CT561: Systems Modelling and Simulation

Week 6: Simulation with Vensim & Introduction to R

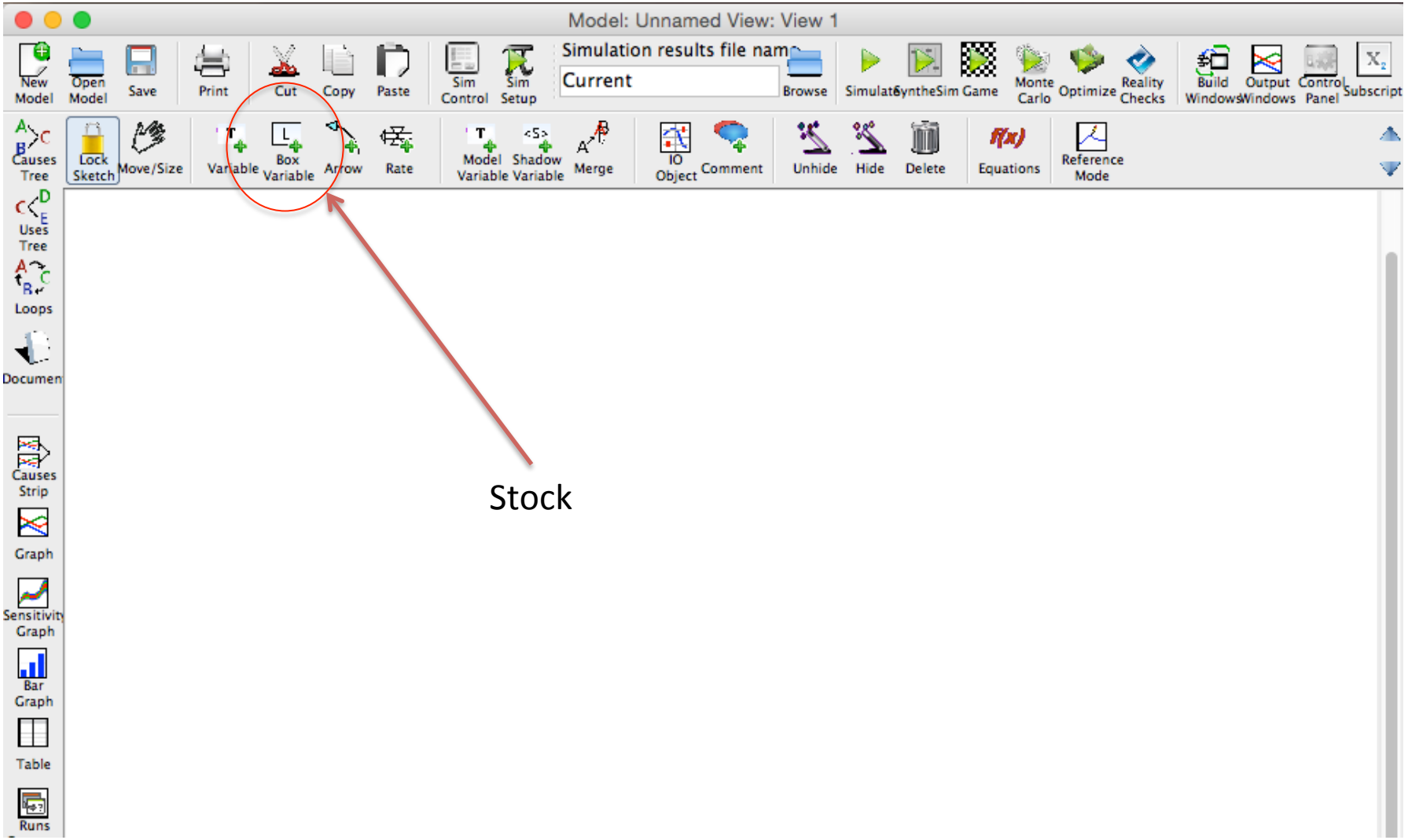
https://github.com/JimDuggan/CT561

Dr. Jim Duggan,
Information Technology,
School of Engineering & Informatics

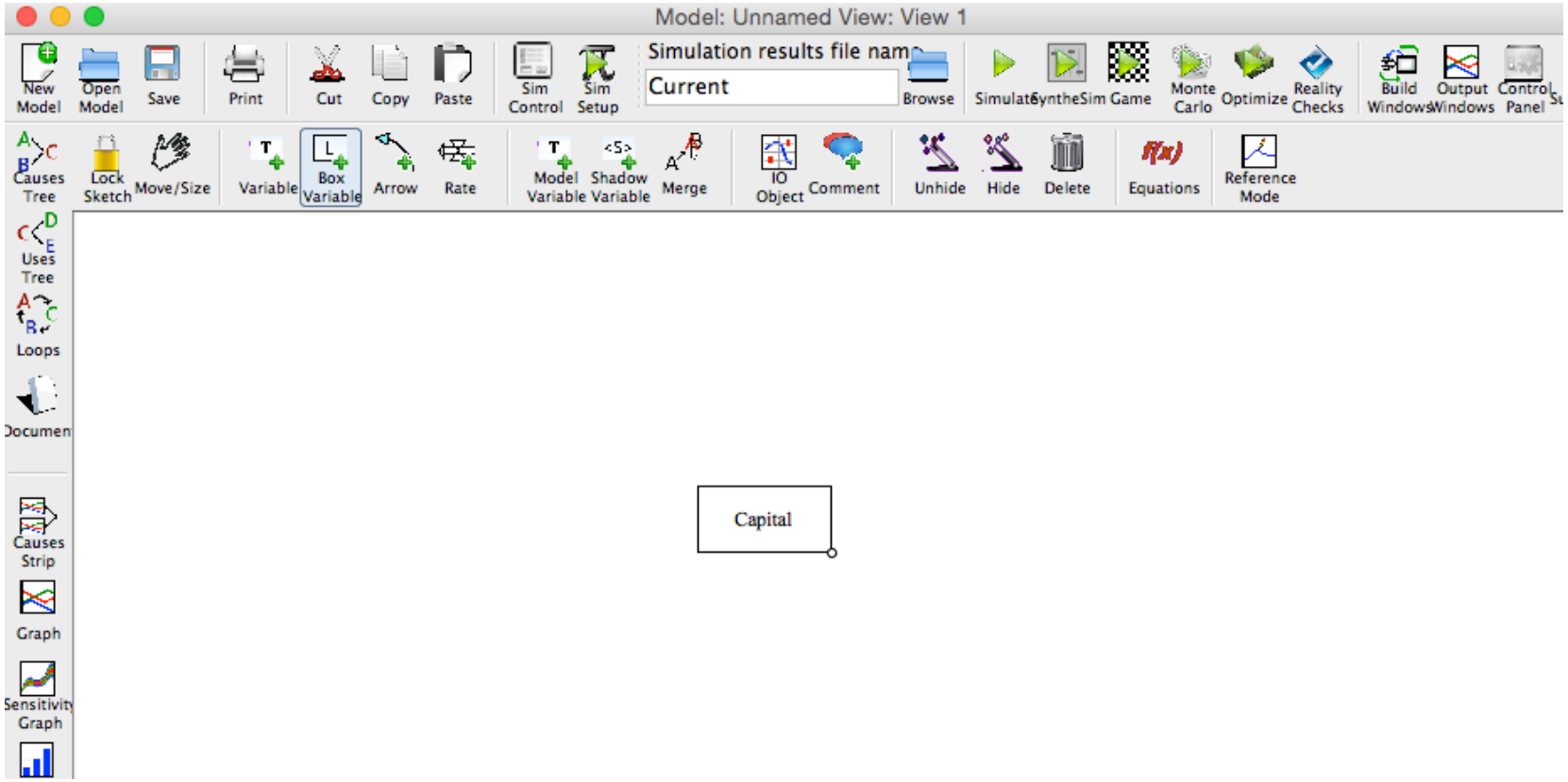
Overview

- Vensim PLE
  - Supports model building with stocks and flows
  - Equation editor
  - Time specs (Start, Finish, DT)
  - Supports graphing
- R and deSolve
  - Functional programming language
  - Supports data science methodology
  - Calibration and sensitivity analysis

