

version 10.1

capture log close

set more off

log using c:\data9\hei2005.log, replace

* ****

* Set local macro flag ON and initiate debugging. Minor debugging occurs *

* when debug = 1; more extensive debugging occurs when debug = 2. *

* ****

local debug = 1

/*

* ****

* Purpose: To calculate the 12 Healthy Eating Index-2005 (HEI-2005) compon-

* ent scores and the total score for population, 2001-2002 *

* ****

* The 12 HEI-2005 components are: HEI Min Max *

* Total Fruit 1 0 5 *

* Whole Fruit 2 0 5 *

* Total Vegetables 3 0 5 *

* Dark Green Vegetables, *

* Orange Vegetables, Legumes 4 0 5 *

* Total Grains 5 0 5 *

* Whole Grains 6 0 5 *

```

* Milk          7  0  10      *
* Meat and Beans      8  0  10      *
* Oils             9  0  10      *
* Saturated Fat     10  0  10      *
* Sodium           11  0  10      *
* Calories from Solid Fats,      *
* Alcohol & Added Sugars  12  0  20      *
* -----      *
* Total HEI-2005      0  100      *

```

* To run this Stata command file, users need five external data sets: *

* *

* 1. MyPyrEquivDB_v1, which should be found at *

* http://www.ars.usda.gov/Services/docs.htm?docid=8503&pf=1&cg_id=0 *

* *

* 2. CNPPMyPyrEquivDB_v1_WJFRT, which should be found at *

* <http://www.cnpp.usda.gov/HealthyEatingIndexSupportFiles0102.htm> *

* *

* Users also need three files from the 2001-2002 NHANES, which can be *

* downloaded from *

* <http://www.cdc.gov/nchs/about/major/nhanes/nhanes01-02.htm> *

* *

* 3. NHANES\2001-2002\Stata\demo_b.dta *

```

* 4. NHANES\2001-2002\Stata\drxtotal_b.dta          *
* 5. NHANES\2001-2002\Stata\drxiff_b.dta          *
*
*
* The MyPyramid Equivalent Database (MPED) and NHANES 2001-2002 datasets *
* are used to create six of the 12 HEI-2005 components: Total Fruit; Total *
* Grains; Whole Grains; Oils; Saturated Fat; and Sodium.          *
* The CNPP MyPyramid Equivalent Database for Whole Fruit and Fruit Juice *
* is used to create one of the components: Whole Fruit. Additional steps *
* are used to create the remaining five components: Total Vegetables; Dark *
* Green and Orange Vegetables and Legumes; Milk; Meat and Beans; and *
* Calories from Solid Fat, Alcohol, and Added Sugar (SoFAAS).      *
*
*
* The NHANES individual food file, drxiff_b, is in univariate (or long) *
* format; the NHANES demographics and food total files are in multivariate *
* (or wide) format.          *
*
*
* 17 data Files are written to c:\data9\ and erased with capture erase *
* commands that have been placed at the end of this program.      *
*
* 1. hei_mped.dta          *
* 2. hei_wjfrt.dta        *
* 3. hei_food.dta         *
* 4. hei_nutrient.dta     *
* 5. hei_demo.dta         *
* 6. hei_pyr.dta          *
* 7. hei_fdpyr.dta        *

```

```

* 8. hei_pyrcalc.dta *
* 9. hei_bwlnt.dta *
* 10. hei_bwlpwr.dta *
* 11. hei_beerwineliq.dta *
* 12. hei_both.dta *
* 13. hei_2005pop.dta *
* 14. hei_pop0102.dta *
* 15. hei_hei2005pop_0102se.dta *
* 16. hei_hei2005pop_0102se_notrunnew.dta *
* 17. hei_pop0102seNEWpopwt *
* *
* Future Modifications: *
* *
* 1. Introduce strict variable typing with every Stata generate command. *
* generate {data type} [variable name] = {expression} *
* *
* 2. Eliminate the use of string variables, which require more storage *
* than a variable stored as byte; *
* *
* 3. Eliminate the use of derived variables like f_total1000. Calculation *
* is more efficient than storage. *
* *
* 3. Reduce all steps to 2 stages: *
* 1) create the variables needed to calculate the 12 component scores; *
* 2) calculate the 12 component scores *

```

```

*
*
* *****
*/

* *****

* Because Stata only operates on the variables within the data set currently *
* loaded in memory, the user must allocate sufficient memory at the start of *
* the program (or prior to the command that requires the larger memory). 75 *
* MB seems excessive and could probably be reduced by restricting work to *
* files that are in wide (multivariate) format. *
* *****

quietly {
  clear
  set mem 75m
  cd c:\data9
}

use "mypyrequivdb_v1.dta", clear

* *****

* mypyrequivdb_v1 is in long or univariate format. It contains 6974 records *
* with 6974 unique foodcodes; and 36 variables. It was retrieved from the *
* USDA site. *
* *****

label variable foodcode "8-digit USDA food code"

```

```
format foodcode %15.0f
sort foodcode
compress
save c:\data9\hei_mped.dta, replace
```

```
* ****
```

```
* Step 1. Calculation of individuals' food group, nutrient, and energy *
```

```
* intakes *
```

```
* *
```

```
* Conversions (to count soy beverages as the Milk food group in HEI-2005) *
```

```
* foodcode==11310000: milk, imitation, fluid, soy based (1 cup=244 grams) *
```

```
* foodcode==11320000: milk, soy, ready-to-drink (1 cup=245 grams) *
```

```
* foodcode==11321000: milk, soy, ready-to-drink, chocolate (1 cup=240 grams)*
```

```
* foodcode==11330000: milk, soy, dry, reconstituted (1 cup=245 grams) *
```

```
* *
```

```
* This code fragment only changes 4 of the 6,974 records that are in a file *
```

```
* that the user has to retrieve and download from the USDA site. *
```

```
* ****
```

```
label variable m_soy "oz equivalents from soybean products"
```

```
label variable d_total "dairy cup equivalents"
```

```
quietly {
```

```
  replace m_soy = 0 if foodcode== 11310000
```

```
  replace d_total = round(100*(1/244),.001) if foodcode== 11310000
```

```
  replace m_soy = 0 if foodcode== 11320000
```

```

replace d_total = round(100*(1/245),.001) if foodcode== 11320000
replace m_soy = 0 if foodcode== 11321000
replace d_total = round(100*(1/240),.001) if foodcode== 11321000
replace m_soy = 0 if foodcode== 11330000
replace d_total = round(100*(1/245),.001) if foodcode== 11330000
}
if `debug'== 2 {
    format d_total %6.3f
    clist m_soy d_total if inlist(foodcode, 11310000, 11320000, 11321000, 11330000)
}

sort foodcode

save c:\data9\hei_mped.dta, replace

```

```

*****
* hei_mped contains 6,974 records and 35 variables.          *
* foodcode  v_drkgr  f_citmlb  m_meat  m_soy  *
* equivflag v_orange f_other  m_organ m_nutsd *
* modcode   v_potato d_total  m_frank  legumes *
* g_total   v_starcy d_milk   m_poult  discfat_oil *
* g_whl     v_tomato d_yogurt m_fish_hi discfat_solid *
* g_nwhl    v_other  d_cheese m_fish_lo add_sug  *
* v_total   f_total  m_mpf    m_egg   a_bev   *
* *****

