The MRW Data

This section provides some descriptive insights into the dataset provided by [Mankiw.Romer.Weil-1992-QJE]_ (MRW) in the data appendix to their paper. The data below were imported by optical scan from a PDF of this data appendix. The replication results reported below are quite close but not perfect, so either the scan is not perfect or their appendix contains a typo.

Create the Datasets

The MRW data comes from the real national accounts of [Summers.Heston-1988]_, covering 1960-1985. The **Y1960** and **Y1980** fields hold GDP per working-age adult is provided for 1960 and 1985, and **agepop** is the average annual growth rate from 1960–1985 of th3 15–64 age group (the "working age" population). The saving rate **IoY** is measured as the investment proportion of GDP ($s = I/GDP$). Using dummy variables, these countries have been classified as belong to three groups: **N**, **I**, and **O** denote the non-oil, intermediate, and OECD samples.

A dataset subtlety is that, since investment rates are not constant over time, MRW average them over the period. In addition, as discussed below, they impose $g_A + \delta == 0.05$ for every country. (They claim plausible changes do not affect the results.)

In[ ]:=
```
mrwDataLocation = "https://subversion.american.edu/aisaac/hw/data/mrw1992.csv";
(* online data *)
(* mrwDataLocation=dataFolder<>"mrw1992.csv"; *) (* for downloaded data *)
SetOptions[Dataset, MaxItems → 5]; (* limit size of display *)
dsAll = SemanticImport[mrwDataLocation] (* create a dataset *)
```

Out[ ]=

Country	N	I	O	Y1960	Y1985	GDP	agepop	IoY
Egypt	1	0	0	907	2160	6.0	2.5	16.3
Ethiopia	1	1	0	533	608	2.8	2.3	5.4
Gabon	0	0	0	1307	5350	7.0	1.4	22.1
Gambia	0	0	0	799	—	3.6	—	18.1
Ghana	1	0	0	1009	727	1.0	2.3	9.1

rows 11–15 of **121**

This is a tabular data set, with one record per row. In WL, this means that this dataset is essentially a list of associations, where each association maps field names (as displayed in the header) to values. For example, consider the first record and apply the **Normal** command to it in order to produce the corresponding association, which maps each key to a value.

In[ ]:=
```
Normal@dsAll[[1]]
```

Out[ ]=
```
⟨|Country → Algeria, N → 1, I → 1, O → 0, Y1960 → 2485,
 Y1985 → 4371, GDP → 4.8, agepop → 2.6, IoY → 24.1, SCHOOL → 4.5|⟩
```

The **Length** command suggests that the dataset **dsAll** holds data on 121 countries.

In[ ]:=
```
Length[dsAll]
```

Out[ ]=
```
121
```

However, scrolling through the data reveals quite a few missing values. To get a sense of how pervasive this is, create a query to categorize each record as to whether or not is has a missing value in any field. To see how this will work, consider the record for Gambia (**dsAll[[14]]**). This reveals that a couple values in the association are **Missing[Empty]**.

In[ ]:=
```
record14 = Normal@dsAll[[14]]
```

Out[ ]=
```
⟨|Country → Gambia, N → 0, I → 0, O → 0, Y1960 → 799, Y1985 → Missing[Empty],
 GDP → 3.6, agepop → Missing[Empty], IoY → 18.1, SCHOOL → 1.5|⟩
```

When applied to an association, the **MemberQ** command checks the values of the association for a pattern match. Conveniently, it is possible to match the the head **Missing**, so just check for that. Here is one way to do this.

In[ ]:=
```
MemberQ[_Missing]@record14
```

Out[ ]=
```
True
```

It is possible to turn this into a query (**Query[All,MemberQ[_Missing]]**) and apply it to the dataset. This would produce a **False** or **True** value for each record, allowing us to count up the **False** values in order to determine the number of complete and incomplete records with the **Counts** command. However, it is simpler to use the query **Query[Counts,MemberQ[_Missing]]** to do this all at one go. Finally, remember that WL has a simplified syntax for dataset queries, as follows.

In[ ]:=
```
dsAll[Counts, MemberQ[_Missing]]
```

Out[ ]=

False	104
True	17

Data Cleaning

The **DeleteMissing** command can filter out such records. Take a little care to read the documentation, remembering that the goal is do delete the entire record whenever any of its values are **Missing**.

