Reading Data & Creating Datasets

Business Intelligence Systems and Data Mining
IMAT5168 Analytics Programming

SAS data sets

- A data set is a 2 dimensional table of rows and columns.
- Each row represents data about one subject.
- Each column is a ‘variable’ that stores a measurable feature of the subject.
- There may be one or more rows per subject.
- One or more columns may be repeated to record the same measurement many times
  - pulse_rate1, pulse_rate2, pulse_rate3
    - represents pulse rates taken at 5, 15 & 30 minutes after arriving at a GP
- For database aficionados only…
  - The data are **not** normalised.

SAS Data Set Example

	Principal income	Secondary income	Size of family	Own or rent house?	Amount of mortgage or rent payment
1	48370.5	20345.5	4	1	669
2	43621	22727	5	1	654
3	49538	0	6	1	606
4	50816.5	0	3	1	533.5
5	46490	22698	3	1	613
6	52071.5	22151	3	1	700.5
7	46041	22857.5	4	0	650
8	51725	0	5	1	617.5
9	50767	21369.5	2	1	563
10	50035	26064	5	0	847
11	45801.5	0	4	0	587.5
12	52546	17568	5	1	756
13	44928.5	0	2	0	281
14	48297	21715.5	2	0	723
15	43468.5	20209	2	1	330
16	47533	0	3	1	582.5
17	53316	0	3	1	554

Anatomy of a SAS variable

Length — Size in characters

Data type — Numeric or text

Informat — The informat determines how data being input will be interpreted.

Outformat — The outformat determines how the variable will be formatted when printed in output.

Name (to use in code) — This is the name programs will use to refer the variable.

Label — This is the name for the variable's column/axis to be used in tables or graphs.

Taxonomy of variables

- Representation of missing values in SAS
  - Text: ''
    - two single quotes
  - Numbers: .
    - dot/period
- Dates are treated as numeric data
  - Each date is the number of days after 1/1/1960

	Text ($)	Numeric
Nominal	Categories coded as text *Convert to a number?*	Categories coded as numbers *(+1 has no meaning)*
Ordinal	Ordered categories *Convert to a number?*	Ordered categories coded as numbers (World ranking) *(+1 has no meaning)*
Interval	*Not applicable*	Numbers where only add & subtract valid (Date) *+1 has a meaning, 0 has no meaning*
Ratio	*Not applicable*	Numbers where add, subtract, multiply & divide valid *+1 has a meaning, 0 has meaning*

Data step processing overview

- Compile phase
  - Declare space to store the line of data from the input file
    - Input buffer
  - Declare space for the required SAS variables
    - Program data vector
  - Initialise useful information about the process

- Execution phase
  - Write the current variables to the data set
  - Reset all variables to missing
  - Build the next observation
    - read a line of data
    - use input statements to translate the text into a SAS variable
    - use data step programming statements to alter/add to SAS variables

Anatomy of the data statement

- **Each data step will contain *data* and *source*; the remaining elements will depend on the task.**

- **data** <output data set>;
  - ***source***
    - set <input another SAS data set>;
    - infile <input a raw data file>;
  - ***length***
    - length <lengths of variables>;
  - ***format***
    - informat <informat definitions for variables>;
  - ***variables***
    - input <variables>;
  - ***programming statements***
    - if <condition> then <action>;
    - output <output the current set of variables>
  - ***output formatting***
    - format <outformat definitions for variables>;
    - label <labels>;
  - ***data***
    - datalines <data>;
- **run;**

Strategy for data statements

- Look at the source data
  - What format is it in?
    - Which input statements will help?
  - Identify any rows that are obviously unusual
    - Put them and one "standard" row in a test file or a `datalines` statement

- Write the **`data`** statement to read the test data
  - **ALWAYS** look at the log
    - Check that the right number of records have been read
    - If there is an error, refine the **`data`** statement
  - Always view the data set itself
    - If the data are incorrect, refine the **`data`** statement

