UCF STA4102 Lecture 2

Alexander V. Mantzaris

Department of Statistics

UNIVERSITY OF CENTRAL FLORIDA

Overview

1. an interesting read
2. advice
3. Running SAS
4. Running our first SAS programs
5. General SAS
6. DATA STEP example
7. data ex

PROBABILITY, STATISTICS AND TRUTH R.V.Mises (prior BIG DATA)

-'All our philosophy is a correction of the common usage of words', Lichtenberg. Many of the quarrels and mistakes occurring in the course of scientific advance could be avoided if this remark were always remembered. Our first step, therefore, will be to inquire more closely into the meaning of the word 'probability'... gradually leading to an adequate scientific definition of the concept of probability, where the key to the relation between statistics and truth may be found in a reasonable definition of probability.

The word 'probable' is commonly used; 'it will probably rain tommorrow'. We speak of something being more or less probable. We can explain what we mean by these statements if the person asking is satisfied by a 'descriptive' answer.

Paul Dirac 1902 (Bristol UK) - 1984 (Tallahassee FL)

situation he created from Florida

Paul Dirac while on vacation in Florida, mailed anonymously to George Gamow in Washington DC a baby alligator in a box. When it was opened, it jumped out and bit the hand. Gamow thought it was a symbol of his favourite experimenter. The alligator languished for a few months till death in the Gamow hotel bath.

current section

1. an interesting read
2. advice
3. Running SAS
4. Running our first SAS programs
5. General SAS
6. DATA STEP example
7. data ex

STACK EXCHANGE

ESSENTIAL is a STACKEXCHANGE network account.

stack overflow and *cross validated* are indespensible.

problems with being stuck

happens all the time

If you get bored of something, read books that are related fiction. Eg sci-fi, history of science, biographies of famous scientists (they do have fascinating lives!)

As I have been told at the start of my degree, if stuck for a long time don't stare at the screen; try something different or talk to someone.

current section

1 an interesting read

2 advice

3 Running SAS

4 Running our first SAS programs

5 General SAS

6 DATA STEP example

7 data ex

Starting SAS! *my.apps.ucf.edu*

I am interested now in trying SAS, how should I go about doing this?

There are a couple of ways to do this as a UCF student. There is the University SAS edition which you can download and install. It requires you to run VMware and from that environment work with SAS. I expect that you might encounter problems with this approach.

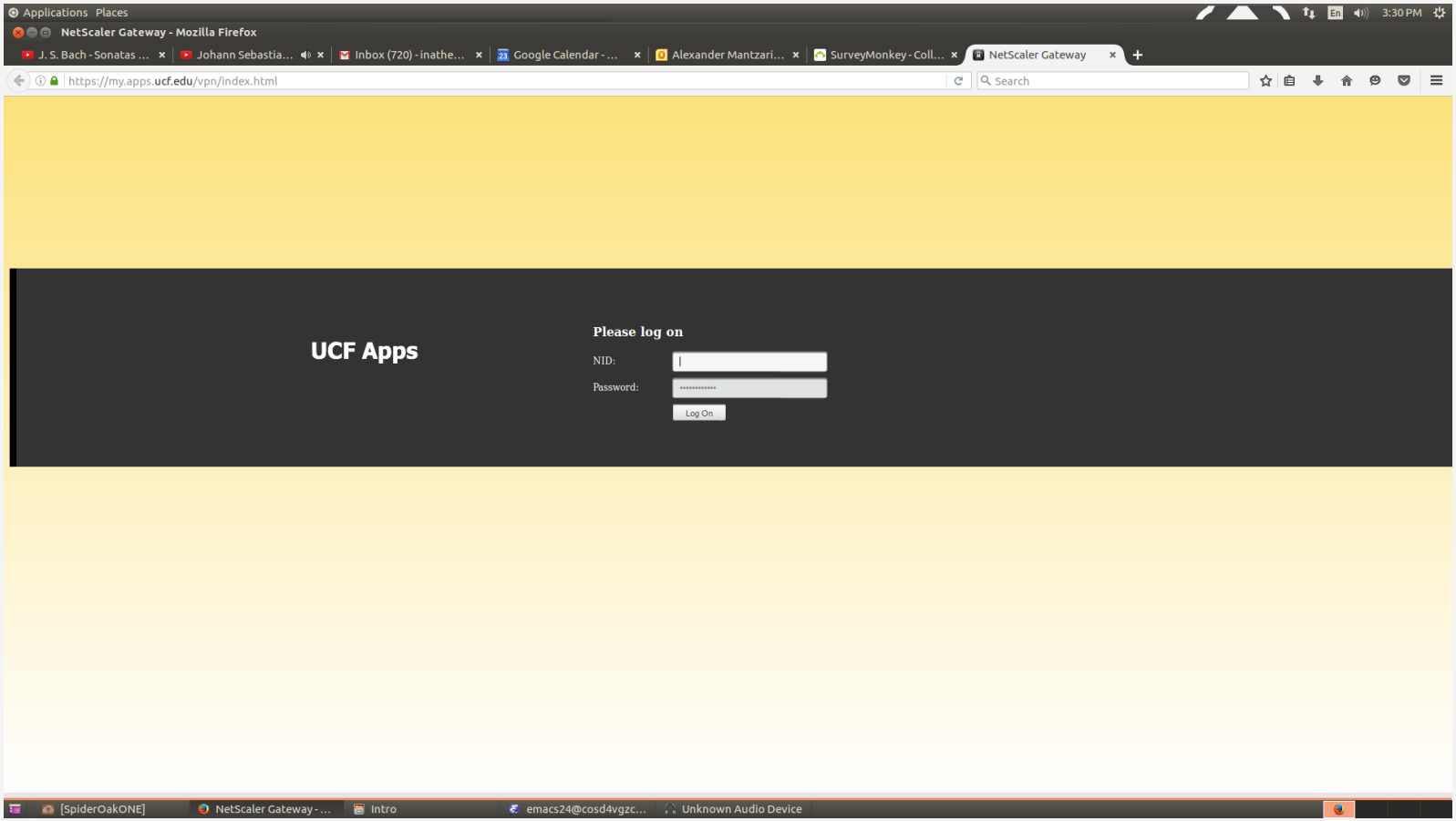
Well, I don't know what VMware is

If you haven't, it is not too challenging and the computer support can assist you through challenges, but I would recommend you use the web interface the university has for SAS:

