Introduction to SAS Language

Instructor

Economist , TERI University

https://welcomedata.wordpress.com/

https://www.linkedin.com/in/sunakshibhatia

What is SAS?

- **SAS** (Statistical Analysis System
- Software suite developed by SAS Institute for advanced analytics, business intelligence, data management, and predictive analytics
- Developed at North Carolina State University from 1966 until 1976, when SAS Institute was incorporated.
- Further developed in the 1980s and 1990s with the additional statistical procedures and components

Components of SAS

Currently, SAS has more than 200 components, some of them are –

- Base SAS – Basic procedures and data management
- SAS/STAT – Statistical analysis
- SAS/GRAPH – Graphics and presentation
- SAS/OR – Operations research
- SAS/ETS – Econometrics and Time Series Analysis
- SAS/IML – Interactive matrix language

SAS University Edition

http://www.sas.com/en_in/software/university-edition/download-software.html

How to download SAS University Edition?

1. Visit the link http://www.sas.com/en_in/software/university-edition/download-software.html

How to Download SAS® University Edition

Step 1: Verify that you have a compatible virtualization software package.

Because SAS University Edition is a virtual application (or vApp), you need virtualization software to run it. Compatible virtualization software packages are listed in the table below. If you don't already have compatible virtualization software, get the free Oracle VM VirtualBox (select release **4.3.12***).

Windows	Oracle VM VirtualBox 4.3.12*	VMware Player 6.0 or later
OS X	Oracle VM VirtualBox 4.3.12*	VMware Fusion for OS X 6.0
Linux	Oracle VM VirtualBox 4.3.12*	VMware Player for Linux 6.0 or later

* Due to known compatibility issues between SAS University Edition and the most recent releases of Oracle VM VirtualBox, you must install **release 4.3.12**. We are investigating and will update this page as soon as the issues are resolved. In the meantime, do not accept any updates suggested for VirtualBox. Thanks for your patience, and enjoy SAS University Edition!

2. Because SAS University Edition is a virtual application , one needs a virtualization software to run it. So, download Oracle VM VirtualBox or VMware Player and install it on your pc/laptop.
3. Now, click on SAS® University Edition for (respective) Virtual application

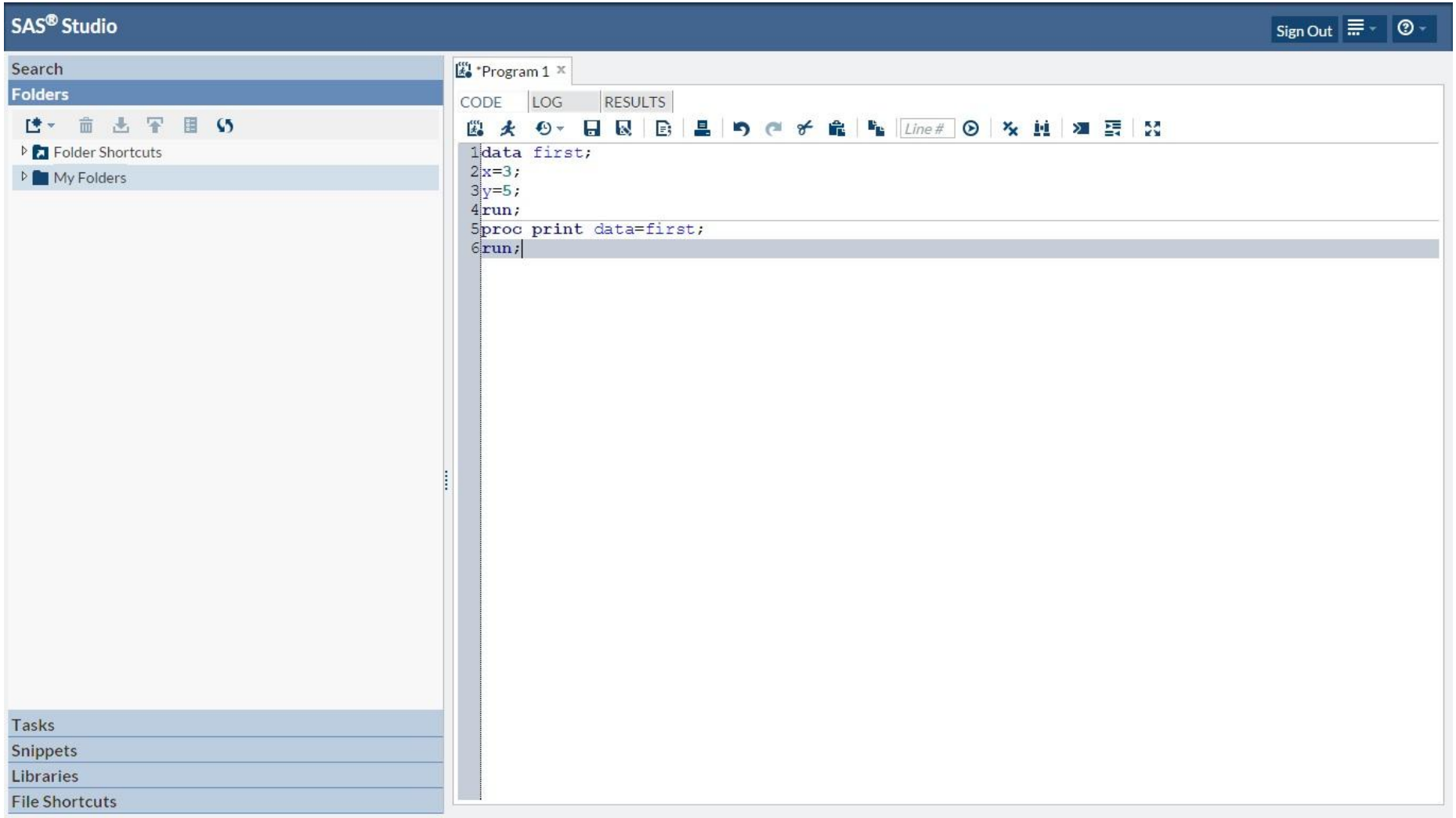
SAS® University Edition for VirtualBox

https://welcomedata.wordpress.com/2015/

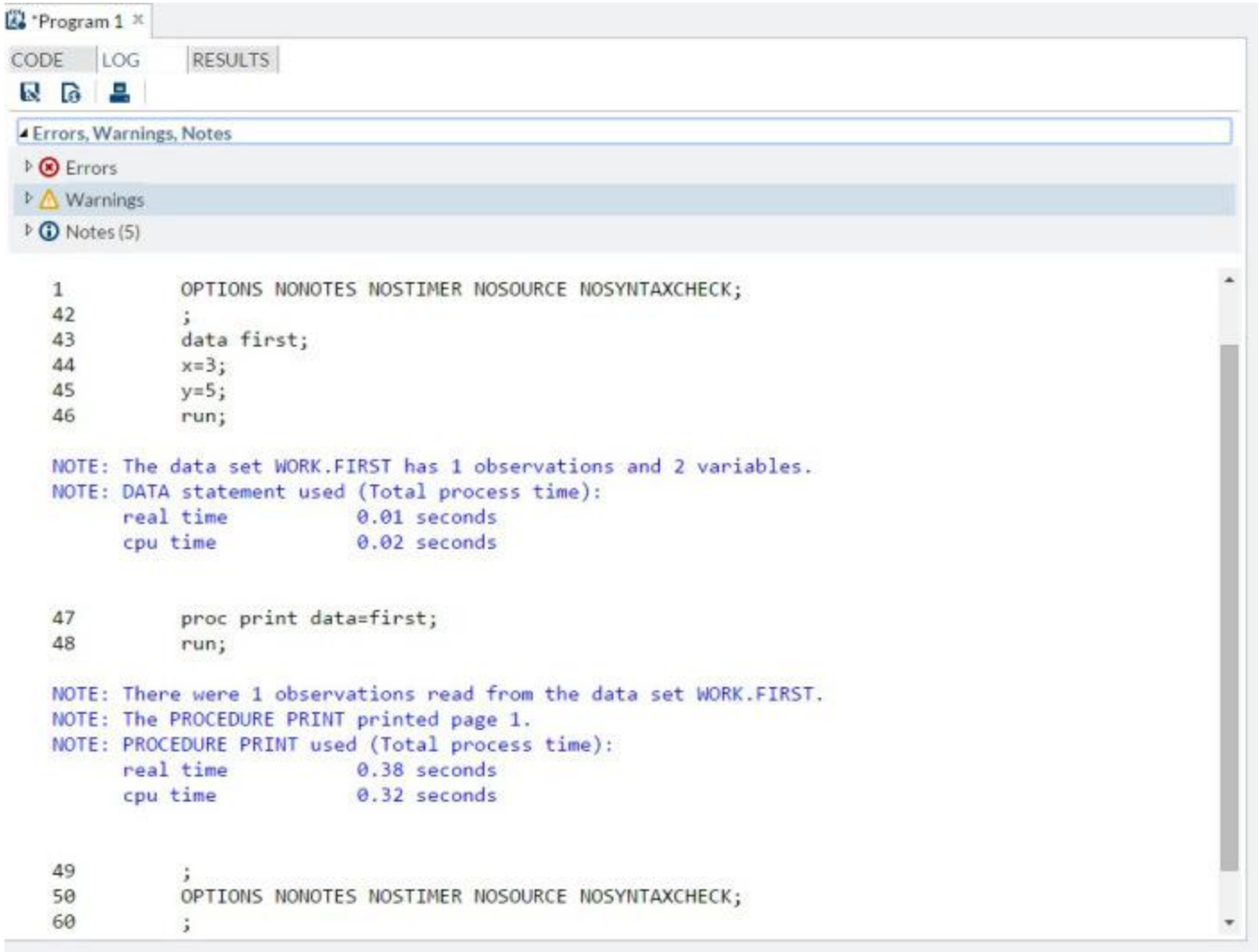
SAS Interface

- Code Editor where we write and modify mode
- Log is a record of everything that we do in SAS session or SAS program :-
  - Program statements identified by line numbers
  - Messages that begin with NOTE, INFO, WARNING, ERROR, or an error number
  - Process time
- Result displays printed output

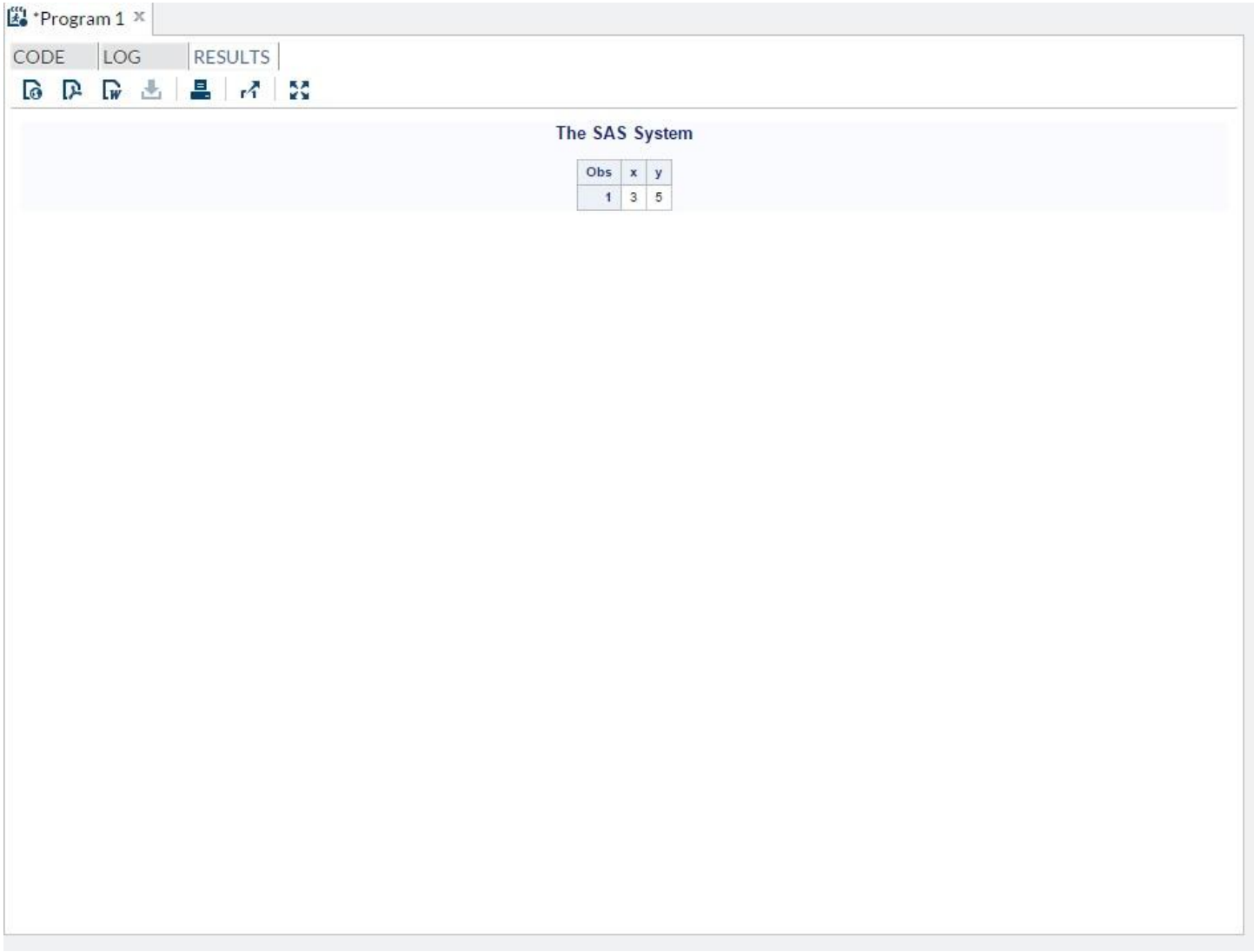
SAS Code Editor



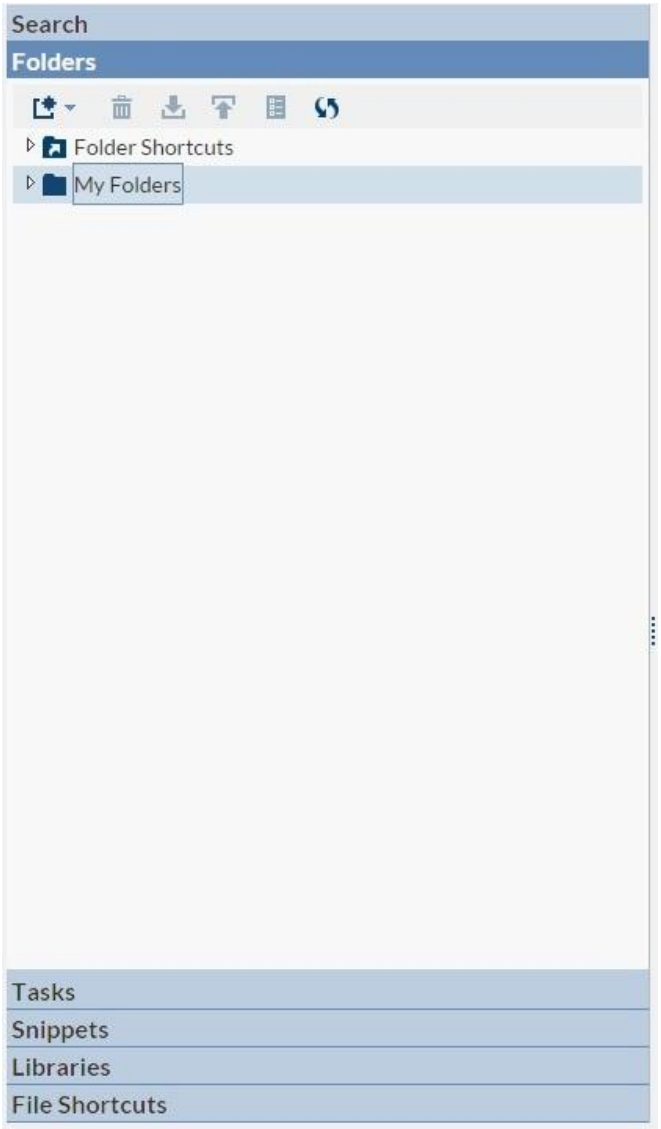
SAS Log


*Program 1
CODE
LOG
RESULTS
Errors, Warnings, Notes
Errors
Warnings
Notes (5)
1 OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
42 ;
43 data first;
44 x=3;
45 y=5;
46 run;
NOTE: The data set WORK.FIRST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
real time 0.01 seconds
cpu time 0.02 seconds
47 proc print data=first;
48 run;
NOTE: There were 1 observations read from the data set WORK.FIRST.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time):
real time 0.38 seconds
cpu time 0.32 seconds
49 ;
50 OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
60 ;

