

# **STAT 425 and STAT 625**

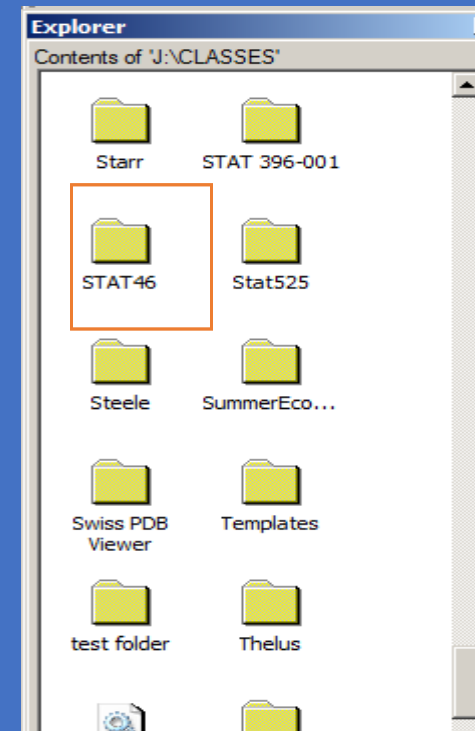
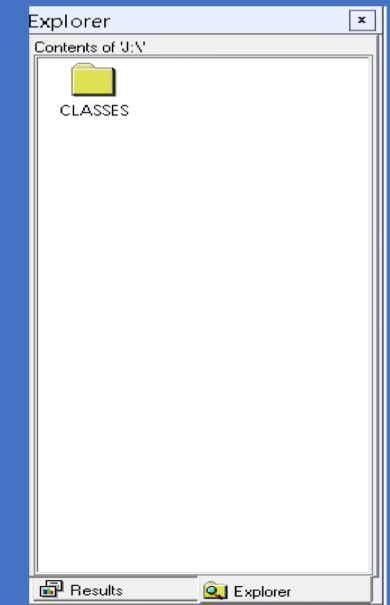
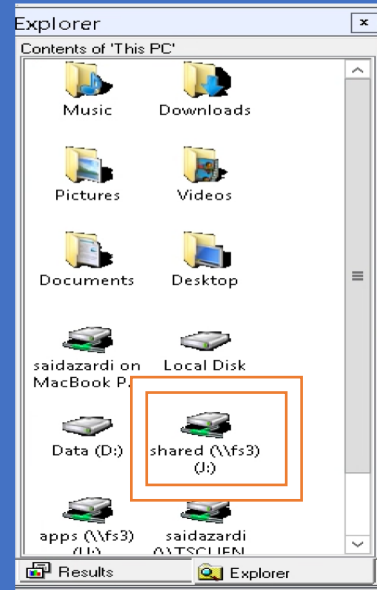
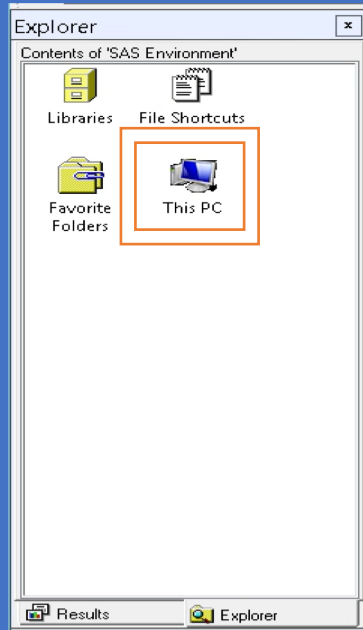
## **Statistical Software**

