Dominion: A Board Game Breakdown

Bryan Alcorn

Stats 157

1   Dominion

You are a monarch, like your parents before you, a ruler of a small pleasant kingdom of rivers and evergreens. Unlike your parents, however, you have hopes and dreams! You want a bigger and more pleasant kingdom, with more rivers and a wider variety of trees. You want a Dominion! In all directions lie fiefs, freeholds, and feodums. All are small bits of land, controlled by petty lords and verging on anarchy. You will bring civilization to these people, uniting them under your banner.

2   Motivation

I have always wanted to try and apply statistics to a game, blending together my interests for computer science and statistics. My roommate recently introduced me to the game and we are currently battling over who can end the semester on top of the leaderboard. We are currently tied and I am hoping this game will change that.

Furthermore, I want to try and play smarter. I want to see if it is possible for me to get real world insights out of testing a game that I built. Not only do I have to build a working game of dominion, but I have to design it in a way such that it can be played automatically so that I can test it.

3   Rules

3.1   Setup

The game starts by selecting the cards to be used for the game. From the 26 possible kingdom cards, you will choose any combination of 10.

Each player receives 7 copper and 3 estates as a starting deck.

3.2 Gameplay

During each turn, you draw 5 cards from your deck, after you play a card it is labeled as played. At the end of every hand, all those cards go into a discard. When you can no longer draw, shuffle the discard pile and make that the new deck. You should be able to always draw unless all cards are played.

Each hand begins with 1 buy and 1 action.

3.3 There are four phases for a hand

1. **Actions** Play an action card, follow the instructions on the card, and deduct one from your action points.

2. **Buy** a card from the board if you have enough money and deduct one from your buy points. Normally this will be just one buy per hand.

3. **Cleanup** Everything in the hand, the purchased card, and the played cards go in discard

4. **Draw** Draw again to 5. You should have 5 in your hand before you play in case another action card requires you to modify your hand in some way.

3.4 End

The game ends if there are no more provinces left or three of the stockpiles are empty. The player with the most victory card count wins.

4 General Approach

1. Build a playable version of the game in Python

2. Calculate some basic probabilities to get an intuition into which strategies might work best or to determine some simple tricks to play better

3. Modify the game so that various strategies can be played against each other with varying number of players

4. Build the strategies

5. Based on the best strategies, build an in game helper that tells you what to play so you can ask it what to play during a real game against friends.

6. Using the simulations, generate test data to run some statistical models on the results of the games to get insight into which cards are most effective

5 Goals

1. Learn about designing a game that can we can run simulations on to learn more about what actions tend to lead to victory

2. Test our assumptions by making different strategies and putting them against each other to see which one is better

3. Determine which cards give us the most value during a game

6 Game Design

The game is built using object oriented programming in Python. The game is broken down into three main operational files

- ***player.py*** - The player controls everything relating to their deck and hand. They draw, discard, play, and win.

- ***gameBoard.py*** - The board manages everything a player has to interact with. It contains the gold, the stockpiles, and methods to interact with the board. The board has a subclass called ***Cards*** and each card can be played by a player on a game board.

- ***play.py*** - Play manages the logic of the game. This is where you select the initial cards to play with, it checks to make sure money ¿= cost of a card when buying, and it also manages every turn until someone wins. Strategies will also be added onto this to made the decisions then their player comes up. Instead of asking the player what to do, the strategies will take care of the buy or play decisions.

From there, ***strategies.py*** was added on as a way to attach different strategies to ***play.py*** and ***models.py*** is used to do tests and calculate probabilities.

