Kellogg School of Management

First version: February 7, 2002
Last update: January 25, 2008

PATRICIA LEDESMA LIÉBANA

Retrieving Data from CRSP and Compustat Using the WRDS's server

The most recent version of this document can be found in the "Training & Publications" page in the Research Computing web site:

www.kellogg.northwestern.edu/researchcomputing/training.htm

For a minimal introduction to UNIX and SAS, refer to the "Basic Unix commands, editing files, and transferring files" and "SAS programming skills" handouts available on the same web page. The former handout provides basic secure FTP instructions to transfer files between Kellogg's and Wharton's UNIX servers.

The Research Computing web site also contains complete sets of the CRSP and Compustat documentation, as well as references to published articles that assess the quality of these datasets.

This document provides an introduction to using the WRDS UNIX shell to extract data from CRSP and Compustat by programming in SAS. Even if you are not a SAS user, there are a number of reasons why it pays off to learn the basics of data manipulation in SAS rather than use the WRDS web interface: (i) Easy replicability; (ii) Easier to refine data extraction; (iii) The web interface rounds the output to decimals; (iv) The web interface deletes observations for which the chosen variables have missing values and there is no simple way of finding out what observations were deleted.

1. Before we begin

- **Goals**: Provide and overview of the CRSP and Compustat data and use SAS to access it. SAS is not the only option for CRSP and Compustat, but for other datasets available at WRDS (such as TAQ or Spectrum), the data is only available in SAS data files. CRSP and Compustat are also available in Fortran binary format.

- **Access to WRDS:** The WRDS UNIX server, wrds.wharton.upenn.edu, will be accessed using SSH Secure Shell, an application that can be downloaded from Northwestern's web site. Users may also connect to the WRDS' server using an X Windows emulator or using WRDS' web interface.

- Kellogg subscribes to the annual updates of CRSP and Compustat. New files for each year become available towards the end of the third quarter. For example, the 2006

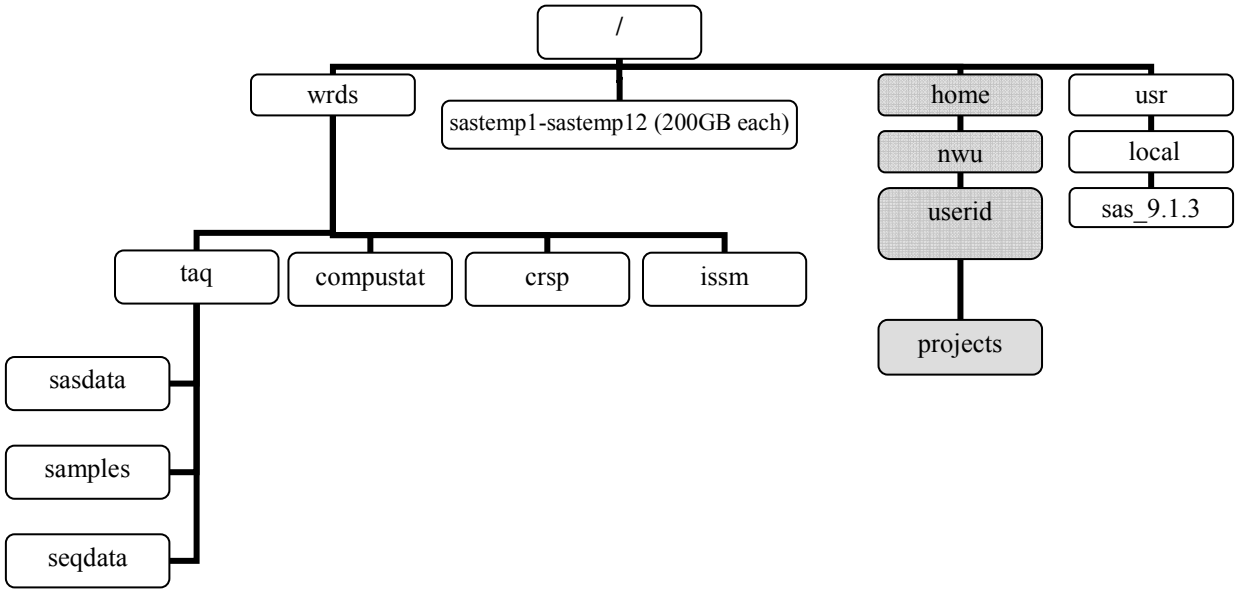
Compustat data will become available around August of 2007; CRSP is generally updated earlier than Compustat, around June.

2. Structure of the WRDS server

- Wharton Research Data Services (WRDS) provides access to a Sun Solaris server that holds datasets to which Kellogg subscribes, as well as software for access. This server can be accessed via the web or via a terminal emulator. The web interface is described in the "Accessing data through Wharton Research Data Services' web interface" handout, which can be used by faculty in MBA classes.

