# Basic SAS skills

January 21, 2007

# Why SAS?

- Can deal with very large datasets – I/O
- All datasets in WRDS are in SAS
- Many features built in:
  - Standard Query Language (PROC SQL)
  - Matrix language (PROC IML)
  - Non linear programming (PROC NLP)

# Syntax rules

- All commands end in a semi-colon
- SAS statements are not case sensitive. You may use upper or lower case.
- Variables names can be upper or lower case.
- When you refer to "external" file name, it is case-sensitive (interaction with UNIX operating system)
- Commands can extend over several lines as long as words are not split
- You may have more than one command per line
- SAS name rules (for datasets and variables): up to 32 characters long; must start with a letter or an underscore (“_”). Avoid special characters.

# Parts of a SAS program

- OPTIONS (control appearance of output and log files)
  - SAS programs produce an output file (.lst) with the printed results of commands
  - Log files (.log) with processing messages
- DATA step, where you manipulate data
- PROC (procedures) are used to run analysis on existing datasets,
  - PROC steps can generate datasets that can be modified in a DATA step
- “Output Delivery System” (ODS)
- Availability of functions to process perl regular expressions (prxparse, prxmatch, prxsubstr, prxposn, prxnext, prxparen)

# How SAS works

- SAS reads and executes data steps statement by statement, observation by observation.
  - All variables in the portion of memory that processes the current each observation are reset to missing in each iteration of the data step. The RETAIN statement prevents this from happening.
- Missing values are negative infinity, denoted by “.”.
  - Operators “propagate” missing values (e.g., “v1+v2+v3”)
  - Functions typically ignore missing values (e.g., SUM(v1,v2,v3)

# Using the SAS documentation

- http://support.sas.com/onlinedoc/913/docMainpage.jsp
- Basic modules for us:
  - BASE (all about DATA steps, SQL, SORT, and a few stat procedures)
  - STAT: REG, ANOVA, PROBIT, etc
  - ETS: Arima, autoreg, expand, qlim, syslin, tscsreg
  - IML (matrix language)
  - OR (nlp)

# Additional sources

- “The Little SAS Book”
- SAS Global Forum proceedings for papers about specific subjects (e.g., “fuzzy merging”):

  http://support.sas.com/events/sasglobalforum/2008/index.html

# Three files:

- Your program: example.sas
  - To run: “sas example.sas”
- Output: example.lst
- Log: example.log

# A simple SAS file

```
options nocenter linesize=80;

libname here '.';

/* === Note: crsp. is the libname for CRSP data on WRDS === */

proc contents data=crsp.msf;
run;
```

The CONTENTS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | CRSP.MSF | Observations | 3656362 |
| Member Type | DATA | Variables | 21 |
| Engine | V9 | Indexes | 5 |
| Created | Friday, February 06, 2009 03:40:33 PM | Observation Length | 168 |
| Last Modified | Friday, February 06, 2009 03:41:47 PM | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | YES |
| Label | Monthly Stock - Securities | | |
| Data Representation | HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64 | | |
| Encoding | latin1 Western (ISO) | | |

Engine/Host Dependent Information

| | |
|---|---|
| Data Set Page Size | 16384 |
| Number of Data Set Pages | 37696 |
| First Data Page | 1 |
| Max Obs per Page | 97 |
| Obs in First Data Page | 68 |
| Index File Page Size | 8192 |
| Number of Index File Pages | 32526 |
| Number of Data Set Repairs | 0 |
| File Name | /wrds/crsp/sasdata/sm/msf.sas7bdat |
| Release Created | 9.0101M3 |
| Host Created | SunOS |
| Inode Number | 360 |
| Access Permission | rw-r--r-- |
| Owner Name | wrdsadmn |
| File Size (bytes) | 617619456 |

Alphabetic List of Variables and Attributes

| # | Variable | Type | Len | Format | Informat | Label |
|---|---|---|---|---|---|---|
| 18 | ALTPRC | Num | 8 | 12.5 | 12.5 | Price Alternate |
| 20 | ALTPRCDT | Num | 8 | YYMMDDN8. | YYMMDD6. | Alternate Price Date |
| 14 | ASK | Num | 8 | 11.5 | 11.5 | Ask |
| 9 | ASKHI | Num | 8 | 12.5 | 12.5 | Ask or High Price |
| 13 | BID | Num | 8 | 11.5 | 11.5 | Bid |
| 8 | BIDLO | Num | 8 | 12.5 | 12.5 | Bid or Low Price |
| 16 | CFACPR | Num | 8 | | | Cumulative Factor to Adjust Prices |
| 17 | CFACSHR | Num | 8 | | | Cumulative Factor to Adjust Shares/Vol |
| 1 | CUSIP | Char | 8 | 8. | 8. | CUSIP Header |
| 7 | DATE | Num | 4 | YYMMDDN8. | | Date of Observation |
| 5 | HEXCD | Num | 8 | 2. | 2. | Exchange Code Header |
| 6 | HSICCD | Num | 8 | 8. | 8. | Standard Industrial Classification Code |
| 4 | ISSUNO | Num | 8 | 8. | 8. | Nasdaq Issue Number |
| 3 | PERMCO | Num | 8 | 8. | 8. | PERMCO |
| 2 | PERMNO | Num | 8 | 8. | 8. | PERMNO |
| 10 | PRC | Num | 8 | 12.5 | 12.5 | Price or Bid/Ask Average |
| 12 | RET | Num | 8 | 11.6 | 11.6 | Returns |
| 21 | RETX | Num | 8 | 11.6 | 11.6 | Returns without Dividends |
| 15 | SHROUT | Num | 8 | | | Shares Outstanding |
| 19 | SPREAD | Num | 8 | 11.5 | 11.5 | Spread Between Bid and Ask |
| 11 | VOL | Num | 8 | 10. | 10. | Volume |

