HSCI 416
Lab 1

FALL 2024

Mina Moeini

Currently PhD student

MSc in Operation Research

BSc in Mathematics and application

Email: mina_moeini@sfu.ca

Objectives before tutorial

1. Overview of the SAS Program Interface
2. Basic SAS Syntax
3. How to access SAS
4. Use SAS on Demand or SAS program
5. Get SAS running

Introduction to SAS

- SAS stands for Statistical Analysis System.
- A powerful software suite for data management and analytics
- **Data Management:**
  - Efficient handling of large datasets.
  - Data cleaning and transformation.
- **Advanced Analytics:**
  - Multivariate analysis
  - Predictive modeling
  - Machine learning integration
- **Reporting and Visualization:**
  - Customizable reports
  - Interactive dashboards

Learning SAS

- Easy-to-learn programming language
- Learn through practicing and correcting errors
- Book Resources
  - ***The Little SAS Book***, Delwiche and Slaughter
  - ***Learning SAS by example***, Cody

Installing SAS:

Option 1: Installing SAS 9.4 locally on your computer (Windows/Linux only).

- SFU IT software includes SAS 9.4 and a license (which will need to be renewed during the semester, but we can cover that in lab when necessary).
- https://www.sfu.ca/information-systems/services/software/sas-9-4/download-sas-9-4.html

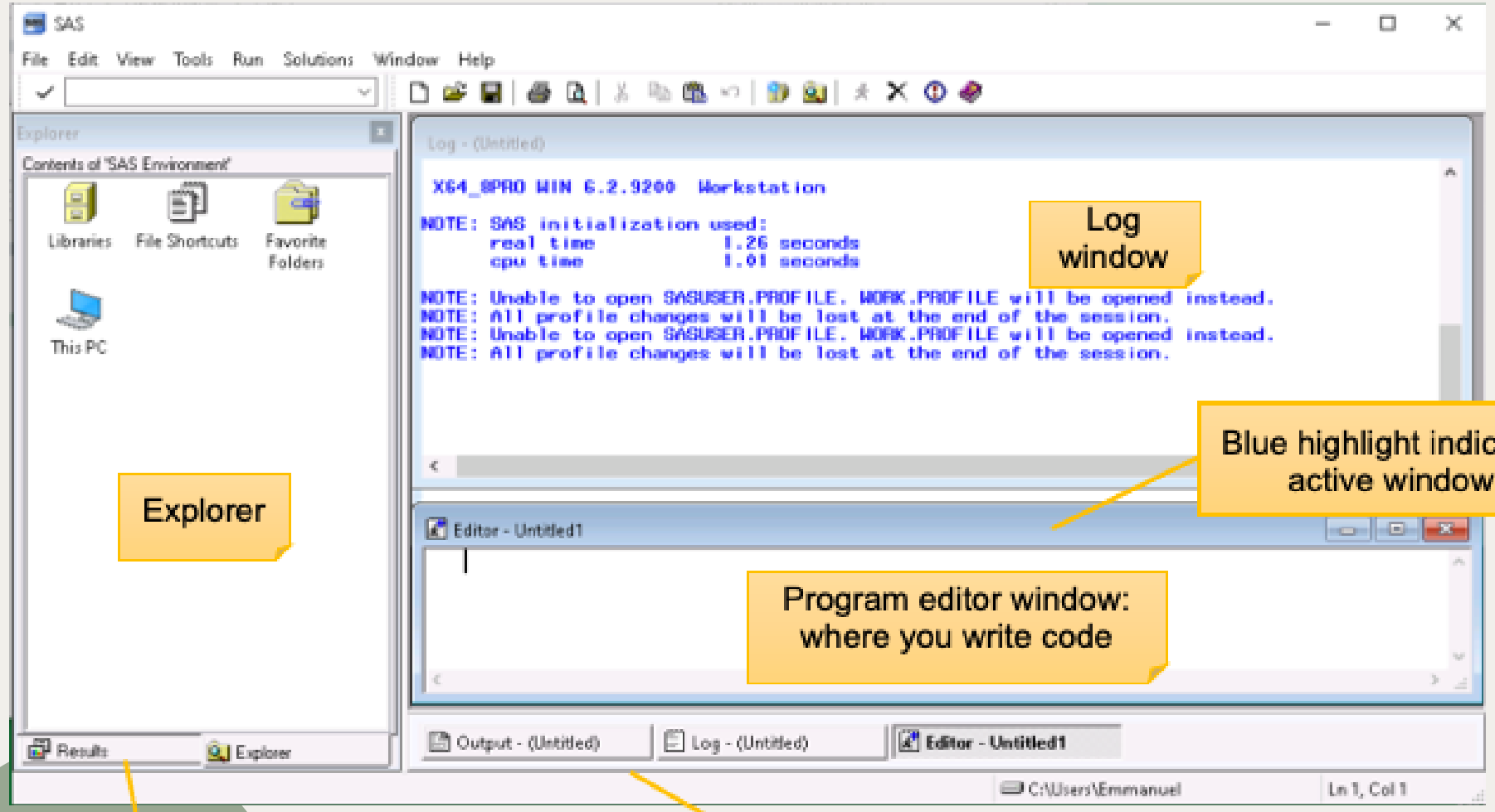
Option 2: SAS on Demand

- SAS provides on online web-based server for students to access SAS. See video on canvas called "How to SAS on Demand" for a walkthrough on setting up and how to use. Video is provided by a previous years TA (with permission).

