Introduction to SAS

Katia Bulekova

Research Computing Services

BOSTON UNIVERSITY

SAS Software Options

- SAS Windowing Environment:
  - for a local desktop submit a request through BUMC IT
  - Shared Computing Cluster (requires a project account)
  - BU's Computer Labs
- SAS Studio
- SAS Enterprise Guide
- SAS OnDemand for Academics (University Edition)

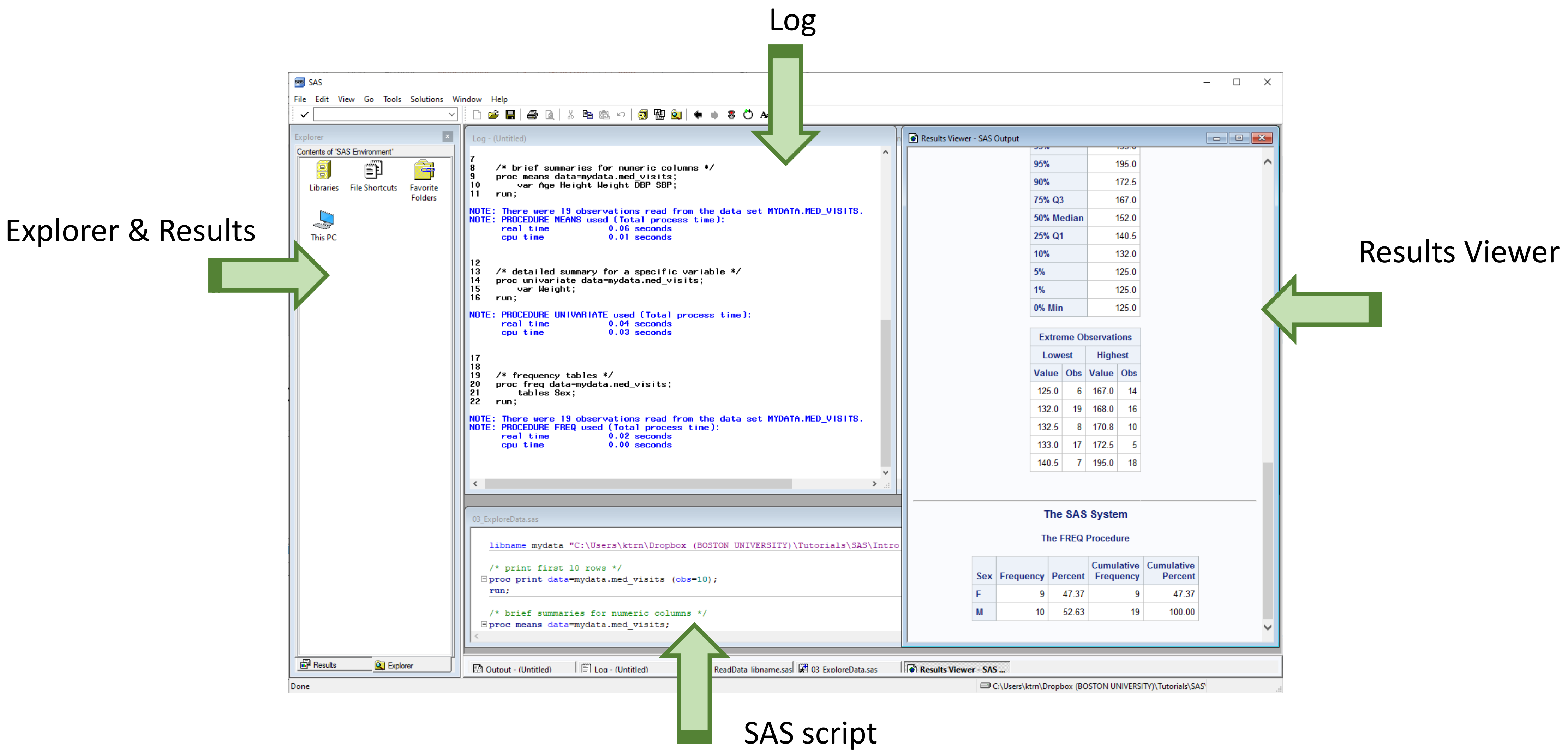
SAS Help

- SAS Online Documentation: http://support.sas.com/documentation/onlinedoc/index.html
- SAS Institute Home page: https://www.sas.com/en_us/home.html
- Within SAS: `Help` menu:
  - SAS Help and Documentation
  - Getting Started with SAS Software

