ARDx capital stock: data manager guide

Contents

1. Overview

1.1 How to read this guide

This guide is intended for those managing the ARDx capital stock code. It should be read in conjunction with the User Guide which explains the options more simply. This document details the specific effects of those options. Users may find this guide sheds light on some of the comments in the User Guide.

This guide also discusses some characteristics of the code. Again these are also discussed in the User guide, which is concerned about the inferences the user can safely draw. This guide shows how the assumptions made or options chosen affect the coding.

1.2 Code structure

The code is broadly split into three parts

- Setting up the options
- Creating the 'capstock input' file
- Creating the capstock file

Creating the capstock input file is a slow process, and should not need to be run very often – only if an additional year of data is available, if a different aggregate data file is to be used, or if one of the imputation methods is changed (imputation for missing observations or missing financial data). The end user should not normally need to change these. It could be helpful for a manager to create all the combination of base input files (at the moment, this is only two) for so that the user does not need to be concerned with this stage at all.

1.3 Data needed

The program needs data from three sources

- Aggregate capex and capstock data at industry/year level
- ARDx financial information – this is where observed capex and the share variables come from
- ARDx Register Panel, either in IDBR-imputed or interpolated form – this creates the analysis frame of all RUs in all years; we also need employment info as the same firms aren't sampled for employment/financial info in the Abs/BRES period (2009 onwards).

++++++++++++++++++ from old notes

2. Section 1: set-up

Various macros to run the code are set up, along with log files, data files and folders.

ONS aggregates define 10 types of capex, but the ABS files only seem to have three (buildings, vehicles, and something which we'll call 'other'), so we will use those three. We set up a macro listing the three capex types so we can loop through them later.

The relevant capex variables are relabelled a capex_b, capex_v and capex_other to make the code more readable.

We also define friendly names for other variables we will use later: turnover from ABI/ABS data, turnover registered by the IDBR, and total purchases.

Finally, we set up the variables to be used for weighting and the weighting method (at the moment, totpurch and employmentweighted are the only options)

3. Section 2: create base files

This section creates the files to be used in the analysis. We need data from all years in a single file, as we are going to interpolate. Five files are created:

'capstock deflators': deflators for various types of capstock (including the three actually used). This is created from an ONS spreadsheet. Note that 'other' is created in the spreadsheet before running the code

'aggregate capstock': aggregated capital stock, by type of industry and asset, real and nominal. Again, data is taken from ONS standard data tables and 'other' is created before running the code

'aggregate capex': aggregate investment by asset and industry, taken from ONS spreadsheets. These spreadsheets are problematic, with much of the information removed for confidentiality or quality reasons, and so we need to calculate our own industry capex totals.

'returned_base': this is all the financial information returned, all rurefs, all years, extracted from the ARDx returned data and reflecting the choices above abut which vars are to be used

'capstock_base': this is the register panel (interpolated according to the global macros), with returned financial information added. It is the subset of firms that returned financial information AT ANY POINT in the period. Note that the previous code claimed to delete those without any financial obs, but in fact deleted only those without two consecutive observations.

Notes: the old capstock code claimed to remove all rurefs with no valid obs, but in fact it seemed to only remove those that didn't have two consecutive observations. Also, shouldn't we perhaps take employment from the IDBR/BSD enterprise data, particularly if there is only one RU?

4. Section 3: financials

4.1 Stage 1: weights

Create the weights for capex imputation, held in the non-method-specific weights capex_weight. This is what will be used to allocate capex for non-observed years. The methods so far identifed are:

1. Employment-weighted: calculate average capex/employee in observed years, apply to non-observed years

Employment-weighted method

The rationale for this method is that employment is generally observed, and so capex/employee when both are observed may be taken as a reasonable proxy when only employment is observed.

The advantage of this method is that it makes use of observed information about non-respondents, and this info is generally non-missing.

The disadvantages are:

- Capex/employment don't necessarily go hand-in-hand eg when setting up a business investment per employee is typically much greater; and when a business is shrinking there might be net disposals per employee

- The link with employment potentially could bias statistical analyses using employment as an explanatory variable alongside capital stock – not aware of any work on this but we should be aware of the possibility
- Many firms have zero employment

What to do about zero employment?

This could reflect setting up the company (in which case there is still some investment), unrecorded working proprietors or other payments. We follow the path of the previous code in replace values with an average; the rationale for this (I believe) is that PIT estimates may not accurately reflect change in employment (particularly in the out-of-scope obs). In our register panel employment is always zero or positive, and so we do this not for missing values (in the old code this equated to 'non-selected') but for zero obs. We use log-linear imputation to ensure that employment is always positive.

To do this we need to ensure determine what is the minimum 'employment' for capital weighting purposes that should be associated with an existing RU with zero observed employment. This is set in a local macro at the beginning of the program called 'min_emp_count'. This must be a positive number.

The imputed employment weights are held in the variable capex_weight.

Alternative methods

None at present

4.2 Stage 2: Work out average capex per employee and impute for missing obs

Capex isn't available for all years, so this stage works out the average capex for each weighting unit over the whole period of existence of the RU. It does this by

- Summing the capex
- Summing the weights for those periods when we also have capex
- Dividing one by the other to get observed capex by unit of weight
- Allocating capex for non-observed years by multiplying the weight by the average

Note that weights must be non-missing.

How do we deal with negative capex (ie net disposal)? These should be exceptional items. There are three options:

a. ignore years where capex is negative - this means we are likely to overestimate capex, as we are not treating small disposals in the same way as large ones, and acquisition the same as disposal
b. exclude the capex series but only if the overall total is negative (ie allow for net disposal in specific years); this means only that truly exception capex is ignored (but does potentially lead to some firms having no capex imputed because of one very large negative value)
c. ignore at the capex imputation stage, only correct at the capstock stage

We follow option (c) – keeping everything in for now, removing later only if the entire capstock is negative. Note that negative capex is observed at the aggregate level for "buildings" in section O (public administration) in every year ie this sector have been permanently selling stuff off since 1997 (although the aggregate cap stocks don't support this story…). This sector will need massive opening stocks to avoid negative capstocks.

Finally, calculate a value for the proportion of imputed obs – allows one to drop RUs with too few obs. In all the future calculations, we drop the cases failing the imputation limit.

4.3 Stage 3: Add aggregate stocks and get grossing adjustment

Aggregate capex and capstock data is added into the dataset. The capstock data will be used to provide the starting values for capstocks when firms are observed; the capex data is used to rescale the capstock data to the observed firms.

The code works whether in current prices or constant price. Both of those are in the aggregate data, and are read in as necessary. If constant prices are used, than the individual capex values are deflated. Note that the key variable used for shares is not deflated, as only the relative share is important.

All of the firms which are out-of-scope (and a very small number of firms in early years) will not have aggregate data associate with them, as we don't know which sector they should be in. Hence, we linearly interpolate these values, on the basis that this firm is moving between two industries. This might not be a valid assumption – the reason firms might change SIC classification is through takeover, merger or divestment, which would be associated with a big jump from one group to another; but we don't have the information to tell us.

At this point we have created the 'merged data'. This takes a considerable amount of time to run, and so is a natural break for efficiency in development. The remaining code is quick and amenable to fiddling.

We only observe a part of the total capex/capstock. If we just allocated aggregate capstock to firms, then we would overestimates, as we would be implying the cap stock for non-respondent firms is zero.

To allow for this, we calculate the share of observed/imputed capex relative to the total (all by industry and asset type), and then use this to create an adjustment factor for capital stock. In doing so, we are assuming that the relationship between investment and capital stocks is constant, overall.

Problem: what if industry level capex is negative? Meaningless to allocate a 'share'.

Problem: what if employment is zero in year 1? We use the capex_weight rather than employment directly.

[We make this adjustment only for those firms which have a tolerable level of imputation – over-imputed firms play no part in this calculation. The new code drops all the obs which do not meet the 'tolerance' level. The old code kept everything in, but created many missing values and didn't seem to use those obs.]

To create the cap stock, we first create a mechanism for allocating initial capital stocks. These are taken as a share of the industry level stocks, allocated in the first year of a firm's existence by a combination of key variable thought to be correlated with capitals stock (such as turnover) and employment. The three mechanisms for calculating shares currently offered are

- Simple: key variable as a proportion of industry (letter-level) total
- Key_share: key variable as a proportion of 3-digit industry total, with 3-digit industries weighted by the proportional contribution of employment in that 3-digit industry to total employment in that letter class

- Emp_share: employment as a proportion of 3-digit industry total, with 3-digit industries weighted by the proportional contribution of the key variable in that 3-digit industry to the letter-level total

In the new coding, employment is replaced by capex_weight. This is (currently) equal to employment but with zero employment changed to an arbitrary small value, so that even start-ups and microbusinesses will have some share of industry capital stocks.