my.apps.ucf.edu

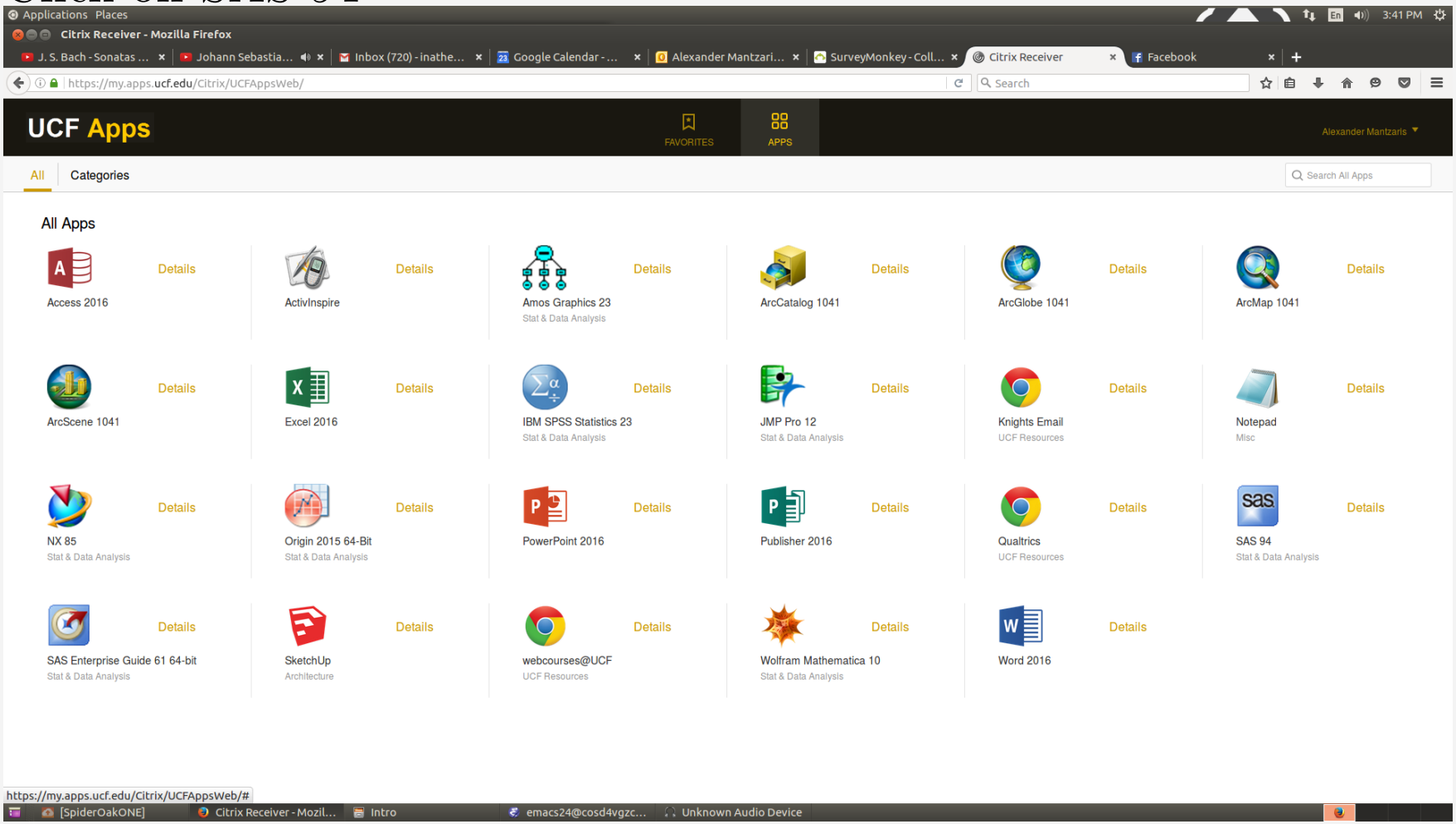
Starting SAS! *my.apps.ucf.edu*

In your browser go to: my.apps.ucf.edu



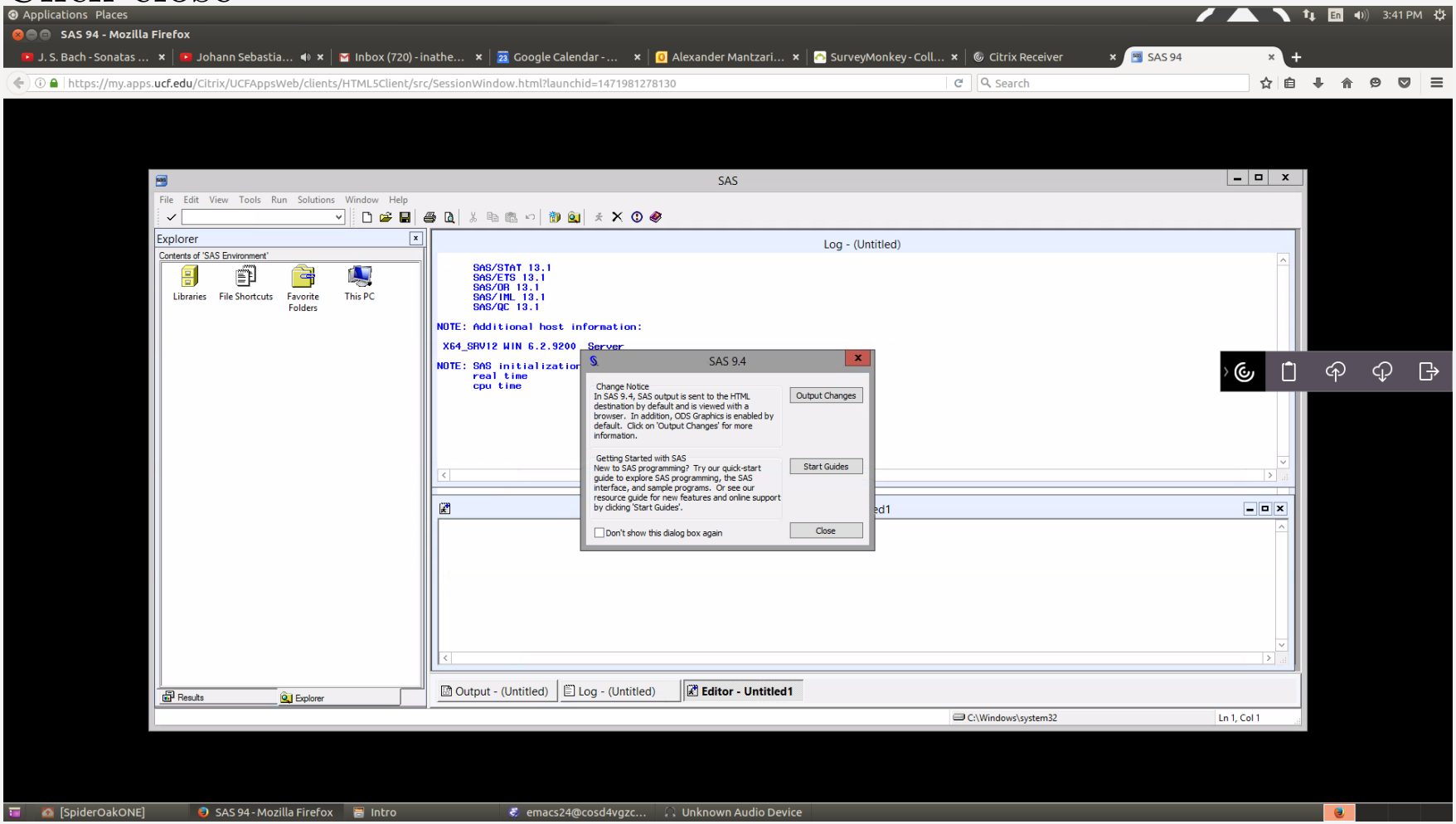
Starting SAS! *my.apps.ucf.edu*

Click on SAS 94



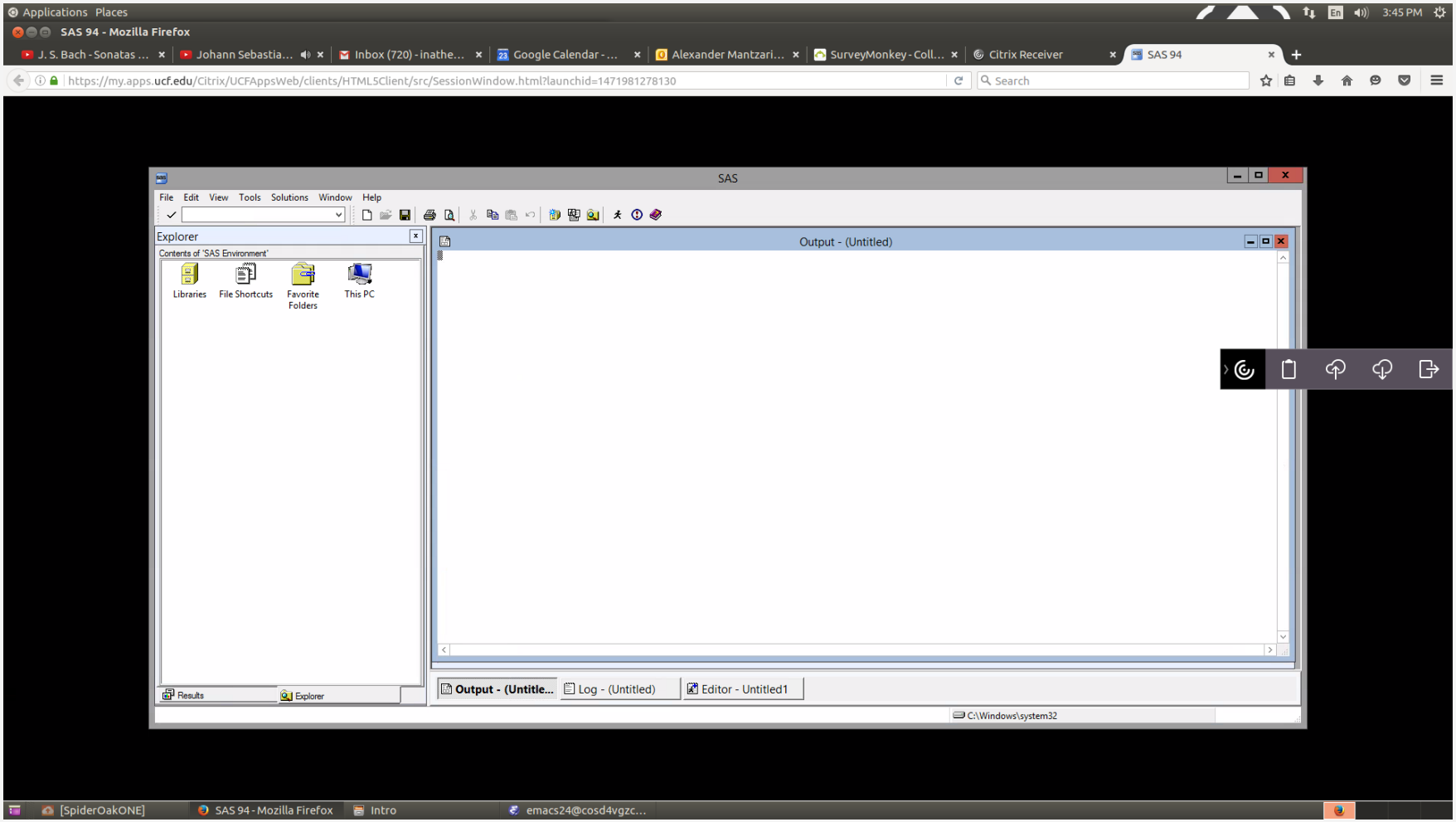
Starting SAS! *my.apps.ucf.edu*

Click close

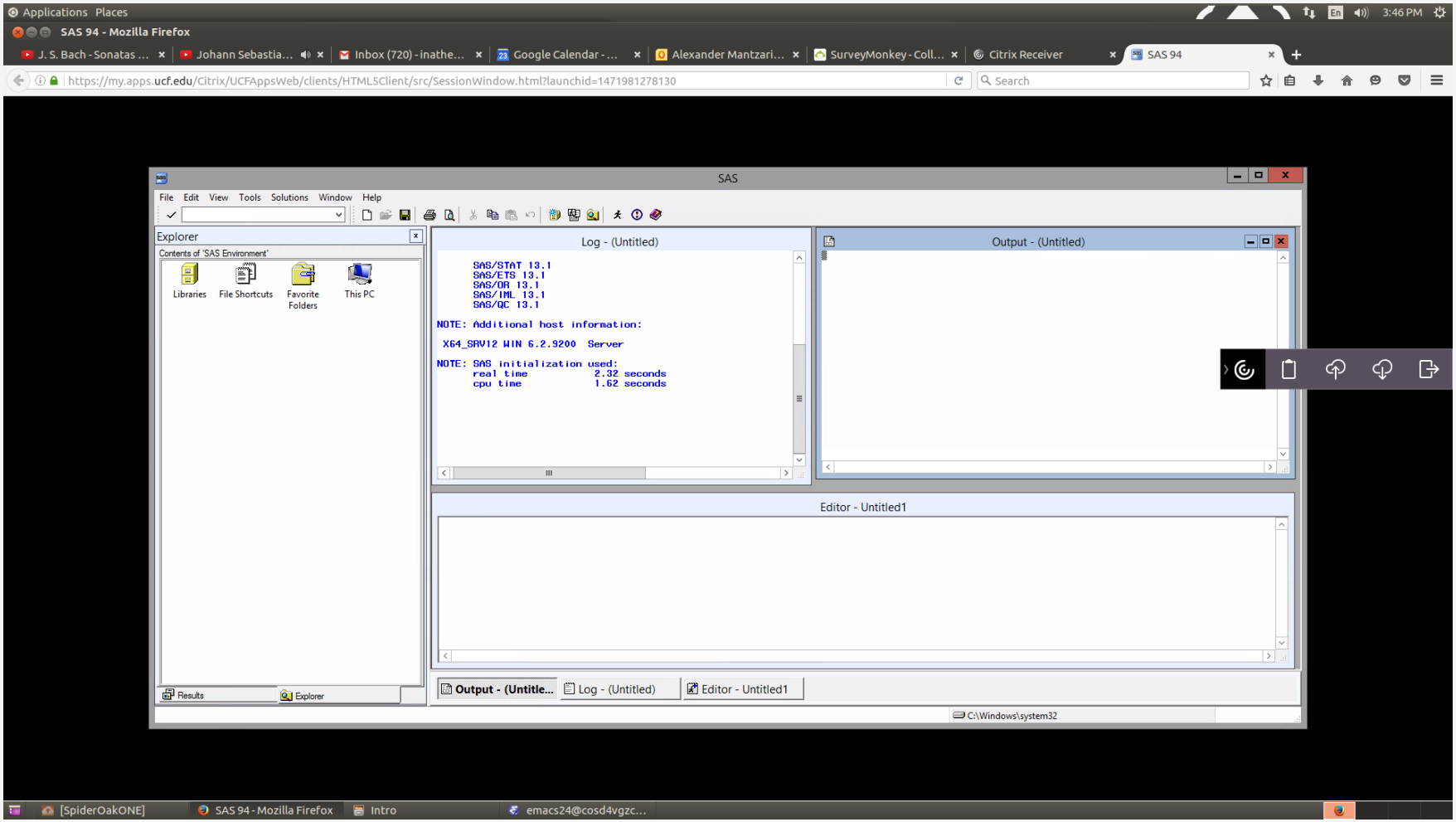


Starting SAS! *my.apps.ucf.edu*

Resize the windows to fit the 'Output window/log window/Editor window' inside



Starting SAS! *my.apps.ucf.edu*



This is the basic view

What am I looking at? What are all these window panes for?
Do not get confused by complexity. That is a common pitfall. The strategy to handle complex things thrown at you is to focus on your immediate goal and build on it.
