

# Course Introduction

Steve Simon

Created 2018-07-24

## Overview

- Introducing your instructor
- Where you can get SAS
- Your first SAS program
- History of SAS
- Directory structure and documentation header
- Permanent storage
- Saving your output
- Getting data from a file

[[Speaker notes]]

author: Steve Simon date: created 2018-08-29 purpose: to produce slides for module01 videos license: public domain

Here is an overview of what I want to cover in module02.

Okay. I want to get you started using SAS. It's either going to be really easy or it's going to be really really hard. I want to give you as much guidance as I can without staring over your shoulder. Just quickly, I wrote this Powerpoint presentation and all the Powerpoint presentations using R Markdown. If you are curious I have a repository. It has some beautiful output.

Greetings! My name is Steve Simon and I am the instructor for the class, MEDB 5507, Introduction to SAS.

## Course instructor, Steve Simon



Photo of Steve Simon

[[Speaker notes]]

Greetings! My name is Steve Simon and I am the instructor for the class, MEDB 5507, Introduction to SAS.

## Original developer, Mary Gerkovich



Photo of Mary Gerkovich

[[Speaker notes]]

This course was originally developed by Dr. Mary Gerkovich. I've changed a few things, but I owe a big debt of gratitude to Mary for all of her hard work.

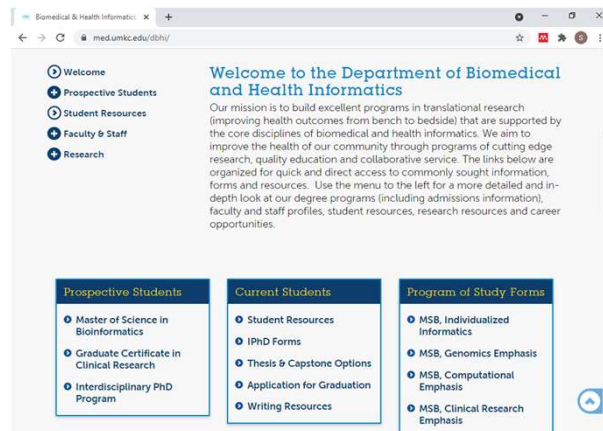
## A bit about myself



[[Speaker notes]]

Let me share a bit about myself.

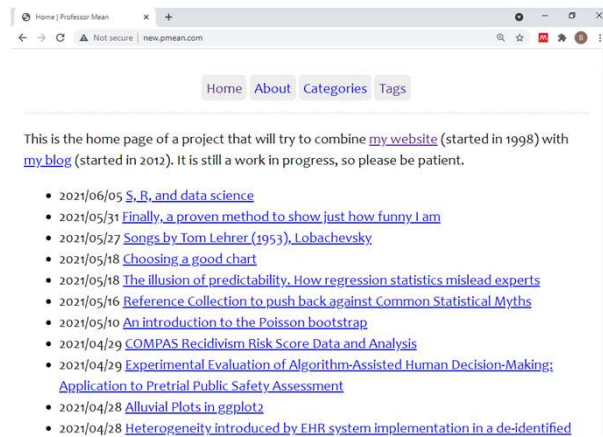
DBHI, <http://med.umkc.edu/dbhi/>



[[Speaker notes]]

I am a part-time faculty member in the Department of Biomedical and Health Informatics.

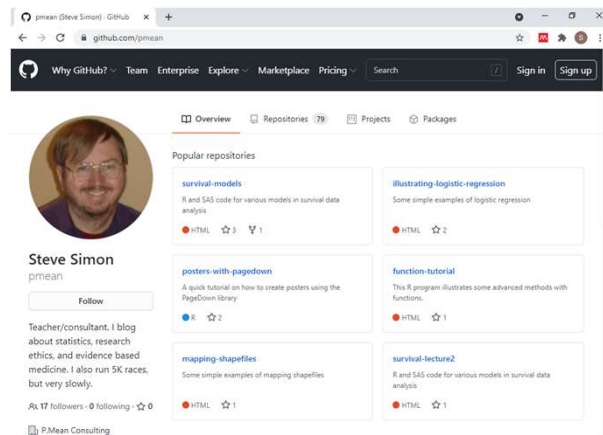
Website, <http://new.pmean.com>



[[Speaker notes]]

I have a blog and a website. I am trying to combine the two into a single site. It's a work in progress, but here is the url for my effort.