Having got to this stage, creating cap stocks is simple

If this is the first time the firm is observed, give it a share of aggregate capital stocks based upon the calculations above, and add investment in the first year.

If not, then take the previous year's capstock, take off deprecation and add on this year's capex (real or imputed).

++++++++++++++++++++++++ end

5. Appendix 1 Detailed code breakdown – 2011 code

Code from old code "capital_stock_BG_1305.do", last edited GA 2012	Meaning and notes
Set-up	

```
*************************************************************************
*CAPITAL STOCK VERSION 2.2x
*Bob Gilhooly's re-run of the Capital Stock data set (2.1x) => updates VICS2 file
*an improved version of the previous files (negative capital stock series removed)
*For a full methodological breakdown users are advised to examine the latest
*capital stock guide and this .do file

*This .do file has been designed to allow researchers to easily create their own
* capital stock data sets with different depreciation rates, key variables and
* values for the calcRatio. All that is required is for users to alter the file
* paths and the locals specified below.

*This work is based on the previous files which were used for the capital stock:
       *run_capstock.do (just defines some globals and then runs the 0-3 files)
       *0.prepare_inputfiles.do
       *1.prepare_deflators.do
       *2.make_raw-rd.do
       *3.pim-rd.do

*************************************************************************
*SETTING UP BASICS:
version 9
clear
cap log close
set mem 1300m
set virtual on
set matsize 10
set more off

* GLOBAL SET-UP
*file paths
global ard_root U:\ARD2\
global prog_root W:\Capital_Stock\
*changed path in next line from 'data' to 2012_Update
global cap_data $prog_root\2012_Update\
global cap_temp  $prog_root\Temp\
global cap_res $prog_root\Results\
global log_root $cap_res
global do_root $prog_root\do_files\
global cleanReg U:\ARD2_Reg_panel\Latest_years\

* Periods and sectors
```

```
global cap_start 1973
global prod_start 1973
global prod_start_v2 1980
global cons_start 1994
global serv_start 1994
global IDBR_start 1994
global pre_IDBR 1993
global pre_ABI 1994
global ABI_start 1997
global first_year $prod_start
*NOTE: final_year + previous_final_year need to be increased by 1 when a new vintage of data is added
(obviously)
global final_year 2009
global previous_final_year 2008

global prod_sec pd
global cons_sec cn
global serv_sec ca pr st re wh mt
global all_sectors $serv_sec $prod_sec $cons_sec

global countries g

*****RESEARCHERS MAY WISH TO CHANGE THE FOLLOWING DEPRECIATION RATES, KEY VARIABLE, METHOD &
calcRatio_value*****
       *copy this do file, change the global file paths and you can tailor your own capital stock!
***set key variableX (i.e. one of the key vars we use to allocate initial capital)
*choice for all years: totpurch, matfuel
*if creating capital stock for 1997 onwards could use: road_trans, lu_count, insurance
local key_var totpurch

***set method type (choice: m0 m1 m2)
local sel_meth m1

***set calcRatio_value (i.e. tolerance to missing values: max ratio of missing to real)
local calcRatio_value 2

****Plant&Machinery (PM) depreciation rate % - use two digits
local pm_dep 08

**** depreciation rates (i.e. exponential)
global d_all=(0.`pm_dep'+0.02+0.2)/3
global d_pm=0.`pm_dep'
global d_b=0.02
global d_v=0.2

*to get StatTransfer to run in the program use:
adopath + "X:\code\stat-transfer-setup"
adopath + "X:\code\ado"
```

*marking version of the capital stock local stver "2.2x" file open ardver using $cap_res\current_version.txt, write replace file write ardver "`stver'" file close ardver  *record steps used in construction: log using $log_root\CapStock_log_v`stver'.smcl, replace	
** ***NOW RUNNING MODIFIED SECTIONS FROM OLD CODE (0,1,2,3): **0.prepare_inputfiles.do *Prepare individual input files from allinputfile *nb no idea where this file originated from. *if (1) { * cd $cap_data * use allinputdata, clear * gen str10 sic92=substr(industry, 1, 2) * gen sic92_2dig=real(sic92) * sort sic92_2dig * cap drop _m * merge sic92_2dig using numlettlook_7.dta, nokeep *ie here we are just bringing in the appropriate SIC letters * replace siclett="CA" if sic92_2dig==10 * replace siclett=industry if sic92_2dig==. * preserve * keep industry sic92_2dig siclett asset d* * drop def* des* * save dep_95, replace * restore * preserve **add def2* to next line to keep yyyy 2000+ - GA220512 * keep industry sic92_2dig siclett asset def1* def2* * save def_95, replace * restore * preserve * keep industry sic92_2dig siclett asset ns* * save ns_95, replace * restore *ie this section creates: dep_95.dta, def_95.dta and ns_95.dta *} ***Coded out previous section as it seems not to be of use in by the code after producing the files** ***GA - 28/05/2012***************	Manipulation of unknown source, apparently not used and commented out in 2012

**1.prepare_deflators.do *input deflators from historic capital investment series *changed inputst path from VICS2_05 to VICS2_09 to VICS_09 foreach asset_type in pm v b { inputst $cap_data\\VICS_09.xls /tdeflators *ie calling in the excel sheet with StatTransfer -> specifies sheet: deflators replace asset="pm" if substr(asset,1,1)=="p" | asset=="int" *(original line (above), next line added by GA 01/03/12 based on new asset types in VICS_09) *replace asset="pm" if substr(asset,1,1)=="p" | asset=="int" | asset=="softao" | asset=="softp" | asset=="com" | asset=="ient" | asset=="imin" mvdecode def* ,mv(0) *ie missing values decode => changes cells=0 to =. drop if asset~="`asset_type'"  collapse (mean) def95*, by (siclett) reshape long def95,i(siclett) j(year) sort year by year: egen mean_def = mean(def95) replace def95 = mean_def if def95==. drop mean_def gen `asset_type'_def95 = def95/100 drop def95 drop if year<1979 sort year siclett save $cap_data\`asset_type'_def95, replace } *NOTE: the VICS2_05.xls contains forecasts for 2003 onwards *so does VICS2_09.xls	Loads up deflators and gets into Stata-readable form, by asset type *Note: new code sticks with 3 assets type: buildings, vehicles, other*
**2.make_raw-rd.do * Makes the cappanel_raw file clear *2.1: prepare stocks file *changed inputst path from VICS2_05 to VICS2_09 to VICS_09 inputst $cap_data\VICS_09.xls /tstocks *ie calling in the excel sheet with StatTransfer -> specifies sheet: stocks replace asset="pm" if substr(asset,1,1)=="p" | asset=="int" *replace asset="pm" if substr(asset,1,1)=="p" | asset=="int" | asset=="softao" | asset=="softp" | asset=="com" | asset=="ient" | asset=="imin" collapse (sum) stock*, by(siclett asset) reshape long stock, i(siclett asset) j(year) drop if year<1973 rename stock ind_capstk95 replace ind_capstk95=ind_capstk95*1000	Create aggregate cap stocks – need to convert from £m to £k