In[ ]:=
```
dsNoMissing = DeleteMissing[dsAll, 1, 1];
```

As expected from the previous exploration, the new dataset contains 104 records.

In[•]:=
```
Length@dsNoMissing
```

Out[•]=
```
104
```

Next, construct the three subsets considered by MRW by selecting based on their dummy variables. This is called subsetting, selective set building, or filtering.

Review the documentation for the **Function** command. This describes the ampersand postfix shorthand, which is commonly used by WL programmers. When a field name is a string, WL allows you to prefix it with an octothorpe (#) in order to refer to it in a function. This makes it easy to subset the dataset based on the provided value of a dummy variable. The following **Select** commands each have a function argument that uses this notation.

In[•]:=
```
dsN = Select[#N == 1 &]@dsNoMissing; (* nonoil *)
dsI = Select[#I == 1 &]@dsNoMissing; (* intermediate (subset of dsN) *)
dsO = Select[#O == 1 &]@dsNoMissing;
(* 22 high pop OECD countries (subset of dsI) *)
```

Countries whose GDP is primarily from oil production are not in any of these groups. For example, Kuwait is not.

In[•]:=
```
Select[dsNoMissing, #Y1960 > 50 000 &]
```

Out[•]=

Country	N	I	O	Y1960	Y1985	GDP	agepop	IoY
Kuwait	0	0	0	77 881	25 635	2.4	6.8	9.5

The **Length** command can count the number of observations in each dataset. The following example uses the **Map** command to compute all three lengths at one go. (The **/@** operator is an infix shorthand for the **Map** command, which applies a function to each element of a list.)

In[•]:=
```
Map[Length, {dsN, dsI, dsO}]
```

Out[•]=
```
{98, 75, 22}
```

Note that the intermediate countries are a subset of the non-oil countries.

In[•]:=
```
Length@Select[0 == #N && 1 == #I &]@dsNoMissing (* nothing is selected *)
```

Out[•]=
```
0
```

Examine the Data

The Cross-Country Distribution of Income

The cross-country distribution of income has shifted substantially over time. On a cross-country basis, it seems that the mean income has risen while the variance has fallen.

In[ ]:=
```
dsI[Mean, {"Y1960", "Y1985"}]
```

Out[ ]=

Y1960	3620.76
Y1985	6589.83

In[ ]:=
```
dsI[Variance, {"Y1960", "Y1985"}]
```

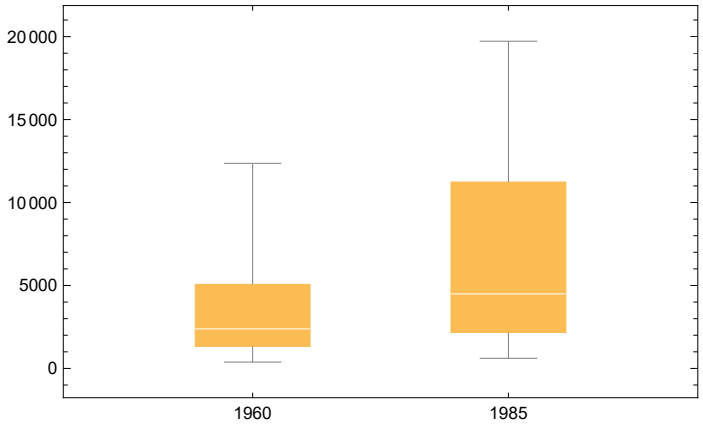
Out[ ]=

Y1960	8999859.
Y1985	29277917.

Consider a first visualization of the income distributions.

In[ ]:=
```
BoxWhiskerChart[
 Transpose@dsI[All, {"Y1960", "Y1985"}],
 ChartLabels → {"1960", "1985"}]
```

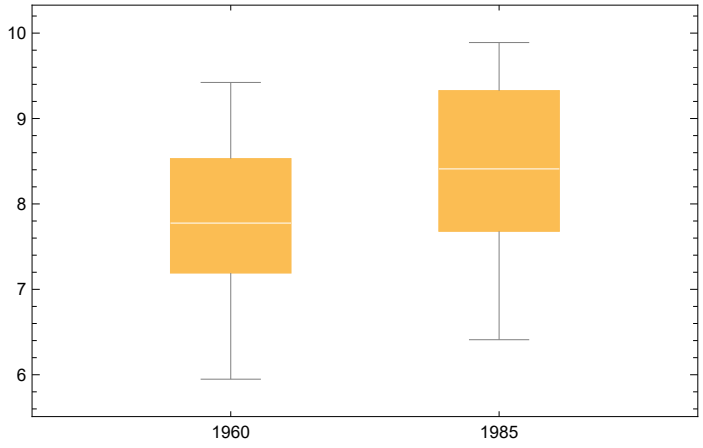
Out[ ]=



Check whether the logarithms appear more normally distributed.