# proc print; dataset options

```
proc print data=crsp.msf (obs=10);
run;

proc print data=crsp.msf (obs=100);
    var permno date prc ret;
run;
```

**crsp.msf conventions:**

- Missing data codes in “ret” (referred as “.C” and “.” )
- ASKHI:
  - High price if there was trading
  - Otherwise highest ask: denoted as Negative!
- BIDLO: Low price if traded; otherwise lowest bid
- PRC: close if there was trading; otherwise P=–(BID+ASK)/2

| Obs | CUSIP | PERMNO | PERMCO | ISSUNO | HEXCD | HSICCD | DATE | BIDLO |
|---|---|---|---|---|---|---|---|---|
| 1 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19851231 | . |
| 2 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860131 | -2.50000 |
| 3 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860228 | -3.25000 |
| 4 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860331 | -3.25000 |
| 5 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860430 | -4.00000 |
| 6 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860530 | -3.06250 |
| 7 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860630 | -2.90625 |
| 8 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860731 | -2.59375 |
| 9 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860829 | -1.03125 |
| 10 | 68391610 | 10000 | 7952 | 10396 | 3 | 3990 | 19860930 | -0.96875 |

| Obs | ASKHI | PRC | VOL | RET | BID | ASK |
|---|---|---|---|---|---|---|
| 1 | . | . | . | . | . | . |
| 2 | -4.43750 | -4.37500 | 1771 | C | . | . |
| 3 | -4.37500 | -3.25000 | 828 | -0.257143 | . | . |
| 4 | -4.43750 | -4.43750 | 1078 | 0.365385 | . | . |
| 5 | -4.31250 | -4.00000 | 957 | -0.098592 | . | . |
| 6 | -4.21875 | -3.10938 | 1074 | -0.222656 | . | . |
| 7 | -3.29688 | -3.09375 | 1069 | -0.005025 | . | . |
| 8 | -3.43750 | -2.84375 | 1163 | -0.080808 | . | . |
| 9 | -2.62500 | -1.09375 | 3049 | -0.615385 | . | . |
| 10 | -1.28125 | -1.03125 | 3551 | -0.057143 | . | . |

| Obs | SHROUT | CFACPR | CFACSHR | ALTPRC | SPREAD | ALTPRCDT | RETX |
|---|---|---|---|---|---|---|---|
| 1 | . | . | . | -2.56250 | . | 19860107 | . |
| 2 | 3680 | 1 | 1 | -4.37500 | 0.25000 | 19860131 | C |
| 3 | 3680 | 1 | 1 | -3.25000 | 0.25000 | 19860228 | -0.257143 |
| 4 | 3680 | 1 | 1 | -4.43750 | 0.12500 | 19860331 | 0.365385 |
| 5 | 3793 | 1 | 1 | -4.00000 | 0.25000 | 19860430 | -0.098592 |
| 6 | 3793 | 1 | 1 | -3.10938 | 0.09375 | 19860530 | -0.222656 |
| 7 | 3793 | 1 | 1 | -3.09375 | 0.06250 | 19860630 | -0.005025 |
| 8 | 3793 | 1 | 1 | -2.84375 | 0.06250 | 19860731 | -0.080808 |
| 9 | 3793 | 1 | 1 | -1.09375 | 0.06250 | 19860829 | -0.615385 |
| 10 | 3793 | 1 | 1 | -1.03125 | 0.06250 | 19860930 | -0.057143 |

| Obs | PERMNO | DATE | PRC | RET |
|---|---|---|---|---|
| 1 | 10000 | 19851231 | . | . |
| 2 | 10000 | 19860131 | -4.37500 | C |
| 3 | 10000 | 19860228 | -3.25000 | -0.257143 |
| 4 | 10000 | 19860331 | -4.43750 | 0.365385 |
| 5 | 10000 | 19860430 | -4.00000 | -0.098592 |
| 6 | 10000 | 19860530 | -3.10938 | -0.222656 |
| 7 | 10000 | 19860630 | -3.09375 | -0.005025 |
| 8 | 10000 | 19860731 | -2.84375 | -0.080808 |
| 9 | 10000 | 19860829 | -1.09375 | -0.615385 |
| 10 | 10000 | 19860930 | -1.03125 | -0.057143 |
| 11 | 10000 | 19861031 | -0.78125 | -0.242424 |
| 12 | 10000 | 19861128 | -0.82813 | 0.060000 |
| 13 | 10000 | 19861231 | -0.51563 | -0.377358 |
| 14 | 10000 | 19870130 | -0.40625 | -0.212121 |
| 15 | 10000 | 19870227 | -0.40625 | 0.000000 |
| 16 | 10000 | 19870331 | -0.25000 | -0.384615 |
| 17 | 10000 | 19870430 | -0.23438 | -0.062500 |
| 18 | 10000 | 19870529 | -0.21875 | -0.066667 |
| 19 | 10000 | 19870630 | . | . |
| 20 | 10001 | 19851231 | . | . |
| 21 | 10001 | 19860131 | -6.12500 | C |
| 22 | 10001 | 19860228 | -6.25000 | 0.020408 |
| 23 | 10001 | 19860331 | -6.31250 | 0.025200 |
| 24 | 10001 | 19860430 | -6.37500 | 0.009901 |
| 25 | 10001 | 19860530 | -6.31250 | -0.009804 |
| 26 | 10001 | 19860630 | -6.12500 | -0.013069 |
| 27 | 10001 | 19860731 | -6.06250 | -0.010204 |
| 28 | 10001 | 19860829 | -6.50000 | 0.072165 |
| 29 | 10001 | 19860930 | 6.37500 | -0.003077 |
| 30 | 10001 | 19861031 | 6.62500 | 0.039216 |

