Preference Levels as Performance Predictors

GIUSSANI RICCARDO

Summary

In situations of conflict each competitor has:

- A pool of characteristics, common to all the competitors. For each characteristic, each competitor has its own proficiency, that can be expressed through a numerical value
- The performance (the outcome of the conflict), also expressible numerically

We assume that one's proficiency cannot be improved up to infinity and all proficiencies must be committed one to each other through trade-offs.

Each competitor can be considered as a 'basket' (in marketing terminology) regarding a trade-off and therefore many levels of preference, that take into account one trade-off only, can be computed, giving many ranks of competitors.

This is the problem I studied:

determine how the goodness in a trade-off (preference level) is significant in determining the performance

'Super Mario Party' for Nintendo Switch is the instance of the problem I took as an example.

The characteristics are the characters' unique dice statistics (explained in Introduction), while the performance is a function of the number of games won by that character in a sample of 60 games.

In order to solve the problem, I chose the following strategy:

- Compute characters' levels of preference for each trade-off
- Use the preferences ranks as independent variables and the performance as dependent variable in a linear regression model
- Exploit the t test to drop non-significant trade-offs through backward elimination

The resulting regression model will show the truly significant trade-offs. Coefficients shall give a numerical quantifier of significance.

Introduction

Mario party is a game in which 4 players move on a game board where the landing squares determine a gain or a loss of coins or a special event (changings in the game board routes, acquisition of item etc.). At the end of every turn the players participate into a minigame that gives away a certain amount of coins. The final winner is determined by whoever has bought the highest number of stars, that can be found in squares of the game board (often changing position).

In the most recent version for Nintendo Switch, each character (the pawn of the player) has its own unique die, that the player can use.

Resources

Each die has some features, that from now on shall be called ‘resources’, that are:

- The expected movement for each turn, that is the mean of values on dice faces that allow movement (‘**movement mean’**)
- The expected gained/lost coins for each turn, that is the mean of values on dice faces that give/get coins (‘**coins mean’**)
- The variances associated to the two means, that are ‘**movement variance’** and ‘**coins variance’**
- The ‘options’ of movement, that is the number of non-zero and different one from each other movement faces. This one was introduced because of the special event squares: players can have an advantage in moving of a specific number of squares (not necessarily of the highest number possible) (‘**variety**’)

Among those resources:

- ‘Movement mean’, ‘coins mean’ and ‘variety’ are favourable to the player. Therefore, the player would like these to be maximized
- ‘Movement variance’ and ‘coins variance’, that indicate the risk associated to each expected value, can be seen by players as favourable (risk seekers) or unfavourable (risk adverse). The former would like variance to be maximized, the latter to be minimized

Ideally any die has these objective functions:

- Maximize the movement mean
- Maximize the coins mean
- Maximize the variety
- Maximize/minimize the movement variance
- Maximize/minimize the coins variance

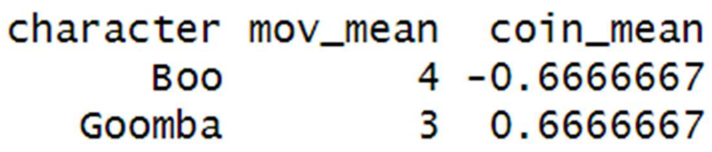
As long as the game is meant to be balanced, it is impossible for a die to maximize/minimize all resources to infinity, but there must occur some trade-offs.

Trade-off

It subsists a trade-off when, given two quantities to be maximized, at a certain point it is no more possible to augment the value of one without diminishing the value of the other.

Example

Consider the two resources ‘movement mean’ and ‘coins mean’ and consider the standard die (1,2,3,4,5,6) with ‘movement mean’ = 3.5 and ‘coins mean’ = 0 along with two dice from the game:



```
character mov_mean  coin_mean
      Boo        4 -0.6666667
   Goomba        3  0.6666667
```

Regarding the standard die:

- *‘Boo’ augments the ‘movement mean’ by 0.5 at the cost of 0.666 ‘coins mean’*
- *‘Goomba’ augments the ‘coins mean’ by 0.666 at the cost of 0.5 ‘movement mean*

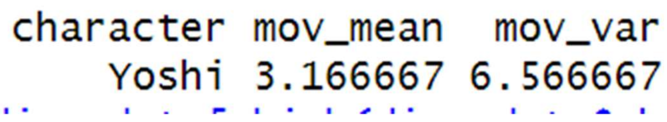
Preference

Regarding two resources a die can be seen as a market basket and consequently can be compared to all the others, giving the possibility to determine better and worse baskets (dice).

Example

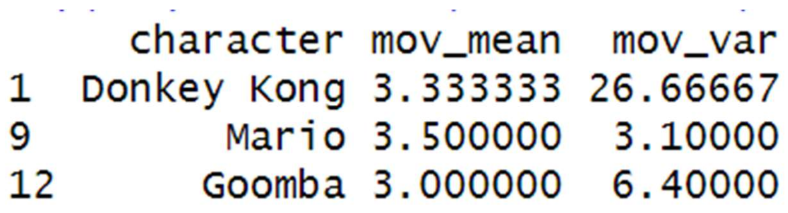
Consider the resources 'movement mean' and 'movement variance' from a risk adverse point of view.

The benchmark shall be 'Yoshi':



```
character mov_mean  mov_var
    Yoshi 3.166667 6.566667
```

'Yoshi' is then compared to three other dice available, specifically considering the trade-off proposed:



```
     character mov_mean  mov_var
1  Donkey Kong 3.333333 26.66667
9        Mario 3.500000  3.10000
12      Goomba 3.000000  6.40000
```

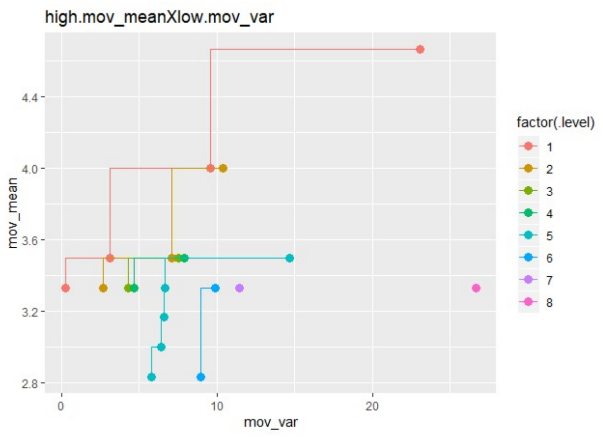
'Goomba' diminishes variance by 0.166 at the cost of a diminishment of mean by 0.1666: the trade seems equal as 'Goomba' is uncapable to get a resource better without getting the other worse.

'Mario' augments the mean by 0.333 and diminishes variance by 3.466. If we could exchange our 'Yoshi' for 'Mario', we would do that immediately.

'Donkey Kong' augments the mean by 0.333, but this come at the cost of an enormous rise of variance. This die is garbage.

This is the conclusion: 'Mario' is better than 'Yoshi', that is equal to 'Goomba', that is better than 'Donkey Kong'.

By using this method on all dice, it is possible to identify the levels of preference that allow us to say that all dice on the same level are 'good the same' and are better than dice in higher levels.



```
"Bowser 1"       "Wario 1"      "Daisy 1"        "Mario 1"
"Boo 2"          "Waluigi 2"    "Shy Guy 2"      "Dry Bones 3"
"Peach 3"        "Luigi 4"      "Monty Mole 4"   "Diddy Kong 5"
"Pom Pom 5"      "Goomba 5"     "Yoshi 5"        "Hammer Bro 5"
"Bowser Jr. 6"   "Rosalina 6"   "Koopa 7"        "Donkey Kong 8"
```

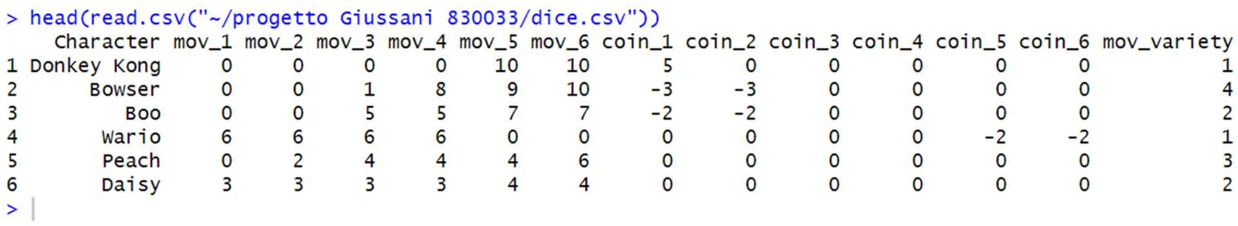
Goal

Taken a sufficiently large sample of games played, winning frequencies of each character can give us a rank of which dice are better and which are worse; this comes by the assumption that, despite the high unpredictability of a single game, there must be dice that have an higher winning chance due to their being more proficient on specific trade-offs.

