Preparing your computer and Stata

Before *Session 1* Mon 27th June

Please follow the advice below to prepare for the computer exercises during the course. We are assuming you have (or will have) Stata installed on your computer. These instructions are simply to set up your work environment so that you are ready to begin: the specific commands needed to do any exercise will be provided with the exercise.

Organisation of Course Material

Create a folder called "Beyond_Classic" on the desktop of your computer (or elsewhere on the hard disk if you prefer) where you will download the various materials provided for the course. Create three sub-folders of "Beyond_Classic", named "lectures", "exercises" and "reading". For the "exercises" folder, create three "daughter" folders named "data", "docs" and "solutions". Prior to the course, all data sets will be available for you to save in the "data" folder, and prior to each session of the course, the instructions and solutions (including code) will be provided in a Zip package organised in this folder structure.

Once you open Stata, you should set the working folder to "data", using the menus or a command such as:

```
cd "c:/Documents/…./Beyond_Classic/exercises/data/"
```

Some of the exercises will require you to create and save temporary data sets. We advise that you always name such files with a prefix "temp" (e.g. "tempExercise1.dta", "tempFramingham.dta) so that you can clean up the data folder when the exercise is completed.

A note on Stata commands, functions and packages

Stata comes with a wide range of built-in commands, but there are also many user written functions for performing specific tasks. Groups of related functions are stored in packages, and a function or package only become available when it is installed. Functions are stored in "ADO" files, and the command "adopath" will tell you where Stata keeps all it's own commands and other commands that you have installed:

```
. adopath
  [1]  (BASE)        "/Applications/Stata/ado/base/"
  [2]  (PERSONAL)    "/Users/marrei/Documents/Stata/ado/personal/"
  ….
```

When you download a package or function from the internet, the ADO file will be automatically stored in the appropriate folder, but we will also provide you with some "personal" ADO files.

If you get a warning from Stata like "command xxx is unrecognized", this usually means that the specific command has not been downloaded yet and you should download it by yourself. `ssc` allows you to easily download a Stata command. For example:

```
ssc install ccmatch
```

will install the command for selecting a matched case-control sample, if you do not already have it.

For a package, type `findit "package name"` and click on install. For example,

- **(i)** `findit meanscor` will direct you to **sg156**, from where you can get this package for using in Exercise 4.1. Click the installation files and click to install the ancillary files. When done, typing **`help meanscor`** gives detailed help
- **(ii)** `findit optimal` offer a list of packages, and you can scroll down to `sxd2` and click on `click here to install` to get the commands we will use in Exercise 4.2 for optimal sampling.

For any Stata command (whether automatically available in Stata or installed by you), you can ask for help:

```
help sttocc   (available in Stata)
help ccmatch  (after installing as above)
```

For each exercise during the course, you will be provided with the Stata code that produces the solutions. A summary of the commands most used is provided in the following pages and may be useful to you later.

FINDING HELP

findit xxxx

findit is the best way to search for information on a topic across all sources, including the system help, the FAQs at the Stata website, the Stata Journal, and all Stata-related Internet sources including user-written additions. For a Stata package, you can type *findit* "package name"and click install, as on page 1.

DATA INPUT/OUTPUT

Input	Output
For a Stata dataset (i.e. auto.dta):	To save a Stata data file (xxx.dta):
`use auto.dta, clear`	`save mydata, replace`
For importing other types of data (e.g., xlsx.):	Exporting as other file types, e.g. Excel
`import excel auto.xls, firstrow`	`export excel mydata.xlsx, replace`

DATA MANIPULATION

expand (expands aggregated dataset to individual records)

duplicates (*reports, displays, lists, tags, or drops* duplicate observations, depending on the subcommand specified. Duplicates are observations with identical values either on all variables or on a specified variable list).

sort (arranges the observations of the current data in order based on the values of a specified variables).

merge (joins corresponding observations from the data set currently in memory, called the master dataset, with those from the .dta file specified)

append (appends a specified .dta file to the end of the data set currently in memory.

set seed (specifies the initial value of the random-number seed used by the random-number functions, such as *runiform()*.

runiform() generates a random number in the interval (0,1).

SIMPLE CALCULATIONS FOR CATEGORICAL DATA

cc (calculates crude odds ratio (OR) and Mantel-Haenszel OR from individual data. For aggregate data, you can specify the option "*fw*=".

cci ("immediate" form of cc, allows you to enter the four numbers of 2x2 table).

csi ("immediate" calculation of the risk ratio from the four numbers of a 2x2 table.

tab (produces a two-way table of frequency counts, along with various measures of association, including chi-square test)

tabodds (performs an approximate chi-squared test of homogeneity of odds and a test for linear trend of the log-odds.

SPECIAL COMMANDS FOR SURVIVAL ANALYSIS

stset (before any survival analysis, you must use *stset* to declare the survival data).

stcox (runs a Cox regression on data that has been "*stset)*".

