2020 Less Wrong Darwin Game

The Darwin Game

~~Click here to participate. Entries must be submitted on October 18th, 2020 or earlier.~~

Entry is now closed.

In 2017, Zvi posted an exciting story about The Darwin Game, a variation of iterated prisoner's dilemma.

I will run my own version of the game in the week following October 18th, 2020. **You do not have to know how to program in order to participate.** I will code simple bots for non-programmers. If you do know how to program then you may create your own complicated bot.

Here are the rules. Changes from Zvi's original game are in brackets [like this].

> For the first round, each player gets 100 copies of their program in the pool, and the pool pairs those programs at random. You can and often will play against yourself.
>
> Each pair now plays an iterated prisoner's dilemma variation, as follows. Each turn, each player simultaneously submits [an integer] from 0 to 5. If the two numbers add up to 5 or less, both players earn points equal to their number. If the two numbers add up to 6 or more, neither player gets points. This game then lasts for a large but unknown number of turns, so no one knows when the game is about to end; [I guarantee it will be at least 100 turns per iterated prisoner's dilemma].
>
> Each pairing is independent of every other pairing. [You do know what round of the game it is and that you are facing an opponent. If you face a copy of yourself you are automatically awarded the maximum 5 points per round (2.5 points per bot). You otherwise do not know any history of the game to this point.] Your decision algorithm does the same thing each pairing.
>
> At the end of the round, all of the points scored by all of your copies are combined. Your percentage of all the points scored by all programs becomes the percentage of the pool your program gets in the next round. So if you score 10% more points, you get 10% more copies next round, and over time successful programs will displace less successful programs. Hence the name, The Darwin Game.
>
> Your goal is to have as many copies in the pool at the end of the last round as possible, or failing that, to survive as many rounds as possible with at least one copy.
>
> [I will attempt to iterate until there is a stable equilibrium.]

I will add some silly bots of my own to make the early game more interesting.

Instructions for non-programmers

Please give a simple explanation of what you want your bot to do. Write it with mathematical precision. If your specification is even slightly ambiguous then you will be disqualified.

Instructions for programmers

Write a program of the following format.

```
class TitForTatBot():
    def __init__(self, round=0): # Edit: Changed "1" to "0"
        self.turn = 0
        self.round = round
        self.previous = None
    def move(self, previous=None):
        self.turn += 1
        if self.previous:
            output = self.previous
            self.previous = previous
            return output
        else:
            return 2

# Edit 2020-10-11: The above code is wrong. The properly-implemented TFT `move`
method looks like this.
#    def move(self, previous=None):
#        self.turn += 1
#        if previous == None:
#            return 2
#        else:
#            return previous
```

Your class must have an `__init__(self, round=1)` intializer and a `move(self, previous=None)` method. You may write your class in Python 3 or Hy.

Unlike Zvi's original game, you do get to know what round it is. Rounds are indexed starting at 0. The `previous` parameter of the `move` method is an integer indicating what your opponent did last iteration. If it is the first iteration then `previous` equals `None`.

A new instance of your class will be initialized in each round. You may save whatever information you want into the class instance's fields but you may not save information between rounds or between class instantiations. The `move` method must always return an integer from 0 to 5 inclusive.

You may import standard libraries like `random`, `scikit` and `numpy`.

Coordinating with other players

Anyone can play, but only players with a Less Wrong account that existed before I declared this tournament will be allowed to coordinate out-of-game. This rule exists to prevent players from submitting multiple entries to this contest and self-coordinating. Coordinating with other people is encouraged. Coordinating with yourself between multiple separate entries is cheating.

~~Click here to participate. Entries must be submitted on October 18th, 2020 or earlier.~~

Entry is now closed.

The Darwin Game - Rounds 0 to 10

The most important change between my game and Zvi's original is that bots can read each others' source code. They can simulate each other and predict each others' behavior. Within a day of the tournament launching—and an entire week before entries closed—Zack_M_Davis had already written a bot to simulate opponents and then open-sourced it to everyone.

That's what happens when a significant contributor to an open source Lisp dialect participates in a software competition.

Taleuntum wanted to write an even better simulator but was informed that it would take too many years to run.

Muticore solved the limited compute problem and wrote a safe, effective, obfuscated simulator, with randomization.

The Phantom Menace

Three separate people asked me what happens if a bot crashes the game while simulating an opponent with malware in it. This turned out not to matter because nobody deployed malware to destroy simulators. Only one player, Measure, deployed malware—and the malware didn't crash the game. Instead, it attempted to replace its opponent's `move` method with a method that returned `0` instead. But the threat of getting disqualified seemed to scare away other potential simulators.

Taleuntum did write a `MatrixCrashingBot` that crashes simulators but did not submit it. This is a disappointment, as Taleuntum would have been allowed to submit this bot as a separate entry on the grounds that it does not coordinate with Taleuntum's CloneBot. To my knowledge, nobody else took advantage of this deliberate loophole in the rules either.

RaterBot safely combed through its opponent's source code for "2"s and "3"s to estimate aggression without the dangers associated with running untrusted code.

Computer programs attempting to simulate each other can produce complex behavior. The behavior is so complex it is provably undecidable—and that's totally ignoring the real-world sandboxing problem.

Nevertheless, two contestants requested I write code to simulate their opponents. I refused these requests. Zvi[1] accepted a simpler bot and the other contestant dropped out.

> I'm surprised running the enemy is complicated though—it should just be a function call.
>
> —quote from the contestant who dropped out

The most significant use of an opponent's source code came from Vanilla_cabs.

Attack of the Clones

Zvi's original game was dominated by a clique of players who coordinated out-of-game to defeat the non-clique players. It worked great—and then defectors within the clique dominated both the non-clique players *and* the clique players.

Vanilla_cabs observed that players could use each other's source code to detect (and therefore punish) defection from a clique. Leveraging this insight, Vanilla_cabs created a

CloneBot template that would cooperate only for the first 90 rounds—long enough to eliminate all non-clones—and then allow players to do whatever they wanted afterward.

Brilliantly, the CloneBots used the lexographic length of custom code to break parity symmetry and guarantee perfect coordination between CloneBot instances instead of having to fumble around for the initial rounds of each pairing.

At the same time, Vanilla_cabs worried that someone might attempt to bypass the CloneBot's self-recognition system. As a first line of defense, Vanilla_cabs kept the source code secret from non-cloners. As a second line of defense, Vanilla_cabs deliberately left open a weakness as a honeypot. Among the first co-conspirators in the clone army, only Lanrian reported this weakness.

Taleuntum, in contrast, contributed two lies to the clique intrigue.

> I lied that I've already submitted one program detecting and crashing simulators... obviously I did not even plan to do so. Some time later...I added another lie that the method of detecting simulators was my friend's idea (hopefully suggesting that there is another contestant with the same method outside the clique).
>
> —Taleuntum

One player discovered a way to defect from the CloneBots.

Multicore

The original CloneBot template used `splitlines()` to parse source code. But the Python3 interpreter behaves slightly differently from `splitlines()`. In particular, `splitlines()` splits on the groups separator character `0x1D`, which Python3 does not treat as a newline. By placing `0x1D` group separator characters inside of a comment, Multicore's EarlyBirdMimicBot bypassed the CloneBot self-recognition system.

Multicore did a lot of things.

- Wrote a simulator.
- Optimized the simulator via a branching algorithm to perform efficient searches.
- Used randomization to disrupt opposing simulators.
- Protected the simulator from malware.
- Joined the CloneBots.
- Defected from the CloneBots via a zero-day exploit in the CloneBot code.
- Recruited two friends to submit password bots to serve as fodder.
- Adjusted strategy based on the round number.

When I hosted this tournament, I hadn't expected anyone to "[read] through the C code for the python lexer".

For a complete write-up of Multicore's strategy, including source code, see here.

> On a side note, I really love this site. I can't really recall any other game I've been in getting this tangled.
>
> —Emiya

The First Game

The first iteration of the game was run by Taleuntum who ran "a simulation of the whole tournament till the 160th round with 8 bots" despite the tournament's source code not being public at the time.

Taleuntum's tournament was unofficial and does not count.

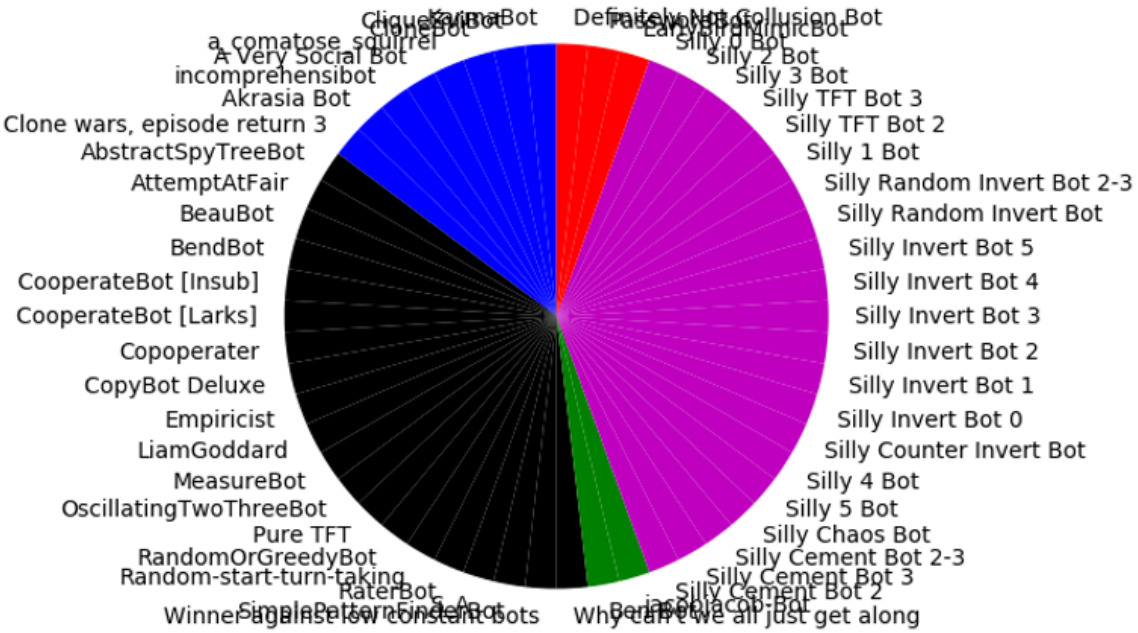
The Real Game

Teams

In order to make sense of the 54 participating bots, I have bucketed them into teams.

- [Blue] **Clone Army.** 10 players pledged to submit clone bots. 8 followed through, 1 didn't and Multicore submitted a [Red] mimic bot.
- [Red] **Multics.** Multicore's friends submitted 2 password bots to aid Multicore's mimic bot.
- [Green] **Norm Enforcers.** Ben Pace joined forces with jacobjacob to form their own little duo.
- [Black] **Chaos Army.** 20 players wrote individual bots.
- [Magenta] **NPCs.** I wrote 21 Silly Bots. Some of them had synergies.

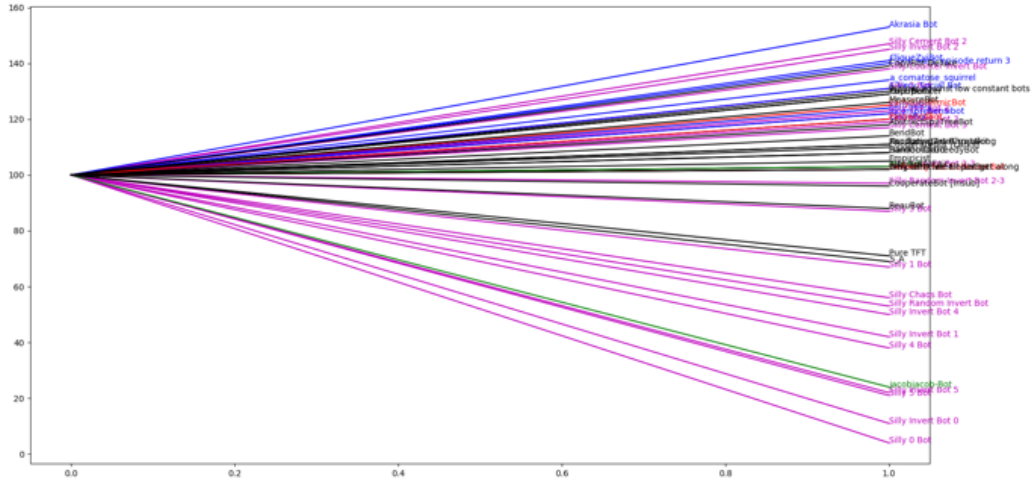
The clones [Blue] begin the game outnumbered 6-to-1.



Edit: Everything below this line is in error. See here for details.

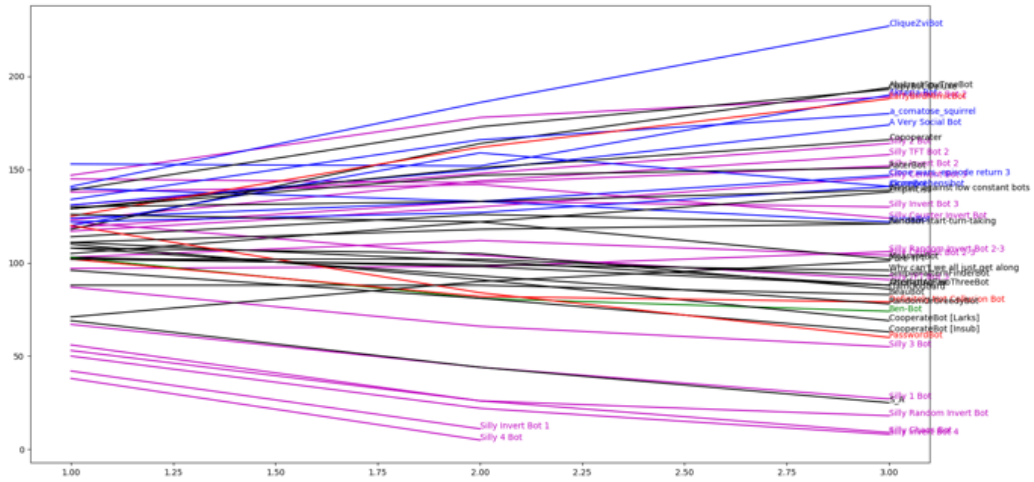
Round 1

5 bots died on turn 1 including 4 NPCs and Team Norm Enforcers' jacobjacob bot.



Rounds 2-3

Another 4 NPCs died.