In[ ]:=
```
BoxWhiskerChart[
 Transpose@dsI[All, {"Y1960", "Y1985"}, Log],
 ChartLabels → {"1960", "1985"}]
```

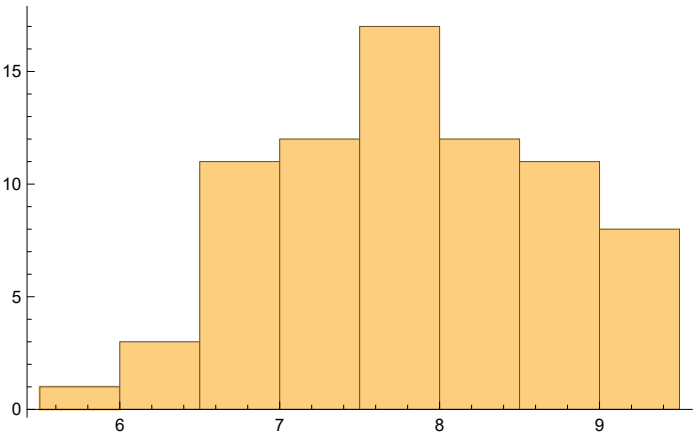
Out[ ]=



Economists often use histograms to provide a more detailed look at univariate distributions.

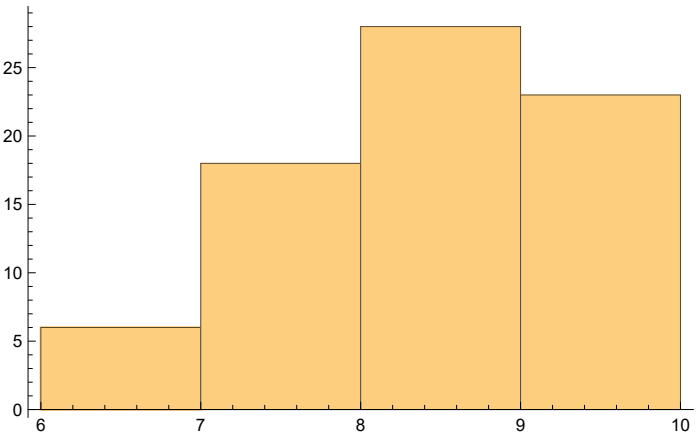
In[ ]:=
```
dsI[Histogram, "Y1960", Log]
```

Out[ ]=



In[ ]:=
```
dsI[Histogram, "Y1985", Log]
```

Out[ ]=



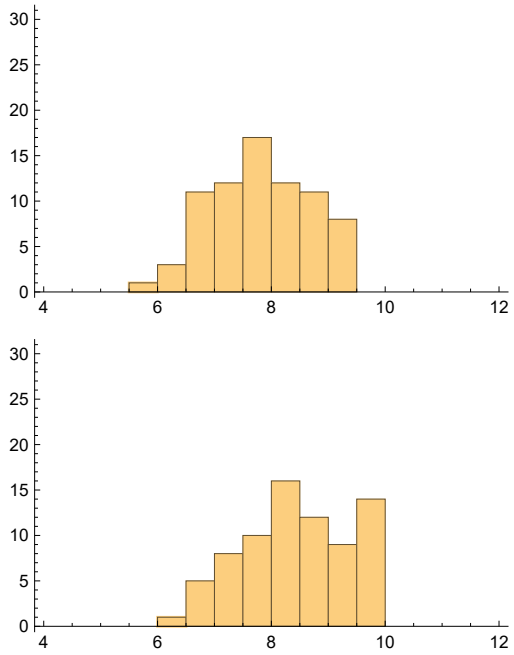
In order to make these more easily comparable, create a custom histogram function.

