P8130 Recitation 1: Reading data in SAS and Descriptive Statistics

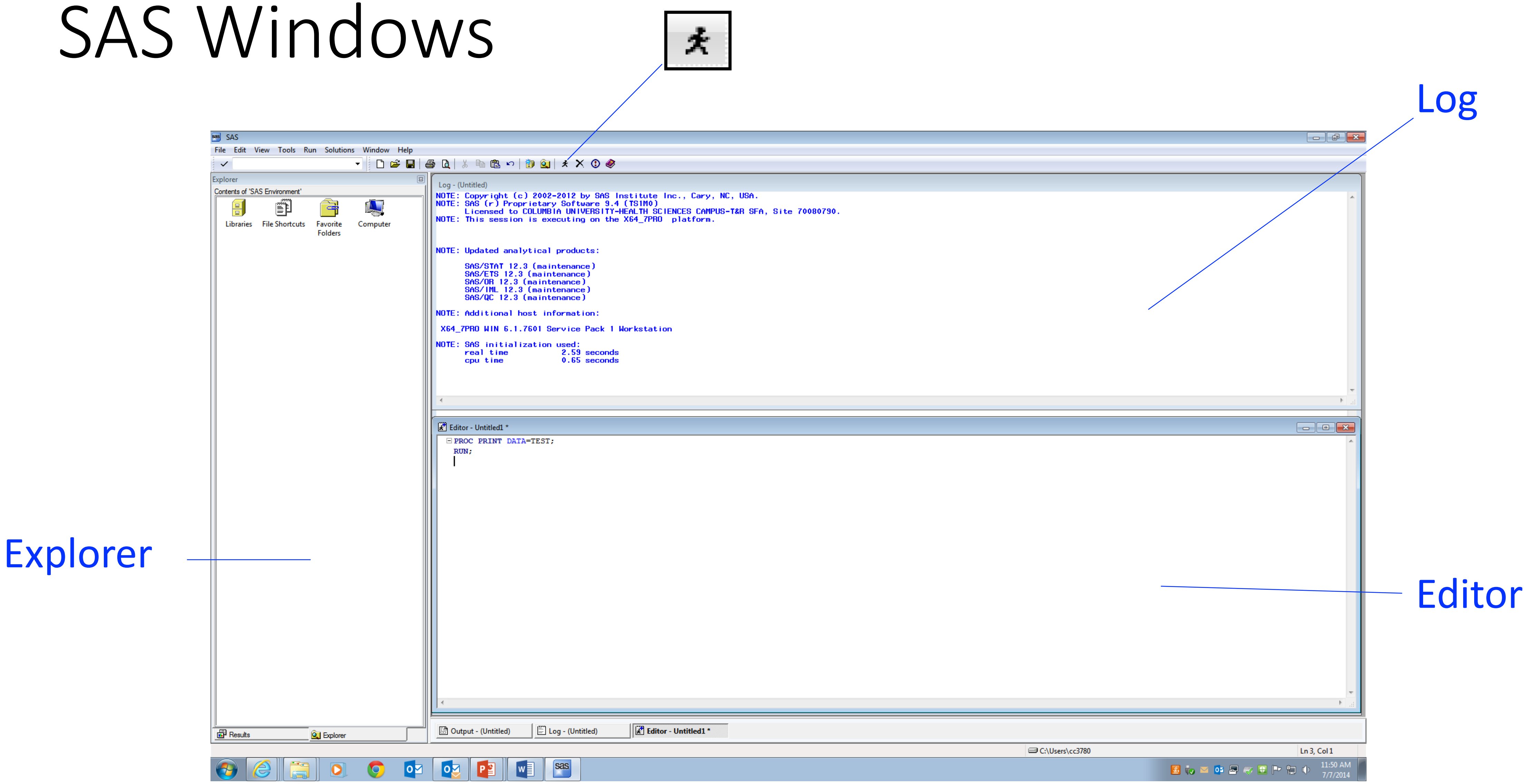
Zilan Chai

Sep. 18th/20th 2017

Outline

- Intro to SAS (windows, basic rules)
- Getting Data into SAS
- Descriptive Statistics