Option 3: Remote connecting to a lab computer at SFU which will have SAS installed.

- This option will differ based on your student status (graduate or undergraduate).
- **Graduate** student remote access instructions: https://sfu.teamdynamix.com/TDClient/255/ITServices/KB/ArticleDet?ID=3671
- **Undergraduate** student remote access instructions:
- https://sfu.teamdynamix.com/TDClient/255/ITServices/KB/ArticleDet?ID=3670

SAS (9.4) Windows environment



Log window

Blue highlight indicates active window

Explorer

Program editor window: where you write code

Click for results window

Click for output window

Terminology

Database	A comprehensive collection of administrative data. Example: All physician visit records (e.g., MSP billing records).
Data File/Dataset	A specific component of a database, containing either the full set or a subset of data. Example: Hospital discharge records for St-Paul's Hospital, focusing on male patients aged 12-24 years or stroke admissions.
Data Record/Observation	A single row of data representing one individual or event (e.g., a patient admission).
Data Field/Variable	A specific column in the data file representing a particular attribute.
Data Value	The actual information within a data field for a specific variable

Basics of SAS syntax

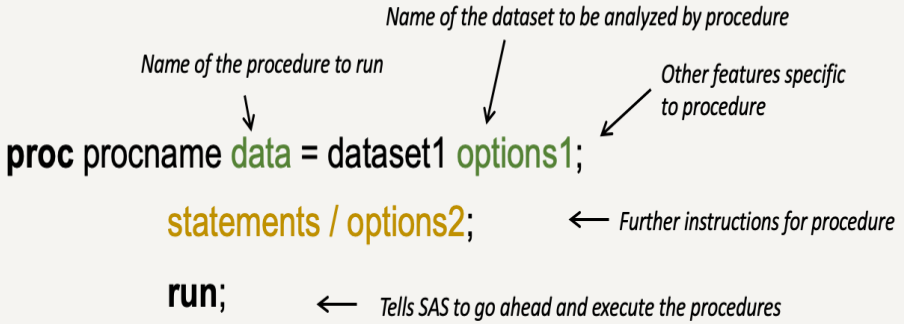
- **DATA steps**: Creates SAS data sets / Read and modify Data

```
data dataset2;      ← New (temporary) dataset to be created after procedures
    set dataset1;   ← Pre-existing dataset to be used
    statements;     ← What you want SAS to do; procedures
    run;            ← Tells SAS to go ahead and execute the procedures
```

- Statements separated by “ ; ”
- **SAS**: “Semicolon - Always Semicolon"

Basics of SAS syntax

Proc: It used to perform various types of data analysis and processing. Each PROC (short for procedure) serves a specific purpose, such as summarizing data, analyzing statistics, or creating graphical reports.



Example:

- PROC PRINT
- PROC PRINT DATA=work.sales_data;
- RUN;

Goals for Today

Reading and Importing Data

- Refer to slides and code examples.
- Access raw data and code for data reading.

Describing the Data

- Use `print` and `contents` to understand data structure.
- Instructions in slides and code.

Saving Datasets

- Guidance provided in slides and code.
- Options: Temporary, Permanent, Libraries, Open.

Subsetting Data

- Focus on writing code for:
  - Selecting observations by random sampling, variables, or specific criteria.

Exercises

- Refer to the Word document for practice exercises.

Practice Makes Perfect

- The best way to improve coding skills is by writing and practicing code.

Reading Data in SAS

SAS reads data from various sources and stores it in a unique format called a **SAS Dataset**, which includes:

- **Descriptor Portion**: Contains metadata about the data, such as variable names and types.
- **Data Portion**: Holds the actual data values.

Rules for SAS Dataset Names:

- **Length**: Must be no longer than 32 characters.
- **Starting Character**: Must begin with a letter or an underscore (_).
- **Restrictions**:
  - Cannot contain spaces or dashes.
  - Only letters, numbers, and underscores are allowed.

Reading Data in SAS

Reading Data from an Excel File:

```
proc import
datafile='/folders/myfolders/sales_data.xlsx'
   out=sales_data
   dbms=xlsx
   replace;
   sheet='Sheet1';
run;


proc print data=sales_data;
run;
```

- The `proc import` procedure reads data from an Excel file specified in the `datafile` option.
- The `out` option specifies the name of the output dataset.
- The `dbms=xlsx` option specifies the type of Excel file being read.
- The `sheet` option specifies which sheet to read from.

Saving datasets

Telling SAS where to FIND and STORE data

- SAS automatically saves datasets temporarily during a session in a workspace called "Work"
  - In the output window, these datasets are shown as `work.(dataset_name)`.
- To save datasets permanently:
  - Create a specific folder (e.g., in MYSFUFILES).
  - Use the `libname` statement to define the storage location and reference it easily in your SAS programs (e.g., `libname saslab2`).

Saving datasets

Using the Temporary Work Library:

```
data work.sales_data;
   input CustomerID $ Product $ SalesAmount;
   datalines;
   C001 Book 20
   C002 Pen 5
   C003 Notebook 10 ;
run;


proc print data=work.sales_data;
run;
```

Creating a Permanent Library Using **`libname:`**

```
libname mydata '/folders/myfolders/permanent_data';
data mydata.employee_data;
   input EmployeeID $ Name $ Department $ Salary;
   datalines;
   E001 John HR 60000
   E002 Alice IT 75000
   E003 Bob Sales 55000  ;
run;
proc print data=mydata.employee_data;
run;
```