```
clear
set matsize 800  /* if you do not know what is matsize, type ’help matsize’ of course*/
set memory 200m, perm
version 10
set more off
cap log close
set logtype text

loc d c:\files

cap mkdir ‘d’
cd  ‘d’
*--------------------------------------------------------------------
*--------------------------------------------------------------------

/********************************************************************************/
/* before we get to the advanced programming let’s practice what we know so far */
/********************************************************************************/


//---combine macros

loc a aaa
loc b bbb
loc c ‘a’ ‘b’
di "‘c’"

//drop it first -- if you do not drop it and run it again it will add things
cap macro drop _all_letters
foreach letter in "aaa" "bbb" "ccc" {
loc all_letters   ‘all_letters’  ‘letter’
}
di "‘all_letters’"


local a= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a1= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a2= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a3= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a4= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b5= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

local aa="‘a’"+"‘a1’"+"‘a2’"+"‘a3’"+"‘a4’" +"‘a5’"

di "‘aa’"    //with the = the length of the macro is limits to 244 characters


local b  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b1= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b2= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b3= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b4= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b5= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

local bb "‘b’"  "‘b1’"  "‘b2’"  "‘b3’"  "‘b4’"  "‘b5’"

di "‘bb’"    //withOUT the = the length of the macro is limits to: see help limits


//----------------------------------extended macros----------------------------------------------
------
/* we have learnt  extended macros for variable labels and value lables and directories -- let’s
practice  */

use http://aok.us.to/class/data_mgmt/gss.dta, clear
d
sum

di "the label of the marital variable is ‘:var lab marital’   "

foreach var of varlist * {
di "the ‘var’ is laballed as ‘:var lab ‘var’ ’ "
}

codebook marital
di "it takes on some value labels, e.g. for val 1 the val label is ‘:label marital 1’ "
di "... for val 2 the val label is ‘:label marital 2’ "

levelsof marital,loc(_marital)
foreach val in ‘_marital’{
di "... for val ‘val’ the val label is ‘:label marital ‘val’  ’ "
}

//---a useful trick to grab the contents of directory...-----

//first generate some datasets
cd ‘d’
ls

sysuse auto, clear

preserve
keep mpg price
save auto1.dta, replace
restore

preserve
keep weight length
save auto2.dta, replace
restore

preserve
keep foreign rep78
save auto3.dta, replace
label data dd
restore

ls

local datafiles1 : dir . files "*.*"
local datafiles2 : dir . files "*.dta"

di ‘"‘datafiles1’"’
di ‘"‘datafiles2’"’


clear all
foreach file of local datafiles2 {
append  using ‘file’
}
d
sum
count

//---exercise 1

//grab and display all the value labels of the income variable in gss data
//hint: use levelsof, loop, and extended macro for value labels

//----------------------------------END extended macros-------------------------------------------
-------



//----------------------------------more loop examples--------------------------------------------
--

//---get results from regression

use http://aok.us.to/class/data_mgmt/gss.dta, clear

loc all_results ""
levelsof region, loc(_reg)
```

```
foreach r in `_reg'{
  reg happy inc if region=="`r'"
  loc result= _b[inc]
  loc name  "`r'"
  loc all_results `all_results' `name' `result'
}

di "`all_results'"

//---we can put them on the graph, too

use http://aok.us.to/class/data_mgmt/gss.dta, clear

loc all_results ""
levelsof region, loc(_reg)

foreach r in `_reg'{
  reg happy inc if region=="`r'"
  loc result= _b[inc]
  loc name  "`r'"
tw(lfit happy inc),title(`r') note(the relationship is `result')
}

//---we can combine graphs with gr combine

use http://aok.us.to/class/data_mgmt/gss.dta, clear

//create a var to hold graph names
gen graph_name=""

loc all_results ""
levelsof region, loc(_region)

loc i=1

foreach r in `_region'{
  reg happy inc if region=="`r'"
  loc result= _b[inc]
  loc name  "`r'"
qui tw(lfit happy inc if region=="`r'" ),title(`r') note(the relationship is `result') saving(`r'
,replace)
replace graph_name="`r'.gph" in `i'
  local ++i
}

l in 1/4
ta  graph_name

levelsof graph_name, loc(graph_name)
gr combine `graph_name', ycommon
gr export gr.eps, replace
! epstopdf gr.eps
! xpdf gr.pdf

//---now do exercise 2
// load gss \url{use http://aok.us.to/class/data_mgmt/gss.dta, clear}
// regress happiness on age by marital status, e.g. \ss{reg happy age if divorced==1}
// collect coefficients on marital status and display them


//----------------------------------END more loop examples---------------------------------------
------




//----------------------------------more nested loop examples and branching----------------------
-------

/* there are 7 days in a week labelled 1 through 7 and there are 4 weeks in a month labelled 1-4
and */
/* there are 12 months in a year labelled 1-12, display all of them! */
/* so we have 3 nested loops, start with the most outer one */

//note nice indenting !
