Hi everyone,

Thanks to everybody who decided to virtually join for this first webinar version of the why&how series hosted by the martinos center.

So, the topic of today’s talk will be an introduction to the MATLAB computing environment. Ideally, I would have asked you how familiar you were with programming in general and with MATLAB, in particular, but given this new online settings I will make the call myself and I will be introducing the basics of how to use Matlab, with a particular focus on people with little-to-no CS background. We will see later on what I mean by that.

---

This is the overall outline of the talk, but without further ado, let’s just go ahead and clarify ‘what’ are we talking about when we say ‘Matlab’.

---

So, the word MATLAB comes from the full name of Matrix Laboratory, and as such we can already imagine how its main key feature should be its ability to deal with linear algebra operations, and matrix calculations. And this is true, and around the idea of simplifying as much as possible matrix-vector operations, Mathworks, which is the company behind Matlab, has built an entire framework comprising a high-level scripting language, and entire, interactive computing environment based on a perfectly tailored graphic interface, and an ever-growing number of additional toolboxes designed to make of MATLAB possibly one of the easiest options when it comes to automating complex processing pipelines, visually exploring datasets, algorithm prototyping, and model development.

---

Let’s now focus on a new question: Why Matlab? How does it compare with other alternatives?

---

When it comes to programming languages, it’s hard not to mention giants like C, C++ or Fortran. Actually, the parallel between Matlab and this class of languages is not exactly straightforward, as they have many differences. Mainly, the target audience is different. Low- level, general-purpose, compiled languages like C++ are the go-to choice when it comes to building applications from the ground up, with fast execution and high performance in mind. On the opposite, Matlab is a high-level language, which should be much easier to learn for people with minimum CS expertise. It already offers a full package of professionally developed tools and applications that often you can use without even needing to write a single line of code. Clearly, this comes with the downside that Matlab is a commercial product and a rather expensive one.

---

The rising star of programming languages, especially in a scientific context, is obviously Python. While still being a true, general-purpose language, python shares with Matlab a few more similarities, like the possibility to be “treated” also as a scripting language, the easier portability related to it being an interpreted language (instead of a compiled one), and a generally more relaxed syntax, when compared to C/C++.

---

The roles kind of flip when we compare Matlab with a scripting language like R. In this case, it is R that is a more specific language, mostly focused on data analysis and statistics, with Matlab able of offering a wider range of tools, while still being a pretty good option to tackle a statistics problem. And Matlab is faster than R, and that’s not something (being faster) that happens often when we talk about Matlab.

---

Ok, so there would have been much more to say about why we could choose Matlab over other options, but let’s just sum it up saying that it is, as usual, a matter of pros and cons. Matlab is really easy to approach, it has an entire company behind it focused on providing the user with the best experience no matter the level of knowledge about programming and coding, but we have to literally pay a price for that, because Matlab is a commercial product that requires a license to be used, and all the toolboxes that extend its functionalities are generally not included in the base license. And this is clearly something that we have always to keep in mind.

---

But let’s now move on and see how we can actually start to use Matlab. In this section, we are going to focus on basic concepts of Matlab as a programming language, to make you able, hopefully, to read and write simple code in Matlab.

---

So, when you fire app your Matlab executable, this nice GUI pops up to give you everything you need to get started writing some code and running some scripts. The main element here is the command windows. You may think of it as a Unix terminal, but it is actually closer to a python console. In there you can write any kind of command using Matlab’s own language, and as soon as you press Enter they will be immediately executed. On the bottom right you have the command history, where every command you typed in the command window is stored for you to backtrack everything you did. I have never actually checked but a fair guess is that it is able to store a few weeks/months of history. Above the command window there is the Workspace. This is one of the best features of Matlab's GUI as in there you see a list of every variable you defined by running your code so that in every moment you can be aware of what names are already taken, and what variable you can call from the command windows to interact with them. On the left, we have the file explorer which shows us the folder we are in and everything inside of it. This is important because this is the way we can decide the working directory in which our code will be executed, and because if we need to read some files, we need to know where we are and if we can access it or not. On the top, there’s the toolbar menu, which reflects the style of the ribbon menu of the Office suite. Using commands in this menu we can perform most of the operations related to writing and running Matlab scripts.