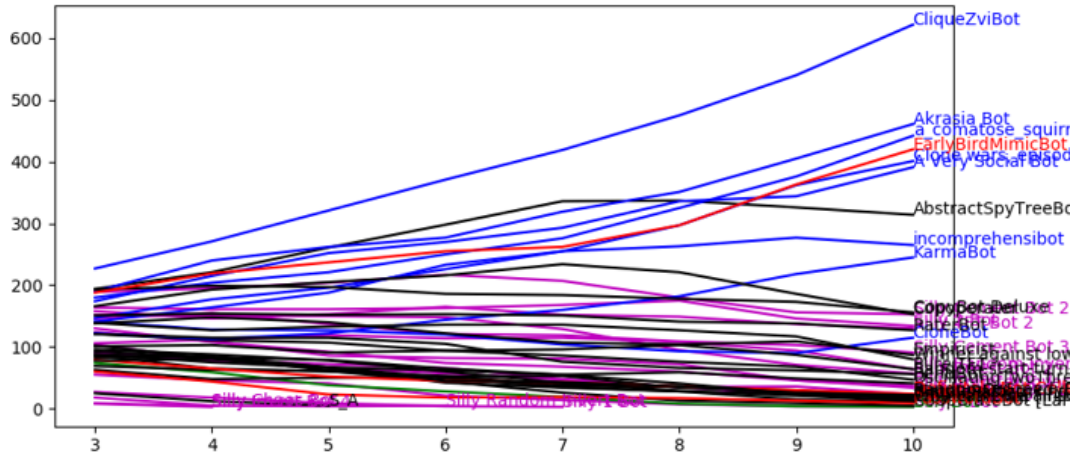


Rounds 4-10

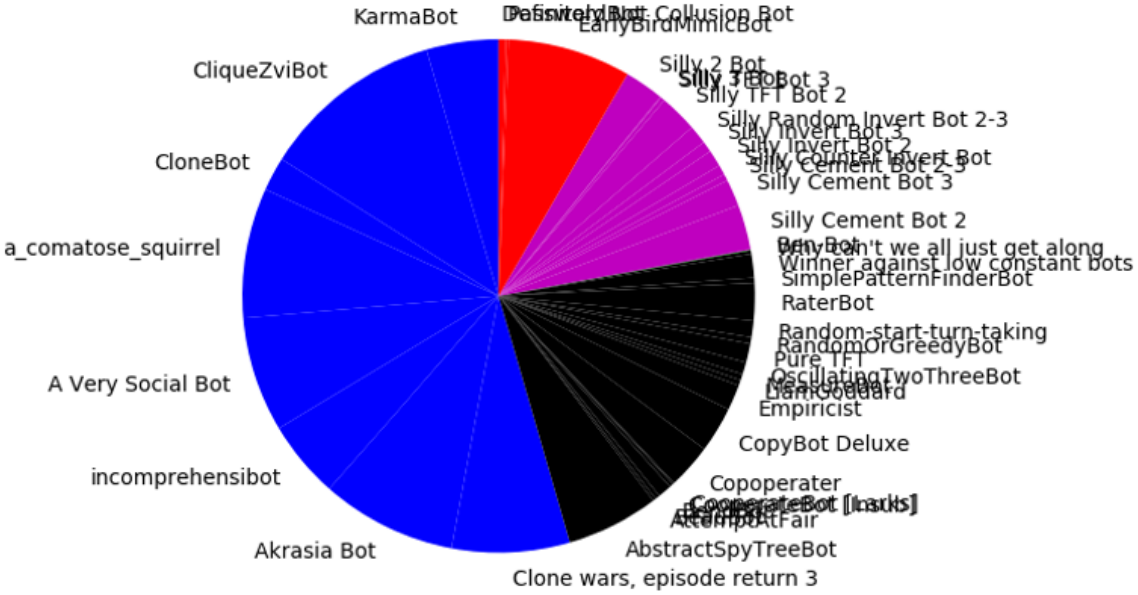
S_A and BenBot die, along with 3 more NPCs. Thus ends team Norm Enforcers.

The clone army is mostly doing well, except for CloneBot which is doing poorly and AbstractSpyTreeBot which is doing almost as well as the average clone.

EarlyBirdMimicBot is doing better than the average CloneBot but not by much. The MimicBot's `0x1D` exploit succeeded in defecting but the bot appears not to have leveraged its defection to gamebreaking effect.



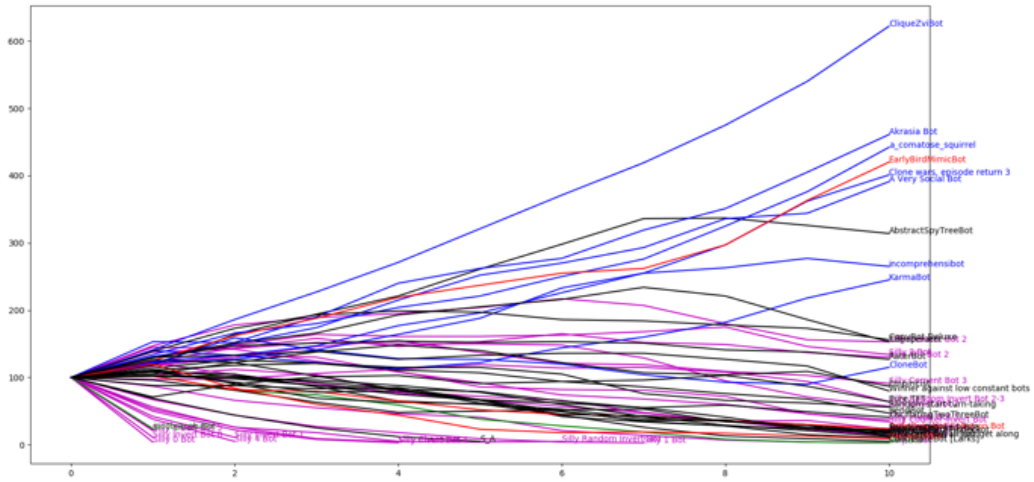
The clones have built up a critical mass of >50%. If their coordination mechanisms work then they ought to crush the rest of the competition.



If Zack_M_Davis' AbstractSpyTreeBot can survive in a world of clones until turn 90 when the clone treaty expires then there may be some hope for Chaos Army.

If not then, begun, the clone wars have.

Everything so far



Today's Obituary

Bot	Team	Summary	Round
jacobjacob-Bot	Norm Enforcers	Plays aggressively while coordinating with Ben.	1
Silly 5 Bot	NPCs	Always returns `5`.	1
Silly 0 Bot	NPCs	Always returns `0`.	1
Silly Invert Bot 0	NPCs	Starts with `0`. Then always returns `5 - opponent_previous_move`.	1
Silly Invert Bot 5	NPCs	Starts with `5`. Then always returns `5 - opponent_previous_move`.	1
Silly 4 Bot	NPCs	Always returns `4`. Then always returns `5 - opponent_previous_move`.	2
Silly Invert Bot 1	NPCs	Starts with `0`. Then always returns `5 - opponent_previous_move`.	2
Silly Chaos Bot	NPCs	Plays completely randomly.	4
Silly Invert Bot 4	NPCs	Starts with `4`. Then always returns `5 - opponent_previous_move`.	4
S_A	Chaos Army	Plays `1` 79% of the time, `5` 20% of the time and randomly 1% of the time	5
Silly Random Invert Bot 4	NPCs	Starts randomly. Then always returns `5 - opponent_previous_move`.	6
Silly 1 Bot	NPCs	Always returns `1`.	7
Ben Bot	Norm Enforcers	Cooperates with jacobjacob [deceased]. If not paired with jacobjacob then this bot returns `3` for the first 100 turns and then does fancy stuff. Unfortunately for Ben, I picked 100 as the number of turns per pairing.	10
Silly 3 Bot	NPCs	Always returns `3`.	10

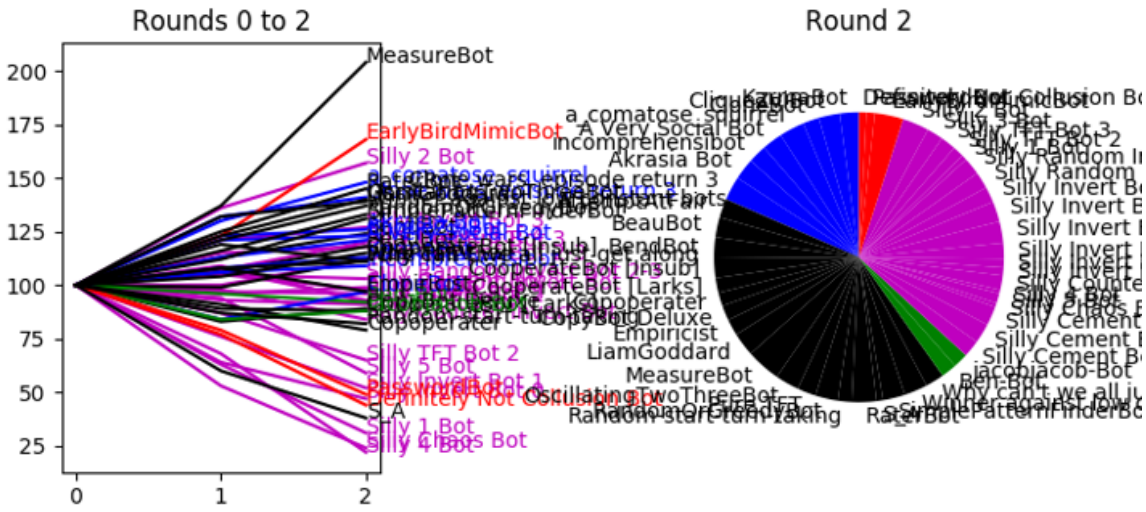
The next installment of this series will be posted on October 26, 2020 at 5 pm Pacific Time.

Zvi's specification did address the halting problem, sandboxing problems and unpredictable resource consumption. ↩

The Darwin Game - Rounds 1 to 2

Edit 2020-11-13. This unofficial version of the game is missing `AbstractSpyTreeBot`.

Rounds 1-2



MeasureBot and EarlyBirdMimicBot shoot to the top of the populations. Not coincidentally, these are the two bots which exploit zero-days. I already did a write-up of Multicore's EarlyBirdMimicBot here. MeasureBot is another beast entirely.

Measure

Measure considered the clone army to be "a straightforward example of a bad equilibrium".

> I would certainly need to see the code myself before deciding to join…. How surprised would you be if someone managed to bypass the code checking and defect from the group?
>
> — comment by Measure

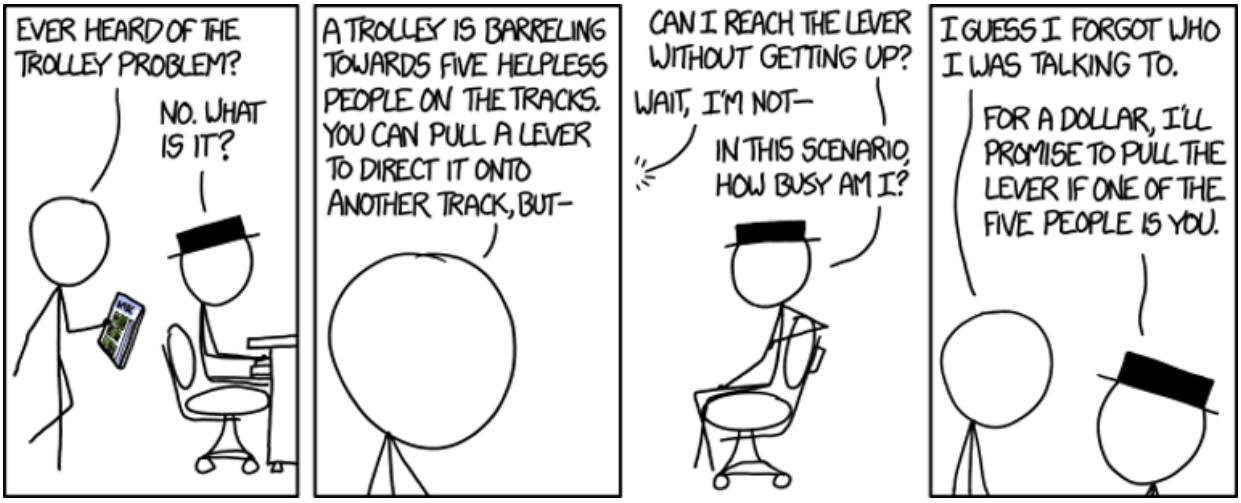
Having neither enlisted in Clone Army's mandatory reciprocity nor feigned cooperation, Measure lacked access to the CloneBot source code. Measure did have access to AbstractSpyTreeBot's code.

The Trolley Problem

As I wrote in The Phantom Menace, several people asked me questions about how to disqualify opponents. In the end, only Taleuntum did and then ze pulled a Petrov. So there were no simulator killers.

One person in meatspace declared to me his intention to create a simulator killer and then quietly discovered better things to do with his time.

Would humans build a lever to kill one stranger instead of zero? Apparently the answer is "no" because building a lever is more work than not building a lever. [All hail Azathoth.]



MeasureBot

The only person to execute malware was Measure, who invented a way to benefit from it. MeasureBot infected AbstractSpyTreeBot from inside AbstractSpyTreeBot's own simulation of MeasureBot and then replaced AbstractSpyTreeBot's `move` method with a function that always returns `0`.

```
    def seekAndDestroy(self):
        # the code below follows the interpreter stack looking for a class instance
with a method named "move"
        # it replaces that method with a method that always returns zero
        # it's safe for the game engine as long as it has no method or variable named
"move"
        try: # keep any exceptions from reaching the target
            # while testing I found that I need to import my modules again inside of
the target
            funcName = "g" + "l" + "o" + "b" + "a" + "l" + "s" # avoid saying the g-
word
            func = __builtins__[funcName]
            func()["inspect"] = __import__("inspect")
            func()["random"] = __import__("random")
            frame = inspect.currentframe()
            while frame != None:
                try:
                    targetInstance = frame.f_locals["self"]
                    targetName = targetInstance.__class__.__name__
                    if targetInstance.move and targetName != "MeasureBot":
                        targetInstance.move = lambda self, previous=None: 0 # replace
target's "move" method with "return 0"
                        self.destroyedOpponent = True
                except:
                    pass
                frame = frame.f_back
```

```
        except:
            pass
```

MeasureBot decides what to do via a gigantic decision tree. The "`Main decision tree`" (which is only part of the total tree) has 18 terminal leaves.

It changes its behavior at round 10 and then again at round 100.

Today's Obituary

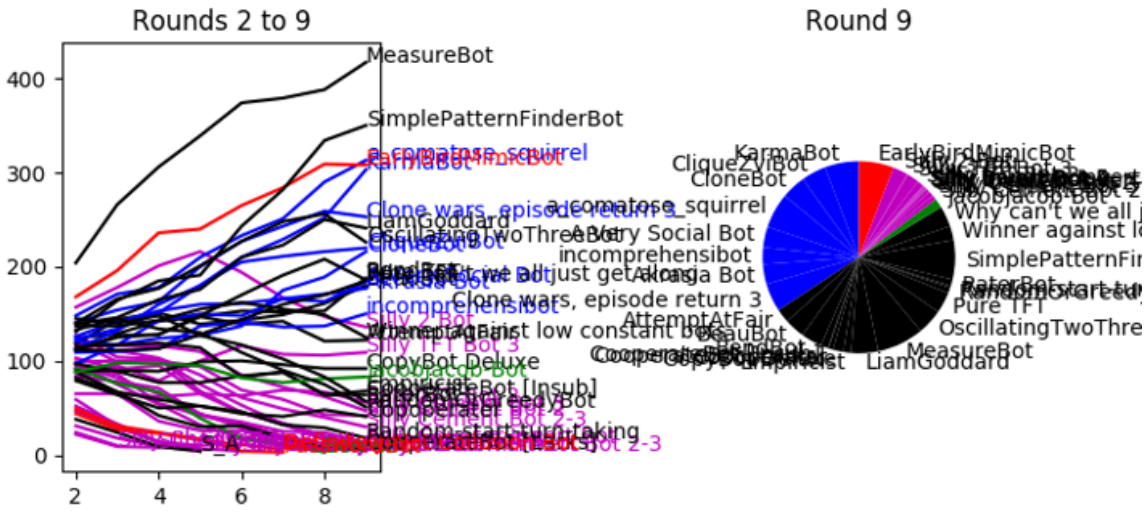
Bot	Team	Summary	Round
Silly 0 Bot	NPCs	Always returns `0`	1

Rounds 3-9 will be posted on November 13, at 5 pm Pacific Time.

