University of Zurich^UZH

Agent-based Financial Economics
Lesson 8: Testing

Luzius Meisser, Prof. Thorsten Hens

luzius@meissereconomics.com

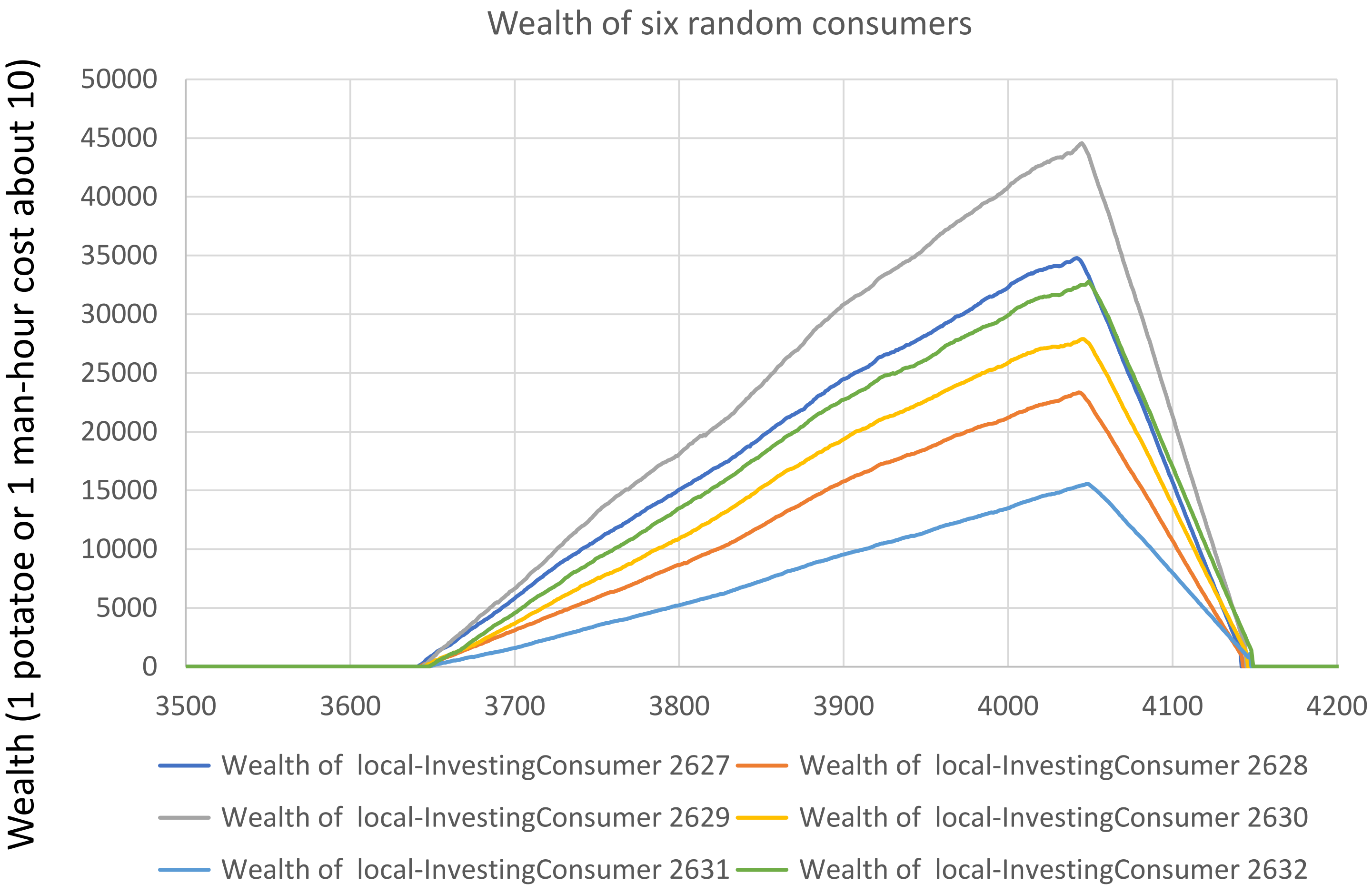
"What I cannot create, I do not understand."

- Richard Feynman

Today

- Discussion of exercise 7
- Outlook for the rest of the course
- The market maker’s problem
- System dynamics group work
- Exercise 8: Testing