```

```
use "C:...\cnppmypyrequivdb_v1_wjfrt.dta", clear
```

```
* ****
```

```
* This file contains 6,974 records and 4 variables: foodcode, foodname *
```

```
* (str60), frtjuice and wholefrt. These data are in a file that the user *
```

```
* has to retrieve and download from the USDA site. *
```

```
* ****
```

```
format foodcode %15.0f
```

```
sort foodcode
```

```
compress
```

```
save c:\data9\hei_wjfrt.dta, replace
```

```
use "C:...\drxiff_bv.dta", clear
```

```
* ****
```

```
* This NHANES individual food file is in long format and is an extract of *
```

```
* the full file that was released November 2007. The original file contains*
```

```
* 143,004 records and 74 variables. The extract contains 143,004 records and*
```

```
* 17 variables. The records were reported by 9882 individuals. After the *
```

```
* unreliable records are deleted, 140,817 records were saved to hei_food. *
```

```
* ****
```

```
label variable drddrstz "reliable dietary data"
```

```
label variable drdifdcd "8-digit USDA food code"
```

```
drop if drddrstz~=1
```

```
gen long foodcode = 1.0*drdifdcd
```

```
sort foodcode
```

```
format foodcode %15.0f
```

```

if `debug'==2 {
    count if foodcode ~=.
    describe
}

compress

save c:\data9\hei_food.dta, replace

use seqn wtdrd1 drddrstz drxtkcal drxtcarb drxtsfat drxtalco drdtsodi /*
*/ using "C:\Projects\NHANES\2001-2002\Stata\drxtot_b.dta", clear

* *****
* This NHANES total nutrient file is in wide format; it contains 10477 *
* records and 141 variables. It was released November 2007. After deleting *
* unreliable records, 9701 records were saved to hei_nutrient.dta. *
* *****

drop if drddrstz~=1

drop drddrstz

gen byte in_N = 1

if `debug'==2 {
    tab in_N
    count if seqn~=.
}

sort seqn

compress

```

```
save c:\data9\hei_nutrient.dta, replace
```

```
* ****
```

```
* hei_nutrient contains 9701 records and 8 variables: *
```

```
* seqn wtdrd1 drxtkcal drxtcarb drxtsfat drdtsodi drxtalco in_N *
```

```
* ****
```

```
use seqn ridageyr riagendr sddsrvyr sdmvpsu sdmvstra /
```

```
*/ using "C:\Projects\NHANES\2001-2002\Stata\demo_b.dta", clear
```

```
* ****
```

```
* This NHANES demographics file is in wide format; it contains 11039 records*
```

```
* and 23 variables. *
```

```
* ****
```

```
sort seqn
```

```
save c:\data9\hei_demo.dta, replace
```

```
* ****
```

```
* Step 2: Merge the datasets and calculate NHANES 2001-2002 individual's *
```

```
* food groups and nutrient intakes. *
```

```
* ****
```

```
use c:\data9\hei_mped, clear
```

```
merge foodcode using c:\data9\hei_wjfrt.dta
```

```
* ****
```

```
* Each file - hei_mped & hei_wjfrt - contains 6974 records that were sorted *
```

```

* on the 1:1 match merge key, foodcode.
*
* *****
if `debug'==2 {
    tab _merge, miss
}
keep if _merge==3
drop _merge
format foodcode %15.0f
sort foodcode
compress
save c:\data9\hei_pyr.dta, replace

if `debug'==2 {
    capture drop record
    by foodcode: gen byte record = _n
    tab record, miss

    dir c:\data9\hei_*.dta
}

* *****

* hei_pyr contains 6,974 records. Each foodcode is unique.
*
* *****

* hei_food contains 140,817 records on 9,700 individuals. There are 1 to 46
* records per individual. This is a 1: many match merge on foodcode. (See

```

* Example 5, pg 397-398, Stata Data Management Manual, Release 10 (2007). *

* ****

```
clear
```

```
use c:\data9\hei_pyr.dta, clear
```

```
merge foodcode using c:\data9\hei_food.dta
```

```
tab _merge if `debug'==1 | `debug'==2
```

```
keep if _merge==3
```

```
drop _merge
```

```
sort foodcode
```

```
format foodcode %15.0f
```

```
compress
```

```
save c:\data9\hei_fdpyr.dta, replace
```

```
if `debug'==2 {
```

```
    describe , simple
```

```
}
```

* Convert NHANES 2001-2002 food intakes from grams to MyPyramid oz and cup equivalents for food groups

```
label variable f_total "fruit cup equivalents"
```

```
label variable wholefrt "whole fruit cup equivalents"
```

```
label variable v_total "vegetable cup equivalents"
```

```
label variable v_drkgr "dark green vegetable cup equivalents"
```

```
label variable v_orange "orange vegetable cup equivalents"
```

```
label variable legumes "cup equivalents of cooked dry beans & peas"
```

```

label variable g_total "oz equivalents of grain group"
label variable g_whl "oz equivalents of whole grains"
label variable d_total "dairy cup equivalents"
label variable m_mpf "meat, poultry, fish oz equivalents"
label variable m_egg "oz equivalents of egg"
label variable m_nutsd "oz equivalents from nuts and seeds"
label variable m_soy "oz equivalents from soybean products"
label variable discfat_oil "gms of discretionary oil"
label variable discfat_sol "gms of discretionary solid fat"
label variable add_sug "teaspoons of added sugar"
label variable drxigrms "grams of food consumed"

foreach var of varlist f_total wholefrt v_total v_drkgr v_orange /*
*/ legumes g_total g_whl d_total m_mpf m_egg m_nutsd m_soy /*
*/ discfat_oil discfat_sol add_sug {
    quietly replace `var' = `var'*(drxigrms/100)
}

sort seqn

if `debug'==2 {
    sum , sep(0)
}

save c:\data9\hei_fdpyr.dta, replace

if `debug'==2 {
    describe , simple
}

```

```
* ****
```

```
* hei_fdpvr.dta contains 140,817 records on 9,700 individuals. There are 1 *
```

```
* to 46 records per individual. There are 56 variables. *
```

```
* ****
```

```
* Calculate individual's food intake amounts for MyPyramid food groups
```

```
collapse (sum) f_total wholefrt v_total v_drkgr v_orange legumes /*
```

```
*/ g_total g_whl d_total m_mpf m_egg m_nutsd m_soy discfat_oil /*
```

```
*/ discfat_sol add_sug, by(seqn)
```

```
gen byte in_P = 1
```

```
sort seqn
```

```
if `debug'==2 {
```

```
    sum , sep(0)
```

```
    clist seqn f_total wholefrt v_total v_drkgr in 1/25
```

```
}
```

```
save c:\data9\hei_pyrcalc.dta, replace
```

```
* ****
```

```
* hei_pyrcalc.dta contains 9,700 records and 18 variables. *
```

```
* ****
```

```
if `debug'==2 {
```

describe , simple

}

```
* *****  
* Nutrient intakes (drxicarb, carbohydrate; drxialco, alcohol) from beer, *  
* wine and distilled spirits, excluding cooking wine. The first 3 digits of*  
* the 8-digit USDA food code are used to indicate alcoholic beverages (931 *  
* through 935); the food code for cooking wine (93401300) is excluded. *  
* *****
```