forval month =1/12{
  forval week=1/4{
    forval day=1/7{
      di " this is ‘day’ day; ‘week’ week and ‘month’ month "
    }
  }
}

//lets pull out weekends

//note even nicer indenting: i usually do not indent, but with nested loops it really helps!
forval month =1/12{
  forval week=1/4{
    forval day=1/7{
      if (‘day’==6|‘day’==7){
        di "hooray this is the ‘week’ weekend of ‘month’"
      }
      else if (‘day’==1){
        di "i hate mondays"
      }
      else{
        di "just a regular day in ‘month’ month"
      }
    }
  }
}


/* let’s do something more useful: */
/* in auto data let’s  get the mean price for each repair record by foreign status */

sysuse auto, clear
d
sum

levelsof rep78, loc(_rep78)
levelsof foreign, loc(_foreign)

foreach rep78 in ‘_rep78’{
  foreach foreign in ‘_foreign’{
    qui sum price if rep78 ==‘rep78’ & foreign == ‘foreign’
    di "the mean for ‘:var lab rep78’  ‘: label rep78 ‘rep78’’ and  ‘:var lab foreign’ ‘: label o
rigin ‘foreign’’ is ‘r(mean)’"
  }
}




//branching gives you control:
foreach t  in "name1" "name2" "name3" {
      if ("‘t’"=="name1")     local a "1894 -1999"
      if ("‘t’"=="name2")     local a "1894 -1949"
      if ("‘t’"=="name3")     local a "1950 -1999"

                di "this is ‘t’ and period ‘a’"
              }
//compare to nested loop
foreach t  in "name1" "name2" "name3" {
  foreach a in "1894 -1999" "1894 -1949" "1950 -1999"{
    di "this is ‘t’ and period ‘a’"
  }
}


//----exercise 3:
/* let’s say that  we are interested in the variables about "family", so we do usual lookfor */
/* but we want to see  the codebook for variables with no more than 10 % of obs missing, write a
```

```
*/
/* loop to do this */
//hint: use branching with condition : (var count) / (tot count) > 0.9

copy http://publicdata.norc.org:41000/gss/documents//OTHR/1972_stata.zip ./gss72,replace
unzipfile gss72

use 1972.dta, clear




//----------------------------------ENDmore nested loop examples and branching-------------------
----



//---------------------------------------bonus---------------------------------------
//what follows is a bonus and not really required though useful and recommended
//--------------------------------------bonus--------------------------------------



//-------------------------loop incrementing-------------------------
//following comes form http://www.survey-design.com.au/

//we can  go by obs, with square brackets
local a=1
sysuse auto, clear
forvalues i=1/‘=_N’ {
if mpg[‘i’]==25 {
display "mgp equals 25  case ‘a’ "
local a=‘a’+1
}
}

//
sysuse auto, clear
local z=1

forvalues i=1/10 {

display mpg[‘z’]
display mpg[‘i’]
display
local ++z

}

//decrementing

sysuse auto, clear
local z=10

forvalues i=1/10 {

display mpg[11-‘z’]
display mpg[‘i’]
display
local --z

}

//more square brackets

clear
sysuse auto
sort mpg
list mpg in 1/10
generate a=1 if mpg[_n]!=mpg[_n-1]
list mpg a in 1/10
```

```
replace a=sum(a)
list mpg a in 1/10

//you can use _N and small _n to reverese order of obs

clear
sysuse auto
keep in 1/15
keep mpg
generate a=mpg[_N-_n+1]
list mpg a


local mpg1= mpg[1]
display "‘mpg1’"


//get var values into macro
sysuse auto, clear
quiet sum

forvalues i=1/‘=_N’{
if ‘i’==1 local make1= make[‘i’]
else local make= "‘make’" + " "+ make[‘i’]
}

display "‘make’"

local macname : list uniq make
display "‘macname’"

//------------------------END loop incrementing------------------------

extended macros
http://www.stata.com/help.cgi?extended_fcn

local <macro name> : <extended-function>

sysuse auto, clear

di "This is ‘:data label’"



//-------------labels again -- ds and tagging vars, labels, notes ------------------------------


// there are characteristics char that can access notes...

