Francesco Checcacci, 30th September 2014

A simulation model to conceptually understand welfare system effects on workforce and consumption

1. Abstract

The paper starts with a brief introduction where I mainly explain how the simulation model, realized with the free software NetLogo 5.0.5 available at https://ccl.northwestern.edu/netlogo/download.shtml, generally works without immediate reference to the code, which will be due in the Body of the paper, in that part we will make a deep dive into the specifics of the code, together with a logic comprehension of the model.
In the conclusions we will see how specific settings of the parameters will lead to different outcomes

2. Introduction

The purpose of this simulation is to understand how, welfare thresholds to be eligible for financial aid, income brackets for taxable income and tax rates, affect the amount of time workers are willing to dedicate to their job.

In this model workers are completely free to decide the amount of time they decide to work with no constraint such as a limited amount of labor supply;
What they cannot decide is the amount they will be paid for a unit time.

Given a increasing utility function we set as independent variable the consumption and we multiply it by a factor representing one's enjoyment of free time, so there will be workers who gives a higher value to free time, and workers who give a lower one, so if U is an utility function, it will be such that: U (consumption, enjoyment of free time, free time).

To simplify we define one threshold of income below which the worker receives a benefit and a threshold above which a worker pays taxes, then the taxable income above such threshold will be deducted at a defined tax rate.

Since we are focusing only on the welfare system, all the money collected will be distributed to those eligible for the benefit

Notice that when workers choose the combinations of work and free time that maximizes their utility they have to consider the benefit/taxes they would receive/pay

Starting the simulation we see how parameters levels affect the amount of time dedicated to work by some type of workers with given characteristics (wage, and enjoyment of free time) it is interesting to see how a given set of parameters such to allow for high benefits reduces the amount of work time in order to receive the benefit, or in order not to fall in the income bracket which make due the payment of taxes: in presence of a very high enjoyment of free time, these preferences won't make an hour work more pay off.

Workers' wage is allowed to increase and vary through time.

It will be also interesting to see the effects on wealth inequality through Gini index and graphically through Lorenz Curve.

The first part of the body will already give a deep comprehension on how to use the model.

To make the most of the following guide, install Netlogo and download the model.

3. Body

3.1 Interactive user guide

This guide explains to the user how to manage the interface tab of the model without reference to the code tab.

3.1.1 The input

The green sliders are input variable defined by the user, on the left, from the top we have:

- *Num-people*: the number of agent in the model.
  Every agent in the model either has an income or is unemployed.
- *Tax_rate*: the tax rate.

We temporary skip *min_income_bracket*.

- *Income_bracket* : workers' income above this value is taxed at the tax rate.
- *Benefit_bracket* : every agent whose income is lower than this value is eligible for a benefit.
- *Min_income_bracket* : in order not to have the user having some agent contemporary pay taxes and receive a benefit, we set this value as the minimum value of *income_bracket* and as the maximum value of *benefit_bracket*.

Below the black screen we have other two sliders defining the number of agents having a certain characteristic regarding their consumption habit; for both this type of agents, their consumption is affected by the consumption of the agents around them.

- *Num_pretentious*: the number of agents whose consumption increase when they found themselves around agents who consumes more than them
- *Num_humble*: the number of agents whose consumption decrease when the average consumption of those around them is lower.

Notice that the number of pretentious and humble agents is within the number of agent in the model defined in the first slider, for example, if you set 250 agents in the model, 100 number of pretentious ones and 100 of humble ones, the total number of agents will still be 250, and the consumption of 50 of these agents will not be influenced by the agents around them.

Be sure to have the red knobs in the sliders visible, if it is not so, click again on the darker line in the slider; do so in order to avoid an error.

If by now you are already bored and you are impatient to start the model, set some random value for these variables, press *setup* and then start the model through the button *go*.

Press again *go* to stop the model.

3.1.2 The output

Every histogram, graph, and number you see with a gray background is output of the model.

- The *Lorenz curve* gives a great representation of how wealth is distributed among the agents, and it works in the following way: take a number in the X axis, say 50%, then take the corresponding value on the Y axis, say you check it and it is 25%, this means that the poorest half of the population owns a fourth of total wealth; you can check the values by passing the cursor on the graph.