SAS Result



SAS Explorer



- All libraries, folders , files that include data, saved code and saved output can be found here

SAS Language

- SAS program consists of SAS Statements
- Each SAS Statement end with a semi-colon (;)
- Basic SAS Statements are :
  - Data Step
  - Proc Step

DATA Step – used to create or modify data sets

PROC Step (Procedure) - pre-written rules that analyze and process data in a SAS dataset and then produce a report

SAS Language

- A SAS program can consist of a DATA step or a PROC step or any combination of DATA and PROC steps.
- Data Step and Proc Step are followed by Run Statement.
- SAS statements are free-format –
  - they can begin and end anywhere on a line
  - one statement can continue over several lines
- To Comment use (/*Comment*/)

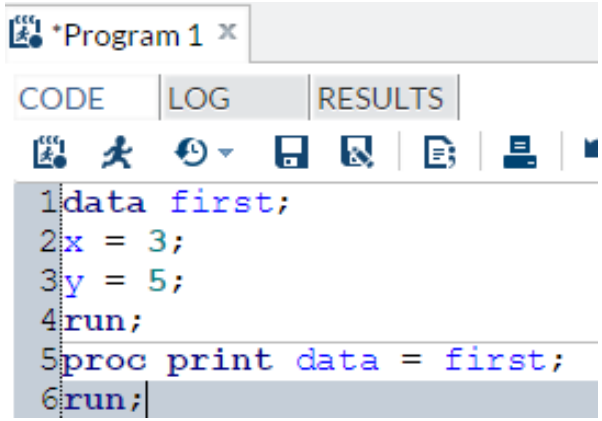
Class Assignment

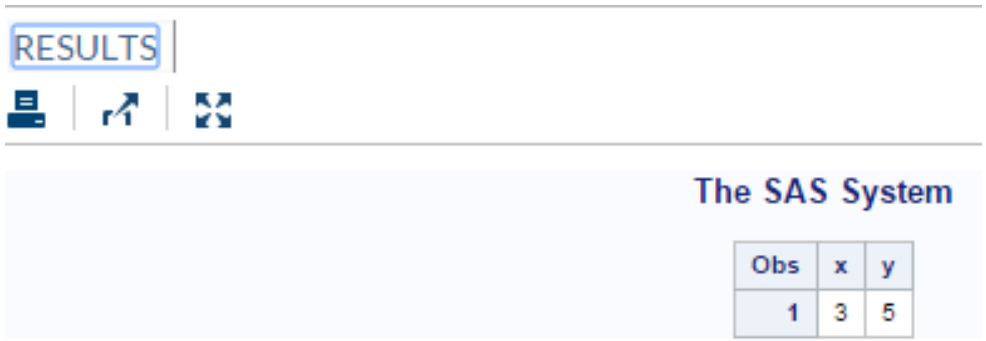
Create a cheat sheet for yourself of the code that you run in the class.

Add short comments for each program or each new step.

Submit the cheat sheet at the end of the class.

Create Temporary Data set





i. There are six SAS statements.

ii. DATA step – creates a new dataset called *first*.

iii. PROC step – prints/displays that dataset in the results/output window.

PROC Step

- Proc PRINT
- Proc DATASETS
- Proc CONTENTS
- Proc SORT
- Proc FREQ
- Proc MEANS
- Proc UNIVARIATE
- Proc CORR

Create Temporary Data set

```
 7 data second;
 8 input m n;
 9 datalines;
10 6 2.13
11 3 4.25
12 1 6.37
13 9 5.67
14 4 7.12
15 2 2.84
16 ;
17 run;
18 proc print data = second;
19 run;
```

The SAS System

Obs	m	n
1	6	2.13
2	3	4.25
3	1	6.37
4	9	5.67
5	4	7.12
6	2	2.84

- Input Statement –
  - Input variable names,
  - Describes arrangement of values in the input data record and assigns
- Datalines/Cards Statement indicates that existence of data lines below

In the above program:

i. There are seven SAS statements;

ii. The INPUT statement defines the variables to be read in each line of data.

iii. The DATALINES statement indicates to SAS that DATA step statements are completed and the next line contains real data.

iv. Notice that the lines of data do not end in a semicolon.

Create Temporary Data set

```
20 data third;
21 input m n o;
22 datalines;
23 6 2.13 apple
24 3 4.25 banana
25 1 6.37 oregano
26 9 5.67 pepper
27 4 7.12 mango
28 2 2.84 orange
29 ;
30 run;
31 proc print data = third;
32 run;
```

The SAS System

Obs	m	n	o
1	6	2.13	.
2	3	4.25	.
3	1	6.37	.
4	9	5.67	.
5	4	7.12	.
6	2	2.84	.

Missing values for Character variable *o?*

Let us check Log for errors

```
NOTE: Invalid data for o in line 46 8-12.
RULE:      ----+----1----+----2----+----3-
46         6 2.13 apple
m=6 n=2.13 o=. _ERROR_=1 _N_=1
NOTE: Invalid data for o in line 47 8-13.
47         3 4.25 banana
m=3 n=4.25 o=. _ERROR_=1 _N_=2
NOTE: Invalid data for o in line 48 8-14.
48         1 6.37 oregano
m=1 n=6.37 o=. _ERROR_=1 _N_=3
```

Create Temporary Data set

For character variable add “$” after variable name in Input statement

```
20 data third;
21 input m n o$;
22 datalines;
23 6 2.13 apple
24 3 4.25 banana
25 1 6.37 oregano
26 9 5.67 pepper
27 4 7.12 mango
28 2 2.84 orange
29 ;
30 run;
31 proc print data = third;
32 run;
```

The SAS System

Obs	m	n	o
1	6	2.13	apple
2	3	4.25	banana
3	1	6.37	oregano
4	9	5.67	pepper
5	4	7.12	mango
6	2	2.84	orange

SAS Options

```
20 data third;
21 input m n o$;
22 datalines;
23 6 2.13 apple
24 3 4.25 banana
25 1 6.37 oregano
26 9 5.67 pepper
27 4 7.12 mango
28 2 2.84 orange
29 ;
30 run;
31 proc print data = third (obs = 4);
32 run;
```

The SAS System

Obs	m	n	o
1	6	2.13	apple
2	3	4.25	banana
3	1	6.37	oregano
4	9	5.67	pepper

Here, PROC step prints only the first 4 observations of the data set *third*

SAS Options

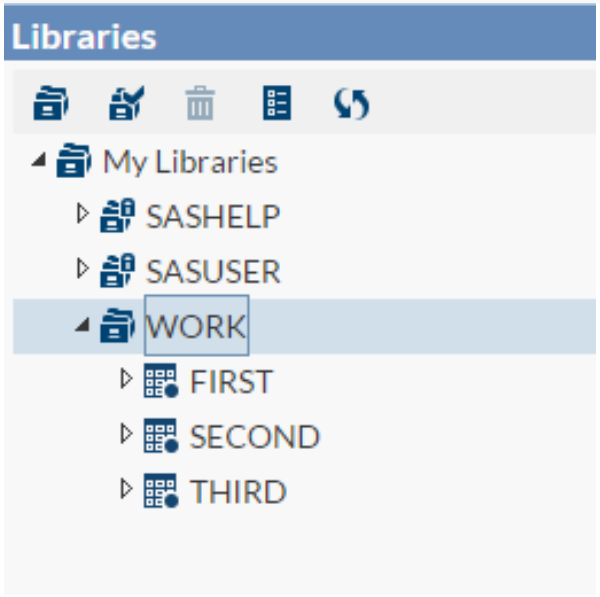
```
20 data third;
21 input m n o$;
22 datalines;
23 6 2.13 apple
24 3 4.25 banana
25 1 6.37 oregano
26 9 5.67 pepper
27 4 7.12 mango
28 2 2.84 orange
29 ;
30 run;
31 proc print data = third (firstobs = 2 obs = 4);
32 run;
```

The SAS System

Obs	m	n	o
2	3	4.25	banana
3	1	6.37	oregano
4	9	5.67	pepper

Here, PROC step prints the data set *third* beginning with observation 2 till observation 4

SAS Libraries



Libraries (Default) –

- SASHELP – contains sample data sets
- SASUSER – stores personal files

Temporary Library –

WORK – stores files only for current session

SAS Files

Rules for data set names –

- i. 1 to 32 characters
- ii. must begin with an alphabet A-Z(uppercase or lowercase) or underscore
- ii. can continue with any combination of alphabets, numbers or underscores

Referencing SAS Files

- Two-Level Names
- To reference a permanent SAS data set in your SAS programs, we use a two-level name
- library name and the filename, or data set name:
- libref.filename

PROC Datasets

PROC DATASETS is used

- to list, copy, remove, or delete SAS files.
- to change variable information such as name, format, informat and label.

PROC Datasets does not require RUN Statement

PROC Datasets

The SAS log gives the name of all SAS datasets in the library called *SASHELP*

```
1 proc datasets lib = sashelp;
```

The SAS System

Directory	
Libref	SASHELP
Levels	4

Level 1	
Engine	V9
Physical Name	/opt/sasinside/SASHome/SASFoundation/9.4/nls/u8/sascfg
Filename	/opt/sasinside/SASHome/SASFoundation/9.4/nls/u8/sascfg
Inode Number	259646
Access Permission	rwxr-xr-x
Owner Name	sas
File Size (bytes)	4096

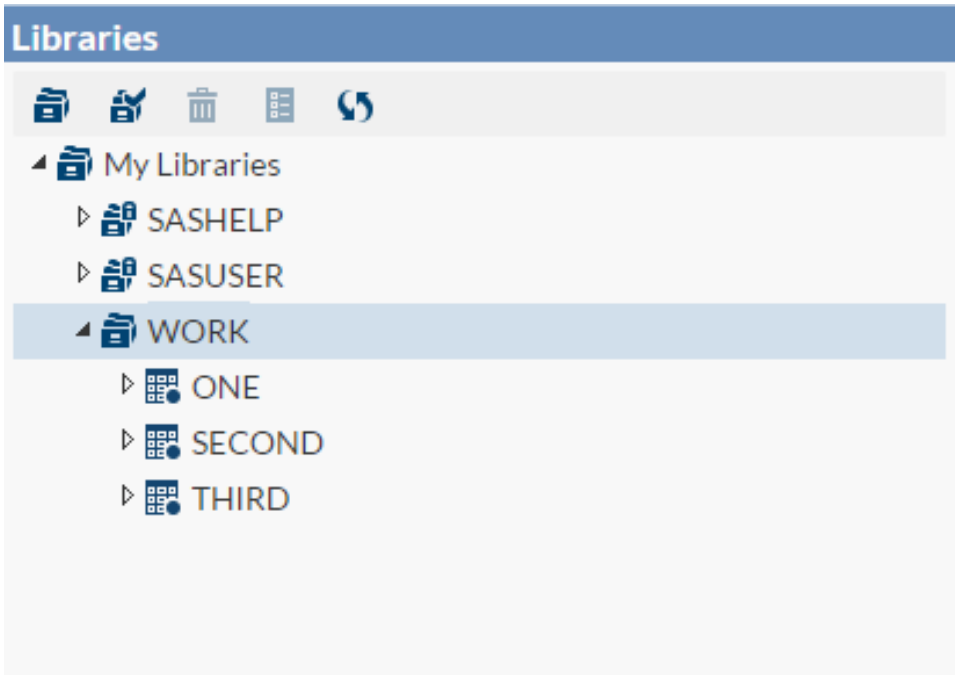
Level 2	
Engine	V9
Physical Name	/opt/sasinside/SASHome/SASFoundation/9.4/nls/u8/sashelp
Filename	/opt/sasinside/SASHome/SASFoundation/9.4/nls/u8/sashelp
Inode Number	259641
Access Permission	rwxr-xr-x
Owner Name	[illegible]

#	Name	Member Type	Level	File Size	Last Modified
1	AACOMP	DATA	4	393216	07/24/2014 04:52:41
	AACOMP	INDEX		147456	07/24/2014 04:52:41
2	AARFM	DATA	4	262144	07/24/2014 04:52:56
	AARFM	INDEX		24576	07/24/2014 04:52:56
3	AC	CATALOG	4	24576	06/13/2013 00:55:17
4	ADSMSG	DATA	4	262144	06/13/2013 00:51:27
	ADSMSG	INDEX		73728	06/13/2013 00:51:27
5	AFCLASS	CATALOG	4	2043904	06/13/2013 01:05:30
6	AFMSG	DATA	4	458752	06/13/2013 00:42:10
	AFMSG	INDEX		106496	06/13/2013 00:42:10
7	AFTOOLS	CATALOG	4	2408448	06/13/2013 01:05:38
8	AIR	DATA	4	131072	06/13/2013 01:00:53
9	ASSCMGR	DATA	4	327680	06/13/2013 01:15:03
10	BASE	CATALOG	4	241664	06/13/2013 02:16:33
11	BASEBALL	DATA	4	196608	07/24/2014 05:04:56
12	BEI	DATA	4	1048576	07/24/2014 05:04:59
13	BMT	DATA	4	131072	06/13/2013 01:23:40
14	BWEIGHT	DATA	4	4128768	07/24/2014 05:04:57
15	CARS	DATA	4	196608	06/13/2013 01:09:53

PROC Datasets – CHANGE Statement

In the following program, we change the name of dataset *first* to *one* using CHANGE Statement.

