LECTURE 8: MORE ON DYNAMIC MODELS

USING THE ODE SOLVER

To integrate a differential equation using ODE solver in R requires
*initial conditions
*differential equation
*parameters

ADDITIONAL PARAMETERS

With the ODE solver we can add additional parameters to the function

- parameters must all be input as a single list
- similar to how we return multiple outputs from a function

see example below..lets add a carrying capacity

R CODE WITH CARRYING CAPACITY

```r
library(deSolve)


source(here("R/dexppop_play.R"))

dexppop_play
```

```
function (time, P, parms)
{
    dexpop <- parms$r * P
    dexpop <- ifelse(P > parms$carry_capacity, 0, dexpop)
    return(list(dexpop))
}
```

```r
# create parameter list
initalrabbits <- 2
years <- seq(from = 1, to = 100, by = 2)

newparms <- list(r = 0.03, carry_capacity = 300)

# apply solver
results <- ode(initalrabbits, years, dexppop_play, newparms)
head(results)
```

```
     time        1
[1,]    1 2.000000
[2,]    3 2.123677
[3,]    5 2.254993
[4,]    7 2.394435
[5,]    9 2.542500
[6,]   11 2.699720
```

```r
# add more meaningful names
colnames(results) <- c("year", "P")
```

PLOT

```r
# plot
ggplot(as.data.frame(results), aes(year, P)) +
  geom_point() +
  labs(y = "Population", "years")
```

TRY AGAIN WITH DIFFERENT PARAMETERS

```r
# same initial condtions
initalrabbits <- 2
years <- seq(from = 1, to = 100, by = 2)

# try again with different parameters
alternativeparms <- list(r = 0.04, carry_capacity = 500)
results2 <- ode(initalrabbits, years, dexppop_play, alternati


# look at results
head(results2)
```

```
     time        1
[1,]    1 2.000000
[2,]    3 2.166576
[3,]    5 2.347022
[4,]    7 2.542500
[5,]    9 2.754258
[6,]   11 2.983651
```

```r
colnames(results2) <- c("year", "P_parm2")

# plot
ggplot(as.data.frame(results2), aes(year, P_parm2)) +
  geom_point() +
  labs(y = "Population", "years")
```

```r
# compare by combining into a single data frame
both <- inner_join(as.data.frame(results), as.data.frame(resu

both_p <- both %>% gather(key = "model", value = "P", -year)
```

PLOT

```r
# plot
ggplot(both_p, aes(year, P, col = model)) +
  geom_point() +
  labs(y = "Population", "years")
```

DIFFERENTIAL EQUATION, DIFFERENCE (ITERATION BY HAND) COMPARISON

Remember we have 3 ways now to calculate population

- analytical solution - based on integration (exppop.R) BEST
- using an ode solver for numerical approximation (exppop_play.R)
- numerical integration using in discrete steps (discrete_logistic_pop.R)

HOW DO THEY DIFFER

```r
# lets also keep the parameters for use later
P0 <- 2
r <- 0.05
K <- 200


# get all models
# discrete
source(here("R/discrete_logistic_popK.R"))
# analytic
source(here("R/exppopK.R"))
# differential for ode
source(here("R/dexppop_play.R"))


# create times we want to see results for
growth_result <- data.frame(time = seq(from = 1, to = 100))

growth_result$Panalytic <- growth_result$time %>% map_dbl(
```

```
     time        1
[1,]    1 2.000000
[2,]    2 2.102545
[3,]    3 2.210342
[4,]    4 2.323669
[5,]    5 2.442807
[6,]    6 2.568053
```

```r
# we already have time - so just extract population
growth_result$Pdifferential <- result[, 2]

# compare all 3 approaches
tmp <- growth_result %>% pivot_longer(cols = -time, names_to
ggplot(tmp, aes(time, P, col = Ptype)) +
  geom_point()
```

```r
# notice Pdifferential is closer to Panalytic than Pdiscrete
```

OTHER EXAMPLES

All differential and difference equations are approximations The analytical solution is exact
Notice that differential equations is a bit more accurate!

