Use R in STEM Courses

Brian Dennis
Dept Fish and Wildlife
and
Dept Statistical Science
University of Idaho

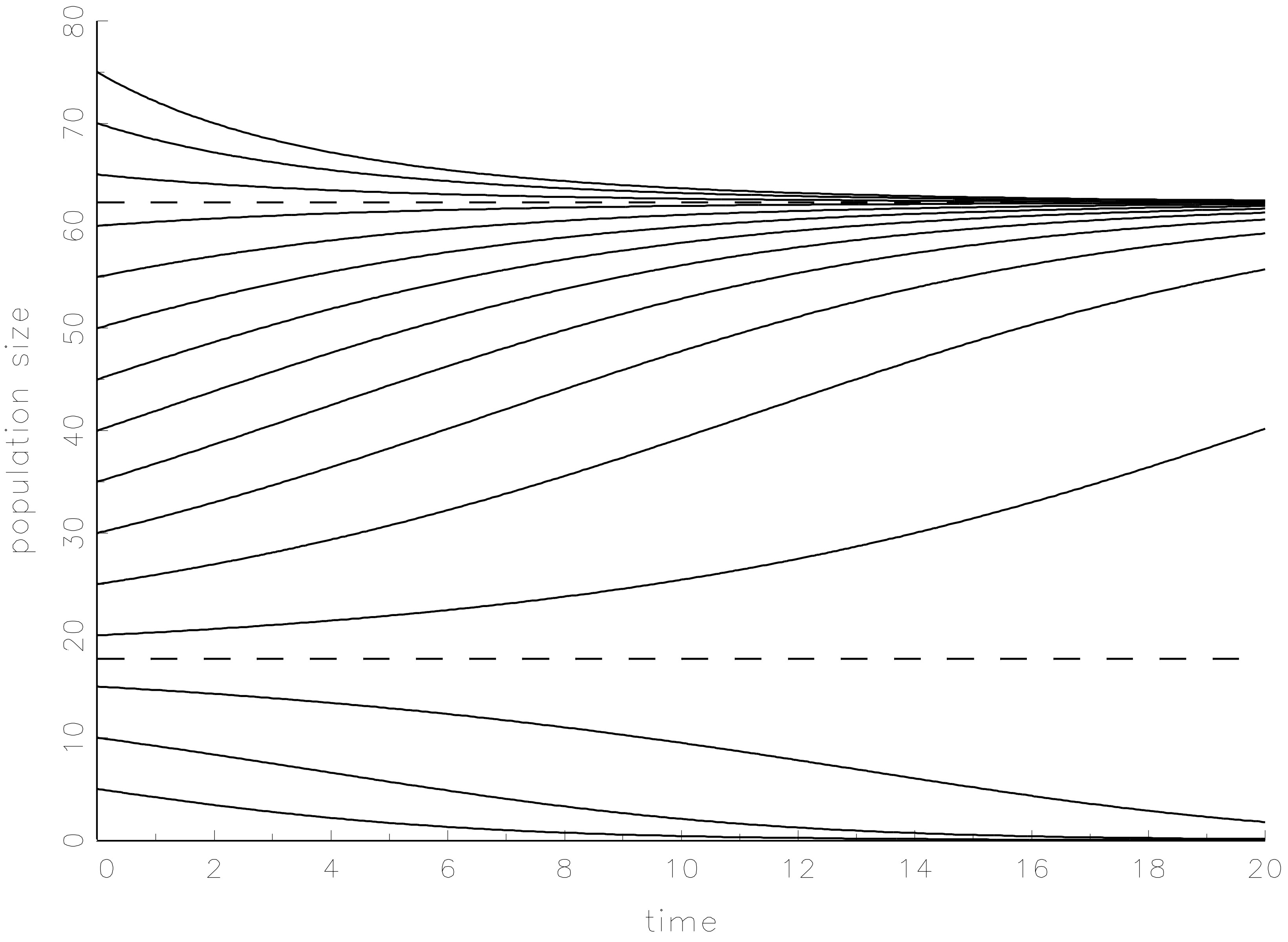
My research interests broadly involve quantitative ecology, the applications of deterministic and stochastic models for improving understanding and solving problems in ecology and natural resource management. Particular topics:

Allee effects (possible extinction due to rare matings or other positive dependence of fitness on population numbers).

Nonlinear dynamics and chaos in ecological populations.

How to connect dynamic mathematical models with ecological data.

What are the stochastic counterparts of deterministic concepts (such as stable & unstable equilibria).