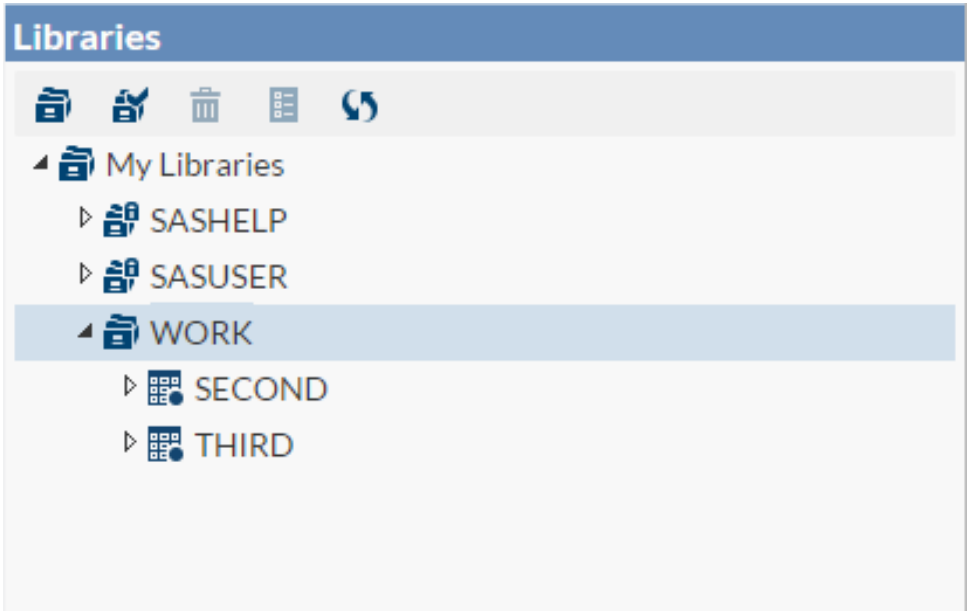
```
2 proc datasets  lib = work;
3 change first = one;
```



PROC Datasets – DELETE Statement

In the following program, we delete dataset *one* using DELETE Statement.

```
5 proc  datasets lib = work;
6 delete  one;
```



Create data set using existing dataset

- Use SET Statement and create a temporary data set for use in current session from an existing data set
- Print the new data set using PROC Print

```
33 data baseball;
34 SET sashelp.baseball;
35 run;
36 proc print data=baseball;
37 run;
```

- Try and see what the following program will do

```
33 data baseball;
34 SET sashelp.baseball (firstobs = 5 obs = 53) ;
35 run;
36 proc print data=baseball;
37 run;
```

PROC Print – TITLE

As the name suggests, TITLE Statements adds Title while printing the output.

```
34 data baseball;
35 set sashelp.baseball;
36 run;
37 proc print data = baseball;
38 title 'BASEBALL Dataset';
39 run;
```

BASEBALL Dataset

Obs	Name	Team	nAtBat	nHits	nHome	nRuns	nRBI	nBB	YrMajor	CrAtBat	CrHits	CrHome	CrRuns	CrRbi	CrBB	League	Division	Positi
1	Thomas, Andres	Atlanta	323	81	6	26	32	8	2	341	86	6	32	34	8	National	West	SS
2	Horner, Bob	Atlanta	517	141	27	70	87	52	9	3571	994	215	545	652	337	National	West	1B
3	Sample, Billy	Atlanta	200	57	6	23	14	14	9	2516	684	46	371	230	195	National	West	OF
4	Murphy, Dale	Atlanta	614	163	29	89	83	75	11	5017	1388	266	813	822	617	National	West	CF

PROC Print – SUM

To generate Column totals use SUM Statement within PROC Step

```
40 data baseball;
41 set sashelp.baseball;
42 run;
43 proc print data = baseball (obs = 10);
44 var nhits nruns;
45 sum nhits;
46 run;
```

The SAS System

Obs	nHits	nRuns
1	66	30
2	81	24
3	130	66
4	141	65
5	87	39
6	169	74
7	37	23
8	73	24
9	81	26
10	92	49
	957	

PROC Sort

Here, PROC Sort is used to create a sorted dataset in ascending order according to *team* variable.

```
37 proc sort data = sashelp.baseball out = baseball;
38 by team;
39 run;
40 proc print data = baseball;
41 var team nhits nhome nruns;
42 run;
```

The SAS System

Obs	Team	nHits	nHome	nRuns
1	Atlanta	81	6	26
2	Atlanta	141	27	70
3	Atlanta	57	6	23
4	Atlanta	163	29	89
5	Atlanta	94	4	42
6	Atlanta	136	5	62
7	Atlanta	84	4	46
8	Atlanta	80	15	45
9	Atlanta	119	8	57
10	Atlanta	68	8	26
11	Atlanta	32	4	14
12	Baltimore	60	0	30
13	Baltimore	177	25	98
14	Baltimore	151	17	61
15	Baltimore	114	23	67
16	Baltimore	37	8	15

Calculate Subtotals

Here, we calculate and print Subtotals for nhits sorted by variable team.

```
37 proc sort data = sashelp.baseball out = baseball;
38 by team;
39 run;
40 proc print data = baseball;
41 var team nhits nruns nhome;
42 sum nhits;
43 by team;
44 run;
```

The SAS System

Team at the End of 1986=Atlanta

Obs	Team	nHits	nRuns	nHome
1	Atlanta	81	26	6
2	Atlanta	141	70	27
3	Atlanta	57	23	6
4	Atlanta	163	89	29
5	Atlanta	94	42	4
6	Atlanta	136	62	5
7	Atlanta	84	46	4
8	Atlanta	80	45	15
9	Atlanta	119	57	8
10	Atlanta	68	26	8
11	Atlanta	32	14	4
Team		1055		

Team at the End of 1986=Baltimore

Obs	Team	nHits	nRuns	nHome
12	Baltimore	60	30	0
13	Baltimore	177	98	25
14	Baltimore	151	61	17
15	Baltimore	114	67	23

PROC Contents

PROC Contents -

- provides information for SAS datasets or libraries
- it gives the name of the dataset or library, the location, when it was created, the host that created it, and the time of the last modification.
- For datasets it also provides the number of observations in the dataset, and attributes for each variable

PROC Contents

The following program requests information for sashelp.baseball dataset using PROC Contents.

```
85 proc contents data = sashelp.baseball;
86 run;
```

The SAS System

The CONTENTS Procedure

Data Set Name	SASHELP.BASEBALL	Observations	322
Member Type	DATA	Variables	24
Engine	V9	Indexes	0
Created	07/24/2014 05:04:55	Observation Length	216
Last Modified	07/24/2014 05:04:55	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	1986 Baseball Data		
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	303
Obs in First Data Page	279
Number of Data Set Repairs	0
Filename	/opt/sasinside/SASHome/SASFoundation/9.4/sashelp/baseball.sas7bdat
Release Created	9.0401M2
Host Created	Linux
Inode Number	261730

PROC Freq

PROC FREQ –

- Counts the number (frequency) of occurrences of each variable(both character and numeric).
- Gives statistics from the data
- Produces one- way to n-way frequency and cross-tabulation tables.
- Produces printed output by default.
- Lists each variable value along with the frequencies and percentages.

PROC Freq

The following program demonstrates the simplest form of PROC FREQ and produces the output

```
33 data baseball;
34 SET sashelp.baseball;
35 run;
36 proc freq data = baseball;
37 run;
```

The output of this program is too long for our purpose to include here.

Add TABLES Statement to limit the output for only one variable and create a one-way frequency table

```
33 data baseball;
34 SET sashelp.baseball;
35 run;
36 proc freq data = baseball;
37 tables team;
38 run;
```

PROC Freq

Create two-way cross-tabulation tables :

In the TABLES statement, state the variable names separated by asterisk

```
33 data baseball;
34 SET sashelp.baseball;
35 run;
36 proc freq data = baseball;
37 tables team*league;
38 run;
```

The SAS System

The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of Team by League

Team(Team at the End of 1986)	League(League at the End of 1986) American	National	Total
Atlanta	0 0.00 0.00 0.00	11 3.42 100.00 7.48	11 3.42
Baltimore	15 4.66 100.00 8.57	0 0.00 0.00 0.00	15 4.66
Boston	10 3.11 100.00 5.71	0 0.00 0.00 0.00	10 3.11
California	13 4.04 100.00 7.43	0 0.00 0.00 0.00	13 4.04
Chicago	13 4.04 54.17 7.43	11 3.42 45.83 7.48	24 7.45
Cincinnati	0 0.00 0.00	12 3.73 100.00	12 3.73

PROC Means

PROC MEANS –

- produces statistics for numeric variables.
- produces printed output by default
- computes N, MEAN, STD, MIN and MAX by default
- similar to PROC SUMMARY however, PROC SUMMARY does not produce a printed output by default.

PROC Means

The following shows simplest use of PROC MEANS procedure.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc MEANS data = baseball;
37 run;
```

The SAS System

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
nAtBat	Times at Bat in 1986	322	390.0745342	143.5958352	127.0000000	687.0000000
nHits	Hits in 1986	322	103.3975155	44.1795091	31.0000000	238.0000000
nHome	Home Runs in 1986	322	11.1024845	8.6987696	0	40.0000000
nRuns	Runs in 1986	322	52.2173913	25.0573661	12.0000000	130.0000000
nRBI	RBIs in 1986	322	49.3726708	25.5011624	8.0000000	121.0000000
nBB	Walks in 1986	322	39.8571429	21.0959408	3.0000000	105.0000000
YrMajor	Years in the Major Leagues	322	7.6801242	4.9697066	1.0000000	24.0000000
CrAtBat	Career Times at Bat	322	2763.08	2328.48	166.0000000	14053.00
CrHits	Career Hits	322	747.6863354	654.7876194	34.0000000	4256.00
CrHome	Career Home Runs	322	74.0900621	90.0651268	0	548.0000000
CrRuns	Career Runs	322	374.2857143	336.4250377	18.0000000	2165.00
CrRbi	Career RBIs	322	347.6149068	338.7903452	9.0000000	1659.00
CrBB	Career Walks	322	273.3944099	273.6253716	8.0000000	1566.00
nOuts	Put Outs in 1986	322	288.9937888	280.6566732	0	1378.00
nAssts	Assists in 1986	322	106.9161491	136.8524541	0	492.0000000
nError	Errors in 1986	322	8.0403727	6.3683591	0	32.0000000
Salary	1987 Salary in $ Thousands	263	535.9258821	451.1186807	67.5000000	2460.00
logSalary	Log Salary	263	5.9272215	0.8891924	4.2121276	7.8079166

PROC Means – VAR

VAR statement –

- selects specified variable ,
- Identifies the analysis variables and their order in the output.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc MEANS data = baseball;
37 VAR salary;
38 run;
```

The SAS System

The MEANS Procedure

Analysis Variable : Salary 1987 Salary in $ Thousands				
N	Mean	Std Dev	Minimum	Maximum
263	535.9258821	451.1186807	67.5000000	2460.00

PROC Means – Statistic

To obtain a specific statistic you must state it in the PROC MEANS statement.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc means data = baseball MEAN;
37 run;
```

The SAS System

The MEANS Procedure

Variable	Label	Mean
nAtBat	Times at Bat in 1986	390.0745342
nHits	Hits in 1986	103.3975155
nHome	Home Runs in 1986	11.1024845
nRuns	Runs in 1986	52.2173913
nRBI	RBIs in 1986	49.3726708
nBB	Walks in 1986	39.8571429
YrMajor	Years in the Major Leagues	7.6801242
CrAtBat	Career Times at Bat	2763.08
CrHits	Career Hits	747.6863354
CrHome	Career Home Runs	74.0900621
CrRuns	Career Runs	374.2857143
CrRbi	Career RBIs	347.6149068
CrBB	Career Walks	273.3944099
nOuts	Put Outs in 1986	288.9937888
nAssts	Assists in 1986	106.9161491
nError	Errors in 1986	8.0403727
Salary	1987 Salary in $ Thousands	535.9258821
logSalary	Log Salary	5.9272215

PROC Means – Class

- The CLASS statement assigns variables used to form subgroups.
- CLASS variables can be either numeric or character.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc means data = baseball mean;
37 CLASS league;
38 run;
```

The SAS System

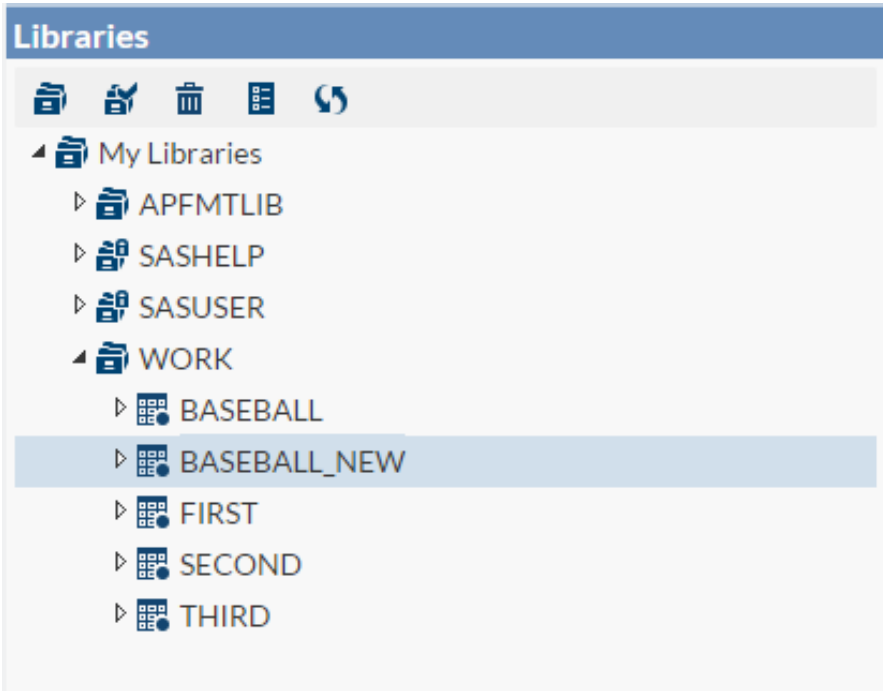
The MEANS Procedure

League at the End of 1986	N Obs	Variable	Label	Mean
American	175	nAtBat	Times at Bat in 1986	405.0514286
		nHits	Hits in 1986	107.6857143
		nHome	Home Runs in 1986	12.4857143
		nRuns	Runs in 1986	55.7771429
		nRBI	RBIs in 1986	52.7828571
		nBB	Walks in 1986	40.7771429
		YrMajor	Years in the Major Leagues	7.9714286
		CrAtBat	Career Times at Bat	2855.15
		CrHits	Career Hits	770.7942857
		CrHome	Career Home Runs	82.3657143
		CrRuns	Career Runs	393.6971429
		CrRbi	Career RBIs	367.1142857
		CrBB	Career Walks	287.4400000
		nOuts	Put Outs in 1986	282.8800000
		nAssts	Assists in 1986	101.2171429
		nError	Errors in 1986	7.5828571
		Salary	1987 Salary in $ Thousands	541.9995468
		logSalary	Log Salary	5.9326225
National	147	nAtBat	Times at Bat in 1986	372.2448980
		nHits	Hits in 1986	98.2925170
		nHome	Home Runs in 1986	9.4557823
		nRuns	Runs in 1986	47.9795918
		nRBI	RBIs in 1986	45.3129252
		nBB	Walks in 1986	38.7619048
		YrMajor	Years in the Major Leagues	7.3333333
		CrAtBat	Career Times at Bat	2653.47
		CrHits	Career Hits	720.1768707
		CrHome	Career Home Runs	64.2380952
		CrRuns	Career Runs	351.1768707
		CrRbi	Career RBIs	324.4013605

PROC Means – Output

Output Statement stores result in new data set.
Here, the new data set is *baseball_new*.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc means data = baseball mean;
37 output out = baseball_new;
38 run;
```



PROC Univariate

PROC UNIVARIATE

- Examines distribution for numeric variables.
- produces printed output by default

The following shows simplest use of PROC Univariate procedure.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc univariate data = baseball;
37 run;
```

The SAS System

The UNIVARIATE Procedure
Variable: nAtBat (Times at Bat in 1986)

Moments			
N	322	Sum Weights	322
Mean	380.074534	Sum Observations	125604
Std Deviation	143.595835	Variance	20619.7639
Skewness	0.07598562	Kurtosis	-1.1786552
Uncorrected SS	55613866	Corrected SS	6618944.21
Coeff Variation	36.8124096	Std Error Mean	8.00228304

Basic Statistical Measures			
Location		Variability	
Mean	380.0745	Std Deviation	143.59584
Median	380.5000	Variance	20620
Mode	209.0000	Range	560.00000

PROC Univariate – NORMAL

NORMAL Option requests for tests for normality that include a series of goodness-of-fit tests based on the empirical distribution function.