---

Now that we know we can use the command windows to write some code, let’s dive deep into the basics of Matlab syntax. The first important thing to keep in mind is that Matlab is a commercial product, and if there is something that commercial products are generally really good at is documentation. There are many ways you can ask for help in Matlab, but the two most common ones are typing help or doc, followed by the name of the function you want to know more about. Using help, you will get a summary of the main properties of a function printed directly in the command window. Using doc, instead, you will fire up the actual documentation, with an in-depth description of everything you can do with a certain function, examples of usage, and also a lot of theory and implementation details. So I’d really suggest you use it as much as possible.

---

After asking for help, the second most important thing to remember is that Matlab stands for Matrix Laboratory, so knowing how to define matrices, vectors, array, is key. Something that makes Matlab way easier the most programming languages is that we don’t need to be too detailed when instantiating variables. We don’t need to specify the type or allocate memory. We just need to choose a name and set a value. One value means our variable is a scalar, more than one means that it is an array. If we want to create a vector, a 1D array, we can use square brackets and list all the values in it. If we use commas to separates values, we will get a row vector,

--

if instead, we use semicolons, we will get a column vector.

--

It may be easier to grasp the difference by looking at a matrix. This is how we define a 3 by 3 matrix: we use commas to separate elements on the same row, and semicolons to separate different rows in the matrix. Other interesting characters it is important to keep in mind are the percentage sign, which is used for inline comments, and the semicolon at the end of each instruction.

---

If we skip the semicolon at the end of each line, after each instruction we will get the output directly printed on the interactive console. Most of the time we don’t want that because it is annoying, but also because printing to console slows down the code significantly.

---

We are not going through all of this command now, but as you could imagine, typing all the values in your array is not the only way you can use to create them. There is a great list of functions you can use to instantiate arrays of predefined shape, filled for instance with zeros, ones or random numbers …

---

Once we created our array we may want to access elements of it. This action is called indexing or slicing, and to do that in Matlab, we need to keep in mind that indexing starts from 1, and not from 0 as usual for many other languages. For instance, if we want to access the element 3,2 of this matrix A, we will use parenthesis, and we will get the element on the third row, second column.

---

If instead, we want to extract the entire first column from A, this is as simple as using a colon meaning that we want to select ALL elements on that dimension of our array and the number 1 as we want the first element of the second dimension. The same will do if we want an entire row, but we will have to switch the order.

---

If instead, we want just a subset of the second row, we can use this syntax, where we specify the first and last column we are interested in. Differently from Python and other languages, both indexes 2 and 5 will be included in slicing.

---

More then that, operation of indexing and slicing can be directly combined with other operation. For instance, we can sum all the element on second row by simply using this command

---

Or we can compute the maximum value of the 3rd column in this way.

---

So this covers pretty much the basics of array creation and manipulation. Then, as I said, Matlab is really built around the idea of making it as easier and intuitive as possible working with multidimensional arrays. In this slide, I list a lot of basic operations you can do on a vector, a 1D array. We are not going to discuss all of them, but you can use it as a cheat sheet if you want. I want to focus your attention on a couple of interesting operations. Let’s say we have 3 vectors. A and B are row vectors; C is a column vector. Ok? Transposing a vector in Matlab is as simple as using an apostrophe after the name of the vector. As you can see, we get a column vector as a result.

---

Then we have an interesting syntax choice when it comes to performing element-by-element operations between vector with the same size. Elementwise multiplication between two row vectors can be performed by pre-pending a dot to the star operator. As a result, we get a new row vector, with each element resulting from …

---

And the same goes for elementwise division, obviously.

---

If we don’t use a dot and we just call for a ‘star’ product, we are implicitly asking Matlab to perform a dot-product between the two vectors. Clearly, this can be done only if the dimension of the shape of the two arrays is suitable for dot product. In this case, we are multiplying a 1 by 3 vector for a 3 by 1 vector, and the result is a scalar.

---

Working with matrices is no more difficult than working with arrays. The interface to basic operations is the same: we can still use dot operator for element-by-element operations, the apostrophe for transposition, and so on.

