Two ways to run SAS

- **batch mode**
  - use text editor to write a file containing SAS code
  - extension “.sas”; e.g. filename might by “conadp.sas”
  - run it in batch mode by entering “sas <filename>”; e.g. “sas conadp”
  - SAS produces two new output files
    - log file with .log extension
    - output file with .lst extension

- **interactive mode**
  - just enter “sas”
  - program editor, log, and output are three windows
  - online help available

How SAS programs and commands are organized

- Use a **DATA step** to organize your data by creating a SAS dataset.
- Then use **PROC steps** to analyze your data using SAS procedures.
  - Once you have created a SAS dataset, you may apply any SAS procedures to it during the current SAS session without recreating the dataset.
- **DATA** and **PROC** steps consist of SAS **statements**.
  - Each statement must end with a semicolon.
  - Most statements include one or more **keywords** that must be spelled exactly as shown.
The DATA step: Creating a SAS dataset

Before it can process data, SAS must read in the data in the form of a table with:

- a row for each observation
- a column for each variable

You must choose a name for the entire dataset and a name for each variable.

SAS distinguishes between two types of variables:

- numeric variables, which contain only digits and decimal points and with which arithmetic operations may be done
- character variables (all other kinds of data).

SAS data step

- **data** statement
  - must begin the data step
  - gives name by which to refer to dataset during this SAS run
- **infile** statement
  - tells SAS in what physical file to find data
  - may optionally give SAS information about file and how to read it

Starting SAS in different environments

- **Running SAS in the ITC**
  Click on Start/All Programs/SAS/SAS 9.1 (English). SAS will come up, but it will be running on the university’s “Virtual Desktop,” not on your local computer. That means that SAS will not be able to find files on the local drives. It should be able to find files in your H: drive, however.
  You can access SAS on the university’s “Virtual Desktop” from other Windows computers, including from home. Go to itc.uiowa.edu and select “Virtual Desktop” on the left panel.

- **Running SAS on the Linux network**
  The Division of Mathematical Sciences has a SAS licence for only one of the computers on the Linux network. If you type “sas” at the Linux prompt, you will automatically be logged into that computer (you will be prompted for your password). SAS will come up in interactive mode.
To make Linux SAS work more like Windows SAS, do the following:

- Select the Program Editor window.
- Click "Tools" and "Options," and select "Preferences”
- Click the "Toolbox" tab and DEPRESS the three toggles (Display Tools Window, Display Command Window, Combine Windows)
- Click the "Editing" tab and UNPRESS the toggle "Automatically store selection”
- Click "OK"

Entering commands and programs

However you access SAS, you will get a screen that shows:

- a menu bar
- a log window
- a program editor window

Click in the program editor window. You may now type commands and programs in this window.

How SAS programs and commands are organized

Use a *DATA step* to organize your data by creating a SAS dataset. Then use *PROC steps* or automated features to analyze your data. Once you have created a SAS dataset, you may apply any SAS procedures or automated features to it during the SAS session without recreating the dataset.

DATA and PROC steps consist of SAS *statements*. Each statement must end with a semicolon. Most statements include one or more *keywords* that must be spelled exactly as shown.

The DATA step: Creating a SAS dataset

Before it can process data, SAS must read in the data in the form of a table with

- a row for each *observation*
- a column for each *variable*

You must choose a name for the entire dataset and a name for each variable. SAS has the following rules for names:

- SAS names must begin with a letter or an underscore. The remaining characters in a SAS name can be letters, numbers, or underscores. There must be no embedded blanks.

