SFM

August 21, 2017

1 Specific Factors Model

1.0.1 Background

The SF model is a workhorse model in trade, growth, political economy and development. We will see variants of the model used to describe rural to urban migration, the Lewis model and other dual sector models of sectoral misallocation, models such as the Harris-Todaro model that explain migration and the urban informal sector. The specific factors model predicts that, in the absence of political redistribution mechanisms, specific factors in declining sectors will organize strong opposition to policies that might otherwise raise growth. The initial distribution of factor endowments may therefore make a big difference in terms of what types of political coalitions mobilize for and against different policies. This is the basic driving force in Moav-Galor's (2006) growth model on why some regions made public investments in human capital which sped the transition from agriculture to manufacturing and enhanced growth, whereas similar policies were delayed in other regions where political/economic resistance was stronger, for instance where landlords had stronger voice in political decisions.

These are but just a few of the applications. The great thing about this model is that it is relatively easy to analyze -- and it can be described very nicely in terms of diagrams -- and yet it is so rich in predictions.

The Specific Factors (SF) or Ricardo-Viner model is a close relative of the Hecksher-Ohlin-Samuelson (HOS) neoclassical trade model. The 2x2 HOS model assumes production in each of two sectors takes place by firms using constant returns to scale technologies with capital and labor as inputs and that both capital and labor are mobile across sectors. In the SF model only labor is inter-sectorally mobile and the fixed amounts of capital become 'specific' to the sector they are trapped within.

In effect the SF model therefore consists of three factors of production: mobile labor and two types of capital, one specific to each sector. Let's label the two sectors are Agriculture and Manufacturing. In agriculture competitive firms bid to hire land and labor. In manufacturing competitive firms bid to hire capital and labor. Each factor of production will be competitively priced in equilibrium but only labor is priced on a national labor market.

The SF model is often described as a short-run version of the HOS model. For example suppose we start with a HOS model equilibrium where labor wage and rental rate of capital have equalized across sectors (which also implies marginal products are equalized across sectors -- no productivity differences across sectors). Now suppose that the relative price of manufacturing products suddenly rises (due to a change of world price, or government trade protection or other policies that favor manufacturing). The higher relative product price should lead firms in the booming manufacturing sector to demand both more capital and labor. Correspondingly, demand for land and labor decline in the agricultural sector. In the short run however only labor can be

move from agriculture to manufacturing. Agricultural workers can become factory workers but agricultural capital (say 'land' or 'tractors') cannot be easily converted to manufacturing capital (say 'weaving machines'). So labor moves from manufacturing to the agriculture, lowering the capital labor ratio in manufacturing and raising the land to labor ratio in agriculture. The model thus predicts a surge in the real return to capital in the expanding sector and a real decline in the real return to capital in agriculture (land). Hence the measured average and marginal product of capital will now diverge across sectors. What happens to the real wage is more ambiguous: it rises measured in terms of purchasing power over agricultural goods but falls in terms of purchasing power over manufacturing goods. Whether workers are better off or worse off following this price/policy change therefore comes down to how important agricultural and manufacturing goods are in their consumption basket. This result is labeled the neo-classical ambiguity.

Over the longer-term weaving machines cannot be transformed into tractors but over time new capital accumulation will build tractors and old weaving machines can be sold overseas or as scrap metal. Hence over time more capital will arrive into the manufacturing sector and leave the agricultural sector, whcih in turn will lead to even more movement of labor to manufacturing. If this process continues capital has, in effect become mobile over time, and we end up getting closer to the predictions of the HOS model.

