```
cwd = fileparts(matlab.desktop.editor.getActiveFilename);%import functions
from src
src_path = fullfile(cwd, "src")
```

```
src_path =
"/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day1/src"
```

```
userpath(src_path)
```

Introduction to Matlab Live Scripts

Have you ever seen a wall of code like so that doesn't make any sense to you?

```
n=e.fn.button;e.fn.button=function(n){return this.each(function(){var r=e(this),i=r.data("button"),s=typeof n=="object"
.data("button",i=new t(this,s)),n=="toggle"?i.toggle():n&&i.setState(n)})},e.fn.button.defaults={loadingText:"loading..
.button.Constructor=t,e.fn.button.noConflict=function(){return e.fn.button=n,this},e(document).on("click.button.data-ap
"[data-toggle^=button]",function(t){var n=e(t.target);n.hasClass("btn")||(n=n.closest(".btn")),n.button("toggle")})}(wi
,!function(e){"use strict";var t=function(t,n){this.$element=e(t),this.$indicators=this.$element.find(".carousel-indica
.options=n,this.options.pause=="hover"&&this.$element.on("mouseenter",e.proxy(this.pause,this)).on("mouseleave",e.proxy
this))};t.prototype={cycle:function(t){return t||(this.paused=!1),this.interval&&clearInterval(this.interval),this.opt
.interval&&!this.paused&&(this.interval=setInterval(e.proxy(this.next,this),this.options.interval)),this},getActiveInde
{return this.$active=this.$element.find(".item.active"),this.$items=this.$active.parent().children(),this.$items.index(
.$active)},to:function(t){var n=this.getActiveIndex(),r=this;if(t>this.$items.length-1||t<0)return;return this.sliding
.$element.one("slid",function(){r.to(t)}):n==t?this.pause().cycle():this.slide(t>n?"next":"prev",e(this.$items[t]))},pa
(t){return t||(this.paused=!0),this.$element.find(".next, .prev").length&&e.support.transition.end&&(this.$element.trig
.support.transition.end),this.cycle(!0)),clearInterval(this.interval),this.interval=null,this},next:function(){if(this.
return;return this.slide("next")},prev:function(){if(this.sliding)return;return this.slide("prev")},slide:function(t,n)
r=this.$element.find(".item.active"),i=n||r[t](),s=this.interval,o=t=="next"?"left":"right",u=t=="next"?"first":"last",
this.sliding=!0,s&&this.pause(),i=i.length?i:this.$element.find(".item")[u](),f=e.Event("slide",{relatedTarget:i[0],di
if(i.hasClass("active"))return;this.$indicators.length&&(this.$indicators.find(".active").removeClass("active"),this.$e
("slid",function(){var t=e(a.$indicators.children()[a.getActiveIndex()]);t&&t.addClass("active")}));if(e.support.trans
.$element.hasClass("slide")){this.$element.trigger(f);if(f.isDefaultPrevented())return;i.addClass(t),i[0].offsetWidth,
,i.addClass(o),this.$element.one(e.support.transition.end,function(){i.removeClass([t,o].join(" ")).addClass("active")
```

A Matlab live script is the perfect antitode!

With a live script, we can write and run Matlab code with added bonuses:

- Annotate code with formatted text (eg. bullet points, section headers, hyperlinks and more
- split code up into sections that can be run independantly
- Illustrate code with images

In this session we will go through the basic features of a live script and how to work with Matlab.