- The *Gini index* is a normalized measure of wealth inequality among agents, graphically it represents the area between the red and black lines in the Lorenz curve, therefore it is good for an immediate understanding of the 'average' inequality, but the Lorenz curve, with its shape, can tell us more;

For an immediate understanding of the importance of the shape of the Lorenz curve, run the model at a fast pace moving the default slider on the top to the right, form normal speed, to faster, with the following settings:

Num-people = 200; Tax_rate = 0.40; income_bracket = 60; benefit_bracket = 60; num-pretentious = 150; num-humble = 0

Press *setup* then *go*.

Wait a while after you have run the model… do you see a particular shape in the Lorenz curve? Is it symmetric? We will reason on this when we will draw all the conclusions.

Press *go* to stop the model.

Let the settings as above, go on reading, we will run the model again soon.

Even if this is not the first time you use NetLogo, in order to make this guide efficient and enjoyable I wrote the commands you will execute in a specified order, so if you want to notice every detail as you read the paper, be ware that the order I wrote the commands matters.

- The *tax revenue* graph represents the amount of taxes collected.

Decrease the number of agent with a taxable income by raising the bracket of taxable income (*income_bracket*), then press go. What happened? Set the value back to 60. Press go again to stop the model.

What happen if we increase the tax rate? Think about the answer, then try it, first press go, then slowly increase it up to 1. Surprised? Would you work more or less if tax rate were at a 100%?

Set the value back to 0.4 then *go* to stop the model.

- *Benefit* is the amount distributed to each agent whose income is lower than the "benefit bracket", now at 60, start imagine this as a daily income, if your income is less than 60 euro per day, then you are eligible for a *benefit*, which, with the parameters we set, should be around 5 euro.

Lower the bracket to be eligible for the benefit (*benefit_bracket*) and start the model.
The *benefit* due to those eligible is now higher because all taxes collected are shared among a lower number of agents. Notice we are redistributing with the benefit every euro collected with the taxes.

Set the value back to 60 and press *go* to stop the model.

- *Class plot* represents through time the number of agents belonging to the lower class in red, the mid class in green and the upper class in blue.
  Classes are based on wealth, not on income.
- Then we have a representation of classes with a *histogram*.

- The *working time* histogram represents the number of individual, for the number of shifts they work where every shift is a four hour one. Take a look at this histogram while you try the following:

Lower again the bracket to be eligible for the benefit down to 1. What will happen? Press go.
To understand this we will have to leave the interface tab and take a look at the code. Stop the model.

3.2 The code

Up on the left you have three tabs. Press on *Code*.

This tab is a set of procedures preceded by the name of the user-defined variables we are going to use (e.g. *gini-index-reserve* or *wealth*); to see all the procedures press on *procedures*.

You should be familiar with two which can be directly invoked from the interface tab: *setup* and *go*.

Click on the *setup* procedure.
This procedure invokes other 5 procedures, those in blue are built-in procedures, and those in black are defined in the code tab itself (user-defined).

Everything in the code which is black or orange is user-defined.
Everything else, is built-in, if at any time you are not satisfied with my explanation about a built-in feature of Netlogo, right click on it and press *Quick Help*.

Every procedure is defined within *to* and *end*.

The *setup* procedure first *clear-all*, for the reader who never used Netlogo will be easier to understand what is the "all" to be cleared if we go on with the code, for now think of it as if I have already invoked some procedure, this *clear all* will cancel its effects on the *agents* and the *world* they act in.

We then *setup-turtles* (user-defined procedure), where *turtles* in NetLogo is a built-in set of agents, one turtle, is an agent of the set, and every turtle has some built-in variable and some user-defined variable.

The user-defined variables for the turtles are those preceded by *turtles-own* at the top of the code.

The *setup-turtles* procedure creates a number *num-people* of agents and then assign them some random value for their variables:

- *Wage* is the amount earned per hour of work by the agent and it can be either 0 (unemployed) or in a range between 5 and 15 euro per hour.
  The number of unemployed agents is displayed in the interface tab;

Right next to it, it is also displayed the number of those who are not seeking a job, the second ones fall in the histogram among those who works 0 hours, the difference will be clearer later.