7 Command Line Interface

The game is playable through the command line with commands to buy, play, or end the turn.

```
Starting Dean's turn

('Stockpile', 'MERCHANT', 'cost 3')
('Stockpile', 'MONEYLENDER', 'cost 4')
('Stockpile', 'CELLAR', 'cost 2')
('Stockpile', 'VASSAL', 'cost 3')
('Stockpile', 'BUREAUCRAT', 'cost 4')
('Stockpile', 'WORKSHOP', 'cost 3')
('Stockpile', 'MILITIA', 'cost 4')
('Stockpile', 'VILLAGE', 'cost 3')
('Stockpile', 'MOAT', 'cost 2')
('Stockpile', 'GARDENS', 'cost 4')
('Golds', 30)
('Silvers', 40)
('Coppers', 60)
('Victory Cards', ['PROVINCE', 'DUCHY', 'ESTATE'])
Make your move, here is your hand Dean
Commands (the space is important):
 end: ends your turn
 play #: will play the #th card in your hand
 buy CARD_NAME: will buy the CARD_NAME from the game board, this can be money, a stockpile, or a victory

('Played:', [])
('Total Money left', 4)
('Total Actions Left', 1)
('Total Buys Left', 1)
('Your hand:', ['COPPER', 'COPPER', 'ESTATE', 'COPPER', 'COPPER'])
Command: buy village
['buy', 'village']
Buying VILLAGE

Make your move, here is your hand Dean
Commands (the space is important):
 end: ends your turn
 play #: will play the #th card in your hand
 buy CARD_NAME: will buy the CARD_NAME from the game board, this can be money, a stockpile, or a victory

('Played:', ['VILLAGE'])
('Total Money left', 1)
('Total Actions Left', 1)
('Total Buys Left', 0)
('Your hand:', ['COPPER', 'COPPER', 'ESTATE', 'COPPER', 'COPPER'])
Command:
```

8   Exploratory Analysis

There are $\binom{26}{10} = 5311735$ ways to set up a game with all 26 cards.

There are $\binom{12}{10} = 66$ ways with the 12 cards. This smaller test set will make things easier to analyze and break down.

We know from how the deck is built and that since each player starts with 10 cards in their deck, they will go through two full hands with no actions to play. After that, assuming that player purchased an action card in the last hand, there is a

From this, there are two approaches, I might just get a throne room, which on its own does not do anything, or the player has X percent chance of getting a throne room and another action card.

9   Strategies

9.1   Intuition

There are many strategies online that talk about effective uses of different cards and different combinations that go well together. The game becomes more complicated when you potentially *Throne Room* a *Smithy*. Also, considering that the game is about building a hand, you want to first buy cards that help you build faster and then buy cards that give you buys/actions/money. One a good deck is built, provinces are easier to afford.

There are two extreme strategies that should be explored in order to find a balance between the two.

1. Build up actions so that the player can keep playing action cards, earning more gold, and drawing more cards and hopefull more money. This extreme is useless however if no more of the cards are money. That being said, there are seven coppers total, so with the proper action cards, it is possible to buy a province.

2. If a player only buys gold, then they only have 5 cards to get money, but the hope is that those cards are of higher value and will add up to at least 8. Actions would be helpful here since they might give more cards and in turn, more money.

Obviously a balance is required between buying money and buying action cards. This is one of the objectives of this project.

9.2   Simulated Strategies

These will be our data generation tools. They will record their move and in the end, record if they win or lose. As a basic rule, they

1. **Random** Randomly chooses an action card if there is one and then buys a random action card that it can afford.

2. **Max Actions** Takes cards that lead to more and more actions and draw cards. This might help if we learn that lots of actions builds long hands with money.

3. **Max Money** Buy money whenever possible that isn't a copper. Otherwise, buy a random action

4. **Balance** Maintain a balance between action cards and buying money. The rationale here is give infinite actions, those don't help much if the player has no money. And vice versa, if the other player only buys money, the might lose still because only getting money might take more time.

9.2.1 Simplifications

Only a subset of cards are being used for these simulations because they require no user input. The rest of the cards require some decision making beyond just picking a card to play. This isn't a big problem since the strategies can be expanded to include also decisions with other players. I am ignoring the rest of the cards however to ensure I get good results with a subset of cards.

Copper should not be purchased, takes up one of the precious five spots in the hand. We will have random that might purchase a copper card so we will still have some player data.

10 Simulations

Simulations is one of the main tools for analyzing different strategies because it allows us to play many games with varying amounts of players and strategies. From there, we can statistically say which strategy is better than the rest by running hypothesis tests on the results. To get these results, again the architecture of the game had to be modified. It was originally designed to be one game at a time, so another game mode is needed to run n numbers of games and generate win numbers from that.

