DYNAMIC SYSTEMS WITH MORE THAN ONE DEPENDENT VARIABLE

SYSTEMS OF EQUATIONS

What if we have more than one variable that is evolving through time and space?

- populations of two species that are changing through time, and interact with each other
- pollutant concentrations in a diffusion-advection model where the two pollutants also interact with each other
- growth of above-ground and below-ground carbon and nitrogen in a forest
- household consumption and savings through time (when consumption depends on savings and vice versa)

These require several differential equations that have to be solved simultaneously

SYSTEMS OF EQUATIONS

- can be ordinary (differentiating with respect to one variable (time or space))
- partial if differentiating with respect to multiple variables (x,y,z, time)

We can estimate trajectories of systems of equations in much the same way that we used numerical integration for our ODE's
Here again methods (especially for complicated parital derivative system of equations) can be complicated
Call your engineering/math when you run into issues!

DYNAMICS OF TWO (OR MORE) VARIABLES

- two variable dynamic models that have feedbacks between variables can create cyclical dynamics (and more complex)
- Two ways to look at results
  - time series of each state variable
  - how state variables interact with each other

PREDATOR-PREY MODELS

Predator-Prey models
A simple approach that assumes prey grow exponentially, with a fixed intrinsic growth rate

- a fixed mortality rate of predators
- a fixed rate of consumption/predation rate of prey by predators
- a fixed conversion rate (ingestion rate) that determines how many “new” predators you get with predation
- no environmental effects (e.g no carrying capacity)

PREDATOR-PREY MODEL

Analogs
As with diffusion, the basic form/ideas in this model can be applied elsewhere

- economics
- infectious disease spread
- combustion

DIFFERENTIAL EQUATIONS FOR A PREDATOR-PREY MODEL

- Prey

$$\frac{\partial prey}{\partial t} = r_{prey} * prey - \alpha * prey * pred$$

- Predator

$$\frac{\partial pred}{\partial t} = eff * \alpha * pred * prey - mort * pred$$

PREDATOR PREY - IMPLEMENTATION IN R

- Ordinary Differential Equation with two dependent variables
- Still use **ODE** solve in R
- Still code the derivative as a function
- Use lists or vectors to bring in initial conditions for all dependent variables; (similar how we bring in multiple parameters to derivative definition function)
- use *with* can help make it easier to code the use of parameters within the derivative definition function (see example below)
- use lists to output derivatives for all dependent variable

EXAMPLE IMPLEMENTATION

```
source(here("R/lotvmod.R"))
lotvmod
```

```
function (t, pop, pars)
{
    with(as.list(c(pars, pop)), {
        dprey <- rprey * prey - alpha * prey
* pred
        dpred <- eff * alpha * prey * pred -
pmort * pred
        return(list(c(dprey, dpred)))
    })
}
```

```
# note the use of with
# initial conditions
currpop <- c(prey = 10, pred = 1)

# time points to see results
days <- seq(from = 1, to = 100, by = 1)

# set parameters
pars <- c(rprey = 0.5, alpha = 0.3, eff =

# run the model
res <- ode(func = lotvmod, y = currpop, ti
```

RUN THIS MODEL - HOW WOULD YOU VISUALIZE RESULTS?

VISUALIZING RESULTS

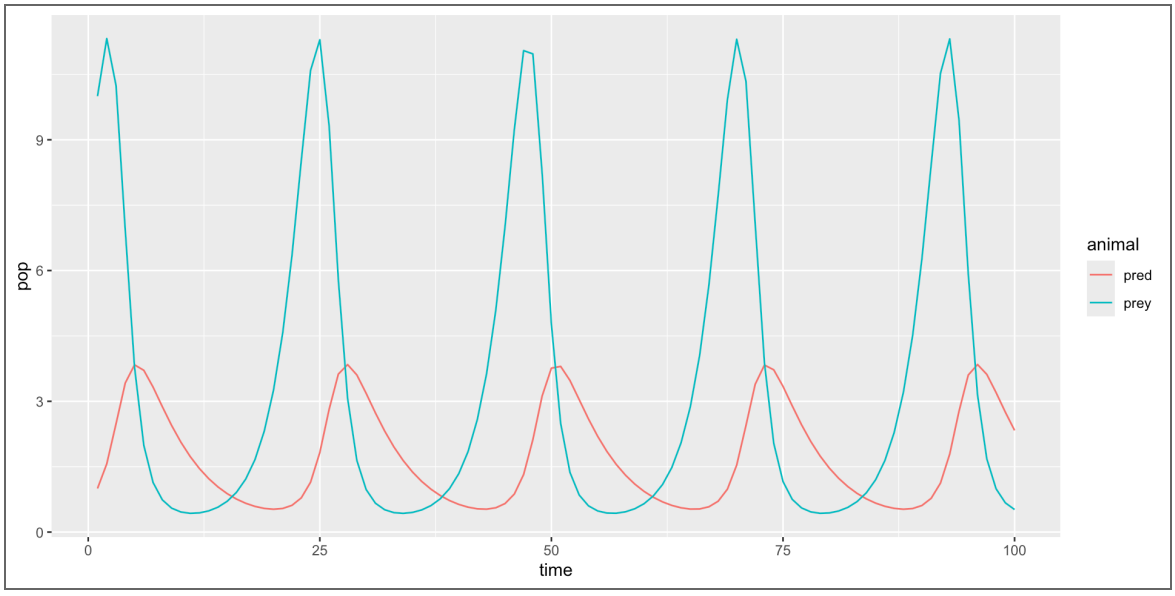
- two variable dynamic models that have feedbacks between variables can create cyclical dynamics (and more complex)
- Two ways to look at results
  - time series of each state variable (pred and prey)
  - how state variables interact with each other
    - interactions through time
    - x versus y colored by time