population size
0
10
20
30
40
50
60
70
80
0
2
4
6
8
10
12
14
16
18
20
time

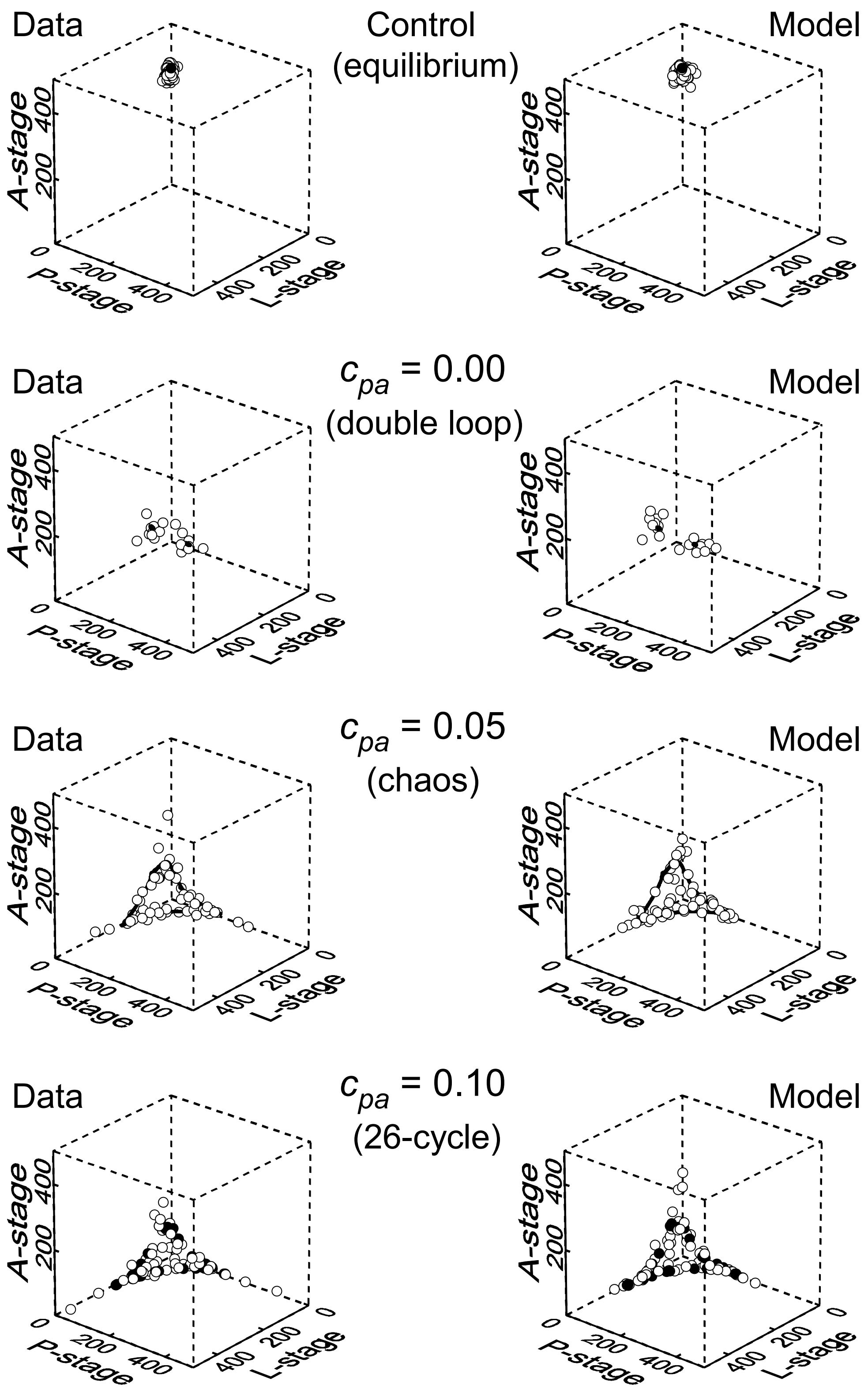

Data
Control
(equilibrium)
Model
Data
c_{pa} = 0.00
(double loop)
Model
Data
c_{pa} = 0.05
(chaos)
Model
Data
c_{pa} = 0.10
(26-cycle)
Model
A-stage
200
400
P-stage
0
200
400
L-stage
0
200
400

Figure 5 (pt. 1)

