POST-PROCESSING SCRIPTS

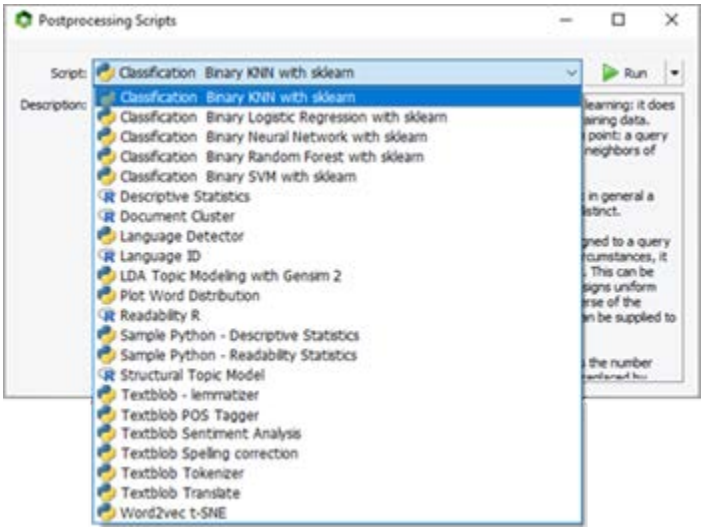
The post-processing scripting feature allows one to extent the capabilities of WordStat by the writing of additional computation or data transformation scripts. Those scripts may be written in Python or R using custom codes or existing libraries for NLP, machine learning, statistical processing, visualization, and more. Such a feature allows data scientists to focus on the crucial processing step of interest, leaving to WordStat the various tasks associated with the data preparation and preprocessing such as document importation, stemming, lemmatization, spelling correction, removal of stop words, and so on. It also allows one to combine WordStat powerful categorization features (with word patterns, phrases, disambiguation rules, etc.) with new analytics techniques not available in WordStat.

The additional possibility to design dialog boxes allows data analysts to create simple graphic users' interfaces that may be used for customizing script execution and facilitate their use by other WordStat users with no programming skills.

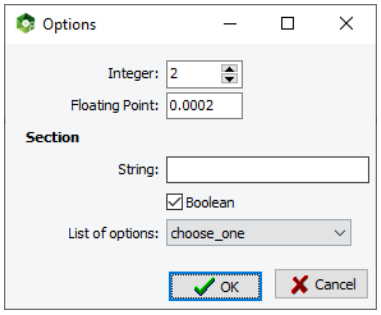
Running an existing Python or R post-processing script:

On the Frequencies page, applying an existing post-processing script can be as simple as these steps:

- Click the  button to open the main post-processing script window.
- The **Script** dropdown menu lists all existing Python and R scripts. Select the desired script. Beneath the **Script** dropdown menu is the **Description** section. Any text describing the script will be displayed here. Resizing the post-processing script window may be required to display all of the description text.



- Click the  button. For some scripts, no other steps are required, and the Report Manager will display the output of your script. Running other scripts will result in the appearance of a dialog box like the one below, allowing you to set some analysis options.

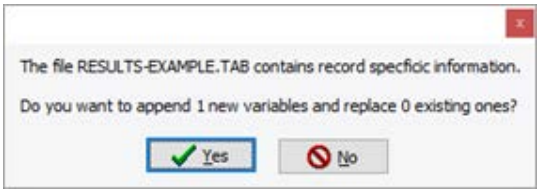


- While the script is running, a Python console or R console will appear, displaying what is logged or printed as output (stdout) or any error messages.

Output

- If a script is executed successfully, WordStat will import any output stored in any supported file format and display them in the Report Manager, which will open automatically. WordStat will import text output stored in any file with a .TXT file extension, assuming an UTF-8 text encoding, or documents stored in .DOCX, .RTF or HTML files. It will also import table output files with either .CSV, .TAB, .TSV or .XLS file extension, and any graphic produced with a .JPG, .JPEG, .PNG, .GIF, or BMP file extension. Once imported, those files are deleted.