- At this stage consider *wealth* as an initial endowment.
- Leisure represents how much an agent enjoys free time, this variable, together with other procedures will explain the presence of agents with a positive wage *not seeking a job*.
- Agents consume a fraction *consumption%* of their *net income*, this fraction, at first is set to be 90% of the *net income*
- *Size* is a built-in variable, where 1 is the size of one *patch* in the *world*.
- There are other two "yes" or "no" variables to be set: *pretentious* and *humble*.
  The next 8 lines of code are necessary to have among all the agents a number *num-pretentious* with the variable pretentious set on "yes" (and "no" the other) and a number *num-humble* with the variable *humble* set on "yes".
  Every other agent will have "no" for both variables
- At last we invoke the procedure *recolor-turtles*
  With "let" we are setting a local variable, this variable *max-turtle* is neither a turtle nor a global variable because it will be used only in this procedure.
  This code sets the built-in variable *color* depending on the wealth of the turtle:
  Blue for the upper class, green for the mid class and red for the lower class.
  I will describe the meaning with an example; it will be useful to understand its limit:
  We have 10 agents, 1 of them owns 3 €, 8 of them own 5 € and the richest owns 12 €, the first will be the only one in the lower class because he or she owns less than 1/3 of the richest, the seconds 8 will represent the mid class because they own less than 2/3 of the richest and the richest will be the upper class;
  Imagine the richest owns 18 euro, you would have 8 agents passing from the mid class, to the lower class.
  This procedure will be invoked again at every tick in the *go* procedure.

With the *set-up-turtle-position* we distribute in an ordered way the *turtles* in the *world*.
We divide the length of the X axis of the world by the square of the number of agents, this local variable *step* is used to define an interval between the agents, at the end of the while cycle you actually set the *xcor* (the position with respect to the abscissa) and *ycor* (w.r.t. the ordinate) variable of the agent through the *setxy* command.

The *update-lorenz-and-gini* procedure as *recolor turtles* will be invoked again at every *tick*.
If we right click on the lorenz curve graph, press edit and press on the little icon shaped as a pencil in correspondence to the lorenz pen name entry, we see this red line plots each number from a vector called lorenz-points, this vector results from this procedure as an ordered set of numbers each

representing the cumulative *wealth of* the *turtles* up to a specific turtle: the turtles are first ordered by their *wealth*, then we update the local variable *wealth-sum-so-far* by adding each time the next turtle in the ordered vector of *wealth*.
The *gini-index-reserve* is the summation of *(index / num-people) - (wealth-sum-so-far / total-wealth)* for a number *num-people* of times, notice that *index* ranges from 1 to *num-people* and *wealth-sum-so-far* is the cumulative wealth up to the index-th agent in the ordered vector wealth.
To understand this, imagine the wealth is equally distributed among all agents, every time, *(index / num-people)* will equal *(wealth-sum-so-far / total-wealth)* on the contrary if one agent owns all wealth, *(wealth-sum-so-far / total-wealth)* will equal zero until the last agent who owns 100% of wealth.
Notice that gini-index is actually the reserve times *2/ (num-people)* it is therefore normalized between 1 and 0.

Reset-ticks set the built-in variable *ticks* to 0.

Before exploring the *go* procedure notice in the interface tab by right clicking on the button go, that the *forever box* is ticked, this means that is procedure is run again and again until the button is clicked a second time.

The procedure starts *asking* the turtles to execute two other procedures, *ask* is a command to have all the agent of an agent set do something.

Update set three variables, for now notice that the maximum income is the *wage* per hour times the maximum number of *hours* to work (three 4 hours-shifts) and *maxConsumption* is the consumption rate times the maximum income, these two variables will be helpful to construct through a linear utility function a simple system of preferences.
So for now it is important to notice that *income* equals the number of *hours* of work times the *wage* per hour.

With *pay-taxes* we ask the turtles to set their *netIncome* equal to the gross *income* minus *tax*, where *tax* is a turtle-own variable representing the amount the worker pays:

```
to pay-taxes ;turtle procedure
  ifelse (income > income_bracket)
  [set tax Tax_Rate * (income - income_bracket)
  set netIncome income - tax]
  [set tax 0]
end
```

Looking at *collect-and-distribute-taxes* we understand the importance of setting *tax* as a variable for each worker; we take the sum of all sums paid, we define the *benefit* by dividing the sum among those who are eligible for it:

```
to collect-and-distribute-taxes
  set tax_revenue sum [tax] of turtles
  if (count turtles with [income < benefit_bracket] > 0)
  [set benefit tax_revenue / (count turtles with [income < benefit_bracket])]

  ask turtles with [income < benefit_bracket]
  [set netIncome income + benefit ]
end
```

Define-netincome-for-non-taxpayers-and-non-subsidy-beneficiary

Later in the code we will have the agents consume and save a part of their *netIncome* variable; it is therefore necessary to set this variable equal to the *income* for the agents who neither pay taxes nor get a benefit.