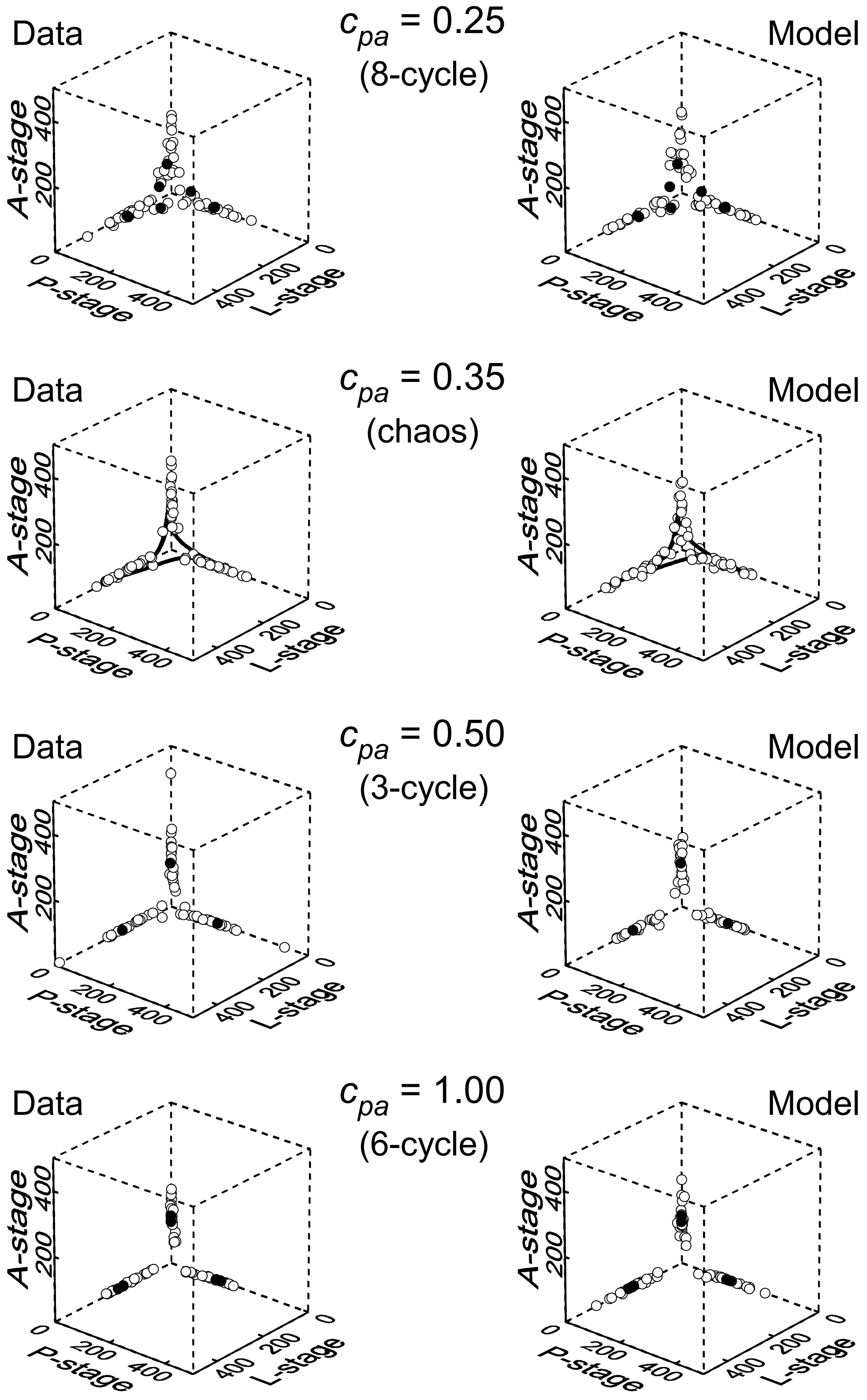
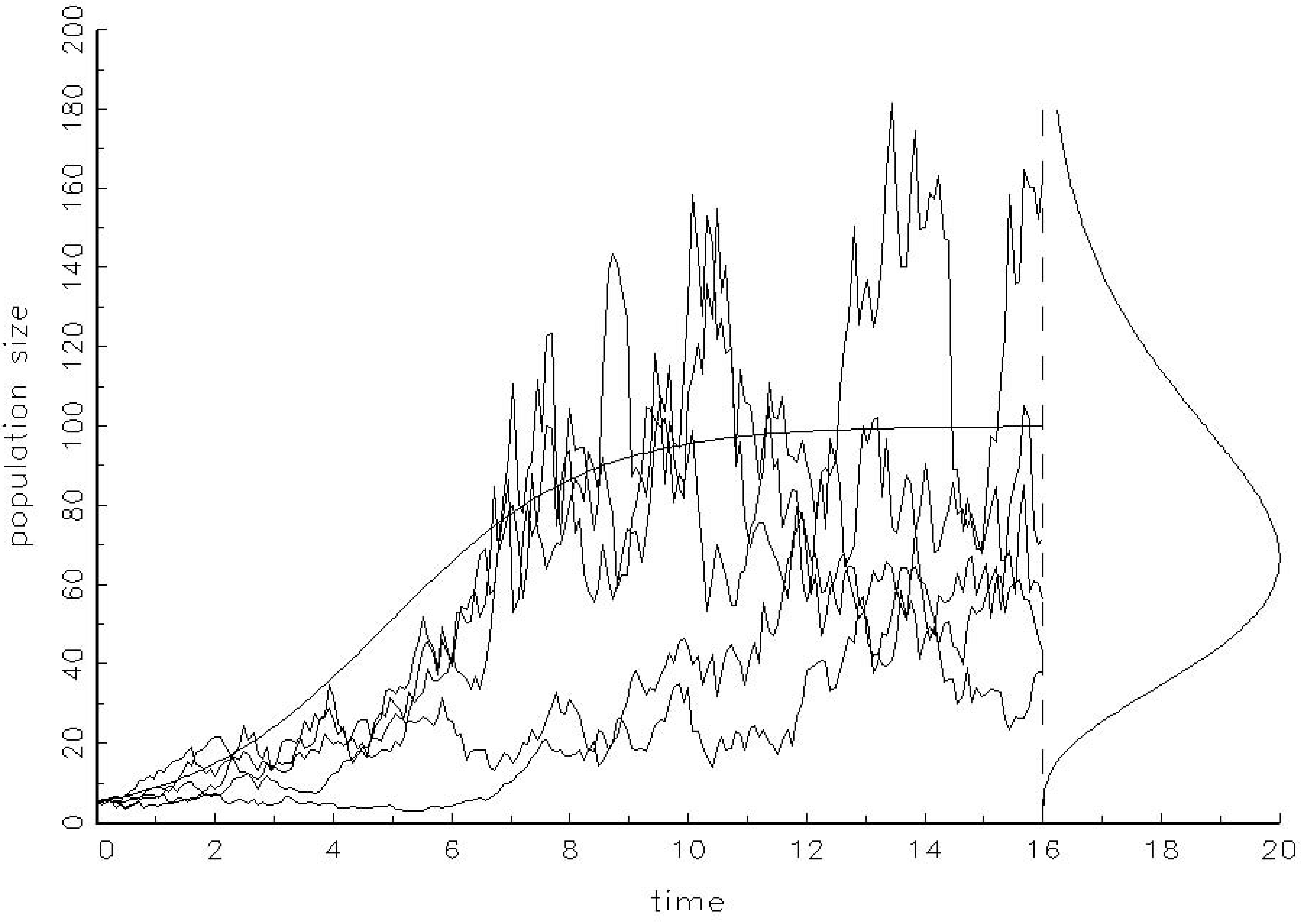

Data
c_{pa} = 0.25
(8-cycle)
Model
Data
c_{pa} = 0.35
(chaos)
Model
Data
c_{pa} = 0.50
(3-cycle)
Model
Data
c_{pa} = 1.00
(6-cycle)
Model
A-stage
P-stage
L-stage
0
200
400

Figure 5 (pt. 2)



population size
0
20
40
60
80
100
120
140
160
180
200
0
2
4
6
8
10
12
14
16
18
20
time

What is R?