- If a table created by the script has a column named **RECNO** containing record numbers, a dialog box will appear, giving the possibility to append data contained in this table to the current project data.



  To append data in this table, click **Yes**. If the table's column names correspond to existing variables, data into those variables will be overwritten, while new column names will result in new variables being appended. Clicking **No** will append the table to the report manager.

Writing a new Python or R post-processing script

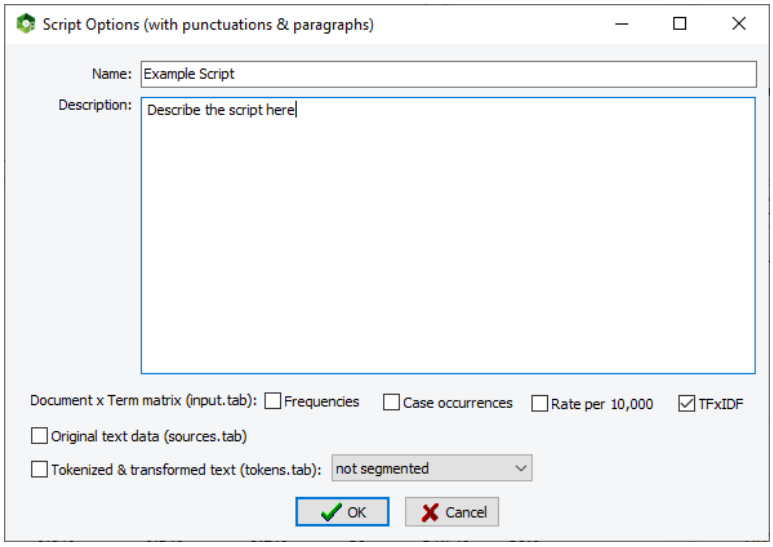
- To create a new post-processing script, open the main post-processing script window by clicking the  button on the **Frequencies** page.

- Then click the  button next to the **Run** button and select **New script** from the dropdown menu.

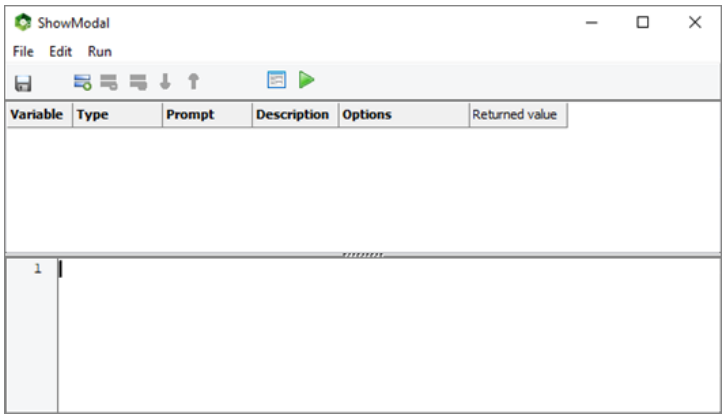
- Select the desired programming language for this script (Python or R)

- A **Script Options** window will appear. Type the **Name** of the new script.

- Optionally, add a **Description**. This description will appear when the script is selected from the script selection dropdown box.