Update-pretentious-and-humbles.

Remember we have some agents who are *pretentious*, these agents cannot stand the fact to have somebody in their neighborhood who consumes more than them, therefore if any of this agents' *consumption* is lower than the *consumption* of any other agent around them they increase their *consumption rate* even up to 98% of their *net income*.

The following is the first part of the procedure, the one regarding *pretentious* agents:

```
to update-pretentious-and-humbles
  ask turtles with [pretentious = "yes"]
  [
   if (consumption < max [consumption] of turtles in-radius 4
     and consumption% < 0.98)
   [set consumption% consumption% + 0.04]

   if ((wage * hours * 0.9) >= max [consumption] of turtles in-radius 4
     and consumption% > 0.9)
   [set consumption% 0.9]
  ]
```

On the contrary *humble* agents decrease their *consumption rate* even down to 70% if the average *consumption* of the agents around them is lower. The second part of the procedure:

```
  ask turtles with [humble = "yes"]
  [
   if (consumption > mean [consumption] of turtles in-radius 4
     and consumption% > 0.7)
   [set consumption% consumption% - 0.04]

   if ((wage * hours * 0.9) <= mean [consumption] of turtles in-radius 4
     and consumption% < 0.9)
   [set consumption% 0.9]
  ]
```

```
end
```

We *ask* the agents to save a fraction of their wealth and consume the other.

Get-a-raise_get-a-cut_get-hired_get-fired.
These procedure increases the *wage* per hour by one euro with a 10% probability or it decreases it with a 10% probability, always remaining in the values (0; 5-15).

Choose-hours.
This is the most important procedure of all; it describes the process through which an agent set his own *hours* variables.
An agent can choose to work either 4, 8, 12 hours (1, 2, 3 shifts) or either not work at all.

Remember we have in the interface tab *unemployed* and *not seeking a job*, let's see first what's inside those *monitor*:
Unemployed: *count turtles with [wage = 0]*
Not seeking a job: *count turtles with [hours = 0 and wage > 0]*
The first ones cannot have a job.
The second ones decided through this procedure not have one.

This procedure takes a simple utility function described in the following report procedure (a special procedure recognized by the *to-report* preceding it) depending directly on *consumption*, *leisure* and *hours* of work and tries the four possible value of hours to set the one that maximizes it.

Utility (*consumption*, *hours*, *leisure*) = *consumption* * (*maxConsumption* - *hours* * *leisure*)
Consumption is again an increasing function of *hours*, *wage* and *consumption%* but we multiply it times a number which is a decreasing function of *hours* and of *leisure*; notice these are turtles-own variable, therefore each agent will pick his or her own combination of working and free time that maximizes his or her utility function.

We write again utility extending consumption

Utility (*wage*, *hours*, *leisure*) =
(*hours* * *wage* * *consumption%)* * (*maxConsumption* - *hours* * *leisure*)

We will focus in the conclusion on the effect of *consumption%* (you already saw one in the user guide, in the shape of Lorenz curve), for now focus on how utility depends on *wage* and *leisure*.

I recall that wage = {0, 5-15} and leisure = {5-15}

MaxConsumption just guarantees a logic system of preferences in the utility function; by this I mean that the above utility function respects four simple rules:
Let *wage* = x, *leisure* = y, then if the solution of Max Utility w.r.t. hours is *hours* = H

- If *wage* = x + K, the solution to Max Utility w.r.t. hours is *hours* = H* ≥ H
- If *wage* = x - K, the solution to Max Utility w.r.t. hours is *hours* = H* ≤ H
- If *leisure* = y + K, the solution to Max Utility w.r.t. hours is *hours* = H* ≤ H
- If *leisure* = y - K, the solution to Max Utility w.r.t. hours is *hours* = H* ≥ H

(K > 0)

In the report procedure before reporting the value *such-utility* into the *choose-hours* procedure, if an agent's *income* (*hours* * *wage*) is less than *benefit_bracket* we add the last *benefit* distributed (the agent does not know which will actually be the *benefit*) to the *income*, if instead it is more than *income_bracket,* we subtract *(Tax_rate * (income - income_bracket))* which is known.