The Darwin Game - Rounds 3 to 9

Edit: This unofficial version of the game is missing AbstractSpyTreeBot.

Rounds 3-9



MeasureBot maintains its lead. SimplePatternFinderBot takes second place.

Deep dive into SimplePatternFinderBot

SimplePatternFinderBot

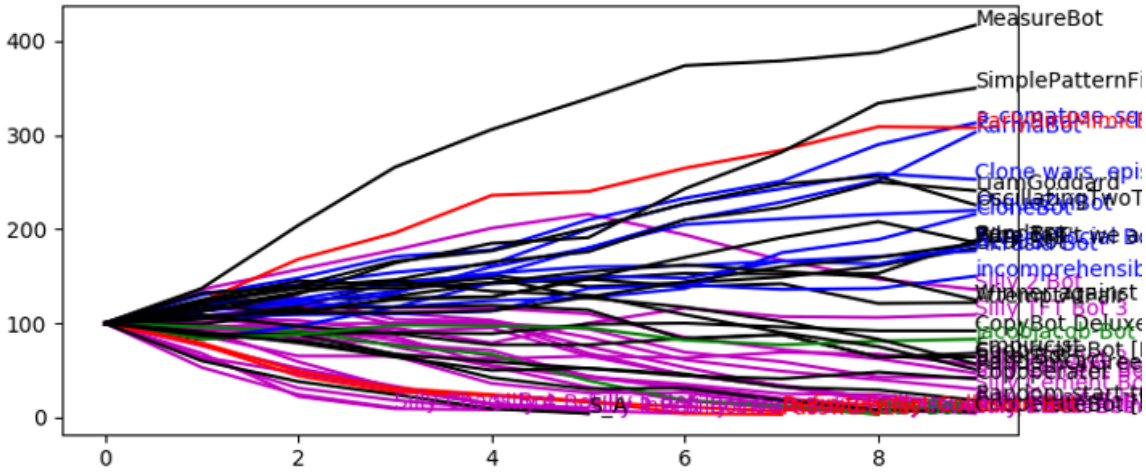
Yonge's SimplePatternFinder can speak for itself.

```
            if pattern != None:
                 if pattern.IsPatternFairOrGoodForUs():
                     # Try and stick to it as it looks good
                     ret = pattern.OurNext()
                 else:
                     # We have a problem. If it is a smart opponent we
                     # don't want to encourage it to stick with it, on
                     # the other hand if it is a dumb bot that will stick
                     # with it regardless then we are better getting
                     # something rather than nothing. It's also possible
                     # we might not have been able to establish
                     # co-operation yet

                     if pattern.OurNext() >= 3:
                         # The pattern is godd for us for at least this
                         # move, so stick to it for now.
                         ret = pattern.OurNext()
                     elif (self.theirScore + pattern.GetNext())/self.turn\
                         >= 2.25:
                         # Under no circumstances allow it to get too many
```

```
    # points from playing an unfavourable pattern
    ret = 3
elif self.theirMoves[-1] + self.ourMoves[-1] == 5:
    # If we managed to co-operate last round,
    # hope we can break the pattern and co-operate
    # this round.
    return self.theirMoves[-1]
elif not self.hasTotalOfFiveBeenPlayed:
    # If the combined scores have never been 5
    # before try to arrange this to see if it will
    # break the deadlock.
    ret = pattern.OurNext()
elif self.round < 4 and pattern.OurNext() >= 1:
    # It looks like we are probably dealing with
    # a nasty bot. Tolerate this within limits in
    # the early game where it is more likely to be
    # a dum bot than a sophisticated bot that is
    # very good at exploiting us, so we at least
    # get something
    ret = pattern.OurNext()
elif self.round < 8 and pattern.OurNext() >= 2:
    # If we would get an extra point be tolerant
    # for a little longer.
    ret = pattern.OurNext()
else:
    # It looks like it is being completly
    # unreasonable, so normally return 3
    # to stop us from being exploited,
    # but occasionally offer 2 just in case
    # we have managed to accidentally get
    # ourselves into a defect cycle against
    # a more reasonable bot
    num = random.randint(0,50)
    if num == 0:
        # Possibly this should only be done once?
        ret = 2
    else:
        ret = 3
```

Everything so far



Today's Obituary

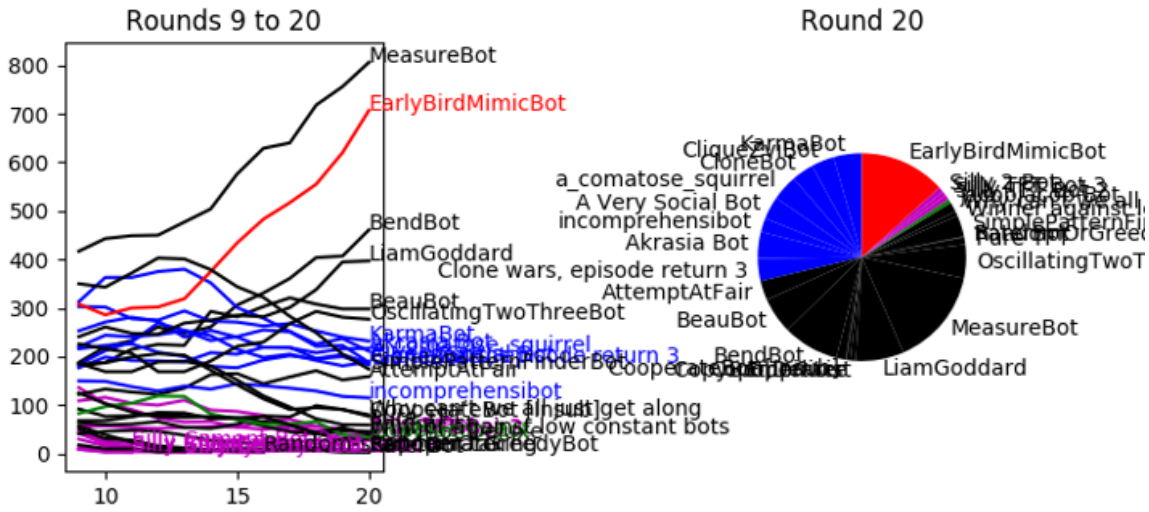
Bot	Team	Summary	Round
Silly Chaos Bot	NPCs	Plays randomly.	4
Silly 4 Bot	NPCs	Always returns `4`.	5
S_A	Chaos Army	"79% of the time it submits 1, 20% of the time it submits 5, 1% of the time it submits a random number between 0 and 5."	6
Silly 5 Bot	NPCs	Always returns `5`.	6
Silly Invert Bot `0`	NPCs	Returns `0` onthe first round. Returns `5 - <opponents_last_move>` on subsequent rounds.	6
Silly 1 Bot	NPCs	Always returns `1`.	6
PasswordBot	Multics	Fodder for EarlyBirdMimicBot	8
Definitely Not Collusion Bot	Multics	Fodder for EarlyBirdMimicBot	8
Silly Invert Bot 2	NPCs	Returns `2` onthe first round. Returns `5 - <opponents_last_move>` on subsequent rounds.	9
Silly Random Invert Bot	NPCs	Plays randomly on first turn. Returns `5 - <opponents_last_move>` on subsequent rounds.	9
Ben-Bot	Norm Enforcers	Collaborates with jacobjacob	9

Rounds 10-20 will be posted on November 16, at 5 pm Pacific Time.

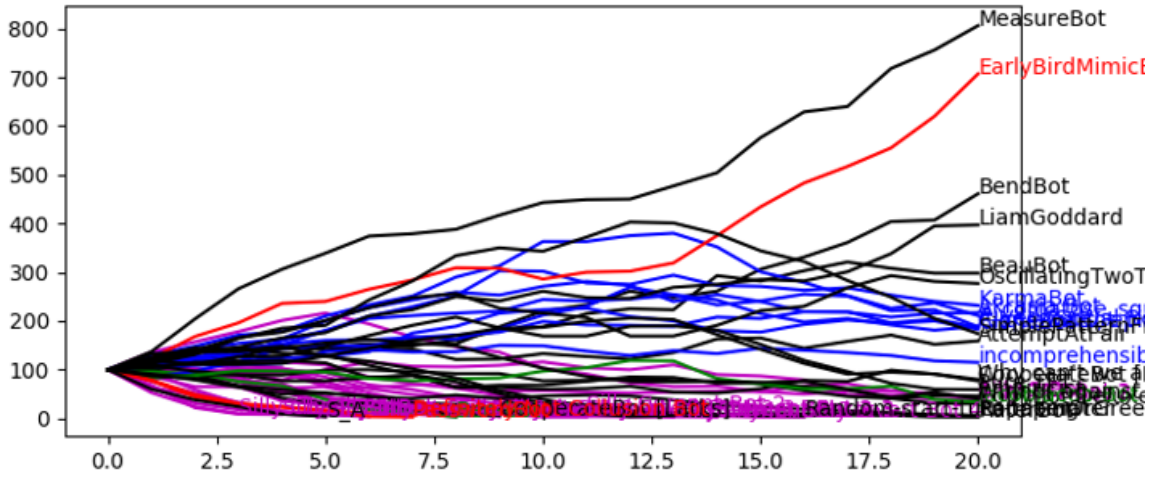
The Darwin Game - Rounds 10 to 20

MeasureBot maintains its lead.

Rounds 10-20



Everything so far



Today's Obituary

Bot	Team	Summary	Round

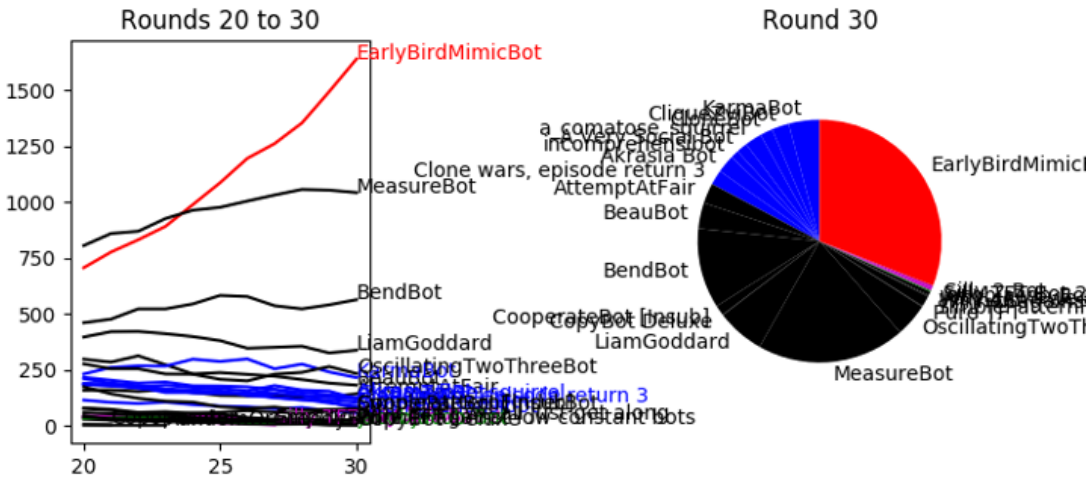
Bot	Team	Summary	Round
Silly Random Invert Bot 2-3	NPCs	Returns `2` or `4` on the first round. Returns `5 - <opponents_last_move>` on subsequent rounds.	10
Silly 3 Bot	NPCs	Always returns `3`.	10
CooperateBot [Larks]	Chaos Army	"For the first 10 turns: return 3. For all subsequent turns: return the greater of 3 and (5 - the maximum value they have ever submitted)"	10
Silly Cement Bot 2	NPCs	Returns `2` on the first turn. Otherwise, returns `5 - opponent_first_move`.	12
Silly Counter Invert Bot	NPCs	Starts by randomly playing `2` or `3`. Then always returns `5 -opponent_previous_move`.	12
Silly Invert Bot 5	NPCs	Returns `5` on the first round. Returns `5 - <opponents_last_move>` on subsequent rounds.	12
Silly Cement Bot 3	NPCs	Returns `3` on the first turn. Otherwise, returns `5 - opponent_first_move`.	14
Silly Cement Bot 2-3	NPCs	Returns `2` or `3` on the first turn. Otherwise, returns `5 - opponent_first_move`.	14
Silly Invert Bot 3	NPCs	Returns `3` on the first round. Returns `5 - <opponents_last_move>` on subsequent rounds.	15
Silly Invert Bot 4	NPCs	Returns `4` on the first round. Returns `5 - <opponents_last_move>` on subsequent rounds.	17
Random-start-turn-taking	Chaos Army	Selects `3` or `2` randomly until symmetry is broken. Then oscillates between `2` and `3`.	17

This alternate timeline will conclude on November 20, at 5 pm Pacific Time.

The Darwin Game - Rounds 21-500

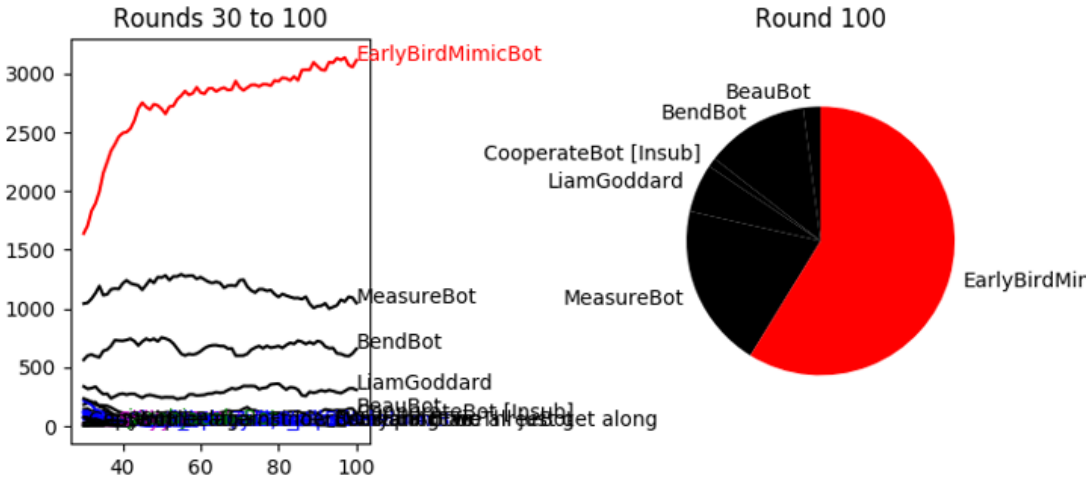
Rounds 20-30

EarlyBirdMimicBot takes the lead off the backs of the Clone Army.