What is our immediate goal?
To become familiar with the DATA-PROC steps.
I've heard that this interface allows us to do a wide range of tasks without any programming necessary. Can we avoid using commands and stick to the tools available from the point and click?
We could, but end up not utilising the full extent of SAS's capability. It is best to get through the tedious parts first.

current section

1 an interesting read

2 advice

3 Running SAS

4 Running our first SAS programs

5 General SAS

6 DATA STEP example

7 data ex

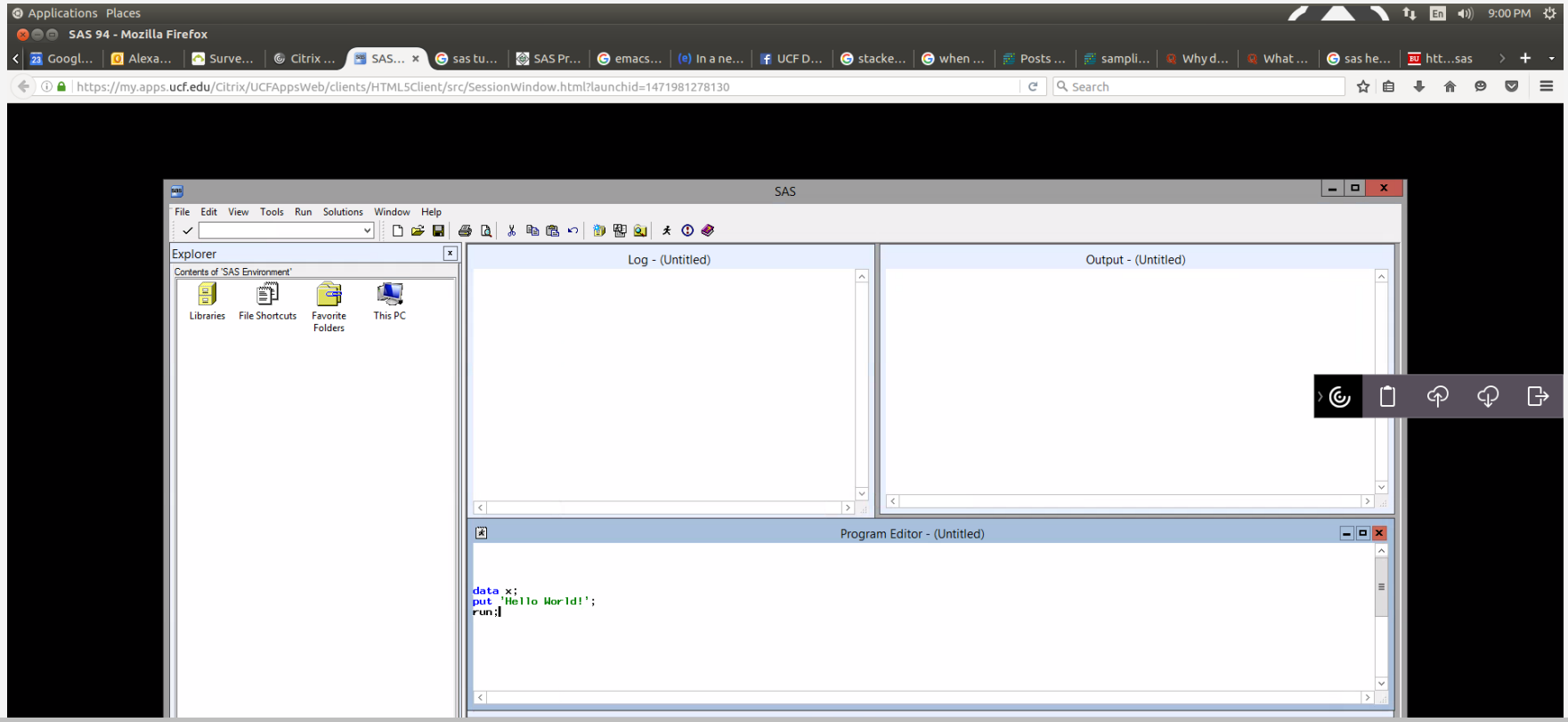
The age long duty of printing out *Hello World!*