Github, <https://github.com/pmean>

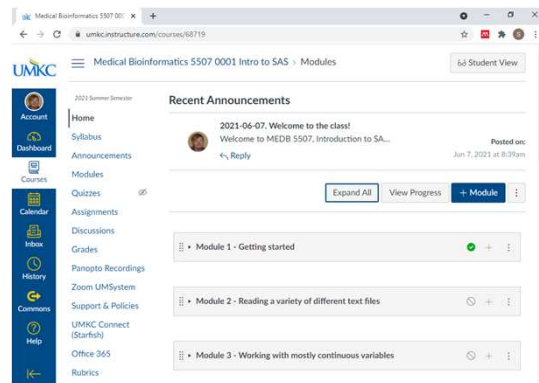


[[Speaker notes]]

I have a github site. It includes some of the programs used in this class. All the material is public domain, but please do not plagiarize your homework from this github site.



# Canvas site



Screenshot of Canvas site

[[Speaker notes]]

My course is currently under development in Canvas. This is the instructors view. You will see a slightly different interface.

## Break #1

- What have you learned
  - Information about me (Steve Simon)
- What's coming next
  - Where can you get SAS

## Where can you get SAS

- On your UMKC computer
  - Desktop, hard-wired to UMKC network
  - No laptops, no home computers
- UMKC Remote Labs
  - Several locations on campus
  - Remote access
- Alternatives not covered in this class
  - SAS University
  - SAS OnDemand for Academics
  - Jupyter lab
  - SASMarkdown, StatWeave

[[Speaker notes]]

There are several ways that you can get access to SAS software. One of these three options should work for you.

You can get it running on your UMKC computer, and I will show you how to do this. You can also access SAS on the UMKC Remote Labs. I'll also show this.

There are some alternatives which I will not show, but if you are interested in investigating, let me know. I am glad to talk privately with anyone about this.

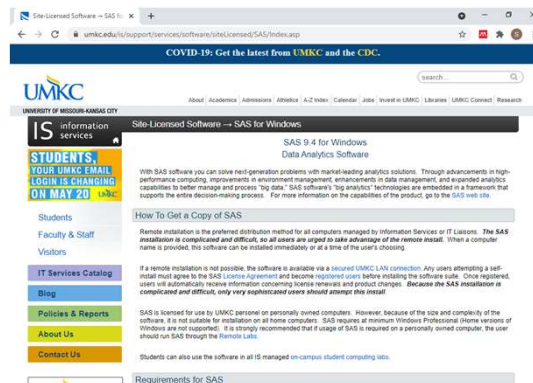
There is a very nice product that I have used in the past called SAS University. It is very difficult to install, but once you have it running, it is very easy to use.

SAS Institute is phasing out SAS University in favor of SAS OnDemand for Academics. This looks like it is much easier to install, though it does require the instructor (me) to set up a few things first. I started work on this, but do not have anything ready yet for this class. Sorry!

If anyone is a fan of Jupyter, you should note that Jupyter lab can run SAS code. You probably need to have SAS already running on your computer, and I have not had time to experiment with this.

If you are familiar with the R programming environment, there are a couple of packages, SASMarkdown and StatWeave, that allow you to integrate SAS code and output into your workflow. I have not had time to experiment with this either.