SAS Rules

- SAS variables must be 32 characters or less; Use only letters, digits, and underscore characters
- SAS is not case-sensitive except for values inside of quoted strings
- Missing numeric values are represented by a period "."
- Each SAS statement should end with a semicolon ";"
- Lines beginning with an asterisk "*" are treated as comments. They should also end with a semicolon ";"

SAS Windowing Environment



```
7
8    /* brief summaries for numeric columns */
9    proc means data=mydata.med_visits;
10       var Age Height Weight DBP SBP;
11   run;

NOTE: There were 19 observations read from the data set MYDATA.MED_VISITS.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.06 seconds
      cpu time            0.01 seconds


12
13   /* detailed summary for a specific variable */
14   proc univariate data=mydata.med_visits;
15       var Weight;
16   run;

NOTE: PROCEDURE UNIVARIATE used (Total process time):
      real time           0.04 seconds
      cpu time            0.03 seconds


17
18
19   /* frequency tables */
20   proc freq data=mydata.med_visits;
21       tables Sex;
22   run;

NOTE: There were 19 observations read from the data set MYDATA.MED_VISITS.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           0.02 seconds
      cpu time            0.00 seconds
```

95%	195.0
90%	172.5
75% Q3	167.0
50% Median	152.0
25% Q1	140.5
10%	132.0
5%	125.0
1%	125.0
0% Min	125.0

Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
125.0	6	167.0	14
132.0	19	168.0	16
132.5	8	170.8	10
133.0	17	172.5	5
140.5	7	195.0	18

The SAS System

The FREQ Procedure

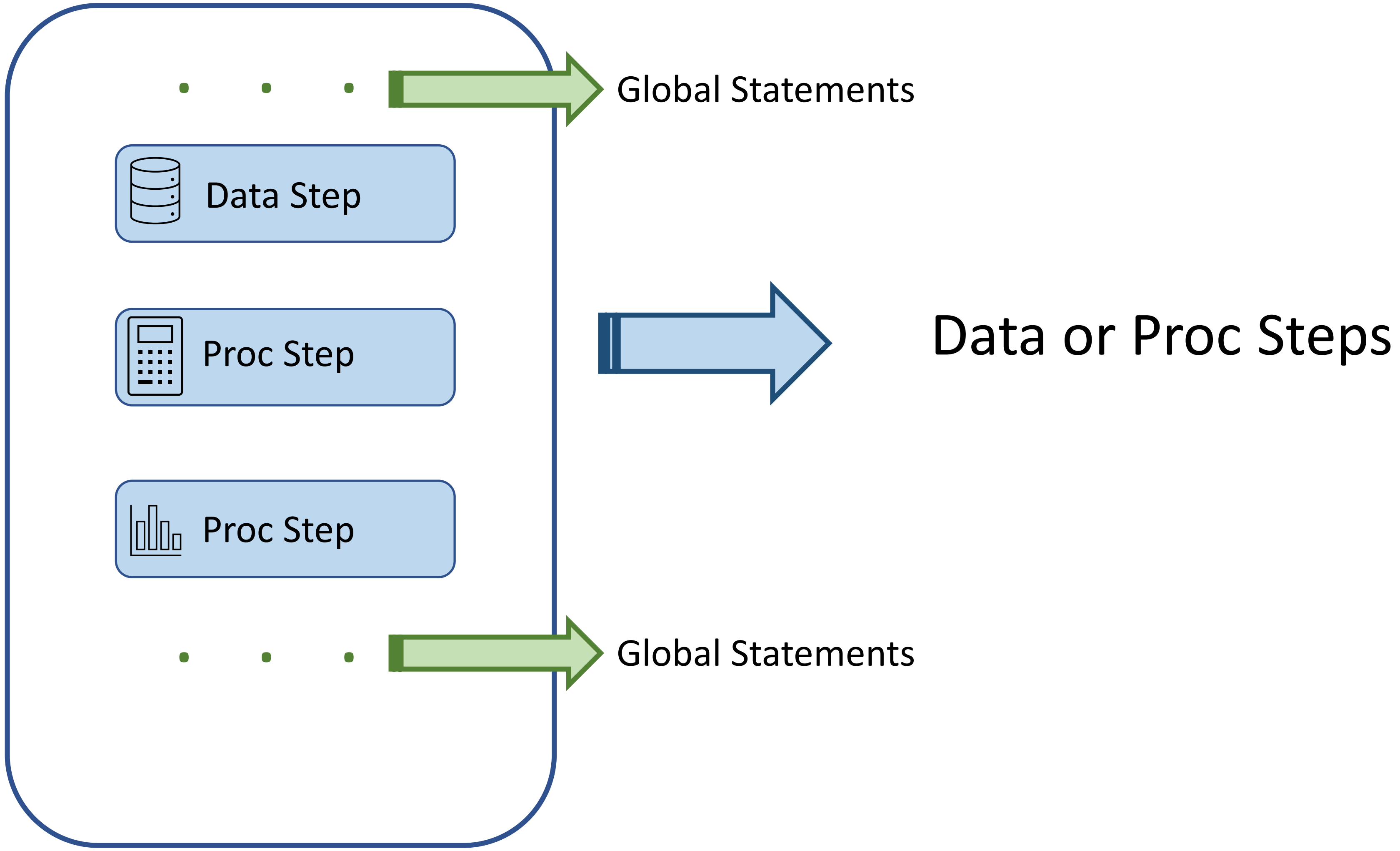
Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	9	47.37	9	47.37
M	10	52.63	19	100.00

```
libname mydata "C:\Users\ktrn\Dropbox (BOSTON UNIVERSITY)\Tutorials\SAS\Intro

/* print first 10 rows */
proc print data=mydata.med_visits (obs=10);
run;

/* brief summaries for numeric columns */
proc means data=mydata.med_visits;
```

