MedPCPy’s FAQ and tips

Some common oversights that might be causing MedPCPy to not work properly or at its best.

1. What type of analyses can this library do?

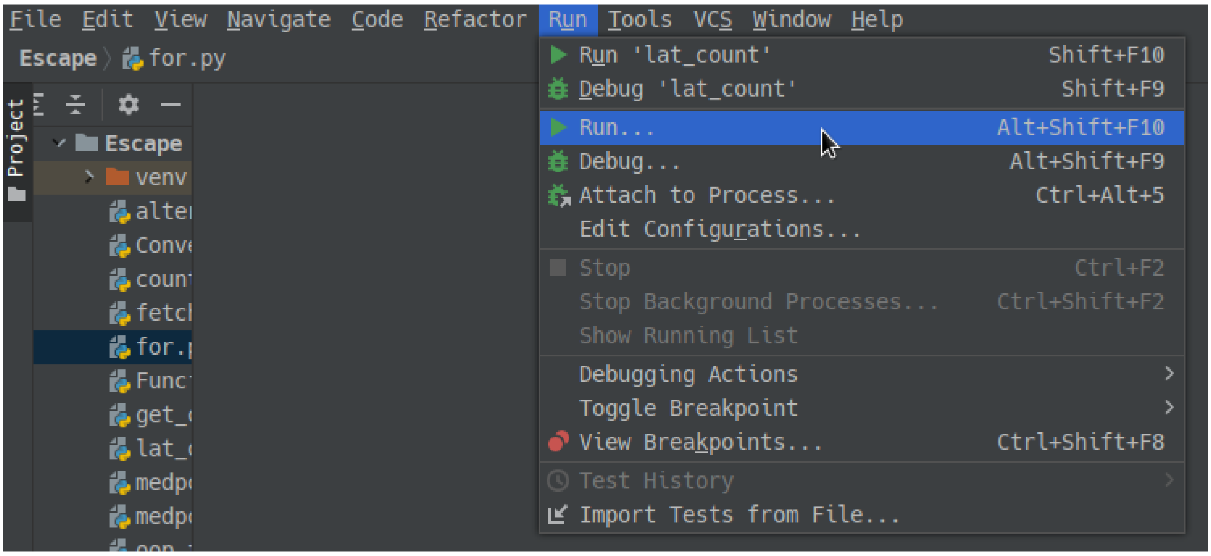
   The library can extract response counts, latencies, response distributions, and entire arrays, and write them and their measures of central tendency (when applicable) on user-specified files.

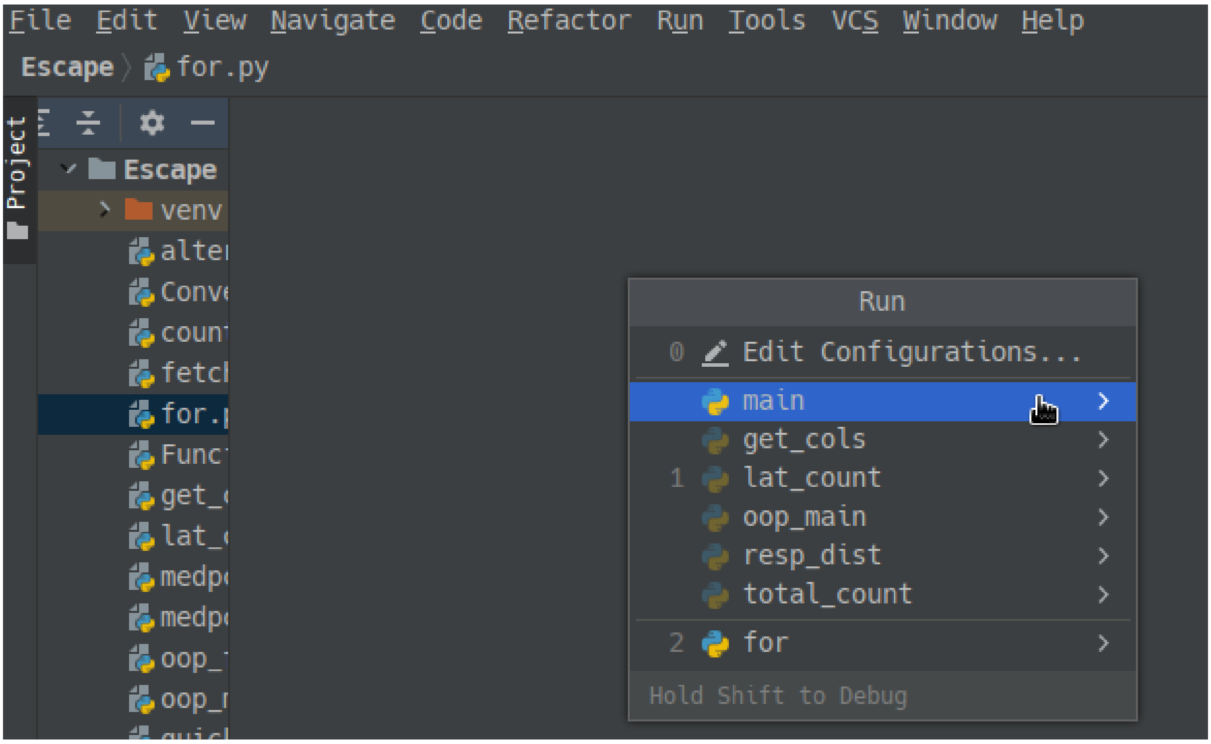
2. My scripts were not working and I discovered that two sessions were recorded in the same data file.