SAS Windows



```
NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M0)
      Licensed to COLUMBIA UNIVERSITY-HEALTH SCIENCES CAMPUS-T&R SFA, Site 70080790.
NOTE: This session is executing on the X64_7PRO  platform.



NOTE: Updated analytical products:

      SAS/STAT 12.3 (maintenance)
      SAS/ETS 12.3 (maintenance)
      SAS/OR 12.3 (maintenance)
      SAS/IML 12.3 (maintenance)
      SAS/QC 12.3 (maintenance)

NOTE: Additional host information:

 X64_7PRO WIN 6.1.7601 Service Pack 1 Workstation

NOTE: SAS initialization used:
      real time           2.59 seconds
      cpu time            0.65 seconds
```

```
PROC PRINT DATA=TEST;
RUN;
```

- **Editor window** – a text editor, where you enter SAS commands and create SAS programs.
- **Log window** - after you submit a SAS program, any notes, errors, or warnings associated with your program as well as the program statements themselves will appear in the Log window.
- **Output window** - displays any printable results
- **Explorer window** – gives easy access to your SAS files and libraries
- **Results** – table of contents for the Output window (tree lists)

SAS Language

- Most programs have two major components:
  - **Data step(s)**–reads data, manipulates/combines it and print reports
  - **Procedure step(s)** –perform analysis on the data and generate output

  Every step ends with a **run;** statement

- Sequential execution of statements
  - Every statement ends with a semicolon **;**
  - Statement can continue on the next line
  - Statements can be on the same line as other statements
  - NOT case sensitive! Exception –“quoted” strings

- Comments: can be inserted in two ways `* ... ;` OR `/* ...*/`

Simple examples

```
* The following data step inputs 6 variables for 4 individuals;
data sbp;
  input pat_name $ pat_id gender $ year1 year2 year3;
/*year1-year3*/
  cards;
  John  1002 M 90 120 125
  Alice 1003 F 140 148 116
  Mike  1004 F 121 130 117
  Barbara 1005 M 151 144 148
;
run;



proc import datafile= 'path:\filename' out= dataname;
run;
```

Rules for SAS names

- Follow these simple rules when making up names for variables and data set members:

  - Names must be 32 characters or less
  - Names can contain only letters, numbers, or underscores.(No %$!*&#@)
  - Names must start with a letter or an underscore
  - Names can contain upper- and lowercase letters.

Methods of getting your dada into SAS

1. Entering data directly into SAS data sets

2. Creating SAS data sets from raw data files

3. Reading Files with the IMPORT Procedure

4. Reading Files with the Import Wizard

5. Using existed SAS data sets

1. Entering data directly into SAS data sets (Data Step – INPUT Statement)

```
* The following data step inputs 6 variables for 4 individuals;
data sbp;
  input pat_name $ pat_id gender $ year1 year2 year3;
/*year1-year3*/
  cards;
  John  1002 M 90 120 125
  Alice 1003 F 140 148 116
  Mike  1004 F 121 130 117
  Barbara 1005 M 151 144 148
;
run;
```

DATA statement names the dataset

INPUT statement lists variable names,

also specifys the data format

CARDS statement lists data

RUN statement tells SAS to execute the block of code of this step

Some modifiers for the INPUT statement

&	tells SAS to use two whitespaces characters to signal the end of a character variable
@	holds the line to allow further input statements in this iteration of the data step on the same data
@n	placed before a variable name, it moves the pointer to column n
+n	placed before a variable name, it moves the pointer to the right n columns
@@	holds the line to allow continued reading from the line on subsequent iterations of the data step (lets SAS know that there are multiple observations per line)

2. Creating SAS data sets from raw data files (Data Step –INFILE Statement)

- Read external files containing data
- The INFILE statement must proceed the INPUT statement

```
data iron;
infile ‘path\iron.dat';  *where to find data;
input ID $ serferr ironaa tibcaa trnsfrec trnsfia; *variable names;
run;
```

- .dat’ files are primarily associated with ‘Data’ and have no specific structure (can be text, graphic, or general binary data)

INFILE Statement - Options

Missover	set values to missing if an input statement would read more than one line
Lrecl=num	treat the input as having a length of *num*characters. Required if input records are longer than 256 characters
Dlm='chars'	use the characters in *chars*instead of blanks and tabs as separators in list (free-form) input
Dsd	read comma-separated data
firstobs=num	start reading data on the *num*line of the file
Obs=num	stop reading after the *num*data line (e.g., obs=5)

3. Reading Files with the IMPORT Procedure

- Easier for certain types of data files
- Scans the data file, automatically determines the variable types, assigns lengths to the character variables, and can recognize some date formats

```
proc import datafile= 'path:\filename' out= dataname;
run;
```