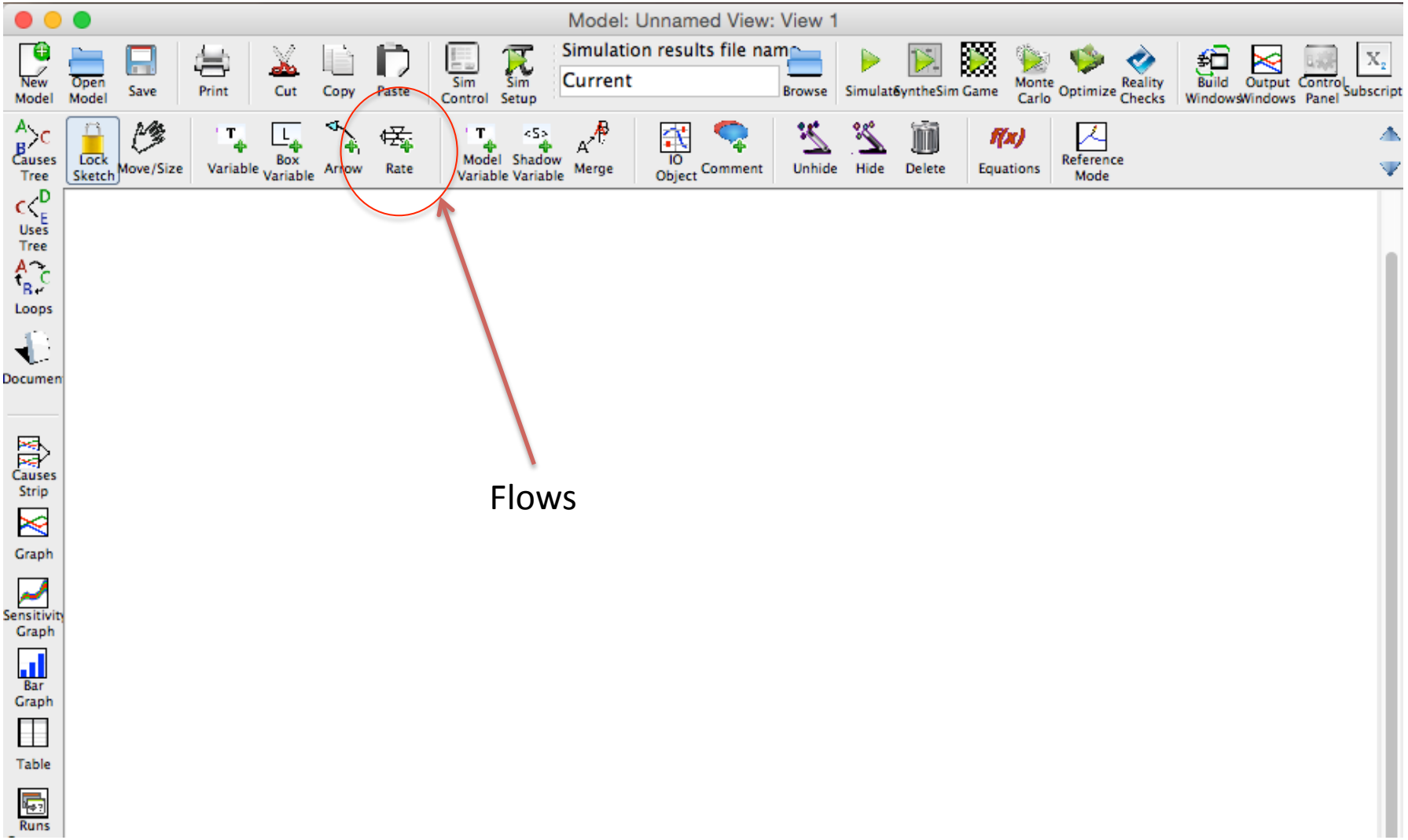
Vensim: Adding a Stock



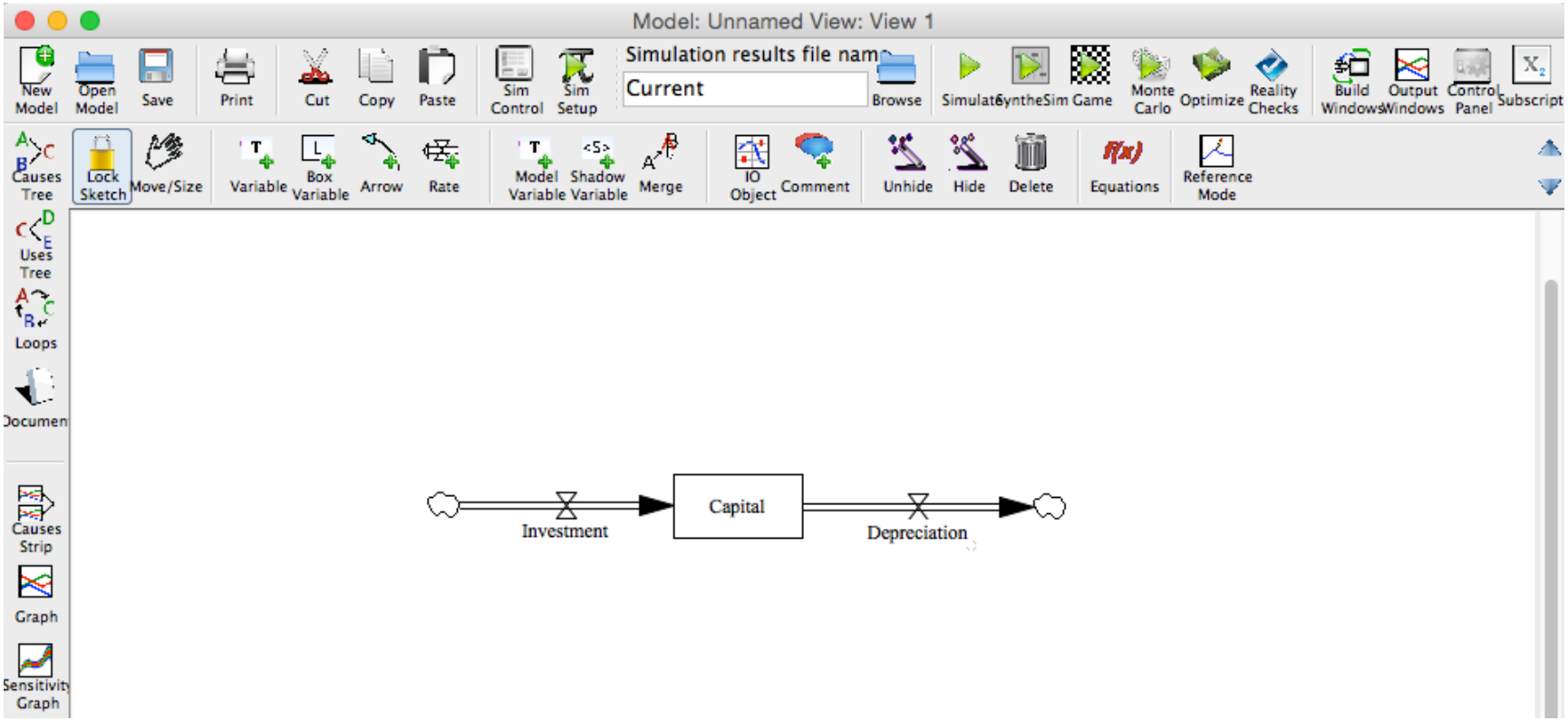
Capital Stock



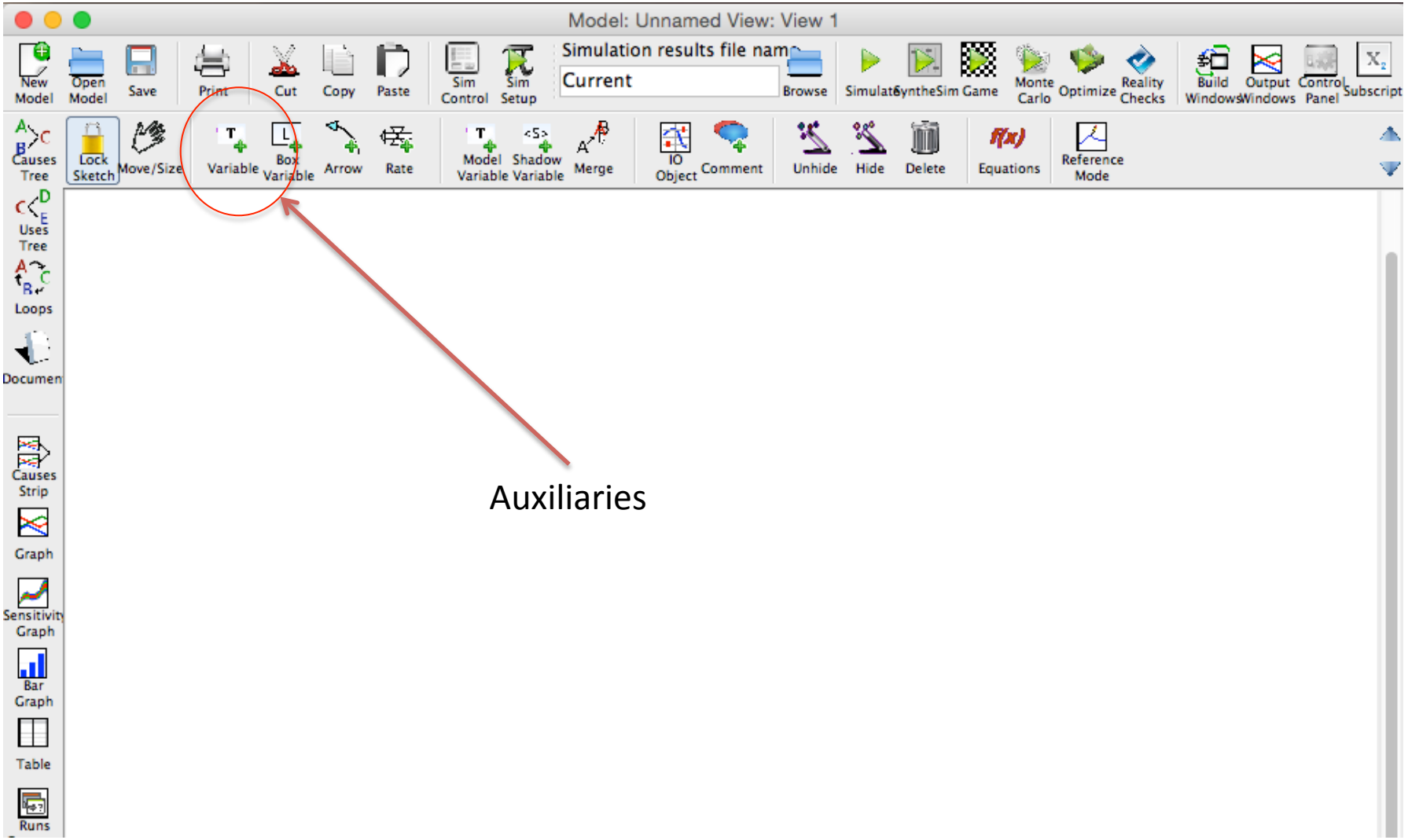
Vensim: Adding Flows



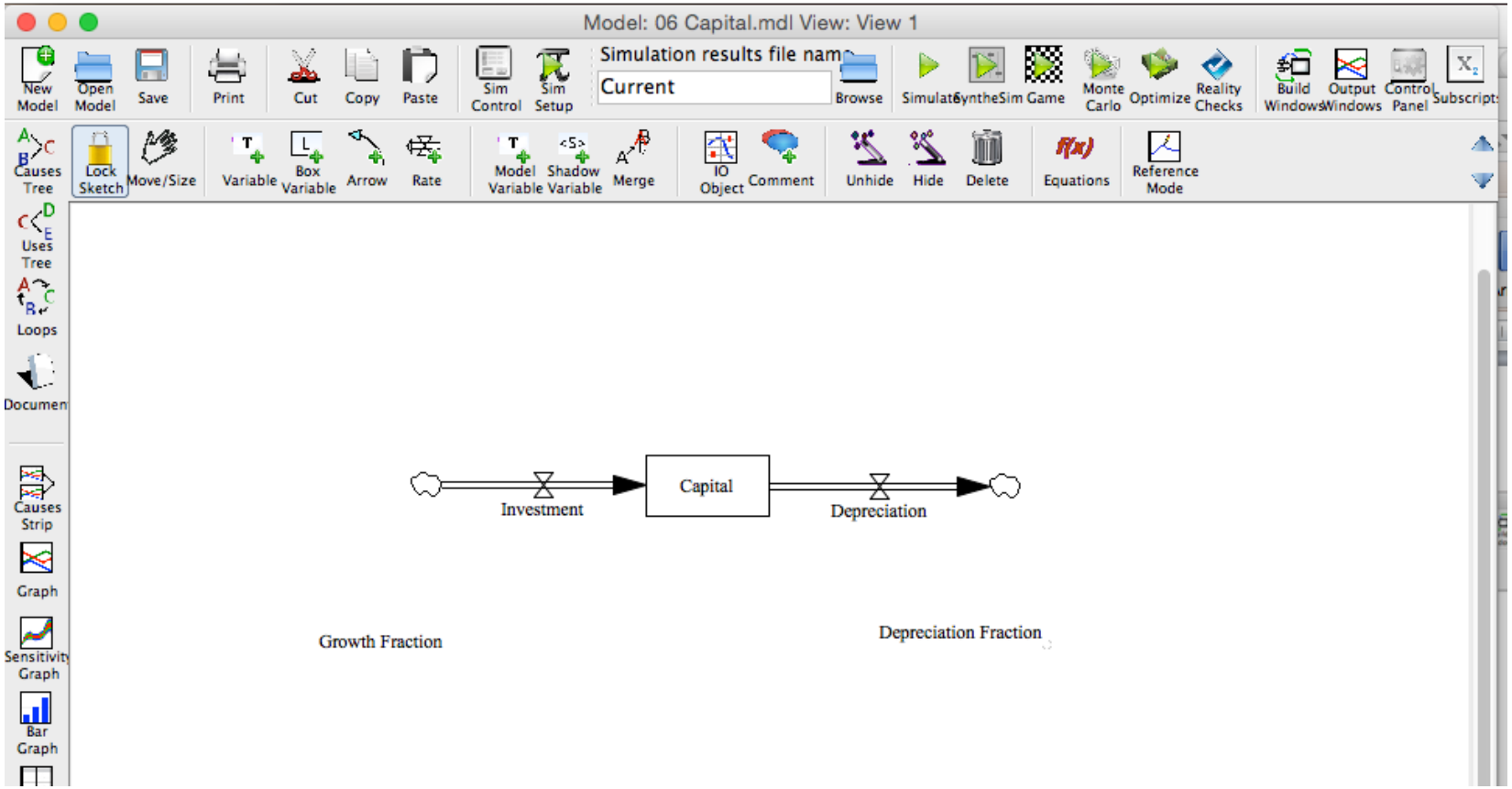
Investment and Depreciation Flows



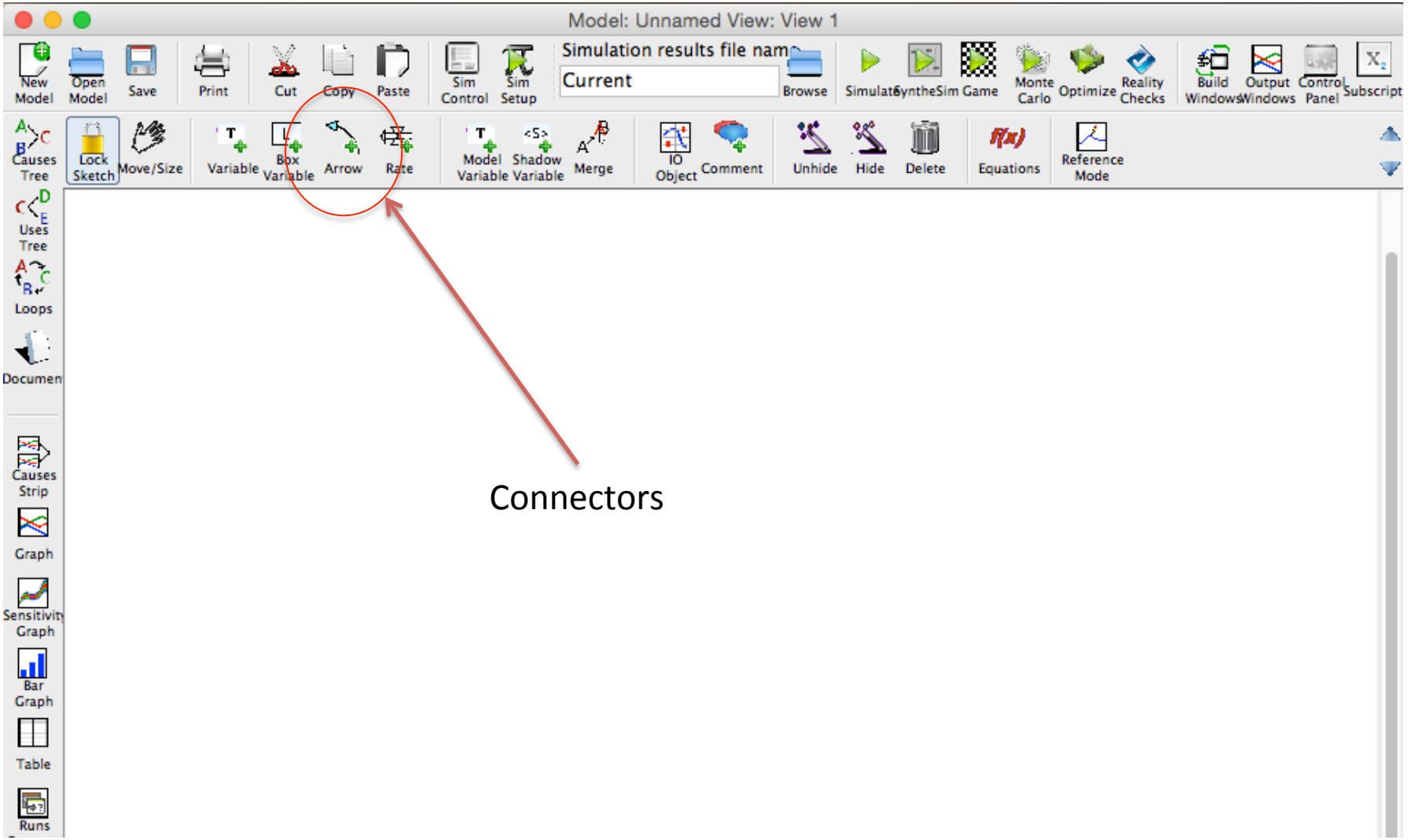
Vensim: Adding Auxiliaries



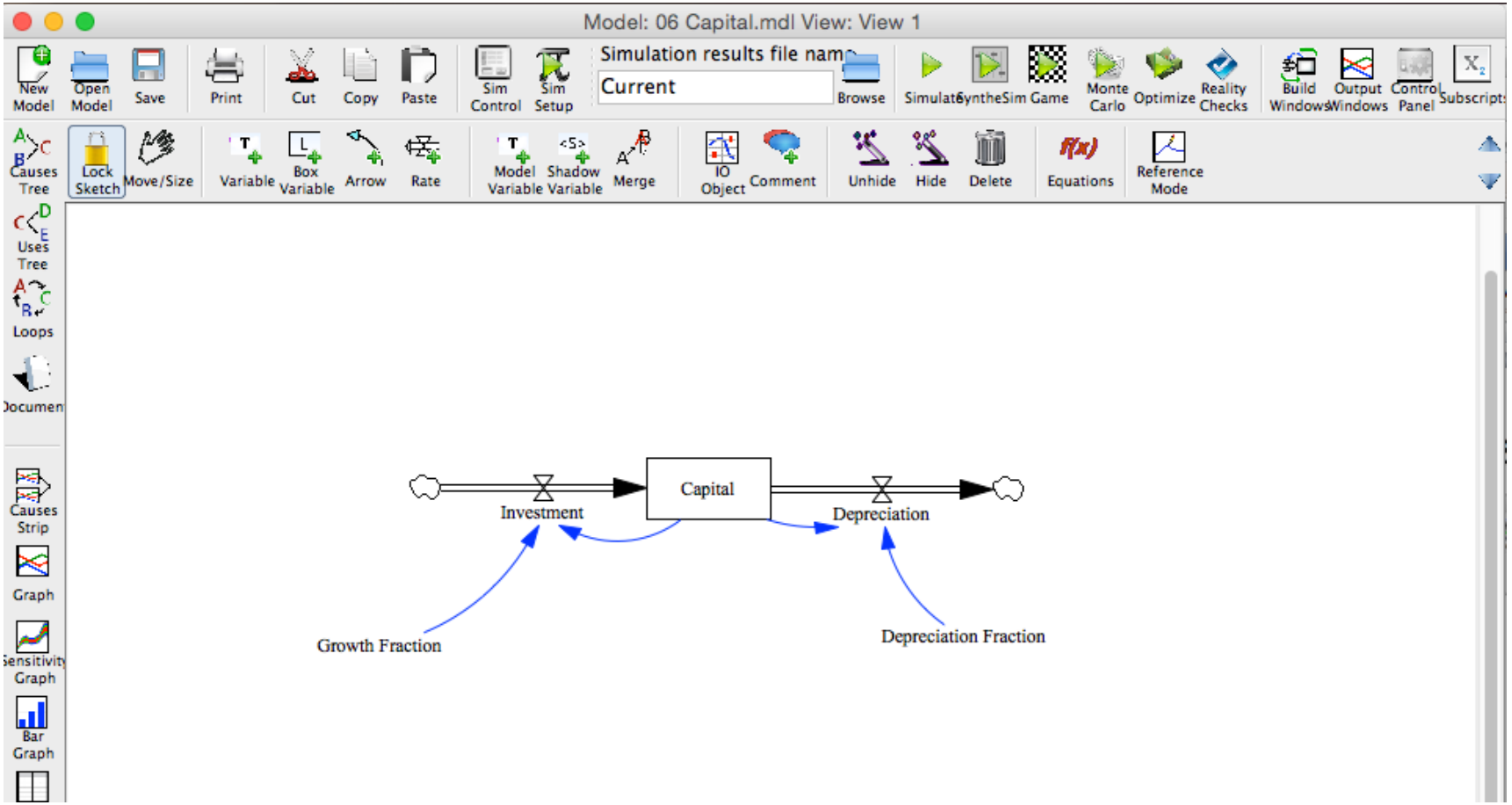
Growth and Depreciation Fractions



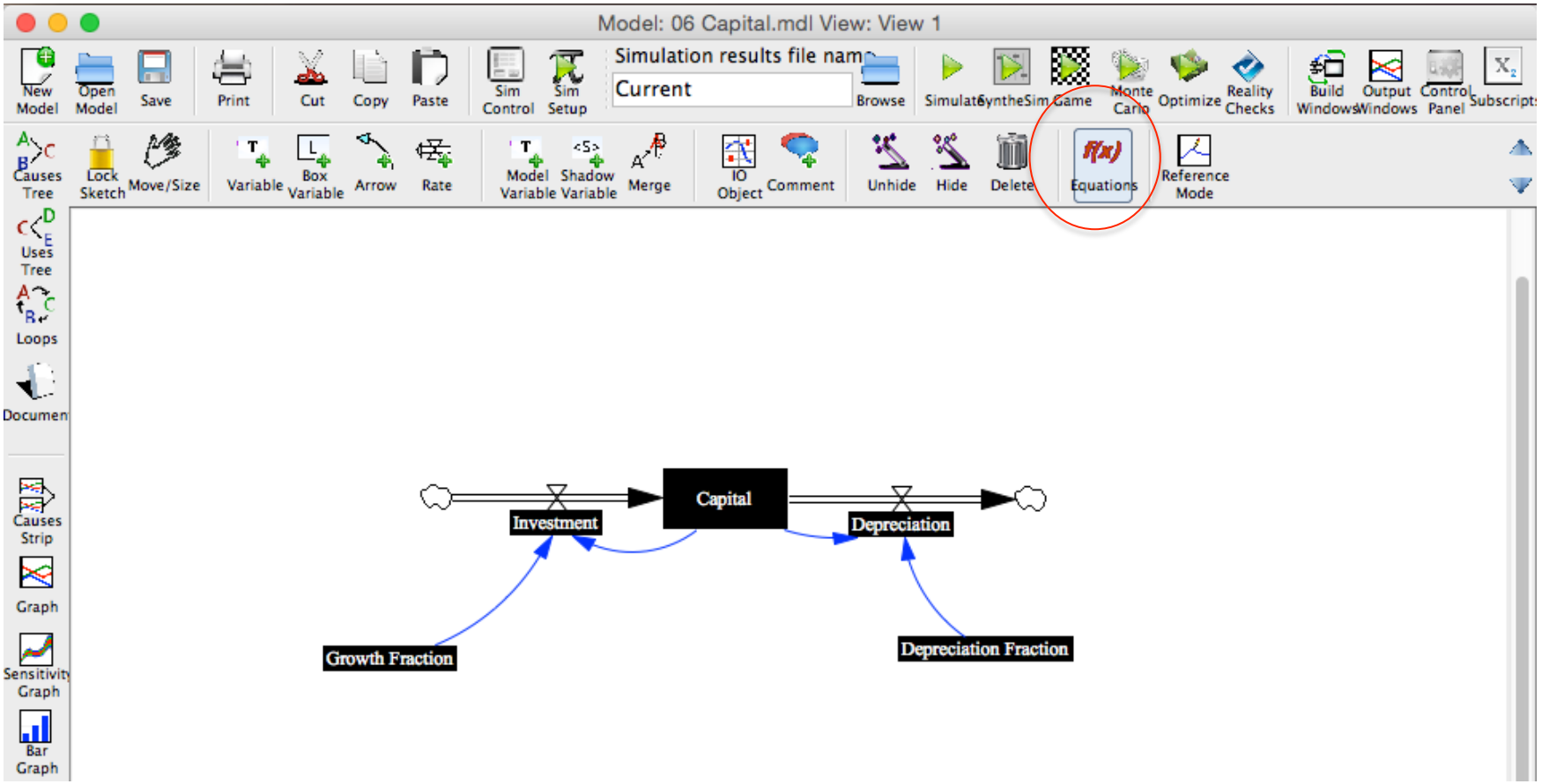
