# Empirical values of fairy chess pieces

**Abstract**

We use *fairy Stockfish* (the current best engine for chess-like games) to play a variant of *chess with different armies* and generate more than 600 millions of positions. With the help of different empirical methods, we estimate the values of many fairy chess pieces.

## Contents

# 1 Introduction

We start with an array summarizing the results from the first method:

| Name | Letter | Notation | Empirical method 1 value |
|---|---|---|---|
| Queen | Q | BR | 8.50 |
| Chancellor | C | NR | 7.62 |
| Archbishop | A | BN | 6.79 |
| Centaur | Y | KN | 6.05 |
| Bers | X | KR | 5.98 |
| DragonHorse | D | BK | 4.89 |
| Rook | R | R | 4.26 |
| Bisroo | I | mBcR | 3.87 |
| Kniroo | M | mNcR | 3.84 |
| Manroo | G | mKcR | 3.74 |
| Roobis | V | mRcB | 3.64 |
| Rooman | W | mRcK | 3.28 |
| Bishop | B | B | 3.28 |
| Manbis | F | mKcB | 3.26 |
| Rookni | U | mRcN | 3.07 |
| Kniman | O | mNcK | 3.04 |
| Knight | N | N | 2.97 |
| Mankni | E | mKcN | 2.89 |
| Bisman | J | mBcK | 2.89 |
| Biskni | H | mBcN | 2.49 |
| Pawn | P | P | 1.00 |

The results from our second method can be illustrated by the following image:

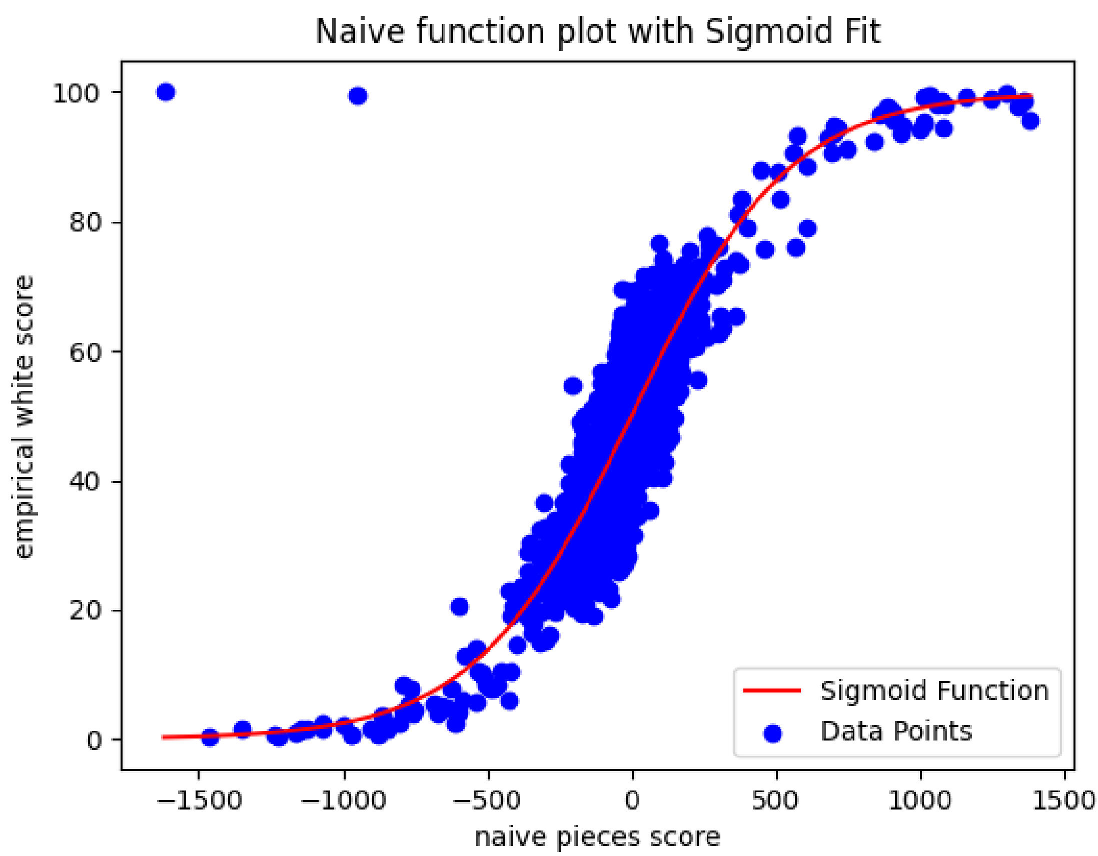


Figure 1: Armies within 50 centipawns of the classical chess armies should be roughly balanced.

# 2 Motivation

Our goal is to define a chess variant akin to Ralph Betza's *Chess with different armies*, where the rules are exactly the same as chess, but white and black pieces may be replaced by fairy chess pieces. This type of variant has been developed in many different ways throughout the years. For instance, one may also mention the following variants implemented in `pychess.org/variants`:

- Orda
- Synochess
- Shinobi
- Empire
- Spartan

The current work tries to fulfil (at least partially) Betza's dream[1]:

> ≪*I wanted to be able to let the players choose their pieces from a list of pieces and values, the way war games work with "buy points", or the way "fantasy leagues" work.* ≫

For instance, one usually assigns the values

- Pawn = 1
- Knight = 3
- Bishop = 3
- Rook = 5
- Queen = 9

which yield a good heuristic for human beings to evaluate any position, and could also be used to find new balanced setups[2] Unfortunately, as noted by Betza :

> ≪I found that the values aren't sufficiently precise, and that the team as a whole must be considered. Some pieces work together well, and choosing them in combination adds to the value of the army as a whole; other pieces do not, and an army made up of them will be weaker on the board than it would seem to be "on paper". ≫

[1]Source: `https://www.chessvariants.com/unequal.dir/cwda.html`

[2]e.g. **rnnqkbbr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1** seems balanced.

For instance, the setup

**nrbqkbrn/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQ - 0 1**

is biased toward white (though one may argue that white advantage may be balanced enough at short time control, or in casual play, or if players switch sides).

In order to alleviate the problem, one may compute piece square tables, or at least values for the starting squares: A1, B1, C1 and D1. For instance, we find below that:

- Knight starting on A1 = 2.81 pawns
- Knight starting on B1 = 2.99 pawns
- Knight starting on C1 = 2.87 pawns
- Knight starting on D1 = 2.99 pawns

These values have to be taken with a grain of salt as margins of error are still unsatisfactory. Nevertheless, the values computed below could be use to generate millions of starting setups that should be balanced for casual players.

# 3 Definition of our variant

In order to generate millions of games with different setups, we use the engine Fairy-Stockfish (version downloaded 2023-08-23) with the added variant:

```
#different armies
[armies:chess]
#customPiece1 = s:fmWfceF
customPiece2 = t:fsmWfceF
customPiece3 = z:mWfceFifmnD
knight = n
bishop = b
rook = r
king = k
customPiece4 = e:mKcN
customPiece5 = f:mKcB
customPiece6 = g:mKcR
customPiece7 = h:mBcN
```

```
customPiece8 = i:mBcR
customPiece9 = j:mBcK
customPiece10 = l:mBcK
customPiece11 = m:mNcR
customPiece12 = o:mNcK
customPiece13 = u:mRcN
customPiece14 = v:mRcB
customPiece15 = w:mRcK
queen = q
archbishop = a
dragonHorse = d
chancellor = c
bers = x
centaur = y
pawnTypes = p
promotionPieceTypes = nbrq
promotedPieceType = p:q
```

In other words, the rules are exactly the same as chess, but one may also play with fairy pieces and there are **no castling rights**. For instance, `customPiece15 = w:mRcK` defines a piece (named Rooman) that can move as a Rook and capture as a Mann (non-royal King).
We make the following remarks:

- The piece with the letter J and the piece with the letter L are the **same**. This allows us to have a better view on our data and our empirical methods: in theory, the value of J given by one empirical method should be exactly the same as the value of L.

- If the starting position is the classical starting position of chess, then this variant is exactly chess. This is partly due to the fact that pawns promotes to Queen, Rook, Knight and Bishop. For practical plays between human beings, it may be better to allow promotion to other fairy pieces.

- The pieces with S, T and Z never appears in the generated games. They represent pawn-type units that may be studied in a subsequent work.

- The starting position of the King plays an important part in how a setup is balanced. This is why we have fixed its starting position to the E1 square.

- Similarly, the symmetry of the starting position may help with balance issues. This is why we have only generated games where, at the start, the A-file corresponds to the H-file, the B-file corresponds to the G-file and the C-file corresponds to the F-file.

- Except for the J and L fairy pieces, we add the restriction that an army cannot have more than two pieces of the same type at start of the game. For instance, we exclude the starting position

  **rnnqknnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w**

- For instance, generic starting positions may look like

  **aergkrea/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w**

  **qdfjkfdq/pppppppp/8/8/8/8/PPPPPPPP/ILYUKYLI w**

- With the above restrictions, there are exactly $21 \times 20 \times 19 \times 18 = 143640$ armies, and $143640^2 = 2.06324496 \times 10^{10}$ starting positions.

# 4 Data generation

The procedure to generate games is as follows:

1. Pick a starting position, e.g.

   **qdfjkfdq/pppppppp/8/8/8/8/PPPPPPPP/ilyukyli w**

2. Use the script available at `github.com/ianfab/chess-variant-mcts` to generate an MCTS book. This yields an mcts-book.epd file with around 1000 positions.

3. Use the scripts available at `github.com/ianfab/chess-variant-stats` to:

   (a) generate 100000 positions[3], starting from the above mcts-book.epd file (this amounts to around 800 games),

[3] 10 ms per position on 1 thread, Processor = 12th Gen Intel® Core™ i5-1235U.

(b) report basic game statistics, such as result distribution, game length, and branching factor,

(c) fit piece values on the generated data using logistic regression (the values are normalized by fixing 1 pawn $= 1.00$).

The above procedure went on 6089 times (for a total amount of 120 GB of data), there are thus more than 600 millions positions generated.

We note that the starting position picked in Step 1 is not random but was chosen (thanks to some heuristics) with the aim to be somewhat balanced with the classical chess armies RNBQKBNR.

# 5 Non-exhaustive list of balanced armies

According to the data generated, we list below some armies that are balanced against the classical chess armies. By *balanced*, we mean a white score between $45\%$ and $55\%$.

```
gioqkoig_RNBQKBNR – White score: 48.2
exdokdxe_RNBQKBNR – White score: 54.0
vmdnkdmv_RNBQKBNR – White score: 54.629999999999995
rnbqkbnr_RNBQKBNR – White score: 52.235
wyifkiyw_RNBQKBNR – White score: 46.699999999999996
imxjkxmi_RNBQKBNR – White score: 48.795
wbahkabw_RNBQKBNR – White score: 46.385000000000005
dmiwkimd_RNBQKBNR – White score: 51.69
ixghkgxi_RNBQKBNR – White score: 50.07000000000001
bgxfkxgb_RNBQKBNR – White score: 49.475
xgrekrgx_RNBQKBNR – White score: 53.44
grmxkmrg_RNBQKBNR – White score: 54.52
ifylkyfi_RNBQKBNR – White score: 45.63
gfxukxfg_RNBQKBNR – White score: 52.235
xjeakejx_RNBQKBNR – White score: 49.394999999999996
jxoakoxj_RNBQKBNR – White score: 46.78
xmeikemx_RNBQKBNR – White score: 53.269999999999996
xdhwkhdx_RNBQKBNR – White score: 50.25
xhdlkdhx_RNBQKBNR – White score: 54.115
havjkvah_RNBQKBNR – White score: 46.82
jvaekavj_RNBQKBNR – White score: 49.97
mynwknym_RNBQKBNR – White score: 46.8
dirwkrid_RNBQKBNR – White score: 49.05
```

```
vxldklxv_RNBQKBNR - White score: 45.45
dlbakbld_RNBQKBNR - White score: 48.335
hryikyrh_RNBQKBNR - White score: 46.239999999999995
nxmfkmxn_RNBQKBNR - White score: 54.67
hydokdyh_RNBQKBNR - White score: 47.645
xevdkvex_RNBQKBNR - White score: 54.815
ivxlkxvi_RNBQKBNR - White score: 50.669999999999995
wvmckmvw_RNBQKBNR - White score: 51.665000000000006
xjidkijx_RNBQKBNR - White score: 47.0
xovmkvox_RNBQKBNR - White score: 48.815
fvwckwvf_RNBQKBNR - White score: 47.184999999999995
fwxikxwf_RNBQKBNR - White score: 53.58
vxfjkfxv_RNBQKBNR - White score: 47.29
xhgdkghx_RNBQKBNR - White score: 54.910000000000004
vwxikxwv_RNBQKBNR - White score: 50.365
hvalkavh_RNBQKBNR - White score: 47.980000000000004
rgmxkmgr_RNBQKBNR - White score: 54.919999999999995
rxwikwxr_RNBQKBNR - White score: 54.325
jxbykbxj_RNBQKBNR - White score: 46.58
xljakjlx_RNBQKBNR - White score: 50.214999999999996
vdrnkrdv_RNBQKBNR - White score: 50.05
wgyikygw_RNBQKBNR - White score: 48.150000000000006
ghaokahg_RNBQKBNR - White score: 45.255
ogyfkygo_RNBQKBNR - White score: 51.379999999999995
mynuknym_RNBQKBNR - White score: 45.785000000000004
oxfdkfxo_RNBQKBNR - White score: 47.535
giuqkuig_RNBQKBNR - White score: 52.995000000000005
xirhkrix_RNBQKBNR - White score: 49.519999999999996
jafwkfaj_RNBQKBNR - White score: 45.209999999999994
xwrikrwx_RNBQKBNR - White score: 51.155
hcufkuch_RNBQKBNR - White score: 46.964999999999996
vrwckwrv_RNBQKBNR - White score: 48.55
uryikyru_RNBQKBNR - White score: 51.339999999999996
fgyhkygf_RNBQKBNR - White score: 47.435
wbalkabw_RNBQKBNR - White score: 48.57
fmuqkumf_RNBQKBNR - White score: 50.6
mvbakbvm_RNBQKBNR - White score: 47.64
wclhklcw_RNBQKBNR - White score: 49.465
hwagkawh_RNBQKBNR - White score: 50.660000000000004
vfuqkufv_RNBQKBNR - White score: 46.56
```

# 6 Empirical method 1: Explanations

The first empirical method simply gather the information generated:

- For each fairy piece (represented by its letter), compute the average value for all starting position (e.g. see `List A` below for the Archbishop)
- For each fairy piece (represented by its letter), compute the average value, but only for starting position where the fairy piece starts on a fixed file (e.g. `A1list` corresponds to starting positions with the archbishops on A1/H1 or A8/H8).

