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

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 - click on the grey cell and press Ctrl + Enter.

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

```
Downloading file to steinmetz2019.pdf
Error using websave
Web service operation terminated by user while accessing URL 'https://uni-bonn.sciebo.de/s/
xUrHMXlJQegOtAN/download'.

Error in download_from_sciebo (line 43)
websave(full_download_location, download_url );
```

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 on 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.

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.opti
.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.$indicators.children()[a.getActiveIndex()]);t&&t.addClass("active")}));if(e.support.transi
.$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, images and more)
- split code up into sections that can be run independantly

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

Table of Contents

Coding Made Easy with Tasks

First, let's download a data file from sciebo for the exercises in this live script. The data is hosted on sciebo at the url: https://uni-bonn.sciebo.de/s/9FxelLhARmHpw85

Fill in the blank ___ below to download the datafile

```
url= "https://uni-bonn.sciebo.de/s/9FxelLhARmHpw85"
```

```
url = 
"https://uni-bonn.sciebo.de/s/9FxelLhARmHpw85"
```

```
download_from_sciebo(url, 'data/steinmetz_winter2017.csv')
```

```
Downloading file to data/steinmetz_winter2017.csv
Done!
```

```
addpath("data")
```

Matlab makes complex tasks like loading data and plotting the work of a few clicks with tasks

Use the Task below to import the file **`steinmetz_winter2017.csv`** from the directory **`data`**

```
% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 15);

% Specify range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["trial", "active_trials", "contrast_left",
"contrast_right", "stim_onset", "gocue_time", "response_type",
"response_time", "feedback_time", "feedback_type", "reaction_time",
"reaction_type", "mouse", "session_date", "session_id"];
opts.VariableTypes = ["double", "categorical", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double",
"categorical", "datetime", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["active_trials", "mouse"], "EmptyFieldRule",
"auto");
opts = setvaropts(opts, "session_date", "InputFormat", "yyyy-MM-dd");
opts = setvaropts(opts, "session_id", "TrimNonNumeric", true);
opts = setvaropts(opts, "session_id", "ThousandsSeparator", ",");

% Import the data
steinmetz_winter2017 = readtable("/home/ben/ibots/iBOTS-Tools/workshops/
matlab-workshop-1/plan/day1/data/steinmetz_winter2017.csv", opts);

% Clear temporary variables
clear opts

% Display results
steinmetz_winter2017
```

```
steinmetz_winter2017 = 7906×15 table
```

...

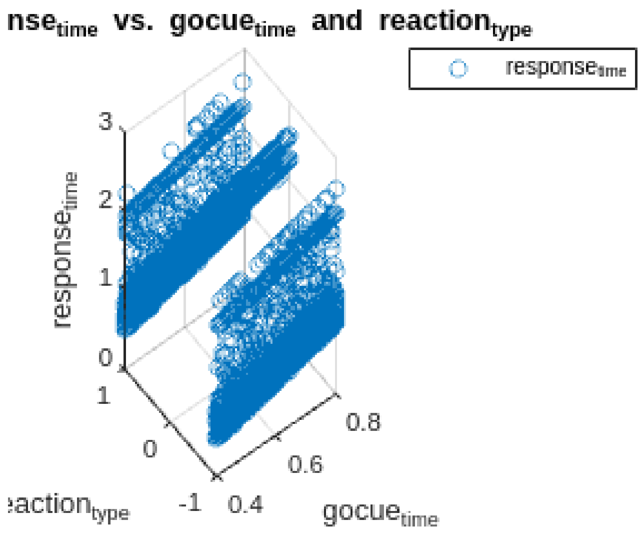
	trial	active_trials	contrast_left	contrast_right	stim_onset
1	1	True	100	0	0.5000
2	2	True	0	100	0.5000
3	3	True	0	100	0.5000
4	4	True	0	25	0.5000

	trial	active_trials	contrast_left	contrast_right	stim_onset
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

⋮

The Create Plot task allows us to make any plot we want - try it out with the data imported from the previous cell