1.0.2 Technology and Endowments

There are two sectors Agriculture and Manufacturing. Production in each sector takes place with a linear homogenous (constant returns to scale) production functions. Agricultural production requires land which is non-mobile or specific to the sector and mobile labor. Manufacturing production requires specific capital and mobile labor.

$$Q_a = F(\bar{T}_a, L_a)$$
$$Q_m = G(\bar{K}_m, L_m)$$

The market for mobile labor is competitive and the market clears at a wage where the sum of labor demands from each sector equals inelastically supplied total labor supply.

$$L_a + L_m = \bar{L}$$

For the numeric simulations below we make the further assumption that each sector uses a CRS Cobb-Douglas production function:

$$F(T_a, L_a) = \bar{T}_a^{(1-\alpha)} \cdot L_a^{\alpha}$$
$$G(K_m, L_m) = \bar{K}_m^{(1-\beta)} \cdot L_m^{\beta}$$

Visualization via Bokeh plots The Bokeh visualization library provides an object-oriented interface and dynamic interactive plotting elements.

```
In [48]: import numpy as np
         import bokeh.plotting as bk
         from bokeh.layouts import row, column
         from bokeh.io import push_notebook
         from bokeh.models import Range1d
         from ipywidgets import interact
```

```
from IPython.display import Latex

output_notebook()
TOOLS = "resize,reset,save,box_select"
```

Model parameters

```
In [5]: Tbar = 100        # Fixed specific land in ag.
        Kbar = 100        # Fixed specific capital in manuf
        Lbar = 400        # Total number of mobile workers
        LbarMax = 400     # Lbar will be on slider, max value.

        p    = 1.00       # initial rel price of ag goods, p = Pa/Pm
        alpha, beta = 0.5, 0.5  # labor share in ag, manuf
```

Firms in each sector hires labor so long as the marginal value product of labor is greater than or equal to the market wage:

$$P_a \cdot MPL_a = w$$

$$P_m \cdot MPL_m = w$$

Dividing each expression above by P_m yields

$$p \cdot MPL_a = \frac{w}{p_m}$$

$$MPL_m = \frac{w}{p_m}$$

$$\text{where } p = \frac{P_a}{P_m}$$

1.1 Static Plot version

First we plot a standard specific factors diagram showing the equilibrium wage and labor allocations. Without a slider.

```
In [55]: La = np.linspace(0.1, LbarMax-0.1,LbarMax)
         Lm = Lbar - La
         Qa = (Tbar**(1-alpha) * La**alpha) * (La<Lbar)
         Qm = Kbar**(1-beta) * (Lbar - La)**beta
         pMPLa = (p * alpha * Qa/La) *(La<Lbar)          # for Cobb-Douglass MPL can be written
         MPLm = beta * Qm/(Lbar-La)
```

Find the equilibrium wage w and labor allocation L_A that clears the market.

$$p \cdot MPL_a(L_A) = MPL_m(\bar{L} - L_A)$$

the following function finds La where closest to that intersection (L_A, w_{eq}).

```
In [56]: def eqn(pMPLa, MPLm):
             diff = (pMPLa-MPLm)*(La<Lbar)+(La>Lbar)*2
             LAidx = np.nanargmin(diff**2)  #value of LA where pMPLa(La) = MPLm(Lbar-La) is at
             return La[LAidx], pMPLa[LAidx]
```