*ie millions we want thousands replace ind_capstk95=0 if ind_capstk95<0 *now reshape data = asset types in columns reshape wide ind_capstk95, i(siclett year) j(asset) string sort siclett year save $cap_data\stocks,replace	
*2.2: prepare VICS industry investment *changed inputst path from VICS2_05 to VICS2_09 to VICS_09 inputst $cap_data\VICS_09.xls /tinvestment replace asset="pm" if substr(asset,1,1)=="p" | asset=="int" *replace asset="pm" if substr(asset,1,1)=="p" | asset=="int" | asset=="softao" | asset=="softp" | asset=="com" | asset=="ient" | asset=="imin" collapse (sum) invest*, by(siclett asset) reshape long invest, i(siclett asset) j(year) drop if year<1973 replace invest=. if invest==0 | invest<0 *NEW STEP: create average investment series (instead of interpolating as in previous code; didn't deal with negatives+ series end) *note: vehicles investment series apperas implausible for several sic letters by siclett asset: egen av_inv=mean(invest) replace invest=av_inv if invest==. drop av_inv ren invest ind_invest95 replace ind_invest95=ind_invest95*1000 *ie millions we want thousands *now reshape data = asset types in columns reshape wide ind_invest95, i(siclett year) j(asset) string sort siclett year save $cap_data\invest,replace	Create aggregate capex in Stata-readable form. Blank investment data is replaced with the average for that asset/industry
*2.3: prepare ARD Panel; generate panel with the standardized variables for all years clear local years "$prod_start_v2/$final_year" local ftype dat forvalues yyyy = `years' { clear capture generate year=. *this loads in a blank data set foreach xx in $all_sectors { display "Sector `xx' years: `yyyy' file type: `ftype'" capture append using "$ard_root\\`ftype'`yyyy'`xx'$countries.dta" capture generate str2 sector = "" replace sector="`xx'" if sector=="" *drop if sector~="pd" & year<1997 & sector~="cn" *note: delete the line above if you want to use all service sector data from '94 onwards } }	Create the panel for capstock firms: • Load up all the returned financial data • Create 'other' asset type as residual excl buildings and vehicles • Keep 3 asset types plus total, plus variables on which to allocate/weight Does some descriptives to show

sort dlink_ref2 rename ncapex_pm wrongncapex_pm gen ncapex_pm=ncapex-(ncapex_b+ncapex_v) *need to reduce the size of the files: drop q* compress save $cap_temp\rpanel_cap_`yyyy', replace }  clear generate year=. forvalues yyyy = `years' { cap tostring cso_ref2, replace format(%14.0g) append using "$cap_temp\rpanel_cap_`yyyy'.dta" erase "$cap_temp\rpanel_cap_`yyyy'.dta" } local list1 dlink_ref2 year sector totpurch ncapex ncapex_b ncapex_v ncapex_pm wrongncapex_pm matfuel road_trans lu_count insurance keep `list1' order `list1' sort dlink_ref2 year save $cap_data\rpanel_cap, replace *ie rpanel_cap= register panel capital (to be merged on to the reg panel in 2.4) *rpanel_cap descriptives: gen Mark_matf=1 if matfuel~=. gen Mark_RT=1 if road_trans~=. gen Mark_lu=1 if lu_count~=. gen Mark_Ins=1 if insurance~=. gen Mark_TP=1 if totpurch~=. gen Mark_ncap=1 if ncapex~=. local list2 sector Mark_matf Mark_RT Mark_lu Mark_Ins Mark_TP Mark_ncap display "Descriptive tables" foreach XX in `list2' { tab year `XX' } *note: we use totpurch and employment as the 'backbone' for capital stock, other variables also investigated; *however, as can be seen from the tabs, local units, road transport, insurance only collected '97 onwards display "from tabs we can see potential for using mat_fuel as part of our calculations" display "previously, capital stock code was run using the totpurch variable"	missing obs in diff variables, helping one to choose weights etc *Note: final part not necessary in ARDx – all vars included all periods? [TEST]*
*now demonstrate that the 'net' capex figures appear to be picked up by local unit figures (sort of) keep if dlink_ref2=="sample RU ref (NB removed FR from code)" keep dlink_ref2 year ncapex_b lu_count save "$cap_data\LUcount_ncapex_example firm.dta", replace display "see example file 'LUcount_ncapex_example firm.dta' for illustration of net capex and buildings"	Unclear what the function of this descriptive is

*2.4: Load ARD Register Panel clear local years "$prod_start/$final_year" gen year=. forvalues yyyy=`years' { append using "$cleanReg\reg_pan`yyyy'.dta", keep(sel_id dlink_ref2 year siccon sel_emp) } save $cap_data/reg_pan.dta, replace count if sel_emp==. *2.5: get rid of those which are never selected * marking enterprise groups which have an observation sel_id==1 compress gen byte mark=0 replace mark=1 if sel_id==1 sort dlink_ref2 year replace mark=1 if mark[_n-1]==1 & dlink_ref2==dlink_ref2[_n-1] gsort dlink_ref2 -year replace mark=1 if mark[_n-1]==1 & dlink_ref2==dlink_ref2[_n-1] sort dlink_ref2 year drop if mark~=1 *ie dropping all ref groups where none of the firm refs has sel_id==1 drop mark	Load register panel and…what? Comment says "get rid of those which are never selected". Think this is what the code does, but a very convoluted way round *Note: new code deletes on the basis of no obs with financial returned data, "by ruref"*
2.6: get other vars (ie here we are merging on the capital information from earlier) sort dlink_ref2 year merge dlink_ref2 year using $cap_data\rpanel_cap,nokeep drop _merge ren ncapex ncapex_all mvdecode ncapex if sel_id~=1 | ncapex_all==. | year==1994 | year==1998 ,mv(0) *ie missing values decode => changes missing values '0' to '.' *now convert SIC variables gen sic92_2dig=real(substr(string(siccon),1,2)) sort sic92_2dig *merging in the associated SIC letters eg DA, DB etc merge sic92_2dig using $cap_data\numlettlook_7,nokeep drop _merge *merge investment deflators foreach aa in pm b v{ sort year siclett merge year siclett using $cap_data/`aa'_def95,nokeep drop _merge }	Add industry letter codes, bring in aggregate deflators and apply to observed capex

*create an average deflator (old code were all the same due to error: problem fixed in this code vintage) egen all_def95=rmean(pm_def95 b_def95 v_def95) *note: the rmean command is rowmean ie takes average of varlist() *create real capex foreach aa in all pm v b{ gen rncapex_`aa'95=ncapex_`aa'/`aa'_def95 } *now that we have created the four real capex vars we can... drop pm_def95 v_def95 b_def95	
***NOW FOR THE KEY STAGES: *2.7: do interpolations (linked to employment) *2.7.1: employment sort dlink_ref2 year ren sel_emp sel_emp_old by dlink_ref2: ipolate sel_emp_old year , generate(sel_emp) drop sel_emp_old *note: brought sel_emp in from ard_reg => shouldn't be many interpolations count if sel_emp==. *want few missing values for the next steps *another fix for the employment figures (zero values often occur in first years of company existence) *the code below creates a 3yr "localised" average of the employment and works backwards to fill in missing values replace sel_emp=0 if sel_emp==. by dlink_ref2: gen n=_n gen mark_emp=n if sel_emp==0 forvalues x=7(-1)1 { gen emp_V`x'=. replace emp_V`x'=sel_emp if n==`x'+1 | n==`x'+2 | n==`x'+3 by dlink_ref2: egen av_emp_3y_`x'=mean(emp_V`x') drop emp_V`x' by dlink_ref2: replace sel_emp=av_emp_3y_`x' if mark_emp==`x' drop av_emp_3y_`x' replace sel_emp=0 if sel_emp==. } count if sel_emp==0 drop n mark_emp	In returned financial data, interpolate employment when missing. The data here is the register panel with out-of-scope firms included, and firms with no valid financial returns at any time excluded. Zero employment is replaced by 3-yr average of following years (code works backwards to fill in details; choice of 7 unclear – perhaps no obs in old code had 10 blank years before valid employment?) *Note: employment is the ABI employment figure, which all firms pre-2009 had – for ABS data, this will be either BRES (real or imputed) or IDBR* *Note: new code use log-linear imputation and minimum employment value to generate employment 'weight'*
*2.7.2: capex *gen total investment per employee => then multiply through *A. gen selected employment when we have rncapex values (withV = with value)	Calculate average investment per employee (all firms here