### **Lecture 3**

#### **Reading Raw Data from External Data Files**

# TO ACCESS THE CLASS FOLDER ON THE J DRIVE:

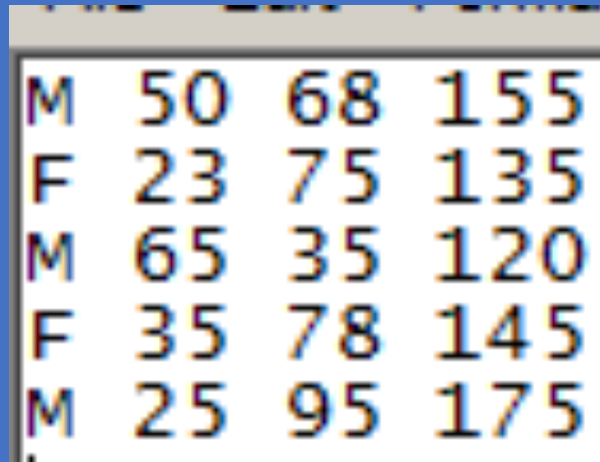
Open your SAS session, Then go to: This PC



STAT 46 is the folder where there will be the data for this course.

# Reading Data Values Seperated by blanks

- List input is one of the easiest ways to read data in SAS. By default SAS assumes that the data vlues are separated by one or more blanks.
- In the following data set called mydata.txt stored in **J:\CLASSES\STAT46** folder we have values representing gender, age(in years), height( in inches) and weight (in pounds) as follows:



M	50	68	155
F	23	75	135
M	65	35	120
F	35	78	145
M	25	95	175

- The following program will read data from this file and create a SAS data file:

```
data demographics;  
  infile 'J:\CLASSES\STAT46\mydata.txt';  
  input Gender $ age height weight;  
Run;  
  
Title "listing of data set demographics";  
  
proc print data=demographics;  
run;
```

## In the Data Step:

- Infile statement tells SAS where to find the data
- The Input statement - > name of variables to be associated with each data value in order. \$ sign for character data.

- Note: Title -> global statement - > Title of the data set
- Proc Print step lists the observations in the data set:

**listing of data set demographics**

Obs	Gender	age	height	weight
1	M	50	68	155
2	F	23	75	135
3	M	65	35	120
4	F	35	78	145
5	M	25	95	175

# Reading Data Values Separated by Commas

- Usually, in Windows or UNIX platforms data is stored in comma-separated values (csv) files.
- In these files data values are delimited by commas instead of blanks.
- They may or may not enclose character data in quotes.

Pam1.csv:

```
20,1,"Angola"  
5,1,"Croatia"  
13,1,"Sierra Leone"  
13,2,"Sierra Leone"  
17,1,"Rwanda"  
31,1,"Philippines"  
8,1,"Mali"  
27,1,"India"  
27,2,"India"  
27,3,"India"  
27,4,"India"
```

- Reading data from a CSV file:

```
data PAM;  
  infile 'J:\CLASSES\STAT46\pam1.csv' dsd;  
  input ID number_years country $;  
Run;
```

Notice the Infile option: “ dsd “: delimiters sensitive data:

- Changes the default delimiter from a blank to a comma
- If there are two delimiters in a row, it assumes there's a missing value in between
- Character values are placed in between quotes) single or double)

# Using an Alternative Method to Specify an External File

- An alternative method: use a separate FILENAME statement to identify the file to use and use this reference (fileref) in the INFILE statement:

```
filename PAM 'J:\CLASSES\STAT46\pam1.csv';  
  
data PAM;  
    infile PAM dsd;  
    input ID number_years country $;  
Run;
```

- Notice: the fileref PAM in the INFILE statement is not placed in between quotes. This is how SAS recognizes that PAM is not the name of the file where to read data, but rather a reference.



