# simon-5507-01-slides

# Topics to be covered

- What you will learn
  - About this course and your instructors
  - Where you can get SAS
  - Your first SAS program
  - History of SAS
  - Documentation header
  - Permanent storage
  - Saving your output
  - Getting data from a file

# 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

This is Steve Simon, one of the instructors for MEDB 5507, Introduction to SAS.

# Course instructor, Suman Sahil

Insert image of Suman Sahil here.

Speaker notes

This is Steve Simon, one of the instructors for MEDB 5507, Introduction to SAS.

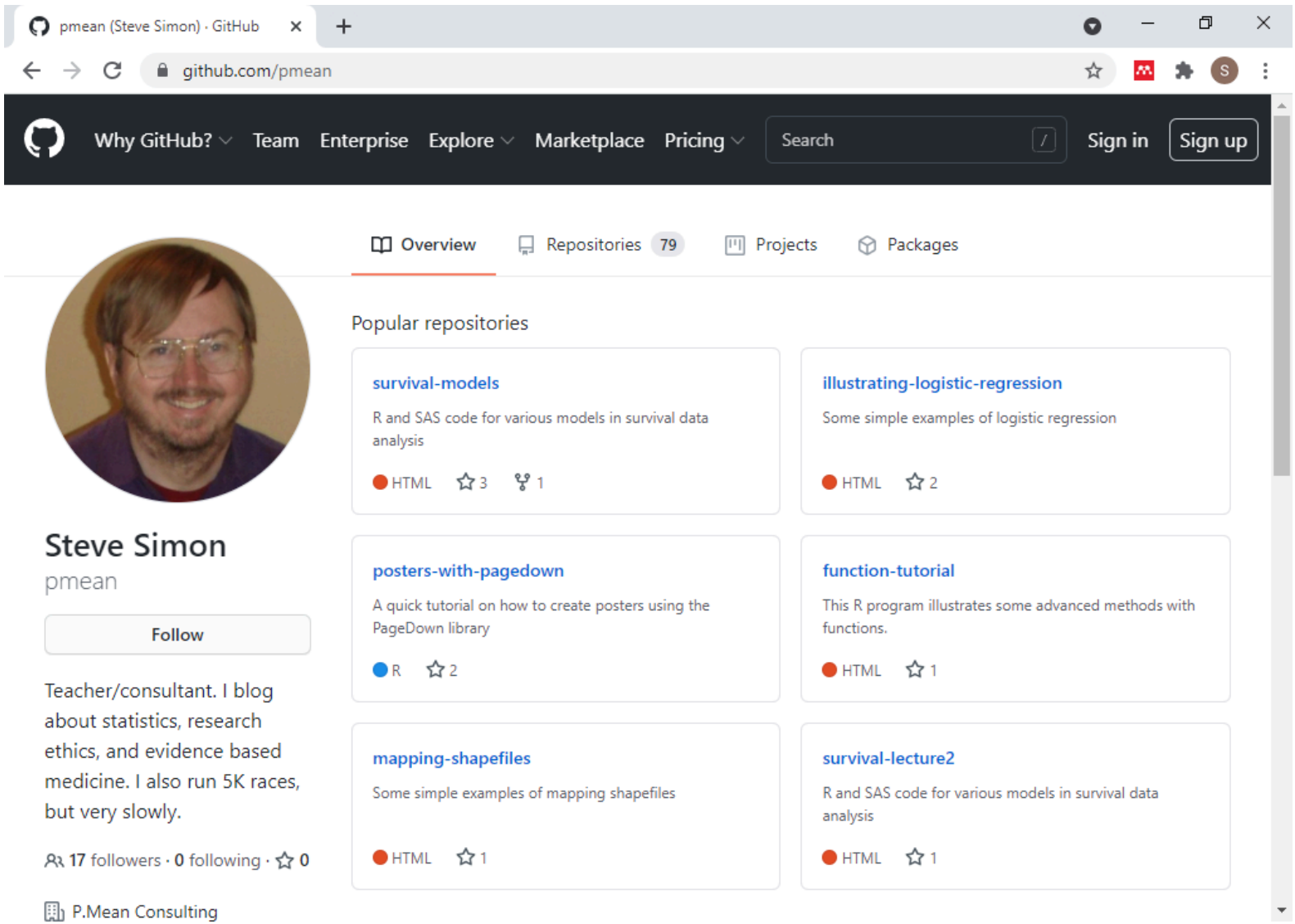
# Original developer, Mary Gerkovich

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.

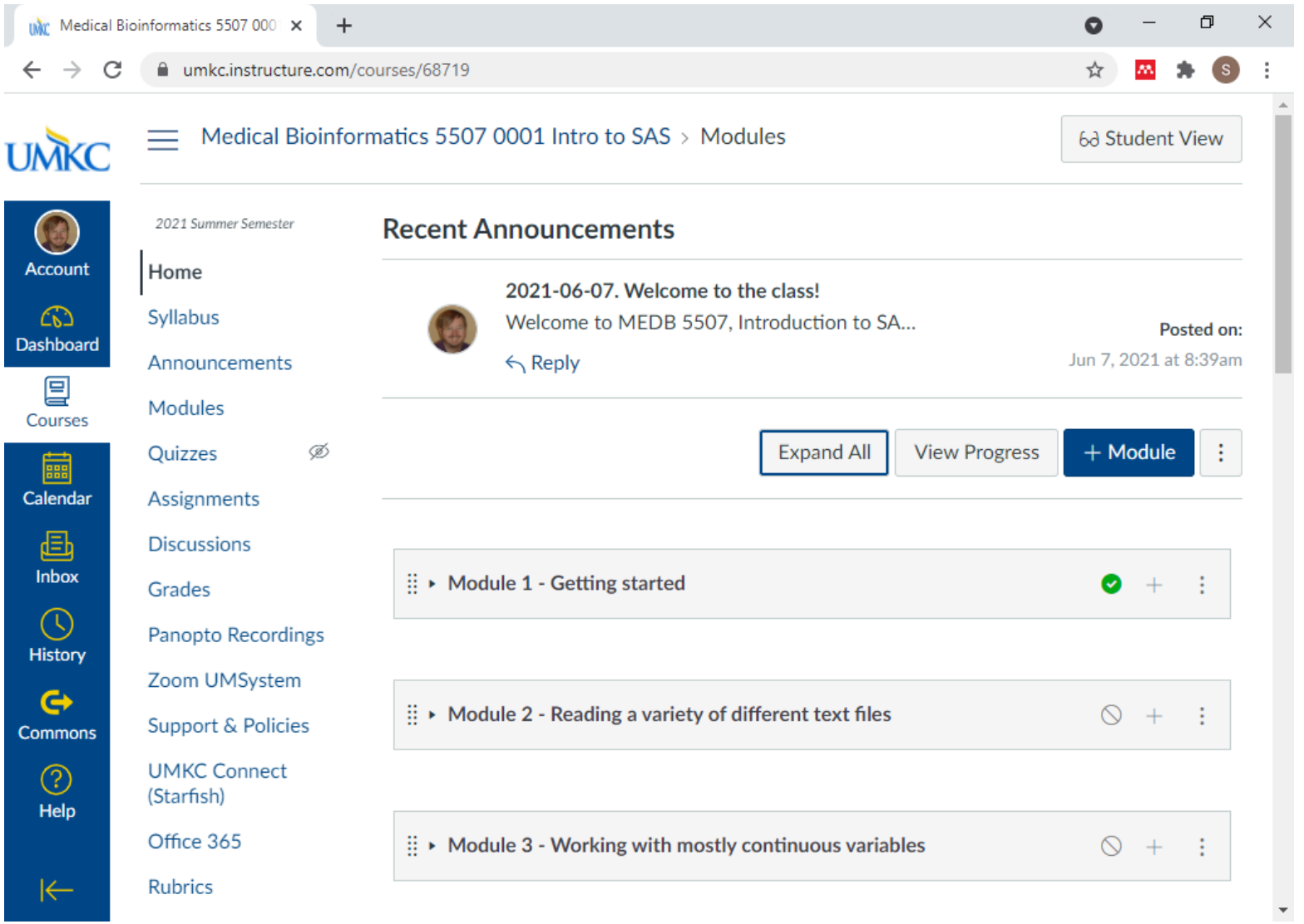
# Github, https://github.com/pmean



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 you have learned
  - About this course and your instructors
- What’s coming next
  - Where you can get SAS

# Where can you get SAS

- SAS OnDemand for Academics (SODA)
- On your UMKC computer
  - Desktop, hard-wired to UMKC network
  - No laptops, no home computers
- UMKC Student labs
  - Royall Hall 303, Lab #17 and #38
- UMKC Remote Labs
- Alternatives not covered in this class
  - SAS University
  - 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.

The recommended option is SAS OnDemand for Academics. This version of SAS runs in the cloud. It is restricted to educational uses only.

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.