10.1 Qualitative Insights

Running the random tests against each other produces long games where provinces are rarely won. This makes sense because in order to get 8 money in a hand, the player either needs a string of actions or to have a lot of silver. This is much less likely if the hands are played at random. When a province is bought, we can look at which cards tend to

be played that give the player enough money to buy the card. This will be useful for determining which cards give us the most value.

Another interesting observation that comes from looking at the simulated games, the scores are higher than normal. This comes from purchasing many estates and gardens. Since the hands tend to be larger with random, the gardens have more yield because their value increases as you build up your hand.

10.2 Strategies Explained

Strategies that were tested

For every strategy, I programmed some basic common sense heuristics into their decision making such as buying a province when that strategy has enough money and the basics rules of playing cards first before buying in order to build up money.

10.2.1 Random

10.2.2 Max Actions

10.2.3 Max Money

### 10.2.4 Balance

10.2.5 Card Specific

Festival and Market

10.3 Win Charts

Games were player with 3 players to start. Here is an example of some of the simulation scores:

Random

The random strategies performed as expected. There were more outliers and the win scores were about the same. This simulation was more of pilot test to see how the simulations would interact.

Random v Random v Random

Players
1000 games were played
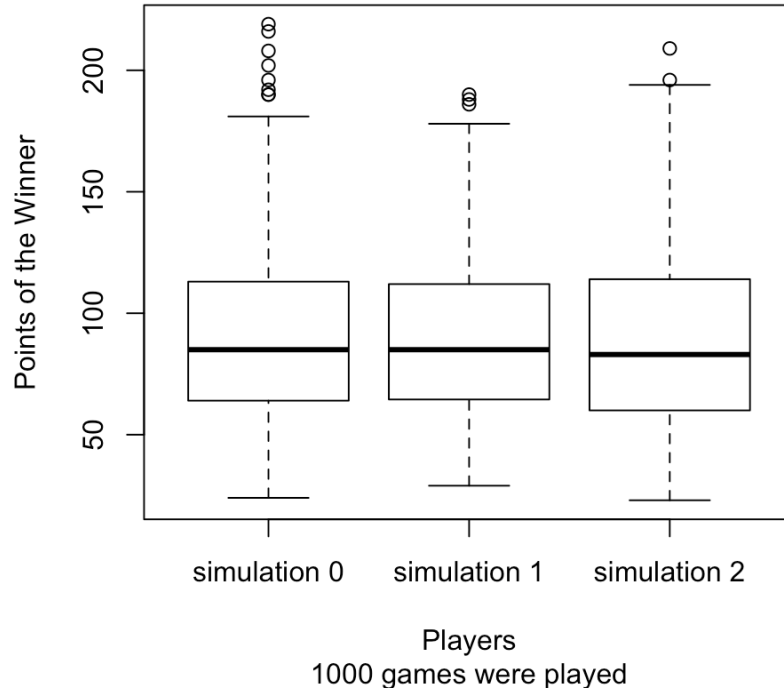
Action vs. Money vs. Random