For weighted Cox regression, *stset* should be run with the weights as a "*pweights*" option, and then *stcox* with the "*vce(robust)*" option to get correct standard errors.

sts generate (creates new variables containing the estimated survivor and failure functions, the cumulative hazard, and other related functions. You must "*stset*" your data before using *sts generate*.)

sts graph (graphs the estimated Kaplan–Meier survivor function).

sts test *(*tests the equality of survivor functions across two or more groups)

FUNCTIONS FOR EPI ANALYSIS

CASE-CONTROL AND CASE-COHORT

ccmatch (matches cases and controls on specified variables).

sttocc (samples a nested case-control study. Before using the *sttocc* command, you need to set the time and failure variables using *stset*. Remember to also set a seed.

stcascoh & stselpre (these commands, from the *sbe41 package* , will select and analyse a case-cohort dataset. The *stselpre* command can only be run following *stcascoh* (which prepares the data in the format expected by *stselpre*).

TWO-STAGE DATA

meanscor & coding

These commands, from the *meanscor* package are used for analyzing two-stage data. The coding function orders the strata formed by different levels of the dependent variable and first stage covariates. Example of getting the different strata formed by levels of dependent variable (mort) and first stage covariates (sex) has been given:

```
coding  mort  sex
```

With the *meanscor* command, the user specifies: the logistic model, first stage variables defining the sampling strata, and the second stage variable(s) in the model:

```
meanscor  y  gonn-chlam,first(gonn-sexptn)  second(chlam)
```

OPTIMAL SAMPLING OF TWO-STAGE DATA

optfixn, optbud & optprec

The *sxd2 package* has several useful commands (***optfixn, optbud & optprec***) of computing optimal sampling designs for two-stage studies.

The ***optfixn*** function calculates the optimal sampling fraction in each of the strata at the second stage, for situations where first stage data is available and the total number of second-stage observations has been decided. An example has been given:

Before running the optfixn function you should run the *coding* function:

```
coding mort sex wtcat
```

then enter first stage sample sizes in the order suggested by *coding* function:

```
matrix fstsamp=(6666, 1228, 144, 58)'
```

and the following command produces the sampling fractions:

```
optfixn mort sex age,first(sex) n1(fstsamp) n2(1000) var(3)
```

The ***optbud*** function calculates the total number of study observations and the second-stage sampling fractions that will maximize precision subject to an available budget. The user must also supply the unit cost of observations at the first and second stage and the vector of prevalences in each of the strata defined by different levels of dependent variable and first stage covariates. An example has been given:

Before running the *optbud* function you should run the *coding* function and then enter the prevalences in the order suggested by *coding* function:

```
matrix prev=(0.02,.134,.670,.054,.05,.047,.001,.004,.013,.002,.003,.002)'
```

and the computation:

```
optbud mort sex-surg,first(sex wtcat) prev(prev) var(7) b(10000) c1(2) c2(15)
```

The ***optprec*** function calculates the total number of study observations and the second-stage sampling fractions that will minimize the study cost subject to a fixed variance (precision) for a specified variable. The user must also supply the unit cost of observations at the first and second stage and the vector of prevalences in each of the strata defined by dependent variable and first stage covariates. As above for optbud, you first run the *coding* function, then enter the prevalences in the order suggested by *coding* function and then the computation:

```
optprec mort sex-surg,first(sex wtcat) prev(prev) var(7) prec(.00038298) c1(2) c2(15)
```

KAPLAN-MEIER TYPE WEIGHTS

cohort_km_weights & sampled_km_weights

The purpose-written Stata commands ***cohort_km_weights*** and ***sampled_km_weights*** compute the inverse probability of being sampled as a control in a nested case-control study, which is known as the Kaplan-Meier type weight.

The command ***cohort_km_weights*** computes the weights from the individual records for the members of a cohort, example:

```
cohort_km_weights , controls(2) id(uniqueid) time(exit) failure(event)
```

Stratified example:

```
cohort_km_weights gender age, controls(3) id(pin) entry(start) time(end) failure(case)
```

The command ***sampled_km_weights*** computes the weights from summary data containing event times with risk set sizes and numbers of failures, example:

```
sampled_km_weights using riskset_estimates, controls(2) id(id)
>time(exit) failure(event) riskset(n_risk) numfail(n_ties)
```

Stratified example:

```
sampled_km_weights gender age using riskset_estimates, controls(3) id(id)
>time(exit) failure(event) riskset(n_risk) numfail(n_ties)
```

The user runs the appropriate command depending on the data available, and both commands call an "invisible" sub-function ***calculate_km_weights*** to do the calculations.

Thus before any application, these 3 functions must be made available to Stata, by placing them (together with the help files *cohort_km_weights.hlp* and *sampled_km_weights.hlp*) in the appropriate *ADO* folders. The *ADO* folder used by Stata for personal or site-specific functions, can be identified by typing "*adopath*" in the Stata command window. Once the *ADO* and *HLP* files have been placed in the *ADO* folders, Stata needs to be restarted to find them.