SAS Program Structure



SAS Program Structure

```
data visits;
   set "C:\work\med_visits.sas7bdat";
   bmi=weight/height**2 *703;
run;

title "Medical Visits Information";

proc print data=visits;
run;

proc means data=visits;
   var bmi;
run;
```

- Each SAS step start with either "**data**" or "**proc**" keyword
- Each step ends with a "**run**" (or sometimes "quit") command
- Each step may contain one or more *statements*
- Each *statement* must end with a semicolon
- Global statements (*title, option, libref*, etc. do not need the "**run**" keyword, but must finish with a semicolon
- Unquoted values can be lowercase, uppercase, or mixed case.

SAS Program Structure

```
data visits; set "C:\work\data.sas7bdat"; bmi=weight/height**2 *703; run;
proc means data=visits; var bmi; run;
```

```
data visits;
   set "C:\work\data.sas7bdat";
   bmi=weight/height**2 *703;
run;

proc means data=visits;
   var bmi;
run;
```

Format your code, so it is easier to read and understand

SAS Program Structure

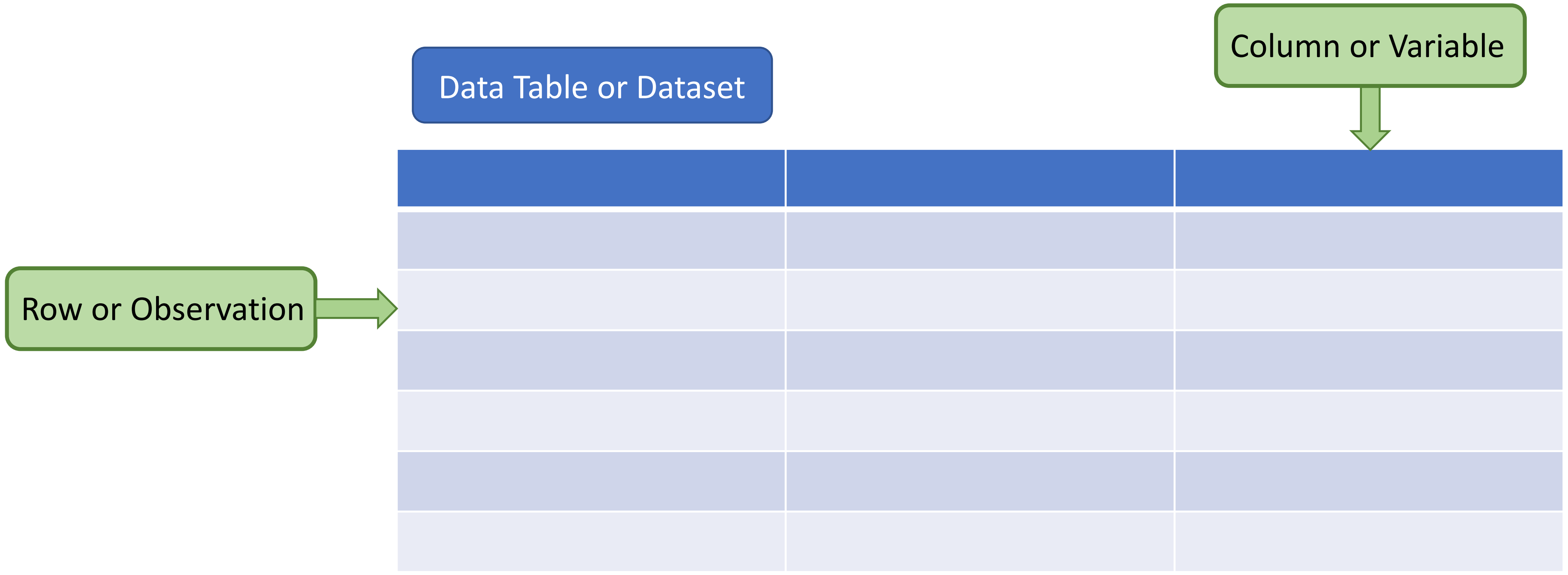
```
/* Read the data and
   define a new variable */
data visits;
  set "C:\work\data.sas7bdat";
  bmi=weight/height**2 *703;
run;
```