- Software available in the WRDS server includes SAS 8, Fortran 77 and Fortran 90, C, perl, and standard UNIX tools and text editors.

- As any UNIX server, the WRDS' system is a hierarchical file system, shown for the WRDS server in the diagram below.



- All datasets are stored in the "/wrds" volume.

- There are 13 "scratch" volumes, named "sastemp0" through "sastemp12", each with 200GB of space. The content of any of these volumes is deleted without warning should it become full. Thus, make sure to move any needed dataset to your skew3 account or your PC. The space used by any one user in a sastemp volume cannot exceed 32GB. You can check the space available in each with the UNIX command "`df -k | more`".

- *Home directory quotas*: Starting August 5, 2004, each user gets a "home" directory, with 750mb of space. If you need to store datasets larger than the space allowed by your quota, write them in one of the scratch volumes. If you need an increase in your

home directory quota, WRDS will do this at an annual charge per year per GB of space. Currently the charge is $60 per GB, per year.

- For old accounts (created prior to the 8/5/04 change), that had two directories (/home/nwu/userid and /projects/nwu/userid), the projects directory now becomes a sub-directory of the home directory (see picture below). A symbolic link, the Unix equivalent of a windows shortcut, allows programs that had /projects/nwu/userid hard-coded to run without problems.

- Default user directory: When a user logs into the server, the default directory or "home directory" is /home/nwu/userid, where userid is the user's login ID for the system, created by WRDS personnel.

- Users can submit only one job at a time.

- Do not write any files to "/tmp".

3. Logic of CRSP and Compustat

- Extended descriptions of CRSP and Compustat are available in the Research Computing web site:

  www.kellogg.northwestern.edu/researchcomputing/crsp.htm

  www.kellogg.northwestern.edu/researchcomputing/compustat.htm

- Both CRSP and Compustat are collections of files that can best be described as relational databases: the data in each dataset is spread across various files that can be conceived as related "spreadsheets". To link these spreadsheets, one must use one or more variables as to link observations from one table with observations of the other.

- In general terms, there are three types of files or tables in CRSP and Compustat: files with information that identifies individual securities or firms (header files), files with historical individual level information (panel data, such as annual financial statements or daily prices and returns, the promised daily yield on a Treasury bond, etc.) and files with market level information (time series, with information such as inflation, value-weighted returns, total market value, etc).

- Both CRSP and Compustat have generated unique identifiers for the firms and securities they cover. Information such as the company name is hard to match, CUSIPs and tickers may change through the history of a firm or issue; they even may be re-assigned to different companies or issues.

- *CRSP unique identifiers*: In CRSP, each security in the stock file is assigned a unique identifier (PERMNO); each company is given a unique identifier called PERMCO. These numbers are consistent through time, even if CUSIP numbers, tickers or company names change. For instruments in the Treasury files, the unique identifier is CRSPID, while in the mutual funds database the identifier is ICDI.

- *Compustat's unique identifiers*: In Compustat, each company has a unique identifier, GVKEY, assigned by Standard & Poor's. This identifier is the one matched with PERMNO by CRSP to create the Merged CRSP/Compustat database. Many use the combination of CNUM (the 6 digit CUSIP number) and CIC (the two digit CUSIP issue number and a check digit) as the unique identifier. In CRSP, the CUSIP variable is the concatenation of the CUSIP number and the CUSIP issue number. Thus, CUSIP in CRSP is an 8-digit variable. For more information about CUSIP numbers, see www.cusip.com.

- *Compustat fiscal years versus calendar years*: The Compustat annual files include a fiscal year variable (yeara) and a fiscal year-end month of data (fyr). The latter takes values from 1 (January) through 12 (December) depending on the month in which a company's fiscal year ends. There are some observations with fyr=0 which should be deleted; data fields for those observations is typically missing.

  For firms with a fiscal year between January and May, the fiscal year (yeara) lags one year behind the calendar year. For example, for a firm with fiscal year-end of April 30th (e.g. Heinz Co, with gvkey 5568), a fiscal year (yeara) of 2002 really corresponds to April 30th, 2003.

  For the quarterly files, the situation is similar, except that the calendar year assignment should be done by quarters. The fiscal year variable in the quarterly files is called "year", and the fiscal quarter is "qtr". The table below spells out the calendar dates you should use to match with CRSP data. Notice the quarters for which the year needs to be shifted. Example 4 in section 8 walks you through an example of matching monthly CRSP data and quarterly Compustat data.

Fiscal and calendar years

	Quarter end calendar date			
FYR	fiscal quarter 1	fiscal quarter 2	fiscal quarter 3	fiscal quarter 4
1	April 30	July 31	October 31	January 31 **fiscal year + 1**
2	May 31	August 31	November 30	February 28/29 **fiscal year + 1**
3	June 30	September 31	December 31	March 31 **fiscal year + 1**
4	July 31	October 31	January 31 **fiscal year + 1**	April 30 **fiscal year + 1**
5	August 31	November 30	February 28/29 **fiscal year + 1**	May 31 **fiscal year + 1**
6	September 30	December 31	March 31	June 30
7	October 31	January 31	April 30	July 31
8	November 30	February 28/29	May 31	August 31
9	December 31	March 31	June 30	September 31
10	January 31	April 30	July 31	October 31
11	February 28/29	May 31	August 31	November 30
12	March 31	June 31	September 30	December 31

- At the end of this document, there is a list of the datasets included in Kellogg's subscription to these datasets.