gen withV_emp=sel_emp if rncapex_v95~=. by dlink_ref2: egen Tot_emp_withV=total(withV_emp) drop withV_emp *B. gen selected investment foreach aa in v b pm all{ by dlink_ref2: egen Tot_I_`aa'_withV=total(rncapex_`aa'95) } *C. generate average investment foreach aa in v b pm all{ by dlink_ref2: gen I_per_emp_`aa'=(Tot_I_`aa'_withV/Tot_emp_withV) } *D. now multiply out for "extrapolation" foreach aa in v b pm all{ by dlink_ref2: replace rncapex_`aa'95=(sel_emp*I_per_emp_`aa') if rncapex_`aa'95==. }  *recode missing values foreach aa in all pm v b{   count if rncapex_`aa'95==. } foreach aa in all pm v b{ replace rncapex_`aa'95=0 if rncapex_`aa'95==. } *E. now get rid of variables used in the construction: drop Tot_emp_withV Tot_I_v_withV Tot_I_b_withV Tot_I_pm_withV Tot_I_all_withV I_per_emp_v I_per_emp_b I_per_emp_pm I_per_emp_all *2.8: generate markers to highlight which firm's estimated capital stock contains large amount of imputation gen calc_mark=0 foreach aa in v b pm all{ replace calc_mark=1 if ncapex_`aa'==. & rncapex_`aa'95~=. } by dlink_ref2: egen N_mark=total(calc_mark) by dlink_ref2: gen firm_count=_N by dlink_ref2: gen firm_withV_count=(firm_count-N_mark) by dlink_ref2: gen calcRatio=(N_mark/firm_withV_count) drop calc_mark N_mark firm_count firm_withV_count *so if calcRatio>1 then there are too many imputations... ie more than half are calculated *2.9: generate sel_idx variable gen sel_idx=1 if calcRatio<`calcRatio_value' | calcRatio==`calcRatio_value' replace sel_idx=0 if sel_idx==.	have at least one valid financial return), by asset type. Apply to obs with missing investment values Generate marker (sel_idx) for amount of imputation carried out (sel_idx=1 means ratio of imputed to valid values is too high to be used)
*2.10: merge capstocks (aggregate figures for sectors) drop if siclett=="" tab year sort siclett year	Merge onto aggregate capstocks and investment, by letter code So, the panel of all firms with at

merge siclett year using $cap_data\stocks,nokeep drop _merge *new checks for infinity problem: count foreach aa in v b pm{ count if ind_capstk95`aa'==. } *note: we do not have any cap stock figures for 2005. Compare counts to year tabs above. *2.11: merge investment figures (aggregate figures for sectors) sort siclett year merge siclett year using $cap_data\invest,nokeep drop _merge *new checks for infinity problem: count foreach aa in v b pm{ count if ind_invest95`aa'==. } *note: we do not have any cap stock figures for 2005. Compare counts to year tabs above. *INTERIM SAVE save $cap_temp\pre_shares_test.dta, replace *INTERIM SAVE use $cap_temp\pre_shares_test.dta, clear	least one valid capex entry contains • Observed capex, deflated • Imputed deflated capex based on employment (real or imputed) • Real total cap stock • Real tot investment
*2.12: Create capital share measures foreach aa in pm v b { bysort year siclett sel_idx: egen sum_`aa'=sum(rncapex_`aa'95) *note: implausible that sum rncapex is negative overall replace sum_`aa'=abs(sum_`aa') bysort year siclett sel_idx: gen invest_inflator`aa'=sum_`aa'/ind_invest95`aa' *we are only interested in sel_idx==1 replace invest_inflator`aa'=. if sel_idx==0 *note: also deal with any zero values by taking averages bysort siclett sel_idx: egen av_inv_infl=mean(invest_inflator`aa') replace invest_inflator`aa'=av_inv_infl if invest_inflator`aa'==0 drop av_inv_infl } *invest_inflator gives the share of selected units investment in terms of total whole economy investment ****NOTE: IF ind_invest95`aa'==. then <=> invest_inflator`aa'==0 ****	• Create pp of observed/total capex using only firms without too much imputation, by industry • Set others shares to missing • Set valid unis with missing shares to industry average *Note: assumptions about non-negative capex wrong eg public sector net disposer 1998-2015 Don't follow rationale for last – only relevant if no observed capex or no aggregate in any year?*

*PROBLEMS WITH SHARES ETC: preserve collapse (mean) ind_invest95pm sum_pm invest_inflatorpm ind_invest95b sum_b invest_inflatorb ind_invest95v sum_v invest_inflatorv, by (year siclett sel_idx) drop if sel_idx==0 save $cap_data\investment_shares_breakdown.dta, replace *from this file we can see the spread of the shares which will be allocated *note: want to constrain values between 0-1, old files had -ve + >1 which impacts on cells later on. **OK to have shares >1 implies that industry numbers are too low??? restore	This bit just generates a dataset for descriptives *Note: concern about shares>1 implies imputation cut-off wrong*
generate selected capital=sel_cap foreach aa in pm v b{ gen sel_cap`aa'=(ind_capstk95`aa')(invest_inflator`aa') } *generate 3 digit sic: gen sic92_3dig=int(siccon/100) if siccon>9999 replace sic92_3dig=int(siccon/10) if siccon<9999 *generate count by firm: sort dlink_ref2 year by dlink_ref2: gen n=_n *ANOTHER INTERIM SAVE... save $cap_data\descriptives_temp.dta, replace ******RUN SOME DESCRIPTIVES + alternatives to show methodology differences (now in its own .do file): *note: this file can be run in conjunction with main capstock do file or ignored if take methodology below as given ** Commented out by Eric Scheffel on 11/08/09 *do $do_root\Descriptives_for_Capital_Stock.do	Firm capstocks created as share of industry total, using observed share of investment as the marker; these will be the initial capstocks when a firm is first observed Set 'n' as the within-RU year counter
*3.1. NOW RUNNING KEY STAGES IDENTIFIED AS BEST PROCEDURE FROM DESCRIPTIVES.DO FILE *set up basic file using 'method X', which the descriptives.do indicates produces fewest negative series use $cap_data\descriptives_temp.dta, clear keep year dlink_ref2 n siclett sic92_* sel_emp calcRatio sel_idx rncapex_* sel_cap* `key_var'  *note: researchers could create their own cap stock data sets by altering the following var at start of do file: *local key_var variableX *now fill-in missing values & create 3 share measures foreach var in `key_var' {	For observations without the relevant share variable, replace with weighted average Set the share in deselected obs back to 0 Then calculate share to be used as (all within year):

```
        *fill in missing values
        replace `var'=. if `var'==0 & n==1
        gen withV_emp_`var'=sel_emp if `var'~=.
        by dlink_ref2: egen Tot_emp_withV_`var'=total(withV_emp_`var')
        by dlink_ref2: egen Tot_`var'_withV=total(`var')
        by dlink_ref2: gen av_T`var'_Temp=(Tot_`var'_withV/Tot_emp_withV_`var')
        by dlink_ref2: replace `var'=(av_T`var'_Temp*sel_emp) if `var'==.
        *now get rid of variables used in the construction:
        drop withV_emp_`var' Tot_emp_withV_`var' Tot_`var'_withV av_T`var'_Temp
        replace `var'=0 if `var'==.

        *create share measures using the 3 different methodologies:
        *method 0:
        bysort year siclett: egen sum_`var'=sum(`var') if sel_idx==1
        gen m0_`var'_sh=`var'/sum_`var'
        *method 1: within (`var')
        bysort year sic92_3dig: egen sum_`var'_3d=sum(`var') if sel_idx==1
        gen m1_`var'_sh_3d=`var'/sum_`var'_3d
        *method 2: across (`var')
        bysort year siclett: egen sum_`var'_siclett=sum(`var') if sel_idx==1
        sort dlink_ref2 year
        gen m2_`var'_sh_3d=(sum_`var'_3d/sum_`var'_siclett)
        *method 1: across (employment)
        bysort year sic92_3dig: egen sum_emp_3d=sum(sel_emp) if sel_idx==1
        bysort year siclett: egen sum_emp_siclett=sum(sel_emp) if sel_idx==1
        sort dlink_ref2 year
        gen m1_emp_sh=(sum_emp_3d/sum_emp_siclett)
        *method 2: within (employment)
        gen m2_emp_sh=(sel_emp/sum_emp_3d)
        ****we want generate new mat_sh variables
        gen m1_`var'_sh=m1_`var'_sh_3d*m1_emp_sh
        gen m2_`var'_sh=m2_`var'_sh_3d*m2_emp_sh
        *get rid of vars used in construction:
        drop sum_`var' sum_`var'_3d m1_`var'_sh_3d sum_`var'_siclett m2_`var'_sh_3d sum_emp_3d sum_emp_siclett
m1_emp_sh m2_emp_sh
}


*Note: descriptives suggest that m1 is best; however, the local can be altered to select a different method
* this may be appropriate as the descriptives file may suggest other method is best if change depreciation
* or calcratio etc.
*note: 3 choices of selected method: m0 m1 m2
*from the start of the .do file we specified:
*local sel_meth mX
```

- Method 0: share by letter
- Method 1: share by 3-digit industry weighted by employment
- Method 2: share by 3-digit industry weighted by employment

