The Dynamics of International Exploitation

Accompanying Simulations

Jonathan F. Cogliano, Roberto Veneziani, and Naoki Yoshihara

Version: June 30, 2022

In[ ]:= [redacted]

Main Simulations

Model Setup

Capital Stock Data for Initial Wealth Endowments

In[ ]:=
```
Data = Import["./InitialEndowments_clean.csv"];
```

In[ ]:=
```
DataNoHeader = SortBy[Drop[Data, 1], #[[5]] &]
```

Out[ ]=
```
{{Burundi, 15963.5, 1.3893, 10.8642, 1469.37}, {Congo - Kinshasa, 175909., 1.666, 81.34, 2162.63},
 {Malawi, 43799.5, 1.9574, 18.6221, 2352.02}, {Mali, 55200.4, 1.3437, 18.542, 2977.05},
```

```
{Sierra Leone, 25355.6, 1.6133, 7.5572, 3355.15}, {Liberia, 16526.4, 1.811, 4.7319, 3492.56},
{Mozambique, 109190., 1.2177, 29.6688, 3680.29}, {Central African Republic, 19002.4, 1.5404, 4.6591, 4078.55},
{Madagascar, 110037., 1.7052, 25.5709, 4303.22}, {Niger, 96472.6, 1.2119, 21.4773, 4491.84},
{Rwanda, 55347.6, 1.838, 12.2084, 4533.57}, {Burkina Faso, 90906.7, 1.2585, 19.1934, 4736.35},
{Ethiopia, 684944., 1.4147, 104.957, 6525.92}, {Zimbabwe, 113144., 2.6482, 16.5299, 6844.83},
{Togo, 55997.2, 1.7909, 7.7977, 7181.24}, {Benin, 81619.9, 1.8408, 11.1757, 7303.34},
{Gambia, 16445.6, 1.6192, 2.1006, 7828.99}, {Kenya, 394330., 2.3053, 49.6999, 7934.22},
{Yemen, 227303., 1.7355, 28.2504, 8046.03}, {Uganda, 357990., 2.325, 42.863, 8351.96},
{Nepal, 250592., 1.7701, 29.305, 8551.18}, {Cambodia, 138382., 1.9022, 16.0054, 8645.97},
{Ivory Coast, 222652., 1.6573, 24.2948, 9164.58}, {Cameroon, 227090., 1.89, 24.0537, 9440.94},
{Pakistan, 1.94586×10^6, 1.7748, 197.016, 9876.66}, {Senegal, 182939, 1.5796, 15.8506, 11541.5},
{Myanmar, 643155., 1.8168, 53.3706, 12050.7}, {Nigeria, 2.46166×10^6, 1.9242, 190.886, 12896.},
{Mauritania, 58348.8, 1.7869, 4.4202, 13200.5}, {Bangladesh, 2.19523×10^6, 2.0488, 164.67, 13331.1},
{Kyrgyzstan, 81286.2, 3.4533, 6.0451, 13446.6}, {Tanzania, 773946., 1.6899, 55.7978, 13870.6},
{Haiti, 158114., 1.7037, 10.9812, 14398.6}, {Lesotho, 32740.4, 1.7582, 2.2333, 14660.1},
{Bolivia, 164294., 2.9092, 11.0516, 14866.1}, {Honduras, 149490., 2.3495, 9.2651, 16134.7},
{Vietnam, 1.55284×10^6, 2.7649, 95.5408, 16253.2}, {Egypt, 1.59062×10^6, 2.6177, 97.5532, 16305.2},
{Belize, 6526.4, 3.5291, 0.3747, 17417.7}, {Nicaragua, 121678., 2.2414, 6.2176, 19570.},
{El Salvador, 129833., 2.2393, 6.3779, 20356.8}, {Guatemala, 349707., 1.9099, 16.9135, 20676.2},
{Sudan, 846309., 1.5911, 40.5333, 20879.3}, {Syria, 382274., 2.5734, 18.2699, 20923.7},
{Laos, 150884., 1.9112, 6.8582, 22000.5}, {Zambia, 378445., 2.6024, 17.0941, 22138.9},
{Moldova, 93525.5, 3.4504, 4.0512, 23085.9}, {Fiji, 22156.6, 2.6811, 0.9055, 24469.},
{India, 33385420, 2.1238, 1339.18, 24929.7}, {Iraq, 1.0465×10^6, 2.2628, 38.2746, 27342.},
{Philippines, 2.87783×10^6, 2.6869, 104.918, 27429.3}, {Paraguay, 188595., 2.6048, 6.8113, 27688.5},
{Armenia, 89271.5, 3.1305, 2.9304, 30463.9}, {Ghana, 918879., 2.4648, 28.8336, 31868.4},
{Jordan, 322888., 2.8746, 9.7024, 33279.2}, {Congo - Brazzaville, 185261., 2.0402, 5.2607, 35216.},
{Angola, 1.08025×10^6, 1.4673, 29.7842, 36269.3}, {Eswatini, 50796.3, 2.0301, 1.3673, 37150.8},
{Peru, 1.20156×10^6, 2.8029, 32.1655, 37355.5}, {Costa Rica, 230187., 2.658, 4.9058, 46921.5},
{Sri Lanka, 981131, 2.8639, 20.8769, 46996.}, {Morocco, 1.71027×10^6, 1.8929, 35.7396, 47853.5},
{Namibia, 124294., 2.2251, 2.5338, 49054.5}, {Ukraine, 2.1785×10^6, 3.2724, 44.2229, 49261.9},
```

{Colombia, 2.4444×10^{6}, 2.563, 49.0656, 49 818.9}, {Tajikistan, 446 238., 3.0509, 8.9213, 50 019.4},
{Gabon, 101 352., 2.7845, 2.0251, 50 047.9}, {South Africa, 2.88631×10^{6}, 2.8092, 56.7172, 50 889.5},
{Mongolia, 156 974., 2.9921, 3.0756, 51 038.5}, {Maldives, 22 589.5, 2.3599, 0.4363, 51 775.2},
{Argentina, 2 364 575, 3.0354, 44.271, 53 411.4}, {Algeria, 2.2363×10^{6}, 2.3028, 41.3181, 54 124.},
{Dominican Republic, 591 543., 2.7196, 10.767, 54 940.3}, {Jamaica, 162 147., 2.5891, 2.8903, 56 100.4},
{Ecuador, 957 913., 2.7496, 16.6249, 57 619.2}, {Bulgaria, 420 190., 3.1628, 7.0846, 59 310.3},
{Tunisia, 714 257., 2.611, 11.5321, 61 936.4}, {Kazakhstan, 1.14236×10^{6}, 3.1358, 18.2045, 62 751.4},
{Serbia, 461 666., 3.391, 7.0347, 65 626.9}, {Albania, 194 145., 2.9512, 2.9302, 66 256.5},
{Iran, 5.4493×10^{6}, 2.4239, 81.1628, 67 140.3}, {Indonesia, 17 984 676, 2.3161, 263.991, 68 126.},
{Poland, 2 638 046, 3.4042, 38.1707, 69 111.8}, {Mexico, 9 272 062, 2.7365, 129.163, 71 785.6},
{Thailand, 5 103 972, 2.7435, 69.0375, 73 930.4}, {China, 105 849 328, 2.5664, 1409.52, 75 096.1},
{Barbados, 21 503.2, 2.8357, 0.2857, 75 265.1}, {Brazil, 15 796 820, 2.9492, 209.288, 75 478.8},
{Panama, 318 240., 2.8572, 4.0986, 77 646.1}, {Venezuela, 2.69051×10^{6}, 2.8227, 31.9771, 84 138.6},
{Chile, 1.5342×10^{6}, 3.1079, 18.0547, 84 975.4}, {Mauritius, 107 904., 2.6031, 1.2651, 85 292.7},
{Uruguay, 301 882., 2.7313, 3.4567, 87 332.4}, {Malaysia, 2 807 267, 3.0341, 31.6243, 88 769.3},
{Botswana, 212 074., 2.8853, 2.2917, 92 539.9}, {Romania, 1.85761×10^{6}, 3.2337, 19.6793, 94 394.},
{Turkey, 8 341 127, 2.4443, 80.745, 103 302.}, {Lithuania, 326 608., 3.2627, 2.8903, 113 001.},
{Russia, 16 505 067, 3.403, 143.99, 114 627.}, {Malta, 53 994.7, 3.1252, 0.4308, 125 336.},
{Slovakia, 689 147., 3.7938, 5.4477, 126 502.}, {New Zealand, 617 120., 3.3607, 4.7058, 131 140.},
{Croatia, 556 734., 3.5237, 4.1894, 132 891.}, {Israel, 1.1703×10^{6}, 3.8071, 8.3216, 140 634.},
{Estonia, 185 511., 3.6161, 1.3096, 141 655.}, {Hungary, 1.422×10^{6}, 3.385, 9.7216, 146 273.},
{Kuwait, 709 136., 2.2432, 4.1365, 171 434.}, {South Korea, 8 896 485, 3.6945, 50.9822, 174 502.},
{Taiwan, 4.17467×10^{6}, 3.2907, 23.5555, 177 227.}, {Japan, 22 900 656, 3.5723, 127.485, 179 635.},
{United States, 60 923 964, 3.7387, 324.46, 187 771.}, {Trinidad and Tobago, 264 846., 3.0679, 1.3691, 193 445.},
{Finland, 1.10275×10^{6}, 3.4662, 5.5232, 199 659.}, {United Kingdom, 13 221 941, 3.7578, 66.1816, 199 783.},
{Cyprus, 172 999., 2.8629, 0.8552, 202 290.}, {Latvia, 397 775., 3.1271, 1.9497, 204 018.},
{Saudi Arabia, 6.74351×10^{6}, 2.668, 32.9382, 204 732.}, {Bahrain, 320 046., 2.2372, 1.4926, 214 422.},
{Czech Republic, 2.2789×10^{6}, 3.6658, 10.6183, 214 620.}, {Slovenia, 453 281., 3.5322, 2.08, 217 923.},
{Greece, 2.4423×10^{6}, 3.0913, 11.1598, 218 848.}, {Canada, 8 071 759, 3.7063, 36.6242, 220 394.},
{Australia, 5.4213×10^{6}, 3.5225, 24.4506, 221 724.}, {France, 15 062 345, 3.1907, 67.2263, 224 054.},

```
{Spain, 10 450 522, 2.9439, 46.3543, 225 449.}, {Iceland, 76 263.6, 3.2278, 0.335, 227 653.},
{Germany, 19 033 204, 3.6705, 82.1142, 231 789.}, {Portugal, 2.39843 × 10^6, 2.4783, 10.3295, 232 192.},
{Sweden, 2.3058 × 10^6, 3.4159, 9.9107, 232 657.}, {Netherlands, 4.16323 × 10^6, 3.3685, 17.0359, 244 380.},
{Denmark, 1.49905 × 10^6, 3.563, 5.7336, 261 451.}, {Belgium, 3.03226 × 10^6, 3.1364, 11.4293, 265 306.},
{Hong Kong, 1.97186 × 10^6, 3.239, 7.3649, 267 737.}, {Ireland, 1.29868 × 10^6, 3.1507, 4.7617, 272 735.},
{Italy, 16 525 972, 3.1217, 59.3599, 278 403.}, {Austria, 2.51829 × 10^6, 3.359, 8.7355, 288 282.},
{Switzerland, 2 498 143, 3.6875, 8.476, 294 731.}, {Norway, 1.57407 × 10^6, 3.6431, 5.3054, 296 692.},
{United Arab Emirates, 2.9535 × 10^6, 2.7401, 9.4001, 314 199.}, {Macao, 200 069., 2.8635, 0.6226, 321 344.},
{Brunei, 148 226., 2.7749, 0.4287, 345 758.}, {Singapore, 2.09948 × 10^6, 3.9742, 5.7088, 367 762.},
{Luxembourg, 219 834., 3.5125, 0.5835, 376 751.}, {Qatar, 1.34575 × 10^6, 3.0921, 2.6392, 509 907.}}
```

In[ ]:=
```
CountryList = DataNoHeader[[All, 1]];
```

In[ ]:=
```
InitialCapital = DataNoHeader[[All, 2]]
```

Out[ ]=
```
{15 963.5, 175 909., 43 799.5, 55 200.4, 25 355.6, 16 526.4, 109 190., 19 002.4, 110 037., 96 472.6, 55 347.6, 90 906.7,
 684 944., 113 144., 55 997.2, 81 619.9, 16 445.6, 394 330., 227 303., 357 990., 250 592., 138 382., 222 652., 227 090.,
 1.94586 × 10^6, 182 939, 643 155., 2.46166 × 10^6, 58 348.8, 2.19523 × 10^6, 81 286.2, 773 946., 158 114., 32 740.4,
 164 294., 149 490., 1.55284 × 10^6, 1.59062 × 10^6, 6526.4, 121 678., 129 833., 349 707., 846 309., 382 274., 150 884.,
 378 445., 93 525.5, 22 156.6, 33 385 420, 1.0465 × 10^6, 2.87783 × 10^6, 188 595., 89 271.5, 918 879., 322 888., 185 261.,
 1.08025 × 10^6, 50 796.3, 1.20156 × 10^6, 230 187., 981 131, 1.71027 × 10^6, 124 294., 2.1785 × 10^6, 2.4444 × 10^6,
 446 238., 101 352., 2.88631 × 10^6, 156 974., 22 589.5, 2 364 575, 2.2363 × 10^6, 591 543., 162 147., 957 913., 420 190.,
 714 257., 1.14236 × 10^6, 461 666., 194 145., 5.4493 × 10^6, 17 984 676, 2 638 046, 9 272 062, 5 103 972, 105 849 328,
 21 503.2, 15 796 820, 318 240., 2.69051 × 10^6, 1.5342 × 10^6, 107 904., 301 882., 2 807 267, 212 074., 1.85761 × 10^6,
 8 341 127, 326 608., 16 505 067, 53 994.7, 689 147., 617 120., 556 734., 1.1703 × 10^6, 185 511., 1.422 × 10^6, 709 136.,
 8 896 485, 4.17467 × 10^6, 22 900 656, 60 923 964, 264 846., 1.10275 × 10^6, 13 221 941, 172 999., 397 775., 6.74351 × 10^6,
 320 046., 2.2789 × 10^6, 453 281., 2.4423 × 10^6, 8 071 759, 5.4213 × 10^6, 15 062 345, 10 450 522, 76 263.6, 19 033 204,
 2.39843 × 10^6, 2.3058 × 10^6, 4.16323 × 10^6, 1.49905 × 10^6, 3.03226 × 10^6, 1.97186 × 10^6, 1.29868 × 10^6, 16 525 972,
 2.51829 × 10^6, 2 498 143, 1.57407 × 10^6, 2.9535 × 10^6, 200 069., 148 226., 2.09948 × 10^6, 219 834., 1.34575 × 10^6}
```

In[ ]:=
```
LaborEndowment = DataNoHeader[[All, 3]] × DataNoHeader[[All, 4]] 100 000
```

Out[ ]=

{1.50936×10^{6}, 1.35512×10^{7}, 3.64509×10^{6}, 2.49149×10^{6}, 1.2192×10^{6}, 856 947., 3.61277×10^{6}, 717 688., 4.36035×10^{6}, 2.60283×10^{6}, 2.2439×10^{6}, 2.41549×10^{6}, 1.48483×10^{7}, 4.37745×10^{6}, 1.39649×10^{6}, 2.05722×10^{6}, 340 129., 1.14573×10^{7}, 4.90286×10^{6}, 9.96565×10^{6}, 5.18728×10^{6}, 3.04455×10^{6}, 4.02638×10^{6}, 4.54615×10^{6}, 3.49664×10^{7}, 2.50376×10^{6}, 9.69637×10^{6}, 3.67303×10^{7}, 789 846., 3.37375×10^{7}, 2.08755×10^{6}, 9.42927×10^{6}, 1.87087×10^{6}, 392 659., 3.21513×10^{6}, 2.17684×10^{6}, 2.64161×10^{7}, 2.55365×10^{7}, 132 235., 1.39361×10^{6}, 1.4282×10^{6}, 3.23031×10^{6}, 6.44925×10^{6}, 4.70158×10^{6}, 1.31074×10^{6}, 4.44857×10^{6}, 1.39783×10^{6}, 242 774., 2.84415×10^{8}, 8.66078×10^{6}, 2.81904×10^{7}, 1.77421×10^{6}, 917 362., 7.10691×10^{6}, 2.78905×10^{6}, 1.07329×10^{6}, 4.37024×10^{6}, 277 576., 9.01567×10^{6}, 1.30396×10^{6}, 5.97894×10^{6}, 6.76515×10^{6}, 563 796., 1.44715×10^{7}, 1.25755×10^{7}, 2.7218×10^{6}, 563 889., 1.5933×10^{7}, 920 250., 102 962., 1.3438×10^{7}, 9.51473×10^{6}, 2.92819×10^{6}, 748 328., 4.57118×10^{6}, 2.24072×10^{6}, 3.01103×10^{6}, 5.70857×10^{6}, 2.38547×10^{6}, 864 761., 1.96731×10^{7}, 6.1143×10^{7}, 1.29941×10^{7}, 3.53455×10^{7}, 1.89404×10^{7}, 3.61739×10^{8}, 81 015.9, 6.17233×10^{7}, 1.17105×10^{6}, 9.02618×10^{6}, 5.61122×10^{6}, 329 318., 944 128., 9.59513×10^{6}, 661 224., 6.3637×10^{6}, 1.97365×10^{7}, 943 018., 4.89997×10^{7}, 134 634., 2.06675×10^{6}, 1.58148×10^{6}, 1.47622×10^{6}, 3.16812×10^{6}, 473 564., 3.29076×10^{6}, 927 900., 1.88354×10^{7}, 7.75141×10^{6}, 4.55413×10^{7}, 1.21306×10^{8}, 420 026., 1.91445×10^{6}, 2.48697×10^{7}, 244 835., 609 691., 8.78791×10^{6}, 333 924., 3.89246×10^{6}, 734 698., 3.44983×10^{6}, 1.3574×10^{7}, 8.61272×10^{6}, 2.14499×10^{7}, 1.36462×10^{7}, 108 131., 3.014×10^{7}, 2.55996×10^{6}, 3.3854×10^{6}, 5.73854×10^{6}, 2.04288×10^{6}, 3.58469×10^{6}, 2.38549×10^{6}, 1.50027×10^{6}, 1.85304×10^{7}, 2.93425×10^{6}, 3.12553×10^{6}, 1.93281×10^{6}, 2.57572×10^{6}, 178 282., 118 960., 2.26879×10^{6}, 204 954., 816 067.}

In[ ]:=
```
Length[InitialCapital]
```

Out[ ]=

144

Initial Parameters

In[ ]:=
```
Clear[N, A, L, b, l, w, p];
N = Length[InitialCapital]; (* Number of agents *)
A = 0.75; (* Technology *)
L = 0.5; (* Labour requirement of technology *)
b = 0.44; (* Subsistence bundle *)
l = LaborEndowment; (* Individual labour endowment *)
```

In[ ]:=
```
T = 50;
```

Countries/Agents and Endowments

In[ ]:=
```
Total[InitialCapital]
```

Out[ ]=

5.11615×10^{8}

In[ ]:=
```
Total[l] ≥ L A^-1 Total[InitialCapital]
```

Out[ ]=

True

In[ ]:=

$$\frac{L\,A^{-1}\,\text{Total}[\text{InitialCapital}]}{\text{Total}[l]}$$

Out[ ]=

0.184985

In[ ]:=
```
Clear[AgentSet];
(* AgentSet=Table[Table[{0,0},{N}],{T}]; *)
AgentSet = Table[{InitialCapital[[i]], LaborEndowment[[i]]}, {i, 1, Length[CountryList]}];
AgentSet // TableForm
```

Out[ ]//TableForm=

15 963.5	1.50936×10^{6}
175 909.	1.35512×10^{7}

43 799.5	3.64509×10^6
55 200.4	2.49149×10^6
25 355.6	1.2192×10^6
16 526.4	856 947.
109 190.	3.61277×10^6
19 002.4	717 688.
110 037.	4.36035×10^6
96 472.6	2.60283×10^6
55 347.6	2.2439×10^6
90 906.7	2.41549×10^6
684 944.	1.48483×10^7
113 144.	4.37745×10^6
55 997.2	1.39649×10^6
81 619.9	2.05722×10^6
16 445.6	340 129.
394 330.	1.14573×10^7
227 303.	4.90286×10^6
357 990.	9.96565×10^6
250 592.	5.18728×10^6
138 382.	3.04455×10^6
222 652.	4.02638×10^6
227 090.	4.54615×10^6
1.94586×10^6	3.49664×10^7
182 939	2.50376×10^6
643 155.	9.69637×10^6
2.46166×10^6	3.67303×10^7
58 348.8	789 846.
2.19523×10^6	3.37375×10^7
81 286.2	2.08755×10^6
773 946.	9.42927×10^6
158 114.	1.87087×10^6
32 740.4	392 659.

164 294.	3.21513×10^{6}
149 490.	2.17684×10^{6}
1.55284×10^{6}	2.64161×10^{7}
1.59062×10^{6}	2.55365×10^{7}
6526.4	132 235.
121 678.	1.39361×10^{6}
129 833.	1.4282×10^{6}
349 707.	3.23031×10^{6}
846 309.	6.44925×10^{6}
382 274.	4.70158×10^{6}
150 884.	1.31074×10^{6}
378 445.	4.44857×10^{6}
93 525.5	1.39783×10^{6}
22 156.6	242 774.
33 385 420	2.84415×10^{8}
1.0465×10^{6}	8.66078×10^{6}
2.87783×10^{6}	2.81904×10^{7}
188 595.	1.77421×10^{6}
89 271.5	917 362.
918 879.	7.10691×10^{6}
322 888.	2.78905×10^{6}
185 261.	1.07329×10^{6}
1.08025×10^{6}	4.37024×10^{6}
50 796.3	277 576.
1.20156×10^{6}	9.01567×10^{6}
230 187.	1.30396×10^{6}
981 131	5.97894×10^{6}
1.71027×10^{6}	6.76515×10^{6}
124 294.	563 796.
2.1785×10^{6}	1.44715×10^{7}
2.4444×10^{6}	1.25755×10^{7}
446 238.	2.7218×10^{6}

101 352.	563 889.
2.88631×10^{6}	1.5933×10^{7}
156 974.	920 250.
22 589.5	102 962.
2 364 575	1.3438×10^{7}
2.2363×10^{6}	9.51473×10^{6}
591 543.	2.92819×10^{6}
162 147.	748 328.
957 913.	4.57118×10^{6}
420 190.	2.24072×10^{6}
714 257.	3.01103×10^{6}
1.14236×10^{6}	5.70857×10^{6}
461 666.	2.38547×10^{6}
194 145.	864 761.
5.4493×10^{6}	1.96731×10^{7}
17 984 676	6.1143×10^{7}
2 638 046	1.29941×10^{7}
9 272 062	3.53455×10^{7}
5 103 972	1.89404×10^{7}
105 849 328	3.61739×10^{8}
21 503.2	81 015.9
15 796 820	6.17233×10^{7}
318 240.	1.17105×10^{6}
2.69051×10^{6}	9.02618×10^{6}
1.5342×10^{6}	5.61122×10^{6}
107 904.	329 318.
301 882.	944 128.
2 807 267	9.59513×10^{6}
212 074.	661 224.
1.85761×10^{6}	6.3637×10^{6}
8 341 127	1.97365×10^{7}
326 608.	943 018.
16 505 067	4.89997×10^{7}

53 994.7	134 634.
689 147.	2.06675×10^{6}
617 120.	1.58148×10^{6}
556 734.	1.47622×10^{6}
1.1703×10^{6}	3.16812×10^{6}
185 511.	473 564.
1.422×10^{6}	3.29076×10^{6}
709 136.	927 900.
8 896 485	1.88354×10^{7}
4.17467×10^{6}	7.75141×10^{6}
22 900 656	4.55413×10^{7}
60 923 964	1.21306×10^{8}
264 846.	420 026.
1.10275×10^{6}	1.91445×10^{6}
13 221 941	2.48697×10^{7}
172 999.	244 835.
397 775.	609 691.
6.74351×10^{6}	8.78791×10^{6}
320 046.	333 924.
2.2789×10^{6}	3.89246×10^{6}
453 281.	734 698.
2.4423×10^{6}	3.44983×10^{6}
8 071 759	1.3574×10^{7}
5.4213×10^{6}	8.61272×10^{6}
15 062 345	2.14499×10^{7}
10 450 522	1.36462×10^{7}
76 263.6	108 131.
19 033 204	3.014×10^{7}
2.39843×10^{6}	2.55996×10^{6}
2.3058×10^{6}	3.3854×10^{6}
4.16323×10^{6}	5.73854×10^{6}
1.49905×10^{6}	2.04288×10^{6}

3.03226×10^6	3.58469×10^6
1.97186×10^6	2.38549×10^6
1.29868×10^6	1.50027×10^6
16 525 972	1.85304×10^7
2.51829×10^6	2.93425×10^6
2 498 143	3.12553×10^6
1.57407×10^6	1.93281×10^6
2.9535×10^6	2.57572×10^6
200 069.	178 282.
148 226.	118 960.
2.09948×10^6	2.26879×10^6
219 834.	204 954.
1.34575×10^6	816 067.

Optimisation Routine

Capital Constrained RS

When the simulation is capital constrained and $\mathrm{N}\, l > L\, A^{-1} \sum \omega_{t-1}^{v}$ it is not possible for $z_t^v = l^v$ for all v and $\mathrm{N}\, l > L\, A^{-1} \sum \omega_{t-1}^{v}$ implies that $w_t = b$ and agents are indifferent concerning their labour supply. By Lemma 1 we can analyze solutions to the agents' maximisation problem with $x_t^v = 0$. Moreover, $w_t = b$ implies $r_t > 0$ and so at any optimal solution it must be $\delta_t^v = 0$ for all v. Thus in the capital constrained case the following procedure can be used to find $(x_t^v; y_t^v; z_t^v)$ given (p_t, r_t):

Step 1: set $x_t^v = \delta_t^v = 0 \; \forall \; v \in \mathcal{N}$

Step 2: set $z_t^v = \frac{L\, A^{-1} \sum \omega_{t-1}^{v}}{l_t}\, l^v \; \forall \; v \in \mathcal{N}$

Step 3: set $y_t^v = A^{-1}\, \omega_{t-1}^{v} \; \forall \; v \in \mathcal{N}$

```
In[•]:= CapitalConstrained[p_, A_, L_, l_, t_, r_, b_] := (
  Activities⟦t, All, 1⟧ = 0.;
  Activities⟦t, All, 4⟧ = 0.;
  Activities⟦t, All, 2⟧ = (A^-1 (If[t == 1, Total[AgentSet⟦All, 1⟧], Total[Activities⟦t - 1, All, 5⟧]])) l/Total[l];
  Activities⟦t, All, 3⟧ = (If[t == 1, AgentSet⟦All, 1⟧, Activities⟦t - 1, All, 5⟧]);
  Activities⟦t, All, 6⟧ = L Activities⟦t, All, 1⟧ + L Activities⟦t, All, 2⟧;
  Activities⟦t, All, 5⟧ = (1 + r) (A Activities⟦t, All, 1⟧ + Activities⟦t, All, 3⟧) +
    ((1 - A (1 + r))/L - b) L (Activities⟦t, All, 1⟧ + Activities⟦t, All, 2⟧)
  (* 1/p (p Activities⟦t,All,1⟧-p Activities⟦t,All,1⟧+
      (p-(1+r) p A) Activities⟦t,All,2⟧+r Activities⟦t,All,3⟧-p b Activities⟦t,All,6⟧); *)
 )
```

Labour Constrained RS

When the simulation is labour constrained and $\mathrm{N}\, l < L\, A^{-1} \sum \omega_{t-1}^{v}$ it is not possible that $A\, y_t^v = \omega_{t-1}^v$ for all v and $\mathrm{N}\, l < L\, A^{-1} \sum \omega_{t-1}^{v}$ implies that $r_t = 0$. Thus agents are indifferent between investing their wealth in productive activities or carrying it to the end of the period and selling it. Hence, $\delta_t^v \in \left[0, A^{-1}\, \omega_{t-1}^v\right]$ is potentially part of the optimal solutions to MP_t^v. By Lemma 1 as in the capital constrained case above, the following procedure can be used to find the optimal $(x_t^v; y_t^v; z_t^v)$ for all v.

Step 1: set $x_t^v = 0\ \forall\ v \in \mathcal{N}$

Step 2: set $z_t^v = l^v\ \forall\ v \in \mathcal{N}$

Step 3: set $y_t^v = \frac{l_t}{L\, A^{-1} \sum \omega_{t-1}^v} A^{-1}\, \omega_{t-1}^v\ \forall\ v \in \mathcal{N}$

Step 4: set $\delta_t^v = \omega_{t-1}^v - \frac{l_t}{L\, A^{-1}\, \omega_{t-1}}\, \omega_{t-1}^v = \left(1 - \frac{l_t}{L\, A^{-1}\, \omega_{t-1}}\right) \omega_{t-1}^v\ \forall\ \in \mathcal{N}$

```
In[ ]:= LabourConstrained[p_, A_, L_, l_, t_, r_, b_] := (
  Activities[[t, All, 1]] = 0.;
  Activities[[t, All, 2]] = L^-1 l;
  Activities[[t, All, 3]] = If[t == 1, AgentSet[[All, 1]], Activities[[t - 1, All, 5]]]
     Total[l] / (L A^-1 If[t == 1, Total[AgentSet[[All, 1]]], Total[Activities[[t - 1, All, 5]]]]);
  Activities[[t, All, 4]] = If[t == 1, Total[AgentSet[[All, 1]]], Total[Activities[[t - 1, All, 5]]]] - Activities[[t, All, 3]];
  Activities[[t, All, 6]] = L Activities[[t, All, 1]] + L Activities[[t, All, 2]];
  Activities[[t, All, 5]] = (1 + r) (A Activities[[t, All, 1]] + Activities[[t, All, 3]]) +
    ((1 - A (1 + r)) / L - b) L (Activities[[t, All, 1]] + Activities[[t, All, 2]])
  (* 1/p (p Activities[[t,All,1]]-p Activities[[t,All,1]]+
      (p-(1+r) p A) Activities[[t,All,2]]+r Activities[[t,All,3]]-p b Activities[[t,All,6]]); *)
 )
```

Knife-Edge RS

In the knife-edge case when $N\,l = L\,A^{-1} \sum \omega_{t-1}^{v}$, by Lemma 1 as above, the following steps can be used to calculate the optimal $(x_t^v; y_t^v; z_t^v)$.

Step 1: set $x_t^v = \delta_t^v = 0 \;\forall\; v \in \mathcal{N}$

Step 2: set $z_t^v = l^v \;\forall\; v \in \mathcal{N}$

Step 3: set $y_t^v = A^{-1}\,\omega_{t-1}^v \;\forall\; v \in \mathcal{N}$

```
In[•]:= KnifeEdge[p_, A_, L_, l_, t_, r_, b_] := (
  Activities⟦t, All, 1⟧ = 0.;
  Activities⟦t, All, 4⟧ = 0.;
  Activities⟦t, All, 2⟧ = L^-1 l;
  Activities⟦t, All, 3⟧ = (If[t == 1, AgentSet⟦All, 1⟧, Activities⟦t - 1, All, 5⟧]);
  Activities⟦t, All, 6⟧ = L Activities⟦t, All, 1⟧ + L Activities⟦t, All, 2⟧;
  Activities⟦t, All, 5⟧ = (1 + r) (A Activities⟦t, All, 1⟧ + Activities⟦t, All, 3⟧) +
    ((1 - A (1 + r))/L - b) L (Activities⟦t, All, 1⟧ + Activities⟦t, All, 2⟧)
  (* 1/p (p Activities⟦t,All,1⟧-p Activities⟦t,All,1⟧+
      (p-(1+r) p A) Activities⟦t,All,2⟧+r Activities⟦t,All,3⟧-p b Activities⟦t,All,6⟧); *)
 )
```

Prices, Wages, and Values

Prices and Wages

A reproducible solution to the simulation procedure consists of a vector (***p***, ***w***) that meets certain conditions described in the associated paper. The determination of $p_t \in \boldsymbol{p}$ and $w_t \in \boldsymbol{w}\ \forall\ t$ has flexibility. Since we are dealing with a single good economy a straightforward choice is $p_t = 1$ for all $t \in T$.

```
In[•]:= (Price = {
  p == (1 + r) p a + w ℓ
    } /. p → 1) // TableForm
Out[•]//TableForm=
  1 == a (1 + r) + w ℓ
```

In[ ]:=
```
TestWage = Solve[Price, w][[1]] // Simplify
```

Out[ ]=

$$\left\{w \to \frac{1 - a\,(1 + r)}{\ell}\right\}$$

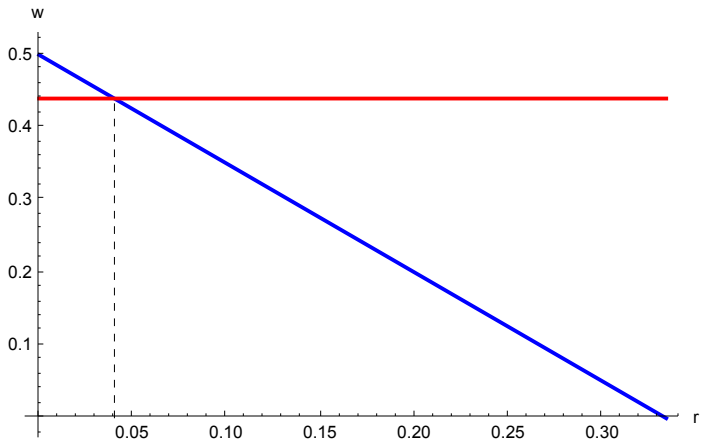
In[ ]:=
```
TestWage /. {p → 1, a → A, ℓ → L, r → 0.05}
```

Out[ ]=

$$\{w \to 0.425\}$$

In[ ]:=
```
Quiet[Plot[{w /. Solve[Price, w][[1]] /. {p → 1, a → A, ℓ → L}, b}, {r, 0, (1 - A)/A}, PlotStyle → {{Thick, Blue}, {Thick, Red}},
  AxesLabel → {"r", "w"}, Epilog → {Dashed, Black, Line[{{r /. Solve[Price, r][[1]] /. {p → 1, a → A, ℓ → L} /. w → b, 0},
      {r /. Solve[Price, r][[1]] /. {p → 1, a → A, ℓ → L} /. w → b, b}}]}]]
```

Out[ ]=



In[ ]:=
```
TestProfit = Solve[Price, r][[1]]
```

Out[ ]=

$$\left\{r \to \frac{1 - a - w\,\ell}{a}\right\}$$

Values

In[ ]:=
```
(EmbodiedValues = Flatten[{
    Solve[v == L (1 - A)^-1, v][[1]]
  }]) // TableForm
```

Out[ ]//TableForm=

v → 2.