char _dta[one] this is char named one of _dta
char _dta[two] this is char named two of _dta
char list

char _dta[two]
char list

char mpg[one] this is char named one of mpg
char mpg[two] this is char named two of mpg
char list

char list mpg[]

char mpg[two]
char list

char rename mpg price
char list price[]

local mynotes: char  _dta[]
di "‘mynotes’"

foreach note in ‘mynotes’ {
```

```
di "‘: char  _dta[‘note’]’"
}


//from: http://www.michaelnormanmitchell.com/stow/the-ds-command-part-3.html

//can tag variables -- tags are useful !

use http://www.MichaelNormanMitchell.com/storage/stowdata/gradsvy, clear
des

//note nice spacing around colon!

//some intresting options of a useful command ds
ds , has(varlabel name*) detail
ds , has(varlabel name*) detail insensitiv

ds , has(vallab) detail
ds , not(vallab) detail
ds , has(vallab yesno) detail
return list
codebook q1
label define yesno_01 1 "Yes" 0 "No"

foreach v of varlist ‘r(varlist)’ {
   recode ‘v’ (2=0)
   label values ‘v’ yesno_01
}
des

ds , has(char note0) detail
ds , not(char note0) detail

ds , has(vallab) detail

char l
//say we are checking vars...
char q1[checked] yes
char q2[checked] yes
char q3[checked] yes
ds , has(char checked)
ds , not(char checked)


//---can put labels from one data to another
sysuse nlsw_labeled
d
sysuse nlsw_unlabeled
d

sysuse nlsw_labeled
keep if (1==2)
append using nlsw_unlabeled
d

//------------------------------END labels again-------------------------------







//-----------------------------------write macro----------------------------------
//this is a great utility -- you can read and write to text files....
//help file

//here the idea is to get the notes/char acteristics out of stata and
//write into a text file ...

//file command is the most powerful stata command--it writes a text
```

```
/* file out of any macro, so out of anything that stata can produce, in */
/* that way sata can communicate with ANY other language, R, latex, */
/* html, PHP, you name it */

file open w using w.tab, write replace
foreach note in ‘mynotes’ {
file write w   "this is note ‘note’" _tab  "‘: char  _dta[‘note’]’" _n
}
file close w

type w.tab

insheet using w.tab, clear
d
l


sysuse auto, clear
loc varlist mpg price weight

file open w using w2.tab, write replace

foreach v in  ‘varlist’{
    qui sum ‘v’
    file write w "‘v’" _tab  ‘"‘: variable label ‘v’’"’  _tab "‘c(filedate)’" _tab "‘r(N)’" _tab
"‘r(mean)’" _tab "‘r(min)’"  _tab "‘r(max)’"  _n
  }
file close w

type w2.tab

insheet using w2.tab, clear
d
l


//----------------------------------ENDwrite macro----------------------------------


/**********************************/
/* system parameters and settings */
/**********************************/

help creturn

c(pwd) returns a string containing the current (working) directory.
c(N) number of obs
c(k) number ov vars
c(filename) what did you use
c(width) returns a numeric scalar equal to the width, in bytes, of the dataset in memory.  If you
 had a
        dataset with two int variables, three floats, one double, and a str20 variable, the width
 of the
        dataset would be 2*2 + 3*4 + 8 + 20 = 44 bytes.  c(width) is equal to r(width), which is
returned by
        describe.