R is a software package for scientific calculations, graphics, and statistical analyses.

R is free and open source.

Versions of R are available for Windows, Mac OS, and Linux/Unix operating systems. (download R here: https://www.r-project.org/)

R is in the form of a command-driven programming language with vast statistical, mathematical, and graphical resources built in as pre-programmed functions:

> R has vector and matrix calculations built in (like Matlab).
>
> R features fantastic graphical displays of data.
>
> R has calculations (random numbers, percentiles, probabilities, etc.) built in for many different statistical distributions.
>
> R has special mathematical functions of all sorts built in (complex numbers, gamma & log-gamma function, numerical integration, optimization, etc.).

R has grown an enormous, international group of users (critical mass).

Lots of online forums for getting help.

Many tutorial websites (often posted by university instructors for courses).

Many instructional videos (Youtube especially).

Many books.

Online courses. Even some MOOCs.

Vast numbers of contributed R routines, posted by users as functions, for producing the latest specialized statistical analyses (genetics, finance/investment, multivariate/social sciences, data mining, etc.)

History of R

Bell Laboratories (now Nokia Bell Laboratories) invented/discovered:

radio astronomy, transistor, information theory, charge-coupled device, laser, cosmic microwave background

Unix, C, C++

The S statistical programming language (John Chambers & others).

S-PLUS

MathSoft distributes commercial version of S, called S-PLUS (around 1995).

S-PLUS spreads rapidly in the statistics world; features dazzling (at the time) graphics such as rotating 3D plots and painting of scatter plots, along with statistical programming for numerically intensive research.

R created by Ross Ihaka and Robert Gentleman (later 1990s).

Uses the syntax of S-PLUS & S (syntax can't be copyrighted).

Free, open source, part of the GNU project.

Currently developed by the *R Development Core Team*
(John Chambers is a member).

The R Journal is started in 2001. Many R users by 2005.

"Data analytics" becomes big in the late 2000s; NYT features an
article about R in 2009.

Early reputation was that R was slow (in comparison to Matlab),
but recent versions of R are substantially speeded up. (Microsoft has
issued a free distribution of R that is reportedly quite fast)

R is an *interpreted* programming language rather than a *compiled* language.

Data Analysts Captivated by R's Power

Stuart Isett for The New York Times

R first appeared in 1996, when the statistics professors Robert Gentleman, left, and Ross Ihaka released the code as a free software package.

By ASHLEE VANCE
Published: January 6, 2009

To some people R is just the 18th letter of the alphabet. To others, it's the rating on racy movies, a measure of an attic's insulation or what pirates in movies say.

R competitors:

S-PLUS (expensive).

Matlab (expensive).

Gauss (expensive).

Octave (free, copies Matlab syntax, graphics not as extensive).

Python (free interpreted programming language, with features
that programmers like more than R, but beginner level documentation
is poor, and the library of user-contributed statistical routines is much
smaller).

SAS (expensive but has established itself in the 70s, 80s, and 90s as
the most extensive and powerful statistical analysis and data management
software).

Let's do some R!

Addition, subtraction, multiplication, division, exponentiation:

```
+   -   *   /   ^
```

```
x = (3 + 2*4^2 - 16)/((-1 + 15)*5)
```

Calculations are vectorized:

```
x = c(3,1,-4,0,2)    # c() is "concatenate"
x
y = c(1,-2,3,5,0)
x + y
```

The usual functions are preprogramed and vectorized (elementwise). Specialty functions of all sorts are in the vast library of user-contributed packages.

```
t = seq(0,10,.1)
t
y = exp(-0.34*t)
y
```

Built-in or user-contributed matrix calculations are extensive.

```
A = rbind(c(1,2),c(-1,4))
b = c(5,6)
x=solve(A,b)
x
Ainv=solve(A)
Ainv
A%*%x
```

Extensive library of random numbers and probability calculations for many probability distributions.

```
mu = 500
sigma = 100
rnorm(1000,mu,sigma)
pnorm(700,mu,sigma)
qnorm(.95,mu,sigma)
```

R is fast enough for most everyday things.

```
A = solve(matrix(runif(1000000),1000,1000))
```

Certain R features are especially attractive for STEM teaching

Graphics! Extensive collection of preprogrammed graphical displays.

```
t = seq(0,10,.1)
y = exp(-0.34*t)
plot(t,y,type="l")

mu = 500
sigma = 100
hist(rnorm(1000,mu,sigma))
```

Data input/output, manipulation. R is all about data.

```
gpa.data=read.table("gpa_data.txt",
  header=TRUE)
gpa.data[3:6,2:5]
gpa.data[housing=="g",]
```

Graphics and statistics seamlessly integrated; for instance, the ggplot2 package is built on Leland Wilkinson's "Grammar of Graphics" and utilizes contemporary research on human perceptions.

```
library(ggplot2)

gpa.data=read.table("gpa_data.txt",
  header=TRUE)
gpa.data=na.omit(gpa.data)

# Quick scatterplot.
qplot(UIGPA, ACT, data = gpa.data)
```

```
# Kernel density estimates of UIGPA
# plotted by HOUSING, superimposed.
ggplot(gpa.data,aes(x=UIGPA,color=HOUSING,
   fill=HOUSING)) +
   geom_density(alpha=I(.05))
   # alpha is degree of transparancy
   # (between 0 and 1)

# Facet by two variables (COLLEGE and SEX)
ggplot(gpa.data,aes(x=ACT,y=UIGPA)) +
   geom_point() + stat_smooth(se=FALSE) +
   facet_grid(COLLEGE~SEX)
```

Students can write their own functions for repetitive calculations.

```
# Function to calculate molar mass.
molar.mass = function(x,y) {
  mm = sum(x*y)
  return(mm)
}

num.atoms = c(2,1)
atomic.weights = c(1.008,16.00)
molar.mass(num.atoms,atomic.weights)
```

Loops! Learning loops is the key for students to be able to understand how mathematics is used to model nature (but keep things in discrete time; calculus can be added later).