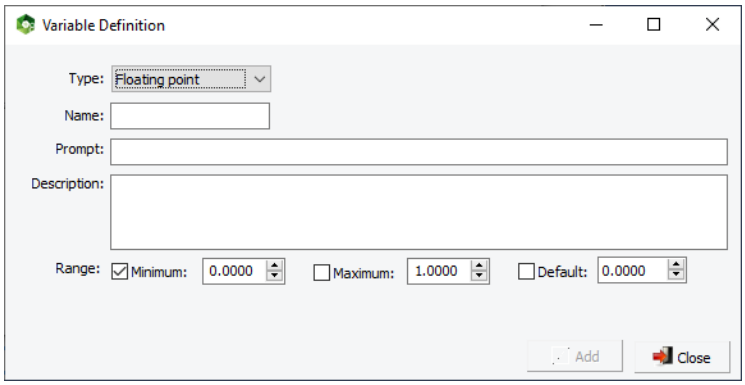


- The options below allow you to select the types of input files that will be generated and processed by the script. Up to three separate data files can be generated from a choice of seven options.
    - The **input.tab** data file contains numerical values resulting from the quantification of text data by WordStat where each row represents a document, and columns consist of frequency information for every item displayed in the Frequencies tab (i.e., either words or content categories). When no categorization process is applied, such a data file corresponds to a Document x Term matrix. A choice of one of four statistics may be stored in this data file: the term's frequencies, the case occurrences (i.e., either 0 or 1), The rate of this term per 10,000 words, or its TF-IDF score. If more than one of these metrics is checked, the user will be asked to select from these options upon execution.
    - The **sources.tab** is the original text data stored in a single file, one document per line. To store a document in a single line carriage returns (ASCII #13) and line feed characters (ASCII #10 had to be replaced with other characters (ASCII 30 and 31).
    - The **tokens.tab** will hold the result of the project's text processing steps including preprocessing, word replacements, stop word removal and categorization. By default, each document is stored on a single line. One may also segment this file by paragraphs or by sentences.
- Once the script options have been set, click **OK** to open the script editor window.



- The **Variables** section at the top allows you to define variables that can be used in the script to customize its execution. Upon execution of a script, if variables have been defined, a dialog box will be presented to set those options.

- To add a new variable to the script, click the button. The variable definition window will appear.



- From the **Type** dropdown menu, select what will be the type of the variable. You may select among seven types of variables: an integer, a floating point, a string, a Boolean, a string, a list of options or a project variable. An additional **Section** type may also be used to group various options into distinct sections. When such an option is selected, you will be asked to enter a string that will be displayed in bold character as the title of the section.

- In the **Name** edit box, type the name of this variable. This is the name that should be inserted into the script source code to customize its execution.

- The **prompt** edit box is the text that will be displayed on the left of the data editing control. The **Description** edit box may also be used to provide additional information about this option. If a description or instructions are provided, a hint window will display this information upon hovering the variable data entry control.

- Depending on the variable type, different specifications can be added:

    - For a **Floating Point** or an **Integer** variable, the range section allows you to set a **Minimum**, **Maximum** and/or **Default** value. First check the box for the desired range specification, then increase or decrease the value via the up/down arrows, or by typing in the numerical value.

    - A Boolean variable can be enabled by default, by checking the default box under the **Description** textbox.

    - For a **List of options**, the strings that will appear as options to be selected must be specified in the **Options** textbox. Each option should be added on a separate line without quotation marks.

    - If a script includes a **Project variable**, choose the type of data that will be expected via the **Data type** dropdown list. Choosing a data type will restrict the list presented to the selected data type. Selecting **Any type** will present a list of all variables on the project. Checking the **Optional** checkbox allows the Project variable to be left blank.