c(filedate) returns a string containing the date and time the file in c(filename) was last saved,
 such
        as "7 Jul 2006 13:51".  c(filedate) is equal to $S_FNDATE.

c(memory) allocated memory


*---------------------------------------------------------------------
//again you should install nonstandard commands [portability]

net from http://fmwww.bc.edu/RePEc/bocode/f
net install find

net from http://fmwww.bc.edu/RePEc/bocode/l
net install lookfor_all
net from  http://fmwww.bc.edu/RePEc/bocode/c
net install  checkvar

net from  http://fmwww.bc.edu/RePEc/bocode/c
net install  ckvar
*---------------------------------------------------------------------



//--------------------------------searching-----------------------------------


/*********************/
/* search text files */
/*********************/

//---find -- search  text files; lets search ado files that begin with "a"
cd "`c(sysdir_base)'a"

loc t : dir . file "*.ado"
find `t', m(version 8)
find, m(version 8)
//search just files
find *.class


//these options may not work for whaever reason... i guess they stop working once you install gre
p
find `t', m(version 8) , show
find `t', m(version 8) , zero

find `t', m(ver)
//you can modify find command to get the output into a macro...

/***************/
/* search data */
/***************/

//produce some data first

cd $d
pwd
ls

// remove files and dirs
loc f : dir . file "*"
di `"`f'"'
loc d : dir . dirs "*"
di `"`d'"'

foreach file in `f'{
  rm "`file'"
}

foreach dir in `d'{
  rmdir  `dir'
}

ls

net from http://www.MichaelNormanMitchell.com/storage/stowdata
net get stowdata

d
ls

lookfor_all wage
lookfor_all wage, describe
//search subdirs
lookfor_all wage, sub
//subdirs upto 3 down
lookfor_all wage, maxdepth(3) de
```

```
//filter files reexp
lookfor_all wage, filter(.2)
//filter subdirs regexp
lookfor_all wage, dirfilter(data) codebook

//search value labels
lookfor_all yes,  vlabs
//search notes
lookfor_all cens,  notes des

//--------------------------------END searching------------------------------------
```

```
clear
set matsize 800  /* if you do not know what is matsize, type ’help matsize’ of course*/
set memory 200m, perm
version 10
set more off
cap log close
set logtype text

loc d c:\files

cap mkdir ‘d’
cd  ‘d’
*--------------------------------------------------------------------
*--------------------------------------------------------------------

/********************************************************************************/
/* before we get to the advanced programming let’s practice what we know so far */
/********************************************************************************/


//---combine macros

loc a aaa
loc b bbb
loc c ‘a’ ‘b’
di "‘c’"

//drop it first -- if you do not drop it and run it again it will add things
cap macro drop _all_letters
foreach letter in "aaa" "bbb" "ccc" {
loc all_letters   ‘all_letters’  ‘letter’
}
di "‘all_letters’"


local a= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a1= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a2= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a3= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local a4= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b5= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

local aa="‘a’"+"‘a1’"+"‘a2’"+"‘a3’"+"‘a4’" +"‘a5’"

di "‘aa’"    //with the = the length of the macro is limits to 244 characters


local b  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b1= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b2= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b3= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b4= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
local b5= "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

local bb "‘b’"  "‘b1’"  "‘b2’"  "‘b3’"  "‘b4’"  "‘b5’"