Given the levels of preference on all trade-offs, the goal is to determine how the goodness in a trade-off (preference level) is significant in determining the performance.

Data Acquisition

dice.csv



```
> head(read.csv("~/progetto Giussani 830033/dice.csv"))
    Character mov_1 mov_2 mov_3 mov_4 mov_5 mov_6 coin_1 coin_2 coin_3 coin_4 coin_5 coin_6 mov_variety
1 Donkey Kong     0     0     0     0    10    10      5      0      0      0      0      0           1
2      Bowser     0     0     1     8     9    10     -3     -3      0      0      0      0           4
3         Boo     0     0     5     5     7     7     -2     -2      0      0      0      0           2
4       Wario     6     6     6     6     0     0      0      0      0      0     -2     -2           1
5       Peach     0     2     4     4     4     6      0      0      0      0      0      0           3
6       Daisy     3     3     3     3     4     4      0      0      0      0      0      0           2
> |
```

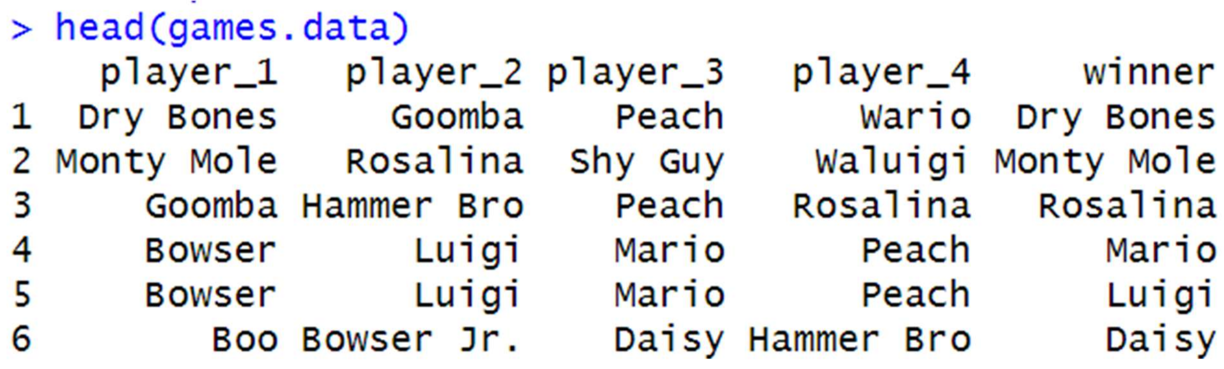
In game there are 20 dice.

All dice have 6 faces. Each face shows or the number of squares the character can move on or an amount of coins that gains/loses, without moving.

Variety is the number of nonzero different numbers in movement faces.

From this dataset ‘resources’ are computed.

gamesResults.csv



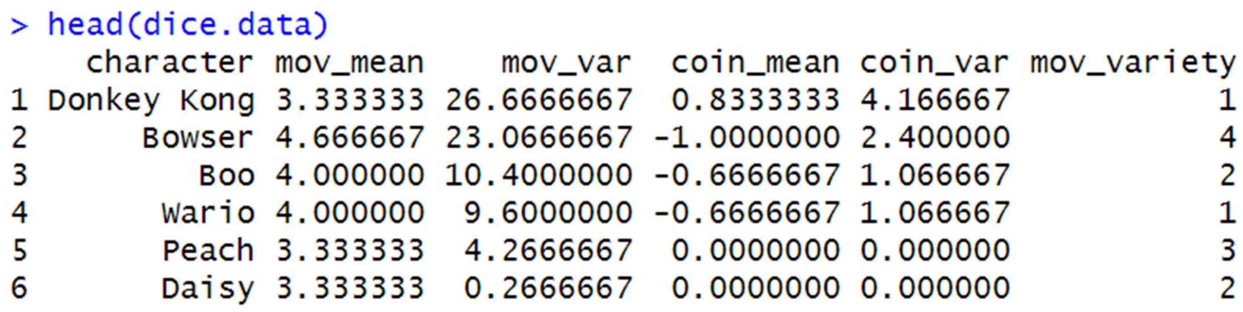
```
> head(games.data)
    player_1    player_2 player_3    player_4     winner
1  Dry Bones      Goomba    Peach       Wario  Dry Bones
2 Monty Mole    Rosalina  Shy Guy     Waluigi Monty Mole
3     Goomba  Hammer Bro    Peach    Rosalina   Rosalina
4     Bowser       Luigi    Mario       Peach      Mario
5     Bowser       Luigi    Mario       Peach      Luigi
6        Boo  Bowser Jr.    Daisy  Hammer Bro      Daisy
```

I took results of 60 ‘parties’ from gameplays on Twitch and YouTube. I reported the 4 characters used and the winning character.

Winning frequency is used as the numerical value that identifies the strength of the die.

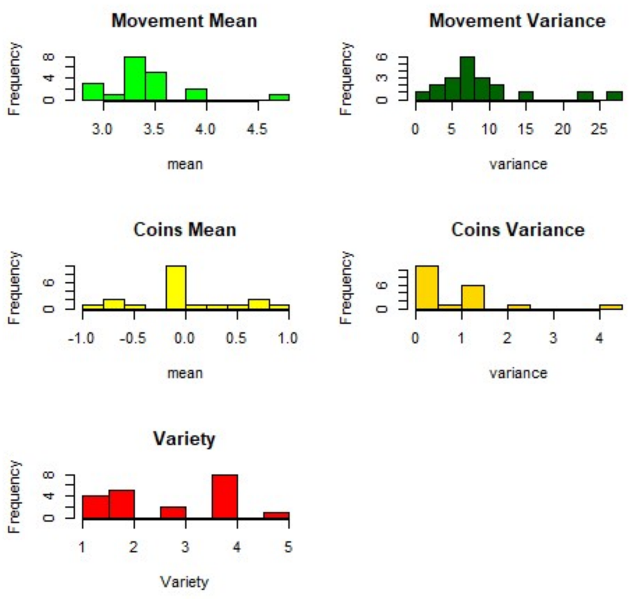
Exploratory Analysis

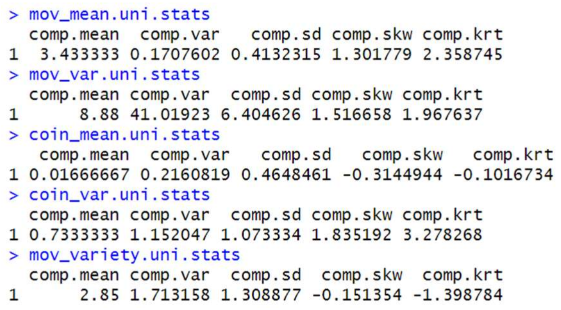
dice.data



```
> head(dice.data)
    character mov_mean    mov_var  coin_mean coin_var mov_variety
1 Donkey Kong 3.333333 26.6666667  0.8333333 4.166667           1
2      Bowser 4.666667 23.0666667 -1.0000000 2.400000           4
3         Boo 4.000000 10.4000000 -0.6666667 1.066667           2
4       Wario 4.000000  9.6000000 -0.6666667 1.066667           1
5       Peach 3.333333  4.2666667  0.0000000 0.000000           3
6       Daisy 3.333333  0.2666667  0.0000000 0.000000           2
```

The dataset of resources.





```
> mov_mean.uni.stats
  comp.mean  comp.var   comp.sd comp.skw comp.krt
1  3.433333 0.1707602 0.4132315 1.301779 2.358745
> mov_var.uni.stats
  comp.mean comp.var  comp.sd comp.skw comp.krt
1      8.88 41.01923 6.404626 1.516658 1.967637
> coin_mean.uni.stats
   comp.mean  comp.var   comp.sd   comp.skw   comp.krt
1 0.01666667 0.2160819 0.4648461 -0.3144944 -0.1016734
> coin_var.uni.stats
  comp.mean comp.var  comp.sd comp.skw comp.krt
1 0.7333333 1.152047 1.073334 1.835192 3.278268
> mov_variety.uni.stats
  comp.mean comp.var  comp.sd  comp.skw  comp.krt
1      2.85 1.713158 1.308877 -0.151354 -1.398784
```

Movement Mean

The mean of all movement means is slightly lower than the one of the standard die (1,2,3,4,5,6). This suggests that this resource is usually traded to lift some other resource.