Testing that 1 > *v b*

In[ ]:=
```
1 > (v /. EmbodiedValues) b
```

Out[ ]=

True

Basic Model

In[ ]:=
```
Clear[Activities, i, Wage, Profit, u];
(Activities = Table[Table[{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}, {N}], {T}]);
Wage = Table[0.0, {T}];
Profit = Table[0.0, {T}];

(* First Stage t=1 simulation *)
(* Setting w1 and π1 *)
If[Total[l] ≥ L A^-1 Total[AgentSet[[All, 1]]],
  If[Total[l] > L A^-1 Total[AgentSet[[All, 1]]], Wage[[1]] = b, Wage[[1]] = RandomReal[{b, (1 - A (1 + 0))/L}]],
  Wage[[1]] = (1 - A (1 + 0))/L];
Profit[[1]] = ((p - p0 A - Wage[[1]] L)/(p0 A)) /. {p → 1, p0 → 1};
```

```
(* Checking whether the simulation is capital constrained, labour constrained,
or on the knife-edge and calling the corresponding function to find optimal (x_t^ν, y_t^ν, z_t^ν, δ_t^ν) for all ν and t=1*)
If[Total[l] ≥ L A^-1 Total[AgentSet⟦All, 1⟧],
  If[Total[l] > L A^-1 Total[AgentSet⟦All, 1⟧],
   CapitalConstrained[1, A, L, l, 1, Profit⟦1⟧, b], KnifeEdge[1, A, L, l, 1, Profit⟦1⟧, b]],
  LabourConstrained[1, A, L, l, 1, Profit⟦1⟧, b]
 ];

(* Simulation for remaining T-1 time steps *)
For[t = 2, t ≤ T, t++,

 (* Updating w_t *)
 If[Total[l] ≥ L A^-1 Total[Activities⟦t - 1, All, 5⟧], If[Total[l] > L A^-1 Total[Activities⟦t - 1, All, 5⟧],
   Wage⟦t⟧ = b, Wage⟦t⟧ = RandomReal[{b, ((1 - A (1 + r))/L /. r → 0)}]], Wage⟦t⟧ = ((1 - A (1 + r))/L /. r → 0)];

 (* Updating r_t *)
 Profit⟦t⟧ = ((p - p0 A - Wage⟦t⟧ L)/(p0 A)) /. {p → 1, p0 → 1};

 (* Checking whether the simulation is capital constrained, labour constrained,
 or on the knife-edge and calling the corresponding function to find optimal (x_t^ν, y_t^ν, z_t^ν, δ_t^ν) for all ν *)
 If[Total[l] ≥ L A^-1 Total[Activities⟦t - 1, All, 5⟧],
  If[Total[l] > L A^-1 Total[Activities⟦t - 1, All, 5⟧],
   CapitalConstrained[1, A, L, l, t, Profit⟦t⟧, b], KnifeEdge[1, A, L, l, t, Profit⟦t⟧, b]],
  LabourConstrained[1, A, L, l, t, Profit⟦t⟧, b]
 ];

]
```

Some Results

Some results for the simulation run over T can be found below. The set of figures shows the aggregate activity levels for (x, y, z, δ), total net output, net output per capita, wealth, the growth rate of endowments g_t, w_t and π_t. The point at which the simulation becomes labour constrained and $\pi_t = 0$ is denoted by the dashed vertical lines in the plots below.

In[ ]:=
```
Clear[q];
t = 1;
While[Profit[[t]] > 0, q = t; t++]
q
```
Out[ ]=
```
44
```

In[ ]:=
```
Profit[[q + 1]]
```
Out[ ]=
```
0.
```

The exact t at which the simulation is labour constrained is given below:

In[ ]:=
```
q + 1
```
Out[ ]=
```
45
```

In[ ]:=
```
s1 = 8
```
Out[ ]=
```
8
```

In[ ]:=
```
s2 = 800
```
Out[ ]=
```
800
```

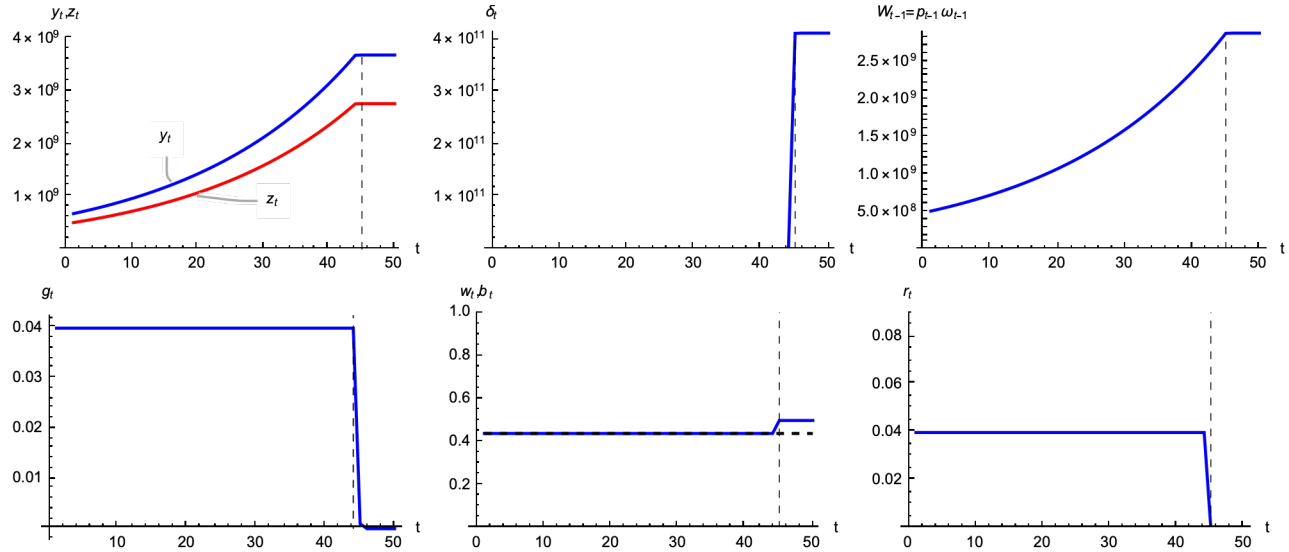
In[ ]:=
```
Clear[Output, Wealth, WealthPlot, XLevel, YLevel, ZLevel, Deltas, j, k];
Output = Table[0.0, {T}];
Wealth = Table[0.0, {T}];
WealthPlot = Table[0.0, {T}];
```

```
XLevel = Table[0.0, {T}];
YLevel = Table[0.0, {T}];
ZLevel = Table[0.0, {T}];
Deltas = Table[0.0, {T}];
For[j = 1, j ≤ T, j ++,
 Output⟦j⟧ = (p x - p1 A x + (p - p1 A (1 + r)) y + (1 + r) z - z) /. {p → 1, p1 → 1, r → Profit⟦j⟧,
    x → Total[Activities⟦j, All, 1⟧], y → Total[Activities⟦j, All, 2⟧], z → Total[Activities⟦j, All, 3⟧]}
]

For[k = 1, k ≤ T, k ++,
 Wealth⟦k⟧ = Total[Activities⟦k, All, 5⟧]
]
WealthPlot⟦1⟧ = Total[AgentSet⟦All, 1⟧];
For[k = 2, k ≤ T, k ++,
 WealthPlot⟦k⟧ = Total[Activities⟦k - 1, All, 5⟧]
]
For[j = 1, j ≤ T, j ++,
 XLevel⟦j⟧ = Total[Activities⟦j, All, 1⟧]
]
For[j = 1, j ≤ T, j ++,
 YLevel⟦j⟧ = Total[Activities⟦j, All, 2⟧]
]
For[j = 1, j ≤ T, j ++,
 ZLevel⟦j⟧ = Total[Activities⟦j, All, 3⟧]
]
For[j = 1, j ≤ T, j ++,
 Deltas⟦j⟧ = Total[Activities⟦j, All, 4⟧]
]

Fig1 = GraphicsGrid[{
   {
```

```
      ListLinePlot[{Callout[YLevel, "y_t", Scaled[0.2]], Callout[ZLevel, "z_t", Scaled[0.35], Scaled[0.25]]},
       PlotStyle → {{Blue}, {Red}}, PlotRange → {0, Max[YLevel] + 20}, AxesLabel → {"t", "y_t,z_t"},
       Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[YLevel] + 20}}]}, LabelStyle → s1, ImageSize → s2],
      ListLinePlot[Deltas, PlotStyle → {Blue}, PlotRange → {0, Max[Deltas] + 10}, AxesLabel → {"t", "δ_t"},
       Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[Deltas] + 10}}]}, LabelStyle → s1, ImageSize → s2],
      ListLinePlot[WealthPlot, PlotRange → {0, Max[WealthPlot] + 10}, AxesLabel → {"t", "W_{t-1}=p_{t-1}ω_{t-1}"}, PlotStyle →
        {Blue}, Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[Wealth] + 10}}]}, LabelStyle → s1, ImageSize → s2]
     },
     {
      ListLinePlot[
       Table[(Total[Activities[[t, All, 5]]] - If[t == 1, Total[AgentSet[[All, 1]]], Total[Activities[[t - 1, All, 5]]]]) /
         If[t == 1, Total[AgentSet[[All, 1]]], Total[Activities[[t - 1, All, 5]]]], {t, 1, T}],
       PlotStyle → {Blue}, PlotRange → All, AxesLabel → {"t", "g_t"}, AxesOrigin → {0, 0},
       Epilog → {Black, Dashed, Line[{{q, 0}, {q, 1}}]}, LabelStyle → s1, ImageSize → s2],
      ListLinePlot[{Wage, Table[b, {T}]}, PlotStyle → {{Blue}, {Dashed, Black}},
       PlotRange → {0, (1 - A (1 + 0))/L + 0.5}, AxesLabel → {"t", "w_t,b_t"}, AxesOrigin → {0, 0},
       Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[Wage] + 0.5}}]}, LabelStyle → s1, ImageSize → s2],
      ListLinePlot[Profit, PlotStyle → {Blue}, PlotRange → {0, Max[Profit] + 0.05}, AxesLabel → {"t", "r_t"}, AxesOrigin →
        {0, 0}, Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[Profit] + 0.05}}]}, LabelStyle → s1, ImageSize → s2],
     }
    }, Spacings → {-10, 20}, ImageSize → Full]
Fig1x = ListLinePlot[XLevel, PlotStyle → {Thick, Blue}, PlotRange → {0, Max[XLevel]},
   AxesLabel → {"t", "x_t"}, Epilog → {Black, Dashed, Line[{{q + 1, -1}, {q + 1, 1}}]}];
Fig1out = ListLinePlot[Output, PlotRange → {0, Max[Output] + 10}, AxesLabel → {"t", "(1-A_t)y_t"},
   PlotStyle → {Thick, Blue}, Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[Output] + 10}}]}];
Fig1outcap = ListLinePlot[Output/N, PlotRange → All, AxesLabel → {"t", "((1-A_t)y_t)/N_t"},
```

```
PlotStyle → {Thick, Blue}, Epilog → {Black, Dashed, Line[{{q + 1, 0}, {q + 1, Max[Output/N] + 0.5}}]};
```

Out[ ]=



In[ ]:=
```
Export["./Fig1.eps", Fig1, "EPS"]
Export["./Fig1x.eps", Fig1x, "EPS"]
Export["./Fig1out.eps", Fig1out, "EPS"]
Export["./Fig1outcap.eps", Fig1outcap, "EPS"]
```

Out[ ]=
```
./Fig1.eps
```

Out[ ]=
```
./Fig1x.eps
```

Out[ ]=
```
./Fig1out.eps
```

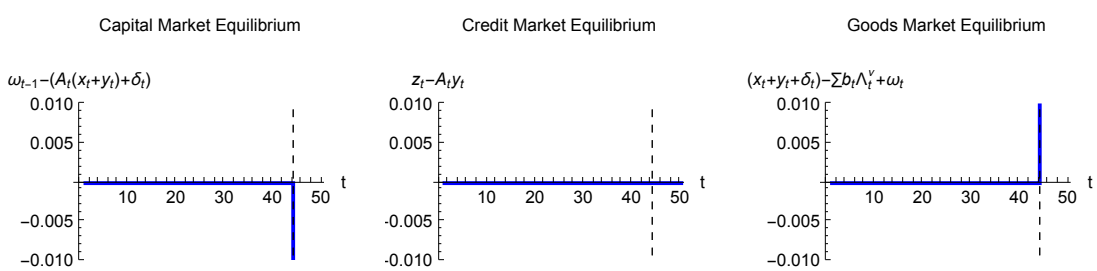
Out[ ]=
```
./Fig1outcap.eps
```

The set of figures below shows the status of the equilibrium over time in terms of the capital market, credit market, goods market, and tests that $A_t\, y_t \leq z_{t-1}$.

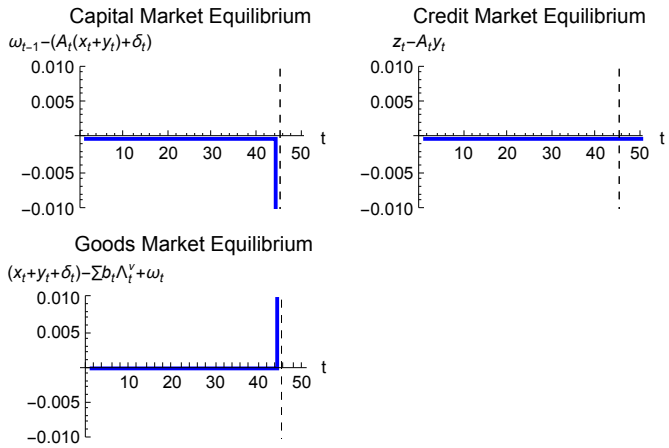
```
In[•]:= Clear[CapitalMarketEqb, GoodsMarketEqb, AyTest, j, k];
CapitalMarketEqb = Table[0.0, {T}];
GoodsMarketEqb = Table[0.0, {T}];
AyTest = Table[0.0, {T}];

CapitalMarketEqb⟦1⟧ = Total[AgentSet⟦All, 1⟧] - (A (XLevel⟦1⟧ + YLevel⟦1⟧) + Deltas⟦1⟧);
For[k = 2, k ≤ T, k++,
 CapitalMarketEqb⟦k⟧ = Wealth⟦k - 1⟧ - (A (XLevel⟦k⟧ + YLevel⟦k⟧) + Deltas⟦k⟧)
]

Clear[k];
For[k = 1, k ≤ T, k++,
 GoodsMarketEqb⟦k⟧ = ((XLevel⟦k⟧ + YLevel⟦k⟧) + Deltas⟦k⟧) - (Total[b Activities⟦k, All, 6⟧] + Wealth⟦k⟧)
]

Clear[k];
AyTest⟦1⟧ = A Total[Activities⟦1, All, 2⟧] - Total[Activities⟦1, All, 3⟧];
For[k = 2, k ≤ T, k++,
 AyTest⟦k⟧ = A Total[Activities⟦k, All, 2⟧] - Total[Activities⟦k, All, 3⟧]
]

Fig2 = GraphicsRow[
  {
   ListLinePlot[CapitalMarketEqb, PlotStyle → {Thick, Blue},
    PlotRange → {-0.01, 0.01}, AxesLabel → {"t", "ω_{t-1}-(A_t(x_t+y_t)+δ_t)"},
    PlotLabel → Style["Capital Market Equilibrium", 9], AxesOrigin → {0, 0}, LabelStyle → 9,
    Epilog → {Black, Dashed, Line[{{q, Max[{Max[CapitalMarketEqb], 1}]}, {q, Min[CapitalMarketEqb] - 10}}]}],
   ListLinePlot[ZLevel - (A YLevel), PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
    AxesLabel → {"t", "z_t-A_t y_t"}, PlotLabel → Style["Credit Market Equilibrium", 9],
    AxesOrigin → {0, 0}, LabelStyle → 9, Epilog → {Black, Dashed, Line[{{q, -0.1}, {q, 0.1}}]}],
   ListLinePlot[GoodsMarketEqb, PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
```

```
    AxesLabel → {"t", "(xt+yt+δt)-∑btΛtγ+ωt"}, PlotLabel → Style["Goods Market Equilibrium", 9],
    AxesOrigin → {0, 0}, LabelStyle → 9, Epilog → {Black, Dashed, Line[{{q, -10}, {q, Max[GoodsMarketEqb] + 10}}]}]
  }, Spacings → {0, 15}
 ]
GraphicsGrid[{
  {
   ListLinePlot[CapitalMarketEqb, PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
    AxesLabel → {"t", "ωt-1-(At(xt+yt)+δt)"}, PlotLabel → "Capital Market Equilibrium", AxesOrigin → {0, 0},
    Epilog → {Black, Dashed, Line[{{q + 1, Max[{Max[CapitalMarketEqb], 1}]}, {q + 1, Min[CapitalMarketEqb] - 10}}]}],
   ListLinePlot[ZLevel - (A YLevel), PlotStyle → {Thick, Blue},
    PlotRange → {-0.01, 0.01}, AxesLabel → {"t", "zt-Atyt"}, PlotLabel → "Credit Market Equilibrium",
    AxesOrigin → {0, 0}, Epilog → {Black, Dashed, Line[{{q + 1, -0.1}, {q + 1, 0.1}}]}]
  },
  {
   ListLinePlot[GoodsMarketEqb, PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
    AxesLabel → {"t", "(xt+yt+δt)-∑btΛtγ+ωt"}, PlotLabel → "Goods Market Equilibrium",
    AxesOrigin → {0, 0}, Epilog → {Black, Dashed, Line[{{q + 1, -10}, {q + 1, Max[GoodsMarketEqb] + 10}}]}]
   (* ListLinePlot[AyTest,PlotStyle→{Thick,Blue},PlotRange→All,AxesLabel→{"t","Atyt-zt-1"},
    PlotLabel→"Testing Atyt≤zt-1",AxesOrigin→{0,0},Epilog→{Black,Dashed,Line[{{q+1,1},{q+1,-100}}]}] *)
  }
 }, Spacings → {0, 15}]
```

Out[ ]=



Out[ ]=



In[ ]:=
```
Export["./Fig2.eps", Fig2, "EPS"]
```

Out[ ]=
```
./Fig2.eps
```

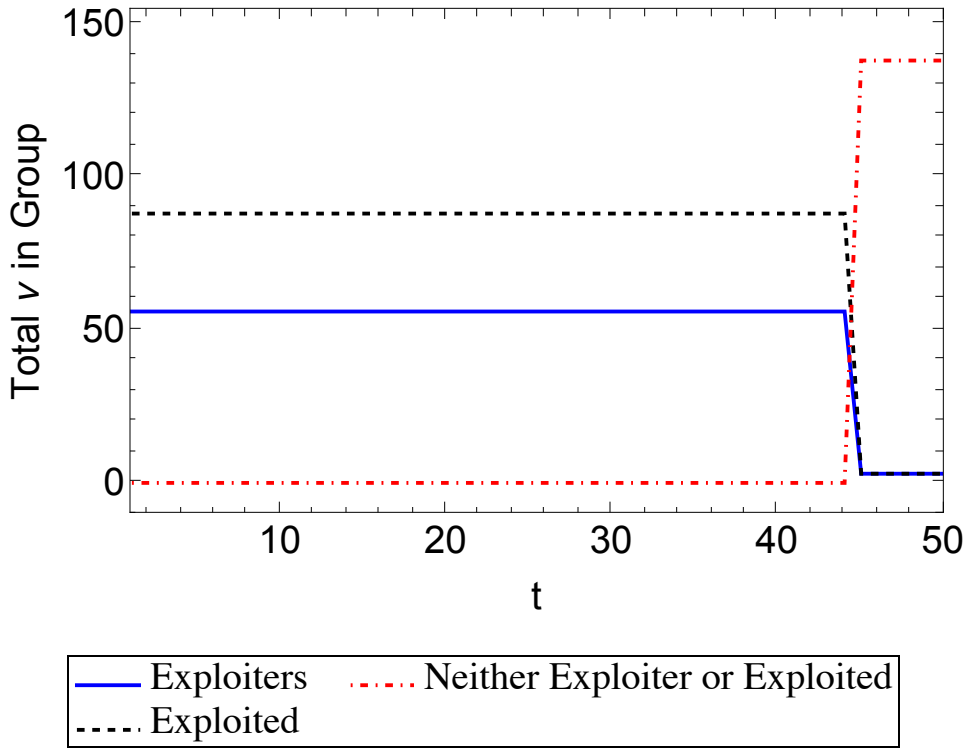
Exploitation and Class Over Time

The chart below captures the number of agents who are exploiters, exploited, or neither over the course of the simulation according to Definition 2, which defines exploitation in relation to Λ_t^v and $v_t\, c_t^v$, where v_t is just the embodied labour value and $c_t^v = \pi_t\, W_{t-1}^v + (w_t - b_t)\; l^v + b_t\, \Lambda_t^v$. An agent v is exploited during t if and only if $\Lambda_t^v > v_t\, c_t^v$, an agent is an exploiter if and only if $\Lambda_t^v < v_t\, c_t^v$, and an agent is neither exploited nor an exploiter if and only if $\Lambda_t^v = v_t\, c_t^v$.

In[ ]:=
```
Clear[IndivExploitation, ConsumptionBundle];
ConsumptionBundle = Table[0.0, {i, T}, {j, N}];
IndivExploitation = Table[0.0, {i, T}, {j, N}];

For[i = 1, i ≤ N, i++,
 ConsumptionBundle⟦1, i⟧ = EmbodiedValues⟦1, 2⟧ (Profit⟦1⟧ × AgentSet⟦i, 1⟧ + (Wage⟦1⟧ - b) l⟦i⟧ + b Activities⟦1, i, 6⟧);
 IndivExploitation⟦1, i⟧ = Activities⟦1, i, 6⟧ - ConsumptionBundle⟦1, i⟧;
```

```
]

For[t = 2, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  ConsumptionBundle⟦t, i⟧ =
   EmbodiedValues⟦1, 2⟧ (Profit⟦t⟧ × Activities⟦t - 1, i, 5⟧ + (Wage⟦t⟧ - b) l⟦i⟧ + b Activities⟦t, i, 6⟧);
  IndivExploitation⟦t, i⟧ = Activities⟦t, i, 6⟧ - ConsumptionBundle⟦t, i⟧;
 ]
]

Clear[Exploiters, Exploited, NonExploit];
Exploiters = Table[0.0, {T}];
Exploited = Table[0.0, {T}];
NonExploit = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  If[IndivExploitation⟦t, i⟧ < 0, Exploiters⟦t⟧ ++, If[IndivExploitation⟦t, i⟧ == 0, NonExploit⟦t⟧ ++, Exploited⟦t⟧ ++]];
 ]
]

ExploitationLegend =
  Grid[{{Row[{Graphics[{Thick, Blue, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["Exploiters", 20], FontFamily → "Times"]}], Spacer[10],
      Row[{Graphics[{Thick, Red, DotDashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["Neither Exploiter or Exploited", 20], FontFamily → "Times"]}]},
     {Row[{Graphics[{Thick, Black, Dashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["Exploited", 20], FontFamily → "Times"]}], Spacer[10],
     }}, Frame → True, Alignment → Left];
ExploitationPlot = Labeled[ListLinePlot[{Exploiters, Exploited, NonExploit},
   PlotStyle → {{Thick, Blue}, {Thick, Dashed, Black}, {Thick, DotDashed, Red}},
   Frame → True, FrameLabel → {"t", "Total ν in Group"}, LabelStyle → 20,
   PlotRange → {{1, T}, {-10, N + 10}}, ImageSize → {500, 350}], ExploitationLegend]
```

Out[ ]=



In[ ]:=
```
Export["./ExploitationPlot.eps", ExploitationPlot, "EPS"]
```

Out[ ]=

```
./ExploitationPlot.eps
```

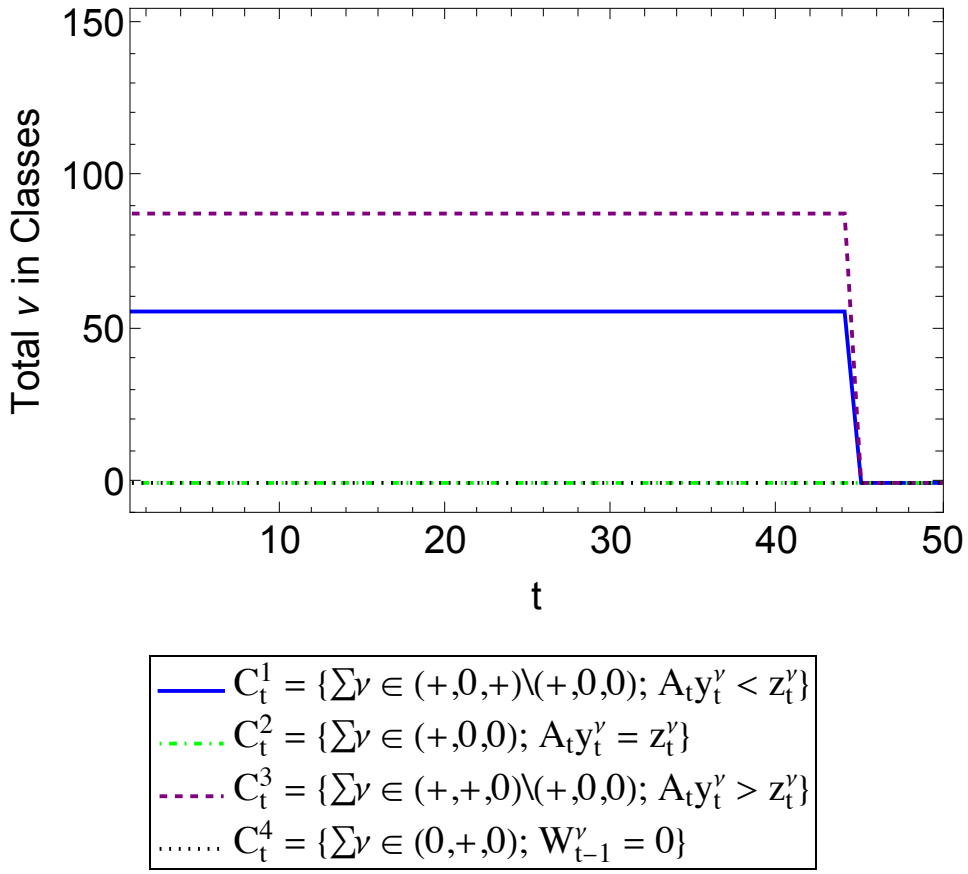
The chart below captures the class composition of the simulation over time according to Corollary 1 of Theorem 3 (class is defined using labour endowments l^v in relation to means of production).

In[ ]:=
```
Clear[Class1, Class2, Class3, Class4, j, k];
Class1 = Table[0.0, {T}];
Class2 = Table[0.0, {T}];
Class3 = Table[0.0, {T}];
```

```
Class4 = Table[0.0, {T}];

For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[A Activities⟦j, k, 2⟧ < Activities⟦j, k, 3⟧ && Profit⟦j⟧ > 0, Class1⟦j⟧++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[A Activities⟦j, k, 2⟧ == Activities⟦j, k, 3⟧ && Profit⟦j⟧ > 0, Class2⟦j⟧++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[A Activities⟦j, k, 2⟧ > Activities⟦j, k, 3⟧ && Profit⟦j⟧ > 0, Class3⟦j⟧++, 0];
 ]
]

Clear[k];
For[k = 1, k ≤ N, k++,
 If[AgentSet⟦k, 1⟧ == 0 && Profit⟦1⟧ > 0, Class4⟦1⟧++, 0];
]

Clear[j, k];
For[j = 2, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[Activities⟦j - 1, k, 5⟧ == 0 && Profit⟦j⟧ > 0, Class4⟦j⟧++, 0];
```

```
 ]
]


ClassLegend =
  Column[{
    Row[{Graphics[{Thick, Blue, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^1 = {∑ν ∈ (+,0,+)\(+,0,0); A_t y_t^ν < z_t^ν}", 20], FontFamily → "Times"]}],
    Row[{Graphics[{Thick, DotDashed, Green, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^2 = {∑ν ∈ (+,0,0); A_t y_t^ν = z_t^ν}", 20], FontFamily → "Times"]}],
    Row[{Graphics[{Thick, Purple, Dashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^3 = {∑ν ∈ (+,+,0)\(+,0,0); A_t y_t^ν > z_t^ν}", 20], FontFamily → "Times"]}],
    Row[{Graphics[{Thick, Black, Dotted, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^4 = {∑ν ∈ (0,+,0); W_{t-1}^ν = 0}", 20], FontFamily → "Times"]}]
   }, Frame → True, Alignment → Left];
ClassPlotCorollary1 = Labeled[
  ListLinePlot[{Class1, Class2, Class3, Class4}, PlotStyle → {{Thick, Blue}, {Thick, DotDashed, Green},
     {Thick, Dashed, Purple}, {Thick, Dotted, Black}, {Thick, DotDashed, Orange}, {Thick, Dashed, Green}},
   PlotRange → {{1, T}, {-10, N + 10}}, Frame → True, FrameLabel → {"t", "Total ν in Classes"},
   LabelStyle → 20, ImageSize → {500, 350}, AxesOrigin → {1, -10}], ClassLegend]
```

Out[ ]=



In[ ]:=
```
Export["./ClassPlotCorollary1.eps", ClassPlotCorollary1, "EPS"]
```

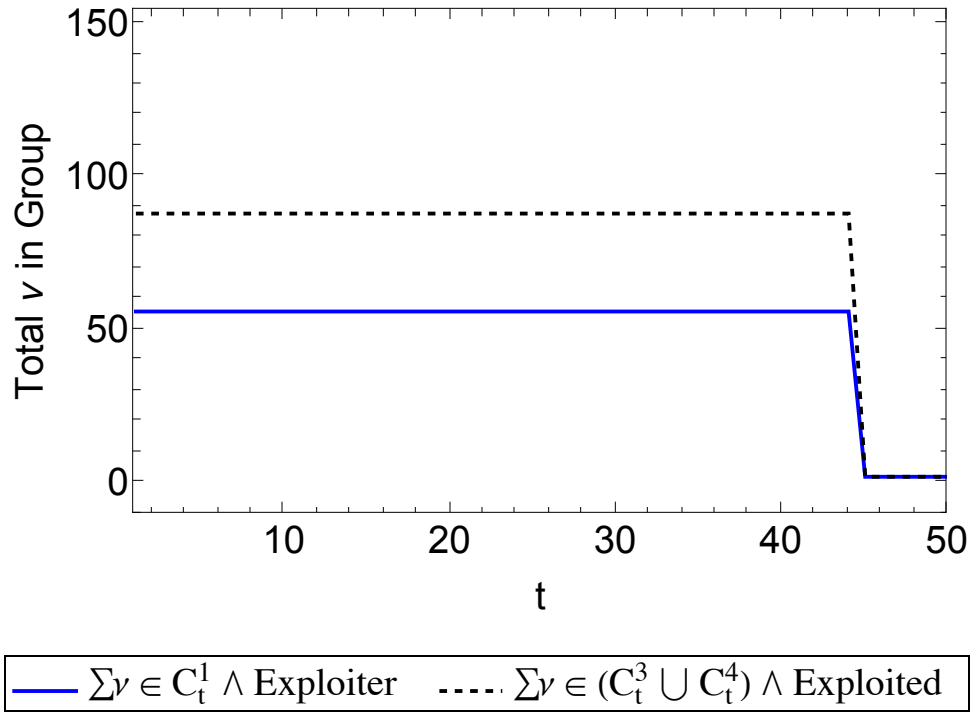
Out[ ]=
```
./ClassPlotCorollary1.eps
```

The chart below captures the intersection of class and exploitation over the simulation. If an agent is in C^1 according to Corollary 1 of Theorem 3 and is also an exploiter according to Definition 2, then the agent is in the group "$C^1 \wedge$ Exploiter". If an agent is in $C^3 \cup C^4$ and is exploited they are in group "$(C^3 \cup C^4) \wedge$ Exploited".

In[ ]:=
```
Clear[CECP1, CECP2, CECP3, j, k];
CECP1 = Table[0.0, {T}];
```

```
CECP2 = Table[0.0, {T}];
CECP3a = Table[0.0, {T}];
CECP3b = Table[0.0, {T}];
CECP3 = Table[0.0, {T}];

For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[A Activities⟦j, k, 2⟧ < Activities⟦j, k, 3⟧ && IndivExploitation⟦j, k⟧ < 0, CECP1⟦j⟧++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[A Activities⟦j, k, 2⟧ > Activities⟦j, k, 3⟧ && IndivExploitation⟦j, k⟧ > 0, CECP3⟦j⟧++, 0];
  If[If[j == 1, AgentSet⟦k, 1⟧, Activities⟦j - 1, k, 5⟧] == 0 && IndivExploitation⟦j, k⟧ > 0, CECP3⟦j⟧++, 0];
 ]
]


CECPLegend =
  Grid[{{Row[{Graphics[{Thick, Blue, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]], Spacer[5],
        Style[Style["∑v ∈ C_t^1 ∧ Exploiter", 20], FontFamily → "Times"]}], Spacer[10],
      Row[{Graphics[{Thick, Black, Dashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["∑v ∈ (C_t^3 ⋃ C_t^4) ∧ Exploited", 20], FontFamily → "Times"]}],
    }}, Frame → True, Alignment → Left];
CECPPlotCorollary1 = Labeled[ListLinePlot[{CECP1, CECP3},
   PlotStyle → {{Thick, Blue}, {Thick, Dashed, Black}}, Frame → True, FrameLabel → {"t", "Total v in Group"},
   LabelStyle → 20, PlotRange → {{1, T}, {-10, N + 10}}, ImageSize → {500, 350}], CECPLegend]
```

Out[ ]=



In[ ]:=
```
Export["./CECPPlotCorollary1.eps", CECPPlotCorollary1, "EPS"]
```

Out[ ]=
```
./CECPPlotCorollary1.eps
```

The charts below display the distribution of an index of the intensity of exploitation across the agents over the simulation. The index itself is calculated as $e_t^v = \Lambda_t^v / v_t\, c_t^v$ and the charts that follow explore some possibilities for visualizing the intensity of exploitation over time.

In[ ]:=
```
Clear[ExploitationIndex];
ExploitationIndex = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  ExploitationIndex[[t, i]] = Activities[[t, i, 6]] / (EmbodiedValues[[1, 2]] (ConsumptionBundle[[t, i]] / EmbodiedValues[[1, 2]])) /. p → 1
 ]
]
```

The chart below uses *Mathematica's* `ArrayPlot` function to display the exploitation index of the N agents over *T*. There is a fixed distribution of uneven exploitation intensity while the simulation is capital constrained, however, this pattern disappears once the simulation is labour constrained, at which point exploitation intensity is the same for all $v \in \mathcal{N}$.