The results are listed in the next section. We make the following remarks:

- In the results below, `Sample Size:  1608` means that there were 1608 setups tested with an Archbishop (i.e. around 160 millions positions).
- We refer to the appendix for pretty pictures representing the results.
- The margins of error are still unsatisfactory.
- The results are consistent with other computations of the values for classical chess. However, we note that the value of the rook is $4.26$ which is slightly less than expected (same remark for the Queen); this may be explained by several factors, one of which is the following:
  - Games in this variant are shorter than in classical chess. The average number of pieces on the board is around $18$ which may higher than in classical chess. As a consequence, heavy pieces like the Rook or the Queen cannot move up to their full potential on the board and thus have a value closer to minor pieces.
- We warn the reader that the value of a piece on a given file (e.g. `A1` below) only means that the piece started on this file in the beginning of the game. One thing to do with the generated games would be to extract all position where a given piece is **always** on a given square (say A1) and compute its value as above.

# 7 Empirical method 1: Results

List A:
Sample Size: 1608
Mean: 6.79
Median: 6.72
Mean \pm 1 Std Dev:
6.79 \pm 0.94
Maximum value: 9.93
Minimum value: 4.05

List B:
Sample Size: 3140
Mean: 3.28
Median: 3.24
Mean \pm 1 Std Dev:
3.28 \pm 0.45
Maximum value: 4.86
Minimum value: 1.72

List C:
Sample Size: 1109
Mean: 7.62
Median: 7.53
Mean \pm 1 Std Dev:
7.62 \pm 1.06
Maximum value: 11.16
Minimum value: 4.72

List D:
Sample Size: 1205
Mean: 4.89
Median: 4.84
Mean \pm 1 Std Dev:
4.89 \pm 0.65
Maximum value: 7.19
Minimum value: 2.66

List E:
Sample Size: 855
Mean: 2.89
Median: 2.84
Mean \pm 1 Std Dev:
2.89 \pm 0.43
Maximum value: 4.18
Minimum value: 1.81

List F:
Sample Size: 1379
Mean: 3.26
Median: 3.23
Mean \pm 1 Std Dev:
3.26 \pm 0.45
Maximum value: 4.78
Minimum value: 1.84

List G:
Sample Size: 1212
Mean: 3.74
Median: 3.71
Mean \pm 1 Std Dev:
3.74 \pm 0.48
Maximum value: 5.44
Minimum value: 2.50

List H:
Sample Size: 1390
Mean: 2.49
Median: 2.47
Mean \pm 1 Std Dev:
2.49 \pm 0.36
Maximum value: 3.62
Minimum value: 1.31

List I:
Sample Size: 1398
Mean: 3.87
Median: 3.84

Mean \pm 1 Std Dev:
3.87 \pm 0.50
Maximum value: 5.58
Minimum value: 2.44

List J:
Sample Size: 1273
Mean: 2.90
Median: 2.88
Mean \pm 1 Std Dev:
2.90 \pm 0.40
Maximum value: 4.19
Minimum value: 1.71

List L:
Sample Size: 1093
Mean: 2.87
Median: 2.84
Mean \pm 1 Std Dev:
2.87 \pm 0.41
Maximum value: 4.30
Minimum value: 1.46

List M:
Sample Size: 1162
Mean: 3.84
Median: 3.80
Mean \pm 1 Std Dev:
3.84 \pm 0.49
Maximum value: 5.69
Minimum value: 2.46

List N:
Sample Size: 3206
Mean: 2.97
Median: 2.94
Mean \pm 1 Std Dev:
2.97 \pm 0.43
Maximum value: 4.39
Minimum value: 1.79

List O:
Sample Size: 1324
Mean: 3.04
Median: 3.00
Mean \pm 1 Std Dev:
3.04 \pm 0.44
Maximum value: 4.49
Minimum value: 1.75

List Q:
Sample Size: 2879
Mean: 8.50
Median: 8.44
Mean \pm 1 Std Dev:
8.50 \pm 1.14
Maximum value: 12.53
Minimum value: 4.85

List R:
Sample Size: 3087
Mean: 4.26
Median: 4.22
Mean \pm 1 Std Dev:
4.26 \pm 0.56
Maximum value: 6.26
Minimum value: 2.34

List U:
Sample Size: 1040
Mean: 3.07
Median: 3.04
Mean \pm 1 Std Dev:
3.07 \pm 0.43
Maximum value: 4.59
Minimum value: 1.87

List V:
Sample Size: 1469
Mean: 3.64

Median: 3.59
Mean \pm 1 Std Dev:
3.64 \pm 0.46
Maximum value: 5.29
Minimum value: 2.43

List W:
Sample Size: 1099
Mean: 3.28
Median: 3.26
Mean \pm 1 Std Dev:
3.28 \pm 0.44
Maximum value: 4.83
Minimum value: 1.70

List X:
Sample Size: 1812
Mean: 5.98
Median: 5.92
Mean \pm 1 Std Dev:
5.98 \pm 0.80
Maximum value: 8.80
Minimum value: 3.36

List Y:
Sample Size: 1424
Mean: 6.05
Median: 5.99
Mean \pm 1 Std Dev:
6.05 \pm 0.80
Maximum value: 8.58
Minimum value: 3.78

A1list:
Sample Size: 361
Mean: 6.92
Median: 6.86
Mean \pm 1 Std Dev:
6.92 \pm 0.92
Maximum value: 9.58
Minimum value: 4.31

A2list:
Sample Size: 350
Mean: 6.74
Median: 6.67
Mean \pm 1 Std Dev:
6.74 \pm 0.91
Maximum value: 9.93
Minimum value: 4.05

A3list:
Sample Size: 376
Mean: 6.80
Median: 6.75
Mean \pm 1 Std Dev:
6.80 \pm 0.87
Maximum value: 9.81
Minimum value: 4.86

A4list:
Sample Size: 521
Mean: 6.71
Median: 6.63
Mean \pm 1 Std Dev:
6.71 \pm 1.02
Maximum value: 9.76
Minimum value: 4.22

B1list:
Sample Size: 277
Mean: 3.07
Median: 3.05
Mean \pm 1 Std Dev:
3.07 \pm 0.39
Maximum value: 4.80
Minimum value: 2.16

B2list:
Sample Size: 297

Mean: 3.15
Median: 3.11
Mean \pm 1 Std Dev:
3.15 \pm 0.42
Maximum value: 4.84
Minimum value: 2.17

B3list:
Sample Size: 2265
Mean: 3.34
Median: 3.30
Mean \pm 1 Std Dev:
3.34 \pm 0.45
Maximum value: 4.86
Minimum value: 1.72

B4list:
Sample Size: 301
Mean: 3.08
Median: 3.02
Mean \pm 1 Std Dev:
3.08 \pm 0.44
Maximum value: 4.55
Minimum value: 1.90

C1list:
Sample Size: 187
Mean: 7.58
Median: 7.42
Mean \pm 1 Std Dev:
7.58 \pm 1.02
Maximum value: 10.43
Minimum value: 4.72

C2list:
Sample Size: 452
Mean: 7.77
Median: 7.68
Mean \pm 1 Std Dev:
7.77 \pm 1.03
Maximum value: 11.16
Minimum value: 5.40

C3list:
Sample Size: 239
Mean: 7.37
Median: 7.30
Mean \pm 1 Std Dev:
7.37 \pm 1.04
Maximum value: 10.10
Minimum value: 4.85

C4list:
Sample Size: 231
Mean: 7.65
Median: 7.59
Mean \pm 1 Std Dev:
7.65 \pm 1.11
Maximum value: 11.08
Minimum value: 4.99

D1list:
Sample Size: 266
Mean: 4.89
Median: 4.84
Mean \pm 1 Std Dev:
4.89 \pm 0.66
Maximum value: 7.01
Minimum value: 2.66

D2list:
Sample Size: 234
Mean: 4.92
Median: 4.88
Mean \pm 1 Std Dev:
4.92 \pm 0.63
Maximum value: 6.93
Minimum value: 2.78

D3list:

Sample Size: 411
Mean: 4.82
Median: 4.73
Mean \pm 1 Std Dev:
4.82 \pm 0.66
Maximum value: 6.88
Minimum value: 3.41

D4list:
Sample Size: 294
Mean: 4.95
Median: 4.92
Mean \pm 1 Std Dev:
4.95 \pm 0.64
Maximum value: 7.19
Minimum value: 3.04

E1list:
Sample Size: 217
Mean: 2.76
Median: 2.74
Mean \pm 1 Std Dev:
2.76 \pm 0.37
Maximum value: 3.94
Minimum value: 1.81

E2list:
Sample Size: 224
Mean: 2.88
Median: 2.80
Mean \pm 1 Std Dev:
2.88 \pm 0.43
Maximum value: 4.12
Minimum value: 1.92

E3list:
Sample Size: 202
Mean: 2.94
Median: 2.93
Mean \pm 1 Std Dev:
2.94 \pm 0.41
Maximum value: 4.07
Minimum value: 2.01

E4list:
Sample Size: 212
Mean: 2.97
Median: 2.96
Mean \pm 1 Std Dev:
2.97 \pm 0.47
Maximum value: 4.18
Minimum value: 1.96

F1list:
Sample Size: 374
Mean: 3.20
Median: 3.17
Mean \pm 1 Std Dev:
3.20 \pm 0.41
Maximum value: 4.74
Minimum value: 2.17

F2list:
Sample Size: 326
Mean: 3.21
Median: 3.17
Mean \pm 1 Std Dev:
3.21 \pm 0.43
Maximum value: 4.76
Minimum value: 2.22

F3list:
Sample Size: 414
Mean: 3.34
Median: 3.35
Mean \pm 1 Std Dev:
3.34 \pm 0.43
Maximum value: 4.78
Minimum value: 2.32

F4list:
Sample Size: 265
Mean: 3.26
Median: 3.20
Mean \pm 1 Std Dev:
3.26 \pm 0.52
Maximum value: 4.67
Minimum value: 1.84

G1list:
Sample Size: 385
Mean: 3.68
Median: 3.65
Mean \pm 1 Std Dev:
3.68 \pm 0.47
Maximum value: 5.30
Minimum value: 2.52

G2list:
Sample Size: 242
Mean: 3.78
Median: 3.74
Mean \pm 1 Std Dev:
3.78 \pm 0.45
Maximum value: 5.03
Minimum value: 2.81

G3list:
Sample Size: 332
Mean: 3.80
Median: 3.77
Mean \pm 1 Std Dev:
3.80 \pm 0.47
Maximum value: 5.13
Minimum value: 2.70

G4list:
Sample Size: 253
Mean: 3.72
Median: 3.71
Mean \pm 1 Std Dev:
3.72 \pm 0.54
Maximum value: 5.44
Minimum value: 2.50

H1list:
Sample Size: 502
Mean: 2.42
Median: 2.38
Mean \pm 1 Std Dev:
2.42 \pm 0.34
Maximum value: 3.41
Minimum value: 1.56

H2list:
Sample Size: 248
Mean: 2.50
Median: 2.49
Mean \pm 1 Std Dev:
2.50 \pm 0.35
Maximum value: 3.62
Minimum value: 1.48

H3list:
Sample Size: 375
Mean: 2.60
Median: 2.57
Mean \pm 1 Std Dev:
2.60 \pm 0.34
Maximum value: 3.57
Minimum value: 1.31

H4list:
Sample Size: 265
Mean: 2.48
Median: 2.45
Mean \pm 1 Std Dev:
2.48 \pm 0.40
Maximum value: 3.62
Minimum value: 1.54

I1list:
Sample Size: 382
Mean: 3.78
Median: 3.73
Mean \pm 1 Std Dev:
3.78 \pm 0.49
Maximum value: 5.58
Minimum value: 2.65

I2list:
Sample Size: 399
Mean: 3.91
Median: 3.86
Mean \pm 1 Std Dev:
3.91 \pm 0.47
Maximum value: 5.57
Minimum value: 2.79

I3list:
Sample Size: 331
Mean: 3.96
Median: 3.94
Mean \pm 1 Std Dev:
3.96 \pm 0.47
Maximum value: 5.54
Minimum value: 2.78