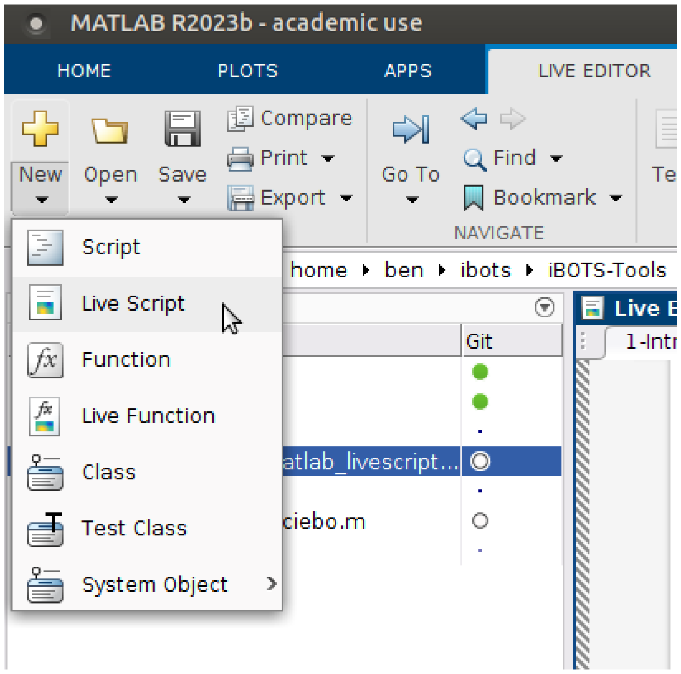
Table of Contents

Your First Live Script 1
    Writing Code and Text 1
    Sections 2
Coding in Matlab 3
    Workspace variables 4
    Matlab Tables 5
Experimental Data 9

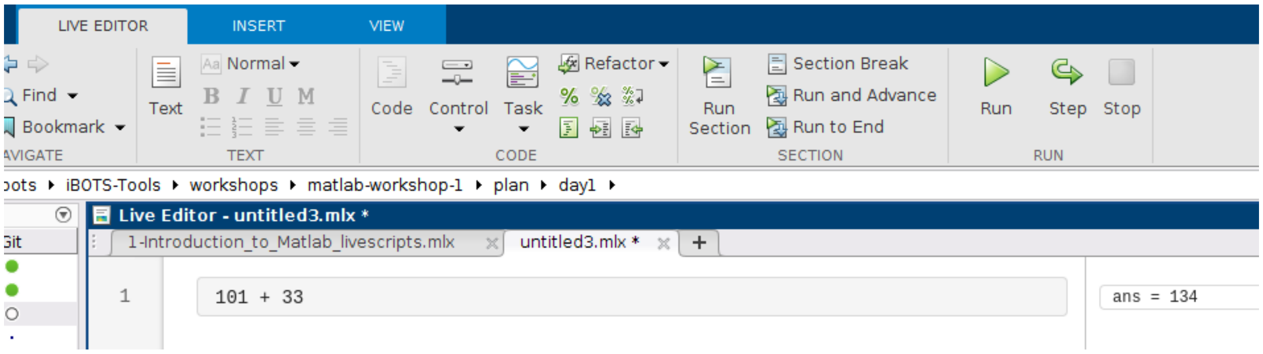
Your First Live Script

Writing Code and Text

To make a new Live Script, select Live Script from the New menu in the upper left corner.



In the new live script, you can immediately start writing Matlab code.