Exercise 7 - Equality

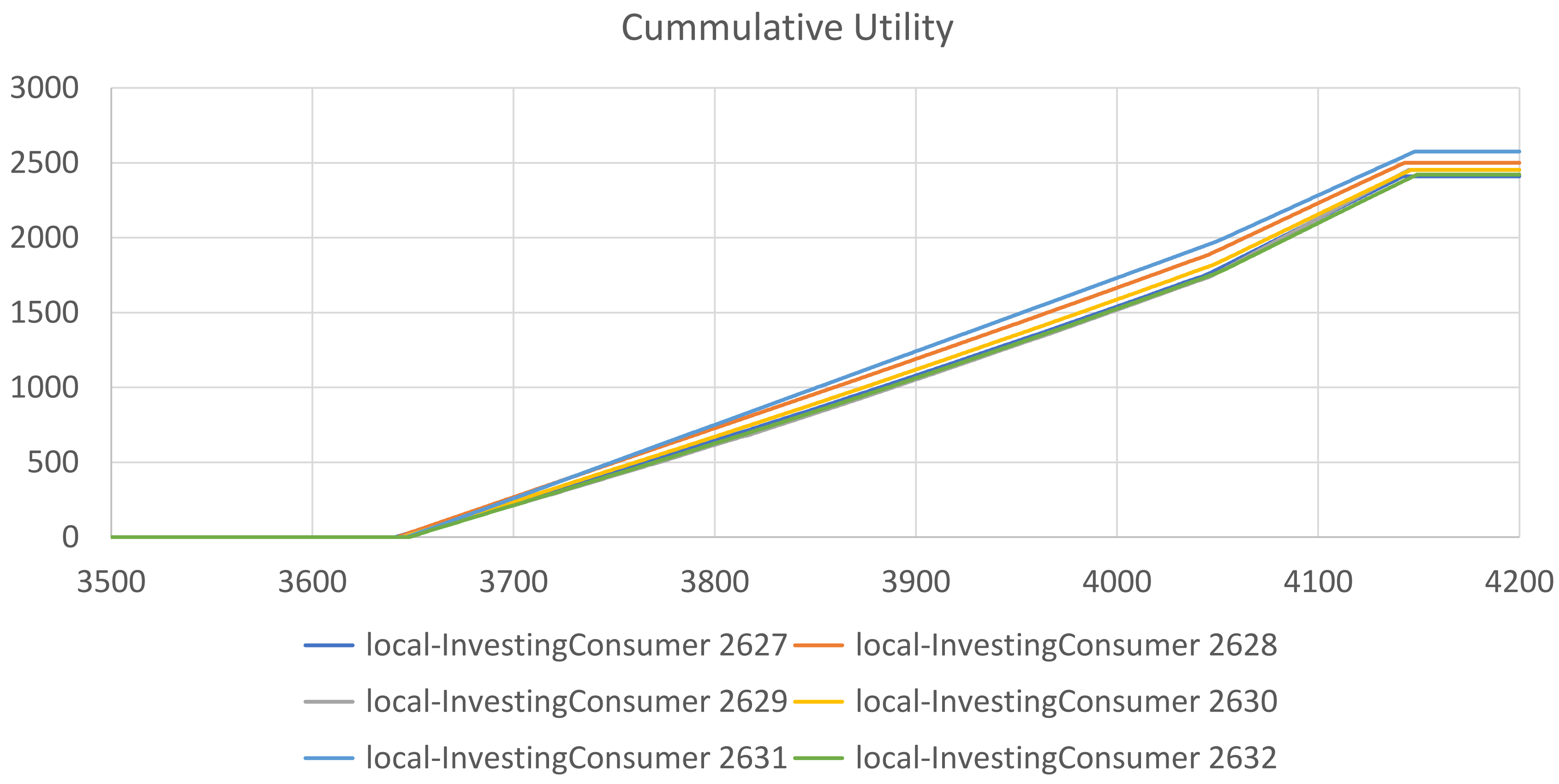


agent	alpha	beta
2627	1.085	0.915
2628	0.694	1.306
2629	1.494	0.506
2630	0.836	1.164
2631	0.57	1.43
2632	1.013	0.987

$$U(x_p, h_l) = \alpha log x_p + \beta log h_l$$

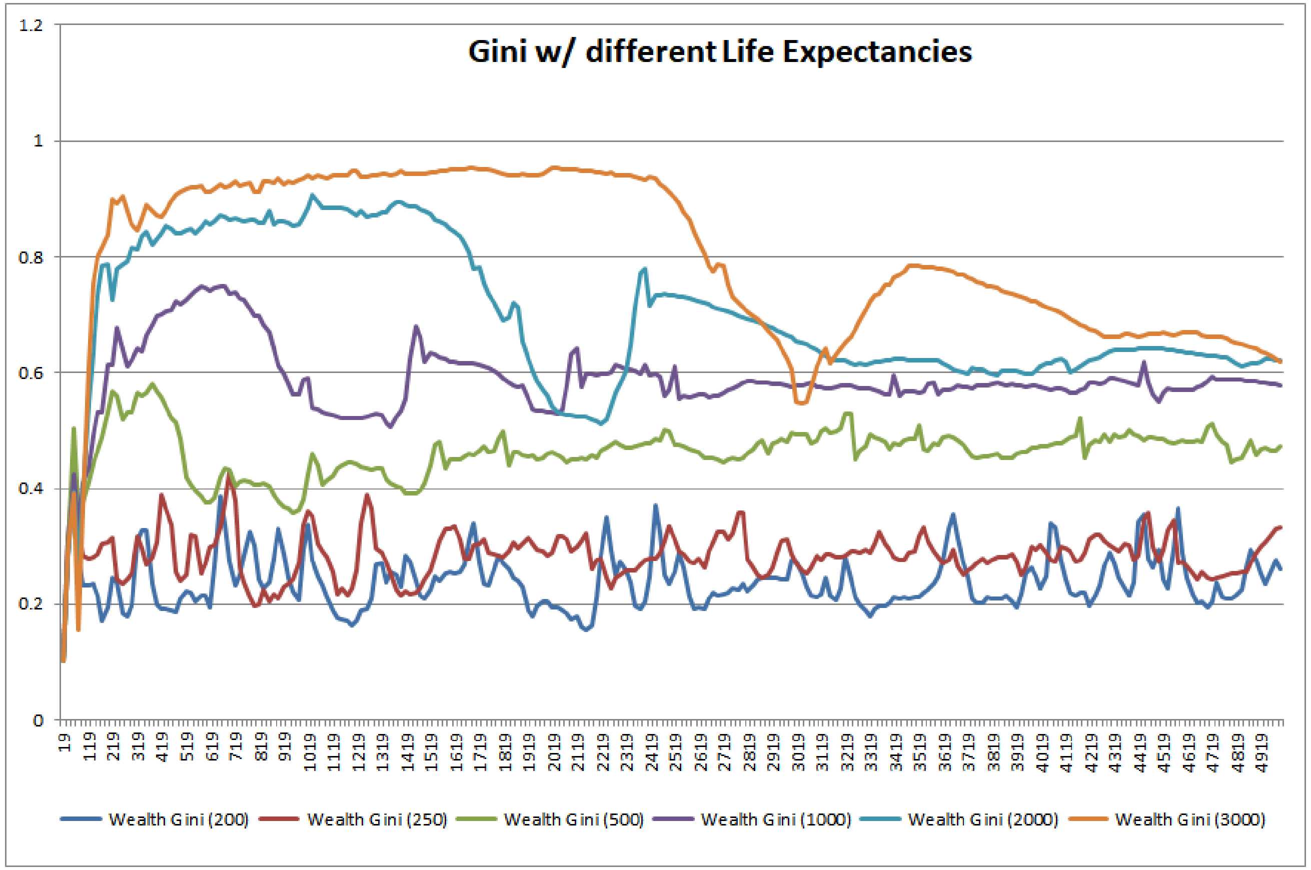
with $\alpha + \beta = 2$ and $0.5 < \alpha < 1.5$ being randomly chosen in that range, using the uniform probability distribution.

Exercise 7 - Equality



Looking at cumulative life-time utility, the world looks much more equal again.

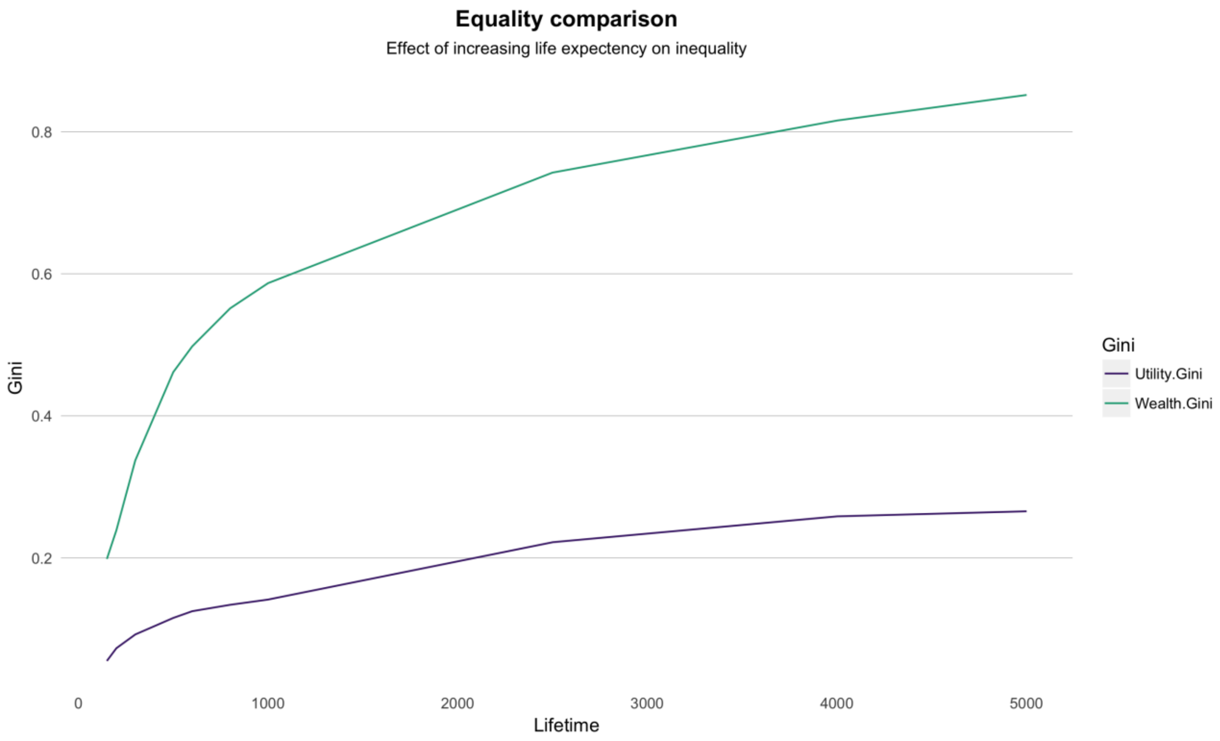
Exercise 7 - Equality



(This nice chart has been made by team 1.)