I4list:
Sample Size: 286
Mean: 3.85
Median: 3.81
Mean \pm 1 Std Dev:
3.85 \pm 0.53
Maximum value: 5.42
Minimum value: 2.44

J1list:
Sample Size: 273
Mean: 2.74
Median: 2.71
Mean \pm 1 Std Dev:
2.74 \pm 0.35
Maximum value: 3.88
Minimum value: 1.94

J2list:
Sample Size: 215
Mean: 2.84
Median: 2.81
Mean \pm 1 Std Dev:
2.84 \pm 0.39
Maximum value: 3.87
Minimum value: 1.71

J3list:
Sample Size: 343
Mean: 2.88
Median: 2.88
Mean \pm 1 Std Dev:
2.88 \pm 0.37
Maximum value: 4.03
Minimum value: 1.87

J4list:
Sample Size: 442
Mean: 3.05
Median: 3.06
Mean \pm 1 Std Dev:
3.05 \pm 0.41
Maximum value: 4.19
Minimum value: 2.03

L1list:
Sample Size: 289
Mean: 2.75
Median: 2.69
Mean \pm 1 Std Dev:
2.75 \pm 0.35
Maximum value: 4.10

```
Minimum value: 1.83

L2list:
Sample Size: 313
Mean: 2.85
Median: 2.82
Mean \pm 1 Std Dev:
2.85 \pm 0.42
Maximum value: 4.30
Minimum value: 1.92

L3list:
Sample Size: 251
Mean: 2.94
Median: 2.95
Mean \pm 1 Std Dev:
2.94 \pm 0.41
Maximum value: 4.30
Minimum value: 1.46

L4list:
Sample Size: 240
Mean: 2.95
Median: 2.96
Mean \pm 1 Std Dev:
2.95 \pm 0.44
Maximum value: 4.24
Minimum value: 1.76

M1list:
Sample Size: 368
Mean: 3.82
Median: 3.79
Mean \pm 1 Std Dev:
3.82 \pm 0.43
Maximum value: 5.46
Minimum value: 2.89

M2list:
Sample Size: 275
Mean: 3.85
Median: 3.79
Mean \pm 1 Std Dev:
3.85 \pm 0.49
Maximum value: 5.43
Minimum value: 2.74

M3list:
Sample Size: 269
Mean: 3.85
Median: 3.83
Mean \pm 1 Std Dev:
3.85 \pm 0.49
Maximum value: 5.08
Minimum value: 2.72

M4list:
Sample Size: 250
Mean: 3.86
Median: 3.82
Mean \pm 1 Std Dev:
3.86 \pm 0.58
Maximum value: 5.69
Minimum value: 2.46

N1list:
Sample Size: 224
Mean: 2.81
Median: 2.81
Mean \pm 1 Std Dev:
2.81 \pm 0.34
Maximum value: 4.26
Minimum value: 2.05

N2list:
Sample Size: 2330
Mean: 2.99
Median: 2.96
Mean \pm 1 Std Dev:
2.99 \pm 0.44
```

Maximum value: 4.39
Minimum value: 1.79

N3list:
Sample Size: 269
Mean: 2.87
Median: 2.83
Mean \pm 1 Std Dev:
2.87 \pm 0.37
Maximum value: 4.27
Minimum value: 1.93

N4list:
Sample Size: 383
Mean: 2.99
Median: 2.98
Mean \pm 1 Std Dev:
2.99 \pm 0.42
Maximum value: 4.31
Minimum value: 1.91

O1list:
Sample Size: 234
Mean: 2.93
Median: 2.92
Mean \pm 1 Std Dev:
2.93 \pm 0.38
Maximum value: 4.03
Minimum value: 1.83

O2list:
Sample Size: 379
Mean: 3.09
Median: 3.07
Mean \pm 1 Std Dev:
3.09 \pm 0.45
Maximum value: 4.35
Minimum value: 2.06

O3list:
Sample Size: 350
Mean: 3.01
Median: 2.98
Mean \pm 1 Std Dev:
3.01 \pm 0.40
Maximum value: 4.32
Minimum value: 1.98

O4list:
Sample Size: 361
Mean: 3.10
Median: 3.06
Mean \pm 1 Std Dev:
3.10 \pm 0.48
Maximum value: 4.49
Minimum value: 1.75

Q1list:
Sample Size: 74
Mean: 7.80
Median: 7.59
Mean \pm 1 Std Dev:
7.80 \pm 1.29
Maximum value: 12.02
Minimum value: 4.85

Q2list:
Sample Size: 83
Mean: 8.01
Median: 7.76
Mean \pm 1 Std Dev:
8.01 \pm 1.32
Maximum value: 11.99
Minimum value: 5.29

Q3list:
Sample Size: 83
Mean: 7.68
Median: 7.57
Mean \pm 1 Std Dev:

7.68 \pm 1.24
Maximum value: 12.08
Minimum value: 5.03

Q4list:
Sample Size: 2639
Mean: 8.56
Median: 8.49
Mean \pm 1 Std Dev:
8.56 \pm 1.11
Maximum value: 12.53
Minimum value: 5.50

R1list:
Sample Size: 2325
Mean: 4.23
Median: 4.19
Mean \pm 1 Std Dev:
4.23 \pm 0.57
Maximum value: 6.26
Minimum value: 2.63

R2list:
Sample Size: 228
Mean: 4.43
Median: 4.41
Mean \pm 1 Std Dev:
4.43 \pm 0.55
Maximum value: 6.13
Minimum value: 2.34

R3list:
Sample Size: 259
Mean: 4.35
Median: 4.25
Mean \pm 1 Std Dev:
4.35 \pm 0.53
Maximum value: 6.12
Minimum value: 3.03

R4list:
Sample Size: 275
Mean: 4.27
Median: 4.23
Mean \pm 1 Std Dev:
4.27 \pm 0.52
Maximum value: 5.90
Minimum value: 2.97

U1list:
Sample Size: 253
Mean: 3.07
Median: 3.02
Mean \pm 1 Std Dev:
3.07 \pm 0.42
Maximum value: 4.48
Minimum value: 1.97

U2list:
Sample Size: 253
Mean: 3.05
Median: 2.99
Mean \pm 1 Std Dev:
3.05 \pm 0.45
Maximum value: 4.59
Minimum value: 2.09

U3list:
Sample Size: 313
Mean: 3.06
Median: 3.06
Mean \pm 1 Std Dev:
3.06 \pm 0.40
Maximum value: 4.29
Minimum value: 2.30

U4list:
Sample Size: 221
Mean: 3.10
Median: 3.07

Mean \pm 1 Std Dev:
3.10 \pm 0.44
Maximum value: 4.42
Minimum value: 1.87

V1list:
Sample Size: 340
Mean: 3.65
Median: 3.60
Mean \pm 1 Std Dev:
3.65 \pm 0.45
Maximum value: 5.17
Minimum value: 2.58

V2list:
Sample Size: 447
Mean: 3.68
Median: 3.62
Mean \pm 1 Std Dev:
3.68 \pm 0.48
Maximum value: 5.29
Minimum value: 2.43

V3list:
Sample Size: 409
Mean: 3.60
Median: 3.56
Mean \pm 1 Std Dev:
3.60 \pm 0.44
Maximum value: 4.96
Minimum value: 2.57

V4list:
Sample Size: 273
Mean: 3.63
Median: 3.61
Mean \pm 1 Std Dev:
3.63 \pm 0.49
Maximum value: 5.29
Minimum value: 2.49

W1list:
Sample Size: 256
Mean: 3.22
Median: 3.18
Mean \pm 1 Std Dev:
3.22 \pm 0.43
Maximum value: 4.36
Minimum value: 2.21

W2list:
Sample Size: 210
Mean: 3.28
Median: 3.21
Mean \pm 1 Std Dev:
3.28 \pm 0.44
Maximum value: 4.55
Minimum value: 2.20

W3list:
Sample Size: 214
Mean: 3.27
Median: 3.28
Mean \pm 1 Std Dev:
3.27 \pm 0.45
Maximum value: 4.83
Minimum value: 1.70

W4list:
Sample Size: 419
Mean: 3.31
Median: 3.28
Mean \pm 1 Std Dev:
3.31 \pm 0.45
Maximum value: 4.77
Minimum value: 2.17

X1list:
Sample Size: 531
Mean: 6.02

Median: 5.96
Mean \pm 1 Std Dev:
6.02 \pm 0.84
Maximum value: 8.64
Minimum value: 4.14

X2list:
Sample Size: 709
Mean: 5.90
Median: 5.83
Mean \pm 1 Std Dev:
5.90 \pm 0.76
Maximum value: 8.80
Minimum value: 3.36

X3list:
Sample Size: 353
Mean: 6.00
Median: 5.94
Mean \pm 1 Std Dev:
6.00 \pm 0.77
Maximum value: 8.20
Minimum value: 3.65

X4list:
Sample Size: 219
Mean: 6.07
Median: 6.10
Mean \pm 1 Std Dev:
6.07 \pm 0.83
Maximum value: 8.33
Minimum value: 3.45

Y1list:
Sample Size: 413
Mean: 6.17
Median: 6.13
Mean \pm 1 Std Dev:
6.17 \pm 0.79
Maximum value: 8.58
Minimum value: 4.01

Y2list:
Sample Size: 400
Mean: 5.90
Median: 5.86
Mean \pm 1 Std Dev:
5.90 \pm 0.76
Maximum value: 8.21
Minimum value: 4.02

Y3list:
Sample Size: 323
Mean: 6.03
Median: 5.97
Mean \pm 1 Std Dev:
6.03 \pm 0.80
Maximum value: 8.58
Minimum value: 3.78

Y4list:
Sample Size: 288
Mean: 6.10
Median: 6.01
Mean \pm 1 Std Dev:
6.10 \pm 0.86
Maximum value: 8.56
Minimum value: 4.29

# 8 Empirical method 2: Explanations

The data generated includes around 6000 starting positions along with white win/draw/loss percentage.
While the first method only took into account wins or losses, the second method tries to use the information about white score[4].
For each fairy piece and each starting file (A-file, B-file, C-file and D-file), we define a variable (e.g. `A1` corresponds to the archbishop starting on the A and H-files).
We define a naive evaluation function as usual. For instance, the evaluation of the starting position

`aefgkfea/.../RNBQKBNR`

is the number `eval(aefgkfea/RNBQKBNR) =`

```
normalization * ( (R1 + N2 + B3 + Q4 + B3 + N2 + R1) -
( 1 - contempt) * (A1 + E2 + F3 + G4 + F3 + E2 + A1))
```

where `normalization` and `contempt` are two other variables with obvious interpretations. Taking all things into account, this defines a function in $4 \times 20 + 2 = 82$ variables called **eval**.
We then consider the sigmoid function

$$S(x) = 50 + 50 \cdot \frac{2}{1+\exp(-0.00368208 \cdot x)}$$

defined for any real number $x$, and we compose it with **eval** in order to have a function **sigmeval** : $\mathbb{R}^{82} \to [0, 1]$.
Our Method 2 consists in minimizing the function **sigmeval** against the 6000 data points (using the $L^1$-norm) modulo the following restrictions:

- We fix `N2 = 300`
- `contempt` is in $[-1, 1]$.
- All other variable are in $[0, 1200]$.

The results are listed in the next section. We make the following remarks:

- The minimized objection function value is around $4$ which means that any setup with a naive evaluation function of $0$ should lead (according to the generated data) to games where the score of white is near $46\% - 54\%$.
  Nevertheless, we remark that, in the generated data, white score has large fluctuations (between $20\%$ and $80\%$) even if the material imbalance is around $0$ centipawns. Even for horizontally symmetrical setup, white score may fall below $30\%$.

[4]White score $=$ White win percentage $+$ (Draw percentage)$/2$.

- If one trusts the sigmoid fit, then a setup with a material imbalance within $100$ centipawns should lead to games where the score of white is (in average) around $45\% - 55\%$.

- We do not know if $4$ is really a global minimum.

- The results have to be taken with a grain of salt. We do not know if the data points generated are of any good quality, or in sufficiently large number (we may have overfitted the data). For instance, the values of the Knight are

  ```
  N1 = 307.8854351675296
  N2 = 300.0
  N3 = 303.74747109415495
  N4 = 296.5143951684805
  ```

  This goes against the intuition that a knight on the rim is less valuable than a knight on the D-file.

- Overall, the results are somewhat consistent with the first method above.

- We remark that the `contempt` variable is slightly negative, which means that black pieces are more important than white pieces. This goes against the usual intuition that white (the first player to move) has a non-negligible advantage. Moreover, one can remark that, over all 6000 setups tested, black score is slightly higher than white score (even in horizontally symmetrical starting position!), which is strange. This may be explain by several factors:
  - Statistical fluctuations in the generated data.
  - The very short time control used.
  - The MCTS-book generation yields starting positions biased towards black.
  - Fairy stockfish has trouble understanding how to play this chess variant.
  - The usual material imbalances in this chess variant puts white in zugzwang.