Variables m[0; 1; 2]_sh are created

```
*3.2. now re-run all as a single section of code:
foreach method in `sel_meth' {
        foreach var in `key_var' {
                **** define initial values & perpetual inventory   **
```

Looping through alternative key vars and methods, if specified:

- For year 1 firms, create initial

1: NEW initial value foreach aa in pm v b{ gen `method'_`var'_rcapstk_`aa'95=. } foreach aa in pm v b{ *gen sel_cap`aa'=(ind_capstk95`aa')*(invest_inflator`aa') replace `method'_`var'_rcapstk_`aa'95=((sel_cap`aa'`method'_`var'_sh)+rncapex_`aa'95) if n==1 & sel_idx==1 *ie here we are allocating initial values of ind cap stock across units }  *2: perpetual inventory foreach aa in pm v b{ bysort dlink_ref2 (year): replace `method'_`var'_rcapstk_`aa'95=((1-${d_`aa'})*`method'_`var'_rcapstk_`aa'95[_n-1])+rncapex_`aa'95 if n>1 & sel_idx==1 }  *3: Aggregate assets gen `method'_`var'_rcapstk_all95=`method'_`var'_rcapstk_pm95+`method'_`var'_rcapstk_b95+`method'_`var'_rcapstk_v95  *tidy variable names renpfix `method'_`var'_ } } *dropping unused vars: keep dlink_ref2 year sel_emp siclett sic92_* sel_idx calcRatio sel_cap* rncapex* rcapstk* `key_var' sh n	capstock by taking m[0/1/2]_sh of valid cap stocks (ie from non-over-imputed firms), plus year1 capex • For other firms, take off depreciation from previous year and add on capex Create a total asset variable
*Now recode any remaining negative series: calculate 'localised' figure to add to series to avoid negative values *note: negative values must be incorrect. Likely to have missed a "lump" of investment *or not allocated enough in the initial allocations *NOTE: we want to run this multiple times to pick up series with many negative values do $do_root\localised_boost do $do_root\localised_boost do $do_root\localised_boost do $do_root\localised_boost do $do_root\localised_boost	See below
*Now recode any remaining negative series (hopefully shouldn't be many!) *let rcapstk stay if total is positive, but get rid of any negative asset series foreach aa in all pm v b{ markg if rcapstk_`aa'95<0 ,by(dlink_ref2) marker(m_`aa') gen Neg_CS_marker_`aa'=. replace Neg_CS_marker_`aa'=1 if m_`aa'==1 & n==1	Describe number of negative capstock series (ie rurefs with negative capstocks, by asset) and replace with missing values; replace all if with missing if total

display "#negative series for asset `aa':" count if Neg_CS_marker_`aa'==1 replace rcapstk_`aa'95=. if m_`aa'==1 } foreach aa in pm v b{ replace rcapstk_`aa'95=. if rcapstk_all95==. }	assets are missing
3.3. final tweaks to the data: destring dlink_ref2, replace format dlink_ref2 %11.0f keep dlink_ref2 year sel_emp siclett sic92_* sel_idx calcRatio rncapex rcapstk* `key_var' order dlink_ref2 year siclett sic92_* sel_idx calcRatio sel_emp `key_var' rncapex_pm95 rncapex_b95 rncapex_v95 rncapex_all95 rcapstk_pm95 rcapstk_b95 rcapstk_v95 rcapstk_all95 label variable dlink_ref2 "Reference number" label variable sel_emp "Employment: zero values recoded using 3yr 'localised' average" label variable siclett "SIC letter coding" label variable sic92_2dig "SIC 2 digit code" label variable sic92_3dig "SIC 3 digit code" label variable sel_idx "marker =1 , if firm is 'selected' " label variable calcRatio "ratio of imputed to actual observations in original data" cap label variable totpurch "Total puchases: missing values calculated" cap label variable matfuel "Materials&Fuel: missing values calculated" label variable rncapex_pm95 "Real net capital expenditure: plant&machinery" label variable rncapex_v95 "Real net capital expenditure: vehicles" label variable rncapex_b95 "Real net capital expenditure: buildings" label variable rncapex_all95 "Real net capital expenditure: all assets=pm+v+b" label variable rcapstk_pm95 "Real capital stock: plant&machinery" label variable rcapstk_v95 "Real capital stock: vehicles" label variable rcapstk_b95 "Real capital stock: buildings" label variable rcapstk_all95 "Real capital stock: all assets=pm+v+b" *note: as discussion with J.Haskel, all should really be weighted by asset prices compress save $cap_res\CapitalStock_Depreciation`pm_dep'_v`stver'.dta, replace *4.0: clean up the temp folder *erase $cap_temp\... *FIN log close	Clean up and label
CODE FROM 'localised_boost.do', last edited 2008	

*Now recode any remaining negative series: calculate 'localised' figure to add to series to avoid negative values *note: negative values must be incorrect. Likely to have missed a "lump" of investment *or not allocated enough in the initial allocations foreach aa in pm v b { by dlink_ref2: egen most_neg`aa'=min(rcapstk_`aa'95) replace most_neg`aa'=. if most_neg`aa'>0 | most_neg`aa'==0 by dlink_ref2: gen mark_`aa'=n replace mark_`aa'=. if most_neg`aa'~=rcapstk_`aa'95 | most_neg`aa'==. *generate boost variable (negate the negative) gen inc_v1_`aa'=0-most_neg`aa' *localise this around the negative value so that the initial values don't become too distorted gen mark2_`aa'=mark_`aa' replace mark2_`aa'=3 if mark2_`aa'>=4 & mark2_`aa'~=. gen insert_`aa'=. replace insert_`aa'=(mark_`aa'-mark2_`aa') if mark_`aa'>=4 replace insert_`aa'=1 if mark_`aa'==3 | mark_`aa'==2 | mark_`aa'==1 by dlink_ref2: egen insert2_`aa'=min(insert_`aa') by dlink_ref2: egen mark3_`aa'=min(mark2_`aa') drop most_neg`aa' mark_`aa' mark2_`aa' insert_`aa' *now must increase the value to take account of the depreciation by dlink_ref2: gen inc_v2_`aa'=inc_v1_`aa'*((1/(1-${d_`aa'}))^mark3_`aa') *now boost capex figures before recalculating rcapstk by dlink_ref2: replace rncapex_`aa'95=(rncapex_`aa'95+inc_v2_`aa') if n==insert2_`aa' drop mark* insert* inc* }	Find the most negative value for each series, by ruref (ignore series with no neg values) Create additional capex by working out what is needed to avoid biggest engative, and then adding it Note: assumption is the neg capstocks due to unobserved investment, not inappropriate starting values
*drop previous rcapstk_all95, replace values for the others drop rcapstk_all95 *1: now recalculate the capital stock figures using methodology above foreach aa in pm v b{ *gen sel_cap`aa'=(ind_capstk95`aa')*(invest_inflator`aa') replace rcapstk_`aa'95=((sel_cap`aa'*sh)+rncapex_`aa'95) if n==1 & sel_idx==1 *ie here we are allocating initial values of ind cap stock across units } *2: perpetual inventory foreach aa in pm v b{ bysort dlink_ref2 (year): replace rcapstk_`aa'95=((1-${d_`aa'})*rcapstk_`aa'95[_n-1])+rncapex_`aa'95 if n>1 & sel_idx==1 } *3: Aggregate assets gen rcapstk_all95=rcapstk_pm95+rcapstk_b95+rcapstk_v95 *quick check of negative series foreach aa in all pm v b{	Drop initial capstock and recalculate Recalculate PIM series Display descriptives

```
        markg if rcapstk_`aa'95<0 ,by(dlink_ref2) marker(m_`aa')
        gen Neg_CS_marker_`aa'=.
        replace Neg_CS_marker_`aa'=1 if m_`aa'==1 & n==1
        display "#negative series for asset `aa':"
        count if Neg_CS_marker_`aa'==1
        drop m_`aa' Neg_CS_marker_`aa'
}

*end of this .do file
```

6. Appendix 2: detailed code breakdown – current code

Note that this code might not exactly reflect the current code which is subject to amendment. This analysis reflects version 00.14.