di "‘bb’"    //withOUT the = the length of the macro is limits to: see help limits


//----------------------------------extended macros----------------------------------------------
------
/* we have learnt  extended macros for variable labels and value lables and directories -- let’s
practice  */

use http://aok.us.to/class/data_mgmt/gss.dta, clear
d
sum

di "the label of the marital variable is ‘:var lab marital’   "

foreach var of varlist * {
di "the ‘var’ is laballed as ‘:var lab ‘var’ ’ "
}

codebook marital
di "it takes on some value labels, e.g. for val 1 the val label is `:label marital 1' "
di "... for val 2 the val label is `:label marital 2' "

levelsof marital,loc(_marital)
foreach val in `_marital'{
di "... for val `val' the val label is `:label marital `val'  ' "
}

//---a useful trick to grab the contents of directory...-----

//first generate some datasets
cd `d'
ls

sysuse auto, clear

preserve
keep mpg price
save auto1.dta, replace
restore

preserve
keep weight length
save auto2.dta, replace
restore

preserve
keep foreign rep78
save auto3.dta, replace
label data dd
restore

ls

local datafiles1 : dir . files "*.*"
local datafiles2 : dir . files "*.dta"

di `"`datafiles1'"'
di `"`datafiles2'"'


clear all
foreach file of local datafiles2 {
append  using `file'
}
d
sum
count

//---exercise 1

//grab and display all the value labels of the income variable in gss data
//hint: use levelsof, loop, and extended macro for value labels

//----------------------------------END extended macros-------------------------------------------
-------



//----------------------------------more loop examples--------------------------------------------
--

//---get results from regression

use http://aok.us.to/class/data_mgmt/gss.dta, clear

loc all_results ""
levelsof region, loc(_reg)
```

```
foreach r in `_reg'{
  reg happy inc if region=="`r'"
  loc result= _b[inc]
  loc name  "`r'"
  loc all_results `all_results' `name' `result'
}

di "`all_results'"

//---we can put them on the graph, too

use http://aok.us.to/class/data_mgmt/gss.dta, clear

loc all_results ""
levelsof region, loc(_reg)

foreach r in `_reg'{
  reg happy inc if region=="`r'"
  loc result= _b[inc]
  loc name  "`r'"
tw(lfit happy inc),title(`r') note(the relationship is `result')
}

//---we can combine graphs with gr combine

use http://aok.us.to/class/data_mgmt/gss.dta, clear

//create a var to hold graph names
gen graph_name=""

loc all_results ""
levelsof region, loc(_region)

loc i=1

foreach r in `_region'{
  reg happy inc if region=="`r'"
  loc result= _b[inc]
  loc name  "`r'"
qui tw(lfit happy inc if region=="`r'" ),title(`r') note(the relationship is `result') saving(`r'
,replace)
replace graph_name="`r'.gph" in `i'
  local ++i
}

l in 1/4
ta  graph_name

levelsof graph_name, loc(graph_name)
gr combine `graph_name', ycommon
gr export gr.eps, replace
! epstopdf gr.eps
! xpdf gr.pdf

//---now do exercise 2
// load gss \url{use http://aok.us.to/class/data_mgmt/gss.dta, clear}
// regress happiness on age by marital status, e.g. \ss{reg happy age if divorced==1}
// collect coefficients on marital status and display them


//----------------------------------END more loop examples---------------------------------------
------




//----------------------------------more nested loop examples and branching----------------------
-------

/* there are 7 days in a week labelled 1 through 7 and there are 4 weeks in a month labelled 1-4
and */
/* there are 12 months in a year labelled 1-12, display all of them! */
/* so we have 3 nested loops, start with the most outer one */

//note nice indenting !
forval month =1/12{
  forval week=1/4{
    forval day=1/7{
      di " this is ‘day’ day; ‘week’ week and ‘month’ month "
    }
  }
}

//lets pull out weekends

//note even nicer indenting: i usually do not indent, but with nested loops it really helps!