- *Time-saving tip for Compustat users*: Many users frequently search three of Compustat's databases to find a specific company (industrial, research and full coverage files). Wharton has combined the three files into one in the SAS version of the data (compann.sas7bdat and compqtr.sas7bdat). There is no comparable version for Fortran. WRDS created these files so that its web interface can perform the queries submitted by subscribers using the web interface. The WRDS web interface consists of a number of perl scripts that customize and run SAS programs.

4. Searching for a company

To search for the PERMNO and CUSIP in CRSP, or for CNUM and DNUM in Compustat, you may use the UNIX "grep" command on the names files from the Fortran data collection. The header files are in ASCII format.

"grep" searches a file line-by-line for a given pattern. Whenever it finds a match for the string, it prints the corresponding line to the screen. The "-i" switch makes "grep" ignore the case of the provided string.

At the prompt in the WRDS Unix server, use the following commands:

For CRSP:

```
grep –i "company name" /wrds/crsp/seqdata/smi/cheadfile.dat
```

For example,

```
grep -i "ibm" /wrds/crsp/seqdata/smi/cheadfile.dat
```

```
12490 20990 45920010 INTERNATIONAL BUSINESS MACHS CO IBM    1 19251231-20051230
75139 22064 03093810 AMERICUS TR FOR IBM SHS         BZP    2 19870731-19920630
75140 22064 03093820 AMERICUS TR FOR IBM SHS         BZS    2 19870731-19920630
75141 22064 03093830 AMERICUS TR FOR IBM SHS         BZU    2 19870731-19920630
```

The columns, left to right, are: PERMNO, CUSIP, Header SIC code, ticker, company name, exchange code and start to end date range for price data.

For Compustat:

```
grep –i "company name" /wrds/compustat/seqdata/ina.names
```

(or res.names or fca.names)

For example:

```
grep -i "ibm" /wrds/compustat/seqdata/ina.names
```

```
7370 459200 101 IBM      INTL BUSINESS MACHINES CORP
```

The columns, left to right, are: DNUM (industry classification code), CNUM (CUSIP issuer code), CIC (CUSIP issuer number and check digit), SMBL (ticker symbol) and company name.

Note that the Compustat names files for Fortran do not include GVKEY. SAS users may write a simple program that searches for a pattern in the ticker (SMBL) and company name (CONAME) variables:

```
data busqueda;
  set comp.namesann;
  where CONAME contains 'IBM' or SMBL contains 'IBM';
proc print data=busqueda;
```

The advantage of this program over the "grep" command is that it will query the combined names file (industrial, full coverage and research names files).

5. WRDS' setup for SAS users

- SAS library names are already defined in all user accounts. Thus, users can invoke those names without defining them (with a LIBNAME statement) in their programs. The configuration of these LIBNAMES is done within a file called "autoexec.sas" in each user's account. Any option or LIBNAME used frequently can be added to this file, which is automatically executed by SAS before any other command. Do not move the autoexec.sas to any subdirectory or it will not work.

- You can customize your "autoexec.sas". For example, my personal preferences for the autoexec.sas:

  ```
  options ls=120 nodate nocenter ps=max formdlim=' '
     nonumber;
  title;
  libname out '~/projects ';
  ```

  These options only have impact on the output (LST) file. They set or eliminate, from left to right: the line size (characters printed per line), date, centering of output, page size, page delimited (set to a space instead of a page break), page numbering. The "title" statement by itself eliminates the SAS default headline ("The SAS System") at the top of every page. I also assign a LIBNAME to a subdirectory in my account where I store some datasets.

- To follow the examples starting in section 8 below, please create a directory called "projects" in your account (`mkdir projects`). Include the following statement in your autoexec.sas and save it:

  ```
  libname out '~/projects';
  ```

  where userid is your WRDS' login ID. Save the changes in the autoexec.sas; they will be used in the examples in section 6 below.