use seqn drxicarb drxialco drxiline foodcode using c:\data9\hei_food.dta, clear

* threed simply extracts the first 3 characters of foodcode

gen int threed = int(foodcode/100000)

if `debug'==2 {

sum threed

}

keep if (threed >= 931 & threed <= 935)

drop if foodcode== 93401300

sort seqn drxiline

compress

save c:\data9\hei_bwlNut.dta, replace

```
* *****  
* hei_bwlNut contains 1,395 records and 6 variables: seqn, drxiline, *
```

```

* drxicarb, drxialco, foodcode, and threed.          *
* ****

if `debug'==2 {
    describe , simple
}

* ****

* Identify Added Sugar intake from beer, wine and distilled spirits.  *
* Exclude cooking wine.          *
* ****

* hei_fdpvr contains 140,817 records on 55 variables.          *
* hei_bwlpvr contains 1,395 records on 6 variables.          *
* ****

use seqn add_sug drxigrms drxiline foodcode using c:\data9\hei_fdpvr.dta, clear
gen int threed = int(foodcode/100000)
if `debug'==2 {
    sum threed
}
keep if (threed >= 931 & threed <= 935)
drop if foodcode== 93401300
sort seqn drxiline
compress
save c:\data9\hei_bwlpvr.dta, replace

```

```
if `debug'==2 {  
    describe , simple  
}
```

```
* ****
```

```
* hei_bwlntut.dta is in long format. It contains 1,395 records. Individuals *  
* have from 1(1,091) to 6(1) records each. Foodcode is not unique within *  
* this file. hei_bwlpvr.dta is in long format. It contains 1,395 records. *  
* Individuals have from 1(1,091) to 6(1) records each. Matching on seqn *  
* and drxiline effects a 1:1 match merge as the seqn-drxiline combinations *  
* are unique. hei_beerwiniq.dta contains 1,395 records; individuals have *  
* 1(1,091) to 6(1) records. *
```

```
* ****
```

```
use c:\data9\hei_bwlntut.dta, clear  
merge seqn drxiline using c:\data9\hei_bwlpvr.dta  
tab _merge if `debug'==1 | `debug'==2  
sort seqn drxiline  
drop _merge  
save c:\data9\hei_beerwiniq.dta, replace  
if `debug'==2 {  
    describe , simple  
}
```

* *****

* Exclude calories from Added Sugars food group in alcoholic beverages *

* *

* Convert teaspoons to grams of added sugars, then to grams of carbohydrate *

* from added sugars *

* *

* Subtract grams of carbohydrate from added sugars from total carbohydrate *

* to calculate SoFAAS calories from alcohol (ethanol) in alcoholic *

* beverages *

* *

* The threed values in hei_beerwineliq.dta are: *

* *

* threed | Freq. Percent Cum. *

*-----+----- *

* 931 | 769 55.13 55.13 *

* 932 | 24 1.72 56.85 *

* 933 | 73 5.23 62.08 *

* 934 | 288 20.65 82.72 *

* 935 | 241 17.28 100.00 *

*-----+----- *

* Total | 1,395 100.00 *

* *****

* *

* bwcarbc has 314 missing values. This subsequently impacts the computation *

* of hei12. Here, the missing values were set equal to 0 on line 466. *

```

*
*
* . gen suggram = 4*add_sug
*
* . gen noscarb = drxicarb - suggram
*
* . gen bwcarbc = 4*noscarb if inlist(threed, 931, 932, 934)
*
* (314 missing values generated)
*
* . gen ethcal = 7*drxialco
*
*
*
* Calories from alcoholic beverages are calculated from ethanol (drxialco)
* and carbohydrate (drxicarb) contained in beverages. To prevent double-
* counting calories from added sugars, calories from added sugars are sub-
* tracted from the calories from alcoholic beverages, which are accounted
* for in the Added Sugars part of the Calories from SoFAAS component.
*
*****

```

```

gen float suggram = 4*add_sug
gen float noscarb = drxicarb - suggram
gen float ethcal = 7*drxialco
gen float bwcarbc = 4*noscarb if inlist(threed, 931, 932, 934)
replace bwcarbc = 0 if bwcarbc==.

```

```

label variable add_sug "the added sugars food group"
label variable suggram "grams of sugar"
label variable drxicarb "intake (carbohydrate)"
label variable noscarb "carbohydrate from non-added-sugar"
label variable drxialco "intake (alcohol)"

```

```
label variable ethcal "calories from ethanol in alcoholic beverages"
```

```
label variable bwcarb "carbohydrate calories from beer, wine & sprits"
```

```
if `debug'==1 | `debug'==2 {
```

```
    count if inlist(threed, 931, 932, 934)
```

```
    sum seqn add_sug suggram drxicarb noscarb bwcarb drxialco ethcal, sep(0)
```

```
}
```

```
sort seqn
```

```
compress
```

```
save c:\data9\hei_beerwineliq.dta, replace
```

```
if `debug'==2 {
```

```
    describe , simple
```

```
}
```

```
* ****
```

```
* hei_beerwineliq contains 1,395 observations and 12 variables: *
```

```
* seqn drxiline drxicarb drxialco foodcode threed *
```

```
* add_sug drxigrms suggram noscarb ethcal bwcarb *
```

```
* ****
```

```
* ****
```

```
* Calculate a person's SoFAAS calories from carbohydrate & alcohol (ethanol)*
```

```
* in alcoholic beverages in 1 day. *
```

```

*
*
* Collapse data by seqn and get totals for bwcarbc and ethcal. Input file *
* was beerwineliq; totals written to beerwineliq.
*
*
* Variable | Obs Mean Std. Dev. Min Max *
*-----+-----*
* seqn | 1091 15546.54 3164.246 9966 21004 *
* bwcarbc | 1091 97.03005 141.7208 0 1278.72 *
* ethcal | 1091 328.1422 378.0433 1.4 3860.5 *
*
*
* Combine all required datasets; include individuals older than 2 *
* Exclusion criteria change (9/17/07): include all preg and lactating women *
* *****

```

```

keep seqn bwcarbc ethcal

sort seqn

collapse (sum) bwcarbc ethcal, by(seqn)

if `debug'==1 | `debug'==2 {
    sum
}

save c:\data9\hei_beerwineliq.dta, replace

if `debug'==2 {

```

```
describe , simple
}

use c:\data9\hei_demo.dta, clear

merge seqn using c:\data9\hei_nutrient c:\data9\hei_pyrcalc.dta c:\data9\hei_beerwineliq.dta

keep if ridageyr>1.99 & ridageyr~=.

keep if in_N==1 & in_P==1

tab _merge

drop _merge

sort seqn

compress

if `debug'==2 {
    count if seqn~=.

    clist seqn in 1/10
}

save c:\data9\hei_both.dta, replace

if `debug'==2 {
    describe , simple

    dir c:\data9\hei_*.dta
}
```