RELATIONSHIP BETWEEN POPULATIONS

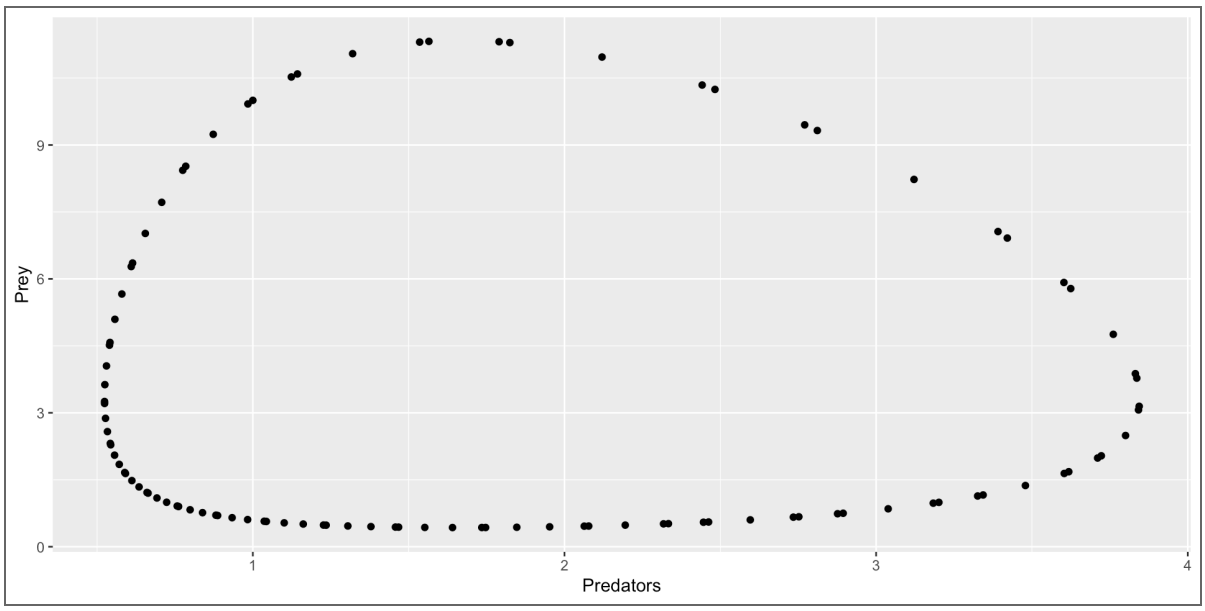
```
# graph the results
head(res)
```

```
     time      prey     pred
[1,]    1 10.000000 1.000000
[2,]    2 11.320956 1.564997
[3,]    3 10.244569 2.482924
[4,]    4  6.914805 3.420960
[5,]    5  3.777091 3.836373
[6,]    6  1.987468 3.710744
```

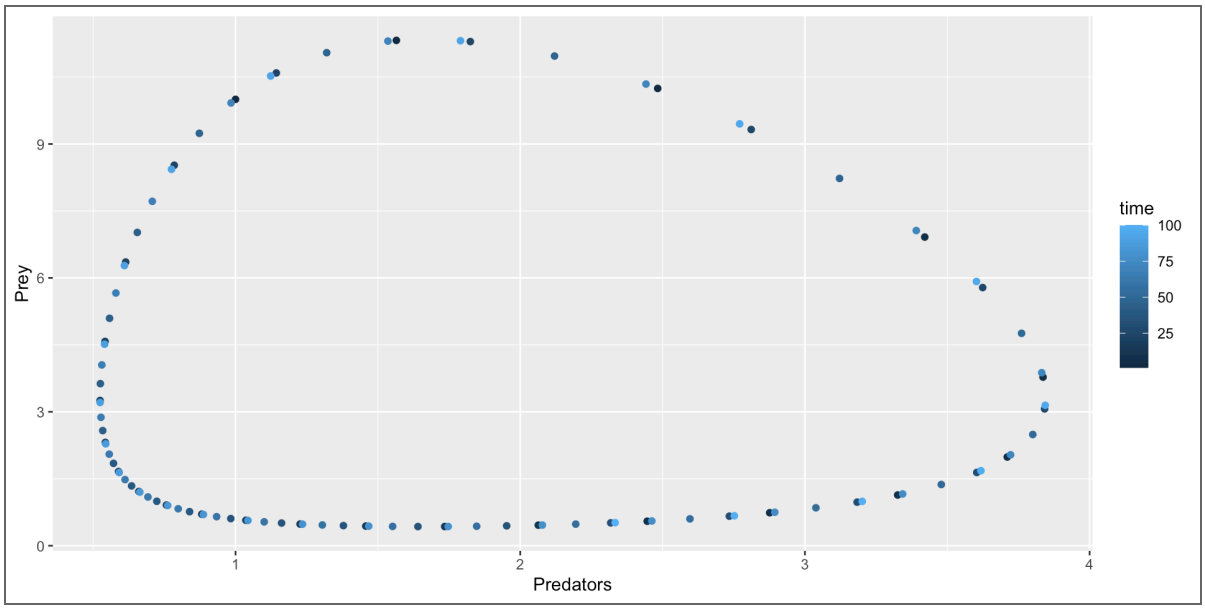
```
# rearrange for easy plotting
resl <- as.data.frame(res) %>% pivot_longe
p1 <- ggplot(resl, aes(time, pop, col = an
  geom_line()

p1
```