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc univariate data = baseball normal;
37 var nhits;
38 run;
```

The SAS System

The UNIVARIATE Procedure
Variable: nHits (Hits in 1986)

Moments			
N	322	Sum Weights	322
Mean	103.397516	Sum Observations	33294
Std Deviation	44.1795091	Variance	1951.82903
Skewness	0.4233842	Kurtosis	-0.5671338
Uncorrected SS	4069054	Corrected SS	626537.118
Coeff Variation	42.7278247	Std Error Mean	2.46202779

Basic Statistical Measures			
Location		Variability	
Mean	103.3975	Std Deviation	44.17951
Median	98.5000	Variance	1952
Mode	53.0000	Range	207.00000
		Interquartile Range	70.00000

Note: The mode displayed is the smallest of 4 modes with a count of 6.

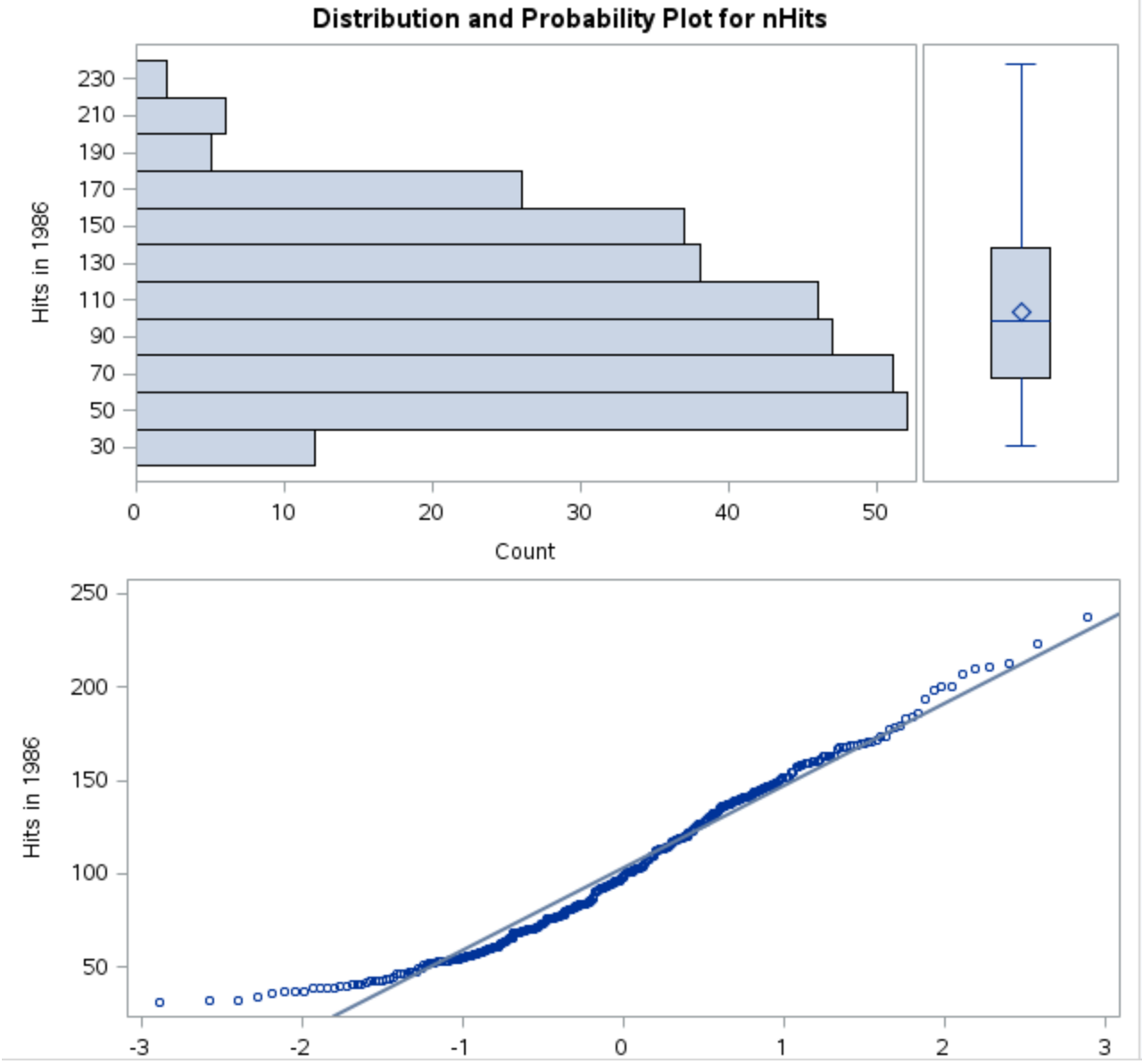
Tests for Location: Mu0=0				
Test	Statistic		p Value	
Student's t	t	41.99689	Pr > |t|	<.0001
Sign	M	161	Pr >= |M|	<.0001
Signed Rank	S	26001.5	Pr >= |S|	<.0001

PROC Univariate – PLOT

PLOT Option in PROC Univariate produces a stem-and-leaf plot (or a horizontal bar chart), a box plot, and a normal quantile plot

```
33 data baseball;
34 set sashelp.baseball;
35 run;
36 proc univariate data = baseball plot;
37 var nhits;
38 run;
```

PROC Univariate – PLOT



DO Loops

```
87 data fourth;
88 do i = 1 to 10;
89    y = i**2;
90    output;
91 end;
92 run;
93 proc print data = fourth;
94 run;
```

The SAS System

Obs	i	y
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25
6	6	36
7	7	49
8	8	64
9	9	81
10	10	100

Incrementing Loop

Here the i is being incremented by 2 in the loop

```
 98 data fifth;
 99 do i = 1 to 10 by 2;
100    y = i**2;
101    output;
102 end;
103 run;
104 proc print data = fifth;
105 title "Fifth";
106 run;
```

Fifth

Obs	i	y
1	1	1
2	3	9
3	5	25
4	7	49
5	9	81

Dropping Variable

Here, we create a dataset *Sixth.* We run a loop using a new variable *i,* but drop it so that it does not include in the dataset.

```
107 data Sixth(drop = i);
108 do i = 1 to 10 by 2;
109    y = i**2;
110    output;
111 end;
112 run;
113 proc print data = Sixth;
114 title "Sixth";
115 run;
```

Sixth

Obs	y
1	1
2	9
3	25
4	49
5	81

Decrementing Loop

Here the i is being decremented by 2 in the loop

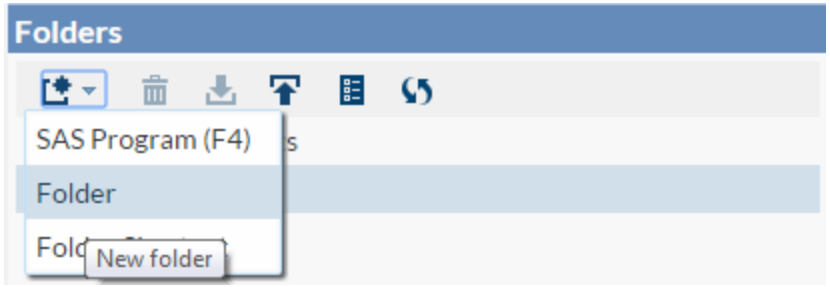
```
116 data Seventh(drop = i);
117 do i = 10 to 1 by -2;
118    y = i**2;
119    output;
120 end;
121 run;
122 proc print data = Seventh;
123 title "Seventh";
124 run;
```

Seventh

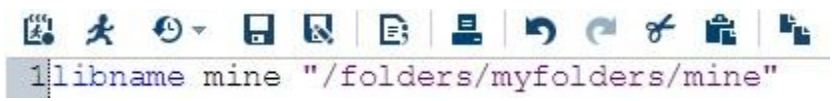
Obs	y
1	100
2	64
3	36
4	16
5	4

Dataset using Raw file

1. Create a new folder in myfolders and call it *mine*.

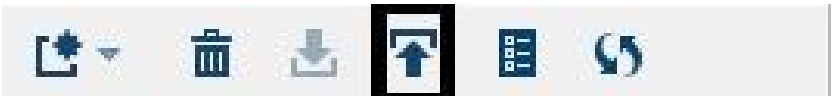


2. Use libname statement to create a new SAS library called *mine*. How do we get the path of mine?

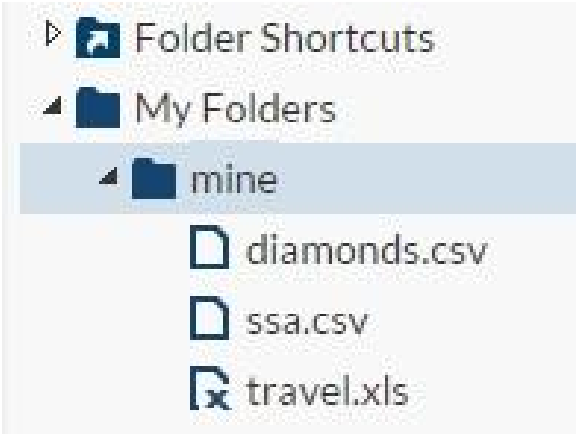


Dataset using Raw file

3. Upload diamonds.csv dataset into the mine folder



4. Check the folder



PROC Import

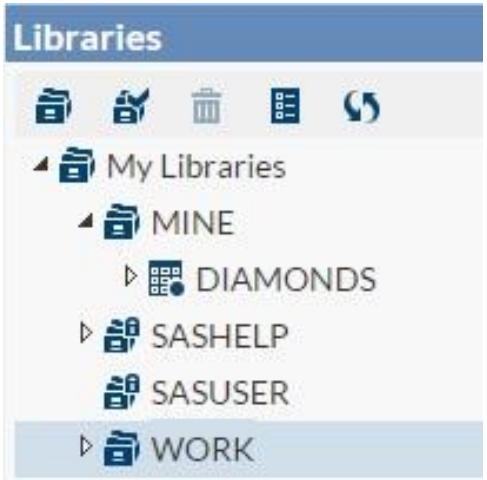
We use Proc IMPORT to import the dataset.

- OUT Option specifies a name for SAS data set
- DATAFILE Statement specifies the location/path of the uploaded file.
- DBMS Statement indicates the type of uploaded file.
- GETNAMES Statement tells SAS whether the first row of the data values is to be considered as variable names or not(Yes - if to be considered as variable names).

```
1 libname mine "/folders/myfolders/mine"
2 proc import out = mine.diamonds
3 datafile= '/folders/myfolders/mine/diamonds.csv'
4 DBMS=CSV;
5 getnames=yes;
6 run;
```

Dataset using Raw file

Check the MINE Library



Create a temporary data set called diamonds, using the existing data set.

```
10 data diamonds;
11 set mine.diamonds;
12 run;
```

PROC Corr

Correlation is one of the first steps to understand the relationship between variables. To compute correlation in SAS, we use PROC CORR.

Corr Procedure calculates pairwise correlation for Numeric variables. This procedure also provides some summary statistics by default - Mean, Standard Deviation, Sum, Minimum and Maximum.