- In your autoexec.sas, you will see the following command:

```
%include '!SASROOT/wrdslib.sas' ;
```

This command calls and executes a SAS programs saved in the directory where SAS is installed (/usr/local/sas_9.1.3). It defines all the SAS libnames for data existing in WRDS (whether we subscribe or not. Do not remove this command. A list of important LIBNAMES already assigned by WRDS through the `%include` statement in the autoexec.sas is provided in the table below:

Sample of WRDS pre-assigned LIBNAMES

Data set	SAS libref	Corresponding directory
Bank regulatory	bank	/wrds/bank/sasdata
Compustat	comp	/wrds/compustat/sasdata
CRSP	crsp	/wrds/crsp/sasdata
Dow Jones averages	dj	/wrds//dj/sasdata
DRI	dri	/wrds/dri/sasdata
Fama French	ff	/wrds/ff/sasdata
FDIC	fdic	/wrds/fdic/sasdata
IRI	iri	/wrds/iri/sasdata
IRRC	irrc	/wrds/irrc/sasdata
ISSM	issm	/wrds/issm/sasdata
PHLX	phlx	/wrds/phlx/sasdata
SEC Disclosure of Order Execution	doe	/wrds/doe/sasdata
TAQ	taq	/wrds/taq/sasdata
Thomson Financial	tfn	/wrds/tfn/sasdata

The `%include` SAS statement allows you to bring SAS code stored in a separate program file.

6. Using SAS sample programs

Sample SAS and Fortran programs for CRSP and Compustat are available in the following paths:

/wrds/compustat/samples

/wrds/crsp/samples

To use and modify one of the available programs, you can copy it to your home directory.

- WRDS' sample files include SAS programs that allow the user to run a CAPM model, create a dataset suitable for an event study, etc.
- A typical mistake after from inexperienced SAS users is to use PROC PRINT to generate an ASCII file for use in another application. Use the FILE, PUT statements or the PROC EXPORT procedures instead. Refer to the *SAS Programming Skills* handout, item 15, for sample code and tips.

Users of other statistical packages, such as Stata, Limdep or SPSS, may use StatTransfer, a utility on Kellogg's UNIX server, skew3, to translate a SAS data file to the desired format. The advantage of this option over creating an ASCII file is that all the variable information (name, formatting, labels, etc) is preserved for use by the other package. Refer to:

www.kellogg.northwestern.edu/researchcomputing/stattransfer.htm

SAS dates and times

Most of the datasets used in finance include some date variable. TAQ includes a date and time stamps for each trade and quote. These dates and times are in SAS date and time formats.

- SAS dates are integers that reflect the number of days elapsed since the SAS epoch, January 1, 1960 (which is stored as 0). For example, the number 15389 represents the date February 18, 2002 as a SAS date. SAS can display this numbers as a readable date in a variety of formats: "February 18, 2002" (with the `worddate20.` format), "18FEB2002" (`date9.` format), "2002:1" (`yyqc6.` format), "20020218" (`yymmddn8.` format), etc. In any of these cases, the date value is 15389. To merge datasets with SAS dates, you do not have to change the format, since it is only a display format.

- SAS times are integers that reflect the number of seconds elapsed since midnight of the current day.

- There is an additional SAS format, the "`DATETIME`" format. These values are integers that represent the number of seconds elapsed since midnight January 1, 1960.

- There are a number of functions that allow calculations with dates: YEAR, MONTH, DAY, WEEKDAY, QTR, etc. For example, you may want to retrieve data for all the trading Fridays during 1999. Using a "`where weekday(datevar)=6;`" would subset Fridays.

- Specific dates and date/times are easy to pass to a SAS program, enclosing the date (e.g., 08feb1999) in single quotes and adding a character that specifies the data type. For example, "`if date ge '08feb1999'd`" would subset observations on or after February 8, 1999, while "`if time ='16:00't`" would subset observations with a time stamp of 4:00pm.

7. Examples of data extraction using SAS

Given the structure of CRSP and Compustat (described in section 3), there are generally two ways of working with these datasets:

1. Starting with a set of variables that identify certain securities or firms (usually tickers or CUSIP numbers), you may query the header files, retrieve the unique IDs (PERMNO or GVKEY) for the relevant observations and then subsets the main data files.

2. Start by sub setting the main data files according to a specific criterion. For example, more than 50,000 employees or a volume traded greater 4.6 million shares at any point during the year 2000. At some point during the analysis (looking at outliers or looking for control variables such as industry or age of the firm/issue), you will need to know more about the data and will use the unique identifiers to retrieve the needed information from the header files.

In any of these two cases, you may add time series information such inflation, the value-weighted return for the market or the level of the Standard & Poor's 500 Composite Index.

The following examples all use short time series and a limited number of firms so that the examples run relatively fast in the WRDS' server.

Example 1: Select variables from Compustat for a series of tickers

In this example we use a list of tickers to subset a group of variables from the Compustat annual file (using WRDS' combined version) for the 1996 through 2000 period.

The logic of the exercise is the following: Researchers who start with a list of companies rarely have the Standard & Poor's assigned GVKEY. Rather, they start either with stock market ticker symbols or CUSIP numbers. In a first step, we want to verify that the matches we find are indeed the firms we are looking for, so we query the combined names file. In the second step, we use the matched GVKEYs to subset the variables and time period of interest.