# 9 Empirical method 2 : Results

```
normalization =
 0.76468623825266
contempt =
 −0.008605831782961692
A1 = 793.7437185216935
A2 = 772.0562579656608
```

A3 = 758.0727813262125
A4 = 768.3326938904349
B1 = 382.19221370431876
B2 = 369.2461715884976
B3 = 366.90499114269977
B4 = 368.63574276080897
C1 = 823.3865206407861
C2 = 805.5544156594655
C3 = 808.1575118379824
C4 = 812.5609112653177
D1 = 545.8636720557945
D2 = 540.2880266373834
D3 = 524.5651680894232
D4 = 537.648601368774
E1 = 266.2844823622911
E2 = 278.0128790015665
E3 = 281.9598525675496
E4 = 276.44900558978657
F1 = 393.0996695763779
F2 = 381.19536768709014
F3 = 367.91325157407573
F4 = 379.30734617792007
G1 = 365.3359368976349
G2 = 377.5506747152516
G3 = 398.20094952108616
G4 = 389.5940989586579
H1 = 235.55791992132689
H2 = 256.8568864898957
H3 = 263.5875949202868
H4 = 250.09364430421206
I1 = 410.57470477145176
I2 = 421.3633740936653
I3 = 422.3818325675987
I4 = 423.10691075500904
J1 = 269.27934035798904
J2 = 276.1727328849483
J3 = 291.6445175148092
J4 = 269.72633306108696
M1 = 431.86402193279866
M2 = 431.53572115786346
M3 = 432.25356887532433
M4 = 432.2057665443132
N1 = 307.8854351675296
N2 = 300.0
N3 = 303.74747109415495
N4 = 296.5143951684805
O1 = 311.8287904059204
O2 = 314.54454918037953
O3 = 300.8017841407945
O4 = 304.04635331015265
Q1 = 896.0756017184152
Q2 = 898.0431654633998
Q3 = 897.5899883004325
Q4 = 895.8760536324891
R1 = 439.4052281777175
R2 = 432.9532148262114
R3 = 436.8651417112539
R4 = 437.31493802272945
U1 = 271.0293235226817
U2 = 275.96651621339737
U3 = 295.4516851698156
U4 = 316.83515338170605
V1 = 434.9709100049079
V2 = 421.7398587120457
V3 = 410.677718567689
V4 = 418.60336845493646
W1 = 285.2408224007906
W2 = 294.69029449956014
W3 = 296.8639779923511
W4 = 296.80273206120273
X1 = 599.4219107443421
X2 = 612.0507277242375
X3 = 616.8376013921578
X4 = 614.7127844720983
Y1 = 676.4746972982648
Y2 = 658.7837857454757
Y3 = 649.0153695968124
Y4 = 662.6557725822203
Minimized Objective Function Value = 4.062751048994891

# 10 Figures

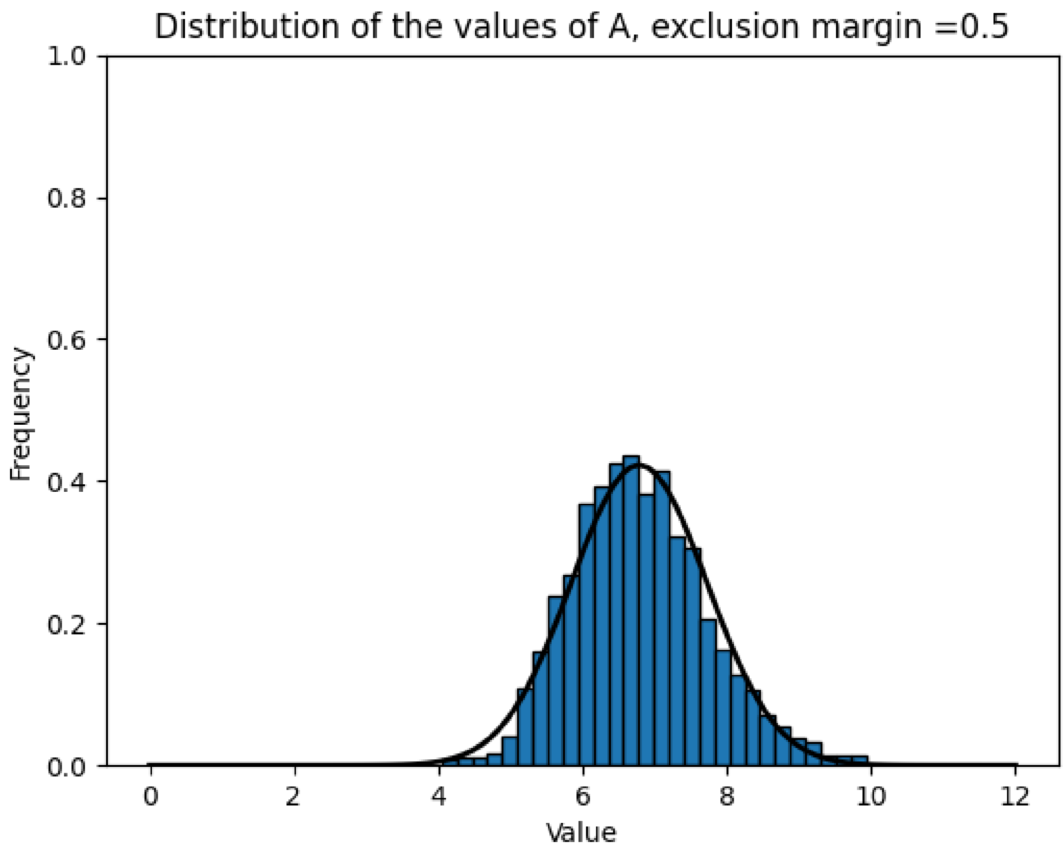


Figure 2: Values of A (Archbishop)

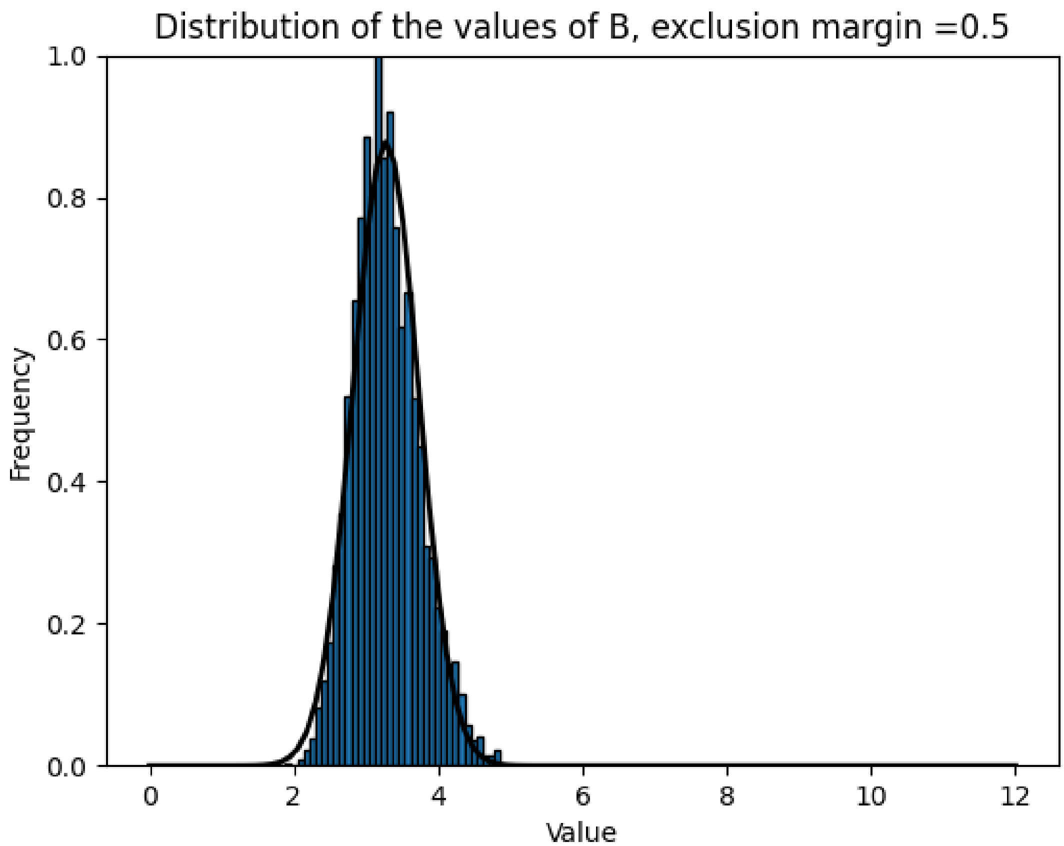


Figure 3: Values of B (Bishop)

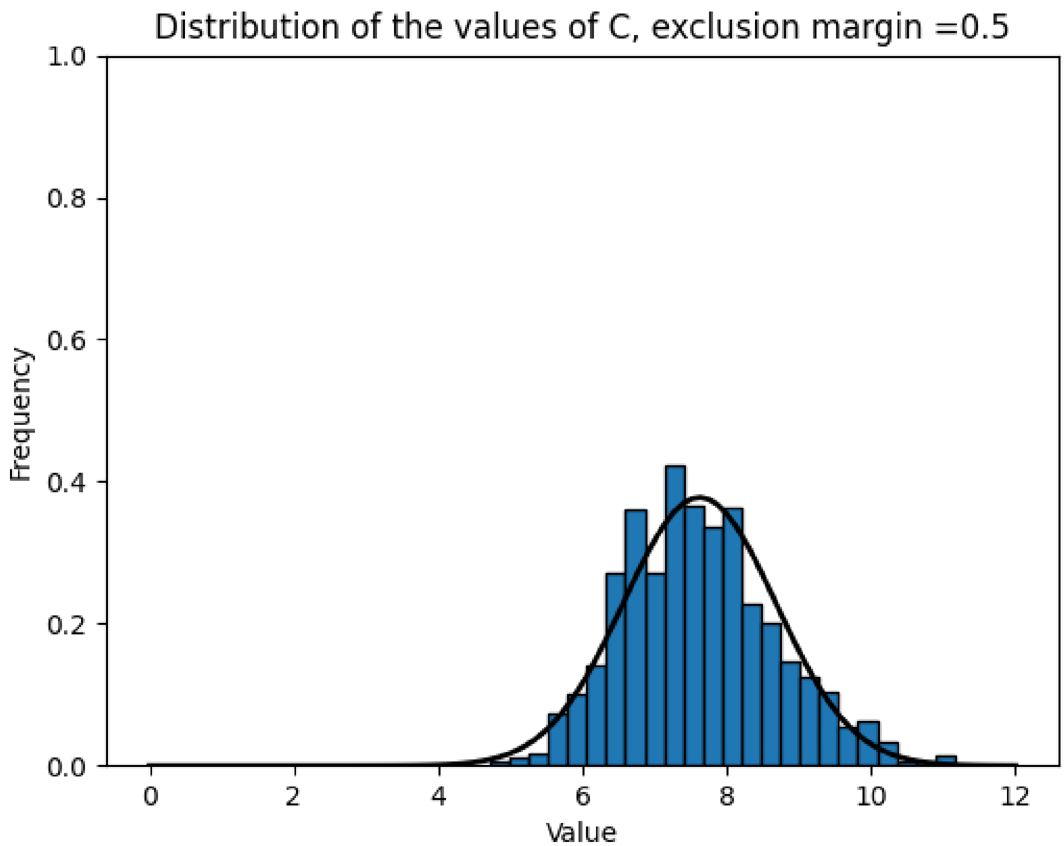


Figure 4: Values of C (Chancellor)

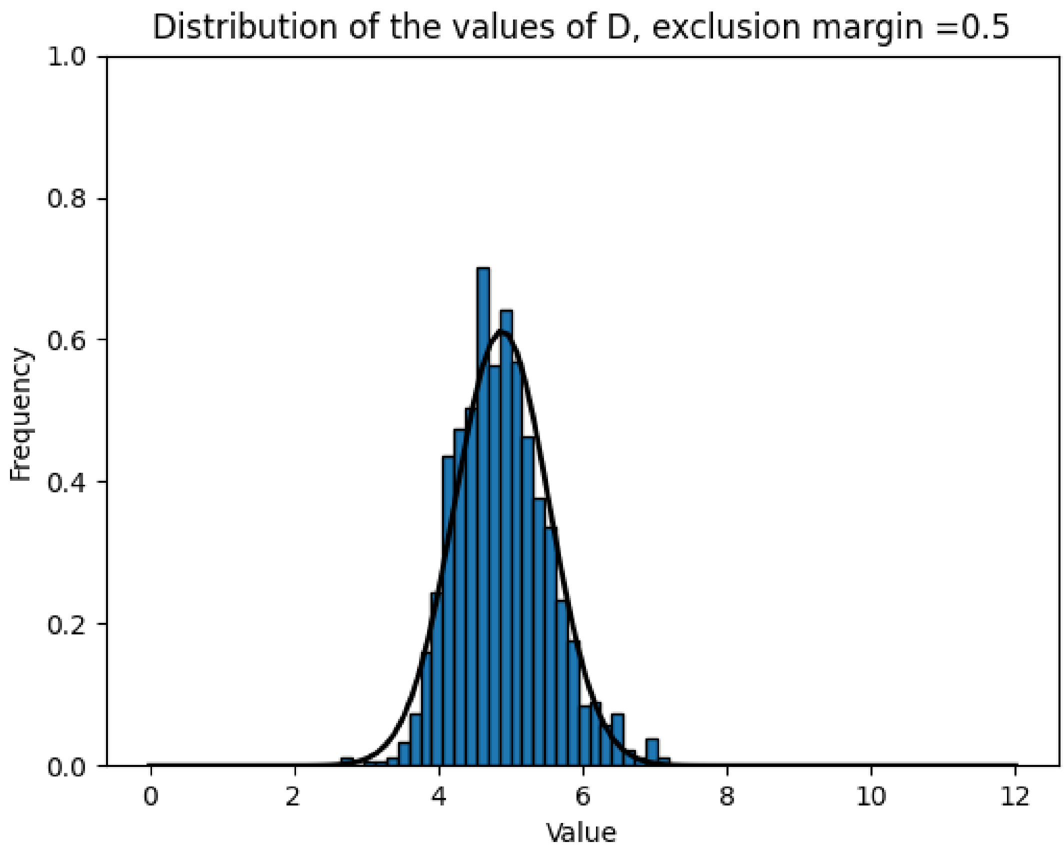


Figure 5: Values of D (DragonHorse)