* ****

* Step 3: Calculate food group and nutrient intakes *

* *

* Note: MyPyramid equivalent values for total vegetable intake (V_TOTAL) *

* in the HEI-2005 may be different from V_TOTAL in the MPED because legumes *

* may be counted as vegetables or meat in the HEI-2005; and total dairy *

* intake D_TOTAL in the HEI-2005 may be different from D_TOTAL in the MPED *

* because soy beverages are counted as milk in the HEI-2005. *

* *

* Calculate HEI-2005 Meat & Beans intake; Standard is 2.5 oz equiv/1000 kcal*

* *

* Calculate total meat intake from meat, poultry, & fish; egg; nuts & seeds;*

* soy products *

* ****

```
use c:\data9\hei_both.dta, clear
```

```
keep if seqn~=.
```

```
compress
```

```
save c:\data9\hei_2005pop.dta, replace
```

```
gen float allmeat = m_mpf + m_egg + m_nutsd + m_soy
```

```
if `debug'==1 | `debug'==2 {
```

```
    describe m_mpf m_egg m_nutsd m_soy allmeat
```

```
}
```

```

if `debug'==2 {
egen nmiss = rowmiss(m_mpf m_egg m_nutsd m_soy)

tab nmiss, miss

drop nmiss

sum allmeat m_mpf m_egg m_nutsd m_soy, sep(0)
}

```

* Create the Meat and Beans standard

```
gen float mbmax= 2.5*(drxtkcal/1000)
```

```
label variable mbmax "maximum standard for meat and beans"
```

```
* ****
```

```
* Legumes intake calculation. Legumes intake counts as meat and beans *
```

```
* until standard is met. Then the rest count as total vegetables. There *
```

```
* are 3 types of calculations for legume intake: *
```

```
* *
```

```
* (a) If total meat intake is less than the Meat and Beans standard, *
```

```
* all Legumes go to Meat and Beans. *
```

```
* *
```

```
* Convert cup equivalents of legumes to oz equivalents *
```

```
* *
```

```
* (b) Some Legumes go to Meat and Beans; the rest go to Total Vegetables *
```

```

*
*
* Convert the extra oz equivalents of legumes back to cup equivalents,
*
* after meeting the Meat and Beans standard
*
*
* (c) If total meat intake exceeds the Meat and Beans standard, then all
*
* Legumes count as Total Vegetables
*
*
* *****
*
* *****
* If legtype were byte, value labels could be used to identify the various
*
* leg intake types. Byte would also require less storage than a string
*
* variable.
*
* The three types were allmeat, meat/veg, and allveg.
*
* *****

```

```
gen str8 legtype = ""
```

```
gen float v_dol = .
```

```
gen float meatleg = .
```

```
gen float needmeat = .
```

```
gen float extrmeat = .
```

```
gen float extrleg = .
```

```
label variable legtype "type of legume intake"
```

```
label variable v_dol "dark green vegetables, orange vegetables, & legume cup equivalents"
```

```
label variable meatleg "oz equivalents of meat from legumes"
```

```
label variable needmeat "oz equivalents of meat needed to meet Meat and Beans standard"
```

```
label variable extrmeat "oz equivalents needed from legumes to meet Meat and Beans standard"
```

```
label variable extrleg "extra legumes in cup equivalents"
```

```
gen byte meatflag = cond(allmeat< mbmax, 1, 0)
```

```
if `debug'==1 | `debug'==2 {
```

```
    sum allmeat mbmax
```

```
    tab meatflag, miss
```

```
}
```

```
replace meatleg = 4*legumes    if (allmeat<mbmax)
```

```
replace needmeat = mbmax - allmeat  if (allmeat<mbmax)
```

```
gen byte test = cond( ((meatleg<=needmeat) & (allmeat<mbmax)), 1, 0)
```

```
replace legtype = "allmeat"    if (meatleg <= needmeat) & (allmeat<mbmax)
```

```
replace allmeat = allmeat + meatleg  if (meatleg <= needmeat) & (allmeat<mbmax)
```

```
replace v_total = v_total    if (meatleg <= needmeat) & (allmeat<mbmax)
```

```
replace v_dol = v_orange + v_drkgr if (meatleg <= needmeat) & (allmeat<mbmax)
```

```
replace meatflag = cond(allmeat< mbmax, 1, 0)
```

```
if `debug'==1 | `debug'==2 {
```

```

sum allmeat mbmax meatleg needmeat

tab meatflag test, miss

}

replace legtype = "meat/veg"    if (meatleg > needmeat) & (allmeat < mbmax)
replace extrmeat = meatleg - needmeat if (meatleg > needmeat) & (allmeat < mbmax)
replace extrleg = extrmeat/4.0    if (meatleg > needmeat) & (allmeat < mbmax)
replace allmeat = allmeat + needmeat if (meatleg > needmeat) & (allmeat < mbmax)
replace v_total = v_total + extrleg if (meatleg > needmeat) & (allmeat < mbmax)
replace v_dol   = v_orange + v_drkgr + extrleg if (meatleg > needmeat) & (allmeat < mbmax)

replace meatflag = cond(allmeat < mbmax, 1, 0)
replace test = cond(((meatleg > needmeat) & (allmeat < mbmax)), 1, 0)

if `debug'==1 | `debug'==2 {

    sum allmeat mbmax meatleg needmeat

    tab meatflag test, miss

}

replace legtype = "allveg" if (allmeat >= mbmax)
replace v_total = v_total + legumes if (allmeat >= mbmax)
replace v_dol   = v_orange + v_drkgr + legumes if (allmeat >= mbmax)

if `debug'==1 | `debug'==2 {

```

```

sum meatleg needmeat legtype allmeat v_total v_dol extrmeat extrleg, sep(0)
}

* ****
* Redefine Total Vegetables consumption after Meat and Bean intake calcula- *
* tion process *
* ****

gen float vegcup = v_total

* ****
* Redefine Dark Green and Orange Vegetables and Legume consumption *
* after Meat and Bean intake calculation process *
* ****

if `debug'==1 | `debug'==2 {
    sum v_total v_dol
}

gen float dgocup = v_dol

label variable dgocup "cup equiv from drkgr veg, orange veg, & legumes"

* Calculate intake of calories from SoFAAS
* Calculate SoFAAS calories from added sugars, solid fat, and
* alcoholic beverages

```

```

gen float addsugc = 16*add_sug
gen float solfatc = 9*discfat_sol
replace ethcal = 0 if ethcal < 0
replace bwcarbc = 0 if bwcarbc < 0
egen float exfaas = rowtotal(addsugc solfatc ethcal bwcarbc)

label variable addsugc "calories from added sugar"
label variable discfat_sol "gms of discretionary solid fat"
label variable solfatc "calories from solid fat"
label variable ethcal "calories from ethanol"
label variable bwcarbc "carbohydrate calories from beer, wine & sprits"
label variable exfaas "calories from solid fats, alcohol, & added sugars "

```

```

save c:\data9\hei_2005pop.dta, replace

```

```

if `debug'==1 | `debug'==2 {
    sum , sep(0)
    describe , simple
}

```

* ****

* Section (II): Calculation of ratios of mean food group and *

* nutrient intakes to mean energy intakes at the population level. *

* *

* *

* Population ratio Sum weighted population food/nutrient consumption *

```

* HEI-2005      = ----- *
*              Sum weighted total kcal consumption *
*
*
* *****

