# MET MA 603:
# SAS Programming and Applications

## *Proc Import*

# Proc Import

Often, the data we want to work with isn't stored in a SAS file (e.g., database, excel spreadsheet, text file, etc.) **Proc Import** is a procedure that creates a SAS dataset based on a non-SAS file.

This is often the fastest and easiest method for data that isn't stored in SAS. However, because much of the procedure is automated, it lacks flexibility. If the source file has irregularities, this method may fail.

Use this method when the data isn't in a SAS file, and the data is organized in a reasonable structure (we will see examples of files where Proc Import will work and examples where it won't).

# Using Proc Import

```
Proc Import
  datafile = "C:\Users\govonlu\Desktop\Data\
scores1.txt"
              out = work.scores           dbms = tab
  REPLACE
;
run;
```

This first statement is a **Required Statement**. It tells SAS where to find the file and what to name the SAS dataset.

When **Replace** is included in the Proc Import statement, any existing SAS dataset of the same name and in the same library as the dataset in the "out=" argument will be replaced. If Replace isn't used the import will be cancelled.

**DBMS** tells SAS the type of data being used. Based on the extension, SAS will make an assumption about how the data is separated. If the assumption SAS makes is appropriate then DBMS is not needed.

# Delimiter Statement

```
Proc Import
  datafile = "C:\Users\govonlu\Desktop\Data\
scores2.txt"
              out = work.scores          dbms = tab
      REPLACE                ;
  delimiter = ',';
run;
```

When reading a file, SAS needs to know where one observation ends and the next begins. **Delimiters** are characters that indicate how the data is separated.

Based on the file extension and DBMS argument SAS will make an assumption about what the delimiter is. If the delimiter in the file is something different than the delimiter statement is needed.

Common choices of delimiters are spaces, commas, or tabs, but almost anything could be used.

# DBMS and File Extensions

The table below shows the DBMS associated with some common file types that might be imported into SAS.

**Available DBMS Specifications**

| Identifier | Input Data Source | Extension |
|---|---|---|
| ACCESS | Microsoft Access database | .MDB |
| DBF | dBASE file | .DBF |
| WK1 | Lotus 1 spreadsheet | .WK1 |
| WK3 | Lotus 3 spreadsheet | .WK3 |
| WK4 | Lotus 4 spreadsheet | .WK4 |
| EXCEL | Excel Version 4 or 5 spreadsheet | .XLS |
| EXCEL4 | Excel Version 4 spreadsheet | .XLS |
| EXCEL5 | Excel Version 5 spreadsheet | .XLS |
| EXCEL97 | Excel 97 spreadsheet | .XLS |
| DLM | delimited file (default delimiter is a blank) | .* |
| CSV | delimited file (comma-separated values) | .CSV |
| TAB | delimited file (tab-delimited values) | .TXT |

# Other Optional Statements

```
Proc Import
    datafile =       "C:\Users\govonlu\Desktop\Data\
occupancy.txt"
                     out = work.occupancy
REPLACE          ;
    getnames = NO ;       datarow = 6 ; guessingrows
= 3000 ;
    run;
```

By default, Proc Import assumes that the first line contains the variable names. Use the **getnames** statement to specify whether to use the first line as the variable names.

By default, Proc Import starts reading data from the first line of the file. Use the **datarow** statement to specify which line to start at.

By default, Proc Import determines the variable types and lengths based on the first 20 rows. Use the guessingrows statement to specify the number of guessing rows (note:

# Import Wizard

Proc Import can also be initiated via point-and-click by choosing Import Data from the File menu.

The Wizard will prompt for the type and location of the file, and the desired library and name for the SAS dataset.

Using the Import Wizard is really just another way of using Proc Import. Each piece of information the Wizard asks for

Based on what the user enters, SAS is generating Proc Import code. In the last step of the Wizard, choosing to save the Proc Import statements instructs SAS to save this code in a file. The code can be viewed, ran, or modified.

# Where Proc Import Fails

Based on a scan of your data, SAS makes assumptions about the data and the dataset it should create. This can be convenient, but sometimes SAS doesn't make the assumptions we want.

Sometimes the data file is structured in a way that Proc Import can't deal with. This causes the method to fail.

When data is structured in columns SAS may not be able to determine how it is delimited.

Sometimes data is formatted, such as with percentages, currencies, or scientific notation. This may cause Proc Import to assign the wrong data type.

Sometimes our data has numbers that we want to be stored as text, such as zip codes or ID numbers. Proc Import will generally create numeric variables.

# Practice

Use Proc Import to import "occupancy.csv".

Use Proc Import to import "days_of_week1.txt".

Use Proc Import to import "days_of_week2.txt". Then, use the Import Wizard to import the same dataset, and save the code that SAS generates. How does the code you wrote compare to the code SAS generated? What is the smallest number that you could set guessingrows to, such that the data will still be imported correctly?

# Readings

- Textbook sections 2.3, 2.16
- https://v8doc.sas.com/sashtml/proc/z0308090.htm

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder