Getting Started

Statistics 426: SAS Programming

Module 1

2021

Introduction to SAS

Foundation SAS has
* Reporting and graphing * Analytics * Visualization and discovery * Business solutions * User interface * Base SAS

Key terms

Key terms to be used in this course:
- SAS programs
- SAS datasets
- SAS files
- SAS libraries

SAS programs

A SAS program is a sequence of steps that the user submits for execution. Structure and components of SAS programs most often contain a DATA step and/or a PROC step (DATA for dataset creation and PROC for some sort of procedure).

All SAS steps begin with either a DATA statement or a PROC statement. SAS will detect the end of a step when it encounters one of the following: (1) A RUN statement (most of the time) (2) A QUIT statement (some of the time) (3) The beginning of another DATA or PROC step

DATA steps

DATA steps typically create or modify SAS data sets. They can also be used to produce custom designed reports. For example, you can use DATA steps to: (1) put your data into a SAS dataset (2) compute values (3) check for and correct errors in your data (4) produce new SAS data sets by subsetting, merging, and updating existing datasets

PROC steps

PROC (procedure) steps are pre-written routines that enable you to analyze and process the data in a SAS dataset and to present the data in the form of a report. PROC steps sometimes create new SAS data sets that contain the results of the procedure. PROC steps can list, sort, and summarize data.

For example, you can use PROC steps to (1) create a report that lists the data (2) produce descriptive statistics (3) create a summary report (4) produce plots and charts

Raw data (or SAS data set)→DATA step→SAS→dataset→PROC steps→Report

Characteristics of SAS Programs

SAS programs consist of SAS statements. A SAS statement has two important characteristics: (1) It usually begins with a SAS *keyword* (2) It always ends with a semicolon

A DATA step begins with a DATA statement, which begins with the keyword DATA. A PROC step begins with a PROC statement, which begins with the keyword PROC

Layout for SAS Programs

SAS statements are in free format. This means that: - they can begin and end anywhere on a line - one statement can continue over several lines - several statements can be on a line - Blanks or special characters separate "words" in a SAS statement - Although the SAS programs are format-free (unstructured), keeping programs a bit formatted will make them easier to read and debug (spaces, tabs, etc., of which I do not always do)

Note: You can specify SAS statements in uppercase or lowercase. SAS is not case-sensitive so if you input variables as uppercase letters, you can still call them with all lowercase letters. In most situations, text that is enclosed in quotation marks is case sensitive.

General form of DATA step with CARDS

The following will read in data that you either input manually or copy/paste. The CARDS statement is interchangeable with the DATALINES statement; fewer keystrokes involved with CARDS, but whatever. To view the data in the results window, use PROC PRINT

```
DATA datasetname;
INPUT var1 var2 var3$ ...;
CARDS;
<input data here>
;
RUN;
```

General form of DATA step with SET

SET allows to read in an existing SAS dataset as the basis of the new dataset being created. In other words, the new dataset specified in the DATA statement is created using an existing SAS dataset specified in the SET statement. Most often the reason to create a new dataset from an existing one is to subset it in some way that modifies it from the original. WHERE, KEEP, etc. statements will be discussed in coming modules

```
DATA datasetname;
SET input-datasetname;
WHERE where-expression;
KEEP variable-list;
LABEL variable1='Label'
variable2='Label'
... ;
FORMAT variable1 format.
variable2 format.
... ;
RUN;
```

General form of DATA step with INFILE

INFILE specifies an external file to read into SAS. The best file types are .csv and .txt files. Excel requires some extra SAS packages to be installed (purchased) to work correctly. Variables are assumed to be numeric unless otherwise specified in the INPUT statement with a `$` following the character variable name(s).

```
DATA libref.datasetname;
INPUT v1 v2 v3$ ... ;
INFILE 'address\filename.ext' DLM=',' firstobs=2;
RUN;
```

Processing SAS Programs

When you submit a SAS program, SAS begins reading the statements and checking them for errors. DATA and PROC statements signal the beginning of a new step. When SAS encounters a subsequent DATA, PROC, or RUN statement (for DATA steps and most procedures) or a QUIT statement (for some procedures), SAS stops reading statements and executes the previous step in the program. In our sample program, each step ends with a RUN statement.

The beginning of a new step (DATA or PROC) implies the end of the previous step. Though the RUN statement is not always required between steps in a SAS program, using it can make the SAS program easier to read and debug, and it makes the SAS log easier to read.

Log Messages

Each time a step is executed, SAS generates a log of the processing activities and the results of the processing. The SAS log collects messages about the processing of SAS programs and about any errors that occur. When SAS processes a program, information about it is written in the log messages shown below. Notice it generates separate sets of messages for each step in the program

Errors

Syntax errors occur when program statements do not conform to the rules of the SAS language. Examples include misspelled words, unmatched quotes (or mismatched), missing semicolons, or invalid operations.

When SAS encounters a syntax error, SAS prints a warning of an error message to the log

SAS dataset and variable names

Dataset and variable names - Can be 32 characters long - Must start with a letter or an underscore (no dashes); subsequent characters can be letters, numbers or underscores - Can be upper case, lower case, or mixed case - Are *not* case sensitive

You can use special characters in variable or dataset names only if you put the name in quotation marks followed immediately by the letter n: `'n'` or `"n"`

Tutorial

Even to do simple calculations, such as subtracting and adding, requires a DATA step and a PROC PRINT to see the results of the calculations. Realistically, a calculator would be easier to use for minor calculations that do not go into a report.

But you can still do it, so we will at least just once

Operations

SAS will always remember the order of operations, even if you do not. PEMDAS... please excuse my dear Aunt Sally. :-) Basic arithmetic functions work rather intuitively, including use of negative numbers. Exponentiation in SAS is done with a double asterisk `**`, not a caret ^ as many other programs do. SAS... double asterisk `**` for exponentiation

Calculations to log instead of creating a dataset

Because we are not creating a DATA set, if we want to actually see these variables, we have to use the PUT statement. And that just tells SAS to put these in the log so that we can view them. This is a good thing for when you are doing little calculations and such, but so is a calculator. :-)

```
DATA _NULL_;
a=mean(1,2,3,4,5);
b=exp(3);
c=var(10,20,30);
d=poisson(1,2);
PUT a b c d;
RUN;
```

Log

Another _NULL_ example

```
DATA _null_;
a=5+-2;
b=5*7;
c=5/7;
d=5**7;
e=5+7*3-12/4-6;
f=(5**3+7)*3-(12/4-6)**2;
PUT a b c d e f;
RUN;
```

Log

Add a comment to program

```
* This is a comment in SAS, it will print in the log but will not be run as a command,
which is good because my babbling will cause errors;
```

```
*/ This is also a comment in SAS, it will print in the log only/*
```

The first is my preferred method, but it matters not which one you use

Libraries

Every SAS file is stored in a SAS library, which is a collection of SAS files. A SAS data library is the highest level of organization for information within SAS. SAS libraries have different implementations depending on your operating environment, but a library usually corresponds to the level of organization that your host operating system uses to access and store files. In some operating environments, a library is a physical collection of files. In others, the files are only logically related. For example, in the Windows and UNIX environments, a library is typically a group of SAS files in the same folder or directory.

Libraries

Storing Files Temporarily or Permanently:
Depending on the library name that you use when you create a file, you can store SAS files temporarily or permanently.

Storing files temporarily:
If you do not specify a library name when you create a file (or if you specify the library named Work), the file is stored in the temporary SAS data library. When you end the session, the temporary library and all of its files are deleted.

Storing files permanently

To store files permanently in a SAS data library, you specify a library name other than the default library name Work. For example, by specifying the library name Clinic when you create a file, you specify that the file is to be stored in a permanent SAS data library until you delete it.

Referencing SAS Files

To reference a permanent SAS dataset in your SAS programs, you use a two-level name:
`libref.filename`

`libref`: name of the SAS data *library* that contains the file
`filename`: name of the file itself

LIBREF examples

The example given shows the name of the file (admit) and the library where it is stored (clinic)

```
DATA clinic.admit2;
SET clinic.admit;
RUN;
```

LIBREF examples

Referencing Temporary SAS Files

```
DATA work.admit2;
SET clinic.admit;
RUN;
```

To reference temporary SAS files, you can specify the default libref Work, a period, and the filename. For example, the two-level name Work.admit2 references the SAS dataset named admit2 that is stored in the temporary SAS library Work.

```
DATA admit2;
SET clinic.admit;
RUN;
```

LIBREF logistics

Alternatively, you can use a one-level name (the filename only) to reference a file in a temporary SAS library. When you specify a one-level name, the default libref Work is assumed. For example, the one-level name admit2 also references the SAS dataset named admit2 that is stored in the temporary SAS library Work. If the USER library is assigned, SAS uses the User library rather than the Work library for one-level names. User is a permanent library.

Referencing Files

By default, SAS defines several libraries for you - Sashelp is a permanent library that contains sample data and other files that control how SAS works at your site. This is a read-only library - Sasuser is a permanent library that contains SAS files in the Profile catalog that store your personal settings, also a convenient place to store your own files - Work is a temporary library for files that do not need to be saved from session to session

You can also define additional libraries. In fact, often the first step in setting up your SAS session is to define the libraries. To define a library, you assign a library name (a libref) to it and specify a path, such as a directory path. You will use the libref as the first part of the file's two-level name (libref.filename) to reference the file within the library. You can use programming statements to assign library names.

Assigning Librefs

To define libraries, you can use a LIBNAME statement. You can store the LIBNAME statement with any SAS program so that the SAS data library is assigned each time the program is submitted

```
LIBNAME libref 'SAS-data-library';
```

Where:
`libref` is 1 to 8 characters long, begins with a letter or underscore, and contains only letters, numbers, or underscores
`SAS-data-library` is the name of a SAS data library in which SAS data files are stored

The specification of the physical name of the library differs by operating environment

Rules for SAS Names

SAS dataset names (1) can be 1 to 32 characters long (2) must begin with a letter (A–Z, either upper- or lower-case) or an underscore (_) (3) can continue with any combination of numbers, letters, or underscores

Examples of valid dataset name include `Payroll`, `LABDATA1995_1997`, `_EstimatedTaxPayments3`

SAS Data Sets

Data sets are one type of SAS file. There are other types of SAS files (such as catalogs). For many procedures and for some DATA step statements, data must be in the form of a SAS dataset to be processed.

Variable creation

Assignment statements are used in the DATA step to update existing variables or to create new ones. You can perform mathematical operations on variables with other variables or constants.

General form:
`variable = expression;`

Create dataset `xvar`

Finally!

```
DATA xvar;
INPUT x @@;
CARDS;
3 -2 4 7 5 -10
;
RUN;
PROC PRINT data=xvar;
RUN;
```

PROC PRINT

The SAS System

Obs	x
1	3
2	-2
3	4
4	7
5	5
6	-10

Log

```
NOTE: SAS initialization used:
      real time           2.50 seconds
      cpu time            1.09 seconds

1     DATA xvar;
2     INPUT x @@;
3     CARDS;

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.XVAR has 6 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.12 seconds
      cpu time            0.01 seconds


5     ;
6     RUN;
7     PROC PRINT data=xvar;
NOTE: Writing HTML Body file: sashtml.htm
8     RUN;

NOTE: There were 6 observations read from the data set WORK.XVAR.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.76 seconds
      cpu time            0.20 seconds
```

Create new variables

```
DATA xyz;
SET xvar;
y=4;
x+y;
x*y;
x**y;
z=x+y;
;
PROC PRINT data=xyz;
RUN;
```

PROC PRINT

The SAS System

Obs	x	y	z
1	7	4	11
2	2	4	6
3	8	4	12
4	11	4	15
5	9	4	13
6	-6	4	-2

Log

```
9    DATA xyz;
10   SET xvar;
11   y=4;
12   x+y;
13   x*y;
14   x**y
14 !     ;
15   z=x+y;
16   ;

NOTE: There were 6 observations read from the data set WORK.XVAR.
NOTE: The data set WORK.XYZ has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           3.34 seconds
      cpu time            0.12 seconds


17   PROC PRINT data=xyz;
18   RUN;

NOTE: There were 6 observations read from the data set WORK.XYZ.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

Intro to basic graphs

SAS is great for computing but is also great at visualizations, even the most basic ones. There are many (many!) options and such but we are starting with the very basic, default graphs for a module or two

Interest I

Scenario: you have $1000 to save and decide on a CD that gets 5% annual interest, to be reinvested back into the total every year. From that we get:

$$P(t) = P_0(1+r)^t$$

$P(t)$: final amount after time t
P_0: starting amount
r: rate
t: time (years)

Interest II

Define time variable below; time is 10 years. More later on the DO statement in coming modules

```
DATA time;
DO t=0 to 10;
OUTPUT; END;
RUN;
PROC PRINT data=time;
```

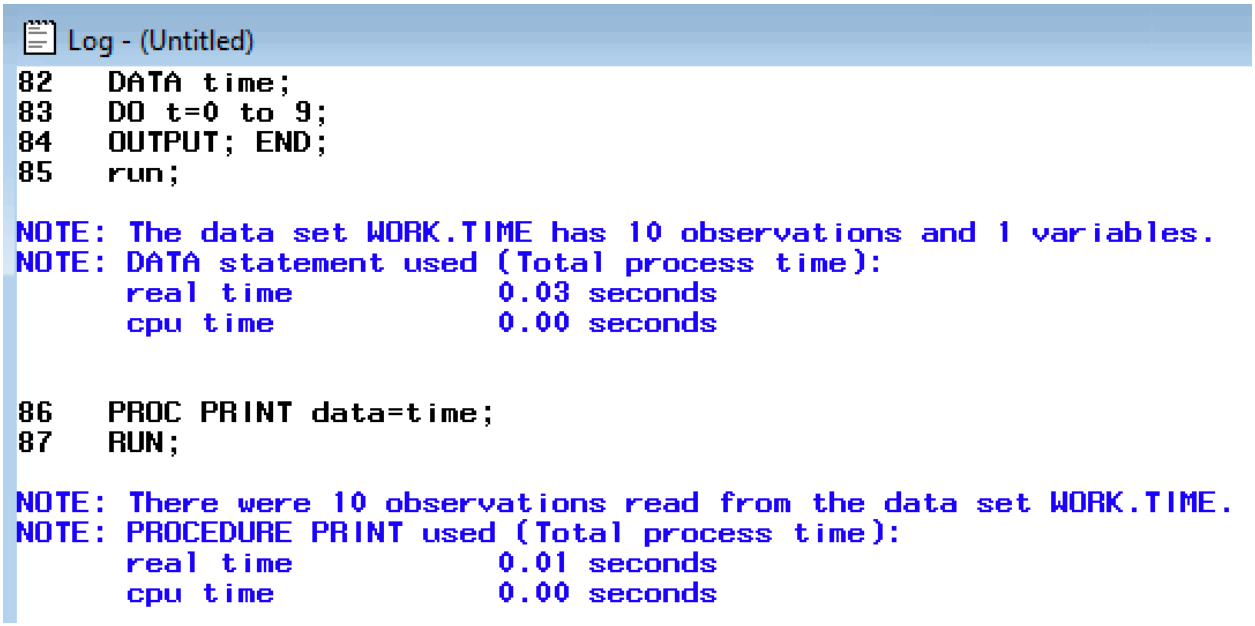
```
RUN;
```

PROC PRINT

The SAS System

Obs	t
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9

Log



Interest III

Using the created time variable t to make calculations for each year

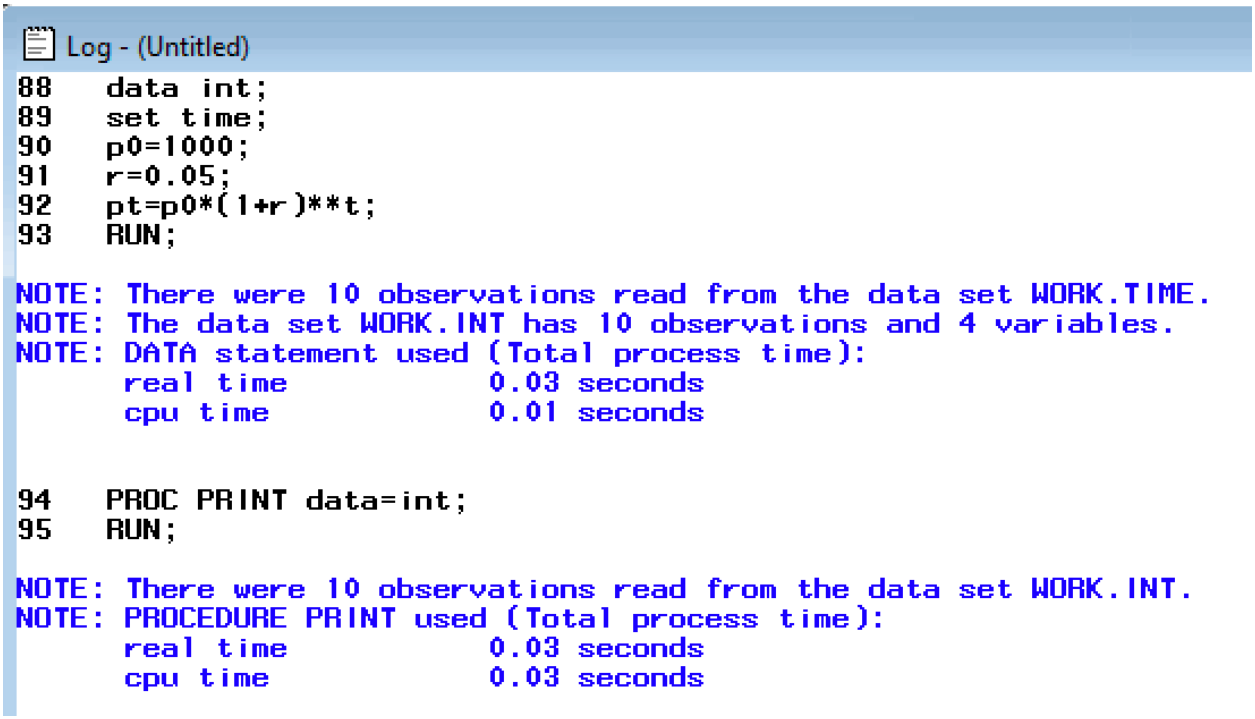
```
DATA int;
SET time;
p0=1000;
r=0.05;
pt=p0*(1+r)**t;
RUN;
PROC PRINT data=int;
RUN;
```

PROC PRINT

The SAS System

Obs	t	p0	r	pt
1	0	1000	0.05	1000.00
2	1	1000	0.05	1050.00
3	2	1000	0.05	1102.50
4	3	1000	0.05	1157.63
5	4	1000	0.05	1215.51
6	5	1000	0.05	1276.28
7	6	1000	0.05	1340.10
8	7	1000	0.05	1407.10
9	8	1000	0.05	1477.46
10	9	1000	0.05	1551.33

Log



Plotting using PROC SGPLOT

General form of `PROC SGPLOT`

```
PROC SGPLOT data=datasetname;
SCATTER x=x y=y;
RUN;
```

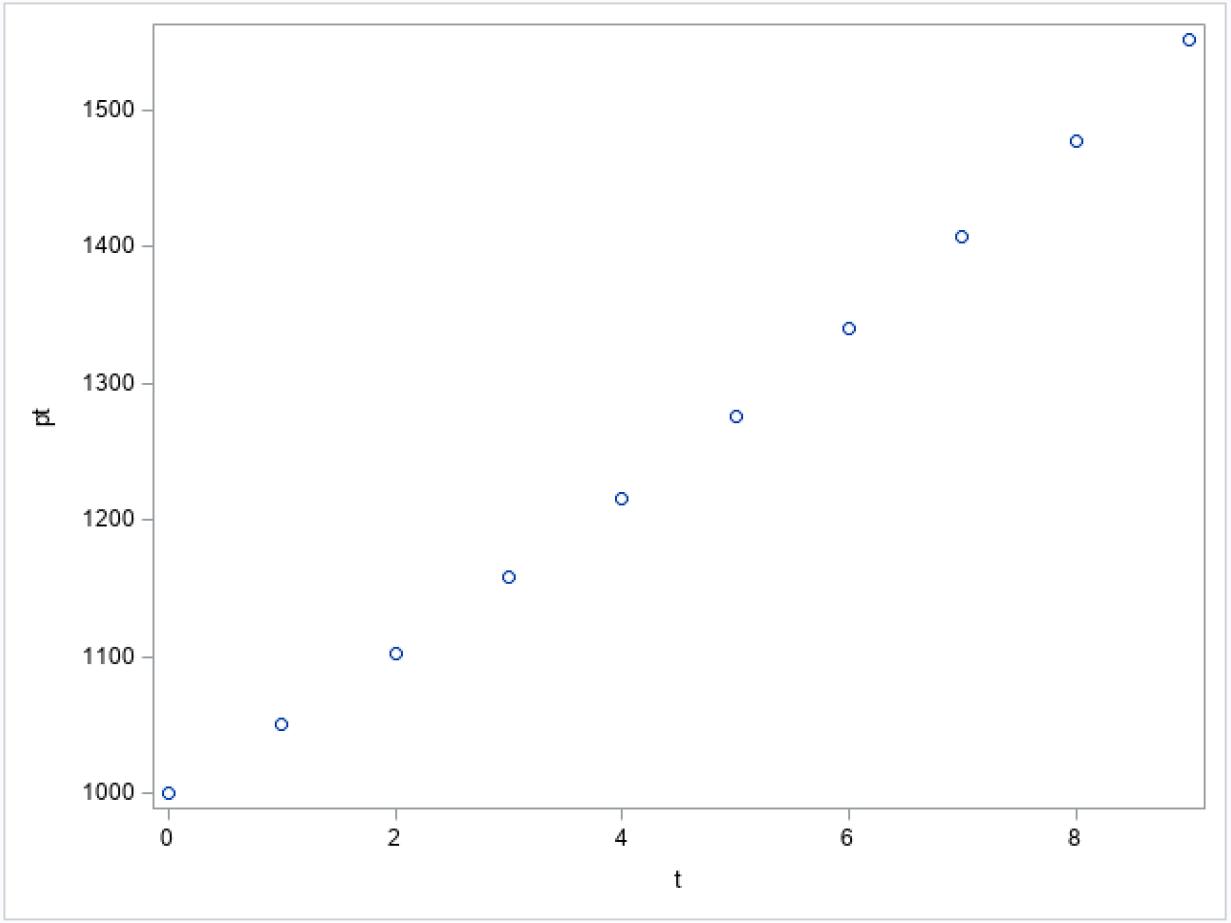
x: x values
y: y values

The first plot

A scatterplot with time t on the x-axis and $P(t)$ values on y-axis with the SCATTER option in PROC SGPLOT

```
PROC SGPLOT data=int;
SCATTER x=t y=pt;
RUN;
```

PROC SGPLOT



Log

Log - (Untitled)

```
96    PROC SGPLOT data=int;
97    SCATTER x=t y=pt;
98    RUN;

NOTE: PROCEDURE SGPLOT used (Total process time):
      real time           0.32 seconds
      cpu time            0.04 seconds

NOTE: There were 10 observations read from the data set WORK.INT.
```

A Room With a Moose!

Real data here on moose density and wolves, specifically the kill rate defined as the average number of moose killed per wolf per 100 days. The moose density is the average number of moose per 1000 km^2.

This time we will manually enter the data. In the future, I would most likely use a datafile and read it in rather than type all values. But here goes!

A Room With a Moose II

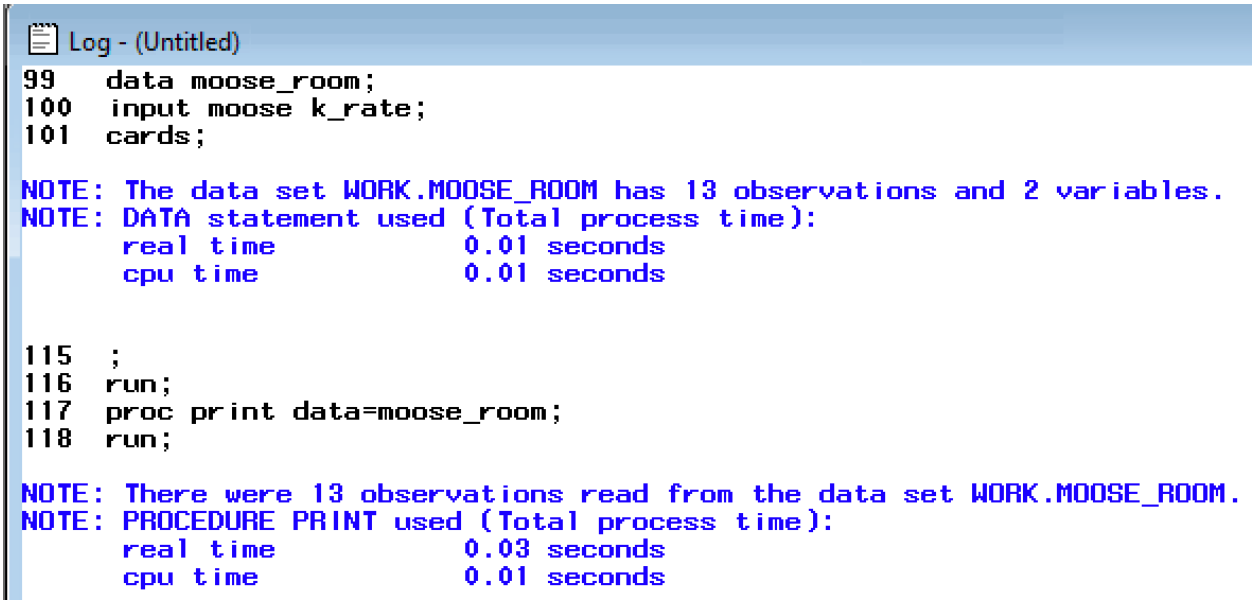
```
data moose_room;
input moose k_rate;
cards;
.17 .37
.23 .47
.23 1.9
.26 2.04
.37 1.12
.42 1.74
.66 2.78
.8 1.85
1.11 1.88
1.3 1.96
1.41 2.44
1.73 2.81
2.49 3.75
;
run;
proc print data=moose_room;
run;
```

PROC PRINT

The SAS System

Obs	moose	k_rate
1	0.17	0.37
2	0.23	0.47
3	0.23	1.90
4	0.26	2.04
5	0.37	1.12
6	0.42	1.74
7	0.66	2.78
8	0.80	1.85
9	1.11	1.88
10	1.30	1.96
11	1.41	2.44
12	1.73	2.81
13	2.49	3.75

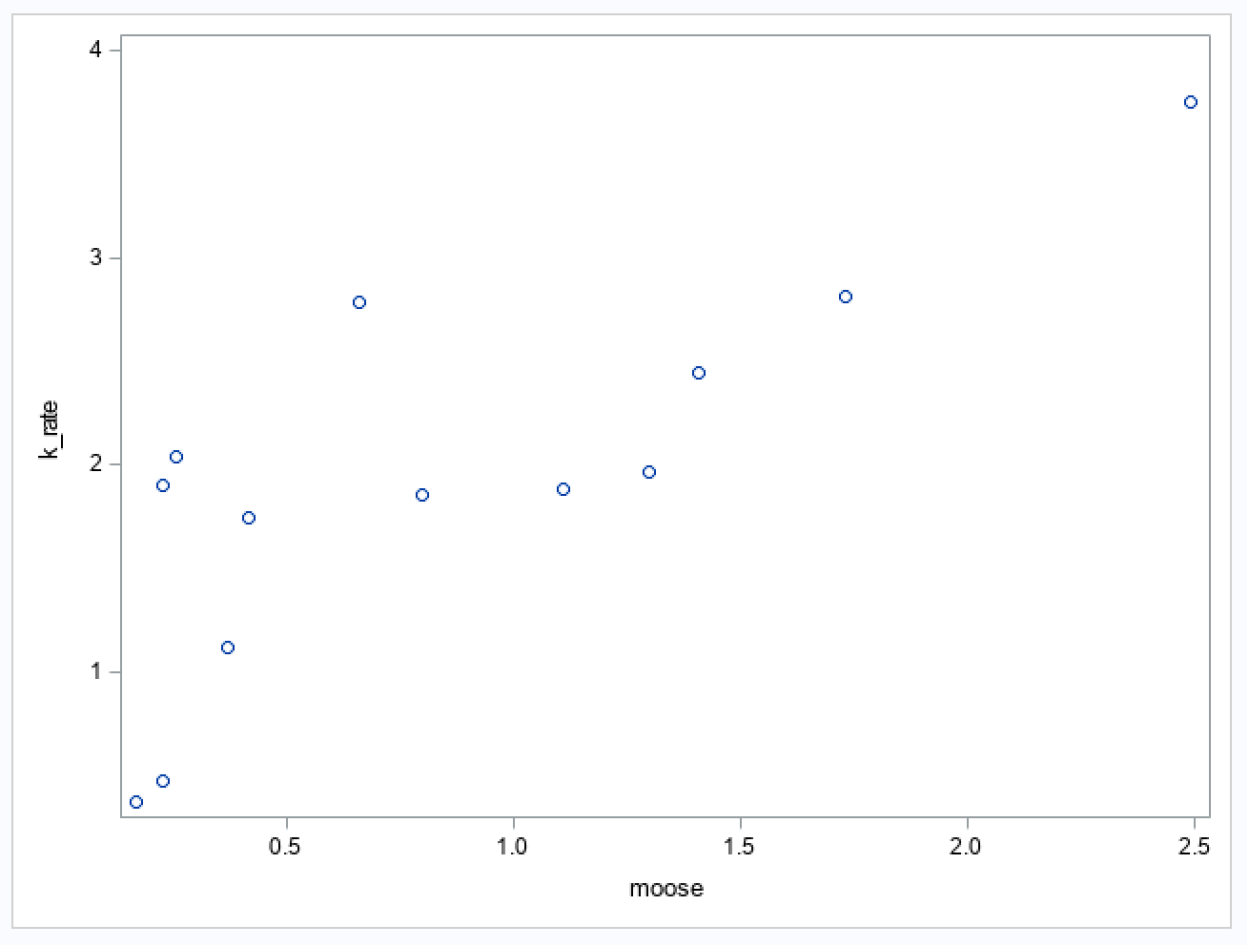
Log



A Room With a Moose III

```
proc sgplot data=moose_room;
scatter x=moose y=k_rate;
run;
```

Scatterplot with points



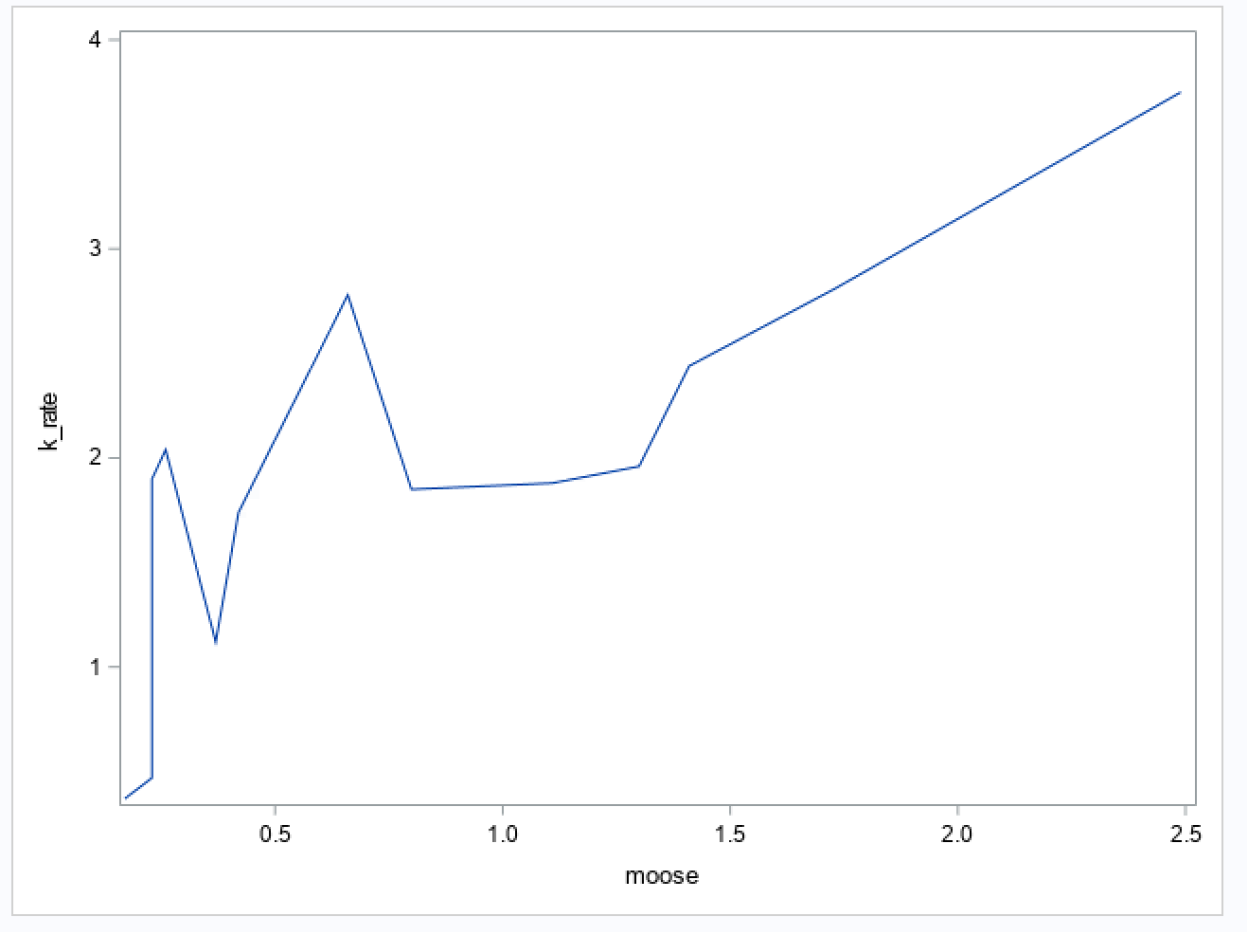
Log

```
Log - (Untitled)
19    proc sgplot data=moose_room;
20    scatter x=moose y=k_rate;
21    run;

NOTE: Writing HTML Body file: sashtml.htm
NOTE: PROCEDURE SGPLOT used (Total process time):
      real time           1.93 seconds
      cpu time            0.78 seconds

NOTE: There were 13 observations read from the data set WORK.MOOSE_ROOM.
```

A Room With a Moose IV

```
proc sgplot data=moose_room;
series x=moose y=k_rate;
run;
```

Lines plot



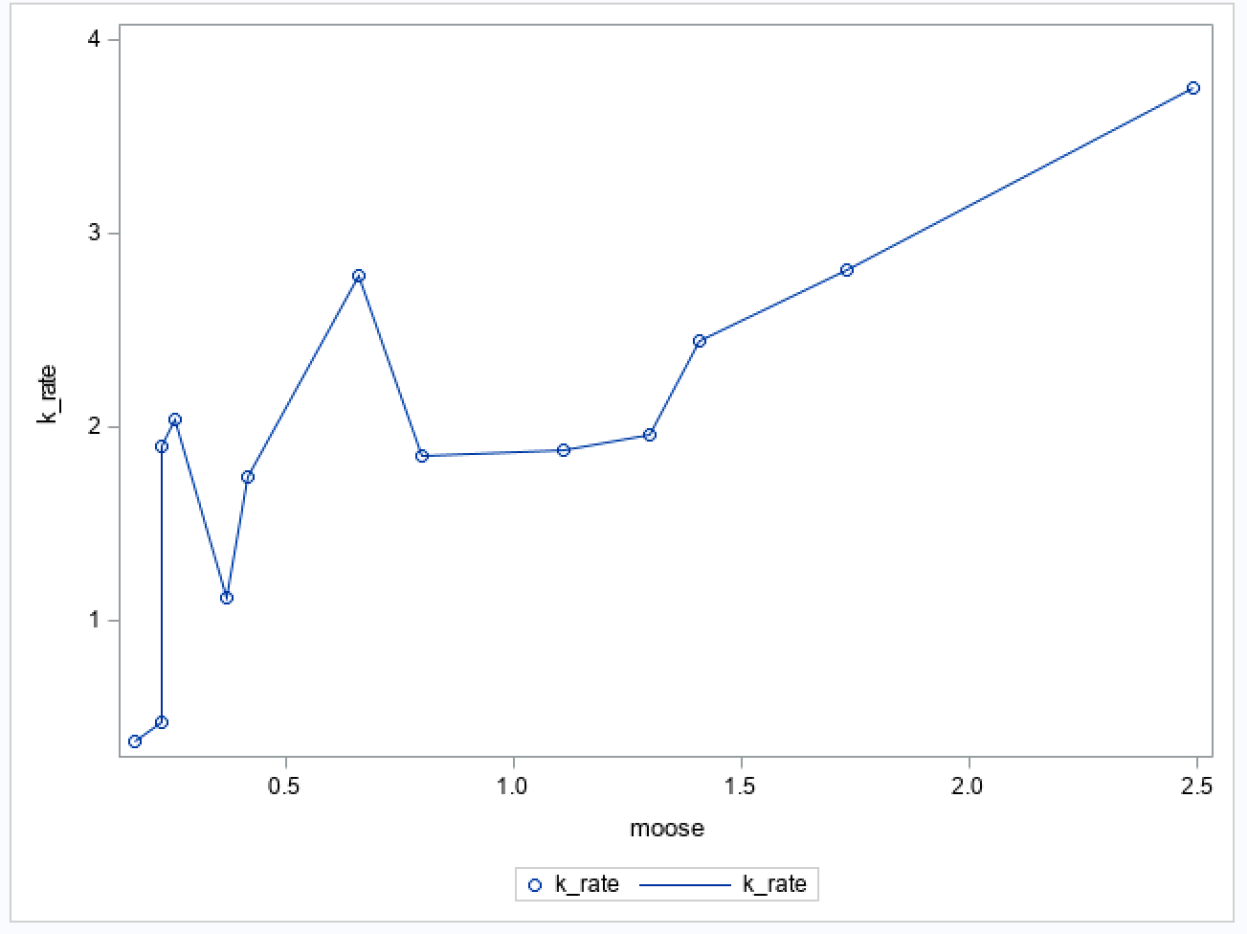
Log

```
22   proc sgplot data=moose_room;
23   series x=moose y=k_rate;
24   run;

NOTE: PROCEDURE SGPLOT used (Total process time):
      real time           0.32 seconds
      cpu time            0.06 seconds

NOTE: There were 13 observations read from the data set WORK.MOOSE_ROOM.
```

A Room With a Moose V

```
proc sgplot data=moose_room;
scatter x=moose y=k_rate;
series x=moose y=k_rate;
run;
```

Scatterplot with line



Log

Log - (Untitled)

```
NOTE: There were 13 observations read from the data set WORK.MOOSE_ROOM.

25    proc sgplot data=moose_room;
26    scatter x=moose y=k_rate;
27    series x=moose y=k_rate;
28    run;

NOTE: PROCEDURE SGPLOT used (Total process time):
      real time           0.40 seconds
      cpu time            0.04 seconds

NOTE: There were 13 observations read from the data set WORK.MOOSE_ROOM.
```

A Room With a Moose VI

We are now going to look at the relationship between the feeding rate of an average wolf and the supply of moose. If you look at the graph, the average would be a line that starts at zero, increases steeply to 2, then increases at a decreasing rate to almost a flat horizontal line at the maximum. Ecologists have expressed

this in a mathematical model, an equation that captures the essential expected relationship under "ideal" circumstances (only moose varies, other factors are fixed).

$$k = \frac{am}{b+m}$$

Where:
k: kill rate of an average predator
m: supply of prey (moose)
a , b: constants obtained from a curve fitting model (later!)

A Room With a Moose VII

Suppose we obtain the curve line parameter estimates and $a = 3.37$ and $b = 0.47$, now we will graph the points with a curve using both SCATTER and SERIES statements. The SCATTER we have seen; just the data points graphed in a scatterplot. SERIES within the same PROC SGPLOT as SCATTER will add a series (a line, trendline, regression line, etc. to the scatterplot). More details covered later

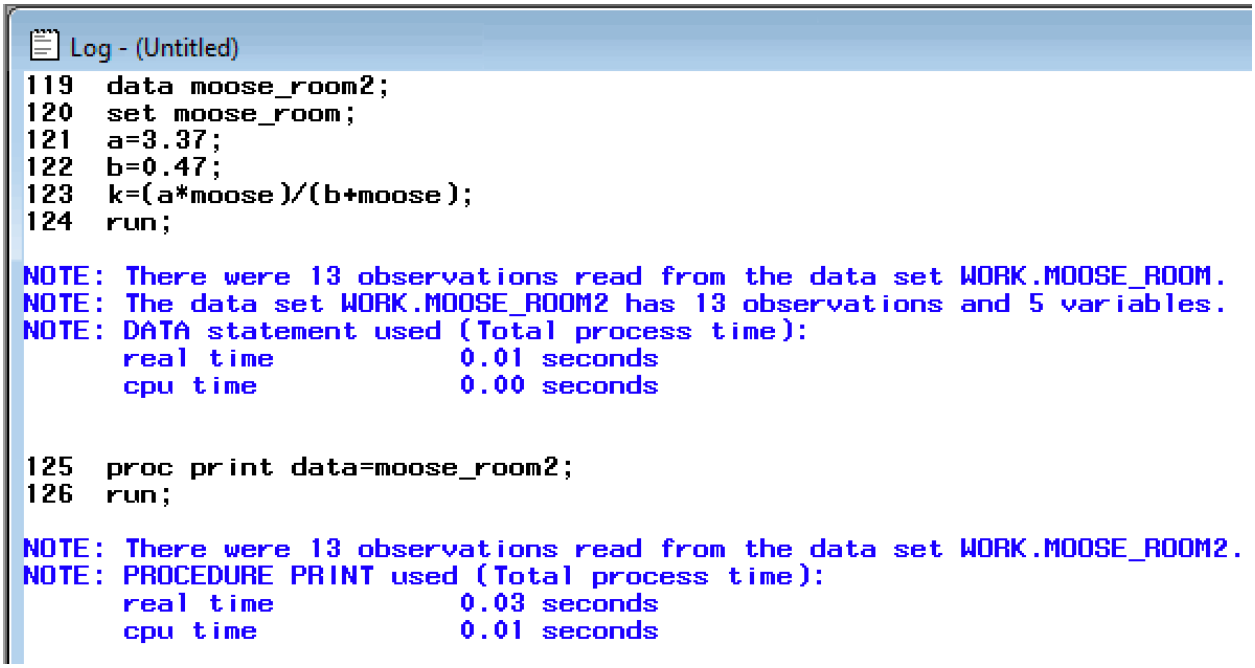
```
data moose_room2;
set moose_room;
a=3.37;
b=0.47;
k=(a*moose)/(b+moose);
run;
proc print data=moose_room2;
run;
```

PROC PRINT

The SAS System

Obs	moose	k_rate	a	b	k
1	0.17	0.37	3.37	0.47	0.89516
2	0.23	0.47	3.37	0.47	1.10729
3	0.23	1.90	3.37	0.47	1.10729
4	0.26	2.04	3.37	0.47	1.20027
5	0.37	1.12	3.37	0.47	1.48440
6	0.42	1.74	3.37	0.47	1.59034
7	0.66	2.78	3.37	0.47	1.96832
8	0.80	1.85	3.37	0.47	2.12283
9	1.11	1.88	3.37	0.47	2.36753
10	1.30	1.96	3.37	0.47	2.47514
11	1.41	2.44	3.37	0.47	2.52750
12	1.73	2.81	3.37	0.47	2.65005
13	2.49	3.75	3.37	0.47	2.83490

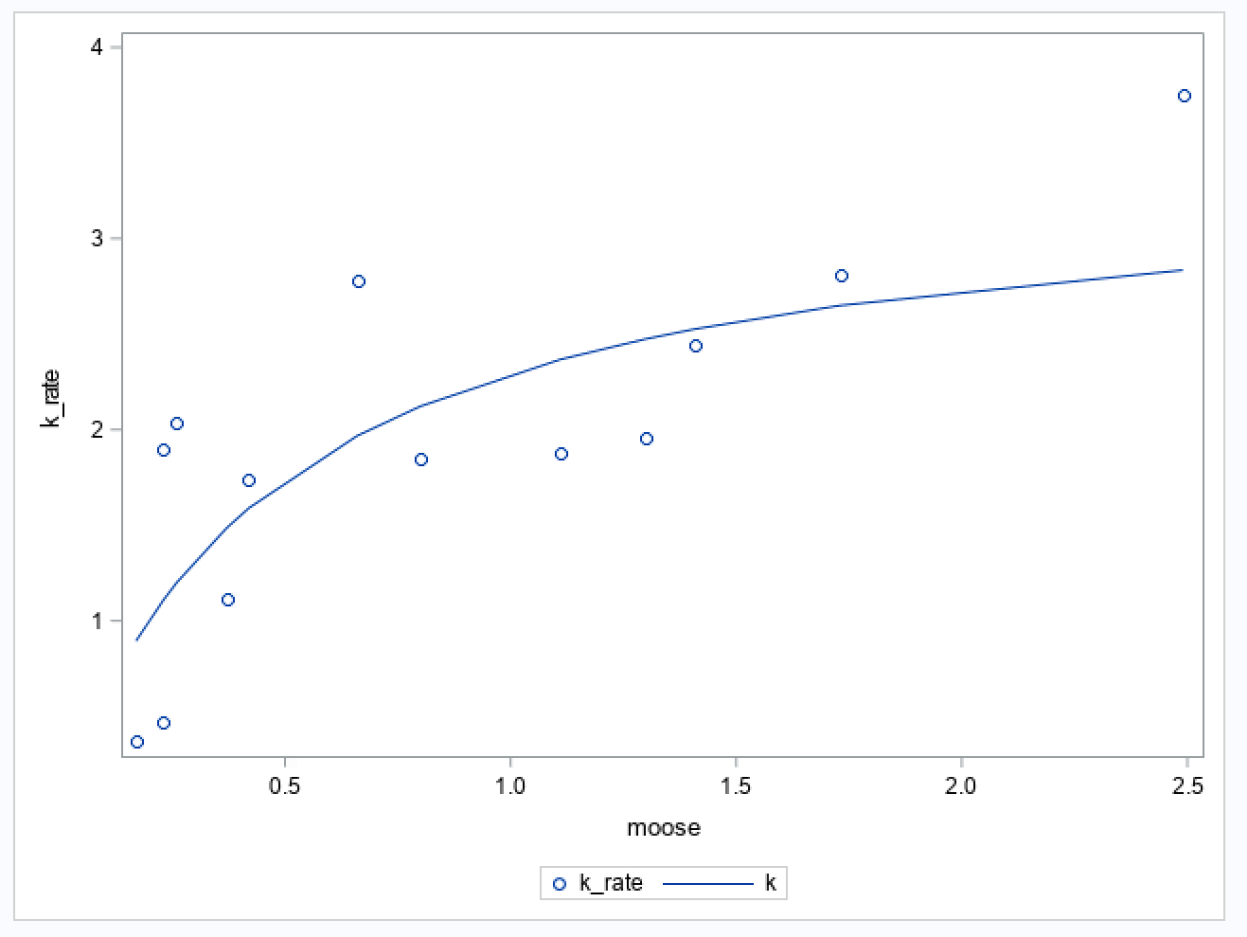
Log



A Room With a Moose VIII

```
proc sgplot data=moose_room2;
scatter x=moose y=k_rate;
series x=moose y=k;
run;
```

Scatterplot with line overlay of model



Log

```
Log - (Untitled)

NOTE: PROCEDURE SGPLOT used (Total process time):
      real time           0.34 seconds
      cpu time            0.04 seconds

NOTE: There were 13 observations read from the data set WORK.MOOSE_ROOM2.
```

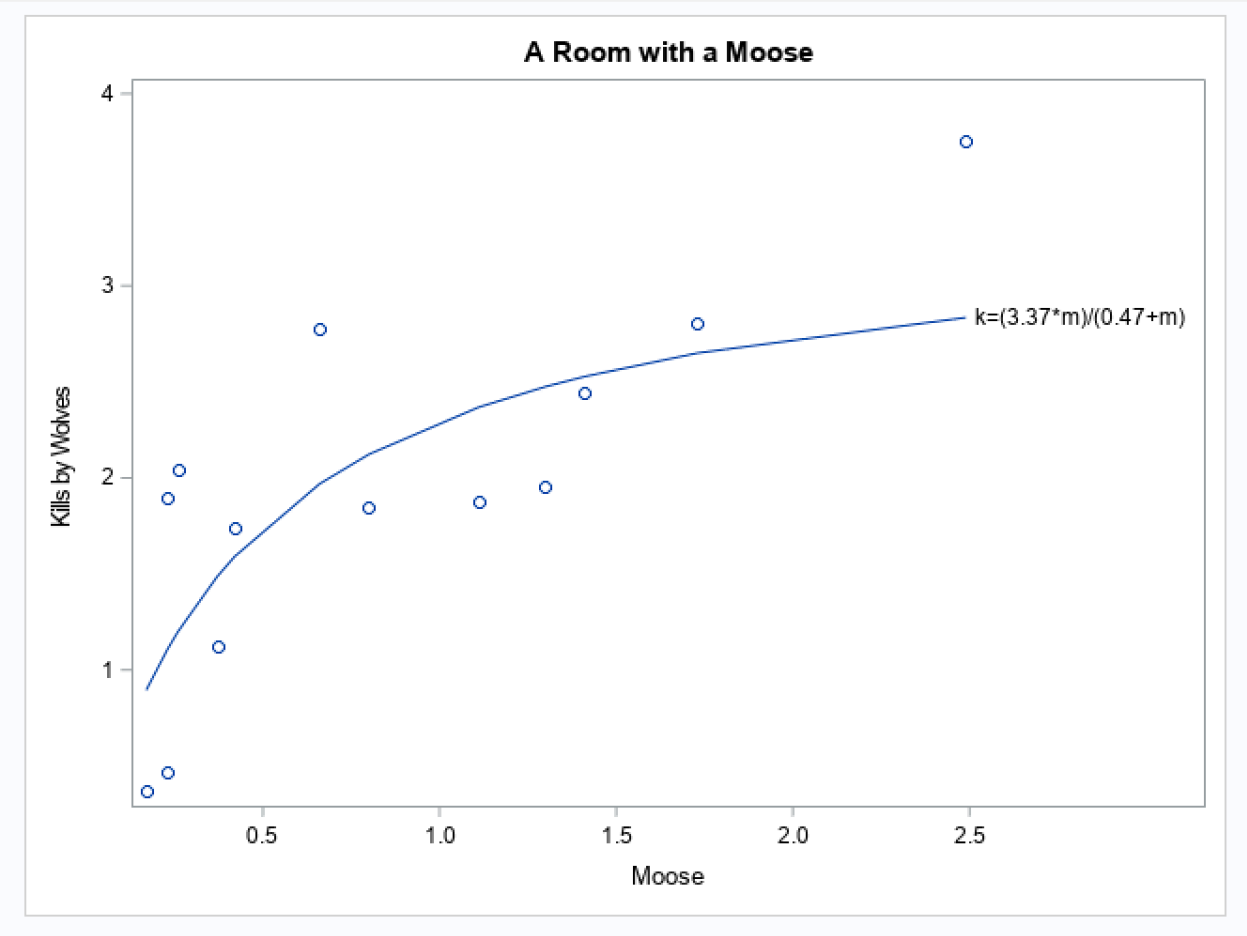
(A secret room with a moose)

With some extra bells and whistles that we will cover later

```
proc sgplot data=moose_room2;
title 'A Room with a Moose';
scatter x=moose y=k_rate;
series x=moose y=k / curvelabel="k=(3.37*m)/(0.47+m)";
xaxis Label='Moose';
yaxis Label='Kills by Wolves';
```

```
run;
title;
```

Secret moose plot



Log

Log - (Untitled)

```
131  proc sgplot data=moose_room2;
132  title 'A Room with a Moose';
133  scatter x=moose y=k_rate;
134  series x=moose y=k / curvelabel="k=(3.37*m)/(0.47+m)";
135  xaxis Label='Moose';
136  yaxis Label='Kills by Wolves';
137  run;

NOTE: PROCEDURE SGPLOT used (Total process time):
      real time           0.43 seconds
      cpu time            0.11 seconds

NOTE: There were 13 observations read from the data set WORK.MOOSE_ROOM2.

138  title;
```