forval month =1/12{
  forval week=1/4{
    forval day=1/7{
      if (‘day’==6|‘day’==7){
        di "hooray this is the ‘week’ weekend of ‘month’"
      }
      else if (‘day’==1){
        di "i hate mondays"
      }
      else{
        di "just a regular day in ‘month’ month"
      }
    }
  }
}


/* let’s do something more useful: */
/* in auto data let’s  get the mean price for each repair record by foreign status */

sysuse auto, clear
d
sum

levelsof rep78, loc(_rep78)
levelsof foreign, loc(_foreign)

foreach rep78 in ‘_rep78’{
  foreach foreign in ‘_foreign’{
    qui sum price if rep78 ==‘rep78’ & foreign == ‘foreign’
    di "the mean for ‘:var lab rep78’  ‘: label rep78 ‘rep78’’ and  ‘:var lab foreign’ ‘: label o
rigin ‘foreign’’ is ‘r(mean)’"
  }
}




//branching gives you control:
foreach t  in "name1" "name2" "name3" {
      if ("‘t’"=="name1")     local a "1894 -1999"
      if ("‘t’"=="name2")     local a "1894 -1949"
      if ("‘t’"=="name3")     local a "1950 -1999"

                di "this is ‘t’ and period ‘a’"
              }
//compare to nested loop
foreach t  in "name1" "name2" "name3" {
  foreach a in "1894 -1999" "1894 -1949" "1950 -1999"{
    di "this is ‘t’ and period ‘a’"
  }
}


//----exercise 3:
/* let’s say that  we are interested in the variables about "family", so we do usual lookfor */
/* but we want to see  the codebook for variables with no more than 10 % of obs missing, write a
```

```
*/
/* loop to do this */
//hint: use branching with condition : (var count) / (tot count) > 0.9

copy http://publicdata.norc.org:41000/gss/documents//OTHR/1972_stata.zip ./gss72,replace
unzipfile gss72

use 1972.dta, clear





//----------------------------------ENDmore nested loop examples and branching-------------------
----



//---------------------------------------bonus---------------------------------------
//what follows is a bonus and not really required though useful and recommended
//--------------------------------------bonus--------------------------------------



//-------------------------loop incrementing-------------------------
//following comes form http://www.survey-design.com.au/

//we can  go by obs, with square brackets
local a=1
sysuse auto, clear
forvalues i=1/‘=_N’ {
if mpg[‘i’]==25 {
display "mgp equals 25  case ‘a’ "
local a=‘a’+1
}
}

//
sysuse auto, clear
local z=1

forvalues i=1/10 {

display mpg[‘z’]
display mpg[‘i’]
display
local ++z

}

//decrementing

sysuse auto, clear
local z=10

forvalues i=1/10 {

display mpg[11-‘z’]
display mpg[‘i’]
display
local --z

}

//more square brackets

clear
sysuse auto
sort mpg
list mpg in 1/10
generate a=1 if mpg[_n]!=mpg[_n-1]
list mpg a in 1/10
```

```
replace a=sum(a)
list mpg a in 1/10

//you can use _N and small _n to reverese order of obs

clear
sysuse auto
keep in 1/15
keep mpg
generate a=mpg[_N-_n+1]
list mpg a


local mpg1= mpg[1]
display "‘mpg1’"


//get var values into macro
sysuse auto, clear
quiet sum

forvalues i=1/‘=_N’{
if ‘i’==1 local make1= make[‘i’]
else local make= "‘make’" + " "+ make[‘i’]
}

display "‘make’"

local macname : list uniq make
display "‘macname’"

//------------------------END loop incrementing------------------------

extended macros
http://www.stata.com/help.cgi?extended_fcn

local <macro name> : <extended-function>

sysuse auto, clear

di "This is ‘:data label’"



//-------------labels again -- ds and tagging vars, labels, notes -----------------------------


// there are characteristics char that can access notes...

char _dta[one] this is char named one of _dta
char _dta[two] this is char named two of _dta
char list

char _dta[two]
char list

char mpg[one] this is char named one of mpg
char mpg[two] this is char named two of mpg
char list

char list mpg[]

char mpg[two]
char list

char rename mpg price
char list price[]

local mynotes: char  _dta[]
di "‘mynotes’"

foreach note in ‘mynotes’ {
```

```
di "‘: char  _dta[‘note’]’"
}


//from: http://www.michaelnormanmitchell.com/stow/the-ds-command-part-3.html

//can tag variables -- tags are useful !

use http://www.MichaelNormanMitchell.com/storage/stowdata/gradsvy, clear
des

//note nice spacing around colon!

//some intresting options of a useful command ds
ds , has(varlabel name*) detail
ds , has(varlabel name*) detail insensitiv

ds , has(vallab) detail
ds , not(vallab) detail
ds , has(vallab yesno) detail
return list
codebook q1
label define yesno_01 1 "Yes" 0 "No"

foreach v of varlist ‘r(varlist)’ {
   recode ‘v’ (2=0)
   label values ‘v’ yesno_01
}
des

ds , has(char note0) detail
ds , not(char note0) detail

ds , has(vallab) detail

char l
//say we are checking vars...
char q1[checked] yes
char q2[checked] yes
char q3[checked] yes
ds , has(char checked)
ds , not(char checked)


//---can put labels from one data to another
sysuse nlsw_labeled
d
sysuse nlsw_unlabeled
d

sysuse nlsw_labeled
keep if (1==2)
append using nlsw_unlabeled
d

//-------------------------------END labels again------------------------------






//-----------------------------------write macro----------------------------------
//this is a great utility -- you can read and write to text files....