really? Yes.

```
1: data x;
2: put 'Hello World!';
3: run;
```

The age long duty of printing out *Hello World!*

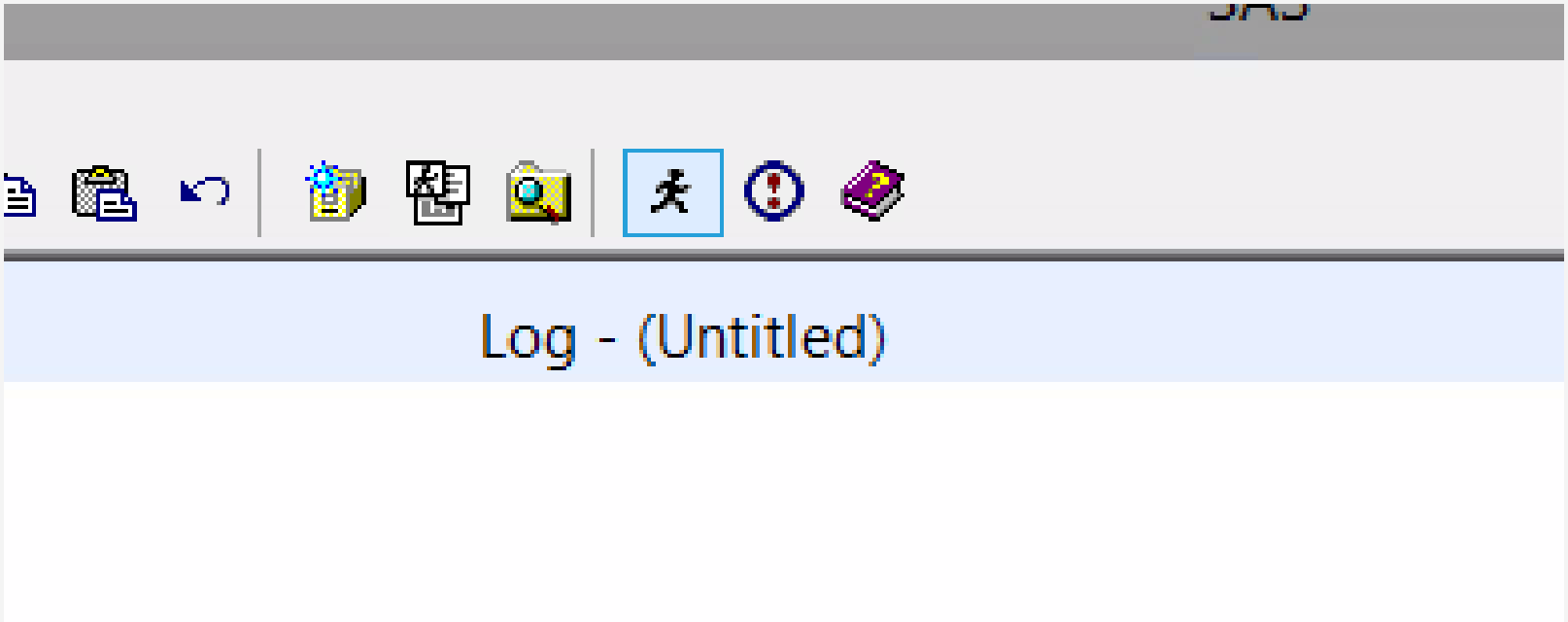
```
1: data x;
2: put 'Hello World!';
3: run;
```



The age long duty of printing out *Hello World!*

```
1: data x;
2: put 'Hello World!';
3: run;
```

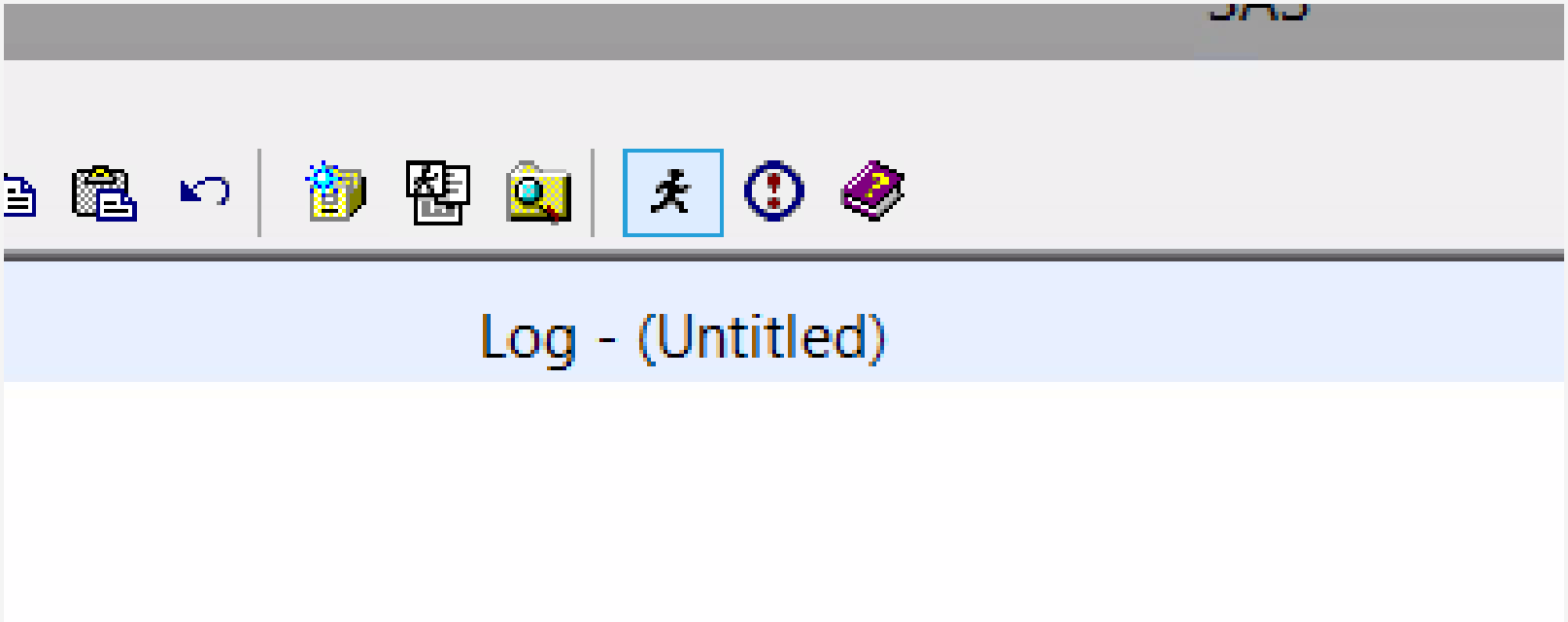
Now push the button on the Toolbar of a person running to execute the script (*Submit* button)



The age long duty of printing out *Hello World!*

```
1: data x;
2: put 'Hello World!';
3: run;
```

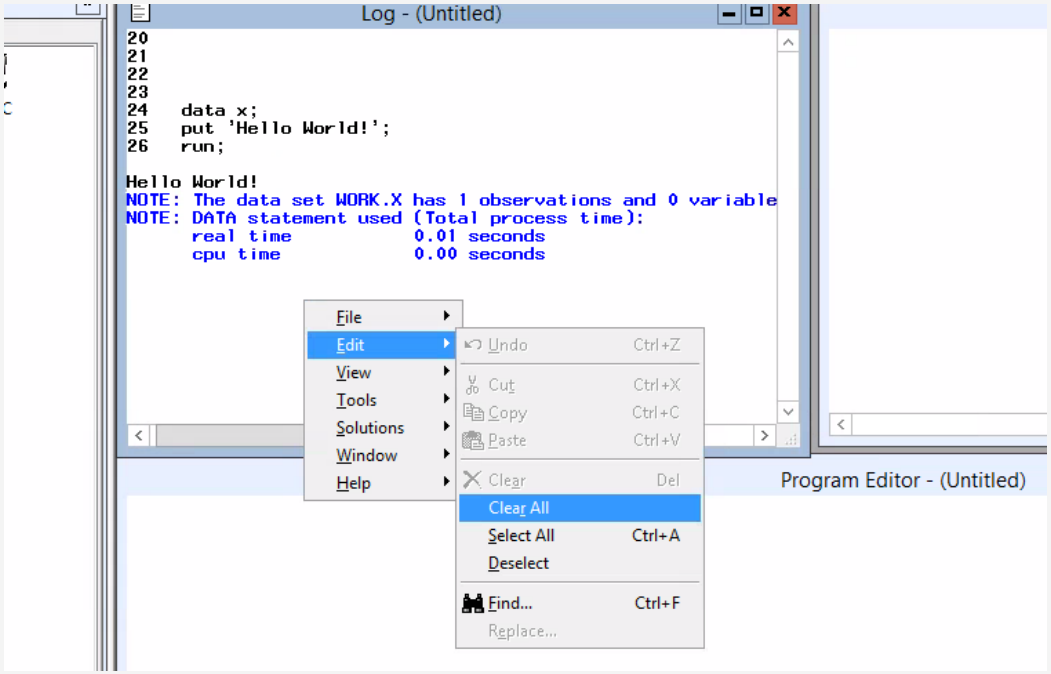
Now push the button on the Toolbar of a person running to execute the script (*Submit* button)



The age long duty of printing out *Hello World!*

```
1: data x;
2: put 'Hello World!';
3: run;
```