Vensim: Adding Connectors



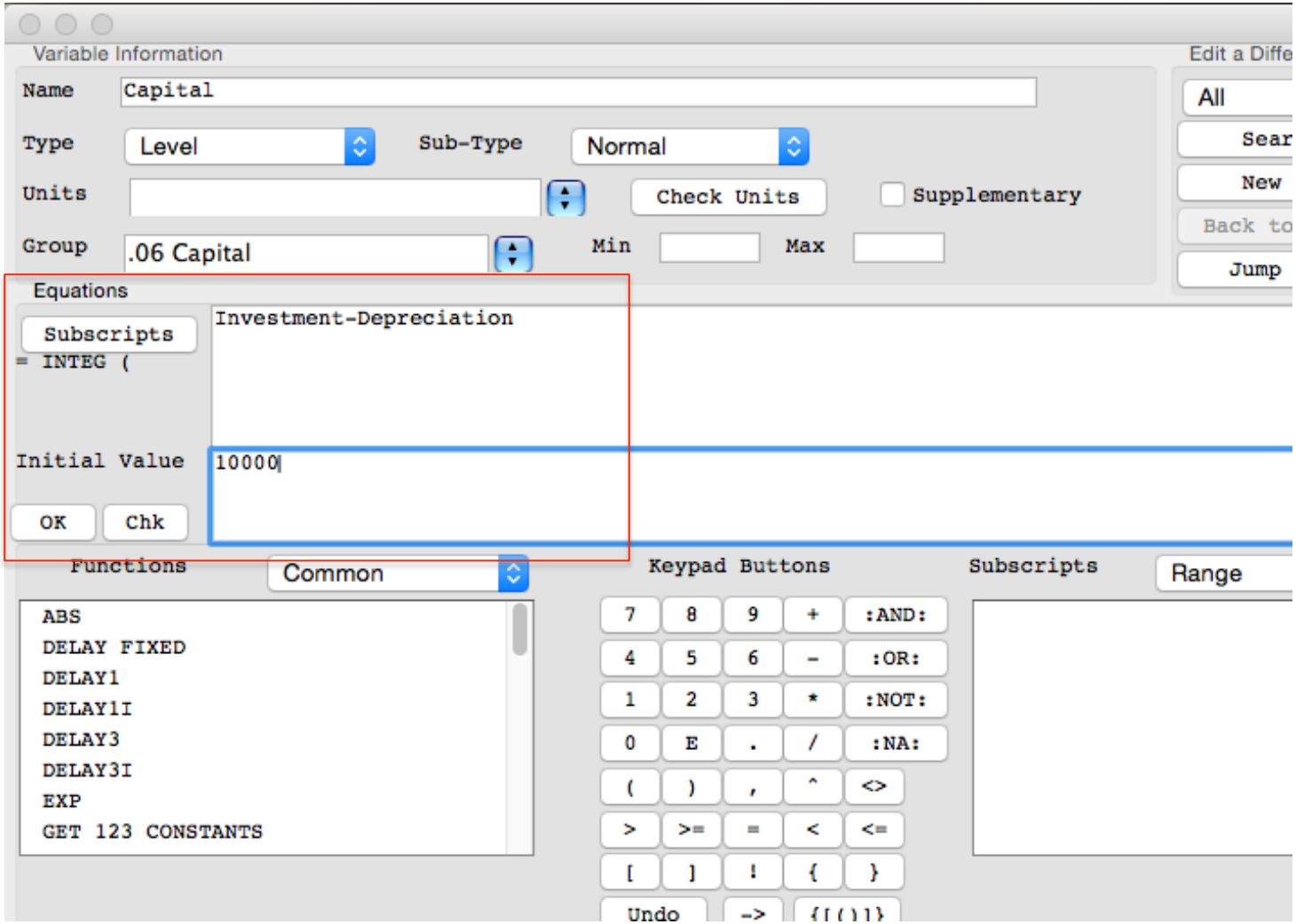
Connect from cause to effect variables



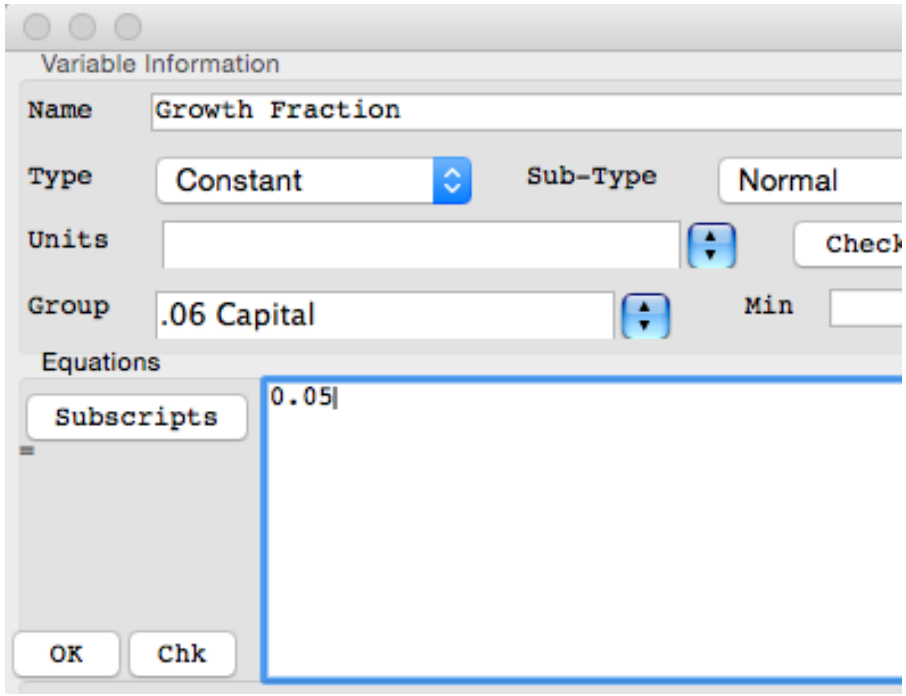
Vensim: Edit equations

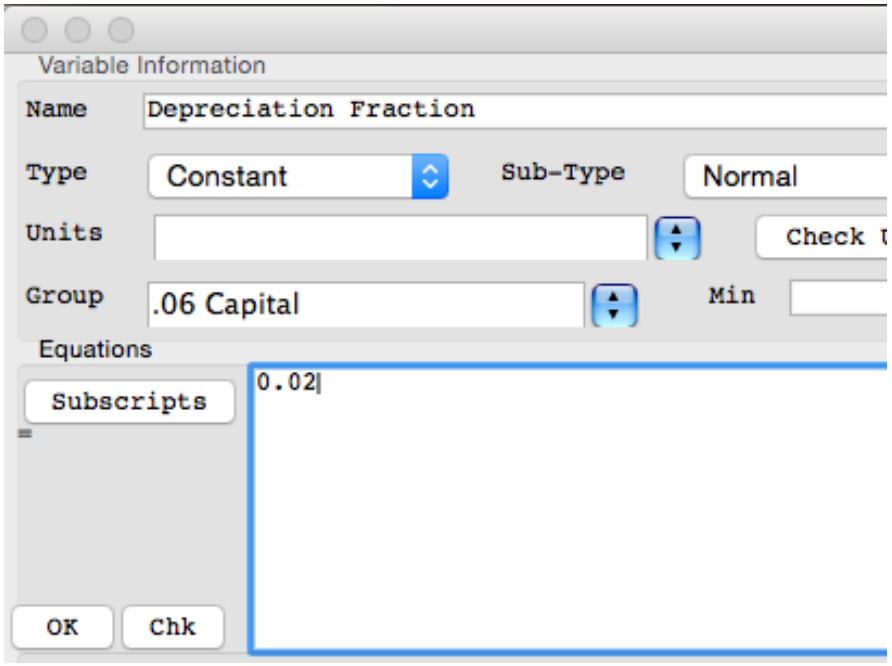


Adding the Capital Equation

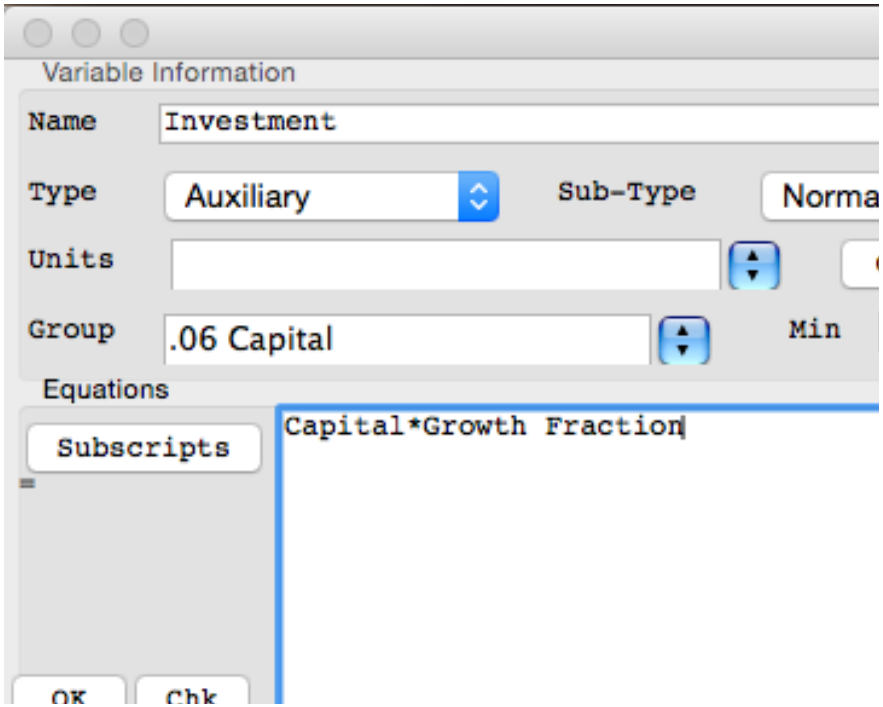


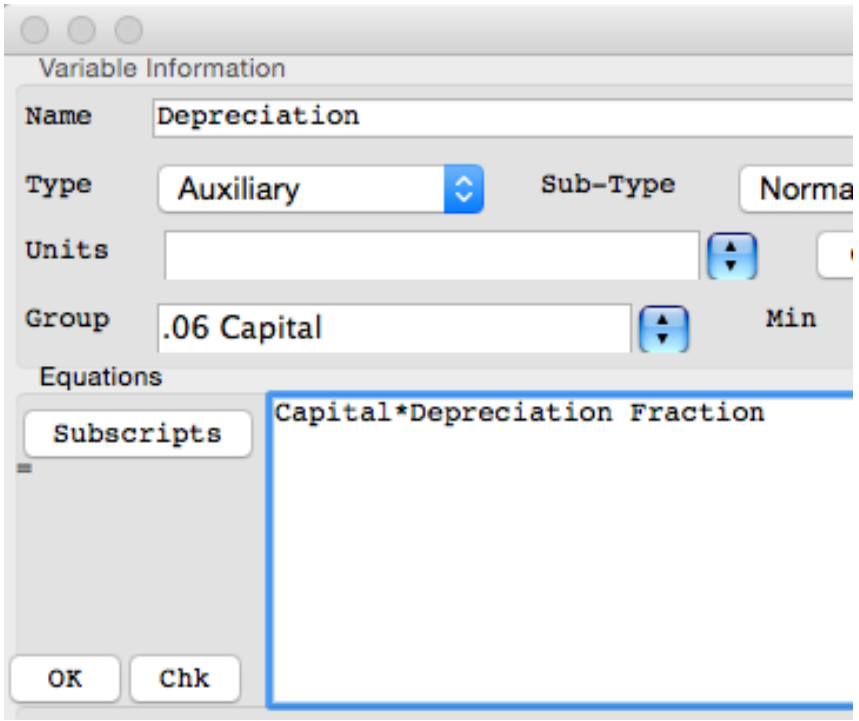
Adding the fractions



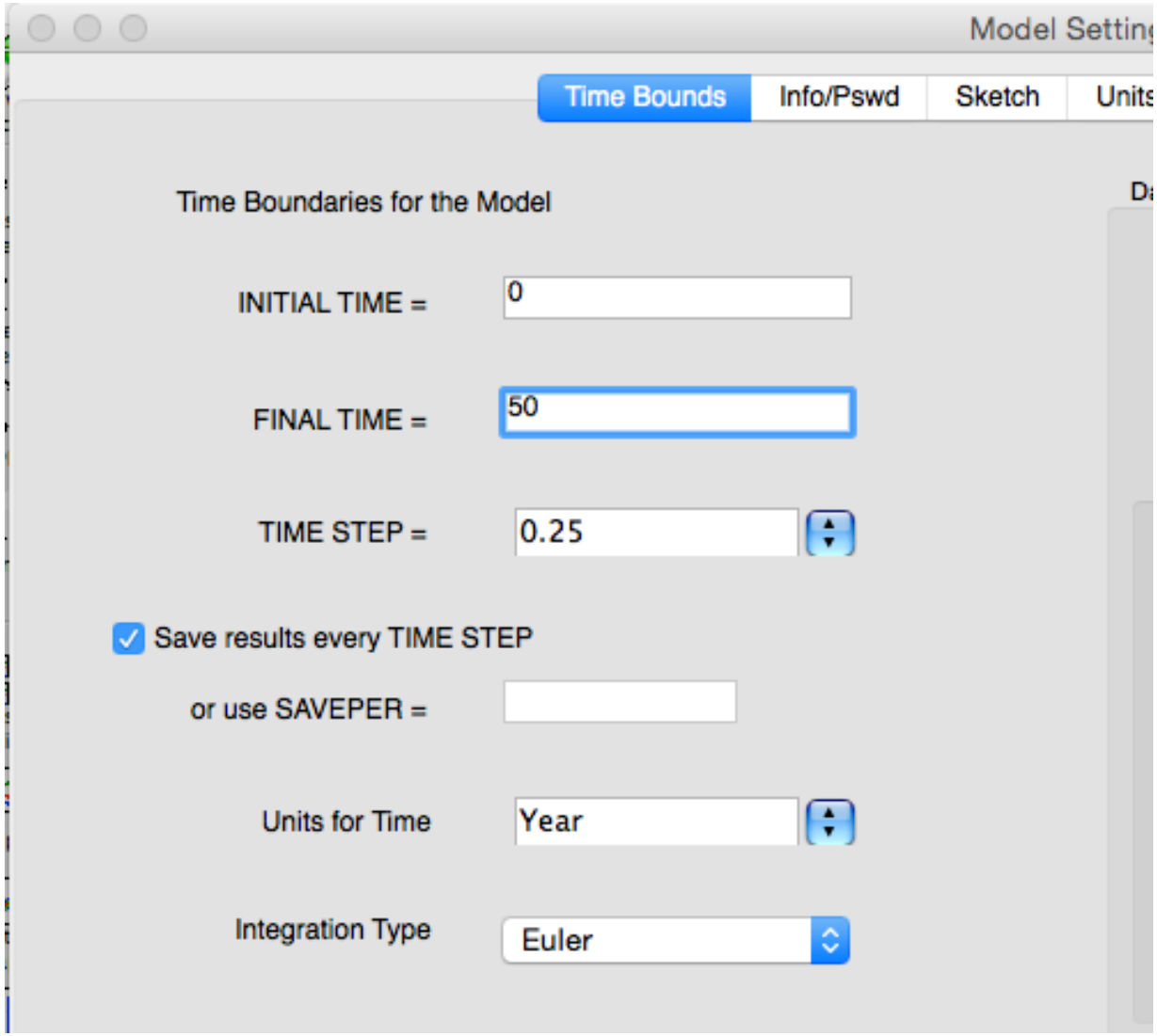


Adding the flows

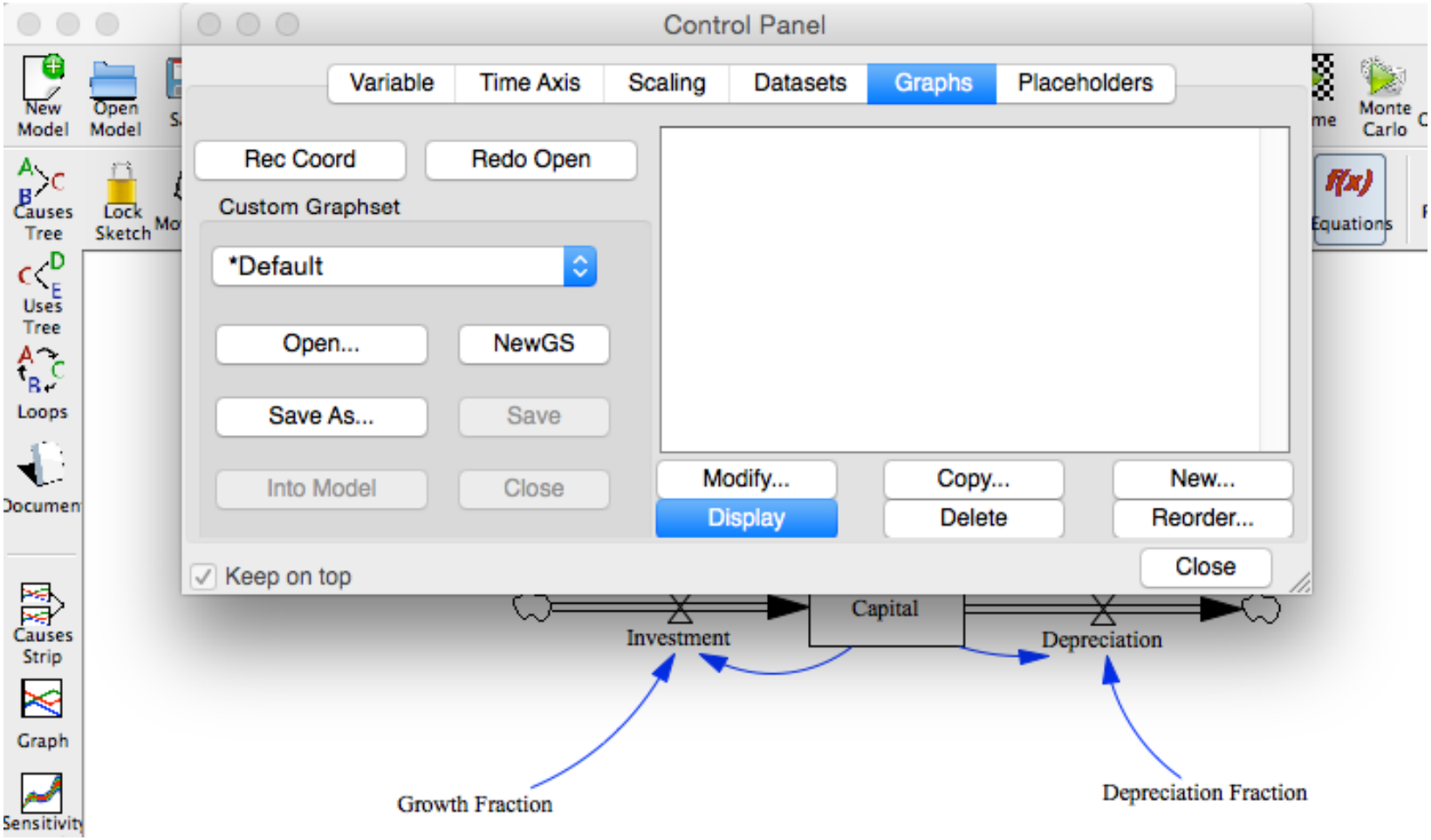