LETS LOOK AT SOMETHING A BIT MORE COMPLICATED

- diffusion (how a contaminent moves through space and time)
- start with 1 dimension in space

DIFFUSION CONCEPTUAL MODEL

MODELING DIFFUSION

Diffusion can be implemented as a partial differential equation
Complicated to solve - but there are tool in R for specific types of partial differential equations **Reactive Transport Example**

SOURCES FOR MORE ON DIFFUSION

More info on differential equations in R **online book differential equation in R**

SIMPLE DIFFUSION MODEL

Diffusionn would require partial derivatives - time and space! it gets much more tricky …beyond this course
Appoximate diffusion with a difference equation - and iterative to get an estimate of how diffusion works
Example of Diffusion - difference equation implementation to see what some issues can be

DIFFUSION IN 1 DIMENSION

DIFFUSION IN ONE DIMENSION THROUGH TIME

DIFFUSION DATA STRUCTURE

R IMPLEMENTATION

```r
source(here("R/diffusion.R"))

# run our diffusion model (iterative difference equation) wit
# using diffusion parameters 0.5 s/m2, 10 m2
result <- diff1(initialC = 10, nx = 10, dx = 1, nt = 8, dt = 

# a list is returned with our 3 data frames for concentration
result
```

```
$conc
          [,1]     [,2]     [,3]      [,4]      [,5]
[,6]        [,7]
[1,] 10.000000 0.000000 0.000000 0.0000000 0.0000000
0.000000000 0.000000000
[2,]  7.500000 2.500000 0.000000 0.0000000 0.0000000
0.000000000 0.000000000
[3,]  6.250000 3.125000 0.625000 0.0000000 0.0000000
0.000000000 0.000000000
[4,]  5.468750 3.281250 1.093750 0.1562500 0.0000000
0.000000000 0.000000000
[5,]  4.921875 3.281250 1.406250 0.3515625 0.0390625
0.000000000 0.000000000
[6,]  4.511719 3.222656 1.611328 0.5371094 0.1074219
0.009765625 0.000000000
[7,]  4.189453 3.142090 1.745605 0.6982422 0.1904297
```

```r
# used filled contour to plot results
head(result$conc)
```

```
          [,1]     [,2]     [,3]      [,4]      [,5]
[,6] [,7] [,8] [,9]
[1,] 10.000000 0.000000 0.000000 0.0000000 0.0000000
0.000000000    0    0    0
[2,]  7.500000 2.500000 0.000000 0.0000000 0.0000000
0.000000000    0    0    0
[3,]  6.250000 3.125000 0.625000 0.0000000 0.0000000
0.000000000    0    0    0
[4,]  5.468750 3.281250 1.093750 0.1562500 0.0000000
0.000000000    0    0    0
[5,]  4.921875 3.281250 1.406250 0.3515625 0.0390625
0.000000000    0    0    0
[6,]  4.511719 3.222656 1.611328 0.5371094 0.1074219
0.009765625    0    0    0
     [,10]
```

```
[1,]      0
```

```r
filled.contour(result$conc, xlab = "Time", ylab = "Distance")
```

```r
# or if you prefer this orientation (Distance on x axis)
filled.contour(t(result$conc), ylab = "Time", xlab = "Distanc
```

CHANGE PARAMETERS (DIFFUSIVITY D, AND SPACE AND TIME STEPS (DX, DT))

```r
# changes diffusivity and other parameters particularly
# diffusivity, dx and dt
res <- diff1(initialC = 10, nx = 10, dx = 1, nt = 10, dt = 30

filled.contour(res$conc, xlab = "Time", ylab = "Distance")
```

```r
# we can also see how much material moved from place to place
filled.contour(res$qin, xlab = "Time", ylab = "Distance")
```

```r
# play with time step, space step and parameters
```

PLAY

Try running the diffusion model with different time steps, space steps and parameters