```

* Convert food group and nutrient intakes in density unit based on HEI-2005

* component scoring standards

```
use c:\data9\hei_2005pop.dta, clear
```

```
gen f_total1000 = 1000*f_total
```

```
gen wholefrt1000 = 1000*wholefrt
```

```
gen vegcup1000 = 1000*vegcup
```

```
gen dgocup1000 = 1000*dgocup
```

```
gen g_total1000 = 1000*g_total
```

```
gen g_whl1000 = 1000*g_whl
```

```
gen d_total1000 = 1000*d_total
```

```
gen allmeat1000 = 1000*allmeat
```

```
gen discfat_oil1000 = 1000*discfat_oil
```

```
gen drxtsfatper = 900*drxtsfat
```

* Change sodium intake in mg to gm then calculate the density per 1000 kcal

```
gen drdtsodigm1000 = 1000*drdtsodi/1000
```

```
gen exfaasper = 100*exfaas
```

```

if `debug'==2 {
    sum f_total1000 wholefrt1000 vegcup1000 dgocup1000 /*
*/ g_total1000 g_whl1000 d_total1000 allmeat1000 /*
*/ discfat_oil1000 drxtsfatper drdtsodigm1000 /*
*/ exfaasper, sep(0)
}

save c:\data9\hei_pop0102.dta, replace

* Population ratio estimates for the HEI-2005 components, age 2+

sort sdmvstra sdmvpsu

if `debug'==2 {
    sum sdmvpsu sdmvstra wt drd1
}

if `debug'==1 | `debug'==2 {
    sum seqn drxtkcal f_total1000 /*
*/ wholefrt1000 vegcup1000 dgocup1000 /*
*/ g_total1000 g_whl1000 d_total1000 /*
*/ allmeat1000 discfat_oil1000 drxtsfatper /*
*/ drdtsodigm1000 exfaasper, sep(0)
}

```

```

svyset sdmvpsu [pweight=wtdrd1], strata(sdmvstra)

sum f_total1000 wholefrt1000 vegcup1000 dgocup1000 /*
*/ g_total1000 g_whl1000 d_total1000 allmeat1000 /*
*/ discfat_oil1000 drxtsfatper drdtsodigm1000 exfaasper , sep(0)

display "#### Running survey ratio command ####"

foreach var of varlist f_total1000 wholefrt1000 vegcup1000 /*
*/ dgocup1000 g_total1000 g_whl1000 d_total1000 allmeat1000 /*
*/ discfat_oil1000 drxtsfatper drdtsodigm1000 exfaasper {
    display "variable is" `var'
    svy: ratio (`var'/drxtkcal)
}

* *****
* Section III: Calculation of HEI-2005 component and total scores at the *
* population level *
* *
* Export output from svy:ratio to popratio0102.xls; calculate the HEI-2005 *
* component and total scores in a different Excel worksheet by copying the *
* output (RHAT and SERHAT) into the yellow highlighted cells in the Excel *
* worksheet, HEI2005_NHANES 2001-2002PopScores.xls. *
* *****

```

* ****

* Section (IV): Calculation of the standard error of the population total *

* HEI-2005 score *

* *

* Create non-truncated, individual HEI-2005 component and total scores; *

* Create a dataset to calculate the standard error for the total HEI-2005 *

* score of a population or a group *

* ****

use c:\data9\hei_both.dta, clear

sort seqn

save c:\data9\hei_hei2005pop_0102se.dta, replace

* ****

* Calculate HEI-2005 TOTAL FRUIT non-truncated component score. Standard *

* for maximum score is ≥ 0.8 cup equiv/1000 kcal; no fruit intake, minimum *

* score is zero. The component score may have a maximum value of 5, 10, or *

* 20 points. *

* ****

* The 12 components are: HEI Min Max *

* *

* Total Fruit 1 0 5 *

* Whole Fruit 2 0 5 *

* Total Vegetables 3 0 5 *

* Dark Green Vegetables, *

* Orange Vegetables,				*
* Legumes,	4	0	5	*
* Total Grains	5	0	5	*
* Whole Grains	6	0	5	*
* Milk	7	0	10	*
* Meat and Beans	8	0	10	*
* Oils	9	0	10	*
* Saturated Fat	10	0	10	*
* Sodium	11	0	10	*
* Calories from Solid Fats,				*
* Alcohol & Added Sugars	12	0	20	*

gen fmax = 0.8*(drxtkcal/1000)

gen hei1 = 5*(f_total/fmax)

replace hei1 = 0 if f_total==0

label variable fmax "maximum standard for Total Fruit component"

label variable hei1 "total fruit"

if `debug'==2 {

sum drxtkcal fmax f_total hei1

}

* Calculate HEI-2005 WHOLE FRUIT non-truncated component score. Standard *
* for maximum score is ≥ 0.4 cup equiv/1000 kcal; no Fruit intake, minimum *
* score is zero. *

* *****

gen njfmax = 0.4*(drxtkcal/1000)

gen hei2 = 5*(wholefrt/njfmax) if njfmax > 0 & njfmax~=.

replace hei2 = 0 if f_total==0

label variable njfmax "maximum standard for Whole Fruit component"

label variable hei2 "whole fruit"

if `debug'==2 {

sum drxtkcal njfmax wholefrt f_total hei2

}

* *****

* Calculate HEI-2005 Total Grains non-truncated component score. Standard *
* for maximum score is ≥ 3.0 cup equiv/1000 kcal; no Total Grains intake, *
* minimum score is zero. *

* *****

gen gmax = 3.0*(drxtkcal/1000)

gen hei5 = 5.0*(g_total/gmax) if gmax > 0 & gmax~=.

label variable gmax "maximum standard for Total Grains component"

label variable hei5 "total grains"

if `debug'==2 {

sum drxtkcal gmax g_total hei5

}

* ****

* Calculate HEI-2005 Whole Grains non-truncated component score. Standard *

* for maximum score is ≥ 1.5 cup equiv/1000 kcal; minimum score is 0. *

* ****

gen wgmax = $1.5 * (\text{drxtkcal} / 1000)$

gen hei6 = $5.0 * (\text{g_whl} / \text{wgmax})$ if wgmax > 0 & wgmax \neq .

label variable wgmax "maximum standard for Whole Grains component"

label variable hei6 "whole grains"

if `debug'==2 {

sum drxtkcal wgmax g_whl hei6

}

* ****

* No reported Total Grains intake, Total Grains and Whole Grains component *

* scores are zero.

*

```
* ****
```

```
replace hei5 = 0 if g_total== 0
```

```
replace hei6 = 0 if g_total== 0
```

```
if `debug'==1 | `debug'==2 {
```

```
    sum g_total hei5 hei6
```

```
}
```

```
* ****
```

```
* Calculate HEI-2005 Milk non-truncated component score. Standard for maxi- *
```

```
* mum score is  $\geq 1.3$  cup equiv/1000 kcal; no Milk intake, minimum score is *
```

```
* zero. *
```

```
* ****
```

```
gen dairymax = 1.3*(drxtkcal/1000)
```

```
gen hei7 = 10.0*(d_total/dairymax)
```

```
label variable dairymax "maximum standard for Milk component"
```

```
label variable hei7 "milk"
```

```
if `debug'==1 | `debug'==2 {
```

```
    sum drxtkcal d_total dairymax hei7
```

```
}
```

```

* *****
* Calculate HEI-2005 Meat and Beans non-truncated component score. Standard *
* for maximum score is >=2.5 oz equiv/1000 kcal; no Meat or Beans intake, *
* minimum score is zero. *
* *****
* Calculate total meat intake from *
* meat, poultry, & fish; egg; nuts & seeds; and soy products *
* *****