Multi-line comments.
Start with /* and end with */

```
* Calculate mean for bmi variable ;
proc means data=visits;
  var bmi;
run;
```

Single-line comments.
Start with * and end with ;

Accessing Data



SAS Data input: *Direct input*

```
* Directly enter the data;
* Use cards or datalines keywords. It must appear
after the input statement;
data mydata;
input id name $;
datalines;
 1 Alex
 2 Bob
 3 Clara
;
```

The name "*mydata*" is a name you assign to your dataset that you can later use in your script. Once this statement is executed, you can find your dataset inside the "Work" library in the Explorer window

Practice

00_InputData.sas

Open this script with SAS and run the existing data statement;

Create your own dataset and check if it appears correctly in the *Work* library;

Do not forget to close the window displaying the dataset!

SAS Data input: *Direct input from a file*

- `Enter the data into a file and then read using the` **`data`** `statement;`

```
data mydata;
infile 'Datasets\mydata.dat' missover;
input id name $;
```

Notice that in this case the `infile` statement precedes the `input` statement

Practice

00_InputDataFromFile.sas

Open this script with SAS and run the existing data statement;

Modify the input file and execute the data statement again;

Check if the dataset in the *Work* library changed.

SAS Data input: *More information on Format*

You can find more information on the format of the input data in the SAS Help Center:

`INPUT Statement`

`INPUT Statement: Formatted`

`Dates, Times, and Datetime values`

SAS Data Import: *CSV*

```
* Import data from a csv file;
proc import
  datafile="/path/to/file.csv"       /* input file name            */
  out=visits                         /* name of the dataset        */
  dbms=csv                           /* file type                  */
  replace;                           /* replace current if needed  */
run;

proc print data=visits (obs=10);     /* print first 10 observations */
run;
```

SAS Data Import: *xlsx*

```
proc import
  datafile="/path/to/file.xlsx"      /* input file name            */
  out=visits                         /* name of the dataset        */
  dbms=xlsx                          /* file type                  */
  replace;                           /* replace current if needed  */

  * Additional Options;
  datarow=2;                      /* data starting row            */
  sheet="MedVisits";              /* default - the first sheet    */
  range="MedVisits$A1:G20";       /* default - all rows and columns */
run;
```

Practice

01_ReadData_CSV.sas

01_ReadData_XLSX.sas

Open these scripts with SAS and run the `proc import` statements;

Execute `proc print` statement to see the data;

Modify the number of observations printed in the "Results Viewer" window.

SAS Data: *reading SAS datasets (sas7bdat)*

You can make SAS read the data directly from the sas7bdat file using **DATA** statement:

```
data visits;
set="/path/to/file.sas7bdat";                    /* sas
file name */
run;
```

In this case, the data for the current session will be stored in a temporary **Work** library

SAS Data: *reading SAS datasets (sas7bdat)*

You can use libname to specify the location where SAS input file(s) are stored on your local drive:

```
libname mylib "path/to/directory/;
```

In this case, you can refer to the dataset directly by its name, specifying mylib as the prefix:

```
proc print data=mylib.my_visits
run;
```

Here "my_visits" is the name of the sas7bdat file

Practice

01_ReadData_sas7bdat.sas

01_ReadData_libname.sas

Open these scripts and execute the code;

For the `01_ReadData_libname.sas` script, check if the directory specified in the `libname` statement appears in the list of your active libraries

SAS Program Structure

SAS Program consists of:

- global statements
- data steps
- proc steps

Practice

02_ProgramStructure.sas

1. Explore the script and adjust the path to the input dataset if needed;
2. In the DATA step we introduce a new variable BMI. Add another variable `MAP=2/3*DBP + 1/3*SBP`
3. Examine the output of the `contents` procedure. Does this new variable appear in the output?
4. Check the visits dataset in the `Work` library. What variables are included in this dataset?