# Alternative places where you can get SAS

- SAS University
- Jupyter lab
- SASMarkdown. StatWeave

Speaker notes

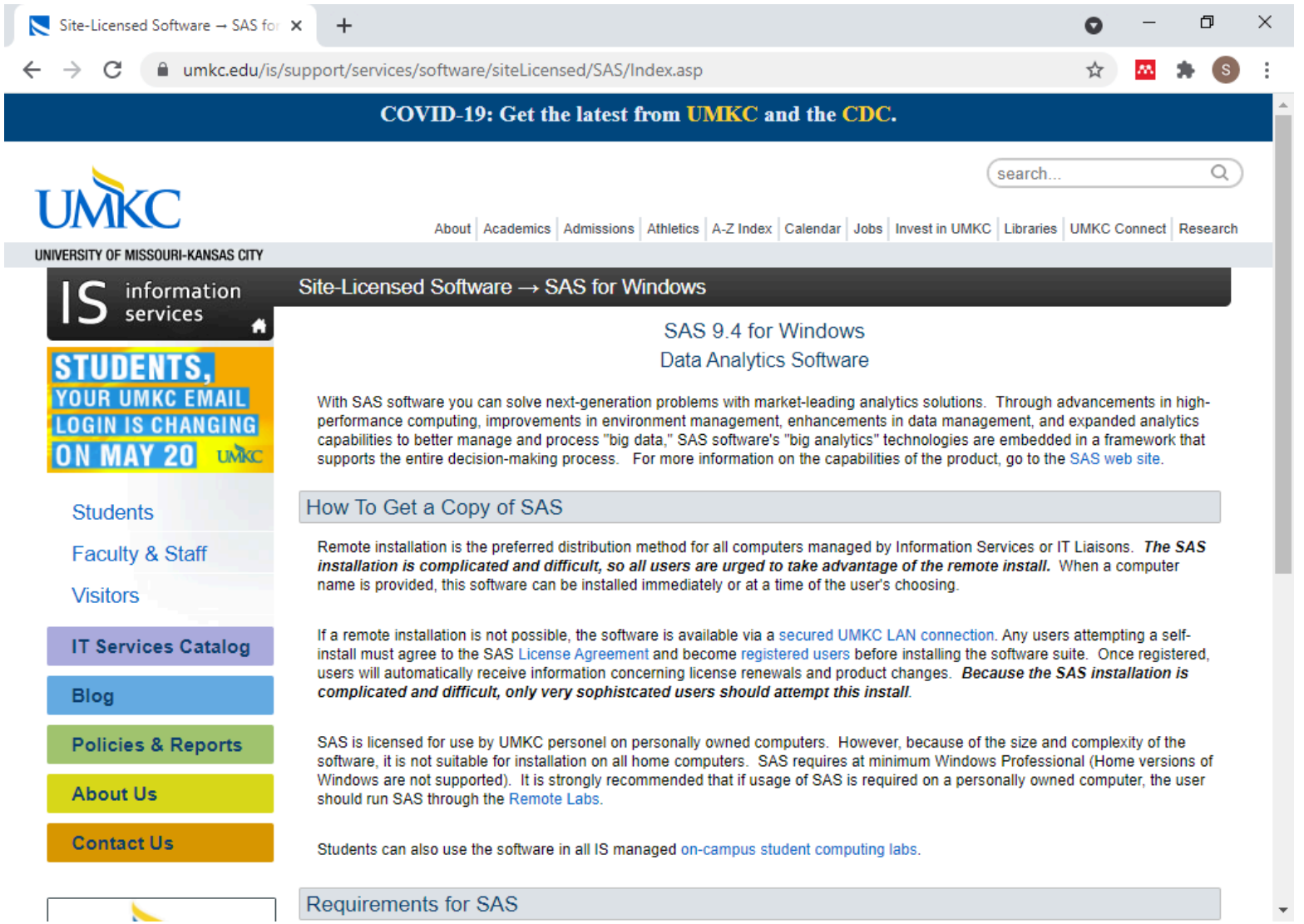
There are some alternatives which I will not show, but if you are interested in investigating, let me know. I am glad to talk privately with anyone about this.

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.

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

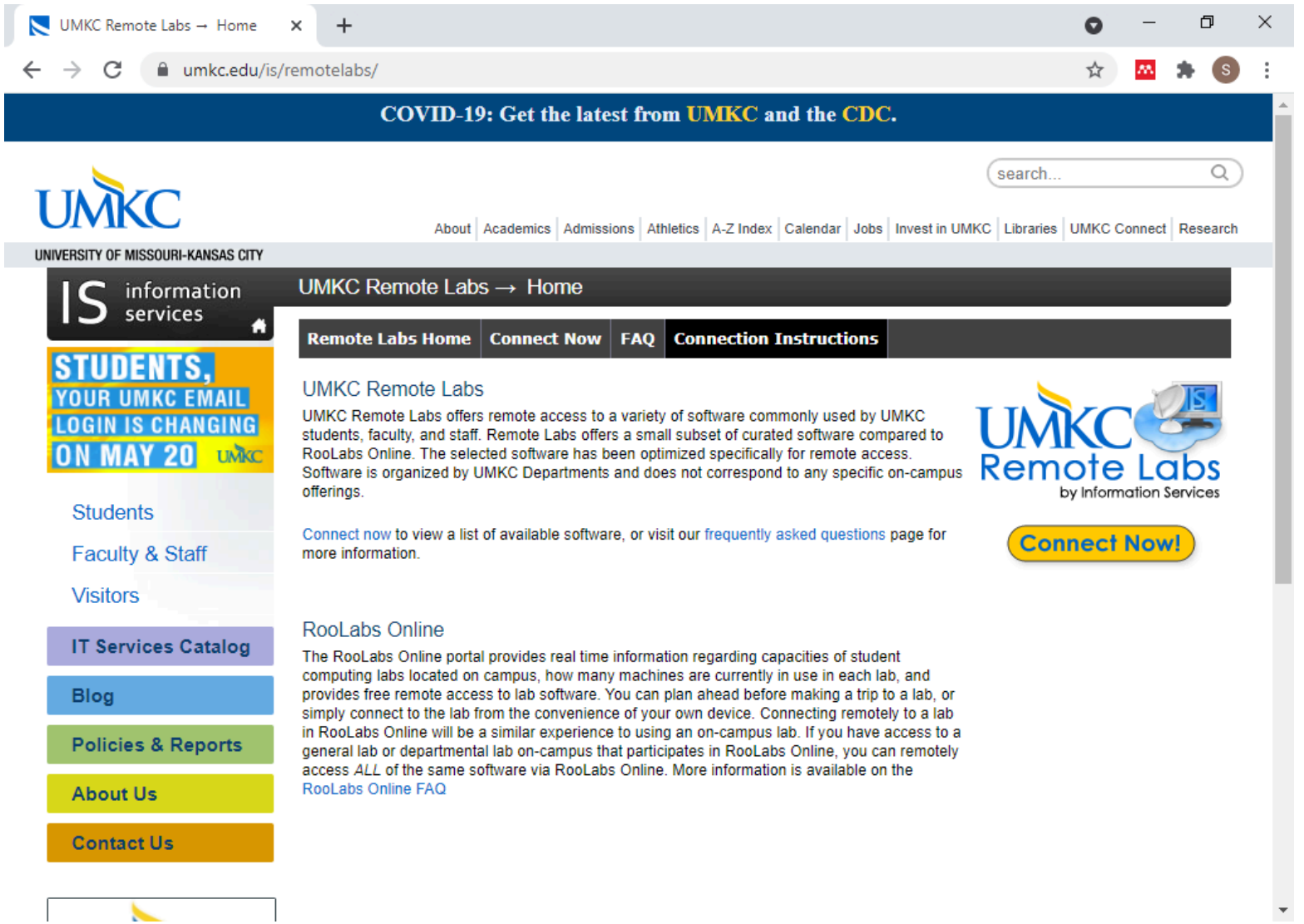
Speaker notes

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

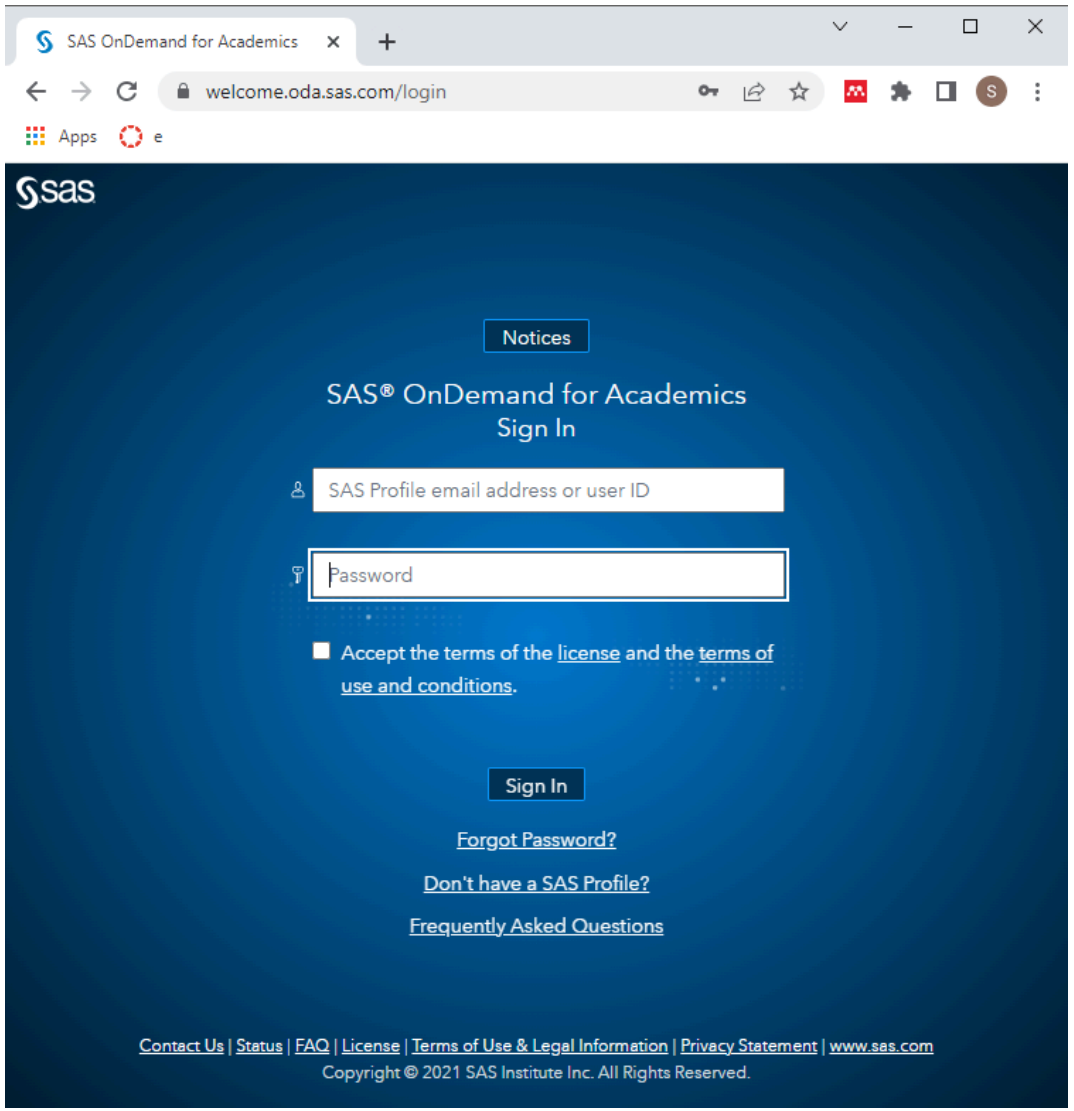



UMKC Remote Labs home page

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.