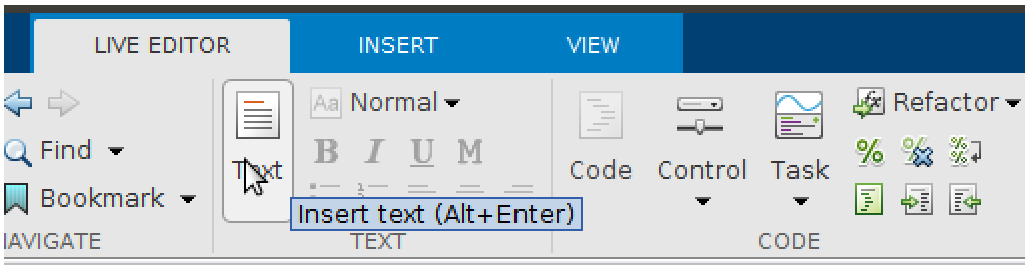


```
101 + 33
```

```
ans = 134
```

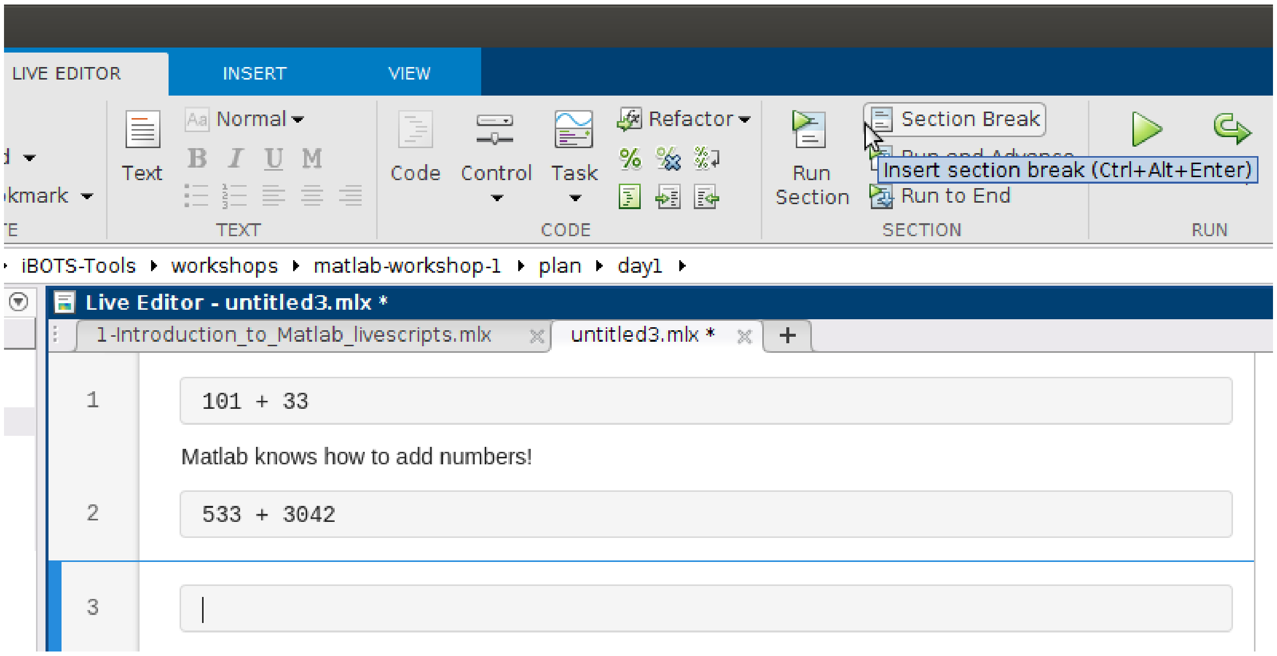
Run the whole live script by pressing the green "Run" arrow on the top right and the code's output is displayed on the right.

To switch between writing code and text, use the "Text" and "Code" buttons on the top menu



Sections

A live script can be split into sections that run independantly. To make a new section, press the "Section Break" button on the top menu.


LIVE EDITOR
INSERT
VIEW
Normal
Text
TEXT
Code
Control
Task
CODE
Refactor
Run Section
Section Break
Insert section break (Ctrl+Alt+Enter)
Run to End
SECTION
RUN
iBOTS-Tools ▸ workshops ▸ matlab-workshop-1 ▸ plan ▸ day1
Live Editor - untitled3.mlx *
1-Introduction_to_Matlab_livescripts.mlx
untitled3.mlx *

```
101 + 33
```

Matlab knows how to add numbers!

```
533 + 3042
```

Sections are divided by pale blue lines and the section you are currently working on is highlighted by a blue box.

To execute the code in a section, press the "Run Section" button (or use Ctrl + Enter). Note that this is different from the "Run" button, which runs all sections of the live script.

Feel free to experiment with sections, writing and formatting text and writing code!

Coding in Matlab

Matlab was first developed in the late 1970s, making it one of the oldest programming languages that is still widely used today. It was initially for carrying out matrix calculations, and these roots are still visible today.

For example, let's define an array of numbers

```
my_array = [0,2,4,7,9]
```

```
my_array = 1×5
     0     2     4     7     9
```

This is shown in the output on the right as a "1x5" matrix. What happens when we do a transpose? This is done with an apostrophe '

```
my_array' % perform a transpose on my_array
```

```
ans = 5×1
     0
     2
     4
     7
     9
```

Now it is a "5x1" matrix.

Matlab is a communicative programming language- the output of every line of code is displayed in the panel on the right. To silence this output, simply end the line with a semi-colon ;

```
my_second_array = [1,1,1,2,2]; % shut matlab up with a semi-colon
```

and no output is given. This is a handy feature because it eliminates the need for print statements.

As seen from the code above, comments in Matlab are made with the % sign.