Data Exploration

Procedures:

- `print`	
- `contents`:	information on dataset
- `means`:	summary statistics
- `univariate`:	detailed summary statistics
- `freq`:	frequency tables

Data Exploration: `contents`

```
proc contents data=<libref>.<dataset>;
run;
```

Optional: TITLE statement

Use `order` to specify the order of the columns (*collate* or *varnum*):

```
proc contents data=<libref>.<dataset> order=collate;
run;
```

Practice

03_ExploreData.sas

1. Explore the script and adjust the path to the input dataset if needed;

2. In the DATA step we introduce a new variable BMI. Add another variable `MAP=2/3*DBP + 1/3*SBP`

3. Examine the output of the `contents` procedure. Does this new variable appear in the output?

4. Check the visits dataset in the `Work` library. What variables are included in this dataset?

Data Analysis : `means`

```
proc means data=<lib>.<data>;
run;
```

By default, all numeric variables are listed. You can select a subset using **var** statement;

The following statistics included:

N (number of observations), **Mean**,

STD (standard deviation), **Max**, **Min**

Data Analysis: **means**

```
proc means data=<lib>.<data> N Mean STD MaxDec=2;
```

Specify statistics

Specify number of decimal places

```
   var SBP DBP;
   class Sex;
   where Age > 50;
run;
```

Select variables

Stratification variable(s)

Condition

Data Analysis : `means (by vs. class)`

```
proc sort data=<lib>.<data>;
by Sex;
run;
```

The data must be sorted first!

```
proc means data=<lib>.<data>;
   var SBP DBP;
   by Sex;
run;
```

The output will contain separate tables for each unique value of the stratification variable

Practice

04_Means.sas

1. Explore the script;
2. Modify the condition to filter the data and examine the output

Data Analysis : `univariate`

```
proc univariate data=<lib>.<data>;
run;
```

`Univariate` procedure is used to explore the distribution of the variables

- Calculate extreme observations
- Supports normality tests
- Can generate plots (histograms, boxplot, etc.)

Data Analysis : `univariate`

```
proc univariate data=<lib>.<data>;
run;
```

The default output includes **5 tables**:

- **Moments**: N, Mean, STD, Skewness, Variance, Kurtosis, …
- **Basic Stats**: mean, median, mode, std, var, range, …
- **Student t-test**
- **Quantiles**
- **Extreme Observations**

Data Analysis : `univariate`

```
proc univariate data=<lib>.<data>;
   var DBP;
   class Sex;
   where Age > 50;
run;
```

Just like the **`mean`**, **`proc univariate`** can use **`var`**, **`class`**, and **`where`** statements. But it has many other statements you can include.

Data Analysis : univariate

```
/* plot a histogram with a normal curve */
proc univariate data=<lib>.<data>;
  var DBP;
  histogram / normal;
run;
```

Practice

05_Univariate.sas

1. Explore and run the script;

2. Change the variables in **`var`** and **`where`** statements and examine how output changes.

3. Plot histogram and examine the output

Data Analysis : `freq`

```
/* frequency tables*/
proc freq data=<lib>.<data>;
   tables Sex;
run;
```

Unlike **`means`** and **`univariate`** procedures, **`freq`** uses "tables" statement to select variables .

Use "*" to produce a crosstabulation table;

Data Analysis : freq

```
/* two-way frequency tables*/
proc freq data=<lib>.<data>;
  tables column1*column2;
run;
```

Practice

06_freq.sas

1. Explore and run the script;
2. The last two proc statement use a dataset from the sashelp library.
   Apply the print and content procedures to explore the dataset first;
3. Modify **`freq`** procedures to build tables for Cylinders and Make variables

ODS Statement

ODS allows the modification of the behavior of the procedures.

Use ODS trace on/off to view the table names generated by SAS:

```
ods trace on;
proc univariate data=mydata.med_visits;
      var Height ;
run;
ods trace off;
```

Check the output in the log;

ODS Statement

Use **ODS** to select specific table(s)

```
ods select BasicMeasures ExtremeObs;
proc univariate data=mydata.med_visits;
       var Height ;
run;
```

ODS Statement

Use **ODS** to change global parameters

```
ODS graphics on; /* enable graphics */

ODS noproctitle; /* no title for the output tables */
title "my own title";    /* Use custom title */
. . .
* Return to the default titles;
title;
ODS proctitle;
```

Practice

07_ods.sas

1. Run the script one procedure at a time;
2. Check the output when you run the procedure with "ODS trace on"
3. Use ODS to control what tables you want to use for the output

Evaluation

Please complete the evaluation form:

rcs.bu.edu/eval