Matching firms by their name is much harder than by any other identifier. Different data sets may use different abbreviations (e.g., CORP. versus CORP or CORPORATION), and mistakes are possible, etc. SAS has tools to deal with this type of matching, including a SOUNDEX function. Users interested in this may look for references about "fuzzy merging".

1. Checking that the tickers match the expected firms: Suppose you have collected the following list of companies and their corresponding tickers. Select 4-5 tickers for this exercise. If you want to try the entire list, it is available for copying in the following URL:

www.kellogg.northwestern.edu/researchcomputing/workshops/updata/tickers.htm

Selected S&P 500 Industrials Constituents as of 14 Feb 2002

Ticker	Company name	Ticker	Company name
ADP	Automatic Data Processing Inc.	GWW	Grainger (W.W.) Inc.
AVY	Avery Dennison Corp.	ITT	ITT Industries, Inc.
CAT	Caterpillar Inc.	ITW	Illinois Tool Works
CTAS	Cintas Corporation	LMT	Lockheed Martin Corp.
DHR	Danaher Corp.	MMM	Minn. Mining & Mfg.
EMR	Emerson Electric	NOC	Northrop Grumman Corp.
ETN	Eaton Corp.	PCAR	PACCAR Inc.

Ticker	Company name	Ticker	Company name
FDC	First Data	PH	Parker-Hannifin
FDX	Federal Express	UNP	Union Pacific
GD	General Dynamics	UTX	United Technologies

Source: S&P Global Data [http://www.spglobaldata.com/]

There are two options to feed this data into SAS. You can either (a) create a text file with the list of tickers, and then read it with SAS; or (b) include the tickers in the SAS program. We will opt for the first option, creating a file called "ticker.txt". Note that the name and extension of the file do not matter to SAS.

2. Create the text file with the tickers you select from the table above. Type one ticker per line, hit enter to go to the next line, type the next ticker, etc. Save the file.

3. Create a new program file (call it "ticksel.sas") and type the following commands:

```
filename ticklist 'tickers.txt';
data readlist;
 infile ticklist;
 input smbl $;
 smbl=upcase(smbl);
proc print data=readlist;
```

The first statement simply assigns a nickname to the file with the ticker list. In the DATA step that follows, we read in the list as an alphanumeric string (hence the "$") to match the name and format of the ticker variable in Compustat (SMBL). As a precautionary step, since all tickers in Compustat (and CRSP) are in uppercase, we can make sure our data is uppercase using the UPCASE function. To make sure our tickers were read correctly, print the list. Run this short program and make sure it works.

4. If you had not problem reading the input file, you are ready to query the names dataset (called "namesann" in the "comp" library). Type the following commands:

```
proc sql;
  create table ticksel as select
  readlist.*, namesann.coname, namesann.gvkey, namesann.dnum
  from readlist, comp.namesann
  where readlist.smbl = namesann.smbl;
quit;
proc print data=ticksel;
```

Notice in the PROC SQL that there are no semi-colons in CREATE statement until the end of the subsetting conditions (WHERE). The spacing and indentation is arbitrary, as in any SAS command.

Tip – limit number of observations: If you are not comfortable with SAS (or with SAS PROC SQL), you may restrict how many observations are written out using the "OUTOBS=" option in the PROC SQL statement; you can also limit the number of

observations that are processed (read from the data files) with the "INOBS=" option . For example, if you just wanted to see the first 10 matches to check if the conditions are working correctly, the PROC SQL statement would read:

```
proc sql outobs=10;
```

In this case, it is convenient to use PROC SQL instead of a merge because "namesann" is not sorted by SMBL and neither may be our input file. With this statement we select the following variables from "namesann": CONAME, GVKEY, and DNUM. In the "WHERE" statement part of SQL we match SMBL in "namesann" with our list of tickers in "readlist". In this example, we subset only those lines of common to "namesann" and "readlist". To verify that we retrieved the firms we needed, we print the list of matches and check them against the original list in the table in [#1].

5. If there are no errors in the program and the matches are correct, we can now subset the variables we need for the period we want. Select two variables from the table below:

Selected Compustat variables

Variable name	*Short description*
data3	Inventories - Total (MM$)
data4	Current Assets - Total (MM$)
data5	Current Liabilities - Total (MM$)
data6	Assets - Total (MM$)
data20	Income before extraordinary items - Adjusted for common stock equivalents (MM$)
data29	Employees (M)
data36	Retained Earnings (MM$)

Now we are ready to create our sample data file. Unlike the previous datasets we created in this example ("readlist", the list of tickers; and "ticksel", the list of matched tickers and GVKEY numbers), we will write the result of this selection to our projects directory.