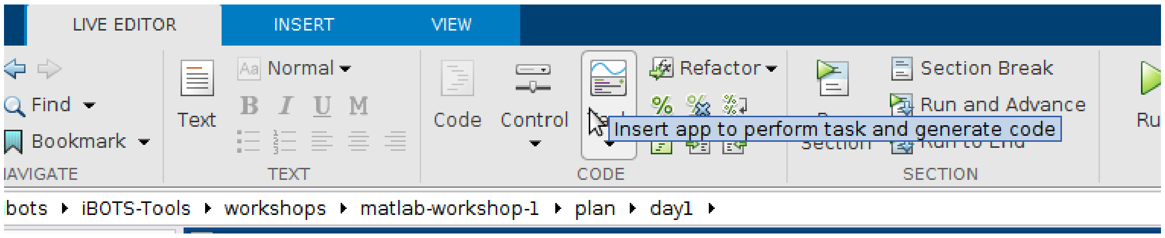
```
% Create scatter3 of selected data
s =
scatter3(steinmetz_winter2017,"gocue_time","reaction_type","response_time","D
isplayName","response_time");

% Add xlabel, ylabel, zlabel, title, and legend
xlabel("gocue_time")
ylabel("reaction_type")
zlabel("response_time")
title("response_time vs. gocue_time and reaction_type")
legend
```



Tasks produce Matlab code in the "Show code" section at the bottom of the task. This code can be further edited to suit your purpose.

Many other tasks are available. You can even create your own custom-made tasks with the "Create a Task" task!

The Task menu is in the **`CODE`** section in the **`LIVE EDITOR`** tab in the top menu.



Explore the "Compute by Group" task to see what is does on the data we just loaded in.

```
% Compute group summary
newTable = groupsummary(steinmetz_winter2017,"mouse",["mean","min"], ...
    vartype("numeric"))
```

```
newTable = 5×26 table
```

...

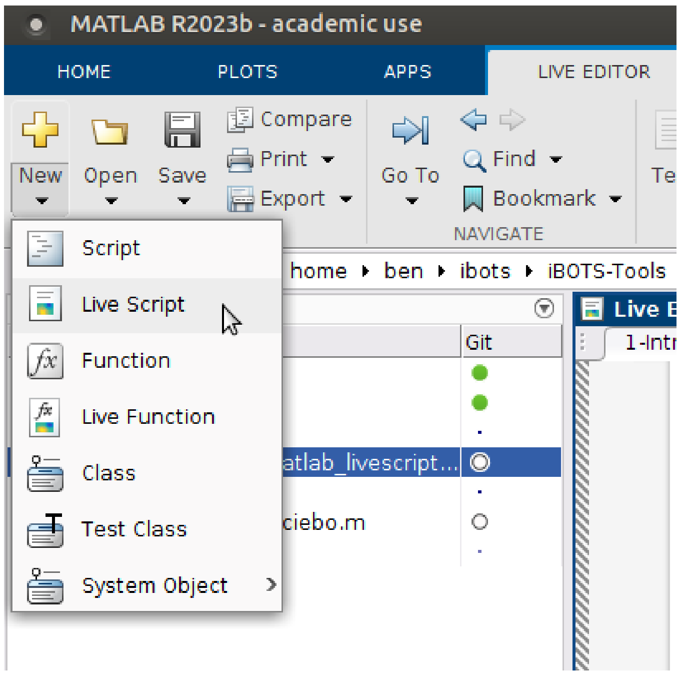
	mouse	GroupCount	mean_trial	min_trial	mean_contrast_left
1	Forssmann	1485	186.5003	1	34.0572
2	Lederberg	2902	211.1844	1	34.1489
3	Richards	1677	172.4603	1	37.3733
4	Tatum	1389	177.9035	1	35.8711
5	Theiler	453	227	1	26.8212

Your First Live Script

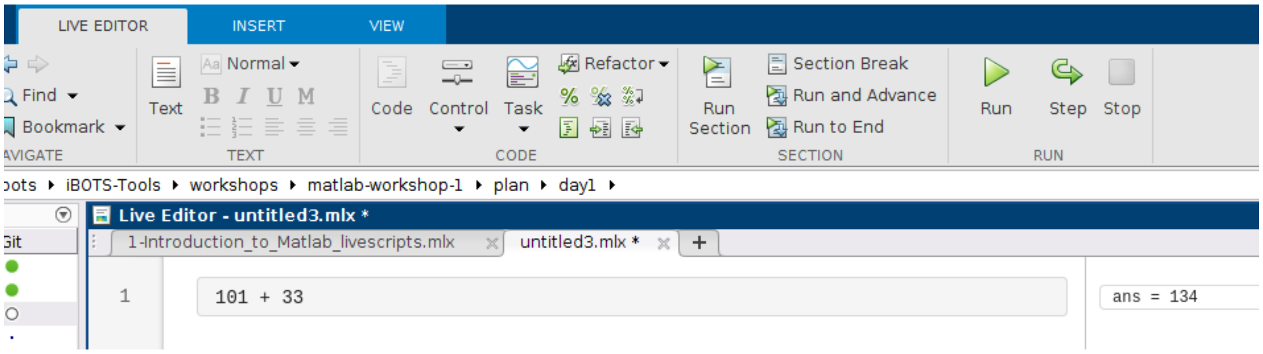
Follow along these instructions to get started with your first live script