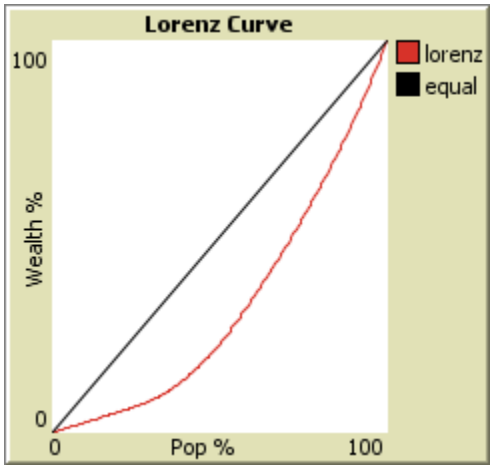
I write first the *choose-hours* procedure and then the reporting procedure:

```
to choose-hours ;turtle procedure
  set h 0
  let best-hours h
  let best-utility such-utility
  let utility0 such-utility
  repeat 3
  [
    set h h + 4
    if (such-utility > best-utility)
    [ set best-hours h
      set hours best-hours;;;
      set best-utility such-utility]
  ]
  if (utility0 = best-utility)
  [set best-hours 0
   set hours best-hours
   set best-utility such-utility]
  set utility best-utility

end

to-report such-utility
  let i h * wage
  if (i < benefit_bracket) [set i i + benefit]
  if (i > income_bracket) [set i i - (Tax_rate * (i - income_bracket))]
  let c i * consumption%
  let u c * (maxConsumption - h * leisure)
  report u
end
```

4. Conclusions

Let's start from where we left in the user guide.

The first curiosity we noticed in the user guide is the asymmetry of the Lorenz curve with particular settings.
Such effect is due to the number of *pretentious*, these agents raise the fraction of income for consumption when somebody around them consumes more than they do, every agents starts from a consumption rate of 90% of their income, 'normal agents' will keep this value along the simulation, the consumption of 'pretentious' increases up to 98% when the conditions are met, therefore those 150 pretentious agents will have a consumption rate in a range between 90% and 98%, they will be saving less on average and as time passes the gap between the accumulated wealth of these individuals and the one of normal agents will increase.
Figure 1

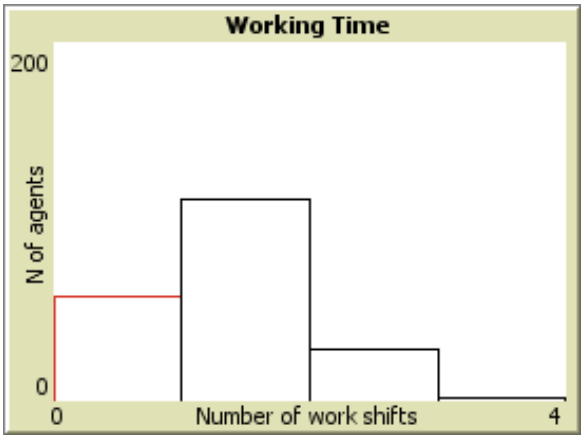


Num-people = 200; Tax_rate = 0.40;
income_bracket = 60; benefit_bracket = 1;
num-pretentious = 150; num-humble = 0

Asymmetry here is very marked, the curve tend to make a rapid movement around the 40% of the population; it is telling us that 40% of the population owns only 13% of total wealth and this wealth is distributed very smoothly across the poorest 40%,
The other 87% of wealth is still pretty evenly spread across the other 60% population.
We know this is due to the presence of *pretentious* agents, indeed most of them belong to the lower class, to see this type in the command center *count turtles with [color = red and pretentious = "yes"]:* it is the number of *pretentious* agents who belongs to the *lower class*, and it will be very close to the total number of agents in the lower class.
Notice that the 'jump' in wealth inequality starts at 40% population, which is 80 individuals, not the 150 as pretentious agents, this is due to the fact that not all 150 agents consumes more than 90% of their *net income*, and even if they do so, this phenomenon could be compensated by a high *income*.

We noticed how *tax revenue* dropped when raising *tax rate* up to 100%, we will run again the experiment taking also a look at the *working time histogram*.