In[ ]:=
```
ExploitationIndex[[1, 1]]
```

Out[ ]=
```
1.13049
```

In[ ]:=
```
ExploitationIndex[[1]]
```

Out[ ]=
```
{1.13049, 1.12916, 1.12969, 1.12412, 1.12487, 1.12569, 1.11973, 1.12177, 1.12244, 1.11604, 1.12275,
 1.11573, 1.11117, 1.12211, 1.1144, 1.11463, 1.10999, 1.11746, 1.11105, 1.11665, 1.11001, 1.11154, 1.1063,
 1.10914, 1.10611, 1.09697, 1.10049, 1.10013, 1.09655, 1.10115, 1.11503, 1.0923, 1.09105, 1.09163, 1.10853,
 1.09926, 1.10446, 1.10261, 1.10945, 1.08961, 1.08777, 1.07896, 1.06752, 1.0927, 1.07552, 1.09076,
 1.10019, 1.08758, 1.07439, 1.07267, 1.08208, 1.07995, 1.0845, 1.06847, 1.07519, 1.04751, 1.01327, 1.0426,
 1.06651, 1.04565, 1.05156, 1.01078, 1.02528, 1.05809, 1.03728, 1.05163, 1.04413, 1.04347, 1.04847,
 1.02577, 1.04592, 1.0187, 1.03374, 1.027, 1.03026, 1.04048, 1.01772, 1.03462, 1.03767, 1.02345, 1.00021,
 0.992845, 1.03327, 1.0066, 1.00347, 0.993497, 1.00524, 1.00941, 1.00248, 0.991169, 1.00176, 0.978759,
 0.982048, 0.993514, 0.981639, 0.993798, 0.940936, 0.971079, 0.97497, 0.94927, 0.976368, 0.95351, 0.958682,
 0.961767, 0.952915, 0.937314, 0.826099, 0.922282, 0.898542, 0.911188, 0.911411, 0.867535, 0.885655,
 0.900965, 0.84347, 0.860474, 0.825178, 0.772503, 0.882461, 0.871979, 0.843054, 0.879379, 0.867892,
 0.844821, 0.825634, 0.843872, 0.867227, 0.778101, 0.851387, 0.837697, 0.835183, 0.802683, 0.808094,
 0.79722, 0.790084, 0.799258, 0.815888, 0.811557, 0.726798, 0.73243, 0.70479, 0.781131, 0.744122, 0.627681}
```

In[ ]:=
```
CountryList
```

Out[ ]=
```
{Burundi, Congo - Kinshasa, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic, Madagascar,
 Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia,
 Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania, Haiti,
 Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria, Laos,
 Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, Congo - Brazzaville, Angola,
 Eswatini, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa,
 Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, Trinidad and Tobago,
 Finland, United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada,
 Australia, France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland,
 Italy, Austria, Switzerland, Norway, United Arab Emirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

In[ ]:=
```
(ExploitTable = Table[{CountryList⟦i⟧, ExploitationIndex⟦1, i⟧, DataNoHeader⟦i, 5⟧}, {i, 1, N}]) // TableForm
Export["./ExploitTable.csv", ExploitTable, "CSV"]
```

Out[ ]//TableForm=

Burundi	1.13049	1469.37
Congo – Kinshasa	1.12916	2162.63
Malawi	1.12969	2352.02
Mali	1.12412	2977.05
Sierra Leone	1.12487	3355.15
Liberia	1.12569	3492.56
Mozambique	1.11973	3680.29
Central African Republic	1.12177	4078.55
Madagascar	1.12244	4303.22
Niger	1.11604	4491.84
Rwanda	1.12275	4533.57
Burkina Faso	1.11573	4736.35
Ethiopia	1.11117	6525.92
Zimbabwe	1.12211	6844.83
Togo	1.1144	7181.24
Benin	1.11463	7303.34
Gambia	1.10999	7828.99
Kenya	1.11746	7934.22
Yemen	1.11105	8046.03
Uganda	1.11665	8351.96
Nepal	1.11001	8551.18
Cambodia	1.11154	8645.97
Ivory Coast	1.1063	9164.58
Cameroon	1.10914	9440.94
Pakistan	1.10611	9876.66
Senegal	1.09697	11 541.5
Myanmar	1.10049	12 050.7
Nigeria	1.10013	12 896.
Mauritania	1.09655	13 200.5
Bangladesh	1.10115	13 331.1
Kyrgyzstan	1.11503	13 446.6
Tanzania	1.0923	13 870.6
Haiti	1.09105	14 398.6

Lesotho	1.09163	14 660.1
Bolivia	1.10853	14 866.1
Honduras	1.09926	16 134.7
Vietnam	1.10446	16 253.2
Egypt	1.10261	16 305.2
Belize	1.10945	17 417.7
Nicaragua	1.08961	19 570.
El Salvador	1.08777	20 356.8
Guatemala	1.07896	20 676.2
Sudan	1.06752	20 879.3
Syria	1.0927	20 923.7
Laos	1.07552	22 000.5
Zambia	1.09076	22 138.9
Moldova	1.10019	23 085.9
Fiji	1.08758	24 469.
India	1.07439	24 929.7
Iraq	1.07267	27 342.
Philippines	1.08208	27 429.3
Paraguay	1.07995	27 688.5
Armenia	1.0845	30 463.9
Ghana	1.06847	31 868.4
Jordan	1.07519	33 279.2
Congo – Brazzaville	1.04751	35 216.
Angola	1.01327	36 269.3
Eswatini	1.0426	37 150.8
Peru	1.06651	37 355.5
Costa Rica	1.04565	46 921.5
Sri Lanka	1.05156	46 996.
Morocco	1.01078	47 853.5
Namibia	1.02528	49 054.5
Ukraine	1.05809	49 261.9
Colombia	1.03728	49 818.9
Tajikistan	1.05163	50 019.4
Gabon	1.04413	50 047.9
South Africa	1.04347	50 889.5
Mongolia	1.04847	51 038.5

Maldives	1.02577	51 775.2
Argentina	1.04592	53 411.4
Algeria	1.0187	54 124.
Dominican Republic	1.03374	54 940.3
Jamaica	1.027	56 100.4
Ecuador	1.03026	57 619.2
Bulgaria	1.04048	59 310.3
Tunisia	1.01772	61 936.4
Kazakhstan	1.03462	62 751.4
Serbia	1.03767	65 626.9
Albania	1.02345	66 256.5
Iran	1.00021	67 140.3
Indonesia	0.992845	68 126.
Poland	1.03327	69 111.8
Mexico	1.0066	71 785.6
Thailand	1.00347	73 930.4
China	0.993497	75 096.1
Barbados	1.00524	75 265.1
Brazil	1.00941	75 478.8
Panama	1.00248	77 646.1
Venezuela	0.991169	84 138.6
Chile	1.00176	84 975.4
Mauritius	0.978759	85 292.7
Uruguay	0.982048	87 332.4
Malaysia	0.993514	88 769.3
Botswana	0.981639	92 539.9
Romania	0.993798	94 394.
Turkey	0.940936	103 302.
Lithuania	0.971079	113 001.
Russia	0.97497	114 627.
Malta	0.94927	125 336.
Slovakia	0.976368	126 502.
New Zealand	0.95351	131 140.
Croatia	0.958682	132 891.
Israel	0.961767	140 634.
Estonia	0.952915	141 655.

Hungary	0.937314	146 273.
Kuwait	0.826099	171 434.
South Korea	0.922282	174 502.
Taiwan	0.898542	177 227.
Japan	0.911188	179 635.
United States	0.911411	187 771.
Trinidad and Tobago	0.867535	193 445.
Finland	0.885655	199 659.
United Kingdom	0.900965	199 783.
Cyprus	0.84347	202 290.
Latvia	0.860474	204 018.
Saudi Arabia	0.825178	204 732.
Bahrain	0.772503	214 422.
Czech Republic	0.882461	214 620.
Slovenia	0.871979	217 923.
Greece	0.843054	218 848.
Canada	0.879379	220 394.
Australia	0.867892	221 724.
France	0.844821	224 054.
Spain	0.825634	225 449.
Iceland	0.843872	227 653.
Germany	0.867227	231 789.
Portugal	0.778101	232 192.
Sweden	0.851387	232 657.
Netherlands	0.837697	244 380.
Denmark	0.835183	261 451.
Belgium	0.802683	265 306.
Hong Kong	0.808094	267 737.
Ireland	0.79722	272 735.
Italy	0.790084	278 403.
Austria	0.799258	288 282.
Switzerland	0.815888	294 731.
Norway	0.811557	296 692.
United Arab Emirates	0.726798	314 199.
Macao	0.73243	321 344.
Brunei	0.70479	345 758.

```
Singapore                    0.781131    367 762.
Luxembourg                   0.744122    376 751.
Qatar                        0.627681    509 907.
```

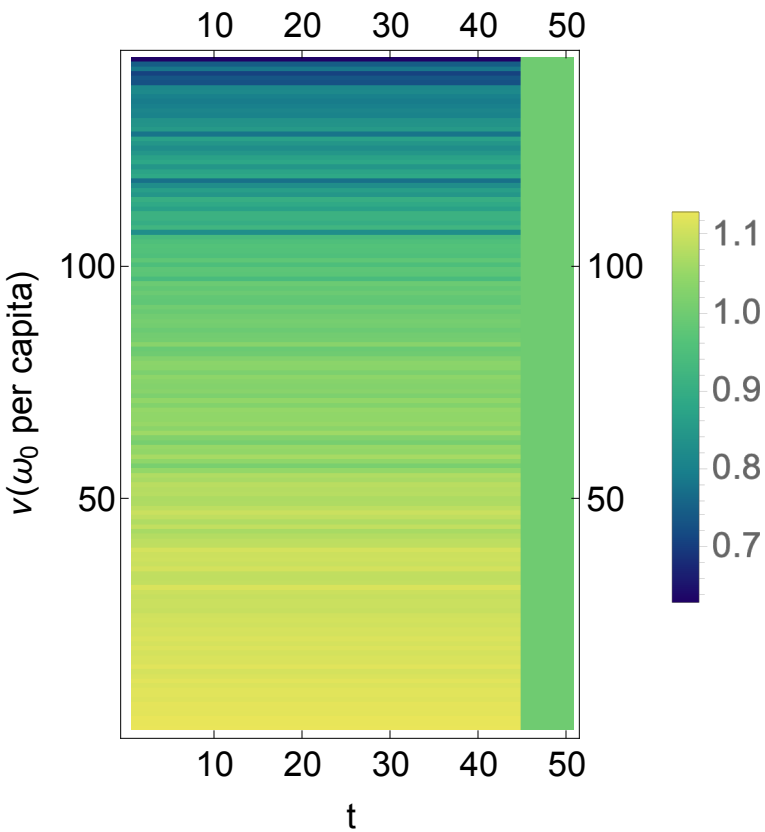
Out[ ]=

```
./ExploitTable.csv
```

In[ ]:=

```
ExploitationArray = ArrayPlot[Transpose[ExploitationIndex], FrameLabel → {"ν(ω0 per capita)", "t"},
  PlotLegends → Automatic, ColorFunction → "BlueGreenYellow", ColorFunctionScaling → True,
  DataReversed → True, FrameTicks → Automatic, AspectRatio → 1.5, LabelStyle → 18]
```

Out[ ]=



In[ ]:=
```
Export["./ExploitationArray.eps", ExploitationArray, "EPS"]
```
Out[ ]=
```
./ExploitationArray.eps
```

In[ ]:=
```
ExploitationIndexDistrib = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  ExploitationIndexDistrib[[t, i]] = {t, ExploitationIndex[[t, i]]}
 ]
]
```

Below the Gini coefficient for the exploitation intensity index is plotted over *T*. The pattern of the Gini coefficient is consistent with the previous charts depicting the exploitation intensity index.

In[ ]:=
```
(* Gini=Table[0.0,{T}];
For[t=1,t≤T,t++,
 Gini[[t]]=N/(N-1) ((Sum[Sum[Abs[ExploitationIndex[[t,i]]-ExploitationIndex[[t,j]]], {j,1,N}], {i,1,N}])/(2 N^2 Mean[ExploitationIndex[[t]]]))
] *)
```

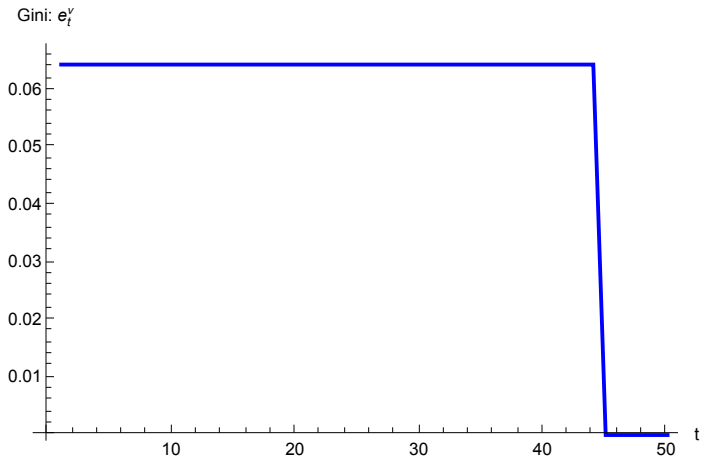
In[ ]:=
```
Clear[SortExploitationIndex];
SortExploitationIndex = Table[Table[0.0, {N}], {t, 1, T}];
GiniTest = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 SortExploitationIndex[[t]] = Sort[ExploitationIndex[[t]]];
 GiniTest[[t]] = N/(N - 1) Sum[((2 i - N - 1) SortExploitationIndex[[t, i]]) / (N^2 Mean[SortExploitationIndex[[t]]]), {i, 1, N}]
]
```

In[ ]:=
```
Chop[GiniTest]
```

Out[ ]=
```
{0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787,
 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787,
 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787,
 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787,
 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0.0644787, 0, 0, 0, 0, 0, 0}
```

In[ ]:=
```
ExploitationGini = ListLinePlot[GiniTest, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "Gini: e_t^v"}]
```

Out[ ]=



In[ ]:=
```
Export["./ExploitationGini.eps", ExploitationGini, "EPS"]
```

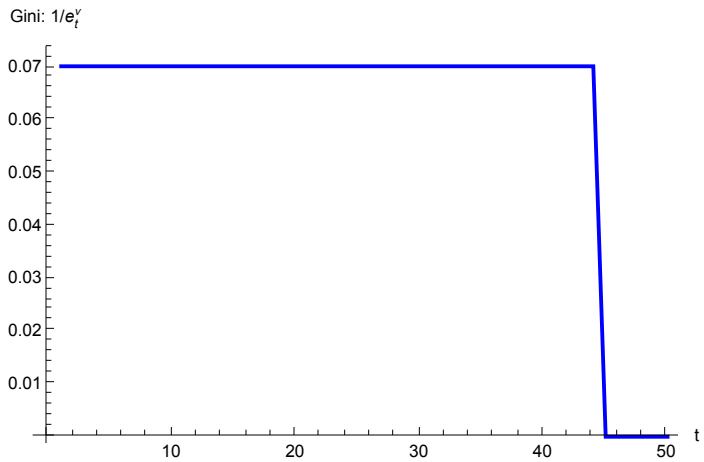
Out[ ]=
```
./ExploitationGini.eps
```

Inverse of exploitation intensity.

In[ ]:=
```
Clear[SortExploitationIndex2];
SortExploitationIndex2 = Table[Table[0.0, {N}], {t, 1, T}];
GiniTest2 = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 SortExploitationIndex2[[t]] = Sort[1 / ExploitationIndex[[t]]];
 GiniTest2[[t]] = N/(N - 1) Sum[((2 i - N - 1) SortExploitationIndex2[[t, i]]) / (N^2 Mean[SortExploitationIndex2[[t]]]), {i, 1, N}]
]
ExploitationGini2 = ListLinePlot[GiniTest2, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "Gini: 1/e_t^γ"}]
```

$$\text{GiniTest2}[[t]] = \frac{N}{N-1}\sum_{i=1}^{N}\left((2i-N-1)\,\text{SortExploitationIndex2}[[t,i]]\right) \Big/ \left(N^2\,\text{Mean}[\text{SortExploitationIndex2}[[t]]]\right)$$

Out[ ]=



In[ ]:=
```
Export["./InverseExploitationGini.eps", ExploitationGini2, "EPS"]
```

Out[ ]=

./InverseExploitationGini.eps

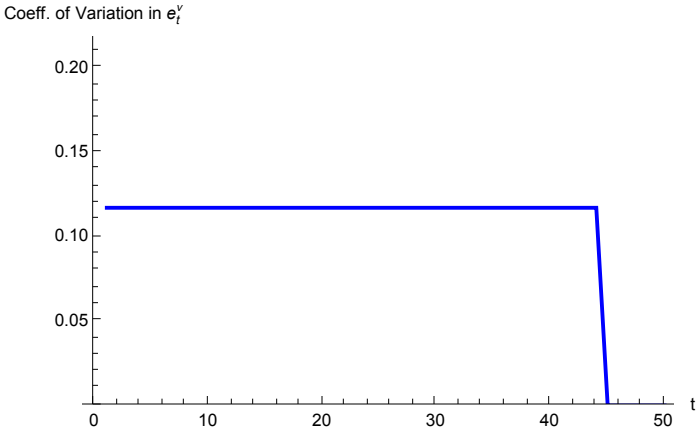
In[ ]:=
```
1 / ExploitationIndex[[1]]
```
Out[ ]=
```
{0.884574, 0.885614, 0.885197, 0.889582, 0.888994, 0.88834, 0.893071, 0.891451, 0.890914, 0.896029, 0.890667,
 0.896276, 0.899949, 0.891178, 0.897341, 0.897158, 0.90091, 0.894884, 0.90005, 0.895535, 0.900892, 0.899657,
 0.903915, 0.901603, 0.904067, 0.911599, 0.908685, 0.908984, 0.911948, 0.90814, 0.89684, 0.915497, 0.916549,
 0.91606, 0.902099, 0.909699, 0.905422, 0.906938, 0.901344, 0.917759, 0.919314, 0.926818, 0.936751, 0.915163,
 0.929783, 0.916791, 0.908935, 0.919469, 0.930764, 0.932256, 0.924149, 0.925971, 0.922085, 0.935915,
 0.930067, 0.954648, 0.986899, 0.959142, 0.937637, 0.956343, 0.950967, 0.98933, 0.975342, 0.945103, 0.964062,
 0.950903, 0.957731, 0.958343, 0.953769, 0.974882, 0.956098, 0.981645, 0.967365, 0.973707, 0.970626,
 0.961098, 0.982587, 0.966542, 0.963697, 0.977092, 0.999791, 1.00721, 0.967799, 0.993448, 0.996539,
 1.00655, 0.994785, 0.990681, 0.997526, 1.00891, 0.998244, 1.0217, 1.01828, 1.00653, 1.0187, 1.00624,
 1.06277, 1.02978, 1.02567, 1.05344, 1.0242, 1.04876, 1.0431, 1.03975, 1.04941, 1.06688, 1.21051, 1.08427,
 1.11291, 1.09747, 1.0972, 1.15269, 1.12911, 1.10992, 1.18558, 1.16215, 1.21186, 1.29449, 1.13319, 1.14682,
 1.18616, 1.13717, 1.15222, 1.18368, 1.21119, 1.18501, 1.1531, 1.28518, 1.17455, 1.19375, 1.19734, 1.24582,
 1.23748, 1.25436, 1.26569, 1.25116, 1.22566, 1.2322, 1.3759, 1.36532, 1.41886, 1.28019, 1.34387, 1.59317}
```

In[ ]:=
```
GiniTest2
```
Out[ ]=
```
{0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773,
 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773,
 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773,
 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773,
 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773, 0.0702773,
 4.89119 × 10^-17, 4.89119 × 10^-17, 4.89119 × 10^-17, 4.89119 × 10^-17, 4.89119 × 10^-17, 4.89119 × 10^-17}
```

Plotting the coefficient of variation for e_t^v.

In[∘]:=
```
Clear[ExpCoeffVar];
ExpCoeffVar = Table[0.0, {T}];
For[i = 1, i ≤ T, i++,
 ExpCoeffVar⟦i⟧ = StandardDeviation[ExploitationIndex⟦i⟧] / Mean[ExploitationIndex⟦i⟧]
]
ListLinePlot[ExpCoeffVar, PlotStyle → {Thick, Blue},
 AxesLabel → {"t", "Coeff. of Variation in e^v_t"}, PlotRange → {0, Max[ExpCoeffVar] + 0.1}]
```

Out[∘]=



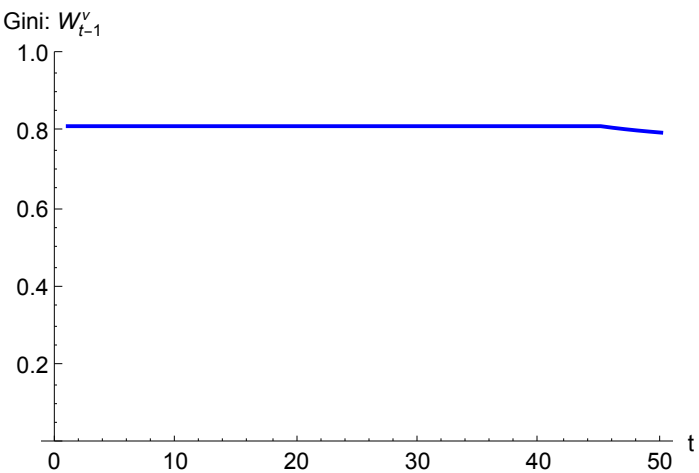
In[∘]:=
```
ExpCoeffVar
```

Out[∘]=
```
{0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198,
 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198,
 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198,
 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198, 0.117198,
 0.117198, 6.08803 × 10^-17, 6.08803 × 10^-17, 6.08803 × 10^-17, 6.08803 × 10^-17, 6.08803 × 10^-17, 6.08803 × 10^-17}
```

Plotting Gini coefficient for wealth $W_{t-1} = p_{t-1}\,\omega_{t-1}$.

In[ ]:=
```
Clear[SortWealth, WealthGini];
SortWealth = Table[Table[0.0, {N}], {t, 1, T}];
WealthGini = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 SortWealth[[t]] = If[t == 1, Sort[AgentSet[[All, 1]]], Sort[Activities[[t - 1, All, 5]]]];
 WealthGini[[t]] = N/(N - 1) Sum[(2 i - N - 1) SortWealth[[t, i]]/(N^2 Mean[SortWealth[[t]]]), {i, 1, N}]
]
WealthGiniPlot = ListLinePlot[WealthGini, PlotStyle → {Thick, Blue},
  AxesLabel → {"t", "Gini: W^ν_{t-1}"}, LabelStyle → 12, PlotRange → {0, 1}]
```

Out[ ]=



In[ ]:=
```
Export["./WealthGini.eps", WealthGiniPlot, "EPS"]
```

Out[ ]=

./WealthGini.eps

In[ ]:=
```
WealthGini
```

Out[ ]=
```
{0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.811503, 0.807881, 0.804632, 0.801721, 0.799042}
```

The figure below plots the distribution of wealth for select *t*.

In[ ]:=
```
(* WealthDistribRow=GraphicsRow[{
    Histogram[AgentSet⟦All,1⟧,10,"Probability",AxesOrigin→{0,0},PlotRange→{0,1},
     ImageSize→{250,300},LabelStyle→12,PlotLabel→Style["t = 1",18],AxesLabel→{"ω^v_{t-1}"}],
    Histogram[Activities⟦24,All,5⟧,20,"Probability",AxesOrigin→{0,0},PlotRange→{0,1},
     ImageSize→{250,300},LabelStyle→12,PlotLabel→Style["t = 25",18],AxesLabel→{"ω^v_{t-1}"}],
    Histogram[Activities⟦49,All,5⟧,20,"Probability",AxesOrigin→{0,0},PlotRange→{0,1},
     ImageSize→{250,300},LabelStyle→12,PlotLabel→Style["t = 50",18],AxesLabel→{"ω^v_{t-1}"}]
   },Spacings→{15,-20}
  ]
  Export["./WealthDistribRow.eps",WealthDistribRow,"EPS"]
*)
```

Levels of wealth for all *v* at *t* = 50.

In[ ]:=
```
Activities[[50, All, 5]]
```

Out[ ]=

{564 656., 5.21434×10^{6}, 1.38695×10^{6}, 1.06026×10^{6}, 511 473., 353 746., 1.66692×10^{6}, 319 194., 1.91523×10^{6}, 1.28006×10^{6}, 979 923., 1.19405×10^{6}, 7.90038×10^{6}, 1.93463×10^{6}, 705 612., 1.03559×10^{6}, 184 331., 5.49989×10^{6}, 2.61373×10^{6}, 4.85048×10^{6}, 2.81024×10^{6}, 1.61076×10^{6}, 2.30635×10^{6}, 2.49609×10^{6}, 2.00837×10^{7}, 1.63184×10^{6}, 6.02945×10^{6}, 2.29525×10^{7}, 517 622., 2.07897×10^{7}, 1.04403×10^{6}, 6.52321×10^{6}, 1.31451×10^{6}, 273 915., 1.78169×10^{6}, 1.37628×10^{6}, 1.55406×10^{7}, 1.54207×10^{7}, 72 253.6, 996 506., 1.04406×10^{6}, 2.61049×10^{6}, 5.86994×10^{6}, 3.23645×10^{6}, 1.09916×10^{6}, 3.13668×10^{6}, 872 795., 177 860., 2.41373×10^{8}, 7.48286×10^{6}, 2.20082×10^{7}, 1.41833×10^{6}, 696 730., 6.40751×10^{6}, 2.34698×10^{6}, 1.17424×10^{6}, 6.22933×10^{6}, 316 496., 8.28789×10^{6}, 1.44931×10^{6}, 6.31514×10^{6}, 9.81201×10^{6}, 736 689., 1.44133×10^{7}, 1.49745×10^{7}, 2.87306×10^{6}, 634 781., 1.80363×10^{7}, 998 491., 134 050., 1.4902×10^{7}, 1.30487×10^{7}, 3.58619×10^{6}, 965 238., 5.75149×10^{6}, 2.59995×10^{6}, 4.15853×10^{6}, 6.94307×10^{6}, 2.83159×10^{6}, 1.1455×10^{6}, 3.06476×10^{7}, 9.99102×10^{7}, 1.59719×10^{7}, 5.27596×10^{7}, 2.88736×10^{7}, 5.88639×10^{8}, 122 045., 9.03789×10^{7}, 1.79707×10^{6}, 1.4907×10^{7}, 8.65228×10^{6}, 587 162., 1.65016×10^{6}, 1.56119×10^{7}, 1.15858×10^{6}, 1.03354×10^{7}, 4.35174×10^{7}, 1.75966×10^{6}, 8.93638×10^{7}, 283 950., 3.73808×10^{6}, 3.25931×10^{6}, 2.95657×10^{6}, 6.23619×10^{6}, 979 172., 7.3947×10^{6}, 3.45437×10^{6}, 4.56902×10^{7}, 2.10845×10^{7}, 1.16649×10^{8}, 3.10378×10^{8}, 1.31416×10^{6}, 5.52601×10^{6}, 6.6883×10^{7}, 848 757., 1.96683×10^{6}, 3.28375×10^{7}, 1.53126×10^{6}, 1.13989×10^{7}, 2.25435×10^{6}, 1.19801×10^{7}, 4.03047×10^{7}, 2.69053×10^{7}, 7.39416×10^{7}, 5.08977×10^{7}, 374 226., 9.44276×10^{7}, 1.14941×10^{7}, 1.13525×10^{7}, 2.03753×10^{7}, 7.32885×10^{6}, 1.46456×10^{7}, 9.5417×10^{6}, 6.26107×10^{6}, 7.94898×10^{7}, 1.21491×10^{7}, 1.21222×10^{7}, 7.62617×10^{6}, 1.39656×10^{7}, 947 265., 697 514., 1.00706×10^{7}, 1.04381×10^{6}, 6.24638×10^{6}}

Income Figures

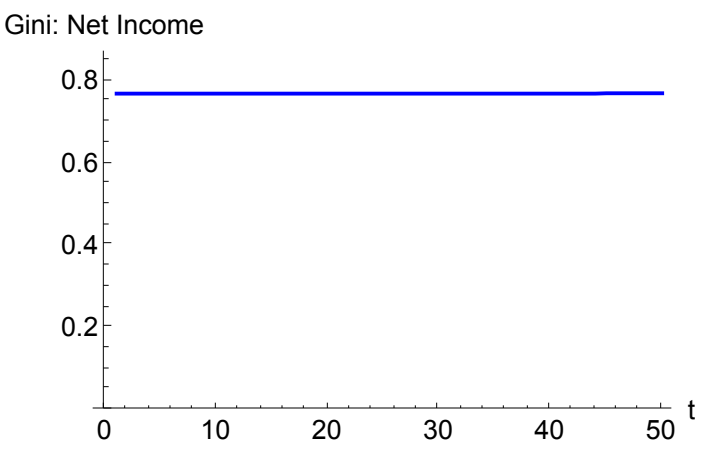
Figures on net and gross income.

In[ ]:=
```
Clear[Income];
Income = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  Income⟦t, i⟧ = Profit⟦t⟧ × Activities⟦t, i, 5⟧ + Wage⟦t⟧ × Activities⟦t, i, 6⟧
 ]
]

IncomeGini = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
```

$$\text{IncomeGini}[[t]] = \frac{N}{N-1}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\text{Abs}[\text{Income}[[t, i]] - \text{Income}[[t, j]]]\right) \Big/ \left(2\,N^2\,\text{Mean}[\text{Income}[[t]]]\right)\right)$$

```
]
IncomeGiniPlot = ListLinePlot[IncomeGini, PlotStyle → {Thick, Blue},
  PlotRange → {0, Max[IncomeGini] + 0.1}, AxesLabel → {"t", "Gini: Net Income"}, LabelStyle → 14]
```

Out[ ]=

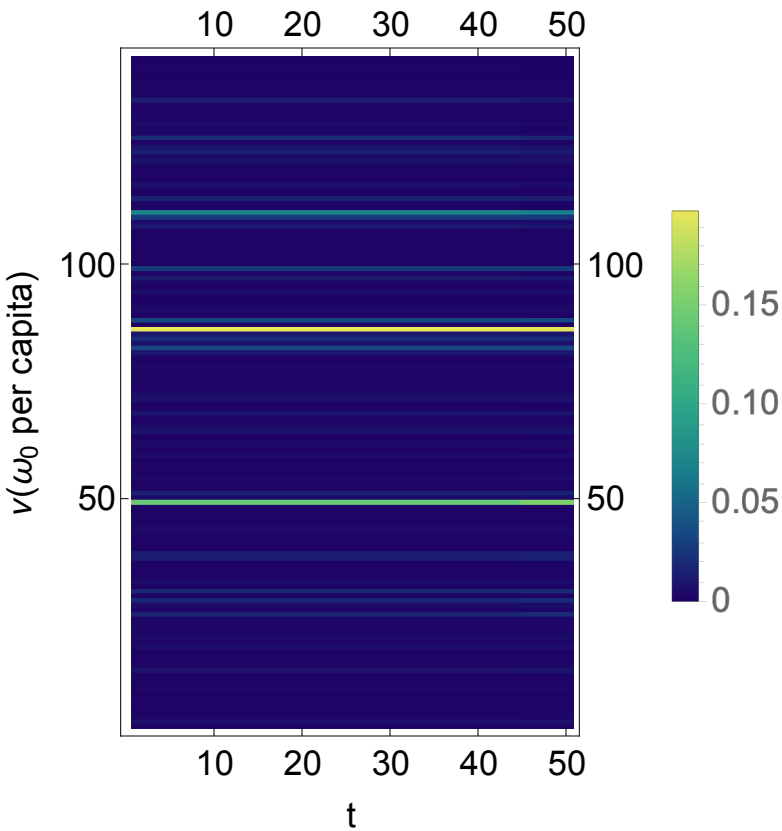


In[ ]:=
```
Export["./NetIncomeGini.eps", IncomeGiniPlot, "EPS"]
```

Out[ ]=
```
./NetIncomeGini.eps
```

In[ ]:=
```
Clear[IncomeShares];
IncomeShares = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  IncomeShares[[t, i]] = Income[[t, i]] / Total[Income[[t]]]
 ]
]
IncomeArray = ArrayPlot[Transpose[IncomeShares], FrameLabel → {"ν(ω0 per capita)", "t"},
  PlotLegends → Automatic, ColorFunction → "BlueGreenYellow", ColorFunctionScaling → True,
  DataReversed → True, FrameTicks → Automatic, AspectRatio → 1.5, LabelStyle → 18]
```

Out[ ]=



In[ ]:=
```
Export["./NetIncomeArray.eps", IncomeArray, "EPS"]
```

Out[ ]=

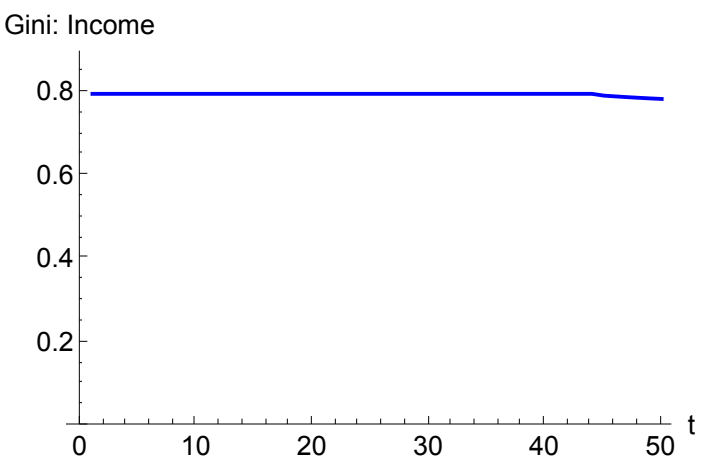
./NetIncomeArray.eps

In[ ]:=
```
Clear[GrossIncome];
GrossIncome = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  GrossIncome[[t, i]] = (1 + Profit[[t]]) Activities[[t, i, 5]] + Wage[[t]] × Activities[[t, i, 6]]
 ]
]

GrossIncomeGini = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
```

$$\text{GrossIncomeGini}[[t]] = \frac{N}{N-1}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\text{Abs}[\text{GrossIncome}[[t, i]] - \text{GrossIncome}[[t, j]]]\right) \Big/ \left(2\,N^2\,\text{Mean}[\text{GrossIncome}[[t]]]\right)\right)$$