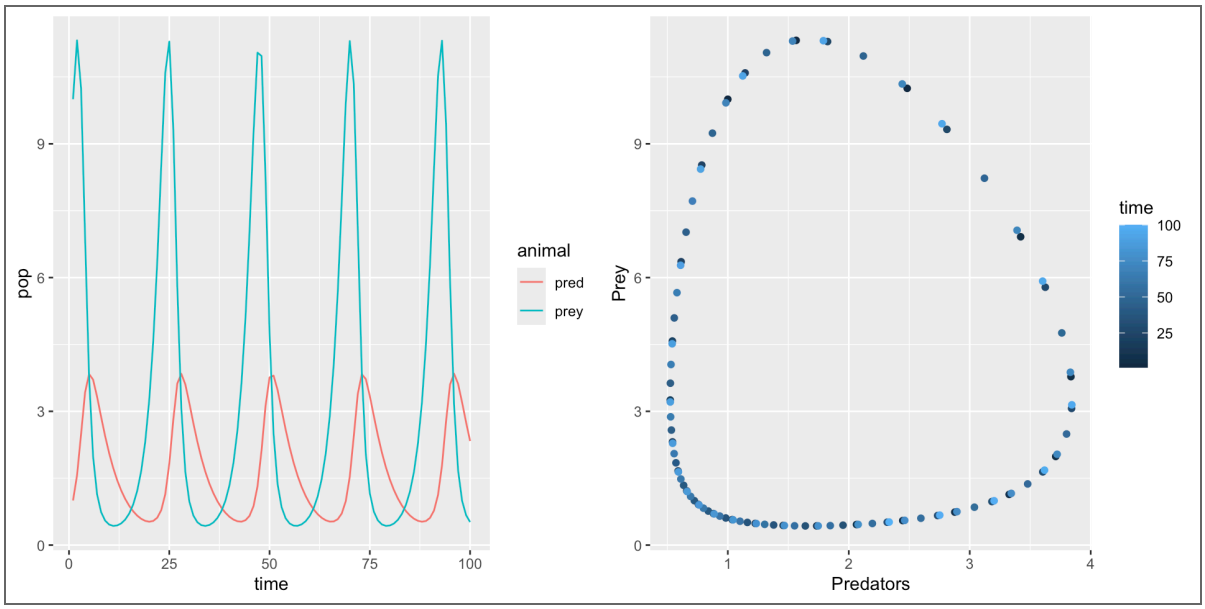
```
p2 <- ggplot(as.data.frame(res), aes(pred,
  geom_point() +
  labs(y = "Prey", x = "Predators")
p2
```




```
# To make this easier to understand - mayb
p2b <- ggplot(as.data.frame(res), aes(pred
  geom_point() +
  labs(y = "Prey", x = "Predators")
p2b
```




```
1 ggarrange(p1, p2b)
```



TRY OTHER PARAMETERS

- try to bring relative size of predators (versus prey) higher
- what if you increase the predation rates (what might that look like in reality)
- how might you add a carrying capacity

OTHER ILLUSTATIONS

- Prey

$$\frac{\partial prey}{\partial t} = r_{prey} * prey - \alpha * prey * pred$$

- Predator

$$\frac{\partial pred}{\partial t} = eff * \alpha * pred * prey - mort * pred$$

Predator and Prey with Carrying Capacity?
How would you code that?

WITH CARRYING CAPACITY

- Prey

$$\frac{\partial prey}{\partial t} = r_{prey} * (1 - \frac{prey}{K}) * prey - \alpha * prey * pred$$