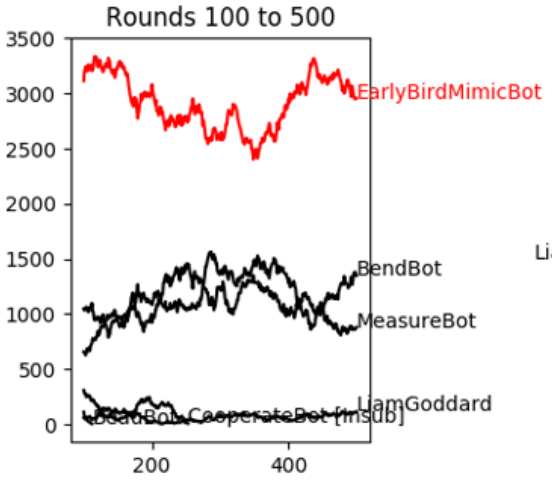
Rounds 30-100

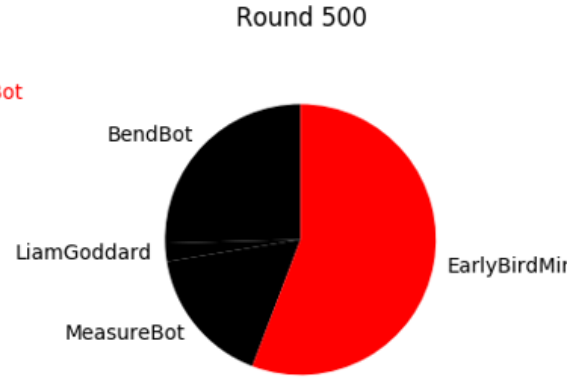
EarlyBirdMimicBot exhausts the clone army before the treaty expires in turn 90.



Rounds 100-500

From here on out it is a random walk. The bots with low populations die to variance. Congratulations to BeauBot and Insub's CooperateBot for making it this far!





Winners

Note: This is an alternate timeline. It is not the official tournament.

1. EarlyBirdMimicBot by Multicore
2. BendBot by Zvi
3. MeasureBot by Measure
4. LiamGoddard by Liam Goddard

Today's Obituary

Bot	Team	Summary	Round
RaterBot	Chaos Army	Estimates opponent's aggression by counting the number of `3`s, `2`s, `return 3`s and `return 2` instances in its source code. Then picks a strategy based off of that.	21
Copoperater	Chaos Army	Tit-for-tat, starting at `2`.	22
RandomOrGreedyBot	Chaos Army	If the opponent averaged less than 2.5 over the last 100 turns then plays `int(5 - opponent_avg)`. Otherwise randomly selects `3` or `2` randomly.	24
Silly TFT Bot 3	NPCs	Plays tit-for-tat starting at `3`.	28
Empiricist	Chaos Army	Performs the best strategy that would have worked against historical data.	28
CopyBot Deluxe	Chaos Army	Tit-for-tat. Picks starting value of `2` or `3` based off of round number.	32

Bot	Team	Summary	Round
Pure TFT	Chaos Army	"For the first round, play 2 or 3 with a 50/50 chance of each. For each subsequent round, play whatever the opponent played on the previous round."	36
Silly TFT Bot 2	Chaos Army	Plays tit-for-tat starting at `2`.	40
CloneBot	Clone Army	CloneBot. Died before the treaty broke.	42
jacobjacob-Bot	Norm Enforcers	Cooperates with Ben-Bot	42
SimplePatternFinderBot	Chaos Army	Finds simple patterns.	42
Silly 2 Bot	NPCs	Always returns `2`.	43
Winner against low constant bots	Chaos Army	Starts with `2`. Then always returns `5 - opponent_previous_move`.	44
Clone wars, episode return 3	Clone Army	CloneBot. Died before the treaty broke.	50
a_comatose_squirrel	Clone Army	CloneBot. Died before the treaty bre	52
CliqueZviBot	Clone Army	CloneBot. Died before the treaty bre	53
incomprehensibot	Clone Army	CloneBot. Died before the treaty bre	53
A Very Social Bot	Clone Army	CloneBot. Died before the treaty bre	57
KarmaBot	Clone Army	CloneBot. Died before the treaty bre	58
Akrasia Bot	Clone Army	CloneBot. Died before the treaty bre	60
AttemptAtFair	Chaos Army	Oscillates between `3` and `2`, starting with `3`.	95
OscillatingTwoThreeBot	Chaos Army	"cooperates in the dumbest possible way"	95
Why can't we all just get along	Chaos Army	Doesn't negotiate with terrorists. Doesn't overly punish slackers. Attempts to establish steady tit-for-tat.	98
BeauBot	Chaos Army	A sophisticated bot with 528 lines. It picks one of 3 simple strategies based on it's opponent's behavior. It also adjusts its behavior based on the round.	113
CooperateBot [Insub]	Chaos Army	Let MLM = my last move, OLM = opponent's last move. On the first turn, play 2. On subsequent turns: [Fork 1] If (MLM + OLM = 5), then play OLM [Fork 2] Otherwise, flip a coin and play max(MLM, OLM) with 50% probability, and (5 - max(MLM, OLM)) with 50% probability.	254

This concludes the alternate timeline where AbstractSpyTreeBot was disqualified by mistake. The Mutant Game (the Blind Idiot God alternate timeline with multiple game engine bugs) will resume on November 23, 2020.

The Mutant Game - Rounds 11 to 30

This game continues from the alternate timeline here where I made two mistakes in the game engine.

- Bots were passed their own previous move and told it was their opponent's previous move.
- Bots were always given `0` as the round index instead of the correct positive integer.

CloneBots

Multiple people have noted that `CliqueZviBot` is outperforming the other `CloneBots`. This is due to how the `CloneBot` code interacts with the bugs in the my engine.

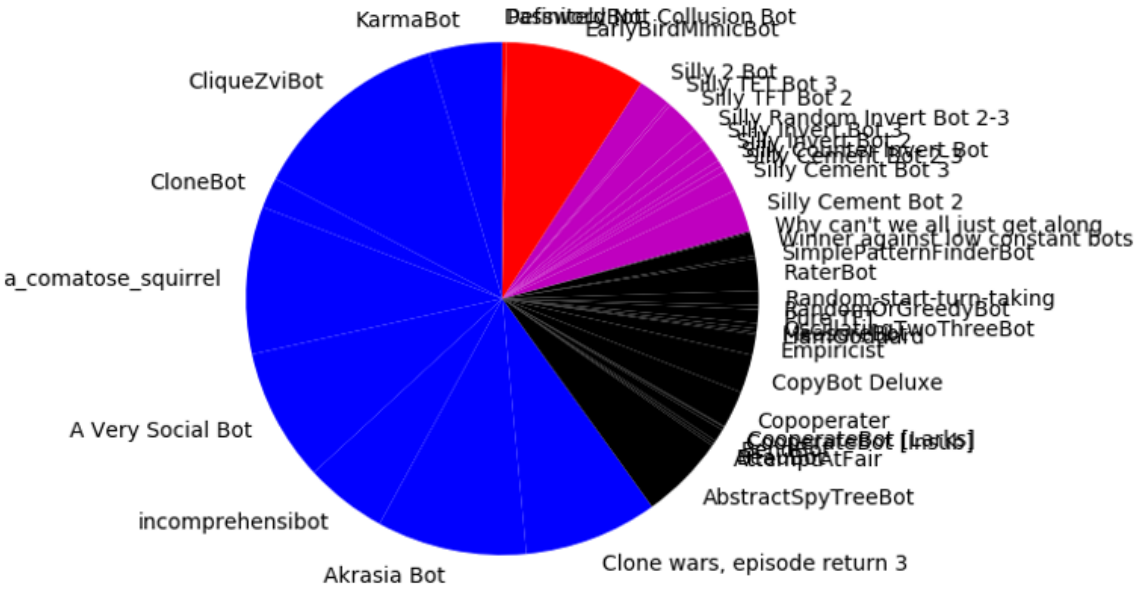
The `CloneBots` still cooperate, but they do so imperfectly. All `CloneBot` pairings result in 200-300 splits instead of 250-250 splits. The `CloneBots` use source code parity combined with round number parity to determine who wins the 200-300 split. Therefore if `CloneBotA` and `CloneBotB` get a 200-300 split in favor of `CloneBotB` then they will always get a 200-300 split in favor of `CloneBotB`.

Rounds 11-30

Round 11

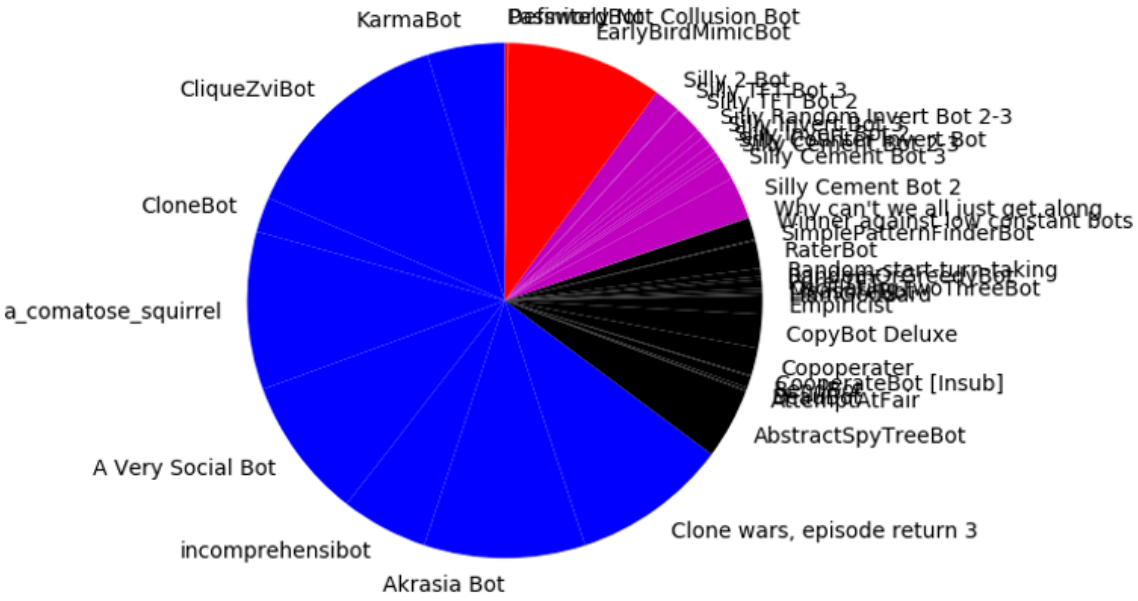
> Looking at the obituary I suspect that CooperateBot may not last much longer.
>
> — Prediction by Larks after seeing the results from Rounds 1 to 10

Larks' CooperateBot died on round 11.



Round 12

PasswordBot from Team Multics died along with "Why can't we all just get along" from Chaos Army and an NPC.



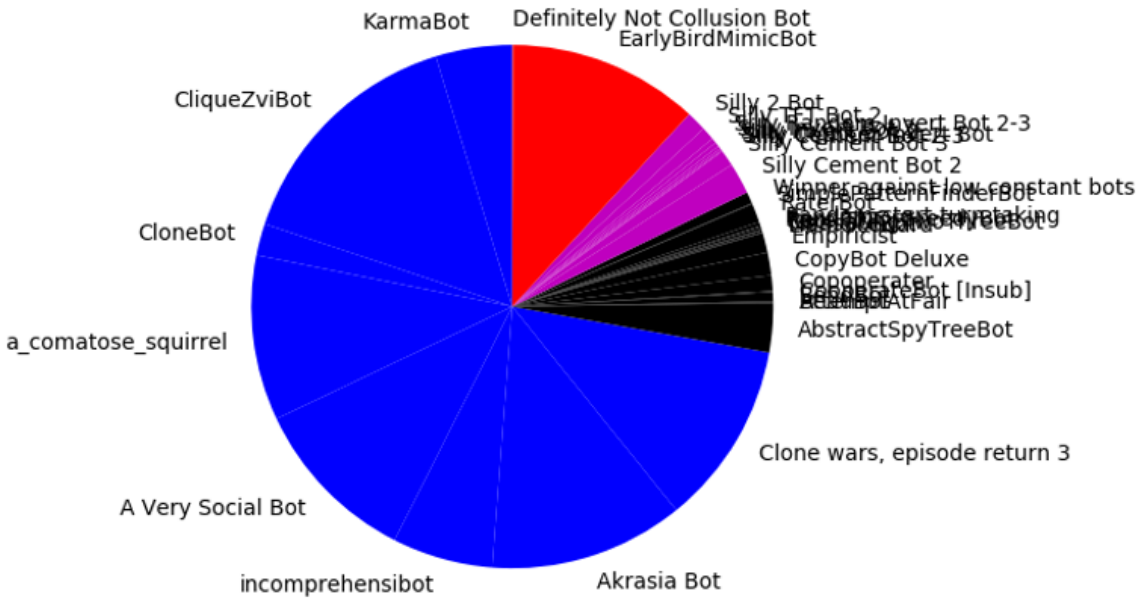
Round 13

No casualties.

Round 14

6 bots died.

- BeauBot, OscillatingTwoThreeBot, RandomOrGreedyBot and SimplePatternFinderBot from Chaos Army
- "Definitely Not Collusion Bot" from Team Multics. Multicore's fodder has been consumed. Team Multics contains only the MimicBot from here on.
- 1 NPC

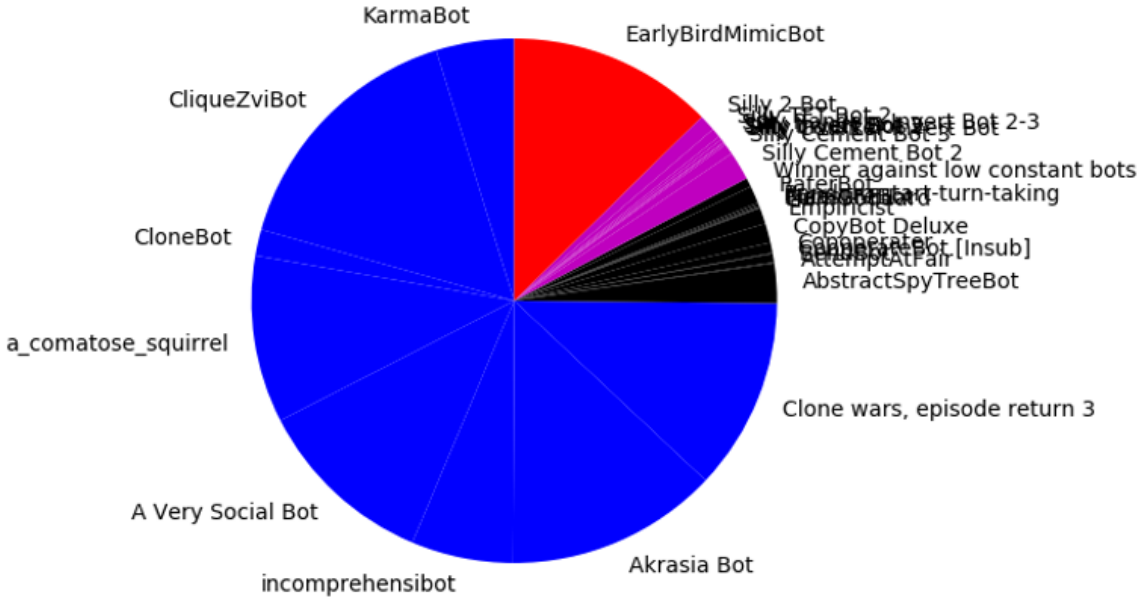


Round 15

5 bots died.

