**MATLAB Basics for Psych 4450 - Part 1**

Prepared by Aahana Bajracharya ([aahana@wustl.edu](mailto:aahana@wustl.edu))

If you have any questions about MATLAB or the class, feel free to email me, post to the #matlab channel on Slack (everyone can benefit!), direct message me on Slack, or set up a time here <https://calendly.com/aahanabajra> to chat via Zoom.

If you run into issues or have specific questions about this document, you can also make a comment and I will try to update the document to address your question.

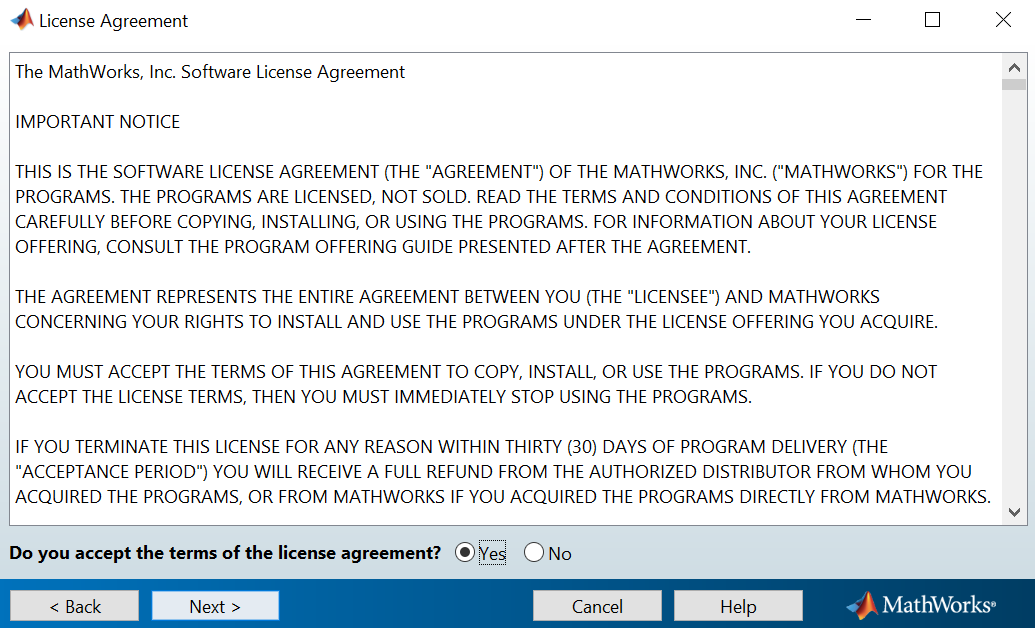
This course will require you to use MATLAB in the context of running SPM (See **SPM Basics for Psych 4450**). But to do that, it will be helpful to get some basic ideas to start using MATLAB. In this part, I will be covering topics such as becoming familiar with the MATLAB interface, variables, loops, vectors, functions, reading data, and plotting simple figures. **Part 2** of the MATLAB Basics guide will focus on using MATLAB to navigate neuroimaging data.