If you let parameters as before, you should have *income_bracket* at 60. If not, set it at 60.
Calculate how many shifts you have to do to earn more than 60 €.
If you earn more or equal than 8 euro even 2 shifts will be enough.
Think the other way around: if you make 3 shifts, is it possible you do not fall within the 60 € bracket?
Start the model, increase the *tax rate* up to 70%, stop the model; this is more or less the effect:
Figure 2

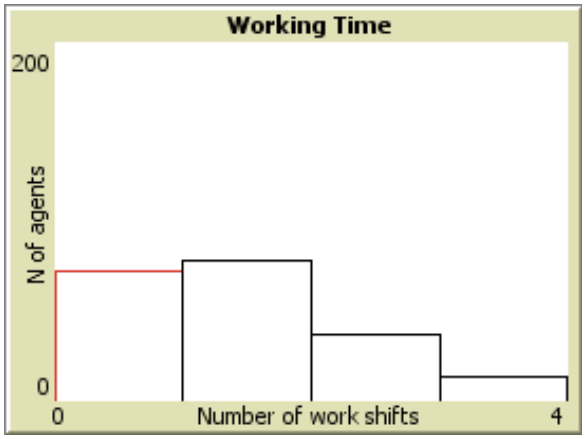


Input:
Num-people = 200; **Tax_rate = 0.7**;
income_bracket = 60; benefit_bracket = 1;
num-pretentious = 150; num-humble = 0
Output:
0 hours: 58 agents
4 hours: 113 agents
8 hours: 28 agents
12 hours: 1 agent

If you work 12 *hours*, even with a 6 € wage per hour, 12 € of your income will be taxed at a 70% rate, therefore very few workers will chose to make 3 shifts.
The only worker who decided to work 12 hours with these conditions has *leisure* equal to 5.3, (the minimum is 5 and maximum is 15) which denotes a very small enjoyment of free time, and a *wage* per hour of 14 €, therefore for him a work shift more will raise his income of 56 €, even if this money will be taxed at 70%, he will be left with 17 € more, which overcome the small pleasure given to him by free time; but this is one case over 250 workers who decided to work less.
Also the number of agents making 2 shifts decreased of about 15 workers since also some of them fall within that bracket.

Tax is not the only thing agents take into account when they decide how much to work, the *benefit* plays an important role.
We saw the decrease in the *tax revenue* when we increase the tax rate from 0.4 to 0.7 or more, and we gave an explanation of this; this has an effect on the *benefit*, it decreases.
We set *tax rate* back to 40% and run the model:

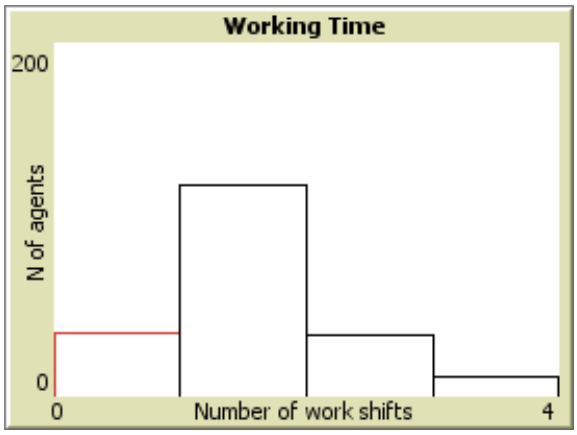
Figure 3



Input:
Num-people = 200; Tax_rate = 0.4;
income_bracket = 60; **benefit_bracket = 1**;
num-pretentious = 150; num-humble = 0
Output:
0 hours: 72 agents
4 hours: 78 agents
8 hours: 37 agents Benefit ≈ 13 €
12 hours: 13 agent Gini index = 0.36

Stop the model. My simulation is giving me a *benefit* which moves around 13 €, this benefit is distributed to every agent who decides or cannot work because the benefit is due to those with no income.
With these setting the *benefit* is big enough to make the difference in the choice between not working and working 1 shift, the numbers of agents choosing the first and the second is similar.