Standard deviation is low, as there are few dice with movement mean higher than 3.5 or lower than 3.33.

Skewness identifies a strong standard deviation direction to the right, stronger than the one of a normal distribution.

Kurtosis is minor than the one of a normal distribution, therefore this distribution has thinner tails.

Movement Variance

The average variance is higher than the standard die by a lot, demonstrating a general risk-seeking tendency.

Nevertheless, there is a strong standard deviation from that value: the ‘excellent’ cases can be noticed at the right side of the histogram. Regardless, most of the dice are near the average. Skewness and Kurtosis values corroborate this consideration.

Coins Mean

The mean of coins mean is almost zero, and the expected coins per turn is never higher than +1 or lower than -1.

Skewness is negative; therefore, coins drop is often positive.

Kurtosis value is also negative despite the strong peak in the middle. This is due to two local maxima in intervals [-1, -0.5] and [+0.5, +1].

Coins Variance

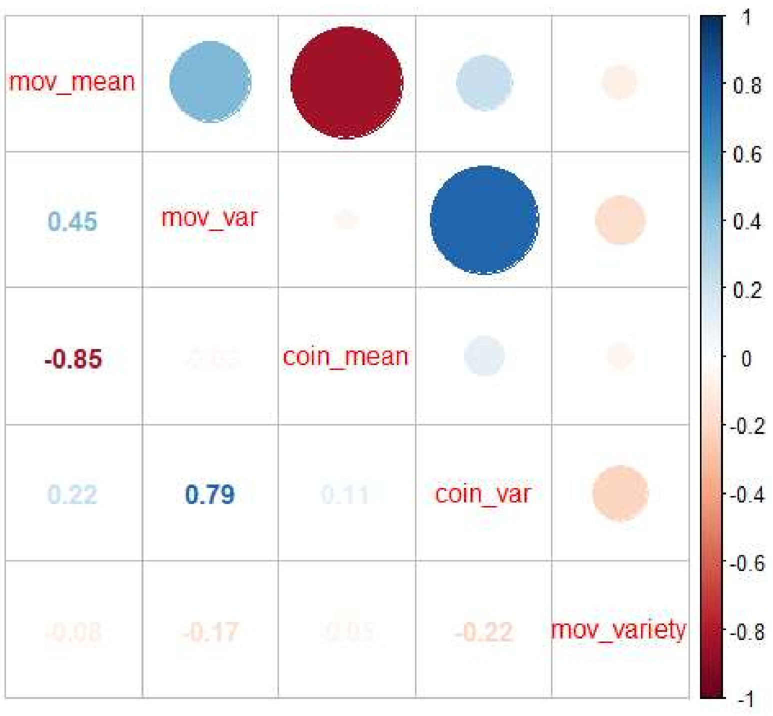
Coins variance provide similar features to movement variance: standard deviation is high because of few ‘excellent cases’, is more right-oriented than a semi-normal and tails are thinner than a normal’s.

Variety

Variety is always lower than the standard’s (6). Indeed, the mean is much lower than 6 and standard deviation is also low.

Kurtosis shows that the distribution is like a uniform one (kurtosis = -6/5). This suggests that this value is uniformly distributed among dice, with no peculiar correlation to any other resource.

Correlations



As previously supposed, variety is not in any relevant correlation.

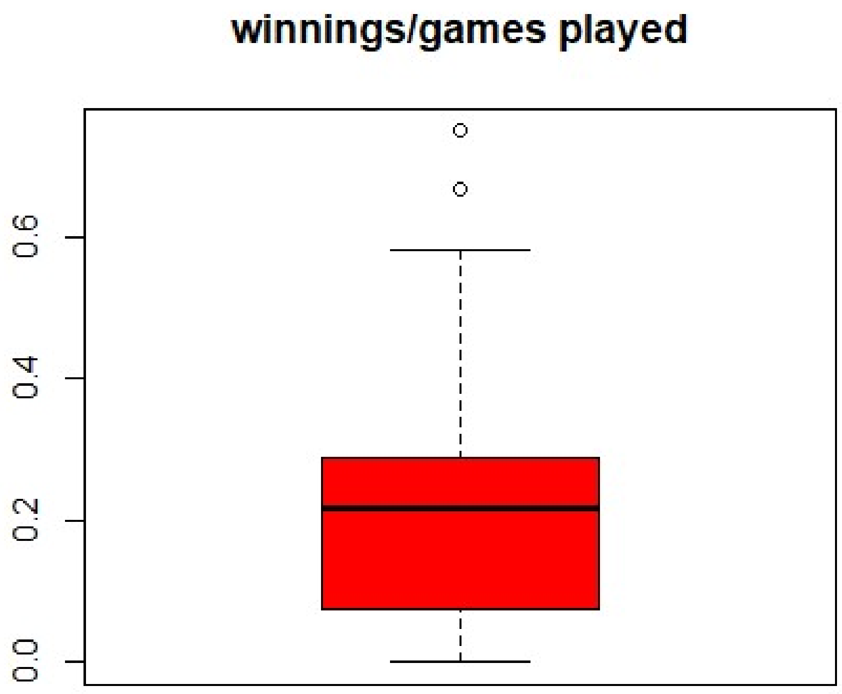
We can see two strong correlations:

- A negative one between movement and coins mean: in the game these two desirable goods are also opposed one to each other; therefore, the player willing to gain coins has to sacrifice some movement, and vice versa.

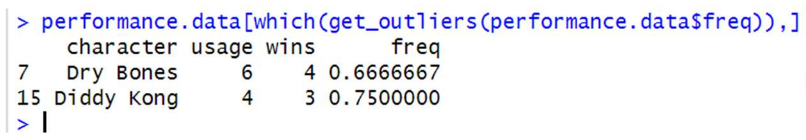
- A positive one between the two variances: regarding this, we can assume that generally dice are totally risky or completely safe.

games.data

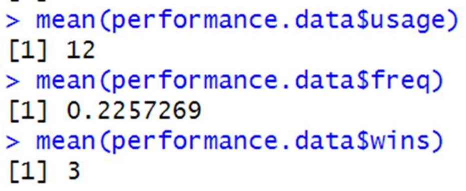
For each of the 20 characters, the winning frequency is calculated.



The boxplot shows the presence of two outliers:



```
> performance.data[which(get_outliers(performance.data$freq)),]
   character usage wins      freq
7   Dry Bones     6    4 0.6666667
15 Diddy Kong     4    3 0.7500000
> |
```



```
> mean(performance.data$usage)
[1] 12
> mean(performance.data$freq)
[1] 0.2257269
> mean(performance.data$wins)
[1] 3
```

The outlying characters are those that have played in a number of games that is almost half of the mean of games played by all characters (usage), but regardless have won an average number of games.

Those two are a little bit ‘penalized’, as their frequency is capped to the nearest whisker limit.

levels.data

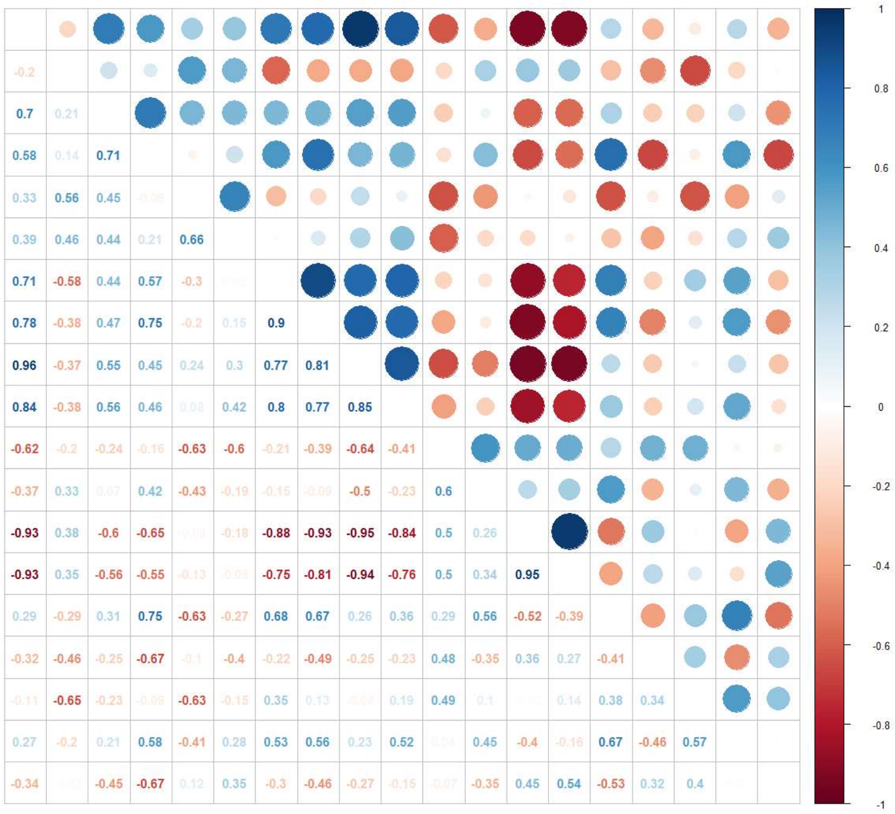
rPref library was then used to get preference levels of all the possible trade-offs (all the 2 elements combinations of different resources, without forgetting the risk-seeking / risk-aversion matter).