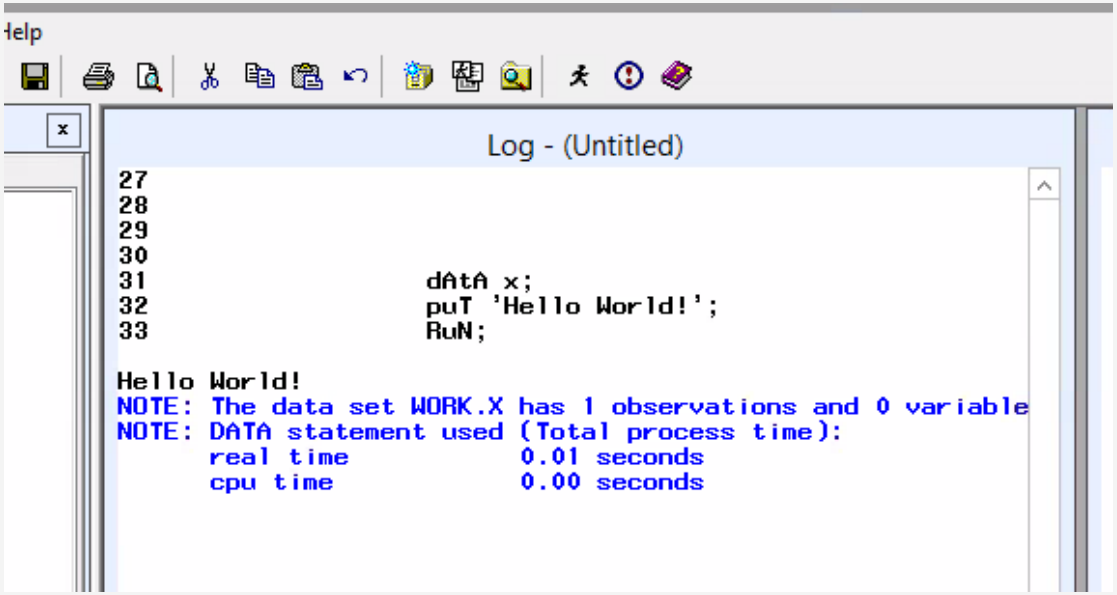
Clearing the Log screen:



The age long duty of printing out *Hello World!*

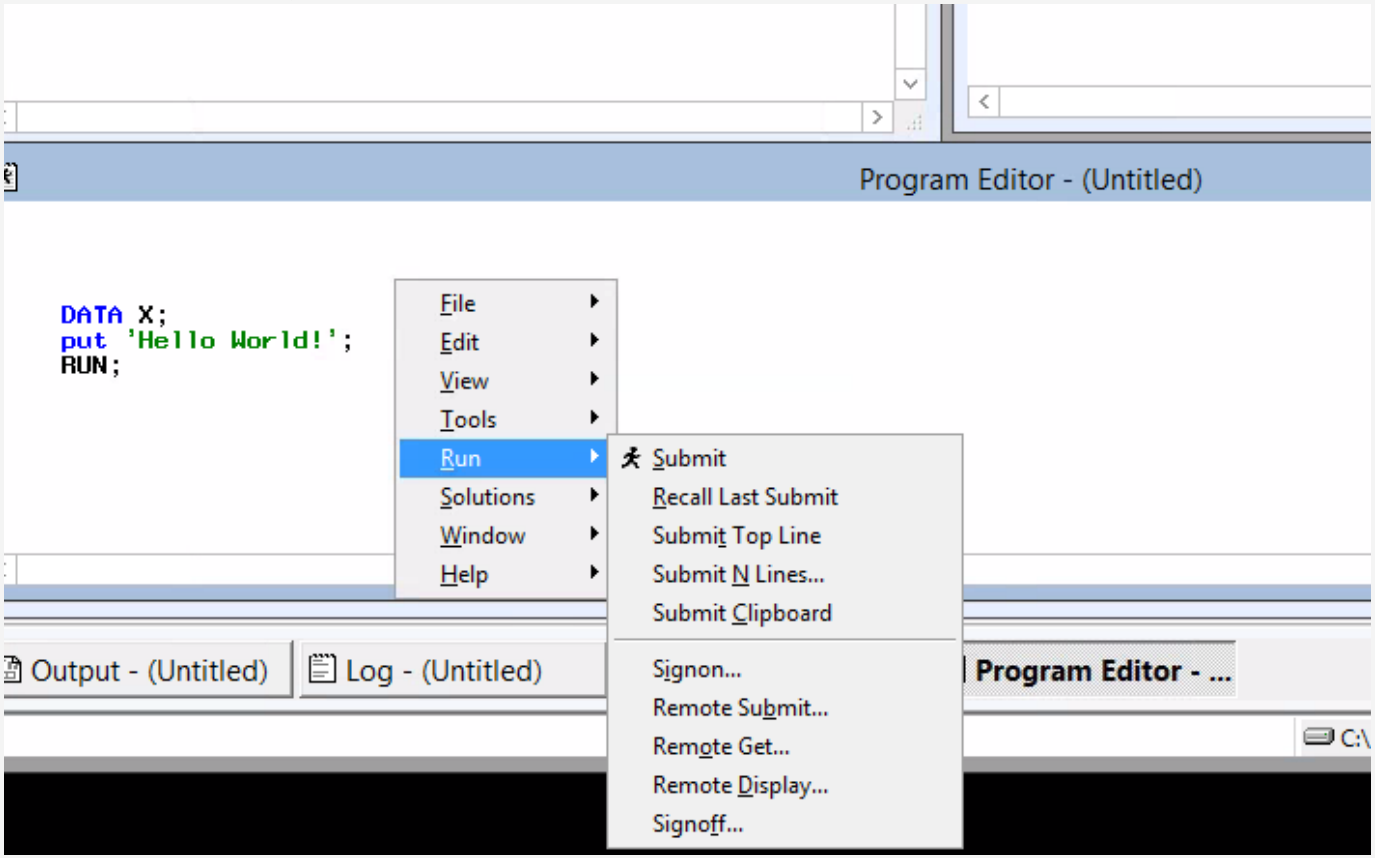
```
1: dAtA x;
2: puT 'Hello World!';
3: RuN;
```

Changing the case from lower to upper does not change results:



The age long duty of printing out *Hello World!*

You can right click and choose the option to run the commands:



current section

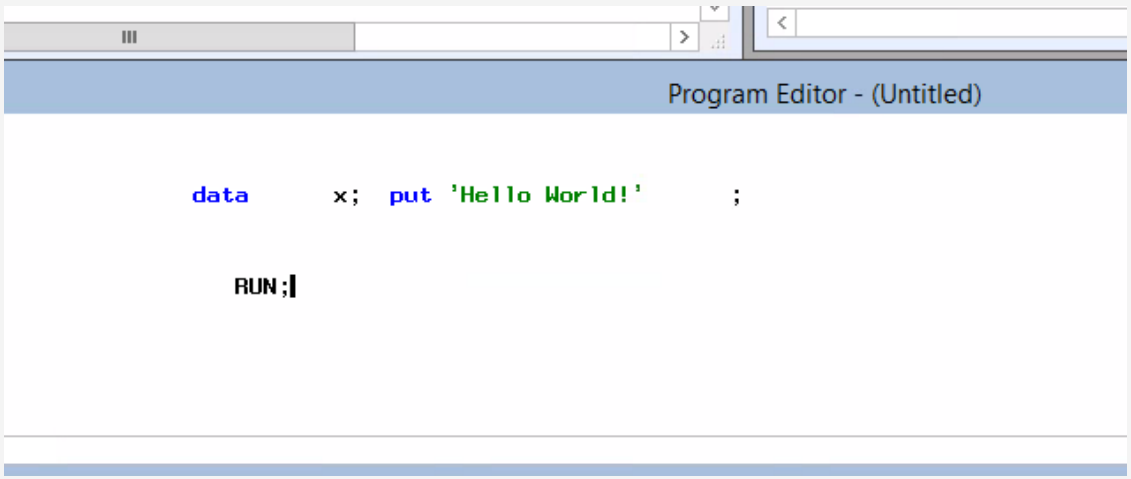
1. an interesting read
2. advice
3. Running SAS
4. Running our first SAS programs
5. General SAS
6. DATA STEP example
7. data ex

SAS Statements

- SAS statements can be chained onto the same line if they have a semi-colon to separate them.
- Statements require no particular alignment in the editor. The interpreter will assemble a correct statement form as long as the end of the line of commands has a semicolon, (;).
- Spaces can be included as you wish.