- Inequality increases as people get older
- Tests with very high life expectancy (above 1000 days) note very meaningful, as the simulation does not have enough time to settle on an equilibrium.

Exercise 7 - Equality

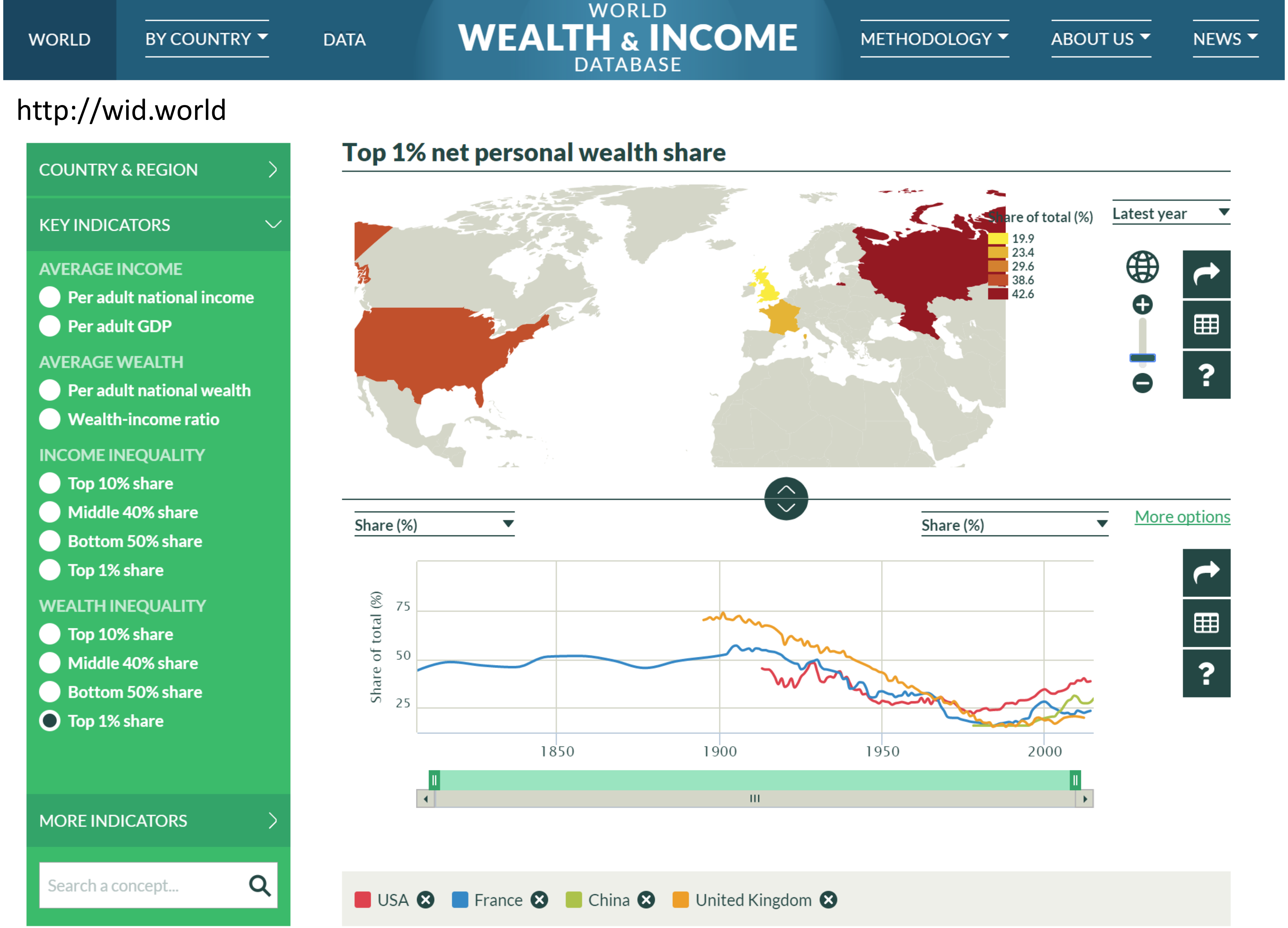


(This nice chart has been made by team 3.)

Some of you concluded here: “money does not seem to buy happiness”

This conclusion depends on the shape of the utility function, or diminishing additional utility when consuming more. I.e. the assumption that spending twice as much on something does not make you twice as happy.

http://wid.world



Top 1% net personal wealth share

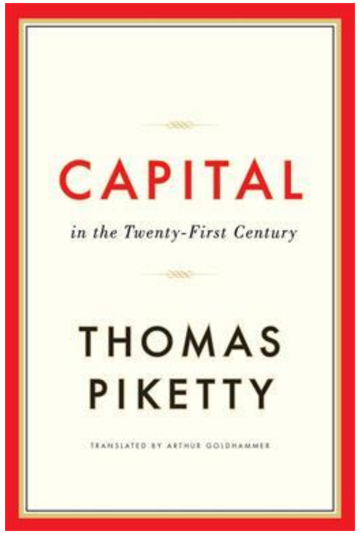
Latest year

Share (%)

Share (%)

🟥 USA 🟦 France 🟩 China 🟧 United Kingdom

Picketty



- Huge collection of historical data on wealth and income in the Western world
- Main finding: inequality has been increasing in the past decades in many western countries
- Model: return on capital is higher than economic growth, so the rich get richer.
- Proposed measure: introduce a global wealth tax.

Criticism:

- Used a questionable method to fill gaps in the data. (http://marginalrevolution.com/marginalrevolution/2017/10/pikettys-data-reliable.html) But this does not change much.
- Only looks at countries individually. It is possible for equality to decreasing in every country but still increase globally.
- Implicitly assumes that "the size of the pie is fixed"
- Disregards other causes of inequality, like ageing of the society (as we have seen in the simulation), people having fewer children, more divorces, etc.

→ My conclusion: started an important discussion about a real risk. In the digital world, the winner takes it all. But maybe everyone can find a niche to win in?

→ Zuckerberg, Gates and others are worried and are loudly thinking about a basic income.