# Data step; set; where.

```
data z;
    set crsp.msf;
    where year(date)=2008 and month(date)=1;
    keep permno date prc ret;
run;

proc print data=z (obs=50);
run;
```

| Obs | CUSIP | PERMNO | PERMCO | ISSUNO | HEXCD | HSICCD | DATE | BIDLO |
|---|---|---|---|---|---|---|---|---|
| 1 | 29274A20 | 10001 | 7953 | 10398 | 3 | 4920 | 20080131 | 13.23000 |
| 2 | 05978R10 | 10002 | 7954 | 10399 | 3 | 6020 | 20080131 | 10.98000 |
| 3 | 00103110 | 10025 | 7975 | 10432 | 3 | 3070 | 20080131 | 28.72000 |
| 4 | 46603210 | 10026 | 7976 | 10433 | 3 | 2052 | 20080131 | 24.56000 |
| 5 | 23323G10 | 10028 | 7978 | 22226 | 2 | 5944 | 20080131 | 4.06000 |
| 6 | 72913210 | 10032 | 7980 | 10437 | 3 | 3670 | 20080131 | 18.54000 |
| 7 | 13886930 | 10042 | 7990 | 10449 | 2 | 1041 | 20080131 | 0.40500 |
| 8 | 77467840 | 10044 | 7992 | 10454 | 3 | 2060 | 20080131 | 11.23000 |
| 9 | 41043F20 | 10051 | 7999 | 10468 | 1 | 3842 | 20080131 | 9.27000 |
| 10 | 00621210 | 10065 | 20023 | 0 | 1 | 6723 | 20080131 | 12.31990 |
| 11 | 86681020 | 10078 | 8021 | 10497 | 3 | 3571 | 20080131 | 14.82000 |
| 12 | 35086510 | 10092 | 8035 | 10522 | 1 | 6794 | 20080131 | 10.95000 |
| 13 | 02514410 | 10100 | 8042 | 10532 | 3 | 8300 | 20080131 | 0.83000 |
| 14 | 68389X10 | 10104 | 8045 | 10536 | 3 | 7370 | 20080131 | 20.07000 |
| 15 | 59491810 | 10107 | 8048 | 10539 | 3 | 7370 | 20080131 | 31.93000 |
| 16 | 00509L70 | 10116 | 8028 | 70081028 | 3 | 2834 | 20080131 | . |
| 17 | 01736110 | 10137 | 20045 | 0 | 1 | 4911 | 20080131 | 53.55000 |
| 18 | 74144T10 | 10138 | 8087 | 10609 | 3 | 6211 | 20080131 | 47.52000 |
| 19 | 67104010 | 10143 | 8092 | 10614 | 3 | 2830 | 20080131 | 38.58000 |
| 20 | 43851610 | 10145 | 22168 | 0 | 1 | 3724 | 20080131 | 53.94990 |
| 21 | 26864810 | 10147 | 8093 | 10615 | 1 | 3572 | 20080131 | 15.81000 |
| 22 | 98385710 | 10163 | 8106 | 10635 | 3 | 3860 | 20080131 | 9.74000 |
| 23 | 00972810 | 10180 | 9743 | 12980 | 3 | 2834 | 20080131 | 7.30000 |
| 24 | 87823710 | 10182 | 8125 | 10660 | 3 | 5040 | 20080131 | 32.66000 |
| 25 | 75991610 | 10200 | 8135 | 10673 | 3 | 2830 | 20080131 | 5.86000 |
| 26 | 50215P10 | 10205 | 8140 | 10678 | 3 | 6035 | 20080131 | 15.75000 |
| 27 | 78080N10 | 10207 | 9806 | 13072 | 3 | 6720 | 20080131 | 7.54000 |
| 28 | 34963110 | 10225 | 20067 | 0 | 1 | 3433 | 20080131 | 65.67000 |
| 29 | 28966010 | 10232 | 8158 | 10697 | 3 | 6030 | 20080131 | 18.57000 |
| 30 | 39091410 | 10238 | 8164 | 10704 | 1 | 6035 | 20080131 | 11.50000 |
| 31 | 05775520 | 10239 | 714 | 924 | 3 | 6330 | 20080131 | 25.65000 |
| 32 | 45383610 | 10252 | 8179 | 10733 | 3 | 6020 | 20080131 | 24.67000 |