The following PROC SQL restricts the variables to data6 and data36 (plus our identification variables, which are already in the "ticksel" data file:

```
proc sql;
  create table out.tickselyr as select
  ticksel.*, compann.yeara,
  compann.data6, compann.data36
  from ticksel, comp.compann
  where ticksel.gvkey = compann.gvkey
    and yeara between 1996 and 2000;
quit;
proc print data=out.tickselyr;
```

Notice that in addition to the variables (data6 and data36), I also retrieve GVKEY and the fiscal year (YEARA) to match and subset the data. We restrict the data to the years 1996 through 2000. The same portion could have been written as "`1996=<yeara=<2000`" or "`yeara in (1996 1997 1998 1999 2000)`". The latter allows you to skip years.

Also notice that in the resulting dataset, each line in ticksel was matched to 5 rows in compann (one row for every year from 1996 through 2000). As a matter of fact, up until the WHERE portion of the SQL statement, SAS created the Cartesian product of the rows in each dataset. The WHERE condition pared the rows down to those we needed.

Once this program runs correctly, we are ready to transfer the resulting data file to its final destination (your computer or skew3).

6. A complete program for this example is available in the following URL:

www.kellogg.northwestern.edu/researchcomputing/workshops/updata/example1.htm

Example 2: Select variables for all the firms in Compustat that belong to a series of industries

This example is very similar to the first one, except that we are less likely to match the wrong industries (unlike the case of tickers, CUSIP numbers or company names), since the same industry code may not be assigned to a different industry.

The current version of Compustat includes two industry classifications: the 1987 SIC classification and the "new" 1997 North American Industrial Classification (NAICS). The variable for the 1987 SIC classification is called "DNUM", while the corresponding to the 1997 NAICS is "NAICS".

For more links to sites with complete listings of industrial codes and concordance tables, and click on "Reference & Papers" in the Research Computing web page. This page contains a link to another page on "Data biases and statistical classifications".

1. For this exercise, we will select two or three of the manufacturing industries listed in the following table, as well as two of the variables listed in the previous table. I have copies the number of firms in each of them in 2000 to get an estimate of the number of observations we will retrieve.

Selected industrial codes for Compustat

NAICS code	Firm count in 2000	NAICS description
334512	13	Automatic environmental controls for monitoring and regulating residential and commercial environments and appliances
311612	20	Meat Processed from Carcasses
322130	20	Paperboard Mills

NAICS code	Firm count in 2000	NAICS description
325611	20	Soap and Other Detergent Manufacturing
325612	20	Polish and Other Sanitation Good Manufacturing
327310	20	Cement Manufacturing
334612	20	Prerecorded Compact Disc (except Software), Tape, and Record Reproducing
311611	21	Animal (except Poultry) Slaughtering
321991	21	Manufactured Home (Mobile Home) Manufacturing
333111	22	Farm Machinery and Equipment Manufacturing

2. Check the WRDS online documentation to verify the format of the NAICS variable. You will notice that it is a 6-character variable. Hence, to select the industries, the NAICS codes must be enclosed in quotations.

3. Create a new program file. Call it "naics-select.sas" and type the following commands (replace in your own choice of industries and variables); you may also add other descriptive information, such as the old industrial classification (DNUM) or "FINC" the incorporation country code for foreign companies.

```
proc sql;
  create table naicsyrs as select
  compann.gvkey, compann.naics, compann.yeara,
   compann.smbl, compann.coname, compann.data6,
      compann.data36
  from comp.compann
  where compann.naics in ("322130" "334612") and
      yeara between 1996 and 2000;
quit;
proc print data=naicsyrs;
```

4. As in the previous case, you can restrict how many observations are written out with the "OUTOBS=" option in PROC SQL (see [#4] in example 1).

5. A complete program for this example is available in the following URL:

www.kellogg.northwestern.edu/researchcomputing/workshops/updata/example2.htm

Example 3: For a given set of tickers, find their permnos in CRSP and check the events file

In this example we will query the CRSP events file. This file has information about events such as distributions, delisting, name changes, etc. In the *CRSP Data Descriptions Guide*, check the chapter on "CRSP Data Coding Schemes." That chapter will contain tables that describe the coding of the various events.

1. Create a new program called "crsp-event.sas". Select any three companies from the table in example 1 (page 8-9), including one of the following tickers: ADP, FDC, UNP. For example:

```
data crsp1;
   set crsp.msfnames (keep=permno ticker comnam
     st_date end_date);
   where ticker in ("ENE" "ADP" "FDC");
proc print data=crsp1;
```

   You will notice in the output that more than once company has the same ticker symbols and that the same permnos are repeated. Keep in mind that tickers can be recycled.

   If this was data for your research project, you should: (i) make sure you keep the permnos corresponding to the companies you are interested; and (ii) examine the repeated permnos by looking at the events file. In this particular example, the repeated permnos are due to name changes (Allied Products Corp).