# SODA login page

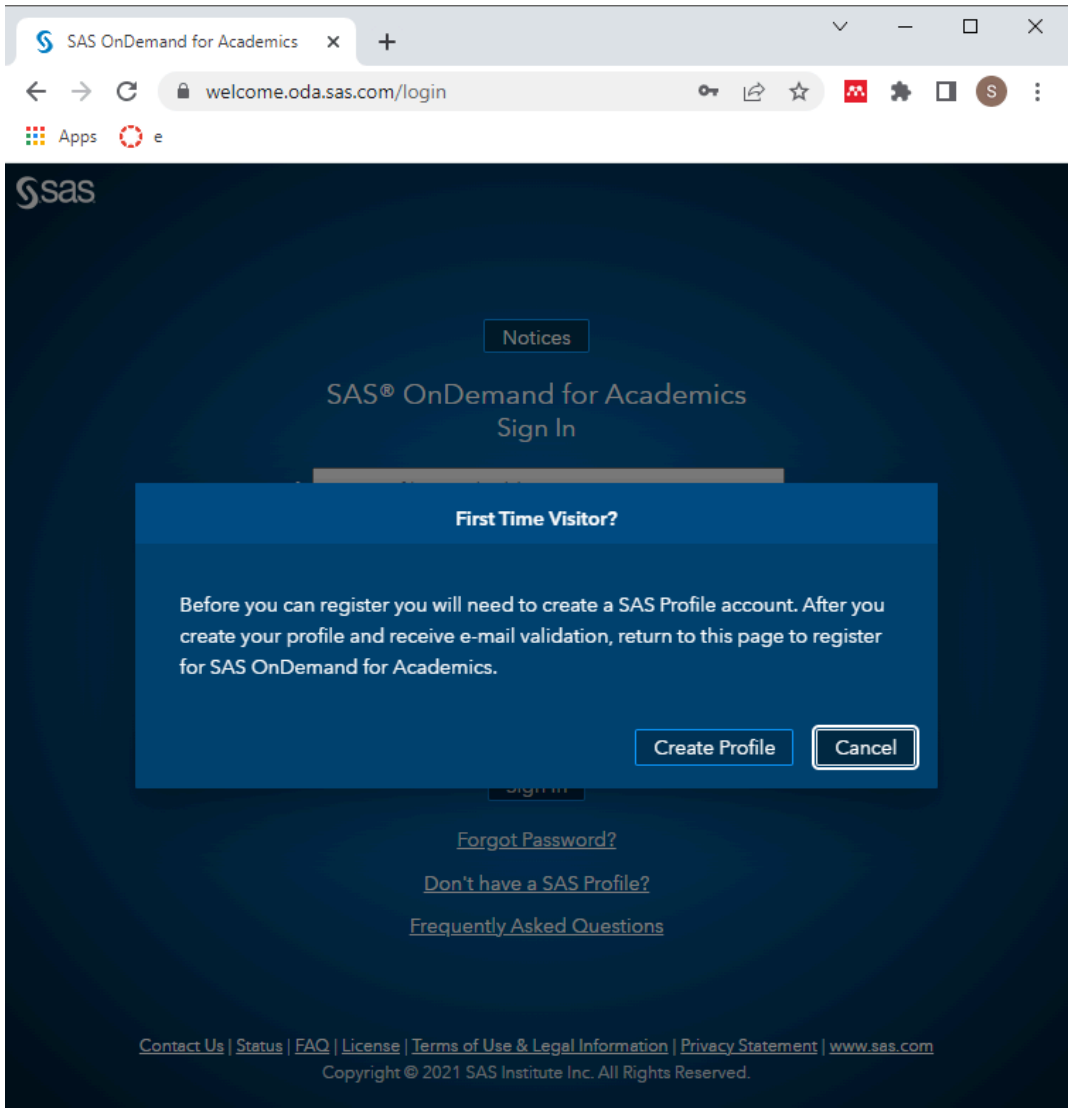



Screenshot of SODA login page

Speaker notes

Here is the login page for SAS OnDemand for Academics. If you have used SAS a lot in the past, you may already have a SAS profile sign-in. More likely, you do not have such an account. You set it up by clicking on the “Don’t have a SAS Profile?” link.

# SODA create profile (1 of 2)

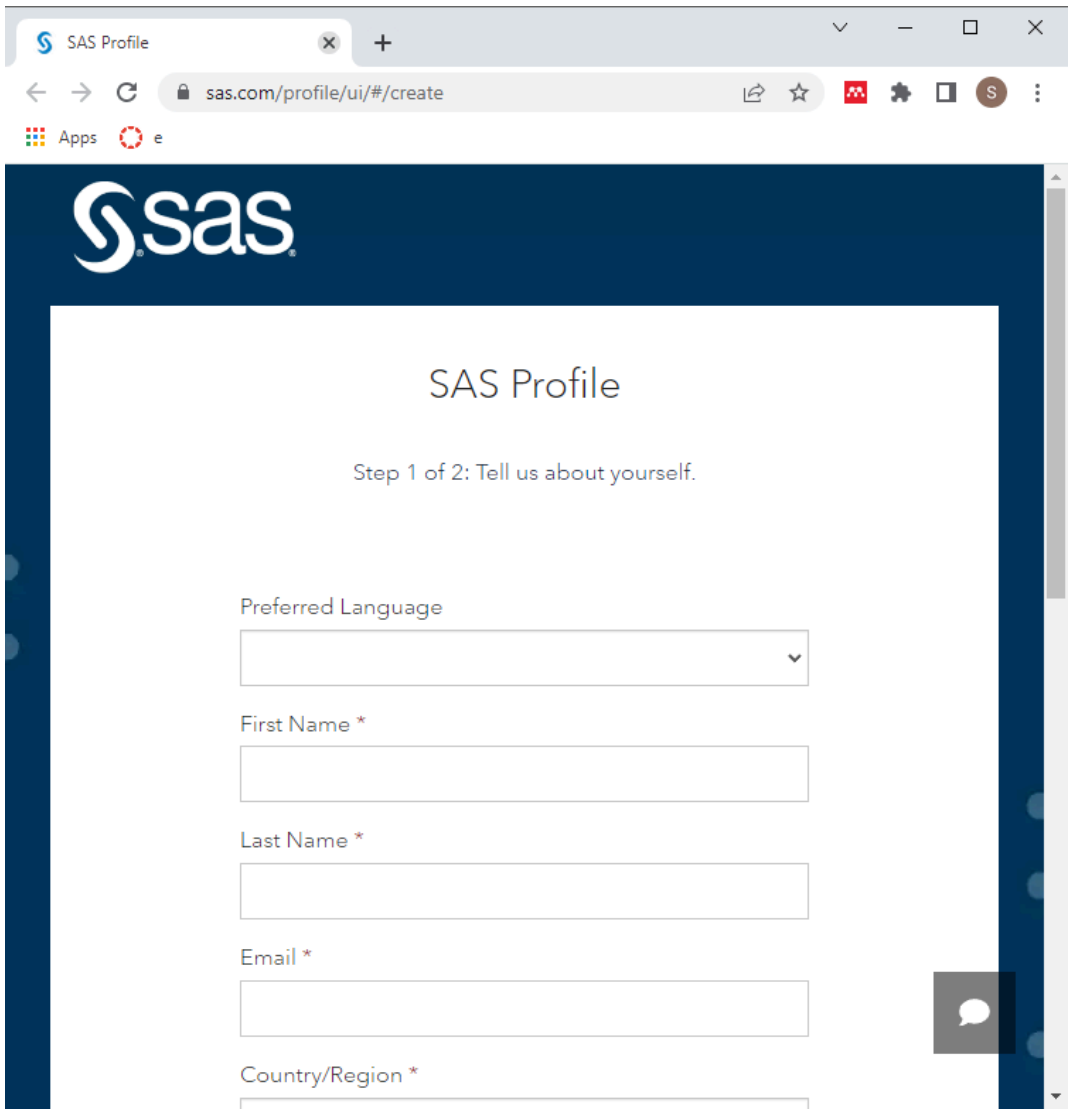



Screenshot of First Time Visitor dialog box

Speaker notes

SAS guides you through the process fairly nicely. Read this dialog box. In particular, note that the profile by itself is not enough. You will still have to register for SAS OnDemand for Academics once the profile is created.