## SAS on your UMKC computer



Screenshot of UMKC IS page on SAS software

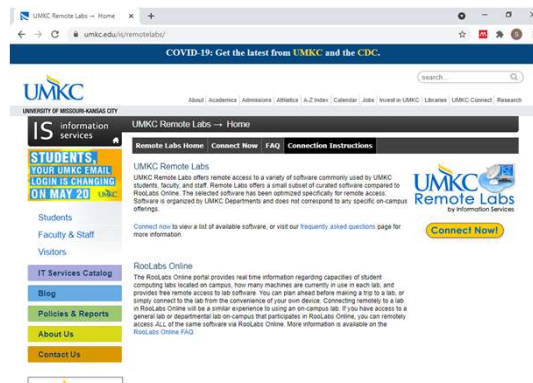
[[Speaker notes]]

This screenshot may be too small for you to read, but you can find the proper link on the recommended readings list for this week on the Canvas website. SAS works for any desktop computer on the UMKC campus. But it has to be hard-wired to the UMKC network. By hard-wired, I mean that there is an ethernet cable connecting your computer to a socket on the wall.

If you are fortunate enough to have access to a hard-wired computer, you can get SAS installed easily. Someone else will do it for you. It may already be sitting on your computer.

With a very few rare exceptions, you cannot get UMKC to load SAS on a laptop computer or on a home computer. This because of the license agreement that UMKC signed with SAS Institute. It does not allow for home use of SAS.

# UMKC Remote Labs



UMKC Remote Labs home page

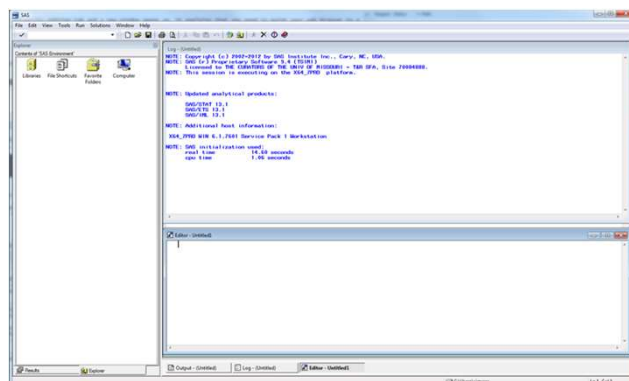
[[Speaker notes]]

You can also run SAS through the UMKC Remote Labs. It should run fine on most browsers, though it can be a bit fussy.

## Break #2

- What have you learned
  - Where you can get SAS
- What's coming next
  - Your first SAS program

## Opening screen - SAS commercial version



Opening screen of SAS with multiple windows

[[Speaker notes]]

If you are running the “regular” version of SAS, click on the icon and here’s an image of what the opening screen looks like. SAS uses a multi-window format. The layout is a bit chaotic. I usually close some windows and re-arrange others. For the benefit of this presentation, I am going to resize everything, close some of the windows, and maximize the one window of greatest importance, the program editor window.

I want to show how to use SAS. I had the screen resolution too high. It is now 800 by 600 and I think it is like going back.

I like to get rid of a couple of windows and use tile vertical for the rest.

Take baby steps. If you can’t run this program, talk to me face-to-face.

You should try typing in the program like I am doing. It helps.

I like to put lots of blank lines in.



Normally you would use a two part name. Not here. This is a simple throw-away example, so it can use a one part name. That means it disappears.

Here is where we type the data. At the end you put in another run statement.

This is bad form. Normally you put the data in a separate file.

I like the obs option a lot.

I'll put in a title as well.

I'm going to indent a bit less.

The thing I always worry about is leaving out a semicolon.

There is no autosave feature.

SAS uses a stupid default 9.4. If you are using a network computer, you want to use a network folder, typically the Q drive.

I don't do this, but do what I preach, not what I do.

I'm going to create a new folder. I can't decide on a name. Am I indecisive? I want to set up a logical directory structure, but I also want it to be reasonably flat. I tend to save things in a folder called "source" (SRC).