//help file

//here the idea is to get the notes/char acteristics out of stata and
//write into a text file ...

//file command is the most powerful stata command--it writes a text
```

```
/* file out of any macro, so out of anything that stata can produce, in */
/* that way sata can communicate with ANY other language, R, latex, */
/* html, PHP, you name it */

file open w using w.tab, write replace
foreach note in ‘mynotes’ {
file write w   "this is note ‘note’" _tab  "‘: char  _dta[‘note’]’" _n
}
file close w

type w.tab

insheet using w.tab, clear
d
l


sysuse auto, clear
loc varlist mpg price weight

file open w using w2.tab, write replace

foreach v in  ‘varlist’{
    qui sum ‘v’
    file write w "‘v’" _tab  ‘"‘: variable label ‘v’’"’  _tab "‘c(filedate)’" _tab "‘r(N)’" _tab
"‘r(mean)’" _tab "‘r(min)’"  _tab "‘r(max)’"  _n
  }
file close w

type w2.tab

insheet using w2.tab, clear
d
l


//-----------------------------------ENDwrite macro-----------------------------------


/***********************************/
/* system parameters and settings */
/***********************************/

help creturn

c(pwd) returns a string containing the current (working) directory.
c(N) number of obs
c(k) number ov vars
c(filename) what did you use
c(width) returns a numeric scalar equal to the width, in bytes, of the dataset in memory.  If you
 had a
        dataset with two int variables, three floats, one double, and a str20 variable, the width
 of the
        dataset would be 2*2 + 3*4 + 8 + 20 = 44 bytes.  c(width) is equal to r(width), which is
returned by
        describe.

c(filedate) returns a string containing the date and time the file in c(filename) was last saved,
 such
        as "7 Jul 2006 13:51".  c(filedate) is equal to $S_FNDATE.

c(memory) allocated memory


*---------------------------------------------------------------------
//again you should install nonstandard commands [portability]

net from http://fmwww.bc.edu/RePEc/bocode/f
net install find

net from http://fmwww.bc.edu/RePEc/bocode/l
net install lookfor_all
net from  http://fmwww.bc.edu/RePEc/bocode/c
net install  checkvar

net from  http://fmwww.bc.edu/RePEc/bocode/c
net install  ckvar
*---------------------------------------------------------------------



//--------------------------------searching------------------------------------


/*********************/
/* search text files */
/*********************/

//---find -- search  text files; lets search ado files that begin with "a"
cd "`c(sysdir_base)'a"

loc t : dir . file "*.ado"
find `t', m(version 8)
find, m(version 8)
//search just files
find *.class


//these options may not work for whaever reason... i guess they stop working once you install gre
p
find `t', m(version 8) , show
find `t', m(version 8) , zero

find `t', m(ver)
//you can modify find command to get the output into a macro...

/***************/
/* search data */
/***************/

//produce some data first

cd $d
pwd
ls

// remove files and dirs
loc f : dir . file "*"
di `"`f'"'
loc d : dir . dirs "*"
di `"`d'"'

foreach file in `f'{
  rm "`file'"
}

foreach dir in `d'{
  rmdir  `dir'
}

ls

net from http://www.MichaelNormanMitchell.com/storage/stowdata
net get stowdata

d
ls

lookfor_all wage
lookfor_all wage, describe
//search subdirs
lookfor_all wage, sub
//subdirs upto 3 down
lookfor_all wage, maxdepth(3) de
```

```
//filter files reexp
lookfor_all wage, filter(.2)
//filter subdirs regexp
lookfor_all wage, dirfilter(data) codebook

//search value labels
lookfor_all yes,  vlabs
//search notes
lookfor_all cens,  notes des

//--------------------------------END searching------------------------------------
```