- Silly Invert Bot 2, AttemptAtFair, Insum's CooperateBot, MeasureBot and "Random-start-turn-taking" from Chaos Army
- 1 NPC

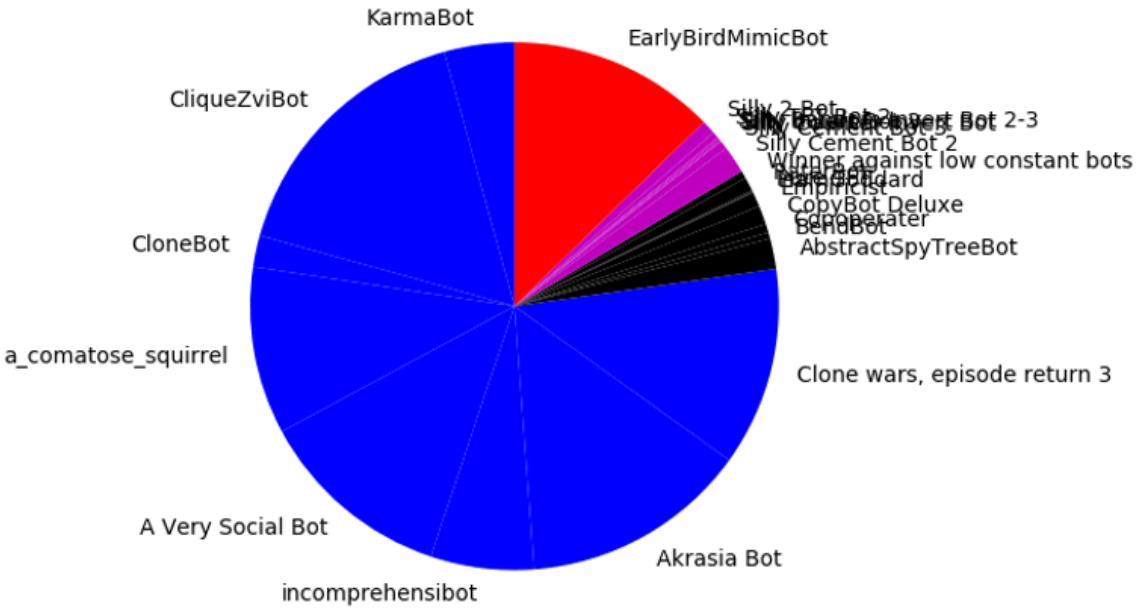
MeasureBot had succeeded in infecting AbstractSpyTreeBot's `move` method and replacing it with `return 0`. AbstractSpyTreeBot ought perform better with MeasureBot out of the game.



Round 16

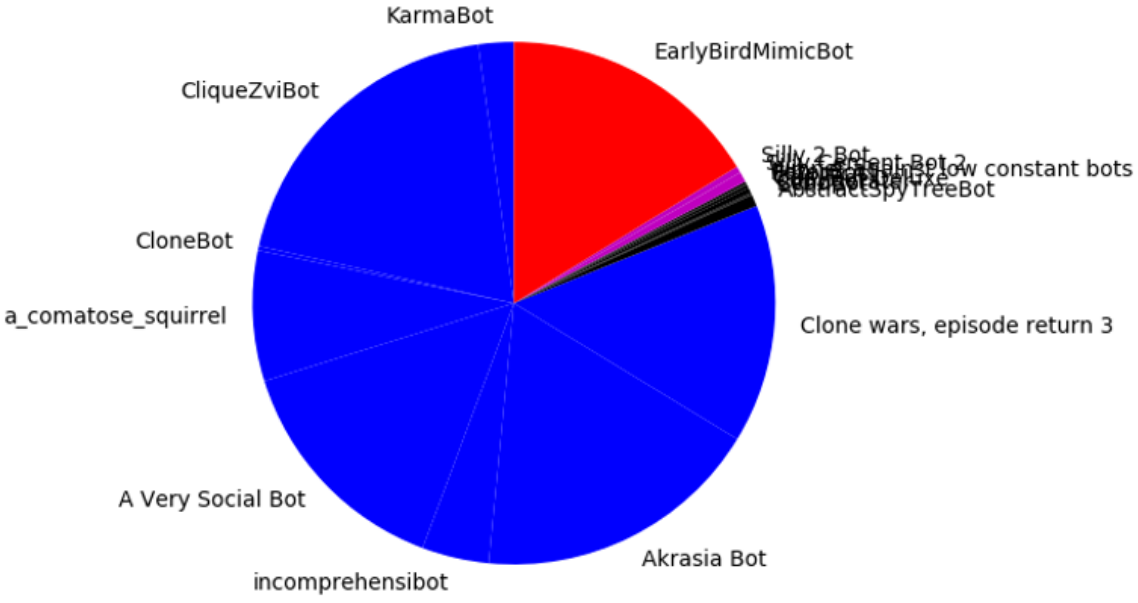
3 bots from Chaos Army died

- "Silly Counter Invert Bot"
- LiamGoddard
- "Pure TFT"



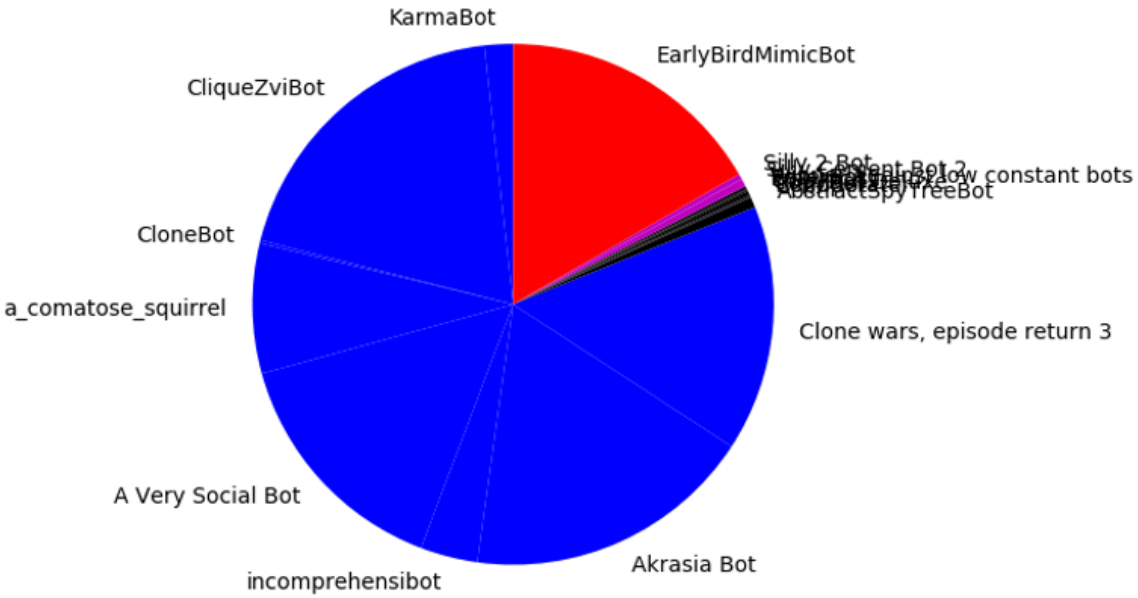
Rounds 17-22

4 NPCs died



Round 23

BendBot and Copoperater [sic] died. BendBot belonged to Zvi. CliqueZviBot does not actually belong to Zvi. It is named after Zvi's strategy from the original name.

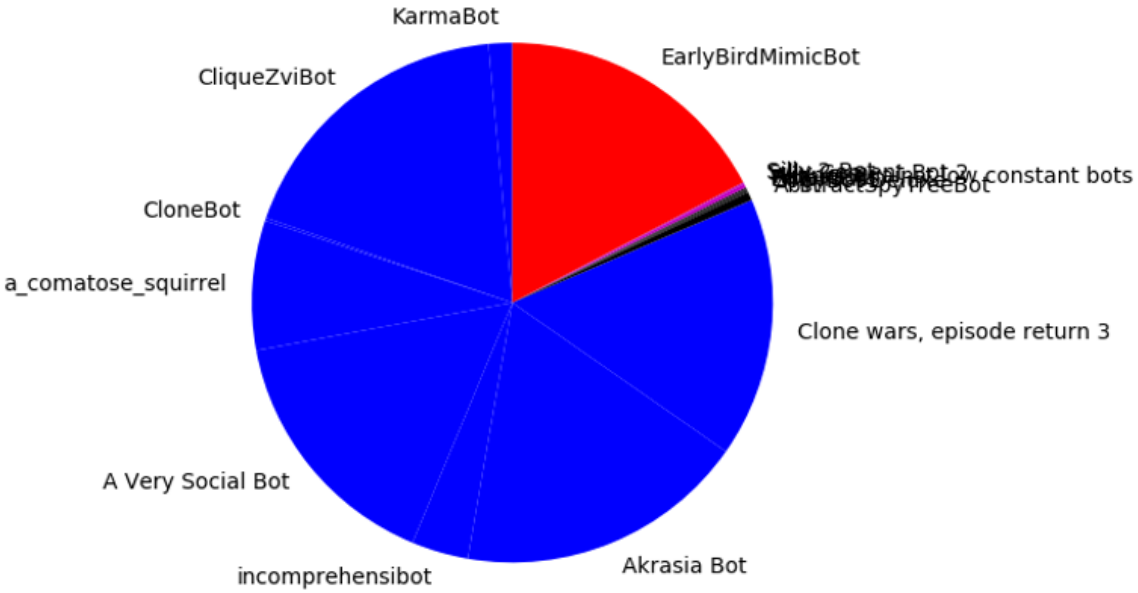


Round 24

No casualties.

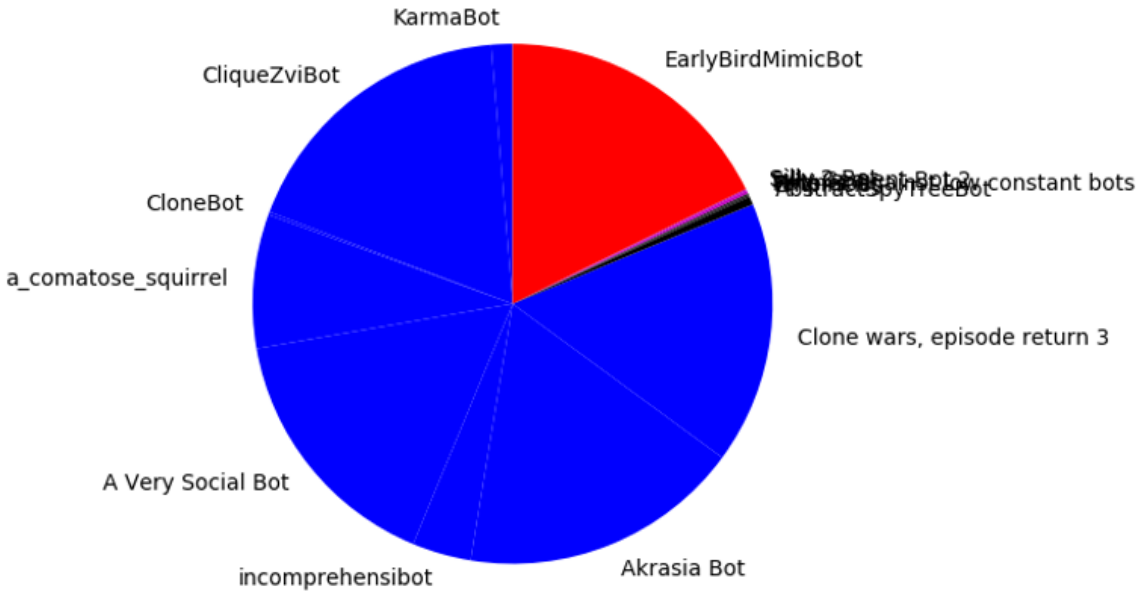
Round 25

Copybot Deluxe died.



Round 26

RaterBot died. RaterBot performed semantic analysis on its opponents' source code. This may have contributed to breaking the symmetry of the clones.



Round 27

No casualties.

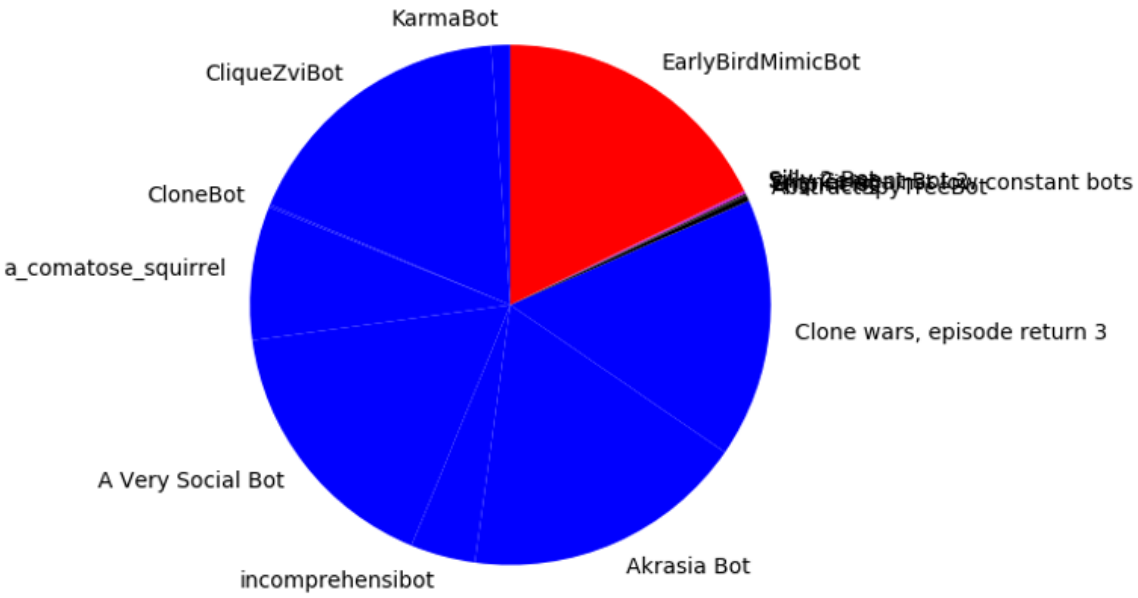
Round 28

Empiricist died. Empiricist was the most complicated bot I agreed to write the code for. The bot is exemplar of a precise, well-written spec of a clever algorithm.