```
]
GrossIncomeGiniPlot = ListLinePlot[GrossIncomeGini, PlotStyle → {Thick, Blue},
  PlotRange → {0, Max[GrossIncomeGini] + 0.1}, AxesLabel → {"t", "Gini: Income"}, LabelStyle → 14]
```

Out[ ]=

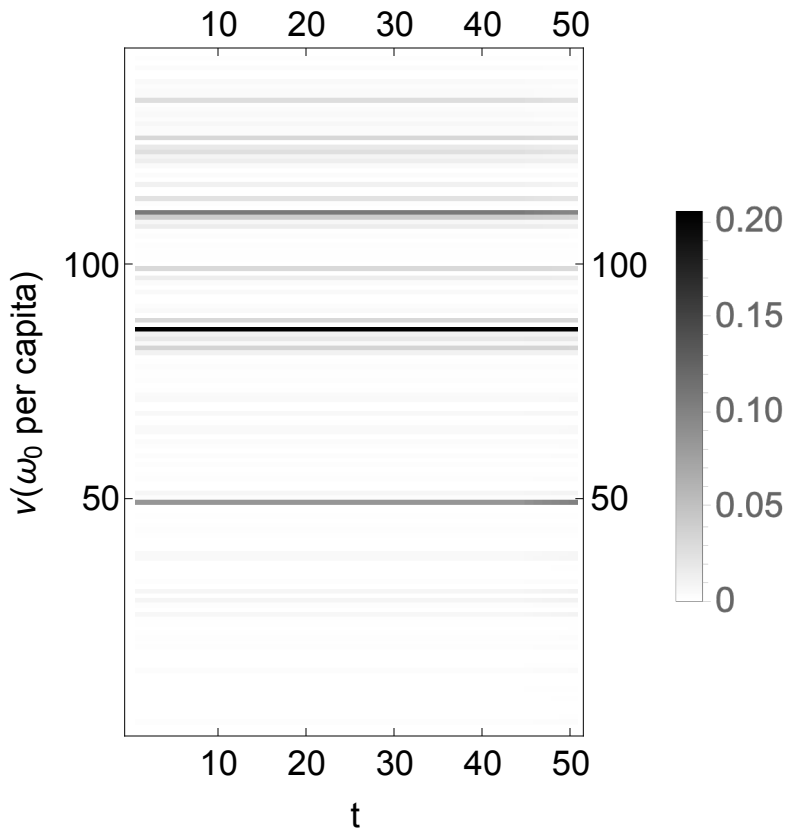


In[ ]:=
```
Export["./GrossIncomeGini.eps", GrossIncomeGiniPlot, "EPS"]
```
Out[ ]=
```
./GrossIncomeGini.eps
```

In[ ]:=
```
GrossIncomeGini
```
Out[ ]=
```
{0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337,
 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337,
 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337,
 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337, 0.796337,
 0.796337, 0.796337, 0.796337, 0.796337, 0.792145, 0.790245, 0.788496, 0.786895, 0.785418, 0.784087}
```

In[ ]:=
```
Clear[GrossIncomeShares];
GrossIncomeShares = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  GrossIncomeShares[[t, i]] = GrossIncome[[t, i]] / Total[GrossIncome[[t]]]
 ]
]
GrossIncomeArray =
 ArrayPlot[Transpose[GrossIncomeShares], FrameLabel → {"ν(ω0 per capita)", "t"}, PlotLegends → Automatic,
  ColorFunctionScaling → True, DataReversed → True, FrameTicks → Automatic, AspectRatio → 1.5, LabelStyle → 18]
```

Out[ ]=



In[ ]:=
```
Export["./GrossIncomeArray.eps", GrossIncomeArray, "EPS"]
```
Out[ ]=
```
./GrossIncomeArray.eps
```

Updated Simulation Reporting

In[ ]:=
```
(ExploitedTable = DeleteCases[Table[If[ExploitationIndex⟦1, i⟧ > 1.0,
      {CountryList⟦i⟧, "&", ExploitationIndex⟦1, i⟧, "\\\\"},], {i, 1, N}], Null]) // TableForm
Export["./ExploitedTable.csv", ExploitedTable, "CSV"]
```
Out[ ]//TableForm=
```
Burundi                    &    1.13049    \\
Congo - Kinshasa           &    1.12916    \\
```

```
Malawi                      &    1.12969    \\
Mali                        &    1.12412    \\
Sierra Leone                &    1.12487    \\
Liberia                     &    1.12569    \\
Mozambique                  &    1.11973    \\
Central African Republic    &    1.12177    \\
Madagascar                  &    1.12244    \\
Niger                       &    1.11604    \\
Rwanda                      &    1.12275    \\
Burkina Faso                &    1.11573    \\
Ethiopia                    &    1.11117    \\
Zimbabwe                    &    1.12211    \\
Togo                        &    1.1144     \\
Benin                       &    1.11463    \\
Gambia                      &    1.10999    \\
Kenya                       &    1.11746    \\
Yemen                       &    1.11105    \\
Uganda                      &    1.11665    \\
Nepal                       &    1.11001    \\
Cambodia                    &    1.11154    \\
Ivory Coast                 &    1.1063     \\
Cameroon                    &    1.10914    \\
Pakistan                    &    1.10611    \\
Senegal                     &    1.09697    \\
Myanmar                     &    1.10049    \\
Nigeria                     &    1.10013    \\
Mauritania                  &    1.09655    \\
Bangladesh                  &    1.10115    \\
Kyrgyzstan                  &    1.11503    \\
Tanzania                    &    1.0923     \\
Haiti                       &    1.09105    \\
Lesotho                     &    1.09163    \\
Bolivia                     &    1.10853    \\
Honduras                    &    1.09926    \\
Vietnam                     &    1.10446    \\
Egypt                       &    1.10261    \\
```

```
Belize                    &   1.10945   \\
Nicaragua                 &   1.08961   \\
El Salvador               &   1.08777   \\
Guatemala                 &   1.07896   \\
Sudan                     &   1.06752   \\
Syria                     &   1.0927    \\
Laos                      &   1.07552   \\
Zambia                    &   1.09076   \\
Moldova                   &   1.10019   \\
Fiji                      &   1.08758   \\
India                     &   1.07439   \\
Iraq                      &   1.07267   \\
Philippines               &   1.08208   \\
Paraguay                  &   1.07995   \\
Armenia                   &   1.0845    \\
Ghana                     &   1.06847   \\
Jordan                    &   1.07519   \\
Congo - Brazzaville       &   1.04751   \\
Angola                    &   1.01327   \\
Eswatini                  &   1.0426    \\
Peru                      &   1.06651   \\
Costa Rica                &   1.04565   \\
Sri Lanka                 &   1.05156   \\
Morocco                   &   1.01078   \\
Namibia                   &   1.02528   \\
Ukraine                   &   1.05809   \\
Colombia                  &   1.03728   \\
Tajikistan                &   1.05163   \\
Gabon                     &   1.04413   \\
South Africa              &   1.04347   \\
Mongolia                  &   1.04847   \\
Maldives                  &   1.02577   \\
Argentina                 &   1.04592   \\
Algeria                   &   1.0187    \\
Dominican Republic        &   1.03374   \\
Jamaica                   &   1.027     \\
```

```
Ecuador                    &    1.03026    \\
Bulgaria                   &    1.04048    \\
Tunisia                    &    1.01772    \\
Kazakhstan                 &    1.03462    \\
Serbia                     &    1.03767    \\
Albania                    &    1.02345    \\
Iran                       &    1.00021    \\
Poland                     &    1.03327    \\
Mexico                     &    1.0066     \\
Thailand                   &    1.00347    \\
Barbados                   &    1.00524    \\
Brazil                     &    1.00941    \\
Panama                     &    1.00248    \\
Chile                      &    1.00176    \\
```

Out[ ]=

./ExploitedTable.csv

In[ ]:=

```
(ExploiterTable = DeleteCases[Table[If[ExploitationIndex⟦1, i⟧ < 1.0,
      {CountryList⟦i⟧, "&", ExploitationIndex⟦1, i⟧, "\\\\"},], {i, 1, N}], Null]) // TableForm
Export["./ExploiterTable.csv", ExploiterTable, "CSV"]
```

Out[ ]//TableForm=

```
Indonesia                &    0.992845    \\
China                    &    0.993497    \\
Venezuela                &    0.991169    \\
Mauritius                &    0.978759    \\
Uruguay                  &    0.982048    \\
Malaysia                 &    0.993514    \\
Botswana                 &    0.981639    \\
Romania                  &    0.993798    \\
Turkey                   &    0.940936    \\
Lithuania                &    0.971079    \\
Russia                   &    0.97497     \\
Malta                    &    0.94927     \\
Slovakia                 &    0.976368    \\
New Zealand              &    0.95351     \\
Croatia                  &    0.958682    \\
```

```
Israel                 &   0.961767   \\
Estonia                &   0.952915   \\
Hungary                &   0.937314   \\
Kuwait                 &   0.826099   \\
South Korea            &   0.922282   \\
Taiwan                 &   0.898542   \\
Japan                  &   0.911188   \\
United States          &   0.911411   \\
Trinidad and Tobago    &   0.867535   \\
Finland                &   0.885655   \\
United Kingdom         &   0.900965   \\
Cyprus                 &   0.84347    \\
Latvia                 &   0.860474   \\
Saudi Arabia           &   0.825178   \\
Bahrain                &   0.772503   \\
Czech Republic         &   0.882461   \\
Slovenia               &   0.871979   \\
Greece                 &   0.843054   \\
Canada                 &   0.879379   \\
Australia              &   0.867892   \\
France                 &   0.844821   \\
Spain                  &   0.825634   \\
Iceland                &   0.843872   \\
Germany                &   0.867227   \\
Portugal               &   0.778101   \\
Sweden                 &   0.851387   \\
Netherlands            &   0.837697   \\
Denmark                &   0.835183   \\
Belgium                &   0.802683   \\
Hong Kong              &   0.808094   \\
Ireland                &   0.79722    \\
Italy                  &   0.790084   \\
Austria                &   0.799258   \\
Switzerland            &   0.815888   \\
Norway                 &   0.811557   \\
United Arab Emirates   &   0.726798   \\
```

```
Macao                  &    0.73243     \\
Brunei                 &    0.70479     \\
Singapore              &    0.781131    \\
Luxembourg             &    0.744122    \\
Qatar                  &    0.627681    \\
```

Out[ ]=

```
./ExploiterTable.csv
```

In[ ]:=

```
(ExploitIncomeCSV = Table[{CountryList⟦i⟧, "&", ExploitationIndex⟦1, i⟧, "\\\\"}, {i, 1, N}]) // TableForm
Export["./ExploitIncomeTable.csv", ExploitIncomeCSV, "CSV"]
```

Out[ ]//TableForm=

```
Burundi                    &    1.13049     \\
Congo - Kinshasa           &    1.12916     \\
Malawi                     &    1.12969     \\
Mali                       &    1.12412     \\
Sierra Leone               &    1.12487     \\
Liberia                    &    1.12569     \\
Mozambique                 &    1.11973     \\
Central African Republic   &    1.12177     \\
Madagascar                 &    1.12244     \\
Niger                      &    1.11604     \\
Rwanda                     &    1.12275     \\
Burkina Faso               &    1.11573     \\
Ethiopia                   &    1.11117     \\
Zimbabwe                   &    1.12211     \\
Togo                       &    1.1144      \\
Benin                      &    1.11463     \\
Gambia                     &    1.10999     \\
Kenya                      &    1.11746     \\
Yemen                      &    1.11105     \\
Uganda                     &    1.11665     \\
Nepal                      &    1.11001     \\
Cambodia                   &    1.11154     \\
Ivory Coast                &    1.1063      \\
Cameroon                   &    1.10914     \\
Pakistan                   &    1.10611     \\
```

```
Senegal                    &   1.09697    \\
Myanmar                    &   1.10049    \\
Nigeria                    &   1.10013    \\
Mauritania                 &   1.09655    \\
Bangladesh                 &   1.10115    \\
Kyrgyzstan                 &   1.11503    \\
Tanzania                   &   1.0923     \\
Haiti                      &   1.09105    \\
Lesotho                    &   1.09163    \\
Bolivia                    &   1.10853    \\
Honduras                   &   1.09926    \\
Vietnam                    &   1.10446    \\
Egypt                      &   1.10261    \\
Belize                     &   1.10945    \\
Nicaragua                  &   1.08961    \\
El Salvador                &   1.08777    \\
Guatemala                  &   1.07896    \\
Sudan                      &   1.06752    \\
Syria                      &   1.0927     \\
Laos                       &   1.07552    \\
Zambia                     &   1.09076    \\
Moldova                    &   1.10019    \\
Fiji                       &   1.08758    \\
India                      &   1.07439    \\
Iraq                       &   1.07267    \\
Philippines                &   1.08208    \\
Paraguay                   &   1.07995    \\
Armenia                    &   1.0845     \\
Ghana                      &   1.06847    \\
Jordan                     &   1.07519    \\
Congo – Brazzaville        &   1.04751    \\
Angola                     &   1.01327    \\
Eswatini                   &   1.0426     \\
Peru                       &   1.06651    \\
Costa Rica                 &   1.04565    \\
Sri Lanka                  &   1.05156    \\
```

```
Morocco                    &    1.01078     \\
Namibia                    &    1.02528     \\
Ukraine                    &    1.05809     \\
Colombia                   &    1.03728     \\
Tajikistan                 &    1.05163     \\
Gabon                      &    1.04413     \\
South Africa               &    1.04347     \\
Mongolia                   &    1.04847     \\
Maldives                   &    1.02577     \\
Argentina                  &    1.04592     \\
Algeria                    &    1.0187      \\
Dominican Republic         &    1.03374     \\
Jamaica                    &    1.027       \\
Ecuador                    &    1.03026     \\
Bulgaria                   &    1.04048     \\
Tunisia                    &    1.01772     \\
Kazakhstan                 &    1.03462     \\
Serbia                     &    1.03767     \\
Albania                    &    1.02345     \\
Iran                       &    1.00021     \\
Indonesia                  &    0.992845    \\
Poland                     &    1.03327     \\
Mexico                     &    1.0066      \\
Thailand                   &    1.00347     \\
China                      &    0.993497    \\
Barbados                   &    1.00524     \\
Brazil                     &    1.00941     \\
Panama                     &    1.00248     \\
Venezuela                  &    0.991169    \\
Chile                      &    1.00176     \\
Mauritius                  &    0.978759    \\
Uruguay                    &    0.982048    \\
Malaysia                   &    0.993514    \\
Botswana                   &    0.981639    \\
Romania                    &    0.993798    \\
Turkey                     &    0.940936    \\
```

```
Lithuania                  &   0.971079   \\
Russia                     &   0.97497    \\
Malta                      &   0.94927    \\
Slovakia                   &   0.976368   \\
New Zealand                &   0.95351    \\
Croatia                    &   0.958682   \\
Israel                     &   0.961767   \\
Estonia                    &   0.952915   \\
Hungary                    &   0.937314   \\
Kuwait                     &   0.826099   \\
South Korea                &   0.922282   \\
Taiwan                     &   0.898542   \\
Japan                      &   0.911188   \\
United States              &   0.911411   \\
Trinidad and Tobago        &   0.867535   \\
Finland                    &   0.885655   \\
United Kingdom             &   0.900965   \\
Cyprus                     &   0.84347    \\
Latvia                     &   0.860474   \\
Saudi Arabia               &   0.825178   \\
Bahrain                    &   0.772503   \\
Czech Republic             &   0.882461   \\
Slovenia                   &   0.871979   \\
Greece                     &   0.843054   \\
Canada                     &   0.879379   \\
Australia                  &   0.867892   \\
France                     &   0.844821   \\
Spain                      &   0.825634   \\
Iceland                    &   0.843872   \\
Germany                    &   0.867227   \\
Portugal                   &   0.778101   \\
Sweden                     &   0.851387   \\
Netherlands                &   0.837697   \\
Denmark                    &   0.835183   \\
Belgium                    &   0.802683   \\
Hong Kong                  &   0.808094   \\
```

```
Ireland                    &   0.79722    \\
Italy                      &   0.790084   \\
Austria                    &   0.799258   \\
Switzerland                &   0.815888   \\
Norway                     &   0.811557   \\
United Arab Emirates       &   0.726798   \\
Macao                      &   0.73243    \\
Brunei                     &   0.70479    \\
Singapore                  &   0.781131   \\
Luxembourg                 &   0.744122   \\
Qatar                      &   0.627681   \\
```

Out[ ]=

```
./ExploitIncomeTable.csv
```

In[ ]:=

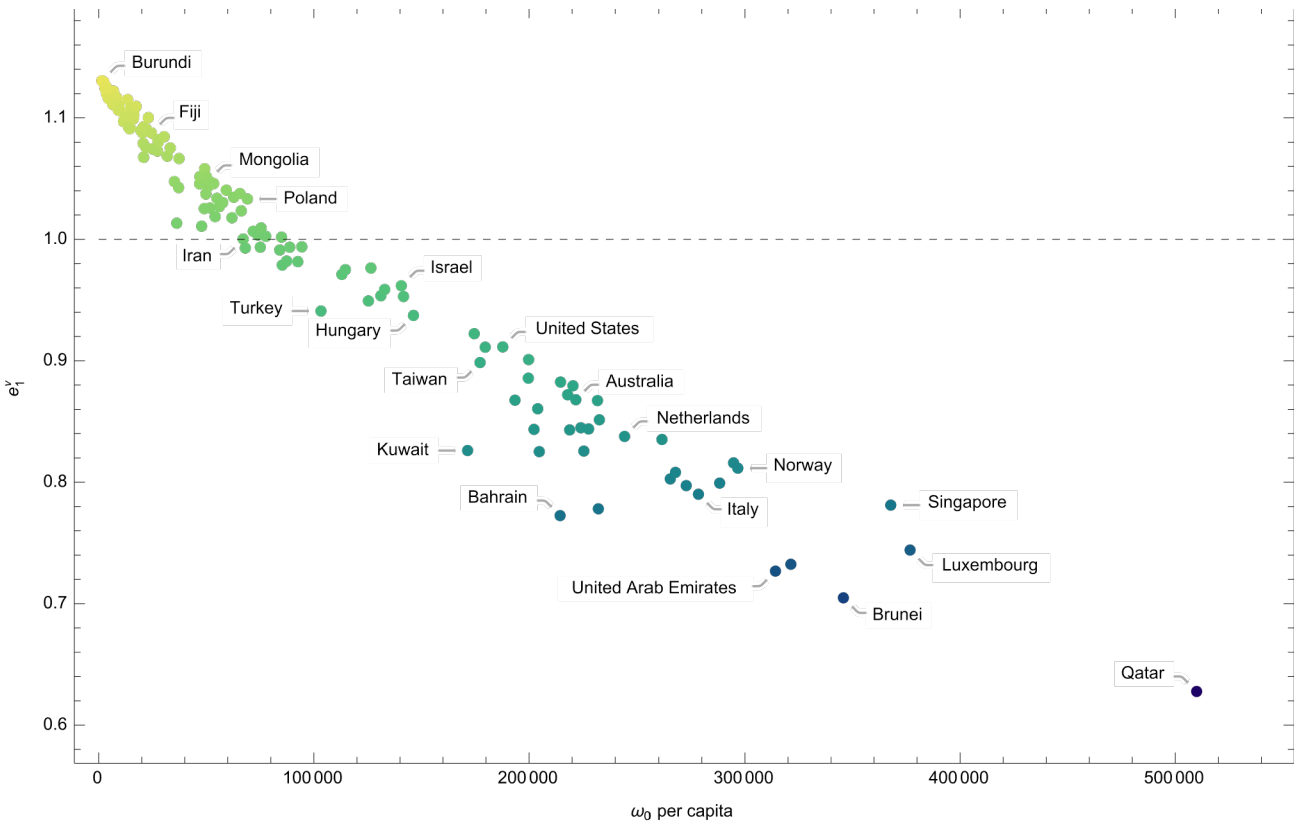
```
DataNoHeader[[1]]
```

Out[ ]=

```
{Burundi, 15963.5, 1.3893, 10.8642, 1469.37}
```

In[ ]:=
```
ExploitWealthPlot = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex[[1, i]]}, {i, 1, N}] → CountryList,
  LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e1^γ"},
  ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}]
Export["./ExploitWealthPlot.eps", ExploitWealthPlot, "EPS"]
```

Out[ ]=



Out[ ]=

./ExploitWealthPlot.eps

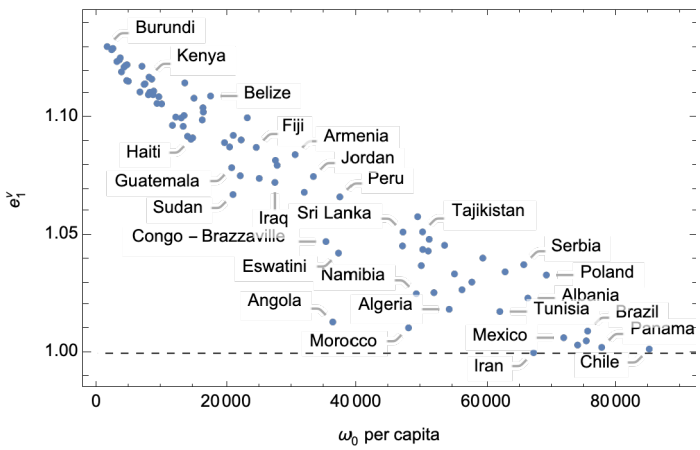
In[ ]:=
```
ExploitedCountries = DeleteCases[Table[If[ExploitationIndex[[1, i]] > 1, i, 0.], {i, 1, N}], 0.]
ExploiterCountries = DeleteCases[Table[If[ExploitationIndex[[1, i]] < 1, i, 0.], {i, 1, N}], 0.]
```

Out[ ]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 83, 84, 85, 87, 88, 89, 91}
```

Out[ ]=
```
{82, 86, 90, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106,
 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125,
 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144}
```

In[ ]:=
```
ExploitedCountriesPlot = ListPlot[
  Table[{DataNoHeader[[i, 5]], ExploitationIndex[[1, i]]}, {i, ExploitedCountries}] → CountryList[[ExploitedCountries]],
  LabelingFunction → Callout[Automatic, Automatic], Frame → True,
  FrameLabel → {"ω0 per capita", "e1^y"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}]
Export["./ExploitedCountriesPlot.eps", ExploitedCountriesPlot, "EPS"]
```

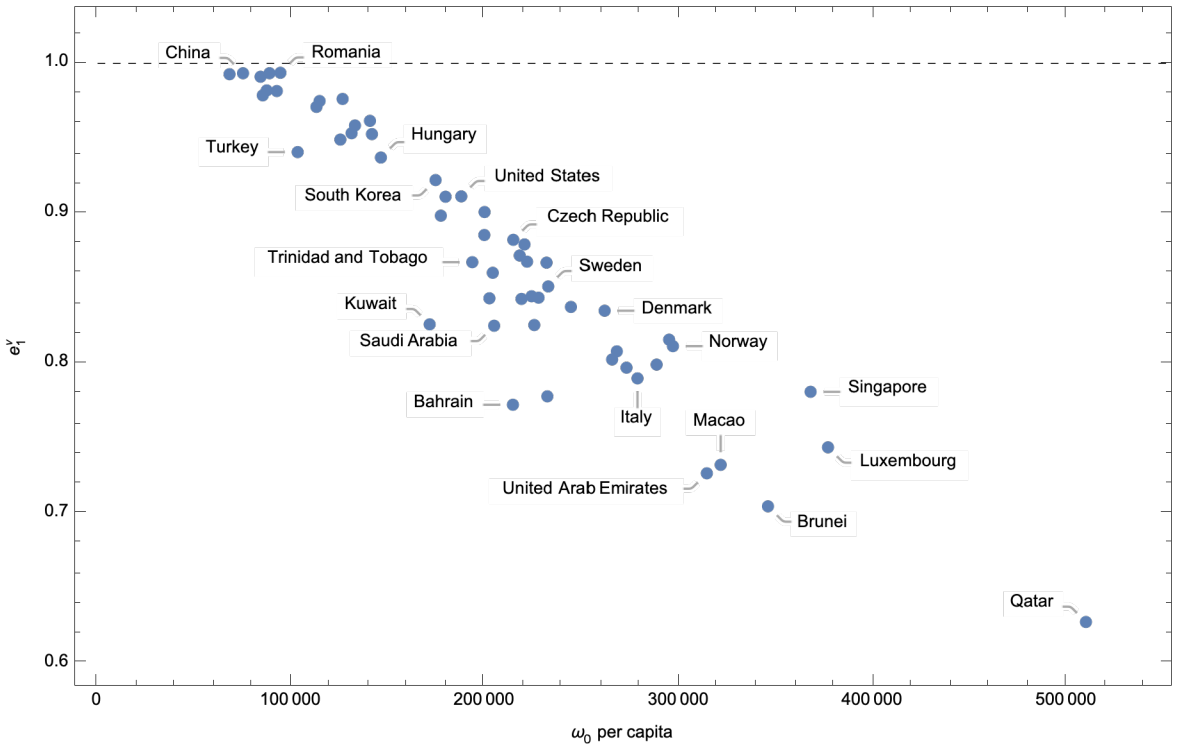
Out[ ]=



Out[ ]=
```
./ExploitedCountriesPlot.eps
```

In[ ]:=
```
ExploiterCountriesPlot = ListPlot[
  Table[{DataNoHeader[[i, 5]], ExploitationIndex[[1, i]]}, {i, ExploiterCountries}] → CountryList[[ExploiterCountries]],
  LabelingFunction → Callout[Automatic, Automatic], Frame → True,
  FrameLabel → {"ω0 per capita", "e1^y"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}]
Export["./ExploiterCountriesPlot.eps", ExploiterCountriesPlot, "EPS"]
```

Out[ ]=



Out[ ]=

./ExploiterCountriesPlot.eps

In[ ]:=
```
(outlierTesting = Table[{CountryList[[i]], AgentSet[[i, 1]], AgentSet[[i, 2]]}, {i, 1, Length[CountryList]}]) // TableForm
Export["./outlierTesting.csv", outlierTesting, "CSV"];
```

Out[ ]//TableForm=

Burundi	15 963.5	1.50936×10^6
Congo – Kinshasa	175 909.	1.35512×10^7

Malawi	43 799.5	3.64509×10^{6}
Mali	55 200.4	2.49149×10^{6}
Sierra Leone	25 355.6	1.2192×10^{6}
Liberia	16 526.4	856 947.
Mozambique	109 190.	3.61277×10^{6}
Central African Republic	19 002.4	717 688.
Madagascar	110 037.	4.36035×10^{6}
Niger	96 472.6	2.60283×10^{6}
Rwanda	55 347.6	2.2439×10^{6}
Burkina Faso	90 906.7	2.41549×10^{6}
Ethiopia	684 944.	1.48483×10^{7}
Zimbabwe	113 144.	4.37745×10^{6}
Togo	55 997.2	1.39649×10^{6}
Benin	81 619.9	2.05722×10^{6}
Gambia	16 445.6	340 129.
Kenya	394 330.	1.14573×10^{7}
Yemen	227 303.	4.90286×10^{6}
Uganda	357 990.	9.96565×10^{6}
Nepal	250 592.	5.18728×10^{6}
Cambodia	138 382.	3.04455×10^{6}
Ivory Coast	222 652.	4.02638×10^{6}
Cameroon	227 090.	4.54615×10^{6}
Pakistan	1.94586×10^{6}	3.49664×10^{7}
Senegal	182 939	2.50376×10^{6}
Myanmar	643 155.	9.69637×10^{6}
Nigeria	2.46166×10^{6}	3.67303×10^{7}
Mauritania	58 348.8	789 846.
Bangladesh	2.19523×10^{6}	3.37375×10^{7}
Kyrgyzstan	81 286.2	2.08755×10^{6}
Tanzania	773 946.	9.42927×10^{6}
Haiti	158 114.	1.87087×10^{6}
Lesotho	32 740.4	392 659.

Bolivia	164 294.	3.21513×10^6
Honduras	149 490.	2.17684×10^6
Vietnam	1.55284×10^6	2.64161×10^7
Egypt	1.59062×10^6	2.55365×10^7
Belize	6526.4	132 235.
Nicaragua	121 678.	1.39361×10^6
El Salvador	129 833.	1.4282×10^6
Guatemala	349 707.	3.23031×10^6
Sudan	846 309.	6.44925×10^6
Syria	382 274.	4.70158×10^6
Laos	150 884.	1.31074×10^6
Zambia	378 445.	4.44857×10^6
Moldova	93 525.5	1.39783×10^6
Fiji	22 156.6	242 774.
India	33 385 420	2.84415×10^8
Iraq	1.0465×10^6	8.66078×10^6
Philippines	2.87783×10^6	2.81904×10^7
Paraguay	188 595.	1.77421×10^6
Armenia	89 271.5	917 362.
Ghana	918 879.	7.10691×10^6
Jordan	322 888.	2.78905×10^6
Congo – Brazzaville	185 261.	1.07329×10^6
Angola	1.08025×10^6	4.37024×10^6
Eswatini	50 796.3	277 576.
Peru	1.20156×10^6	9.01567×10^6
Costa Rica	230 187.	1.30396×10^6
Sri Lanka	981 131	5.97894×10^6
Morocco	1.71027×10^6	6.76515×10^6
Namibia	124 294.	563 796.
Ukraine	2.1785×10^6	1.44715×10^7
Colombia	2.4444×10^6	1.25755×10^7
Tajikistan	446 238.	2.7218×10^6

Gabon	101 352.	563 889.
South Africa	2.88631×10^6	1.5933×10^7
Mongolia	156 974.	920 250.
Maldives	22 589.5	102 962.
Argentina	2 364 575	1.3438×10^7
Algeria	2.2363×10^6	9.51473×10^6
Dominican Republic	591 543.	2.92819×10^6
Jamaica	162 147.	748 328.
Ecuador	957 913.	4.57118×10^6
Bulgaria	420 190.	2.24072×10^6
Tunisia	714 257.	3.01103×10^6
Kazakhstan	1.14236×10^6	5.70857×10^6
Serbia	461 666.	2.38547×10^6
Albania	194 145.	864 761.
Iran	5.4493×10^6	1.96731×10^7
Indonesia	17 984 676	6.1143×10^7
Poland	2 638 046	1.29941×10^7
Mexico	9 272 062	3.53455×10^7
Thailand	5 103 972	1.89404×10^7
China	105 849 328	3.61739×10^8
Barbados	21 503.2	81 015.9
Brazil	15 796 820	6.17233×10^7
Panama	318 240.	1.17105×10^6
Venezuela	2.69051×10^6	9.02618×10^6
Chile	1.5342×10^6	5.61122×10^6
Mauritius	107 904.	329 318.
Uruguay	301 882.	944 128.
Malaysia	2 807 267	9.59513×10^6
Botswana	212 074.	661 224.
Romania	1.85761×10^6	6.3637×10^6
Turkey	8 341 127	1.97365×10^7
Lithuania	326 608.	943 018.
Russia	16 505 067	4.89997×10^7

Malta	53 994.7	134 634.
Slovakia	689 147.	2.06675×10^6
New Zealand	617 120.	1.58148×10^6
Croatia	556 734.	1.47622×10^6
Israel	1.1703×10^6	3.16812×10^6
Estonia	185 511.	473 564.
Hungary	1.422×10^6	3.29076×10^6
Kuwait	709 136.	927 900.
South Korea	8 896 485	1.88354×10^7
Taiwan	4.17467×10^6	7.75141×10^6
Japan	22 900 656	4.55413×10^7
United States	60 923 964	1.21306×10^8
Trinidad and Tobago	264 846.	420 026.
Finland	1.10275×10^6	1.91445×10^6
United Kingdom	13 221 941	2.48697×10^7
Cyprus	172 999.	244 835.
Latvia	397 775.	609 691.
Saudi Arabia	6.74351×10^6	8.78791×10^6
Bahrain	320 046.	333 924.
Czech Republic	2.2789×10^6	3.89246×10^6
Slovenia	453 281.	734 698.
Greece	2.4423×10^6	3.44983×10^6
Canada	8 071 759	1.3574×10^7
Australia	5.4213×10^6	8.61272×10^6
France	15 062 345	2.14499×10^7
Spain	10 450 522	1.36462×10^7
Iceland	76 263.6	108 131.
Germany	19 033 204	3.014×10^7
Portugal	2.39843×10^6	2.55996×10^6
Sweden	2.3058×10^6	3.3854×10^6
Netherlands	4.16323×10^6	5.73854×10^6
Denmark	1.49905×10^6	2.04288×10^6

Belgium	3.03226×10^6	3.58469×10^6
Hong Kong	1.97186×10^6	2.38549×10^6
Ireland	1.29868×10^6	1.50027×10^6
Italy	16 525 972	1.85304×10^7
Austria	2.51829×10^6	2.93425×10^6
Switzerland	2 498 143	3.12553×10^6
Norway	1.57407×10^6	1.93281×10^6
United Arab Emirates	2.9535×10^6	2.57572×10^6
Macao	200 069.	178 282.
Brunei	148 226.	118 960.
Singapore	2.09948×10^6	2.26879×10^6
Luxembourg	219 834.	204 954.
Qatar	1.34575×10^6	816 067.

Map Plots of Exploitation Intensity

In[ ]:=
```
CountryList
```

Out[ ]=
```
{Burundi, Congo - Kinshasa, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic, Madagascar,
 Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia,
 Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania, Haiti,
 Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria, Laos,
 Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, Congo - Brazzaville, Angola,
 Eswatini, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa,
 Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, Trinidad and Tobago,
 Finland, United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada,
 Australia, France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland,
 Italy, Austria, Switzerland, Norway, United Arab Emirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

Converting list of countries to those in Mathematica geographical data.

In[ ]:=
```
CountryListRevise = CountryList /. {"Congo - Kinshasa" → "DemocraticRepublicCongo",
   "Congo - Brazzaville" → "RepublicCongo", "Eswatini" → "Swaziland", "Trinidad and Tobago" → "TrinidadTobago"}
```

Out[ ]=
```
{Burundi, DemocraticRepublicCongo, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic,
 Madagascar, Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal,
 Cambodia, Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania,
 Haiti, Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria,
 Laos, Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, RepublicCongo, Angola,
 Swaziland, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa,
 Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, TrinidadTobago, Finland,
 United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada, Australia,
 France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland,
 Italy, Austria, Switzerland, Norway, United Arab Emirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

Converting country names to strings compatible with Mathematica’s geographical data.