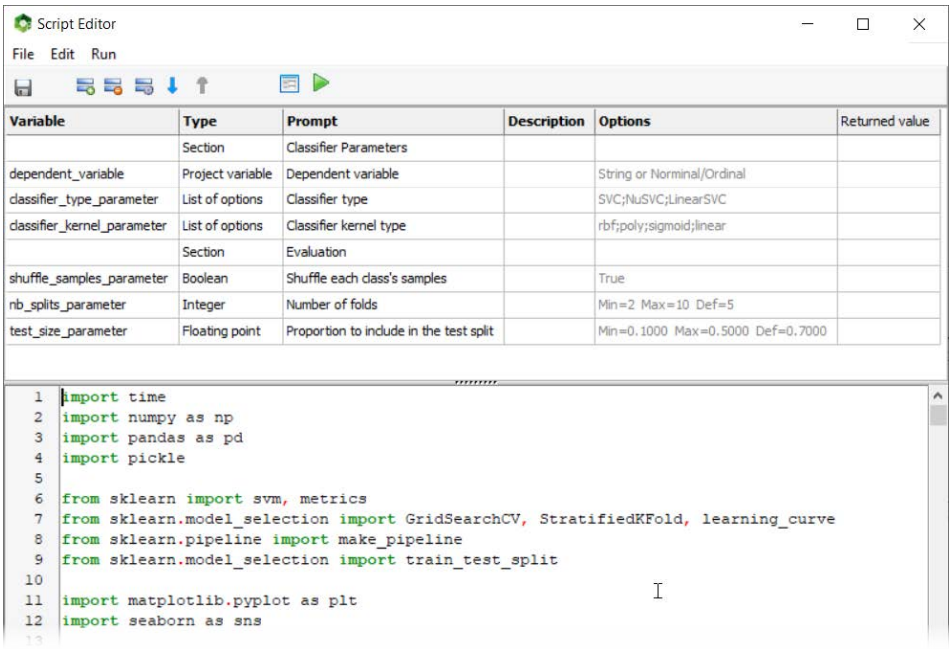
- To remove an existing variable, first select the variable to be deleted and then click the button.

- Select a variable and click the edit variable button to open the variable definition window and make any changes to the existing specifications.

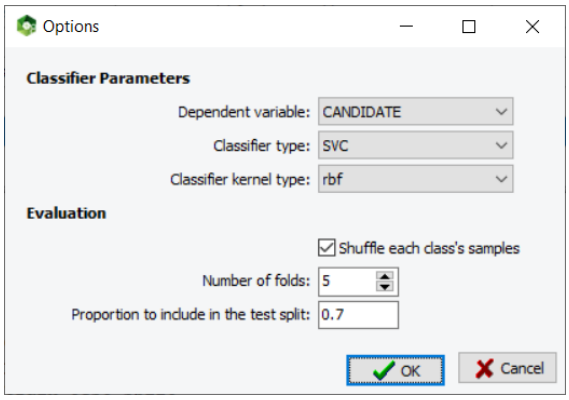
- Click the down arrow to change the order of the variables and bring the selected variable down in the list of variables, or the up arrow to move the variable up.

- The preview button shows the **Options** dialog box as it would display when the script is run, without having to run the script. The dialog box can also be previewed by selecting **Test Dialog Box** under the **Run** menu.

  In the example below, the dialog box consists of six variables grouped under two sections: Classifier Parameters and Evaluation.



  Running the script will cause the following dialog box to appear:



- The bottom section of the script editor window should contain valid Python or R code, and the formatting will reflect the syntax of the script's language. Proper commands for importing needed libraries should be specified at the beginning of the script. Names of the specified variables may be used in the script to customize its execution. The Python script below illustrates how one may use the defined variables in a script (highlighted in yellow) and their associated values set by the user using the dialog box.

```
    data = pd.read_csv(file, sep='\t', encoding='ISO-8859-1')
    data_clean = data.copy()
    data_clean = data_clean.drop(columns=[entry_number])
    if data[dependent_variable].dtype == 'object':
        # Convert categorical variable to numeric
        tags_list = data[dependent_variable].unique()
        tags = {x: i for i, x in enumerate(tags_list)}
        data_clean['cleaned_tag'] = data_clean[dependent_variable].apply(lambda x: tags[x])
    else:
        data_clean['cleaned_tag'] = data_clean[dependent_variable]
    data_clean = data_clean.drop(columns=dependent_variable)
    data_X = data_clean.drop(columns='cleaned_tag')
    data_y = data_clean['cleaned_tag']

    # Split the data into training and testing sets
    return train_test_split(data_X, data_y,
                            test_size=test_size_parameter,
                            random_state=1)

def main():
    """Train SVM classifier
    """
    try:
        # read and split data
        X_train, X_test, y_train, y_test = prepare_data(input_file)

        # cross-validation object
        kfolds = StratifiedKFold(n_splits=nb_splits_parameter, shuffle=shuffle_samples_parameter, random_state=1)

        # Training
        np.random.seed(1)
        svc_param_grid = {}
        # Linear classifier
        if classifier_type_parameter == 'LinearSVC':
            pipeline_svm = make_pipeline(
                getattr(svm, classifier_type_parameter)(class_weight="balanced"))
            svc_param_grid = {'linearsvc__C': [0.01, 0.1, 0.5, 1]}
        else:
            pipeline_svm = make_pipeline(
                getattr(svm, classifier_type_parameter)(probability=True, class_weight="balanced"))
            if classifier_type_parameter == 'NuSVC':
                if not classifier_kernel_parameter == 'linear':
```