# SODA create profile (2 of 2)

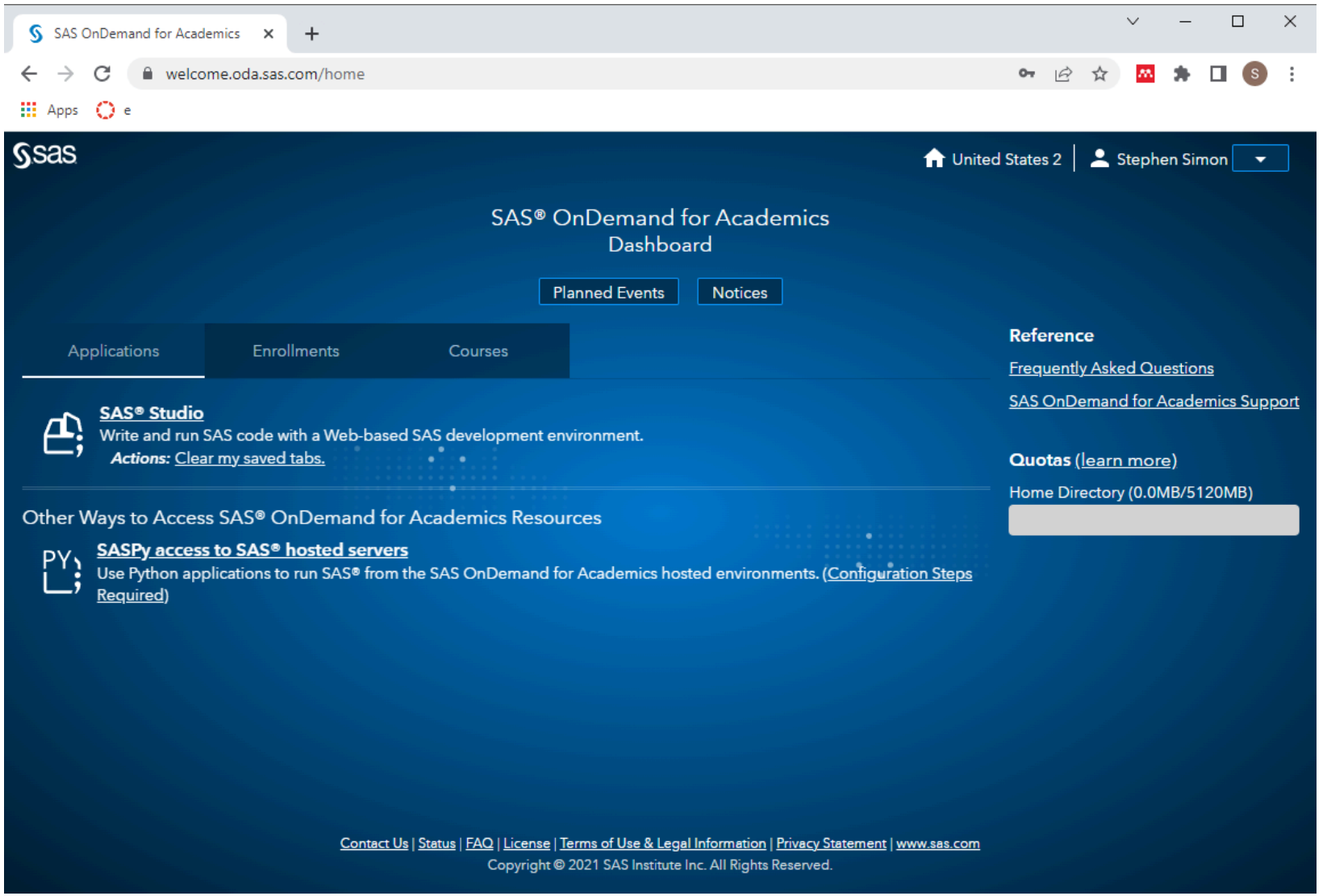



Screenshot of First Time Visitor dialog box

Speaker notes

I won’t take you through all of the steps, but SAS is not asking for a lot of information. Note the box at the bottom of the screen (not shown in this screenshot) that asks if you want to get promotional emails from SAS Institute. It’s easy enough to say “Yes” here and then unsubscribe later if you find the emails are not worth the trouble. Or you can say “No” right away if you prefer.

# SODA dashboard (1 of 2)




Screenshot of SODA dashboard

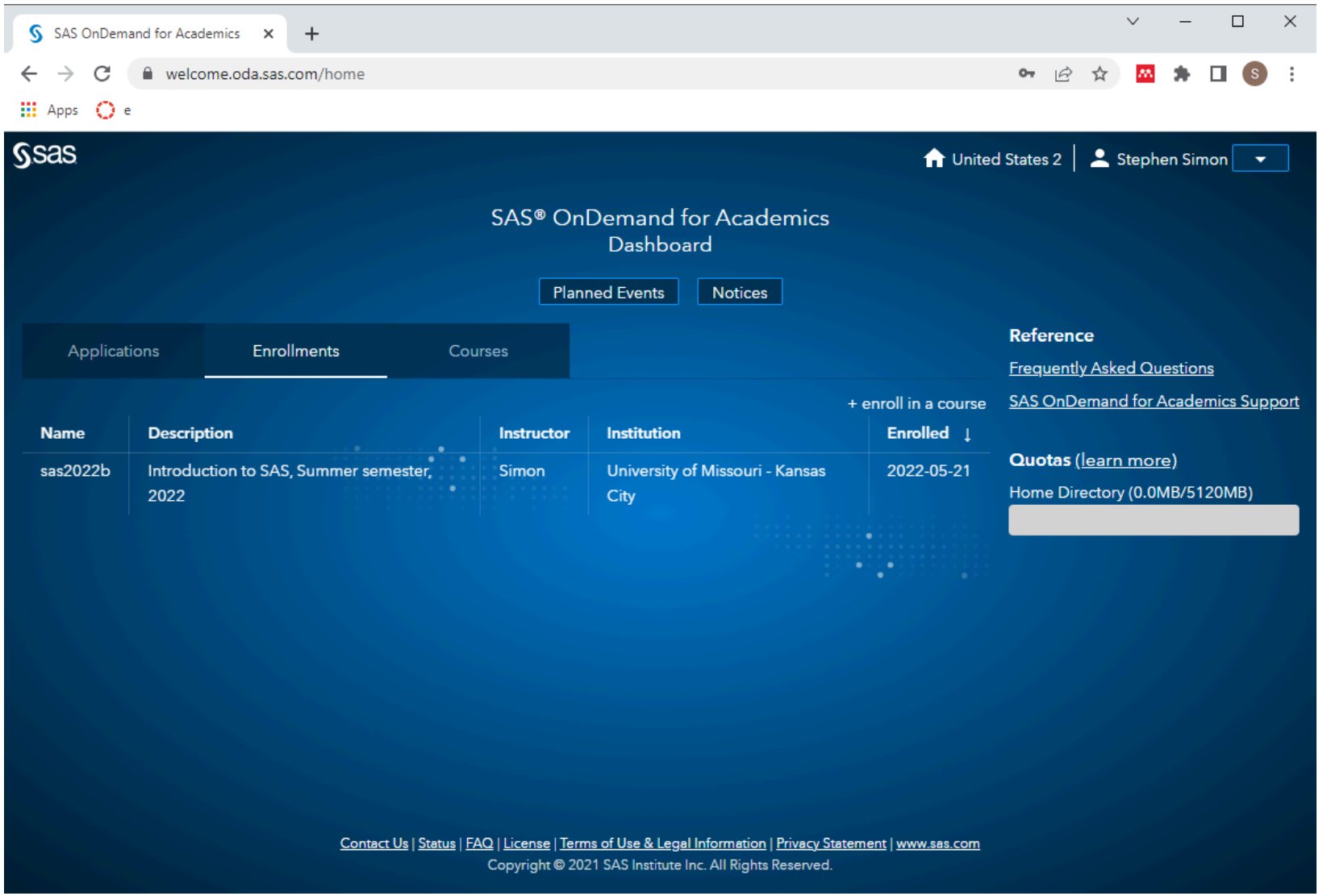
Speaker notes

This is the SAS OnDemand Dashboard. It offers two options, SAS Studio and a Python interface to SAS. I have not used the Python interface, but I’m sure it is pretty simple for someone already experienced with Python.

The top of the page has links to Planned Events and Notices. Beneath that are links to Applications, Enrollments, and Courses.

To load SAS Studio, click on the top link.