In[ ]:=
```
countriesNoSpace = StringReplace[#, " " → ""] & /@ CountryListRevise
```

Out[ ]=
```
{Burundi, DemocraticRepublicCongo, Malawi, Mali, SierraLeone, Liberia, Mozambique, CentralAfricanRepublic, Madagascar,
 Niger, Rwanda, BurkinaFaso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia,
 IvoryCoast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania,
 Haiti, Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, ElSalvador, Guatemala, Sudan, Syria,
 Laos, Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, RepublicCongo, Angola,
 Swaziland, Peru, CostaRica, SriLanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, SouthAfrica,
 Mongolia, Maldives, Argentina, Algeria, DominicanRepublic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, NewZealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, SouthKorea, Taiwan, Japan, UnitedStates, TrinidadTobago, Finland,
 UnitedKingdom, Cyprus, Latvia, SaudiArabia, Bahrain, CzechRepublic, Slovenia, Greece, Canada, Australia,
 France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, HongKong, Ireland,
 Italy, Austria, Switzerland, Norway, UnitedArabEmirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

Attaching countries in current sub-sample to geographical entities in Mathematica geographical data.

In[ ]:=
```
countryEntities = Table[Entity["Country", ToString[countriesNoSpace[[i]]]], {i, 1, Length[CountryList]}]
```

Out[ ]=

{Burundi, Democratic Republic of the Congo, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic, Madagascar, Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia, Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania, Haiti, Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria, Laos, Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, Republic of the Congo, Angola, Eswatini, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa, Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan, Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile, Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand, Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, Trinidad and Tobago, Finland, United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada, Australia, France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland, Italy, Austria, Switzerland, Norway, United Arab Emirates, Macau, Brunei, Singapore, Luxembourg, Qatar}

Threading country entities to exploitation intensity index.

In[ ]:=
```
Thread[countryEntities → ExploitationIndex[[1]]]
```

Out[ ]=

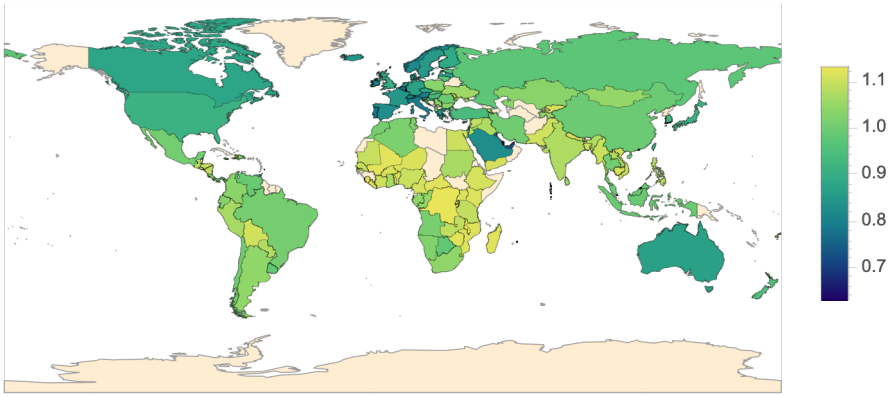
{Burundi → 1.13049, Democratic Republic of the Congo → 1.12916, Malawi → 1.12969, Mali → 1.12412, Sierra Leone → 1.12487, Liberia → 1.12569, Mozambique → 1.11973, Central African Republic → 1.12177,

Madagascar → 1.12244, Niger → 1.11604, Rwanda → 1.12275, Burkina Faso → 1.11573, Ethiopia → 1.11117,
Zimbabwe → 1.12211, Togo → 1.1144, Benin → 1.11463, Gambia → 1.10999, Kenya → 1.11746, Yemen → 1.11105,
Uganda → 1.11665, Nepal → 1.11001, Cambodia → 1.11154, Ivory Coast → 1.1063, Cameroon → 1.10914,
Pakistan → 1.10611, Senegal → 1.09697, Myanmar → 1.10049, Nigeria → 1.10013, Mauritania → 1.09655,
Bangladesh → 1.10115, Kyrgyzstan → 1.11503, Tanzania → 1.0923, Haiti → 1.09105, Lesotho → 1.09163,
Bolivia → 1.10853, Honduras → 1.09926, Vietnam → 1.10446, Egypt → 1.10261, Belize → 1.10945,
Nicaragua → 1.08961, El Salvador → 1.08777, Guatemala → 1.07896, Sudan → 1.06752, Syria → 1.0927,
Laos → 1.07552, Zambia → 1.09076, Moldova → 1.10019, Fiji → 1.08758, India → 1.07439, Iraq → 1.07267,
Philippines → 1.08208, Paraguay → 1.07995, Armenia → 1.0845, Ghana → 1.06847, Jordan → 1.07519,
Republic of the Congo → 1.04751, Angola → 1.01327, Eswatini → 1.0426, Peru → 1.06651, Costa Rica → 1.04565,
Sri Lanka → 1.05156, Morocco → 1.01078, Namibia → 1.02528, Ukraine → 1.05809, Colombia → 1.03728,
Tajikistan → 1.05163, Gabon → 1.04413, South Africa → 1.04347, Mongolia → 1.04847, Maldives → 1.02577,
Argentina → 1.04592, Algeria → 1.0187, Dominican Republic → 1.03374, Jamaica → 1.027, Ecuador → 1.03026,
Bulgaria → 1.04048, Tunisia → 1.01772, Kazakhstan → 1.03462, Serbia → 1.03767, Albania → 1.02345,
Iran → 1.00021, Indonesia → 0.992845, Poland → 1.03327, Mexico → 1.0066, Thailand → 1.00347,
China → 0.993497, Barbados → 1.00524, Brazil → 1.00941, Panama → 1.00248, Venezuela → 0.991169,
Chile → 1.00176, Mauritius → 0.978759, Uruguay → 0.982048, Malaysia → 0.993514, Botswana → 0.981639,
Romania → 0.993798, Turkey → 0.940936, Lithuania → 0.971079, Russia → 0.97497, Malta → 0.94927,
Slovakia → 0.976368, New Zealand → 0.95351, Croatia → 0.958682, Israel → 0.961767, Estonia → 0.952915,
Hungary → 0.937314, Kuwait → 0.826099, South Korea → 0.922282, Taiwan → 0.898542, Japan → 0.911188,

```
United States → 0.911411, Trinidad and Tobago → 0.867535, Finland → 0.885655, United Kingdom → 0.900965,
Cyprus → 0.84347, Latvia → 0.860474, Saudi Arabia → 0.825178, Bahrain → 0.772503, Czech Republic → 0.882461,
Slovenia → 0.871979, Greece → 0.843054, Canada → 0.879379, Australia → 0.867892, France → 0.844821,
Spain → 0.825634, Iceland → 0.843872, Germany → 0.867227, Portugal → 0.778101, Sweden → 0.851387,
Netherlands → 0.837697, Denmark → 0.835183, Belgium → 0.802683, Hong Kong → 0.808094, Ireland → 0.79722,
Italy → 0.790084, Austria → 0.799258, Switzerland → 0.815888, Norway → 0.811557, United Arab Emirates → 0.726798,
Macau → 0.73243, Brunei → 0.70479, Singapore → 0.781131, Luxembourg → 0.744122, Qatar → 0.627681}
```

In[ ]:=
```
mapExploitationIntensity = GeoRegionValuePlot[Thread[countryEntities → ExploitationIndex[[1]]],
  ColorFunction → "BlueGreenYellow", GeoBackground → {"CountryBorders", "Ocean" → White}, GeoRange → All]
Export["./mapExploitationIntensity.eps", mapExploitationIntensity, "EPS"];
Export["./mapExploitationIntensity.jpg", mapExploitationIntensity, "JPEG"];
```

Out[ ]=



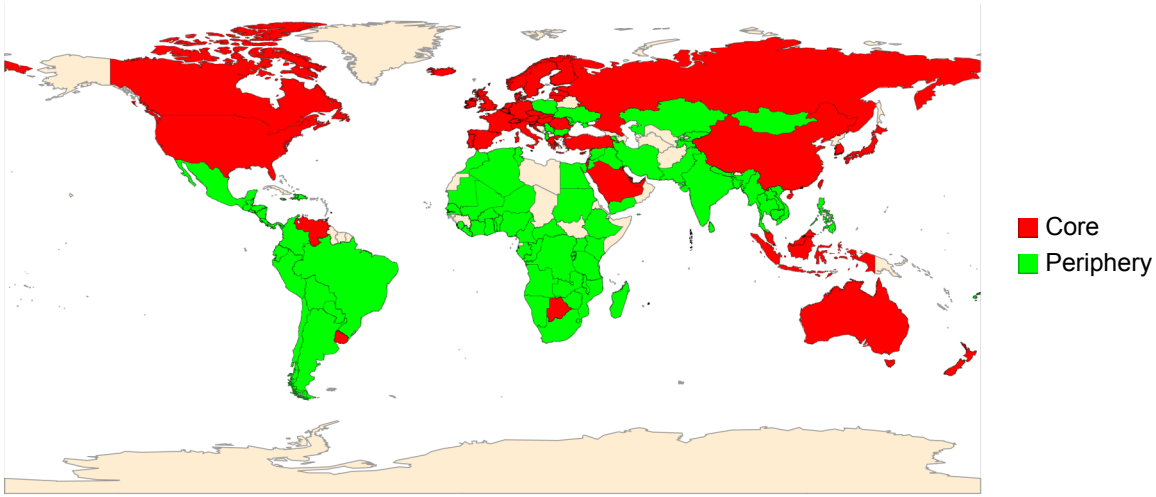
Partitioning exploitation intensity index into periphery and core groups.

In[ ]:=
```
corePerip = Table[If[ExploitationIndex[[1, i]] > 1, 1., 0.], {i, 1, Length[CountryList]}]
```

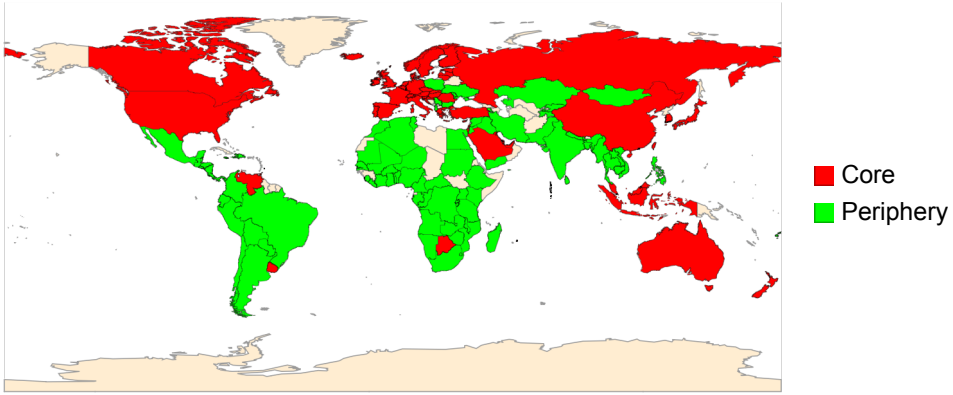
Out[ ]=
```
{1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.,
 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}
```

In[ ]:=
```
mapCorePeriphery = GeoRegionValuePlot[Thread[countryEntities → corePerip],
  ColorRules → {1. → Green, 0. → Red}, GeoBackground → {"CountryBorders", "Ocean" → White},
  GeoRange → All, PlotLegends → SwatchLegend[{Red, Green}, {"Core", "Periphery"}]]
Export["./mapCorePeriphery.eps", mapCorePeriphery, "EPS"];
Export["./mapCorePeriphery.jpg", mapCorePeriphery, "JPEG"];
```

Out[ ]=



Out[ ]=



Endogenous Consumption and Technical Change

The following section introduces an endogenous subsistence bundle and technical change. The optimisation procedures remain the same as above: **CapitalConstrained**, **LabourConstrained**, and **KnifeEdge**.

In[ ]:=
```
T = 50
```
Out[ ]=
```
50
```

Changing Subsistence

Subsistence b_t updates at the same rate as overall endowments grow g_t, with $g_t = \frac{\omega_{t-1}-\omega_{t-2}}{\omega_{t-2}}$. Subsistence b_1 begins at some initial value **Subsistence[[1]]** and then updates as needed according to $b_t = b_{t-1}\left(1 + \phi \frac{\omega_{t-1}-\omega_{t-2}}{\omega_{t-2}}\right)$, but never goes above a point at which $1 > v_t\, b_t$ is violated. The parameter ϕ is introduced to dampen the growth of subsistence if desired.

In[ ]:=
```
Clear[Subsistence];
Subsistence = Table[0.0, {T}];
Subsistence[[1]] = 0.3;
ϕ = 1;
```

Technical Change

In[ ]:=
```
rStar = 0.01;
```

In[ ]:=
```
Clear[𝒜, ℒ];
𝒜 = Table[0.0, {T}];
ℒ = Table[0.0, {T}];
𝒜[[1]] = 0.5;
ℒ[[1]] = 1.;
```

Starting Price and Wage

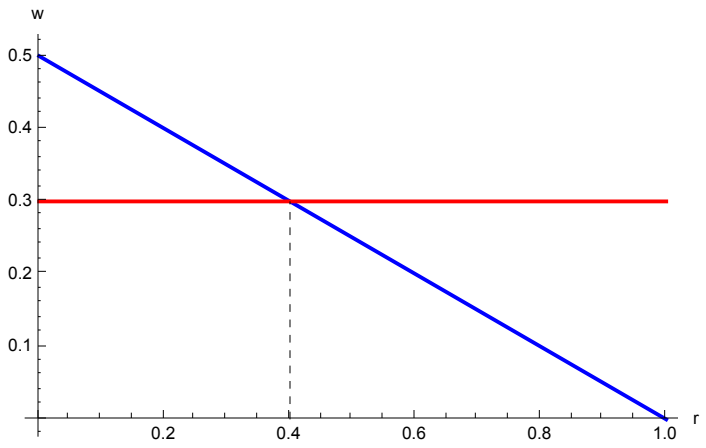
In[ ]:=
```
(TechChangePrice = {
    p == (1 + r) p a + w ℓ
  } /. p → 1) // TableForm
```

Out[ ]//TableForm=

$1 == a\,(1 + r) + w\,\ell$

In[ ]:=
```
(TechChangeWage = Solve[TechChangePrice, w] // Simplify);
Quiet[Plot[{w /. TechChangeWage /. {p → 1, a → 𝒜[[1]], ℓ → ℒ[[1]]}, Subsistence[[1]]}, {r, 0, (1 - 𝒜[[1]])/𝒜[[1]]},
  PlotStyle → {{Thick, Blue}, {Thick, Red}}, AxesOrigin → {0, 0}, AxesLabel → {"r", "w"}, Epilog →
   {Dashed, Black, Line[{{r /. Solve[TechChangePrice, r][[1]] /. {p → 1, a → 𝒜[[1]], ℓ → ℒ[[1]]} /. w → Subsistence[[1]], 0},
      {r /. Solve[TechChangePrice, r][[1]] /. {p → 1, a → 𝒜[[1]], ℓ → ℒ[[1]]} /. w → Subsistence[[1]], Subsistence[[1]]}}]}]]
```

Out[ ]=



Values

In[ ]:=
```
EmbodiedValues42 = Solve[v == ℓ (1 - a)^-1, v][[1]]
```

Out[ ]=

$$\left\{v \to -\frac{\ell}{-1+a}\right\}$$

In[ ]:=
```
EmbodiedValues42 /. {a → 𝒜[[1]], ℓ → ℒ[[1]]}
```

Out[ ]=

$$\{v \to 2.\}$$

Simulation

The simulation itself occurs in the following order: (1) initialisation, $t = 1$; (2) update b_t; (3) determine w_t and π_t; (4) update $\mathcal{A}_t$ and $\mathcal{L}_t$ if appropriate, and update π_t again if appropriate; (5) agents solve their optimisation problem; (6) repeat (2)-(5) for T.

In[ ]:=
```
Clear[Activities, i, Wage42, Profit42, u];
(Activities = Table[Table[{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}, {N}], {T}]);
Wage42 = Table[0.0, {T}];
Profit42 = Table[0.0, {T}];

(* First Stage t=1 simulation *)
(* Setting w1 and π1 *)
If[Total[l] ≥ ℒ[[1]] 𝒜[[1]]^-1 Total[AgentSet[[All, 1]]],
  If[Total[l] > ℒ[[1]] 𝒜[[1]]^-1 Total[AgentSet[[All, 1]]],
   Wage42[[1]] = Subsistence[[1]], Wage42[[1]] = RandomReal[{Subsistence[[1]], (1 - 𝒜[[1]] (1 + 0))/ℒ[[1]]}]],
  Wage42[[1]] = (1 - 𝒜[[1]] (1 + 0))/ℒ[[1]]];
Profit42[[1]] = ((p - p0 𝒜[[1]] - Wage42[[1]] × ℒ[[1]])/(p0 𝒜[[1]])) /. {p → 1, p0 → 1};
```

```
(* Checking whether the simulation is capital constrained, labour constrained,
or on the knife-edge and calling the corresponding function to find optimal (x_t^ν, y_t^ν, z_t^ν, δ_t^ν) for all ν and t=1*)
If[Total[l] ≥ ℒ⟦1⟧ 𝒜⟦1⟧^-1 Total[AgentSet⟦All, 1⟧],
  If[Total[l] > ℒ⟦1⟧ 𝒜⟦1⟧^-1 Total[AgentSet⟦All, 1⟧], CapitalConstrained[1, 𝒜⟦1⟧, ℒ⟦1⟧, l,
    1, Profit42⟦1⟧, Subsistence⟦1⟧], KnifeEdge[1, 𝒜⟦1⟧, ℒ⟦1⟧, l, 1, Profit42⟦1⟧, Subsistence⟦1⟧]],
  LabourConstrained[1, 𝒜⟦1⟧, ℒ⟦1⟧, l, 1, Profit42⟦1⟧, Subsistence⟦1⟧]
 ];

(* Simulation for remaining T-1 time steps *)
For[t = 2, t ≤ T, t++,
 (* Updating subsistence bundle *)
 Subsistence⟦t⟧ = Subsistence⟦t - 1⟧
   (1 + ϕ ((Total[Activities⟦t - 1, All, 5⟧] - (If[t == 2, Total[AgentSet⟦All, 1⟧], Total[Activities⟦t - 2, All, 5⟧]])) /
        (If[t == 2, Total[AgentSet⟦All, 1⟧], Total[Activities⟦t - 2, All, 5⟧]]))));

 (* Updating w_t *)
 If[Total[l] ≥ ℒ⟦t - 1⟧ 𝒜⟦t - 1⟧^-1 Total[Activities⟦t - 1, All, 5⟧],
  If[Total[l] > ℒ⟦t - 1⟧ 𝒜⟦t - 1⟧^-1 Total[Activities⟦t - 1, All, 5⟧], Wage42⟦t⟧ = Subsistence⟦t⟧,
   Wage42⟦t⟧ = RandomReal[{Subsistence⟦t⟧, ((1 - 𝒜⟦t - 1⟧ (1 + r)) / ℒ⟦t - 1⟧ /. r → 0)}]], Wage42⟦t⟧ = ((1 - 𝒜⟦t - 1⟧ (1 + r)) / ℒ⟦t - 1⟧ /. r → 0)];

 (* Updating π_t *)
 Profit42⟦t⟧ = ((p - p0 𝒜⟦t - 1⟧ - Wage42⟦t⟧ × ℒ⟦t - 1⟧) / (p0 𝒜⟦t - 1⟧)) /. {p → 1, p0 → 1};

 (* Updating 𝒜_t and ℒ_t if appropriate *)
 Clear[rNew];
 If[Profit42⟦t⟧ < rStar, rNew = RandomVariate[UniformDistribution[{Profit42⟦t - 1⟧, Max[Profit42]}]], 0];
```

```
Clear[AChange];
AChange = RandomVariate[UniformDistribution[{0.01, 0.03}]];
If[Profit42[[t]] < rStar,
 𝒜[[t]] = Min[{𝒜[[t - 1]] + AChange, 0.991}], 𝒜[[t]] = 𝒜[[t - 1]]];

Clear[v1, v2, ℒnew, s, h];
If[Profit42[[t]] < rStar, ℒ[[t]] = (1 - 𝒜[[t]] - 𝒜[[t]] rNew)/Subsistence[[t]], ℒ[[t]] = ℒ[[t - 1]]];
If[ℒ[[t]] < 0, ℒ[[t]] = (1 - 𝒜[[t]] - 𝒜[[t]] (rNew - 0.02))/Subsistence[[t]], 0];


(* Updating profit rate again in the event that technical change has taken place *)
Profit42[[t]] = ((p - p0 𝒜[[t]] - Wage42[[t]] × ℒ[[t]])/(p0 𝒜[[t]])) /. {p → 1, p0 → 1};

(* Checking whether the simulation is capital constrained, labour constrained,
or on the knife-edge and calling the corresponding function to find optimal (x_t^v, y_t^v, z_t^v, δ_t^v) for all v *)
If[Total[l] ≥ ℒ[[t]] 𝒜[[t]]^-1 Total[Activities[[t - 1, All, 5]]],
 If[Total[l] > ℒ[[t]] 𝒜[[t]]^-1 Total[Activities[[t - 1, All, 5]]], CapitalConstrained[1, 𝒜[[t]], ℒ[[t]], l,
   t, Profit42[[t]], Subsistence[[t]]], KnifeEdge[1, 𝒜[[t]], ℒ[[t]], l, t, Profit42[[t]], Subsistence[[t]]]],
 LabourConstrained[1, 𝒜[[t]], ℒ[[t]], l, t, Profit42[[t]], Subsistence[[t]]]
];

]
```

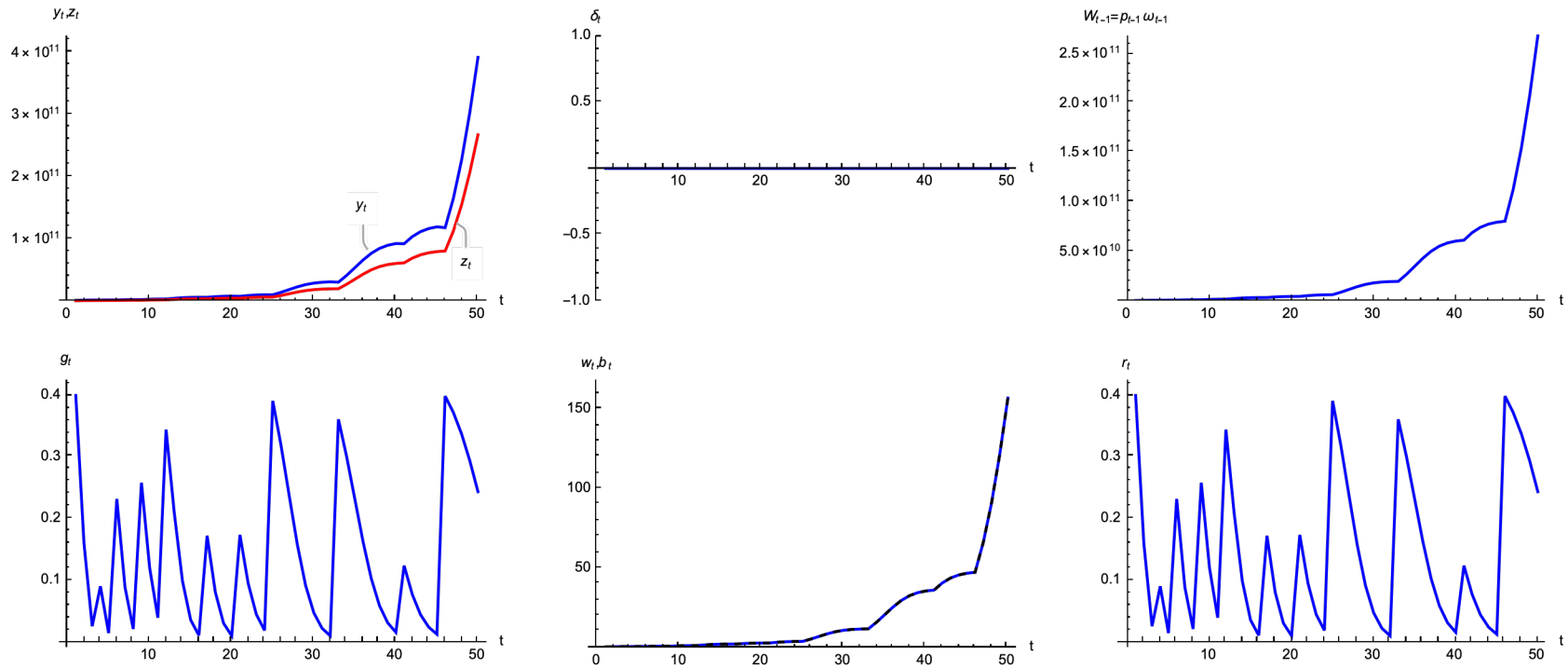
Results of Simulation with Endogenous Subsistence and Technical Change

In[ ]:=
```
Clear[Output42, Wealth42, WealthPlot42, XLevel42, YLevel42, ZLevel42, Deltas42, j, k];
Output42 = Table[0.0, {T}];
```

```
Wealth42 = Table[0.0, {T}];
WealthPlot42 = Table[0.0, {T}];
XLevel42 = Table[0.0, {T}];
YLevel42 = Table[0.0, {T}];
ZLevel42 = Table[0.0, {T}];
Deltas42 = Table[0.0, {T}];
For[j = 1, j ≤ T, j++,
 Output42[[j]] = (p x - p1 𝒜[[j]] x + (p - p1 𝒜[[j]] (1 + r)) y + (1 + r) z - z) /. {p → 1, p1 → 1, r → Profit42[[j]],
    x → Total[Activities[[j, All, 1]]], y → Total[Activities[[j, All, 2]]], z → Total[Activities[[j, All, 3]]]}
]

For[k = 1, k ≤ T, k++,
 Wealth42[[k]] = Total[Activities[[k, All, 5]]]
]
WealthPlot42[[1]] = Total[AgentSet[[All, 1]]];
For[k = 2, k ≤ T, k++,
 WealthPlot42[[k]] = Total[Activities[[k - 1, All, 5]]]
]
For[j = 1, j ≤ T, j++,
 XLevel42[[j]] = Total[Activities[[j, All, 1]]]
]
For[j = 1, j ≤ T, j++,
 YLevel42[[j]] = Total[Activities[[j, All, 2]]]
]
For[j = 1, j ≤ T, j++,
 ZLevel42[[j]] = Total[Activities[[j, All, 3]]]
]
For[j = 1, j ≤ T, j++,
 Deltas42[[j]] = Total[Activities[[j, All, 4]]]
]
```

```
Fig1TechChange = GraphicsGrid[{
   {
    ListLinePlot[{Callout[YLevel42, "yt", Scaled[0.2]], Callout[ZLevel42, "zt", Scaled[0.65], Scaled[0.5]]},
     PlotStyle → {{Blue}, {Red}}, PlotRange → {0, Max[YLevel42]}, AxesLabel → {"t", "yt,zt"},
     LabelStyle → s1, ImageSize → s2, ImagePadding → {{50, 0}, {0, 0}}],
    ListLinePlot[Deltas42, PlotStyle → {Blue},
     PlotRange → {0, Max[Deltas42]}, AxesLabel → {"t", "δt"}, LabelStyle → s1, ImageSize → s2],
    ListLinePlot[WealthPlot42, PlotRange → {0, Max[WealthPlot42]}, AxesLabel → {"t", "Wt-1=pt-1ωt-1"},
     PlotStyle → {Blue}, LabelStyle → s1, ImageSize → s2, ImagePadding → {{50, 0}, {0, 0}}]
   },
   {
    ListLinePlot[
     Table[(Total[Activities⟦t, All, 5⟧] - If[t == 1, Total[AgentSet⟦All, 1⟧], Total[Activities⟦t - 1, All, 5⟧]]) /
       If[t == 1, Total[AgentSet⟦All, 1⟧], Total[Activities⟦t - 1, All, 5⟧]], {t, 1, T}], PlotStyle → {Blue},
     PlotRange → All, AxesLabel → {"t", "gt"}, AxesOrigin → {0, 0}, LabelStyle → s1, ImageSize → s2],
    ListLinePlot[{Wage42, Subsistence}, PlotStyle → {{Blue}, {Dashed, Black}}, PlotRange → {0, Max[Subsistence] + 10},
     AxesLabel → {"t", "wt,bt"}, AxesOrigin → {0, 0}, LabelStyle → s1, ImageSize → s2],
    ListLinePlot[Profit42, PlotStyle → {Blue}, PlotRange → All,
     AxesLabel → {"t", "rt"}, AxesOrigin → {0, 0}, LabelStyle → s1, ImageSize → s2]
   }
  }, Spacings → {-10, 20}, ImageSize → Full]
Fig1TCx = ListLinePlot[XLevel42, PlotStyle → {Blue}, PlotRange → {0, Max[XLevel42]}, AxesLabel → {"t", "xt"}];
Fig1TCout = ListLinePlot[Output42, PlotRange → {0, Max[Output42]}, AxesLabel → {"t", "(1-At)yt"}, PlotStyle → {Blue}];
Fig1TCoutcap = ListLinePlot[Output42/N, PlotRange → All, AxesLabel → {"t", "((1-At)yt)/Nt"}, PlotStyle → {Thick, Blue}];
Export["./Fig1TechChange.eps", Fig1TechChange, "EPS"]
Export["./Fig1TCx.eps", Fig1TCx, "EPS"]
Export["./Fig1TCout.eps", Fig1TCout, "EPS"]
Export["./Fig1TCoutcap.eps", Fig1TCoutcap, "EPS"]
```

Out[ ]=



Out[ ]=

```
./Fig1TechChange.eps
```

Out[ ]=

```
./Fig1TCx.eps
```

Out[ ]=

```
./Fig1TCout.eps
```

Out[ ]=

```
./Fig1TCoutcap.eps
```

In[ ]:=

```
Clear[CapitalMarketEqb42, GoodsMarketEqb42, AyTest42, j, k];
CapitalMarketEqb42 = Table[0.0, {T}];
GoodsMarketEqb42 = Table[0.0, {T}];
AyTest42 = Table[0.0, {T}];
```

```
CapitalMarketEqb42[[1]] = Total[AgentSet[[All, 1]]] - (𝒜[[1]] (XLevel42[[1]] + YLevel42[[1]]) + Deltas42[[1]]);
For[k = 2, k ≤ T, k++,
 CapitalMarketEqb42[[k]] = Wealth42[[k - 1]] - (𝒜[[k]] (XLevel42[[k]] + YLevel42[[k]]) + Deltas42[[k]])
]

Clear[k];
For[k = 1, k ≤ T, k++,
 GoodsMarketEqb42[[k]] =
  ((XLevel42[[k]] + YLevel42[[k]]) + Deltas42[[k]]) - (Total[Subsistence[[k]] × Activities[[k, All, 6]]] + Wealth42[[k]])
]

Clear[k];
For[k = 1, k ≤ T, k++,
 AyTest42[[k]] = 𝒜[[k]] × Total[Activities[[k, All, 2]]] - Total[Activities[[k, All, 3]]]
]

Fig2TechChange = GraphicsRow[
   {
    ListLinePlot[CapitalMarketEqb42, PlotStyle → {Thick, Blue},
     PlotRange → {-0.01, 0.01}, AxesLabel → {"t", "ω_{t-1}-(A_t (x_t+y_t)+δ_t)"},
     PlotLabel → Style["Capital Market Equilibrium", 9], AxesOrigin → {0, 0}, LabelStyle → 9],
    ListLinePlot[ZLevel42 - (𝒜 YLevel42), PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01}, AxesLabel →
      {"t", "z_t-A_t y_t"}, PlotLabel → Style["Credit Market Equilibrium", 9], AxesOrigin → {0, 0}, LabelStyle → 9],
    ListLinePlot[GoodsMarketEqb42, PlotStyle → {Thick, Blue},
     PlotRange → {-0.01, 0.01}, AxesLabel → {"t", "(x_t+y_t+δ_t)-∑b_t Λ_t^γ+ω_t"},
     PlotLabel → Style["Goods Market Equilibrium", 9], AxesOrigin → {0, 0}, LabelStyle → 9]
   }, Spacings → {0, 15}
  ];
GraphicsGrid[{
  {
```

```
    ListLinePlot[CapitalMarketEqb42, PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
     AxesLabel → {"t", "ω_{t-1}-(A_t(x_t+y_t)+δ_t)"}, PlotLabel → "Capital Market Equilibrium", AxesOrigin → {0, 0}],
    ListLinePlot[ZLevel42 - (ℛ YLevel42), PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
     AxesLabel → {"t", "z_t-A_t y_t"}, PlotLabel → "Credit Market Equilibrium", AxesOrigin → {0, 0}]
   },
   {
    ListLinePlot[GoodsMarketEqb42, PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
     AxesLabel → {"t", "(x_t+y_t+δ_t)-∑b_t Λ_t^ν+ω_t"}, PlotLabel → "Goods Market Equilibrium", AxesOrigin → {0, 0}],
    ListLinePlot[AyTest42, PlotStyle → {Thick, Blue}, PlotRange → {-0.01, 0.01},
     AxesLabel → {"t", "A_t y_t-z_t"}, PlotLabel → "Testing A_t y_t≤z_t", AxesOrigin → {0, 0}]
   }
  }, Spacings → {0, 15}]
```

Out[ ]=

In[ ]:=
```
Export["./Fig2TechChange.eps", Fig2TechChange, "EPS"]
```

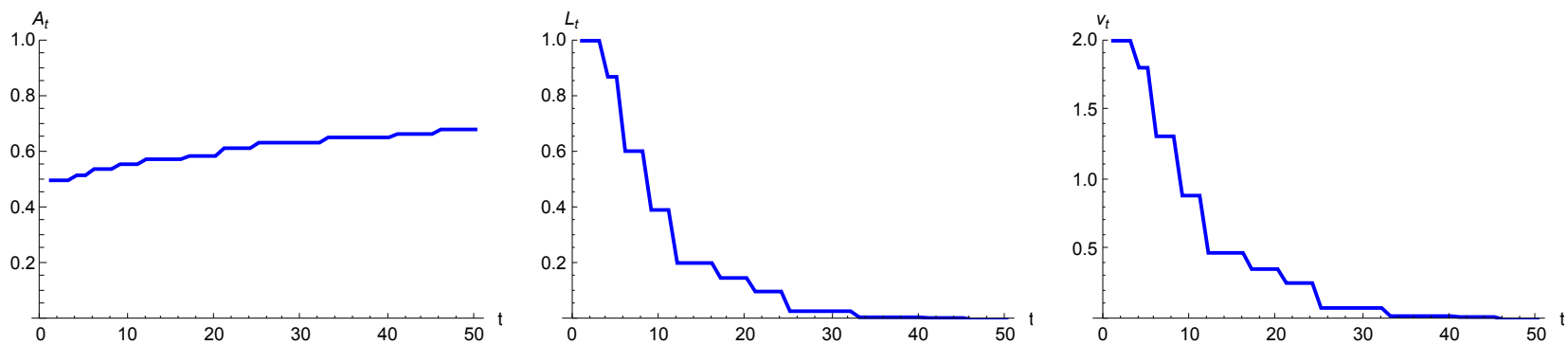
Out[ ]=

./Fig2TechChange.eps

The chart below plots out $\mathcal{A}_t$, $\mathcal{L}_t$, and the embodied labour values for the simulation.

In[ ]:=
```
Clear[PlotValues];
PlotValues = Table[v /. EmbodiedValues42 /. {a → 𝒜[[t]], ℓ → ℒ[[t]]}, {t, 1, T}];