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.

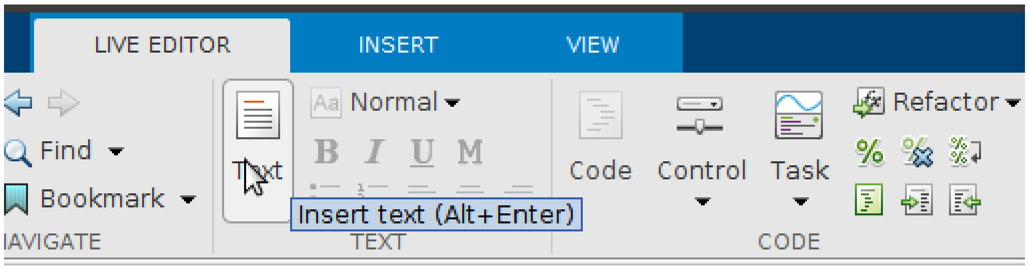


```
101 + 33
```

```
ans = 134
```

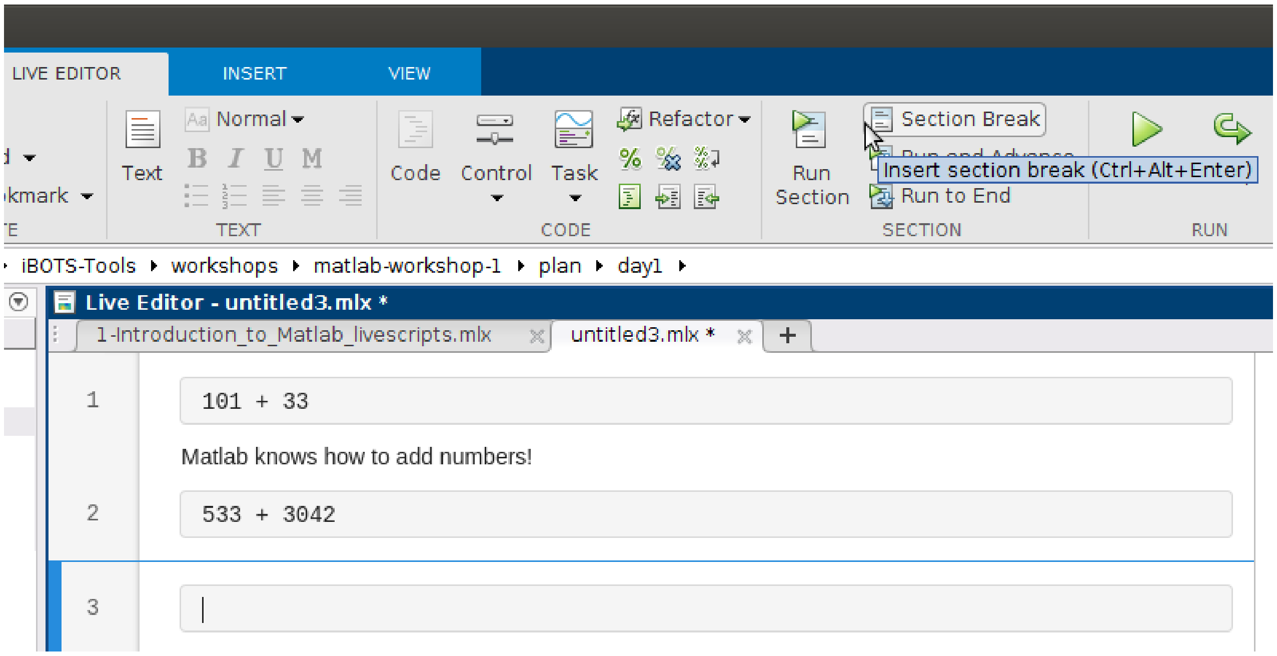
Run the whole live script by pressing the green "Run" arrow on the top right and the code's output is displayed on the right.