# SODA enrollments

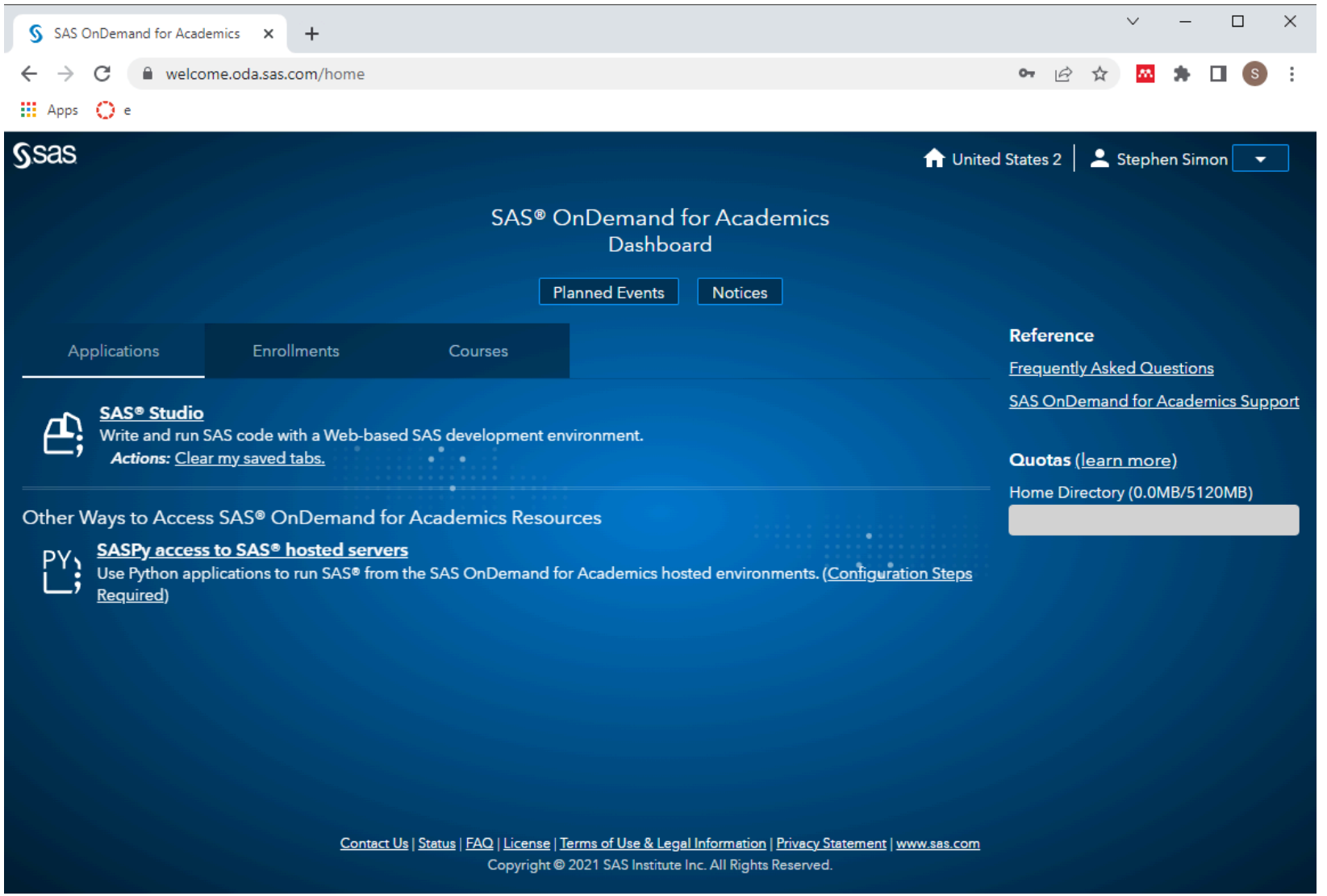



Screenshot of SODA enrollments page

Speaker notes

The enrollments page allows you to “enroll” in my class. This isn’t really needed. It adds a folder with all the data files that I use in this class, but you could also get these from my github site.

# SODA dashboard (2 of 2)

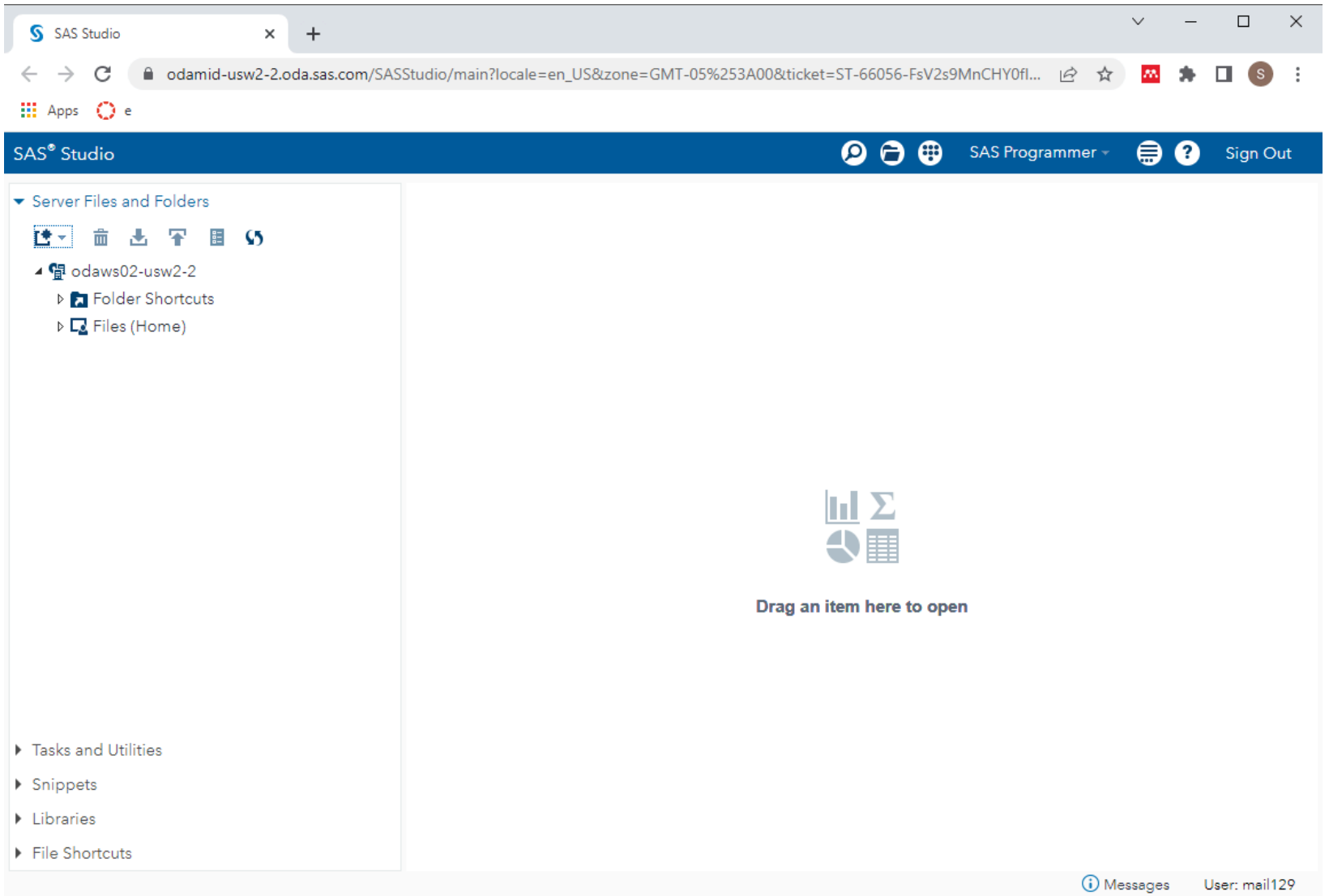



Screenshot of SODA dashboard

Speaker notes

Go back to the dashboard by clicking on the Applications link. To load SAS Studio, click on SAS Studio link.

# SODA studio (1 of 3)

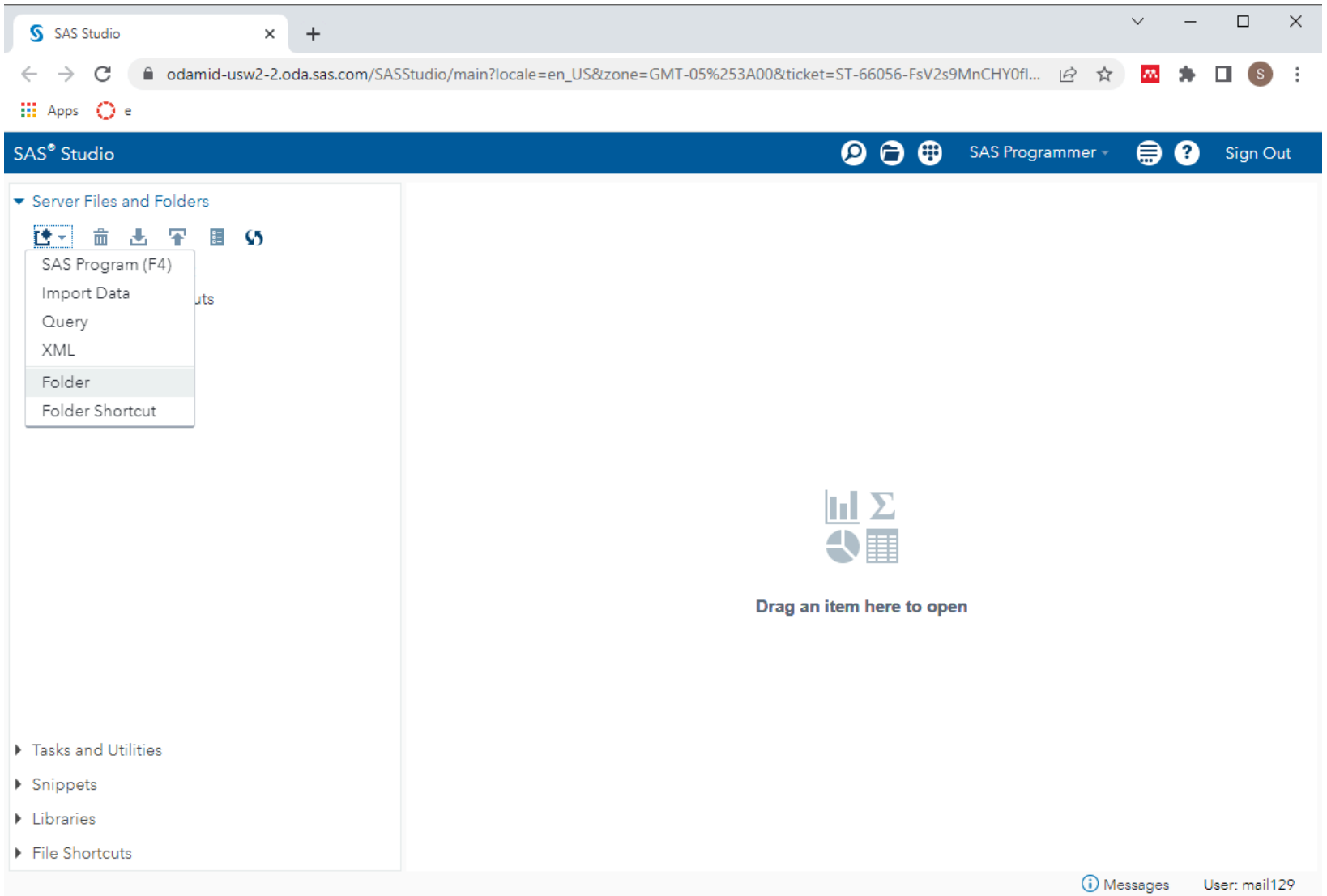



Screenshot of SODA dashboard

Speaker notes

This is what SAS Studio looks like. The first thing you should do is to create a directory structure.