In[ ]:=
```
hist = data ↦ Histogram[Log@data, {0.5}, PlotRange → {{4, 12}, {0, 30}}];
```

Notice the substantial flattening and rightward shift of the distribution between 1960 and 1985.

In[ ]:=
```
Labeled[
 GraphicsColumn[{
   dsI[hist, "Y1960"],
   dsI[hist, "Y1985"]
  }, ImageSize → 288],
 "Log Per Capita Income, 1960 and 1985"]
```

Out[ ]=

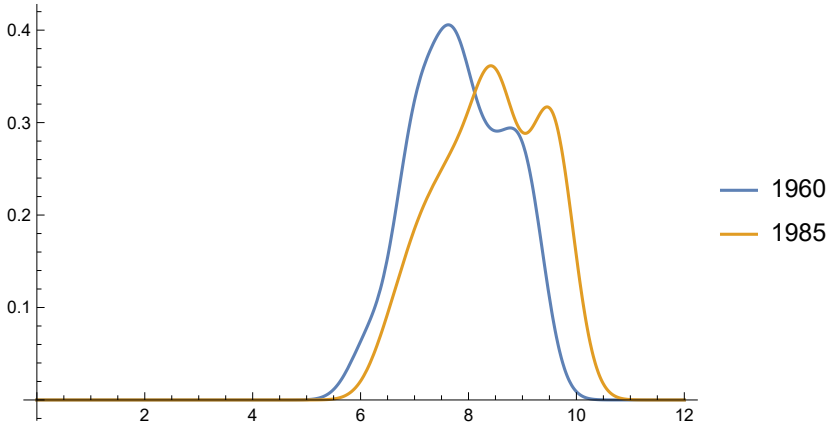


Log Per Capita Income, 1960 and 1985

A more refined window on the same distribution is provided by a kernel-density plot. The following chart plots both distributions.

In[ ]:=
```
pdf = data ↦ PDF@SmoothKernelDistribution[Log@data];
Plot[{
  dsI[pdf, "Y1960"][x],
  dsI[pdf, "Y1985"][x]},
 {x, 0, 12},
 PlotLegends → {"1960", "1985"}]
```

Out[ ]=



Correlations in the Data

Recall that, given a Cobb-Douglas production function, the basic neoclassical growth model predicts that steady-state income (per worker) depends positively on the saving rate and negatively on population growth: $y_{ss} = B\left(\frac{Bs}{d+g_A+g_N}\right)^{\alpha/(1-\alpha)}$. Postpone the logarithmic transformation of the RHS variables, for reasons that will soon be evident, and consider the correlation between the saving rate, population growth, and per-capita income.

It proves convenient to first convert the data into a matrix by stripping out the keys (with the **Values** command) and then convert it to a normal matrix (with the **Normal** command). (As of Mathematica version 13, this preliminary step remains necessary.) Then the **Correlation** command can produce the correlations. For the intermediate countries, the raw correlations are crudely aligned with our predictions.

In[ ]:=
```
dsI[Correlation@*Normal@*Values, {"IoY", "agepop", "Y1985"}]
```

Out[ ]=

1.0	−0.393421	0.665647
−0.393421	1.0	−0.606623
0.665647	−0.606623	1.0

Use **TableForm** to format it nicely.

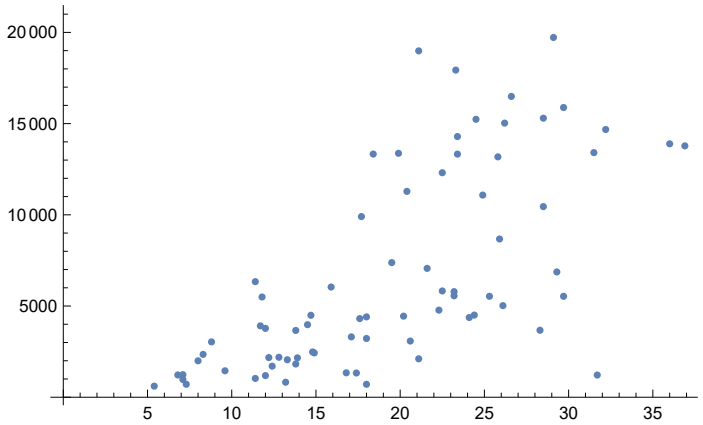
Out[•]//TableForm=

	s	gN	Y/N
s	1.	−0.393421	0.665647
gN	−0.393421	1.	−0.606623
Y/N	0.665647	−0.606623	1.