   This usually happens when two different data outputs are contained in the same file, probably due to running two different sessions without specifying a new data file name on Med-PC-IV. If this is the case, it’s necessary to separate both sessions, making sure the spacing of the file stays consistent. The script should run smoothly afterwards.

3. I hit the “Run” button, but it seems like something is not working.

   On the PyCharm IDE, when one is working with more than one .py file, it is important to make sure the correct script is selected for running. This can be verified by clicking the “Run” menu, then, the name of the desired script should appear in the first option. If a different script is selected, then clicking the third option, ‘Run…’, and selecting the desired script will solve the problem.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Escape
for.py
Project
Escape
venv
alte
Conv
coun
fetc
for.
Func
get_
lat_
medp
medp
Run 'lat_count' Shift+F10
Debug 'lat_count' Shift+F9
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Process... Ctrl+Alt+5
Edit Configurations...
Stop Ctrl+F2
Stop Background Processes... Ctrl+Shift+F2
Show Running List
Debugging Actions
Toggle Breakpoint
View Breakpoints... Ctrl+Shift+F8
Test History
Import Tests from File...



4. The script seems to be working, but I keep getting zeros instead of the actual expected numbers.

   Make sure the timeColumn and markColumn arguments (the letters identifying the columns in which, in the individual converted files, the time and marks are written) in Analyzer are correctly declared, and that all your marks are correct as well.

5. I am using the Resp_dist function, and some data is displayed, but it does not make complete sense.

   Make sure to adjust the “unit” and/or “bin_size” arguments to your preferences. The default in the sample script is “unit” = 1, which indicates that time is measured in seconds in the timeColumn declared in the analyzer object, and “bin_size” = 1, which means each trial is divided in 1-second segments (bins).