- Predator

$$\frac{\partial pred}{\partial t} = eff * \alpha * pred * prey - mort * pred$$

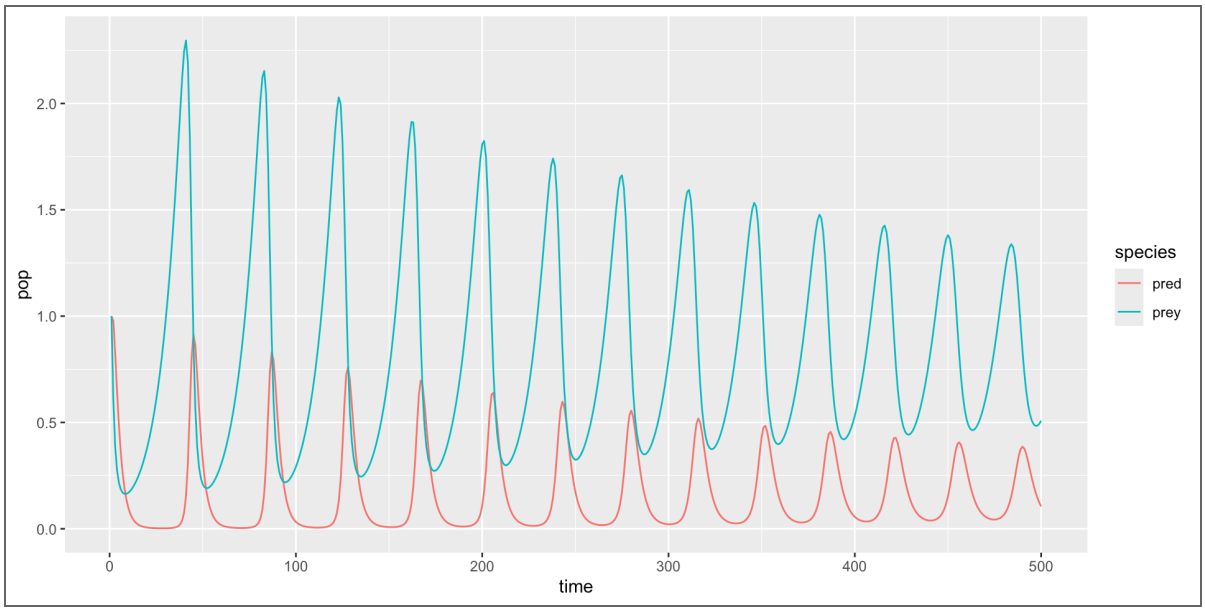
IMPLEMENTATION

```
source(here("R/lotvmodK.R"))
lotvmodK
```

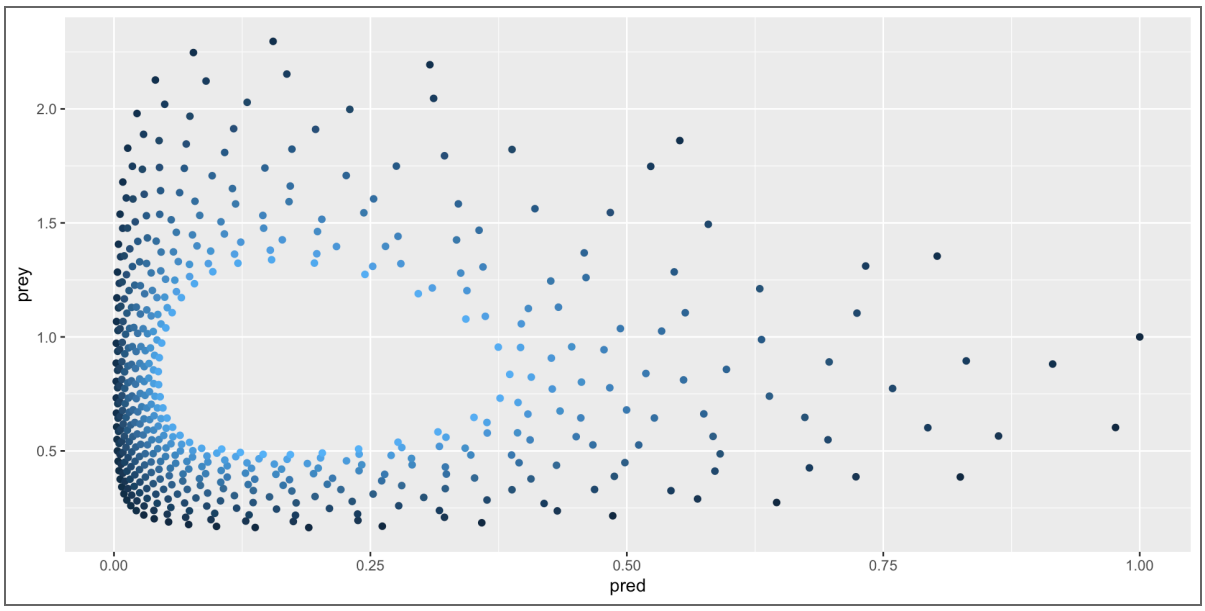
```
function (t, pop, pars)
{
    with(as.list(c(pars, pop)), {
        dprey <- rprey * (1 - prey/K) * prey
- alpha * prey *
            pred
        dpred <- eff * alpha * prey * pred -
pmort * pred
        return(list(c(dprey, dpred)))
    })
}
```

```
# initial conditions
currpop <- c(prey = 1, pred = 1)

# set parameter list
pars <- c(rprey = 0.1, alpha = 0.6, eff =

# times when you want to evaluate
days <- seq(from = 1, to = 500)

# run our differential equation solver
res <- ode(func = lotvmodK, y = currpop,

# rearrange for plotting
resl <- as.data.frame(res) %>% pivot_long