R syntax for loops is easy:

```
for (i in some.vector) {
    expression 1
    expression 2
        ...
}
```

Here `i` takes the first value in *`some.vector`* for the first time through the loop, `i` takes the second value in *`some.vector`* the second time through the loop, etc.

Example: Spotted Owl Model. Spotted Owls have three age classes, juveniles, subadults (nonreproductive), and adults. A main model used by wildlife scientists is

$$J_{t+1} = f A_t \qquad \text{juveniles}$$

$$S_{t+1} = p_1 J_t \qquad \text{subadults}$$

$$A_{t+1} = p_2 S_t + p_3 A_t \qquad \text{adults}$$

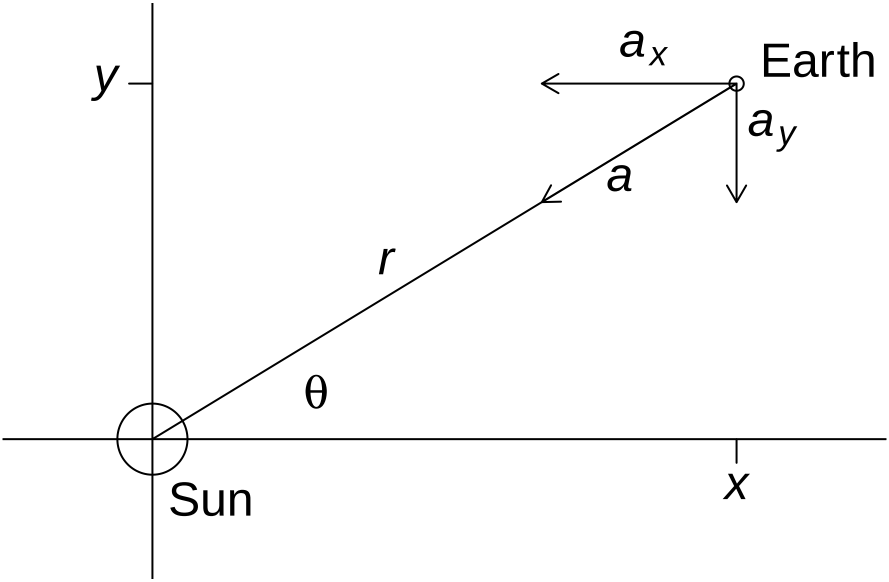
f is adult fecundity, p_1 is juvenile survival probability, p_2 is subadult survival probability, p_3 is adult survival probability.

```
#=======================================
# R script to calculate and plot age class
# sizes through time for an age-structured
# wildlife population, using projection
# equations. Demographic rates for the
# Northern Spotted Owl are from Noon and
# Biles(Journal of Wildlife Management 1990).
#=======================================
num.times=20  # Number of years to project
              # population.
p0=.11      # Age-specific survival and
            # fecundity rates.
p1=.71      #
p2=.94      #
f=.24       #
```

```
J.t=numeric(num.times+1) # Vector of zeros:
                         # will eventually
                         # contain juveniles.
S.t=J.t                  # Vector for
                         # subadults.
A.t=J.t                  # Vector for adults.

J.t[1]=1200 # Initial age class sizes.
S.t[1]=800  #
A.t[1]=2000 #

for (i in 1:num.times) {
   J.t[i+1]=f*A.t[i]           # Recursion
                               # equations
   S.t[i+1]=p0*J.t[i]          # for
```

```
                                         # projection
                                         # of age
                                         # classes.
   A.t[i+1]=p1*S.t[i]+p2*A.t[i] #
}

J.t      # Print the results to the console.
S.t      #
A.t      #

# Plot the age classes through time.
time.t=0:num.times
plot(time.t,J.t,type="l",lty=2,
   xlab="time in years",
   ylab="population size",
   ylim=c(0,2600))
points(time.t,S.t,type="l",lty=5)
points(time.t,A.t,type="l",lty=1)
```

With R, using vectors, loops, functions, and graphics all working together, precalculus physics can be dynamic!
(Just don't mention calculus!)


a_x
Earth
y
a_y
a
r
θ
Sun
x

$$F = ma$$

$$F_{Mm} = -G\frac{Mm}{r^2}$$

$$a_x = -\frac{GMx}{(x^2+y^2)^{3/2}}$$

$$a_y = -\frac{GMy}{(x^2+y^2)^{3/2}}$$

Change in the position of the earth over a small time interval Δt:

$$\Delta v_x = a_x \Delta t$$
$$\Delta v_y = a_y \Delta t$$
$$\Delta x = v_x \Delta t$$
$$\Delta y = v_y \Delta t$$

