# Handbook Vetoboxing

## Version 1.0, 05-06-2016

### Developed by: Kim de Bie

This handbook describes the vetoboxing software package in its current version. It consists of two sections. Firstly, the software and its possibilities are described from the viewpoint of the user: what functionalities does the software currently have, and how can it be used? Secondly, the software, its inner functioning and a detailed outline of the used scripts/functions are described.

#### User guide

In short, the vetoboxing software can be used to play multidimensional strategic voting games. Being a first prototype, the current version of the vetoboxing software has limited functionality, but allows the user nonetheless to perform tedious, repeated calculations in a much higher speed than was previously possible. A first important fact to note is that the software currently does not have a user interface, and has to be run via the command line. In addition, the source code of the software needs to be edited to change the inputs of the simulation. Although this might slightly stretch the comfort zone of those unfamiliar, no great technical expertise is required to run the project in this fashion.

After downloading the software from [www.github.com/kimdebie/vetoboxing](http://www.github.com/kimdebie/vetoboxing) (click *Download ZIP*), open the file *vetoboxing.py*. At the top of the file, the inputs of the simulation can be changed. The following variables should currently be defined as per the input specifications below.

##### Input

This section describes the parameters that can be altered by the user, the way in which these should be entered and their effects on the outcomes of the program.

|  |  |  |
| --- | --- | --- |
| *Variable* | *Meaning* | *Input* |
| runs | The number of times the simulation should be run. | a single integer |
| distance\_type | The method used to calculate distance between two points. Can be Pythagorean or city-block distance. | ‘pyth’ or ‘city-block’ |
| number\_dimensions | The number of dimensions that the simulation uses. When elliptical preference curves are used (see preference\_shape), two dimensions must be used. Currently, the maximum number of dimensions that can be used is 3 (to be extended in later versions). | a single integer: 1, 2 or 3 |
| voter\_A, voter\_B, voter\_C … | The parameters of the voters that participate in the game. Specify the position of the voter, whether they are agenda setter or not and whether they are veto player or not. Voters may be removed when not used. Currently, the maximum number of voters that can be used is 5 (to be extended). Note that input is different for elliptical or circular preferences. | *for circular preferences:*  Voter( (X\_POSITION, Y\_POSITION), **AGENDA\_SETTER**, **VETO\_PLAYER** )  *for elliptical preferences:*  Voter ( (X\_POSITION, Y\_POSITION, WIDTH\_ELLIPS, HEIGHT\_ELLIPS, ANGLE), **AGENDA\_SETTER**, **VETO\_PLAYER** )  *All underlined variables are decimal numbers; bold variables are Booleans: True or False* |
| preference\_shape | Specify whether preferences of voters are circular or elliptical. | ‘ellipse’ or ‘circle’ |
| voters | A list of all the voters that the user wants to include in the program. The voters must be specified as per the instructions above. | [name\_voter1, name\_voter2, name\_voter3] |
| status\_quo | The current policy position. | Decimal numbers; corresponding to the number of dimensions. Like so:  (2.0, 4.5, 2.1) |
| alter\_status\_quo | Specifies whether the status quo is to be altered in consecutive runs of the visualization. Options: the status quo is not altered; the status quo has a random drift; the status quo is the outcome of the previous game; the status quo is the outcome of the previous game in addition to some random drift. | ‘no’, ‘random’, ‘history’, ‘history and drift’ |
| drift\_status\_quo | Specifies the drift of the status quo in each dimension – only necessary when alter\_status\_quo is defined as ‘history and drift’. | Decimal numbers; corresponding to the number of dimensions. Like so:  [0.2, 0.4, 0.0] |
| distribution\_type | Determine what type of distribution is used for random draws in altering the positions of players. | ‘normal’, ‘uniform’, ‘exponential’, ‘paretian’ |
| dummy\_type | Determine what method is used for determining the outcome of the game. Currently, a point-grid method is used: a grid of points is created with a distance from each other as specified by the user. For each iteration, the optimal point from the grid of points is selected. | 0 |
| grid\_size | The size of the grid or the ‘policy playing field’; the outcome on each dimension will lie within the boundaries of the grid. The grid is of equal size on all dimensions. | An integer >0 |
| breaks | The interval at which points should be placed in the grid. | A decimal number >0 |
| save\_results | Specifies whether results of the simulation are saved in a .csv file. | True; False |
| visualize | Specifies whether the simulation is to be visualized. Only two-dimensional games can be visualized. | True; False |

##### Running the software

To run the software, open a command line interface (Windows PowerShell or Terminal for Mac). Change the directory to the folder ‘vetoboxing’ using “cd */filepath-to-folder-vetoboxing/*”. From here, run the software “*python vetoboxing.py”.* The command line will provide you with an indication of the status of the program. Speed of the program varies a lot with the parameters as specified by the user.

##### Reviewing the results

Results of simulations are stored in a folder inside the folder ‘vetoboxing’, with a name specified by the user. Textual outputs of results are stored in .csv format. Filenames are formatted as follows:

*results\_modelnumber-dimensions-vetoplayers\_\_datestamp.csv*

The model number specifies one of the following:

1. baseline veto player model (random draw status quo - no history)
2. dynamic veto player model (random walk status quo - with history)
3. dynamic veto player model (random walk status quo - with history and biased drift)
4. dynamic veto player model (random walk status quo - with history and biased drift, and preferences with drift)

Dimensions specifies the number of dimensions (1, 2 or 3), vetoplayers specifies the number of veto players (1, 2 or 3).

In the .csv file, the following variables are stored: the positions of players at the start of each game, the positions of veto players, the status quo, the outcome or policy decision, the total distance travelled across all dimensions (Pythagorean and Manhattan distance) and the distance travelled in each single dimension.

If specified by the user, visual representations of the output are also generated. Visual representations can only be created of two-dimensional simulations. These files are stored in a folder with a name equal to the filename of the .csv file. For each run, a .png file is stored, where the name of the file simply represents the number of the run. In addition, an .mp4 file is generated which displays the simulation dynamically.

#### Developers’ guide

The vetoboxing software is written in Python version 2.7.10. The software consists of two files, namely *vetoboxing.py*, which contains all main functions, and *error.py*, which provides an error-checking module. Extensive descriptions of functions used are provided in the software. *This developers guide is to be extended.*