```

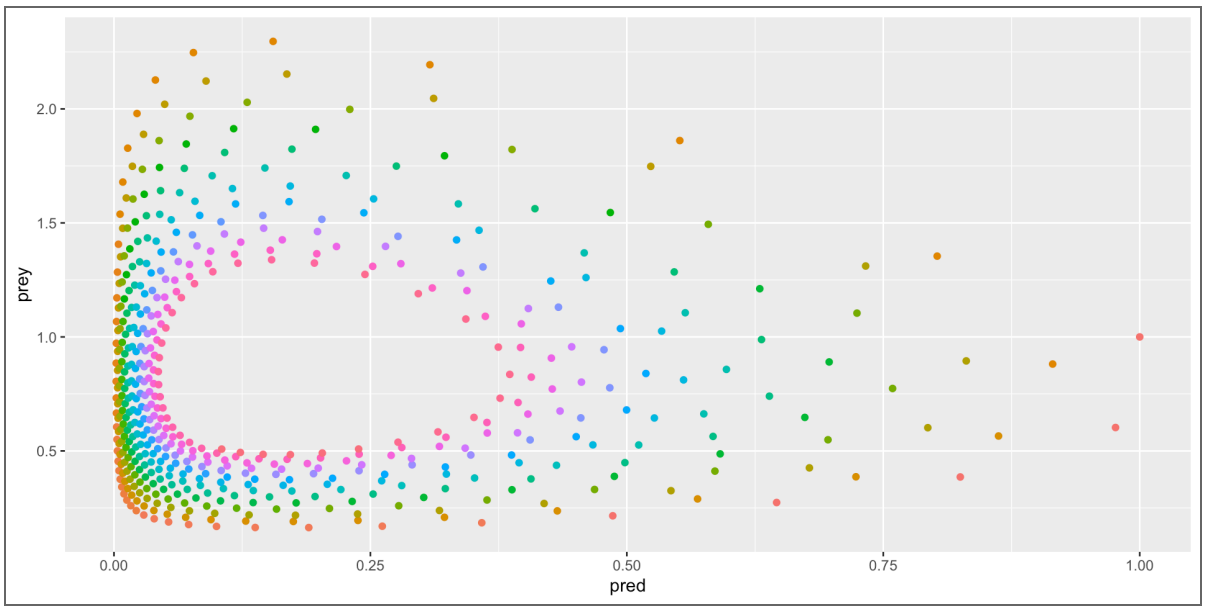
```
# graph both populations over time
p1 <- ggplot(resl, aes(time, pop, col = s
  geom_line()
```



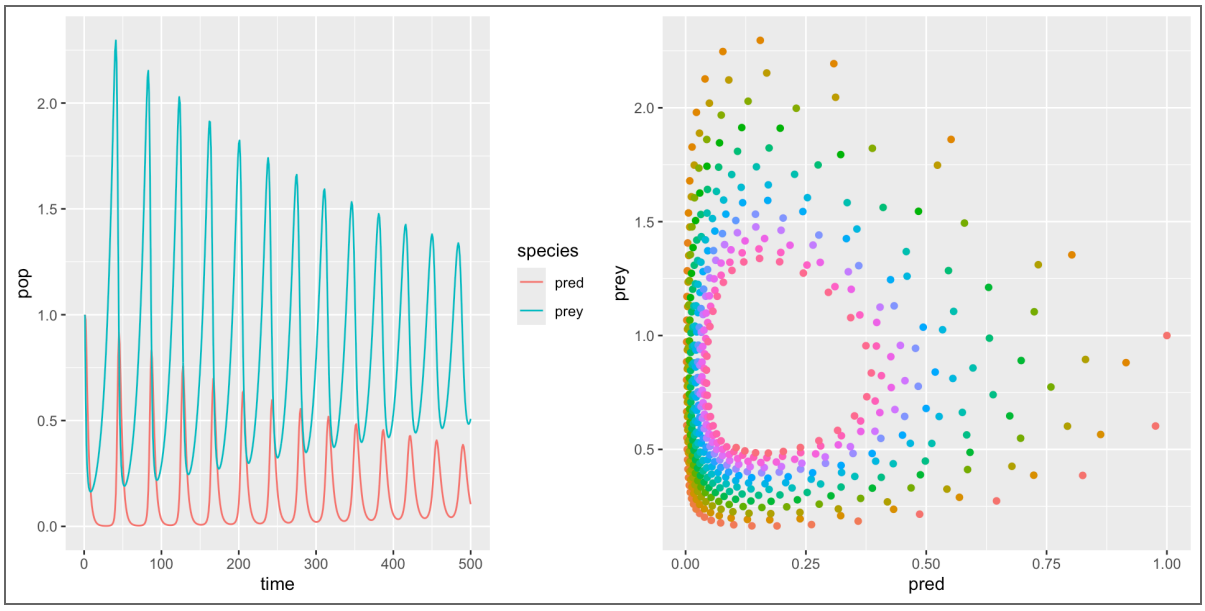

```
# also look at relationships between predi
# I will remove the legend here to make it
p2 <- ggplot(as.data.frame(res), aes(pred,
  geom_point() +
  theme(legend.position = "none")
p2
```



```
p2 <- ggplot(as.data.frame(res), aes(pred,
  geom_point() +
  theme(legend.position = "none")
p2
```




```
ggarrange(p1, p2)
```




```
# try with different parameter sets, can y
```