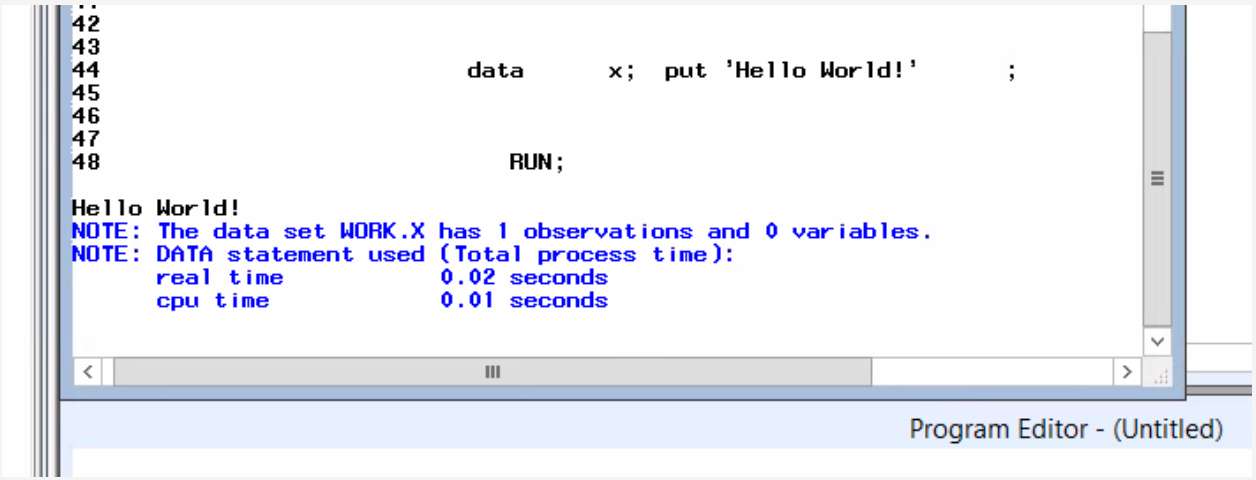
SAS Statements

- SAS statements can be chained onto the same line if they have a semi-colon to separate them.
- Statements require no particular alignment in the editor. The interpreter will assemble a correct statement form as long as the end of the line of commands has a semicolon, (;).
- Spaces can be included as you wish.



SAS Statements

- SAS statements can be chained onto the same line if they have a semi-colon to separate them.
- Statements require no particular alignment in the editor. The interpreter will assemble a correct statement form as long as the end of the line of commands has a semicolon, (;).
- Spaces can be included as you wish.



SAS variable names

Some rules about the names.

- Maximum 32 characters long. (Sounds like something you would never do but names like *dataSetRetrialAugust23of2003goodSample* is not uncommon for you to want to remember the contents' contexts)
- No blanks. (variable 'TEMP' cannot be written as 'TE MP').
- Variables can start with a letter or underscore(_).
- Numbers can be included in the name but not at the start.

current section

1 an interesting read

2 advice

3 Running SAS

4 Running our first SAS programs

5 General SAS

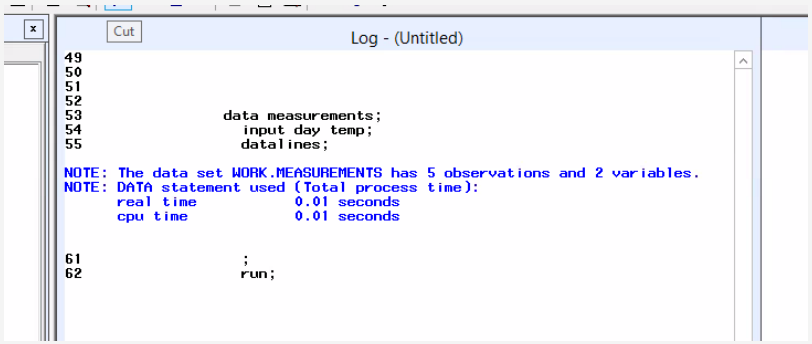
6 DATA STEP example

7 data ex

Another simple example

```
 1: data measurements;
 2: input day temp;
 3: datalines;
 4: 1 50
 5: 2 52
 6: 3 49
 7: 4 48
 8: 8 55
 9: ;
10: run;
```

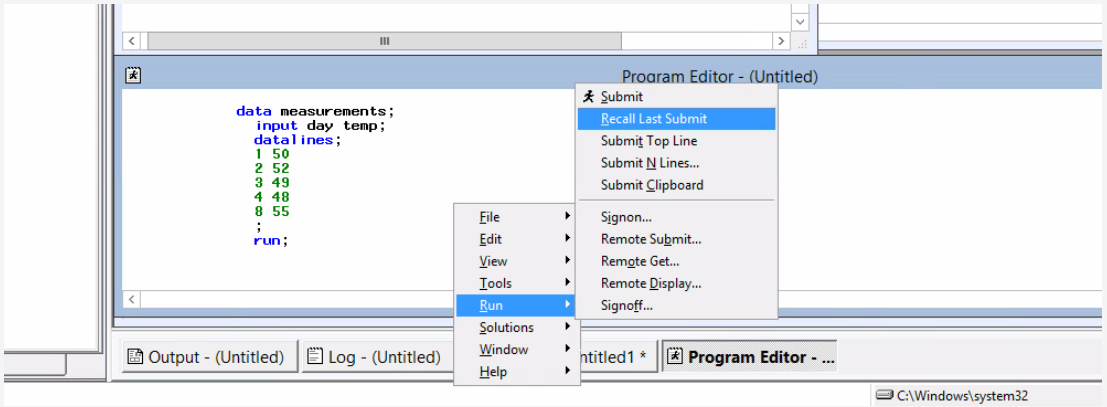
```
 1: data measurements;
 2: input day temp;
 3: datalines;
 4: 1 50
 5: 2 52
 6: 3 49
 7: 4 48
 8: 8 55
 9: ;
10: run;
```



Recall Last Submit

(Very important to not waste time!) Because the submit button sends things to the interpreter and clears the Program Editor pane on default options, you might end up retyping things, which is a waste.

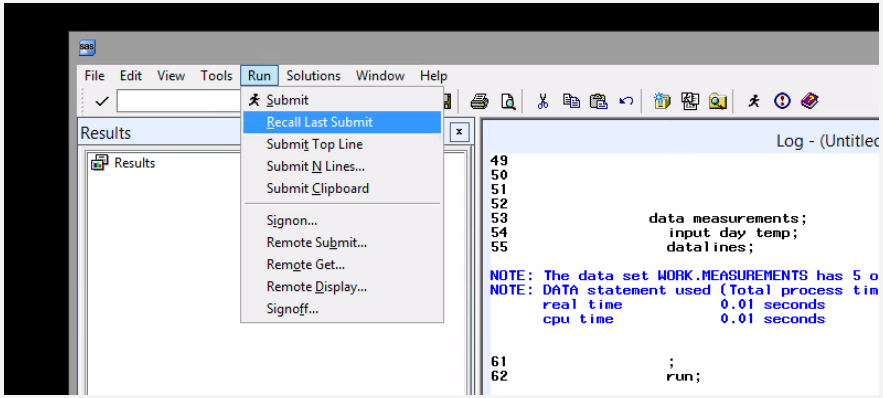
•You can right click and select from Run-Recall Last Submit to bring back the commands you had.



Recall Last Submit

(Very important to not waste time!) Because the submit button sends things to the interpreter and clears the Program Editor pane on default options, you might end up retyping things, which is a waste.

- Use the drop down menues to access the Recall of the previous command set.