   The second problem that jumps at you is that there are two permnos with the same ticker (ADP). How would you deal with this? First of all, in the output, check the start (st_date) and ending (end_date) dates of the CRSP series. One of them ends in October 31, 2000. Again, you want to check the events file. You could write the following commands:

```
data test;
   set crsp. mseall (keep=ticker permno comnam
      exchcd date distcd dlstcd);
   where ticker='ADP' and year(date)=2000;
proc print data=test;
```

   The variables exchcd, distcd and dlstcd are codes for the exchange, distribution and delisting, respectively. In the output to this program, you will see that Allied Products Corp in October 31, 2000 has a delisting code of "574". If you check the CRSP Coding Schemes chapter of the CRSP US Stock and Indices Database Data Descriptions Guide, you will find that 574 is a delisting due to bankruptcy or insolvency.

   There are two flavors of the monthly stock events file: mse and mseall (or, for the daily data, dse and dseall). The latter is one where each event is one observation, instead of several events in one.

In practice you may be using hundreds of tickers. One simple way of narrowing down on the permnos you are after, is to use the dates of interest to you and compare those to the starting date of the series (st_date), and the end date of the series (end_date – for active stocks, this will always correspond the vintage of the CRSP file) in the msfnames file. For example, if you needed data for 1998 through 2001, a better way of writing the query of msfnames would be:

```
data crsp1;
   set crsp.msfnames (keep=permno ticker comnam
     st_date end_date);
```

```
where ticker in ("ENE" "ADP" "FDC")
 and year(st_date)<=1998 and year(end_date)>=2001;
```

In this specific example, this would narrow the query to precisely the tickers we want. Suppose, however, that it did not and you had hundreds of tickers for which you need to query the events file. A way to generate the query is to create a list of the problematic permnos, i.e., tickers for which you got more than one permno in the period of interest:

1. Use the NODUPKEY option in a PROC SORT to generate a list that has all the permno-ticker combinations. Here, I also use the OUT= option to generate an output file and leave the crsp1 dataset intact:

```
proc sort data=crsp1 nodupkey out=checkdups;
  by ticker permno;
```

2. Use PROC FREQ to create a list that will have only the tickers that have "multiple" permno (note the WHERE condition on the output dataset, checkdups2):

```
proc freq data=checkdups noprint;
   tables ticker
     / out=checkdups2 (where=(count>=2) keep=ticker count);
```

3. Then, recover the permnos by using the list of tickers generated with PROC FREQ (checkdups2) and selecting the matching permnos in our original list (checkdups):

```
proc sql;
   create table checkdups3 as select
    checkdups, checkdups2
   where checkdups2.ticker=checkdups.ticker;
quit;
```

4. Finally, query the events file:

```
proc sql;
  create table checkevents as select
   mseall.ticker, mseall.permno, mseall.comnam,
   mseall.shrcd, mseall.exchcd, mseall.shrcls, mseall.date,
   mseall.dlstcd
  from checkdups3, crsp.mseall
  where checkdups3.permno=mseall.permno
     and year(mseall.date) between 1998 and 2001
quit;
```

5. A complete program for this example is available in the following URL:

www.kellogg.northwestern.edu/researchcomputing/workshops/updata/example3.htm

Example 4: For a given set of tickers, retrieve data from CRSP and match it to Compustat data

In the previous examples, we have encountered a couple of issues: the first is that one of multiple matches in CRSP for a given stock ticker symbol. The key to solving this problem is know what company we are looking for and narrow the data retrieval period to the one in which the company was in existence. More important issues are those of matching company data in CRSP and Compustat, and making sure we address the fiscal year issue. This example shows one way of addressing these issues.

Create a new SAS program called "fiscal.sas"

1. For this example, we will start by providing a list of GVKEYS within the SAS program:

```
data firms;
input gvkey;
cards;
1240
1468
1072
3980
27828
;
proc print data=firms;
```

2. Now we retrieve some annual Compustat data (assets and retained earnings) for our selected firms between the fiscal years 2000 and 2004:

```
proc sql;
 create table firms2 as select
   firms.gvkey, compann.data6, compann.data36, compann.yeara,
   compann.coname, compann.smbl, compann.fyr
 from firms, comp.compann
 where firms.gvkey=compann.gvkey
and compann.yeara between 2000 and 2004;
quit;
proc print data=firms2;
```

We also retrieve some quarterly Compustat data, including gvkey, fiscal year ("year": in the quarterly file), assets (data44), the earnings announcement date (rdqe). Finally, we create a calendar date for each quarter using a couple of SAS functions.

```
proc sql;
   create table firms3 as select
   firms.gvkey, compqtr.year as yeara,
   compqtr.data44, compqtr.rdqe, compqtr.qtr,
     intnx('month',yyq(compqtr.year,compqtr.qtr),2,'end')
     as date format=yymmddn8.
  from firms, comp.compqtr
```

```
  where firms.gvkey=compqtr.gvkey

and compqtr.year between 2000 and 2004;
quit;
proc print data=firms3;
```

The expression "intnx(•) as date format=yymmddn8." creates this calendar date and saves it to our output file as a variable called "date" with format yymmddn8..