# proc means: Equally weighted portfolio

```
proc means data=crsp.msf nway noprint;
    by date;
    where year(date)>=2000;
    var ret;
    output out=eqret mean=;
run;

proc print data=eqret (obs=50);
run;
```

- Note: year() is a SAS **function**

# Proc means: output file…

The SAS System 14:51 Wed

| Obs | DATE | _TYPE_ | _FREQ_ | RET |
|---|---|---|---|---|
| 1 | 20000131 | 0 | 8606 | 0.050586 |
| 2 | 20000229 | 0 | 8624 | 0.120475 |
| 3 | 20000331 | 0 | 8617 | -0.000429 |
| 4 | 20000428 | 0 | 8580 | -0.088543 |
| 5 | 20000531 | 0 | 8610 | -0.057305 |
| 6 | 20000630 | 0 | 8627 | 0.079153 |
| 7 | 20000731 | 0 | 8622 | -0.019355 |
| 8 | 20000831 | 0 | 8603 | 0.059271 |
| 9 | 20000929 | 0 | 8590 | -0.040113 |
| 10 | 20001031 | 0 | 8542 | -0.066834 |
| 11 | 20001130 | 0 | 8469 | -0.118659 |
| 12 | 20001229 | 0 | 8380 | -0.007087 |
| 13 | 20010131 | 0 | 8293 | 0.225205 |
| 14 | 20010228 | 0 | 8232 | -0.074514 |
| 15 | 20010330 | 0 | 8178 | -0.072848 |
| 16 | 20010430 | 0 | 8118 | 0.076360 |
| 17 | 20010531 | 0 | 8048 | 0.061981 |
| 18 | 20010629 | 0 | 7985 | 0.007678 |
| 19 | 20010731 | 0 | 7916 | -0.028517 |
| 20 | 20010831 | 0 | 7858 | -0.034321 |
| 21 | 20010928 | 0 | 7784 | -0.127933 |
| 22 | 20011031 | 0 | 7762 | 0.077762 |
| 23 | 20011130 | 0 | 7730 | 0.080751 |
| 24 | 20011231 | 0 | 7656 | 0.057688 |

# Proc means: class statement

```
28          proc means data=crsp.msf nway noprint;
29              by date;
NOTE: An index was selected to execute the BY statement.
      The observations will be returned in index order rather than in physical
      order.  The selected index is for the variable(s):
 DATE
30              where year(date)>=2000;
31              var ret;
32              output out=eqret mean=;
33          run;

NOTE: There were 789584 observations read from the data set CRSP.MSF.
      WHERE YEAR(date)>=2000;
NOTE: The data set WORK.EQRET has 108 observations and 4 variables.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           6:05.00
      cpu time            6:05.16
```

- This is inefficient code:
  - The data is sorted by permno and then date
    - "by date" ONLY worked because there is an index
    - But processing is slow!
  - Also: a function `year()` is calculated for all date values

# Average returns for each security: Proc means: class; date constants

```
/* ===  Cant use "by" with nonsorted data!    === */
/* ...  Note an alternative form of 'where'   ... */

proc means data=crsp.msf (where=(date>='01JAN2000'd)) nway noprint;
    class permno;
     var ret;
     output out=avgret mean=;
run;

proc print data=avgret (obs=50);
run;
```

- “class date” does not require sorting
- works faster than “by” with an index
- Note the a SAS date constant!

```
41          proc means data=crsp.msf (where=(date>='01JAN2000'd)) nway noprint;
42              class permno;
43              var ret;
44              output out=avgret (drop=_type_ _freq_) mean=;
45          run;

NOTE: There were 789584 observations read from the data set CRSP.MSF.
      WHERE date>='01JAN2000'D;
NOTE: The data set WORK.AVGRET has 12700 observations and 2 variables.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           8.00 seconds
      cpu time            9.50 seconds
```

| Obs | PERMNO | RET |
|---|---|---|
| 1 | 10001 | 0.011799 |
| 2 | 10002 | 0.010482 |
| 3 | 10009 | 0.125462 |
| 4 | 10012 | -0.020922 |
| 5 | 10016 | 0.018921 |
| 6 | 10019 | -0.040217 |
| 7 | 10025 | 0.009598 |
| 8 | 10026 | 0.016486 |
| 9 | 10028 | 0.006038 |
| 10 | 10032 | 0.012382 |
| 11 | 10035 | 0.012812 |
| 12 | 10037 | 0.017887 |
| 13 | 10039 | -0.104072 |
| 14 | 10042 | 0.010017 |
| 15 | 10044 | 0.021770 |
| 16 | 10051 | 0.022800 |
| 17 | 10056 | 0.010606 |

# ComputingCumulative Returns: Data step: by; if; retain; first.var; last.var

```
data cumret;
   set crsp.msf (obs=1000);
   retain cumret;
   by permno date;
   if first.permno then cumret = 1;
   else if not missing(ret) then cumret = cumret*(1+ret);
   keep permno date ret cumret;
run;

proc print data=cumret (obs=100);
run;
```