- When the **`data`** statement is correct
  - Read the full dataset
  - Check the log for errors
  - Use univariate stats (continuous) or frequencies (categorical) to check data

About data in text files

- Terminology
  - ‘Delimiter’ is used for two purposes
  - It is used to mean:

- Separator: (which SAS calls a delimiter)
  - the character(s) that separate data items

```
23 24 25 …  /* a space character separates each number */
```

- Delimiter: (for which SAS has no specific name)
  - the character(s) that surround the character(s) representing a data item

```
'Hello, World!' /* the quote (") delimits text */
```

Types of input

Column input
General form: V[V]

```
123456789+123456789+
1 Smith    196412081
2 Williams197311075
```

V = value

V_1= name 1-8,
V_2 = Year 9-12
V_3= SBP 13-15,
V_4 = DBP 16-17

List input
General form: DVDS[DVDS]

D = delimiter ("), V = value, S = separator (,)

```
123456789+123456789+1234
1 "Smith", 1964, 120, 80
2 "Jones", 1973, 110, 75
```

Free formatted input
General form: [hvs]V[[hvs]V]

V = value [hvs] = horizontal or vertical space

```
123456789+123456789+
1 Smith
2 1964       120  80
3 Jones
4 1973       110  75
```

Data arranged in columns

sleep.txt

1	African elephant	6654.000	5712.000	-999.0	-999.0	3.3	38.6	645.0	3	5	3
2	African giant pouched rat	1.000	6.600	6.3	2.0	8.3	4.5	42.0	3	1	3
3	Arctic Fox	3.385	44.500	-999.0	-999.0	12.5	14.0	60.0	1	1	1
4	Arctic ground squirrel	.920	5.700	-999.0	-999.0	16.5	-999.0	25.0	5	2	3
5	Asian elephant	2547.000	4603.000	2.1	1.8	3.9	69.0	624.0	3	5	4
6	Baboon	10.550	179.500	9.1	.7	9.8	27.0	180.0	4	4	4
7	Big brown bat	.023	.300	15.8	3.9	19.7	19.0	35.0	1	1	1
8	Brazilian tapir	160.000	169.000	5.2	1.0	6.2	30.4	392.0	4	5	4
9	Cat	3.300	25.600	10.9	3.6	14.5	28.0	63.0	1	2	1
10	Chimpanzee	52.160	440.000	8.3	1.4	9.7	50.0	230.0	1	1	1
11	Chinchilla	.425	6.400	11.0	1.5	12.5	7.0	112.0	5	4	4

credit.txt

1	1	1	42	2	2	7882	1	4	2	3	3	4	2	45	3	3	1	3	2	1
2	2	1	24	3	0	4870	1	3	3	3	1	4	4	53	3	3	2	3	2	1
3	3	4	36	2	6	9055	5	3	2	3	1	4	4	35	3	3	1	2	2	2
4	4	4	24	2	2	2835	3	5	3	3	1	4	2	53	3	2	1	3	1	1
5	5	2	36	2	1	6948	1	3	2	3	1	2	3	35	3	1	1	4	1	2

Data step for columnar data


```
data customer;
        infile
                'credit.txt'
                missover
        ;
        input
                customer 1-3        /* ID Number */
                account  5         /* Chequing account */
                duration 7-8        /* Duration in months */
                history  10        /* Credit history */

        /* more variable definitions here */
        ;
run;
```