Course Outlook

November 10th: testing, last exercise: market maker

November 17th: presentation setup, the three agent types

November 24th: control theory, learning

December 1st: leverage

December 8th: presentations: land developer

December 15th: presentations: farm with capital

December 22nd:: presentations: investment fund

(Dropped topic: endogenous technology.
Maybe topics: heterogeneous discount rates, blockchain talk.)

Presentations

Teams specialize on one of three firm agents:

- Two teams implement a **land developer**: buys and sells land, and converts man-hours into new land
- Two teams implement an extended **farm with capital** investments (buying and selling land).
- Two teams implement an **investment fund**, that trades on the stock market and in which the consumers can invest.

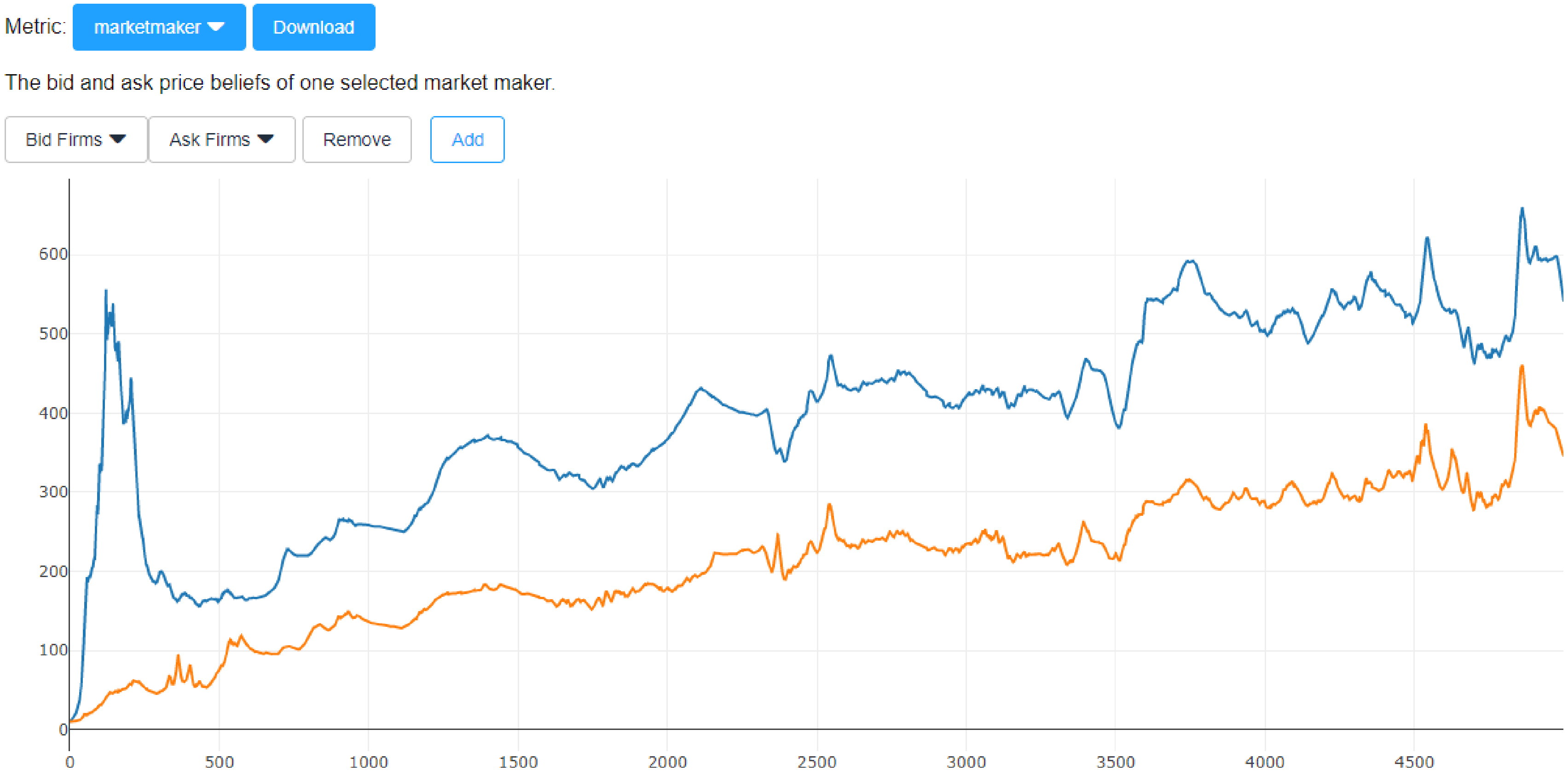
All agent types will be integrated into one big simulation.

Metric to optimize: dividends paid to consumer-shareholders.

Presentations: 10 minutes per team member.
Economists explain how the code work, computer scientists explain the economics behind their agents. Deliverables: slides, code.

Final grading: 60% exercise based grade + 40% presentation grade

Market Maker



Metric: marketmaker

The bid and ask price beliefs of one selected market maker.

Under the current configuration, the spread is way too high!

Naïve attempts at improving this failed or did lead to other problems…

→ We need more than parameter tuning.

Market Maker - Literature

Garman, M.B., 1976. Market microstructure. *Journal of financial Economics*, *3*(3), pp.257-275.

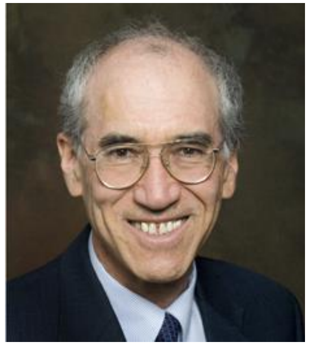
Assumptions:

- Arrivals of buy and sell orders to the market are Poisson distributed in time, with stationary rate functions A&) and A,@): 4 (order quantity) is assumed equal to 1.
- All exchanges are made through a single central ‘market-maker’, who possesses a monopoly on all trading. No direct exchanges between buyers and sellers are permitted.
- The market-maker is a price-setter, in the sense that he may control the price-probability functions for aggregate demand and supply (for example, by refusing all orders that do not meet his price). Specifically, we assume that he sets a price pe at which he will fill buy orders and correspondingly a price ps for sell orders, yielding the resultant order rates d,(p,) and As(ps), respectively. The asset that is bought and sold shall hereafter be termed ‘stock’, the numirairc asset ‘cash’.
- At time 0, the central market-maker has cash and stock inventories of 1,(O) and I,(O), respectively. Subsequent negative inventories imply the market-maker’s ‘failure’, i.e., inability to continue in his role.
- The market-maker seeks to maximize expected prolit per unit time, subject to the avoidance of certain ultimate failure.
- There are no transactions costs for the market-maker.

Findings:

- The exact solution of eqs. (6) and (7) for the ultimate failure probabilities is quite complicated, due to the fact that (a) there are two interrelated state variable equations, and (b) eq. (6) alone requires the solution of a polynomial of order pB+ps [cf. Feller (1968, p. 363ff.)]. As an alternative, we may approximate the ultimate failure probabilities as a function of the market-maker’s price strategy as follows.
- By being willing to take profits in the form of stock inventory increases, the market-maker can artificially inflate prices by maintaining the inequality pe > ps > p*; in no case, however, will the market-maker be able to set both prices below p* without ultimate stock failure.
- it clear that the specialists must pursue a policy of relating their prices to their inventories in order to avoid failure.