```
23 proc corr data=diamonds;
24 run;
```

The SAS System

The CORR Procedure

7 Variables:	carat depth table price x y z

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
carat	53940	0.79794	0.47401	43041	0.20000	5.01000
depth	53940	61.74940	1.43262	3330763	43.00000	79.00000
table	53940	57.45718	2.23449	3099241	43.00000	95.00000
price	53940	3933	3989	212135217	326.00000	18823
x	53940	5.73116	1.12176	309139	0	10.74000
y	53940	5.73453	1.14213	309320	0	58.90000
z	53940	3.53873	0.70570	190879	0	31.80000

Pearson Correlation Coefficients, N = 53940
Prob > |r| under H0: Rho=0

	carat	depth	table	price	x	y	z
carat	1.00000	0.02822 <.0001	0.18162 <.0001	0.92159 <.0001	0.97509 <.0001	0.95172 <.0001	0.95339 <.0001
depth	0.02822 <.0001	1.00000	-0.29578 <.0001	-0.01065 0.0134	-0.02529 <.0001	-0.02934 <.0001	0.09492 <.0001
table	0.18162 <.0001	-0.29578 <.0001	1.00000	0.12713 <.0001	0.19534 <.0001	0.18376 <.0001	0.15093 <.0001
price	0.92159 <.0001	-0.01065 0.0134	0.12713 <.0001	1.00000	0.88444 <.0001	0.86542 <.0001	0.86125 <.0001
x	0.97509 <.0001	-0.02529 <.0001	0.19534 <.0001	0.88444 <.0001	1.00000	0.97470 <.0001	0.97077 <.0001
y	0.95172 <.0001	-0.02934 <.0001	0.18376 <.0001	0.86542 <.0001	0.97470 <.0001	1.00000	0.95201 <.0001
z	0.95339 <.0001	0.09492 <.0001	0.15093 <.0001	0.86125 <.0001	0.97077 <.0001	0.95201 <.0001	1.00000

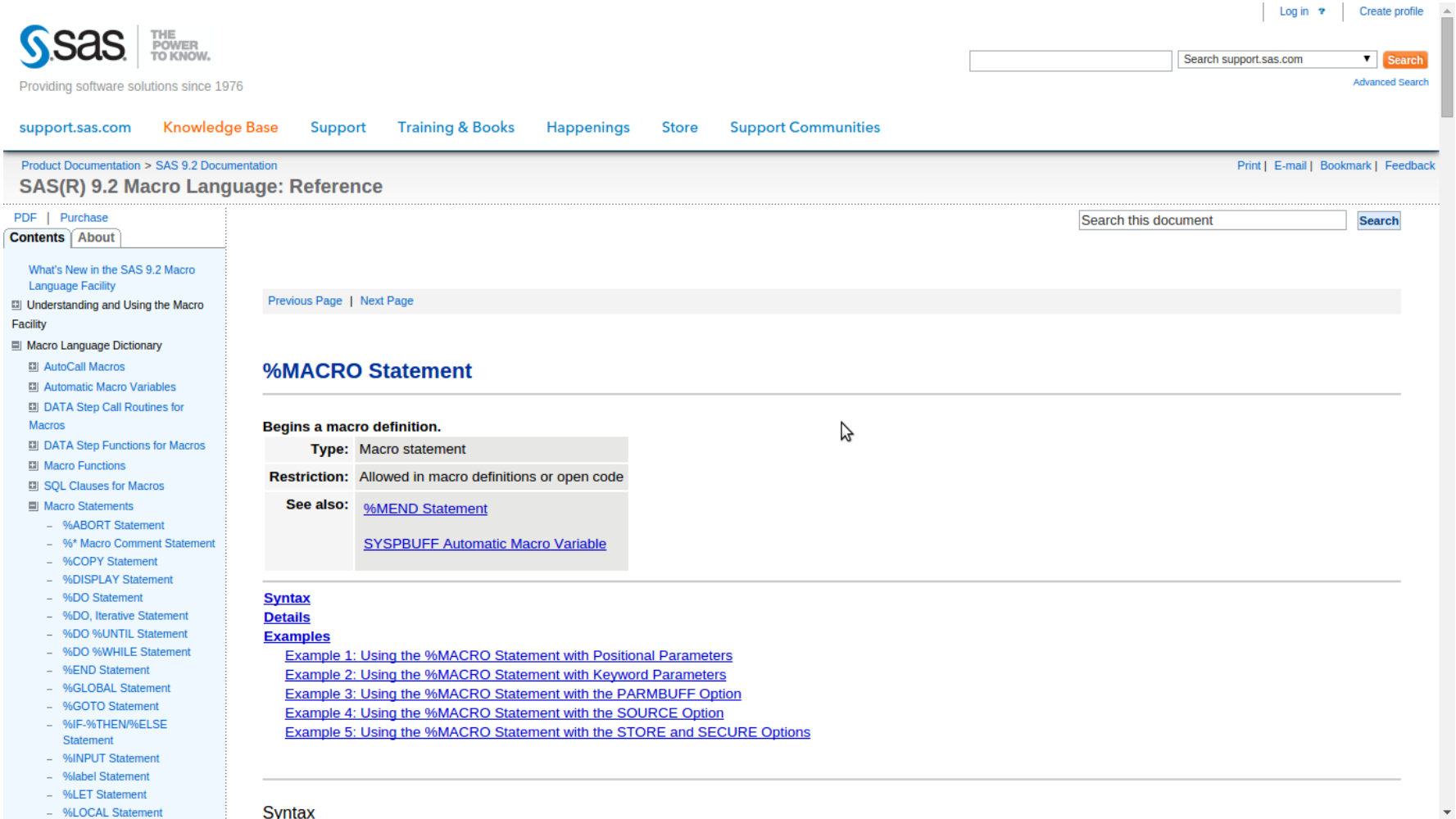
Assignment

Make use of the procedures learnt in the class to compute statistics for diamonds data set.

https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Diamond.csv

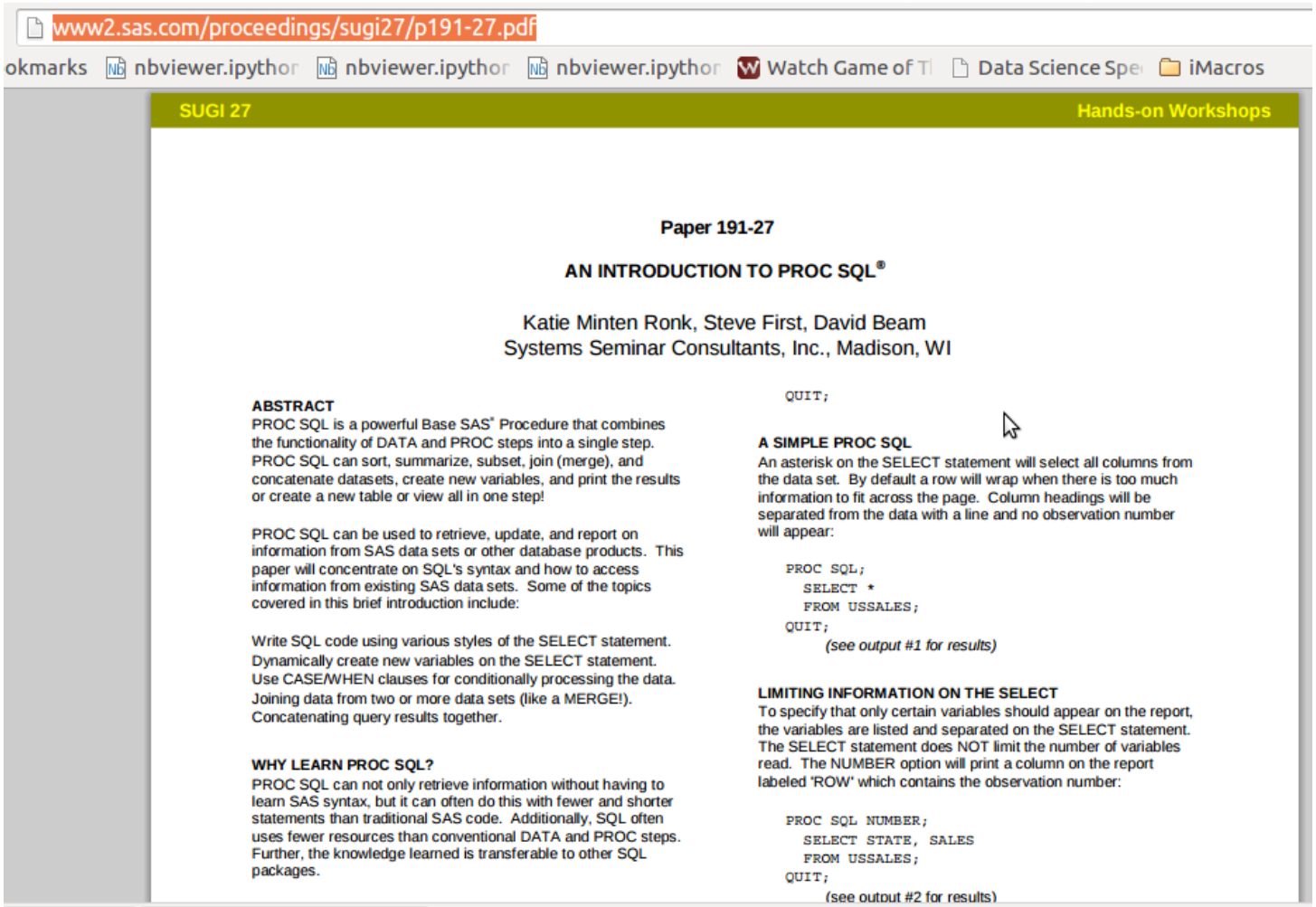
SAS DOCUMENTATION = Hallelujah!

http://support.sas.com/documentation/



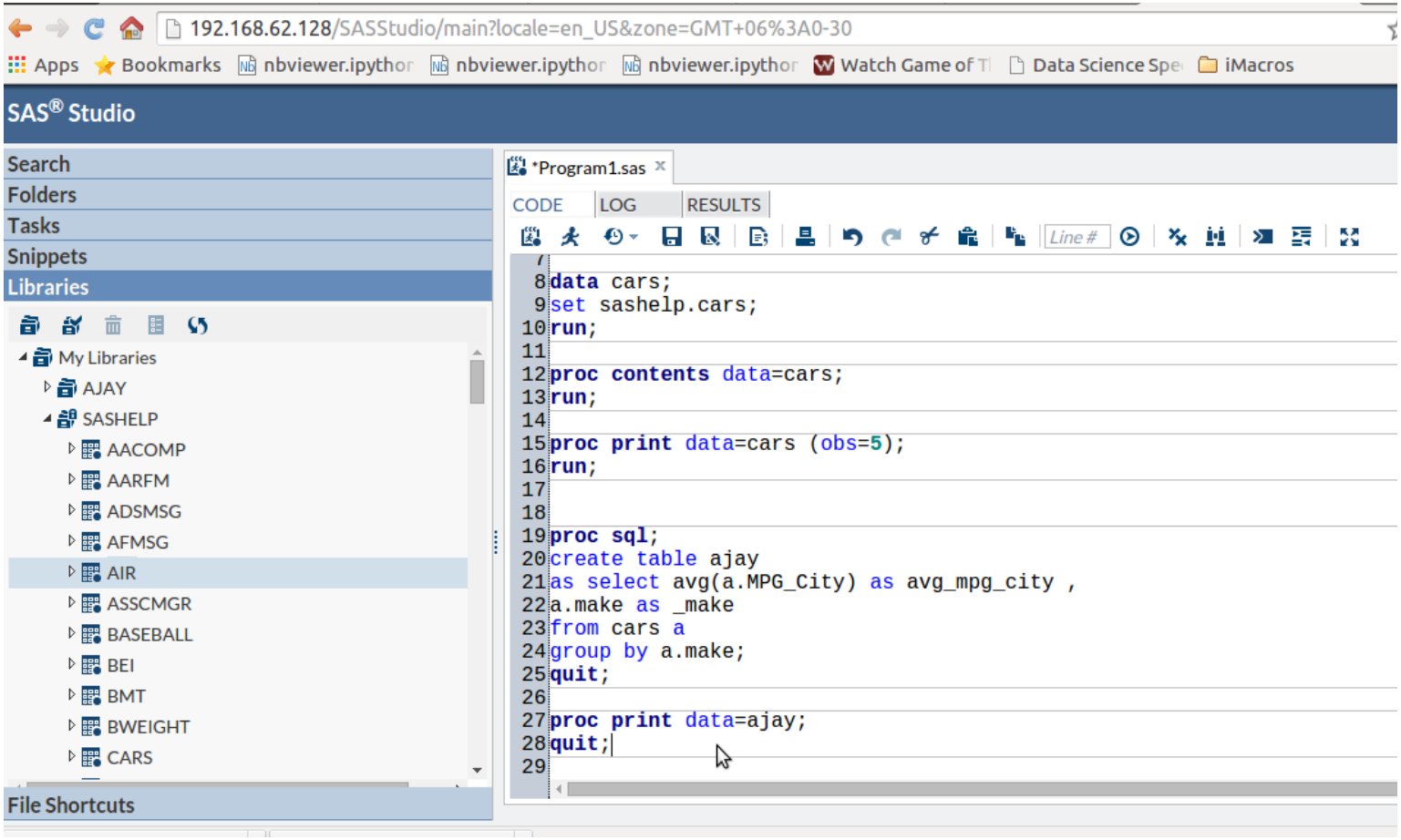
PROC SQL

http://www2.sas.com/proceedings/sugi27/p191-27.pdf



PROC SQL

Use SQL within SAS



MACRO LANGUAGE

https://v8doc.sas.com/sashtml/macro/znemacro.htm

Generating SAS Code Using Macros

Macros allow you to substitute text in a program and to do many other things. A SAS program can contain any number of macros, and you can invoke a macro any number of times in a single program.

To help you learn how to define your own macros, this section presents a few examples you can model your own macros after. Each of these examples is fairly simple; by mixing and matching the various techniques, you can create advanced, flexible macros that are capable of performing complex tasks.

Each macro you define has a distinct name, which is subject to the standard SAS naming conventions. (See the base SAS language documentation for more information on SAS naming conventions.) A macro definition is placed between a %MACRO statement and a %MEND (macro end) statement, as follows:

%MACRO *macro-name*;

macro definition

%MEND *macro-name*;

The **macro-name** specified in the %MEND statement must match the **macro-name** specified in the %MACRO statement.

Note: While specifying the **macro-name** in the %MEND statement is not required, it is recommended. It makes matching %MACRO and %MEND statements while debugging easier. ■

Here is a simple macro definition:

```
%macro dsn;
   Newdata
%mend dsn;
```

This macro is named DSN. **`Newdata`** is the text of the macro. A string inside a macro is called **constant text** or **model text** because it is the model, or pattern, for the text that becomes part of your SAS program.

To call (or **invoke**) a macro, precede the name of the macro with a percent sign (%), as follows:

%macro-name

Although the call to the macro looks somewhat like a SAS statement, it does not have to end in a semicolon.

For example, here is how you might call the DSN macro:

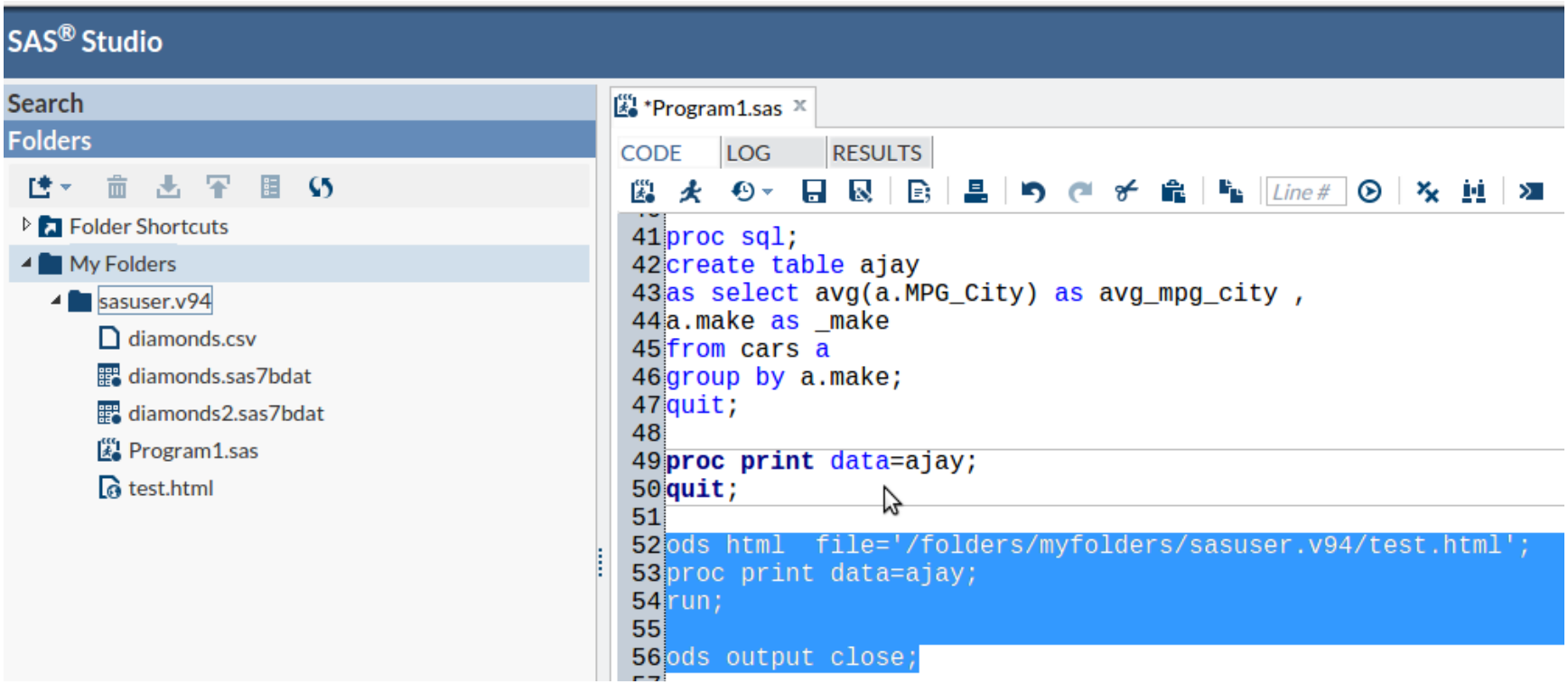
```
title "Display of Data Set %dsn";
```

MACRO LANGUAGE