Code	Meaning and notes
* Create Capstock.do * * Code to generate RU-level capital stock * * Last modified: * 02.12.15 FR Created (VML) * * This capital stock has the following variablesRegister panel has the following variables: * taken from ARDx (IDBR info): ruref, egrp_ref, entref, ultfoc * taken from ARDx (ARDx surveyed/created): sector, sic03/07, selection_type * calculated/updated here: * year * selection_type Added 'out of scope' as possible value - implies observation has been interpolated * age Number of years firm has been in reg_panel * * Assumptions: * set_ARDx_globals_vxx.xx.do is available (note: hard-coded here so it can be run stand-alone by researchers) * ARDx register panel FOR THE SAME PERIOD AS THE CAPSTOCK is available * ARDx register panel has complete set of obs with out-of-scope data tkane from IDBR/interpolated * year_start and year_end need to be defined here if not running through "Rebuild ARDx" * * Note that, because of the imputation, Capstock CHANGES AS EACH YEAR IS ADDED * * Program structure * 1. Set up - check that global macros have been set up, set local ones for this run * Edit the local macros in this setion to control what the program does * 2. Create dataset to be used (if not already done - use flags to switch off) * Load up all the returned ARD data: pool over years, keeping only min var set * * Start with the Register Panel * Strip all vars but ruref, year, SIC vars * Generate SIC letter code and drop SIC vars * Merge with ARDx files to acquire IDBR emp, IDBR turnover, raw mat purch, sales * Save under capstock_base (so that we don't need to do again for different parameterisations)	**Program and macro set-up** Introduction

`` * 3. Create aggregate files: deflators and cap stock (if not already done - use flags to swtich off) * Update infor in input spreadhseet from ONS files, if necessary * Create and save cap stock by asset type and industry, and deflators by asset type * * *** * Run quietly, displaying only specific output quietly { *** * Part I: Set-up * *** capture log close set more off if "$globals_run"!="Y" { *** Global macros not set up yet! *** noisily display "Global macros not run" stop } ```	
`` * These macros control how the program operates - EDIT THESE * == *These first group are for development and testing - users wanting to create a new file should have these all switched on, except the last local version_no 00.13 local create_base 0 local create_capstock_base 0 local create_aggregates 1 local create_weights_file 1 local create_input_file 1 * produce lots of stats, or not local verbose 1 * aggregate data input files global XLS_capstock_aggregates = "$data_folder\capstock\capstock_aggregates_to_$last_year.xls" global capstock_aggregates_sheet = "cleaned for Stata" global deflators_sheet = "deflators_cleaned_for_stata" global input_XLS = "$data_folder\capstock\capex_aggregates_to_$last_year.xls" ```	Set up of user-controlled macros. The first group should only be changed for rebuilding the base input file eg when a new year is available, or to create a version with a different imputation method.

`  global insheet_b = "Buildings and transfer costs"` `  global inrange_b = "B3:DB40"` `  global insheet_v = "Transport"` `  global inrange_v = "B3:DB40"` `  global insheet_other = "Other"` `  global inrange_other = "B3:DB40"` `  * this is how many employees we should treat a firm as having for weighting purposes - CAN'T BE ZERO!` `  local min_emp_count 0.1` `  global first_year = $ABI_start` `*  global last_year = $year_end` `  global last_year = 2014` `  global capstock_root = "$data_folder\capstock"`	
`  * tolerance can be set as minim number of non-imputed or as min proportion of non-imputed. Test is for the word 'proportion'` `  local tolerance 2` `  * local tolerance_type proportion` `  local tolerance_type number` `  * should we drop firms that show negative capex when observed?` `  * options: 'all' - overall neg capex, 'neg_only' - only drop neg years when calcuating ave capex per emp, 'none' - do nothing` `  local drop_neg_capex all` `*  local drop_neg_capex neg_only` `*  local drop_neg_capex none` `  * variables to be used for the initial weighting, given friendly names` `  local ARDx_vars totpurch ABSturnover IDBRturnover` `  local totpurch_var wq499` `  local ABSturnover_var wq550` `  local IDBRturnover_var turnover` `  local key totpurch` `  * depreciation rates` `  local depreciation_v 0.00` `  local depreciation_b 0.0`	This second group is of more interest to end-users – likely to play around with these more

`` local depreciation_other 0.0 * Method to generate imputed capex * currently recognised options: emp_weighted only local capex_method emp_weighted * Method to allocate initial cap stocks (see manual for details) * currently recognised options: simple, key_share, emp_share local capstock_methods simple key_share emp_share local capstocks_to_create simple key_share emp_share * Method to deal with negative capital stocks * options available: missing_capex rebase_init_stock * missing_capex assumes one or more lumps of missing investment * rebase_init_stock assumes initial casptock is too low * Program records how many iterations are needed to remove negative cap stocks * max_backfill_attempts specifies how hard you should try to avoid negative capstocks ie how many times to retrosepctviely adjust the data * local backfill_method missing_capex local backfill_method rebase_init_stock local max_backfill_attempts 5 * use current price or chained volume measure? local current_price 0 ```	
`` * General macros - SHOULDN'T need changing by users * == * ONS aggregates have 10 types of capex but ABS only claims to have 3 - buildings, vehicles and the old 'plant + machinery' = "other" * As a general rule, follow old code in describing capex_pm as residual "other" * local cap_types b v pm c ps oas ao mr ca rd local cap_types ="$cap_types" local capex_b wq531 local capex_v wq532 local capex_other wq533 global num_letter_codes local ss = substr("$first_year", 3,2) local ee = substr("$last_year", 3,2) local file_name1 = "`ss'_`ee'_minemp`min_emp_count'" ```	This third group control the program – generally shouldn't need changing

`` local file_name2 = "t`tolerance'_k`key'_m`method'_`backfill_method'" global reg_panel = "$ARDx_folder\\$reg_panel_file`ss'_`ee'_$missing_emp" global extended_reg_panel = "$capstock_root\extended_reg_panel" global capstock_base = "$capstock_root\capstock_base" global returned_base = "$capstock_root\returned_base" global capstock_with_weights = "$capstock_root\capstock_base_with_weights" global capstock_input = "$capstock_root\capstock_input_`file_name1'"   global capstock_file = "$capstock_root\capstock_`file_name1'_`file_name2'.dta"   global ARDX_input = "$ARDx_folder\ARDx_file"   local universe = "$universe_file"   local base_creation_log = "$output_folder\create_capstock_`file_name1'_v`version_no'.smcl"   local capstock_creation_log = "$output_folder\create_capstock_`file_name1'_`file_name2'_v`version_no'.smcl" * tempfile temp_base local curr_name real if `current_price' { local curr_name nom } ```	
`` if (`create_base'+`create_capstock_base'+`create_aggregates'+`create_weights_file'+`create_input_file') >0 {     * only log file if recreatign the base files      log using "`base_creation_log'", replace set linesize 255 noisily display "***" noisily display "Capstock base file creation log" noisily display "Program running from $first_year to $last_year" noisily display "Options:" noisily display "create base : `create_base'"     noisily display "capex weighting method   : `capex_method'" noisily display "minimum employment count : `min_emp_count'"     noisily display "output file              : $capstock_merged"     noisily display "log file                 : `base_creation_log'" noisily display "***" } ```	Set up a separate log file for the input file creation, if necessary, and helpfully print the parameters

