Same Different Task behavioral data Script

The automated data script for the Same Different Task was made using Python 3.9 and will likely not work on older version. I would suggest opening the script in an editor such as PyCharm with a python 3.9 environment to run it (at the time of writing this, psychopy is still using Python 3.6 which will not work).

The script has two parts:
Data_Extraction_SameDiff.py – This is the main script that loops through the files
Behavioral_SameDiff_Script.py – This script extracts the data from a single file

You only need to run the *Data_Extraction_SameDiff.py* file, which will call a function written in the *Behavioral_SameDiff_Script.py* file for each file in the folder. Once it is done it will then save all the data into an excel sheet called "SameDiff_Behavioral_Data.xlsx" in a "Results" folder. If these last two do not exist it will create a results folder (in the same directory as the script) and the excel file.

The folder should look something like below, with the scrips and the data in one folder. The data in the data-folder need to be text files, but the naming does not matter that much. Currently it is set up to look for a folder named *"data"*. This can be changed on lines 15 and 16 of the *Data_Extraction_SameDiff.py* file.

Within this raw data folder, it looks for subfolders that correspomd to the different kinds of stimuli that can be used in this task (see the task description document). These are created by the task as well. To add folders to the script you simply add the folder name between the brackets on line 15. Since the names can be quite long the script will number the sheets in the excel file according to the order the folders have been processed (e.g. univariate1 is the data from the first folder)

…Folder/
- Data_Extraction_SameDiff.py
- Behavioral_SameDiff_Script.py
- data/
  - 1) Practice Trials
    - ppn001.dat
    - ppn002.dat
  - 4) 3-Val_Uncrossed/
    - ppn001.dat
    - ppn002.dat
  - Results/
    - SameDiff_Behavioral_Data.xlsx

Data_Extraction_SameDiff.py
For the script to work it needs to be in the same folder as the "data" folder, not in the data folder but next to the data folder, because it will try to find a "data" folder in the same directory as itself. This can be specified on lines 15 and 16 though.

It will loop through all the folders specified in the settings structure (written as `['folder1, 'folder2']` on line 15). Within each folder, it will loop through every .dat file. If the IDE has a debug-window or something similar, a message will appear

```
Processing Folder: Folder1
```

It will proceed by loading in all the .dat files in the data folder and loop through each file, processing it using *Behavioral_SameDiff_Script.py*. A message will appear showing which file is currently being processed

```
Processing: PP01.txt
```

If something goes wrong, the message will indicate which file caused an error

```
Error Occurred at file: PP01.txt
```

Upon completion, all the data is saved to an excel file in the Results folder. This can be loaded into an SPSS file or processed further in excel.

Sheet	Description
Bins	A key for the numerical codes used in the univariate file
Univariate	▪ A univariate data structure. Each participant has 8 rows, one for each condition. ▪ PP: the participant number ▪ Age: the age of the participant ▪ Handedness: Left or right-handed ▪ Bin: The condition comprised of range, canonicity, and congruency ▪ Range: Numerical Range based on the symbol shown ▪ Canonicity: the canonicity of the hands shown in the experiment ▪ Congruency: Congruency between the quantity of the had and the symbol ▪ Acc: Average accuracy ▪ RT_Overall: Average Response Time ▪ RT_Correct: Average Response Time Correct only
Multivariate_Canonical	▪ A multivariate version of the above data

Behavioral_SameDiff_Script.py

This script extract the data for individual files, the main script will call the "behavioral_samediff_runner" function that outputs the data from a single text file.

The first thing it does is check which line contains "Trial_Number" which, at the time of writing, denotes where the participant data starts.

```
Number-Hand Stroop Task
-----------------------------------
Participant  ;001
Version      ;4.5
File         ;4) 3-Val_Uncrossed.csv
Start Time   ;15-05-2019 12:17:42
Trials       ;96
Blocks       ;1
Blocklength  ;96
Age          ;26
Sex          ;Female
Handedness   ;Right
-----------------------------------

Block;Trial_Number;Trial Code;Symbol_Num;Hand_Num;Dominant Hand;Non Dominant Hand;
Distance;Stim_Hand_Dom;Canonical;buttonPressed;Correct;fixationTime;StimOnset;SOA;
responseTime;Buttonbox RT
1;1;38;8;8;3;5;0;0;1;2;1;0.01612;76623.96657;1.16629;0.71709;727.00000
1;2;55;4;2;2;0;2;1;0;3;1;0.00706;2.38340;0.95715;0.80047;819.00000
1;3;91;2;2;0;2;0;0;0;2;1;0.00354;2.53363;1.00372;0.55030;555.00000
1;4;39;8;9;4;5;-1;0;1;2;-1;0.00443;2.10000;0.83779;0.76712;770.00000
1;5;9;4;4;4;0;0;1;1;999;0;0.01596;2.46691;0.99948;-1.69980;770.00000
```

← find this line

Participant Information is the first block, everything above the data. It reads this in, transposes the rows into columns, and saves it for the output.

Task data is the second block. The script reads all the lines and separates the data entries by semicolons. Various new columns are created based on the data

0) Block;	10) buttonPressed
1) Trial_Number;	11) Correct
2) Trial Code	12) fixationTime
3) Symbol_Num	13) StimOnset
4) Hand_Num	14) SOA
5) Dominant Hand	15) responseTime
6) Non Dominant Hand	16) Buttonbox RT
7) Distance	**17) Can/Con Bins**
8) Stim_Hand_Dom	**18) Low/High**
9) Canonical	**19) Low/High Can/Con bins**

Can/Con Bins is created based on the canonical and Distance columns:

1 = Incongruent Non-Canonical
2 = Incongruent Canonical
3 = Congruent Non-Canonical
4 = Congruent Canonical

Low/High is created based on the Symbol_Num column

0 = Low
1 = High

Low/High Can/Con bins is based on the previous two

1 = Low Incongruent Non-Canonical
2 = Low Incongruent Canonical
3 = Low Congruent Non-Canonical
4 = Low Congruent Canonical
6 = High Incongruent Non-Canonical
7 = High Incongruent Canonical
8 = High Congruent Non-Canonical
9 = High Congruent Canonical

Based on the structure shown above, it is possible to calculate the accuracy and the response times separately for each of the 8 conditions created by range, canonicity, and congruency

- Accuracy is the average of the “correct” column
- Response time is the average of the RT column
- Response time for correct only is the average of the RT column is Correct = 1

The script then creates a multivariate and univariate version of the results and returns this to the main script.