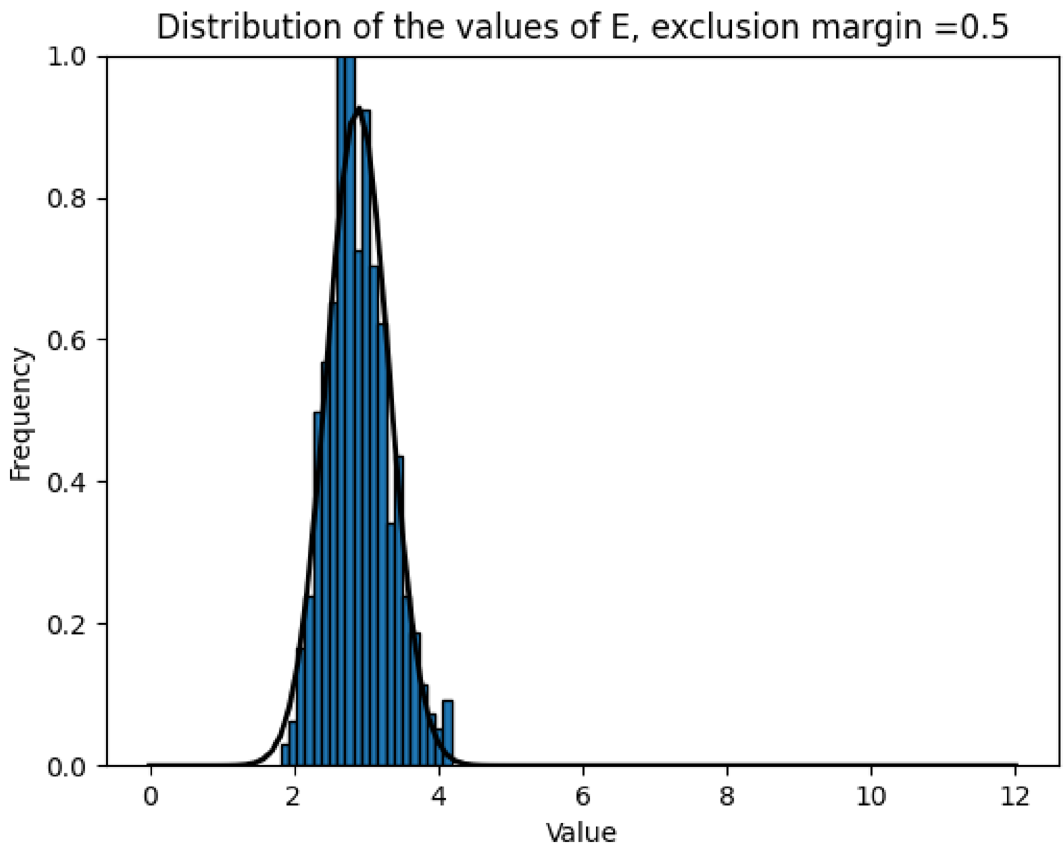


Figure 6: Values of E (Mankni)

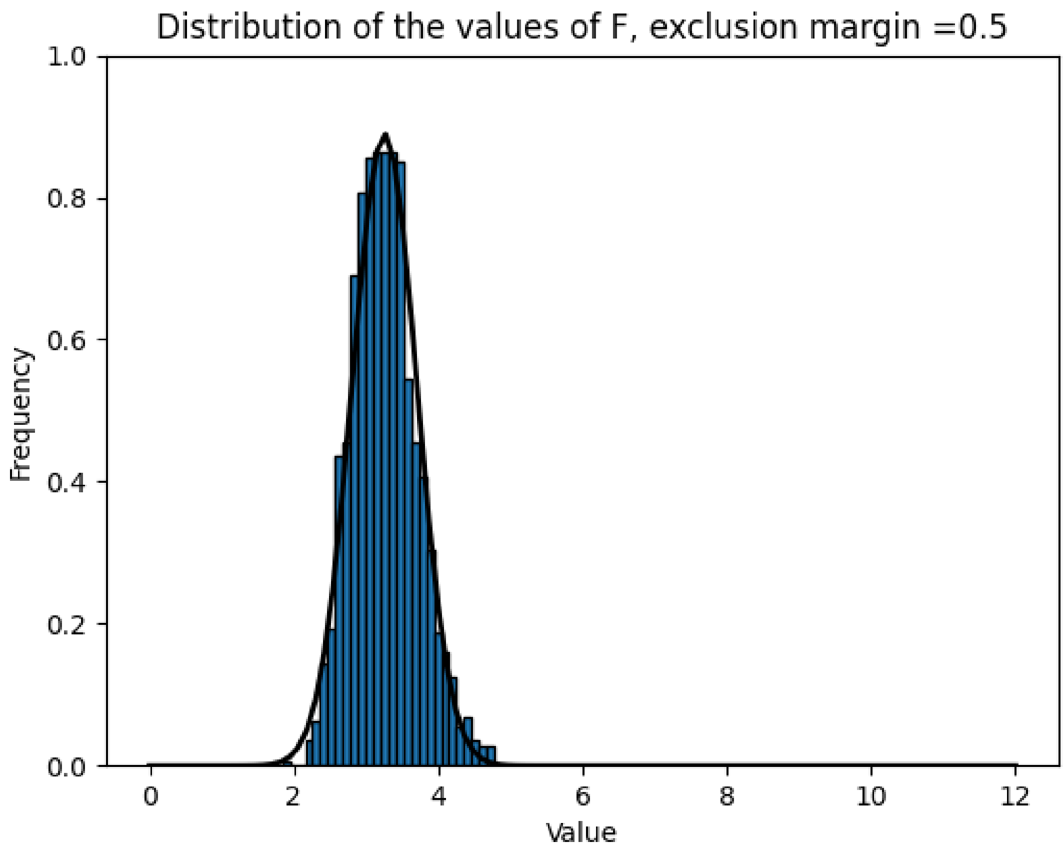


Figure 7: Values of F (Manbis)

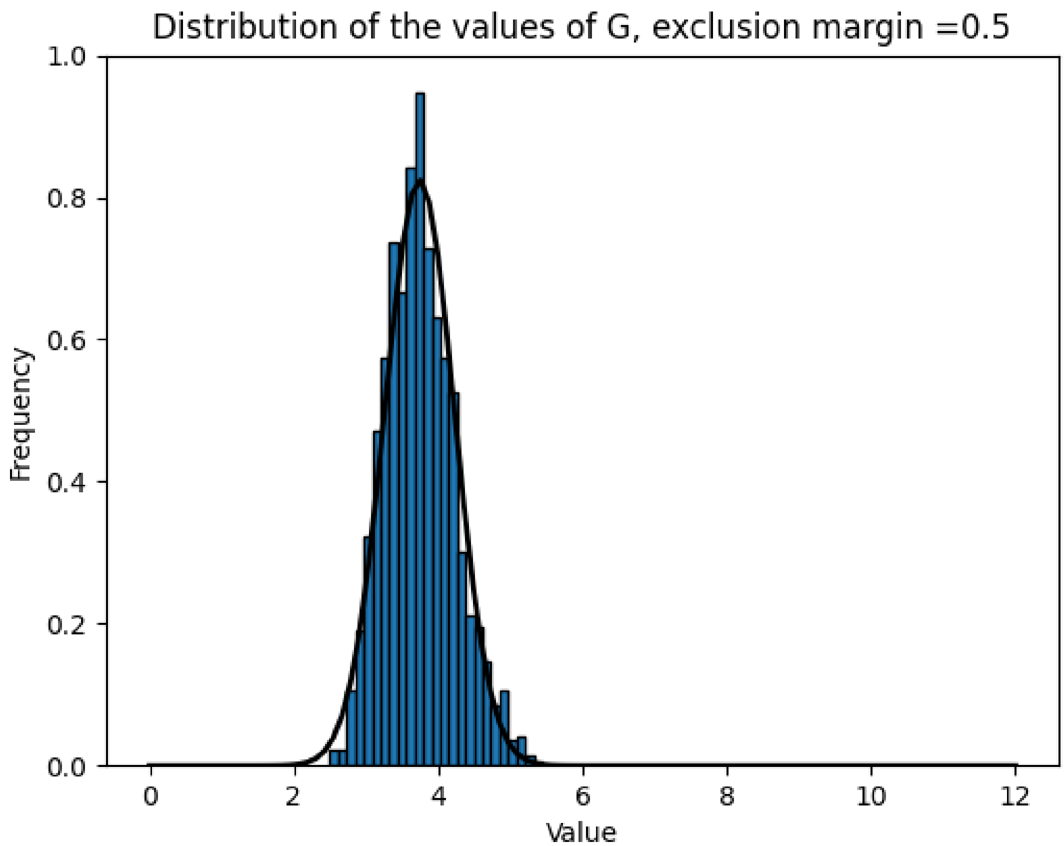


Figure 8: Values of G (Manroo)

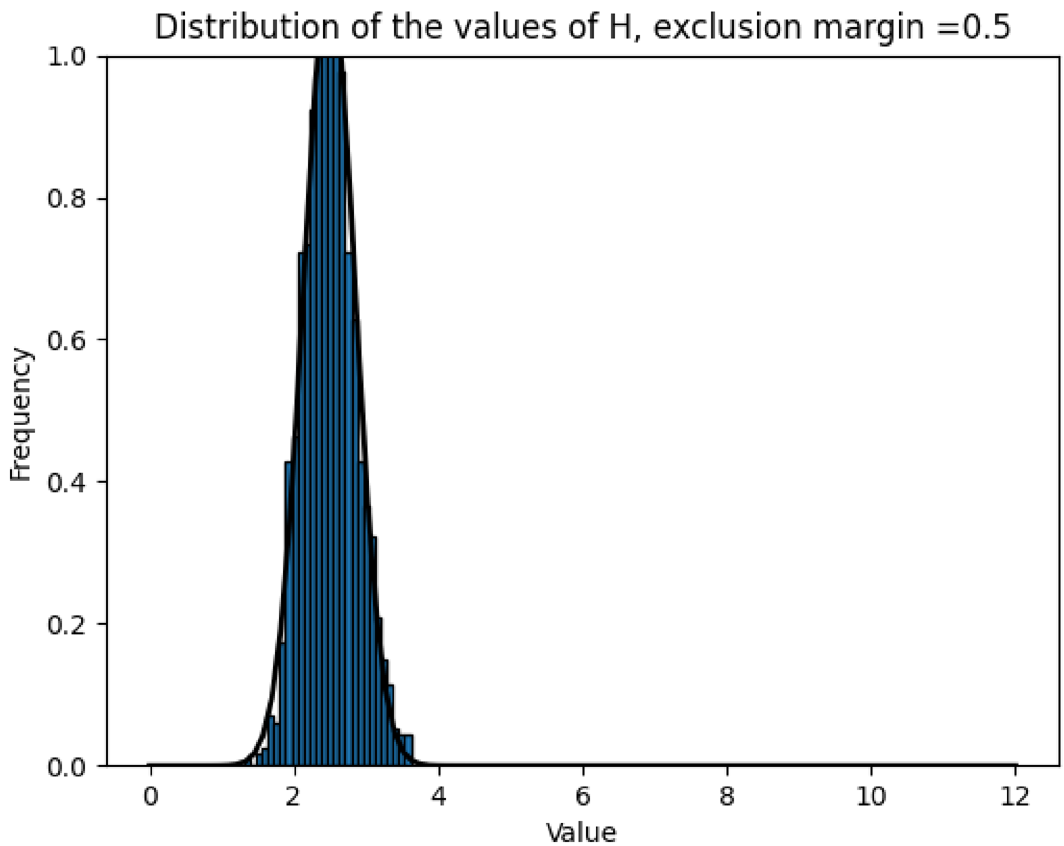


Figure 9: Values of H (Biskni)