New earth position coordinates, velocities, time:

$$x_{\texttt{NEW}} = x_{\texttt{OLD}} + \Delta x$$
$$y_{\texttt{NEW}} = y_{\texttt{OLD}} + \Delta y$$
$$v_{x\,\texttt{NEW}} = v_{x\,\texttt{OLD}} + \Delta v_x$$
$$v_{y\,\texttt{NEW}} = v_{y\,\texttt{OLD}} + \Delta v_y$$
$$t_{\texttt{NEW}} = t_{\texttt{OLD}} + \Delta t$$

```
#===========================================
# R program to calculate and plot earth’s
# orbit around the sun, using Newtonian
# gravity.  Sun is at the origin, x and y
# coordinates are measured in astronomical
# units.  Time is measured in sidereal years.
#===========================================

#-------------------------------------------
# 0. Set initial location, velocity and
# direction of the earth here.
#-------------------------------------------
x0=.983291     # Periapsis distance in
               # astronomical units.
y0=0           # Start on x-axis, at
```

```
               # periapsis.
phi=pi/2       # Initial direction angle of
               # the earth's orbit.
v0=30.3        # Earth's initial velocity at
               # periapsis, km/s.

#-----------------------------------------
# 1. Set number of time increments and end
# time for trajectory calculation.
#-----------------------------------------
n.int=10000    # Number of time increments
               # per earth year to be used.
t.end=1        # Number of years duration of
               # trajectory.
```

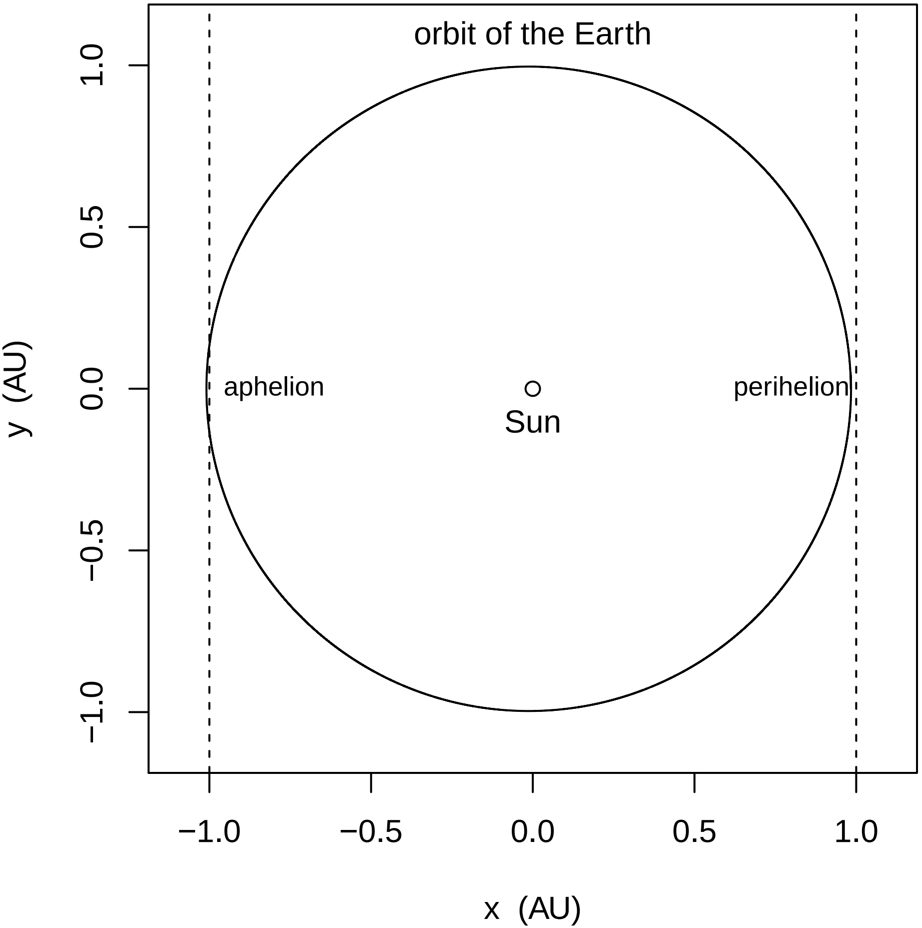
```
#-------------------------------------------
# 2. Various constants gleaned from astronomy
books or online.
#-------------------------------------------
G=6.67428e-11        # Gravitational
                     # constant, m^3/(kg*s^2).
m=6.0e24             # Mass of the earth, kg.
M=m*3.33e5           # Mass of the sun.
km.per.AU=149.6e6           # One astronomical
                            # unit, km.
sec.per.year=31558149.540  # Seconds in a
                            # sidereal year.
```

```
#-------------------------------------------
# 3. Do all calculations in units convenient
# for plotting.
#-------------------------------------------
me=m/m      # Mass of earth expressed in
            # earth units.
Me=M/m      # Sun's mass in earth units.
G=G*m*sec.per.year^2/((km.per.AU^3)*(1000^3))
                        # New units of G
                        # are AU^2/(me*yr^2).
v0=v0*sec.per.year/km.per.AU  # New units of
                              # velocity are
                              # AU/yr.
```

```
#-------------------------------------------
# 4. Initialize various quantities.
#-------------------------------------------
v.x0=v0*cos(phi)    # Earth's velocity,
                    # x-component, at
                    # periapsis.
v.y0=v0*sin(phi)    # Earth's velocity,
                    # y-component, at
                    # periapsis.
dt=1/n.int          # Duration of one tiny
                    # time interval (delta t).
tot.int=round(t.end*n.int);  # Total number
                             # of tiny
                             # intervals
                             # (must be an
                                # integer).

#-------------------------------------------
# 5.  Pre-allocate vectors which will store
# all the values.  Note:  x is x-position, y
# is y-position, v.x is x-direction velocity,
# v.y is y-direction velocity, t is time.
#-------------------------------------------
x=numeric(tot.int+1)    #  x starts as a
                        # vector of tot.int+1
                        # zeros.
                        # The "plus one" is
                        # for the initial
                        # condition.
y=x            # Allocate y the same as x.
```

```
v.x=x           # Allocate v.x the same as x.
v.y=x           # Allocate v.y the same as x.
t=x             # Allocate t the same as x;

#-------------------------------------------
# 6.  Insert the initial conditions into the
Vectors.
#-------------------------------------------
x[1]=x0          # Initial value of x-position.
y[1]=y0          # Initial value of y-position.
v.x[1]=v.x0      # Initial value of x-direction
                 # velocity.
v.y[1]=v.y0      # Initial value of y-direction
                 # velocity.
t[1]=0             # Initial value of time.
```

```
c=G*Me                # Pre-calculate a constant
                      # that appears repeatedly
                      # in the equations.

#-------------------------------------------
# 7.  Loop to calculate trajectory.
#-------------------------------------------
for (i in 1:tot.int) {
  dx=v.x[i]*dt       # Change in x.
  dy=v.y[i]*dt       # Change in y.
  dv.x=-c*x[i]/(x[i]^2+y[i]^2)^(3/2)*dt
             # Change in x-velocity.
  dv.y=-c*y[i]/(x[i]^2+y[i]^2)^(3/2)*dt
             # Change in y-velocity.
  x[i+1]=x[i]+dx       # New value of x.
  y[i+1]=y[i]+dy      # New value of y.
  v.x[i+1]=v.x[i]+dv.x  # New value of x-
                        # velocity.
  v.y[i+1]=v.y[i]+dv.y  # New value of y-
                        # velocity.
  t[i+1]=t[i]+dt      # New value of time.
}

#-------------------------------------------
# 8.  Plot the trajectory.
#-------------------------------------------
par(pin=c(4,4))  # Equal screen size in x and
                 # y directions.
plot(x,y,type="l",xlim=c(-1.1,1.1),
   ylim=c(-1.1,1.1)) # Line plot of y vs x.
```