# SODA studio (2 of 3)



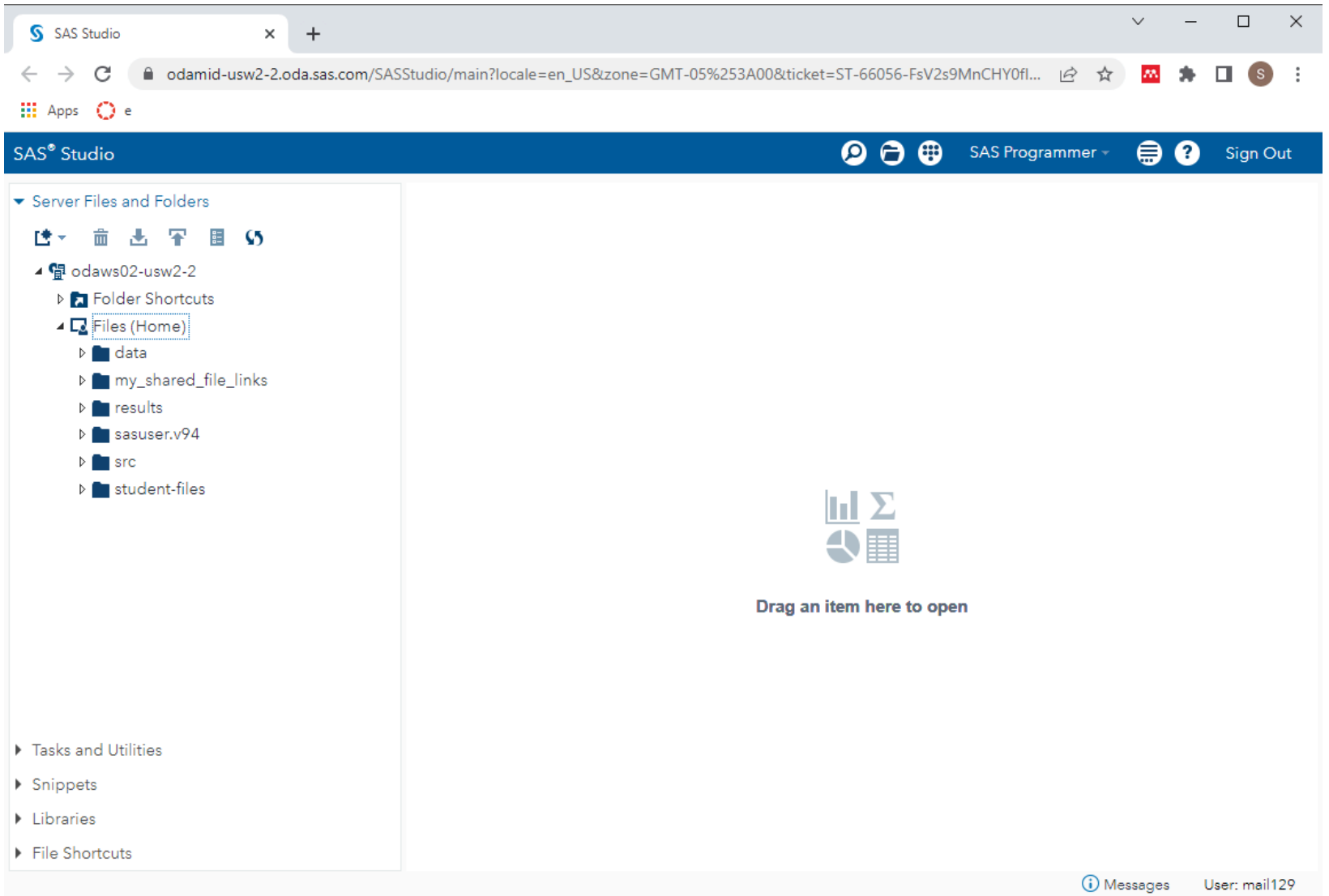

Screenshot of SODA dashboard

Speaker notes

Click on the leftmost icon beneath “Server Files and Folders”.

Create three directories: data, results, and src.

# SODA studio (3 of 3)




Screenshot of SODA dashboard

Speaker notes

Create three directories: data, results, and src.

There are some folders already in there from SAS, my_shared_file_links, sasuser.v94, and student-files. Don’t worry about these folders for now.

# Directory structure

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

Speaker notes

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

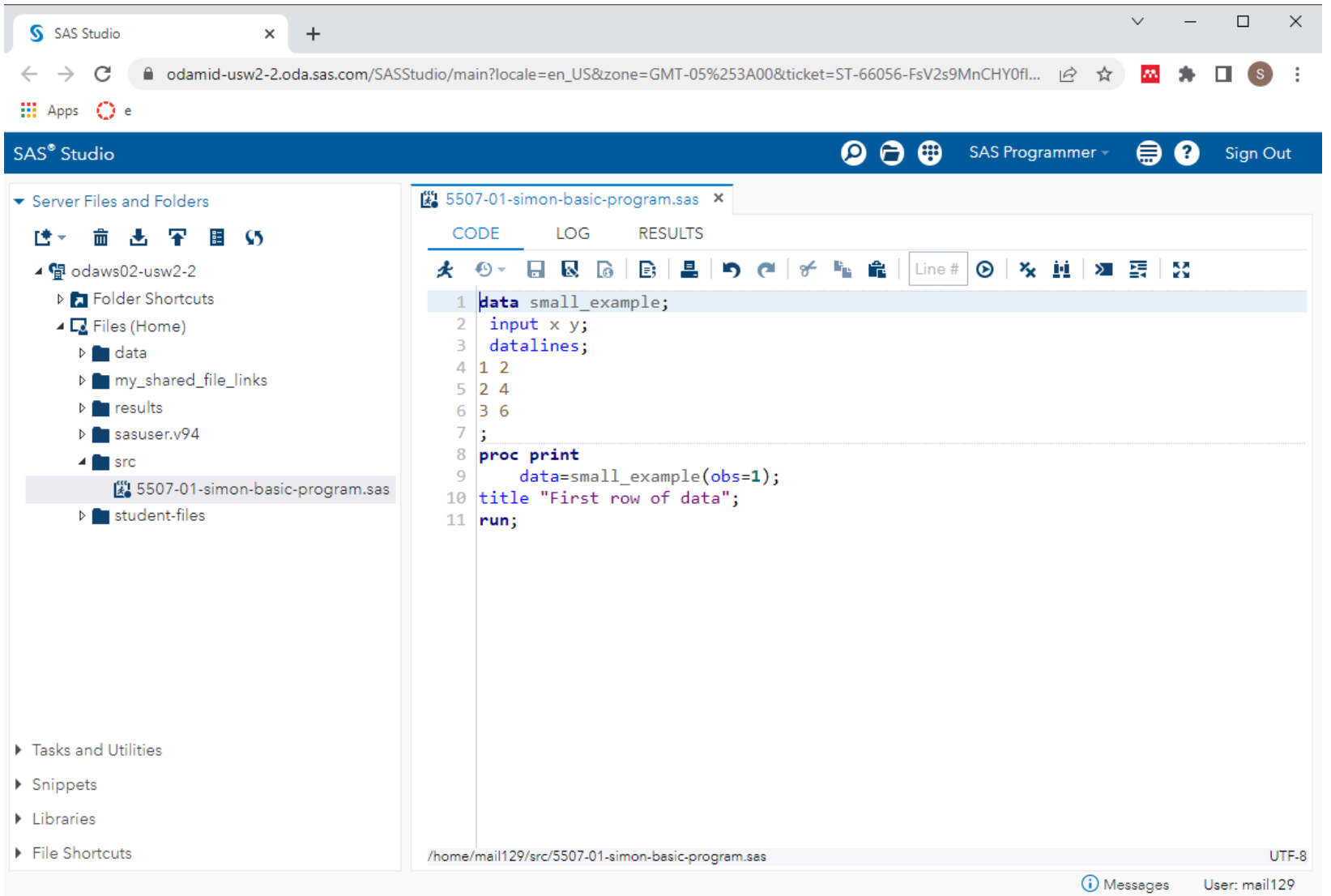
Store all your programs for this class in a single directory. You can call it “sas” or “5507” or anything you like. Some people may prefer to create a separate directory for each module in this class. That’s fine also.

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 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.

# Break #2

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

# SAS program editor




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 (1 of 2)

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

Speaker notes

Here is the program I want you to type in.

I like to put lots of blank lines in.

The data statement creates a dataset with the name “small_example”. 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 once the program is done. We’ll explain this further in a later video.

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.

# SAS Test program (2 of 2)

```
proc print
    data=small_example(obs=1);
  title "First row of data";
run;
```

Speaker notes

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. I like the obs option a lot.

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.