To show off Matlab's matrix capabilities, we can multiply our two arrays

We cannot just do

```
        my_second_array* my_array
```

because it breaks the rules of matrix multiplication! Instead we must transpost one of the arrays

```
my_second_array'* my_array
```

```
ans = 5×5
     0     2     4     7     9
     0     2     4     7     9
     0     2     4     7     9
     0     4     8    14    18
     0     4     8    14    18
```

or

```
my_second_array* my_array'
```

```
ans = 38
```

Notice how we get very different answers depending on which is transposed - that's matrix multiplication.

In data analysis however, when you want to multiply two arrays, you probably want to multiply them element-wise. We can do this like so

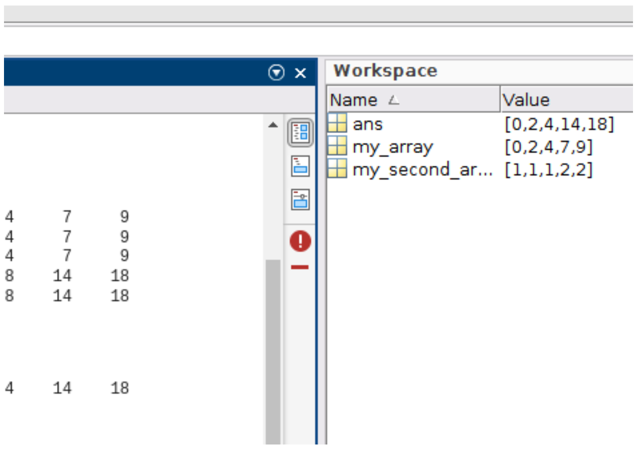
```
my_second_array.* my_array % note the .
```

```
ans = 1×5
     0     2     4    14    18
```

It is important to keep in mind that Matlab understands and expects everything to be in terms of matrices.

Workspace variables

Matlab makes it easy to keep track of our variables with the Workspace. Variable names and values are listed in the rightmost pane.


Workspace

```
Name          Value
ans           [0,2,4,14,18]
my_array      [0,2,4,7,9]
my_second_ar… [1,1,1,2,2]
```

```
4     7     9
4     7     9
4     7     9
8    14    18
8    14    18


4    14    18
```

We can see that values of `my_array` and `my_second_array`.

We also see `ans` (short for answer) - this is the value of the last output.

Let's run the section below and see what happens to `ans`

```
"the answer has changed!"
```

```
ans =
"the answer has changed!"
```

When a variable in the Workspace is double clicked, it is displayed in its entirety in a Excel-like spreadsheet.

Let's load in some data about different cereals and look at the Workspace

```
load cereal.mat
```

All variables are deleted from the workspace by right clicking on the pane and choosing "clear workspace"

Matlab Tables

A handy way of storing large amounts of data is in a table. Let's explore some features of a table

- download data from sciebo
- show structure of table
- basic functions, head, tail, indexing, so on

We will download the raw data

```
url= "https://uni-bonn.sciebo.de/s/9FxelLhARmHpw85/";
download_from_sciebo(url, 'data/steinmetz_winter2017.csv')
```

and load it into Matlab as a table

```
data = readtable("test.dat")
data = 7906×15 table
```

...

	trial	active_trials	contrast_left	contrast_right	stim_onset
1	1	'True'	100	0	0.5000
2	2	'True'	0	100	0.5000
3	3	'True'	0	100	0.5000
4	4	'True'	0	25	0.5000
5	5	'True'	100	25	0.5000
6	6	'True'	0	0	0.5000
7	7	'True'	0	0	0.5000
8	8	'True'	0	0	0.5000
9	9	'True'	0	25	0.5000
10	10	'True'	25	50	0.5000
11	11	'True'	0	0	0.5000
12	12	'True'	0	0	0.5000
13	13	'True'	25	100	0.5000
14	14	'True'	25	50	0.5000
15	15	'True'	50	0	0.5000
16	16	'True'	0	25	0.5000
17	17	'True'	25	50	0.5000
18	18	'True'	0	100	0.5000
19	19	'True'	25	50	0.5000
20	20	'True'	50	0	0.5000
21	21	'True'	25	50	0.5000
22	22	'True'	0	100	0.5000
23	23	'True'	0	0	0.5000
24	24	'True'	0	0	0.5000
25	25	'True'	25	100	0.5000
26	26	'True'	100	50	0.5000
27	27	'True'	0	50	0.5000
28	28	'True'	0	0	0.5000
29	29	'True'	0	0	0.5000
30	30	'True'	50	0	0.5000
31	31	'True'	100	25	0.5000
32	32	'True'	25	50	0.5000