ANOTHER EXAMPLE: COMPETITION

Species 1 (or Company 1)

$$\frac{\partial s_1}{\partial t} = r_1 * s_1 * (1 - (\frac{s_1 + \alpha_{12} * s_2}{K_1}))$$

Species 2 (or Company 2)

$$\frac{\partial s_2}{\partial t} = r_2 * s_2 * (1 - (\frac{s_2 + \alpha_{21} * s_1}{K_2}))$$

- How might you explain what this is doing?
- What do the coefficients “mean”

COMPETITION

Species 1 (or Company 1)

$$\frac{\partial s_1}{\partial t} = r_1 * s_1 * (1 - (\frac{s_1 + \alpha_{12} * s_2}{K_1}))$$

Species 2 (or Company 2)

$$\frac{\partial s_2}{\partial t} = r_2 * s_2 * (1 - (\frac{s_2 + \alpha_{21} * s_1}{K_2}))$$

- s_1 , s_2 are species populations
- r_1 , r_2 are growth rates of each species
- K_1 , K_2 are carrying capacities; could be the same for both species but maybe not?
- α_{12} , α_{21} are competitive effect of the other species; could be the same for both species

AND JUST FOR FUN (I'LL EXXPLAN WHY)

- Lorenz Equations (for fluid dynamics),
- x,y,z variables that change with time that describe how convection in the atmosphere works - a cell that is warmed from below and cooled from above
- x rate of convective overturning
- y departure from linear horizontal (upwelling/downwelling) temperature gradient
- z departure from linear vertical temperature difference
- 3 equations (dx/dt, dy/dt, dz/dt) that describe how these variables change with time
- 3 parameters (a,b,c) related to atmospheric properties
- Developed by Meteorologist Edward Lorenz - early climate model development in 1960s
- Lorenz equations are example of dynamic systems that can exhibit stable and chaotic states depending on parameters and initial conditions

CODE FOR LORENZ SYSTEM

Lets look at a Lorenz System Code

```
# lorenz
source(here("R/lorenz.R"))

lorenz
```

```
function (t, pt, parms)
{
    with(as.list(c(pt, parms)), {
        dx <- a * (y - x)
        dy <- x * (b - z) - y
        dz <- x * y - c * z
        return(list(c(dx, dy, dz)))
    })
}
```

APPLICATION OF LORENZ

```
pars <- list(a = 10, b = 28, c = 8 / 3)
res <- ode(func = lorenz, c(x = 0.1, y = 0
```

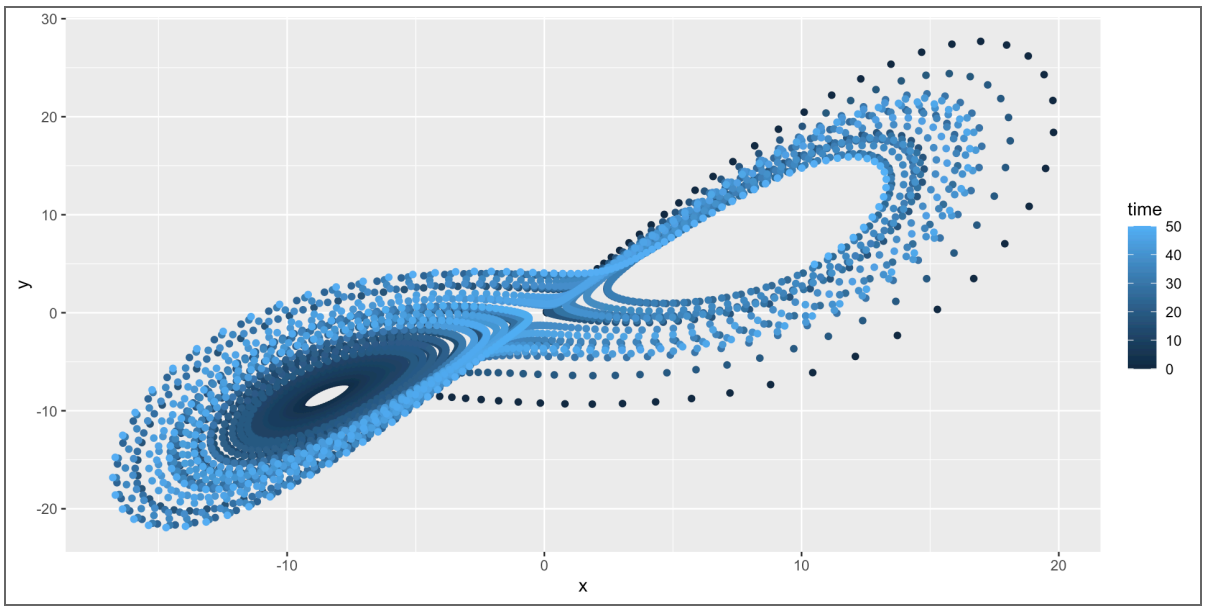
now plot

- as a phase space diagram (x vs y, x vs z, y vs z) - using time for color
- as a time series (e.g time on the x axis) add each variable with a different color