Data arranged in lists

smsa.txt

```
city    JanTemp JulyTemp    RelHum  Rain    Mortality   Education   PopDensity  %NonWhite   %WC pop pop/house   income  HCPot   NOxPot  SO2Pot  NOx
Akron, OH   27  71  59  36  921.87  11.4    3243    8.8 42.6    660328  3.34    29560   21  15  59  15
Albany-Schenectady-Troy, NY 23  72  57  35  997.87  11.0    4281    3.5 50.7    835880  3.14    31458   8   10  39  10
Allentown, Bethlehem, PA-NJ 29  74  54  44  962.35  9.8 4260    0.8 39.4    635481  3.21    31856   6   6   33  6
Atlanta, GA 45  79  56  47  982.29  11.1    3125    27.1    50.2    2138231 3.41    32452   18  8   24  8
Baltimore, MD   35  77  55  43  1071.29 9.6 6441    24.4    43.7    2199531 3.44    32368   43  38  206 38
Birmingham, AL  45  80  54  53  1030.38 10.2    3325    38.5    43.1    883946  3.45    27835   30  32  72  32
```

country.csv

```
Name,Area,Popnsize,Pcturban,Lang,Liter,Lifemen,Lifewom,PcGDP
Afghanistan,647500,..,..,Pashto,..,..,..,800
Albania,28748,3.1,43.8,Albanian,98.75,71,76.7,4900
Algeria,2381740,31.9,58.8,Arabic,69.8,69.8,72.4,7200
American Samoa,199,..,..,..,..,..,..,8000
Andorra,468,..,..,Catalan,..,..,..,24000
Angola,1246700,15,35.7,Portuguese,67.95,39.3,42.3,3200
Anguilla,102,..,..,..,..,..,..,7500
Antigua and Barbuda,443,0.1,37.8,English,..,..,..,11000
Argentina,2766890,38,90.1,Spanish,97.2,70.7,78.2,13700
```

List data example


```
data work.NewSalesEmps;
        infile
                'newemps.txt'
        ;
        input
                First_Name      $       /* text */
                Last_Name       $
                Job_Title       & $     /* text with spaces */
                Salary                  /* numeric */
        ;
run;
```

List data problems

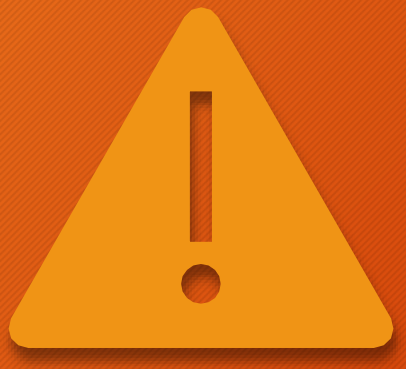
- Comma delimited
  - Commas in text variables

```
e.g. 323, "George Bush, Jnr", 44.50
```

  - Handle by adding `DSD` to `INFILE` command
  - NB: add `DLM='09'x` to use tab as a delimiter

- Space delimited
  - Spaces in text variables (George Bush, Jnr)
  - e.g. `323 George Bush, Jnr 44.50`
  - Handle by using '`&`' before '`$`' so that 2 spaces end input
    - Note: You may need to reformat the data to ensure 2 spaces are between data items.

Input from Excel in 32-bit Office



- Problem: 64-bit SAS 9.4
  - cannot rad Excel or Access files directly if you are using 32-bit Microsoft Office
- Solution:
  - Install SAS PC Files Server
    - See http://support.sas.com/kb/43/802.html
  - Import the data using: File -> Import data…
    - Try: "Microsoft Excel Workbook on PC Files Server"
    - Or: XLS or XLSX format
  - Save the program that SAS creates for later re-use.
- Note:
  - Importing Excel files is not easy and requires a bit of patience and experimentation!

