "Log(inc+.5)"
note inc_log5: `tag’
```

3. Since `local tag` is in all do-files it is easy to keep track of how every variable was created.

Principle 4: Keep the source variables

Why keep source variables?

1. Verify the new variables, and re-verify them if you have doubts later.

2. Fix errors discovered later.

3. Create additional variables.

Exceptions

1. If the dataset is *very* large, you might need to delete variables.

2. If source variables have *dangerous errors*, you might want to delete them.

If you delete a variable

```
note: inc was deleted due to a coding error; use incV2. ///
  If you need inc see binlfp2.dta | `tag'
```

If you keep an incorrect variable

```
note inc: DO NOT USE inc due to incorrect coding of high ///
    incomes; use incV2 | `tag'
label var inc: "DO NOT USE; see incV2 | `tag'"
```

Renaming source variables

1. I do not want to risk using source variables:

- They don’t have the labels
- They haven’t been cleaned.

2. To make it easier to keep track of which variables are source variables, I rename them the start with **S**:

```
rename gender Sgender
note Sgender: renamed source variable gender | `tag’
```

3. As an exercise, use a loop to do this with all of your source variables.

- Make this one of your templates

Selected commands for creating variables

generate

generate *newvar* = *exp* [*if*] [*in*]

For example,

```
generate agesqrt=sqrt(age)
gen agesqrt=sqrt(age) if age>5
```

clonevar

clonevar *newvar* = *sourcevar* [*if*] [*in*]

The clone retains values, labels and other metadata.

---

generate versus clonevar

```
. use wf-lfp, clear
(Workflow data on labor force participation | 2008-04-02)

. gen lfp_gen = lfp
. note lfp_gen: `tag’

. clonevar lfp_clone = lfp
. note lfp_clone: `tag’

. summarize lfp*

  Variable |       Obs        Mean    Std. Dev.       Min        Max
-----------+--------------------------------------------------------
       lfp |       753    .5683931    .4956295          0          1
   lfp_gen |       753    .5683931    .4956295          0          1
 lfp_clone |       753    .5683931    .4956295          0          1

. describe lfp*

variable     storage  display    value
name         type     format     label   variable label
--------------------------------------------------------------------
lfp          byte     %9.0g      lfp     In paid labor force? 1=yes 0=no
lfp_gen      float    %9.0g
lfp_clone    byte     %9.0g      lfp     In paid labor force? 1=yes 0=no
```

replace

Changes the values of an existing variable:

replace *newvar* = *exp* [*if*] [*in*]

Example

1. Copy original variable:

```
. gen educcat = edyears
(159 missing values generated)
. label var educcat "Categorized years of education"
. note educcat: `tag’
```

2. Change some values:

```
. replace educcat = 1 if edyears>=0 & edyears<=8   // no HS
(278 real changes made)
. replace educcat = 2 if edyears>=9 & edyears<=11  // some HS
(501 real changes made)
. replace educcat = 3 if edyears==12               // HS
(205 real changes made)
. replace educcat = 4 if edyears>=13 & edyears<=15 // some college
(517 real changes made)
. replace educcat = 5 if edyears>=16 & edyears<=24 // college plus
(135 real changes made)
```

---

3. Label categories:

```
. label def educcat 1 NoHS 2 SomeHS 3 HS 4 SomeCol 5 ColPlus ///
>      .b b_Refused .c c_DontKnow .d d_AtSchool .e e_AtCollege ///
>      .f f_NoFrmlSchl
. label val educcat educcat
```

4. The new variable:

```
. tab1 educcat, missing

-> tabulation of educcat

     educcat |      Freq.     Percent        Cum.
-------------+-----------------------------------
        NoHS |        281       15.63       15.63
      SomeHS |        501       27.86       43.49
          HS |        205       11.40       54.89
     SomeCol |        517       28.75       83.65
     ColPlus |        135        7.51       91.16
   b_Refused |          3        0.17       91.32
  c_DontKnow |         61        3.39       94.72
  d_AtSchool |          7        0.39       95.11
 e_AtCollege |         73        4.06       99.17
f_NoFrmlSchl |         15        0.83      100.00
-------------+-----------------------------------
       Total |      1,798      100.00
```

New variables with missing values

1. It is easy to:

- Create missing values that should *not* be missing.
- Turn missing values into seemingly valid values.

2. I want a variable that is 1 if married; 0 if not:

```
. use wf-russia01, clear
(Workflow data to illustrate creating variables | 2008-04-02)

. tab1 marstat, miss

-> tabulation of marstat

    Marital |
     status |      Freq.     Percent        Cum.
------------+-----------------------------------
  1_married |        931       51.78       51.78
  2_widowed |        321       17.85       69.63
 3_divorced |        215       11.96       81.59
4_separated |         33        1.84       83.43
   5_single |        279       15.52       98.94
         .b |         19        1.06      100.00
------------+-----------------------------------
      Total |      1,798      100.00
```

---

3. I use an equality comparison:

```
. gen ismar_wrong = (marstat==1)
. label var ismar_wrong "Is married created incorrectly."
. note ismar_wrong: `tag’
. label def Lyesno 0 0_no 1 1_yes
. label val ismar_wrong Lyesno
```

4. Missing values are *not 1* so they are coded as 0.

```
. tabulate marstat ismar_wrong, miss

            |  Is married created
    Marital |     incorrectly.
     status |      0_no      1_yes |     Total
------------+----------------------+----------
  1_married |         0        931 |       931
  2_widowed |       321          0 |       321
 3_divorced |       215          0 |       215
4_separated |        33          0 |        33
   5_single |       279          0 |       279
         .b |        19          0 |        19
------------+----------------------+----------
      Total |       867        931 |     1,798
```

---

5. To exclude missing values from the comparison:

```
. gen ismar_right = (marstat==1) if !missing(marstat)
(19 missing values generated)
. label var ismar_right "Is married?"
. note ismar_right: `tag’
. label val ismar_right Lyesno
. tabulate marstat ismar_right, miss

    Marital |             Is married?
     status |      0_no      1_yes          . |     Total
------------+---------------------------------+----------
  1_married |         0        931          0 |       931
  2_widowed |       321          0          0 |       321
 3_divorced |       215          0          0 |       215
4_separated |        33          0          0 |        33
   5_single |       279          0          0 |       279
         .b |         0          0         19 |        19
------------+---------------------------------+----------
      Total |       848        931         19 |     1,798
```

6. This error caused a recent retraction.

Other commands for creating variables

Read the manuals to review the dozens of commands you can use.

recode: an easy way to recode categories

Convert: 1 to 1; then 2 through 5 to 0

```
. recode marstat 1=1   /// married stays married
                 2/5=0 /// 2 to 5 becomes not married
                , gen(ismar2_right)
(848 differences between marstat and ismar2_right)
. label var ismar2_right "Is married?"
. note ismar2_right: `tag’
. tabulate marstat ismar2_right, miss
    Marital |             Is married?
     status |         0          1         .b |     Total
  1_married |         0        931          0 |       931
  2_widowed |       321          0          0 |       321
 3_divorced |       215          0          0 |       215
4_separated |        33          0          0 |        33
   5_single |       279          0          0 |       279
         .b |         0          0         19 |        19
------------+---------------------------------+----------
      Total |       848        931         19 |     1,798
```

Recode years of education to categories

```
. recode edyears  0/ 8=1  /// 0 to 8 is 1
                 9/11=2  /// 9 to 11 is 2
                   12=3  /// 12 is 3
                13/15=4  /// 13 to 15 is 4
                16/24=5  /// 16 to 24 is 5
              , gen(edcat2)
(1636 differences between edyears and educcat2)
. note edcat2: `tag’
```

Recode 1 to 0, *all* other values to 1:

```
. recode edyears 1=0 *=1, gen(edtest1)
(1798 differences between edyears and edtest1)
. note edtest1: `tag’
```

To retain missing values:

```
. recode edyears 1=0 *=1 if !missing(edyears), gen(edtest2)
(1639 differences between edyears and edtest2)
. note edtest2: `tag’
```

I prefer replace to recode

1. `recode` is powerful and fast.
2. `replace` makes me more thoughtful.

---

egen

1. **egen** stands for *extended generate*.
2. There are dozens of powerful **egen** commands.
3. Be sure you know what the command is doing before using it!

Standardized variables

```
. use wf-lfp, clear
(Workflow data on labor force participation | 2008-04-02)
. summarize age

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |       753    42.53785    8.072574         30         60

. gen agestd = (age - r(mean)) / r(sd)
. label var agestd "Age standardized using generate"
. note agestd: using r(mean) | `tag’
. egen agestdV2 = std(age)
. label var agestdV2 "Age standardized using egen"
. note agestdV2: using egen | `tag’
. summarize agestd agestdV2

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      agestd |       753   -7.05e-09           1  -1.553141   2.163145
    agestdV2 |       753   -7.05e-09           1  -1.553141   2.163145
```

Count 0's

1. **anycount** counts how many variables have specified values:

```
. gen count0 = anycount(lfp k5 k618 age wc hc lwg inc), ///
              values(0)
. label var count0 "# of 0's in lfp k5 k618 age wc hc lwg inc"
. note count0: `tag’
. tabulate count0, miss

# of 0's in |
lfp k5 k618 |
  age wc hc |
    lwg inc |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         11        1.46        1.46
          1 |         94       12.48       13.94
          2 |        157       20.85       34.79
          3 |        251       33.33       68.13
          4 |        169       22.44       90.57
          5 |         71        9.43      100.00
------------+-----------------------------------
      Total |        753      100.00
```

2. How do I know this is right?

---

3. I try counting another way:

```
gen count0v2 = 0
label var count0v2 "v2:lfp k5 k618 age wc hc lwg inc == 0"
note count0v2: `tag’

foreach var in lfp k5 k618 age wc hc lwg inc {
    replace count0v2 = count0v2 + 1 if `var'==0
}
```

4. **`compare count0 count0v2`** confirms the variables are identical.

---

tabulate, gen()

To create indicator variables for the categories of a variable:

tabulate *varname* [*if*] [*in*] **, generate(***stub***)** [**missing**]

- **gen(*stub*)** requests an indicator variables for each value of *varname*.
- Indicators start with *stub* and ends with the value of *varname*.
- **missing** option creates indicators for missing value categories.

Indicators of marital status:

```
. use wf-russia01, clear
(Workflow data to illustrate creating variables | 2008-04-02)
. tabulate marstat, gen(ms_is)

    Marital |
     status |      Freq.     Percent        Cum.
------------+-----------------------------------
  1_married |        931       52.33       52.33
  2_widowed |        321       18.04       70.38
 3_divorced |        215       12.09       82.46
4_separated |         33        1.85       84.32
   5_single |        279       15.68      100.00
------------+-----------------------------------
      Total |      1,779      100.00
```

```
. codebook ms_is*, compact

Variable    Obs Unique      Mean  Min  Max  Label
-------------------------------------------------------------------
ms_is1     1779      2  .5233277    0    1  marstat==1_married
ms_is2     1779      2  .1804384    0    1  marstat==2_widowed
ms_is3     1779      2  .1208544    0    1  marstat==3_divorced
ms_is4     1779      2  .0185497    0    1  marstat==4_separated
ms_is5     1779      2  .1568297    0    1  marstat==5_single

. tabulate marstat ms_is1, miss

    Marital |        marstat==1_married
     status |         0          1          . |     Total
------------+---------------------------------+----------
  1_married |         0        931          0 |       931
  2_widowed |       321          0          0 |       321
 3_divorced |       215          0          0 |       215
4_separated |        33          0          0 |        33
   5_single |       279          0          0 |       279
         .b |         0          0         19 |        19
------------+---------------------------------+----------
      Total |       848        931         19 |     1,798
```

Next add variable labels, notes, and value labels.

Verifying variables

1. You will make a mistake when you create variables.

2. Here are some ways to find those mistakes.

Method 1: Check the code

Writing the program

1. My comments, notes, and labels are brief since I focus on the logic.

2. The code might not be legible.

Verifying a program before posting

1. Review the commands.

2. Make the code legible.

3. Add comments explaining the logic of what I am doing.

4. Improve metadata.

Method 2: List variables

1. Listing values helps when something is wrong, but you aren't sure why.

2. You can also use **`browse`**, but *do not* used **`edit`**!

Error in recoding income

1. **`finc_mid`** is the midpoint of the range of income:

```
recode finc_mid /// with illegible code
1=1.5 2=4 3=6 4=8 5=9.5 6=10.5 7=11.5 8=12.5 ///
9=13.5 10=14.5 11=16 12=18.5 13=21 14=23.5 15=23.5 16=32.5 17=37.5 1///
8=42.5 19=47.5 20=55 21=67.5 22=82.5 23=97.5 24=131.25
```

2. If I list the first 100 cases, they might all have **`fincome=1`**.

3. To randomly select N cases

```
. set seed 1951
. tempvar selvar // variable disappears when do file ends.
. gen `selvar' = int( (runiform()*_N)+ 1 ) // it works!
. label var `selvar' "Random numbers from 1 to _N"
. sort income // to list records in order
```

This code should be one of your templates.

---

4. I list 20 randomly selected cases:

```
. list fincome finc_mid if `selvar' <= 20, clean

            fincome   finc_mid
  92.       2_3-5K          4
 211.       3_5-7K          6
 242.       4_7-9K          8
 333.      5_9-10K        9.5
 479.     8_12-13K       12.5
 727.    12_17-20K       18.5
 819.    13_20-22K         21
 876.    14_22-25K       23.5
 930.    14_22-25K       23.5
1105.    15_25-30K       23.5
1118.    15_25-30K       23.5
1174.    16_30-35K       32.5
1236.    16_30-35K       32.5
1338.    17_35-40K       37.5
```

(output omitted)

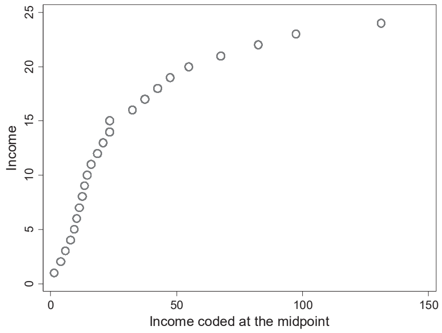
Method: use by command

```
. sort finc_mid
. by finc_mid: sum fincome
-> finc_mid = 1.5
Variable |        Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
 fincome |         67           1           0          1          1
<snip>
-> finc_mid = 23.5
Variable |        Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
 fincome |        300       14.58    .4943832         14         15

-> finc_mid = 32.5
Variable |        Obs        Mean    Std. Dev.       Min        Max
---------+--------------------------------------------------------
 fincome |        178          16           0         16         16
```

Method 3: Plotting

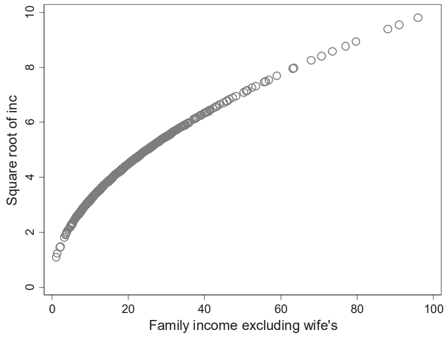
Creating finc_mid

```
scatter fincome finc_mid, msymbol(circle_hollow)
```



Taking the square root of inc

```
gen incsqrt = sqrt(inc) if !missing(inc)
label var incsqrt "Square root of inc"
note incsqrt: `tag’
scatter incsqrt inc, msym(circle_hollow)
```



Method 4: Tabulate with missing option

1. When variables have many values, the table is large and hard to check.

```
. gen incsqrt = sqrt(inc)
(1764 missing values generated)
. label var incsqrt "Sqrt family income excluding wife's"
. note incsqrt: `tag’
```

2. To check only missing:

```
. tabulate inc incsqrt if missing(inc) | missing(incsqrt), miss

           |   Sqrt
           |   family
    Family |   income
    income | excluding
 excluding |   wife's
    wife's |         . |     Total
-----------+-----------+----------
 -.0290001 |         1 |         1
         . |     1,742 |     1,742
        .a |         5 |         5
        .b |        16 |        16
-----------+-----------+----------
     Total |     1,764 |     1,764
```

Method 5: Construct variables multiple ways

1. I need to verify that I used **`recode`** correctly.