Money strategy did the best while random rarely won. From this we have an indicator that maximizing money might be a better first strategy to get money towards buying a province. It is worth noting that the action strategy tends to lead to more cards and actions, this is effective if there is money to draw, but if only actions are drawn, then more actions are not as effective as they can be.
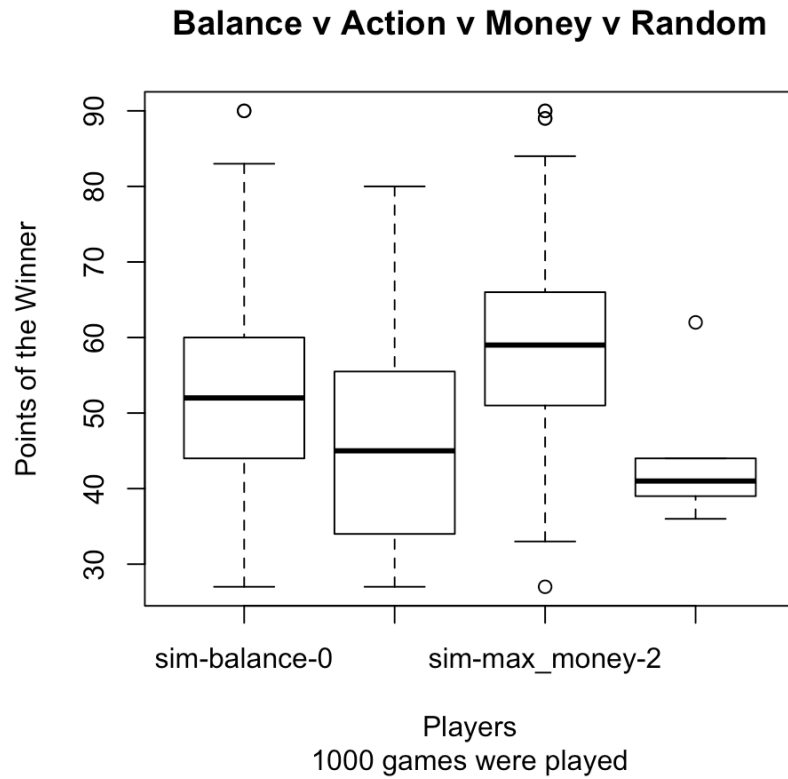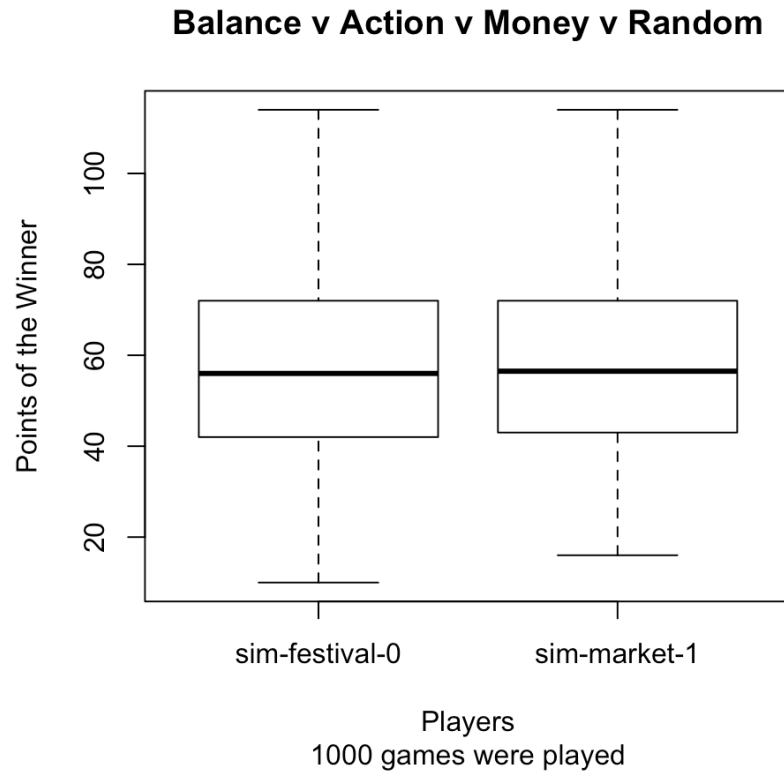
Actions vs. Buys vs. Random

Players
1000 games were played
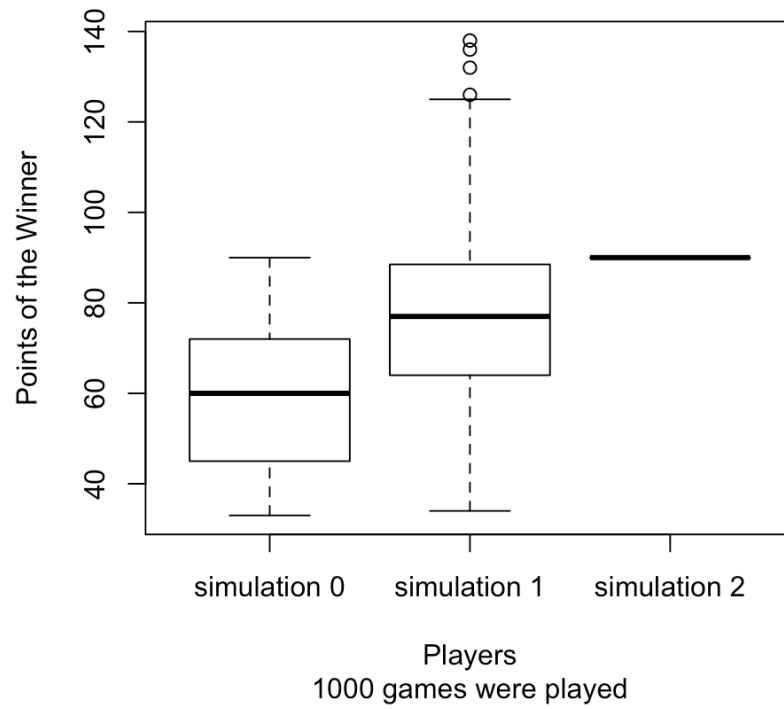
Balance vs. Action vs. Money vs. Random