---

---

Moving on from data type and basic operations, as any proper programming language, Matlab needs tools to control the flow of how the code is executed. Every main type of control flow is there. For instance, we can use if-else condition to execute a snippet of code only if a condition is matched. We just need an IF opening keyword and an END closing keyword to determine the portion of the code interested. We don need brackets or special indentation. ELSE and ELSEIF conditions are also available to test multiple options at once.

---

Also, a FOR loop is available to allow you to run a piece of code for a fixed and known number of times. And the way you call is again by using a FOR opening keyword, and END to close the loop. You can also nest more the one loop inside the other.

---

The last major control flow mechanism is the while loop, which you can use when you need to run some code multiple time but you don’t have an exact number of repetitions. The while loop checks the value of a condition and keeps looping until that condition is met. Or you can use a break statement to exit the loop while checking a secondary condition.

---

Ok, we made it this far covering as quickly as possible the basic syntax of Matlab language.

Let’s now move a step forward and see the best practices of code writing.

---

I said previously that you can write code in the command window. Well, that doesn’t mean that you should! Command window is great to check variables, calling scripts, debugging and so on, but it’s not a great place to write actual code. If you click the first button on the menu bar, which says “new script”, a new window will pop up which is the Editor window.

---

Matlab’s editor is a dedicate text editor where you can write text and save it as an m-file. And m-file, at its core, is nothing more than a text file containing a sequence of Matlab instruction that will be executed one at a time, top to bottom. Running a script of execting each line of it in the command windows produce the exact same result, but allow you to keep track of what you are doing, saving your script, and reuse it later on, maybe on a different dataset.

---

It is as easy as it sounds, but this comes at the cost of some ‘security’ issues. In fact, scripts do not have a dedicate memory space, and they share the same workspace as the command window. This means that if we create a variable A using the console, and then we run a script which redefines a new variable A, we lost the original one and there is no coming back. Being aware of this is a matter of practice, but it is something you need to be aware of and careful about while using scripts.

---

If we want to have better control over what our code does, and which part of memory it can read and write, we can resort to a different type of m-file which is called ‘Function’. Writing a function allows us to encapsulate a set of command in a dedicate space, and to give them a name, an alias, we can use to call that piece of code multiple times during our processing pipeline. The main difference between a script m-file and a function m-file is in the first row of the function file. It starts with a keyword ‘function’ and closes with an ‘end’ keyword like we have ween for all Matlab statements that encloses a portion of our code. Then we can list a set of output variables using square brackets (which avoid the need of using return statements) and input variables using parenthesis. The body of the function will be used to process the inputs and to produce the outputs. We can then give a name to our function and save the file using the same name.

---

Choosing a unique name is important because that’s how we can call the function later on to use its functionalities. And what’s good about functions is that, unlike scripts, they have their own memory space which does not interfere with the main workspace, so we are safe that the only way running a function will affect our code is through the outputs of the function themselves.

---

As a general rule, my suggestion is every time you need to write some code in Matlab try to write only one big script, which will act as your main file, and use it to call as many functions as you can. And try to refactor as much code as possible into as small as possible functions

---

I said previously that Matlab editor is mostly a text editor. Well, a better definition of it is probably a text editor on steroids, as it has a lot, really a lot, of hidden functionalities to assist you while writing code. One of the most interesting ones is that it continuously performs automatic code checking of what you are writing, giving you suggestions about how to improve your code, or warnings about errors you may have done. You will see all of this on a sidebar on the right side of the editor window. You can click on each marker to jump to the relevant line of code and read the message that the editor is trying to give you.

---

Using the editor you can also resort to one of the easiest ways to debug your code “live” while coding. It is as easy as highlighting a section of your code and pressing F9: as long as al the variables needed by that snippet have already been defined in the workspace, you will immediately see the output pop out on the command window, so that you can check for errors, or just see if the outcome is what you were expecting.