```
. recode edyears 0/8=1 9/11=2 12=3 13/15=4 16/24=5, gen(educcat)
(1636 differences between edyears and educcat)
. note edyears: `tag’
```

2. I create what should be the same variable using **`replace`**:

```
. gen educcatV2 = edyears
(848 missing values generated)
. label var educcatV2 "categorize educ using replace."
. note edyearsV2: `tag’
. replace educcatV2 = 1 if edyears>=0  & edyears<=8   // no HS
(278 real changes made)
. replace educcatV2 = 2 if edyears>=9  & edyears<=11  // some HS
(501 real changes made)
. replace educcatV2 = 3 if edyears==12                // HS
(205 real changes made)
. replace educcatV2 = 4 if edyears>=13 & edyears<=15  // some college
(517 real changes made)
. replace educcatV2 = 5 if edyears>=16 & edyears<=24  // college+
(135 real changes made)
```

---

3. Compare the variables:

```
. compare educcat educcatV2

                                        ---------- difference ----------
                            count       minimum      average     maximum
------------------------------------------------------------------------
educcat=educcatV2            1639
                       ----------
jointly defined              1639             0            0           0
jointly missing               848
                       ----------
total                        2487
```

4. If they are not the same, I tabulate the two variables.

5. I would delete **`educcatV2`** before saving the dataset.

Example of preparing data

Using the Russian ISSP for 2002, I create the analysis variables in steps:

wflec-add-data-russia1-controls.do

- Add control variables

wflec-add-data-russia2-binary.do

- Binary indicators

wflec-add-data-russia3-noneutral.do

- Exclude neutral

wflec-add-data-russia4-analysisvars.do

- Add variables for analysis

---

wflec-add-data-russia1-controls.do

1. Loading the data and checking signature:

```
. use wf-russia01, replace
(Workflow data to illustrate creating variables | 2008-04-02)
. datasignature confirm
  (data unchanged since 02apr2008 13:29)
```

2. Create local for tagging new variables and notes

3. **`gender`** equals 1 for men and 2 for women. I want a pair of binary variables for gender:

```
gen female = gender - 1
label var female "Female?"
label def female 0 0_male 1 1_female
label val female female
note female: based on gender | `tag'
```

---

4. Checking:

```
. tab gender female, miss

       Gender: |
       1=male, |        Female?
      2=female |    0_male   1_female |     Total
---------------+----------------------+----------
       1. Male |       695          0 |       695
     2. Female |         0      1,103 |     1,103
---------------+----------------------+----------
         Total |       695      1,103 |     1,798
```

5. Next, create **`male`**:

```
gen male = 1 – female
label var male "Male?"
label def male 1 1_male 0 0_female
label val male male
note male: `tag'
```

6. Aside: Why does this work? How can it be used to reverse code a 5-point scale?

7. For an indicator of being married I use `recode`:

```
. recode marstat (1 2 3 4=1) (5=0), gen(married)
(848 differences between marstat and married)
. label def married 1 1_married 0 0_never
. label val married married
. label var married "Ever married?"
. note married: recoding of marstat | married includes ///
    married, widowed, divorced, separated | `tag'

. tab marstat married, miss

    Marital |          Ever married?
     status |   0_never  1_married         .b |     Total
------------+---------------------------------+----------
  1_married |         0        931          0 |       931
  2_widowed |         0        321          0 |       321
 3_divorced |         0        215          0 |       215
4_separated |         0         33          0 |        33
   5_single |       279          0          0 |       279
         .b |         0          0         19 |        19
------------+---------------------------------+----------
      Total |       279      1,500         19 |     1,798
```

---

8. Indicators of degree and being employed full time:

```
recode edlevel (1 2 3 4 5=0)(6 7=1)(99=.n), gen(hidegree)
label var hidegree "Any higher education?"
label def hidegree 0 0_not 1 1_high_ed
label val hidegree hidegree
note hidegree: recode of edlevel | `tag'
tab edlevel hidegree, miss

recode empstat (1 7=1)(2 3 5 6 8 9 10=0)(98=.d)(99=.n), gen(fulltime)
label def fulltime 1 1_fulltime 0 0_not
label val fulltime fulltime
label var fulltime "Ever worked full time?"
note fulltime: ///
  recoding  empstat; includes fulltime & retired | `tag'
tab empstat fulltime, miss
```

---

9. Before saving the file, I check the new variables:

```
. codebook female-fulltime, compact

Variable    Obs Unique      Mean  Min  Max  Label
--------------------------------------------------------------------
female     1798      2  .6134594    0    1  Female?
male       1798      2  .3865406    0    1  Male?
married    1779      2  .8431703    0    1  Ever married?
hidegree   1795      2  .2250696    0    1  Any higher education?
fulltime   1765      2  .7852691    0    1  Ever worked full time?
```

10. House cleaning and save the file:

```
. local savename wf-russia02.dta
. sort id
. quietly compress
. label data "`savename' | Russia add controls | `dte'"
. note: `savename' / `tag'
. datasignature set, reset
  1798:19(15177):591800297:459057199       (data signature reset)
. save `savename', replace
file wf-russia02.dta saved
```

---

11. After saving a file, I make sure things are working properly:

```
. use `savename', clear
(wf-russia02.dta | Russia add controls | 2017-06-19)
. datasignature confirm
  (data unchanged since 19jun2017 12:38)

. notes

_dta:
  1.  wf-russia01.dta \ wf-isspru01.dta \ wf-russia01-support.do
      jsl 2008-04-02.
  2.  wf-russia02.dta / wflec-add-data-russia1.do Scott Long
      2017-06-19
::: snip :::
female:
  1.  based on gender / wflec-add-data-russia1.do Scott Long
      2017-06-19
male:
  1.  based on gender / wflec-add-data-russia1.do Scott Long
      2017-06-19
::: snip :::
```

Comparing datasets before and after changes

1. `cf` compares data in memory to a saved file and indicates which variables differ:

```
. cf _all using wf-russia01
           female:  does not exist in using
             male:  does not exist in using
          married:  does not exist in using
         hidegree:  does not exist in using
         fulltime:  does not exist in using
r(9);
```

2. The files differ the way they are supposed to.

3. Since `cf` returned an error code, the log file is not closed. The solution is

```
capture noisily cf _all using wf-russia01
```

---

wflec-add-data-russia2-binary.do

1. I load and check the dataset (*wf6-create02-binary.do*):

```
. use wf-russia02, clear
(wf-russia02.dta | Russia add controls | 2017-06-19)

. datasignature confirm
  (data unchanged since 19jun2017 12:38)
```

2. Next, I check the six questions about working women:

```
. codebook momwarm kidsuffer famsuffer wanthome housesat workbest, compact

Variable    Obs Unique      Mean  Min  Max  Label
-------------------------------------------------------------------------------
momwarm    1765      5  2.324646    1    5  Working mom can have warm relat...
kidsuffer  1755      5  2.373789    1    5  Pre-school child suffers?
famsuffer  1759      5  2.401933    1    5  Family life suffers?
wanthome   1717      5  2.639487    1    5  Women really want is home & kids?
housesat   1680      5  2.855357    1    5  Housework satisfies like paid job?
workbest   1710      5  2.071345    1    5  Work best for women's independe...
-------------------------------------------------------------------------------
```

3. Each variable is a five-point scale. For example,

```
. tab1 momwarm, miss

-> tabulation of momwarm

  Working mom |
can have warm |
  relations w |
        kids? |      Freq.     Percent        Cum.
--------------+-----------------------------------
     1StAgree |        464       25.81       25.81
       2Agree |        712       39.60       65.41
     3Neither |        197       10.96       76.36
    4Disagree |        336       18.69       95.05
  5StDisagree |         56        3.11       98.16
a_Can't choose |         26        1.45       99.61
    b_Refused |          7        0.39      100.00
--------------+-----------------------------------
        Total |      1,798      100.00
```

4. **`momwarm`** and **`workbest`** are coded so that agreeing is positive; **`kidsuffer`**, **`famsuffer`**, **`wanthome`**, and **`housesat`** are coded so that agreeing is negative.

5. An easy way to verify this:

```
. pwcorr momwarm kidsuffer famsuffer wanthome housesat workbest, obs

             |  momwarm kidsuf~r famsuf~r wanthome housesat workbest
-------------+------------------------------------------------------
     momwarm |   1.0000
             |     1765
             |
   kidsuffer |  -0.2494   1.0000
             |     1736     1755
             |
   famsuffer |  -0.2517   0.5767   1.0000
             |     1737     1738     1759
             |
    wanthome |  -0.1069   0.2357   0.2977   1.0000
             |     1698     1688     1698     1717
             |
    housesat |  -0.0148   0.1465   0.1921   0.4133   1.0000
             |     1664     1657     1662     1649     1680
             |
    workbest |   0.0624  -0.0220  -0.0717  -0.1369  -0.2019   1.0000
             |     1691     1684     1690     1659     1636     1710
```

6. To recode variables so 1 is positive; 0 is negative. My first attempt is:

```
label def Lagree 1 1_agree 0 0_not .a a_Unsure  ///
    .b b_Refused .n n_Neutral
```

7. This is confusing since someone can agree with a positive statement and can also agree with a negative statement. So:

```
label def Lprowork 1 1_yesPos0 0_noNeg .a a_Unsure  ///
    .b b_Refused .n n_Neutral
```

8. I dichotomize **`momwarm`** and add labels and notes. I treat neutral as missing (i.e., **`3=.`**) and add a note to remind me of this decision:

```
. * momwarm: 1=SA working mom can have warm relationship
. * Bwarm:   1=agree (not reversed)
. recode momwarm (1/2=1) (4/5=0) (3=.n), gen(Bwarm)
(1301 differences between momwarm and Bwarm)

. label var Bwarm "Working mom can have warm relations?"
. label val Bwarm Lprowork
. note Bwarm: 3=neutral in source was coded .n | `tag'
```

9. To verify the recode,

```
. tab Bwarm momwarm, miss

   Working |
   mom can |
 have warm |      Working mom can have warm relations w kids?
relations? |  1StAgree     2Agree   3Neither  4Disagree  5StDisagr |     Total
-----------+-------------------------------------------------------+----------
   0_noNeg |         0          0          0        336         56 |       392
  1_yesPos |       464        712          0          0          0 |     1,176
  a_Unsure |         0          0          0          0          0 |        26
 b_Refused |         0          0          0          0          0 |         7
 n_Neutral |         0          0        197          0          0 |       197
-----------+-------------------------------------------------------+----------
     Total |       464        712        197        336         56 |     1,798

   Working | Working mom can have
   mom can |   warm relations w
 have warm |         kids?
relations? | a_Can't c  b_Refused |     Total
-----------+----------------------+----------
   0_noNeg |         0          0 |       392
  1_yesPos |         0          0 |     1,176
  a_Unsure |        26          0 |        26
 b_Refused |         0          7 |         7
 n_Neutral |         0          0 |       197
-----------+----------------------+----------
     Total |        26          7 |     1,798
```

10. Since each variable is transformed in the same way, I automate:

```
local vin  momwarm
local vout Bwarm

recode `vin' (1/2=1)(4/5=0)(3=.n), gen(`vout')
label var `vout' "Working mom can have warm relations?"
label val `vout' Lprowork
note `vout': 3=neutral in source was coded .n | `tag'
tab `vout' `vin', miss
```

11. Then, to recode **`workbest`**:

```
local vin  workbest
local vout Bindep
gen `vout' = `vin'
label var `vout' "Agree work creates independence?"
recode `vout' (1/2=1)(4/5=0)(3=.n)
note `vout': 3=neutral in source was coded .n | `tag'
tab `vin' `vout', miss
```

And so on...

12. I check that the binary variables are coded in the same direction:

```
. pwcorr B*, obs

             |    Bwarm    Bkids  Bfamily  Bnohome  Bjobsat   Bindep
-------------+------------------------------------------------------
       Bwarm |   1.0000
             |     1568
             |
       Bkids |   0.2351   1.0000
             |     1311     1483
             |
     Bfamily |   0.2289   0.5775   1.0000
             |     1314     1312     1481
             |
     Bnohome |   0.1382   0.2327   0.2850   1.0000
             |     1208     1157     1161     1352
             |
     Bjobsat |   0.0396   0.1446   0.1672   0.4659   1.0000
             |     1119     1071     1071     1033     1248
             |
      Bindep |   0.0345   0.0185   0.0546   0.1048   0.1538   1.0000
             |     1279     1217     1211     1122     1058     1438
```

13. Housekeeping and save:

```
. local savename wf-russia03.dta
. sort id
. quietly compress
. label data "`savename' | Russia adding example step 2 | `dte'"
. note: `savename' / `tag'
. datasignature set, reset
  1798:25(35160):2565153827:1461624542     (data signature reset)

. capture noisily cf _all using wf-russia02.dta
           Bwarm:  does not exist in using
           Bkids:  does not exist in using
         Bfamily:  does not exist in using
         Bnohome:  does not exist in using
         Bjobsat:  does not exist in using
          Bindep:  does not exist in using
```

---

wflec-add-data-russia3-noneutral.do

1. I exclude neutral and code them in the same direction. I start by loading the data and setting up a value label:

```
. use wf-russia03.dta, replace
(wf-russia03.dta | Russia adding example step 2 | 2017-06-19)
. datasignature confirm
  (data unchanged since 19jun2017 12:44)

. label def Lsa_sd 1 1_SA_Pos 2 2_A_Pos 3 3_D_Neg ///
>     4 4_SD_Neg .a a_Unsure .b b_Refused .n n_Neutral
```

2. I want to use the same code for each variable. I use locals to indicate the variable being transformed and the new variable:

```
. * momwarm: 1=SA working mom can have warm relationship
. * C4warm:  1=SA (not reversed)
. local vin momwarm
. local vout C4warm
```

More on next page...

```
. * momwarm: 1=SA working mom can have warm relationship
. * C4warm:  1=SA (not reversed)
. local vin momwarm
. local vout C4warm

. recode `vin' (1=1) (2=2) (3=.n) (4=3) (5=4), gen(`vout')
(589 differences between momwarm and C4warm)
. label var `vout' "Working mom can have warm relations?"
. label val `vout' Lsa_sd
. note `vout': 3=neutral in source was coded .n | `tag'
. tab `vin' `vout', m

   Working mom |
 can have warm |
   relations w |    Working mom can have warm relations?
         kids? |  1_SA_Pos    2_A_Pos    3_D_Neg   4_SD_Neg |     Total
---------------+--------------------------------------------+----------
      1StAgree |       464          0          0          0 |       464
        2Agree |         0        712          0          0 |       712
      3Neither |         0          0          0          0 |       197
     4Disagree |         0          0        336          0 |       336
   5StDisagree |         0          0          0         56 |        56
a_Can't choose |         0          0          0          0 |        26
     b_Refused |         0          0          0          0 |         7
---------------+--------------------------------------------+----------
         Total |       464        712        336         56 |     1,798
```

```
   Working mom |
 can have warm |    Working mom can have warm
   relations w |           relations?
         kids? |  a_Unsure  b_Refused  n_Neutral |     Total
---------------+---------------------------------+----------
      1StAgree |         0          0          0 |       464
        2Agree |         0          0          0 |       712
      3Neither |         0          0        197 |       197
     4Disagree |         0          0          0 |       336
   5StDisagree |         0          0          0 |        56
a_Can't choose |        26          0          0 |        26
     b_Refused |         0          7          0 |         7
---------------+---------------------------------+----------
         Total |        26          7        197 |     1,798
```

3. After creating the other scales, I check that they are coded in the same direction with `pwcorr C4*`. This is why I start the variables with **`C4`**. I also check with the corresponding binary variable. Output not shown.

```
. pwcorr C4*, obs
             |   C4warm   C4kids C4family C4nohome C4jobsat  C4indep
-------------+------------------------------------------------------
      C4warm |   1.0000
             |     1568
             |
      C4kids |   0.1942   1.0000
             |     1311     1483
∷ snip ∷
```

---

4. I want correlations between the B version and the C4 versions of each variable, but don't want correlations for all B and all C4 variables, so I use a loop:

```
. * through stem of variables
. foreach s in warm kids family nohome jobsat indep {

.         * correlate the B and C4 version of the stem
  2.      pwcorr B`s' C4`s', obs
  3. }
             |    Bwarm   C4warm
-------------+------------------
       Bwarm |   1.0000
             |     1568
             |
      C4warm |  -0.8239   1.0000
             |     1568     1568
             |    Bkids   C4kids
-------------+------------------
       Bkids |   1.0000
             |     1483
             |
      C4kids |  -0.8114   1.0000
             |     1483     1483
```

(output omitted)

---

5. Bookkeeping and save the file.