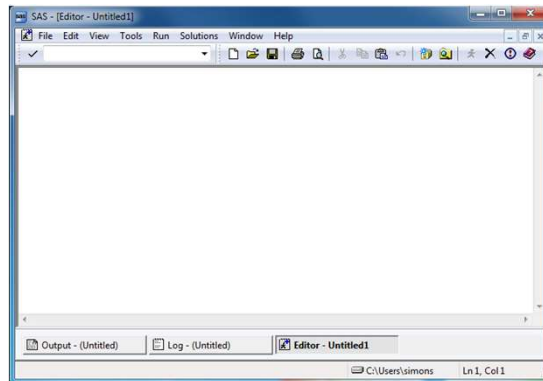
The run guy is a guy who looks like he is running.

You see some diagnostic information. Work is a scratch. We are not using this again. Three observations, but then one because we said obs=1.

There is the output. SAS by default centers things, and it doesn't work well with 800 by 600.

Try this yourself. It's a baby step. If it works then you can take more baby steps.

## SAS program editor (1 of 2)

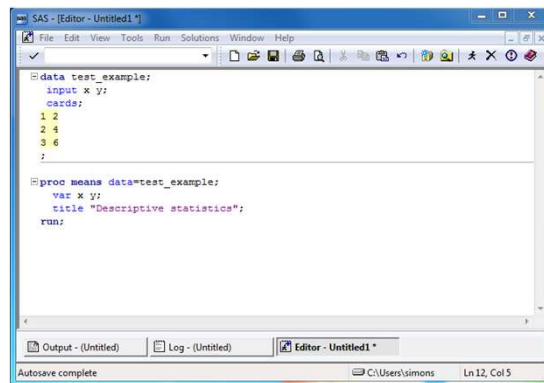


Maximized view of empty SAS program editor window

[[Speaker notes]]

This is the program editor window. You type in your program in this window, or read an existing program from another window. The other two remaining tabs, the log window, and the output window, are also important.

## SAS program editor (2 of 2)

The screenshot shows the SAS program editor window titled "SAS - [Editor - Untitled1 \*]". The menu bar includes File, Edit, View, Tools, Run, Solutions, Window, and Help. The toolbar contains icons for file operations and execution. The main text area contains the following SAS code:

```
data test_example;  
  input x y;  
  cards;  
  1 2  
  2 4  
  3 6  
  ;  
  
proc means data=test_example;  
  var x y;  
  title "Descriptive statistics";  
run;
```

The status bar at the bottom indicates "Autosave complete", "C:\Users\simons", and "Ln 12, Col 5".

Program editor window with simple SAS program

[[Speaker notes]]

If you have SAS running, try running the following sample program. Here's a simple test program. After you type this program in, click on FILE | SAVE and store your program somewhere safe. Save it to a location where you can remember things.

If you are using the computer labs, you need to save things on a network folder. You can't use a USB stick.

## SAS Test program

```
data test_example;  
  input x y;  
  cards;  
1 2  
2 4  
3 6  
;  
  
proc means data=test_example;  
  var x y;  
  title "Descriptive statistics";  
run;
```

[[Speaker notes]]

After you type this program in, click on FILE | SAVE and store your program somewhere safe.

## SAS log window (1 of 2)

[[Speaker notes]]

The font here is a bit small, but notice that there are no red messages indicating warnings or errors. We're thrilled when we see no warnings or error messages. We're always looking for warnings and errors. We also watch closely the number of observations.

## SAS log window (2 of 2)

[[Speaker notes]]

Always start looking for error messages at the top. The very first error or warning message is most likely to be helpful, and later errors/warnings are often of less value.

## Log messages (1 of 2)

```
1 data test_example;  
2 input x y;  
3 cards;
```

NOTE: The data set WORK.TEST\_EXAMPLE has 3 observations and 2 variables.

NOTE: DATA statement used (Total process time):  
real time 0.51 seconds  
cpu time 0.04 seconds

[[Speaker notes]]

Always watch the log to see that you have read in the proper number of observations.

## Log messages (2 of 2)

```
9   proc means data=test_example;  
10      var x y;  
11      title "Descriptive statistics";  
12  run;  
  
NOTE: Writing HTML Body file: sashtml.htm  
NOTE: There were 3 observations read from the  
data set WORK.TEST_EXAMPLE.  
NOTE: PROCEDURE MEANS used (Total process time):  
      real time          1.72 seconds  
      cpu time           0.20 seconds
```

[[Speaker notes]]

..and that you are analyzing the proper number of observations.