* now on to the main program *** * Part 2: Create base files and aggregates, if needed * ***	**Creating input file**
* first, create the aggregate files in a useful format from ONS spreadsheets if `create_aggregates' { noisily do "$code_folder\create_aggregates_for_capstock_v04.do" }	Run the .do file that generates aggregates capex and capstocks (year/industry level) and deflators
* second, combine returned, universe and aggregate data for all obs if `create_base' {    * if necessary, combine all the ARDx returned data files and merge onto the register panel   if `create_capstock_base' { foreach vv in `ARDx_vars' {   local using_vars = "`using_vars' ``vv'_var'" } foreach vv in `cap_types' {   local using_vars = "`using_vars' `capex_`vv''" } noisily display "Creating returned base (panel of information from financial surveys)..." use ruref year sic03 sic07 selection_type employment using "$reg_panel" , clear local first 1 forvalues yyyy= $first_year/$last_year { noisily display "Extracting `using_vars' from $ARDx_folder\ARDx`yyyy'.dta" if `first' {   use `using_vars' ruref year using "$ARDx_folder\ARDx`yyyy'.dta" , clear   local first 0   }   else {   append using "$ARDx_folder\ARDx`yyyy'.dta" , keep(`using_vars' ruref year)   }   }   foreach vv in `ARDx_vars' { rename ``vv'_var' `vv'   }   foreach vv in `cap_types' {	Create base file [skip if not necessary] Create returned base [skip if not necessary] Create a macro containing all the variables to be taken from the ARDx Extract these from all the years of the ARDx and put into a file called ‘returned_base’

```
        rename `capex_`vv'' capex_`vv'
      }

      keep `ARDx_vars' capex* ruref year

      compress
      noisily describe
      noisily save "$returned_base" , replace
    }
    else {
      noisily display "Reading returned base file $returned_base..."
      use "$returned_base" , clear
    }

    * Now create subset of usable obs by getting rid of those never selected to give financial info
    * NB old code seemed (section 2.5) to only keep those with consecutive observations
    * complicated in BRES/ABS because firms might have employment but no financial info

    noisily display "Creating subset with financial info"
    use ruref year employment sic03 sic07 selection_type using "$reg_panel" , clear
    sort ruref year
    gen fin_data = (selection_type == 0) | (selection_type == 2)
    by ruref: egen n_finances = sum(fin_data)
    keep if n_finances > 0
    drop n_finances fin_data

    noisily display "Merging with returned data"
    noisily merge 1:1 ruref year using "$returned_base" , nogenerate keep(match master)

    * now, put the letter codes onto usable obs in the register panel, and then merge with ARDx
returned data
    * Note that leading zeros need to be added
    * Also save sic3d - will use it later
    noisily display "Adding letter codes..."
    gen sic_to_do = real(sic03)
    replace sic_to_do = real(sic07) if year>2007
    gen int sic2d = sic_to_do/1000
    gen int sic3d = sic_to_do/100
    drop sic03 sic07 sic_to_do
    do "$code_folder\sic letter lookup.do"
    drop sic2d

    compress
    noisily describe
```

[end optional 'returned base' creation; if file not created fresh, read it in]

Drop all the ones that have never given financial info – can't be used (unless you were to give them investment based on employment & industry only)

Create the letter codes

`noisily save "$capstock_base" , replace` `}` `else {` `  if `create_input_file' {` ` noisily display "Reading capstock base file $capstock_base..."` ` use "$capstock_base" , clear` ` }` `}`	Save this as 'capstock_base' [end optional 'capstock base' creation; if file not created fresh, read it in]
`* Third, create the weighting variable` `if `create_weights_file' {`  ` if "`capex_method'" == "emp_weighted" {` `  * create the employment weights = actual for non-zero, weighted average for zeros` `  * Create non-linear weighting by converting to and from logs - also avoid problems of neg values.` `  * Set min emp to some positive value; all will then have imputed values with some allocation of capital` `  noisily display "Generating employment weights..."` `  sort ruref year` `  by ruref: gen emp2 = employment` `  by ruref: replace emp2 = `min_emp_count' if (emp2==0) & ((_n==1) | (_n==_N))` ` gen log_emp = ln(emp2)` ` by ruref: ipolate log_emp year , generate(log_emp_i) epolate` ` gen capex_weight = exp(log_emp_i)` ` drop log_emp_i emp2 log_emp` ` }`  ` else {` ` * other methods - to be determined` ` }`  ` noisily save "$capstock_with_weights" , replace`  `}`  `else if `create_input_file' {` `  noisily display "Reading file with imputed weights $capstock_with_weights..."` `  use "$capstock_with_weights" , clear`	Create the weights file [skip if not necessary] Make sure employment is always non missing by doing a log-linear interpolation (this why the 'minimum' employment value must be positive, for logging) [end optional 'capstock with weights' creation; if file not created fresh, read it in]

} * end create_weighted_file or not	
	[Most parts of the remaining code loop over the asset types defined in the options; for this section, assume each set of notes begins “For each asset type…”]
* Fourth, the financial stuff * Create deflators and aggregate capital stocks, if needed, cap stocks to be deflated * Also need to create an 'other', (deflate before collapsing) * Finally, not that the ONS spreasheets can have extraneous spaces in letter codes - remove if `create_input_file' { * create imputed investment values - ignore net disposals at this stage * also create the marker for imputation * must be sorted by ruref at this point gen byte nonimputed = ((selection_type==0) | (selection_type==2)) by ruref: egen tot_nonimputed = sum(nonimputed) foreach vv in `cap_types' { noisily display "Creating imputed capex for `vv'" by ruref: egen tot_observed_weight = sum(capex_weight) if nonimputed by ruref: egen tot_usable_weight = max(tot_observed_weight) by ruref: egen tot_observed_capex = sum(capex_`vv') if nonimputed by ruref: egen tot_usable_capex = max(tot_observed_capex) gen ave_capex = tot_usable_capex/tot_usable_weight replace ave_capex = 0 if ave_capex==. noisily replace capex_`vv' = ave_capex*capex_weight if capex_`vv' == . drop ave_capex tot_usable_capex tot_observed_capex tot_observed_weight tot_usable_weight label variable capex_`vv' "Actual or imputed capex, nominal values" } noisily display "Adding aggregate data..." gen byte current_price = `current_price' noisily merge m:1 current_price year letter07 using "$aggregate_capex" , nogenerate keep(match master) noisily merge m:1 current_price year letter07 using "$aggregate_capstock" , nogenerate	Bring together firms existence data, financial data and aggregates to create the ‘capstock input’ file [skip if not necessary] Generate markers for whether firm has provided capex data or not Calculate per-employee investment data for a ruref over all years and apply to years where capex is not observed Merge in aggregate data using industry letter codes; assume that both constant

keep(match master) * if constant prices, need to deflate returned capex - do in next section when we have a handy loop sort ruref year foreach vv in `cap_types' {       foreach var in ind_capex ind_capstock deflator{         noisily display "Interpolating industry data `var', asset type `vv', for out-of-scope firms..."         ren `var'_`vv' old_var         by ruref: ipolate old_var year , generate(`var'_`vv') epolate drop old_var } * need to do something about the out-of-scope firms - no letter match(plus a v small number in early years where SIC is missing) ****** TEMP FIX for missing SICs in early years ******** noisily display "Marking missing SIC codes as out-of-scope... noisily replace sel = 5 if ind_capex_`vv'==. } noisily save "$capstock_input" , replace } else { noisily display "Reading input file $capstock_input..." use "$capstock_input" , clear } * end create_input_file or not	and current prices are merged Where RUs are out of scope (and so we don't know what industry to put them in). do linear interpolation of missing aggregate values [end optional 'capstock input' creation; if file not created fresh, read it in]
if (`create_base'+`create_capstock_base'+`create_aggregates'+`create_weights_file'+`create_input_file')>0 { * log file only open if recreating the base files log close }	Save and close the log file, if necessary
** * Part 3: Apply the various options and calculate the cap stock * * Do here so we can play around with options quickly - if done in merged file * * above means that needs to be re-created every time, which is a lengthy process * ** log using "`capstock_creation_log'", replace	**Create capital stock file** Display info for log file

```
set linesize 255

noisily display "*********************************************************************************"
noisily display "Capstock creation log"
noisily display "Program running from $first_year to $last_year"
noisily display "Options:"
noisily display "create base              : `create_base'"
noisily display "tolerance for imputation : Non-imputed `tolerance_type' >= `tolerance'"
noisily display "key allocation variable  : `key'"
noisily display "capex weighting method   : `capex_method'"
noisily display "init capstock methods    : `capstocks_to_create'"
noisily display "Drop negative capex      : `drop_neg_capex'"
noisily display "minimum employment count : `min_emp_count'"
noisily display "backfilling method       : `backfill_method'"
noisily display "current prices?          : `current_price' `curr_name'"
noisily display "output file              : $capstock_file"
noisily display "log file                 : `capstock_creation_log'"
noisily display "*********************************************************************************"
```

```
* First, make the necessary adjustmetns to the data

* Set to current or constant price

foreach vv in `cap_types' {
  if `current_price'==0 {
    replace capex_`vv' = capex_`vv'/deflator_`vv'
  }
}

* get rid of negative capex values
if "`drop_neg_capex'" == "all" {
  * dropping all obs where neg capex observed
  gen drop_this = 0
  foreach vv in `cap_types' {
    gen neg_capex_`vv' = (capex_`vv'<0) & (nonimputed==0)
    by ruref: egen drop_`vv' =max(neg_capex_`vv')
    by ruref: replace drop_this = drop_this + drop_`vv'
  }
  noisily display "Getting rid of firms with overall negative capex"
  noisily drop if drop_this
}
if "`drop_neg_capex'" == "neg_only" {
```