```

capture drop allmeat

```
gen allmeat = m_mpf + m_egg + m_nutsd + m_soy
```

```
label variable allmeat "total meat oz equivalents"
```

```
label variable m_mpf "meat, poultry, fish oz equivalents"
```

```
label variable m_egg "oz equivalents of eggs"
```

```
label variable m_nutsd "oz equivalents of nuts and seeds"
```

```
label variable m_soy "oz equivalents of soybean products"
```

```
if `debug'==1 | `debug'==2 {
```

```
sum m_mpf m_egg m_nutsd m_soy allmeat , sep(0)
```

```
}
```

```
* Create the Meat and Beans standard
```

```
gen mbmax = 2.5*(drxtkcal/1000)
```

```
label variable mbmax "max standard for Meat and Beans component"
```

```
* ****
```

```
* Legumes intake calculation; Legumes intake counts as Meat and Beans until *
```

```
* the standard is met, then the rest count as Total Vegetables *
```

```
* ****
```

```
* If total meat intake is more than the Meat and Beans standard then *
```

```
* all Legumes go to Meat and Beans *
```

```
* ****
```

```
gen str8 legtype = ""
```

```
gen meatleg = .
```

```
gen needmeat = .
```

```
gen extrmeat = .
```

```
gen extrleg = .
```

```
gen v_dol = .
```

```
label variable legtype "type of legume intake"
```

```
label variable meatleg "oz equiv of meat from legumes"
```

```
label variable needmeat "oz equiv of meat needed to meet Meat & Beans standard"
```

```
label variable extrmeat "oz equiv needed from legumes to meet Meat & Beans standard"
```

```
label variable extrleg "extra legumes in cup equivalents"
```

```
label variable v_dol "drkgr veg, orange veg, & legume cup equiv"
```

```
gen byte allmeat_chk = cond(allmeat<mbmax, 1, 0)
```

```
gen byte meat_chk = cond(meatleg>needmeat, 1, 0)
```

```
if `debug'==1 | `debug'==2 {
```

```
tab allmeat_chk meat_chk, miss  
sum allmeat mbmax meatleg needmeat, sep(0)  
}
```

```
replace meatleg = 4*legumes if allmeat_chk==1  
replace needmeat = mbmax-allmeat if allmeat_chk==1  
replace legtype = "allmeat" if meat_chk==0 & allmeat_chk==1  
replace allmeat = allmeat + meatleg if meat_chk==0 & allmeat_chk==1  
replace v_total = v_total if meat_chk==0 & allmeat_chk==1  
replace v_dol = v_orange + v_drkgr if meat_chk==0 & allmeat_chk==1
```

```
replace allmeat_chk = cond(allmeat<mbmax, 1, 0)  
replace meat_chk = cond(meatleg>needmeat, 1, 0)  
if `debug'==1 | `debug'==2 {  
tab allmeat_chk meat_chk, miss  
  
sum allmeat mbmax meatleg needmeat, sep(0)  
}
```

```
replace legtype = "meat/veg" if allmeat_chk==1 & meat_chk==1  
replace extrmeat = meatleg - needmeat if allmeat_chk==1 & meat_chk==1  
replace extrleg = extrmeat/4 if allmeat_chk==1 & meat_chk==1  
replace allmeat = allmeat + needmeat if allmeat_chk==1 & meat_chk==1  
replace v_total = v_total + extrleg if allmeat_chk==1 & meat_chk==1  
replace v_dol = v_orange + v_drkgr + extrleg if allmeat_chk==1 & meat_chk==1
```

```
replace allmeat_chk = cond(allmeat<mbmax, 1, 0)
replace meat_chk = cond(meatleg>needmeat, 1, 0)
if `debug'==1 | `debug'==2 {
    tab allmeat_chk meat_chk, miss
    sum allmeat mbmax meatleg needmeat, sep(0)
}
```

```
replace legtype = "allveg" if allmeat_chk==0
replace v_total = v_total + legumes if allmeat_chk==0
replace v_dol = v_orange + v_drkgr + legumes if allmeat_chk==0
```

```
replace allmeat_chk = cond(allmeat<mbmax, 1, 0)
replace meat_chk = cond(meatleg>needmeat, 1, 0)
if `debug'==1 | `debug'==2 {
    tab allmeat_chk meat_chk, miss
    sum allmeat mbmax meatleg needmeat, sep(0)
}
```

```
gen hei8=10*(allmeat/mbmax) if mbmax > 0 & mbmax~=.
```

```
label variable hei8 "meat and beans"
```

```
if `debug' ==2 {
    sum allmeat mbmax hei8
}
```

* ****

* Calculate HEI-2005 Total Vegetable non-truncated component score. *

* Standard for maximum score is ≥ 1.1 cup equiv/1000 kcal. No Vegetable *

* intake, minimum score is zero. *

* ****

gen vmax = 1.1*(drxtkcal/1000)

gen vegcup = v_total

gen hei3=5.0*(vegcup/vmax) if vmax > 0 & vmax~=. .

label variable vmax "max standard for Total Vegetables component"

label variable vegcup "cup equivalents from total vegetables"

label variable hei3 "total vegetables"

if `debug' ==2 {

sum drxtkcal vmax v_total vegcup

}

* ****

* Calculate HEI-2005 Dark Green and Orange Vegetables and Legumes *

* non-truncated component score *

* ****

* Standard for maximum score is ≥ 0.4 cup equiv/1000 kcal

```
gen dgmax = 0.4*(drxtkcal/1000)
```

```
gen dgocup = v_dol
```

```
gen hei4 = 5.0*(dgocup/dgmax) if dgmax > 0 & dgmax~=.
```

```
label variable dgmax "max for Total DrkGr Veg, Orange Veg, & Legumes comp"
```

```
label variable dgocup "cup equiv from DrkGr Veg, Orange Veg, & Legumes"
```

```
label variable hei4 "vegetables and legumes"
```

```
if `debug' ==2 {
```

```
    sum drxtkcal dgmax v_dol dgocup hei4
```

```
}
```

```
* ****
```

```
* Calculate HEI-2005 Oils non-truncated component score *
```

```
* Standard for maximum score is >=12 grams/1000 kcal *
```

```
* ****
```

```
gen oilmax = 12.0*(drxtkcal/1000)
```

```
gen hei9 = 10.0*(discfat_oil/oilmax) if oilmax > 0 & oilmax~=.
```

```
label variable oilmax "maximum standard for Oil component"
```

```
label variable hei9 "oils"
```

```
if `debug' ==2 {
```

```
    sum drxtkcal oilmax discfat_oil hei9
```

```
}
```