```
. local savename wf-russia04.dta
. sort id
. quietly compress

. label data "`savename' | Russia adding example step 3 | `dte'"
. note: `savename' / `tag'
. datasignature set, reset
  1798:31(64962):3525882739:3285584125       (data signature
reset)
. save `savename', replace
file wf-russia04.dta saved

. capture noisily cf _all using wf-russia03.dta
          C4warm:  does not exist in using
          C4kids:  does not exist in using
        C4family:  does not exist in using
        C4nohome:  does not exist in using
        C4jobsat:  does not exist in using
         C4indep:  does not exist in using
```

Challenge

Find a better way to compute the correlations.

wflec-add-data-russia4-analysisvars.do

```
capture log close
log using wflec-add-data-russia4-analysis, replace text
version 13.1
clear all
set linesize 80
macro drop _all
set scheme s1manual

//  Reproducible results example: adding variables - russia step 1
//  Data revisions during analysis

local pgm wflec-add-data-russia4
local dte 2017-06-19
local who Scott Long
local tag "`pgm'.do `who' `dte'"

//  #1 load data

use wf-russia04, replace
datasignature confirm

//  #2 dataset revisions during for analysis

gen agesq = age*age
lab var agesq "age*age"
note agesq: `tag'
```

```
//  #3 cleanup and save

local savename wf-russia05.dta
sort id
quietly compress
label data "`savename' | Russia adding example step 4 | `dte'"
note: `savename' / `tag'
datasignature set, reset
save `savename', replace

codebook, compact

//  #5 check the changes

capture noisily cf _all using wf-russia04.dta

log close
exit
```

---

wflec-add-data-russia0-master.do

```
    //  Reproducible results example: adding variables
    //  Russian data master do-file Scott Long 2017-06-19

    do wflec-add-data-russia1-controls.do
    do wflec-add-data-russia2-binary.do
    do wflec-add-data-russia3-noneutral.do
    do wflec-add-data-russia4-analysisvars.do

    exit
```

Adding variables during analysis

1. Adding variables often occurs during analysis.
2. Analysis interferes with carefully creating variables.
3. Part 18 considers how to integrate adding variables with analysis.

Part 17*: Importing data

WFDAUS pages 198-210.

Assume that data are imported incorrectly.

1. Converting data between formats can introduce errors.
   - Missing data is sometimes incorrectly converted
   - Names or labels might be truncated
   - Metadata might be lost.
2. If the conversion introduces errors, everything that follows can be wrong.

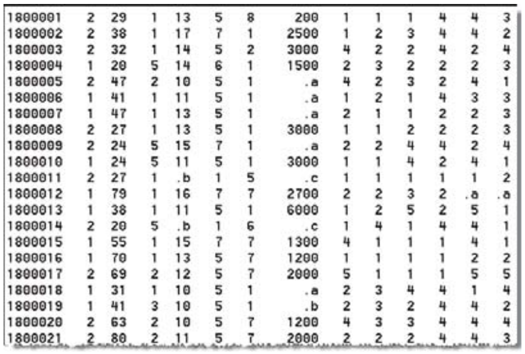
Data formats

1. There is no universal format for data.
2. Among the 100s of formats, there are two basic types:
   - c. *ASCII* formats
   - d. *Binary* formats

ASCII data formats

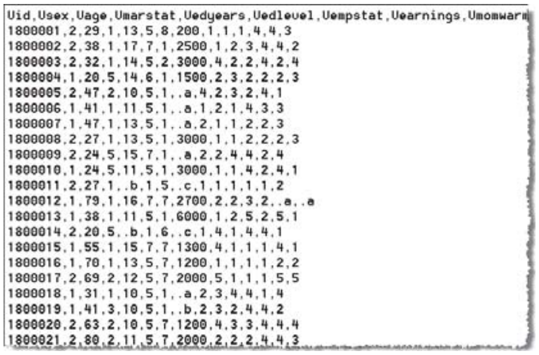
1. **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange is a standard for data files that originated in 1967.
2. Data are stored as *plain text*.
3. The disadvantage is that it does not include metadata except as auxiliary files.
4. ASCII come in two flavors:
   - a. *Fixed format*: column locations are fixed
   - b. *Free format*: spaces or characters separate variables
5. Examples of ASCII data:

Fixed format



```
1800001   2  29   1  13   5   8    200   1   1   1   4   4   3
1800002   2  38   1  17   7   1   2500   1   2   3   4   4   2
1800003   2  32   1  14   5   2   3000   4   2   2   4   2   4
1800004   1  20   5  14   6   1   1500   2   3   2   2   2   3
1800005   2  47   2  10   5   1     .a   4   2   3   2   4   1
1800006   1  41   1  11   5   1     .a   1   2   1   4   3   3
1800007   1  47   1  13   5   1     .a   2   1   1   2   2   3
1800008   2  27   1  13   5   1   3000   1   1   2   2   2   3
1800009   2  24   5  15   7   1     .a   2   2   4   4   2   4
1800010   1  24   5  11   5   1   3000   1   1   4   2   4   1
1800011   2  27   1  .b   1   5     .c   1   1   1   1   1   2
1800012   1  79   1  16   7   7   2700   2   2   3   2  .a  .a
1800013   1  38   1  11   5   1   6000   1   2   5   2   5   1
1800014   2  20   5  .b   1   6     .c   1   4   1   4   4   1
1800015   1  55   1  15   7   7   1300   4   1   1   1   4   1
1800016   1  70   1  13   5   7   1200   1   1   1   1   2   2
1800017   2  69   2  12   5   7   2000   5   1   1   1   5   5
1800018   1  31   1  10   5   1     .a   2   3   4   4   1   4
1800019   1  41   3  10   5   1     .b   2   3   2   4   4   2
1800020   2  63   2  10   5   7   1200   4   3   3   4   4   4
1800021   2  80   2  11   5   7   2000   2   2   2   4   4   3
```

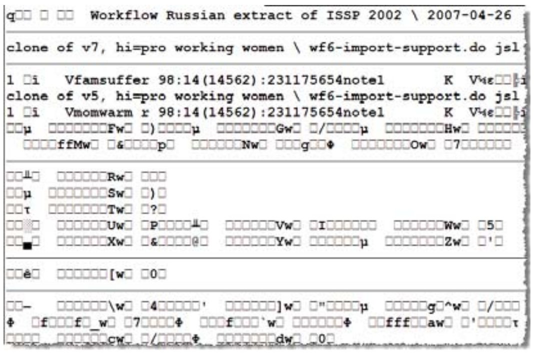
Free format



```
Uid,Usex,Uage,Umarstat,Uedyears,Uedlevel,Uempstat,Uearnings,Umomwarm
1800001,2,29,1,13,5,8,200,1,1,1,4,4,3
1800002,2,38,1,17,7,1,2500,1,2,3,4,4,2
1800003,2,32,1,14,5,2,3000,4,2,2,4,2,4
1800004,1,20,5,14,6,1,1500,2,3,2,2,2,3
1800005,2,47,2,10,5,1,.a,4,2,3,2,4,1
1800006,1,41,1,11,5,1,.a,1,2,1,4,3,3
1800007,1,47,1,13,5,1,.a,2,1,1,2,2,3
1800008,2,27,1,13,5,1,3000,1,1,2,2,2,3
1800009,2,24,5,15,7,1,.a,2,2,4,4,2,4
1800010,1,24,5,11,5,1,3000,1,1,4,2,4,1
1800011,2,27,1,.b,1,5,.c,1,1,1,1,1,2
1800012,1,79,1,16,7,7,2700,2,2,3,2,.a,.a
1800013,1,38,1,11,5,1,6000,1,2,5,2,5,1
1800014,2,20,5,.b,1,6,.c,1,4,1,4,4,1
1800015,1,55,1,15,7,7,1300,4,1,1,1,4,1
1800016,1,70,1,13,5,7,1200,1,1,1,1,2,2
1800017,2,69,2,12,5,7,2000,5,1,1,1,5,5
1800018,1,31,1,10,5,1,.a,2,3,4,4,1,4
1800019,1,41,3,10,5,1,.b,2,3,2,4,4,2
1800020,2,63,2,10,5,7,1200,4,3,3,4,4,4
1800021,2,80,2,11,5,7,2000,2,2,2,4,4,3
```

Binary data formats

1. Binary formats use a complex format that requires software to decode.
2. Each group of eight 0's or 1's is translated to a single character which may or may not be visible:



```
q□□ □ □□  Workflow Russian extract of ISSP 2002 \ 2007-04-26

clone of v7, hi=pro working women \ wf6-import-support.do jsl

1 □i   Vfamsuffer 98:14(14562):231175654notel        K  V¼ε□□
clone of v5, hi=pro working women \ wf6-import-support.do jsl
1 □i   Vmomwarm r 98:14(14562):231175654notel        K  V¼ε□□
□□µ  □□□□□□□Fw□ □)□□□□µ  □□□□□□□Gw□ □/□□□□µ  □□□□□□□Hw□ □□□□□
  □□□□ffMw□ □&□□□□p□  □□□□□□Nw□ □□□g□□  □□□□□□□Ow□ □7□□□□□□

□□╨□  □□□□□□Rw□ □□□
□□µ  □□□□□□□Sw□ □)□
□□τ  □□□□□□□Tw□ □?□
□□□□  □□□□□□Uw□ □P□□□□╨□  □□□□□□Vw□ □I□□□□□□  □□□□□□Ww□ □5□
□□■□  □□□□□□Xw□ □&□□□□@□  □□□□□□Yw□ □□□□□□µ  □□□□□□□Zw□ □'□

□□ê□  □□□□□□[w□ □0□

□□–   □□□□□□\w□ □4□□□□□'  □□□□□□]w□ □"□□□□µ  □□□□□g□^w□ □/□□□
Φ  □f□□□f□_w□ □7□□□□Φ  □□□f□□□`w□ □□□□□□Φ  □□fff□□aw□ □'□□□□τ
□□□□  □□□□□□cw□ □/□□□□Φ  □□□□□□□dw□ □0□
```

3. Binary files contain *metadata* such as variable labels, value labels, and notes that are loaded with the dataset.
4. Statistical packages use their own format; new versions of a program often use enhanced formats to incorporate new features.

Warnings

1. If you save data in a binary format that becomes obsolete, you risk having the file but not being able to decode it.
2. Converting data from one binary format discards information that is not compatible with the destination format.

Stata commands for ASCII formats

1. **`insheet`** reads tab-delimited or comma-delimited files.
2. **`infile`** reads space, tab, or comma delimited free-format files and fixed format files using a dictionary with locations of variables.
3. **`infix`** reads fixed format files that do not have a dictionary.
4. Note: Stata has unified commands for importing and exporting data into the **`import`** and **`export`** commands. I use the old names, which still work.

Importing free format ASCII (wf6-import.do)

Since the first row contains names of the variables:

```
. insheet using wf6-import-free.txt, clear
(14 vars, 1798 obs)
. list vid-vempstat in 1/5, clean

           vid   vsex   vage   vmarstat   vedyears   vedlevel   vempstat
  1.   1800001      2     29          1         13          5          8
  2.   1800002      2     38          1         17          7          1
  3.   1800003      2     32          1         14          5          2
  4.   1800004      1     20          5         14          6          1
  5.   1800005      2     47          2         10          5          1
```

Importing SAS XPORT files

If your data are in SAS XPORT transport format, use **`fdause`** command.

`fdause wf6-import-fdause.xpt, clear`

`fdause` can also read value labels from a formats.xpf XPORT file.

Importing odbc files: rare

ODBC (Open DataBase Connectivity) is a standard format for exchanging data between programs and can be read with the **`odbc`** command.

Importing XML data: rare

`xmluse` command reads Extensible Markup Language (XML) which is designed to be highly portable. **`xmlsave`** saves data in XML format.

Use statistical packages to export data

1. Many programs save data in alternative formats.
    - SPSS can write data in Stata format
    - SAS can write data in SAS XPORT format
2. This is an effective way to convert data between formats.

Using a data conversion program

1. Stat/Transfer converts among many formats.

Verifying data conversion

1. Things go wrong when you convert data from one format to another.
2. With ASCII in fixed format, it is easy to make a mistake specifying columns.
3. When using other methods to convert data the problems can be more subtle.

Assume the imported data is wrong

Step 1: Compare statistics from source and destination

1. Compute descriptive statistics including frequencies using:
    a. Source statistical package
    b. Unconverted source data.
2. Compute the same information using:
    a. Stata
    b. Converted data.
3. Verify that *everything* matches.

Step 2: Examine the missing values

1. Compare the distribution of missing values for *all* variables using the source program with the source data and Stata with the converted data.
2. Tabulate values with **`tab1`** *variable-list*, **`missing`**.
3. The most frequent problem I see is that multiple missing value codes in the source data are merged to a single value when converted.

Step 3: Convert the data two ways

1. For example, use SPSS to export the data into Stata format.
2. Use Stat/Transfer to convert the same source data into Stata format.
3. In Stata, use **`cf`** to compare the two files. If the files match exactly, you can be *more* confident in the conversions.

Example: Converting the ISSP 2002 data from Russia

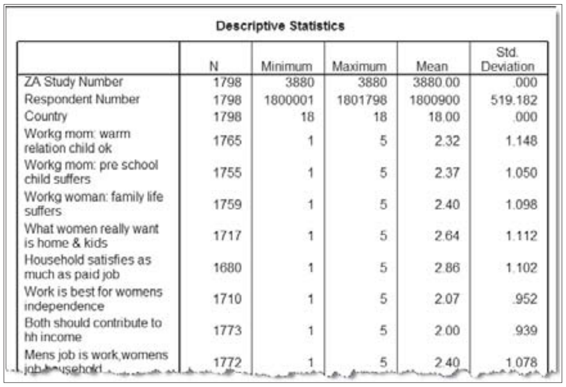
2002 Russian Family and Changing Gender Roles III Study (ISSP 2004).

Step 1: Examining the data in SPSS

1. The source file **`04106-0001-Data.por`** is a SPSS Portable file which includes variable labels, value labels, and information on missing values.
2. I open the dataset with SPSS Version 14 and selected cases with **`v3`** equal to 18, the code for Russia, and save a SPSS binary file **`wf6-isspru-spss01.sav`**.

---

3. In SPSS I computed descriptive statistics:

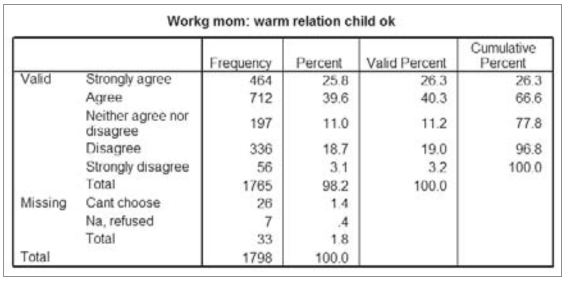


Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
ZA Study Number	1798	3880	3880	3880.00	.000
Respondent Number	1798	1800001	1801798	1800900	519.182
Country	1798	18	18	18.00	.000
Workg mom: warm relation child ok	1765	1	5	2.32	1.148
Workg mom: pre school child suffers	1755	1	5	2.37	1.050
Workg woman: family life suffers	1759	1	5	2.40	1.098
What women really want is home & kids	1717	1	5	2.64	1.112
Household satisfies as much as paid job	1680	1	5	2.86	1.102
Work is best for womens independence	1710	1	5	2.07	.952
Both should contribute to hh income	1773	1	5	2.00	.939
Mens job is work,womens job household	1772	1	5	2.40	1.078

---

4. I computed frequency distributions for all variables:



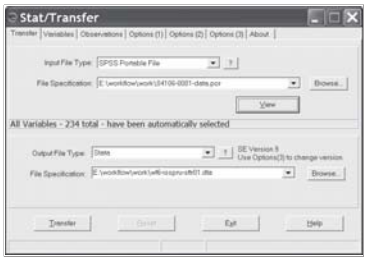
Workg mom: warm relation child ok

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly agree	464	25.8	26.3	26.3
	Agree	712	39.6	40.3	66.6
	Neither agree nor disagree	197	11.0	11.2	77.8
	Disagree	336	18.7	19.0	96.8
	Strongly disagree	56	3.1	3.2	100.0
	Total	1765	98.2	100.0	
Missing	Cant choose	26	1.4		
	Na, refused	7	.4		
	Total	33	1.8		
Total		1798	100.0		

5. And so on...

Step 2: Converting data to Stata format

1. Using Stat/Transfer Version 8 I create **`wf6-isspru-sttr01.dta`**:



Stat/Transfer
Transfer | Variables | Observations | Options (1) | Options (2) | Options (3) | About
?
Browse...
View
?
Browse...
Transfer
Reset
Exit
Help

Input File Type: SPSS Portable File

File Specification: E:\workflow\work\04106-0001-data.por

All Variables - 234 total - have been automatically selected

Output File Type: Stata

SE Version 8
Use Options(3) to change version

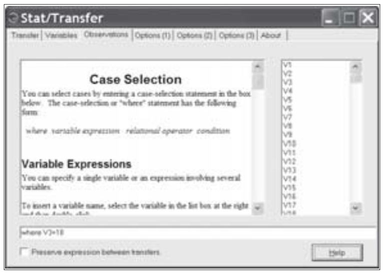
File Specification: E:\workflow\work\wf6-isspru-sttr01.dta

2. The file to convert **`04106-0001-data.por`**, not **`wf6-isspru-spss01.sav`**.

- o Avoid converting a converted dataset.

---

3. Switching to the Observations tab, I select cases from Russia:



Stat/Transfer
Transfer | Variables | Observations | Options (1) | Options (2) | Options (3) | About
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
Help

Case Selection

You can select cases by entering a case-selection statement in the box below. The case-selection or "where" statement has the following form:

where variable expression relational operator condition

Variable Expressions

You can specify a single variable or an expression involving several variables.

To insert a variable name, select the variable in the list box at the right

where V3=18

☐ Preserve expression between transfers

4. Returning to the Transfer tab, I click Transfer to create **`wf6-isspru-sttr01.dta`**.

- o If asked choose double precision.
- o The file will be larger, but you can compress it later.

Step 3: Verifying the transfer

1. To check the transferred data, I use **`codebook`** (*wf6-isspru-sttr01.do*):

```
. codebook, compact

Variable  Obs Unique      Mean      Min      Max  Label
-------------------------------------------------------------------------------
V1       1798      1      3880     3880     3880  ZA Study Number
V2       1798   1798   1800900  1800001  1801798  Respondent Number
V3       1798      1        18       18       18  Country
V4       1765      5  2.324646        1        5  Workg mom: warm relation child ok
V5       1755      5  2.373789        1        5  Workg mom: pre school child suffers
V6       1759      5  2.401933        1        5  Workg woman: family life suffers
V7       1717      5  2.639487        1        5  What women really want is home & kids
V8       1680      5  2.855357        1        5  Household satisfies as much as paid job
V9       1710      5  2.071345        1        5  Work is best for womens independence
V10      1773      5  2.003948        1        5  Both should contribute to hh income
V11      1772      5  2.401806        1        5  Mens job is work,womens job household
```

(output omitted)

2. Results match SPSS except that variable names are upper case.

---

3. Since descriptive statistics exclude missing values, I use **`tab1`** to verify that missing values have been converted correctly:

```
-> tabulation of V4

  Workg mom: warm relation |
                  child ok |      Freq.     Percent        Cum.
---------------------------+-----------------------------------
            Strongly agree |        464       25.81       25.81
                     Agree |        712       39.60       65.41
Neither agree nor disagree |        197       10.96       76.36
                  Disagree |        336       18.69       95.05
         Strongly disagree |         56        3.11       98.16
                        .c |         26        1.45       99.61
                        .n |          7        0.39      100.00
---------------------------+-----------------------------------
                     Total |      1,798      100.00
```

4. OK. The first one matches? Is that good enough?

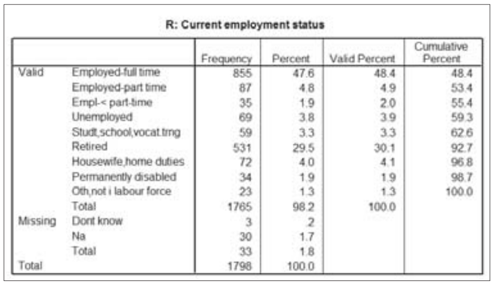
---

5. Hours later, I found a problem with **`V239`**. Where did 33 come from?