Fig3TechChange = GraphicsRow[{
   ListLinePlot[𝒜, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "A_t"}, PlotRange → {0, 1},
    AxesOrigin → {0, 0}, LabelStyle → 10, ImagePadding → {{20, 10}, {Automatic, Automatic}}],
   ListLinePlot[ℒ, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "L_t"}, PlotRange → {0, Max[ℒ]},
    AxesOrigin → {0, 0}, LabelStyle → 10, ImagePadding → {{20, 10}, {Automatic, Automatic}}],
   ListLinePlot[PlotValues, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "v_t"}, PlotRange → {0, Max[PlotValues]},
    AxesOrigin → {0, 0}, LabelStyle → 10, ImagePadding → {{20, 10}, {Automatic, Automatic}}]
  }, Spacings → 10, ImageSize → Full]
```

Out[ ]=



In[ ]:=
```
Export["./Fig3TechChange.eps", Fig3TechChange, "EPS"]
```

Out[ ]=

./Fig3TechChange.eps

In[ ]:=
```
Column[{"A_max" → Max[𝒜], "L_min" → Min[ℒ], "v_min" → Min[PlotValues]}, Alignment → Left]
```

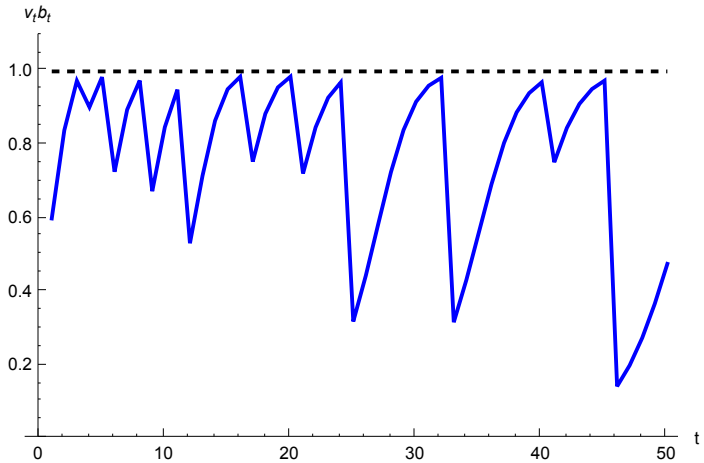
Out[ ]=

A_{max} → 0.682261
L_{min} → 0.000964239
v_{min} → 0.00303469

The chart below verifies that the $1 > v_t\, b_t$ condition is not violated over the course of the simulation.

In[ ]:=
```
ListLinePlot[{Table[(ℒ[[t]] (1 - 𝒜[[t]])^-1) Subsistence[[t]], {t, 1, T}], Table[1., {T}]},
 AxesLabel → {"t", "v_t b_t"}, PlotStyle → {{Thick, Blue}, {Thick, Dashed, Black}}, PlotRange → {0, 1.1}]
```

Out[ ]=



Exploitation and Class Over Time

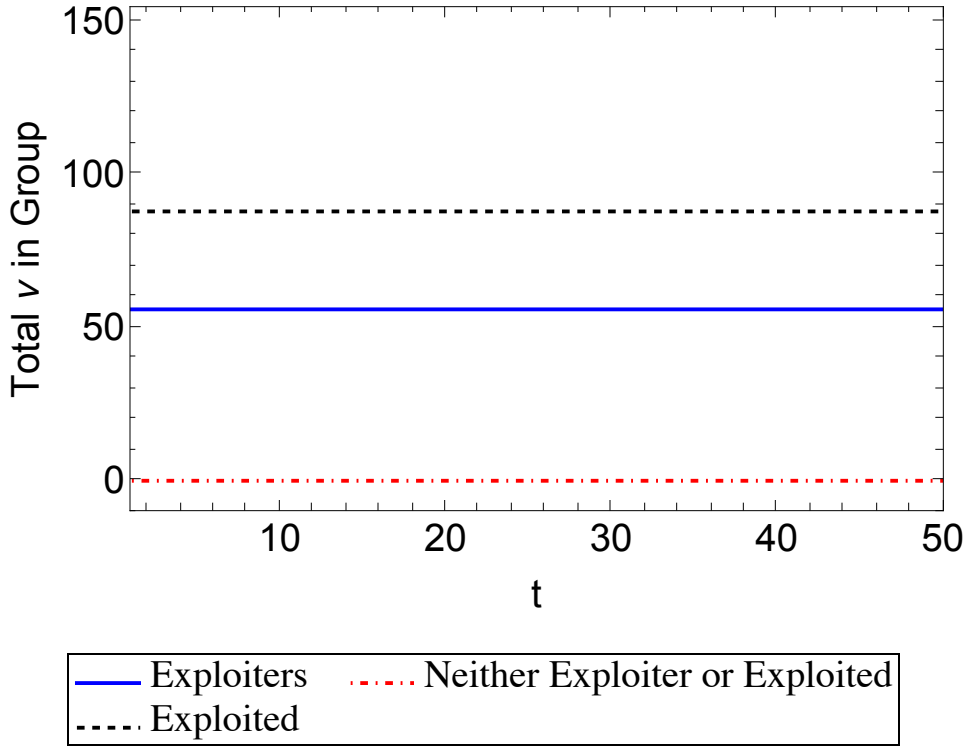
The charts below track exploitation and class over the course of the simulation as in the previous sections. The presence of increasing subsistence and technical change induces the presence of persistent exploitation.

In[ ]:=
```
Clear[IndivExploitation42, ConsumptionBundle42];
ConsumptionBundle42 = Table[0.0, {i, T}, {j, N}];
IndivExploitation42 = Table[0.0, {i, T}, {j, N}];

For[i = 1, i ≤ N, i++,
 ConsumptionBundle42[[1, i]] = (ℒ[[1]] (1 - 𝒜[[1]])^-1)
   (Profit42[[1]] × AgentSet[[i, 1]] + (Wage42[[1]] - Subsistence[[1]]) l[[i]] + Subsistence[[1]] × Activities[[1, i, 6]]);
 IndivExploitation42[[1, i]] = Activities[[1, i, 6]] - ConsumptionBundle42[[1, i]];
```

```
]

For[t = 2, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  ConsumptionBundle42[[t, i]] = (ℒ[[t]] (1 - 𝒜[[t]])^-1)
    (Profit42[[t]] × Activities[[t - 1, i, 5]] + (Wage42[[t]] - Subsistence[[t]]) l[[i]] + Subsistence[[t]] × Activities[[t, i, 6]]);
  IndivExploitation42[[t, i]] = Activities[[t, i, 6]] - ConsumptionBundle42[[t, i]];
 ]
]

Clear[Exploiters42, Exploited42, NonExploit42];
Exploiters42 = Table[0.0, {T}];
Exploited42 = Table[0.0, {T}];
NonExploit42 = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  If[IndivExploitation42[[t, i]] < 0, Exploiters42[[t]]++,
    If[IndivExploitation42[[t, i]] == 0, NonExploit42[[t]]++, Exploited42[[t]]++]];
 ]
]

ExploitationLegend =
  Grid[{{Row[{Graphics[{Thick, Blue, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["Exploiters", 20], FontFamily → "Times"]}], Spacer[10],
      Row[{Graphics[{Thick, DotDashed, Red, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["Neither Exploiter or Exploited", 20], FontFamily → "Times"]}]},
    {Row[{Graphics[{Thick, Black, Dashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
        Spacer[5], Style[Style["Exploited", 20], FontFamily → "Times"]}], Spacer[10],
    }}, Frame → True, Alignment → Left];
ExploitationPlot42 = Labeled[ListLinePlot[{Exploiters42, Exploited42, NonExploit42},
   PlotStyle → {{Thick, Blue}, {Thick, Dashed, Black}, {Thick, DotDashed, Red}},
```

```
Frame → True, FrameLabel → {"t", "Total v in Group"}, PlotRange → {{1, T}, {-10, N + 10}},
LabelStyle → 20, ImageSize → {500, 350}], ExploitationLegend]
```

Out[ ]=



In[ ]:=
```
Export["./ExploitationTechChange.eps", ExploitationPlot42, "EPS"]
```

Out[ ]=

```
./ExploitationTechChange.eps
```

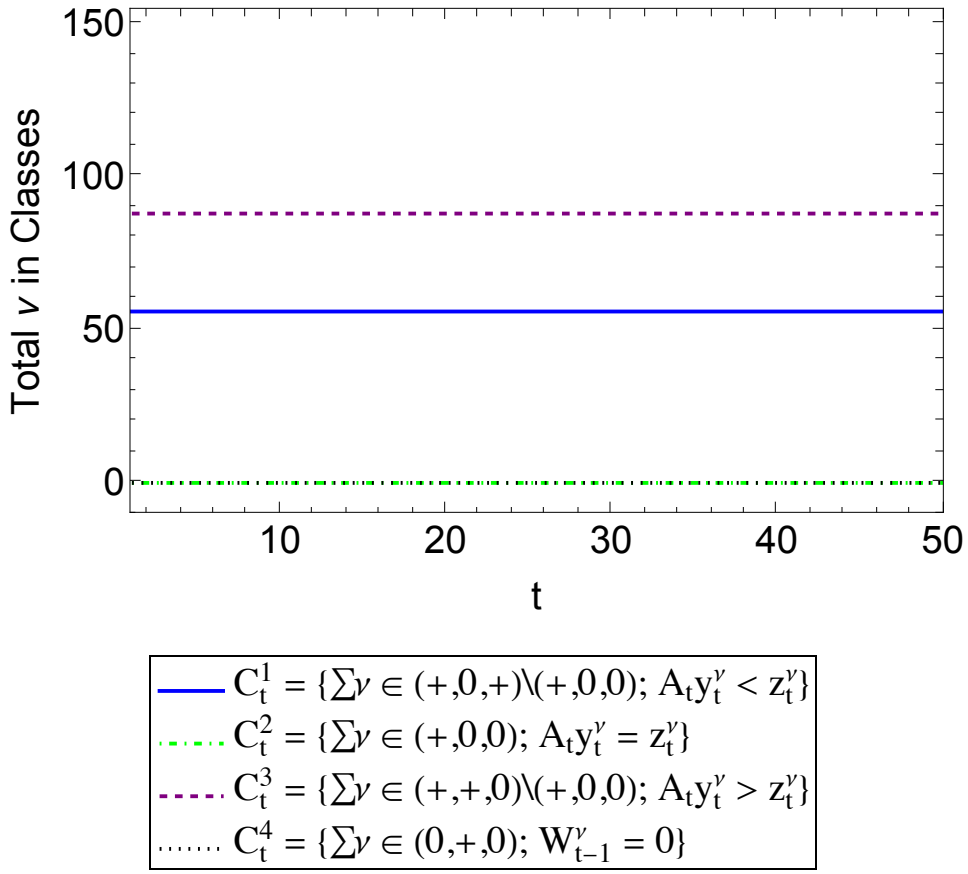
The chart below captures the composition of class over the simulation according to Corollary 1 of Theorem 3.

In[ ]:=
```
Clear[Class142, Class242, Class342, Class442, j, k];
Class142 = Table[0.0, {T}];
Class242 = Table[0.0, {T}];
```

```
Class342 = Table[0.0, {T}];
Class442 = Table[0.0, {T}];

For[j = 1, j ≤ T, j ++,
 For[k = 1, k ≤ N, k ++,
  If[𝒜[[j]] × Activities[[j, k, 2]] < Activities[[j, k, 3]], Class142[[j]] ++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j ++,
 For[k = 1, k ≤ N, k ++,
  If[𝒜[[j]] × Activities[[j, k, 2]] == Activities[[j, k, 3]], Class242[[j]] ++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j ++,
 For[k = 1, k ≤ N, k ++,
  If[𝒜[[j]] × Activities[[j, k, 2]] > Activities[[j, k, 3]], Class342[[j]] ++, 0];
 ]
]

Clear[k];
For[k = 1, k ≤ N, k ++,
 If[AgentSet[[k, 1]] == 0., Class442[[1]] ++, 0];
]

Clear[j, k];
For[j = 2, j ≤ T, j ++,
 For[k = 1, k ≤ N, k ++,
```

```
  If[Activities⟦j - 1, k, 5⟧ == 0., Class442⟦j⟧ ++, 0];
 ]
]


ClassLegend =
  Column[{
    Row[{Graphics[{Thick, Blue, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^1 = {∑ν ∈ (+,0,+)\(+,0,0); A_t y_t^ν < z_t^ν}", 20], FontFamily → "Times"]}],
    Row[{Graphics[{Thick, DotDashed, Green, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^2 = {∑ν ∈ (+,0,0); A_t y_t^ν = z_t^ν}", 20], FontFamily → "Times"]}],
    Row[{Graphics[{Thick, Purple, Dashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^3 = {∑ν ∈ (+,+,0)\(+,0,0); A_t y_t^ν > z_t^ν}", 20], FontFamily → "Times"]}],
    Row[{Graphics[{Thick, Black, Dotted, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
      Spacer[5], Style[Style["C_t^4 = {∑ν ∈ (0,+,0); W_{t-1}^ν = 0}", 20], FontFamily → "Times"]}]
   }, Frame → True, Alignment → Left];
ClassPlot42 = Labeled[
  ListLinePlot[{Class142, Class242, Class342, Class442}, PlotStyle → {{Thick, Blue}, {Thick, DotDashed, Green},
     {Thick, Dashed, Purple}, {Thick, Dotted, Black}, {Thick, DotDashed, Orange}, {Thick, Dashed, Green}},
   PlotRange → {{1, T}, {-10, N + 10}}, Frame → True, FrameLabel → {"t", "Total ν in Classes"},
   LabelStyle → 20, ImageSize → {500, 350}, AxesOrigin → {1, -10}], ClassLegend]
```

Out[ ]=



In[ ]:=
```
Export["./ClassPlotCorollary1TC.eps", ClassPlot42, "EPS"]
```

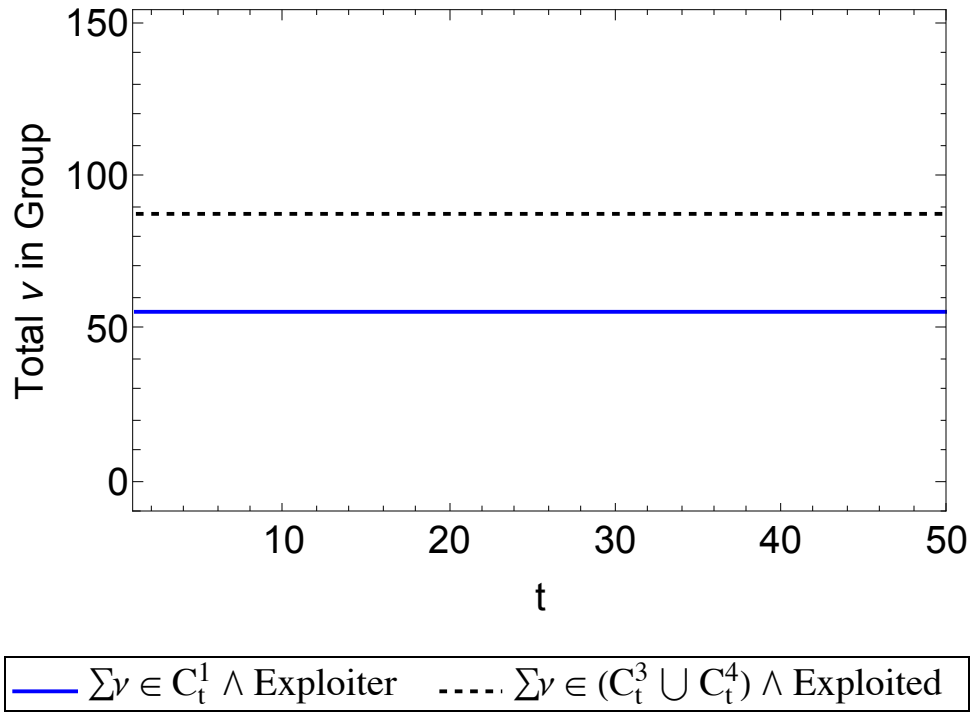
Out[ ]=
```
./ClassPlotCorollary1TC.eps
```

In[ ]:=
```
Clear[CECP142, CECP242, CECP342, j, k];
CECP142 = Table[0.0, {T}];
CECP242 = Table[0.0, {T}];
```

```
CECP3a42 = Table[0.0, {T}];
CECP3b42 = Table[0.0, {T}];
CECP342 = Table[0.0, {T}];

For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[𝒜[[j]] × Activities[[j, k, 2]] < Activities[[j, k, 3]] && IndivExploitation42[[j, k]] < 0., CECP142[[j]]++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[𝒜[[j]] × Activities[[j, k, 2]] == Activities[[j, k, 3]] && IndivExploitation42[[j, k]] == 0, CECP242[[j]]++, 0];
 ]
]

Clear[j, k];
For[j = 1, j ≤ T, j++,
 For[k = 1, k ≤ N, k++,
  If[𝒜[[j]] × Activities[[j, k, 2]] > Activities[[j, k, 3]] && IndivExploitation42[[j, k]] > 0, CECP342[[j]]++, 0];
 ]
]


CECPLegend =
  Grid[{{Row[{Graphics[{Thick, Blue, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]], Spacer[5],
       Style[Style["∑v ∈ C_t^1 ∧ Exploiter", 20], FontFamily → "Times"]}], Spacer[10],
     Row[{Graphics[{Thick, Black, Dashed, Line[{{0, 0}, {1, 0}}]}], AspectRatio → 0.15, ImageSize → Scaled[0.04]],
       Spacer[5], Style[Style["∑v ∈ (C_t^3 ⋃ C_t^4) ∧ Exploited", 20], FontFamily → "Times"]}],
    }}, Frame → True, Alignment → Left];
```

```
CECPPlot42 = Labeled[ListLinePlot[{CECP142, CECP342},
   PlotStyle → {{Thick, Blue}, {Thick, Dashed, Black}}, Frame → True, FrameLabel → {"t", "Total ν in Group"},
   PlotRange → {{1, T}, {-10, N + 10}}, LabelStyle → 20, ImageSize → {500, 350}], CECPLegend]
```

Out[ ]=



In[ ]:=
```
Export["./CECPPlotCorollary1TC.eps", CECPPlot42, "EPS"]
```

Out[ ]=

./CECPPlotCorollary1TC.eps

In[ ]:=
```
TableForm[{Exploiters42[[1]], Exploited42[[1]], Class142[[1]], Class242[[1]],
  Class342[[1]], Class442[[1]], CECP142[[1]], CECP242[[1]], CECP342[[1]]}, TableDirections → Row,
 TableHeadings → {{"Exploiters", "Exploited", "C_t^1", "C_t^2", "C_t^3", "C_t^4", "CECP^1", "CECP^2", "CECP^3"}}]
```

Out[ ]//TableForm=

Exploiters	Exploited	C_t^1	C_t^2	C_t^3	C_t^4	$CECP^1$	$CECP^2$	$CECP^3$
56.	88.	56.	0.	88.	0.	56.	0.	88.

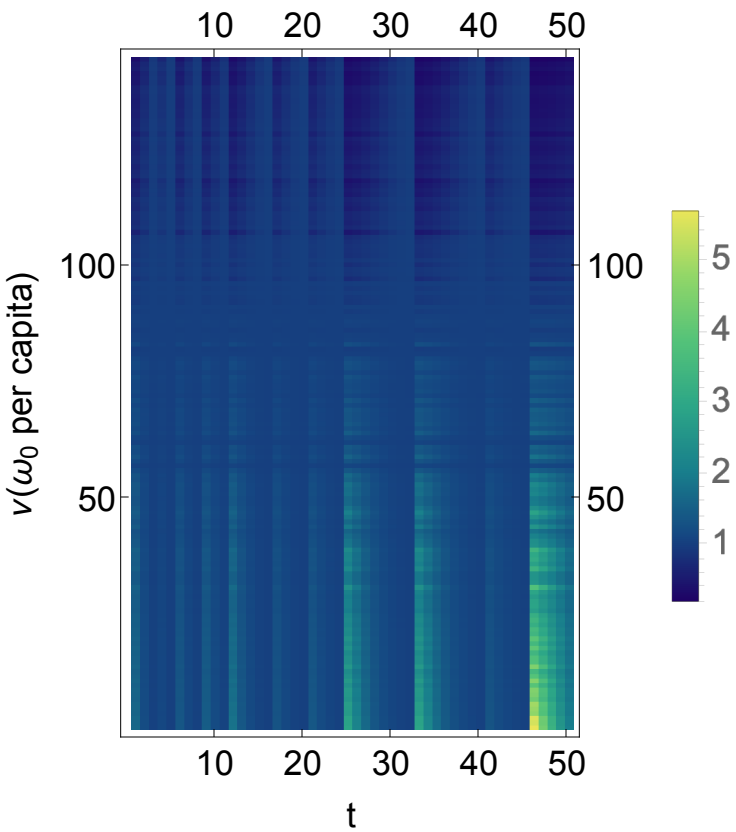
The charts below display the distribution of an index of the intensity of exploitation across the agents over the simulation. The index itself is calculated as $e_t^v = \Lambda_t^v / v_t\, c_t^v$.

In[ ]:=
```
Clear[ExploitationIndex42];
ExploitationIndex42 = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  ExploitationIndex42[[t, i]] = Activities[[t, i, 6]] / (PlotValues[[t]] (ConsumptionBundle42[[t,i]] / PlotValues[[t]]))
 ]
]
```

The chart below uses the `ArrayPlot` function to display the exploitation intensity index for all $v \in \mathcal{N}$ over T. The addition of technical change induces “cycles of exploitation” in the exploitation intensity index. As the simulation runs and accumulation progresses with a constant technique for some periods, exploitation converges across the agents. However, as soon as a new technique comes into use exploitation is restored and the pattern continues until the next new technique of production comes into use. This pattern can be seen in all the charts below.

In[ ]:=
```
ExploitationPlotTC = ArrayPlot[Transpose[ExploitationIndex42], FrameLabel → {"ν(ω₀ per capita)", "t"},
  PlotLegends → Automatic, ColorFunction → "BlueGreenYellow", ColorFunctionScaling → True,
  DataReversed → True, FrameTicks → Automatic, AspectRatio → 1.5, LabelStyle → 18]
```

Out[ ]=



In[ ]:=
```
Export["./ExploitationPlotTC.eps", ExploitationPlotTC, "EPS"]
```

Out[ ]=

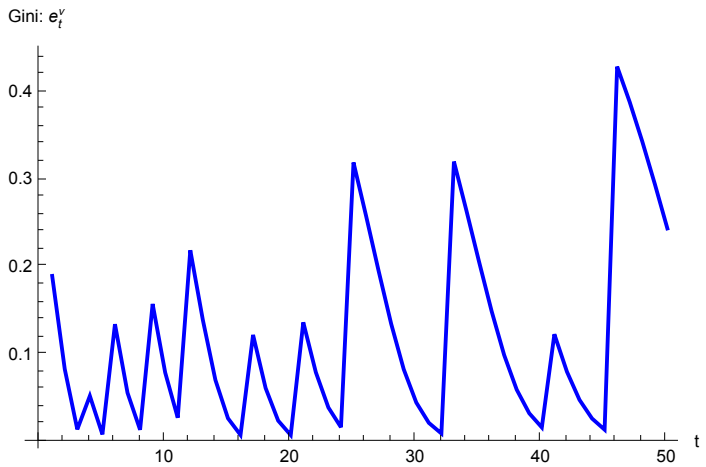
./ExploitationPlotTC.eps

In[ ]:=
```
ExploitationIndexDistrib42 = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  ExploitationIndexDistrib42[[t, i]] = {t, ExploitationIndex42[[t, i]]}
 ]
]
```

Below the Gini coefficient for the exploitation intensity index is plotted over T. The pattern of the Gini coefficient is consistent with the previous charts depicting the exploitation intensity index. The reason for the fluctuations is that as b_t (and so w_t) increases profits and exploitation decrease. When profits go below π^*, an innovation occurs that reestablishes profitability and so exploitation.

In[ ]:=
```
Clear[Gini42, SortExploitation42];
Gini42 = Table[0.0, {T}];
SortExploitation42 = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 SortExploitation42[[t]] = Sort[ExploitationIndex42[[t]]];
 Gini42[[t]] = N/(N - 1) Sum[(2 i - N - 1) SortExploitation42[[t, i]]/(N^2 Mean[SortExploitation42[[t]]]), {i, 1, N}]
]
ExploitationGiniTC = ListLinePlot[Gini42, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "Gini: e_t^v"}]
```

Out[ ]=



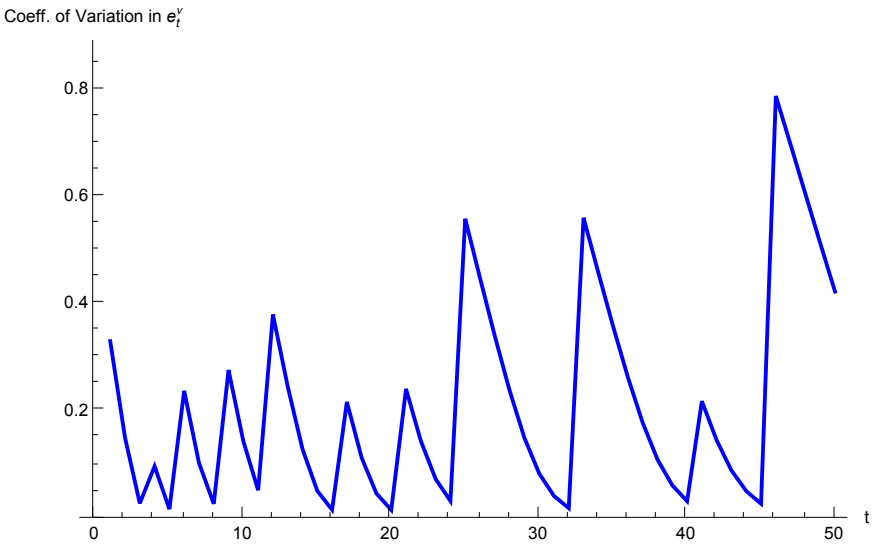
In[ ]:=
```
Export["./ExploitationGiniTC.eps", ExploitationGiniTC, "EPS"]
```

Out[ ]=
```
./ExploitationGiniTC.eps
```

The next figure plots the coefficient of variation in e_t^v over T. The figure shows a cycle in the dispersion of exploitation intensity that is driven by technical change.

In[ ]:=
```
Clear[ExpCoeffVar42];
ExpCoeffVar42 = Table[0.0, {T}];
For[i = 1, i ≤ T, i++,
 ExpCoeffVar42[[i]] = StandardDeviation[ExploitationIndex42[[i]]] / Mean[ExploitationIndex42[[i]]]
]
ExpCoeffVar42;
ListLinePlot[ExpCoeffVar42, PlotStyle → {Thick, Blue},
 AxesLabel → {"t", "Coeff. of Variation in e_t^y"}, PlotRange → {0, Max[ExpCoeffVar42] + 0.1}]
```

Out[ ]=

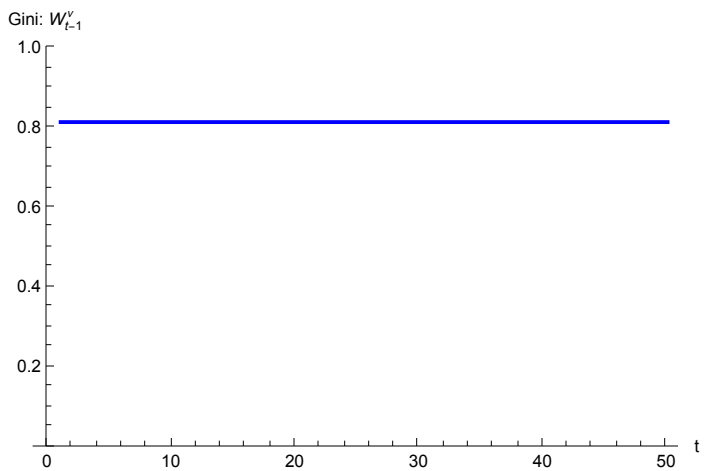


Below the Gini coefficient of wealth is plotted over the course of the simulation.

In[ ]:=
```
Clear[SortWealth42, WealthGini42];
SortWealth42 = Table[Table[0.0, {N}], {t, 1, T}];
WealthGini42 = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
 SortWealth42[[t]] = If[t == 1, Sort[AgentSet[[All, 1]]], Sort[Activities[[t - 1, All, 5]]]];
 WealthGini42[[t]] = N/(N - 1) Sum[(2 i - N - 1) SortWealth42[[t, i]]/(N^2 Mean[SortWealth42[[t]]]), {i, 1, N}]
]
WealthGiniPlot =
 ListLinePlot[WealthGini42, PlotStyle → {Thick, Blue}, AxesLabel → {"t", "Gini: W^v_{t-1}"}, PlotRange → {0, 1}]
```

$$\text{WealthGini42}[[t]] = \frac{N}{N-1} \sum_{i=1}^{N} \frac{(2\,i - N - 1)\ \text{SortWealth42}[[t, i]]}{N^2\ \text{Mean}[\text{SortWealth42}[[t]]]}$$

Out[ ]=



In[ ]:=
```
Export["./WealthGiniTC.eps", WealthGiniPlot, "EPS"]
```

Out[ ]=

./WealthGiniTC.eps

In[ ]:=
```
WealthGini42
```

Out[ ]=
```
{0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604,
 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604, 0.815604}
```

Income Figures

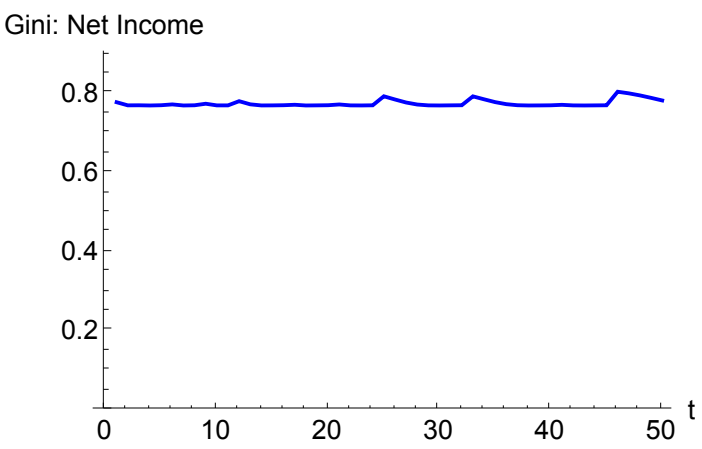
Figures on net and gross income.

In[ ]:=
```
Clear[Income];
Income = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  Income⟦t, i⟧ = Profit42⟦t⟧ × Activities⟦t, i, 5⟧ + Wage42⟦t⟧ × Activities⟦t, i, 6⟧
 ]
]

IncomeGini = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
```

$$\text{IncomeGini}[[t]] = \frac{N}{N-1}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\text{Abs}[\text{Income}[[t,i]] - \text{Income}[[t,j]]]\right) \Big/ \left(2\,N^2\,\text{Mean}[\text{Income}[[t]]]\right)\right)$$

```
]
IncomeGiniPlot = ListLinePlot[IncomeGini, PlotStyle → {Thick, Blue},
  PlotRange → {0, Max[IncomeGini] + 0.1}, AxesLabel → {"t", "Gini: Net Income"}, LabelStyle → 14]
```

Out[ ]=

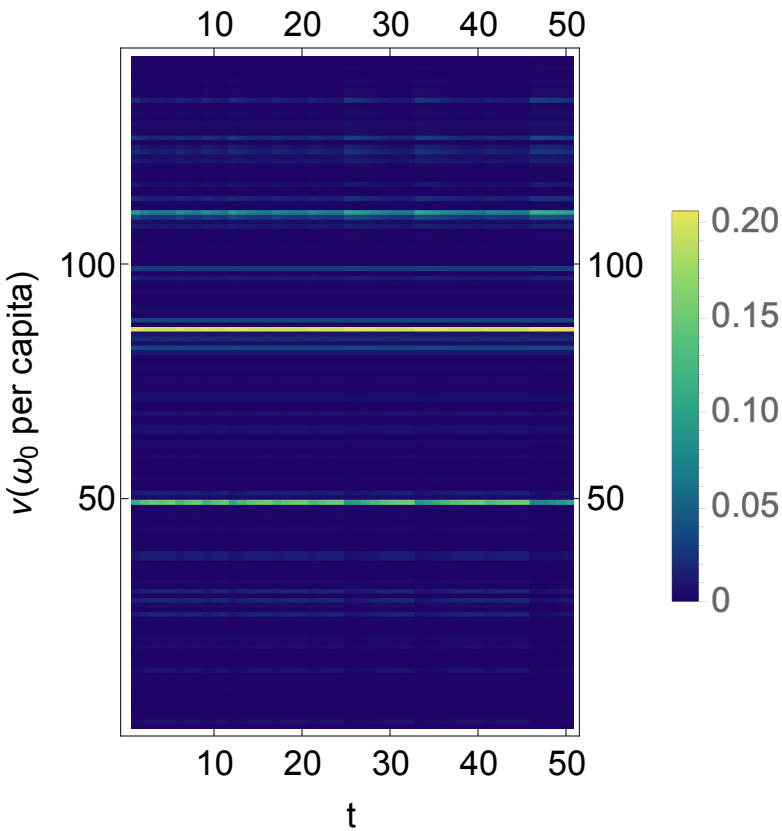


In[ ]:=
```
Export["./NetIncomeGiniTC.eps", IncomeGiniPlot, "EPS"]
```

Out[ ]=
```
./NetIncomeGiniTC.eps
```

In[ ]:=
```
Clear[IncomeShares];
IncomeShares = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  IncomeShares[[t, i]] = Income[[t, i]] / Total[Income[[t]]]
 ]
]
IncomeArray = ArrayPlot[Transpose[IncomeShares], FrameLabel → {"ν(ω0 per capita)", "t"},
  PlotLegends → Automatic, ColorFunction → "BlueGreenYellow", ColorFunctionScaling → True,
  DataReversed → True, FrameTicks → Automatic, AspectRatio → 1.5, LabelStyle → 18]
```

Out[ ]=



In[ ]:=
```
Export["./NetIncomeArrayTC.eps", IncomeArray, "EPS"]
```

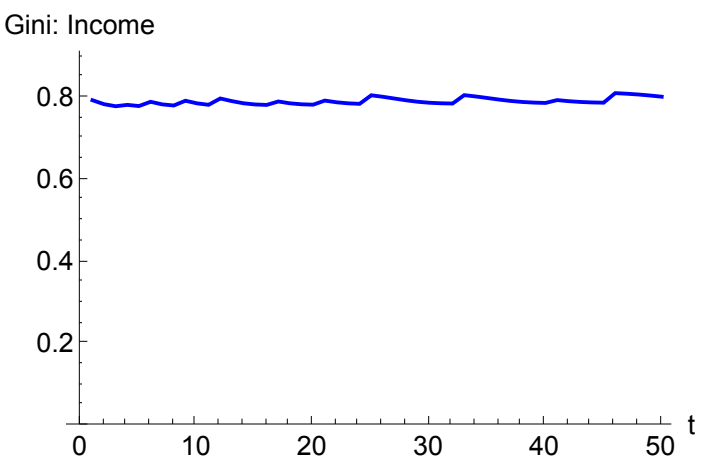
Out[ ]=
```
./NetIncomeArrayTC.eps
```

In[ ]:=
```
Clear[GrossIncome];
GrossIncome = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  GrossIncome⟦t, i⟧ = (1 + Profit42⟦t⟧) Activities⟦t, i, 5⟧ + Wage42⟦t⟧ × Activities⟦t, i, 6⟧
 ]
]