Balance takes on some of the flaws of the action strategy. Thus, it sits in-between the best strategy so far, Max Money, and the okay action strategy.

Balance v Action v Money v Random

Players
1000 games were played

Festival vs. Market

While market is slightly higher, they are similar enough that there is probably no significant difference. This will be tested in the hypothesis tests. Notice that the scores are lower for these than the other tests. Lower scores might indicate quicker wins. A quick win means someone is able to buy a province faster than normal.

Balance v Action v Money v Random

Points of the Winner

100

80

60

40

20

sim-festival-0   sim-market-1

Players
1000 games were played

Get some win results and say if they are statistically significant with some tests, get those p-values

11   Gathering Data

Data was stored in CSV files. For example, the random vs. random games had rows with values $< Winner, Winner_score >$. This entry was added to the CSV file after every simulation

Simulations were run with game parameters in one file called ***commands.in***

```
sim
1
2
3
4
5
6
```

```
7
8
9
10
random,random,random
```

Then, a bash script was run to run the commands over and over again

```
#!/bin/sh
for i in 'seq 1 1000';
        do
                python play.py < commands.in
        done
```
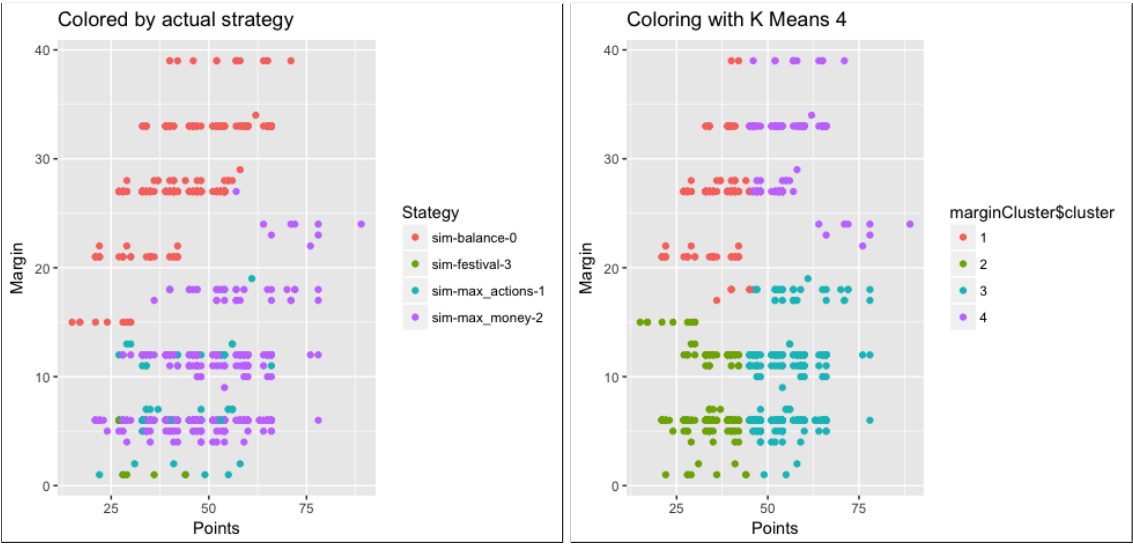
12   Brief Model Overview