```
-> tabulation of V239

  R: Current employment |
status                  |      Freq.     Percent        Cum.
------------------------+-----------------------------------
      Employed-full time |        855       47.55       47.55
      Employed-part time |         87        4.84       52.39
         Empl-< part-time |         35        1.95       54.34
              Unemployed |         69        3.84       58.18
Studt,school,vocat.trng |         59        3.28       61.46
                 Retired |        531       29.53       90.99
   Housewife,home duties |         72        4.00       94.99
    Permanently disabled |         34        1.89       96.89
   Oth,not i labour force |         23        1.28       98.16
                      .a |         33        1.84      100.00
------------------------+-----------------------------------
                   Total |      1,798      100.00
```

---

6. The 33 **`.a`**'s in Stata correspond to 3 \`Dont know' + 30 \`Na' in SPSS:



R: Current employment status

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Employed-full time	855	47.6	48.4	48.4
	Employed-part time	87	4.8	4.9	53.4
	Empl-< part-time	35	1.9	2.0	55.4
	Unemployed	69	3.8	3.9	59.3
	Studt,school,vocat.trng	59	3.3	3.3	62.6
	Retired	531	29.5	30.1	92.7
	Housewife,home duties	72	4.0	4.1	96.8
	Permanently disabled	34	1.9	1.9	98.7
	Oth,not i labour force	23	1.3	1.3	100.0
	Total	1765	98.2	100.0	
Missing	Dont know	3	.2		
	Na	30	1.7		
	Total	33	1.8		
Total		1798	100.0		

7. In Stat/Transfer 8 I used the default option

"Use all, Map to extended (a-z) missing values."

6. I tried "Use None, Map to extended (a-z) missing values".
7. This did not combine missing values. Using this option, I created `wf6-isspru-sttr02.dta`. Now:

```
-> tabulation of V239

  R: Current employment |
       status           |      Freq.     Percent        Cum.
------------------------+-----------------------------------
     Employed-full time |        855       47.55       47.55
     Employed-part time |         87        4.84       52.39
       Empl-< part-time |         35        1.95       54.34
             Unemployed |         69        3.84       58.18
Studt,school,vocat.trng |         59        3.28       61.46
                Retired |        531       29.53       90.99
  Housewife,home duties |         72        4.00       94.99
   Permanently disabled |         34        1.89       96.89
 Oth,not i labour force |         23        1.28       98.16
              Dont know |          3        0.17       98.33
                     Na |         30        1.67      100.00
------------------------+-----------------------------------
                  Total |      1,798      100.00
```

8. The 33 cases are correctly split into two categories, but:

```
-> tabulation of V239

 R: Current |
 employment |
     status |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        855       47.55       47.55
          2 |         87        4.84       52.39
          3 |         35        1.95       54.34
          5 |         69        3.84       58.18
          6 |         59        3.28       61.46
          7 |        531       29.53       90.99
          8 |         72        4.00       94.99
          9 |         34        1.89       96.89
         10 |         23        1.28       98.16
         98 |          3        0.17       98.33
         99 |         30        1.67      100.00
------------+-----------------------------------
      Total |      1,798      100.00
```

9. Since I want missing value codes, not numbers, I tried DBMS/Copy and encountered the same problem. (DBMS/Copy is no longer available)

10. I used SPSS to save the data directly to Stata format. While SPSS did not combine categories, it also did not use extended missing value codes.
11. SPSS writes:

> "The problem comes when converting SPSS data to another platform like Stata. During the translation to Stata, SPSS treats these user-missings as system missing values. So, when we convert to another program like Stata, these values get treated like system missing values (e.g. a user-missing value of 8 gets translated as .)."

I don't follow this exactly, but it tells to be very careful with missing data conversions.

Summary

1. I try to avoid converting data from one format to another.
2. Some data providers provide data in multiple formats.
3. Some programs let you read data in multiple formats.
4. If you must convert, check every variable and pay particular attention to missing values.

Part 18: Data analysis

WFDAUS pages 287-318.

Overview

1. Reproducible data management makes analysis more efficient and accurate.
2. Start with a plan, since the next "obvious" thing might not be the best thing.
3. The computing workflow makes two critical tasks simpler:
   - Retaining provenance
   - Revising analysis

Planning analysis

1. It is tempting to start analysis without a plan.
   - I might do this when I get a long-awaited dataset.
   - Prudence overcomes enthusiasm and I make a plan.
2. Three dimensions of planning (Oliveira and Stewart, 2006: 59-70)
   - **In the large** *Objectives* of the work
   - **In the middle** *Manageable tasks* within objectives
   - **In the small** *Necessary details* to accomplish tasks
3. Your plan depends on many things.
   - How specific is your research question?
   - Are you testing well defined hypotheses or exploring new ideas?
   - Are you familiar with the data and the statistical methods?
   - Is this a "one and done" paper or part of a sequence where consistency must be maintained across papers?

Planning and organization

1. Plans should include explicit ideas on organizing your results.
2. Decide on names for variables to be added.
3. Decide on how to name new files.

Plan tables and graphs

1. Dummy tables and figures are an effective way to plan your analysis. For example:

	Women		Men		Test of equal means	
Variable	Mean	Std. Dev.	Mean	Std. Dev.	*t*-test	Prob
Tenure						
Year						
Selectivity						
Articles						
Prestige						

Planning and Collaboration

1. For collaborations, planning is even more important.
2. Collaborators might *initially* say:
   "Whatever you think is fine. When do we get the results?"
3. Collaborators *later* say:
   "Why did you do it that way? Wouldn't this have been better?
4. Insist on discussing the plan.
   - Amazingly useful ideas can come out of these discussions.

Planning in the small

1. How will you accomplish the tasks from the middle level plan.
   - The **nitty-gritty details** (Oliveira and Stewart 2006:61)
2. Decide
   - Which variables to useand how to code them (e.g., Do you want to treat two years of college as the same as having a two year degree from a trade or professional school?)
   - Which commands to use (e.g., Should the hurdle model be estimated with **`logit`** and **`ztp`** or with **`hhplogit`**?).

Dangers

1. When planning in the small, periodically step back to make sure you understand what the larger objectives are.
2. Planning in the small can take you far off track as you learn new tools. Solving a puzzle dominates your thinking.
3. Always remember why you are doing what you are doing.

One way to approach analysis

1. Analysis involves many small decisions.
2. Suppose you have 5 choices such as:

Choice 1	*a*: Exclude middle category	*b*: Keep middle category
Choice 2	*a*: Dichotomize SA + A vs not	*b*: SA + A + N vs not
::		

3. It is tempting to try all combinations:

	Choices				
	1	*2*	*3*	*4*	*5*
Set 1:	Y	Y	Y	Y	Y
Set 2:	Y	Y	Y	Y	N
::					
Set 32:	N	N	N	N	N

4. There are 32 ways to proceed; 1024 ways with 10 decisions!

---

5. Prioritize the choices by importance.
6. Decide on the first most important and stick with it.
7. Keep that choice and move on to the next most important decision.
8. At each step, keep track of issues you want to revisit later.
9. When all issues have been resolved, explore concerns you have and assess how robust your results are (more on this later).

Example

1. Variable A is most critical and variables b c d e f and g are the default options for othr variables.
2. Model: logit y A b c d e f g and logit y a b c d e f g
   - A is chosen
3. Model logit y A b c d e f g and logit y A B c d e f g
   - b is chosen
4. And so on...

Baseline statistics

1. This is basic but often overlooked.
2. Create a do-file that produces descriptive statistics for key variables. Keep the results handy.
3. Start all analyses do-files by checking descriptive statistics.

Example of baseline statistics

1. Select the sample: verify the N's

```
//  #1 load data and select sample

. use wf-tenure, clear
(Workflow data for gender differences in tenure | 2008-04-02)
. datasignature confirm
  (data unchanged since 02apr2008 13:29)
```

```
. tabulate sampleis

 Sample for |
     tenure |
   analysis |      Freq.     Percent        Cum.
------------+-----------------------------------
      0_Not |        148        5.03        5.03
 1_InSample |      2,797       94.97      100.00
------------+-----------------------------------
      Total |      2,945      100.00

* select analysis sample
. keep if sampleis
(148 observations deleted)
```

2. Descriptive statistics for the analysis sample. Verify that names and labels are clear; if not, go back to data management.

```
. //  #2 desc statistics for men & women combined

. codebook female male tenure year yearsq select articles prestige, compact

Variable    Obs Unique      Mean  Min  Max  Label
-----------------------------------------------------------------------------
female     2797      2  .3775474    0    1  Scientist is female?
male       2797      2  .6224526    0    1  Is male?
```

```
tenure     2797      2  .1229889    0    1  Is tenured?
year       2797     10  3.855917    1   10  Years in rank
yearsq     2797     10  20.16911    1  100  Years in rank squared
select     2797      8  4.995048    1    7  Baccalaureate selectivity
articles   2797     48  7.050411    0   73  Total number of articles
prestige   2797     98  2.646591  .65  4.8  Prestige of department
-----------------------------------------------------------------------------
```

3. Descriptive statistics by gender since I am compare men and women

```
//  #3 desc statistics for women

codebook female male tenure year yearsq select articles prestige ///
    if female, compact

//  #4 desc statistics for men

codebook female male tenure year yearsq select articles prestige ///
    if male, compact
```

Baseline statistics in analysis files

1. At the start of an analysis do-file, create a local with the variables you will be using, say **`analysisvars`**.

```
local analysisvars female male tenure year ///
    yearsq select articles prestige
```

2. After you have selected the sample:

```
codebook `analysisvars', compact
```

3. Check the results and compare to your baseline statistics.

Pavalko, Gong, and Long 2007

1. Part 6 discussed planning the cohort, work and health paper.

2. Substantive plans were translated into tasks that evolved as work proceeded.

3. New tasks emerged when initial analyses showed that issues were more complex than anticipated.

- How do new findings affect other plans?
- Do they imply changes to earlier analyses?
- Are unanticipated variables needed?

The CWH middle plan

CWH in the middle: specific tasks

Red is for data management; Blue is for data analysis.

cwh01: **Descriptive statistics.** Basic descriptive statistics

cwh02: **Compare count models for number of health limitations.**

cwh03: **Logit model for having any limitations.** Due to many 0's logit of no limitations or any.

cwh04: **Hurdle model for number of limitations.** Combine logit with count models.

cwh05: **Data management.** Add interaction variables.

cwh06: **Data management.** Add data from the 1971 panel.

cwh07: **Count models with 1971 data included**.

cwh08: **Hurdle model using alternative parameterizations**.

cwh09: **Sensitivity analysis of hurdle model**.

Using these results, we wrote the first draft

- After discussing draft, we planned new analyses with new variables.
- New analyses could have been added to the tasks above, be we created new tasks organized around the tables in for the revised paper.
- When a paper is almost ready to circulate, having tasks organized around tables/figures is often convenient for making changes later.
- The results in the revised paper were all from these tasks:

cwh10: **Data management**. Add additional variables.

cwh11: **Descriptive statistics for Tables 1, 2, and 3**.

cwh12: **Hurdle models and predictions for Figures 1 through 5**.

cwh13: **Supplementary analyses with the hurdle model for Table 4**.

Revised paper submitted

- After R&R, added variables, refined the coding of other variables, and updated our figures and tables. Since the analyses in the paper were included in tasks **cwh10** through **cwh13**, revisions were simple.

cwh14: **Data management.** Add work and smoking variables; revise some operational definitions.

cwh15: **Re-estimate models and create plots**.

cwh16: **Estimate additional models for Table 4.**

The paper went through two more revisions before it was published

- The project was put on hold for months at a time while we waited for reviews.
- Having the work divided into tasks and the provenance of all results documented made it easy to pick up the work where we had left off.

How many do-files?

1. The number of tasks and do-files depend on:
    a. How complicated are your analyses?
    b. How you like to work?
2. I prefer more tasks and shorter do-files. A single paper might have 100 do-files each with less than 100 commands. Some might be 1000 lines long.
3. Here is an example of the risk of long do-files.
    a. I was asked for advice on adjusting for clustered observations.
    b. The do-file I received was long, without comments, produced 163 pages of output, and did not reproduce the tables.
    c. I had evolved as new analyses were added and prior analyses were revised. Results were not posted.
    d. Later changes to the do-file affected earlier results.
    e. We had to start over.

Why short do files

1. If you correct an error in a do-file, are later commands affected?
    - You can accidentally change something or make a change earlier in the file that affects something later.
    - Do I check everything in the new log file?
    - Do I share the entire log file with collaborators who only the changes?
2. I find it easier to verify corrections in shorter do-files.
3. I prefer reviewing shorter log files, especially in collaborative work.
    - For me, checking too much at one time is error prone.
    - In collaborative work, lengthy logs waste time looking for the result that someone else is discussing. It works better to discuss smaller sets of results.
    - It makes it simpler to apply the posting principle.
4. When a draft is complete, I create a do-files that produce all of the results.
    - .This is convenient for revisions.
    - It gives me one more chance to find errors.

Objections to many short do-files

- It is difficult to keep track of the files and it is hard to rerun them.
- But, if you use naming conventions it is easy to keep track of your files.
- *Master do-files* make it easy to re-run all of your scripts.

Using master do-files

1. A master do-file contains `do` commands that run other do-files.
2. In a study of race differences in sexual well-being, I used ten do-files.
3. To rerun these files, I ran the master do-file `swb-all.do`:

```
capture log close master
log using swb-all, name(master) replace text

//  program:    swb-all.do
//  task:       swb | may 2007 analyses
//  project:    workflow - chapter 7
//  author:     jsl | 2007-03-08
//  note:       all programs required swb-00-loaddata.doi

// Task 01: descriptive statistics and data checking
do swb-01a-desc.do
do swb-01b-descmisc.do
do swb-01c-barchart.do

// Task 02: logit - sexual relationships
do swb-02aV2-srlogit.do
do swb-02b-srlogit-checkage.do
do swb-02c-srlogit-ageplot.do
```

```
// Task 03: logit - own sexuality
do swb-03a-os2logit.do
do swb-03b-os2Vos1logit.do

// Task 04: logit - self attractiveness
do swb-04a-salogit.do

// Task 05: logit - miscellaneous
do swb-05a-sr-os2-cor.do

log close master
exit
```

4. To rerun everything, I enter the command:

```
do swb-all
```

5. If I only want to re-run some of the do-files, I comment out the others:

```
capture log close master
log using swb-all, name(master) replace text

//  program:    swb-all.do
//  task:       swb | may 2007 analyses
//  project:    workflow - chapter 7
//  author:     jsl | 2008-03-07
//  note:       all programs include swb-00-loaddata.doi
```

```
/*
  // Task 01: descriptive statistics and data checking
  do swb-01a-desc.do
  do swb-01b-descmisc.do
  do swb-01c-barchart.do

  // Task 02: logit - sexual relationships
  do swb-02aV2-srlogit.do
  do swb-02b-srlogit-checkage.do
  do swb-02c-srlogit-ageplot.do
*/
// Task 03: logit - own sexuality
do swb-03a-os2logit.do
do swb-03b-os2Vos1logit.do

// Task 04: logit - self attractiveness
do swb-04a-salogit.do

// Task 05: logit - miscellaneous
do swb-05a-sr-os2-cor.do

log close master
exit
```

6. If you don't want a single log file, drop the three `log` commands.

Effectively using long do-files

Ross Stolzenberg at Chicago prefers a single do-file for each paper.

> Here attached is a do-file that is pretty much a complete project from inception to the date and time that the file was run. In case you wonder who wrote all this, I refer to myself by name, to prevent existential questions about the identity of "I" and "me." Generally, I try to identify in the code who wrote the code, if there is any ambiguity at all. I use a little subroutine at the beginning to generate names of log files. If you search for the word "argument" you will get to a set of comments that outline the model and its implementation. What looks like excessive computation in that section is really just my efforts to make sure that I don't stumble across convenient results that are somehow misleading. *I do everything 6 ways to Sunday, just to be sure. I know that you do the same, but it does seem to me that this degree of checking has gone out of style for more recent cohorts of researchers.* (e-mail 2011-03-21)

Dual workflow and data analysis

The problem

1. During analysis you will need additional variables.

2. It is dangerous to use variables that are not in the dataset

3. In the midst of analysis, it is distracting to return to data management.

Solution with dual do-file

1. Create a *dual do-file* that simply:
   - Loads the current dataset
   - Saves it as the *dual dataset*.

2. Analysis do-fles use the dual dataset.
   - If I need a new variable, I edit the dual do-file and run it. The dual dateset is no longer identical to the prior version (which is ok since it was not posted).
   - When I am done, I post the analysis files, the dual files, and dual dataset.

3. By having the dual script in place, it is easy to maintain the dual workflow.

Example of dual workflow during analysis

1st data run: Create `wflec-dual.dta` identical to the posted dataset

```
capture log close
log using wflec-analysis-dual-stat, replace text
version 13.1
clear all
macro drop _all
set linesize 80
set scheme s1manual

//  Reproducible results example: analysis using dual workflow
//  1st run - analysis with original variables

local pgm wflec-analysis-dual-stat
local who Scott Long
local dte 2017-06-19
local tag "`pgm'.do `who' `dte'"

//  #1 load current version of dataset

use wflec-dual
datasignature confirm
codebook, compact
```

```
//  #2 analysis using log of wages

logit lfp lwg k5 wc

//  #3 analysis using wages - which is not in use wf-lec

log close
exit
```

1st stat run: Unposted analysis program using `wflec-dual`

```
capture log close
log using wflec-analysis-dual-stat, replace text
version 13.1
clear all
macro drop _all
set linesize 80
set scheme s1manual