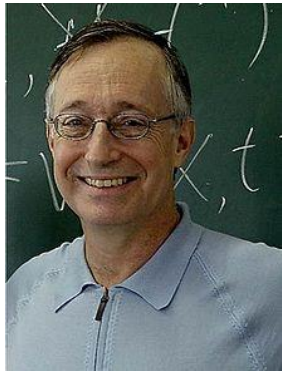
Market Maker - Literature



Amihud, Y. and Mendelson, H., 1980. Dealership market: Market-making with inventory. *Journal of Financial Economics*, *8*(1), pp.31-53.

“It is proved that the prices are monotone decreasing functions of the stock at hand, and that the resulting spread is always positive.”

Market Maker - Literature



Paul Milgrom

Lawrence Glosten

Glosten, L.R. and Milgrom, P.R., 1985. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, *14*(1), pp.71-100.

“The presence of traders with superior information leads to a positive bid-ask spread even when the specialist is risk-neutral and makes zero expected profits.”

“in this paper is based on the idea that a bid-ask spread can be a purely informational phenomenon, occurring even when all the specialist’s fixed and variable transactions costs (including his time, inventory costs, etc.) are zero and when competition forces the specialist’s profit to zero. “

“In this paper, we use a formal model to show how the spread arises from **adverse selection**”.

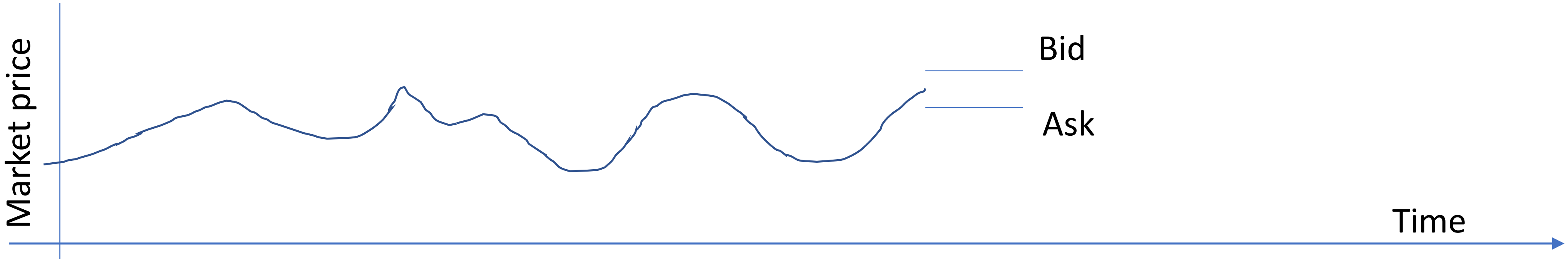
“The welfare loss we have described is at least partly due to the requirement that the specialist must break even on each trade.” (Bancor trader does not have that requirement.)

Adverse selection: market maker can never know whether an incoming trade comes from a liquidity trader (who buys or sells to reallocate his assets) or from an information trader with new insights about the true value of the stock. Whenever the market maker trades with an information trader who knows the true value of the stock, the market maker makes a small loss.

→ The spread is needed to recover the money lost to information traders.
→ If there are too many information traders, spreads go through the roof and the market breaks down.

Market Maker Problem

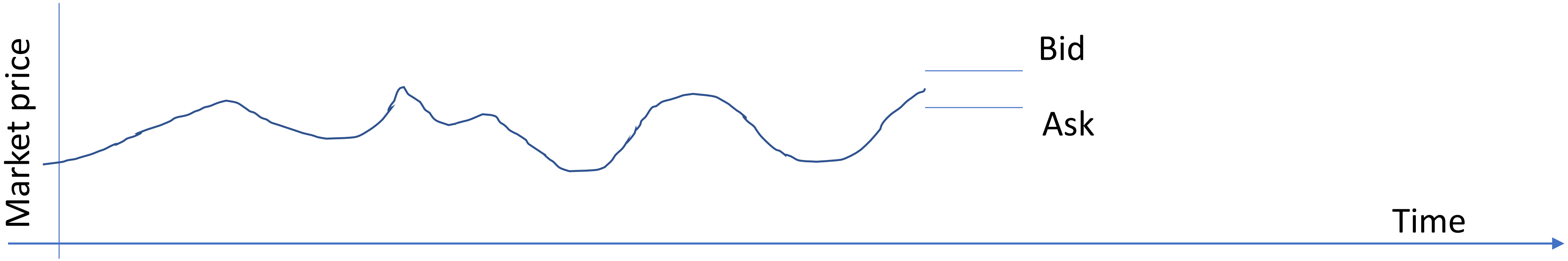
- The market makers are the only agents that are allowed to place limit orders
- Everyone else can only do market orders, and must therefore always trade with a market maker
- Market makers are risk-neutral
- Market makers maximize profits and compete with each other (Bertrand competition)
- Market makers should avoid running out of money or out of inventory. If they run out of both, they are bankrupt.
- There is an exogenous, stochastic inflow of money, i.e. 1000 USD on a particular day
- There is an exogenous, stochastic outflow of shares, i.e. 10 shares on a particular day
- The market maker has no access to external information about the true value of the stock



Market Maker Problem

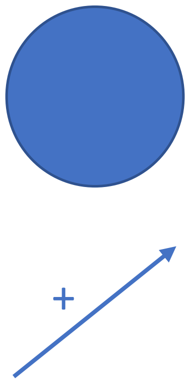
Order of events:

1. Market makers place their limit orders in random order
2. Market maker orders are matched against each other
3. Other market participants place their market orders



Market Making – System Dynamics

Take 20 minutes to sketch the system dynamics with your team.

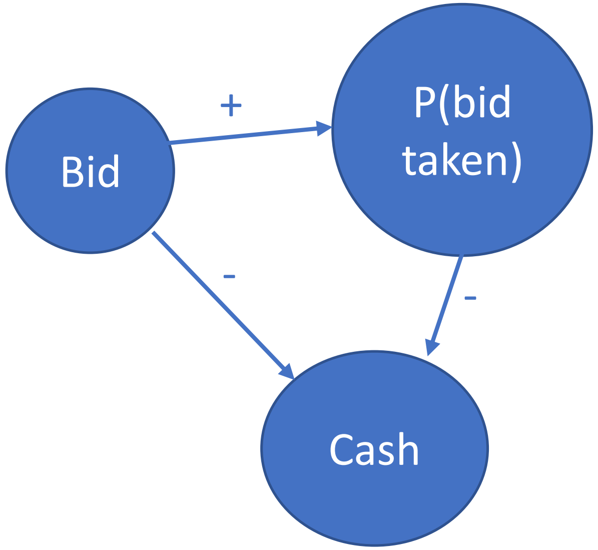


Circles are variables

Arrows indicate a positive (+) or negative (-) relationship.