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
Refactor
Code
Control
Task
CODE
Section Break
Run and Advance
Insert section break (Ctrl+Alt+Enter)
Run to End
Run Section
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 a list 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 `'`

Use the cell below to transpose **`my_array`**

```
my_array'
```

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

What does the transpose do?

Matlab is a communicative programming language- the output of every line of code is displayed in the panel on the right.

Define a new variable **`my_second_array`** that contains 5 elements which are numbers.

End your line of code with a semi-colon. Do you notice what happend with/without a semi-colon? **Hint** - look at the panel on the right

```
my_second_array = [5,5,5,5,5];
```

Multiplication

Let's multiply **`my_array`** and **`my_second_array`** together and see what happens.

Do that in the cell below

```
 my_array*my_second_array
```

```
Error using  * 
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first
matrix matches the number of rows in the second matrix. To operate on each element of the matrix
individually, use TIMES (.*) for elementwise multiplication.

Related documentation
```

Oh no! It doesn't work because Matlab explicitly assumes that we want to do matrix multiplication. Matrix multipication has its own set of rules, and we have to follow them.

Instead, do the multiplication with one of the matrices transposed.

```
  my_array' *my_second_array
```

```
ans = 5×5
     0     0     0     0     0
    10    10    10    10    10
    20    20    20    20    20
    35    35    35    35    35
    45    45    45    45    45
```

Now do the multiplication with the other matrix transposed

```
my_array * my_second_array'
```

```
ans = 110
```

What do you notice? Matrix multiplication is quite strange indeed...

In data analysis however, when you want to multiply two arrays, you probably want to multiply them element-wise. We can do this with the `.*` operator.

In the cell below, multiply **`my_array`** and **`my_second_array`** together element-wise

```
my_array.*my_second_array
```

```
ans = 1×5
     0    10    20    35    45
```

Division

In many programming languages, division is done with the `/` operator

Use the `/` operator on **`my_array`** and **`my_second_array`**

```
my_array / my_second_array
```

```
ans = 0.8800
```

Did you expect that?

In Matlab, the `/` operator does not perform a division. It infact solves a system of linear matrix equations.

To find out more, highlight `/` in the cell above, right click and choose "Help on "/" " .

In the cell below, use the `./` operator on **`my_array`** and **`my_second_array`** to find out what it does

```
my_array./my_second_array
```

```
ans = 1×5
         0    0.4000    0.8000    1.4000    1.8000
```

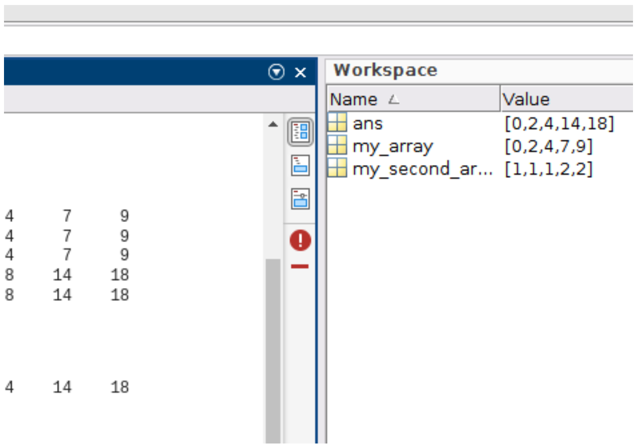
The lesson here is that Matlab understands and expects everything to be in terms of matrices - this can either be a blessing or a curse!

Comments in Matlab are made with the % sign. Try it out in the cell below

```
% Bonjour tout le monde!
```

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.



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

4    14    18
```

Workspace

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

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.

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

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

Load the data downloaded earlier into Matlab as a table using the **`readtable`** function. Name the result **`data`**

```
data = readtable("data/steinmetz_winter2017.csv")
```

```
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

⋮

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.

Answer the following questions using the Workspace

- How many rows does **`data`** contain?

ANSWER - 7906

- What is the largest value in the contrast_left column?

ANSWER - 100

- What is the smallest value in the response_time column?

ANSWER - 0.479414497674441