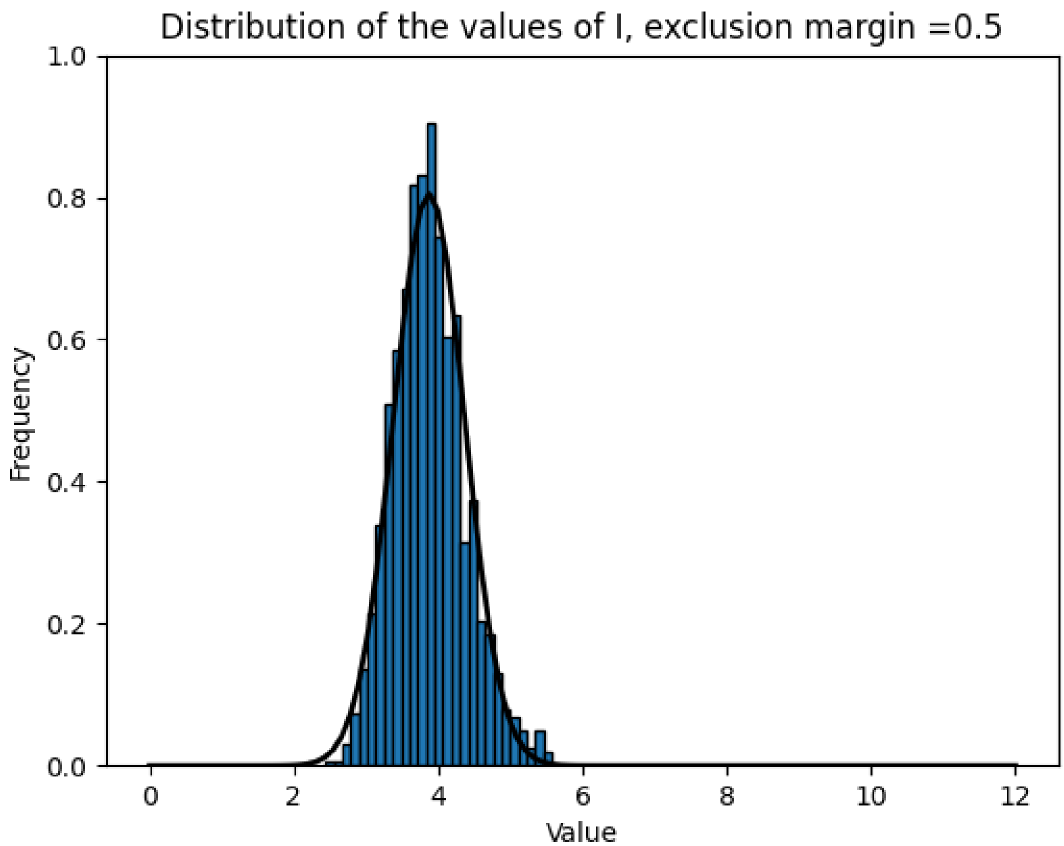


Figure 10: Values of I (Bisroo)

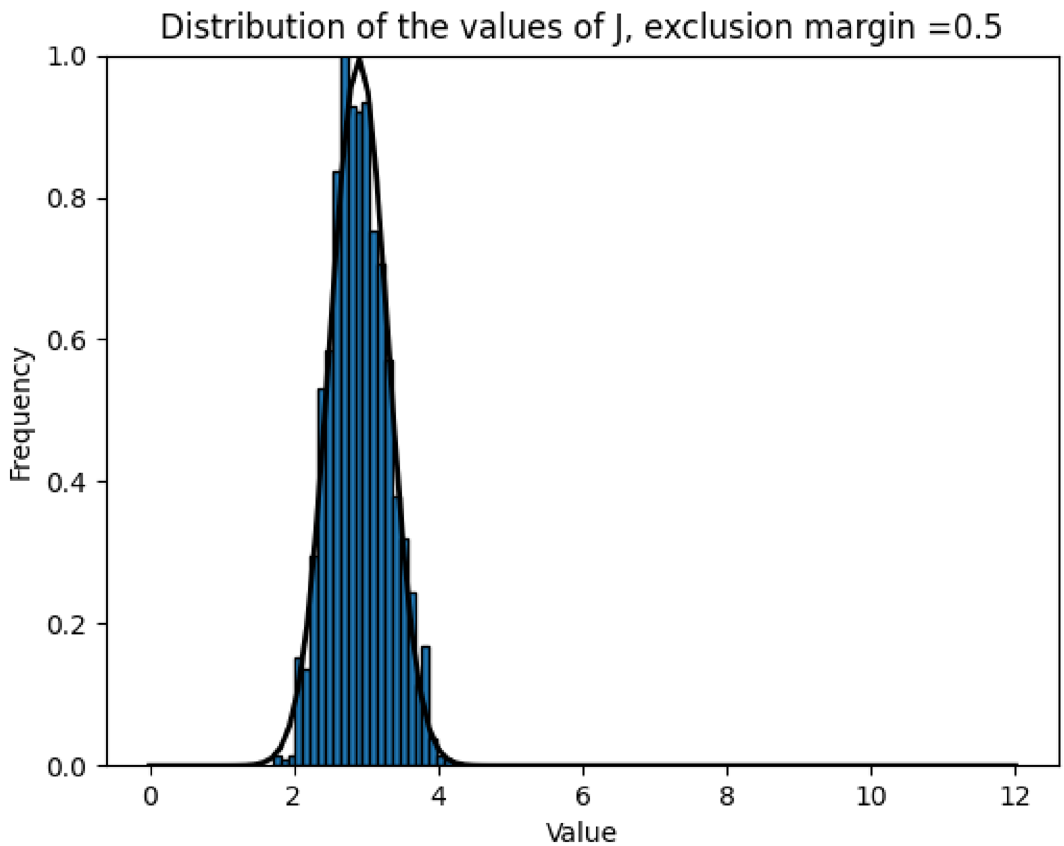


Figure 11: Values of J (Bisman)

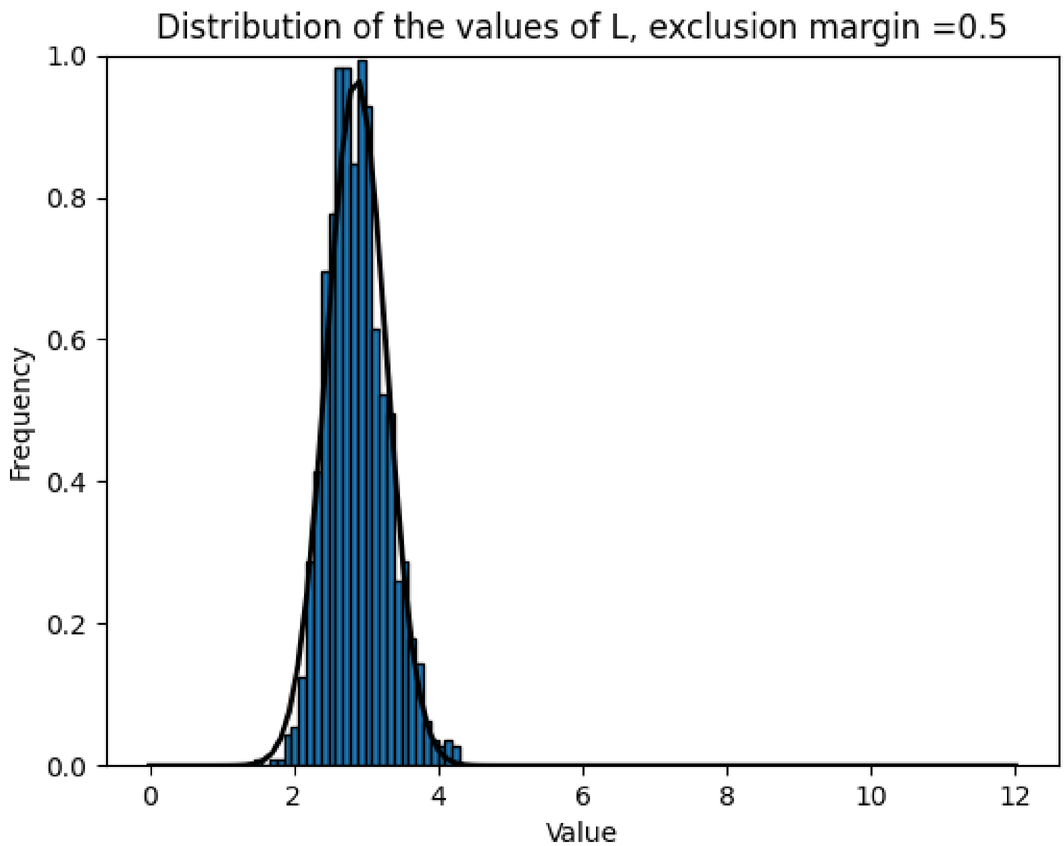


Figure 12: Values of L (also Bisman)

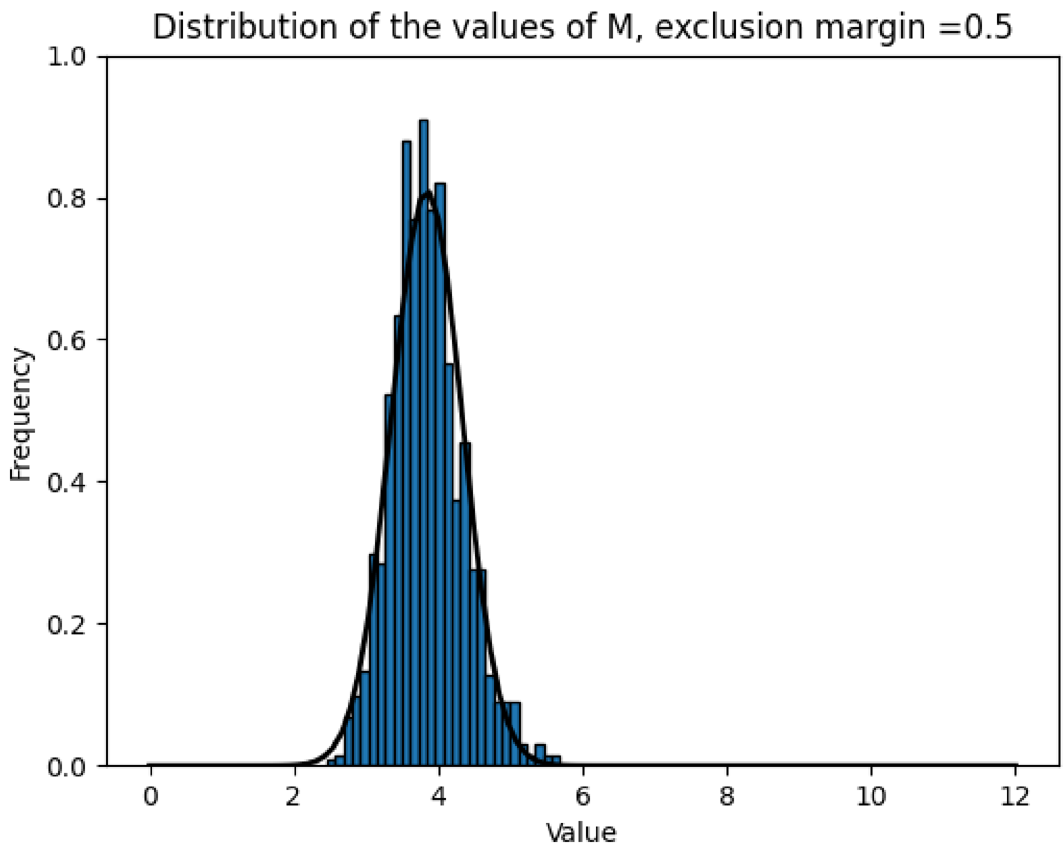


Figure 13: Values of M (Kniroo)

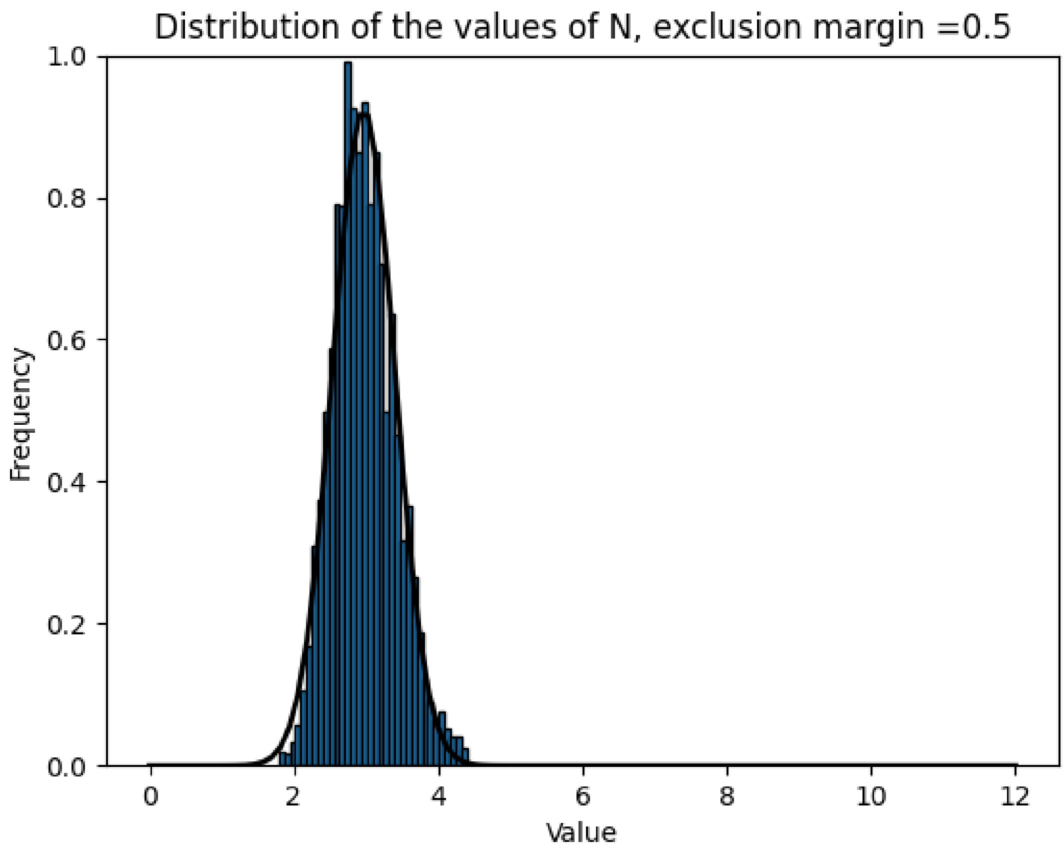


Figure 14: Values of N (Knight)