//  Reproducible results example: analysis using dual workflow
//  1st run - analysis with original variables

local pgm wflec-analysis-dual-stat
local who Scott Long
local dte 2017-06-19
local tag "`pgm'.do `who' `dte'"
```

```
//  #1 load current version of dataset

use wflec-dual
datasignature confirm
codebook, compact

//  #2 analysis using log of wages

logit lfp lwg k5 wc

//  #3 analysis using wages - which is not in use wf-lec

log close
exit
```

2nd data run: Add wages to `wflec-dual`

```
capture log close
log using wflec-analysis-dual-data, replace text
version 13.1
clear all
macro drop _all
set linesize 80
set scheme s1manual

//  Reproducible results example: analysis using dual workflow
//  2nd run : add wages
```

```
local pgm wflec-analysis-dual-data
local who Scott Long
local dte 2017-06-19
local tag "`pgm'.do `who' `dte'"

//  #1 load current version of dataset

use wf-lfp
datasignature confirm
codebook, compact

//  #2 create new variables here if they are needed

gen wages = log(lwg) if !missing(lwg)
label var wages "wages unlogged"
note wages: `tag'
sum lwg wages
scatter lwg wages
graph export `pgm'-wagecheck.png, replace

//  #3 save the version of the dataset used in analysis do-files

local savename "wflec-dual.dta"
compress
label data "dual of wf-lfp to use during analysis | `dte'"
note: `savename' / `tag' / updated dataset during data analysis
datasignature set, reset
```

```
save `savename', replace
notes
log close
exit
```

2nd stat run: analysis of wages using `wflec-dual`

```
capture log close
log using wflec-analysis-dual-stat, replace text
version 13.1
clear all
macro drop _all
set linesize 80
set scheme s1manual

//  Reproducible results example: analysis using dual workflow
//  2nd run - analysis with added variable

local pgm wflec-analysis-dual-stat
local who Scott Long
local dte 2017-06-19
local tag "`pgm'.do `who' `dte'"

//  #1 load current version of dataset

use wflec-dual
datasignature confirm
```

```
codebook, compact

//  #2 analysis using log of wages

logit lfp lwg k5 wc
estimates store lfplwg

//  #3 analysis using wages - which is not in wf-lfp

logit lfp wages k5 wc
estimates store lfpwages

estimates table lfplwg lfpwages

log close
exit
```

Dual workflow with multiple analyses

Data management		*Data analysis*	
`data01.do`	`=> cwh01.dta`	`desc01a.do`	`≪ uses cwh01.dta`
`data02.do`	`=> cwh02.dta`	`desc01b.do`	`≪ uses cwh02.dta`
		`desc02.do`	`≪ uses cwh02.dta`
		`compare01a.do`	`≪ uses cwh02.dta`
`data03.do`	`=> cwh03.dta`	`compare01a1.do`	`≪ uses cwh03.dta`
		`compare021.do`	`≪ uses cwh03.dta`
		`logit01V2.do`	`≪ uses cwh03.dta`
`data04.do`	`=> cwh04.dta`	`logit02.do`	`≪ uses cwh04.dta with no new variables.`
`data04.do`	`=> cwh04.dta`	`logit03.do`	`≪ uses cwh04.dta with new variable`
		`logit04.do`	`≪ uses cwh04.dta`

When ready to post, I finalized **`data04.do`** to create the official **`cwh04.dta`**. Then rerun **`logit02.do`** through **`logit03.do`** using the posted version of **`cwh04.dta`**.

Temporary variables

1. There are two ways you might reasonably add variables in an analysis file.
- Stata's factor syntax create variables "on the fly". This is safe, but causes problems when exporting data.
- Temporary variables disappear when the do-file ends. In general, a dual workflow makes such variables unnecessary.

Adding age-squared

My dataset includes age and I want to add age-squared. I can do this four ways

1. Factor syntax
2. Temporary variables
3. Create the variable in my analysis file
4. Use a dual workflow

Factor syntax

```
logit lfp lwg k5 i.wc c.age##c.age, nolog
```

tempvar

```
tempvar agesq
gen `agesq' = age*age
label var `agesq' "tempvar: age-squared"
logit lfp lwg k5 i.wc age `agesq', nolog
```

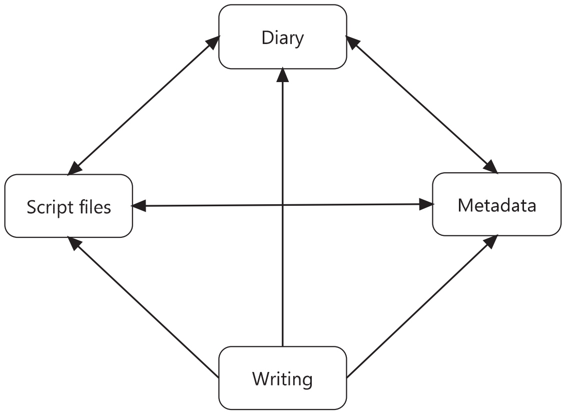
Violate dual workflow

```
gen agesq = age*age
label var agesq "bad idea! age-squared"
logit lfp lwg k5 i.wc age agesq, nolog
```

Use a dual workflow

You know how to do this!

Documentation of data analysis



Diary

Script files

Metadata

Writing

Project diary

Diary and do-files

1. Begin with an analysis plans and a list of tasks.
   - o In the plan, name each do-file and sketch what it will do.
   - o I sometimes create dummy do-files in my working directory.

2. *As I work, the diary becomes a dated record of the completed scripts.*

3. The diary only includes details on a do-file if there is a problem, analyses are complex or non-standard, or results are surprising.

Project diary and results

Second on results

1. I drafts finding (or add them as comments in the do-file)

2. I start outlining the paper.

Section on To do/Ideas

1. I keep a list of ideas for the current paper and future papers.

2. I include reactions and suggestions from collaborators.

Old example of analysis project diary

First complete set of analysis for FLIM measures paper

```
f2alt01a.do - 24May2002
  Descriptive information on all rhs, lhs, and flim measures
f2alt01b.do - 25May2002
  Compute bic’ for each of four outcomes and all flim measures.
   **  Outcome: Can Work                   global lhs "qcanwrk95"
   **  Outcome: Work in three categories   global lhs "dhlthwk95"
   **  Outcome: bath trouble               global lhs "bathdif95"
   **  Outcome: adlsum95 - sum of adls     global lhs "adlsum95"
f2alt01c.do - 25May2002
  Compute bic’ for each of four outcomes and with only these restricted flim
measures.
   *     1.  ln(x+.5) and ln(x+1)
   *     2.   9 counts: >=5=5  >=7=7  (50% and 75%)
   *     3.   8 counts: >=4=4  >=6=6  (50% and 75%)
   *     4.   18 counts: >=9=9 >=14=14  (50% and 75%)
   *     5.   probability splits at .5; these don't work well in prior tests
f2alt01d.do - 25May2002
  bic’ for all four outcomes in models that include all raw flim measures
(fla*p5; fll*p5);
  pairs of u/l measures; groups of LCA measures
f2alt01e.do - all LCA probabilities - 25May2002

:::
```

Newer style of project diary

First complete set of analysis for FLIM measures paper

```
2002-05-24: model flims as a count

f2alt01a.do – desc stats
f2alt01b.do – bic to compare models for four outcomes

2002-05-25

f2alt01c.do - model with restricted flim measures
f2alt01d.do – model with raw flim measures
f2alt01e.do – LCA of flims

:::
```

Automation in data analysis

Why automation?

Automation primarily affects efficiency and accuracy, not reproducibility.

Example

1. I use data about gender differences in the receipt of tenure for academic biochemists (Long, Allison, and McGinnis 1993).

2. The binary outcome indicates receipt of tenure with predictors such as gender, departmental prestige, time in rank, and research productivity.

3. Each person has multiple observations corresponding to each year they were in the rank of assistant professor.

Locals to define sets of variables

1. I could do this:

```
//  #2 desc statistics for men & women combined
codebook female tenure year yearsq select articles prestige, compact

//  #3 desc statistics for women
codebook female tenure year yearsq select articles prestige ///
    if female, compact

//  #4 desc statistics for men
codebook female tenure year yearsq select articles prestige ///
    if male, compact
```

2. If I decide to drop `yearsq`, I must remove it in three locations.

3. Instead use a local

```
local varset female tenure year yearsq select articles prestige

//  #5 desc statistics for men & women combined
codebook `varset', compact

//  #6 desc statistics for women
codebook `varset' if female, compact

//  #7 desc statistics for men
codebook `varset' if male, compact
```

Links among models

1. Locals make it easier to see the links among models and prevent errors in specification.

2. Suppose I want to estimate a series of nested logit models:

```
//  baseline gender only model
logit tenure female

//  time
logit tenure female year yearsq

//  department
logit tenure female year yearsq select prestige

//  productivity
logit tenure female year yearsq select prestige articles
```

---

3. Using locals is more efficient and less likely to create mistakes:

```
local Vtime "year yearsq"      // time in rank
local Vdept "select prestige"  // characteristics of departments
local Vprod "articles"         // research productivity
```

4. I estimate four models:

```
//  baseline gender only model
logit tenure female

//  + time
logit tenure female `Vtime'

//  + department
logit tenure female `Vtime' `Vdept'

//  + productivity
logit tenure female `Vtime' `Vdept' `Vprod'
```

5. To revise the model, I only have to change the locals:

```
local Vtime "year yearsq yearcu" // time in rank
local Vprod "articles citations" // research productivity
```

6. Since the `logit` commands are unchanged, I know the models are correctly specified.

Nested models with a loop

Method 1

```
local Vtime "year yearsq"      // time in rank
local Vdept "select prestige"  // characteristics of departments
local Vprod "articles"         // research productivity

local model1 female
local model2 `model1' `Vtime'
local model3 `model2' `Vdept'
local model4 `model3' `Vprod'

foreach number in 1 2 3 4 {
    logit tenure `model`number''  // N.B. '' not '
}
```

Method 2

```
foreach varset in Vfem Vtime Vdept Vprod {
    local rhsvars "`rhsvars' ``varset''" // N.B. `` and ''
    logit tenure `rhsvars'
}
```

Computing t-tests with loops

1. I want to test gender differences in all of the variables (*wf7-loops-ttest.do*):

```
. ttest tenure, by(female)

Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
  0_Male |    1741    .1315336    .0081025    .3380801    .1156419    .1474253
1_Female |    1056    .1089015    .0095908    .3116632    .0900824    .1277207
---------+--------------------------------------------------------------------
combined |    2797    .1229889    .0062111    .3284832    .1108102    .1351677
---------+--------------------------------------------------------------------
    diff |            .0226321    .0128075               -.002481    .0477451
------------------------------------------------------------------------------
    diff = mean(0_Male) - mean(1_Female)                          t =   1.7671
Ho: diff = 0                                     degrees of freedom =     2795

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.9613         Pr(|T| > |t|) = 0.0773          Pr(T > t) = 0.0387
```

---

2. For the other variables, I could run the commands:

```
ttest year,     by(female)
ttest select,   by(female)
ttest articles, by(female)
ttest prestige, by(female)
```

3. With a loop:

```
local varlist "tenure year select articles prestige"
foreach var in `varlist' {
    ttest `var', by(female)
}
```

4. The `ttest` command isn't echoed so the variable isn't indicated.

```
Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
  0_Male |    1741    .1315336    .0081025    .3380801    .1156419    .1474253
1_Female |    1056    .1089015    .0095908    .3116632    .0900824    .1277207
combined |    2797    .1229889    .0062111    .3284832    .1108102    .1351677
    diff |            .0226321    .0128075               -.002481    .0477451
------------------------------------------------------------------------------
    diff = mean(0_Male) - mean(1_Female)                          t =   1.7671
Ho: diff = 0                                     degrees of freedom =     2795

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.9613         Pr(|T| > |t|) = 0.0773          Pr(T > t) = 0.0387
```

---

5. The solution is:

```
foreach var in `varlist' {
    di _new ". ttest `var', by(female)"
    ttest `var', by(female)
}
```

6. For example, the first time through the loop:

```
. ttest tenure, by(female)

Two-sample t test with equal variances
------------------------------------------------------------------------------
  Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
-------+----------------------------------------------------------------------
```

(output omitted)

Loops for alternative model specifications

1. I want to evaluate alternative transformations of articles in a logit model.

2. I create 9 variables and their names:

```
local artvars ""

forvalues root = 1(1)9 {

    gen art_root`root' = articles^(1/`root')
    label var art_root`root' "articles^(1/`root')"

    * accumulate list of variable names
    local artvars "`artvars' art_root`root'"

}
```

3. Now I estimate models:

```
foreach avar in `artvars' {
    di _new "== logit with `avar'"
    logit tenure `avar' female year yearsq select prestige
}
```

Advanced: Collecting statistics

1. You often compute 100s of statistics, but only want to report a few.

2. Extracting the numbers is tedious and error prone.

3. Automation can help. You invest time, you gain time and accuracy.

Collecting t-tests

1. I want a table of t-tests that looks like this:

	Women		Men		Test of equal means	
Variable	Mean	Std. Dev.	Mean	Std. Dev.	*t*-test	Prob
Tenure						
Year						
Selectivity						
Articles						
Prestige						

The table is the plan that guides what follows.

---

2. `ttest` returns the information I need:

```
. ttest tenure, by(female)

Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
  0_Male |    1741    .1315336    .0081025    .3380801    .1156419    .1474253
1_Female |    1056    .1089015    .0095908    .3116632    .0900824    .1277207
---------+--------------------------------------------------------------------
combined |    2797    .1229889    .0062111    .3284832    .1108102    .1351677
---------+--------------------------------------------------------------------
    diff |            .0226321    .0128075               -.002481    .0477451
------------------------------------------------------------------------------
    diff = mean(0_Male) - mean(1_Female)                          t =   1.7671
Ho: diff = 0                                     degrees of freedom =     2795

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.9613         Pr(|T| > |t|) = 0.0773          Pr(T > t) = 0.0387
```

3. The returns are:

```
  . return list

  scalars:
                    r(sd) =  .3284832119751412
                  r(sd_2) =  .3116632125613366
                  r(sd_1) =  .3380801147013905
                    r(se) =  .0128074679461748
```

```
                   r(p_u) =  .0386602339087719
                   r(p_l) =  .9613397660912281
                     r(p) =  .0773204678175438
                     r(t) =  1.767100751070975
                  r(df_t) =  2795
                  r(mu_2) =  .1089015151515152
                   r(N_2) =  1056
                  r(mu_1) =  .1315336013785181
                   r(N_1) =  1741
```

4. I combine the returns from multiple t-tests in the matrix `tmatrix`:

```
capture matrix drop tmatrix

local varlist "tenure year select articles prestige"

local ncols = 6

matrix vecorig = J(1,`ncols',-999)

matrix colnames vecorig = Wmn Wsd Mmn Msd ttest Prob
```

Next, a loop to populate the matrix...

---

5. I loop through the t-tests and save the results to the matrix:

```
foreach var in `varlist' {

    qui ttest `var', by(female)

    mat myvec = vecorig

    mat myvec[1,1] = r(mu_2)
    mat myvec[1,2] = r(sd_2)
    mat myvec[1,3] = r(mu_1)
    mat myvec[1,4] = r(sd_1)
    mat myvec[1,5] = r(t)
    mat myvec[1,6] = r(p)

    mat rownames myvec = `var'
    mat tmatrix = nullmat(tmatrix) \ myvec

}
```

---

6. I print the matrix with formatting

```
 . local header "t-tests: men compared to women"
 . matlist tmatrix, title(`header') format(%8.3f)

t-tests: men compared to women

           |      Wmn       Wsd       Mmn       Msd     ttest      Prob
-----------+------------------------------------------------------------
    tenure |    0.109     0.312     0.132     0.338     1.767     0.077
      year |    3.974     2.380     3.784     2.252    -2.121     0.034
    select |    5.001     1.475     4.992     1.365    -0.170     0.865
  articles |    7.415     7.430     6.829     5.990    -2.284     0.022
  prestige |    2.658     0.765     2.640     0.784    -0.612     0.540
```

7. The matrix can be exported as a csv file using `svmat` and `export excel`.

8. Run `search frmttable` to find a command to export matrices to LaTeX with formatting; `search estout` for another option.

9. The `putexcel` command will write Excel spreadsheets.

Collecting nested regressions

1. Load the data and specify sets of variables:

```
. //  #1 load data and select sample

. use wf-tenure, clear
(Workflow data for gender differences in tenure | 2008-04-02)
. datasignature confirm
  (data unchanged since 02apr2008 13:29)

. keep if sampleis
(148 observations deleted)

. //  #2 define groups of variables
.
. local Vbase female
. local Vtime year yearsq      // time in rank
. local Vdept select prestige  // characteristics of departments
. local Vprod articles         // research productivity
```

---

2. Set up a matrix and locals to use in a loop:

```
capture matrix drop matnest
local rowlist Vbase Vtime Vdept Vprod
local ncols = 4
matrix vecorig = J(1,`ncols',-999)
matrix colnames vecorig = ORfem zfem pfem

local rhs "" // add names of rhs variables below
```

3. Loop through models and collect what I want:

```
foreach mdl in Vbase Vtime Vdept Vprod {
    mat myvec = vecorig

    * estimate model
    local rhs "`rhs' ``mdl''"
    qui logit tenure `rhs'

    * get stats and compute pvalue
    local b = _b[female] // retrieve b for female
    local or = exp(`b') // compute odds ratio
    local se = _se[female] // retrieve std error for b
    local z = `b'/`se' // z-value
    local pval = 2*(1-normal(abs([`z']))) // pvalue
```