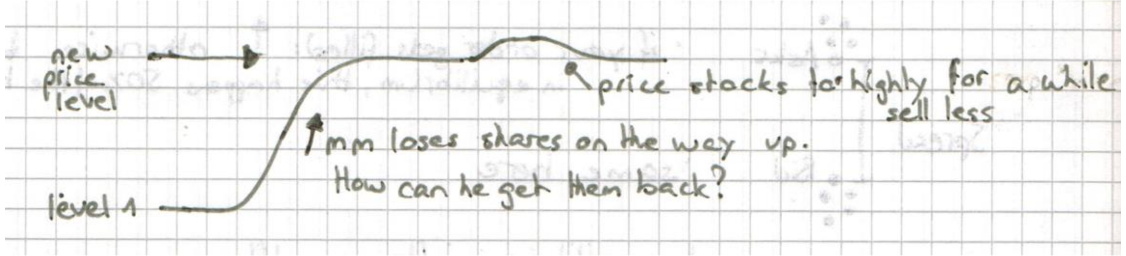
Ideas for variables:

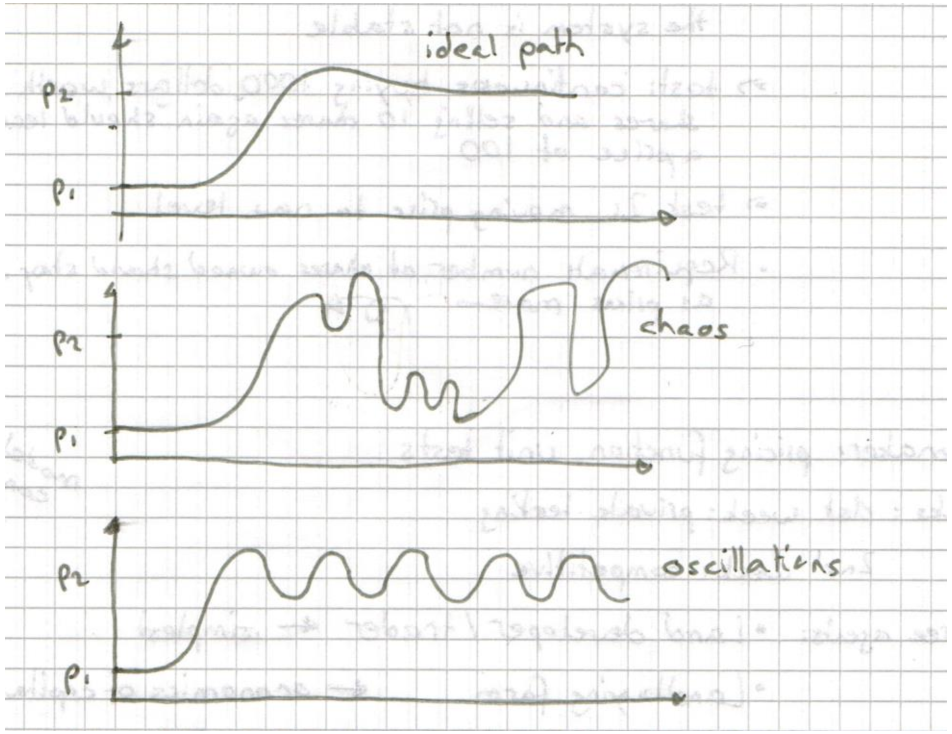
- Bid price
- Ask price
- Spread
- Equilibrium price
- Inventory: stocks
- Inventory: cash
- Inflow
- Outflow
- Profit
- …

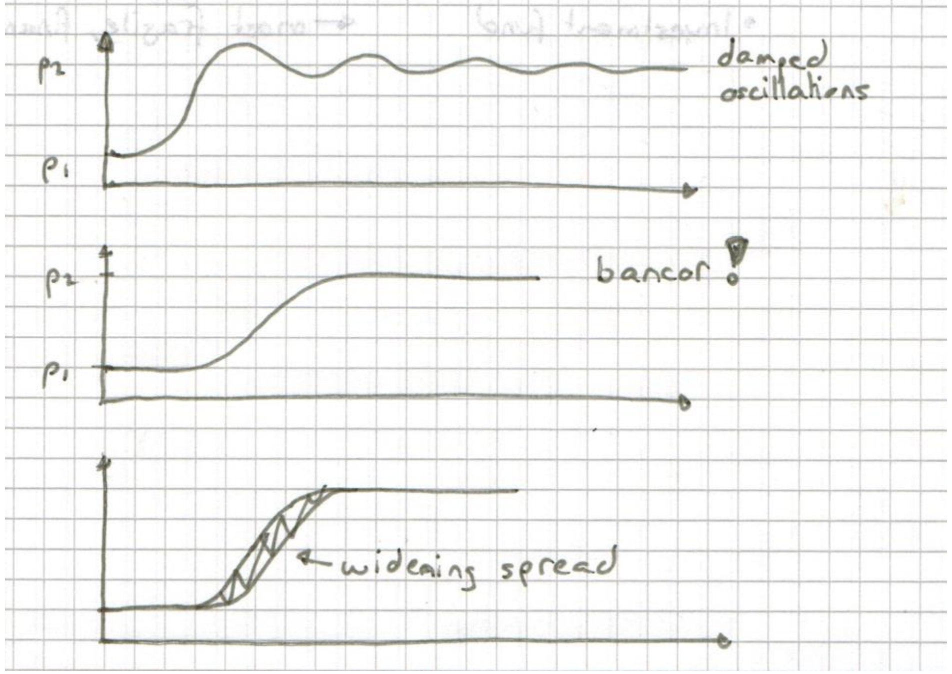


Example relation: a higher bid leaves the market maker with less cash. Also, it increases the probability of the bid being taken by a market participant, thereby further reducing cash.

Market Making


new price level
level 1
price stocks to highly for a while
sell less
mm loses shares on the way up.
How can he get them back?





Unit Tests

→ Look at code

Exercise 8 – Market Maker

See exercise 8 on github:

https://github.com/meisser/course/blob/master/exercises/journal/exercise08-task.md

How to comment code

1. Make sure the code itself is readable, “the code is the documentation”. The code tells **what** the program does.
2. Add comments to describe **why** you are doing something.