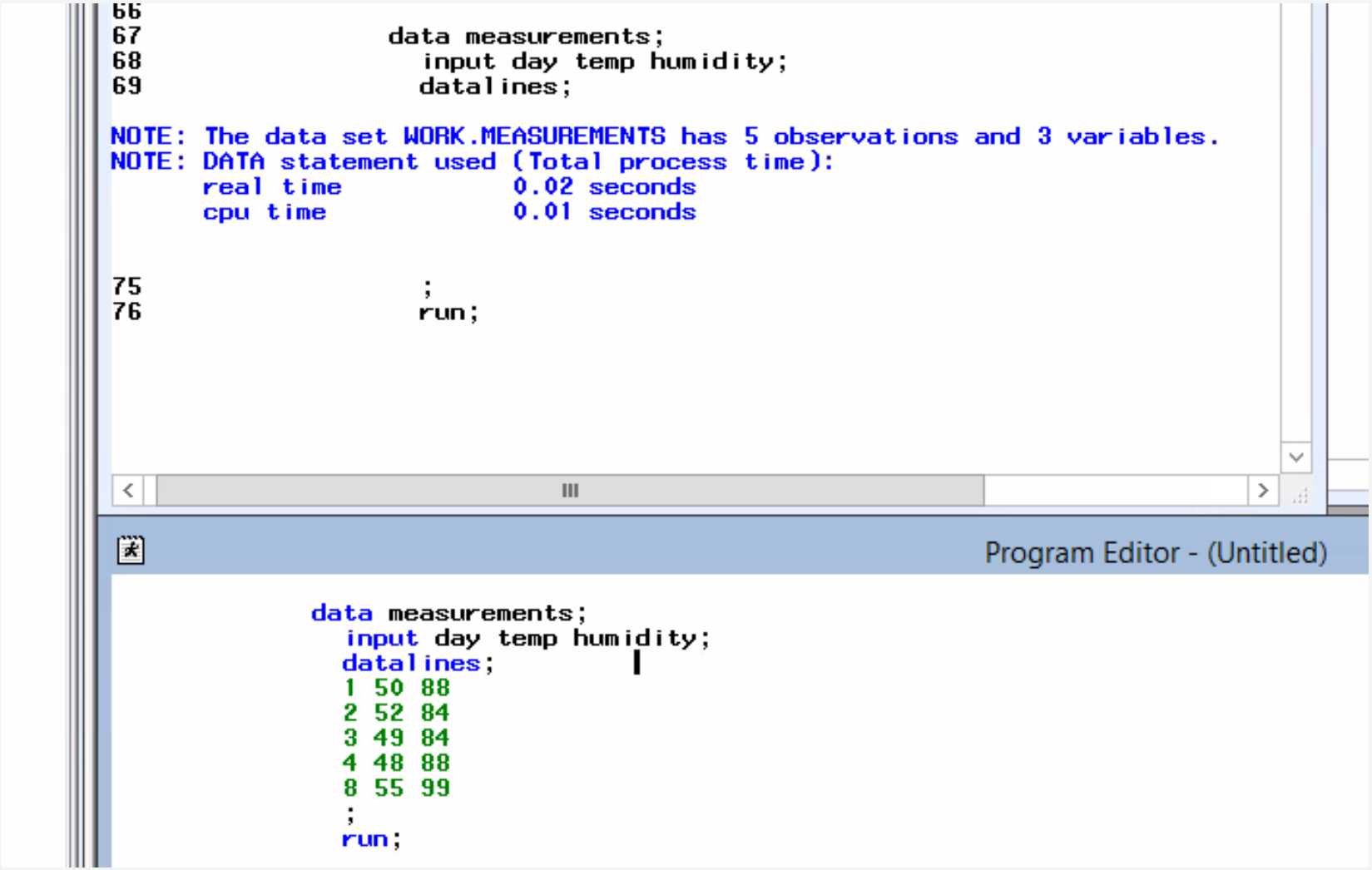


Another simple example

We can recall the previous submission and then add manually another variable of humidity.

```
 1: data measurements;
 2: input day temp humidity;
 3: datalines;
 4: 1 50 88
 5: 2 52 84
 6: 3 49 84
 7: 4 48 88
 8: 8 55 99
 9: ;
10: run;
```

Another simple example



current section

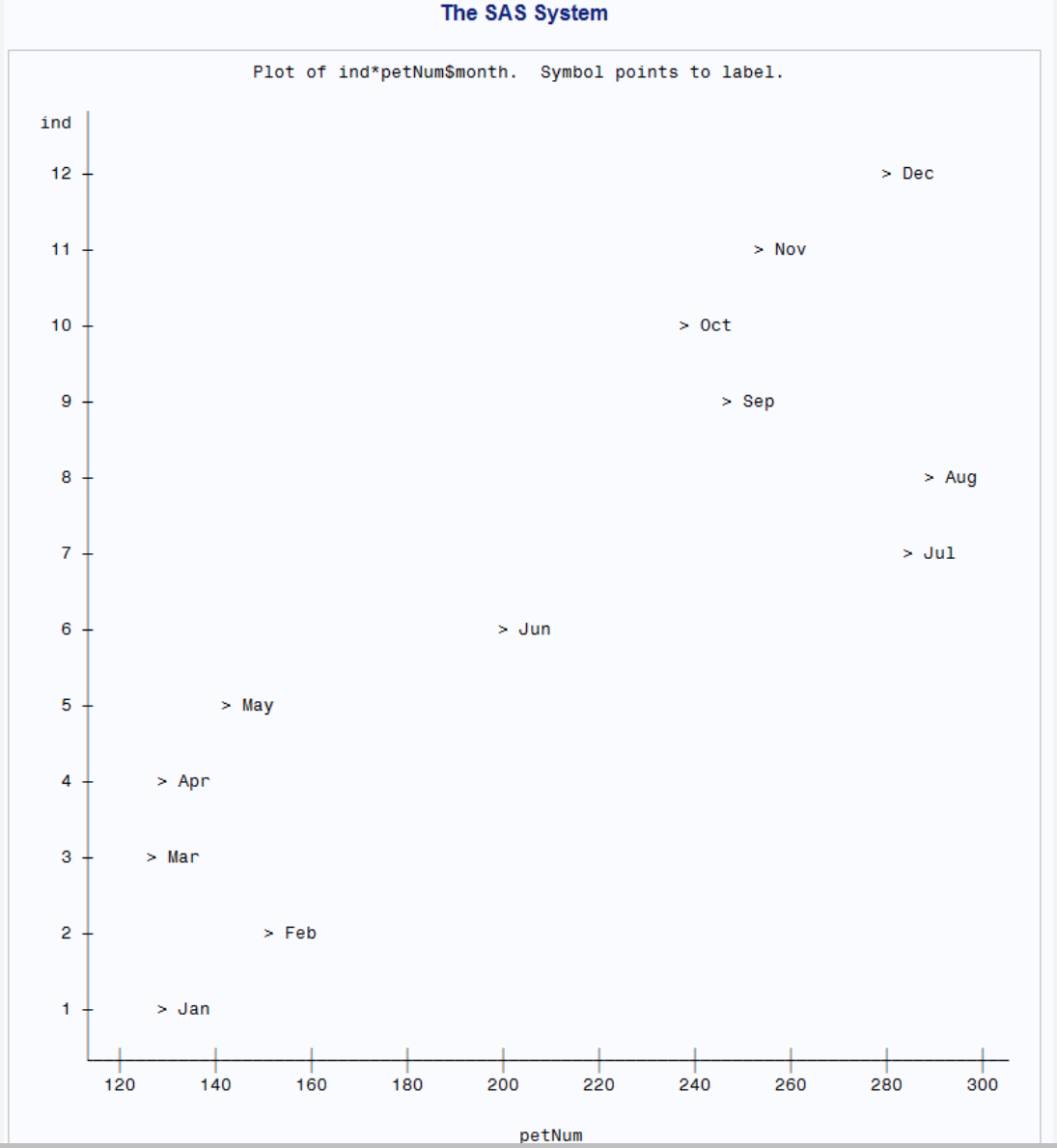
1. an interesting read
2. advice
3. Running SAS
4. Running our first SAS programs
5. General SAS
6. DATA STEP example
7. data ex

King County, Washington PetData 2016 Pet Adoptions

```
data one;
input ind month $ petNum;
datalines;
1 Jan 129
2 Feb 151
3 Mar 126
4 Apr 128
5 May 143
6 Jun 200
7 Jul 285
8 Aug 288
9 Sep 247
10 Oct 238
11 Nov 253
12 Dec 279
;
run;
```

```
proc plot data = one;
plot ind*petNum $month;
by month; run;
```