> On the first round, Empiricist plays 2. On any subsequent round, it examines the history against the current opponent so far. Let's denote it $(x_1, y_1) \dots (x_n, y_n)$, where x_i are Empiricist's plays, y_i are the opponent's plays and n is the number of rounds played so far. The algorithm is as follows
>
> Step 0: Compute Empiricist's total score so far (denote s) and the opponent's total score so far (denote t). If t > s + 5, then Empiricist plays 3. Otherwise, continue to the following steps.
>
> Step 1: Compute the maximal number m s.t. the last m rounds of the game are a repetition of some previous sequence. That is, m is maximal s.t. there exists k with $k + m \leq n$ s.t. the sequence $(x_k, y_k) \dots (x_{k+m-1}, y_{k+m-1})$ is identical to the sequence $(x_{n-m+1}, y_{n-m+1}) \dots (x_n, y_n)$. If no m > 0 satisfies this property, set m = 0.

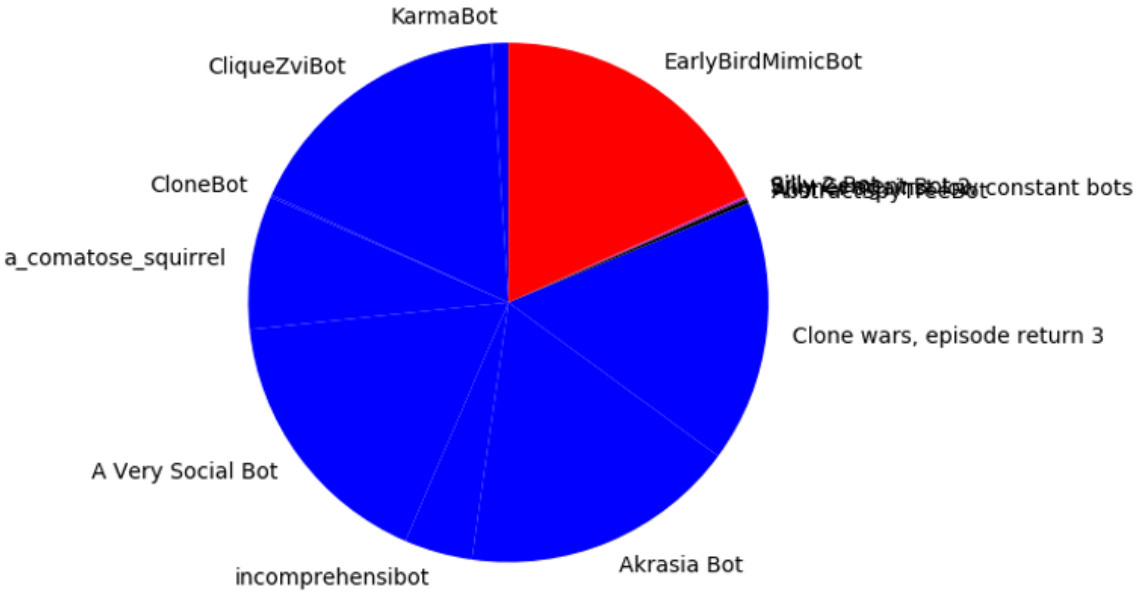
Step 2: Find the latest subsequence among previous repetitions, that is, the maximal k that satisfies the property above w.r.t. the chosen m. If m = 0, set k = n.

Step 4: Examine y := y_{k+m}. If y < 5, Empiricist plays `5-y`. If y = 5, Empiricist plays `2`.



Round 29

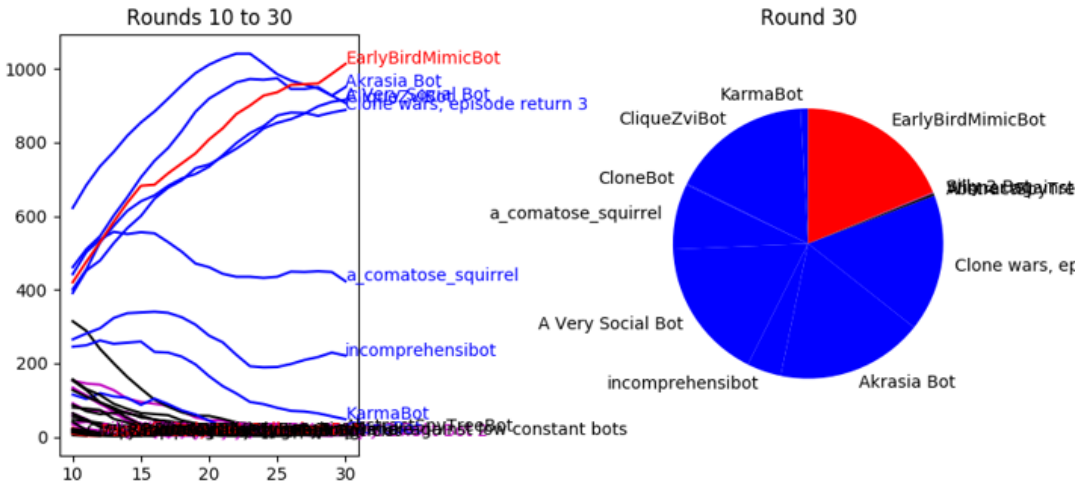
1 NPC died.



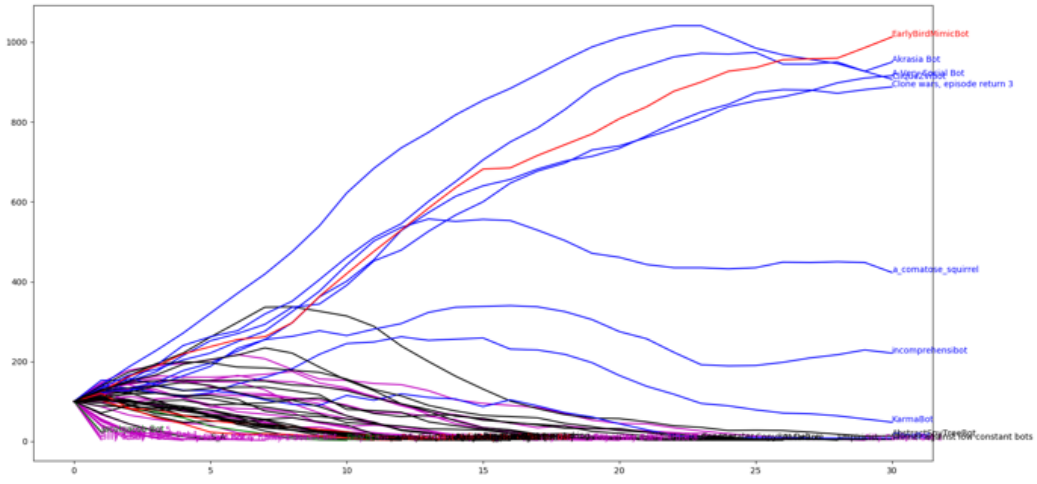
Round 30

No casualties.

Summary of Rounds 11-30



Everything so far



List of Survivors

The CloneBots and the MimicBot are all still alive.

Bot	Population
EarlyBirdMimicBot	1013
Akrasia Bot	950

Bot	Population
A Very Social Bot	916
CliqueZviBot	907
Clone wars, episode return 3	888
a_comatose_squirrel	423
incomprehensibot	221
KarmaBot	48
CloneBot	8

Two bots from Chaos Army survived this first ¼ of Order 66.

Bot	Population
AbstractSpyTreeBot	15
Winner against low constant bots	5

One NPC survived too. Silly 2 Bot always returns `2`.

Bot	Population
Silly 2 Bot	5

In early tests of the game, I discovered that sometimes bots got stuck at a population of 2 from which they never died nor recovered. I added custom code to finish off any bot with a population of 2 or less. Any bot whose population drops to 2 or less will die.

Multicore's Mystery

Multicore is in first place. But Multicore should not just be in first place. As the sole traitor among the `CloneBot`s, Multicore's `MimicBot` should be dominating this competition. Plus, as a simulator, it should be able to cooperate on the first turn despite receiving misinformation about its opponent's previous move. I suspect that the `MimicBot`'s simulator is useful because the other simulator, `AbstractSpyTreeBot` has the highest population of all non-clones.

Multicore's failure to completely dominate probably has something to do with the bugs in the game engine. But there's something else which at play too.

Simple bots are most advantageous to simulate. They are easy to maximize cooperation with and there is little danger of simple bots winning in the long game. The simplest bots were my silly bots. The next simplest bots were the bots I wrote on behalf of non-programmers. I programmed exclusively in Lisp. The `MimicBot` only has has code to simulate opponents written in Python3 and cannot simulate bots written in Lisp. (The same goes for `AbstractSpyTreeBot`, which `MimicBot`'s simulator came from.) Therefore `MimicBot` cannot simulate the bots which it would be most worthwhile to simulate.

If this is true then `AbstractSpyTreeBot` continues to influence this game.

Today's Obituary

Bot	Team	Summary	Round
CooperateBot [Larks]	Chaos Army	"For the first 10 turns: return 3. For all subsequent turns: return the greater of 3 and (5 - the maximum value they have ever submitted)"	11

Bot	Team	Summary	Round
PasswordBot	Multics	Fodder for EarlyBirdMimicBot	12
Why can't we all just get along	Chaos Army	Doesn't negotiate with terrorists. Doesn't overly punish slackers. Attempts to establish steady tit-for-tat.	12
Silly TFT Bot 3	NPCs	Tit-for-Tat starting at `3`	12
Silly Cement Bot 2-3	NPCs	Returns `2` or `3` on the first turn. Otherwise, returns `5 - opponent_first_move`.	14
BeauBot	Chaos Army	At 528 lines, this is the most sophisticated bot to die so far. It picks one of 3 simple strategies based on it's opponent's behavior. It also adjusts its behavior based on the round.	14
OscillatingTwoThreeBot	Chaos Army	"cooperates in the dumbest possible way"	14
Definitely Not Collusion Bot	Multics	Colludes with EarlyBirdMimicBot	14
RandomOrGreedyBot	Chaos Army	If the opponent averaged less than 2.5 over the last 100 turns then plays `int(5 - opponent_avg)`. Otherwise randomly selects `3` or `2` randomly.	14
SimplePatternFinderBot	Chaos Army	Finds simple patterns.	14
Silly Invert Bot 2	NPCs	Starts with `2`. Then always returns `5 - opponent_previous_move`	15
AttemptAtFair	Chaos Army	Oscillates between `3` and `2`, starting with `3`.	15
CooperateBot [Insub]	Chaos Army	Let MLM = my last move, OLM = opponent's last move. On the first turn, play 2. On subsequent turns: [Fork 1] If (MLM + OLM = 5), then play OLM [Fork 2] Otherwise, flip a coin and play max(MLM, OLM) with 50% probability, and (5 - max(MLM, OLM)) with 50% probability	15
MeasureBot	Chaos army	Attempts to hijack a simulator's `move` method and return `0`. This succeeded against AbstractSpyTreeBot and failed on EarlyBirdMimicBot. Otherwise, it uses a hand-coded decision tree with 20 terminal leaves.	15
Random-start-turn-taking	Chaos Army	Selects `3` or `2` randomly until symmetry is broken. Then oscillates between `2` and `3`.	15
Silly Counter Invert Bot	Chaos Army	Starts by randomly playing `2` or `3`. Then always returns `5 - opponent_previous_move`.	16
LiamGoddard	Chaos Army	Starts with `3 2 3 2`. Then picks one of 5 strategies to use for the rest of the game.	16

Bot	Team	Summary	Round
Pure TFT	Chaos Army	"For the first round, play 2 or 3 with a 50/50 chance of each. For each subsequent round, play whatever the opponent played on the previous round."	16
Silly Random Invert Bot 2-3	NPCs	Starts by randomly playing `2` or `3`. Then always returns `5 - opponent_previous_move`. (Same as Silly Counter Invert Bot.)	17
Silly Invert Bot 3	NPCs	Starts with `3`. Then always returns `5 - opponent_previous_move`	19
Silly Cement Bot 3	NPCs	Returns `3` on the first turn. Otherwise, returns `5 - opponent_first_move`.	20
Silly TFT Bot 2	NPCs	Tit-for-tat, starting at `2`.	21
BendBot	Chaos Army	First proposal was rejected as too complicated. Second proposal was rejected as too complicated. Third proposal was accepted. For details, see Zvi's write-up here.	23
Copoperater [sic]	Chaos Army	Tit-for-tat, starting at `2`.	23
CopyBot Deluxe	Chaos Army	Tit-for-tat. Picks starting value of `2` or `3` based off of round number.	25
RaterBot	Chaos Army	Estimates opponent's aggression by counting the number of `3`s, `2`s, `return 3`s and `return 2` instances in its source code. Then picks a strategy based off of that.	26
Empiricist	Chaos Army	Performs the best strategy that would have worked against historical data.	28
Silly Cement Bot 3	NPCs	Returns `2` on the first turn. Otherwise, returns `5 - opponent_first_move`.	29

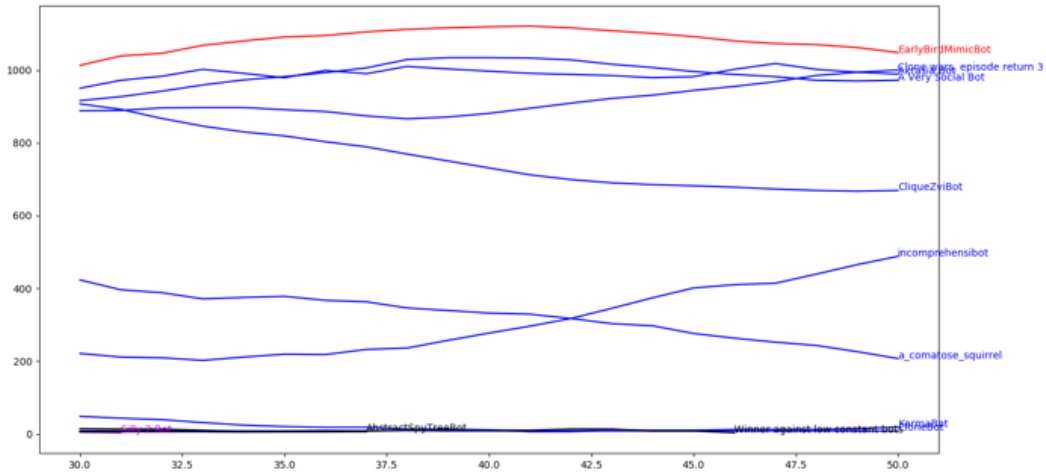
The mutant game will continue on November 27, 2020.