Since every trade-off has a different number of maximum levels, I normalized each levels vector on a [0,1] interval, so that:

- 0: the die is on the best level
- 1: the die is on the worst level

Correlations

These are the correlations between levels vectors:



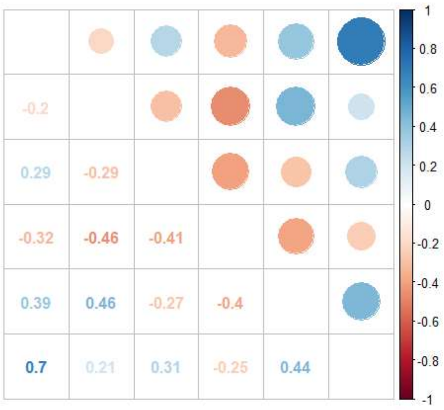
A high correlation between two trade-offs (may that be positive or negative) means that those two give us the same information on the proficiencies of the dice. It is therefore ideal to cut at least one of them from the study.

The cutting threshold is 0.7 (positive and negative).

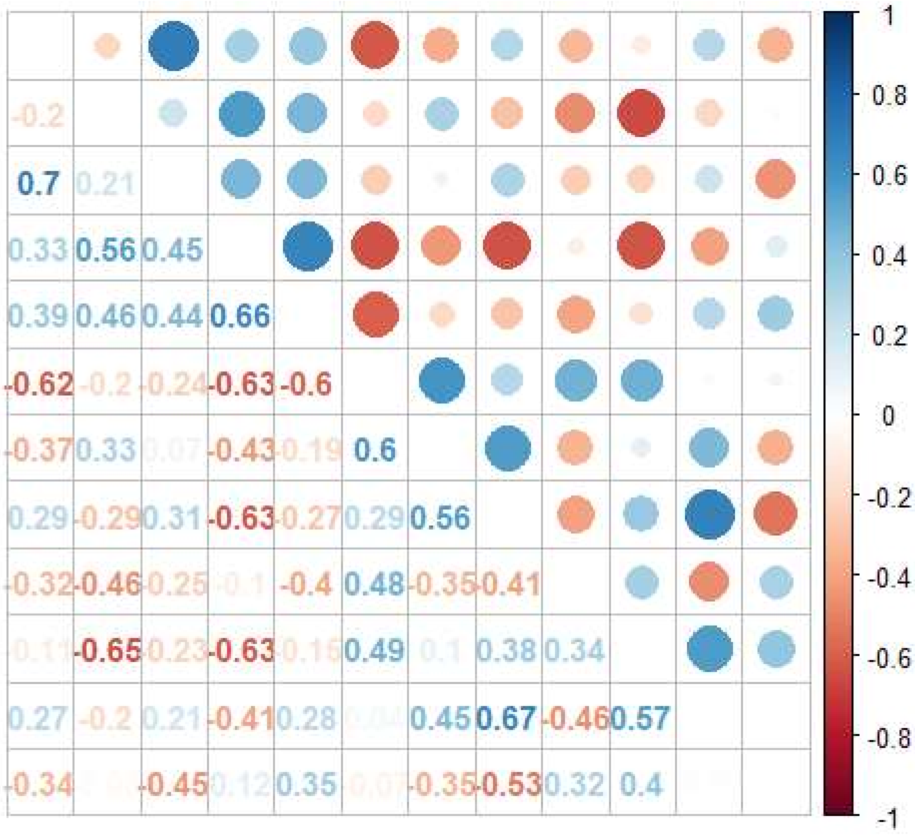
In order to skim trade-offs, I picked a few 'fundamental' ones (or that 'seem to be more natural to the human eye') and non-fundamentals are cut off if correlated to one of these:

- Trade-offs between a mean and its variance (both risk seeker and risk adverse)
- Trade-off between movement mean and movement variety
- Trade-off between movement mean and coin mean

I previously checked that these were not correlated:



This kind of selection leaves 12 trade-offs standing:



- 'Movement mean' – 'movement variance' (risk seeker)
- 'Movement mean' – 'movement variance' (risk adverse)
- 'Movement mean' – 'coins mean'
- 'Movement mean' – 'coins variance' (risk adverse)
- 'Movement mean' – 'movement variety'
- 'Coins mean' – 'movement variance' (risk adverse)
- 'Movement variance' (risk adverse) – 'coins variance' (risk seeker)
- 'Movement mean' – 'coins variance' (risk seeker)
- 'Movement mean' – 'coins variance' (risk adverse)
- 'Movement mean' – 'movement variety'
- 'Movement variety' – 'coins variance' (risk seeker)

- 'Movement variety' – 'coins variance' (risk adverse)

Performance

Levels values lie in a [0,1] range, so that:

if **a** < **b** then **a** is better than **b**

Winning frequencies lie in a [0,1] range, so that:

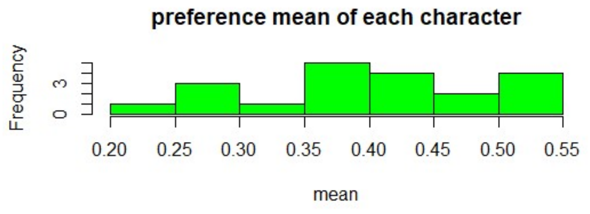
if **a** < **b** then **a** is worse than **b**

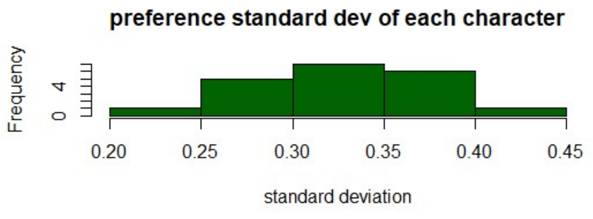
Instead of dealing with the winning frequency I decided to take a function of that value, so that it has the same semantic as level values (If **a** < **b** then **a** is better than **b**).

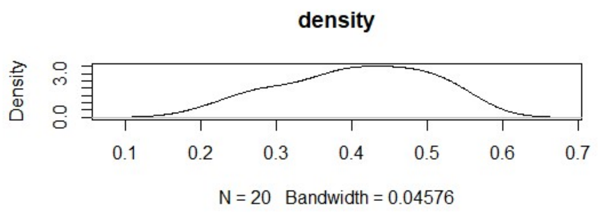
I specifically took the function $performance = {}^{1}/_{freq + 1}$ that maps frequency onto the range [0.5, 1], so that:

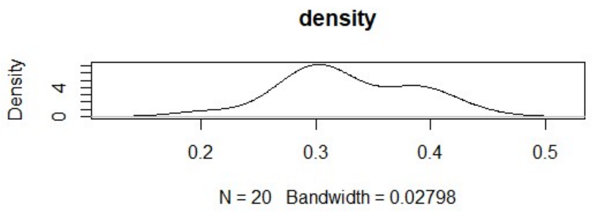
- 0.5: 100% winning rate (best die)
- 1: 0% winning rate (worst die)

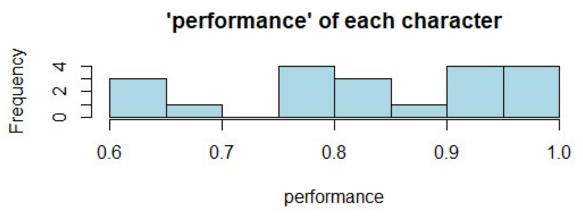
Then I considered mean and standard deviation of preferences and performance of each character.