## 1.0 Installation

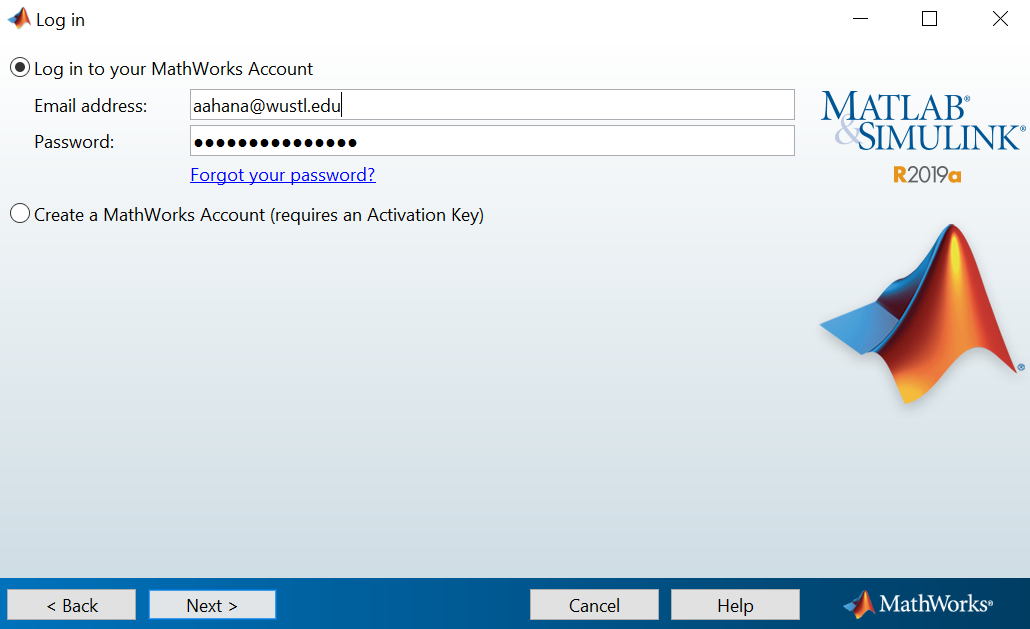
You can get access to MATLAB through WashU: <https://sts.wustl.edu/matlab-free-for-students/> (see “installation instructions” PDF). If you have version compatibility issues, please refer to the information here <https://www.mathworks.com/support/requirements/previous-releases.html> to figure out which version of MATLAB works best for you. We recommend that you download a version of MATLAB 2018 or higher.

After you download your installation file, run the executable file. The next steps will look something like this if you get MATLAB through the WashU license (might have a slightly different interface for the more recent version).