orbit of the Earth
aphelion
perihelion
Sun
x (AU)
y (AU)
−1.0
−0.5
0.0
0.5
1.0

Why use R in STEM classes?

(1) With R, scientific calculations and graphs are fun and easy to produce.

(2) R could be used across a wide variety of STEM courses, promoting the integration of STEM subjects. R skills would follow a student from course to course.

(3) For students, R is probably the most universally available computational tool (aside from counting on fingers). R runs on most platforms, so R can be made instantly available to every student in every course. No additional proprietary software or calculators have to be purchased. Once R is installed on a machine, the internet is not needed to use R.

(4) R invites collaboration among students.

Graphical displays are big, and easily saved as files.

R expressions are easily shared and texted.

R scripts (lists of expressions) are easily saved, modified, shared, emailed. *R builds on itself*.

(5) R skills follow a student to college and professional life.

(6) There is excellent, free documentation available online (including videos).

(7) Inexpensive tutorial and reference books are available.

Calculators

Graphing calculators are hard to learn: a student rarely actually masters the daunting sequence of keystrokes needed to produce even the simplest graph.

The graphs themselves are low-resolution toys that are near-useless.

Graphing calculators are expensive.

Spreadsheets

The most widely used spreadsheet is proprietary.

The quality of the graphical displays produced by the proprietary spreadsheet is reviled by many working scientists, because the common default graphing options are unacceptable in scientific journals (axis label sizes and fonts, line thicknesses, tic marks, plotting symbols, and so on).

For many years, the proprietary spreadsheet has had statistical routines known to be poorly programmed (see http://practicalstats.com/xlsstats/excelstats.html).

One can use an open source spreadsheet, but then one must spend time dealing with installation, compatibility issues and uneven documentation.

On the web

A self-learn tutorial at the National Center for Ecological Analysis and Synthesis:
http://www.nceas.ucsb.edu/files/scicomp/Dloads/RProgramming/BestFirstRTutorial.pdf

R for biologists at U Tennessee:
http://www.nimbios.org/products/RforBiologistv1.1.pdf

R tutorial at Illinois State U:
http://math.illinoisstate.edu/dhkim/rstuff/rtutor.html

R resource website at UCLA:
http://www.ats.ucla.edu/stat/r/

R tutorial (with videos) at U Colorado/Denver:
http://math.ucdenver.edu/RTutorial/

Intro to R at the R Project website:
http://cran.r-project.org/doc/manuals/R-intro.html