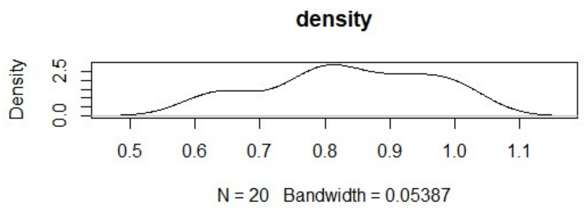


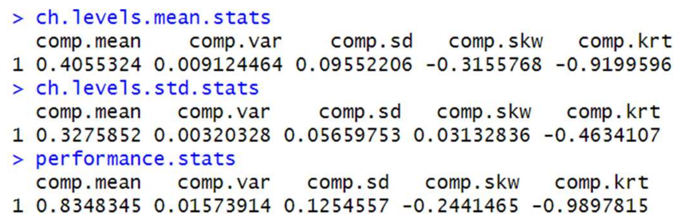












```
> ch.levels.mean.stats
  comp.mean    comp.var    comp.sd   comp.skw   comp.krt
1 0.4055324 0.009124464 0.09552206 -0.3155768 -0.9199596
> ch.levels.std.stats
  comp.mean   comp.var    comp.sd   comp.skw   comp.krt
1 0.3275852 0.00320328 0.05659753 0.03132836 -0.4634107
> performance.stats
  comp.mean   comp.var   comp.sd   comp.skw   comp.krt
1 0.8348345 0.01573914 0.1254557 -0.2441465 -0.9897815
```

Preference Mean

On average, characters have a level of 0.4, with the regarding standard deviation relatively exiguous (0.09).

This fact underlies that characters have the tendency to behave overall ‘good enough’ without much ‘excellent’ cases.

Kurtosis value shows that the distribution looks like a uniform one, rather than a normal, with negative Skewness.

Preference Standard Deviation

The average standard deviation is high (0.32) with small standard deviation (0.06).

As I said before, characters tend to have a ‘good enough’ mean, but this comes from the fact that dice are great in some trade-offs and very bad in others.

From Skewness and Kurtosis, it is derived that this distribution has a symmetry similar to a normal one, but flatter.

Performance

Both mean and standard deviation of the performance are high, which shows that characters have generally poor performances, but with both a lot of better cases and worse cases.

Performance is distributed like a uniform distribution.

Regression

Levels are then used as regressors of the performance value in order to find out which trade-offs are truly significant in telling which dice are good and which aren’t.

Conclusions about the proficiencies of dice can be extrapolated from coefficients, that allow us to compare the utilities that a die has in being strong in a trade-off instead of in another one.

A remainder of the meaning of levels values and performance values:

- Levels
  - 0: the die is on the best level
  - 1: the die is on the worst level
- Performance
  - 0.5: 100% winning rate (best die)
  - 1: 0% winning rate (worst die)

Any coefficient **c** is to be interpreted as it follows:

- **c** > 0: performance grows faster as the levels grows and grows slower as the level decreases. The higher is **c**, the ‘better’ it is to have a level near to **0**, to keep performance low.

- **c** < 0: performance decreases faster as the level grows and decreases slower as the level decreases. The lower is **c**, the it is to have a level near **1**, to keep performance low. The better a die is on this level, the worse the die is.

Example

Suppose the regression model gives us those coefficients:

- *Trade-off movement mean – movement variance <u>risk adverse</u>: 0.1*
- *Trade-off movement mean – movement variance <u>risk seeker</u>: 0.9*

Then we want to pick the best die among these two:

- *'RiskSeeker' die: risk adverse level = 1; risk seeker level =0*
- *'RiskAdverse' die: risk adverse level= 0; risk seeker level= 1*

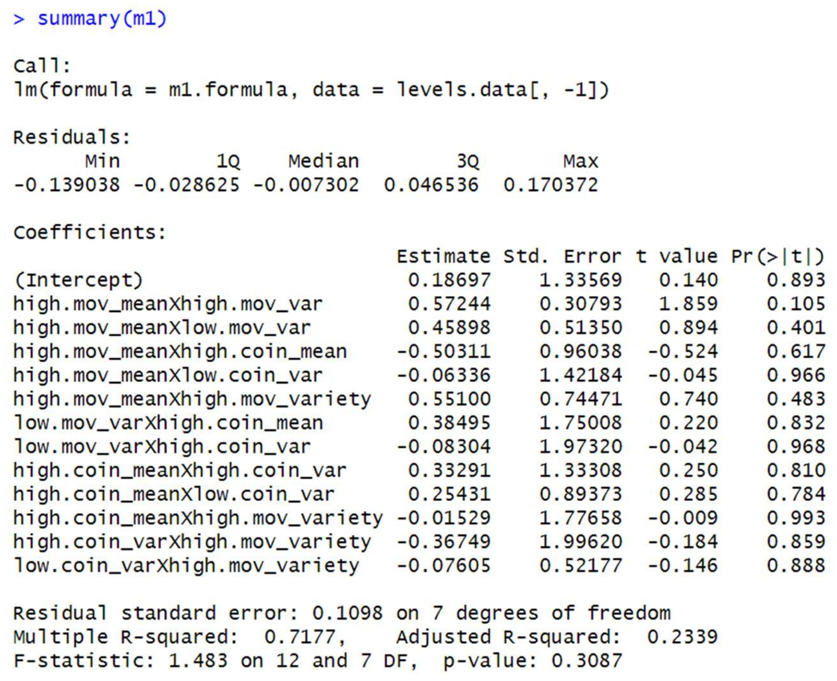
Which one to choose?

RiskSeeker is better. As a matter of facts, its performance grows by 0.1 against the 0.9 growth of RiskAdverse. Therefore, the mean-variance risk seeker trade-off level describes the effective strength of a die better than the risk adverse counterpart.

If we had found the opposite coefficients (risk adverse = -0.1, risk seeker = -0.9) it would mean that RiskAdverse die is better, as it benefits from the negativity of the risk-seeker coefficient.

Backward Elimination

The first regression model is the one with all trade-offs. Step by step the least significant one according to the t-test is removed.



```
> summary(m1)

Call:
lm(formula = m1.formula, data = levels.data[, -1])

Residuals:
      Min        1Q    Median        3Q       Max 
-0.139038 -0.028625 -0.007302  0.046536  0.170372 

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                      0.18697    1.33569   0.140    0.893
high.mov_meanXhigh.mov_var       0.57244    0.30793   1.859    0.105
high.mov_meanXlow.mov_var        0.45898    0.51350   0.894    0.401
high.mov_meanXhigh.coin_mean    -0.50311    0.96038  -0.524    0.617
high.mov_meanXlow.coin_var      -0.06336    1.42184  -0.045    0.966
high.mov_meanXhigh.mov_variety   0.55100    0.74471   0.740    0.483
low.mov_varXhigh.coin_mean       0.38495    1.75008   0.220    0.832
low.mov_varXhigh.coin_var       -0.08304    1.97320  -0.042    0.968
high.coin_meanXhigh.coin_var     0.33291    1.33308   0.250    0.810
high.coin_meanXlow.coin_var      0.25431    0.89373   0.285    0.784
high.coin_meanXhigh.mov_variety -0.01529    1.77658  -0.009    0.993
high.coin_varXhigh.mov_variety  -0.36749    1.99620  -0.184    0.859
low.coin_varXhigh.mov_variety   -0.07605    0.52177  -0.146    0.888

Residual standard error: 0.1098 on 7 degrees of freedom
Multiple R-squared:  0.7177,	Adjusted R-squared:  0.2339 
F-statistic: 1.483 on 12 and 7 DF,  p-value: 0.3087
```

The final model has six variables:

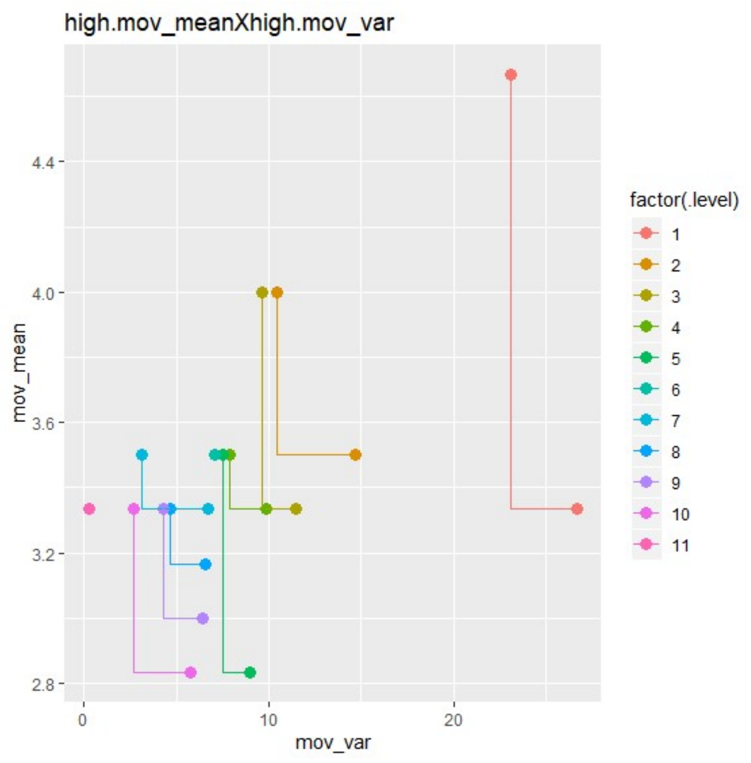
```
>     print(s7)

Call:
lm(formula = m7.formula, data = levels.data[, -1])

Residuals:
      Min        1Q    Median        3Q       Max 
-0.132863 -0.035432 -0.009791  0.042868  0.195932 

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)   
(Intercept)                     0.35144    0.12999   2.704  0.01807 * 
high.mov_meanXhigh.mov_var      0.65120    0.19560   3.329  0.00543 **
high.mov_meanXlow.mov_var       0.33679    0.11295   2.982  0.01061 * 
high.mov_meanXhigh.coin_mean   -0.36860    0.14278  -2.582  0.02279 * 
high.mov_meanXhigh.mov_variety  0.29796    0.11882   2.508  0.02620 * 
low.mov_varXhigh.coin_mean      0.48110    0.13137   3.662  0.00287 **
high.coin_varXhigh.mov_variety -0.25237    0.08555  -2.950  0.01128 * 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08868 on 13 degrees of freedom
Multiple R-squared:  0.6581,	Adjusted R-squared:  0.5003 
F-statistic: 4.171 on 6 and 13 DF,  p-value: 0.01479
```

R^2 shows that the 66% of variations in performance are well descripted by variations in trade-offs levels.

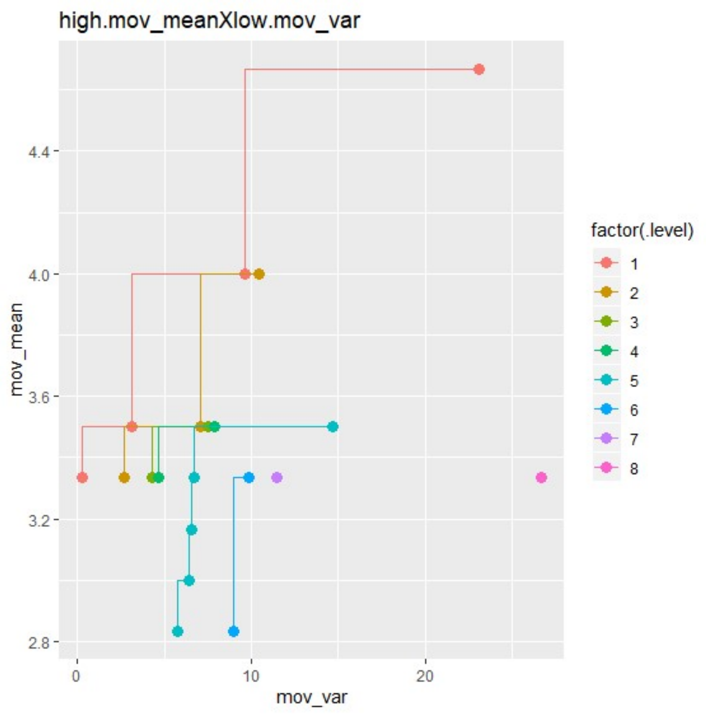
Despite the initial model had a 72%, this last one is better because:

- R^2 $Adjusted$ is greater, which tells that by dropping regressors the model improved
- $f - statistic\ p - value$ < 0.05: it is rejected the null hypothesis that this model has the same performance of a model without predictors.

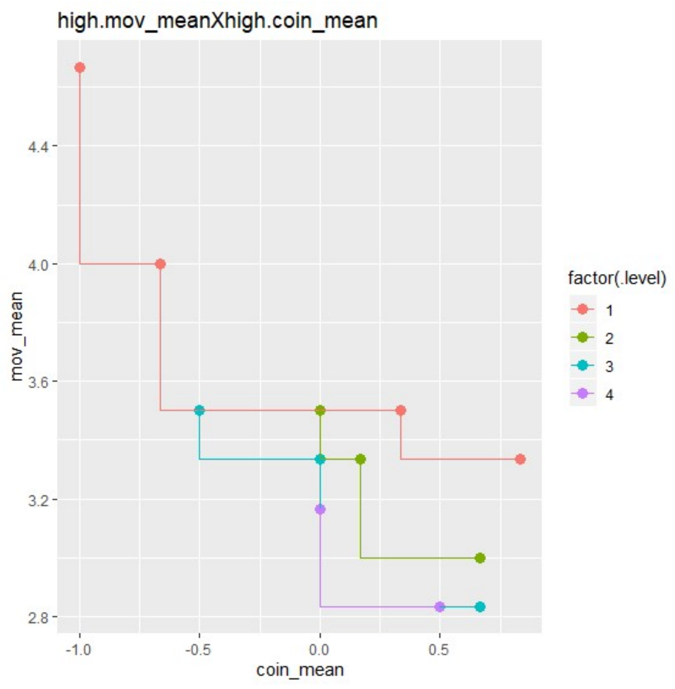
The surviving trade-offs are these:



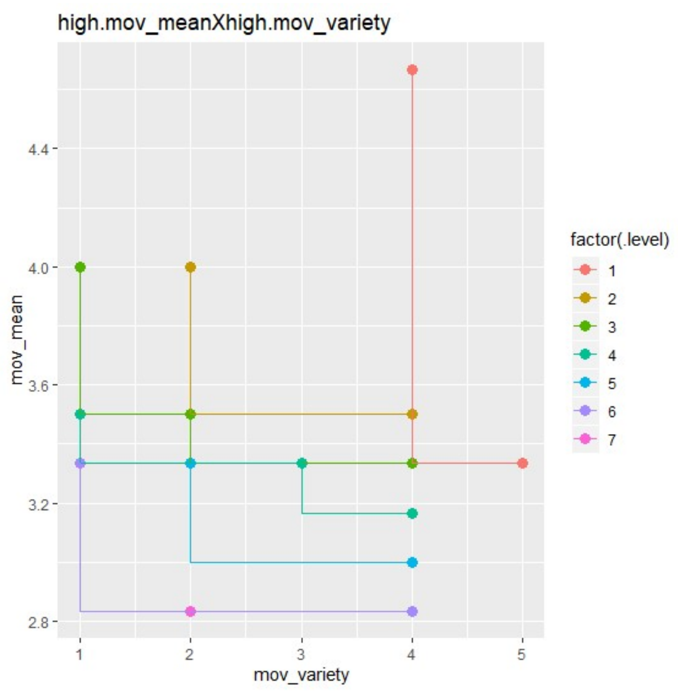
```
"Donkey Kong 1" "Bowser 1"      "Boo 2"         "Diddy Kong 2"
"Wario 3"       "Koopa 3"       "Luigi 4"       "Bowser Jr. 4"
"Dry Bones 5"   "Rosalina 5"    "Waluigi 6"     "Pom Pom 7"
"Mario 7"       "Monty Mole 8"  "Yoshi 8"       "Peach 9"
"Goomba 9"      "Shy Guy 10"    "Hammer Bro 10" "Daisy 11"
```