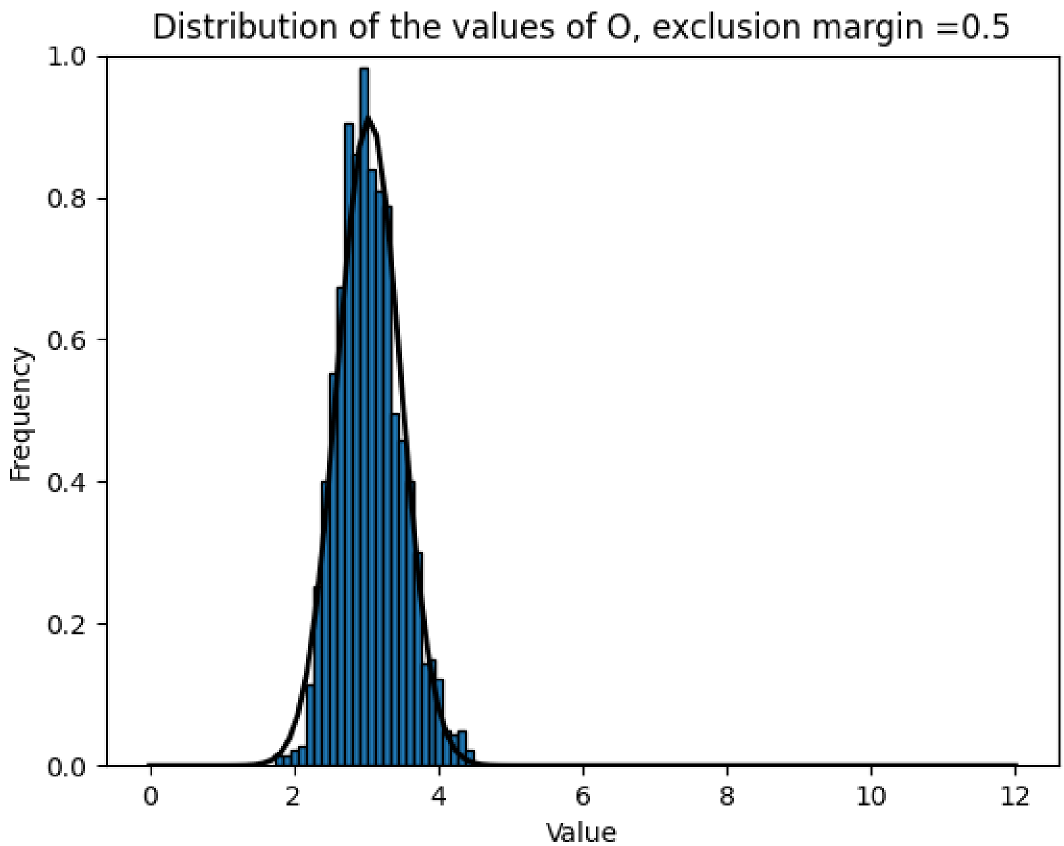


Figure 15: Values of O (Kniman)

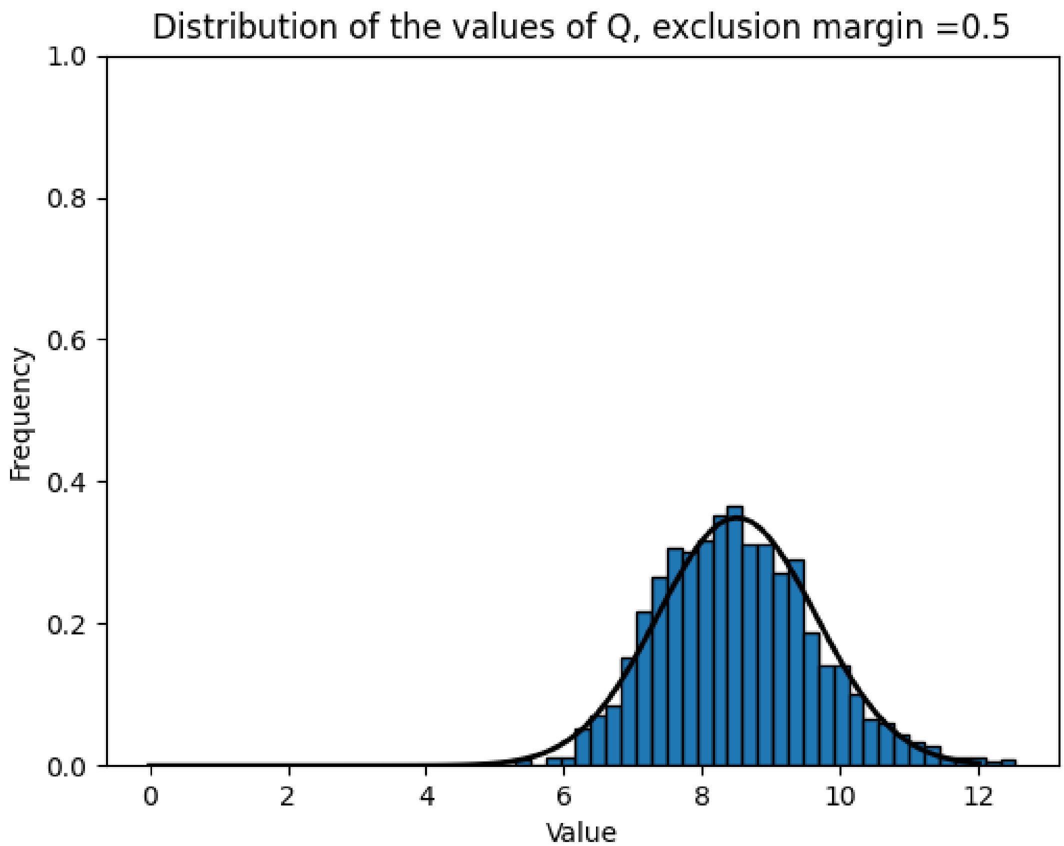


Figure 16: Values of Q (Queen)

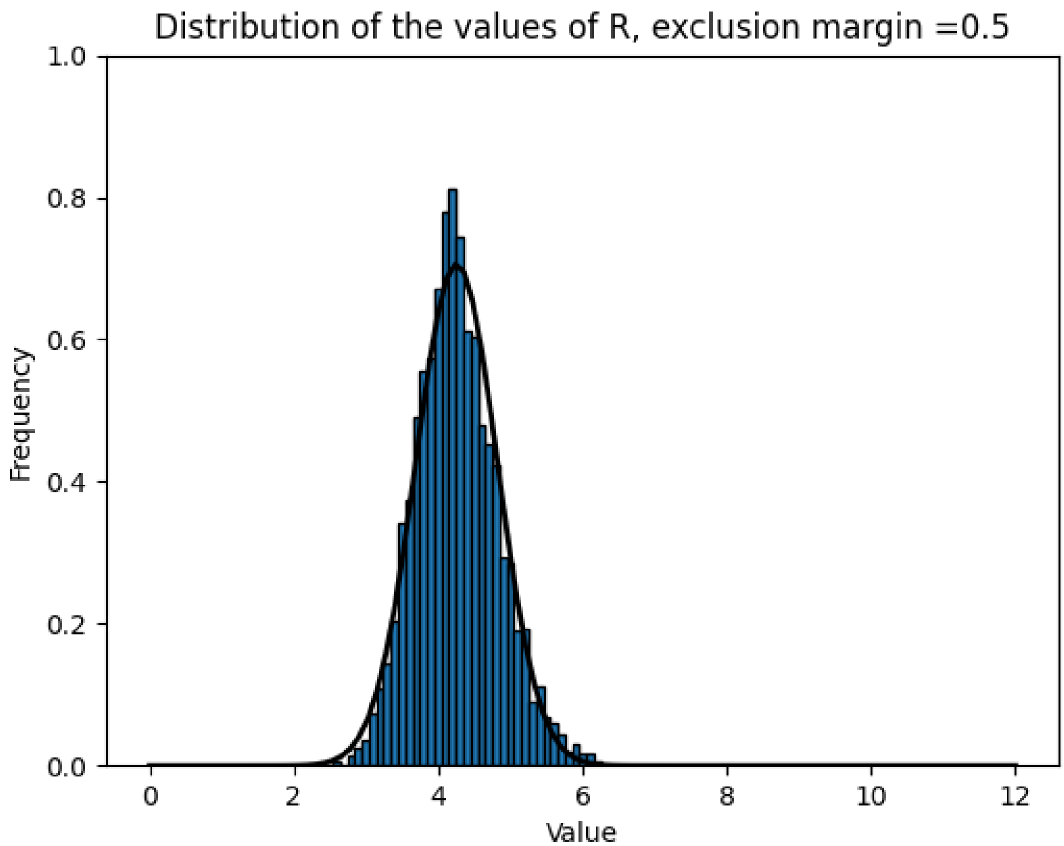


Figure 17: Values of R (Rook)

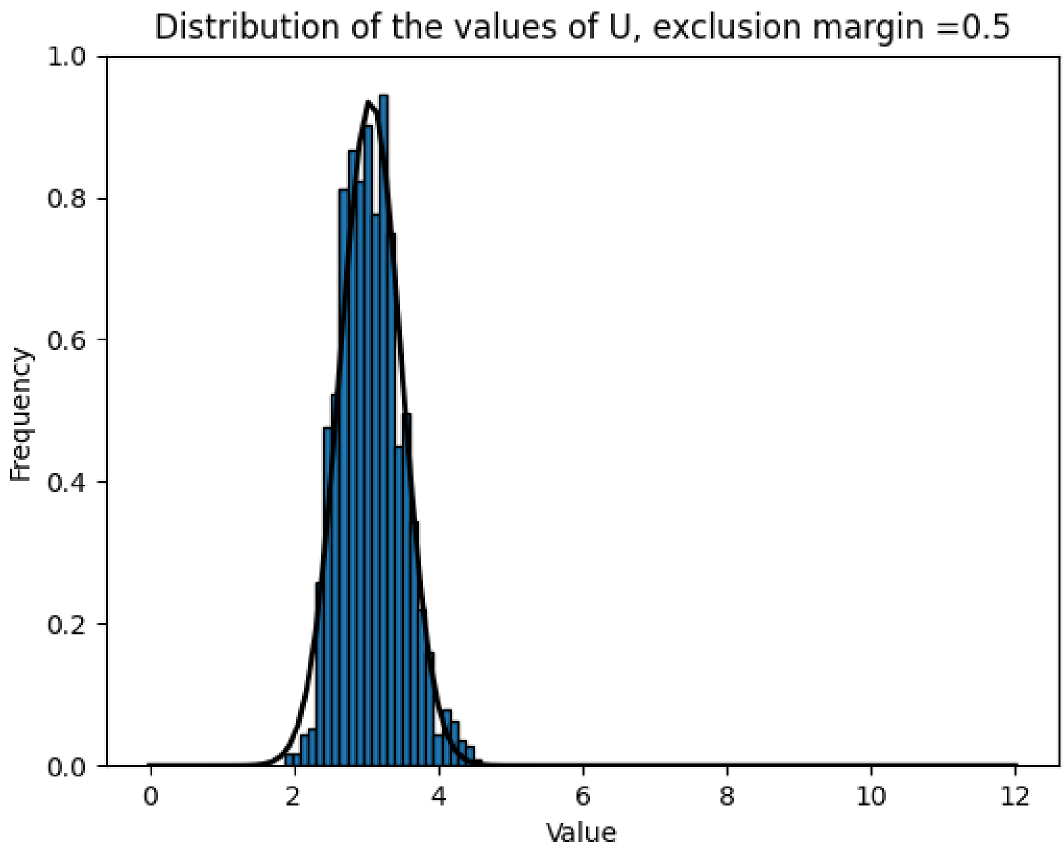


Figure 18: Values of U (Rookni)