```
    * save results
    mat myvec[1,1] = `or'
    mat myvec[1,2] = `z'
    mat myvec[1,3] = `pval'
    mat rownames myvec = `mdl' // label row
    mat matnest = nullmat(matnest) \ myvec // stack vector

}
```

4. Here is the table:

```
. //  #4 print results
.
. matlist matnest, format(%8.3f)

             |    ORfem      zfem      pfem
-------------+------------------------------
       Vbase |    0.807    -1.764     0.078
       Vtime |    0.723    -2.511     0.012
       Vdept |    0.721    -2.520     0.012
       Vprod |    0.702    -2.678     0.007
```

Summary

1. Analysis is the exciting part for most of us.

2. A fully documented dataset makes analysis easier.

3. Posting, dual workflow, and run order naming make reproducibility easy.

4. Additional benefits are seen when we present our results and need to document the provenance of every result.

Part 19: Provenance & presentations

WFDAUS pages 318-327.

Overview

1. Document the provenance of every result you present.
  - If you don't know where a number came from, how can you reproduce it?

2. Documenting provenance makes revisions much easier.

3. Effective presentations are more convincing.

4. Things go wrong in presentations which you need to anticipate

---

Documenting provenance

1. For reproducibility, you must document the *provenance of every result*.
  - The woeful tale of the 18 month delay in a dissertation

2. It is easier to add a result to a paper than to find where it came from.
  - Rounding makes searches unreliable. `.1234567` rounds to `.1235`. Searching `.1235` will not find the source in your log files.

3. Documenting provenance makes revisions more efficient.

Method 1: Not recommended

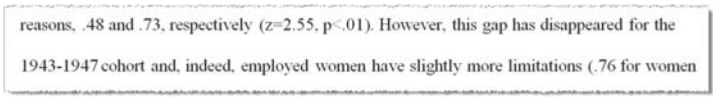
  - Annotate printed or PDF output.
  - Finding results is possible but tedious.

Method 2: Not recommended

  - In your diary, record which results were obtained from which log files.
  - Tedious and error prone.

Method 3: My preference in Word

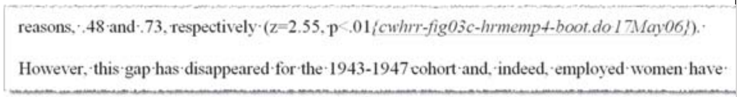
1. This is an extract from Pavalko, Gong and Long (2007). The results came from **`cwhrr-fig03c-hrmemp4.do.`** How do I know?



reasons, .48 and .73, respectively (z=2.55, p<.01). However, this gap has disappeared for the

1943-1947 cohort and, indeed, employed women have slightly more limitations (.76 for women

2. In Word, I turn on the option to make hidden text visible:

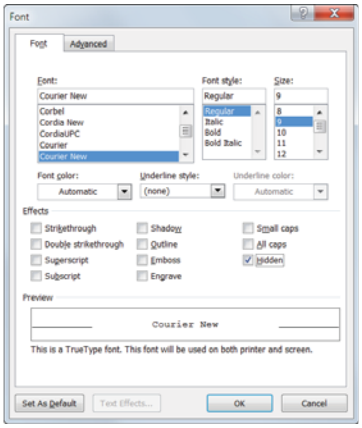


reasons,·.48·and·.73,·respectively·(z=2.55,·p<.01*{cwhrr-fig03c-hrmemp4-boot.do·17May06}*).·

However,·this·gap·has·disappeared·for·the·1943-1947·cohort·and,·indeed,·employed·women·have·

3. Demonstration of searching for do-file.

Software: Word and hidden text



☐ Strikethrough ☐ Shadow ☐ Small caps
☐ Double strikethrough ☐ Outline ☐ All caps
☐ Superscript ☐ Emboss ☑ Hidden
☐ Subscript ☐ Engrave

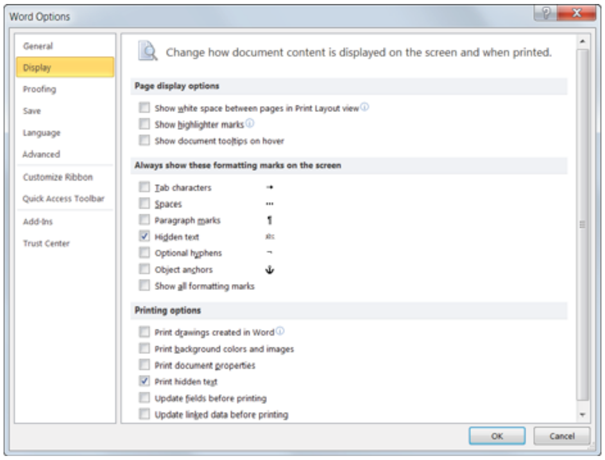
This is a TrueType font. This font will be used on both printer and screen.

1. Right clicking and selecting font:

2. Better yet, create a keystroke, say **ctrl-alt-h** to make it hidden which shows up with a faint, dotted underline.

---

3. To view or print hidden text and other things:



Change how document content is displayed on the screen and when printed.

Page display options

☐ Show white space between pages in Print Layout view
☐ Show highlighter marks
☐ Show document tooltips on hover

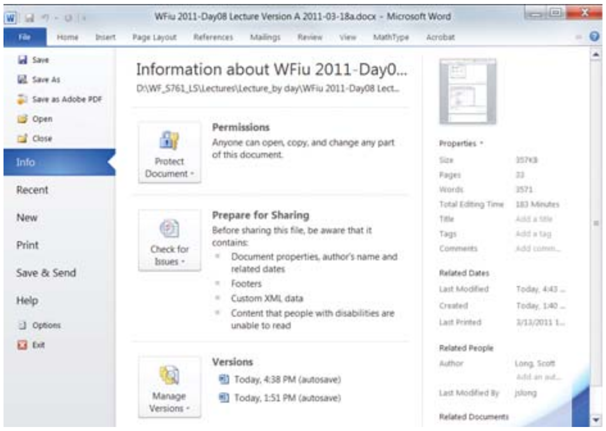
Always show these formatting marks on the screen

☐ Tab characters →
☐ Spaces ···
☐ Paragraph marks ¶
☑ Hidden text abc
☐ Optional hyphens ¬
☐ Object anchors ⚓
☐ Show all formatting marks

Printing options

☐ Print drawings created in Word
☐ Print background colors and images
☐ Print document properties
☑ Print hidden text
☐ Update fields before printing
☐ Update linked data before printing

Aside: Hidden text you might regret



Information about WFiu 2011-Day0...

D:\WF_5761_LS\Lectures\Lecture_by day\WFiu 2011-Day08 Lect...

Permissions

Anyone can open, copy, and change any part of this document.

Prepare for Sharing

Before sharing this file, be aware that it contains:

- Document properties, author's name and related dates
- Footers
- Custom XML data
- Content that people with disabilities are unable to read

Versions

Today, 4:38 PM (autosave)
Today, 1:51 PM (autosave)

Size 357KB
Pages 33
Words 3571
Total Editing Time 183 Minutes
Title Add a title
Tags Add a tag
Comments Add comm...

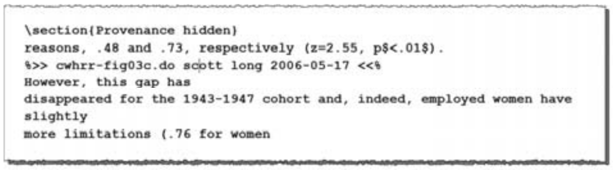
Related Dates
Last Modified Today, 4:43 ...
Created Today, 1:40 ...
Last Printed 3/13/2011 1...

Related People
Author Long, Scott
Add an aut...
Last Modified By jslong

Related Documents

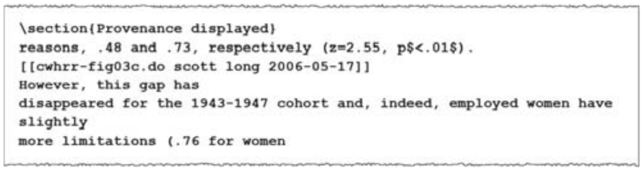
Method 3: My preference in LaTeX

1. Add comments to your TeX file that are not be shown in formatted output.



```
\section{Provenance hidden}
reasons, .48 and .73, respectively (z=2.55, p$<.01$).
%>> cwhrr-fig03c.do scott long 2006-05-17 <<%
However, this gap has
disappeared for the 1943-1947 cohort and, indeed, employed women have
slightly
more limitations (.76 for women
```

2. To make provenance visible, change **%>>** to **[[** and **<<%** to **]]**.



```
\section{Provenance displayed}
reasons, .48 and .73, respectively (z=2.55, p$<.01$).
[[cwhrr-fig03c.do scott long 2006-05-17]]
However, this gap has
disappeared for the 1943-1947 cohort and, indeed, employed women have
slightly
more limitations (.76 for women
```

Provenance in graphs

1. When you create graphs, use a caption to indicate provenance.

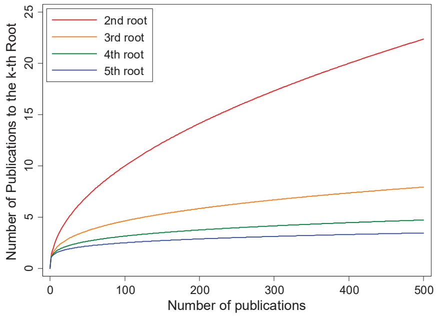
2. If you don't, you have no way to know where the graph came from unless you know the file name (which is hard when embedded in Word).

3. Before a paper is submitted or accepted, you can create graphs that don't show this information.

4. For example…

Graph captions

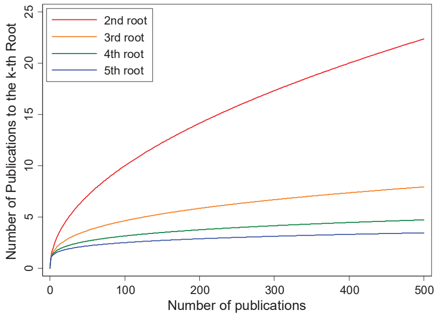
```
twoway (line art_root2 art_root3 art_root4 art_root5 articles, ///
    lwid(medium) lcol(red orange green blue)), ///
    ytitle(Number of Publications to the k-th Root) ///
    yscale(range(0 8.)) legend(pos(11) rows(4) ring(0)) ///
    caption(`pgm'-with.emf | `tag', size(vsmall))
```



wflec-present-provenance-caption-with.emf | wflec-present-provenance-caption.do Scott Long 2017-06-20

---

5. To hide the caption, crop the image



6. Or create a version without the caption.

Provenance and revisions

1. Documents should include metadata with the provenance of *every* number you report and every data driven conclusion you draw.
2. Graphs should indicate their source or the text should have metadata indicating the source of the graph.
3. You can strip the information from the file when you submit a paper.
   - Word has options to remove hidden text.
   - A good editor makes it simple to remove provenance in LaTeX files if you enter it systematically.
4. You can print your document to a PDF file without showing provenace for drafts you want to distribute.

Let's consider how I make revisions...

Revisions using hidden text

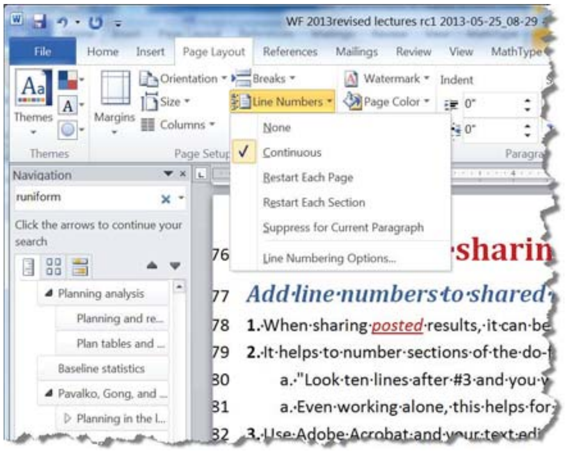
1. Print the paper with the hidden text shown.
2. Highlight results that might change and the do-files used.
3. Put the names of these do-files in a new master do file.
4. Copy do-files to change to files with same name but different version (e.g., V2).
5. Revise the V2 do-files run the new master do-file.
6. Revise the paper.

Sharing results for feedback

Add line numbers to shared results

1. When sharing *posted* results, it can be confusing to talk about specific results.
2. Use location comments (e.g., `// #3 logit on lfp`) to make it easy to refer to and find results.
   - "Look ten lines after #3 and you will see ...."
3. Your editor or word processor lets you add line numbers to the document.
   - Then print to a PDF for distribution.

In Word...



76 sharin

77 *Add line numbers to shared*

78 **1.** When sharing *posted* results, it can be

79 **2.** It helps to number sections of the do-f

80 a. "Look ten lines after #3 and you v

81 a. Even working alone, this helps for

82 **3.** Use Adobe Acrobat and your text edi

Editing before sharing

1. You might want to edit a log before sharing it.
2. For example,
   - Copy **`mywork01.log`** to **`mywork01.logedited`**
   - Revise **`mywork01.logedited`** and share it as a PDF.

Effective tables & graphs

King's *Publication, Publication* gives excellent:

> Referees are busy people looking for a way to finish the thankless (anonymous) task of reviewing your paper as quickly as possible. Since you're not likely to have as much time from them as you think, you need to make reading your paper as easy as possible. And in this game a tie doesn't go to the runner: *If a referee didn't read carefully, pay attention, or understand you, or missed or misunderstood something important in your paper, it is your fault.* And since it is your paper and not you that matters, anonymous referees will not (and for the sake of the literature normally should not) give an anonymous paper writer the benefit of the doubt. Anonymous referees are not normally prone to spontaneous generosity and do not generally impute favorable motives to authors who are not clear or impute appropriate assumptions when you leave them unstated.

When Scott Long reviews a paper with careless presentation, he assumes the author was equally careless in data management and analysis.

General principles for tables & graphs

1. Effective tables and graphs takes careful planning and execution.
2. Outsiders must understand the tables and figures (avoid tacit knowledge).
   - Tables and graphs should be self-contained.
3. Aesthetically pleasing graphs and tables are more convincing.
4. Consistently use of significant digits and avoid silly content (b=0.00, p=0.00).
5. Don't undermine your good work when you present it.

Use templates and exemplars

1. Following standard convention makes it easier for other to understand.
2. Study journals and books in your field for find excellent examples of presentations.
   - Save these in \Templates\Papers
3. Sources on presentations
   - Wong, D. M. (2010). The Wall Street journal guide to information graphics: the dos and don'ts of presenting data, facts, and figures. Norton & Co.
   - Publication Manual of the American Psychological Association.
   - The Chicago Manual of Style.

Presenting

Try it *before* you present

1. Try your presentation *every way* you might present it.
2. The same graph or table is *not* effective in *all* media.
   - How it looks on your monitor is not how it looks projected or on paper.
3. Never trust someone who tells your presentation file work on their system.
   - It doesn't help you if someone says, sorry your file didn't work. I thought it would be OK.
   - Have a PDF version of your talk that you carry with you and have available on the internet.

Have you ever heard

- Unfortunately, the colors look the same on the screen, but the bar on the left is red and the one on the right is blue.
- This graph is in your handout, although it is hard to tell the lines apart since they are in black and white instead of color.
- You probably can't read the numbers, so I'll tell you what they are.

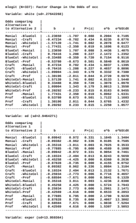
Have you ever seen

- A projector that doesn't work correctly
- The clicker/pointer doesn't work
- Software on the presentation computer is incompatible with presenter's file
- Files for presentation can't be found

If it isn't your fault

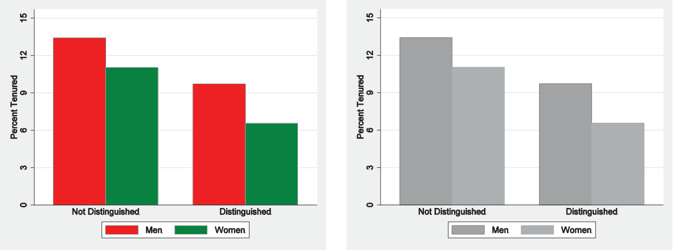
- It is your presentation

Tables that do not work

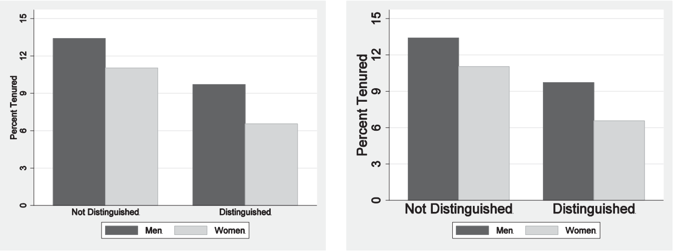


```
mlogit (N=337): Factor Change in the Odds of occ

Variable: white (sd=.27642268)

Odds comparing    |
Alternative 1     |
to Alternative 2  |      b         z     P>|z|     e^b   e^bStdX
------------------+---------------------------------------------
Menial  -BlueCol  | -1.23650   -1.707   0.088   0.2904   0.7105
Menial  -Craft    | -0.47234   -0.782   0.434   0.6235   0.8776
Menial  -WhiteCol | -1.57139   -1.741   0.082   0.2078   0.6477
Menial  -Prof     | -1.77431   -2.350   0.019   0.1696   0.6123
BlueCol -Menial   |  1.23650    1.707   0.088   3.4436   1.4075
BlueCol -Craft    |  0.76416    1.208   0.227   2.1472   1.2352
BlueCol -WhiteCol | -0.33488   -0.359   0.720   0.7154   0.9116
BlueCol -Prof     | -0.53780   -0.673   0.501   0.5840   0.8619
Craft   -Menial   |  0.47234    0.782   0.434   1.6037   1.1395
Craft   -BlueCol  | -0.76416   -1.208   0.227   0.4657   0.8096
Craft   -WhiteCol | -1.09904   -1.343   0.179   0.3332   0.7380
Craft   -Prof     | -1.30196   -2.011   0.044   0.2720   0.6978
WhiteCol-Menial   |  1.57139    1.741   0.082   4.8133   1.5440
WhiteCol-BlueCol  |  0.33488    0.359   0.720   1.3978   1.0970
WhiteCol-Craft    |  1.09904    1.343   0.179   3.0013   1.3550
WhiteCol-Prof     | -0.20292   -0.233   0.815   0.8163   0.9455
Prof    -Menial   |  1.77431    2.350   0.019   5.8962   1.6331
Prof    -BlueCol  |  0.53780    0.673   0.501   1.7122   1.1603
Prof    -Craft    |  1.30196    2.011   0.044   3.6765   1.4332
Prof    -WhiteCol |  0.20292    0.233   0.815   1.2250   1.0577
----------------------------------------------------------------