GrossIncomeGini = Table[0.0, {T}];
For[t = 1, t ≤ T, t++,
```

$$\text{GrossIncomeGini}[[t]] = \frac{N}{N-1}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\text{Abs}[\text{GrossIncome}[[t, i]] - \text{GrossIncome}[[t, j]]]\right) \Big/ \left(2\,N^2\,\text{Mean}[\text{GrossIncome}[[t]]]\right)\right)$$

```
]
GrossIncomeGiniPlot = ListLinePlot[GrossIncomeGini, PlotStyle → {Thick, Blue},
  PlotRange → {0, Max[GrossIncomeGini] + 0.1}, AxesLabel → {"t", "Gini: Income"}, LabelStyle → 14]
```

Out[ ]=

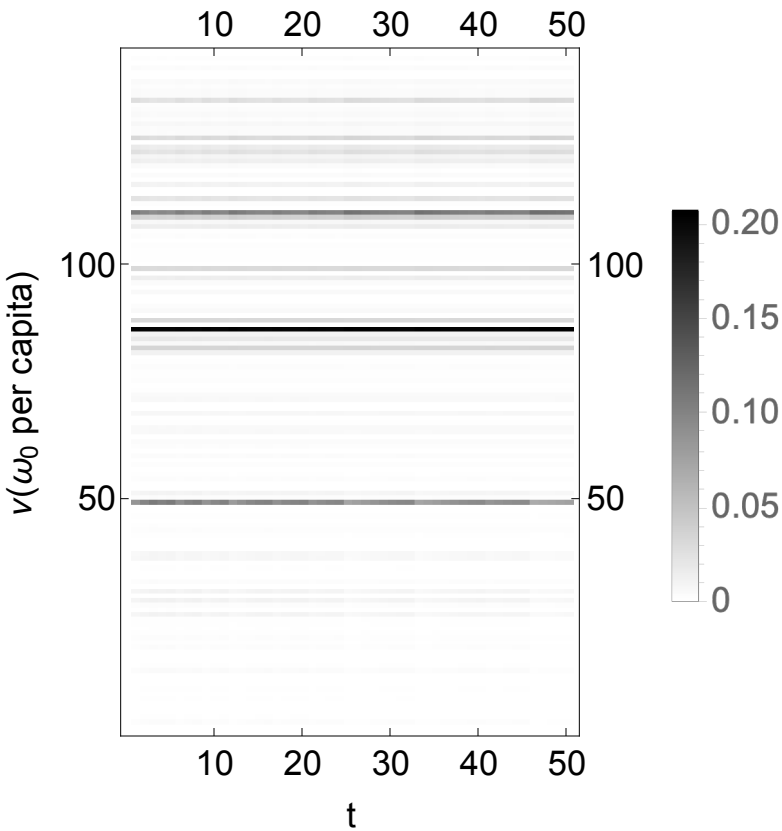


In[ ]:=
```
Export["./GrossIncomeGiniTC.eps", GrossIncomeGiniPlot, "EPS"]
```
Out[ ]=
```
./GrossIncomeGiniTC.eps
```

In[ ]:=
```
Min[GrossIncomeGini]
Max[GrossIncomeGini]
```
Out[ ]=
```
0.780087
```
Out[ ]=
```
0.812067
```

In[ ]:=
```
Clear[GrossIncomeShares];
GrossIncomeShares = Table[Table[0.0, {N}], {T}];
For[t = 1, t ≤ T, t++,
 For[i = 1, i ≤ N, i++,
  GrossIncomeShares[[t, i]] = GrossIncome[[t, i]] / Total[GrossIncome[[t]]]
 ]
]
GrossIncomeArray =
 ArrayPlot[Transpose[GrossIncomeShares], FrameLabel → {"ν(ω0 per capita)", "t"}, PlotLegends → Automatic,
  ColorFunctionScaling → True, DataReversed → True, FrameTicks → Automatic, AspectRatio → 1.5, LabelStyle → 18]
```

Out[ ]=



In[ ]:=
```
Export["./GrossIncomeArrayTC.eps", GrossIncomeArray, "EPS"]
```

Out[ ]=
```
./GrossIncomeArrayTC.eps
```

Updated Results Presentations

In[ ]:=
```
TimeSlices = {1, 10, 25, 40, 50};
```

In[ ]:=
```
(ExploitIncomeCSVexploit =
   DeleteCases[Table[If[ExploitationIndex42[[1, i]] > 1.0, {CountryList[[i]], "&", ExploitationIndex42[[1, i]],
       "&", ExploitationIndex42[[10, i]], "&", ExploitationIndex42[[25, i]], "&", ExploitationIndex42[[40, i]],
       "&", ExploitationIndex42[[50, i]], "\\\\"},], {i, 1, N}], Null]) // TableForm
Export["./ExploitIncomeTableTCexploit.csv", ExploitIncomeCSVexploit, "CSV"]
```

Out[ ]//TableForm=

Burundi	&	1.62537	&	1.17074	&	2.89308	&	1.0293	&	2.01431	\\
Congo - Kinshasa	&	1.61626	&	1.16894	&	2.84457	&	1.02902	&	1.99607	\\
Malawi	&	1.6199	&	1.16966	&	2.86384	&	1.02913	&	2.00335	\\
Mali	&	1.58243	&	1.16213	&	2.67351	&	1.02799	&	1.9294	\\
Sierra Leone	&	1.58735	&	1.16314	&	2.69753	&	1.02814	&	1.93899	\\
Liberia	&	1.59286	&	1.16426	&	2.72477	&	1.02831	&	1.94978	\\
Mozambique	&	1.55384	&	1.15621	&	2.53923	&	1.02708	&	1.87436	\\
Central African Republic	&	1.56698	&	1.15895	&	2.59986	&	1.0275	&	1.89952	\\
Madagascar	&	1.57139	&	1.15986	&	2.6206	&	1.02764	&	1.90801	\\
Niger	&	1.53038	&	1.15123	&	2.43551	&	1.02631	&	1.83008	\\
Rwanda	&	1.57342	&	1.16028	&	2.63023	&	1.02771	&	1.91193	\\
Burkina Faso	&	1.52846	&	1.15082	&	2.42724	&	1.02625	&	1.82649	\\
Ethiopia	&	1.50038	&	1.14471	&	2.31045	&	1.02529	&	1.77454	\\
Zimbabwe	&	1.56922	&	1.15942	&	2.61035	&	1.02757	&	1.90382	\\
Togo	&	1.52021	&	1.14904	&	2.39217	&	1.02597	&	1.81111	\\
Benin	&	1.52162	&	1.14935	&	2.39813	&	1.02602	&	1.81374	\\
Gambia	&	1.4932	&	1.14312	&	2.28174	&	1.02505	&	1.76144	\\
Kenya	&	1.53937	&	1.15315	&	2.47462	&	1.02661	&	1.84697	\\
Yemen	&	1.49963	&	1.14454	&	2.30742	&	1.02527	&	1.77317	\\
Uganda	&	1.53425	&	1.15206	&	2.45223	&	1.02644	&	1.83733	\\
Nepal	&	1.49334	&	1.14315	&	2.28227	&	1.02505	&	1.76169	\\
Cambodia	&	1.50258	&	1.14519	&	2.31935	&	1.02537	&	1.77857	\\
Ivory Coast	&	1.4712	&	1.13818	&	2.19638	&	1.02427	&	1.72169	\\
Cameroon	&	1.48808	&	1.14198	&	2.26148	&	1.02487	&	1.75212	\\
Pakistan	&	1.47011	&	1.13793	&	2.19223	&	1.02423	&	1.71973	\\
Senegal	&	1.41778	&	1.12574	&	2.00459	&	1.02228	&	1.62775	\\
Myanmar	&	1.43757	&	1.13042	&	2.07323	&	1.02304	&	1.66213	\\
Nigeria	&	1.43552	&	1.12994	&	2.06598	&	1.02296	&	1.65854	\\
Mauritania	&	1.41544	&	1.12518	&	1.99666	&	1.02219	&	1.62372	\\

Bangladesh	&	1.44134	&	1.13131	&	2.08661	&	1.02318	&	1.66874	\\
Kyrgyzstan	&	1.52408	&	1.14988	&	2.40856	&	1.0261	&	1.81832	\\
Tanzania	&	1.39213	&	1.11953	&	1.91956	&	1.02128	&	1.58391	\\
Haiti	&	1.38536	&	1.11787	&	1.89782	&	1.02101	&	1.57247	\\
Lesotho	&	1.3885	&	1.11864	&	1.90787	&	1.02114	&	1.57777	\\
Bolivia	&	1.48442	&	1.14116	&	2.24718	&	1.02474	&	1.74549	\\
Honduras	&	1.43062	&	1.12879	&	2.04883	&	1.02277	&	1.65001	\\
Vietnam	&	1.46041	&	1.13572	&	2.15591	&	1.02388	&	1.70242	\\
Egypt	&	1.44971	&	1.13325	&	2.11671	&	1.02349	&	1.68347	\\
Belize	&	1.48999	&	1.1424	&	2.269	&	1.02493	&	1.75559	\\
Nicaragua	&	1.37767	&	1.11596	&	1.87343	&	1.0207	&	1.55952	\\
El Salvador	&	1.3679	&	1.11352	&	1.843	&	1.0203	&	1.5432	\\
Guatemala	&	1.32265	&	1.10189	&	1.70901	&	1.01838	&	1.46899	\\
Sudan	&	1.26715	&	1.08686	&	1.55899	&	1.01584	&	1.38108	\\
Syria	&	1.39429	&	1.12006	&	1.92656	&	1.02137	&	1.58757	\\
Laos	&	1.30558	&	1.09736	&	1.66129	&	1.01762	&	1.4416	\\
Zambia	&	1.38382	&	1.11749	&	1.89291	&	1.02095	&	1.56987	\\
Moldova	&	1.43585	&	1.13002	&	2.06715	&	1.02297	&	1.65913	\\
Fiji	&	1.36694	&	1.11328	&	1.84002	&	1.02026	&	1.54159	\\
India	&	1.30003	&	1.09587	&	1.64608	&	1.01737	&	1.43275	\\
Iraq	&	1.29168	&	1.09361	&	1.62348	&	1.01699	&	1.41952	\\
Philippines	&	1.3384	&	1.106	&	1.75438	&	1.01906	&	1.49456	\\
Paraguay	&	1.32761	&	1.10319	&	1.72316	&	1.01859	&	1.47701	\\
Armenia	&	1.35084	&	1.1092	&	1.79115	&	1.01959	&	1.51494	\\
Ghana	&	1.27164	&	1.0881	&	1.57058	&	1.01606	&	1.38806	\\
Jordan	&	1.30397	&	1.09692	&	1.65686	&	1.01755	&	1.43903	\\
Congo – Brazzaville	&	1.17809	&	1.06078	&	1.34607	&	1.01131	&	1.24665	\\
Angola	&	1.04566	&	1.01683	&	1.08023	&	1.00324	&	1.06062	\\
Eswatini	&	1.15767	&	1.05443	&	1.30145	&	1.01018	&	1.21691	\\
Peru	&	1.26243	&	1.08554	&	1.54687	&	1.01562	&	1.37374	\\
Costa Rica	&	1.17031	&	1.05838	&	1.32889	&	1.01088	&	1.23526	\\
Sri Lanka	&	1.19538	&	1.06604	&	1.38498	&	1.01224	&	1.27212	\\
Morocco	&	1.03688	&	1.01367	&	1.06438	&	1.00264	&	1.04882	\\
Namibia	&	1.08955	&	1.03216	&	1.1625	&	1.00612	&	1.12054	\\
Ukraine	&	1.22398	&	1.07453	&	1.45183	&	1.01372	&	1.31491	\\
Colombia	&	1.1361	&	1.04757	&	1.25586	&	1.00894	&	1.18593	\\

```
Tajikistan                  &    1.19568    &    1.06614    &    1.38568    &    1.01225    &    1.27257    \\
Gabon                       &    1.16401    &    1.05642    &    1.31514    &    1.01053    &    1.22609    \\
South Africa                &    1.16125    &    1.05556    &    1.30916    &    1.01038    &    1.22209    \\
Mongolia                    &    1.18218    &    1.06204    &    1.35516    &    1.01153    &    1.25264    \\
Maldives                    &    1.09138    &    1.03278    &    1.16604    &    1.00623    &    1.12307    \\
Argentina                   &    1.17143    &    1.05873    &    1.33135    &    1.01094    &    1.2369     \\
Algeria                     &    1.06517    &    1.02374    &    1.11614    &    1.00455    &    1.08704    \\
Dominican Republic          &    1.12206    &    1.04301    &    1.227      &    1.00811    &    1.166      \\
Jamaica                     &    1.09606    &    1.03436    &    1.17517    &    1.00653    &    1.12957    \\
Ecuador                     &    1.10854    &    1.03854    &    1.19979    &    1.0073     &    1.14698    \\
Bulgaria                    &    1.14899    &    1.05169    &    1.28293    &    1.00969    &    1.2044     \\
Tunisia                     &    1.06162    &    1.0225     &    1.10952    &    1.00431    &    1.08221    \\
Kazakhstan                  &    1.12552    &    1.04414    &    1.23407    &    1.00832    &    1.17091    \\
Serbia                      &    1.13767    &    1.04807    &    1.25913    &    1.00903    &    1.18818    \\
Albania                     &    1.08267    &    1.02981    &    1.14925    &    1.00568    &    1.11103    \\
Iran                        &    1.0007     &    1.00026    &    1.00119    &    1.00005    &    1.00091    \\
Poland                      &    1.12024    &    1.04241    &    1.22331    &    1.008      &    1.16344    \\
Mexico                      &    1.02233    &    1.00835    &    1.03858    &    1.00162    &    1.02943    \\
Thailand                    &    1.01167    &    1.00439    &    1.02001    &    1.00085    &    1.01533    \\
Barbados                    &    1.01769    &    1.00663    &    1.03046    &    1.00129    &    1.02328    \\
Brazil                      &    1.03206    &    1.01191    &    1.05577    &    1.0023     &    1.04238    \\
Panama                      &    1.00832    &    1.00314    &    1.01423    &    1.00061    &    1.01091    \\
Chile                       &    1.00589    &    1.00222    &    1.01005    &    1.00043    &    1.00772    \\
```

Out[ ]=

```
./ExploitIncomeTableTCexploit.csv
```

In[ ]:=

```
(ExploitIncomeCSVexploiter =
   DeleteCases[Table[If[ExploitationIndex42[[1, i]] ≤ 1.0, {CountryList[[i]], "&", ExploitationIndex42[[1, i]],
       "&", ExploitationIndex42[[10, i]], "&", ExploitationIndex42[[25, i]], "&", ExploitationIndex42[[40, i]],
       "&", ExploitationIndex42[[50, i]], "\\\\"},], {i, 1, N}], Null]) // TableForm
Export["./ExploitIncomeTableTCexploiter.csv", ExploitIncomeCSVexploiter, "CSV"]
```

Out[ ]//TableForm=

```
Indonesia                &    0.976542    &    0.990977    &    0.96075     &    0.998226    &    0.96952     \\
China                    &    0.978647    &    0.991798    &    0.964221    &    0.998389    &    0.972237    \\
Venezuela                &    0.971159    &    0.988868    &    0.951922    &    0.997808    &    0.962587    \\
```

```
Mauritius            &   0.932541   &   0.973312   &   0.890451   &   0.994677   &   0.913514   \\
Uruguay              &   0.942566   &   0.977425   &   0.906101   &   0.995513   &   0.926142   \\
Malaysia             &   0.978704   &   0.991819   &   0.964314   &   0.998393   &   0.97231    \\
Botswana             &   0.94131    &   0.976912   &   0.904128   &   0.995409   &   0.924556   \\
Romania              &   0.979623   &   0.992177   &   0.965832   &   0.998464   &   0.973497   \\
Turkey               &   0.826967   &   0.926517   &   0.737545   &   0.984758   &   0.785026   \\
Lithuania            &   0.909691   &   0.963735   &   0.855553   &   0.99271    &   0.885013   \\
Russia               &   0.921171   &   0.968582   &   0.872954   &   0.993709   &   0.899283   \\
Malta                &   0.848798   &   0.936749   &   0.767486   &   0.986994   &   0.810939   \\
Slovakia             &   0.925343   &   0.970326   &   0.879343   &   0.994067   &   0.904493   \\
New Zealand          &   0.8602     &   0.941971   &   0.783455   &   0.98812    &   0.824605   \\
Croatia              &   0.874384   &   0.948357   &   0.803648   &   0.989484   &   0.841736   \\
Israel               &   0.882995   &   0.952175   &   0.816089   &   0.990293   &   0.852208   \\
Estonia              &   0.858585   &   0.941237   &   0.781179   &   0.987962   &   0.822665   \\
Hungary              &   0.817711   &   0.922084   &   0.725095   &   0.983777   &   0.774139   \\
Kuwait               &   0.58765    &   0.789904   &   0.45592    &   0.950653   &   0.521281   \\
South Korea          &   0.780708   &   0.903774   &   0.676724   &   0.979644   &   0.731199   \\
Taiwan               &   0.726544   &   0.875146   &   0.609717   &   0.972911   &   0.669976   \\
Japan                &   0.754777   &   0.890352   &   0.644103   &   0.97653    &   0.701651   \\
United States        &   0.755286   &   0.890621   &   0.644734   &   0.976593   &   0.702227   \\
Trinidad and Tobago  &   0.662704   &   0.838276   &   0.53602    &   0.963715   &   0.600197   \\
Finland              &   0.699124   &   0.859751   &   0.577397   &   0.969146   &   0.639697   \\
United Kingdom       &   0.731849   &   0.878052   &   0.616091   &   0.973611   &   0.675889   \\
Cyprus               &   0.61782    &   0.810059   &   0.48732    &   0.956242   &   0.552611   \\
Latvia               &   0.649139   &   0.82996    &   0.521043   &   0.961553   &   0.58569    \\
Saudi Arabia         &   0.586098   &   0.78884    &   0.454333   &   0.950352   &   0.519684   \\
Bahrain              &   0.504631   &   0.728814   &   0.374605   &   0.932299   &   0.437686   \\
Czech Republic       &   0.69253    &   0.855951   &   0.569776   &   0.968201   &   0.632484   \\
Slovenia             &   0.671417   &   0.843525   &   0.545763   &   0.965062   &   0.609573   \\
Greece               &   0.617075   &   0.809574   &   0.486533   &   0.95611    &   0.551831   \\
Canada               &   0.686239   &   0.852291   &   0.56256    &   0.967284   &   0.625629   \\
Australia            &   0.663397   &   0.838696   &   0.536792   &   0.963823   &   0.600941   \\
France               &   0.62024    &   0.811633   &   0.489885   &   0.956669   &   0.555146   \\
Spain                &   0.586865   &   0.789366   &   0.455117   &   0.950501   &   0.520473   \\
Iceland              &   0.618539   &   0.810527   &   0.488081   &   0.956369   &   0.553363   \\
Germany              &   0.662105   &   0.837913   &   0.535354   &   0.963621   &   0.599554   \\
```

```
Portugal              &   0.512663   &   0.735119   &   0.382164   &   0.934299   &   0.445611   \\
Sweden                &   0.632173   &   0.819304   &   0.502628   &   0.958734   &   0.567698   \\
Netherlands           &   0.607596   &   0.803341   &   0.476563   &   0.954403   &   0.541935   \\
Denmark               &   0.603207   &   0.800422   &   0.471982   &   0.953597   &   0.537371   \\
Belgium               &   0.54963    &   0.763012   &   0.417788   &   0.942849   &   0.482531   \\
Hong Kong             &   0.55816    &   0.769198   &   0.426209   &   0.944681   &   0.491155   \\
Ireland               &   0.541166   &   0.756784   &   0.409508   &   0.940981   &   0.474012   \\
Italy                 &   0.530328   &   0.748673   &   0.399014   &   0.938514   &   0.463161   \\
Austria               &   0.544306   &   0.759105   &   0.412571   &   0.94168    &   0.477168   \\
Switzerland           &   0.570712   &   0.778138   &   0.43874    &   0.947289   &   0.503919   \\
Norway                &   0.563698   &   0.773166   &   0.431717   &   0.945844   &   0.496776   \\
United Arab Emirates  &   0.443854   &   0.677991   &   0.319391   &   0.915172   &   0.378807   \\
Macao                 &   0.450912   &   0.684192   &   0.325629   &   0.917363   &   0.385547   \\
Brunei                &   0.417326   &   0.653923   &   0.296338   &   0.906384   &   0.353693   \\
Singapore             &   0.517068   &   0.738539   &   0.386337   &   0.935374   &   0.449972   \\
Luxembourg            &   0.465936   &   0.69712    &   0.339056   &   0.921836   &   0.399979   \\
Qatar                 &   0.335883   &   0.571603   &   0.229218   &   0.872398   &   0.278729   \\
```

Out[ ]=

```
./ExploitIncomeTableTCexploiter.csv
```

In[ ]:=

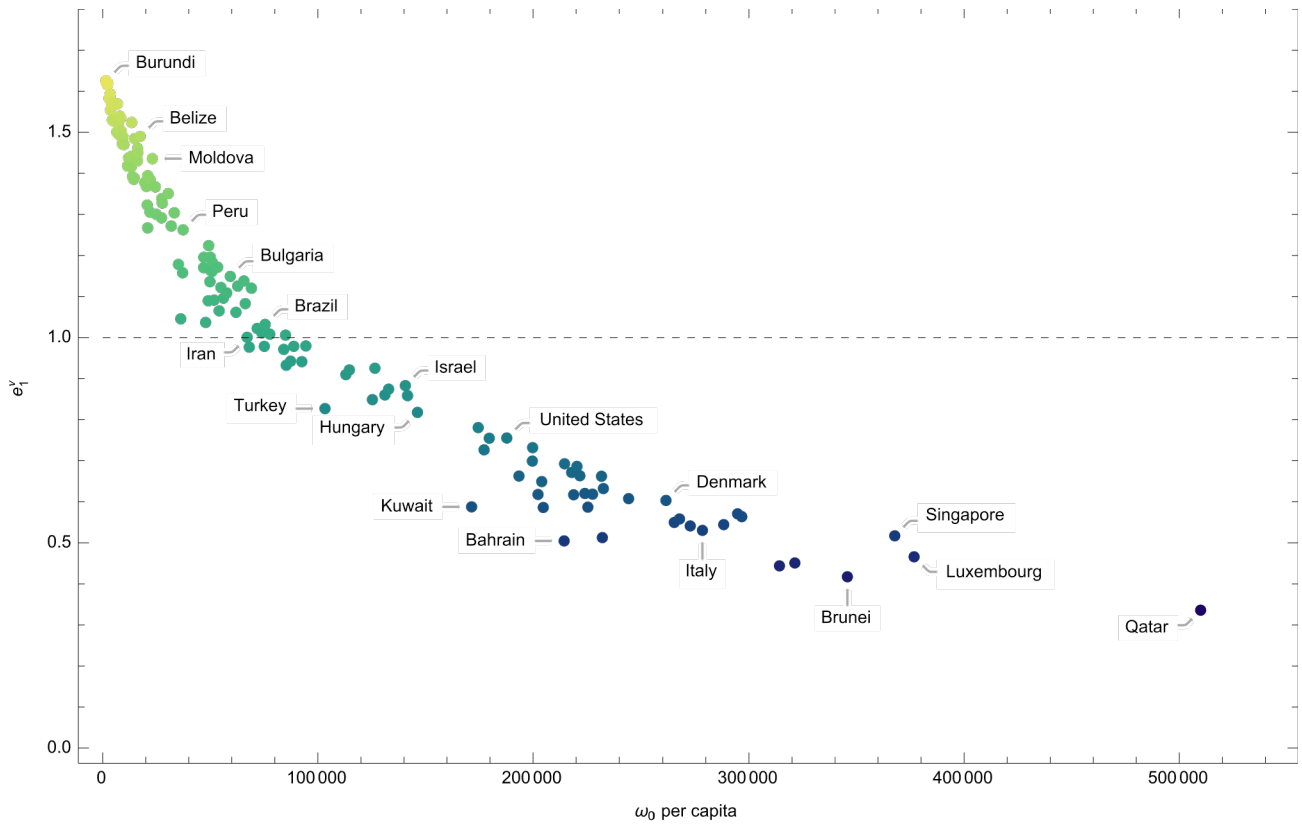
```
(ExploitIncomeCSV =
    Table[{CountryList⟦i⟧, "&", ExploitationIndex42⟦1, i⟧, "&", ExploitationIndex42⟦10, i⟧, "&", ExploitationIndex42⟦
       25, i⟧, "&", ExploitationIndex42⟦40, i⟧, "&", ExploitationIndex42⟦50, i⟧, "\\\\"}, {i, 1, N}]) // TableForm;
(* Export["./ExploitIncomeTableTC.csv",ExploitIncomeCSV,"CSV"] *)
```

In[ ]:=

```
CountryLabels = {"United States", "China", "India", "South Africa",
   "Brazil", "Russia", "United Kingdom", "Germany", "Sweden", "Norway", "France", "Italy"};
```

In[ ]:=
```
ExploitWealthPlot = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[1, i]]}, {i, 1, N}] → CountryList,
  LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e1^γ"},
  ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600000, 1}}]}]
Export["./ExploitWealthPlotTC.eps", ExploitWealthPlot, "EPS"]
```

Out[ ]=



Out[ ]=

./ExploitWealthPlotTC.eps

In[ ]:=
```
ExploitWealthPlot1 = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[1, i]]}, {i, 1, N}] → CountryList,
   LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e_t^v"},
   ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitWealthPlot10 = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[10, i]]}, {i, 1, N}] → CountryList,
   LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e_t^v"},
   ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitWealthPlot25 = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[25, i]]}, {i, 1, N}] → CountryList,
   LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e_t^v"},
   ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitWealthPlot40 = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[40, i]]}, {i, 1, N}] → CountryList,
   LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e_t^v"},
   ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitWealthPlot50 = ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[50, i]]}, {i, 1, N}] → CountryList,
   LabelingFunction → Callout[Automatic, Automatic], Frame → True, FrameLabel → {"ω0 per capita", "e_t^v"},
   ColorFunction → "BlueGreenYellow", Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
Export["./ExploitWealthPlotTC1.eps", ExploitWealthPlot1, "EPS"]
Export["./ExploitWealthPlotTC10.eps", ExploitWealthPlot10, "EPS"]
Export["./ExploitWealthPlotTC25.eps", ExploitWealthPlot25, "EPS"]
Export["./ExploitWealthPlotTC40.eps", ExploitWealthPlot40, "EPS"]
Export["./ExploitWealthPlotTC50.eps", ExploitWealthPlot50, "EPS"]
```

Out[ ]=
```
./ExploitWealthPlotTC1.eps
```

Out[ ]=
```
./ExploitWealthPlotTC10.eps
```

Out[ ]=
```
./ExploitWealthPlotTC25.eps
```

Out[ ]=
```
./ExploitWealthPlotTC40.eps
```

Out[ ]=

```
./ExploitWealthPlotTC50.eps
```

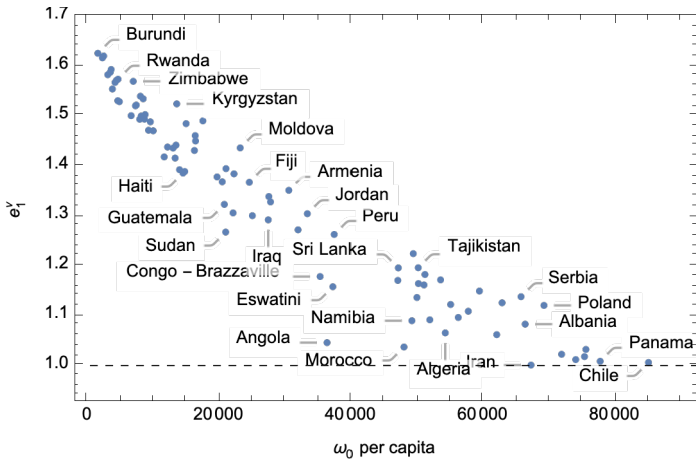
In[ ]:=

```
(* ExploitWealthPlotTable=Table[ListPlot[Table[{DataNoHeader[[i,5]],ExploitationIndex42[[j,i]]},{i,1,N}]→CountryList,
   LabelingFunction→Callout[Automatic,Automatic],AxesLabel→{"Ω0 per capita","e_t^v"},
   ColorFunction→"BlueGreenYellow",PlotLabel→StringJoin["t=",ToString[j]],PlotRange→All,
   ImageSize→Large,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}],{j,TimeSlices}];
ExploitWealthPlotTC=GraphicsGrid[{
    {ExploitWealthPlotTable[[1]],ExploitWealthPlotTable[[2]]},
    {ExploitWealthPlotTable[[3]],ExploitWealthPlotTable[[4]]},
    {ExploitWealthPlotTable[[5]],}
   },ImageSize→Large
  ]
  Export["./ExploitWealthPlotTCgrid.eps",ExploitWealthPlotTC,"EPS"] *)
```

```
In[•]:= (* ExploitWealthPlotTC2=GraphicsGrid[{
    {ListPlot[Table[{DataNoHeader[[i,5]],ExploitationIndex42[[1,i]]},{i,1,N}]→CountryList,LabelingFunction→
       Callout[Automatic,Automatic,LabelStyle→6],PlotLabel→"t=1",AxesLabel→{"Ω0 per capita","e_t^v"},
      ColorFunction→"BlueGreenYellow",LabelStyle→8,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}],
     ListPlot[Table[{DataNoHeader[[i,5]],ExploitationIndex42[[10,i]]},{i,1,N}]→CountryList,LabelingFunction→
       Callout[Automatic,Automatic,LabelStyle→6],PlotLabel→"t=10",AxesLabel→{"Ω0 per capita","e_t^v"},
      ColorFunction→"BlueGreenYellow",LabelStyle→8,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}]},
    {ListPlot[Table[{DataNoHeader[[i,5]],ExploitationIndex42[[25,i]]},{i,1,N}]→CountryList,LabelingFunction→
       Callout[Automatic,Automatic,LabelStyle→6],PlotLabel→"t=25",AxesLabel→{"Ω0 per capita","e_t^v"},
      ColorFunction→"BlueGreenYellow",LabelStyle→8,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}],
     ListPlot[Table[{DataNoHeader[[i,5]],ExploitationIndex42[[40,i]]},{i,1,N}]→CountryList,LabelingFunction→
       Callout[Automatic,Automatic,LabelStyle→6],PlotLabel→"t=40",AxesLabel→{"Ω0 per capita","e_t^v"},
      ColorFunction→"BlueGreenYellow",LabelStyle→8,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}]},
    {ListPlot[Table[{DataNoHeader[[i,5]],ExploitationIndex42[[50,i]]},{i,1,N}]→CountryList,LabelingFunction→
       Callout[Automatic,Automatic,LabelStyle→6],PlotLabel→"t=50",AxesLabel→{"Ω0 per capita","e_t^v"},
      ColorFunction→"BlueGreenYellow",LabelStyle→8,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}],}
   },Spacings→{-3,Automatic}]
  Export["./ExploitWealthPlotTCgrid2.eps",ExploitWealthPlotTC2,"EPS"] *)

In[•]:= ExploitedCountries =
  Table[DeleteCases[Table[If[ExploitationIndex42[[j, i]] > 1, i, 0.], {i, 1, N}], 0.], {j, TimeSlices}];
ExploiterCountries =
  Table[DeleteCases[Table[If[ExploitationIndex42[[j, i]] < 1, i, 0.], {i, 1, N}], 0.], {j, TimeSlices}];
```

In[ ]:=
```
ExploitedCountriesPlot =
 ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[1, i]]}, {i, ExploitedCountries[[1]]}] →
   CountryList[[ExploitedCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
  FrameLabel → {"ω0 per capita", "e1ʸ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600000, 1}}]}]
Export["./ExploitedCountriesPlotTC.eps", ExploitedCountriesPlot, "EPS"]
```

Out[ ]=



Out[ ]=

./ExploitedCountriesPlotTC.eps

In[ ]:=
```
ExploitedCountriesPlot1 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[1, i]]}, {i, ExploitedCountries[[1]]}] →
    CountryList[[ExploitedCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitedCountriesPlot10 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[10, i]]}, {i, ExploitedCountries[[2]]}] →
    CountryList[[ExploitedCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitedCountriesPlot25 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[25, i]]}, {i, ExploitedCountries[[3]]}] →
    CountryList[[ExploitedCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitedCountriesPlot40 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[40, i]]}, {i, ExploitedCountries[[4]]}] →
    CountryList[[ExploitedCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploitedCountriesPlot50 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[50, i]]}, {i, ExploitedCountries[[5]]}] →
    CountryList[[ExploitedCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
Export["./ExploitedCountriesPlotTC1.eps", ExploitedCountriesPlot1, "EPS"]
Export["./ExploitedCountriesPlotTC10.eps", ExploitedCountriesPlot10, "EPS"]
Export["./ExploitedCountriesPlotTC25.eps", ExploitedCountriesPlot25, "EPS"]
Export["./ExploitedCountriesPlotTC40.eps", ExploitedCountriesPlot40, "EPS"]
Export["./ExploitedCountriesPlotTC50.eps", ExploitedCountriesPlot50, "EPS"]
```

Out[ ]=
```
./ExploitedCountriesPlotTC1.eps
```

Out[ ]=
```
./ExploitedCountriesPlotTC10.eps
```

Out[ ]=