6. My script(s) were working perfectly, but I restarted PyCharm and now they don’t.

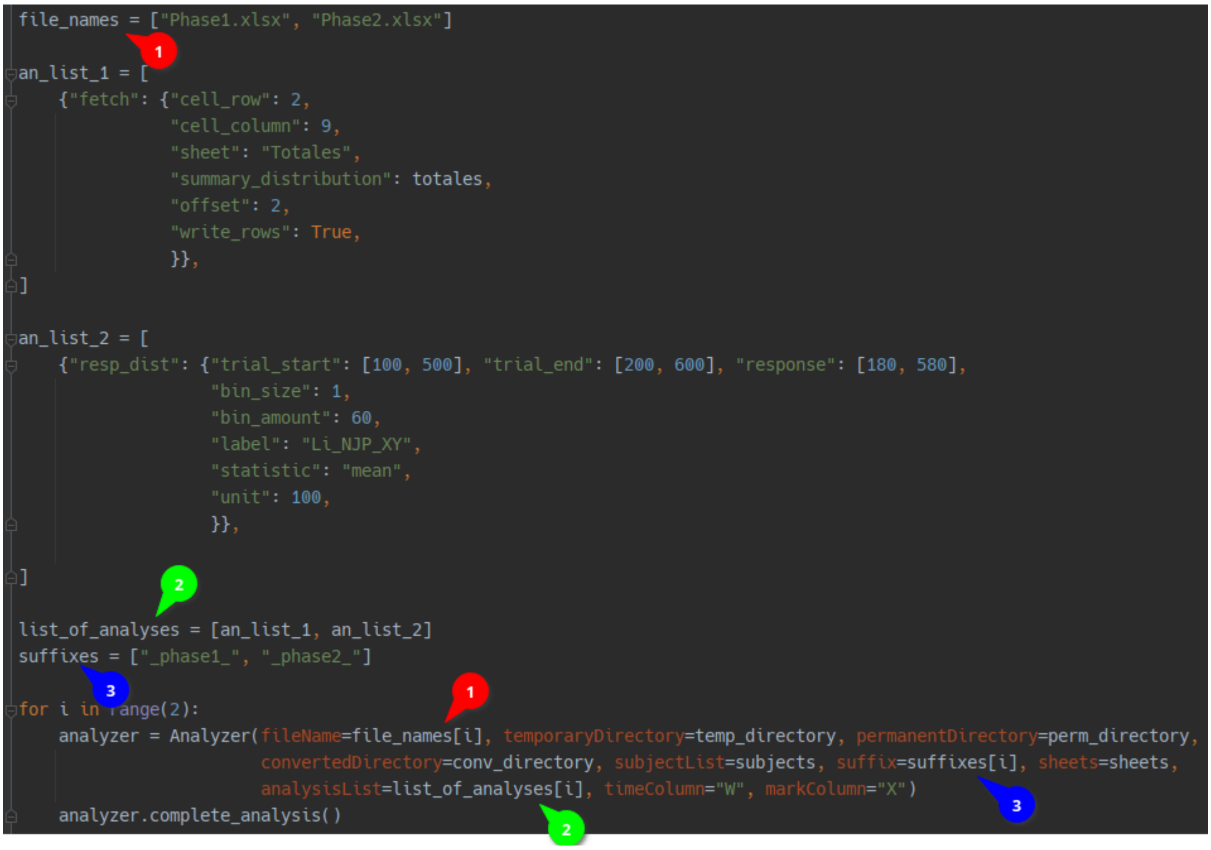
   When working on the PyCharm IDE, it is important to make sure your scripts are saved in the project in which MedPCPy is installed, otherwise they will not work since the script cannot access libraries installed outside the current project. If you have two projects open, one where MedPCPy is installed, and one where it is not, and you run a script saved in the second project, it will run properly, but only while the first project is open.

7. Is it possible to analyze data from subjects that are currently in different sessions?

Yes, it is possible to analyze subjects in different sessions (e.g. session 3 of subject 1, and session 6 of subject 2), and also to analyze a different number of sessions for each subject (e.g. 2 sessions of subject 4, and 8 sessions of subject 2). Note that it is important to respect the number of sessions, as they will determine in which cell the data is saved.

8. My experiment has many phases in which subjects stay for different amounts of sessions, so on the same day I have to analyze data from different procedures. Is there a way to use the same script to analyze all my different files at the same time?

   An easy fix would be to create as many analysisLists as scripts you would require. Then you could create a variable with a list of the different summary file names (e.g., file_names = [‘Phase1.xlsx’, ‘Phase2.xlsx’]), another with a list of the suffixes you need (e.g., suffixes = [“_phase1_”, “_phase2_”]), and another one with a list of all the analysisLists you need (e.g., list_of_analyses = [an_list_1, an_list_2]). Then, you can use a *for* loop to iteratively feed them into the Analyzer object. Here is an example of the code:



```
file_names = ["Phase1.xlsx", "Phase2.xlsx"]

an_list_1 = [
    {"fetch": {"cell_row": 2,
               "cell_column": 9,
               "sheet": "Totales",
               "summary_distribution": totales,
               "offset": 2,
               "write_rows": True,
               }},
]

an_list_2 = [
    {"resp_dist": {"trial_start": [100, 500], "trial_end": [200, 600], "response": [180, 580],
                   "bin_size": 1,
                   "bin_amount": 60,
                   "label": "Li_NJP_XY",
                   "statistic": "mean",
                   "unit": 100,
                   }},

]

list_of_analyses = [an_list_1, an_list_2]
suffixes = ["_phase1_", "_phase2_"]

for i in range(2):
    analyzer = Analyzer(fileName=file_names[i], temporaryDirectory=temp_directory, permanentDirectory=perm_directory,
                        convertedDirectory=conv_directory, subjectList=subjects, suffix=suffixes[i], sheets=sheets,
                        analysisList=list_of_analyses[i], timeColumn="W", markColumn="X")
    analyzer.complete_analysis()
```

   Note that Python lists are ordered, and thus the first item of the file_names list must correspond with the first item of all other lists, and so on.