Variable: ed (sd=2.9464271)

Odds comparing    |
Alternative 1     |
to Alternative 2  |      b         z     P>|z|     e^b   e^bStdX
------------------+---------------------------------------------
Menial  -BlueCol  |  0.09942    0.972   0.331   1.1045   1.3404
Menial  -Craft    | -0.09382   -0.962   0.336   0.9105   0.7585
Menial  -WhiteCol | -0.35316   -3.011   0.003   0.7025   0.3533
Menial  -Prof     | -0.77885   -6.795   0.000   0.4589   0.1008
BlueCol -Menial   | -0.09942   -0.972   0.331   0.9054   0.7461
BlueCol -Craft    | -0.19324   -2.494   0.013   0.8243   0.5659
BlueCol -WhiteCol | -0.45258   -4.425   0.000   0.6360   0.2636
BlueCol -Prof     | -0.87828   -8.735   0.000   0.4155   0.0752
Craft   -Menial   |  0.09382    0.962   0.336   1.0984   1.3184
Craft   -BlueCol  |  0.19324    2.494   0.013   1.2132   1.7671
Craft   -WhiteCol | -0.25934   -2.773   0.006   0.7716   0.4657
Craft   -Prof     | -0.68504   -7.671   0.000   0.5041   0.1329
WhiteCol-Menial   |  0.35316    3.011   0.003   1.4236   2.8308
WhiteCol-BlueCol  |  0.45258    4.425   0.000   1.5724   3.7943
WhiteCol-Craft    |  0.25934    2.773   0.006   1.2961   2.1471
WhiteCol-Prof     | -0.42569   -4.616   0.000   0.6533   0.2853
Prof    -Menial   |  0.77885    6.795   0.000   2.1790   9.9228
Prof    -BlueCol  |  0.87828    8.735   0.000   2.4067  13.3002
Prof    -Craft    |  0.68504    7.671   0.000   1.9838   7.5264
Prof    -WhiteCol |  0.42569    4.616   0.000   1.5307   3.5053
----------------------------------------------------------------

Variable: exper (sd=13.959364)
```

Colors in BW



Labels that are too small



```
graph bar (mean) Mbg (mean) Wbg, over(Vbg, label(labsize(vlarge))) ///
    legend(label(1 Men) label(2 Women)) ///
    ytitle("Percent Tenured", size(vlarge)) ///
   ylab(0(3)15, labsize(large)) legend(label(1 Men) label(2 Women)) ///
   bar(1,fcolor(gs4)) bar(2,fcolor(gs13)) legend(size(vlarge))
```

A recent example: as intended

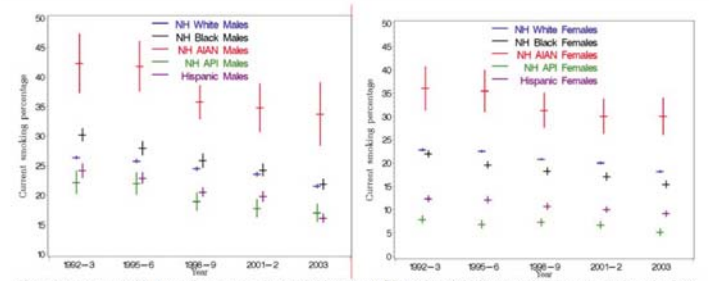


Figure 2. Current smoking prevalence rate estimates (with 95% confidence intervals) by race/ethnicity, year, and gender from TUS-CPS for the period 1992 to 2003

Davis et al. 2007 color.png

A recent example: as seen

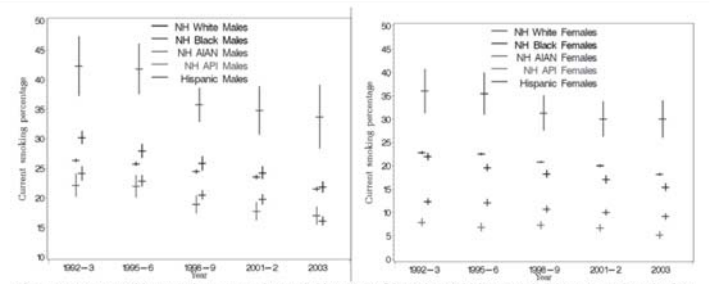
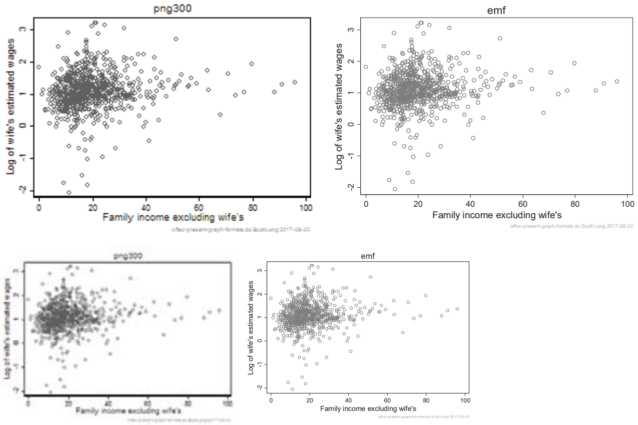


Figure 2. Current smoking prevalence rate estimates (with 95% confidence intervals) by race/ethnicity, year, and gender from TUS-CPS for the period 1992 to 2003

Davis et al. 2007 BW.png

pngs that don't scale



Tips for papers and presentations

When circulating drafts of papers

1. Cover page
   - title, *date*, authors, file name, and *who controls the document*.
2. *Make it clear which file is the current version*.
   - Name papers: groupdiff-2016-01-31.docx, not groupdiff-v13.docx
3. If you send a docx or tex file, name the file to indicate who controls the file for future edits.
   - groupdiff-2016-01-31 mustillo has control.docx
4. When asking for comments, add line numbers to the paper.
   - ASR now sends out papers to review with line numbers added.

Presentations

1. Talks are produced with Word, LaTeX, PowerPoint, Keynote, or Beamer.
   - Unless you have time to test, save your presentation in multiple formats.
   - PowerPoint 2007 might not work with PowerPoint 2003.
   - PDF files should work on almost any computer you use.
2. Most presentations use projectors.
3. Aspect ratios vary of output devices vary possibly distorting graphs

Monitor 1080p	1920	x	1080	Ratio: 1.78
Projectors	1024	x	768	Ratio: 1.33

4. Have redundant copies
   - On the web
   - In the cloud
   - On a USB stick; get a “clicker” that has a USB drive in it
   - E-mail it to yourself

---

5. Tufte (2006) hates PowerPoint:

> Imagine a widely used and expensive prescription *drug that claimed to make us beautiful* but didn't. Instead the drug had *frequent, serious side effects: making us stupid*, degrading the quality and credibility of our communication, turning us into bores, wasting our colleagues' time. The side effects, and the resulting unsatisfactory cost/benefit ratio, would rightly lead to a worldwide product recall.

He suggested using a word processor to create talks, not a presentation program. PowerPoint makes it easy to do "bad things". He sells a poster of Stalin addressing the masses with PowerPoint.

6. Peter Norvig, Director of Research at Google, makes the same points more humorously: *The Gettysburg PowerPoint Presentation* (www.norvig.com/Gettysburg/).

---

7. Tantau's (2007) *User's Guide to the Beamer Class* (Ch 5) has useful suggestions:
   - Fit the time you have.
   - Never use a smaller font to fit more on a page.
   - Do not include things that you will not discuss.
   - Use colors carefully, maximize contrast, and avoid shaded backgrounds.
   - Test your presentation.
   - For small tables or intricate graphs, distribute paper copies.
8. Wong, Dona M. 2010. *The Wall Street Journal Guide to Information Graphics: The Dos and Don'ts of Presenting Data, Facts, and Figures*. Reasonably priced and very effective.

Making tables

1. A command's output does not match the format of your table.
   - Never use computer output for your table!
2. Creating tables by hand is error prone and slow; better approaches take time to learn but can save time.

Save results to csv files

1. Use `putexcel` to create Excel files that you can edit.
2. Or, paste output into Excel and use text to columns.

Table making programs in Stata

1. Ben Jann's `esttab` and John Gallup's `outreg` are great.
2. Both take an investment to use effectively.

Save results to matrices

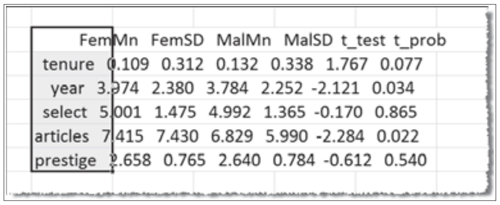
1. Use Gallup's `frmttable` or Jann’s `estout` to save results to latex or rtf.
2. This requires Stata automation, but is very effective.

Stata to Excel: The old way

1. A spreadsheet allows you to:
   a. Compute derivative statistics
   b. Revise the format (e.g., # of digits).
2. I often keep tables in a spreadsheet till the first circulation draft is written. Then, clean them up in Word.
3. Here is how to move a "matrix" into Excel.
4. List the matrix:

```
. matrix list stats, format(%9.3f)

stats[5,6]
             FemMn    FemSD    MalMn    MalSD  t_test  t_prob
  tenure     0.109    0.312    0.132    0.338   1.767   0.077
    year     3.974    2.380    3.784    2.252  -2.121   0.034
  select     5.001    1.475    4.992    1.365  -0.170   0.865
articles     7.415    7.430    6.829    5.990  -2.284   0.022
prestige     2.658    0.765    2.640    0.784  -0.612   0.540
```

5. Select the table in the editor, copy it, and paste into Excel:

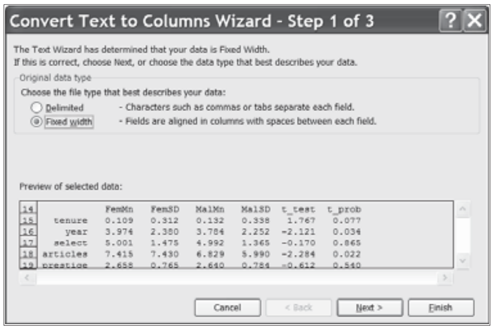


	FemMn	FemSD	MalMn	MalSD	t_test	t_prob
tenure	0.109	0.312	0.132	0.338	1.767	0.077
year	3.974	2.380	3.784	2.252	-2.121	0.034
select	5.001	1.475	4.992	1.365	-0.170	0.865
articles	7.415	7.430	6.829	5.990	-2.284	0.022
prestige	2.658	0.765	2.640	0.784	-0.612	0.540

6. Each cell within the box on the left holds the entire row of text and numbers shown to the right. We want to split this information into multiple columns.

---

7. To convert the pasted text into numbers within cells, the *Convert Text to Columns Wizard* (how you invoke this depends on the version of Excel you are using):



Convert Text to Columns Wizard - Step 1 of 3

The Text Wizard has determined that your data is Fixed Width.
If this is correct, choose Next, or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

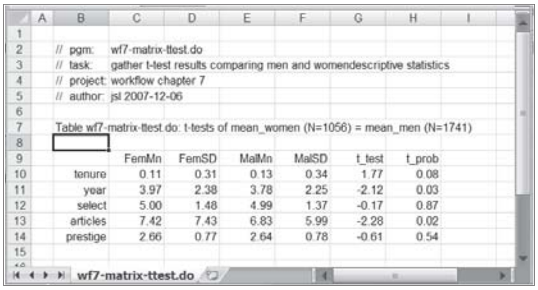
○ Delimited - Characters such as commas or tabs separate each field.
● Fixed width - Fields are aligned in columns with spaces between each field.

Preview of selected data:

```
14           FemMn   FemSD   MalMn   MalSD  t_test  t_prob
15   tenure  0.109   0.312   0.132   0.338   1.767   0.077
16     year  3.974   2.380   3.784   2.252  -2.121   0.034
17   select  5.001   1.475   4.992   1.365  -0.170   0.865
18 articles  7.415   7.430   6.829   5.990  -2.284   0.022
19 prestige  2.658   0.765   2.640   0.784  -0.612   0.540
```

---

8. Edit the table in Excel. For example, notice the provenance:



// pgm: wf7-matrix-ttest.do
// task: gather t-test results comparing men and womendescriptive statistics
// project: workflow chapter 7
// author: jsl 2007-12-06

Table wf7-matrix-ttest.do: t-tests of mean_women (N=1056) = mean_men (N=1741)

	FemMn	FemSD	MalMn	MalSD	t_test	t_prob
tenure	0.11	0.31	0.13	0.34	1.77	0.08
year	3.97	2.38	3.78	2.25	-2.12	0.03
select	5.00	1.48	4.99	1.37	-0.17	0.87
articles	7.42	7.43	6.83	5.99	-2.28	0.02
prestige	2.66	0.77	2.64	0.78	-0.61	0.54

Stata to Excel: the new way

Stata **`putexcel`** moves information from Stata directly into an Excel worksheet.

http://blog.stata.com/2013/09/25/export-tables-to-excel/

Regression tables with `esttab`

1. Stata's **`estimates table`** creates tables but is very limited.

2. Jann's **`estout`** is powerful but hard to learn. **`esttab`** is easier.

- o Gallup's **`outreg`** is easier to use but less powerful

Create a table for the nested regressions with esttab

```
//  #3a - baseline gender only model
logit tenure female, nolog or
eststo

//  #3b + time
logit tenure female `Vtime', nolog or
eststo

//  #3c + department
logit tenure female `Vtime' `Vdept', nolog or
eststo
```

---

3. Using the default options for esttab, I easily create a basic table:

```
. esttab
                      (1)             (2)             (3)
                   tenure          tenure          tenure
------------------------------------------------------------
female             -0.215          -0.324*         -0.327*
                  (-1.76)         (-2.51)         (-2.52)

year                                1.805***        1.818***
                                  (11.21)         (11.23)

yearsq                             -0.129***       -0.130***
                                  (-9.35)         (-9.35)

select                                              0.141**
                                                   (3.12)

prestige                                           -0.262**
                                                  (-3.15)

_cons              -1.887***       -6.927***       -7.002***
                 (-26.62)        (-15.59)        (-13.30)
------------------------------------------------------------
N                    2797            2797            2797
------------------------------------------------------------
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

---

4. With a few simple options, I can fine tune the format:

```
. esttab, eform nostar bic label varwidth(33) ///
>     title("Table 7.1: WF Example of Jann's esttab Command.") ///
>     mtitles("Model A" "Model B" "Model C") ///
>     addnote("Source: wflec-present-tables-esttab.do")

Table 7.1: Workflow Example of Jann's esttab Command.
------------------------------------------------------------------------
                                           (1)          (2)          (3)
                                       Model A      Model B      Model C
------------------------------------------------------------------------
Scientist is female?                     0.807        0.723        0.721
                                       (-1.76)      (-2.51)      (-2.52)

Years in rank.                                        6.079        6.161
                                                    (11.21)      (11.23)
<snip>
Baccalaureate selectivity.                                         1.151
                                                                  (3.12)

Prestige of department.                                            0.770
                                                                 (-3.15)
------------------------------------------------------------------------
Observations                              2797         2797         2797
BIC                                     2098.4       1768.7       1767.4
------------------------------------------------------------------------
Exponentiated coefficients; t statistics in parentheses
Source: wflec-present-tables-esttab.do
```

5. You can save the table in a format to read into a spreadsheet or a word processor:

```
esttab using wf7-estout.tex, eform nostar bic label ///
    varwidth(33) mtitles("Model A" "Model B" "Model C") ///
    addnote("Source: wflec-present-tables-esttab.do")
```

6. The resulting table looks like this:

	Model A	Model B	Model C
Scientist is female?	0.807	0.723	0.721
	(-1.76)	(-2.51)	(-2.52)
Years in rank.		6.079	6.161
		(11.21)	(11.23)
Years in rank squared.		0.879	0.878
		(-9.35)	(-9.35)
Baccalaureate selectivity.			1.151
			(3.12)

7. See Jann's web site (Google Jann estout).

Graphs in Stata

graph export

1. Stata's `graph export` converts from Stata gph to other formats.

   graph export *newfilename.suffix* [**, replace width()**]

   See `help export` for details.

2. Key formats:

.eps	Encapsulated postscript; for publishers and Tex/LaTex
.emf	Windows Enhanced Metafile (.wmf is obsolete)
.pdf	PDF graph format for Tex/LaTex or OS X
.png	PNG (Portable Network Graphics) *Can print terribly!*
.tif	TIFF if you are editing the graph

Suggested formats

1. With LaTex, use eps, pdf, or tif. If you plan to publish a book, create eps files along with other formats.
2. For Word, I use emf since it *scales* (it prints fine if larger or smaller).
3. With PNG you need to set width to match the output device. Use `width()` or `height()`.
4. *Take the time to thoroughly test graphic formats before you make a lot of graphs.*
5. See:
   - template-stata-graph-formats.do
   - templates-stata-graph-formats-2017-06-20.docx

Sweat the details

1. In presentations, it is worth sweating the details.
2. Things go wrong that are not your fault, so prepare for them.
   - Not your fault, but listeners remember that your talk wasn't very good
3. The look of a paper affects how it is reviewed.
4. This might not be universalistic, but would you feel comfortable if
   - Your lawyer didn't look or act like a lawyer?
   - You surgeon was awkward and sloppy?