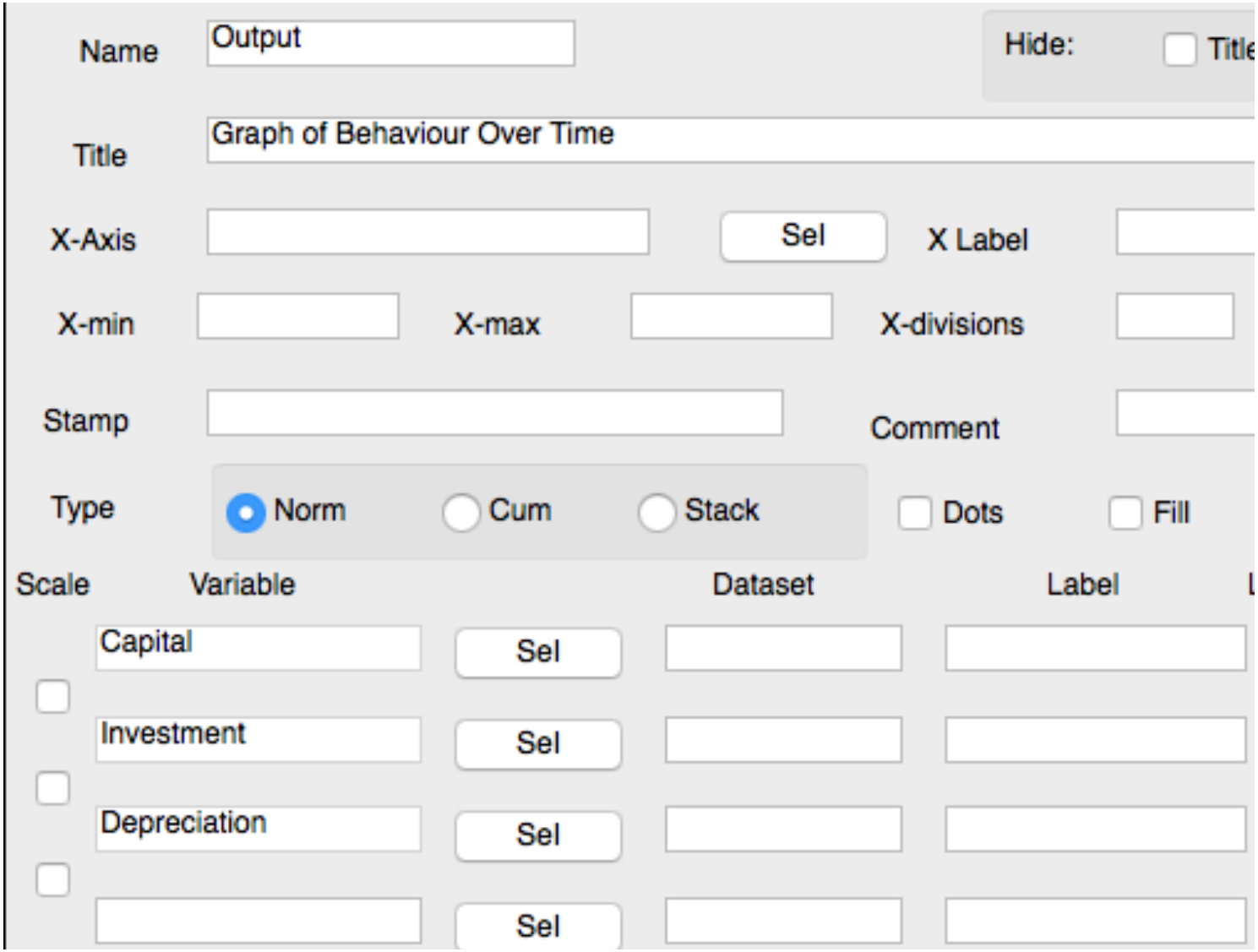
Setting the run parameters



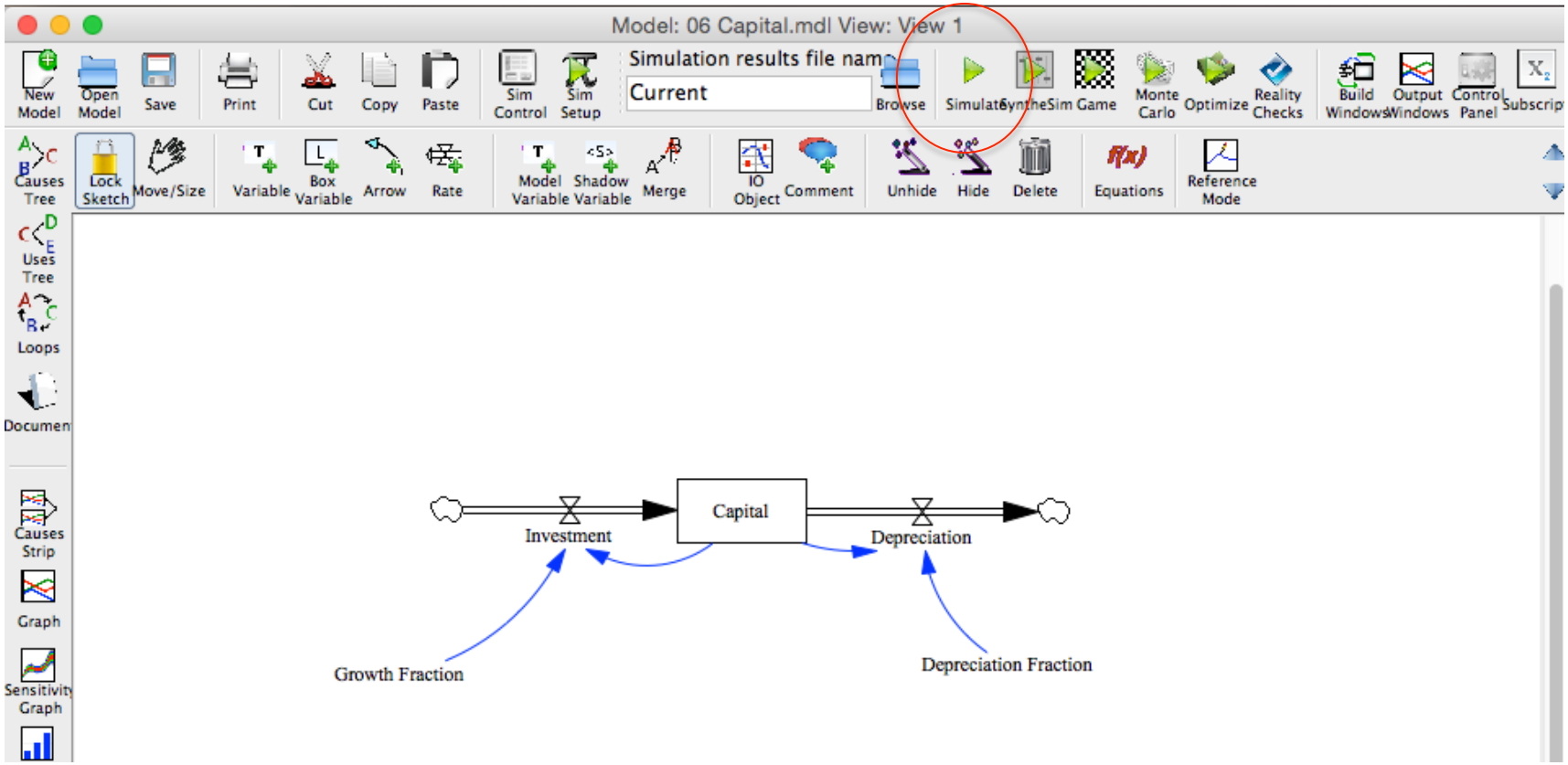
Add a graph


Control Panel
Variable
Time Axis
Scaling
Datasets
Graphs
Placeholders
Rec Coord
Redo Open
Custom Graphset
*Default
Open...
NewGS
Save As...
Save
Into Model
Close
Modify...
Copy...
New...
Display
Delete
Reorder...
Keep on top
Close
New Model
Open Model
Causes Tree
Lock Sketch
Uses Tree
Loops
Causes Strip
Graph
Sensitivity
Monte Carlo
Equations
Capital