Let's give the benefit to a bigger number of people, set *benefit_bracket* to 60 and run the model:
Figure 4

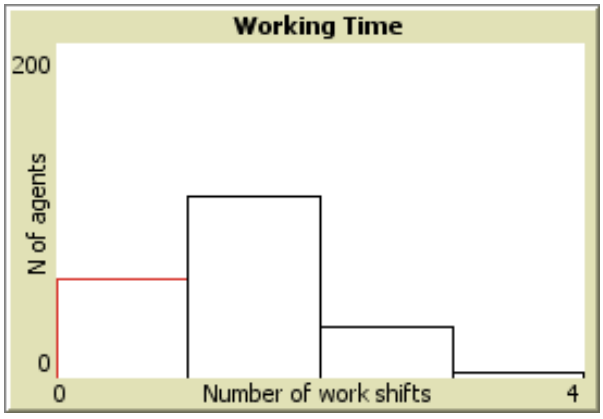


Input:
Num-people = 200; Tax_rate = 0.4;
income_bracket = 60; **benefit_bracket = 60**;
num-pretentious = 150; num-humble = 0
Output:
0 hours: 44 agents
4 hours: 115 agents
8 hours: 29 agents **Benefit ≈ 6**
12 hours: 12 agent

Stop the simulation. The *benefit* now moves around 6 euro, but most important, it is given also to all those making one work shift, since the maximum they can earn is 60 with a 15 euro wage per hour, therefore it is no more considered in the choice between not working or making one shift, there is no difference, you get the *benefit* either way.
It is important to notice that of the 44 agents not working, around 30 have no choice, they cannot work, and therefore the number of people who actually decides to live only on the benefit is lower.

We now introduce *humble* agents in our model; let's see the outcome, and the settings, we comment later:

Figure 5



Input:
Num-people = 200; Tax_rate = 0.4;
income_bracket = 60; **benefit_bracket = 1**;
num-pretentious = 0; num-humble = 150
Output:
0 hours: 59 agents
4 hours: 109 agents
8 hours: 30 agents **Benefit ≈ 9 €**
12 hours: 2 agent **Gini index = 0.24**

If you like to see the effects in your model, remember to *setup* before staring the simulation, this time it is necessary since we are changing different types of agent. Confront this with figure 3, the inputs are the same, except for the presence of *humble* agents instead of pretentious ones, but the outcome is very different; here more agents than with the setup of figure 3, prefer to work; we can still see the effect of the benefit, how it drives some agent into choosing not to work: the number of not working agents is still greater than that in figure 4 where the benefit is given to both not workers and workers making one shift, but the effect is diminished by the presence of *humble* agents.

I write again the extended utility function: Utility (*wage*, *hours*, ***consumption%***, *leisure*) =
(*hours* * *wage* * ***consumption%***) * (*maxConsumption* - *hours* * *leisure*)

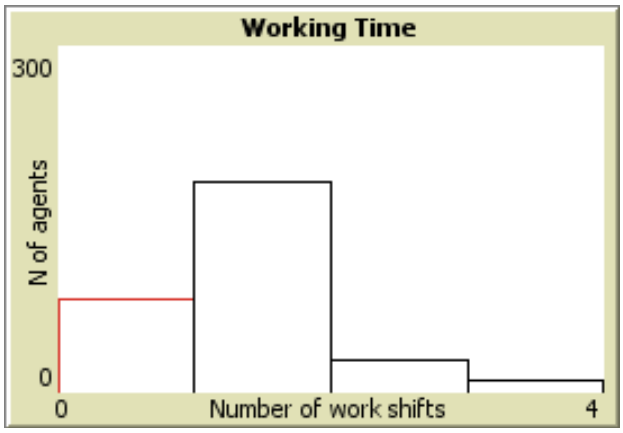
Humble agents consumes less on average than the other agents; their normal level of consumption is 90% of the net income as the others, but under certain circumstances they decrease the fraction of income to consume even down to 70%; this has a direct effect on the marginal utility than an hour more of work will give to them; these workers cannot enjoy 'all' the money they earn, therefore ceteris paribus they will prefer an hour off work, on right side consumption has no effect: (*maxConsumption* - *hours* * *leisure*).
Some could argue that the enjoyment of free time would lower too if consumption were to be cut, and they would make an extremely compelling argument since one consumes most of the money she earns during free time. I imagine that would partially drive the model back to the outcome in figure 3; but for how this model works, reducing the fraction of consumption smooth the effects of every settings the user sets, or, if you like, of every 'work reform' you make in this small *world*.

It is also worth noticing the lower *Gini index* in presence of *humble* agents.