## Where is the output?

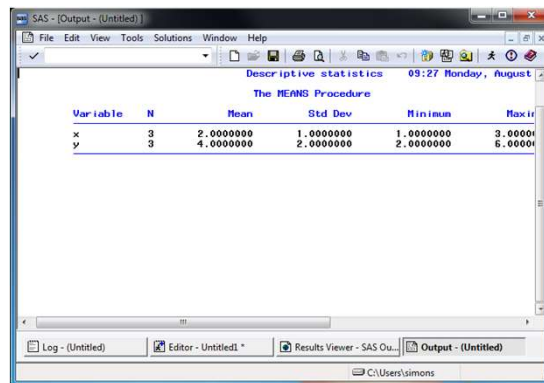
SAS has several options for storing output.

- In the output window
- As an html file
- As a pdf file

[[Speaker notes]]

Output is tricky. I want to talk in more detail later about this, but you can take the output and save it. There are several ways to do this. If you already have output, click on the CREATE LISTING option box to send the output to the output window. Click on the CREATE HTML option box to send the output to an html file. Click on the BROWSE button to select a default folder for your html file.

## SAS output window



The screenshot shows the SAS Output window titled "SAS - [Output - (Untitled)]". The window displays the results of a MEANS procedure. The title bar includes "Descriptive statistics" and the date/time "09:27 Monday, August". The main content area shows a table with the following data:

Variable	N	Mean	Std Dev	Minimum	Maximum
x	3	2.0000000	1.0000000	1.0000000	3.0000000
y	3	4.0000000	2.0000000	2.0000000	6.0000000

The window has a menu bar (File, Edit, View, Tools, Solutions, Window, Help) and a toolbar. The status bar at the bottom shows the path "C:\Users\simons".

Maximized view of default SAS output

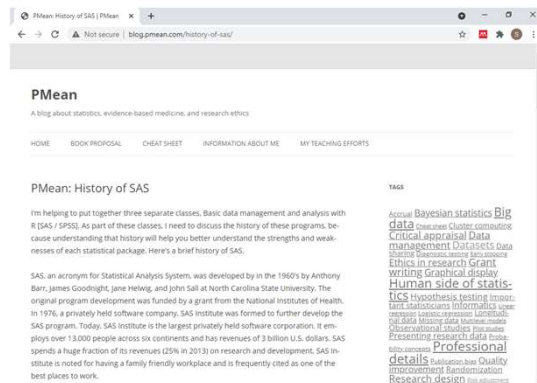
[[Speaker notes]]

Here's what the output window looks like. Notice that SAS uses a monospaced font here.

## Break #3

- What have you learned
  - Your first SAS program
- What's coming next
  - History of SAS

# History of SAS



Blog post on the history of SAS

[[Speaker notes]]

History of SAS. If you understand where SAS comes from, you understand some of the limitations.

This is on my blog.

## History of SAS

- SAS=Statistical Analysis System
- Founders come from NCSU
  - Anthony Barr
  - James Goodnight
  - Jane Helwig
  - John Sall
- Originally for IBM mainframes
  - PL/1, FORTRAN, Assembler
  - Translated to C in 1985

[[Speaker notes]]

SAS was developed at NC State.

In the 1960's, IBM mainframes dominated. So SAS was originally written just for these systems. Originally, SAS was written in a mix of PL/1, Fortran, and Assembler. Millions of lines of code re-written in C in 1985 so that SAS could run on personal computers.

SAS was built in the 1960s. There is very little software written in the 1960s that is still in regular use. It shows the staying power of SAS, but it is “long in the tooth.”

## History of SAS

### – SAS Institute

- Founded 1976
- Privately held
- Huge spending on R&D
- Great place to work

[[Speaker notes]]

SAS Institute was formed in 1976. It is a privately held company.