SOME EXAMPLES WITH DIFFERENT PARAMETERS AND SPACE/TIME STEPS

```r
# what if we increase diffusivity
resfast <- diff1(initialC = 10, nx = 10, dx = 0.5, nt = 10, d
filled.contour(resfast$conc, xlab = "Time", ylab = "Distance"
```

```r
# Discretization Issue Example
resunstable <- diff1(initialC = 10, nx = 10, dx = 1, nt = 10,
filled.contour(resunstable$conc, xlab = "Time (fraction of ho
```

```r
# this illustrates the problem with difference equations (and
# if things are changing quickly we need to use much smaller

# so lets cut our step size by 10 (dt) (but then add 10 more
resunstable <- diff1(initialC = 10, nx = 100, dx = 1, nt = 10
filled.contour(resunstable$conc, xlab = "time", ylab = "Dista
```

DYNAMICS MODELS

- Diffusion example illustrates the challenge of numerical integration
- We see evidence of “overshoot”
- Correct by reducing the time step (but then we have to increase the number of time steps to cover the same period)
  - recall total time is number of time steps (nt) multiplied by time interval (dt)

DIFFUSION EXAMPLE

```r
source(here("R/diffusion.R"))


# Change parameters (diffusivity D, and space and time steps

res <- diff1(initialC = 100, nx = 10, dx = 1, nt = 10, dt = 3
filled.contour(res$conc, xlab = "Time", ylab = "Distance", ma
```

```r
# we can also see how much material is moving in to each cell
filled.contour(res$qin, xlab = "Time", ylab = "Distance", mai
```

```r
# we can also see net amount of material moved from place to
filled.contour(res$qin - res$qout, xlab = "Time", ylab = "Dis
```

```r
# what if we increase diffusivity
resfast <- diff1(initialC = 100, nx = 10, dx = 0.5, nt = 10,
filled.contour(resfast$conc, xlab = "Time", ylab = "Distance"
```

```r
filled.contour(resfast$qin, xlab = "Time", ylab = "Distance",
```

```r
# this illustrates the problem with difference equations (and
# if things are changing quickly we need to use much smaller

# so lets cut our step size by 10 (dt) (but then  multiply nu
resfast_fixtime <- diff1(initialC = 100, nx = 10, dx = 0.5, n
filled.contour(resfast_fixtime$conc, xlab = "time", ylab = "D
```

```r
filled.contour(resfast_fixtime$qin, xlab = "Time", ylab = "Di
```

```r
filled.contour(resfast_fixtime$qin - resfast_fixtime$qout, xl
```

EXTRACTING MEANING FROM TIME SERIES OUTPUT

Useful to brainstorm about what is important
For example

- time it takes to evenly diffuse?

How would we implement that?

EXTRACTING INFORMATION FROM SPACE-TIME RESULTS

- pictures can be hard to interpret
- summarizing over one of the dimensions (either space or time) can help
- looking at a single trajectory through time
- looking at spatial variation for one point in time
- looking at spatial variation for multiple points in time

TRY IT

```r
# View(resfast_fixtime$conc)


# graph a single point in space through time
# single column (time)
plot(resfast_fixtime$conc[, 3], ylab = "Concentration for a l
```

```r
# plot all trajectories
# add a time column to concentration data frame and transform
resl <- as.data.frame(resfast_fixtime$conc) %>%
  mutate(time = seq(from = 1, to = 100)) %>%
  pivot_longer(-time, names_to = "distance", values_to = "con
ggplot(resl, aes(time, conc, col = distance)) +
  geom_line()
```

```r
# plot all places at each point in time
ggplot(resl, aes(time, conc, group = time)) +
  geom_boxplot()
```

```r
# use apply to calculate the spatial variation for each row (
cvar <- resfast_fixtime$conc %>% apply(1, var)
cmean <- resfast_fixtime$conc %>% apply(1, mean)

spatial_aver <- cbind.data.frame(cvar, cmean, time = seq(from
length(cvar)
```

```
[1] 100
```

```r
# notice its the same as the number of time units (nt) used a

# plot spatial variation through time
ggplot(spatial_aver, aes(time, cvar)) +
  geom_line() +
  labs(y = "Spatial Variation")
```

```r
# plot coefficient of variation (so standard deviation divide
ggplot(spatial_aver, aes(time, 100 * sqrt(cvar) / cmean)) +
  geom_line() +
  labs(y = "COV (as percent")
```