Simple example with $\bar{L}$ at $\bar{L}^{max}$

```
In [57]: def foo(Lbar=LbarMax):
             La = np.arange(0, Lbar)
             Lm = Lbar - La
             Qa = (Tbar**(1-alpha) * La**alpha) * (La<Lbar)
             Qm = Kbar**(1-beta) * (Lbar - La)**beta
             pMPLa = (p * alpha * Qa/La) *(La<Lbar)          # for Cobb-Douglass MPL can be wri
             MPLm = beta * Qm/(Lbar-La)
             LA, weq = eqn(pMPLa, MPLm)
             titletxt = "Ricardo Viner"
             ymax = 1.25
             rv = figure(title=titletxt, tools=TOOLS,width=600, height=400)
             #rv.x_range = Range1d(0, LbarMax)
             rv.y_range = Range1d(0, ymax)
             rv.line(La, pMPLa,line_width=3,legend="Ag LD")
             rv.line(La, MPLm,line_width=3, color = 'red',legend="Mf LD")
             rv.line([Lbar,Lbar],[0,ymax], line_width=3, color='black')
             rv.line([0,Lbar],[weq,weq], line_width=1, line_dash=[4, 4], color='black')
             rv.line([LA, LA],[0, weq], line_width=1, line_dash=[4, 4], color='black')
             rv.xaxis.axis_label = 'Labor'
             rv.yaxis.axis_label = 'real wage'

             print("Real equilibrium wage: ( w/Pm ={0:5.2f}, w/Pa ={1:5.2f} )".format(weq, weq/
             print("Labor allocations : ( La ={0:5.0f}, Lm ={1:5.0f} )".format(LA, Lbar-LA))
             show(rv)

In [60]: foo(200)

C:\Users\jconning\AppData\Local\Continuum\Anaconda3\lib\site-packages\ipykernel\__main__.py:6:


    ---------------------------------------------------------------------------

    ValueError                                Traceback (most recent call last)

    <ipython-input-60-22818efa4acb> in <module>()
    ----> 1 foo(200)


    <ipython-input-57-4ed0510ad6e4> in foo(Lbar)
          6     pMPLa = (p * alpha * Qa/La) *(La<Lbar)          # for Cobb-Douglass MPL can be w
          7     MPLm = beta * Qm/(Lbar-La)
    ----> 8     LA, weq = eqn(pMPLa, MPLm)
          9     titletxt = "Ricardo Viner"
         10     ymax = 1.25
```

```
        <ipython-input-56-7cfedf00fc24> in eqn(pMPLa, MPLm)
          1 def eqn(pMPLa, MPLm):
    ----> 2     diff = (pMPLa-MPLm)*(La<Lbar)+(La>Lbar)*2
          3     LAidx = np.nanargmin(diff**2)  #value of LA where pMPLa(La) = MPLm(Lbar-La) is
          4     return La[LAidx], pMPLa[LAidx]


        ValueError: operands could not be broadcast together with shapes (200,) (400,)
```

```
In [28]: LA, weq = eqn(pMPLa, MPLm)
         titletxt = "Ricardo Viner"
         ymax = 1.25
         rv = figure(title=titletxt, tools=TOOLS,width=600, height=400)
         #rv.x_range = Range1d(0, LbarMax)
         rv.y_range = Range1d(0, ymax)
         rv.line(La, pMPLa,line_width=3,legend="Ag LD")
         rv.line(La, MPLm,line_width=3, color = 'red',legend="Mf LD")
         rv.line([Lbar,Lbar],[0,ymax], line_width=3, color='black')
         rv.line([0,Lbar],[weq,weq], line_width=1, line_dash=[4, 4], color='black')
         rv.line([LA, LA],[0, weq], line_width=1, line_dash=[4, 4], color='black')
         rv.xaxis.axis_label = 'Labor'
         rv.yaxis.axis_label = 'real wage'

         print("Real equilibrium wage: ( w/Pm ={0:5.2f}, w/Pa ={1:5.2f} )".format(weq, weq/p))
         print("Labor allocations : ( La ={0:5.0f}, Lm ={1:5.0f} )".format(LA, Lbar-LA))
         show(rv)
```

```
Real equilibrium wage: ( w/Pm = 0.35, w/Pa = 0.35 )
Labor allocations : ( La =  200, Lm =  200 )
```

```
In [ ]:
```