PROC IMPORT -Options

DBMS=	specify the file extension if there is none: CSV (comma-delimited), TAB (tab-delimited), XLS (Excel), ACCESS, DTA (Stata), DLM (other), etc.
Replace	overwrite an existing data set with the one specified in out= option
Delimiter	specify what delimiter is used; default is a blank space (e.g., delimiter='/ ')
Getnames=NO	do not get variable names from the first line of input file. Default is YES. In NO, the variables are named Var1, Var2, …

4. IMPORT WIZARD

- SAS has a nice feature for importing data:
    1. Go to File -> Import Data
    2. Select the type of file you want to import (e.g., .csv, .txt)
    3. Locate the file
    4. Choose destination (default WORK library) and name for the data set (member)
- Proc Import statements will automatically be generated

EXPORT WIZARD

- Similarly, you can export data files (e.g., Excel)

    1. Go to File -> Export Data

    2. Choose Library and Member

    3. Select the type of data you wish to export

    4. Choose destination folder and filename

5. Reading SAS dataset – SET statement

- If the data already exits as a temporary/permanent SAS data set, we do not need to use INFILE/INPUT statements
- SET statement reads a temporary/permanent SAS data set into a new one, so that you can modify it (add new variables, subset, etc.)

```
data new-data-set;
 set old-data-set;
run;

* If permanent data set;
data new-data-set;
 set library.old-data-set;
run;
```

Permanent SAS data set and library – next week.

Descriptive Statistics

- Data Description
- Summarize your data
  - PROC MEANS
- Visualize data
  - Histograms, bos plots, scatter plots

Data Description

- PROC CONTENTS

```
proc contents data=data-name options;
run;
```

- The output displays info about the data set, such as number of variables and their types, number of observations, dates of creation, etc.

- PROC PRINT

```
proc print data = data-name;
run;
```

- Used to list the data

PROC MEANS

- Primarily used for reporting sums and means of numeric variables

```
proc means data = data_name;
  ... <statements>;
  run;
```

- Default statistics: N (number of non-missing obs), Mean, StdDev, Min and Max
- Other Options:

QRANGE	MAXDEC = *n*–number of decimal places to be displayed
MODE	MISSING –treats missing values as valid summary groups
RANGE	NMISS –number of missing values
SUM	MEDIAN, Q1 (25thpercentile), Q3 (75thpercentile)

PROC MEANS: VAR statement

- Without VAR statement, it will show the summary statistics for all numeric variables

- You can specify which variables to include in the report using VAR statement

```
proc means data = data_name;
  var var1 var2 va3;
run;
```