```
./ExploitedCountriesPlotTC25.eps
```

Out[ ]=

```
./ExploitedCountriesPlotTC40.eps
```

Out[ ]=

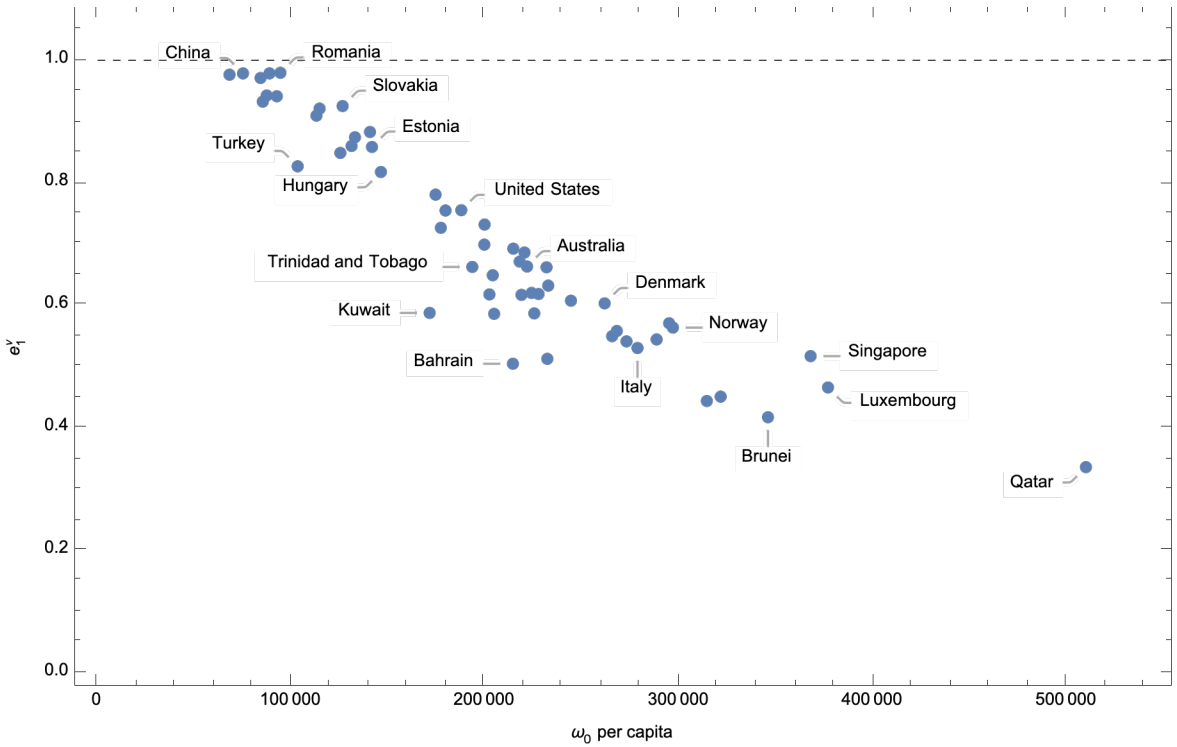
```
./ExploitedCountriesPlotTC50.eps
```

In[ ]:=

```
(* ExploitationIndexSlices=Table[ExploitationIndex42⟦j,All⟧,{j,TimeSlices}];
ExploitedCountriesPlotTC=
 Table[ListPlot[Table[{DataNoHeader⟦i,5⟧,ExploitationIndexSlices⟦j,i⟧},{i,ExploitedCountries⟦j⟧}]→
    CountryList⟦ExploitedCountries⟦j⟧⟧,LabelingFunction→Callout[Automatic,Automatic],
   AxesLabel→{"Ω0 per capita","e_t^γ"},PlotLabel→StringJoin["t=",ToString[TimeSlices⟦j⟧]],
   ImageSize→Medium,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}],{j,Length[TimeSlices]}];
ExploitedCountriesTCgrid=GraphicsGrid[{
    {ExploitedCountriesPlotTC⟦1⟧,ExploitedCountriesPlotTC⟦2⟧},
    {ExploitedCountriesPlotTC⟦3⟧,ExploitedCountriesPlotTC⟦4⟧},
    {ExploitedCountriesPlotTC⟦5⟧,}
   },Spacings→{-10,Automatic}
  ]
  Export["./ExploitedCountriesTCgrid.eps",ExploitedCountriesTCgrid,"EPS"] *)
```

In[ ]:=
```
ExploiterCountriesPlot =
 ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[1, i]]}, {i, ExploiterCountries[[1]]}] →
   CountryList[[ExploiterCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
  FrameLabel → {"ω0 per capita", "e1ʸ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}]
Export["./ExploiterCountriesPlotTC.eps", ExploiterCountriesPlot, "EPS"]
```

Out[ ]=



Out[ ]=

./ExploiterCountriesPlotTC.eps

In[ ]:=
```
ExploiterCountriesPlot1 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[1, i]]}, {i, ExploiterCountries[[1]]}] →
    CountryList[[ExploiterCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploiterCountriesPlot10 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[10, i]]}, {i, ExploiterCountries[[2]]}] →
    CountryList[[ExploiterCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploiterCountriesPlot25 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[25, i]]}, {i, ExploiterCountries[[3]]}] →
    CountryList[[ExploiterCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploiterCountriesPlot40 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[40, i]]}, {i, ExploiterCountries[[4]]}] →
    CountryList[[ExploiterCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
ExploiterCountriesPlot50 =
  ListPlot[Table[{DataNoHeader[[i, 5]], ExploitationIndex42[[50, i]]}, {i, ExploiterCountries[[5]]}] →
    CountryList[[ExploiterCountries[[1]]]], LabelingFunction → Callout[Automatic, Automatic], Frame → True,
   FrameLabel → {"ω0 per capita", "e_t^γ"}, Epilog → {Black, Dashed, Line[{{0, 1}, {600 000, 1}}]}];
Export["./ExploiterCountriesPlotTC1.eps", ExploiterCountriesPlot1, "EPS"]
Export["./ExploiterCountriesPlotTC10.eps", ExploiterCountriesPlot10, "EPS"]
Export["./ExploiterCountriesPlotTC25.eps", ExploiterCountriesPlot25, "EPS"]
Export["./ExploiterCountriesPlotTC40.eps", ExploiterCountriesPlot40, "EPS"]
Export["./ExploiterCountriesPlotTC50.eps", ExploiterCountriesPlot50, "EPS"]
```

Out[ ]=
```
./ExploiterCountriesPlotTC1.eps
```

Out[ ]=
```
./ExploiterCountriesPlotTC10.eps
```

Out[•]=

```
./ExploiterCountriesPlotTC25.eps
```

Out[•]=

```
./ExploiterCountriesPlotTC40.eps
```

Out[•]=

```
./ExploiterCountriesPlotTC50.eps
```

In[•]:=

```
(* ExploitationIndexSlices=Table[ExploitationIndex42⟦j,All⟧,{j,TimeSlices}];
ExploiterCountriesPlotTC=
 Table[ListPlot[Table[{DataNoHeader⟦i,5⟧,ExploitationIndexSlices⟦j,i⟧},{i,ExploiterCountries⟦j⟧}]→
    CountryList⟦ExploiterCountries⟦j⟧⟧,LabelingFunction→Callout[Automatic,Automatic],
   AxesLabel→{"Ω0 per capita","e_t^γ"},PlotLabel→StringJoin["t=",ToString[TimeSlices⟦j⟧]],
   ImageSize→Medium,Epilog→{Black,Dashed,Line[{{0,1},{600000,1}}]}],{j,Length[TimeSlices]}];
ExploiterCountriesTCgrid=GraphicsGrid[{
    {ExploiterCountriesPlotTC⟦1⟧,ExploiterCountriesPlotTC⟦2⟧},
    {ExploiterCountriesPlotTC⟦3⟧,ExploiterCountriesPlotTC⟦4⟧},
    {ExploiterCountriesPlotTC⟦5⟧,}
   },Spacings→{-10,Automatic}
  ]
  Export["./ExploiterCountriesTCgrid.eps",ExploiterCountriesTCgrid,"EPS"] *)
```

Map Plots of Exploitation Intensity

In[ ]:=
```
CountryList
```

Out[ ]=
```
{Burundi, Congo - Kinshasa, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic, Madagascar,
 Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia,
 Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania, Haiti,
 Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria, Laos,
 Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, Congo - Brazzaville, Angola,
 Eswatini, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa,
 Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, Trinidad and Tobago,
 Finland, United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada,
 Australia, France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland,
 Italy, Austria, Switzerland, Norway, United Arab Emirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

Converting list of countries to those in Mathematica geographical data.

In[ ]:=
```
CountryListRevise = CountryList /. {"Congo - Kinshasa" → "DemocraticRepublicCongo",
   "Congo - Brazzaville" → "RepublicCongo", "Eswatini" → "Swaziland", "Trinidad and Tobago" → "TrinidadTobago"}
```

Out[ ]=
```
{Burundi, DemocraticRepublicCongo, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic,
 Madagascar, Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal,
 Cambodia, Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania,
 Haiti, Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria,
 Laos, Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, RepublicCongo, Angola,
 Swaziland, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa,
 Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, TrinidadTobago, Finland,
 United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada, Australia,
 France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland,
 Italy, Austria, Switzerland, Norway, United Arab Emirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

Converting country names to strings compatible with Mathematica’s geographical data.

In[ ]:=
```
countriesNoSpace = StringReplace[#, " " → ""] & /@ CountryListRevise
```

Out[ ]=
```
{Burundi, DemocraticRepublicCongo, Malawi, Mali, SierraLeone, Liberia, Mozambique, CentralAfricanRepublic, Madagascar,
 Niger, Rwanda, BurkinaFaso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia,
 IvoryCoast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania,
 Haiti, Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, ElSalvador, Guatemala, Sudan, Syria,
 Laos, Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, RepublicCongo, Angola,
 Swaziland, Peru, CostaRica, SriLanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, SouthAfrica,
 Mongolia, Maldives, Argentina, Algeria, DominicanRepublic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan,
 Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile,
 Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, NewZealand,
 Croatia, Israel, Estonia, Hungary, Kuwait, SouthKorea, Taiwan, Japan, UnitedStates, TrinidadTobago, Finland,
 UnitedKingdom, Cyprus, Latvia, SaudiArabia, Bahrain, CzechRepublic, Slovenia, Greece, Canada, Australia,
 France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, HongKong, Ireland,
 Italy, Austria, Switzerland, Norway, UnitedArabEmirates, Macao, Brunei, Singapore, Luxembourg, Qatar}
```

Attaching countries in current sub-sample to geographical entities in Mathematica geographical data.

In[ ]:=
```
countryEntities = Table[Entity["Country", ToString[countriesNoSpace[[i]]]], {i, 1, Length[CountryList]}]
```

Out[ ]=

{Burundi, Democratic Republic of the Congo, Malawi, Mali, Sierra Leone, Liberia, Mozambique, Central African Republic, Madagascar, Niger, Rwanda, Burkina Faso, Ethiopia, Zimbabwe, Togo, Benin, Gambia, Kenya, Yemen, Uganda, Nepal, Cambodia, Ivory Coast, Cameroon, Pakistan, Senegal, Myanmar, Nigeria, Mauritania, Bangladesh, Kyrgyzstan, Tanzania, Haiti, Lesotho, Bolivia, Honduras, Vietnam, Egypt, Belize, Nicaragua, El Salvador, Guatemala, Sudan, Syria, Laos, Zambia, Moldova, Fiji, India, Iraq, Philippines, Paraguay, Armenia, Ghana, Jordan, Republic of the Congo, Angola, Eswatini, Peru, Costa Rica, Sri Lanka, Morocco, Namibia, Ukraine, Colombia, Tajikistan, Gabon, South Africa, Mongolia, Maldives, Argentina, Algeria, Dominican Republic, Jamaica, Ecuador, Bulgaria, Tunisia, Kazakhstan, Serbia, Albania, Iran, Indonesia, Poland, Mexico, Thailand, China, Barbados, Brazil, Panama, Venezuela, Chile, Mauritius, Uruguay, Malaysia, Botswana, Romania, Turkey, Lithuania, Russia, Malta, Slovakia, New Zealand, Croatia, Israel, Estonia, Hungary, Kuwait, South Korea, Taiwan, Japan, United States, Trinidad and Tobago, Finland, United Kingdom, Cyprus, Latvia, Saudi Arabia, Bahrain, Czech Republic, Slovenia, Greece, Canada, Australia, France, Spain, Iceland, Germany, Portugal, Sweden, Netherlands, Denmark, Belgium, Hong Kong, Ireland, Italy, Austria, Switzerland, Norway, United Arab Emirates, Macau, Brunei, Singapore, Luxembourg, Qatar}

Threading country entities to exploitation intensity index.

In[ ]:=
```
Thread[countryEntities → ExploitationIndex42[[1]]]
```

Out[ ]=

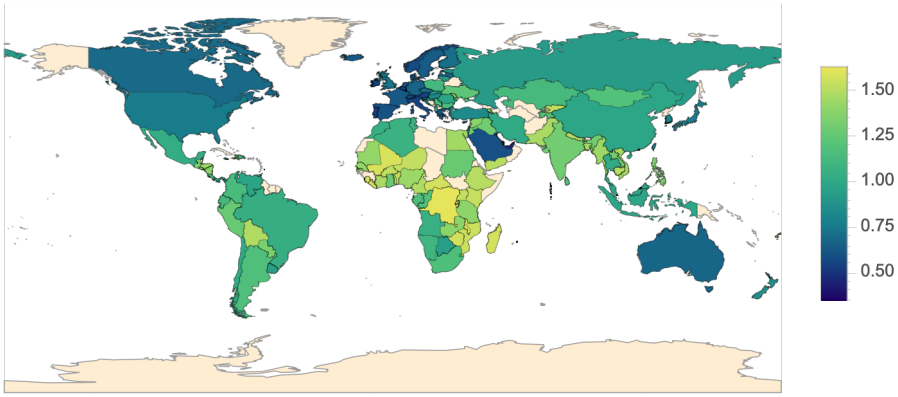
{Burundi → 1.62537, Democratic Republic of the Congo → 1.61626, Malawi → 1.6199, Mali → 1.58243, Sierra Leone → 1.58735, Liberia → 1.59286, Mozambique → 1.55384, Central African Republic → 1.56698,

Madagascar → 1.57139, Niger → 1.53038, Rwanda → 1.57342, Burkina Faso → 1.52846, Ethiopia → 1.50038,
Zimbabwe → 1.56922, Togo → 1.52021, Benin → 1.52162, Gambia → 1.4932, Kenya → 1.53937, Yemen → 1.49963,
Uganda → 1.53425, Nepal → 1.49334, Cambodia → 1.50258, Ivory Coast → 1.4712, Cameroon → 1.48808,
Pakistan → 1.47011, Senegal → 1.41778, Myanmar → 1.43757, Nigeria → 1.43552, Mauritania → 1.41544,
Bangladesh → 1.44134, Kyrgyzstan → 1.52408, Tanzania → 1.39213, Haiti → 1.38536, Lesotho → 1.3885,
Bolivia → 1.48442, Honduras → 1.43062, Vietnam → 1.46041, Egypt → 1.44971, Belize → 1.48999,
Nicaragua → 1.37767, El Salvador → 1.3679, Guatemala → 1.32265, Sudan → 1.26715, Syria → 1.39429,
Laos → 1.30558, Zambia → 1.38382, Moldova → 1.43585, Fiji → 1.36694, India → 1.30003, Iraq → 1.29168,
Philippines → 1.3384, Paraguay → 1.32761, Armenia → 1.35084, Ghana → 1.27164, Jordan → 1.30397,
Republic of the Congo → 1.17809, Angola → 1.04566, Eswatini → 1.15767, Peru → 1.26243, Costa Rica → 1.17031,
Sri Lanka → 1.19538, Morocco → 1.03688, Namibia → 1.08955, Ukraine → 1.22398, Colombia → 1.1361,
Tajikistan → 1.19568, Gabon → 1.16401, South Africa → 1.16125, Mongolia → 1.18218, Maldives → 1.09138,
Argentina → 1.17143, Algeria → 1.06517, Dominican Republic → 1.12206, Jamaica → 1.09606, Ecuador → 1.10854,
Bulgaria → 1.14899, Tunisia → 1.06162, Kazakhstan → 1.12552, Serbia → 1.13767, Albania → 1.08267,
Iran → 1.0007, Indonesia → 0.976542, Poland → 1.12024, Mexico → 1.02233, Thailand → 1.01167,
China → 0.978647, Barbados → 1.01769, Brazil → 1.03206, Panama → 1.00832, Venezuela → 0.971159,
Chile → 1.00589, Mauritius → 0.932541, Uruguay → 0.942566, Malaysia → 0.978704, Botswana → 0.94131,
Romania → 0.979623, Turkey → 0.826967, Lithuania → 0.909691, Russia → 0.921171, Malta → 0.848798,
Slovakia → 0.925343, New Zealand → 0.8602, Croatia → 0.874384, Israel → 0.882995, Estonia → 0.858585,
Hungary → 0.817711, Kuwait → 0.58765, South Korea → 0.780708, Taiwan → 0.726544, Japan → 0.754777,

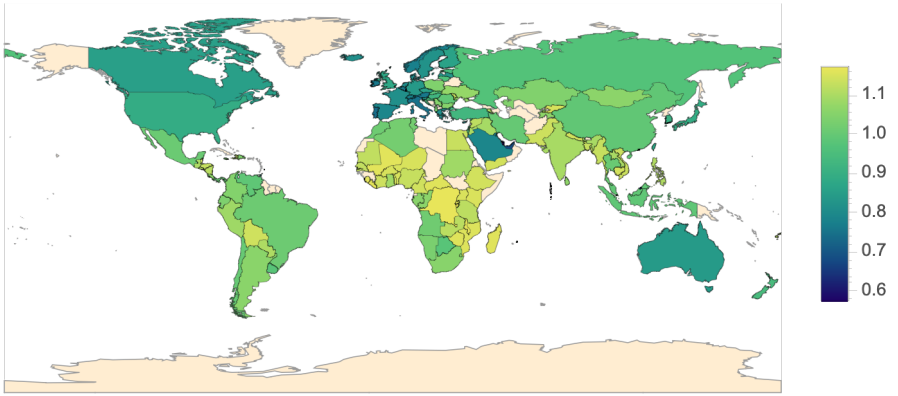
```
United States → 0.755286, Trinidad and Tobago → 0.662704, Finland → 0.699124, United Kingdom → 0.731849,
Cyprus → 0.61782, Latvia → 0.649139, Saudi Arabia → 0.586098, Bahrain → 0.504631, Czech Republic → 0.69253,
Slovenia → 0.671417, Greece → 0.617075, Canada → 0.686239, Australia → 0.663397, France → 0.62024,
Spain → 0.586865, Iceland → 0.618539, Germany → 0.662105, Portugal → 0.512663, Sweden → 0.632173,
Netherlands → 0.607596, Denmark → 0.603207, Belgium → 0.54963, Hong Kong → 0.55816, Ireland → 0.541166,
Italy → 0.530328, Austria → 0.544306, Switzerland → 0.570712, Norway → 0.563698, United Arab Emirates → 0.443854,
Macau → 0.450912, Brunei → 0.417326, Singapore → 0.517068, Luxembourg → 0.465936, Qatar → 0.335883}
```

In[ ]:=
```
mapExploitationIntensity1 = GeoRegionValuePlot[Thread[countryEntities → ExploitationIndex42[[1]]],
  ColorFunction → "BlueGreenYellow", GeoBackground → {"CountryBorders", "Ocean" → White}, GeoRange → All]
mapExploitationIntensity10 = GeoRegionValuePlot[Thread[countryEntities → ExploitationIndex42[[10]]],
  ColorFunction → "BlueGreenYellow", GeoBackground → {"CountryBorders", "Ocean" → White}, GeoRange → All]
mapExploitationIntensity25 = GeoRegionValuePlot[Thread[countryEntities → ExploitationIndex42[[25]]],
  ColorFunction → "BlueGreenYellow", GeoBackground → {"CountryBorders", "Ocean" → White}, GeoRange → All]
mapExploitationIntensity40 = GeoRegionValuePlot[Thread[countryEntities → ExploitationIndex42[[40]]],
  ColorFunction → "BlueGreenYellow", GeoBackground → {"CountryBorders", "Ocean" → White}, GeoRange → All]
mapExploitationIntensity50 = GeoRegionValuePlot[Thread[countryEntities → ExploitationIndex42[[50]]],
  ColorFunction → "BlueGreenYellow", GeoBackground → {"CountryBorders", "Ocean" → White}, GeoRange → All]
Export["./mapExploitationIntensityTC1.eps", mapExploitationIntensity1, "EPS"];
Export["./mapExploitationIntensityTC1.jpg", mapExploitationIntensity1, "JPEG"];
Export["./mapExploitationIntensityTC10.eps", mapExploitationIntensity10, "EPS"];
Export["./mapExploitationIntensityTC25.eps", mapExploitationIntensity25, "EPS"];
Export["./mapExploitationIntensityTC40.eps", mapExploitationIntensity40, "EPS"];
Export["./mapExploitationIntensityTC50.eps", mapExploitationIntensity50, "EPS"];
```

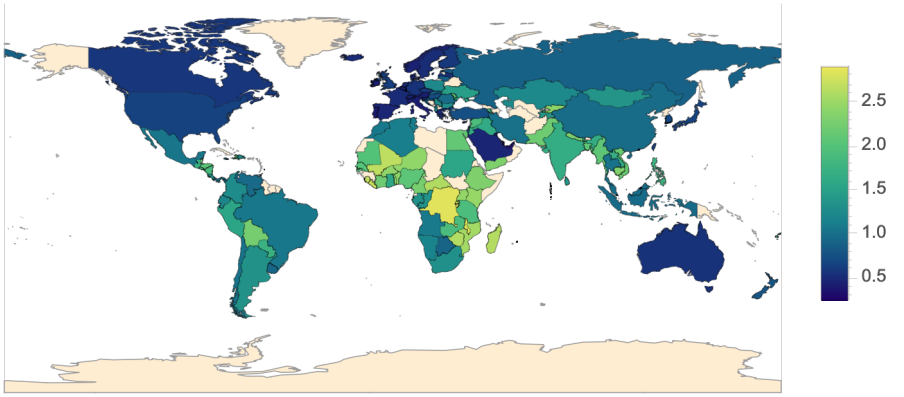
Out[ ]=



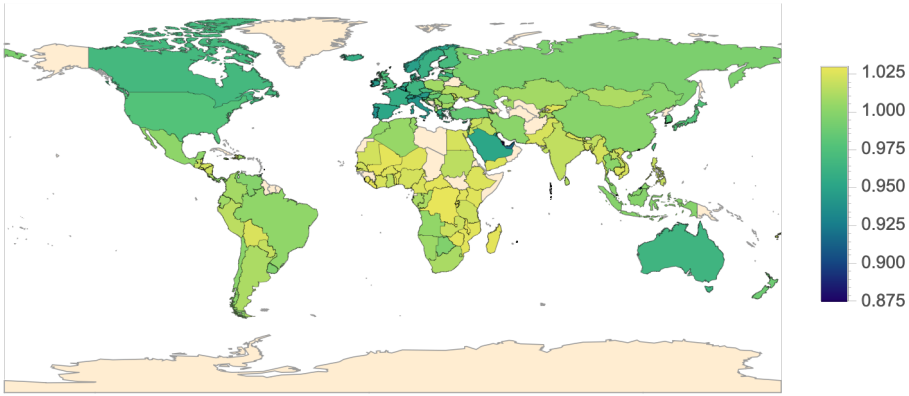
Out[ ]=



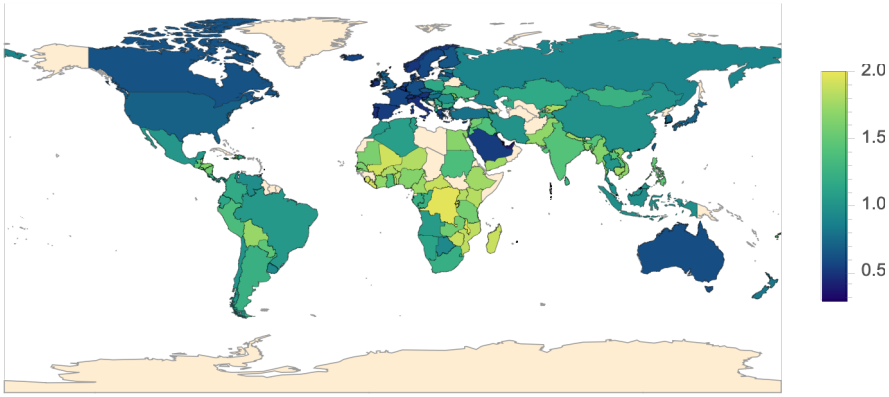
Out[ ]=



Out[ ]=



Out[ ]=

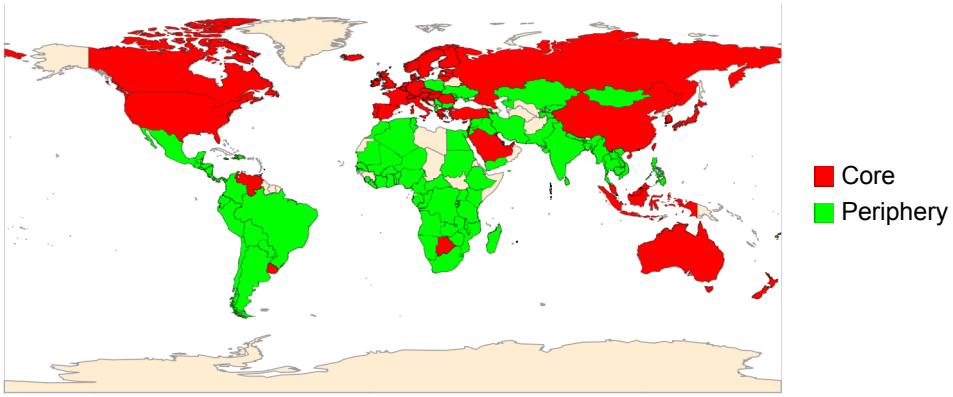


Partitioning exploitation intensity index into periphery and core groups.

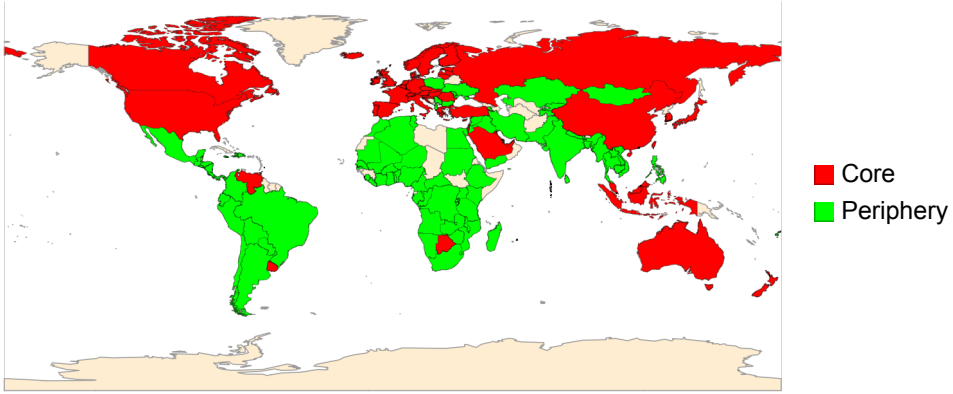
In[ ]:=
```
corePerip1 = Table[If[ExploitationIndex42[[1, i]] > 1, 1., 0.], {i, 1, Length[CountryList]}];
corePerip10 = Table[If[ExploitationIndex42[[10, i]] > 1, 1., 0.], {i, 1, Length[CountryList]}];
corePerip25 = Table[If[ExploitationIndex42[[25, i]] > 1, 1., 0.], {i, 1, Length[CountryList]}];
corePerip40 = Table[If[ExploitationIndex42[[40, i]] > 1, 1., 0.], {i, 1, Length[CountryList]}];
corePerip50 = Table[If[ExploitationIndex42[[50, i]] > 1, 1., 0.], {i, 1, Length[CountryList]}];
```

```
In[ ]:= mapCorePeripheryTC1 = GeoRegionValuePlot[Thread[countryEntities → corePerip1],
          ColorRules → {1. → Green, 0. → Red}, GeoBackground → {"CountryBorders", "Ocean" → White},
          GeoRange → All, PlotLegends → SwatchLegend[{Red, Green}, {"Core", "Periphery"}]]
        mapCorePeripheryTC10 = GeoRegionValuePlot[Thread[countryEntities → corePerip10],
          ColorRules → {1. → Green, 0. → Red}, GeoBackground → {"CountryBorders", "Ocean" → White},
          GeoRange → All, PlotLegends → SwatchLegend[{Red, Green}, {"Core", "Periphery"}]]
        mapCorePeripheryTC25 = GeoRegionValuePlot[Thread[countryEntities → corePerip25],
          ColorRules → {1. → Green, 0. → Red}, GeoBackground → {"CountryBorders", "Ocean" → White},
          GeoRange → All, PlotLegends → SwatchLegend[{Red, Green}, {"Core", "Periphery"}]]
        mapCorePeripheryTC40 = GeoRegionValuePlot[Thread[countryEntities → corePerip40],
          ColorRules → {1. → Green, 0. → Red}, GeoBackground → {"CountryBorders", "Ocean" → White},
          GeoRange → All, PlotLegends → SwatchLegend[{Red, Green}, {"Core", "Periphery"}]]
        mapCorePeripheryTC50 = GeoRegionValuePlot[Thread[countryEntities → corePerip50],
          ColorRules → {1. → Green, 0. → Red}, GeoBackground → {"CountryBorders", "Ocean" → White},
          GeoRange → All, PlotLegends → SwatchLegend[{Red, Green}, {"Core", "Periphery"}]]
        Export["./mapCorePeripheryTC1.eps", mapCorePeripheryTC1, "EPS"];
        Export["./mapCorePeripheryTC1.jpg", mapCorePeripheryTC1, "JPEG"];
        Export["./mapCorePeripheryTC10.eps", mapCorePeripheryTC10, "EPS"];
        Export["./mapCorePeripheryTC25.eps", mapCorePeripheryTC25, "EPS"];
        Export["./mapCorePeripheryTC40.eps", mapCorePeripheryTC40, "EPS"];
        Export["./mapCorePeripheryTC50.eps", mapCorePeripheryTC50, "EPS"];
```

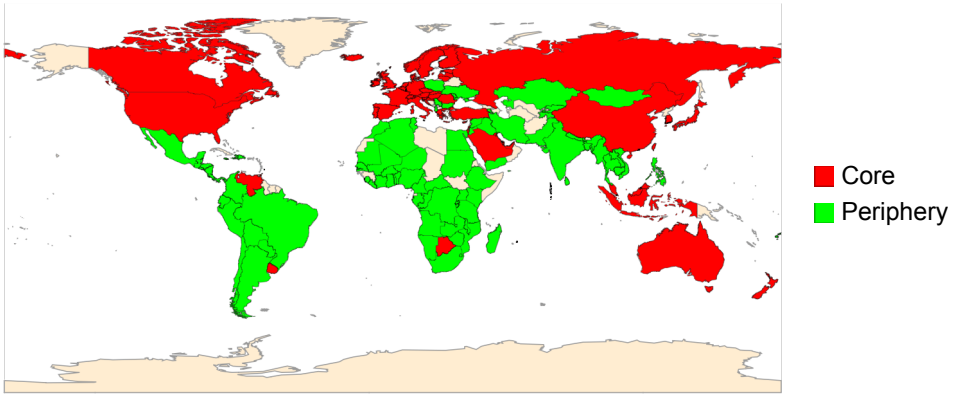
Out[ ]=



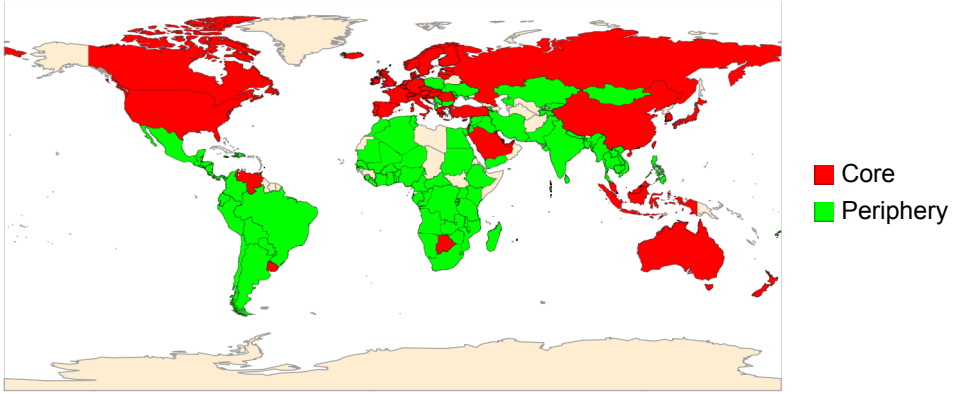
Out[ ]=



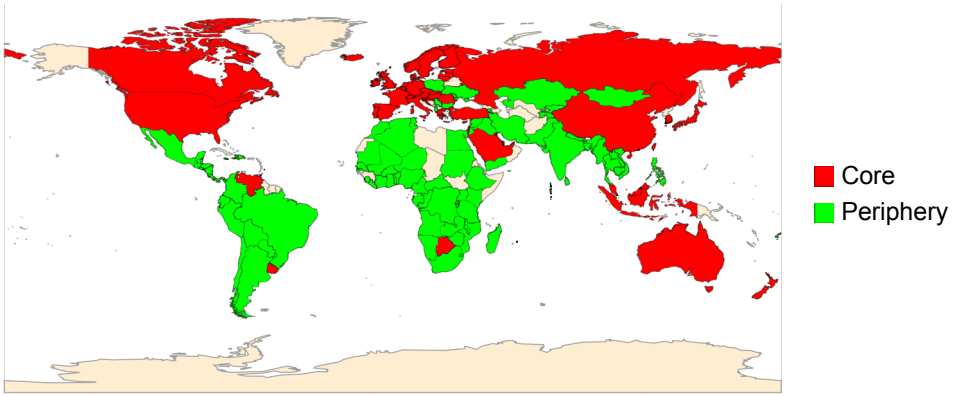
Out[ ]=



Out[ ]=



Out[ ]=



Robustness Simulations

Standard of Living Consumption and Technical Change, v1

Standard of Living Consumption and Technical Change, v2

Standard of Living Consumption and Technical Change, v3

Standard of Living Consumption and Technical Change, v4

Standard of Living Consumption and Technical Change, v5

Standard of Living Consumption and Technical Change, v6

Exogenous Technical Change and Endogenous Subsistence

Basic Model - proxies for human capital

Exogenous Technical Change and Endogenous Subsistence - proxies for human capital

Endogenous Consumption and Technical Change - proxies for human capital

Basic Model - persons engaged

Exogenous Technical Change and Endogenous Subsistence - persons engaged

Endogenous Consumption and Technical Change - persons engaged