```
30
31 options macrogen symbolgen;
32
33
34 %macro ajay(newvar,numb);
35
36
37    proc print data=cars (obs=&numb);
38       var &newvar;
39
40    run;
41 %mend ajay;
42
43 %ajay(MPG_City,10)
44 %ajay(make,5)
45 %ajay(cylinders,4)
46
```

ODS

http://support.sas.com/rnd/base/ods/scratch/ods-tips.pdf



ODS

SAS 9 ODS Tip Sheet

Common Destinations

"Destination" is a term for the ODS driver that generates a specific output format.

Destination Name	Description
Listing	Plain text
HTML	Primary format used on the World Wide Web (WWW)
XML	Plain text data interchange format
PDF	Works well for printing as well as on-screen viewing
Postscript	Works well for printing
RTF	Used by most word processors
Tagsets.ExcelXP	Share SAS data with Microsoft Excel
Output	Creates SAS data sets from ODS tables

Opening and Closing Destinations

ods *destination-name <option(s)>*;
Opens the ODS destination, ***destination-name***, with options, ***option(s)***.

ods *destination-name* close;
Closes the ODS destination, ***destination-name***.

ods _all_ close;
Closes all ODS destinations (including Listing).

Basic ODS Usage

```
ods destination-name;
... procedure code ...
ods destination-name close;
```

HTML Destination

Basic Usage

```
ods html <option(s)>;
... procedure code ...
ods html close;
```

File Options

file="*filename*" or **body="*filename*"**
Specifies the name of the file that will contain the output tables.

contents="*filename*"
Specifies the name of the file that will contain a table of contents for the output.

frame="*filename*"
Specifies the name of the file that includes the body file and table of contents into a set of scrollable frames.

stylesheet="*filename*"
Specifies an external filename for styles.

The following file sub-options can be specified in parentheses following each of the above options.

no_top_matter
no_bottom_matter
Specifies that no beginning or ending markup, respectively, should be inserted into the output file.

url="*URL*"
Specifies a URL to be used in place of the filename in links to the file.

Other Commonly Used Option

style=*style-definition*
Specifies the style definition to use.

PDF Destination

Basic Usage

```
ods pdf <option(s)>;
... procedure code ...
ods pdf close;
```

File Options

file="*filename*"
Specifies the name of the file that will contain the output tables.

Document Metadata Options

author="*text*"
keywords="*text*"
subject="*text*"
title="*text*"
Specifies the author, keywords, subject, and title in the metadata of the PDF document.

Other Commonly Used Options

columns=*n*
Specifies the number of columns on each page.

compress=*n*
Specifies the level of compression.

notoc
Specifies that the table of contents bookmarks should not be generated.

startpage=yes | no | now
Controls page breaks.

style=*style-definition*
Specifies the style definition to use.

uniform
Specifies that multiple page tables should retain their width across pages.

RTF Destination

Basic Usage

```
ods rtf <option(s)>;
... procedure code ...
ods rtf close;
```

File Options

file="*filename*"
Specifies the name of the file that will contain the output tables.

Document Metadata Options

author="*text*"
operator="*text*"
title="*text*"
Specifies the author, operator, and title in the metadata of the RTF document.

Other Commonly Used Options

bodytitle
Specifies that the titles and footnotes should appear in the body of the report, not as RTF instructions.

columns=*n*
Specifies the number of columns on each page.

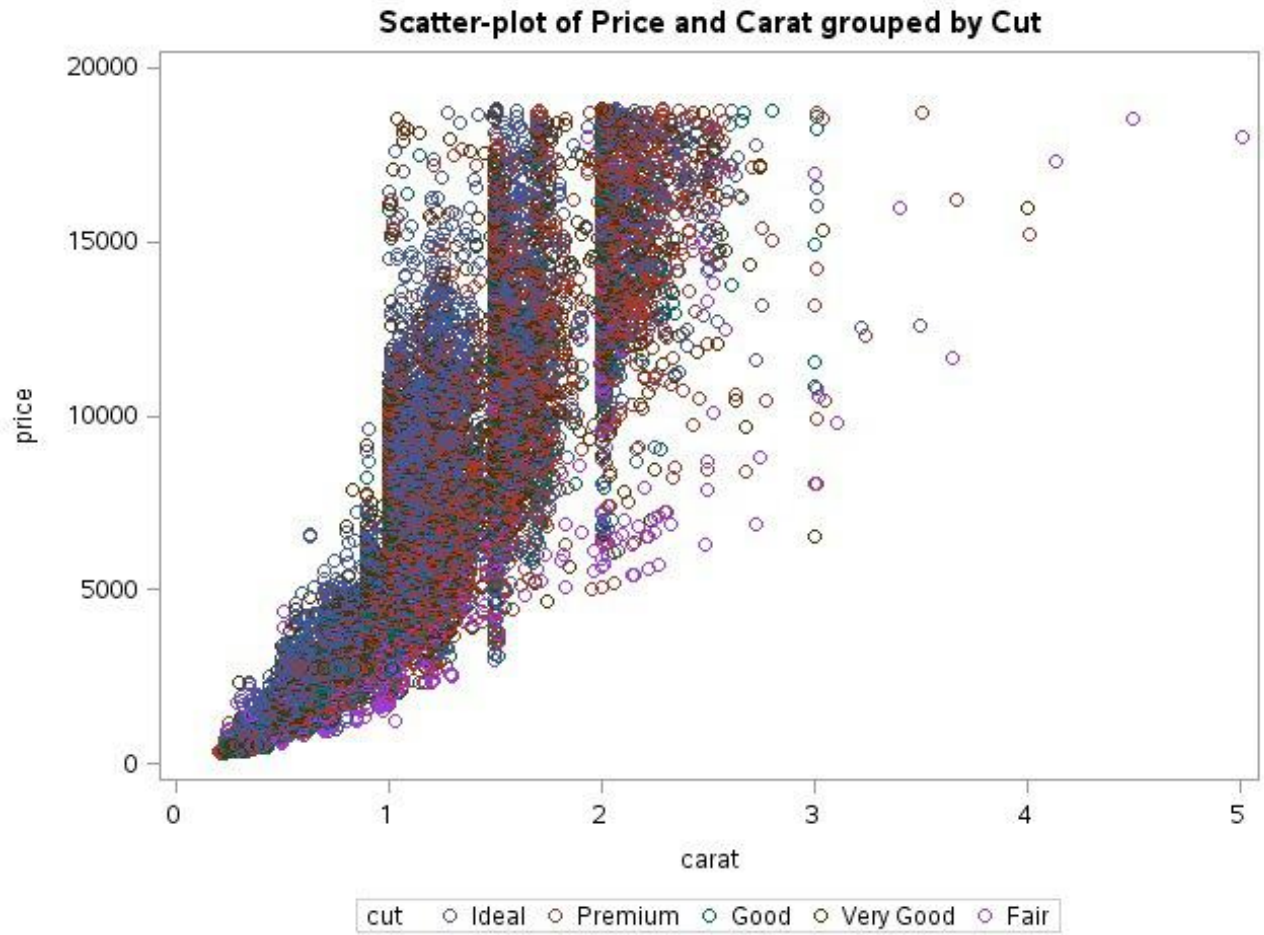
sasdate
Specifies that the time and date that the SAS program was submitted should be written to the RTF, instead of the time that the file was opened.

startpage=yes | no | now
Controls page breaks.

style=*style-definition*
Specifies the style definition to use.

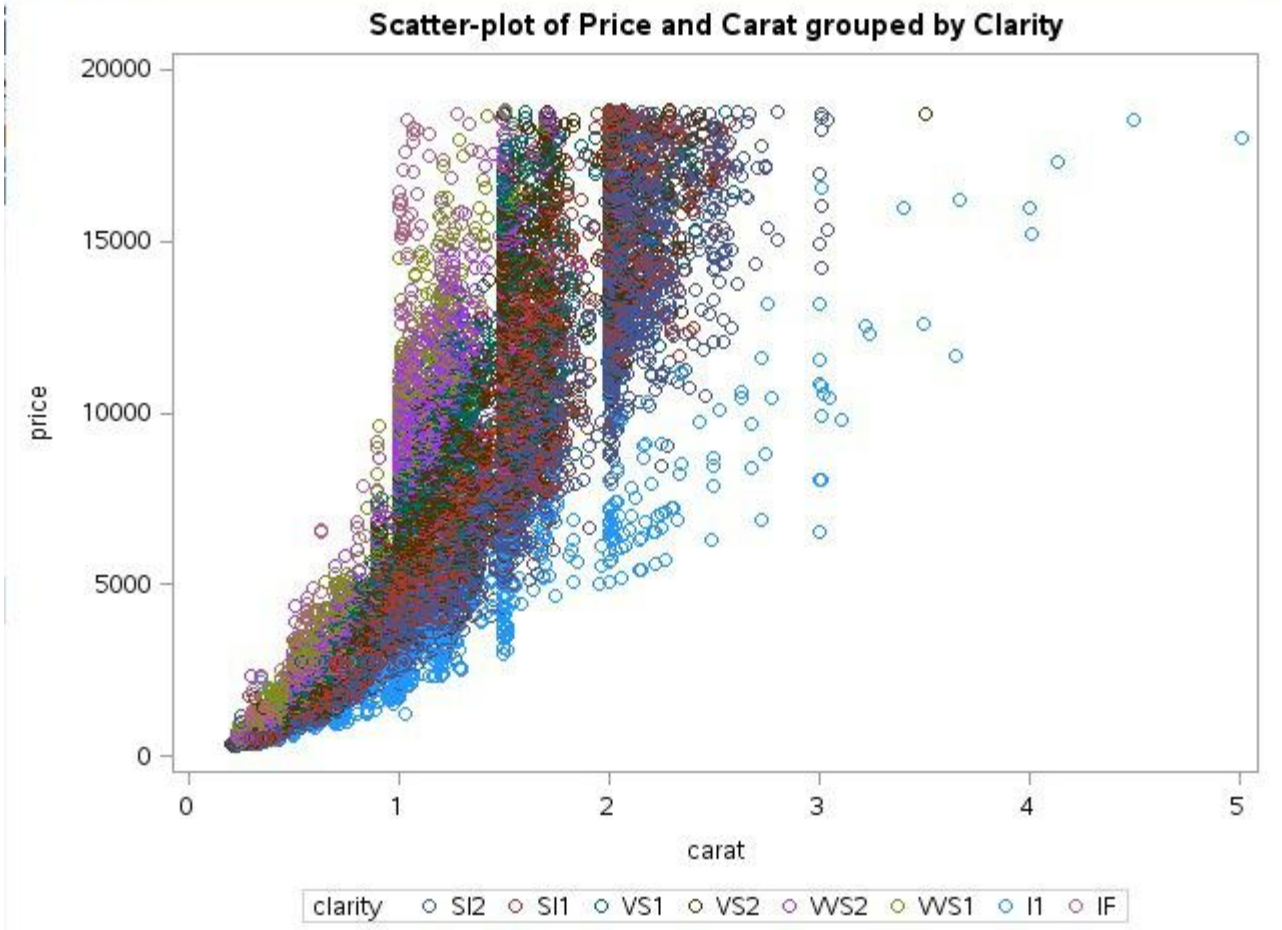
Data Visualization using SAS

```
35 title 'Scatter-plot of Price and Carat grouped by Cut';
36 proc sgplot data=diamonds;
37 scatter x=carat y=price / group=cut;
38 run;
```



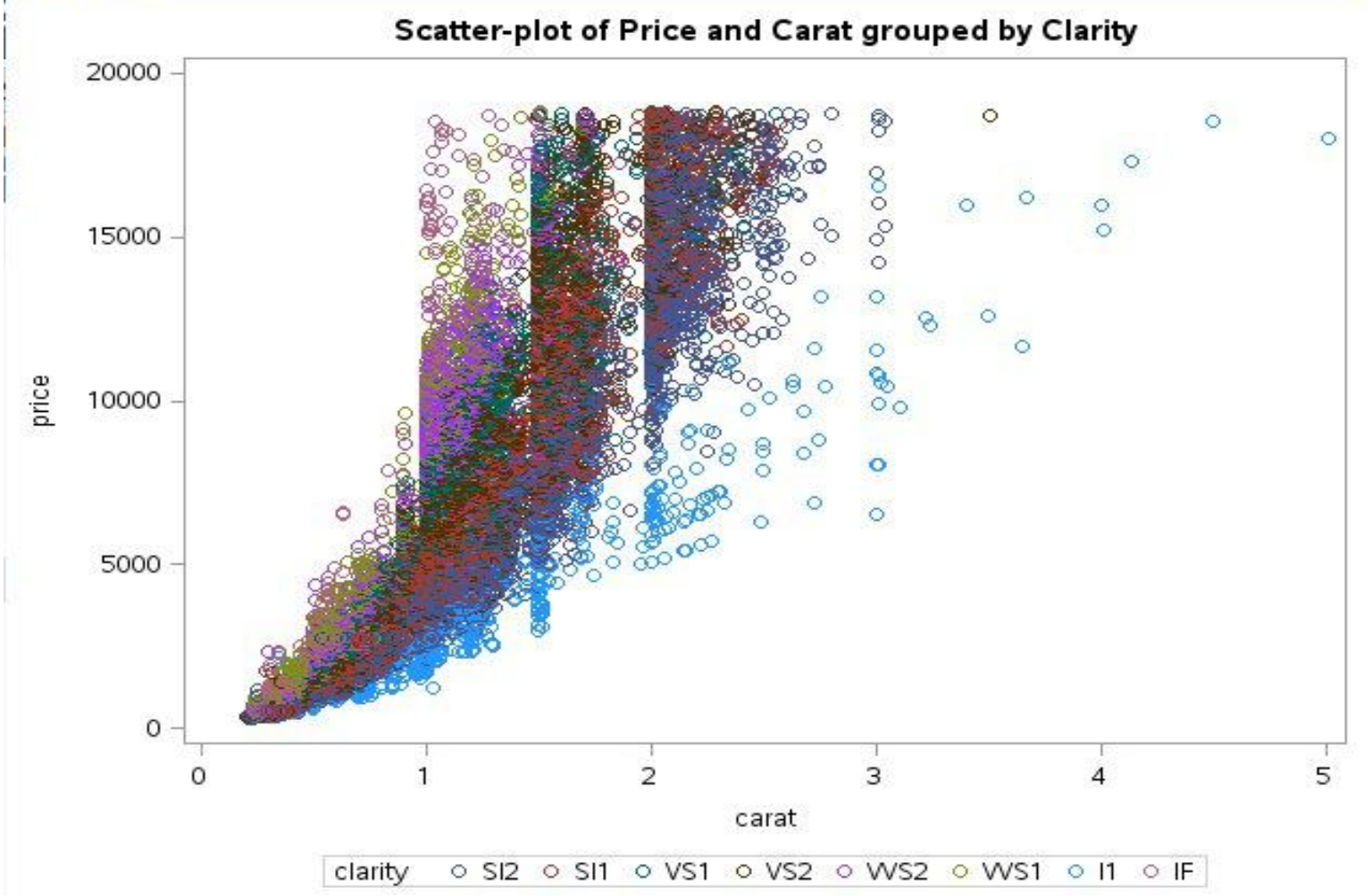
Data Visualization using SAS