# Reading Data Values Separated by Other Delimiters

- It's possible to have other delimiters
- Add DLM=option to the INFILE statement
- Example:

M:50:68:155

F:23:75:135

M:65:35:120

F:35:78:145

M:25:95:175

*Infile 'file-description' dlm=':' ;*                      Or                      *Infile 'file-description'*  
**delimiter=':' ;**

For a TAB key - > use its hexadecimal equivalent. For ASCII files( American Standard Code for Information Interchange used in Windows or UNIX operating systems) use:

*Infile 'file-description' dlm='09' x;* For EBCDIC files ( used in most mainframe computers: Extended Binary –Coded Decimal Interchange Code) use:

*Infile 'file-description' delimiter='05'x ;*

# Placing Data Lines Directly in your Program

```
data Demos;  
  Input Gender $ age height weight;  
  
  datalines;  
M 50 68 155  
F 23 75 135  
M 65 35 120  
F 35 78 145  
M 25 95 175  
;  
run;
```

- INFILE statement removed and DATALINE statement added

# Reading Raw Data from Fixed Columns

- Many raw data files store specific information in fixed columns
- Advantages over data values separated by delimiters:
  1. No worries about missing data: you can leave the appropriate column blank
  2. In INPUT statement you can choose what column to read and in what order

There are two methods:

- Method 1: Column Input
- Method 2: Formatted Input

# Method 1: Column Input

File: Bank.txt

Description of the file:

Variable	Description	Starting Column	Ending Column	Data Type
Subj	Subj number	1	3	Character
DOB	Date of Birth	4	13	Character
Gender	Gender	14	14	Character
Balance	Bank Account Balance	15	21	Numeric

File J:\Classes\STAT46\Bank.txt

00110/21/1955M 1145

00211/18/2001F 18722

00305/07/1944M 123.45

00407/25/1945F -12345

- Sample Program demonstrating column input:

```
data Financial;  
  
  infile 'J:\Classes\STAT46\Bank.txt';  
  input Subj $      1-3  
        DOB  $      4-13  
        Gender $     14  
        Balance 15-21;  
  
run;  
  
title "Listing of Financial";  
proc print data=Financial;  
run;
```

Specify:

- a variable, a dollar sign if variable is a character
- the starting column and the ending column
- The number of column for each character value determines the number of bytes SAS uses to store these values, for numeric variables SAS uses 8 bytes regardless of the number of columns

## Alternative way to write the program:

```
data Financial;  
  
infile 'J:\Classes\STAT46\Bank.txt';  
input Subj $ 1-3   DOB $ 4-13   Gender $ 14   Balance 15-21;  
run;
```

The specification can be written in one line, but it's harder to read. It's better to specify each variable on a separate line.

Output:

**Listing of Financial**

Obs	Subj	DOB	Gender	Balance
1	001	10/21/1955	M	114.0
2	002	11/18/2001	F	1872.0
3	003	05/07/1944	M	123.4
4	004	07/25/1945	F	-1234.0

# Method 2: Formatted Input

- Formatted input reads data from fixed columns
- Reads character data, standard numeric data and non standard numeric data such as numbers with dollar signs and commas, dates in a variety of formats.
- Formatted input is the most common and most powerful of all input methods, to be used with nonstandard data in fixed columns.

```
data Financial;  
  
  infile 'J:\Classes\STAT46\Bank.txt';  
  input @1 Subj $3.  
        @4 DOB mmddyy10.  
        @14 Gender $1.  
        @15 Balance 7.;  
  
run;  
  
title "Listing of Financial";  
proc print data=Financial;  
run;
```

- @ signs in INPUT statement are called pointers
- Ex: @4 -> “go to column 4”
- Following variables names are Informats: they tell SAS how to read data
- The choice of informats is dictated by the data:
  - w.d -> reads standard numeric values: w tells SAS the number of columns to read

Ex: to read 123 with 3.0 informat - > 123.0

3.1 informat -> 12.3

If the number has already a decimal point, SAS ignores the d portion of the informat

Ex: if you read 1.23 with a 4.1 informat, SAS store a value of 1.23.

- \$w. -> reads w columns of character data

Ex: Subj takes up to 3 columns and Gender takes up a single column



- Reading the date:

- mmddyy10. -> mm/dd/yyyy.