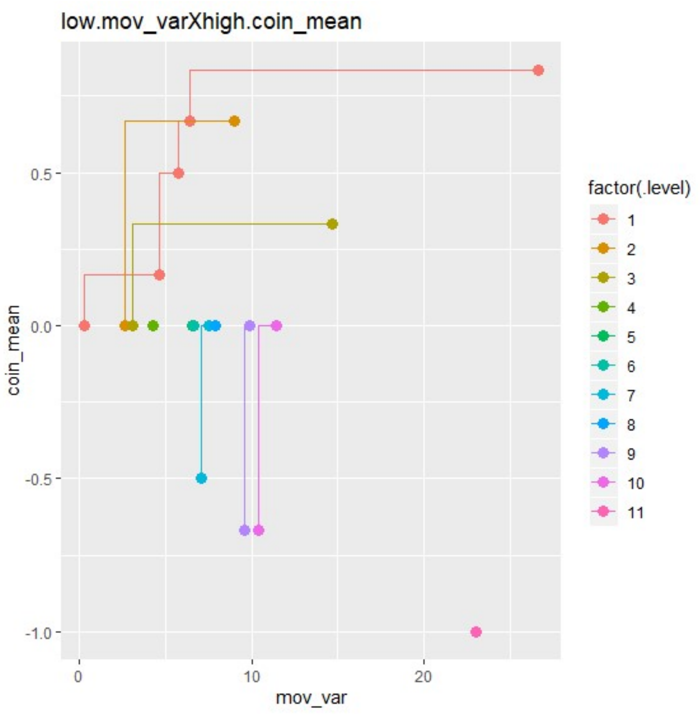
```
"Bowser 1"       "Wario 1"        "Daisy 1"         "Mario 1"
"Boo 2"          "Waluigi 2"      "Shy Guy 2"       "Dry Bones 3"
"Peach 3"        "Luigi 4"        "Monty Mole 4"    "Diddy Kong 5"
"Pom Pom 5"      "Goomba 5"       "Yoshi 5"         "Hammer Bro 5"
"Bowser Jr. 6"   "Rosalina 6"     "Koopa 7"         "Donkey Kong 8"
```



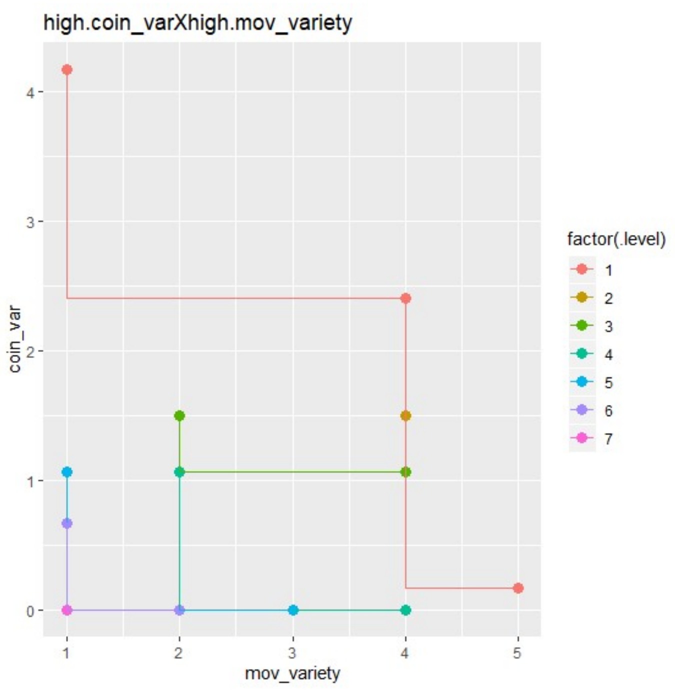
```
"Donkey Kong 1" "Bowser 1"       "Boo 1"          "Wario 1"
"Diddy Kong 1"  "Goomba 2"       "Dry Bones 2"    "Mario 2"
"Luigi 2"       "Monty Mole 2"   "Rosalina 3"     "Peach 3"
"Daisy 3"       "Pom Pom 3"      "Waluigi 3"      "Bowser Jr. 3"
"Shy Guy 3"     "Koopa 3"        "Yoshi 4"        "Hammer Bro 4"
```



```
"Bowser 1"       "Monty Mole 1"  "Boo 2"         "Mario 2"
"Luigi 2"        "Waluigi 2"     "Wario 3"       "Dry Bones 3"
"Koopa 3"        "Diddy Kong 4"  "Peach 4"       "Bowser Jr. 4"
"Yoshi 4"        "Daisy 5"       "Pom Pom 5"     "Goomba 5"
"Donkey Kong 6"  "Shy Guy 6"     "Rosalina 6"    "Hammer Bro 7"
```

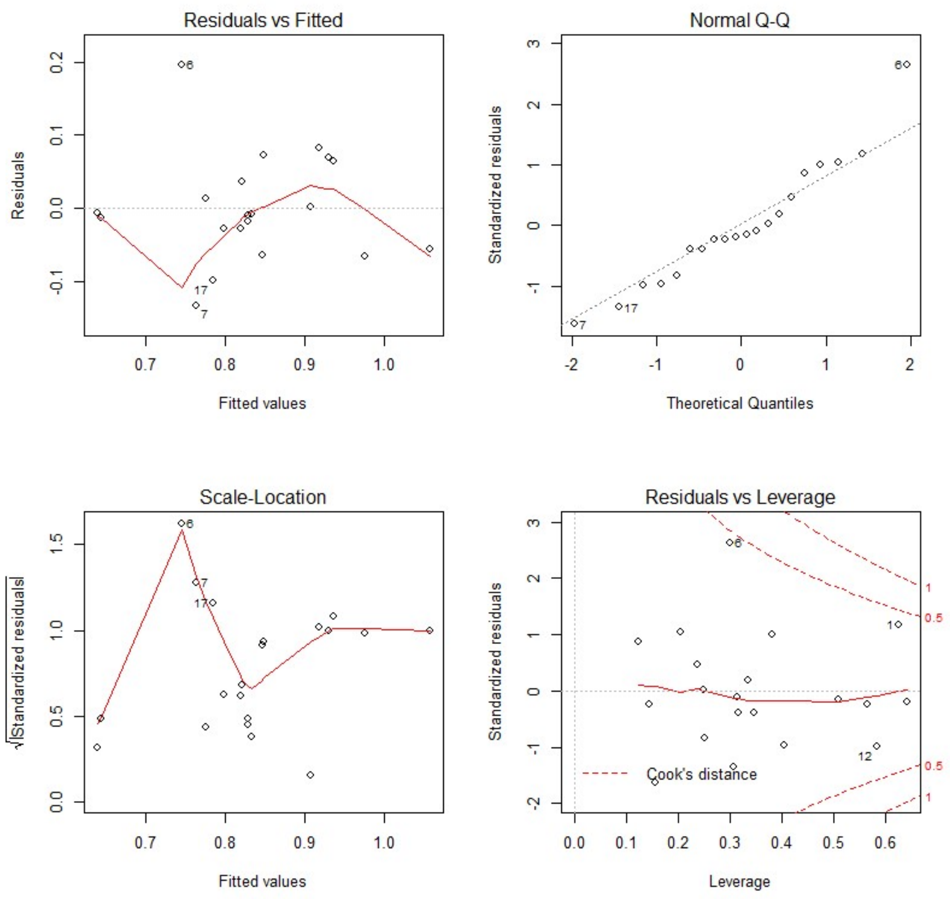


```
"Donkey Kong 1"  "Daisy 1"       "Goomba 1"      "Monty Mole 1"
"Hammer Bro 1"   "Rosalina 2"    "Shy Guy 2"     "Diddy Kong 3"
"Mario 3"        "Peach 4"       "Yoshi 5"       "Pom Pom 6"
"Dry Bones 7"    "Waluigi 7"     "Luigi 8"       "Wario 9"
"Bowser Jr. 9"   "Boo 10"        "Koopa 10"      "Bowser 11"
```