Investment
Depreciation
Growth Fraction
Depreciation Fraction

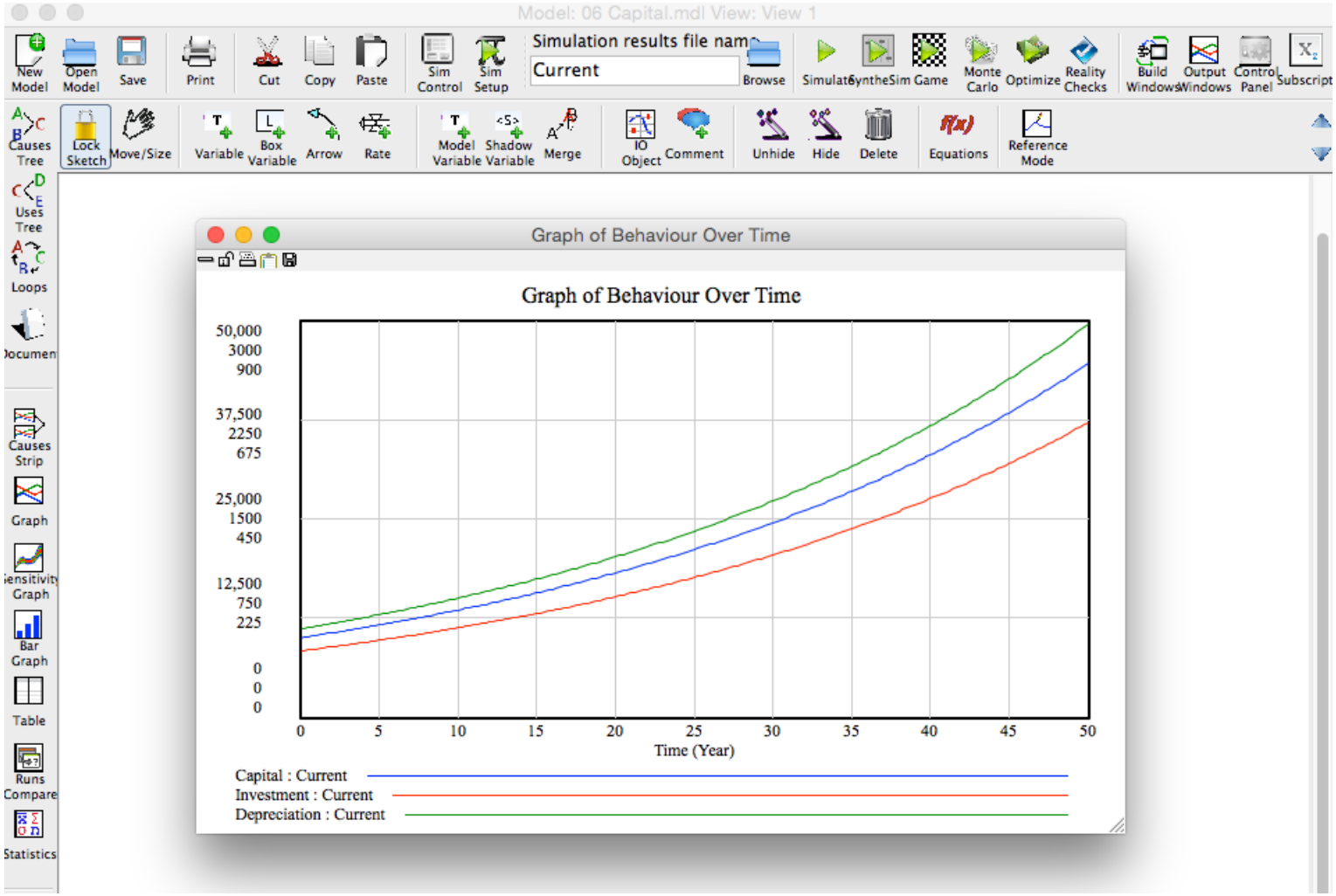
Finishing the graph



Running the simulation



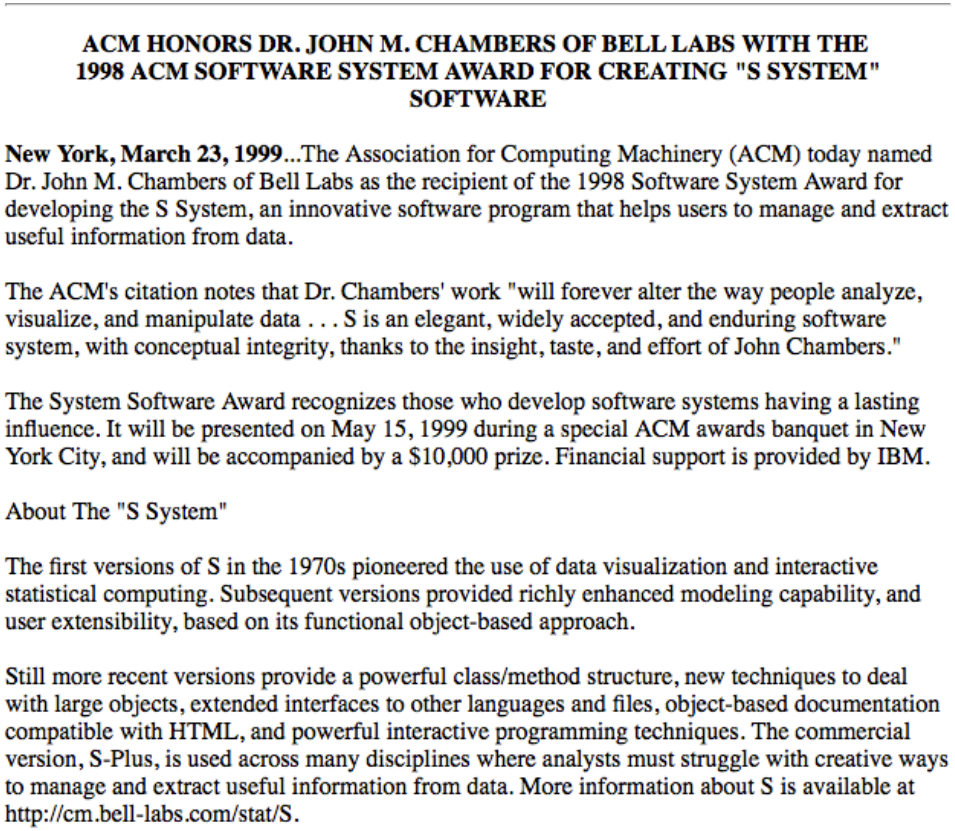
Display the results



Introduction to R

Overview

- R's *mission* is to enable the best and most thorough exploration of data possible (Chambers 2008).
- It is a dialect of the S language, developed at Bell Laboratories
- ACM noted that *S "will forever alter the way people analyze, visualize, and manipulate data"*



ACM HONORS DR. JOHN M. CHAMBERS OF BELL LABS WITH THE 1998 ACM SOFTWARE SYSTEM AWARD FOR CREATING "S SYSTEM" SOFTWARE

New York, March 23, 1999...The Association for Computing Machinery (ACM) today named Dr. John M. Chambers of Bell Labs as the recipient of the 1998 Software System Award for developing the S System, an innovative software program that helps users to manage and extract useful information from data.