- Grouped data (indicated by “by permno date”):
  - First.permno is equal to 1 if the observation is first in a group of observations with identical permno’s; 0 otherwise
  - Last.permno is similarly 1 if the observation is last in a group
  - Data set needs to be indexed or (preferably) sorted
    - If grouped but *not* sorted one *sometimes* can use “`by permno date notsorted`”

| Obs | PERMNO | DATE | RET | cumret |
|---|---|---|---|---|
| 1 | 10000 | 19851231 | . | 1.00000 |
| 2 | 10000 | 19860131 | C | 1.00000 |
| 3 | 10000 | 19860228 | -0.257143 | 0.74286 |
| 4 | 10000 | 19860331 | 0.365385 | 1.01429 |
| 5 | 10000 | 19860430 | -0.098592 | 0.91429 |
| 6 | 10000 | 19860530 | -0.222656 | 0.71071 |
| 7 | 10000 | 19860630 | -0.005025 | 0.70714 |
| 8 | 10000 | 19860731 | -0.080808 | 0.65000 |
| 9 | 10000 | 19860829 | -0.615385 | 0.25000 |
| 10 | 10000 | 19860930 | -0.057143 | 0.23571 |
| 11 | 10000 | 19861031 | -0.242424 | 0.17857 |
| 12 | 10000 | 19861128 | 0.060000 | 0.18929 |
| 13 | 10000 | 19861231 | -0.377358 | 0.11786 |
| 14 | 10000 | 19870130 | -0.212121 | 0.09286 |
| 15 | 10000 | 19870227 | 0.000000 | 0.09286 |
| 16 | 10000 | 19870331 | -0.384615 | 0.05714 |
| 17 | 10000 | 19870430 | -0.062500 | 0.05357 |
| 18 | 10000 | 19870529 | -0.066667 | 0.05000 |
| 19 | 10000 | 19870630 | . | 0.05000 |
| 20 | 10001 | 19851231 | . | 1.00000 |
| 21 | 10001 | 19860131 | C | 1.00000 |
| 22 | 10001 | 19860228 | 0.020408 | 1.02041 |
| 23 | 10001 | 19860331 | 0.025200 | 1.04612 |
| 24 | 10001 | 19860430 | 0.009901 | 1.05648 |
| 25 | 10001 | 19860530 | -0.009804 | 1.04612 |
| 26 | 10001 | 19860630 | -0.013069 | 1.03245 |
| 27 | 10001 | 19860731 | -0.010204 | 1.02192 |

**Note the missing return code!**

- "`if ret~=.`" would **not** have worked!
- "`missing(var)`" works for all missing values – i.e.
  - Numeric : `.  .A  .B  .C  .D  …  .Z`
  - Character: `""`

# Counting positive returns: proc freq; data views; more missing…

```
data gt0 (keep=gg) /view=gt0;
 set crsp.msf (where=(date>='01JAN2000'd));
 if not missing(ret) then gg=ret>0;
run;

proc freq data=gt0;
  tables gg / missing;
run;
```

The FREQ Procedure

| gg | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|---|---|---|---|---|
| . | 25096 | 3.18 | 25096 | 3.18 |
| 0 | 378061 | 47.88 | 403157 | 51.06 |
| 1 | 386427 | 48.94 | 789584 | 100.00 |

- A data view is a “fake” dataset that is stored as compiled code
  - Takes (almost) no space; easy to use.
  - Faster processing – at least if you only need it once!
- “`if not missing(ret)`” is required since missing values (including “.”) compare as if they are just large negatives!

# Other useful procedures

- **`proc sort`** : sorting a dataset
- **`proc append`** : append a dataset at the end of another
- **`proc import`** : import data from other formats (e.g. csv)
  - Must run SAS with "`-nodisplay`" command line option
  - Be careful when importing datasets with many variables!
    - Make sure to set `guessingrows=` option
- **`proc export`** : export data into other formats
  - Actual export data step code is written into the log!
    - Can be copy-pasted and re-used!
    - This allows specifying dataset options; etc.
- **`proc reg`**: a linear regression
- **`proc glm`**: fixed/random effects etc.
- **`proc model`**: can use for many things – e.g. GMM

# Reading raw files; proc reg

```
filename file1 '~ple531/training/table97.raw';
filename file2 '~ple531/training/pop6080.txt';

data consumption; infile file1 missover;
   input year gnp c; lagcon=lag(c);

data population; infile file2; input year pop;

data matched; merge consumption population;
  by year;

proc print data=matched;
proc reg data=d1;
  Test_run: model c = gnp lagcon / dw;
  output out = d2 p=chat r=ehat student=normehat;
```

# A recap:

- Errors due to forgotten semi-colons and quotes are frequent and often hard to recognize.
- Missing values are negative infinity in SAS (positive infinity in Stata)
- PROCs can generate data files that you can manipulate in DATA steps.
- SAS can keep as many files as you need “open” during a session (unlike Stata).
- Merge statement requires datasets to be sorted. Sorting takes time and space, but merge can sometimes be more efficient than SQL.
- Know your data; always check the log.
- Always test your code with small datasets that represent all the cases in your actual sample.
- Document your program and add variable labels to your data!

# “Permanent” vs temporary SAS data files

- Remember?

```
data consumption; infile file1 missover;
input year gnp c; lagcon=lag(c);
```

- “consumption” is a temporary dataset. It is deleted at the end of the SAS session.
- For permanent datasets: **LIBNAME *libref* ‘path’;**
  A library is a directory with SAS datasets.
- Example:

```
libname finc488 '~ple531/training/mydata';
data finc488.consumption; infile file1
missover; input year gnp c; lagcon=lag(c);
```

# Merging data – options

- In SAS, we have two options for merging datasets:
  - MERGE statement in a DATA step
  - PROC SQL
- The trick is understanding your data and controlling how the merge takes place.
- A DATA step merge requires sorted datasets. Sorting typically takes more space and time.
- SQL does not require sorting and can also create variables on the fly.