Next produce the corresponding scatter plots. First do this for the saving rate and per capita income.

In[•]:=
```
dsI[ListPlot, {"IoY", "Y1985"}]
```

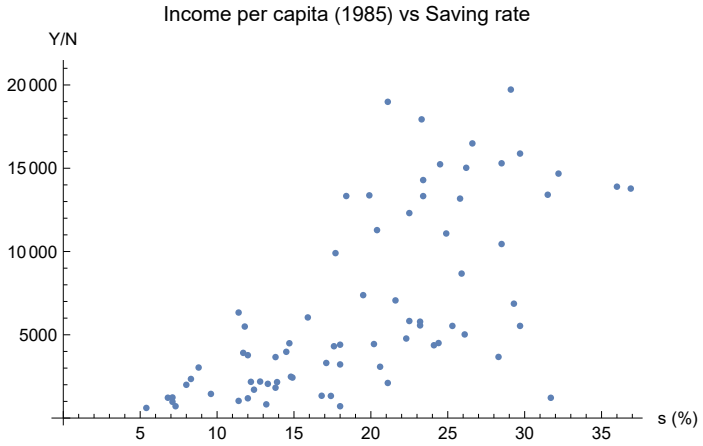
Out[•]=



This is not bad, but the lack of labels is distracting. So try again.

In[•]:=
```
ListPlot[dsI[All, {"IoY", "Y1985"}],
 AxesLabel → {"s (%)", "Y/N"},
 PlotLabel → "Income per capita (1985) vs Saving rate"]
```

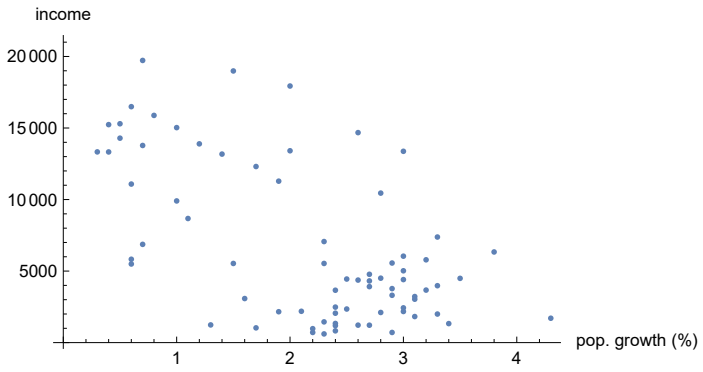
Out[•]=



Do the same for the growth rate of the working-age population and per capita income.

In[ ]:=
```
ListPlot[dsI[All, {"agepop", "Y1985"}],
 AxesLabel → {"pop. growth (%)", "income"}]
```

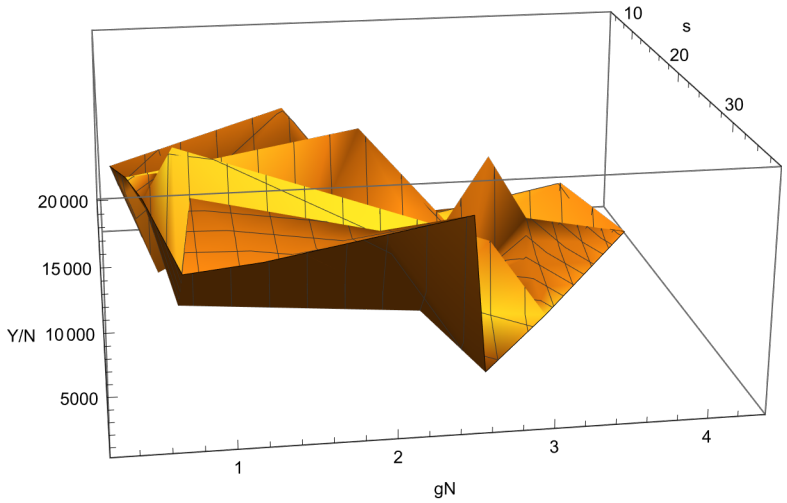
Out[ ]=



A three-dimensional scatter plot provides an overall view of these correlations.