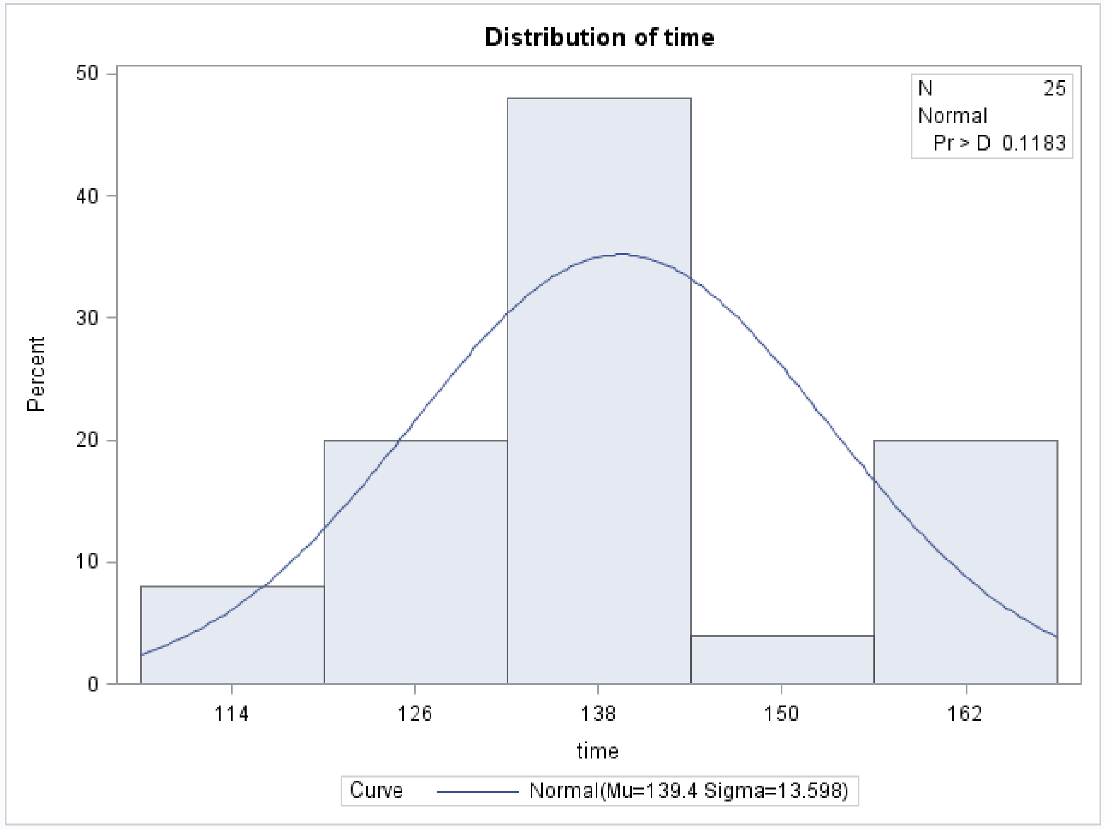
PROC UNIVARIATE

- Also produces simple summary statistics (mean, standard deviation)
- Generates histograms, boxplots, normal probability plots (QQ plots)
- Conducts tests for normality

```
proc univariate data = data_name;
  <statements>;
run;
```

PROC UNIVARIATE - Histogram

- Histogram and normality check



```
proc univariate data=prob2;
histogram time / normal(percents=20 40 60 80);
inset n normal(ksdpval) / pos = ne;
run;
```

Percents option specifies the quantiles of the normal distribution

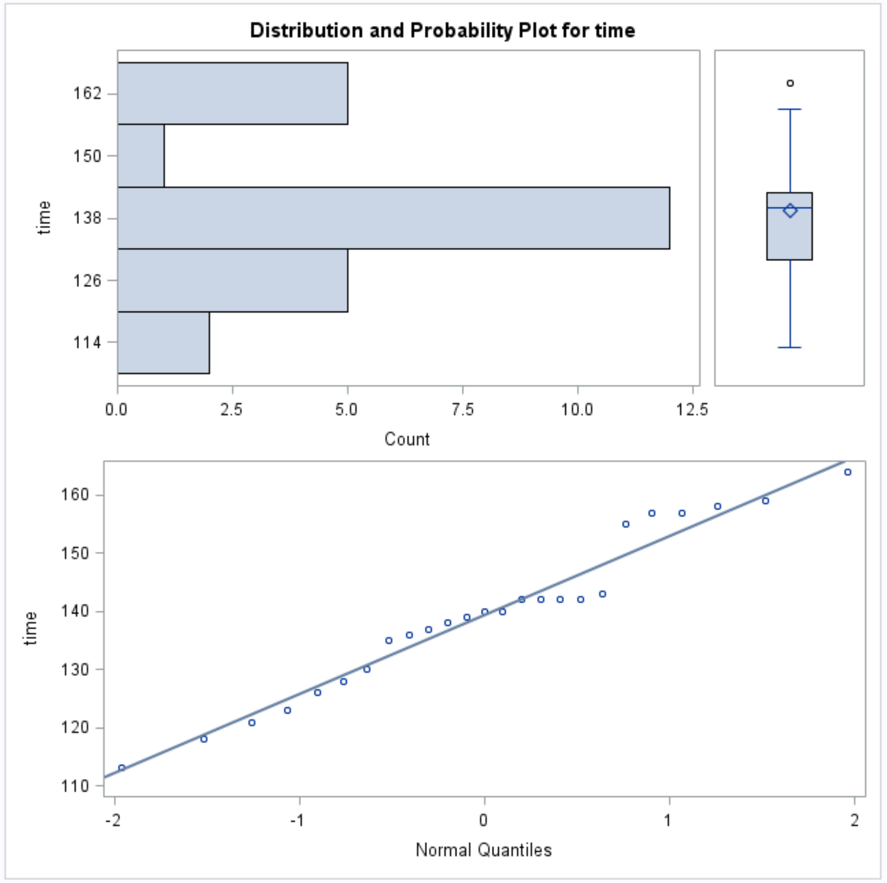
normal(ksdpval) generates the Kolmogorov-Smirnov test of normality;

Pos indicates the location (NE) of where to put this test result

PROC UNIVARIATE - Plots

- The PLOTS option in the PROC UNIVARIATE statement requests several basic summary plots.

```
* PROC UNIVARIATE - boxplot and qqplot;
proc univariate data=prob2 plot;
var time;
run;
```


Distribution and Probability Plot for time
time
Count
time
Normal Quantiles