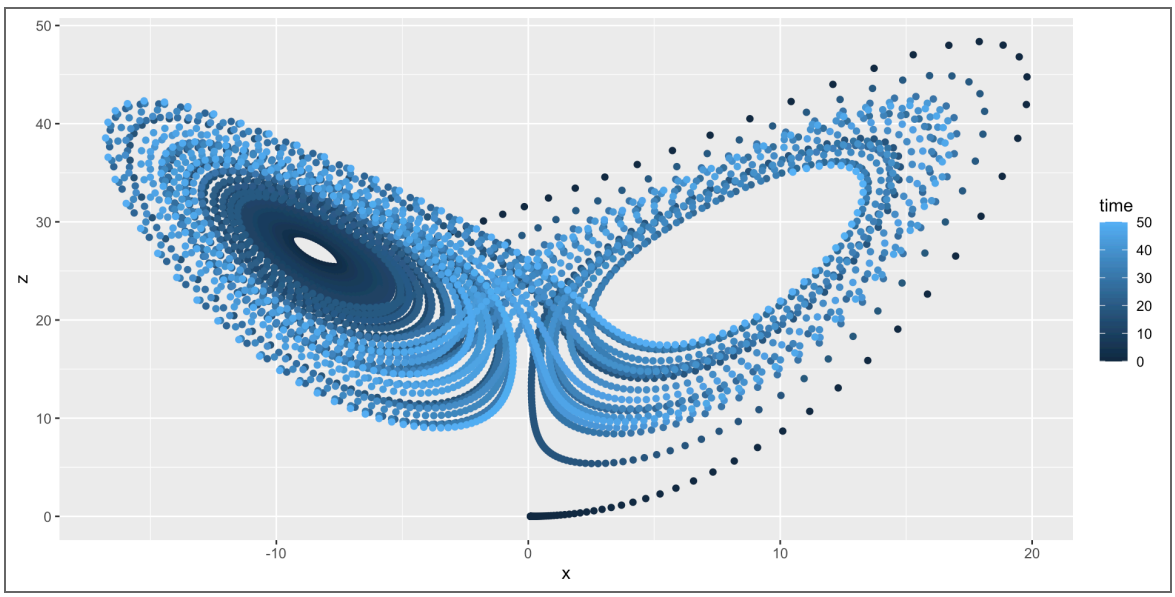
  then try with different initial conditions

PLOTS

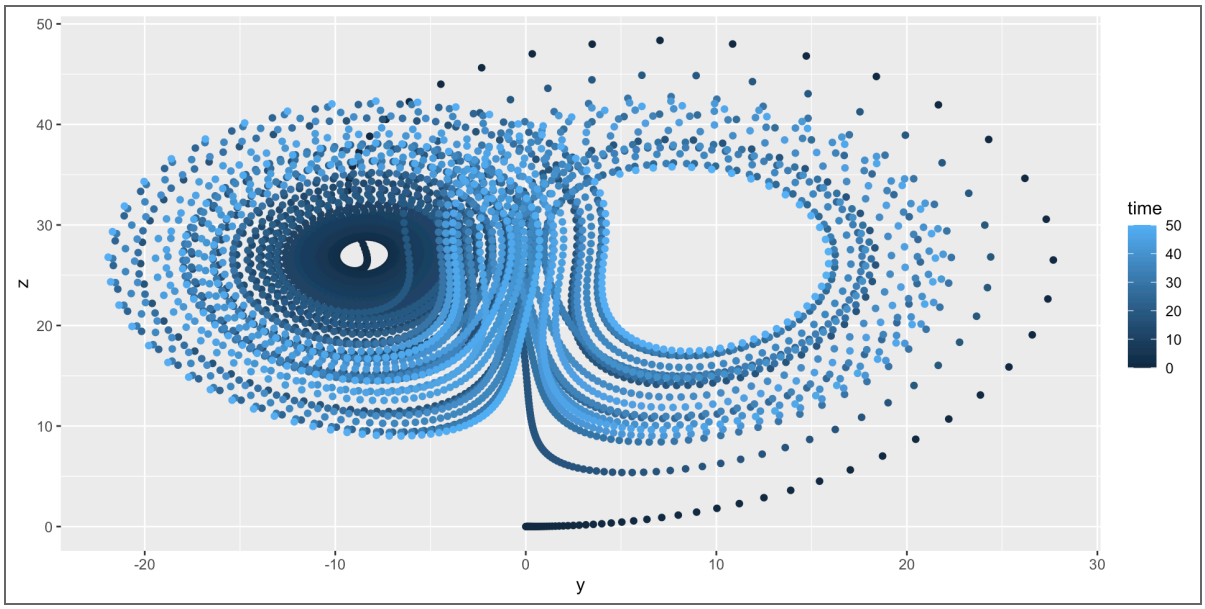
```
1 ggplot(as.data.frame(res), aes(x, y, col =
2   geom_point()
```