SAS spends over a quarter of its budget, which is a huge fraction, on Research and Development.

The SAS Headquarters is in Cary, NC, and it is huge. It is a pretty nice place to work with many family friendly policies even from the 1980s. SAS Institute has been rated in many magazine reviews as one of the top places to work.

## History of SAS

- SAS licensing model
  - Great for large organizations
  - Prohibitively expensive for individuals
- Excellent training resources
  - SAS publications
  - Certification program
  - SAS user conferences
- Other products
  - JMP, 1989
  - Viya, 2017

[[Speaker notes]]

SAS has a licensing model that is aggressively priced for large corporations but prohibitively expensive for individual consultants.

SAS is oriented around various data sets and procedures. There is a menu driven version of SAS, but it is not very good. If you want a good package that is totally menu driven, use SPSS.

SAS has a licensing model that is prohibitive for individual consultants. They offer a free product, SAS University, that is intended for teaching.

SAS has literally hundreds of books published through an in-house publisher. It has a certification program that allows you to earn credentials that can help you in your job search. SAS also sponsors some very extravagant user conferences.

SAS Institute has many products beyond SAS. Most notable of these is JMP (pronounced “jump”). This is an acronym for John’s Macintosh Product. It was released in 1989 when the Macintosh series of computers had many graphical user

interface features that were not yet available for other personal computers. It pioneered (and continues to lead) in many interactive and dynamic graphic features.

SAS Viya is a cloud based platform with many advanced visualization and machine learning algorithms not found in SAS.



## Break #4

- What have you learned
  - History of SAS
- What's coming next
  - Directory structure/Documentation header

## Directory structure

- One directory for the entire class
  - Possibly one directory for each module
- Subdirectory structure
  - src
  - results
  - data
  - others?
    - images
    - doc

Make sure you store things consistently. This part of the requirements for this class.

Store all your programs for this class in a single directory. You can call it “sas” or “5507” or anything you like. Some people may prefer to create a separate directory for each module in this class. That’s fine also.

Most importantly, create three subdirectories, src, results, and data. Store your programs in src, your output in results and your raw and intermediate datasets in data. You may wish to create other subdirectories, such as an images folder for any graphs you produce, a doc folder for any documentation you collect, but the three important subdirectory folders are src, results, and data.

## Basic program, part 1

```
data small_example;  
  input x y;  
  datalines;  
1 2  
2 4  
3 6  
;  
proc print  
  data=small_example(obs=1);  
  title "First row of data";  
run;
```

Here is the basic program I want you to run.

The data statement creates a dataset with the name “small\_example”.

The input statement tells SAS to expect two variables and assigns them the names x and y.

The datalines statement tells SAS to read the data directly below this line. This is NOT a recommended practice. You should normally keep your data separate from your code. I am doing this only for the sake of simplicity. You will see in just a minute how to handle data that comes in a separate file.

The three lines of data follow. A single semicolon tells SAS that this is the end of the data.

The print procedure will print part or all of a dataset.

You specify the name of the dataset with the data= option. The options obs=1 tells SAS to print only the first row.

The title1 statement prints a descriptive title on the first line of output.

The run statement tells SAS that there is no more information associated with this procedure.

## Break #5

- What have you learned
  - How to read a small dataset
  - How to print a small dataset
- What's coming next
  - Permanent storage

## Permanent storage, part 1

```
* 5507-01-simon-permanent-storage.sas
* author: Steve Simon
* date: created 2021-05-30
* purpose: to store data set in a permanent
location
* license: public domain;
```

Let's look at a modified program that stores the data in a permanent location.

Here is the documentation header. You should include a documentation header with any program you run for this class.

## Permanent storage, part 2

```
%let path=q:/introduction-to-sas;  
  
libname perm "&path/data";  
  
data perm.simple_example;  
    input x y;  
datalines;  
1 2  
2 4  
3 6  
;
```

Notice the top line in this section. This is the libname statement. It tells SAS that you want to establish a permanent storage location at ../data. The ../data tells the computer system to go one level closer to the root directory, and then slide into the data subdirectory.