# A basic merge

- As with a DATA step with a BY statement, SAS generates some internal dummies that you can use. In a MERGE, SAS keeps track of the source of the observation. Use the IN= data set option to name the internal tracking variable.

```
data temp;
  merge dsA (in=a) dsB (in=b);
  by id;
  if a;
```

- a=1 when the case was present in dataset “dsA”, and missing otherwise. “`if a;`” is a short form for “`if a ne .;`”

# Basic PROC SQL

```
PROC SQL;
  CREATE TABLE temp AS SELECT
  dsA.v1, dsA.v2, dsB.v3, dsB.v4
  FROM dsA, dsB
  WHERE dsA.id = dsB.id;
QUIT;
```

- This is a “inner join”: it s in dsA that match cases in dsB.

Name of output dataset

Variables in format: source.var_name

Source datasets

Conditions to merge

# crsp.msf and crsp.stocknames

**Contents of Dataset: crsp:msf**

Descriptions: Monthly Stock - Securities

| Position | Variable | Format | Length | Description |
|---|---|---|---|---|
| 20 | ALTPRCDT | Num | 8 | Alternate Price Date |
| 18 | ALTPRC | Num | 8 | Price Alternate |
| 9 | ASKHI | Num | 8 | Ask or High Price |
| 14 | ASK | Num | 8 | Ask |
| 8 | BIDLO | Num | 8 | Bid or Low Price |
| 13 | BID | Num | 8 | Bid |
| 16 | CFACPR | Num | 8 | Cumulative Factor to Adjust Prices |
| 17 | CFACSHR | Num | 8 | Cumulative Factor to Adjust Shares/Vol |
| 1 | CUSIP | Char | 8 | CUSIP Header |
| 7 | DATE | Num | 4 | Date of Observation |
| 5 | HEXCD | Num | 8 | Exchange Code Header |
| 6 | HSICCD | Num | 8 | Standard Industrial Classification Code |
| 4 | ISSUNO | Num | 8 | Nasdaq Issue Number |
| 3 | PERMCO | Num | 8 | PERMCO |
| 2 | PERMNO | Num | 8 | PERMNO |
| 10 | PRC | Num | 8 | Price or Bid/Ask Average |
| 21 | RETX | Num | 8 | Returns without Dividends |
| 12 | RET | Num | 8 | Returns |
| 15 | SHROUT | Num | 8 | Shares Outstanding |
| 19 | SPREAD | Num | 8 | Spread Between Bid and Ask |
| 11 | VOL | Num | 8 | Volume |

**Contents of Dataset: crsp:stocknames**

Descriptions: Company Names with Effective Name Dates

| Position | Variable | Format | Length | Description |
|---|---|---|---|---|
| 5 | CUSIP | Char | 8 | CUSIP Identifier - Header |
| 15 | END_DATE | Num | 8 | End Date of Price Data |
| 10 | EXCHCD | Num | 8 | Exchange Code - Historical |
| 9 | HEXCD | Num | 8 | Exchange Code - Header |
| 3 | NAMEDT | Num | 8 | Start Date of Effective Name |
| 16 | NAMEDUM | Num | 8 | 1: Daily Only, 2: Daily and Monthly |
| 4 | NAMEENDDT | Num | 8 | End Date of Effective Name |
| 6 | NCUSIP | Char | 8 | CUSIP Identifier - Historical |
| 2 | PERMCO | Num | 8 | CRSP Permanent Company Number |
| 1 | PERMNO | Num | 8 | CRSP Permanent Number |
| 12 | SHRCD | Num | 8 | Share Code as of Name Start Date |
| 13 | SHRCLS | Char | 4 | Share Class as of Name Start Date |
| 11 | SICCD | Num | 8 | SIC Code as of Name Start Date |
| 14 | ST_DATE | Num | 8 | Start Date of Price Data |
| 8 | comnam | Char | 32 | Company Name - Historical |
| 7 | ticker | Char | 5 | Exchange Ticker Symbol - Historical |

# proc sql: historical returns of IBM

```
proc sql;
    select b.ticker
        ,a.date
        ,a.ret
    from crsp.msf as a
    left join crsp.stocknames as b
        on a.permno = b.permno
        and b.namedt <= a.date <= b.nameenddt
    where b.ticker = 'IBM'
    order by a.permno, a.date;
quit;
```

| Exchange Ticker Symbol - Historical | Date of Observation | Returns |
|---|---|---|
| IBM | 19620731 | 0.140752 |
| IBM | 19620831 | 0.025840 |
| IBM | 19620928 | -0.107886 |
| IBM | 19621031 | -0.021216 |
| IBM | 19621130 | 0.153902 |
| IBM | 19621231 | -0.021330 |
| IBM | 19630131 | 0.086538 |
| IBM | 19630228 | -0.054277 |
| IBM | 19630329 | 0.053158 |
| IBM | 19630430 | 0.101544 |
| IBM | 19630531 | 0.031267 |
| IBM | 19630628 | -0.085909 |
| IBM | 19630731 | -0.004011 |
| IBM | 19630830 | 0.025892 |
| IBM | 19630930 | 0.016301 |
| IBM | 19631031 | 0.092920 |
| IBM | 19631129 | -0.015182 |
| IBM | 19631231 | 0.044822 |
| IBM | 19640131 | 0.069034 |
| IBM | 19640228 | 0.052122 |
| IBM | 19640331 | 0.044376 |
| IBM | 19640430 | -0.040387 |
| IBM | 19640528 | 0.054910 |

- This code assumes that the ticker has not changed over this period.
- Putting commas in front of variable names in lists helps prevent **very** annoying syntax errors!!!