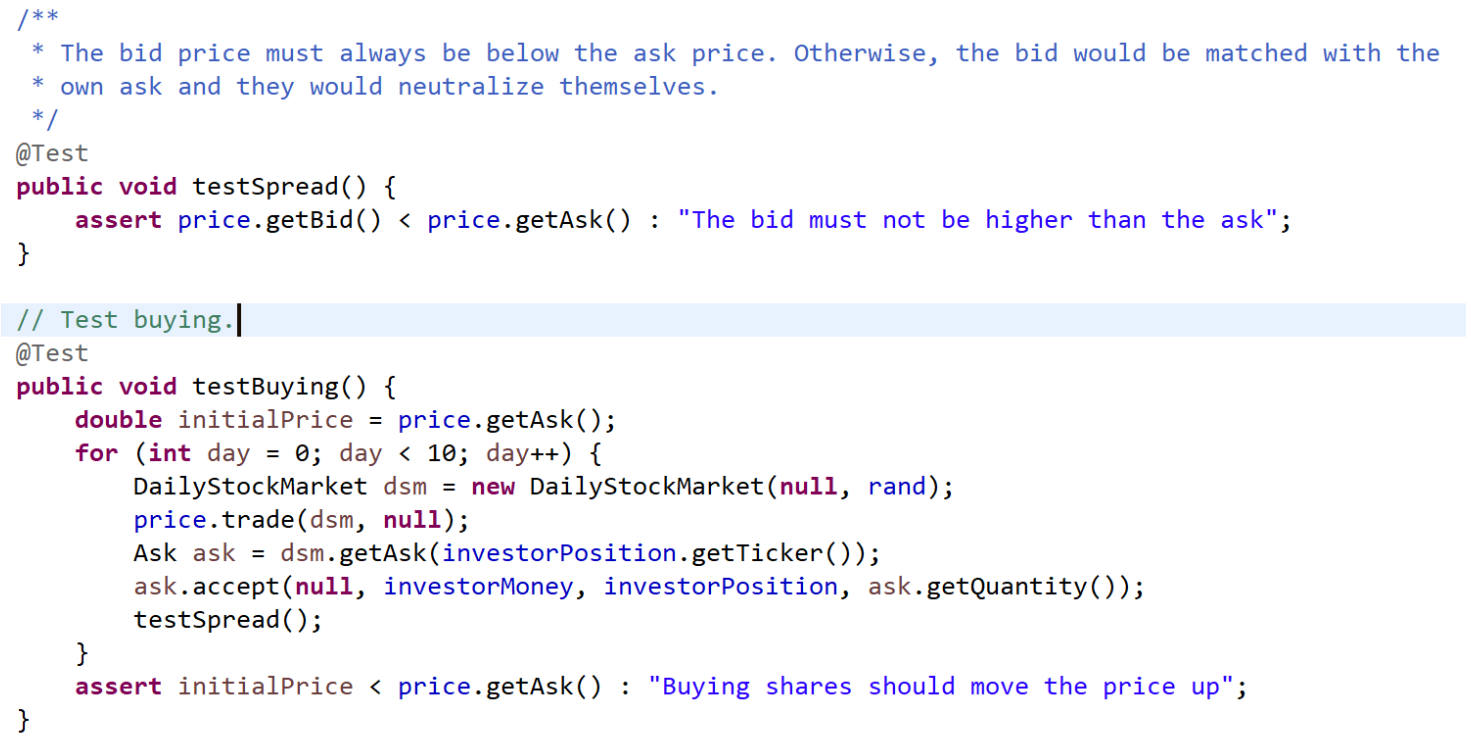
Readable code:

- Adhere to coding conventions and naming guidelines
- Short methods (only a few lines of code) with meaningful names
- Variables with meaningful names (unlike mathematics!)

Good comments:

- Specifies your intent, the “why”
- Refer to documentation or other sources where appropriate
- APIs: describe how an interface is supposed to be used

Commenting Example



```
/**
 * The bid price must always be below the ask price. Otherwise, the bid would be matched with the
 * own ask and they would neutralize themselves.
 */
@Test
public void testSpread() {
    assert price.getBid() < price.getAsk() : "The bid must not be higher than the ask";
}

// Test buying.
@Test
public void testBuying() {
    double initialPrice = price.getAsk();
    for (int day = 0; day < 10; day++) {
        DailyStockMarket dsm = new DailyStockMarket(null, rand);
        price.trade(dsm, null);
        Ask ask = dsm.getAsk(investorPosition.getTicker());
        ask.accept(null, investorMoney, investorPosition, ask.getQuantity());
        testSpread();
    }
    assert initialPrice < price.getAsk() : "Buying shares should move the price up";
}
```

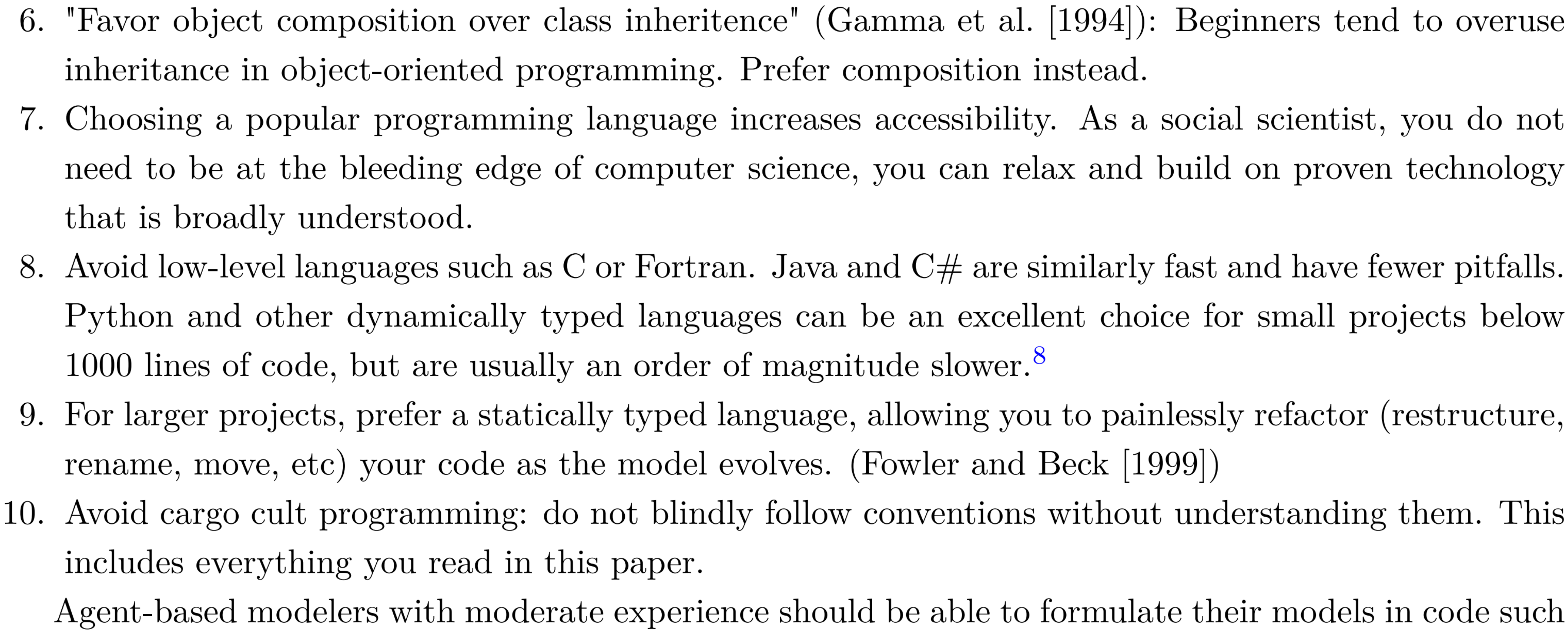
Good comment.

Bad comment.