* ****

* Linear extrapolation for Saturated fat & Sodium non-truncated component *

* scores: Individuals with truncated score of >0 and <8 receive the same *

* score; Individuals with truncated score of >=8 and 0 will receive a *

* linear extrapolated score based on the scores of 0-8 *

* *

* Calculate HEI-2005 Saturated Fat non-truncated component score *

* Standards for score of 8 is 10% of total kcal and score of zero is >=15% *

* *

* Calculate percent of calories from Saturated Fat *

* ****

gen pctsfat=(100*(9*drxtsfat))/drxtkcal if (drxtkcal > 0 & drxtkcal~=.)

label variable pctsfat "% calories from saturated fat"

* Truncated score is >0 and <8 for Saturated fat intake of 10%-15% of total kcal

gen hei10 = .

replace hei10 = (8 - (8*(pctsfat-10)/5)) if (pctsfat>10.0 & pctsfat<15.0)

label variable hei10 "saturated fat"

* hei10 is not calculated if the logical condition on percent of calories

* from saturated fat is not met (if (pctsfat>10.0 & pctsfat<15.0)).

```
if `debug'==2 {
```

```
    sum pctsfat hei10
```

```
    sum pctsfat if hei10==.
```

```
    count if pctsfat<= 10.0
```

```
    count if pctsfat>= 15.0
```

```
}
```

```
* ****
```

```
* Calculate HEI-2005 Sodium non-truncated component score *
```

```
* Standards for score of 8 is 1.1 grams/1000 kcal and score of zero is *
```

```
* >=2.0 grams/1000 kcal *
```

```
* ****
```

```
* Calculate the sodium density
```

```
gen sodiden=(drdtsodi*1000)/drxtkcal if drxtkcal > 0 & drxtkcal~=.
```

```
label variable sodiden "sodium density"
```

```
* Truncated score is >0 and <8 for Sodium intake of 1.1g - 2g/1000 kcal
```

```
gen hei11 = 8 - (8 *(sodiden-1100)/(2000-1100)) if (sodiden>1100 & sodiden<2000)
```

```
label variable hei11 "sodium"
```

```
if `debug'==2 {
```

```
    sum drdtsodi drxtkcal hei11
```

```
}
```

* *****

* Calculate HEI-2005 calories from SoFAAS non-truncated component score *

* Standard for maximum score is <=20% total kcal, Maximum score is 20; *

* >=50% total kcal, minimum score is zero *

* *****

* Calculate SoFAAS calories from Added sugars, Alcohol, and Solid fats

* Calories from added sugars

gen addsugc = 16.0*add_sug

sum add_sug addsugc if `debug'==1 | `debug'==2

* Calories from solid fat

gen solfatc = 9.0*discfat_sol

sum discfat_sol solfatc if `debug'==1 | `debug'==2

replace ethcal = 0 if ethcal < 0 | ethcal==.

replace bwcarb = 0 if bwcarb < 0 | bwcarb==.

* Total SoFAAS calories as in kcal

gen exfaas = addsugc + solfatc + ethcal + bwcarb

sum addsugc solfatc ethcal bwcarb exfaas drxtkcal if `debug'==1 | `debug'==2 , sep(0)

gen alcoholc = ethcal + bwcarb

gen addsugc_perc = (100*addsugc)/drxtkcal

```
gen alcoholc_perc = (100*alcoholc)/drxtkcal
```

```
gen solfatc_perc = (100*solfatc)/drxtkcal
```

```
* Calculate SoFAAS as in %kcal*
```

```
gen sofa_perc=(exfaas*100)/drxtkcal if drxtkcal > 0 & drxtkcal~=.
```

```
gen sofamin = 20
```

```
gen sofamax = 50
```

```
gen hei12 = 20 - (20*(sofa_perc-sofamin)/(sofamax-sofamin))
```

```
if `debug'==1 | `debug'==2 {
```

```
    sum sofa_perc hei12 addsugc addsugc discfat_sol solfatc /*
```

```
*/ ethcal bwcarb exfaas alcoholc addsugc_perc /*
```

```
*/ alcoholc_perc solfatc_perc , sep(0)
```

```
}
```

```
* ****
```

```
* Calculate linear equations for Saturated fat and Sodium components based *
```

```
* on the individuals with HEI-2005 component score of >0 and <8, then *
```

```
* extrapolate the equations to the individuals with score >=8 or score <=0 *
```

```
* ****
```

```
save c:\data9\hei_hei2005pop_0102se.dta, replace
```

```
* ****
```

```

* These two drop statements were explicit in the SAS program. They are not *
* needed in this Stata translation because Stata, drops cases with missing *
* data. *
* *****
* drop if hei10==.
* drop if hei11==.
* *****
* The SAS code fit an OLS regression and did not account for the complex *
* sample survey design. The statements were changed to simple summarize *
* commands. Doing so does not effect the calculation of the desired ratios.*
* *****
sum hei10 pctsfat
sum hei11 sodiden

* Apply the linear extrapolation of the equations to individuals with scores
* >=8 or score <=0

use c:\data9\hei_hei2005pop_0102se.dta, clear

replace hei10 = -1.60000*(pctsfat) + 24.00000 if (pctsfat<=10 | pctsfat>=15)
replace hei11 = -0.00889*(sodiden) + 17.77778 if (sodiden<=1100 | sodiden>=2000)

regress hei10 pctsfat
regress hei11 sodiden

```

compress

sum hei1 hei2 hei3 hei4 hei5 hei6 /*

*/ hei7 hei8 hei9 hei10 hei11 hei12 if `debug'==1 | `debug'==2 , sep(0)

order seqn sddsrstyr wttrd1 sdmvpsu sdmvstra sddsrstyr riagendr ridageyr /*

/ hei in_*

save c:\data9\hei_hei2005pop_0102se_notrunnew.dta, replace

* ****

* Calculate the HEI-2005 total score. At the individual level, the total *

* HEI-2005 non-truncated score is the sum of the 12 non-truncated component *

* scores.

*

*

*

* hei1 "total fruit"

*

* hei2 "whole fruit"

*

* hei3 "total vegetables"

*

* hei4 "vegetables and legumes"

*

* hei5 "total7 grains"

*

* hei6 "whole grains"

*

* hei7 "milk"

*

* hei8 "meat and beans"

*

* hei9 "oils"

*

```

*      hei10 "saturated fat"                *
*      hei11 "sodium"                      *
*      hei12 "calories from SFAAS"         *
* *****

```

```
egen hei2005new = rowtotal(hei1 hei2 hei3 hei4 hei5 hei6 hei7 hei8 hei9 hei10 hei11 hei12)
```

```
label variable hei2005new "Total individual's non-truncated HEI-2005"
```

```
save c:\data9\hei_hei2005pop_0102se_notrunnew.dta, replace
```

```

* *****
*
* Create population total to estimate the standard error of HEI-2005 total *
* score for the population:                *
*
*      Population total = individual's total HEI-2005 non-truncated score *
*      x individual's total energy intake *
*
* *****