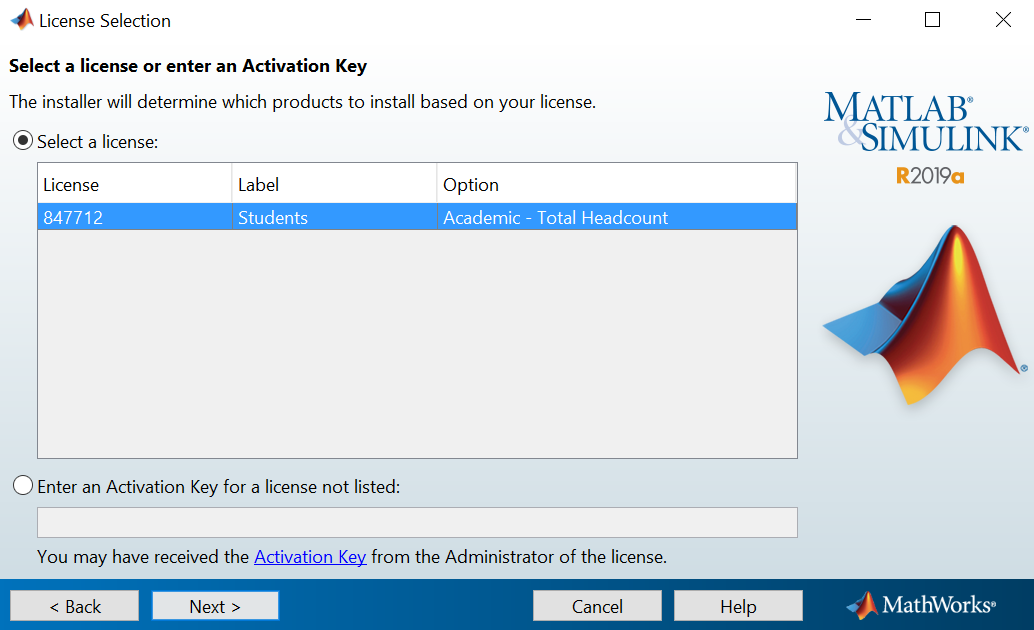
**Step 1: License Agreement**



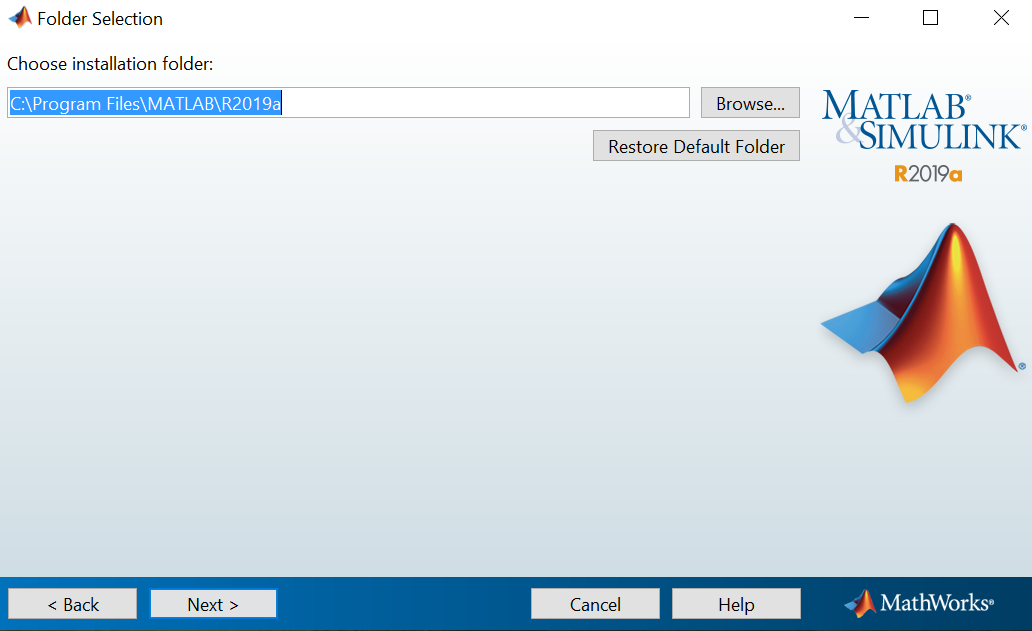
**Step 2: Log in to Mathworks (WUSTL Key or the same info you used to download the installation file)**

****

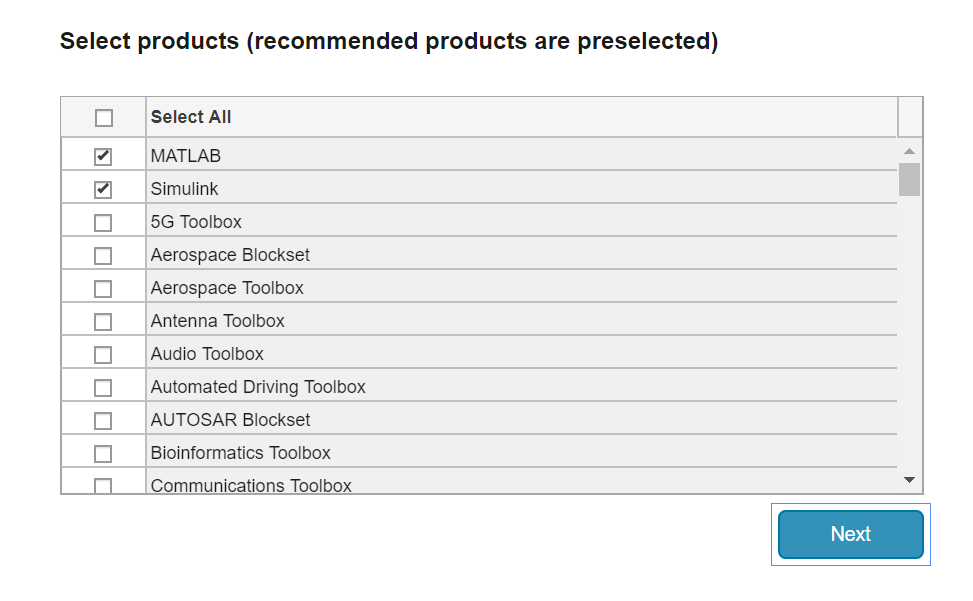
**Step 3: License Selection**

****

**Step 4: Destination Folder Selection**

****

**Step 5: Product Selection (MATLAB, Simulink, Statistics and Machine Learning Toolbox, Symbolic Math Toolbox) are preselected.**