The ACM's citation notes that Dr. Chambers' work "will forever alter the way people analyze, visualize, and manipulate data . . . S is an elegant, widely accepted, and enduring software system, with conceptual integrity, thanks to the insight, taste, and effort of John Chambers."

The System Software Award recognizes those who develop software systems having a lasting influence. It will be presented on May 15, 1999 during a special ACM awards banquet in New York City, and will be accompanied by a $10,000 prize. Financial support is provided by IBM.

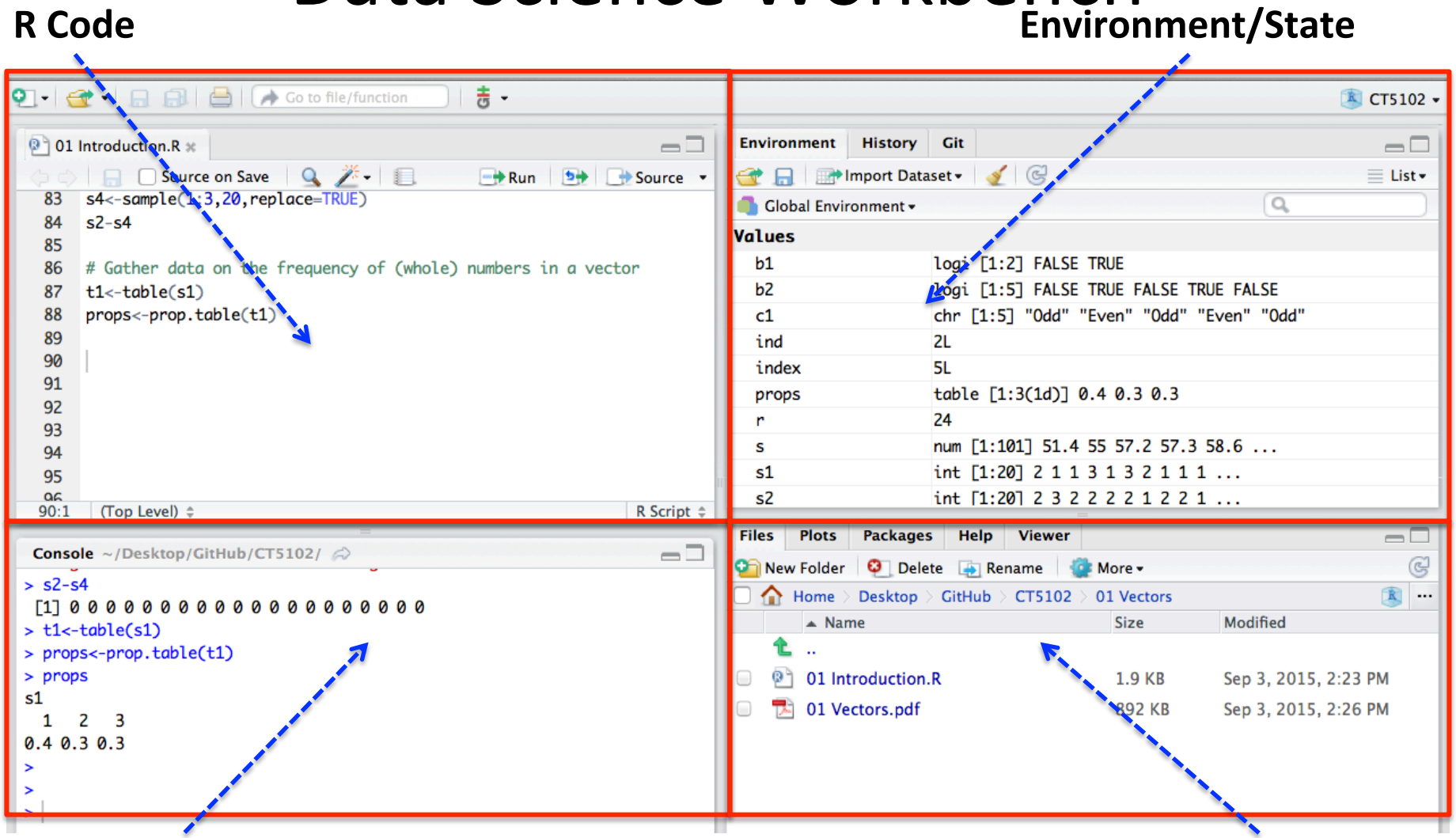
About The "S System"

The first versions of S in the 1970s pioneered the use of data visualization and interactive statistical computing. Subsequent versions provided richly enhanced modeling capability, and user extensibility, based on its functional object-based approach.

Still more recent versions provide a powerful class/method structure, new techniques to deal with large objects, extended interfaces to other languages and files, object-based documentation compatible with HTML, and powerful interactive programming techniques. The commercial version, S-Plus, is used across many disciplines where analysts must struggle with creative ways to manage and extract useful information from data. More information about S is available at http://cm.bell-labs.com/stat/S.

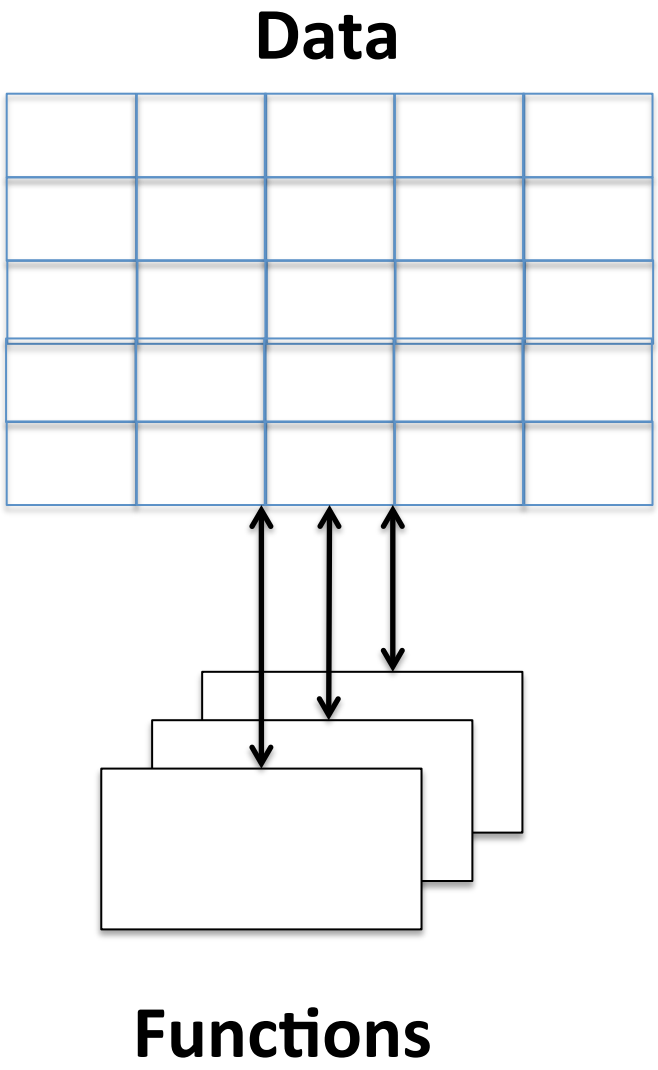
http://www.acm.org/announcements/ss99.html

R Studio: A Data Science Workbench



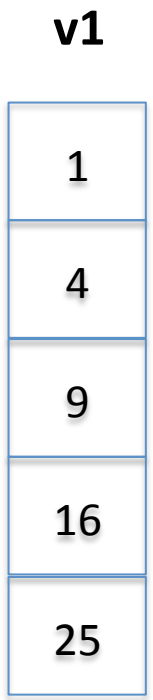
Functional Programming

- R is a functional programming language, where software programs are organized into *functions* that can be called to transform data.
- Users of R should adopt the habit of creating simple functions which will make their work **more effective** and also **more trustworthy** (Chambers 2008).



Data Structures - Vector

- The fundamental data type in R is the vector
- Similar to 1-D arrays in C and Java
- A variable that contains a sequence of elements that have the same data type (Matloff 2009).
- Create using c(e_1, …, e_n)
- ***Assignment statement <-***




```
> v1<-c(1,4,9,16,25)
> v1
[1]  1  4  9 16 25
```

Index

- The concept of an index is powerful in R, as it allows access to individual data elements of a vector, using the square brackets notation.
- In R, unlike programming languages such as C and Java, the index for a vector starts at 1.

```
> v1
[1]  1  4  9 16 25

> v1[1]
[1] 1

> v1[2]
[1] 4

> v1[5]
[1] 25
```

Creating sequences

- The colon operator (:) generates regular sequences within a specified range.

```
> v1<-1:10
> v1
 [1]  1  2  3  4  5  6  7  8  9 10

> v2<-3:13
> v2
 [1]  3  4  5  6  7  8  9 10 11 12 13
```

Using **seq()** function

- Useful function to create sequences, and allows for further detail via the **by** argument

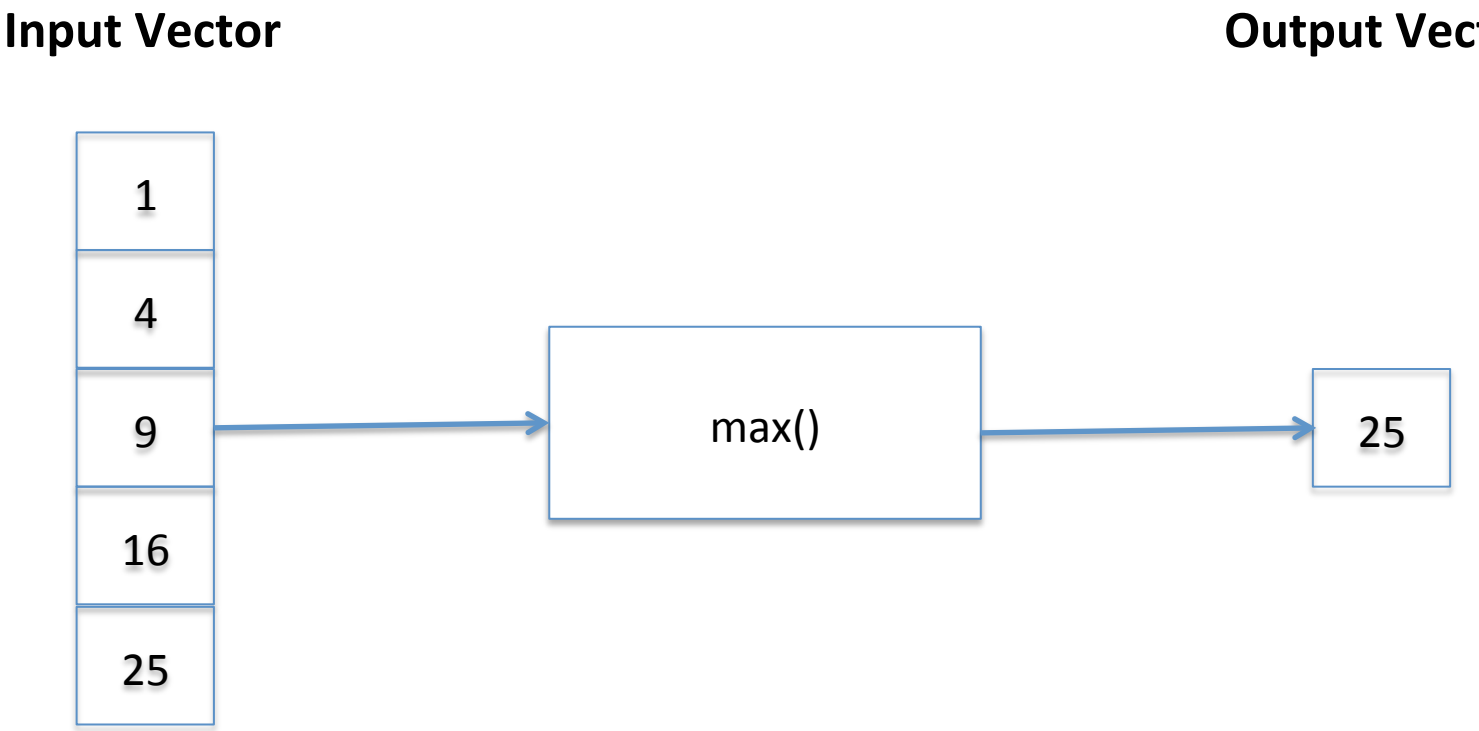
```
> v6<-seq(1,5)
> v6
[1] 1 2 3 4 5

> v7<-seq(1,5,by=.5)
> v7
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

> v8<-seq(1,100,by=20)
> v8
[1]  1 21 41 61 81
```