You assign a brief name (no more than eight characters!) to this location. In my program, I use the name "perm" but anything is fine here.

Then you prefix the dataset name simple\_example with the libname location and a dot. This is called a two part name by SAS. The first part gives the permanent location folder and the second part gives the file name.

Once you establish a two-part name for a dataset, you assure that it is stored for later re-use.

## Permanent storage, part 3

```
proc print  
  data=perm.simple_example(obs=1);  
  title1 "First row";  
run;
```

Once you establish a two-part name, use it everywhere.



## Re-using data in permanent storage, part 1

```
* 5507-01-simon-re-use.sas
* author: Steve Simon
* date: created 2021-05-30
* purpose: to re-use stored data
* license: public domain;
```

Here's a program that re-uses the dataset you just placed in permanent storage.

First, let's show the documentation header.

## Re-using data in permanent storage, part 2

```
%let path=q:/introduction-to-sas;  
  
libname perm "&path/data";  
  
proc means  
    data=perm.simple_example;  
    title1 "Descriptive statistics";  
run;
```

Notice that there is no data step in this program, you start with a libname statement that reminds SAS where you stored the permanent dataset. Then you just refer to the two-part name in the data= option of any SAS procedure. Here we are computing some simple descriptive statistics using proc means.

## Break #6

- What have you learned
  - Permanent storage
- What's coming next
  - Saving your output

## Saving output as pdf, part 1

```
* 5507-01-simon-save-output.sas
* author: Steve Simon and Steve Simon
* date: created 2021-06-12
* purpose: to create a permanent dataset
* license: public domain;
```

Let's look at a modified program that stores your output as a pdf file.

Here is the documentation header.

## Saving output as pdf, part 2

```
%let path=q:/introduction-to-sas;

libname perm "&path/data";

ods pdf file=
    "&path/results/5507-01-simon-save-output.pdf";

data perm.small_example;
    input x y;
    datalines;
1 2
2 4
3 6
;
```

The ods statement should be placed near the top of the code, certainly before any SAS procedure that produces output.

## Saving output as pdf, part 3

```
proc print  
    data=perm.small_example(obs=1);  
    title1 "First row of data";  
run;  
  
ods pdf close;
```

Near the bottom, you turn off the output. Place this AFTER the last SAS procedure that produces output.

## Break #7

- What have you learned
  - Saving your output
- What's coming next
  - Getting data from a file

## Reading data from a file, part 1

```
* 5507-01-simon-input-text.sas
* author: Steve Simon
* date: created 2021-05-30
* purpose: to read data from a separate file
* license: public domain;
```

It is a very basic principle of good computing practices that you keep your data and your program in separate files. This code shows you how to do this using the infile statement.



## Reading data from a file, part 2

```
%let path=q:/introduction-to-sas;  
  
libname perm "&path/data";  
  
filename rawdata "&path/data/six-numbers.txt";  
  
ods pdf file="&path/results/5507-01-simon-input-  
text.pdf";  
  
data perm.simple_example;  
  infile rawdata;  
  input x y;  
run;
```

The filename statement tells SAS where a particular dataset is stored: both the path and the name of the file. It associates that path and filename with a variable that you refer to using the infile statement.

## Reading data from a file, part 3

```
proc print  
    data=perm.simple_example(obs=1);  
    title1 "First row";  
run;  
  
ods pdf close;
```

The last part of the program remains unchanged.

## Reading data from a file, part 4

```
1 2  
2 4  
3 6
```

This is what your data file looks like. It is just six numbers arranged in a grid.

You will see many variations on the layout of data, and SAS can handle just about any variation. You will see how to handle many of those variations in an upcoming module.

## Summary

- What have you learned?
  - Introducing your instructor
  - Where you can get SAS
  - Your first SAS program
  - History of SAS
  - Directory structure and documentation header
  - Permanent storage
  - Saving your output
  - Getting data from a file