# Self-join: computing past returns…

```
proc sql;
    create table pastret (drop=lcumret) as select distinct
        a.permno
        ,a.date
        ,a.ret
        ,sum(log(1+b.ret)) as lcumret
        ,exp(calculated lcumret)-1 as pastret
    from crsp.msf as a
    left join crsp.msf as b
        on a.permno = b.permno
        and 0 <= intck('month',b.date, a.date) <= 6
        and a.date>='01JAN2000'd
    group by a.permno, a.date
    having count(b.ret) = 6;

proc print data=pastret (obs=1000);
run;
```

- Computing cumulative returns over a rolling 6-mo. window:
  - Intck() here gives the range between 2 dates in months
  - sum() computes a sum over a group
  - Need to use having() rather than where() with summary statistics
  - “calculated” keyword used to refer to newly created variables!

# Merging data: the perfect merge

**dataset A**

| permno | date | ret | shrout |
|---|---|---|---|
| 10107 | 20061222 | -0.011341 | 9830460 |
| 10107 | 20061226 | 0.011808 | 9830460 |
| 10107 | 20061227 | 0.001000 | 9830460 |
| 10107 | 20061228 | -0.001332 | 9830460 |
| 10107 | 20061229 | -0.004003 | 9830460 |
| 11081 | 20061222 | -0.000393 | 2271614 |
| 11081 | 20061226 | 0.001572 | 2271614 |
| 11081 | 20061227 | -0.001177 | 2271614 |
| 11081 | 20061228 | -0.008641 | 2271614 |
| 11081 | 20061229 | -0.005943 | 2271614 |
| 12490 | 20061222 | -0.006881 | 1506352 |
| 12490 | 20061226 | 0.004305 | 1506352 |
| 12490 | 20061227 | 0.016099 | 1506352 |
| 12490 | 20061228 | -0.002366 | 1506352 |
| 12490 | 20061229 | 0.001856 | 1506352 |

**dataset B**

| permno | date | vol |
|---|---|---|
| 10107 | 20061222 | 26661631 |
| 10107 | 20061226 | 20701973 |
| 10107 | 20061227 | 31366625 |
| 10107 | 20061228 | 26765285 |
| 10107 | 20061229 | 33456606 |
| 11081 | 20061222 | 8723071 |
| 11081 | 20061226 | 6773208 |
| 11081 | 20061227 | 10262437 |
| 11081 | 20061228 | 9988658 |
| 11081 | 20061229 | 6792287 |
| 12490 | 20061222 | 451906 |
| 12490 | 20061226 | 338929 |
| 12490 | 20061227 | 402196 |
| 12490 | 20061228 | 442416 |
| 12490 | 20061229 | 225953 |

# Decision time…

| permno | date | ret | shrout |
|---|---|---|---|
| 10107 | 20061222 | -0.011341 | 9830460 |
| 10107 | 20061226 | 0.011808 | 9830460 |
| 10107 | 20061227 | 0.001000 | 9830460 |
| 10107 | 20061228 | -0.001332 | 9830460 |
| 10107 | 20061229 | -0.004003 | 9830460 |
| 11081 | 20061222 | -0.000393 | 2271614 |
| 11081 | 20061226 | 0.001572 | 2271614 |
| 11081 | 20061227 | -0.001177 | 2271614 |
| 11081 | 20061228 | -0.008641 | 2271614 |
| 11081 | 20061229 | -0.005943 | 2271614 |
| 12490 | 20061222 | -0.006881 | 1506352 |
| 12490 | 20061226 | 0.004305 | 1506352 |
| 12490 | 20061227 | 0.016099 | 1506352 |
| 12490 | 20061228 | -0.002366 | 1506352 |
| 12490 | 20061229 | 0.001856 | 1506352 |

| permno | date | vol |
|---|---|---|
| 10107 | 20061221 | 25679912 |
| 10107 | 20061227 | 31366625 |
| 10107 | 20061228 | 26765285 |
| 11081 | 20061227 | 10262437 |
| 11081 | 20061228 | 9988658 |
| 88394 | 20061227 | 45439500 |
| 88394 | 20061228 | 30126200 |

# Only complete records

| merge 1 | | | | |
|---|---|---|---|---|
| permno | date | ret | shrout | vol |
| 10107 | 12/27/2006 | 0.001 | 9830460 | 31366625 |
| 10107 | 12/28/2006 | -0.001332 | 9830460 | 26765285 |
| 11081 | 12/27/2006 | -0.001177 | 2271614 | 10262437 |
| 11081 | 12/28/2006 | -0.008641 | 2271614 | 9988658 |

```
proc sql;
    create table dsmerge2 as select
    dsa.*, dsb.vol
    from dsa, dsb
    where dsa.permno=dsb.permno and
    dsa.date=dsb.date;
quit;
```