Can talk about the models I used to for strategies

12.1   K-Means

Errors

withinss	8833.396 10936.840 24079.819 14742.027
betweenss	192660.3

12.2   Hypothesis Testing

Using hypothesis tests, we can now answers questions like, 'Is playing festival on average a better move than market?' or 'Is strategy A, on average, better than strategy B?'

```
#now strategy differences
colnames(win_all) <- c("Name", "Points")
money_v_random = win_all[(win_all$Name == "sim-max_money-2" |
win_all$Name == "sim-random-3"), ]

counts = c()
for (item in money_v_random$Name) {
  if (item == "sim-max_money-2") {
    counts = c(counts, 1)
  } else {
    counts = c(counts, 0)
  }
}
money_v_random$counts = counts
money_v_random$Name <- droplevels(money_v_random$Name)
prop.test(table(money_v_random$counts, money_v_random$Name), correct = FALSE)
```

12.2.1   Results

```
        2-sample test for equality of proportions without continuity correction

data:  table(fest_v_market$counts, fest_v_market$Name)
X-squared = 1002, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 1 1
sample estimates:
prop 1 prop 2
     1      0
```

```
        2-sample test for equality of proportions without continuity correction

data:  table(money_v_random$counts, money_v_random$Name)
X-squared = 544, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -1 -1
sample estimates:
prop 1 prop 2
     0      1
```

13   Risks

Strategies don't produce good results for testing because having the entire game come down to just just one win/loss classification doesn't make realistic sense for scaling this approach. Every move could be taken into consideration for more accurate results. There would need to be a way to measure if each move is good or bad though.

The action strategy might not be good because it could be more of a support role, meaning it works well in combination to other cards, but not by itself against a money strategy. This also accounts for the weaknesses in balance. To fix this, strategies need to be smarter with more rules to discover these additional benefits from pairing action cards. Money cards don't pair well which makes a direct comparison harder.

Not testing more of the cards, but would make the simulations much more complicated. This was still a good approach for understanding the relationship between drawing, buying money, and buying cards that give actions.

14 Results

15 Summary

16 Coding Reflection

1. Testing framework should be part of the design upfront. I had to test most things manually which was not as efficient as it could have been and it doesn't let me catch as many bugs. What was useful was simulations which caught a lot of errors.

17 Miscellaneous and further ideas

There are many different approaches to analyzing this game and many more questions that could have been asked. Throughout the process, I wrote down different aspects of the game that could have been tested. Here are some of the things that could be done on top of what I already did. Since the game can be easily adapted to include other modes of play and other simulations, they are not difficult to add on.

One thing to look at is getting more information out of each simulation, right now the outputs are only who wins and the points that were scored by the winner. However, other good information might be things such as typo of victory (either based on the stockpiles or provinces) and margin of victory. These numbers could be used to figure out a ranking of scores. So this way instead of having to run a lot of simulations with different strategies, instead one could find scores of how well the simulations perform. This might help quantify how good specific modifications are to strategies.

Additional modes:
A game helper mode would make all of this information practical. It would take in information about a game someone is playing and recommend a move for them. This would be useful for putting the project insights to the test by actually providing insights into a particular game. Another mode would be person versus computer strategy mode where individuals can play against their strategies.

Additional Strategies:
There are many more strategies that can be tried to find the best solution. We could run a simulation to see if Village is better than Cellar. This is fun for answering simple queries

like "is card x better than card y," but to really figure out more into an optimal strategy, we need to do more machine learning to figure out our best moves.

Additional things to explore time permitting:

1. Average score for each simulation

2. Probability of particular hands to help guide the strategies

3. more simulations

4. probability models

5. Incorporate ML concepts to Dominion, the goal is to maximize points and we would train on the moves in a game that lead to increased points only if that game leads to a win

6. Redesign the game in some aspects, making the game modes more modular and the simulations easier to adjust

7. Program more cards to auto play so simulations still work

TO ADD based on the cost of village and lab, I said cards are more valuable than actions if they are both present (smitty)

how I designed the strategies, actions vs money and always buying a province. How the base should be better than random but its a good first thing.

make it so each strategy buys a province if they can and redo your simulations!

k means

significant tests

run some cards against each other, is festival better than market, obviously a combo is probably best but maybe it can help make a better first move

check this doc with Grammarly - test run

add in your caveots, like these scores are unrealistic because they rack up points rather than going for the kill

18   Open Source

I wanted to make sure all the code can be viewed and experimented with. My github can be found here: `https://github.com/rollonbears234/dominion-game`.