---

Clearly, you can also choose to opt for more proper ways to debug your code, and this is done from the editor windows, as well. Again, this is as simple as double-clicking a line number: this will create a breakpoint at that position and the next time you will run the script, the execution will stop right at that point for you to check if everything is going as expected. You may place as many breakpoints as you want in your code, and while executing a green arrow while highlighting the current breakpoints. Debugging is quite intuitive because there is no real ‘debug mode’: if there are breakpoints, running the script will stop at them, if there are no breakpoints, all the code will be executed. That’s it.

---

While Matlab editor is super cool and full of functionalities, in the latest releases Mathworks added also a new type of editor called Live Editor. It shares all the same basic functionalities of the basic editor, but can also combine MATLAB code with formatted text, latex equations, and images. In addition, live scripts can store and display output alongside the code that creates it, allowing you to produce high-quality report files in PDF, HTML or LATEX, in which your code is exported alongside your results. This is basically a duplicate of the idea behind project Jupyter, for those of you who know Python … but don’t tell them I said that!

---

Ok, so, another interesting way Matlab can be very useful for people working in research and data analysis, is because it offers plenty of resources designed for the purpose of data exploration and visualization. The following set of slides don’t want to be an exhaustive list of all the graphic functionalities provided by Matlab, but I thought that it could be interesting to provide you an overview of the most important ones and to show you how creating even very complex images is often a matter of calling one ad hoc function.

---

Let’s start with the simplest one, plotting arrays. The function ‘plot’ requires 2 main inputs: an independent and a dependent variable. And for the latter, it can also be a matrix: in which case, as shown here, all the rows in that matrix will be plotted on the graph versus the independent variable. Then you can personalize almost everything: labels, axis name, title, legend, and so on.

---

Let’s now imagine we have some data and we want to fit a model to it. The results of our fitting would be a continuous curve and a confidence interval. Well, the function ‘line’ uses the same interface as plot and stem, and allow us to easily show upper and lower confidence interval together with the result of the fitting.

---

Something else that can happen is that we want to show two sets of data on the same plot, but if we use the same scale for the y axis, one of them will be almost invisible. Well, using two different y axis on the same plot, in Matlab is just a matter of calling the right keyword, right before executing the plot command.

---

Clearly, 2D plots are not the only thing that can be done. We can use the plot3 function to obtain 3D plots, very useful to show MR spectra, for instance.

---

Or bar, to obtain a bar plot that is by default able to group together data relevant to the same value of the x axis.

---

Histogram, or histogram2 provide straightforward ways to compute and show histogram of arrays or matrices

---

Then, of course, every plotting library worth of its name should give you the full power of assembling your figure as you see fit. In Matlab, this is as simple as calling the function subplot before any other plotting instruction. The first two inputs are the total number of rows and columns of the subplot grid, and the third one is the position in the grid we want to use for the current plot. And now instead of having 4 separate figures, you have just one, with 4 subplots.

---

(quiver)

---

If you want to go even fancier, you can be interested in trying surface rendering, for which Matlab provides, between others, functions like surfc, or mesh, which you can use to visualize brain meshing resulting from your freesurfer reconstruction.

---

And what if you want instead to show images? Well, you can use the ‘image’ function to let Matlab take care of everything or the ‘imagesc’ that allow you more control on the scaling of the visualization. With these functions, you can have fun changing colormaps until you find the best one for your specific purpose.

---

But that’s just the tip of the iceberg. How crazy you want to go with complex visualizations, is totally up to you, your imagination … and obviously, your data!

---

Great, so, I began this presentation saying that I wanted to target an audience of people without a CS background, but so far I have been just talking about writing code, defining variables, functions, scripts, and so on. Well, is it true or not that you can use Matlab even without writing a single line of code? Well, as weird as it may sound, while all other languages, like C++, Python, and even R, to get something done you need to sit down and need to speak the same language as the compiler, Matlab has a lot of tools that allow people who don’t know how to code, or BETTER, that DON’T WANT to code, to enjoy it capabilities.

---

And even better, you don’t need strange workarounds to do that, because a lot of functionalities are already available within Matlab itself. Let’s see what I mean.