output of petplot, *conclusions?*

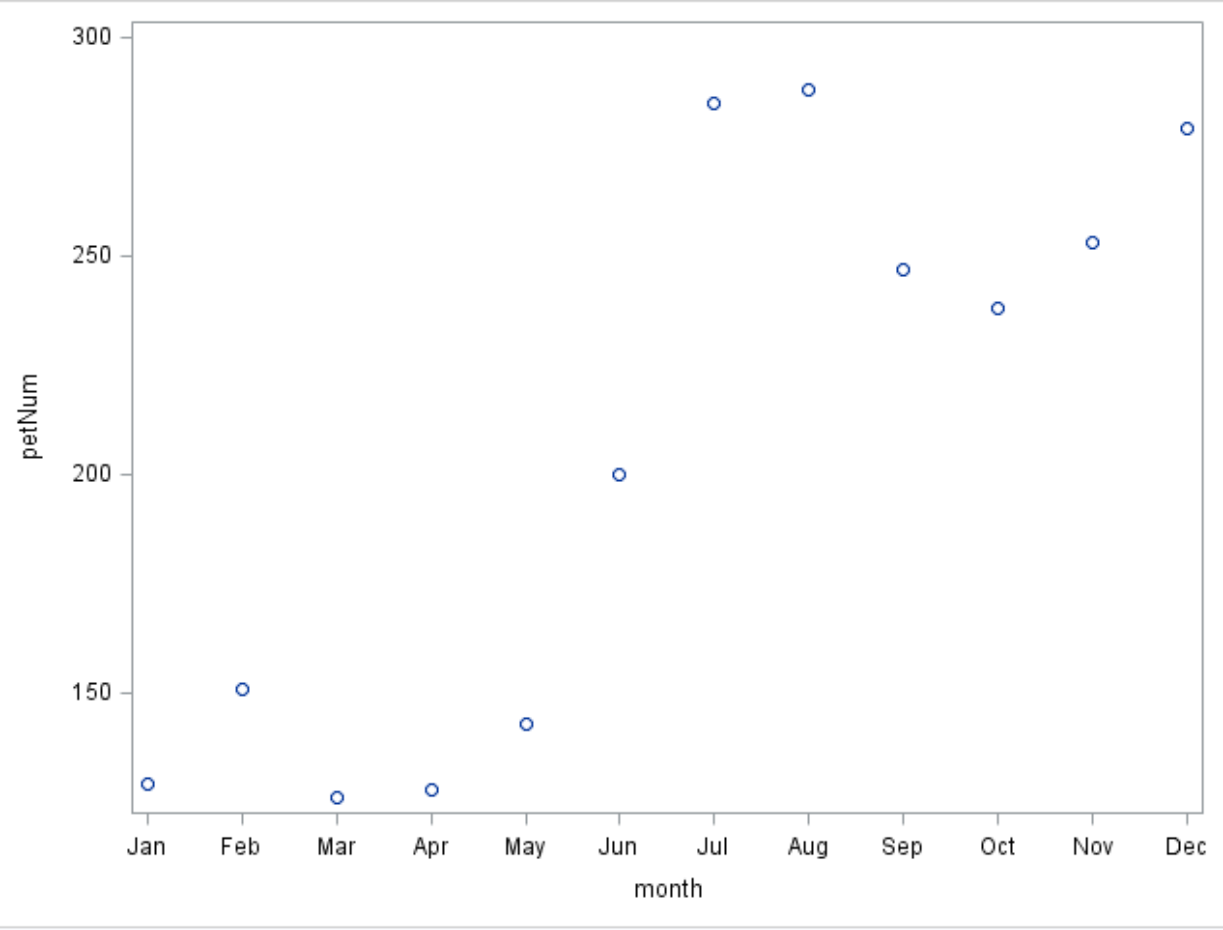


King County, Washington PetData 2016 Pet Adoptions

```
data one;
input ind month $ petNum;
datalines;
1 Jan 129
2 Feb 151
3 Mar 126
4 Apr 128
5 May 143
6 Jun 200
7 Jul 285
8 Aug 288
9 Sep 247
10 Oct 238
11 Nov 253
12 Dec 279
;
run;
```

```
proc sgplot data = one;
scatter x = month y = petNum;
run;
```

output of petplot 'sgplot', *different conclusions?*

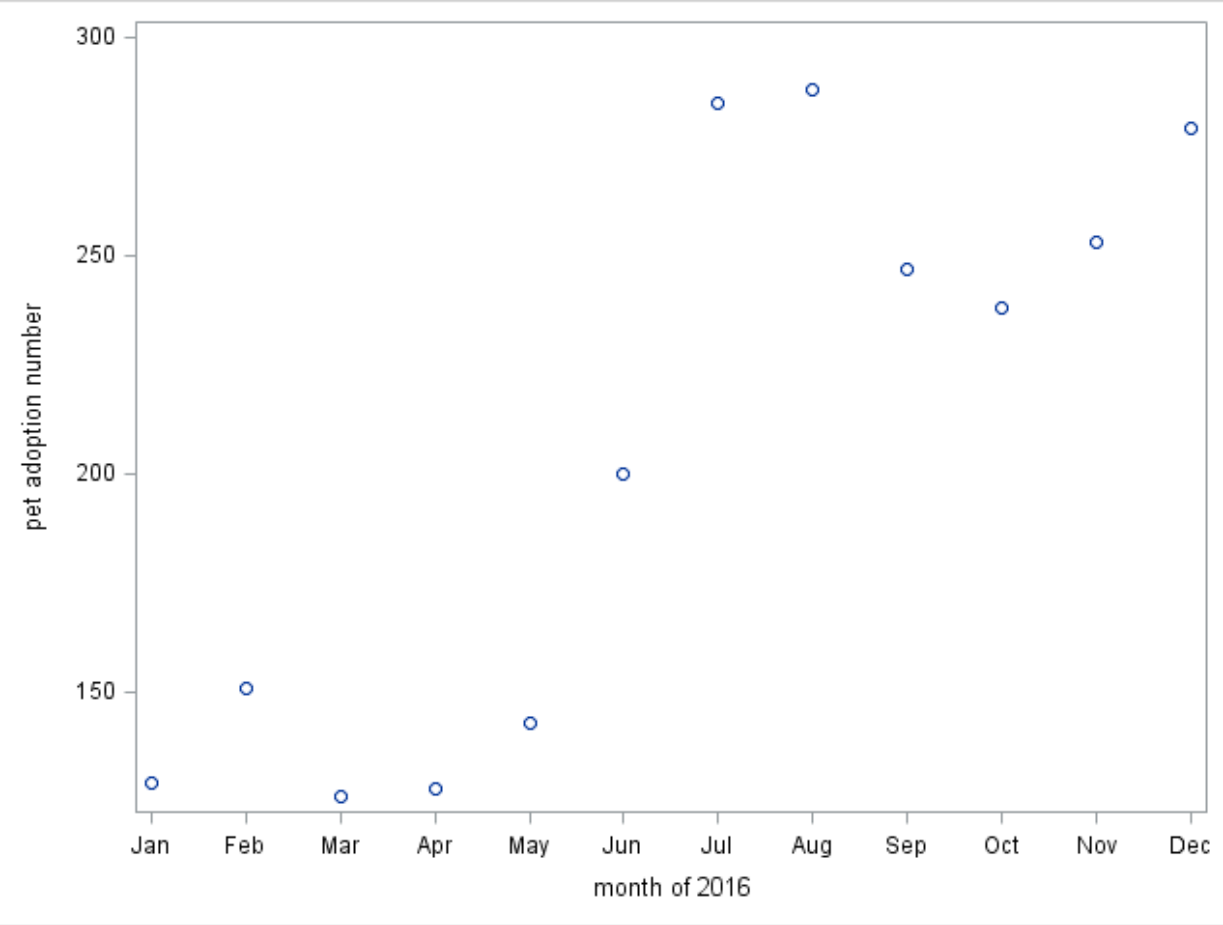


King County, Washington PetData 2016 Pet Adoptions

```
data one;
input ind month $ petNum;
datalines;
1 Jan 129
2 Feb 151
3 Mar 126
4 Apr 128
5 May 143
6 Jun 200
7 Jul 285
8 Aug 288
9 Sep 247
10 Oct 238
11 Nov 253
12 Dec 279
;
run;
```

```
proc sgplot data = one;
xaxis label = "month of 2016";
yaxis label = "pet adoption
number";
scatter x = month y = petNum;
run;
```

output of petplot 'sgplot', *with better labels*



Longley economic data: US dept of labor March1963

```
data econ;
infile datalines delimiter=',';
input Year $ GNPdeflator GNP
Unemployed ArmedForces
Population Year Employed;
datalines;
"1947",83,234.289,235.6,159,107.608,1947,60.323
"1948",88.5,259.426,232.5,145.6,108.632,1948,61.122
"1949",88.2,258.054,368.2,161.6,109.773,1949,60.171
"1950",89.5,284.599,335.1,165,110.929,1950,61.187
"1951",96.2,328.975,209.9,309.9,112.075,1951,63.221
"1952",98.1,346.999,193.2,359.4,113.27,1952,63.639
"1953",99,365.385,187,354.7,115.094,1953,64.989
"1954",100,363.112,357.8,335,116.219,1954,63.761
"1955",101.2,397.469,290.4,304.8,117.388,1955,66.019
"1956",104.6,419.18,282.2,285.7,118.734,1956,67.857
"1957",108.4,442.769,293.6,279.8,120.445,1957,68.169
"1958",110.8,444.546,468.1,263.7,121.95,1958,66.513
"1959",112.6,482.704,381.3,255.2,123.366,1959,68.655
"1960",114.2,502.601,393.1,251.4,125.368,1960,69.564
"1961",115.7,518.173,480.6,257.2,127.852,1961,69.331
"1962",116.9,554.894,400.7,282.7,130.081,1962,70.551

;run;
```

```
proc print data = econ;
run;
```

Obs	Year	GNPdeflator	GNP	Unemployed	ArmedForces	Population	Employed
1	1947	83.0	234.289	235.6	159.0	107.608	60.323
2	1948	88.5	259.426	232.5	145.6	108.632	61.122
3	1949	88.2	258.054	368.2	161.6	109.773	60.171
4	1950	89.5	284.599	335.1	165.0	110.929	61.187
5	1951	96.2	328.975	209.9	309.9	112.075	63.221
6	1952	98.1	346.999	193.2	359.4	113.270	63.639
7	1953	99.0	365.385	187.0	354.7	115.094	64.989
8	1954	100.0	363.112	357.8	335.0	116.219	63.761
9	1955	101.2	397.469	290.4	304.8	117.388	66.019
10	1956	104.6	419.180	282.2	285.7	118.734	67.857
11	1957	108.4	442.769	293.6	279.8	120.445	68.169
12	1958	110.8	444.546	468.1	263.7	121.950	66.513
13	1959	112.6	482.704	381.3	255.2	123.366	68.655
14	1960	114.2	502.601	393.1	251.4	125.368	69.564
15	1961	115.7	518.173	480.6	257.2	127.852	69.331
16	1962	116.9	554.894	400.7	282.7	130.081	70.551

Longley economic data: US dept of labor March1963

```
proc means data=econ;
run;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
GNPdeflator	16	101.6812500	10.7915534	83.0000000	116.9000000
GNP	16	387.6984375	99.3949378	234.2890000	554.8940000
Unemployed	16	319.3312500	93.4464247	187.0000000	480.6000000
ArmedForces	16	260.6687500	69.5919604	145.6000000	359.4000000
Population	16	117.4240000	6.9561016	107.6080000	130.0810000
Employed	16	65.3170000	3.5119684	60.1710000	70.5510000

Longley economic data: US dept of labor March1963

```
proc plot data=econ;
plot Year*Employed $Year;
run;
```