Part 20: Replication

Reproduction of results

1. Obtaining *exactly the same results* with the same data
   - You post your files for an article and others get your results
2. Not as easy as it looks, but you now have the necessary tools
3. With practice, these tools also make you more efficient

Reproducing wrong results

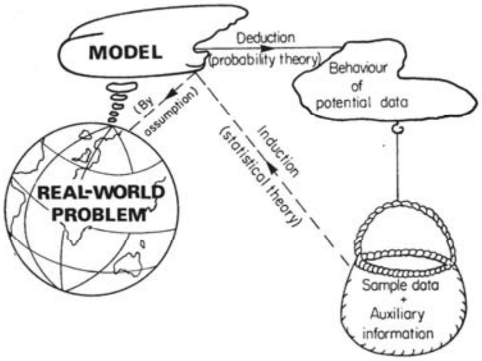
1. "Wrong" results occur for many reasons
   - Errors in data
   - Misspecification of models
   - Mistakes in analysis model
   - Other things?
2. Replication involves deeper aspects of being "wrong"

Replication of results

1. Replication requires that results generalize
   - Another lab runs the experiment based on your description
   - Someone tries your analysis with different dataset
2. What can go wrong?
   - Tacit knowledge
   - Failure to disclose procedures
   - Misunderstanding of the implications of how analyses were done
   - Lack of integrity

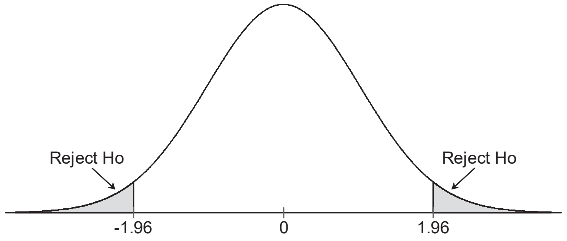
Paradigm for statistical inference

1. Vic Barnett’s Comparative Statistical Inference



---

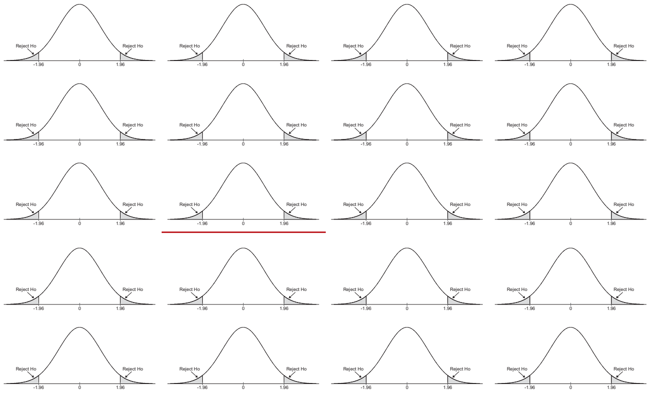
2. This paradigm implies a distribution of estimates from repeated sampling



3. What proportion is in the tail? What shape does it have?

4. What invalidates the sampling distribution?

Multiple tests at 5% level if there is no effect



Numbers are identical, but meanings differ

```
                  Test of       Stepwise
    Variable |    theory        selection
-------------+----------------------------
         bmi |     1.074***      1.074***
       white |     0.543***      0.543***
         age |     1.288***      1.288***
       agesq |     0.998***      0.998***
      female |     0.854*        0.854*
    hsdegree |     0.749***      0.749***
      weight |     1.004**       1.004**
       _cons |     0.000***      0.000***
-------------+----------------------------
           N |      8035          8035
         bic |  7594.215      7594.215
------------------------------------------
       legend: * p<.1; ** p<.05; *** p<.01
```

Discovery and verification

1. Randomly split sample three ways (wflec-replication-split.do)

```
lab def Lsample 1 Explore 2 Verify
qui sum hhid
local nhalf = round(r(N)/2,1)

// Split 1: first random splitting of sample
local split 1
set seed 193211
gen double _random = runiform()
lab var _random "uniform random"
sort _random
gen sample`split' = 2
replace sample`split' = 1 in 1/`nhalf'
lab var sample`split' "split `split' random subsample indicator"
lab val sample`split' sample
drop _random

// Split 2: second random splitting of sample
local split 2
set seed 19321 // split 2: female versus height
<snip>

// Split 3: third random splitting of sample
local split 3
set seed 192 // split 3: no height weight versus no female
<snip>
```

---

2. Stepwise selection of models with three different random splits of the sample into an exploration sample and a verification sample:

```
local rhs "age agesq bmi female height weight hsdegree white"

foreach split in 1 2 3 { // run on three random splits of data

    * stepwise selection with exploration sample
    stepwise, pe(.05): logit diabetes age agesq bmi female ///
        height weight hsdegree white if sample`split'==1
    est store explore`split'

    * stepwise selection with verification sample
    stepwise, pe(.05): logit diabetes age agesq bmi female ///
        height weight hsdegree white if sample`split'==2
    est store verify`split'

    estimates table explore`split' verify`split', ///
        eform p b(%9.3f) p(%9.2f) title(Randomization split `split')

}
```

Comparing results from six subsamples

```
------------------------------------------------------------------------------
         |       Split 1                Split 2                Split 3
Variable |  explore    verify      explore    verify      explore    verify
---------+--------------------------------------------------------------------
      bmi |  1.067***   1.066***    1.004      1.074***    1.101***   0.971
    white |  0.518***   0.547***    0.521***   0.543***    0.505***   0.562***
      age |  1.262***   1.351***    1.324***   1.288***    1.282***   1.341***
    agesq |  0.999***   0.998***    0.998***   0.998***    0.998***   0.998***
 hsdegree |  0.720***   0.680***    0.662***   0.749***    0.780***   0.650***
   weight |  1.006***   1.006***    1.016***   1.004**                1.022***
   height |                         0.936**                           0.909***
   female |                                    0.854*      0.733***
    _cons |  0.000***   0.000***    0.000***   0.000***    0.000***   0.000***
---------+--------------------------------------------------------------------
        N |   8036       8035        8036       8035        8036       8035
      bic | 7557.1     7479.6      7450.6     7594.2      7622.1     7405.3
------------------------------------------------------------------------------
                                            legend:  p<.1; ** p<.05; *** p<.01
```

Is data driven analysis worthless?

Do you always need to “register” analyses before you collect data?

1. Why are some funding agencies requiring “pre-registration of analyses”?
    - Does this preclude specification searches
2. Can you look at your data and then exclude observations?
    - Does astronomy do this?
    - Can you add dummy variables for each outlier?
3. Can you try multiple measures of a concept? Which one do you select
4. Are you more likely to replicate a simple or a complicated model?
5. What should you report to others to facilitate replication? Reproduction?
6. Is data mining software worthless?
    - VP for research at drug company
    - Steen Anderson and cost of mining software

Model Robustness

Extracted from: Young and Holsteen. 2015. Model Uncertainty and Robustness: A Computational Framework for Multimodel Analysis. *Sociological Methods and Research*.

Introduction

1. *Model uncertainty* is pervasive and inherent
2. Social theory provides testable ideas but not concrete direction on testing
    - Which control variables?
    - How to operationally define variables?
    - What functional form?
3. When the ‘‘true’’ model is unknown, which imperfect approximation is best?
    - Theory can be tested in many ways
    - Modest differences in methods can have large influence on the results

4. Empirical findings are a joint product of the data and the model.
    - Data do not speak for themselves
    - Different methods/models applied to same data often allow different conclusions
5. Choosing which model to report is “difficult, fraught with ethical and methodological dilemmas, and not covered in any serious way in classical statistical texts” (Ho et al. 2007:232).

Method

1. Framework to demonstrate robustness across sets of model specifications.
    - Fit all combinations of specified model ingredients
    - Report key statistics on the *modeling distribution of estimates*.
    - Identify model details that are empirically most influential.
2. Emphasize the parallel between
    - Uncertainty about the data
    - Uncertainty about the model

3. Standard errors reflect uncertainty about the data from sampling
4. Young’s method considers uncertainty about the model— how much an estimate changes in repeated modeling.
    - Do the results depend on minor and idiosyncratic aspects of model specification?
    - Important to probe inside the model to see which elements are critical to obtaining current results: *model influence analysis*

Point Estimates as model assumption sets

1. In classical statistics
    - True model is assumed known
    - One model is applied to a sample of data.
2. In practice, true model is not known and there are many possible variants
3. Analysts select options from a large menu of modeling assumptions making choices about the ‘‘best’’ functional form, set of control variables, operational definitions, and standard error calculations.

4. Point estimates cannot be calculated until these modeling decisions are made.
    - Point estimates represents a package of model assumptions and frequently captures just ‘‘one ad-hoc route through the thicket of possible models’’ (Leamer 1985:308).
    - When just one estimate is reported, these assumptions are effectively elevated to ‘‘dogmatic priors’’ that the data must be analyzed only with the exactly specified model (Leamer 2008:4).
5. Multi-model analysis is a way of relaxing these assumptions. For example,

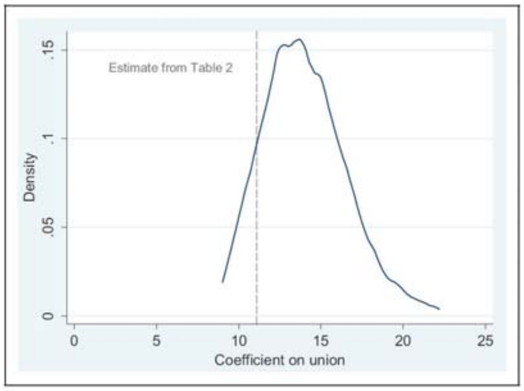


Figure 1. Modeling distribution of union wage premium.
Note: Kernel density graph of estimates from 1,024 models. Vertical line indicates the preferred estimate of an 11 percent union wage premium as reported in Table 2.

Science Isn't Broken

By Christie Aschwanden fivethirtyeight.com/features/science-isnt-broken/

1. If you follow the headlines, your *confidence in science may have taken a hit* lately. (edited by JSL)
   - **Peer review**? More like self-review. …a scam in which researchers were rubber-stamping their own work, circumventing peer review at five high-profile publishers.
   - **Scientific journals**? International Journal of Advanced Computer Technology accepted paper "Get Me Off Your F…ing Mailing List"… Two journals allowed an engineer posing as Maggie Simpson and Edna Krabappel to publish "Fuzzy, Homogeneous Configurations."
   - **Revolutionary findings? Possibly fabricated**. Berkeley, grad students discovered irregularities in LaCour's influential paper suggesting that an in-person conversation with a gay person could change how people felt about same-sex marriage. The journal Science retracted the paper when LaCour's co-author could find no record of the data.

---

2. Taken together, headlines like these suggest that science is a shady enterprise that spits out a bunch of dressed-up nonsense. But I've spent months investigating the problems hounding science, and I've learned that the headline-grabbing cases of misconduct and fraud are mere distractions.
3. But

   The state of our science is strong, but it's plagued by a universal problem: *Science is hard – really f…ing hard*.

   If we're going to rely on science as a means for reaching the truth - and it's still the best tool we have - it's important that we understand and respect just how difficult it is to get a rigorous result.

Part 21: Review of Workflow

The guiding philosophy

1. Reproduction is essential when you complete a project.
2. Replication is essential for scientific progress.

Workflow must address

The *universal aptitude for ineptitude* makes any human accomplishment an incredible miracle. --*Dr. John Paul Stapp*

Is a reproducible WF too hard?

Richard Ball at Haverford College (e-mail 17 March 2016)

And indeed, it is striking how naturally the whole reproducibility business comes to undergraduates--they simply can't imagine it could be acceptable to do work that isn't reproducible. And although some of them gnash their teeth a bit when I make them do everything in do files, it is just a matter of weeks before they totally get it and can't imagine how anyone could do things any other way. When I break the news to them that in fact a lot research by professional economics cannot be reproduced, they simply don't believe me. (Just like professional economists don't believe me when I tell them sophomores in an intro stats class are routinely producing comprehensive and accurate replication documentation for their papers.)

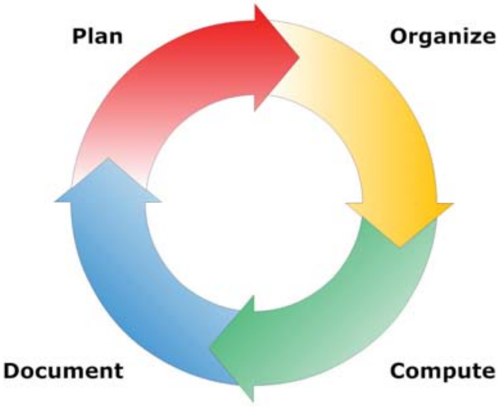
Criteria for choosing your WF

1. Reproducibility
2. Accuracy
3. Efficiency
4. Scalability
5. Standardization
6. Automation
7. Usability
8. Simplicity

Ask yourself repeatedly

1. How are my methods for documentation, planning, execution, and organization related to these criteria?
2. How do my tools relate to these criteria?
3. If I change something, how does it affect each criterion?

Never forget POD (and backups)



Critical computing WF

Posting principle

- You post it, you don't change it.

Dual workflow

- *Data management* and *data analysis* are kept distinct.

Run order naming

- Name do-files in the order they are to be run to be efficient and to provide implicit documentation.

If you disagree

1. There are many ways to create reproducible results.
2. If you develop alternative procedures, keep all criteria in mind.
3. A workflow must be coordinated, so think carefully about tinkering with things unless you think them through and test the changes.

Finally...

1. This class makes explicit a workflow I discovered and borrowed over decades.
2. Some topics seem far removed from substantive goals.
    - Research begins with an exciting idea that you hope will contribute to our knowledge of the world.
3. Do *not* make developing an effective workflow your goal.
    - WF is an essential means to producing work that makes a contribution.
4. *You will be tempted* to violate the principles of a good workflow.
    - More times than not, short-cuts will make things take longer.
    - An effective workflow must a balance competing demands.
5. An effective workflow:
    - Creates reproducible results.
    - Prevents big problems with sometimes cumbersome procedures.
    - It makes research less stressful.

---

the end

Workflow, not slow. – Bruce Frasier

Bill Gould on asking for help

Bill Gould: blog.stata.com/2010/12/ (edited by jsl)

If you ask a question poorly on statalist you probably won't be flamed, but you are unlikely to get an answer. To increase chances of a helpful response:

1. Subject line

Make the subject line of your email meaningful. Some good subject lines are: Survival analysis; Confusion about -stcox-; Unexpected error from -stcox-

2. First sentence

The purpose of the first sentence is to catch the attention of members who have an interest in your topic and let the others move on.

- I'm getting an unexpected error message from -stcox-.
- I'm using -stcox- and getting a result I know is wrong, so I know I'm misunderstanding something.

3. Second sentence

- I am using Stata 11.1 for Windows.; I am using Stata 10 for Mac.

4. The second (3rd, 4th,...) paragraph

Describe the problem concisely but completely. Sacrifice conciseness for completeness if you must or you think it will help. To the extent possible, simplify your problem by getting rid of extraneous details.

- I have 100,000 observations and 1,000 variables on firms, but 4 observations and 3 variables will be enough to show the problem. My data looks like this

```
firm_id     date      x
  10043       17     12
  10043       18      5
  13944       17     10
  27394       16      1
```

- I need data for each date with the # of firms and the average value of x:

```
date    no_of_firms   avg_x
  16              1       1
  17              2      11
  18              1      12
```

Here's another example for the second and subsequent paragraphs:

- Patients enter and leave the hospital, sometimes more than once over the period. I think it would be appropriate to combine the separate stays so that a patient who was in for 2 days and later for 4 days could be treated as being in for 6 days, except I also record how many separate stays there were. I'm evaluating cost, so treating cost as proportional to days in hospital, whatever their distribution, will be adequate. I'm looking at total days as a function of number of stays. The idea is that letting patients out too early results in an increase in total days, and I want to measure this. I realize that more stays and days might also arise simply because the patient was sicker. ...
- Is there some way I could estimate the model seperately within disease code, constraining the coefficient on number of stays to be the same? I saw something in the manual about stratified estimates, is that right?

---

5. You're asking someone to invest their time, so invest yours

1. Read what you have written and improve it. Help them by making your problem easy to understand.
2. The easier your problem is to understand, the more likely you are to get a response.
3. Sparkling prose or proper grammar are not required. Organization is more important than the style.
4. Avoid or explain jargon.

6. Tone

1. Write as if you are writing to a colleague you know well. Assume interest in your problem. Do not write as you might write to your research assistant, employee, servant, slave, or family member. "I'm busy & really I don't have time to check the Statalist postings, so respond to me directly, and soon."
2. Just as when writing to a colleague, in general you do not need to apologize, beg, or play on sympathies. Usually when I write to colleagues I know well, I just jump right in. The same rule works with Statalist.

---

7. What's appropriate

Questions appropriate for Stata's Technical Services are not appropriate for Statalist, and vice versa. Some questions aren't appropriate for either one, but those are rare. If you ask an inappropriate question, and ask it well, someone will usually direct you to a better source.

8. Who can ask, and how

You must join Statalist to send questions. Yes, you can join, ask a question, get your answer, and quit, but if you do, don't mention this at the outset. List members know this happens, but if you mention it when you ask the question, you'll sound superior and condescending. Also, stick around for a few days after you get your response, because sometimes your question will generate discussion. If it does, you should participate.