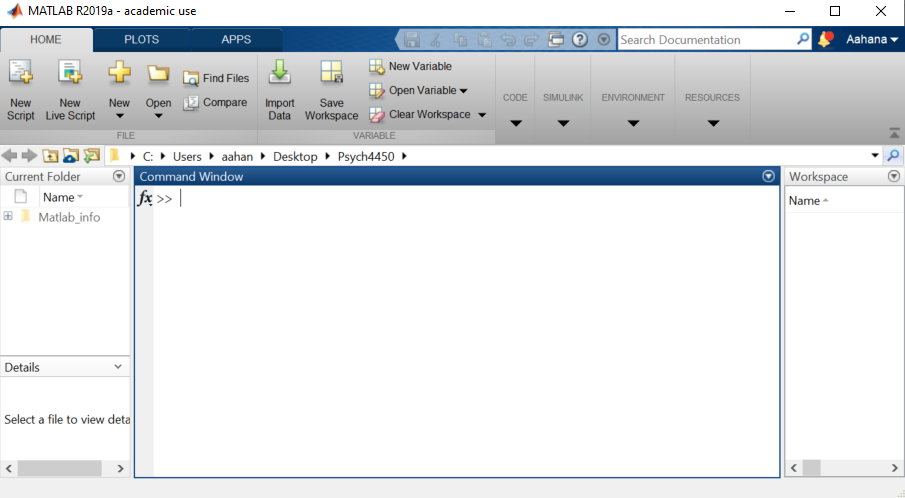
****

For our purposes, the preselected options are sufficient (Note: we will not be using Simulink, so if you are concerned about download time or space, you can uncheck this). The installation will take slightly longer depending on the number of products you select.

The next steps should be pretty intuitive (Let me know if you have any trouble). The default installation of MATLAB occupies around **11GB.** Please make sure that you have enough space.

## 1.1 MATLAB interface

Load MATLAB. You should see an interface like this.



The **Command Window** allows you to run lines of code.

The **Workspace** will show you the variables that have been executed.

The first thing to do is to make sure we are in the desired folder where we want our code to be saved or where our data is located. You can see this information in the **Current Folder** panel on the left. But you can do much more on the **Command Window** with these simple commands. After you type the commands, hit enter to execute it.

|  |  |
| --- | --- |
| Type **cd** in the Command Window to check your current directory.  If you want to step out one level in our path, type **cd ..**  Type **cd** again to check  Or you can type **cd C:\Path** to set it to something different |  |

## 1.2 Variables

A variable refers to a storage location that you have assigned a name (a letter or a word) and a known or unknown quantity. The quantity can be strings or numeric.

An important point to note is to make sure that your variable names are sensible and short (but maybe not too short). It cannot contain any spaces, MATLAB keywords, special characters (except underscore), and cannot begin with a number (More info [here](https://www.mathworks.com/help/matlab/matlab_prog/variable-names.html?s_tid=srchtitle)). Variable names are also case sensitive, variable **a** is not the same as **A**. Variables can contain integers, floating points, strings or booleans (1 or 0).

Let’s try out some basic mathematical operations to understand this better.

|  |  |
| --- | --- |
| Here, **num1, num2, num3, num4,** and **num5** are all variables that have been assigned numerical values.  In the 3rd command, we did not assign the value to any variable, thus it is not stored yet. It is only temporarily assigned to **ans**. The value of which can change if you carry out another operation without assigning a variable.  The 4th command assigns the output of the addition to a variable named **num3**.  Next, **num4** is created by multiplying a constant to our variable **num3**.  Similarly, **num5** is created by raising the power of **num1** by 2.  You can also reassign values contained in the variables to other existing variables. | Adding a semicolon at the end of the command suppresses the content from being displayed.  You can also see the values inside all the variables in the Workspace window on the right. |

Type **clc** to clean the Command Window

You can also type **clear** or **clear all**, but this will also clear your variables in the workspace

Typing **clear NameOfVariable** will erase only the variable of your choice

|  |  |
| --- | --- |
| **Try it yourself:**  a = 10;  b = 20;  c = a + b  d = a - b  e = a \* b  f = a / b  g = a \ b  x = 7;  y = 3;  z = x ^ y | **Answer:**  c = 30  d = -10  e = 200  f = 0.50000  g = 2  z = 343 |

All the operations that you carried out above can also be done by using functions, more on that later (**see Section 1.5**).