SAS distinguishes between two types of variables: *numeric variables*, which contain only digits and decimal points and with which arithmetic operations may be done, and *character variables* (all other kinds of data).
Controlling print width

Put this line at the beginning of every SAS program if you want output to print correctly on 8-1/2 by 11 inch paper:

```sas
options linesize = 75 ;
```

Reading data in from an existing datafile

Suppose you have saved the file “billion.dat” in a directory called SASdata on your H: drive. Use an “infile” statement to tell SAS to use it.

```sas
data billion ; * gives dataset a name for SAS
infile 'h:\my documents\SASdata\billion.dat' ; * tells SAS where the data is
input with age region $ ; * names the variables in each row
run ; * end of data step
```

Using `datalines`

Alternatively, you may use a `datalines` statement instead of an `infile` statement, and copy and paste the data directly into the data step. This will avoid any issues with mapping local drives to virtual drives. You must put a semicolon on a line by itself after the last row of data. This would look like:

```sas
data billion ; * gives dataset a name for SAS
input with age region $ ; * names the variables in each row
&$ after region identifies character vbl
run ; * end of data step
```

type these lines into the program editor window. To make SAS run these statements and create the dataset, use the mouse to highlight the block of statements and then click on the icon of the running man. SAS will use the log window to tell you what it has done. Be sure to check the log window for any error messages. If any errors are reported, click in the program editor window to make it active. Correct the errors in the code and then rerun the block of code.
Using SAS procedures to list and tabulate the dataset

Once the dataset is created, you may run SAS procedures to analyze it. To list the entire dataset:

```
proc print data = billion ;
run ;
```

To get a frequency distribution of the regions in which billionaires lived:

```
proc freq data = billion ;
tables region ;
run ;
```

The output is:

```
The FREQ Procedure

<table>
<thead>
<tr>
<th>region</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>38</td>
<td>16.31</td>
<td>38</td>
</tr>
<tr>
<td>E</td>
<td>80</td>
<td>34.33</td>
<td>118</td>
</tr>
<tr>
<td>M</td>
<td>22</td>
<td>9.44</td>
<td>140</td>
</tr>
<tr>
<td>O</td>
<td>29</td>
<td>12.45</td>
<td>169</td>
</tr>
<tr>
<td>U</td>
<td>64</td>
<td>27.47</td>
<td>233</td>
</tr>
</tbody>
</table>
```

Proc univariate: SAS workhorse of descriptive statistics

Use `proc univariate` for quantitative variables when you want the following:

- means
- medians
- quartiles
- 5-number summary
- stem plots (for small datasets) or histograms (large datasets)
- boxplots

```
proc univariate plot data = billion ;
var weight ;
run ;
```

The output is:

```
The UNIVARIATE Procedure

Variable: weight

Moments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>233</td>
</tr>
<tr>
<td>Sum Weights</td>
<td>233</td>
</tr>
<tr>
<td>Mean</td>
<td>2.68</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>3.32</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.65</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>5.69</td>
</tr>
<tr>
<td>UnCorrected SS</td>
<td>420</td>
</tr>
<tr>
<td>Corrected SS</td>
<td>250</td>
</tr>
</tbody>
</table>

Basic Statistical Measures

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Variability</td>
</tr>
<tr>
<td>Mean</td>
<td>2.68</td>
</tr>
<tr>
<td>Median</td>
<td>1.80</td>
</tr>
<tr>
<td>Mode</td>
<td>1.00</td>
</tr>
<tr>
<td>Interquartile Range</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Tests for Location: Mu0=0

<table>
<thead>
<tr>
<th>Test</th>
<th>Statistic</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student's t</td>
<td>12.33</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Sign</td>
<td>118.5</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Signed Rank</td>
<td>13630.5</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

Quantiles (Definition 5)

<table>
<thead>
<tr>
<th>Quantile</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>3.0</td>
</tr>
<tr>
<td>25%</td>
<td>1.3</td>
</tr>
<tr>
<td>50%</td>
<td>1.8</td>
</tr>
<tr>
<td>75%</td>
<td>3.0</td>
</tr>
<tr>
<td>90%</td>
<td>4.5</td>
</tr>
<tr>
<td>99%</td>
<td>14.0</td>
</tr>
</tbody>
</table>

Extreme Observations

<table>
<thead>
<tr>
<th>Value</th>
<th>Obs</th>
<th>Value</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>233</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>232</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>231</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>230</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>229</td>
<td>37</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Histogram # Boxplot

* may represent up to 3 counts