Clean Code Hints 1-5

1. Keep your code clean and simple. Its purpose is not to show how smart you are, but to provide an accessible formulation of your model to others and your future self. Given two options, choose the one that astonishes your readers the least.
2. Split code into small units with descriptive names. For example, one should split large functions into multiple small ones even if they are only called from one place. The purpose of a function is not to enable the reuse of its code, but to structure the program nicely. The same applies to classes.
3. Choose the simplest tool that does the job. You won't need gimmicks like dependency-injection frameworks or aspect-oriented programming. They are often poorly supported by the development environment and tend to make relevant information less accessible.
4. Avoid premature generalization: do not write a function to draw polygons if all you need for now is rectangles.
5. Avoid premature optimization: Your first priority is to get things right. You can still optimize later if necessary. Most of the time, it is not. "Premature optimization is the root of all evil." (Knuth [1974])

Clean Code Hints 6-10



6. "Favor object composition over class inheritence" (Gamma et al. [1994]): Beginners tend to overuse inheritance in object-oriented programming. Prefer composition instead.
7. Choosing a popular programming language increases accessibility. As a social scientist, you do not need to be at the bleeding edge of computer science, you can relax and build on proven technology that is broadly understood.
8. Avoid low-level languages such as C or Fortran. Java and C# are similarly fast and have fewer pitfalls. Python and other dynamically typed languages can be an excellent choice for small projects below 1000 lines of code, but are usually an order of magnitude slower.[8]
9. For larger projects, prefer a statically typed language, allowing you to painlessly refactor (restructure, rename, move, etc) your code as the model evolves. (Fowler and Beck [1999])
10. Avoid cargo cult programming: do not blindly follow conventions without understanding them. This includes everything you read in this paper.

Agent-based modelers with moderate experience should be able to formulate their models in code such

Source: “Is the Code the Model?”, meissereconomics.com/2017/06/22/code.html

The Cargo Cult

https://www.youtube.com/watch?time_continue=29&v=c7RA4UnEuQ0

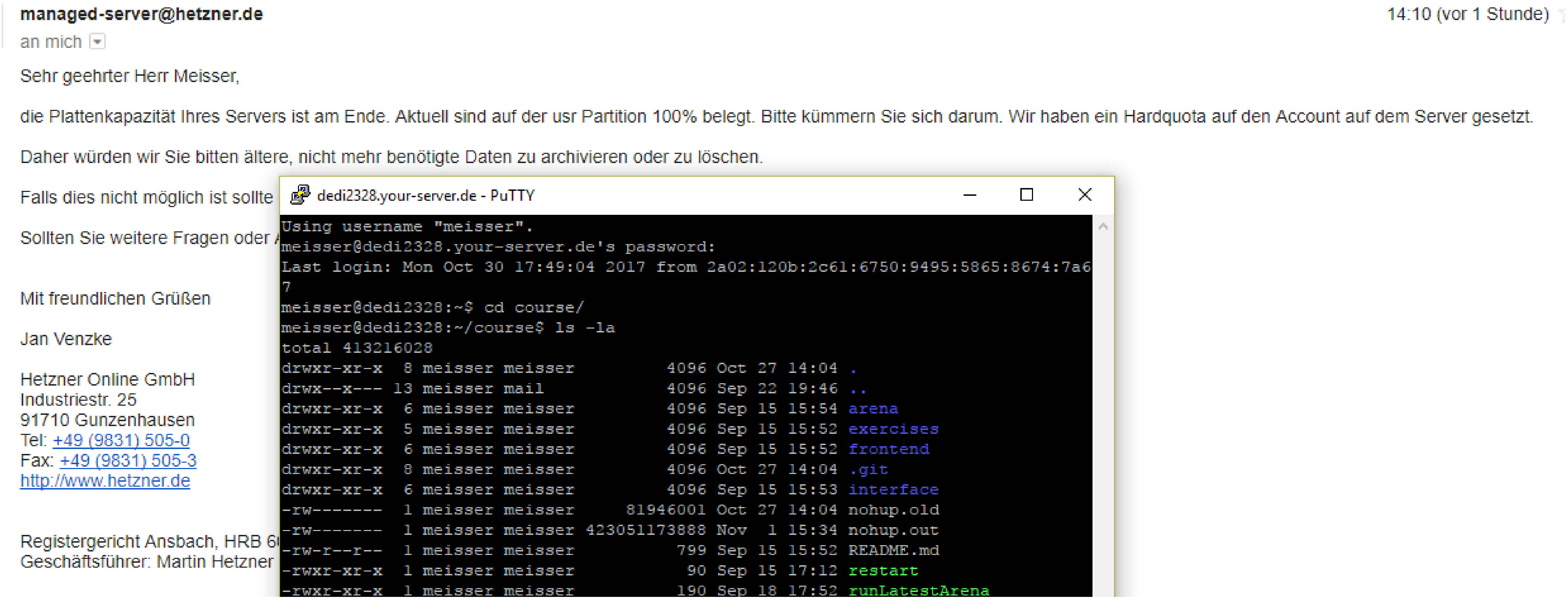
Video of New Guinea natives building fake airports to make the gods send them gifts from the sky.

The started doing this after they observed the Westerners successfully doing the same.

Note that this is a rational response to what they observed: create a testable hypothesis and then test it. That’s empirical science. It only turns into a cult if you refuse to adjust your beliefs after having done the test.

Today, “cargo cult programming” or “cargo cult science” is an euphemism for adopting best practices because everyone else follows them, and not because one has understood them and decided that they make sense.

Administrative Note



managed-server@hetzner.de
an mich

14:10 (vor 1 Stunde)

Sehr geehrter Herr Meisser,

die Plattenkapazität Ihres Servers ist am Ende. Aktuell sind auf der usr Partition 100% belegt. Bitte kümmern Sie sich darum. Wir haben ein Hardquota auf den Account auf dem Server gesetzt.

Daher würden wir Sie bitten ältere, nicht mehr benötigte Daten zu archivieren oder zu löschen.

Falls dies nicht möglich ist sollte

Sollten Sie weitere Fragen oder A

Mit freundlichen Grüßen

Jan Venzke

Hetzner Online GmbH
Industriestr. 25
91710 Gunzenhausen
Tel: +49 (9831) 505-0
Fax: +49 (9831) 505-3
http://www.hetzner.de

Registergericht Ansbach, HRB 6
Geschäftsführer: Martin Hetzner

```
Using username "meisser".
meisser@dedi2328.your-server.de's password:
Last login: Mon Oct 30 17:49:04 2017 from 2a02:120b:2c61:6750:9495:5865:8674:7a6
7
meisser@dedi2328:~$ cd course/
meisser@dedi2328:~/course$ ls -la
total 413216028
drwxr-xr-x  8 meisser meisser         4096 Oct 27 14:04 .
drwx--x--- 13 meisser mail            4096 Sep 22 19:46 ..
drwxr-xr-x  6 meisser meisser         4096 Sep 15 15:54 arena
drwxr-xr-x  5 meisser meisser         4096 Sep 15 15:52 exercises
drwxr-xr-x  6 meisser meisser         4096 Sep 15 15:52 frontend
drwxr-xr-x  8 meisser meisser         4096 Oct 27 14:04 .git
drwxr-xr-x  6 meisser meisser         4096 Sep 15 15:53 interface
-rw-------  1 meisser meisser     81946001 Oct 27 14:04 nohup.old
-rw-------  1 meisser meisser 423051173888 Nov  1 15:34 nohup.out
-rw-r--r--  1 meisser meisser          799 Sep 15 15:52 README.md
-rwxr-xr-x  1 meisser meisser           90 Sep 15 17:12 restart
-rwxr-xr-x  1 meisser meisser          190 Sep 18 17:52 runLatestArena
```

→ Please remove all “System.out” statements from your Code before pushing it to github!
→ Also, it would be great if you could remove them from your old agents.