# The left rules

**merge 2**

| permno | date | ret | shrout | vol |
|---|---|---|---|---|
| 10107 | 12/22/2006 | -0.011341 | 9830460 | . |
| 10107 | 12/26/2006 | 0.011808 | 9830460 | . |
| 10107 | 12/27/2006 | 0.001 | 9830460 | 31366625 |
| 10107 | 12/28/2006 | -0.001332 | 9830460 | 26765285 |
| 10107 | 12/29/2006 | -0.004003 | 9830460 | . |
| 11081 | 12/22/2006 | -0.000393 | 2271614 | . |
| 11081 | 12/26/2006 | 0.001572 | 2271614 | . |
| 11081 | 12/27/2006 | -0.001177 | 2271614 | 10262437 |
| 11081 | 12/28/2006 | -0.008641 | 2271614 | 9988658 |
| 11081 | 12/29/2006 | -0.005943 | 2271614 | . |
| 12490 | 12/22/2006 | -0.006881 | 1506352 | . |
| 12490 | 12/26/2006 | 0.004305 | 1506352 | . |
| 12490 | 12/27/2006 | 0.016099 | 1506352 | . |
| 12490 | 12/28/2006 | -0.002366 | 1506352 | . |
| 12490 | 12/29/2006 | 0.001856 | 1506352 | . |

```
proc sql;
   create table dsmerge3 as select
   dsa.*, dsb.vol
   from dsa left join dsb
   on dsa.permno=dsb.permno and
   dsa.date=dsb.date;
quit;
```

# The right rules

| merge 3 | | | | |
|---|---|---|---|---|
| permno | date | vol | ret | shrout |
| 10107 | 12/21/2006 | 25678912 | . | . |
| 10107 | 12/27/2006 | 31366625 | 0.001 | 9830460 |
| 10107 | 12/28/2006 | 26765285 | -0.001332 | 9830460 |
| 11081 | 12/27/2006 | 10262437 | -0.001177 | 2271614 |
| 11081 | 12/28/2006 | 9988658 | -0.008641 | 2271614 |
| 88394 | 12/27/2006 | 45439500 | . | . |
| 88394 | 12/28/2006 | 30126200 | . | . |

```
proc sql;
create table dsmerge4 as select
dsb.*, dsa.ret, dsa.shrout
from dsa right join dsb
on dsa.permno=dsb.permno and dsa.date=dsb.date;
quit;
```

# All the records

**merge 4**

| permno | date | ret | shrout | vol |
|---|---|---|---|---|
| 10107 | 12/21/2006 | . | . | 25678912 |
| 10107 | 12/22/2006 | -0.011341 | 9830460 | . |
| 10107 | 12/26/2006 | 0.011808 | 9830460 | . |
| 10107 | 12/27/2006 | 0.001 | 9830460 | 31366625 |
| 10107 | 12/28/2006 | -0.001332 | 9830460 | 26765285 |
| 10107 | 12/29/2006 | -0.004003 | 9830460 | . |
| 11081 | 12/22/2006 | -0.000393 | 2271614 | . |
| 11081 | 12/26/2006 | 0.001572 | 2271614 | . |
| 11081 | 12/27/2006 | -0.001177 | 2271614 | 10262437 |
| 11081 | 12/28/2006 | -0.008641 | 2271614 | 9988658 |
| 11081 | 12/29/2006 | -0.005943 | 2271614 | . |
| 12490 | 12/22/2006 | -0.006881 | 1506352 | . |
| 12490 | 12/26/2006 | 0.004305 | 1506352 | . |
| 12490 | 12/27/2006 | 0.016099 | 1506352 | . |
| 12490 | 12/28/2006 | -0.002366 | 1506352 | . |
| 12490 | 12/29/2006 | 0.001856 | 1506352 | . |
| 88394 | 12/27/2006 | . | . | 45439500 |
| 88394 | 12/28/2006 | . | . | 30126200 |

```
proc sql;
 create table dsmerge1 as select
  coalesce(dsa.permno,
   dsb.permno) as permno,
  coalesce(dsa.date, dsb.date)
   as date format=mmddyys10.,
   dsa.ret, dsa.shrout, dsb.vol
 from dsa full join dsb
 on dsa.permno=dsb.permno and
   dsa.date=dsb.date;
quit;
```

# Merging: DATA step versus PROC SQL

| *Type of merge* | *Conditions in DATA step* | *PROC SQL jargon and conditions* |
|---|---|---|
| All records | No conditions necessary | "outer join"<br>`from dsA full join dsB` |
| Complete records | if a and b; | "inner join"<br>`from dsA, dsB` |
| Left rules | if a; | "left outer join"<br>`from dsA left join dsB` |
| Right rules | if b; | "right outer join"<br>`from dsA right join dsB` |

# References

Good references are

- http://ftp.sas.com/techsup/download/technote/ts644.html
- A book called "Combining and modifying SAS data sets: examples", which is in the RC library (for Kellogg students). It has a lot of examples. Unfortunately, it does not exist in an online version (only the code is available).
- And “Getting Started with the SQL Procedure”