```
42 title 'Scatter-plot of Price and Carat grouped by Clarity';
43 proc sgplot data=diamonds;
44 scatter x=carat y=price / group=clarity;
45 run;
```



Data Visualization using SAS

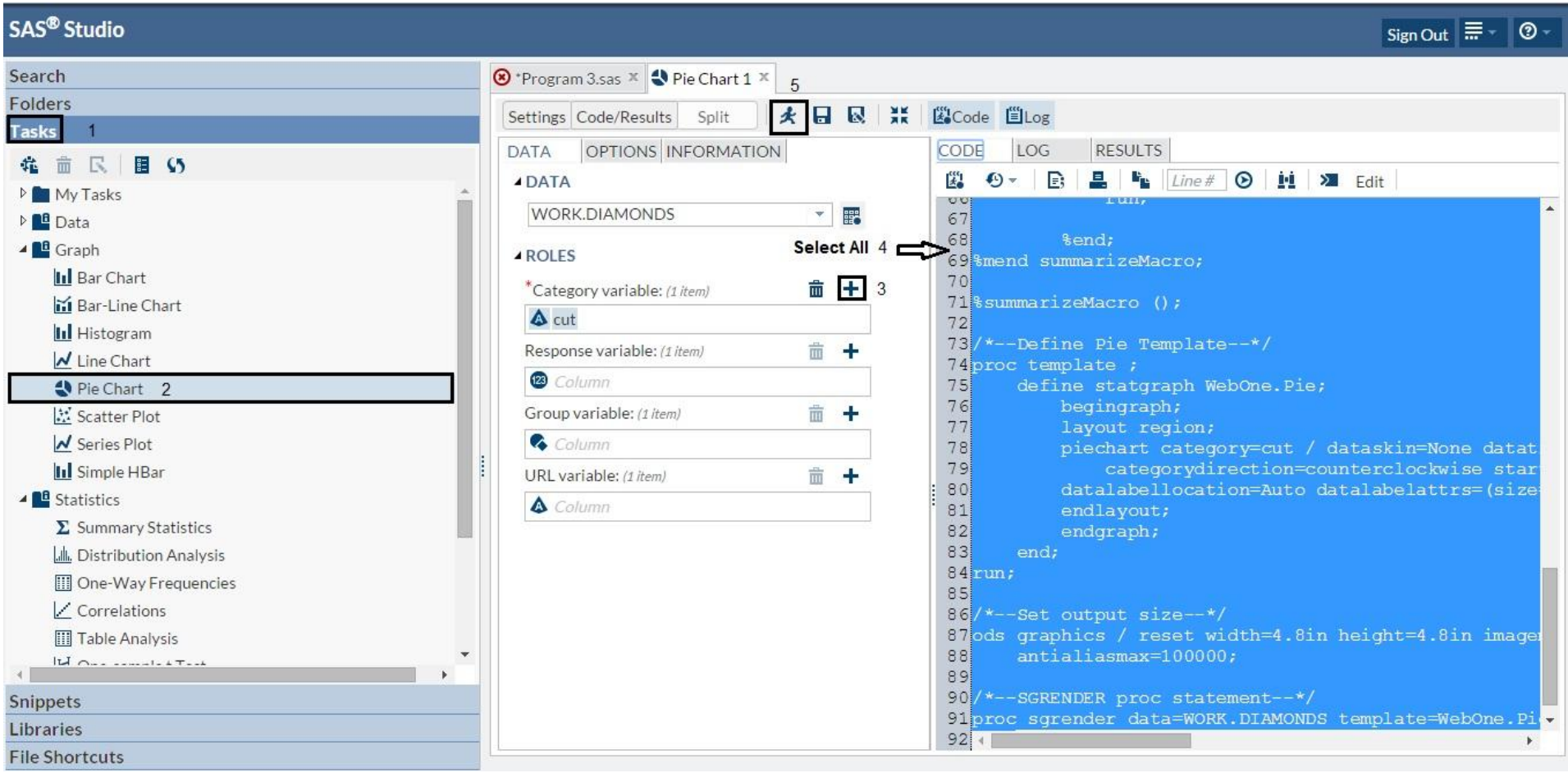
```
42 title 'Scatter-plot of Price and Carat grouped by Clarity';
43 proc sgplot data=diamonds;
44 scatter x=carat y=price / group=clarity;
45 run;
```



Data Visualization using SAS

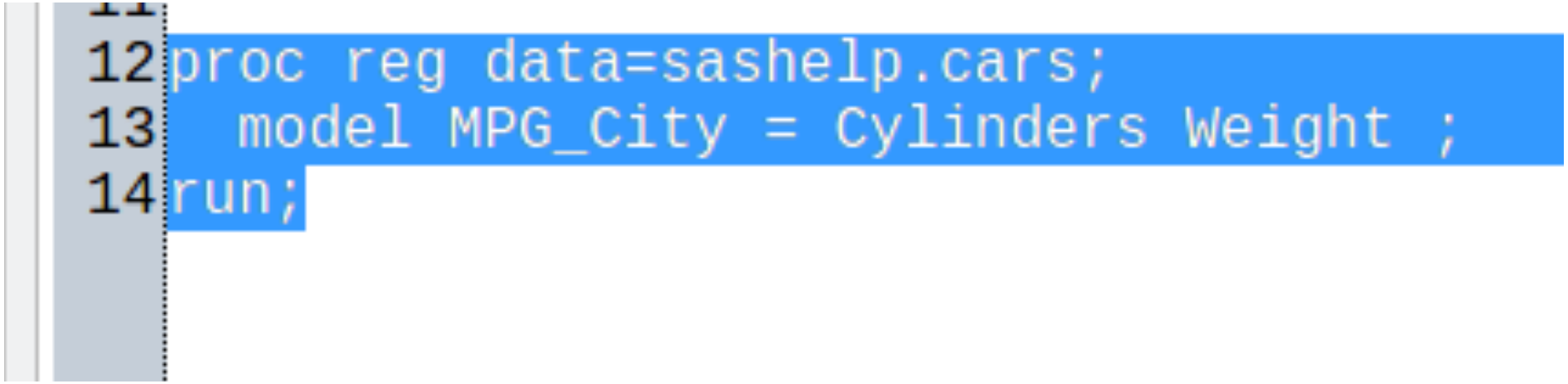
Use Tasks

see https://welcomedata.wordpress.com/2015/03/23/quick-pie-charts-in-sas-university-edition/



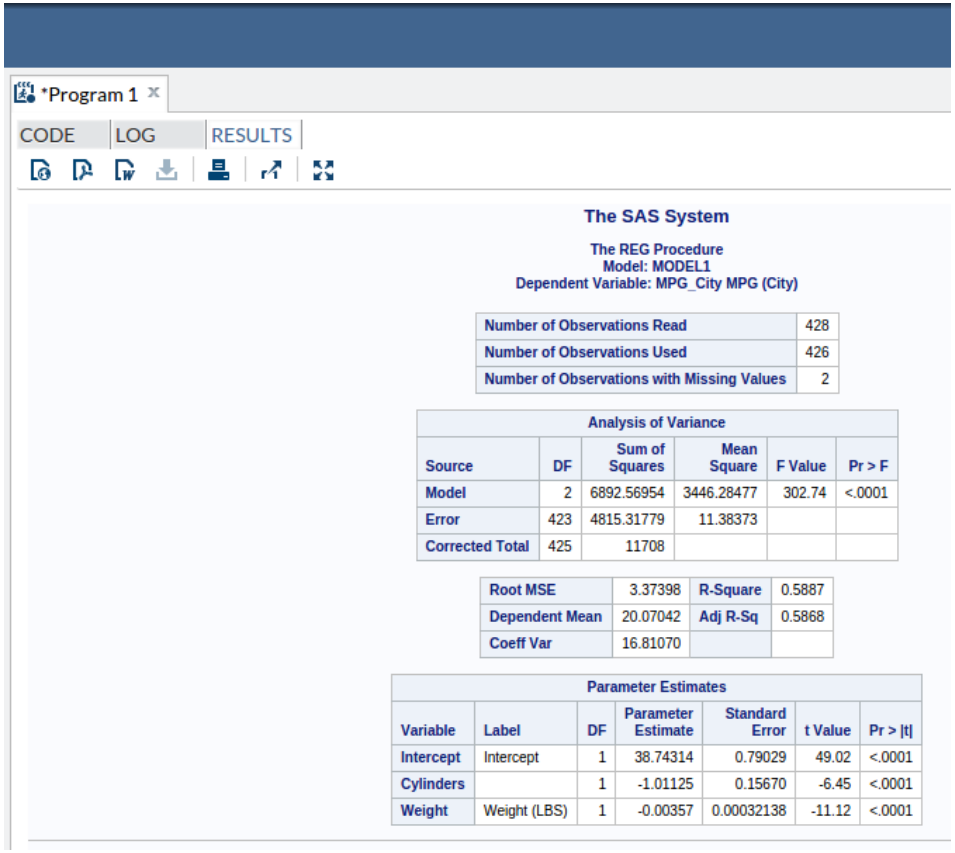
Modeling using SAS

```
proc reg data=sashelp.cars;
  model MPG_City = Cylinders Weight ;
run;
```



Modeling using SAS

```
proc reg data=sashelp.cars;
  model MPG_City = Cylinders Weight ;
run;
```



*Program 1

CODE | LOG | RESULTS

The SAS System

The REG Procedure
Model: MODEL1
Dependent Variable: MPG_City MPG (City)

Number of Observations Read	428
Number of Observations Used	426
Number of Observations with Missing Values	2

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	6892.56954	3446.28477	302.74	<.0001
Error	423	4815.31779	11.38373		
Corrected Total	425	11708			

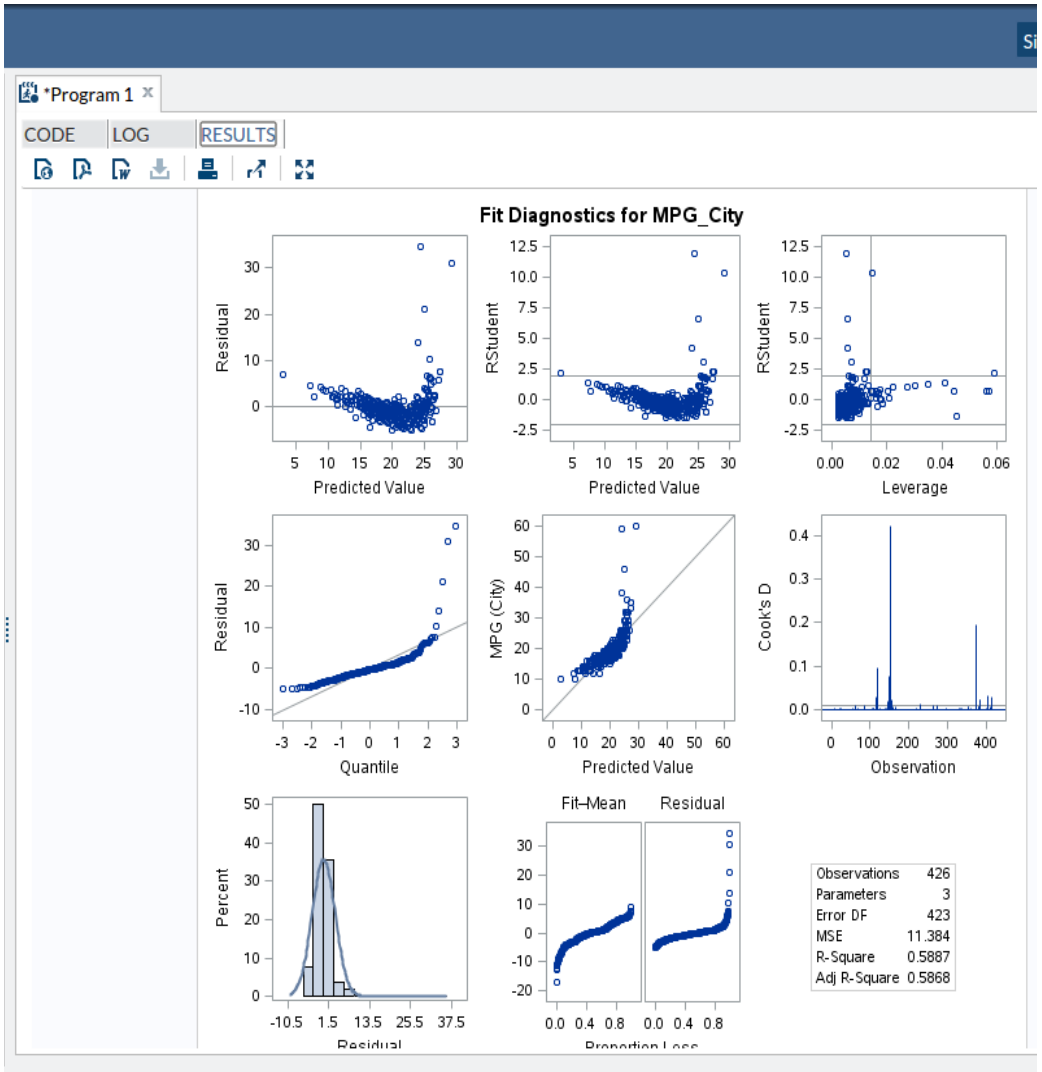
Root MSE	3.37398	R-Square	0.5887
Dependent Mean	20.07042	Adj R-Sq	0.5868
Coeff Var	16.81070		

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	38.74314	0.79029	49.02	<.0001
Cylinders		1	-1.01125	0.15670	-6.45	<.0001
Weight	Weight (LBS)	1	-0.00357	0.00032138	-11.12	<.0001

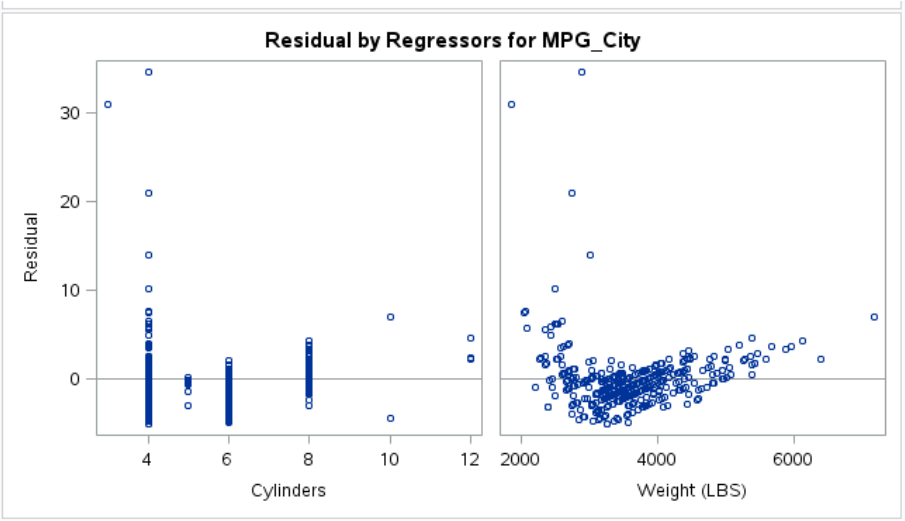
Modeling using SAS