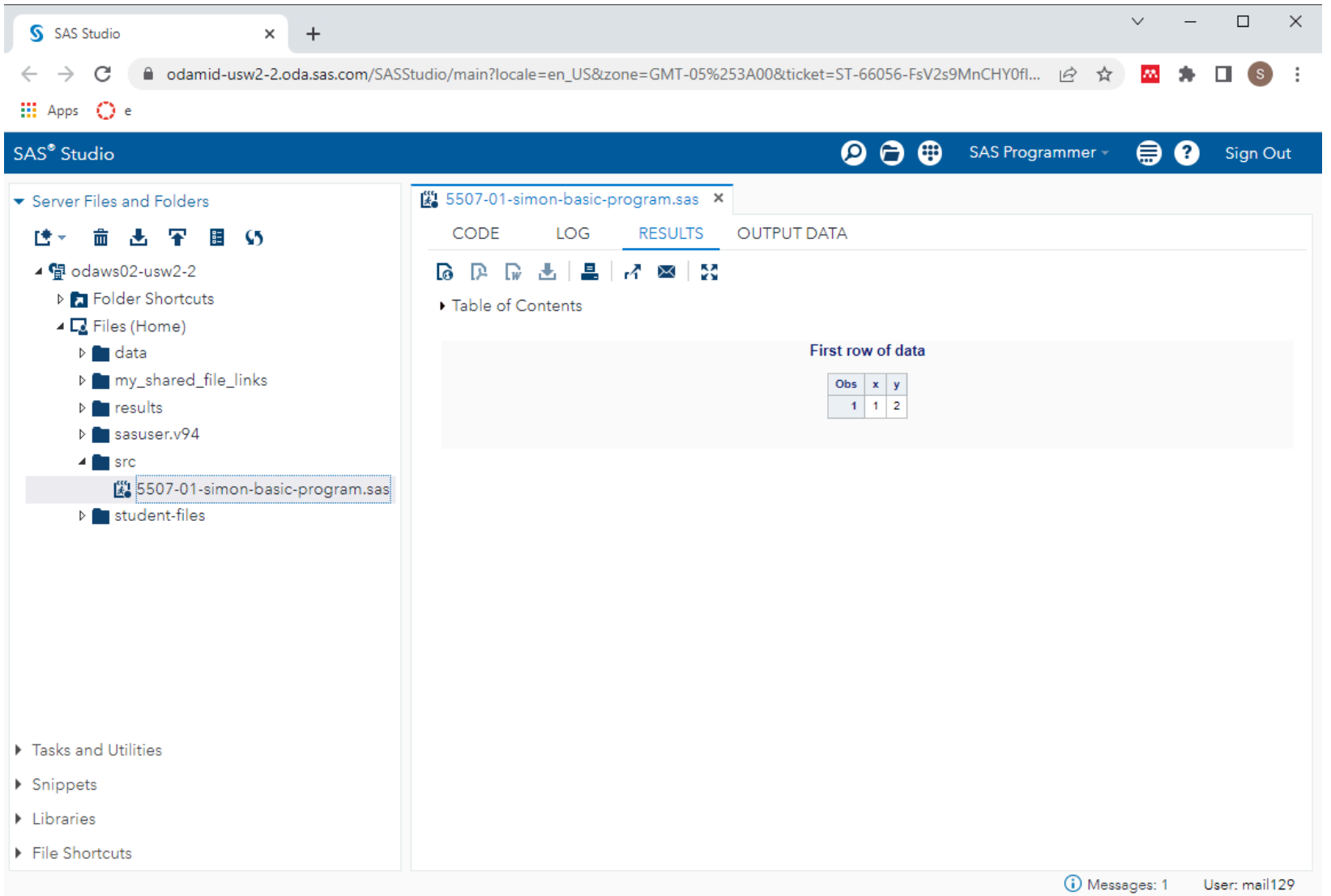
The thing I always worry about is leaving out a semicolon. Please watch these closely.

Be careful and back up your programs regularly. There is no autosave feature in SAS.

When you finish the program, save it in the src directory. Then click on the run button. The run button is a guy who looks like he is running.

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

# SAS results window (1 of 2)

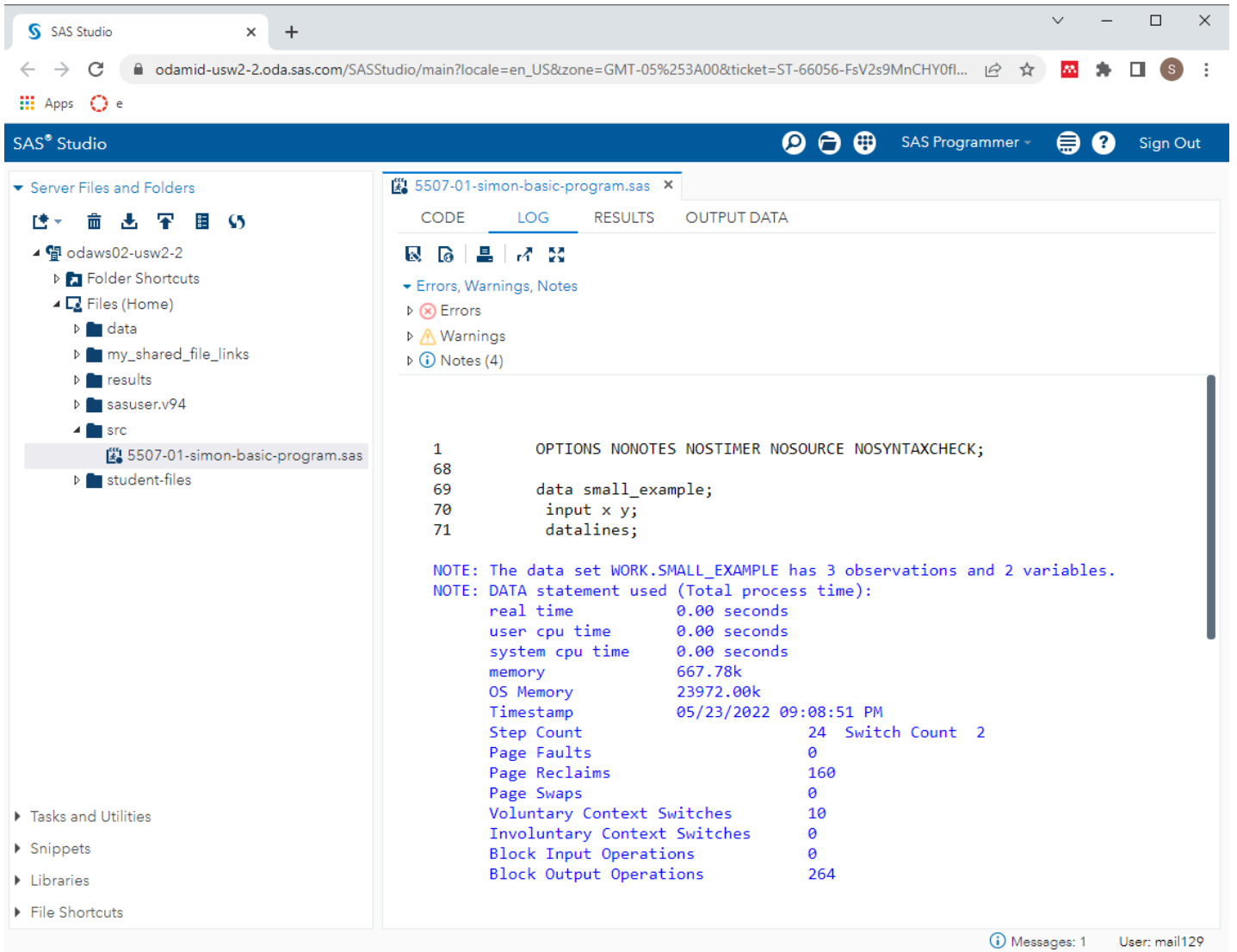



Screenshot of results window

Speaker notes

When the program runs, your output appears in the results window.