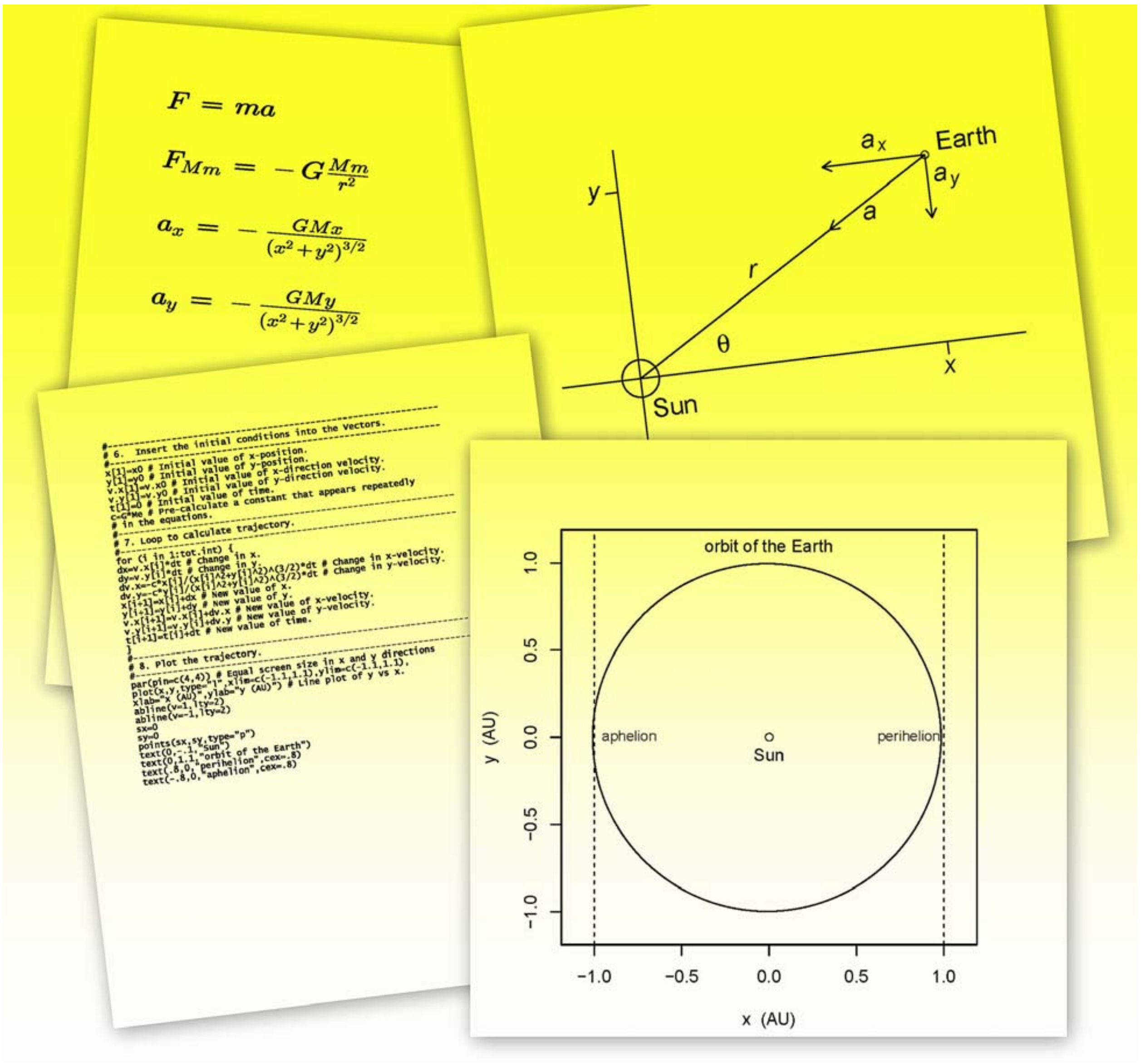
Introductory-level instructional books about R

R in Action
http://www.manning.com/kabacoff/

R in a Nutshell
http://shop.oreilly.com/product/0636920022008.do

R for Dummies
http://www.dummies.com/store/product/R-For-Dummies.productCd-1119962846.html

The R Student Companion (for high school!)
http://www.crcpress.com/product/isbn/9781439875407



$$F = ma$$

$$F_{Mm} = -G\frac{Mm}{r^2}$$

$$a_x = -\frac{GMx}{(x^2+y^2)^{3/2}}$$

$$a_y = -\frac{GMy}{(x^2+y^2)^{3/2}}$$

```
#----------------------------------------------------------
# 6.  Insert the initial conditions into the vectors.
#----------------------------------------------------------
x[1]=x0 # Initial value of x-position.
y[1]=y0 # Initial value of y-position.
v.x[1]=v.x0 # Initial value of x-direction velocity.
v.y[1]=v.y0 # Initial value of y-direction velocity.
t[1]=0 # Initial value of time.
c=G*Me # Pre-calculate a constant that appears repeatedly
# in the equations.
#----------------------------------------------------------
# 7. Loop to calculate trajectory.
#----------------------------------------------------------
for (i in 1:tot.int) {
dx=v.x[i]*dt # Change in x.
dy=v.y[i]*dt # Change in y.
dv.x=-c*x[i]/(x[i]^2+y[i]^2)^(3/2)*dt # Change in x-velocity.
dv.y=-c*y[i]/(x[i]^2+y[i]^2)^(3/2)*dt # Change in y-velocity.
x[i+1]=x[i]+dx # New value of x.
y[i+1]=y[i]+dy # New value of y.
v.x[i+1]=v.x[i]+dv.x # New value of x-velocity.
v.y[i+1]=v.y[i]+dv.y # New value of y-velocity.
t[i+1]=t[i]+dt # New value of time.
}
#----------------------------------------------------------
# 8. Plot the trajectory.
#----------------------------------------------------------
par(pin=c(4,4)) # Equal screen size in x and y directions
plot(x,y,type="l",xlim=c(-1.1,1.1),ylim=c(-1.1,1.1),
xlab="x (AU)",ylab="y (AU)") # Line plot of y vs x.
abline(v=1,lty=2)
abline(v=-1,lty=2)
sx=0
sy=0
points(sx,sy,type="p")
text(0,-.1,"Sun")
text(0,1.1,"orbit of the Earth")
text(.8,0,"perihelion",cex=.8)
text(-.8,0,"aphelion",cex=.8)
```

Happy R-ing!
These slides can be downloaded at:

webpages.uidaho.edu/~brian/reprints/BDennis_reprint_list.htm

Brian Dennis
brian@uidaho.edu