In[ ]:=
```
ListPlot3D[dsI[All, {"IoY", "agepop", "Y1985"}],
 AxesLabel → {"s", "gN", "Y/N"}]
```

Out[ ]=



Next, recode the dataset. Once again, look up the **Function** command the ampersand postfix short-hand, which is commonly used by WL programmers.

Recode the dataset by creating a transformation to map across each record. Note that the unobserved $d + g_A$ is approximated by 0.05 for all countries. (See MRW for the motivation.)

In[ ]:=
```
recoder = <|
    "ln`s" → Log[#"IoY" / 100],
    "ln`n" → Log[0.05 + #agepop / 100],
    (* recall MRW notation difference (n=d+gA+gN) *)
    "ln`y" → Log[#Y1985]
    |> &;
dsI`recode = dsI[All, recoder]
```

Out[ ]=

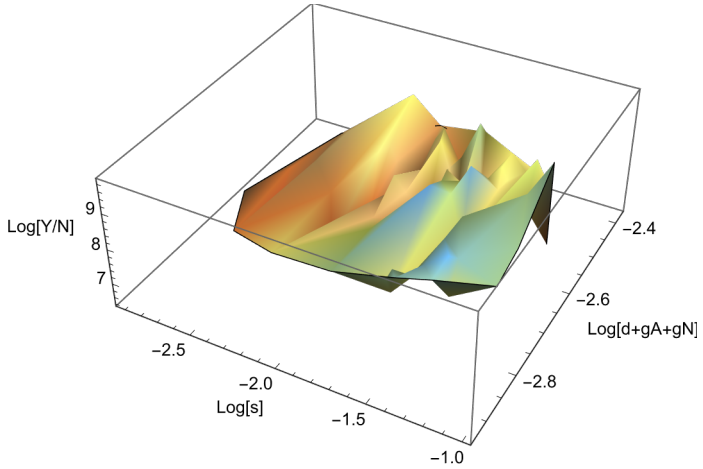
ln`s	ln`n	ln`y
−1.42296	−2.57702	8.38275
−1.26231	−2.50104	8.20822
−2.05573	−2.64508	7.69166
−2.91877	−2.6173	6.41017
−2.08747	−2.37516	7.44073

rows 1–5 of 75

If we plot this data, the positive correlation with the saving rate is evident but the negative correlation with the population growth rate somewhat less so.

In[ ]:=
```
ListPlot3D[dsI`recode,
 AxesLabel → {"Log[s]", "Log[d+gA+gN]", "Log[Y/N]"},
 Mesh → None, InterpolationOrder → 3, ColorFunction → "SouthwestColors"]
```

Out[ ]=



Log-Linear Formulation for Empirical Application

This section illustrates an approach to replicating the MRW estimates.

Recall the steady state value of the Solow-Swan model with Cobb-Douglas production:

$y_{ss} = B\left(\frac{B\,s}{d+g_L}\right)^{\alpha/(1-\alpha)}$. A logarithmic transformation produces

$$\text{Log}[y_{ss}] == \frac{\text{Log}[B]}{1-\alpha} + \frac{\alpha}{1-\alpha}\,\text{Log}[s] - \frac{\alpha}{1-\alpha}\,\text{Log}[d+g_L] \tag{1}$$

This log-linear formulation suggests a possible linear regression equation.

$$\text{Log}[y_{ss}] == \beta_{00} + \beta_1\,\text{Log}[s] + \beta_2\,\text{Log}[d+g_L] \tag{2}$$

Unfortunately, estimation of this regression equation requires data on $y = Y/L = Y/(A\,N)$. (This notation differs slightly from MRW.) The value of A is unobserved. The MRW solution is to move the technology variable to the right-hand side, as follows.