1.2 PPF plot

We can plot the associated production possibility frontier.
It will be useful to draw this as a paremetric plot by defining QM(La) and QA(La) and create a new columndatasource.

```
In [9]: QA = lambda La: (Tbar**(1-alpha) * La**alpha)
        QM = lambda La: Kbar**(1-beta) * (Lbar - La)**beta
```

```
In [10]: LA, weq = eqn(pMPLa, MPLm)
         priceline = (QM(LA)+p*QA(LA))- p*QA(La)
         ppfsource = ColumnDataSource(data=dict(
                     QA = QA(La),
                     QM = QM(La),
                     priceline = priceline
             )  )
```

```
In [11]: titletxt = "PPF"
         ppf = figure(title=titletxt, tools=TOOLS,width=500, height=500)
         ppf.x_range = Range1d(0, max(Qa)*1.25)
         ppf.y_range = Range1d(0, max(Qa)*1.25)
         ppf.line('QA','QM', source=ppfsource, line_width=3)
         ppf.circle(QA(LA), QM(LA), size=10)
         ppf.line('QA','priceline', source=ppfsource, line_width=1, line_dash=[4, 4])
         ppf.xaxis.axis_label = 'Qa'
         ppf.yaxis.axis_label = 'Qm'

         print("At Pa/Pm ={0:5.2f} country produces:".format(p))
         print("    Qa ={0:5.0f}, Qm ={1:5.0f} ".format(QA(LA), QM(LA)))
         show(ppf)

At Pa/Pm = 1.00 country produces:
    Qa =  141, Qm =  141
```

1.3 Interactive Plots

We'll use the slider from the ipywidgets library for Ipython working with Bokeh's ability to refresh a graph when a ColumnDataSource is refreshed. First we define a plot again but this time also give Bokeh a data source that can be refreshed.

```
In [12]: LA, weq = eqn(pMPLa, MPLm)

         weqLine = (np.ones(LbarMax)*weq) * (La<Lbar)      # we will use this to plot equilibr

         sourceRV = ColumnDataSource(data=dict(
                     La = La,
                     pMPLa = pMPLa,
                     MPLm = MPLm,
                     weqLine = weqLine
             )  )

         rvs = figure(tools=TOOLS, title='Ricardo-Viner', width=600, height=400)
         rvs.x_range = Range1d(0, LbarMax)
         rvs.y_range = Range1d(0, ymax)
         rvs.line('La', 'pMPLa',line_width=3, source=sourceRV,legend="Agric. LD")
         rvs.line('La', 'MPLm', line_width=3, source=sourceRV, color='red',legend="Manuf. LD")
         rvs.line('La', 'weqLine', line_width=1, line_dash=[4, 4], source=sourceRV, color='bla
         rvs.xaxis.axis_label = 'Labor'
         rvs.xaxis.axis_label_text_font_size = "12pt"
         rvs.yaxis.axis_label = r'Real wage'
         rvs.yaxis.axis_label_text_font_size = "12pt"
         rvs.legend.location = "top_right"
```