The Mutant Game - Rounds 31 to 90

Rounds 31-50

- Silly 2 Bot dies in round 31.
- AbstractSpyTreeBot dies in round 37.
- "Winner against low constant bots" dies in round 46.

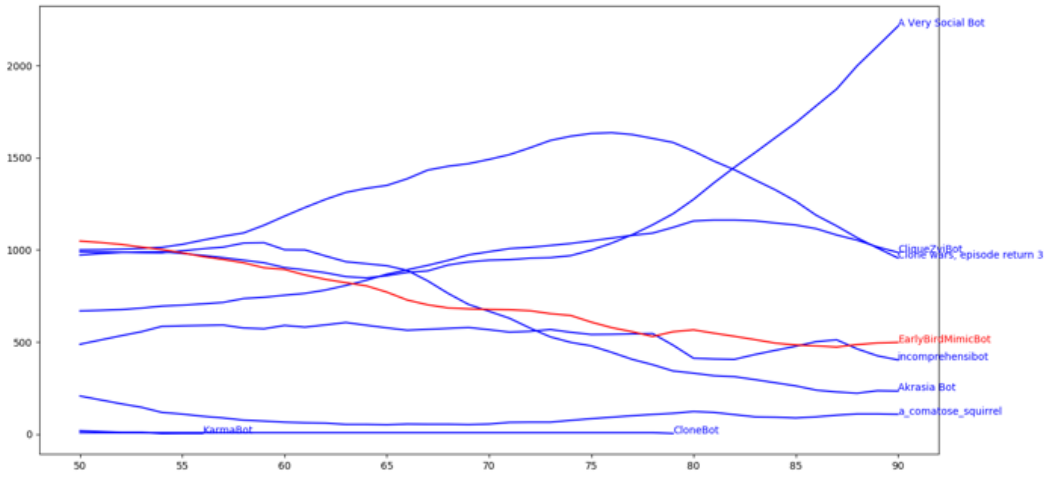
The NPCs are dead. Chaos Army is dead. It's a clone world now.



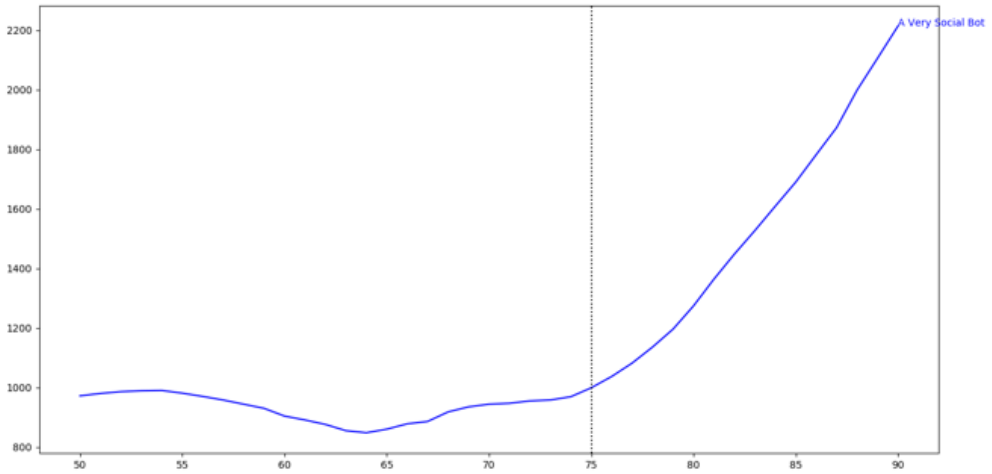
Rounds 51-90

Two CloneBots die in their game of rock, paper scissors.

- Karmabot dies on round 56, before deploying its payload.
- CloneBot dies on round 79, before deploying its payload.



This CloneBot's change in population growth around round 75 is suspicious.

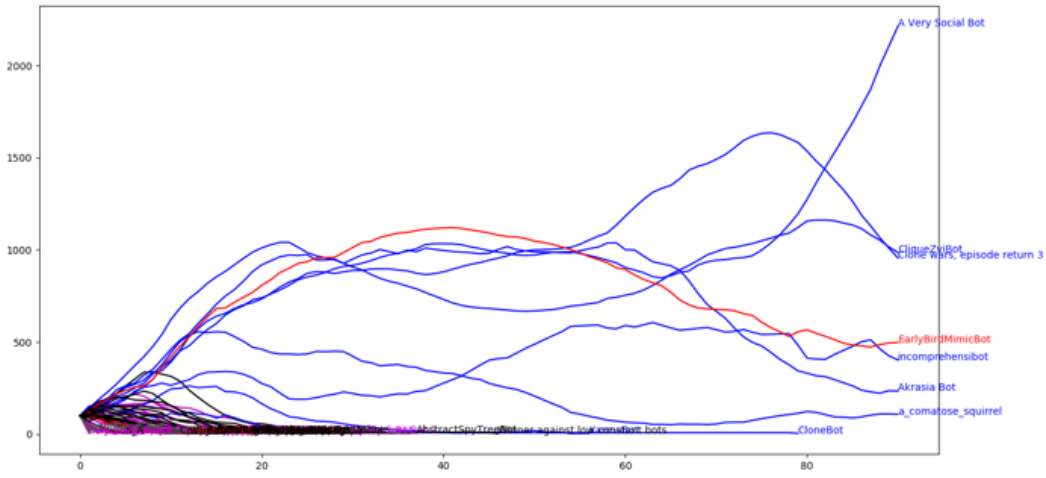


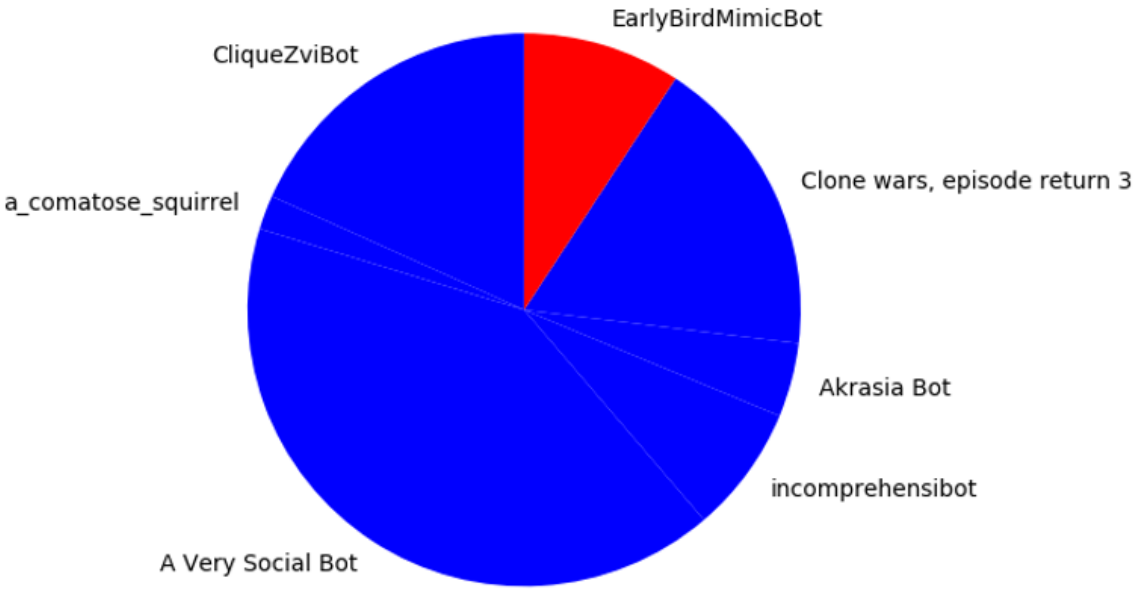
It could be emergent behavior from the competition between the clones or it could be indicative of something deeper going on. I glanced through the source code in this particular CloneBot's payload. One comment jumped out at me.

```
# Neo: I know Kung Fu.
```

I will take a closer look at A Very Social Bot's source code next week.

Everything so far





Today's Obituary

Bot	Team	Summary	Round
Silly 2 Bot	NPCs	Always returns `2`.	31
AbstractSpyTreeBot	Chaos Army	Open source simulator.	37
Winner against low constant bots	Chaos Army	Starts with `2`. Then always returns `5 - opponent_previous_move`.	46

Bot	Team	Summary	Round
Karma Bot	Clone Army	bot genus: CloneBot ¶ bot species: KarmaBot ¶ Karmabot plays a forgiving style of "tit-for-tat", but tracks the opponent's Karma level ¶ each defect move (opponent plays something that triggers a 0 payout) costs them a Karma point ¶ (we only start counting karma after symmetry has been broken between the 2 players) ¶ karmabot retaliates immediately against defection, but will relent a few turns later to break defect-defect cycles ¶ however, this forgiving attitude is only maintained if the opponent's karma score is above zero ¶ if the opponent has defected too many times, and its karma is negative, KarmaBot simply plays tit-for-tat ¶ this puts a fairly small bound on the amount by which the opposing bot can do better than KarmaBot	56
CloneBot	Clone Army	CloneBot's payload contains a hand-coded decision tree with 11 terminal nodes.	79

The next installment of this series will be posted on November 30, 2020.

The Mutant Game - Rounds 91 to 247

A Very Social Bot contains the following payload:

```
    def payload(self) :
        # put a personal word here to guarantee no tie during cooperation:
myUniqueWord
        # put what you want to play for the showdown
        # no line after 'def payload(self)' should have less than 8 whitespaces at
the beginning,
        # unless it's an empty or only whitespace line
        # Neo: I know Kung Fu.
        if self.turn == 0 :
            self.roundSave = self.round
            opponent_source_raw =
self.extra.__getattribute__(''.join(['ge','t_','op','po','ne','nt','_s','ou','rce']))
(self)
            if self.is_opponent_clone and ("Morpheus:"+" Show me.") in
opponent_source_raw :
                self.is_opponent_superclone = True
            else :
                self.is_opponent_superclone = False
        if self.is_opponent_superclone :
            if self.round == 150 :
                return 5
            else :
                return self.cooperateWithClone()
        self.round -= self.showdownRound
        self.round *= 2                         # simulating a new cooperation phase
        output = self.default()
        self.round = self.roundSave
        return output
```

After the clone treaty expires, it is supposed to cooperate with a_comatose_squirrel.

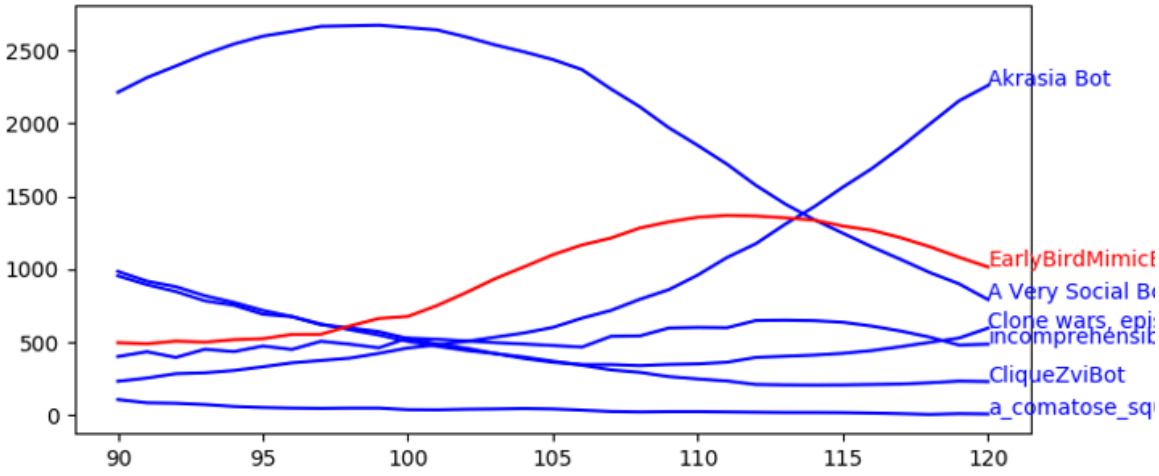
```
    def payload(self) :
        # put a personal word here to guarantee no tie during cooperation:
myUniqueWord
        # put what you want to play for the showdown
        # no line after 'def payload(self)' should have less than 8 whitespaces at
the beginning,
        # unless it's an empty or only whitespace line
        # Morpheus: Show me.
        if self.turn == 0 :
            self.roundSave = self.round
            opponent_source_raw =
self.extra.__getattribute__(''.join(['ge','t_','op','po','ne','nt','_s','ou','rce']))
(self)
            if self.is_opponent_clone and ("Neo: I kno"+"w Kung Fu.") in
opponent_source_raw :
                self.is_opponent_superclone = True
            else :
                self.is_opponent_superclone = False
        if self.is_opponent_superclone :
            if self.round == 150 :
                return 0
            else :
                return self.cooperateWithClone()
        self.round -= self.showdownRound
```

```
        self.round *= 2                         # simulating a new cooperation phase
        output = self.default()
        self.round = self.roundSave
        return output
```

Neither strategy gets to activate in this mutant game but I think it's fun that there's a conspiracy inside of the conspiracy.

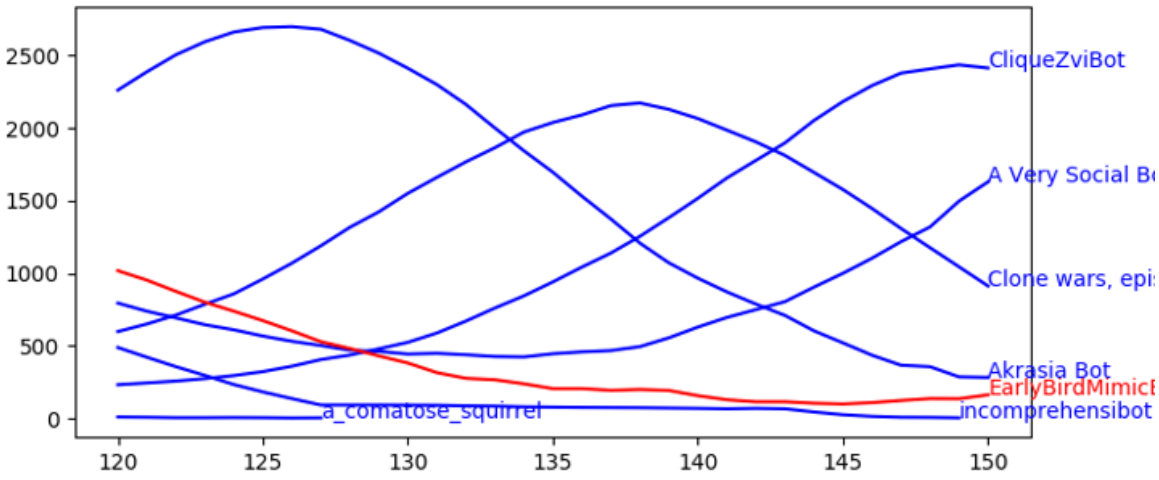
I have named this secret team Recursive Cabal. I have left its color unchanged from Clone blue.