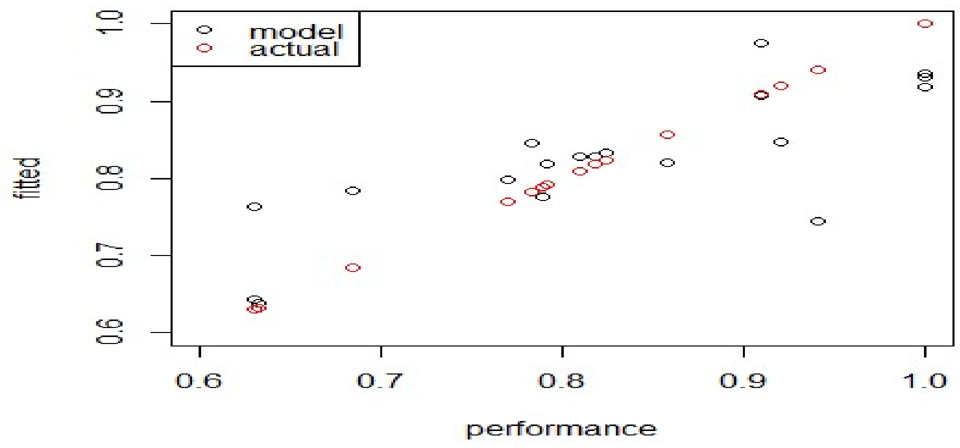


```
"Donkey Kong 1" "Bowser 1"    "Monty Mole 1"  "Waluigi 2"
"Goomba 3"      "Rosalina 3"  "Hammer Bro 3"  "Boo 4"
"Mario 4"       "Luigi 4"     "Yoshi 4"       "Koopa 4"
"Wario 5"       "Peach 5"     "Bowser Jr. 5"  "Daisy 6"
"Dry Bones 6"   "Pom Pom 6"   "Diddy Kong 6"  "Shy Guy 7"
```

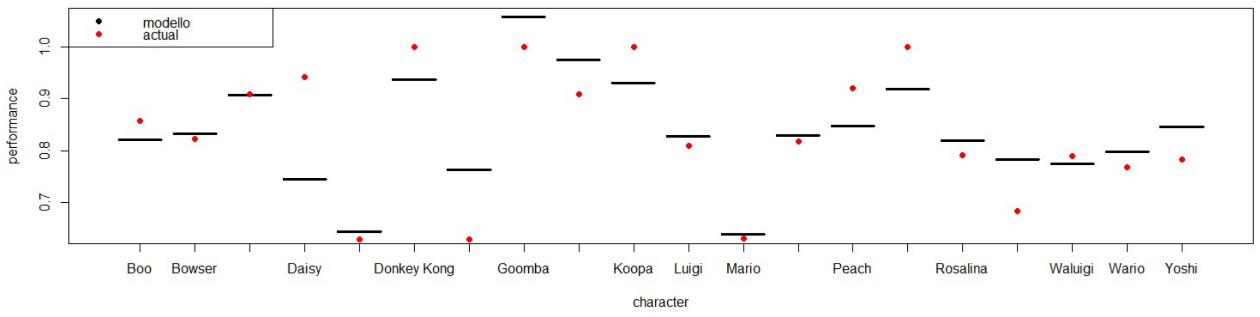
Model Analysis



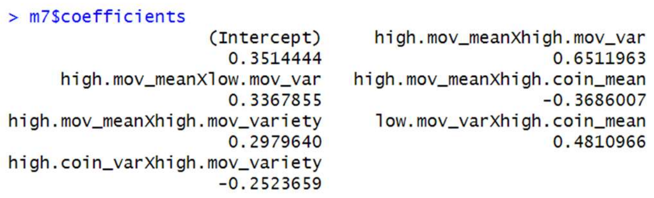
- Residuals don't show a non-linear pattern
- The QQ plot shows that residuals are well-adaptable to a straight line. We can assume that those are normally distributed
- Residuals appear to be randomly distributed on predicted values
- All points are inside the Cook's distance. There are no 'influential cases' that, if excluded, would give us a different model



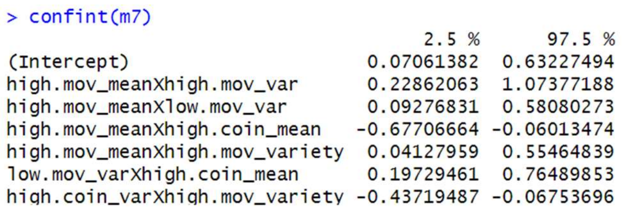
Inside the [0.7, 0.9] range the model seems to adapt relatively well to the actual performance. Outside the box, the model can deviate a lot from the true value, with cases of well-fittingness



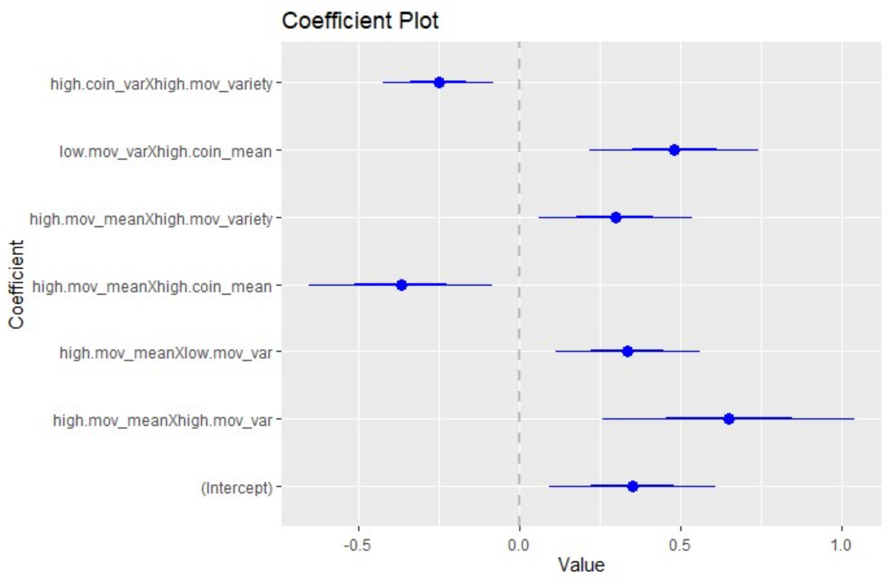
Results



```
> m7$coefficients
                  (Intercept)      high.mov_meanXhigh.mov_var
                    0.3514444                       0.6511963
      high.mov_meanXlow.mov_var  high.mov_meanXhigh.coin_mean
                    0.3367855                      -0.3686007
high.mov_meanXhigh.mov_variety      low.mov_varXhigh.coin_mean
                    0.2979640                       0.4810966
high.coin_varXhigh.mov_variety
                   -0.2523659
```



```
> confint(m7)
                                     2.5 %      97.5 %
(Intercept)                     0.07061382  0.63227494
high.mov_meanXhigh.mov_var      0.22862063  1.07377188
high.mov_meanXlow.mov_var       0.09276831  0.58080273
high.mov_meanXhigh.coin_mean   -0.67706664 -0.06013474
high.mov_meanXhigh.mov_variety  0.04127959  0.55464839
low.mov_varXhigh.coin_mean      0.19729461  0.76489853
high.coin_varXhigh.mov_variety -0.43719487 -0.06753696
```



Comments:

- The 'movement mean' – 'movement variance' trade-off has a positive coefficient for both risk seeker and risk adverse points of view. Therefore, both those trade-offs are good, but the higher 'risk seeker' coefficient suggests that this game privileges the seeking of risk
- Since trade-off between the two means has a negative coefficient, trading movement for coins seems bad, and dice with movement faces may be better.
  However, the 'coins mean' – 'movement variance' (risk adverse) has a positive coefficient. The meaning of this can be that trading movement for coins (diminishing movement to augment coins) can have a good impact if it reduces the movement variance but has a bad impact as it lowers the movement mean.
- A similar conclusion can be given about 'variety': it is good to diminish 'variety' to augment 'movement mean', but it is bad if the aim is to augment 'coins variance'

Surprisingly the 'movement mean' - 'coins mean' trade-off has not the great positive impact on performance that could have been expected.

Note that confidence levels are relatively large and overlying. Therefore, no comment can be taken as an absolute rule.

Conclusion

Truly applicable conclusions about the composition of dice could be possible by sampling a much larger number of games, possibly simulating many combinations of characters competing.

Results could be used to intervene on dice to adjust them with a view on a hypothetical future version of the game.

For instance, developers may decide to change dice proficient in the 'movement mean' – 'coin mean' trade-off to make this one have a positive impact on performance.

Another possible application is the usage of coefficients to predict the performance of newly created dice, in order to decide which are suitable to be released in a new (hypothetical) version of the game.

The gist of this is that a similar strategy could be applied on a great extent of cases, whenever we want to decide, among many resources, which trade-offs are truly important regarding the final result.

An example from the marketing area:

Cars have some characteristics (engine displacement, trunk volume, gasoline consumption, etc.); during construction phase the industry must deal with trade-offs among characteristics in the pursuit to make the best car possible. Comparing all the cars models available on the market, preference levels for each trade-off can be computed.

Once gained knowledge of this, the industry could use regression on the number of models sold (the performance on the market) to understand which trade-offs are to be taken more in account than others.