---

If we look at the menu toolbar, we can find a tab called APPS. In there you can find a library of Matlab Apps, which are self-contained MATLAB programs that allow the user to complete a task just using an ad hoc user interface. If you expand the drop-down menu …

---

You will see a huge list of app that ready to be used. Well. At least you will see all the apps that come with the toolboxes for which you bought a license, but that’s another story. To understand what I mean, let’s see a couple of examples of such apps.

---

One of the most used and always available is the interactive data import app. Let’s say you have done some processing, maybe in excel, and you have a CSV file you want to open in Matlab. You can either click the import data button, or drag and drop the file in the command window.

---

Either choice will fire up the interactive import app that will scan your file, trying to guess the best way to read it and showing you a preview. Once there, you may adjust some import parameter, like choosing a different column delimiter, if the autodetection somehow failed; or you can decide the type of workspace variable you want to use to read your file; you can choose a strategy to fill missing data, and eventually, you can import what has been read from your file. Quite interesting, you can also save a script or a function for the input: in this way, next time you want to import the same type of file, you can just call the function instead of having to set all the parameters manually again.

---

Ok, so let’s say now that the imported data are some time series we want to fit. Well, instead of going through the doc and looking for the best way to write a function to fit our data, we could give a try at the curve fitting app available. This is as simple as selecting from the workspace the independent variable, and our noisy time series and picking a model from a dropdown menu. Depending on the chosen model we may have to set some additional parameters, or we may want to write down our own custom function. After that, we may want to adjust some optimization settings and we are good to go.

---

The app will immediately show us the fitted curve and the residual plots. On the left panel we could read all the details about the estimated parameters, R squared, error and so on. And if we like the result we can export them to Matlab’s workspace. And as before, we could also save a fit object that we may call later on to fit the same model to another dataset. And all this without writing a single line of code ourselves.

---

Other interesting apps are for example the DICOM browser, which allows you to scan the header information of all dicoms in the current folder and provides also a few details and a preview of the images. Once we find the right image, we may export it as a matrix, or send it to other apps for better visualization.

---

The volume viewer app, for instance, which in a few seconds gives us a 3D rendering or our scan with which we can play around adjusting light, camera angle, and so on.

---

Just to quickly mention a couple more, there is a barebone image segmenter

---

And also an entire deep learning network designer in which you can import pre-trained networks, study their structure, modifying it by simple drag and drop, and train. Definitely, something you don’t usually see every day.

---

If we want to find a limit to what offered by the Matlab Apps library is certainly the fact that they are designed to cover the broader possible range of users and tasks. If we are looking for an ultra-specific software for our research field, we may not be lucky enough to find it listed in the built-in apps. But this doesn’t mean we won’t find it anywhere. If it is true the Matlab is a closed source commercial project, it is also true that the academic community using it is huge. As a result, a lot of third-party tools have been developed over the years, and some of them are almost as good as commercial software on their own. The first place to look for them is certainly the Mathworks File Exchange Platform, where everyone can upload new functions, scripts, apps, and so on, and everyone is free to download and use them.  
---

Towards the end of this talk, I thought it could be interesting to see what the neuroscience community has been doing in the last few years in this field. A little disclaimer: 1st - what follows doesn’t aim at being an exhaustive list of every tool available; 2nd – each one of this tool would require more than one day to be properly presented; 3rd – my goal here is just to show you a few examples what’s out there so that if you get curious about something, you can go ahead and see if it fits the kind of research you are doing.

---

Let’s start with EEGlab, a tool that if you work on EEG and MEG data you would probably know. It is built around an interactive GUI that allows you to process EEG and MEG data, and also to build custom scripts and processing pipeline to make it easy to replicate your experiment. It provides functions to study each channel location and field map, perform ICA decomposition, and also guided group analysis with very little knowledge about statistics required.

---

It’s also an open-source platform through which researchers can share new methods as EEGLAB plug-in functions, like SIFT, which is a tool to study causality and connectivity,

---

Or BCILAB, which is focused on the design and analysis of brain-computer interface experiments.  
---

Simply put, you can think of it as a competitor of our own MNE-toolbox, which used to have Matlab interface, but that I fear has been now replaced by the success of MNE python.