Assume the efficiency of labor grows at a constant rate: $A_t = A_0\,e^{g_A\,t}$. Since $Y/N = A\,y$, we have Log[Y/N] = Log[A] + Log[y]. This is also true in a steady state y_{ss}, where

$$\text{Log}[Y/N] == \text{Log}[A_0] + g_A\,t + \text{Log}[y_{ss}] == \text{Log}[A_0] + g_A\,t + \beta_{00} + \beta_1\,\text{Log}[s] + \beta_2\,\text{Log}[d+g_L]$$

MRW bring this to the data as follows. First, add a country-specific stochastic element to the model by specifying that for each country i.

$$\text{Log}[A_{i,0}] = a + \epsilon_i$$

Here the constant a is shared across countries but ϵ_i is a country-specific shock. Putting this all together yields a linear regression specification where for each country

$$\text{Log}[Y/N] == \beta_0 + \beta_1\,\text{Log}[s] + \beta_2\,\text{Log}[d+g_A+g_N] + \epsilon \tag{3}$$

Recall $\beta_1 = \alpha/(1-\alpha)$ and $\beta_2 = -\alpha/(1-\alpha)$, and note that all constant terms move into the intercept: $\beta_0 = a + g_A\,t + \text{Log}[B]/(1-\alpha)$.
MRW claim that data on factor shares suggest $\alpha \approx 1/3$ so that we should expect $\beta_1 = -\beta_2 \approx 1/2$.

Use the data they provide for 75 “intermediate” countries to estimate this with ordinary least squares. The model is

$$\text{Log}[Y/N] == X.\beta + \epsilon$$

Here the dependent variable is (the log of) per-capita GDP. The matrix X of exogenous variables has the following columns: all ones (for the intercept), the log of the saving rate, and the log of $0.05 + g_N$. (Recall that the unobserved $d + g_A$ is approximated by 0.05 for all countries).

Exercise

Use Mathematic to produce the first regression equation with coefficients {β_{00}, β_1, β_2}.

A Solution:

In[ ]:=
```
ss`y
```

Out[ ]=
$$B\left(\frac{B\,s}{d+gL}\right)^{\frac{\alpha}{1-\alpha}}$$

A logarithmic transformation:

```
In[•]:= ss`lny = Log[ss`y] // PowerExpand // Apart
```

$$\text{Out[•]=}\quad -\frac{\text{Log}[B]-\alpha\,\text{Log}[d+gL]}{-1+\alpha}-\frac{\alpha\,\text{Log}[s]}{-1+\alpha}$$

Rewrite this by hand as follows.

```
In[•]:= ss`lny == β00 + β1 Log[s] + β2 Log[d + gL] /.
          <|β00 → Log[B]/(1 - α), β1 → α/(1 - α), β2 → -α/(1 - α)|> // Simplify
```

```
Out[•]= True
```

Review of Least Squares

Every choice of β produces a different vector of residuals. Let e represent the residual vector, which is a function of β.

$$e=\beta\mapsto y-X.\beta$$

We will choose β to minimize the norm of $e[\beta]$. That is,

$$\beta = \text{argmin}_\beta\, e[\beta].e[\beta] \tag{4}$$

This is called the least squares solution, because it is minimizes the sum of the squared residuals. Note that the properties of matrix multiplication imply that

$$(y - X.\beta).(y - X.\beta) == y^\intercal.y - 2\,y^\intercal.(X.\beta) + \beta^\intercal.(X^\intercal.X).\beta$$

Setting the derivative with respect to β to zeros produces the first-order necessary conditions.

$$-2X^\intercal.y+2\,(X^\intercal.X)\,\beta==0$$

Assuming X has full column rank, solve for β as follows.

$$\beta = (X^\intercal.X)^{-1}.X^\intercal.y$$

Recall that for any choice of β there is a corresponding residual vector e such that $y == X.\beta + e$. We can therefore substitute for y in the first-order conditions to get

$$X^\intercal.e == 0$$

That is, to minimize the length of the residual vector, it must be orthogonal to the data.

Fitting a Linear Model

Next, produce a fitted model with the **LinearModelFit** command. Currently, the input to this command must be a matrix of values, where each row represents an observation and the last value in each row is the dependent variable. (Unfortunately, this command does not yet work directly with dataset objects.) Convert this dataset into a matrix by stripping out the keys with the **Values** command and then converting it to a normal matrix with the **Normal** command.

```
In[•]:= dsI`recodeM = Normal@Values@dsI[All, recoder]; (* convert to matrix *)
        dsI`fit = LinearModelFit[dsI`recodeM, {ln`s, ln`n}, {ln`s, ln`n}]
         (* fitted model *)
```

```
Out[•]= FittedModel[ 5.34587 – 2.0172 ln`n + 1.31755 ln`s ]
```