## 1.3 Vectors

A vector is a matrix with either one row or column. In MATLAB, a vector is most easily created by using square brackets [ ].

|  |  |
| --- | --- |
|  | Row vector dimension 1 X 3  Column vector dimension 3 X 1  Matrix dimension 2 X 3 |

In the command window

|  |  |
| --- | --- |
| Here, x is a row vector and y is a column vector.  You create a row vector by separating the scalars with a comma and use a semicolon to create a column vector.  You can create the same row and column vectors by using a colon instead.  x= [1:4] will give the same output  y= [5:8]’ will give the same output. The ’ after the square bracket will convert a row vector to a column vector. In mathematical terms, this is the transpose.  For other increments and decrements,  x = [1:2:10]  Here, the output is a row vector with all the numbers between 1 and 10 in increments of 2.  y = [1:2:10]’  Here, the output is a column vector.  z = [10:-2:0]  Here, the output is a list of numbers between 10 and 0 in decreasing order with decrements of 2. |  |

## 1.4 For Loop

As the name suggests, loops enable the execution of a command or a set of commands a number of times. One of the most common types of loops in MATLAB is the ‘for’ loop. The general syntax of a for loop is as follows:

for index = values

Statements or commands

end

|  |  |
| --- | --- |
| The goal of the code on the right is to obtain a sum of the squares of all the even numbers in the given range.  The loop iterates through the list of numbers contained in the variable, i = [0,2,4,6], one at a time.  When i = 0 , total = 0 + 0 ^ 2 = 0  When i = 2 , total = 0 + 2 ^ 2 = 4  When i = 4 , total = 4 + 4 ^ 2 = 20  When i = 6 , total = 20 + 6 ^ 2 = 56  Notice that the value contained in total changes after each iteration of the loop. |  |

**Try it yourself:** write a for loop to obtain the sum of odd numbers in a given range.

## 1.5 Functions

Functions are a set of commands that accept input and return output after performing some action. Let’s use MATLAB’s built-in functions to do some of the examples that we have seen above.

|  |  |
| --- | --- |
| Here, **num** is a variable consisting of numbers from 1 to 10 with the default increment of 1.  **sum** is a built-in function that takes the sum of all the numbers contained in the variable **num**.  The For loop example in **section 1.4** can be simplified further by using functions.  **num** consists of even numbers between 0 and 6  **sum (num .^2)** squares each of the numbers contained in the variable **num** and then adds them up.  The ‘.’ represents an element-wise operation. |  |

[Here](https://www.mathworks.com/help/matlab/referencelist.html?type=function&category=index&s_tid=CRUX_lftnav_function_index) is a list of all the built-in functions in MATLAB. The commands **clc** and **clear** mentioned in **Section 1.2** are also functions.

You can explore what these functions do by typing **help FunctionName** onto the command window. This will give you examples of what inputs you need to supply to your function and also examples of how the function can be used.

## 1.6 Writing/Opening MATLAB files

All of the command examples above have been executed directly on the command window. However, for more involved tasks, it is better to write your own executable script. MATLAB files have the extension **‘.m’**. To write your own script, click on **New** (yellow +) under **Home** tab and select **Script**. This will open an untitled Editor window above the Command Window.

A demo MATLAB file (**Psych4450\_demo.m**) can be found on Canvas. Download this and make sure that this file is located at a location accessible by MATLAB, more info on setting path below **(Section 1.8.1).** Double-clicking a **.m** file will open it in MATLAB. In order to **run** the file hit the **green play button** under the **Editor** tab. The output from the script will be displayed in the command window.

Any text followed by % are comments in MATLAB. In the demo script, the first three lines convey information about the creator of the file and give a brief description of the purpose of the file. It is preferable to comment on your code whenever possible. This will especially come in handy when you are sharing your code with other people.

The double percent symbols, %%, separate code to different sections. You can independently execute these code chunks by placing your cursor in any of the sections and clicking **Run Section** under the **Editor** tab.

## 1.6 Reading Data

One of the key strengths of MATLAB is the ability to deal with large volumes of data. Here are some ways in which you can read/write data using MATLAB.

For MATLAB versions before 2019a, the **xlsread** function can be used to read data in a spreadsheet format. But, the **readtable** function is recommended for more recent versions of MATLAB. The documentation on this provides a detailed explanation of the usage <https://www.mathworks.com/help/matlab/ref/readtable.html> .

There are plenty of sample datasets in MATLAB that you can play around with. To load any of the datasets listed <https://www.mathworks.com/help/stats/sample-data-sets.html> , you can type **load NameOfDataset** on the command window. Once you load the datasets, you can view the contents of the dataset from the workspace window. Refer to **Psych4450\_demo.m** for an example.

## 1.7 Plotting

Let’s try loading polydata.mat which is a sample data for polynomial fitting and try some plotting.