The building blocks of this are as follows: first we use the YYQ(*year*,*quarter*) function to create a date that will be the first day of the quarter: January 1, April 1, July 1 or October 1 of the fiscal year in question. Clearly, this not the usual way we would think about the date. Hence, we use the INTNX function to "shift" our date to the last day of the quarter (*i.e.*, by two months): March 31, June 30, September 30 or December 31st of each fiscal year. The INTNX function is a date and time function that adds an interval to a date (or time) and returns another date (or time). The syntax is:

```
INTNX(interval<multiple><.shift-index>, start-from, increment<,alignment>)
```

3. Now merge the annual and quarterly data. We will use a DATA step, which requires having both datasets sorted by the variables used as keys for the merge. In this DATA step we also address the fiscal year issue by defining a new date called "calendar":

```
proc sort data=firms2; by gvkey yeara;
proc sort data=firms3; by gvkey yeara;
data cstatfirms;
 merge firms2 firms3;
 by gvkey yeara;
 if 0<fyr<=5 then calendar=
 intnx('month',mdy(month(date),day(date),yeara+1),0,'end');
 if fyr>5 then calendar=date;
 format calendar yymmddn8.;
 if fyr=0 then delete;
proc print data=cstatfirms;
```

Notice that "calendar" is defined differently based on the fiscal year.

4. Using the Merged CRSP/Compustat link file, we look up the permnos for our firms, in order to get some CRSP data. We only use some specific link types. Check the *Merged CRSP/Compustat Data Definitions* manual for the meaning of each link type:

```
proc sql;
 create table permlist as select
   cstlink2.*
 from firms, crsp.cstlink2
 where firms.gvkey=cstlink2.gvkey and
  cstlink2.linktype in
    ("LU" "LC" "LD" "LF" "LN" "LO" "LS" "LX");
quit;
proc print data=permlist;
```

Notice in the output of the PROC PRINT above that instead of "permno" and "permco", CSTLINK2 contains "npermno" and "npermco." This is a special version of permno and permco that is augmented by additional codes, such as npermcos of 0 for firms in Compustat that do not have a match in CRSP. For companies that have a match, the npermno is the same as the permno. As a result of the query above, we also retrieve two additional and useful variables: linkdt, the starting date for the validity of a link (a missing value indicates that the link was valid before there was data for that firm in Compustat), and linkenddt, the ending date for the validity of a link (with a missing value indicating that the link is valid beyond the current vintage of CRSP data).

5. We retrieve returns and dates from the CRSP monthly stock file. Notice how we use permno and npermno to do the match and, furthermore, we compare the starting and ending date for the validity of links to ensure that we are choosing the right series for each time period.

   Since CRSP includes trading days, the last trading day of the month is not often the last calendar day of the month. To simplify our lives, we can "shift" the date to align it with the end of each month. To check that the shifting worked as expected, we print the observations for selected variables:

```
proc sql;
  create table crspfirms as select
  permlist.*, msf.ret, msf.date
  intnx('month',date,0,'end') as calendar format=yymmddn8.
 from permlist, crsp.msf
   where permlist.npermno=msf.permno
     and (msf.date between '01jan2000'd
     and '31dec2004'd) and (permlist.linkdt<=msf.date
       or permlist.linkdt=.)
     and (msf.date<=permlist.linkenddt
       or permlist.linkenddt=.);
quit;
```

6. And, finally, we merge the Compustat and CRSP data, again printing selected variables for a final check.

```
proc sort data=crspfirms; by gvkey calendar;
proc sort data=cstatfirms; by gvkey calendar;
libname mydata '~/projects';
data mydata.alldata;
 merge cstatfirms crspfirms;
 by gvkey calendar;
proc print data=alldata;
  var gvkey permno yeara fyr date calendar ret data6 data44;
```

   At this stage you can also define a LIBNAME if you have not done so already, and save the resulting data set as a "permanent" file in your home directory. If it is a very large file, you can also save it in one of the scratch volumes (sastemp0-sastemp12) and transfer it with sFTP.

8. Final tips

- Always check your programs on a reduced sample of the data to make sure your selection conditions work. If your program processes the data and you are getting errors, write a simpler program and build up from there to determine where the error (or errors) is (are).

- When naming datasets and variables, avoid words that could be SAS keywords. If the word you chose is actually a keyword (for example, "output"), your program may produce cryptic error messages in the context of a macro, for example.

- To try new routines for data manipulation, clean up, etc, create small, fake, datasets with 3 or 4 variables, 10 observations (enough to show the different cases in your data, including missing values).

- Common SAS mistakes: leaving a quote (or double-quote) open, forgetting a semi-colon at the end of a command, mistyping the name of a variable or dataset, and leaving a necessary portion of a program as a comment.

- To save time, before transferring a data file from WRDS to skew3 or your personal workstation, compress it; use "gzip" or "zip". If you are transferring to skew3, initiate the sftp session in skew3; with a lower processing load on the CPUs, the transfer will be faster than if you start the sftp session in WRDS.