```

```
* use c:\data9\hei_hei2005pop_0102se_notrunnew.dta, clear
```

```
gen hei2005kcal=hei2005new*drxtkcal
```

```
save c:\data9\hei_pop0102seNEWpopwt.dta, replace
```

```
sort sdmvstra sdmvpsu
```

```
svyset sdmvpsu [pweight=wtdrd1], strata(sdmvstra)
```

```
* ****
```

```
* HEI-2005 non-truncated total score at the individual level; Linear *
```

```
* extrapolation of LE 0 and GE8 based on <0 and >8 for Saturated Fat and *
```

```
* Sodium components, Age 2+ *
```

```
* ****
```

```
svy: ratio (ratio_1: hei2005kcal/drxtkcal)
```

```
* ****
```

```
* Copy standard error calculated in Stata into the pink highlighted cell in *
```

```
* HEI2005_NHANES 2001-2002PopScores.xls Excel worksheet to obtain the 95% *
```

```
* confidence interval for the population total HEI-2005 score *
```

```
* ****
```

```
capture erase c:\data9\hei_mped.dta
```

```
capture erase c:\data9\hei_wjfrt.dta
```

```
capture erase c:\data9\hei_food.dta
```

```
capture erase c:\data9\hei_nutrient.dta
```

```
capture erase c:\data9\hei_demo.dta
```

```
capture erase c:\data9\hei_pyr.dta
```

```
capture erase c:\data9\hei_fdpyr.dta
```

```
capture erase c:\data9\hei_pyrcalc.dta
```

```
capture erase c:\data9\hei_bwlnt.dta
capture erase c:\data9\hei_bwlpyr.dta
capture erase c:\data9\hei_beerwineliq.dta
capture erase c:\data9\hei_both.dta
capture erase c:\data9\hei_pop0102.dta
capture erase c:\data9\hei_hei2005pop0102.dta
capture erase c:\data9\hei_2005pop.dta
capture erase c:\data9\hei_2005pop_0102se.dta
capture erase c:\data9\hei_slop.dta
capture erase c:\data9\hei_hei2005pop_0102se.dta
capture erase c:\data9\hei_hei2005pop_0102se_notrunnew.dta
capture erase c:\data9\hei_pop0102seNEWpopwt.dta
```

```
exit
```

```
exit
```

```
* ****
```

```
*          Variable labels          *
```

```
* ****
```

```
label variable a_bev      "drinks of alcohol"
```

```
label variable add_sug    "teaspoons of added sugars food group"
```

```
label variable addsugc    "calories from added sugar"
```

```
label variable addsugc_perc "% calories from added sugars"
```

label variable alcoholc "calories from ethanol in alcoholic beverages"

label variable alcoholc_perc "% calories from ethanol in alcoholic beverages"

label variable allmeat "total meat ounce equivalents"

label variable bwcarb "carbohydrate calories from beer, wine & sprits"

label variable d_cheese "cheese cup equivalents"

label variable d_milk "milk cup equivalents"

label variable d_total "dairy cup equivalents"

label variable d_yogurt "yogurt cup equivalents"

label variable dairymax "max. standard for Total Milk component"

label variable dgmax "max. standrd. for Total Dark Green+Orange Vegett & Legumes"

label variable dgocup "cup equivalents from dark green + orange vegetables, & legumes"

label variable discfat_oil "gms of discretionary oil"

label variable discfat_solid "gms of discretionary solid fat"

label variable drddrstz "reliable dietary data"

label variable drdifdcd "8-digit USDA food code"

label variable drdtsodi "total sodium consumed per day"

label variable drxialco "intake (alcohol)"

label variable drxicarb "intake (carbohydrate)"

label variable drxichol "intake (cholesterol)"

label variable drxifibe "intake (fiber)"

label variable drxigrms "grams of food consumed"

label variable drxikcal "intake (kilocalories)"

label variable drxiline "food intake data line"

label variable drximfat "intake (monounsaturated fat)"

label variable drxipfat "intake (polyunsaturated fat)"

label variable	drxiprot	"intake (protein)"
label variable	drxisfat	"intake (saturated fat)"
label variable	drxisugr	"intake (sugar)"
label variable	drxitfat	"intake (total fat)"
label variable	drxtalco	"total alcohol consumed per day"
label variable	drxtcarb	"total carbohydrate consumed per day"
label variable	drxtkcal	"total energy consumed per day"
label variable	drxtsfat	"total saturated fat consumed per day"
label variable	drxtsfatper	"percent of calories from saturated fat"
label variable	equivflag	"food code MyPyramid equivalents data"
label variable	ethcal	"calories from ethanol"
label variable	exfaas	"calories from solid fats, alcohol and added sugars"
label variable	exfaasper	"percent of calories from solid fats, alcohol, and added sugars"
label variable	extrleg	"extra legumes in cup equivalents"
label variable	extrmeat	"oz equiv. needed from legumes to meet Meat and Beans standard"
label variable	f_citmlb	"citrus, melon, & berries cup equivalents"
label variable	f_other	"other fruit cup equivalents"
label variable	f_total	"fruit cup equivalents"
label variable	fmax	"maximum standard for Total Fruit component"
label variable	foodcode	"8-digit USDA food code"
label variable	foodname	"food description"
label variable	frtjuice	"fruit juice cup equivalents"
label variable	g_nwhl	"oz equivalents total nonwhole grains"
label variable	g_total	"oz equivalents total grain group"
label variable	g_whl	"oz equivalents whole grains"

label variable	gmax	"maximum standard for Total Grains component"
label variable	legtype	"type of legume intake"
label variable	legumes	"cup equivalents of cooked dry beans & peas"
label variable	m_egg	"oz equivalents of eggs"
label variable	m_fish_hi	"oz cooked seafood (high omega-3)"
label variable	m_fish_lo	"oz cooked seafood (low omega-3)"
label variable	m_frank	"oz cooked lean sausage meats"
label variable	m_meat	"oz cooked lean meat"
label variable	m_mpf	"meat, poultry, fish oz equivalents"
label variable	m_nutsd	"oz equivalents from nuts and seeds"
label variable	m_organ	"oz cooked lean organ meats"
label variable	m_poult	"oz cooked poultry meat"
label variable	m_soy	"oz equivalents from soybean products"
label variable	mbmax	"max standard for Meat and Beans component"
label variable	meatleg	"oz equivalents of meat from legumes"
label variable	modcode	"modification code"
label variable	needmeat	"oz equivalents of meat needed to meet Meat and Beans standard"
label variable	njfmax	"maximum standard for Whole Fruit component"
label variable	noscarb	"carbohydrate from non-added-sugar"
label variable	oilmax	"maximum standard for Oil component"
label variable	pctsfat	"% calories from saturated fat"
label variable	sofa_perc	"% calories from solid fats, alcohol, and added sugars"
label variable	sofamax	"maximum standard for SoFAAS component"
label variable	sofamin	"minimum standard for SoFAAS component"
label variable	sofatc	"calories from solid fat"

label variable solfatc_perc "% calories from solid fat"
label variable suggram "grams of sugar"
label variable threed "beer, wine & distilled sprits"
label variable v_dol "dark green and orange vegetables, & legume cup equivalents"
label variable v_drkgr "dark green vegetable cup equivalents"
label variable v_orange "orange vegetable cup equivalents"
label variable v_other "other vegetable equivalents"
label variable v_potato "potato cup equivalents"
label variable v_starcy "starchy vegetable cup equivalents"
label variable v_tomato "tomato cup equivalents"
label variable v_total "total vegetable cup equivalents"
label variable vegcup "cup equivalents from total vegetables"
label variable vmax "maximum standard for Total Vegetables component"
label variable wgmax "maximum standard for Whole Grains component"
label variable wholefrt "whole fruit cup equivalents"