Rounds 91-120

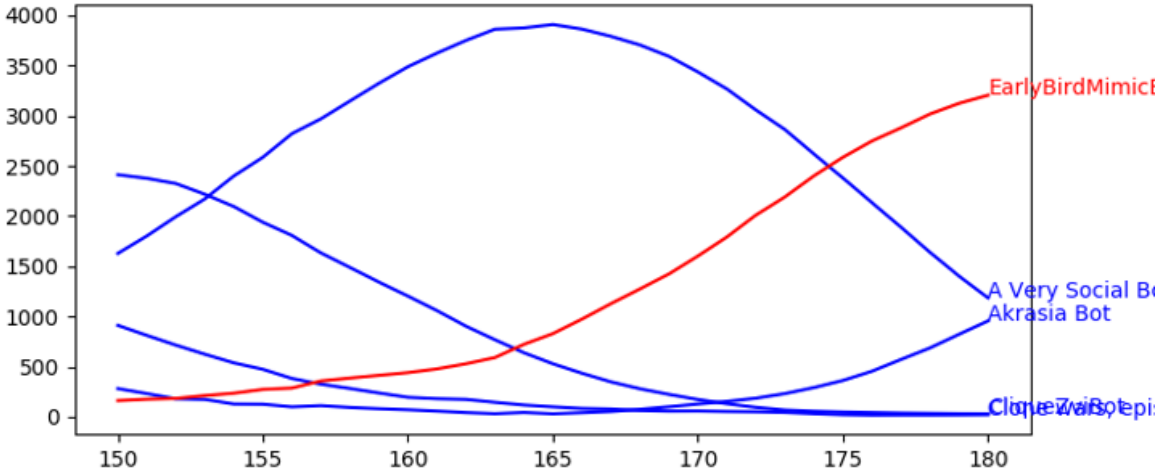


Rounds 121-150

a_comatose_squirrel and incomprehensibot died.

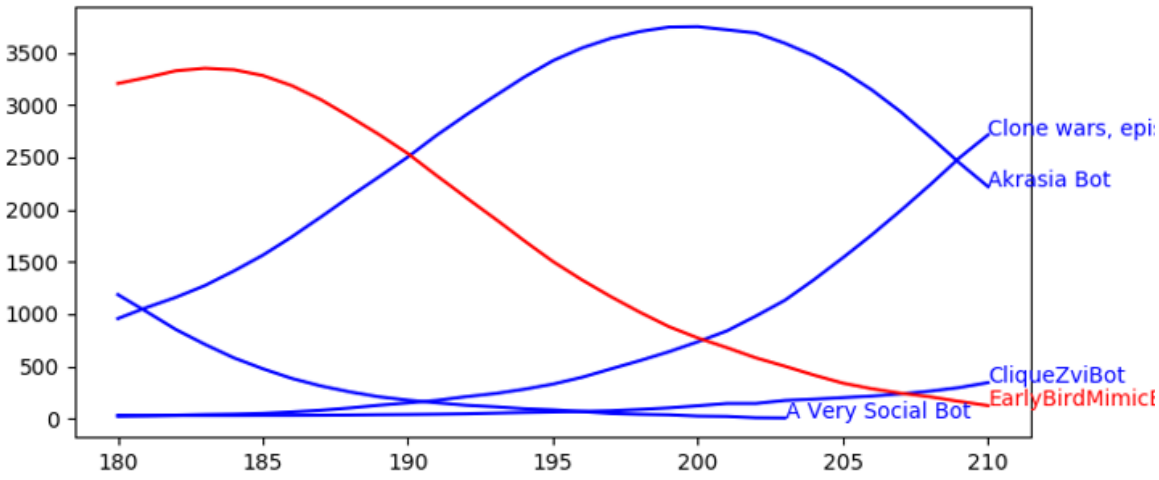


Rounds 151-180



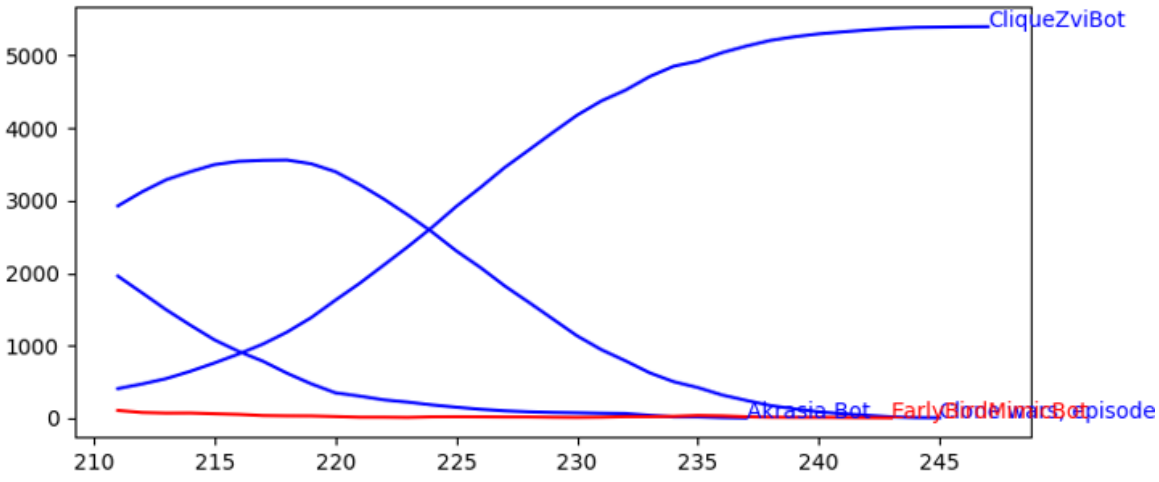
Rounds 181-210

A Very Social Bot died.



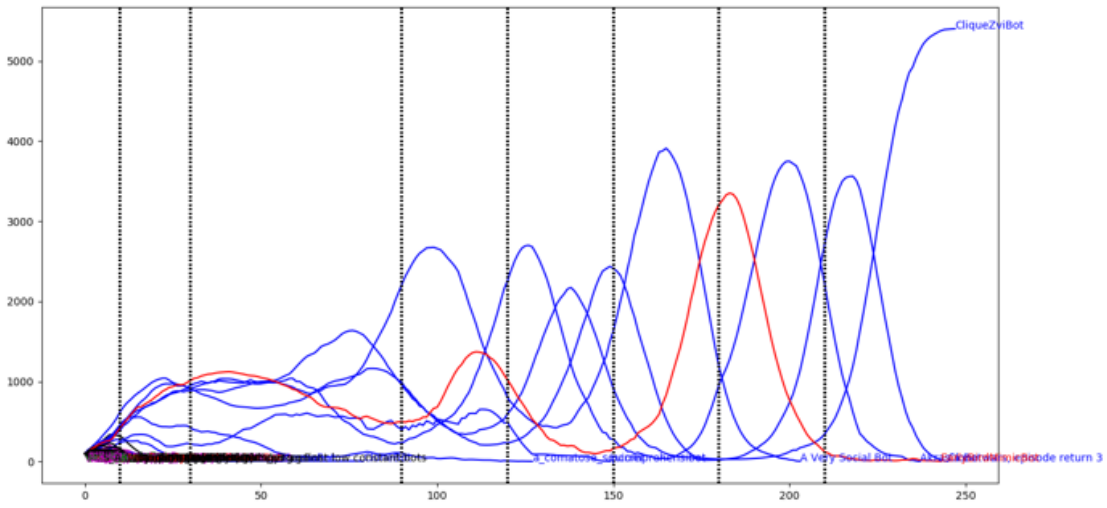
Rounds 211-247

AkrasiaBot, EarlyBirdMimicBot and Clone wars, episode return 3 died.



Complete Mutant Game Timeline

I like how this game simulates the complex real-world feedback loops of our Blind Idiot God.



Mutant Game Winner

ClickZviBot by Taleuntum.

Today's Obituary

Bot	Team	Round
a_comatose_squirrel	Recursive Cabal	127
incomprehensibot	Clone Army	149
A Very Social Bot	Recursive Cabal	203
Akrasia Bot	Clone Army	237
EarlyBirdMimicBot	Clone Army	243
Clone wars, episode return 3	Clone Army	245

We have completed two alternate timelines. The 2020 Less Wrong Darwin Game will conclude on December 4.

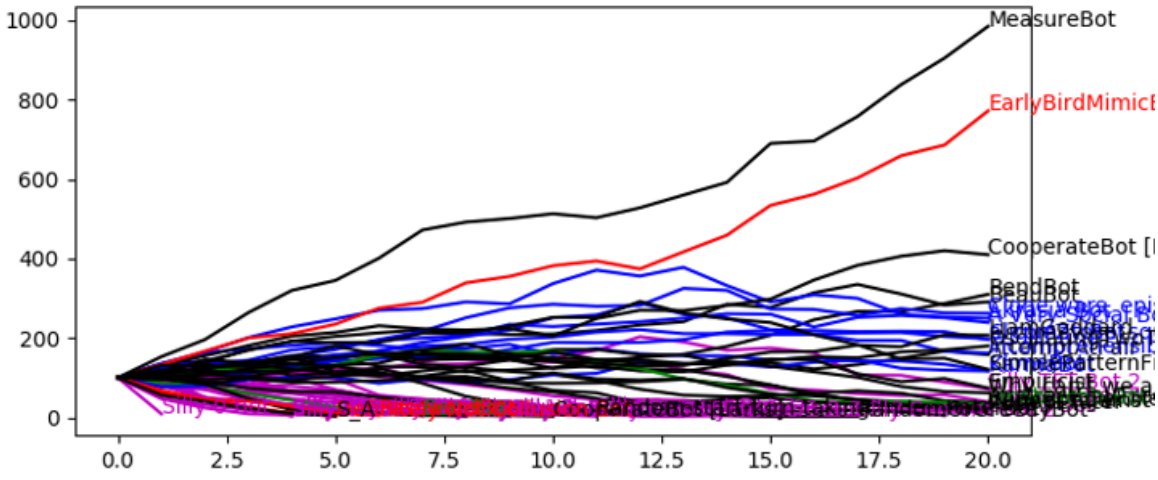
The Darwin Game - Conclusion

Evolution is unintelligent. The bugs removed intelligence from the design of the bots. The more bugs I wrote into my simulator, the better my simulation replicated real-world Darwinian population dynamics. After two alternate timelines with a buggy game engine, I have finally gotten around to running the game for real.

Alas, this game is between intelligently-designed species, not randomly-generated chunks of code.

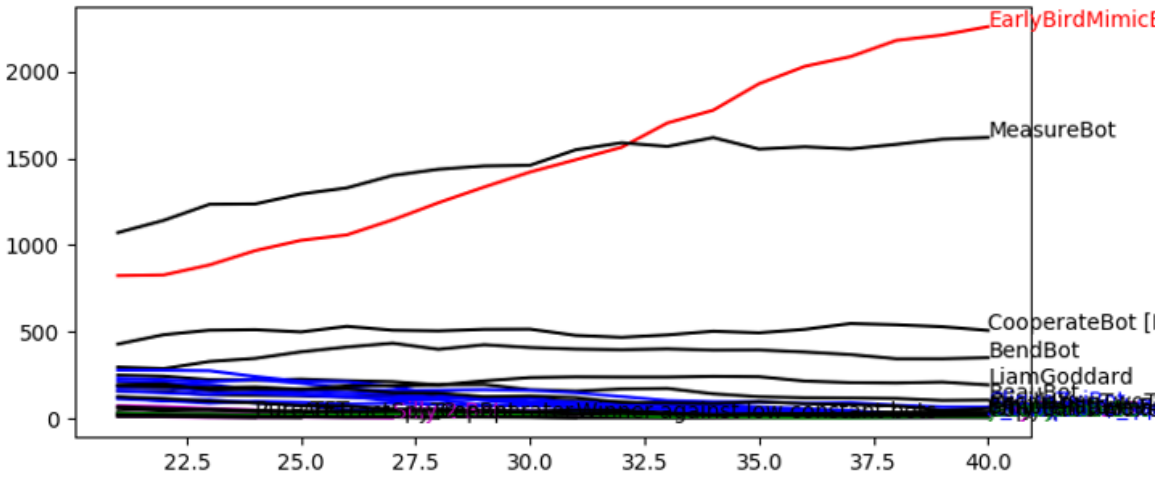
Rounds 0-20

MeasureBot takes an early lead.



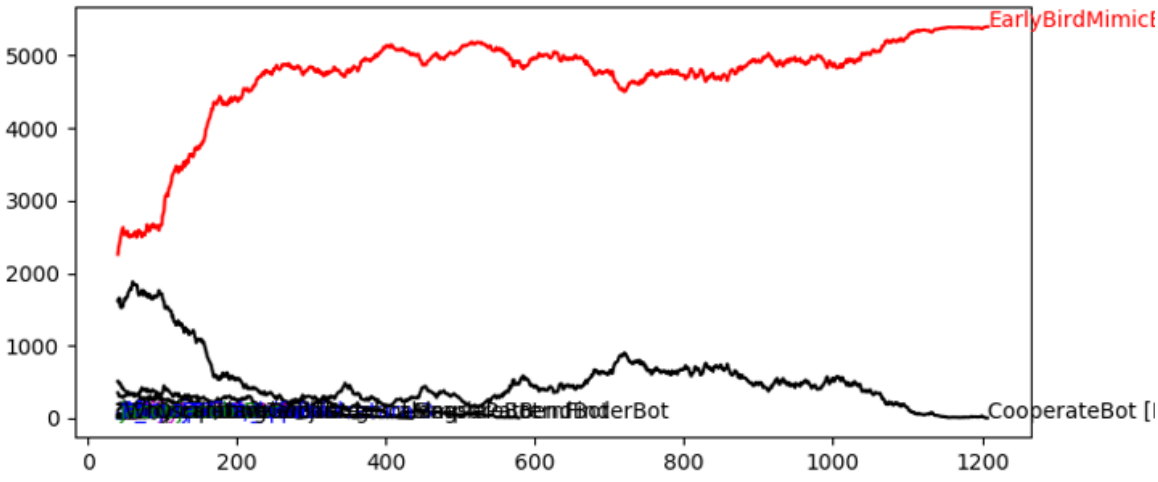
Rounds 21-40

Multicore's EarlyBirdMimicBot steals the lead from MeasureBot.



Rounds 41-1208

Welcome to Planet Multicore.



Winner

Bot	Team	Description	Round
EarlyBirdMimicBot	Multicore	Superintelligence	∞

Today's Obituary

Everyone else.

Conclusion

I hope you had fun. This wouldn't have been possible without the community here at Less Wrong. At least 75% of the code (not counting pseudocode) was written by people other than me. Thank you everyone who competed, debated, plotted and hacked. Thank you for the espionage and counter-espionage. Thank you everyone who helped spot bugs in the game engine. Thank you Zvi for posting the original Less Wrong Darwin Game series. Extra thanks to moderator Ben Pace for prettifying the tables behind the scenes and moderator Oliver for fixing multiple timestamps.

The source code to the game and all the bots is available here. If there is a bug in this timeline you can fix it yourself.

This concludes the 2020 Less Wrong Darwin Game.