This next function takes slider values for $\bar{L}$ and $p = \frac{P_a}{P_m}$, updates the curves to be plotted and pushes those values to later Bokeh plots' data 'source' for a refresh.

```
In [18]: def updatervs(Lbar=LbarMax, p=1, Tbar=100,Kbar=100,):
             La = np.arange(0, Lbar)
             Lm = Lbar - La
             Qa = (Tbar**(1-alpha) * La**alpha) * (La<Lbar)
             Qm = Kbar**(1-beta) * (Lbar - La)**beta
             pMPLa = (p * alpha * Qa/La) *(La<Lbar)          # for Cobb-Douglass MPL can be wri
             MPLm = beta * Qm/(Lbar-La)
             rvs = figure(tools=TOOLS, title='Ricardo-Viner', width=600, height=400)
             rvs.x_range = Range1d(0, Lbar)
             rvs.y_range = Range1d(0, ymax)
             rvs.line('La', 'pMPLa',line_width=3, source=sourceRV,legend="Agric. LD")
             rvs.line('La', 'MPLm', line_width=3, source=sourceRV, color='red',legend="Manuf. L
             rvs.line('La', 'weqLine', line_width=1, line_dash=[4, 4], source=sourceRV, color=
             rvs.xaxis.axis_label = 'Labor'
             rvs.xaxis.axis_label_text_font_size = "12pt"
             rvs.yaxis.axis_label = r'Real wage'
             rvs.yaxis.axis_label_text_font_size = "12pt"
             rvs.legend.location = "top_right"
             show(rvs)

In [19]: updatervs()

In [13]: def updateRV(Lbar=LbarMax, p=1, Tbar=100,Kbar=100,):
             Qa = (Tbar**(1-alpha) * La**alpha) * (La<Lbar)
             Qm = Kbar**(1-beta) * (Lbar - La)**beta
             pMPLa = (p * alpha * Qa/La) *(La<Lbar)
             MPLm = beta * Qm/(Lbar-La)

             LA, weq = eqn(pMPLa, MPLm)
             weqLine = (np.ones(LbarMax)*weq) * (La<Lbar)

             QA = (Tbar**(1-alpha) * LA**alpha)
             QM = Kbar**(1-beta) * (Lbar - LA)**beta
             sourceRV.data['MPLm'] = MPLm
             sourceRV.data['pMPLa'] = pMPLa
             sourceRV.data['weqLine'] = weqLine
             sourceRV.push_notebook()
             print("Equilibrium wages: ( w/Pm ={0:5.2f}, w/Pa ={1:5.2f} )".format(weq, weq/p))
             print("Labor and output : ( La ={0:5.0f}, Lm ={1:5.0f} ), \
               (Qa = {2:5.0f}, Qm = {3:5.0f}) ".format(LA, Lbar-LA, QA, QM))
```

The next two cells display the plot and seupt the interactive slider.
NOTE: you can only use the slider when this notebook is run interactively on a jupyter server.

```
In [14]: rvs.plot_width=600
         show(rvs)

In [15]: interact(updateRV, p =(0.25,2,0.25), Lbar=(100, LbarMax, 10), Tbar=(50, 200, 10), Kba

Out[15]: <function __main__.updateRV>
```

1.4 Two synchronized plots

Let's try to show the labor allocation and PPF plots side by side and responsive to the same slider. For this we are going to setup two data sources, one for each plot.

```
In [ ]: def updateBoth(Lbar=LbarMax, p=1, Tbar=100,Kbar=100,):
            Qa = (Tbar**(1-alpha) * La**alpha) * (La<Lbar)
            Qm = Kbar**(1-beta) * (Lbar - La)**beta
            pMPLa = (p * alpha * Qa/La) *(La<Lbar)
            MPLm = beta * Qm/(Lbar-La)

            LA, weq = eqn(pMPLa, MPLm)
            weqLine = (np.ones(LbarMax)*weq) * (La<Lbar)

            sourceRV.data['MPLm'] = MPLm
            sourceRV.data['pMPLa'] = pMPLa
            sourceRV.data['weqLine'] = weqLine
            sourceRV.push_notebook()

            QA = lambda La: (Tbar**(1-alpha) * La**alpha)
            QM = lambda La: Kbar**(1-beta) * (Lbar - La)**beta
            priceline = (QM(LA)+p*QA(LA)) - p*QA(La)

            ppfsource.data['QA'] = QA(La)
            ppfsource.data['QM'] = QM(La)
            ppfsource.data['priceline'] = priceline
            ppfsource.push_notebook()
```

As of 10/1/15 THe plots that follow are not quite working.

Major issue: only the RV plot is updating... PPF plot is not changed, not sure why. Minor: need to rescale to fit the two plots side by side.

```
In [ ]: ppf2=ppf
        rvs2=rvs

        ppf2.plot_height=350
        ppf2.plot_width=350
        rvs2.plot_height=350
        rvs2.plot_width=350
        show(vplot(hplot(ppf, rvs,)))
```

```
In [ ]: interact(updateBoth, p =(0.25,2,0.25), Tbar=(50, 200, 10), Kbar=(50, 200, 10), Lbar=(1
```