- SAS reads the date and converts the value into a SAS date equal to the number of days from January 1, 1960

- 01/01/1960 with mmddyy10. informat -> SAS stores the value of 0

- 01/02/1960 -> 1

- @4 moves the pointers to the 4<sup>th</sup> column

- Mmddyy10. tells SAS to read the next 10 columns as a date

- Output:

**Listting of Financial**

Obs	Subj	DOB	Gender	Balance
1	001	-1533	M	114.0
2	002	15297	F	1872.0
3	003	-5717	M	123.4
4	004	-5273	F	-1234.0

To have the DOB written in a standard way and the Balance figures with a dollar sign:

```
proc print data=Financial;  
    format DOB mmddyy10.  
           Balance dollar11.2;  
run;
```

**Listting of Financial**

Obs	Subj	DOB	Gender	Balance
1	001	10/21/1955	M	\$114.00
2	002	11/18/2001	F	\$1,872.00
3	003	05/07/1944	M	\$123.40
4	004	07/25/1945	F	\$-1,234.00

- Alternatively, with a different format:

```
title "Listting of Financial";  
proc print data=Financial;  
    format DOB      date9.  
           Balance  dollar11.2;  
run;
```

**Listting of Financial**

Obs	Subj	DOB	Gender	Balance
1	001	21OCT1955	M	\$114.00
2	002	18NOV2001	F	\$1,872.00
3	003	07MAY1944	M	\$123.40
4	004	25JUL1945	F	\$-1,234.00

- Writing the format in a PROC step, associates the format only for that procedure.
- Writing the format in a DATA step makes it more permanent
- It's possible to remove a format for one or more variables by issuing a FORMAT statement and not specify a format:
- ex: *format Age;*

# Using Informats with List Input

If you have a blank or a comma delimited file with dates and character values longer than 8 bytes ( or other values that require an informat):

- > provide informat with list input: in INPUT statement, each such variable name must be followed by a colon then the appropriate informat:

Example:

“121”, “Christoph Douglas”, 21/4/1945, “\$35,300”

“122”, “Gerard Philippe”, 22/5/1945, “\$34,600”

“123”, “Michelle Noiret”, 23/4/1945, “\$54,200”

Program: Using Informats with List Input:

```
data List;  
  infile 'J:\\CLASSES\\STAT46\\list.csv' dsd;  
  input Subj      :      $3.  
        Name      :      $20.  
        DOB       :      mmddyy10.  
        Salary    :      dollar8.;  
  format DOB date9.    Salary dollar8.;  
run;
```

# Supplying an Informat Statement with List Input

- Alternatively the informats with list input could be provided before the INPUT statement as in the following example:

```
data List;
    informat Subj      :   $3.
              Name      :   $20.
              DOB       :   mmddyy10.
              Salary    :   dollar8.;
    infile 'J:\\CLASSES\\STAT46\\list.csv' dsd;
    input Subj
          Name
          DOB
          Salary
          format DOB date9.      Salary dollar8.;
run;
```

# Using List Input with Embedded Delimiters:

- If the data in list.csv file was placed in a text file with blanks as delimiters instead of commas and without quotes, as in list.txt:

121 Christoph Douglas 21/4/1945 \$35,300

122 Gerard Philippe 22/5/1945 \$34,600

123 Michelle Noiret 23/4/1945 \$54,200

So, use the Ampersand “&” informat modifier following Name and have two or more spaces between the end of the Name and the DOB as in:

121 Christoph Douglas    21/4/1945   \$35,300

122 Gerard Philippe     22/5/1945   \$34,600

123 Michelle Noiret      23/4/1945   \$54,200

- Demonstrating the Ampersand Modifier for List Input

```
data List;  
  infile 'J:\\CLASSES\\STAT46\\list.txt';  
  input      Subj      :   $3.  
             Name      &   $20.  
             DOB       :   mmddyy10.  
             Salary    :   dollar8.;  
  format DOB date9.    Salary dollar8.;  
run;
```