Data step for free format data


```
1.      data club;
2.           length
3.                          name $       20
4.           ;
5.           input
6.           /*             line         pos          variable                format */
7.                          #2           @1           team $                  6.
8.                          #1           @6           name & $
9.                                       @1           id
10.                         #3           @1           (wt0-wt1)               (3.)
11.          ;
12.          datalines;
13.     /* data: 3 lines per subject */
14.     1023 David Shaw
15.     red
16.     189 165
17.     1049 Amelia Serrano
18.     yellow
19.     145 124
20.          ;
21.     run;
```

	name	team	id	wt0	wt1
1	David Shaw	red	1023	189	16
2	Amelia Serrano	yellow	1049	145	12

Data step for free format data

- Handle using cursor movement commands
  - `@n` set starting position
  - `+n` move n characters forward
  - `/` new line
  - `#n` read from line n
  - `n1-n2` read from position n1 to position n2

Input buffer underflow

- Underflow: less data than used by all variables
    - flowover (default)
        - read next record & continue inputting variables
    - missover
        - set current variable to missing
        - start a new data step iteration with a new record
    - truncover
        - set current variable using available characters
        - start a new data step with a new record
    - stopover
        - stop data step processing

Strategies for: hierarchical input Multiple data subjects in input buffer

- Hierarchical input strategy
  - Read variable using the input statement
  - @ = hold the data in the input buffer
    - use variables to decide how to proceed
    - then continue with the next **input** statement
- Multiple data subjects in input buffer strategy
  - @@ = hold the data in the input buffer
    - then continue processing at the start of the **data** statement
- See examples on next slides

Hierarchical input: Select subsets of the data, using @


```
1.   data red_team;
2.      input Team $ 13-18 @; /* hold the data */
3.      if Team='red';    /* Sub-setting if statement */
4.                        /* only continue if condition true */
5.      input             /* code only reached if Team='red' */
6.              IdNumber          1-4
7.              StartWeight       20-22
8.              EndWeight         24-26;
9.      datalines;
10.  /*--+----+----+----+----+-*/
11.  1023 David  red    189 165
12.  1049 Amelia yellow 145 124
13.  *...;
```

Code to handle multiple data rows on one input line


```
data amess1978;
        input
                ventilation
                folate
                @@ /* more data on the same line */
        ;
        datalines;
1 243   2 206    3 241
1 251   2 210    3 258
1 275   2 226    3 270
1 291   2 249    3 293
1 347   2 255    3 328
1 354   2 273
1 380   2 285
1 392   2 295
2 309
;
run;
```

Example informat


```
1. proc format;
2.   invalue team
3.         'red'      = 0
4.         'yellow'   = 1
5.         'green'    = 2
6.         'blue'     = 3
7.         other      = .
8.   ;
9. run;
```

Using put to check...

Specific solution

```
1.   input
2.   . . .
3.       team      team6.
4.   . . .
5.   ;
6.   put 'team = ' team;

Log entries...
team = 0
team = 3
team = 2
team = 1
```

General solution

```
1.    input
2.         . . .
3.         tmp $       7-12
4.        . . . ;
5.    put 'input = ' tmp;
6.    team = input(tmp, team.);
7.    put 'team = ' team;

Log entries...
input = red
team = 0
input = blue
team = 3
input = green
team = 2
input = yellow
team = 1
```

Type conversion using informat

Specific solution

```
1.    * only works with raw data;
2.   input
3.         name $      1-6
4.         team        team6.
5.         sbp         13-15
6.         dbp         16-18
7.         pulse       19-21
8.         wt0         22-24
9.         wt1         25-27
10.  ;
11.  datalines;
12.  Mike   red    128 80 95110 95
13.  John   blue   135 75 72 95 94
14.  Bill   green 12080 72 75 75
15.  Jack   yellow110 75 78 90 85
```

General solution

```
1.   * works with all $ data;
2.   input
3.         name $      1-6
4.         tmp $       7-12
5.         sbp         13-15
6.         dbp         16-18
7.         pulse       19-21
8.         wt0         22-24
9.         wt1         25-27
10.  ;
11.  team = input(tmp, team.);
12.  drop tmp;
13.  datalines;
14.  Mike   red    128 80 95110 95
15.  John   blue   135 75 72 95 94
16.  Bill   green 120 80 72 75 75
17.  Jack   yellow110 75 78 90 85
```