You can type these line by line on a command window or execute them altogether as a script. Any two lines of command can also be entered in a single line as long as it is separated by a semicolon.

load polydata.mat; % loads the built-in dataset

figure; % generates a new figure

plot (x,y,'r'); % plots the variable x against y, line color red

hold on % holds the most recent plot and enables another plot to be overlayed

plot (x1,y1,'b') % plots the variable x1 against y1, line color blue

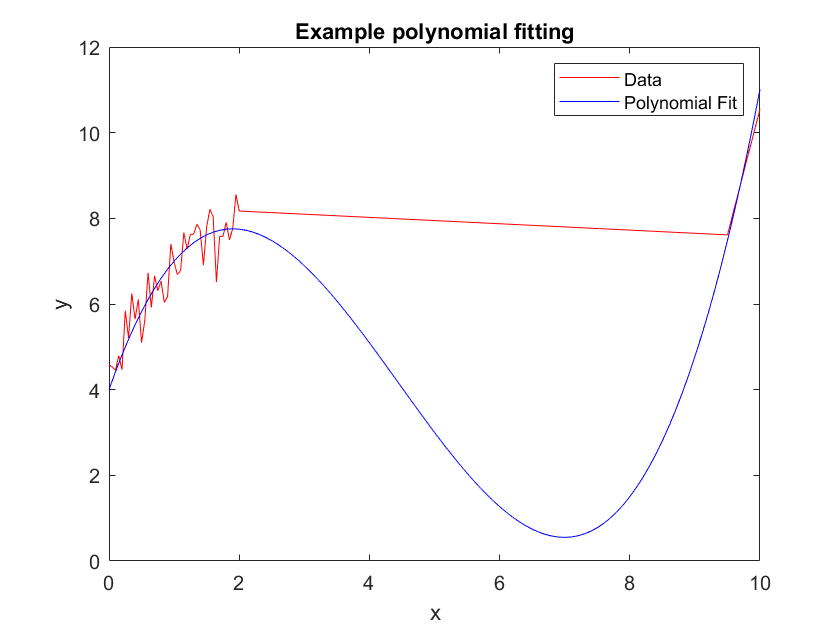
title ('Example polynomial fitting');

legend ('Data','Polynomial Fit');

xlabel ('x');

ylabel ('y');

hold off



Refer to **Psych4450\_demo.m** for the code.

## 1.8 Miscellaneous

### 1.8.1 Setting a path

A path refers to a subset of all the folders in the file system that **MATLAB** uses to locate files efficiently. It is good coding practice to set your path early on. You can either do this on the Command Window or through the tool strip.

Home > Set Path > Add Folder > Save > Close

Refer to **Section 1.1** on **SPM Basics for Psych 4450** guide for a visual demo of this process.

### 1.8.2 Saving the workspace

To save the variables in your current workspace for future reference, type

**save NameOfChoice.mat** on the command window. You can load this by double clicking the filename or by typing **load NameOfChoice.mat** on the command window**.**

### 1.8.3 Resources

* The MATLAB Onramp and **MATLAB Fundamentals** course <https://matlabacademy.mathworks.com/> might be helpful to get a quick overview.
* MATLAB Online is a browser version of MATLAB <https://www.mathworks.com/products/matlab-online.html>.
* Dr. Peelle has lots of resources listed here <http://jonathanpeelle.net/learning-matlab/>.

**MATLAB Quiz Mock Questions:**

1. Which of the following can be used as a variable name?
   1. 200Num
   2. Num of items
   3. dist
   4. n!$b
2. A semicolon in MATLAB can be used to do the following
   1. construct arrays
   2. suppress output from a **MATLAB** command
   3. separate commands entered on the same line
   4. all of the above
3. Which of the following will generate a set of even numbers between 0 and 10 in descending order
   1. 1:2:10
   2. 0:2:10
   3. 10:-2:0
   4. 1:10
4. What is the value of the variable **Val** at the end of the for loop

Val = 0

for index = 1:3:9

Val = Val + index;

end

* 1. Val = 12
  2. Val = 3
  3. Val = 16
  4. Val = 0

1. Which of the following is used before a comment in **MATLAB**?
   1. #
   2. &
   3. $
   4. %

**Answers: 1 C, 2 D, 3 C, 4 A, 5 D**