- Clicking the ▶ button will run the script. Any changes made will prompt a dialog box asking if changes should be saved before running. Running a script from within the script editor opens a Python or R console. In contrast to running a script from the main post-processing scripts window, the console will also display the script's code in the upper half of the window. The code will include all the variable assignments. Once the script terminates, the console window will have to be closed for the output to be processed. The script can also be run by selecting **Run Script** under the **Run** menu

- A new script can also be created by selecting **New** under the **File** menu of the script editor. There is an option to choose between Python or R.

- Save a new script with the 💾 button or by selecting **Save** from the **File** menu. Selecting **Save as** will open the **Script Options** window allowing you to enter a new name for the script.

Editing a script

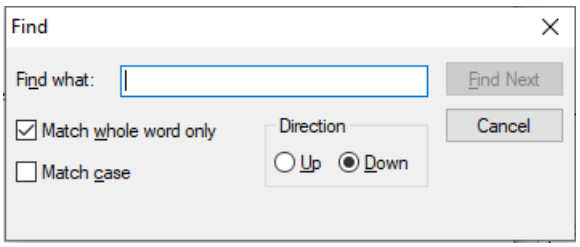
- To edit an existing script, first select it from the **Script** dropdown menu, then click the ▾ button next to the **Run** button and select **Edit Script**.

- Variables can be added, deleted, or edited (See details above)

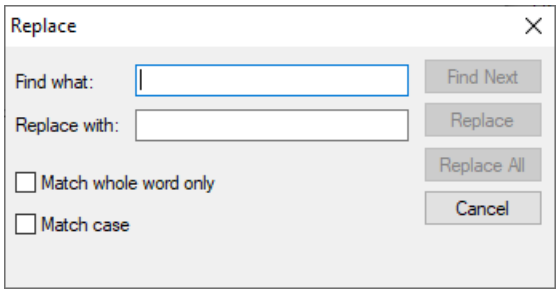
- Script Options of an existing script can also be edited by selecting **Settings** under the **Edit** menu.

- Changes can be made to the code of the script in the Code section of the script editor. Any edits can be typed into this section or using the **Edit** menu. Under the **Edit** menu, you will find common edits such as **Undo**, **Cut**, **Copy**, **Paste**, **Select all**.

- Click **Find** to search for any particular sequence in the code.



- Select **Replace** in the Edit menu to substitute a sequence in the code.



Importing/Exporting a script

Post-processing scripts created in WordStat (whether Python or R) are saved as '.wscr' files in a distant folder not easily accessible. The import and export features have been designed to easily copy an existing script to this folder or create a copy outside of this folder, allowing one to share scripts with other users.

- To import a script, click the button next to the run button and select Import. Select an .wscr script from the file explorer and click Open. A script can also be imported by selecting the **Import** command from the **File** menu in the script editor.

- To export a script, first select it, click the button next to the RUN button and select Edit. Then choose the Export command from the file menu of the script editor. Choose its destination in the file explorer and click Save

Console

Any libraries that are imported in a post-processing script will automatically be installed as the script is run. In some cases, a package requires extra steps for installation, which Wordstat cannot perform under-the-hood. A Python or R console can be opened to perform these installations or any other tasks.

- From the Script Editor window, select **Console** from the **Run** menu
- Depending on the language of the script, a Python or R console will open