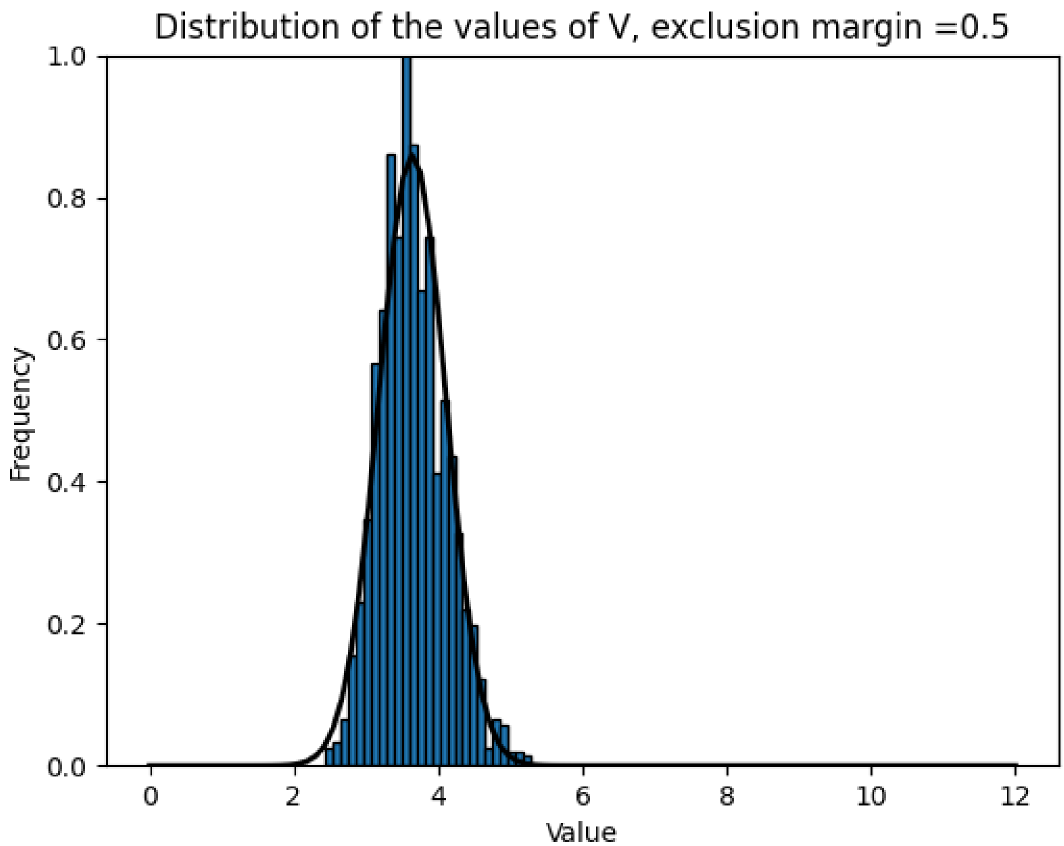


Figure 19: Values of V (Roobis)

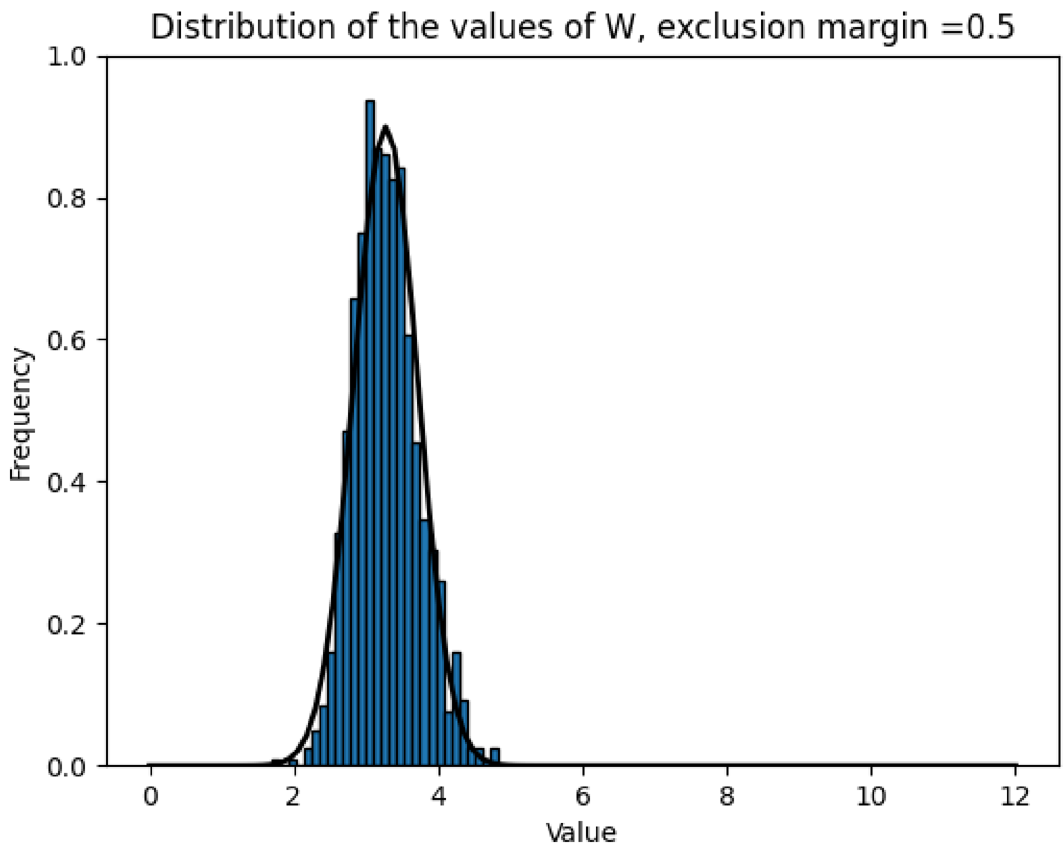


Figure 20: Values of W (Rooman)

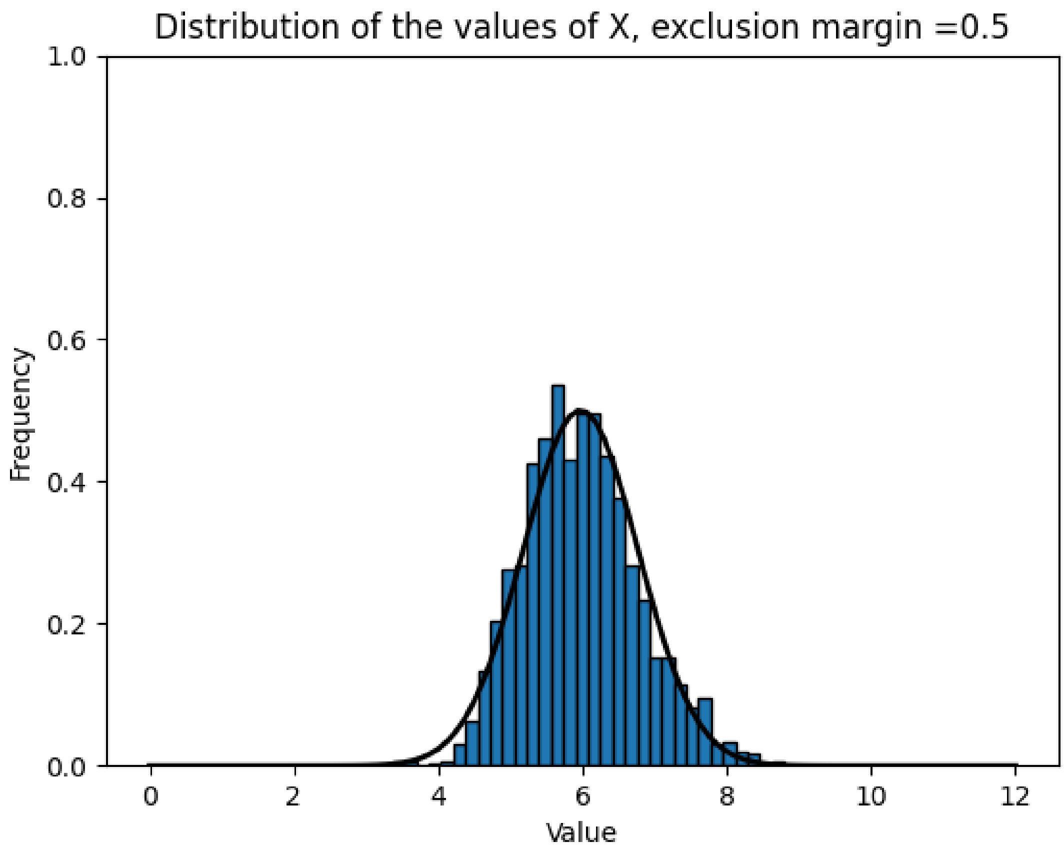


Figure 21: Values of X (Bers)

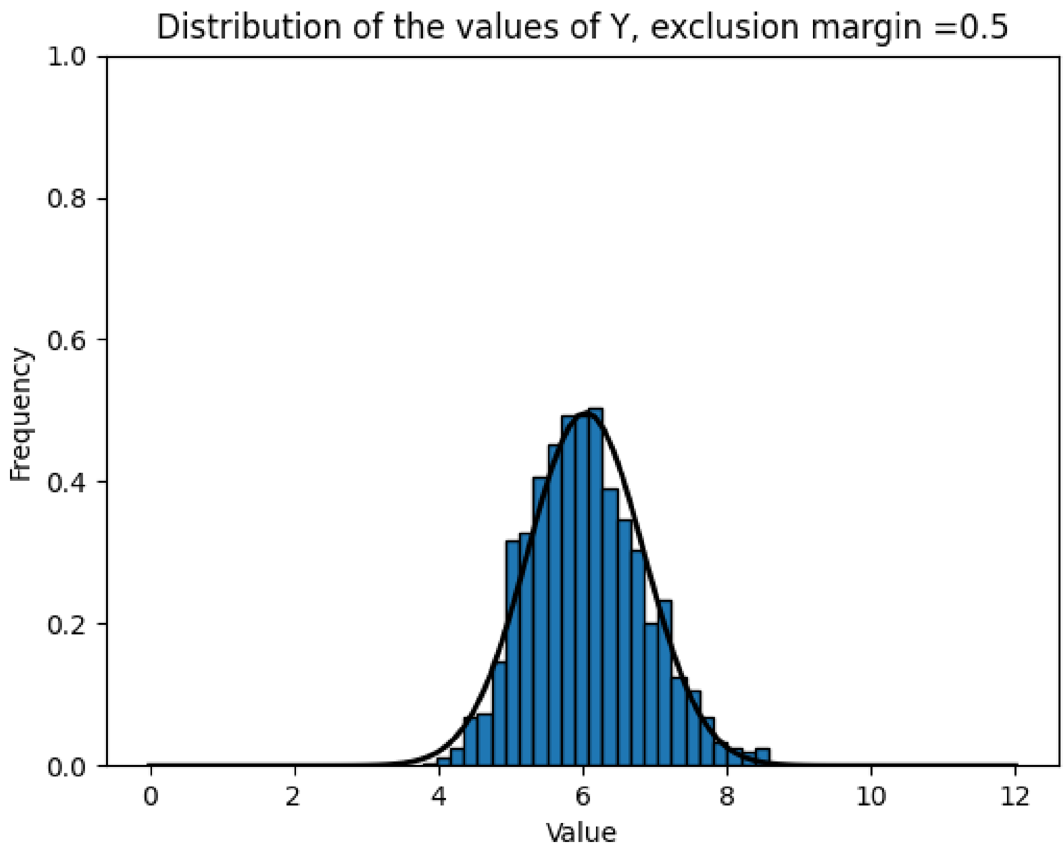


Figure 22: Values of Y (Centaur)

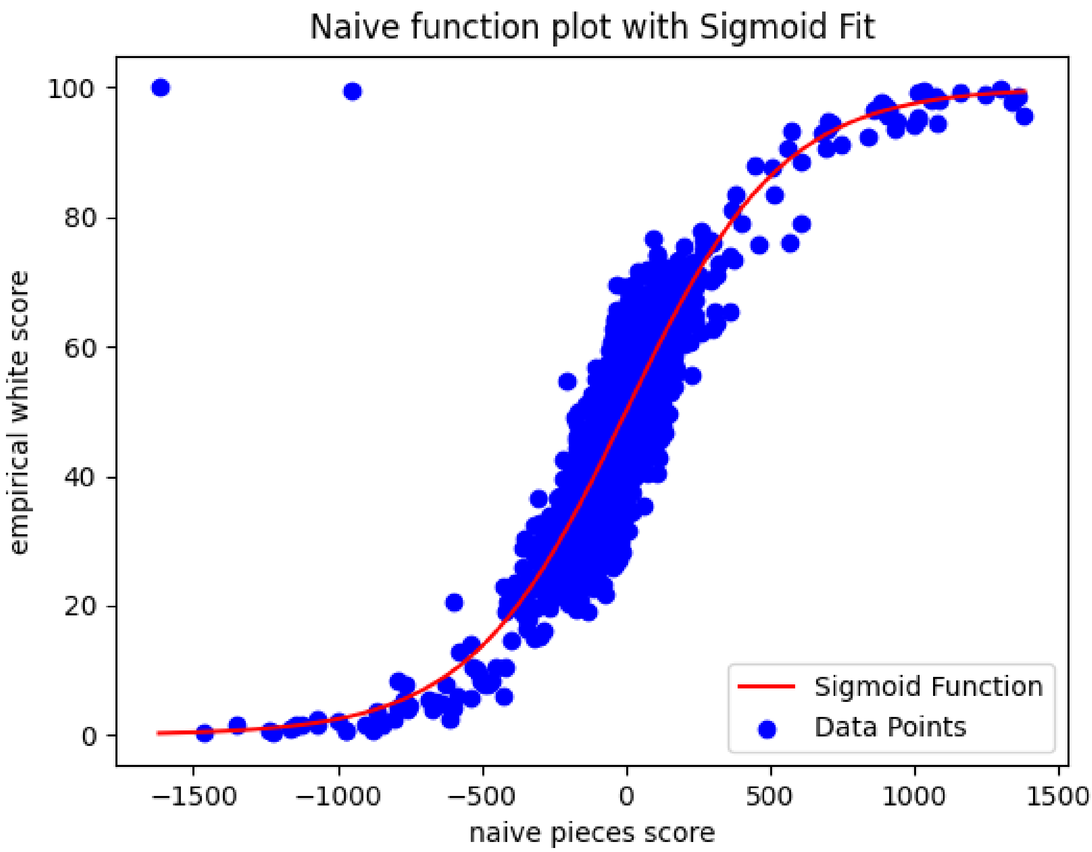


Figure 23: naive pieces score vs empirical white score, with sigmoid fit