Sequences as indices

- Sequences can be used an indices for vectors, with the minus sign used for exclusion

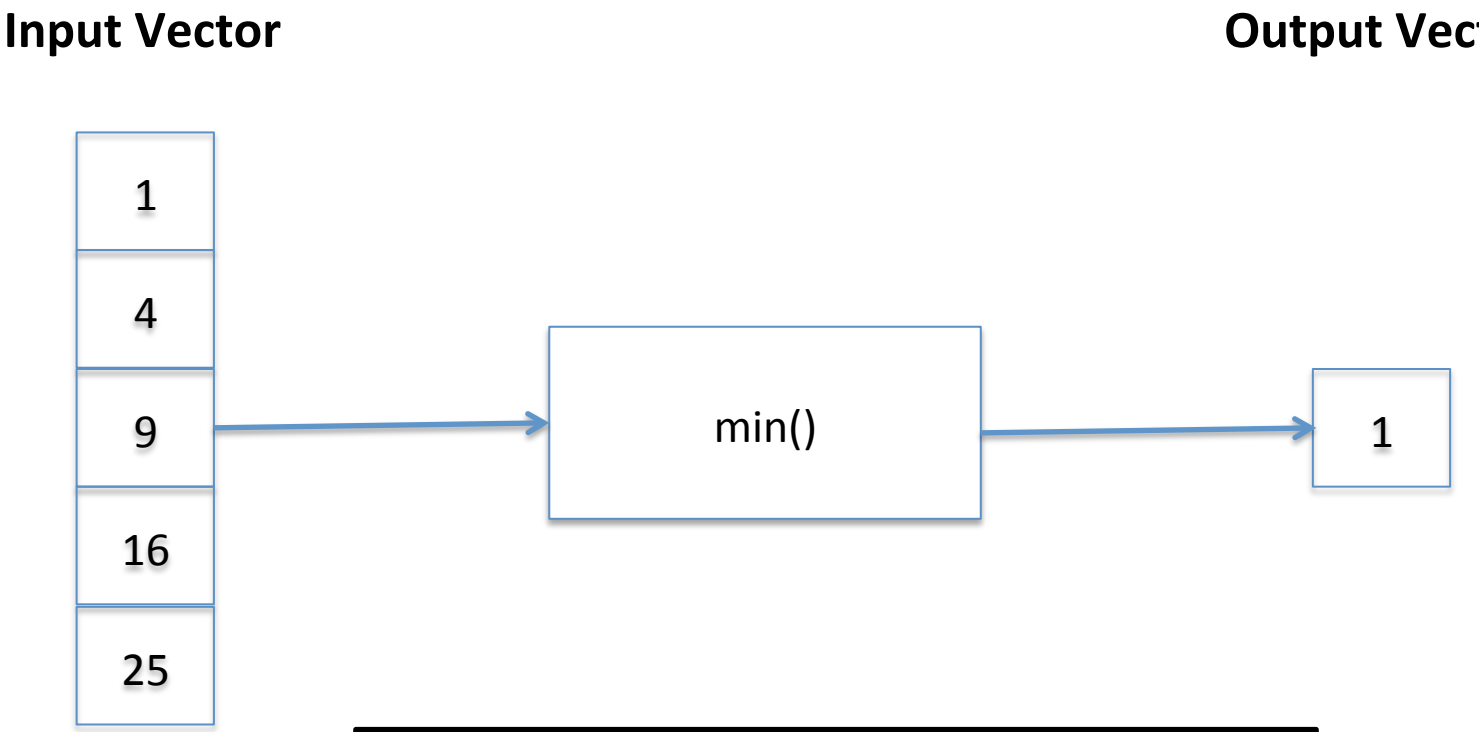
```
> v1
[1]  1  4  9 16 25

> v1[1:2]
[1] 1 4

> v1[-1]
[1]  4  9 16 25

> v1[-(1:3)]
[1] 16 25
```

max function




```
> v1
[1]  1  4  9 16 25
> max(v1)
[1] 25
```

min function




```
> v1
[1]  1  4  9 16 25
> min(v1)
[1] 1
```

Challenge 1.1

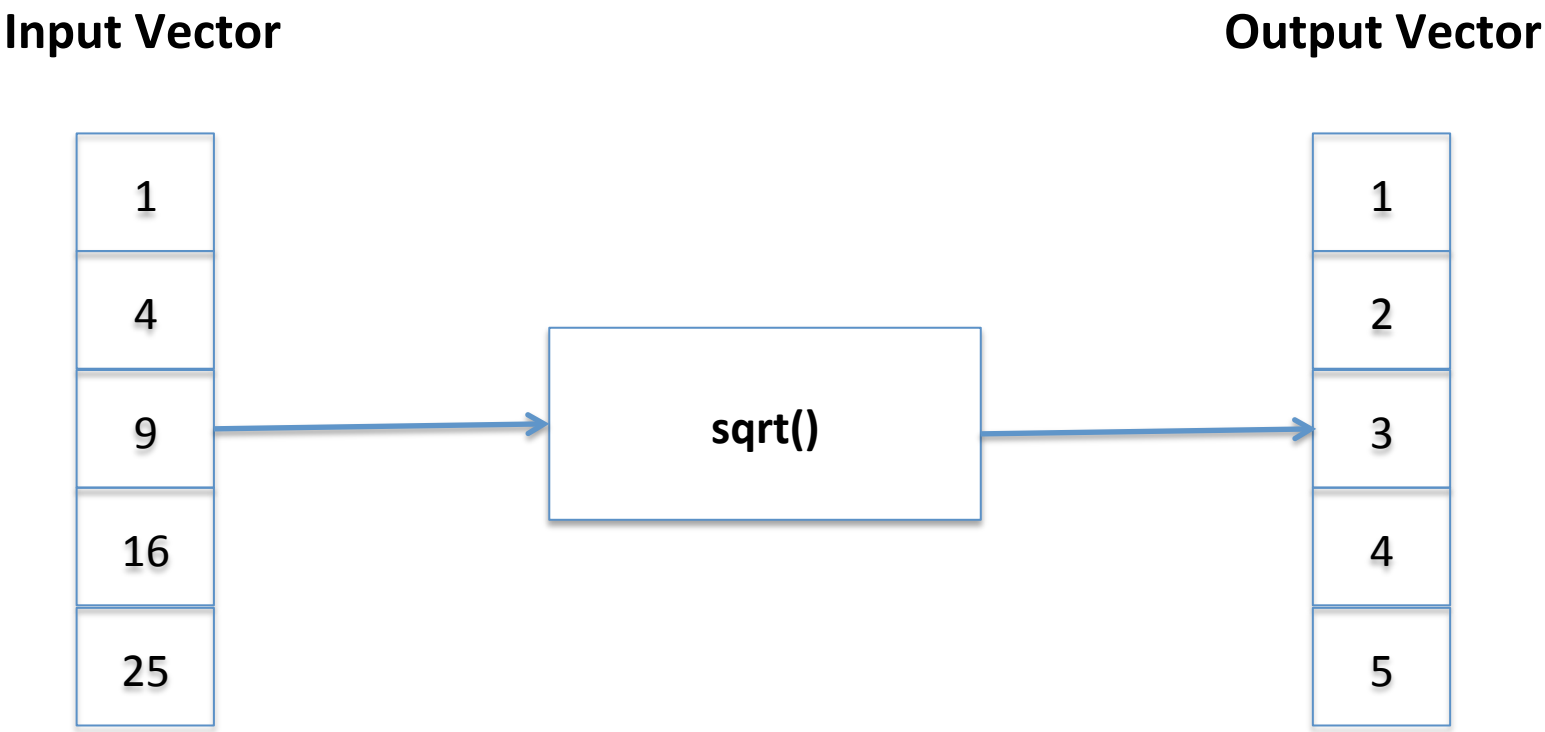
- Create an R vector of squares of 1 to 10
- Find the minimum
- Find the maximum

Vectorization

- A powerful feature of R is that it supports *vectorization*
- Functions can operate on every element of a vector, and return the results of each individual operation in a new vector.

```
> v1
[1] 1 2 3 4 5

> r<-sqrt(v1)

> r
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

Key Idea



Conditional Expressions on Vectors

- Conditional expressions can be applied to vectors, and this operation returns a boolean vector

```
> v1
[1]  1  4  9 16 25

> v1 %% 2==0
[1] FALSE  TRUE FALSE  TRUE FALSE
```

Boolean vectors used as indices

- A target vector can be filtered by the TRUE locations in a boolean vector.

```
> v1
[1]  1  4  9 16 25

> b1<-v1 %% 2 == 1

> b1
[1]  TRUE FALSE  TRUE FALSE  TRUE

> v1[b1]
[1]  1  9 25
```

which()

- Give the TRUE indices of a logical object, allowing for array indices.

```
> v1
[1]  1  4  9 16 25
> ind<-which(v1==4)
> ind
[1] 2
> v1[ind]
[1] 4
```

Naming vector elements

- The elements of a vector can also be given names, and this can be useful in defining parameters for further analysis. The name can be used as an index.

```
> names(v1)<-
c("Var1","Var2","Var3","Var4","Var5")

> v1
Var1 Var2 Var3 Var4 Var5
   1    4    9   16   25

> v1["Var5"]
Var5
  25
```

Vectorized if/else

- Vectors can also be processed using the vectorized **ifelse(b,u,v)** function, which accepts a boolean vector **b** and allocates the element-wise results to be either **u** or **v**.

```
> v1
[1]  1  4  9 16 25

> c1<-ifelse(v1%%2==0,"Even","Odd")

> c1
[1] "Odd"  "Even" "Odd"  "Even" "Odd"
```

Challenge 1.2

- Write 2 “parallel” vectors that store a course code in one, and the number of students in the other. Write a script that returns the number of students (output) in a course (input). Make use of the == operator for string comparison.

```
> "CT102"=="CT101"
[1] FALSE

> "CT102"=="CT102"
[1] TRUE
```

Summary

- Vectors: a key data structure in R
- Vectorization – apply an operation on all elements and return a new vector
- Boolean vectors – very useful!

References

- Chambers, J. 2008. Software for data analysis: programming with R. Springer Science & Business Media. Chicago.
- Chang, W. 2013. R Graphics Cookbook. " O'Reilly Media, Inc." Sebastapol, CA.
- Matloff, N. 2009. The Art of R Programming. No Starch Press, San Francisco, CA.