Convert to constant prices, if necessary

Drop obs with negative capex, depending on options:
‘all’: drop ruref if overall capex is negative for a ruref
[‘neg_only’: sensible? not implemented]
‘none’: no action necessary

`` * should we code this? likely to lead to significant overestimation } else { * take no action } * Create tolerance marker and drop over-imputed rurefs - done at ruref not asset level if "`tolerance_type'"=="proportion" {      by ruref: gen tolerable = (tot_nonimputed/_N) >= `tolerance' } else { by ruref: gen tolerable = tot_nonimputed >= `tolerance'    }    noisily display "Dropping rurefs with too much imputation..."    noisily drop if tolerable==0    drop nonimputed tot_nonimputed    * Calculate proportion of observed/total investment to adjust total capstock to be allocated to observed rurefs    sort year letter07   foreach vv in `cap_types' { *** PROBLEM: QUITE POSSIBLE THAT sum_capex_observed IS NEGATIVE!!! by year letter07: egen sum_capex_observed_`vv' = sum(capex_`vv') gen sum_capstock_observed_`vv' = ind_capstock_`vv'*sum_capex_observed_`vv'/ind_capex_`vv' } ```	Drop rurefs which don't meet the tolerable level of non-imputed values Calculate share of ARDx observed capex to aggregate capex
`` * Second, create original cap stocks (ie on first appearance in data). Obs missing key variables are dropped * remember the key variable 'key' was set at the beginning * Set missing 'key' (allcoation) variable to average per employee if missing sort ruref year by ruref: replace `key' = . if `key'==0 & _n==1 by ruref: gen key_weight = capex_weight if `key'~=.   by ruref: egen tot_key_weight = total(key_weight)   by ruref: egen tot_key = total(`key') by ruref: gen key_average = tot_key/tot_key_weight by ruref: replace `key' = key_average*capex_weight if `key' == . drop key_weight tot_key_weight tot_key key_average * Get the vars needed for calcs. Note that sic3d must be a subset of letter07 sort year letter07 sic3d ```	[this section uses the 'key' variable for initial capstock allocation defined at the beginning] Set any missing 'key' values to the ruref average (none, letter,

by year letter07: egen tot_key_letter = sum(`key') by year letter07: egen tot_weight_letter = sum(capex_weight) by year letter07 sic3d: egen tot_key_3d = sum(`key') by year letter07 sic3d: egen tot_weight_3d = sum(capex_weight) * Now calculate, depending on the method foreach method in `capstocks_to_create' { * Three ways to calculate initial weights currently recognised options: simple, key_share, emp_share if "`method'"=="simple" { gen init_share_`method' = `key'/tot_key_letter } else if "`method'"=="key_share" { gen init_share_`method' = (`key'/tot_key_3d)*(tot_weight_3d/tot_weight_letter) } else if "`method'"=="emp_share" { gen init_share_`method' = (capex_weight/tot_weight_3d)*(tot_key_3d/tot_key_letter) } }	Calculate the info needed to do the different methods: totals for different levels of industry breakdown (none, letter, SIC 2digit), by year Calculate observation weights for each allocation method
sort ruref year foreach method in `capstocks_to_create' { foreach vv in `cap_types' { by ruref: gen capstock_`vv'_`method' = (init_share_`method'*sum_capstock_observed_`vv') + capex_`vv' if _n==1 by ruref: replace capstock_`vv'_`method' = capstock_`vv'_`method'[_n-1]*(1-`depreciation_`vv'') + capex_`vv' if _n>1 label variable capstock_`vv'_`method' "Estimated `curr_name' cap stock (method:`method' key:`key' dep:`depreciation_`vv'')" } }	Generate the initial capstock (year 1 for each ruref) as 'share' x 'observed' capstock Generate the capstock in other years via the PIM (ie capstock plus capex minus depreciation)
** * Part 4: Summary statistics prior to backfilling * * Along the way, create markers for negative vlaues which will beuseful later * ** if `verbose' { sort ruref	Generate summary stats on the number of negative capstocks

`foreach method in `capstocks_to_create' {` ` foreach vv in `cap_types' {` `  gen neg_capstock_`method'_`vv' = capstock_`vv'_`method' <0` `  by ruref: egen neg_ruref_`method'_`vv' = max(neg_capstock_`method'_`vv')` `  noisily display "Number of negative `curr_name' observations for asset `vv' for method `method'"` ` noisily table letter07 year , contents(mean neg_capstock_`method'_`vv')` ` noisily table letter07 year , contents(mean neg_ruref_`method'_`vv')` ` }` `}`  `gen n_obs=1` `preserve` `noisily display "Summary of negative capstocks"` `collapse (mean) neg_capstock* (sum) n_obs, by(year letter07)` `sort year letter07` `noisily list` `sort letter07 year` `noisily list` `restore`  `preserve` `noisily display "Summary of rurefs with at least one negative capstock"` `sort ruref` `by ruref: drop if _n>1` `collapse (mean) neg_ruref* (sum) n_obs, by(year letter07)` `sort year letter07` `noisily list` `sort letter07 year` `noisily list` `restore` `}`	
`*****************************************************************************` `* Part 5: Deal with negative capital stocks                                *` `* options available: missing_capex rebase_init_stock                        *` `*   missing_capex assumes one or more lumps of missing investment              *` `*   rebase_init_stock assumes initial casptock is too low                      *` `* Program records how many iterations are needed to remove negative cap stocks  *` `*****************************************************************************` `* create vars here and then empty within the loops to speed up programme` `gen lowest = 0`	Now apply the backstop adjustment

```
  foreach method in `capstocks_to_create' {
    foreach vv in `cap_types' {
      gen num_iterations_`method'_`vv' = 0
      gen capex_`method'_`vv'_adj = capex_`vv'
      gen capstock_`vv'_`method'_adj = capstock_`vv'_`method'
      forvalues iteration = 1/`max_backfill_attempts' {
        noisily display "Backfilling for method `method' for asset `vv'; iteration `iteration'"

        * first find the lowest value, check whether its negative, and mark it in a ruref
        * if we have multiple obs on the lowest value, take the last one
        by ruref: egen low_value = min(capstock_`vv'_`method'_adj)
        replace num_iterations_`method'_`vv' = num_iterations_`method'_`vv' +1 if low_value <0
        replace lowest=0
        by ruref: replace lowest = _n if (capstock_`vv'_`method'_adj==low_value) & (low_value<0)
        by ruref: egen last_low = max(lowest)
        replace lowest=0 if last_low!=lowest

        * now backfill, remembering 'low_value' is negative...
        if "`backfill_method'"=="missing_capex" {
          replace capex_`method'_`vv'_adj = capex_`method'_`vv'_adj-low_value if lowest!=0
        }
        else if "`backfill_method'"=="rebase_init_stock" {
          by ruref: replace capex_`method'_`vv'_adj = capex_`method'_`vv'_adj -low_value/((1-
`depreciation_`vv'')^(lowest-1)) if (_n==1) & (lowest!=0)
        }
        else {
          noisily display "No backfill method specified!"
          stop
        }
        by ruref: replace capstock_`vv'_`method'_adj =
(init_share_`method'*sum_capstock_observed_`vv') + capex_`method'_`vv'_adj if _n==1
        by ruref: replace capstock_`vv'_`method'_adj = capstock_`vv'_`method'[_n-1]*(1-
`depreciation_`vv'')  + capex_`method'_`vv'_adj if _n>1

        if `verbose' {
          *produce some stats
          gen test_neg = capstock_`vv'_`method'_adj <0
          by ruref: egen test_neg_ruref = max(test_neg)
          noisily table letter07 year , contents(mean test_neg)
          noisily table letter07 year , contents(mean test_neg_ruref)
          drop test_neg test_neg_ruref
        }
        * always produce this one as it's quite handy
        noisily tab letter07 num_iterations_`method'_`vv'
```

For each of the methods used…
- For each of asset types…
  - For the max number of backstop changes allowed…

Identify which is the lowest capstock value, and by how much

Calculate the appropriate correction to capex

Add the adjusted capex back in the appropriate place, and recalculate the PIM

Produce some stats

drop low_value last_low } } } drop lowest noisily save "$capstock_file", replace	
log close } * end 'quietly'	Fin