```
proc reg data=sashelp.cars;
  model MPG_City = Cylinders Weight ;
run;
```



Modeling using SAS

```
proc reg data=sashelp.cars;
  model MPG_City = Cylinders Weight ;
run;
```



Modeling using SAS

Try this

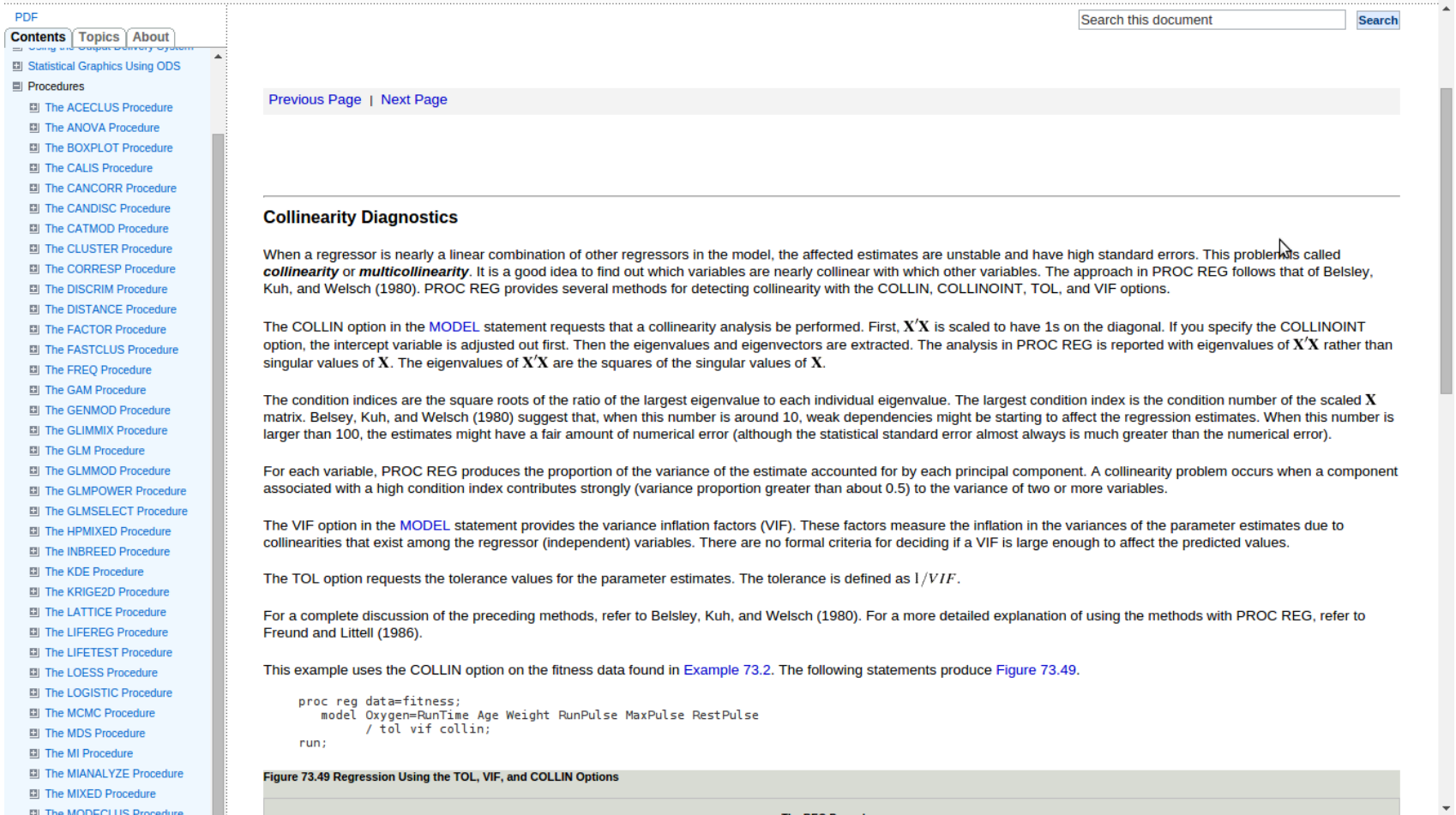
```
proc reg data=sashelp.iris;
  model SepalLength =      SepalWidth PetalLength       PetalWidth ;
run;
```

NOW TRY THIS

```
proc reg data=sashelp.iris;
  model SepalLength =      SepalWidth PetalLength       PetalWidth  /vif collin;
run;
```

Modeling using SAS : Multicollinearity

http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_reg_sect038.htm



PDF

Contents | Topics | About

- Statistical Graphics Using ODS
- Procedures
  - The ACECLUS Procedure
  - The ANOVA Procedure
  - The BOXPLOT Procedure
  - The CALIS Procedure
  - The CANCORR Procedure
  - The CANDISC Procedure
  - The CATMOD Procedure
  - The CLUSTER Procedure
  - The CORRESP Procedure
  - The DISCRIM Procedure
  - The DISTANCE Procedure
  - The FACTOR Procedure
  - The FASTCLUS Procedure
  - The FREQ Procedure
  - The GAM Procedure
  - The GENMOD Procedure
  - The GLIMMIX Procedure
  - The GLM Procedure
  - The GLMMOD Procedure
  - The GLMPOWER Procedure
  - The GLMSELECT Procedure
  - The HPMIXED Procedure
  - The INBREED Procedure
  - The KDE Procedure
  - The KRIGE2D Procedure
  - The LATTICE Procedure
  - The LIFEREG Procedure
  - The LIFETEST Procedure
  - The LOESS Procedure
  - The LOGISTIC Procedure
  - The MCMC Procedure
  - The MDS Procedure
  - The MI Procedure
  - The MIANALYZE Procedure
  - The MIXED Procedure

Search this document | Search

Previous Page | Next Page

Collinearity Diagnostics

When a regressor is nearly a linear combination of other regressors in the model, the affected estimates are unstable and have high standard errors. This problem is called ***collinearity*** or ***multicollinearity***. It is a good idea to find out which variables are nearly collinear with which other variables. The approach in PROC REG follows that of Belsley, Kuh, and Welsch (1980). PROC REG provides several methods for detecting collinearity with the COLLIN, COLLINOINT, TOL, and VIF options.

The COLLIN option in the MODEL statement requests that a collinearity analysis be performed. First, $\mathbf{X}'\mathbf{X}$ is scaled to have 1s on the diagonal. If you specify the COLLINOINT option, the intercept variable is adjusted out first. Then the eigenvalues and eigenvectors are extracted. The analysis in PROC REG is reported with eigenvalues of $\mathbf{X}'\mathbf{X}$ rather than singular values of $\mathbf{X}$. The eigenvalues of $\mathbf{X}'\mathbf{X}$ are the squares of the singular values of $\mathbf{X}$.

The condition indices are the square roots of the ratio of the largest eigenvalue to each individual eigenvalue. The largest condition index is the condition number of the scaled $\mathbf{X}$ matrix. Belsey, Kuh, and Welsch (1980) suggest that, when this number is around 10, weak dependencies might be starting to affect the regression estimates. When this number is larger than 100, the estimates might have a fair amount of numerical error (although the statistical standard error almost always is much greater than the numerical error).

For each variable, PROC REG produces the proportion of the variance of the estimate accounted for by each principal component. A collinearity problem occurs when a component associated with a high condition index contributes strongly (variance proportion greater than about 0.5) to the variance of two or more variables.

The VIF option in the MODEL statement provides the variance inflation factors (VIF). These factors measure the inflation in the variances of the parameter estimates due to collinearities that exist among the regressor (independent) variables. There are no formal criteria for deciding if a VIF is large enough to affect the predicted values.

The TOL option requests the tolerance values for the parameter estimates. The tolerance is defined as $1/VIF$.

For a complete discussion of the preceding methods, refer to Belsley, Kuh, and Welsch (1980). For a more detailed explanation of using the methods with PROC REG, refer to Freund and Littell (1986).

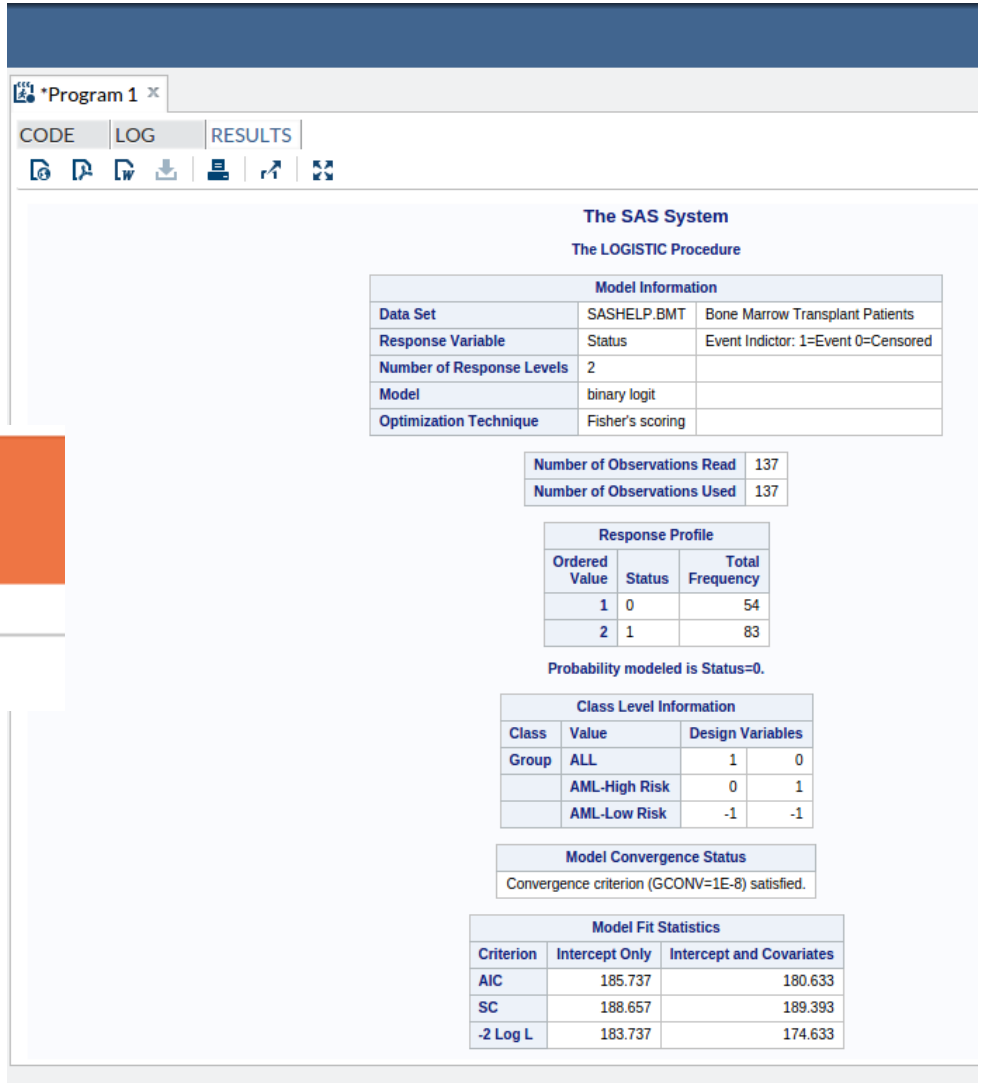
This example uses the COLLIN option on the fitness data found in Example 73.2. The following statements produce Figure 73.49.

```
proc reg data=fitness;
   model Oxygen=RunTime Age Weight RunPulse MaxPulse RestPulse
         / tol vif collin;
run;
```

Figure 73.49 Regression Using the TOL, VIF, and COLLIN Options

Modeling using SAS : PROC LOGISTIC

```
proc logistic data=sashelp.Bmt;
   class Group;
   model Status=Group;
run;
```



Modeling using SAS : PROC LOGISTIC

```
proc logistic data=sashelp.Bmt;
   class Group;
   model Status=Group;
run;
```

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	185.737	180.633
SC	188.657	189.393
-2 Log L	183.737	174.633

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	9.1039	2	0.0105
Score	8.9456	2	0.0114
Wald	8.6190	2	0.0134

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
Group	2	8.6190	0.0134

Analysis of Maximum Likelihood Estimates

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.5063	0.1850	7.4943	0.0062
Group	ALL	1	-0.0326	0.2682	0.0148	0.9031
Group	AML-High Risk	1	-0.6221	0.2726	5.2078	0.0225

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
Group ALL vs AML-Low Risk	0.503	0.215	1.175
Group AML-High Risk vs AML-Low Risk	0.279	0.117	0.662

Association of Predicted Probabilities and Observed Responses

Percent Concordant	48.1	Somers' D	0.283
Percent Discordant	19.8	Gamma	0.416
Percent Tied	32.0	Tau-a	0.136
Pairs	4482	c	0.642

Questions or Feedback

Email us at

info@decisionstats.org