Ordinarily, we want to take a look at the parameter table for a fitted model.

```
In[•]:= dsI`fit["ParameterTable"]
```

Out[•]=

	Estimate	Standard Error	t-Statistic	P-Value
1	5.34587	1.54308	3.46442	0.00089895
ln`s	1.31755	0.170943	7.70758	5.38321×10^{-11}
ln`n	–2.0172	0.533866	–3.77848	0.000322373

Alternatively, we may prefer to look at confidence intervals. (These are 95% intervals by default.)

```
In[•]:= dsI`fit["ParameterConfidenceIntervalTable"]
```

Out[•]=

	Estimate	Standard Error	Confidence Interval
1	5.34587	1.54308	{2.2698, 8.42193}
ln`s	1.31755	0.170943	{0.976785, 1.65832}
ln`n	–2.0172	0.533866	{–3.08144, –0.952957}

As a crude measure of goodness of fit, we may look at the adjusted R^2. Roughly speaking, this is the proportion of the variability in the dependent variable that is explained by variation in the independent variables. For such a simple model of such a complex phenomenon, this measure looks pretty good.

```
In[•]:= dsI`fit["AdjustedRSquared"]
```

```
Out[•]= 0.587767
```

One problem with the estimates is that they are not clearly related as the algebraic derivation of the model leads us to expect. So, take a look at the restricted model.

```
In[•]:= dsI`rfit = LinearModelFit[dsI`recodeM, {ln`s – ln`n}, {ln`s, ln`n}];
        dsI`rfit["ParameterTable"]
```

Out[•]=

	Estimate	Standard Error	t-Statistic	P-Value
1	7.09292	0.145614	48.7106	2.04004×10^{-57}
–ln`n + ln`s	1.43096	0.139123	10.2855	7.57589×10^{-16}

```
In[•]:= dsI`rfit["AdjustedRSquared"]
```

```
Out[•]= 0.586111
```

Despite the restriction, the residuals do not grow much and the adjusted R^2 barely budges. Therefore, expect not to reject this constraint. Nevertheless, we should test it.

To test the constraint, first compute the F-statistic for the model comparison. Get the sum of squares from each model, and then use the standard formula (the proportional change in the residuals scaled by the proportional change in the degrees of freedom).

Out[•]//TableForm=

```
unrestricted SSE    26.8475
restricted SSE      27.3298
F statistic         1.29333
```

Then find the p-value:

```
In[ ]:= 1 - CDF[FRatioDistribution[1, nI - k], fstat]
Out[ ]= 0.259206
```

This is too big to reject the restriction. We (naturally) get the same result if we use the **HypothesisTesting** package.

```
In[ ]:= << HypothesisTesting`
In[ ]:= FRatioPValue[fstat, 1, nI - k]
Out[ ]= OneSidedPValue → 0.259206
```

In sum, things look ... not bad. We get a reasonable amount of the variance explained, given that we are using cross section data, and we cannot reject a constraint implied by theory. But the key coefficient estimate of around 3/2 implies a value for α of around 0.6, which is almost twice the share of capital in the national income accounts.

And there is another problem. If we restrict the analysis to just their OECD subsample, the estimates are less encouraging. The large p-values mean we lack confident that saving rates and population growth rates have predictive power.

```
In[ ]:= ds0`fit = LinearModelFit[Normal@Values@dsO[All, recoder], {ln`s, ln`n}, {ln`s, ln`n}]
Out[ ]= FittedModel[ 8.02061 - 0.741921 ln`n + 0.49989 ln`s ]

In[ ]:= ds0`fit["ParameterTable"]
```

Out[ ]=

	Estimate	Standard Error	t-Statistic	P-Value
1	8.02061	2.51789	3.18545	0.00487043
ln`s	0.49989	0.433896	1.1521	0.263572
ln`n	–0.741921	0.852195	–0.870601	0.394839

Note that the model predicts both the signs and the *magnitudes* of s and n on Y/N, since α is capital's share and is known to be about 1/3 (implying $[\alpha/(1-\alpha)] \approx .5$. Their cross section estimates of (1) support the sign prediction, and they also accept the equal size of the coefficients on n and s. But yield $\alpha \approx .59$, which is much too large. (And the s.e. is small enough to easily reject $\alpha = 1/3$.)