# SAS log window (1 of 4)

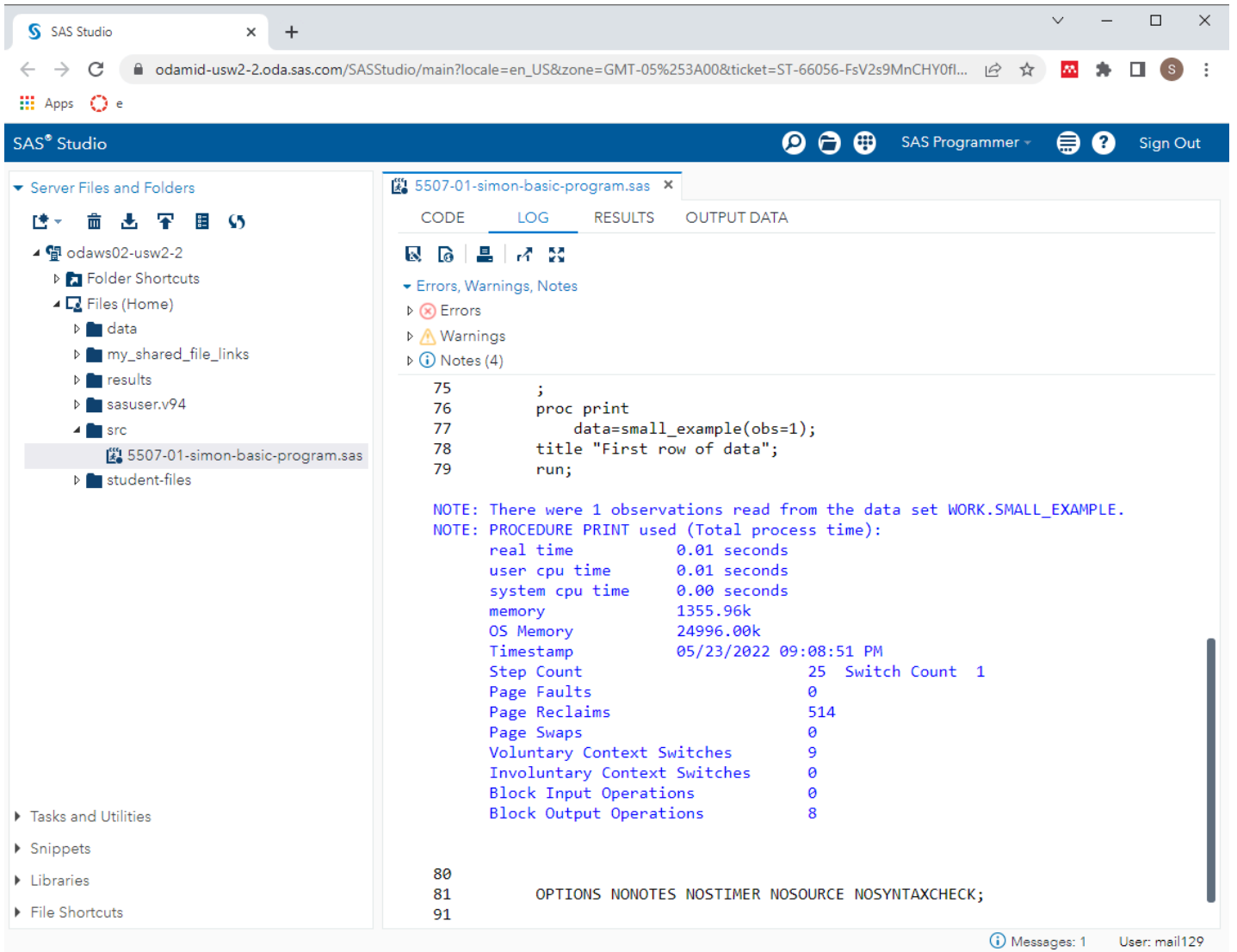



Screenshot of log window

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 4)




Screenshot of log window

Speaker notes

Always start looking from the top, and scroll slowly down to the bottom. No warnings or error messages here either.

# SAS log window (3 of 4)

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

NOTE: The data set WORK.TEST_EXAMPLE has 3 observations and 2 variables.
```

Speaker notes

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

# Log messages (4 of 4)

```
75          ;
76          proc print
77              data=small_example(obs=1);
78          title "First row of data";
79          run;

NOTE: There were 1 observations read from the data set WORK.SMALL_EXAMPLE.
```

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

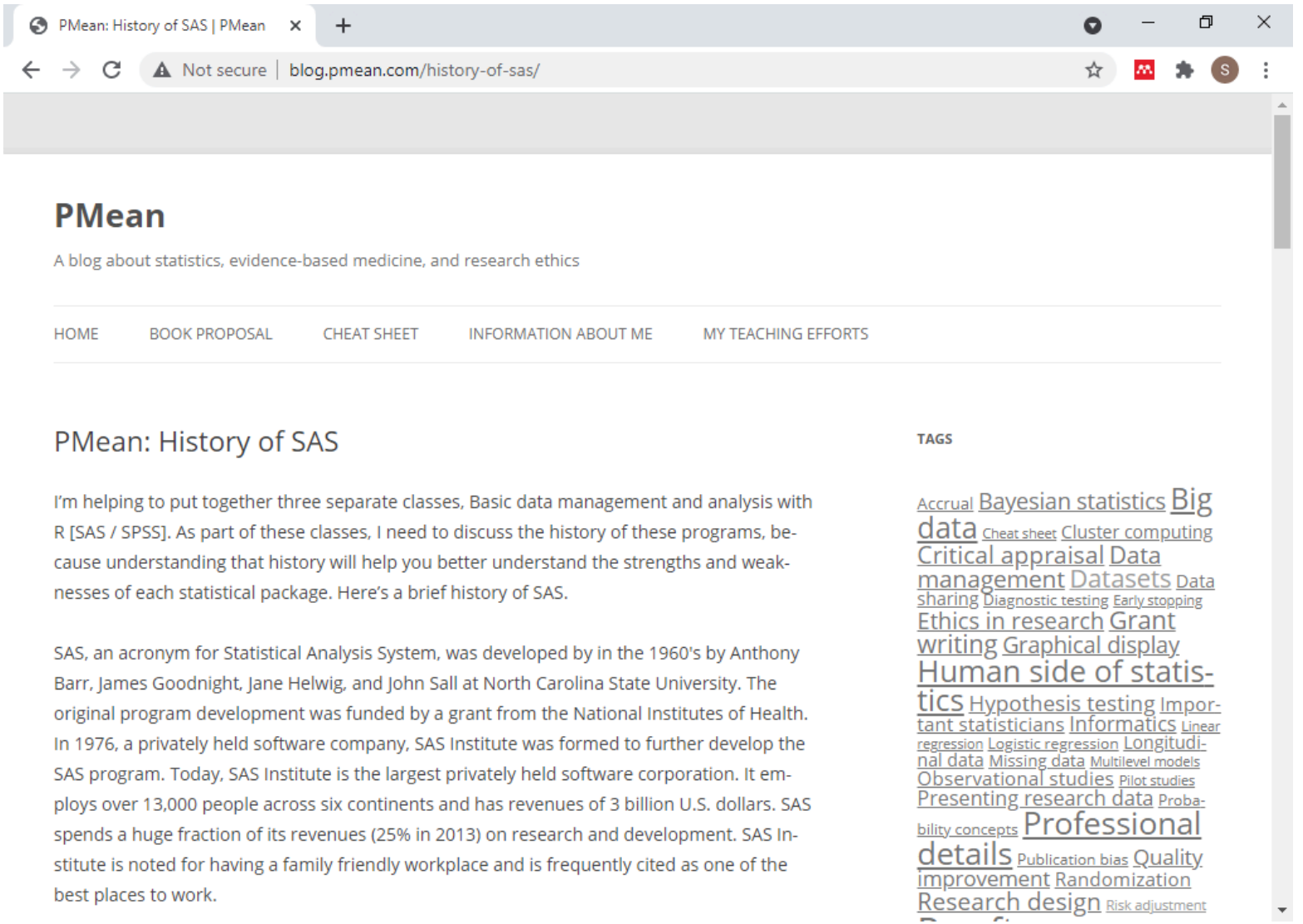
Output is tricky. I want to talk in more detail later about this, but you can take the output and save it several different ways.

# Live demonstration (1 of 5)

# Break #3

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

# History of SAS, blog post




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.

# Origins 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.”

# Corporate structure

- 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.

# Licensing and training

- SAS licensing model
  - Great for large organizations
  - Prohibitively expensive for individuals
- Excellent training resources
  - SAS publications
  - Certification program
  - SAS user conferences

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.

# Other products from SAS Institute

- JMP, 1989
- Viya, 2017

Speaker notes

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 you have learned
  - History of SAS
- What’s coming next
  - Documentation header

# Documentation header

```
* 5507-01-simon-documentation-header.sas
  author: Steve Simon
  date: created 2022-06-06
  purpose: to read and print a small dataset
  license: public domain;
```

Speaker notes

Here is a documentation header. It belongs on the top of every SAS program that you run. Suman is going to modify the documentation header in a live demo.

# Live demonstration (2 of 5)

# Break #5

- What you have learned
  - Documentation header
- What’s coming next
  - Permanent storage

# Permanent storage (1 of 4)

```
* 5507-01-simon-permanent-storage.sas
  author: Steve Simon
  date: created 2022-06-06
  purpose: to read and print a small dataset
  license: public domain;
```

Speaker notes

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 (2 of 4)

```
libname perm "q:/introduction-to-sas/data";
```

Speaker notes

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 (3 of 4)

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

Speaker notes

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 (4 of 4)

```
proc print
    data=perm.small_example(obs=1);
title "First row of data";
run;
```

Speaker notes

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 2022-06-06
  purpose: calculate descriptive statistics
    for strored data
  license: public domain;
```

Speaker notes

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

```
libname perm "q:/introduction-to-sas/data";

proc means
    data=perm.simple_example;
  title1 "Descriptive statistics";
run;
```

Speaker notes

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.

# Live demonstration (3 of 5)

# Break #6

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

# Saving output as pdf (1 of 4)

```
* 5507-01-simon-save-output-data.sas
  author: Steve Simon
  date: created 2022-06-06
  purpose: to read a small dataset
    from a separate file
  license: public domain;
```

Speaker notes

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

Here is the documentation header.

# Saving output as pdf (2 of 4)

```
libname perm "q:/introduction-to-sas/data";

ods pdf file=
    "q:/introduction-to-sas/results/5507-01-simon-save-output.pdf";
```

Speaker notes

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

# Saving output as pdf (3 of 4)

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

Speaker notes

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

# Saving output as pdf (4 of 4)

```
proc print
    data=perm.small_example(obs=1);
title "First row of data";
run;

ods pdf close;
```

Speaker notes

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

# Live demonstration (4 of 5)

# Break #7

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

# Reading data from a file (1 of 4)

```
* 5507-01-simon-read-data.sas
  author: Steve Simon
  date: created 2022-06-09
  purpose: to read a small dataset
    from a separate file
  license: public domain;
```

Speaker notes

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 (2 of 4)

```
libname perm "q:/introduction-to-sas/data";

filename rawdata
  "q:/introduction-to-sas/data/six-numbers.txt";

ods pdf file=
```

Speaker notes

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 (3 of 4)

```
data perm.small_example;
 infile rawdata;
 input x y;
run;
```

Speaker notes

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 (4 of 4)

```
proc print
    data=perm.small_example(obs=1);
title "First row of data";
run;

ods pdf close;
```

Speaker notes

The last part of the program remains unchanged.

# Reading data from a file, part 4

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

Speaker notes

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.

# Live demonstration (5 of 5)

# Summary

- What you have learned
  - About this course and your instructors
  - Where you can get SAS
  - Your first SAS program
  - History of SAS
  - Documentation header
  - Permanent storage
  - Saving your output
  - Getting data from a file