---

Another very famous Matlab-based tool is SPM. Developed at UCL, this is a suite of many Matlab programs focused on the analysis of brain imaging data sequences, with a particular focus on functional imaging like fmri, pet, spect and meg. It again offers a quite intuitive user interface that guide that guides the user through the entire processing pipeline,

---

Taking care of everything from image registration, normalization to atlases, to the estimation of a GLM based on the functional task performed which produces a set of estimated parameters that are then shown as statistical maps. Again, this description does not make justice to the tool, but I think that their training course is a multiday one, so bear with me on this very quick outline.

---

If you had the chance to work with data coming from the Siemens BrainPET scanner in Bay 6 here at the Martinos Center, you have already met Masamune, which is a Matlab tool for PET-MR image reconstruction maintained by people in Ciprian Catana’s Lab, to which I belong myself. This tool takes care of everything from attenuation estimation and correction, to motion correction. It performs PET image reconstruction in such a way that MR and PET images are registered and in the same space. It provides also tools for exporting the final image in many different formats and also to prepare everything required by kinetic modeling when a dynamic dataset is acquired.

---

Speaking of kinetic modeling, COMKAT was one of the first open-source tools to provide an alternative to commercial software like PMOD to perform pharmacokinetic modeling on PET and SPECT data. It supports a wide set of models already implemented that the user can select just using a drop-down menu. Interactive interfaces to deal with blood input function and region of interest selection are also provided so that no real command-line interaction is needed by the user.

---  
Lastly, MIAKAT is a newer software for the analysis of PET-MR dynamic data, which leverages a Matlab based GUI to provides a point and click access to a complex library of underlying functionalities. The overall pipeline provided takes care of everything, from the coregistration of functional PET and structural MR data to the import of blood input data, to everything that lies in between having a time series of PET volumes and producing a set of parametric maps.

---

That was the last one, but again, it wasn’t meant to be an exhaustive list. I just hope I captured your interest by talking about these virtuous examples of writing open source tools while using Matlab. Before closing this webinar, however, I wanted to dedicate some time to the 3 viewers (I am optimistic, I know!) that may be interested in something a bit more ‘advanced’.

---

I am going to start mentioning how we can use Matlab here, at the Martinos Center.

---

Well, the easiest way is to use any of the many Linux machines we have around. While logged in to any of them, you can just type ‘matlab’ and the GUI we saw before will fire up and you will be up and running in no time. The executable ‘matlab’ is linked to the default version of Matlab (which version this actually is I think is up the helpdesk). It is not necessary the latest. However, there are many versions of Matlab installed in our server and you can use anyone you want by simply calling the right executable.

---

Second question: can you use Matlab from your laptop? Yes and no. Matlab needs a license to run, but you can apply for a network license following this link. Network license, however, means that you will be able to use Matlab only while connected to a Partners network, either ethernet or wifi. Another option, probably the easiest one, is to use your laptop to remotely access one of the Martinos machines, and use Matlab from there. These links give you instructions on how you can do this. Ultimately, the only way to have Matlab with you all the time, also at home is to buy a standalone license.

---

the biggest problem that scares CS people away from Matlab, which is its execution speed. Matlab code, especially the code written by the average user (myself included) is almost often painfully slow. This topic will require an entire hour probably, so as for the graphics functionalities, I will just give you a few hints so that if something catches your attention, you can go deeper and see how to do it. Use functions instead of scripts, because the first time you call a function the interpreter performs a sort of compilation, and from the following call, the execution will be faster. Preallocate array in which you want to save some output data: you can do that initializing them as zero-filled arrays. Try to avoid for loops as much as possible, because Matlab is REALLY bad at handling them. For the same reason, keep the body of a for loop as short as possible, and do as many operations as possible within it. Finally, DO NOT change the working directory or modify your path during execution: every time you do that the code will be recompiled affecting performance.

---

If you really have to deal with for loop, Matlab offers you a very nice way to speed up your code by leveraging parallel execution. Doing so, in Matlab is as easy as changing a keyword from ‘for’ to ‘parfor’. The speed-up factor will clearly depend on the number of cores available on your machine, but you can trust me that you will see a difference ... or you can try It yourself!

---

Unfortunately, you cannot use parfors all the time, as this solution will usually work only