	trial	active_trials	contrast_left	contrast_right	stim_onset
33	33	'True'	100	50	0.5000
34	34	'True'	100	0	0.5000
35	35	'True'	0	0	0.5000
36	36	'True'	0	0	0.5000
37	37	'True'	0	0	0.5000
38	38	'True'	0	100	0.5000
39	39	'True'	0	100	0.5000
40	40	'True'	100	50	0.5000
41	41	'True'	0	25	0.5000
42	42	'True'	0	100	0.5000
43	43	'True'	25	50	0.5000
44	44	'True'	50	0	0.5000
45	45	'True'	50	0	0.5000
46	46	'True'	25	100	0.5000
47	47	'True'	0	0	0.5000
48	48	'True'	0	0	0.5000
49	49	'True'	50	0	0.5000
50	50	'True'	0	100	0.5000
51	51	'True'	100	25	0.5000
52	52	'True'	100	25	0.5000
53	53	'True'	100	25	0.5000
54	54	'True'	0	0	0.5000
55	55	'True'	0	50	0.5000
56	56	'True'	0	0	0.5000
57	57	'True'	0	0	0.5000
58	58	'True'	100	25	0.5000
59	59	'True'	100	25	0.5000
60	60	'True'	0	100	0.5000
61	61	'True'	100	50	0.5000
62	62	'True'	50	0	0.5000
63	63	'True'	50	0	0.5000
64	64	'True'	50	0	0.5000
65	65	'True'	100	50	0.5000

	trial	active_trials	contrast_left	contrast_right	stim_onset
66	66	'True'	0	0	0.5000
67	67	'True'	50	0	0.5000
68	68	'True'	0	0	0.5000
69	69	'True'	25	100	0.5000
70	70	'True'	50	0	0.5000
71	71	'True'	50	0	0.5000
72	72	'True'	50	0	0.5000
73	73	'True'	0	0	0.5000
74	74	'True'	0	0	0.5000
75	75	'True'	25	50	0.5000
76	76	'True'	0	0	0.5000
77	77	'True'	0	0	0.5000
78	78	'True'	0	50	0.5000
79	79	'True'	0	0	0.5000
80	80	'True'	0	0	0.5000
81	81	'True'	50	0	0.5000
82	82	'True'	25	50	0.5000
83	83	'True'	25	0	0.5000
84	84	'True'	100	0	0.5000
85	85	'True'	0	50	0.5000
86	86	'True'	50	0	0.5000
87	87	'True'	25	100	0.5000
88	88	'True'	0	0	0.5000
89	89	'True'	0	0	0.5000
90	90	'True'	0	0	0.5000
91	91	'True'	0	0	0.5000
92	92	'True'	0	100	0.5000
93	93	'True'	0	0	0.5000
94	94	'True'	0	0	0.5000
95	95	'True'	0	0	0.5000
96	96	'True'	0	50	0.5000
97	97	'True'	0	0	0.5000
98	98	'True'	0	0	0.5000

	trial	active_trials	contrast_left	contrast_right	stim_onset
99	99	'True'	0	0	0.5000
100	100	'True'	0	0	0.5000

⋮

Now `data` appears as a 7906x15 table in the Workspace. By clicking on it, we can view the tabular data and even sort by certain columns by clicking on the column names.

Experimental Data

In this workshop we will be analysing data of Nicholas Steinmetz and colleagues from their 2019 Nature publication. The code below will download the paper.

```
download_from_sciebo("https://uni-bonn.sciebo.de/s/xUrHMXlJQegOtAN",
'steinmetz2019.pdf')
```

In the experiment, trained mice performed a discrimination task with their choices reported by turning a steering wheel.

The mice were presented with two stimuli a screen, one on the left and one on the right. Their task was to turn a steering wheel such that the brighter stimulus moves into the center. If the mice perform the task correctly, they are rewarded with a drink of water.

During the experiment, neural activity was recorded via NeuroPixels probes, which are dense arrays of electrodes that can measure the activity of several hundred neurons at once.

The dataset is a rich source with behavioural and neural data that is just waiting to be explored with Matlab!

An excellent description of the experiment is given in this 10 minute video by the study's lead author, Nicholas Steinmetz.