Established the above, in order see clearly the effects of other 'policy' we will use pretentious agents to increase the effects of the chosen settings.

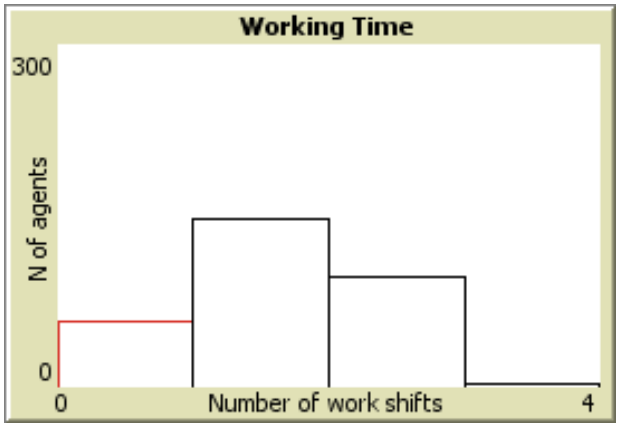
So far we focused more on the choice if working or not, finally we confront two cases to see how taxable income alone influences whether working or not and the work load chosen.
The following is a wonderful example of how complexity determines an outcome that would have either been difficult to predict.

Figure 6



Input:
Num-people = 300; Tax_rate = 0.6;
income_bracket = 40; **benefit_bracket = 40**; num-pretentious = 250; num-humble = 0
Output:
0 hours: 80 agents
4 hours: 183 agents
8 hours: 28 agents **Benefit ≈ 15 €**
12 hours: 9 agents

Figure 7



Input:
Num-people = 300; Tax_rate = 0.6;
income_bracket = 90; **benefit_bracket = 40**; num-pretentious = 250; num-humble = 0
Output:
0 hours: 56 agents
4 hours: 147 agents
8 hours: 96 agents **Benefit ≈ 5 €**
12 hours: 1 agent

From experiment 6 to 7 we decreased taxable income raising from 40 to 90 € *income_bracket*, a first and easy effect to predict is the drop in the *benefit*, let's consider the others:
A decrease in the number of agents not working and workers making 1 or 3 shifts.
An increase in the number of workers making 2 shifts.
Let's start reasoning on the one effect we were able to predict: the drop in the benefit, what does it imply? Who is receiving the benefit?
Benefit is being distributed to those who earns less than 40 euro per day, that includes unemployed and a part of the workers making 1 shift: if you work 4 hours a day, you must have a *wage* per hour lower than 10 € in order to receive the *benefit*, therefore more or less a half of these workers will receive a benefit. That support the outcome we saw; I believe some agent with a low wage and/or high leisure

who before preferred not to work, now switched to one shift a day, and some agent with a higher wage and/or lower leisure switched to 2 shifts a day.
That would partially explain the increase in the number of workers making 2 shifts.

Together with the previous statement, the following explain the high increase in the 2 shifts:
We have 8 agents who moved from the 12 hours shifts, it is possible that before, for these workers, to leave the high 60% *tax rate* they should have moved from 3 shifts to 1, which is a big jump, a jump that brings to such a low gross *income* which is hardly compensated by the increase in free time; now instead these agents find themselves with the possibility to make just one work shift less enough to exit, at least partially, the 60% tax rate and enjoying more free time, which now, given the lower earning they have to give up, in order to exit the *income bracket*, are willing to do so.

This last consideration is very interesting, we have some people who decided to work less, not because there was an increase in taxes, with a high probability most of these agents would not have been directly influenced by the rose in taxable income since it rose up to 90 € and are very few the workers belonging to the '3 shifts agent set' earning less than 90 € a day, already with a wage per hour higher than 7 € (from 8 to 15, so most of them) you do not earn less than 90 € a day, this means that there has been some agent earning at least more than 95 € a day who decided to earn less (the minimum they gave up is, in the case their *wage* was 8 euro, 32 € before taxes), in order to fall among those who do not pay taxes, or at least to have a smaller amount as taxable income.

If you like, to see this effect, run the model with settings in figure 6, stop the model after a while, then type in the command center *ask turtles with [hours = 12 and wage >= 8][ask patches in-radius 2 [set pcolor yellow]]* this will allow you to locate them, then set *income_bracket* up to 90 € and, without pressing setup, run again the model for a few steps, then stop it and look at the *hours* variable of those agents.