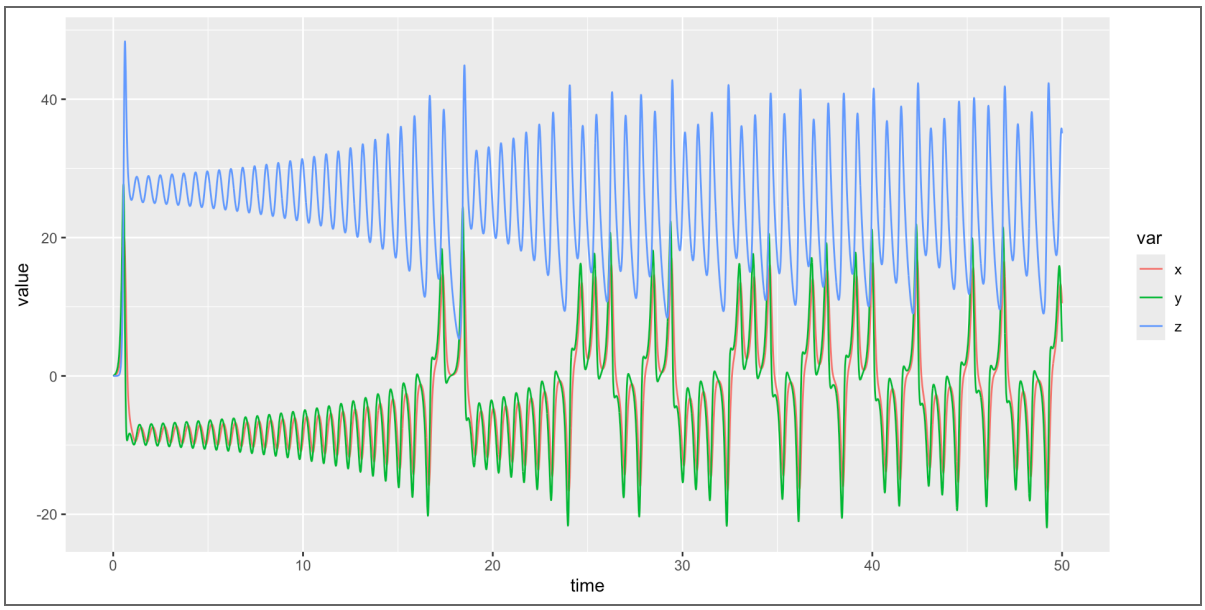

```
1 ggplot(as.data.frame(res), aes(x, z, col =
2   geom_point()
```



```
ggplot(as.data.frame(res), aes(y, z, col =
  geom_point()
```

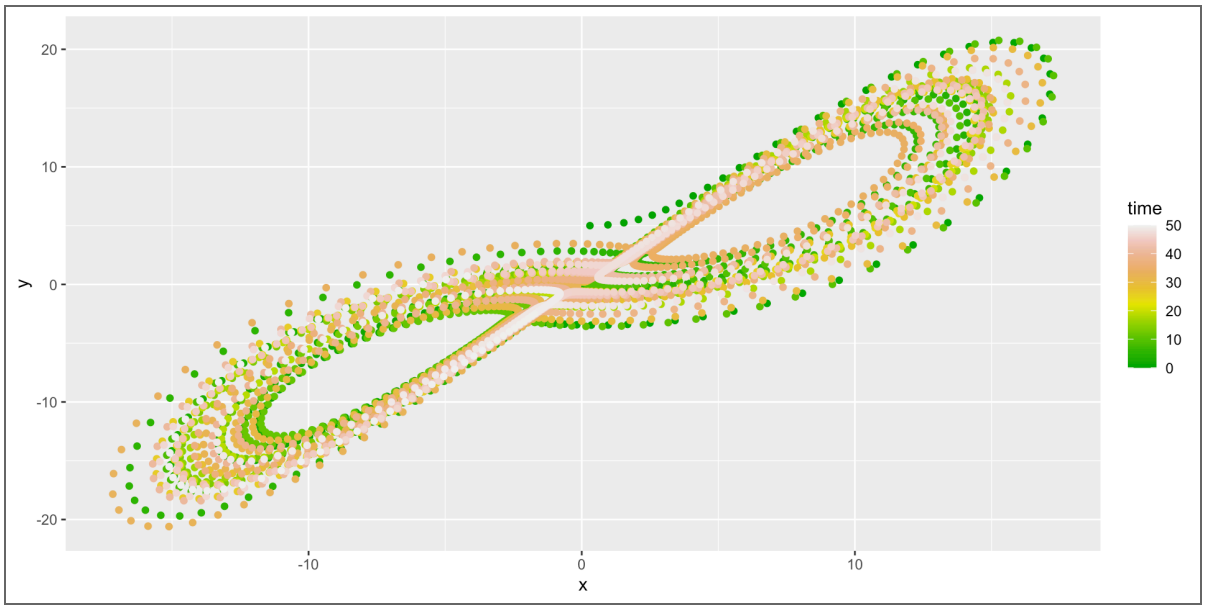



```
resl <- as.data.frame(res) %>% gather(key
ggplot(resl, aes(time, value, col = var))
  geom_line()
```

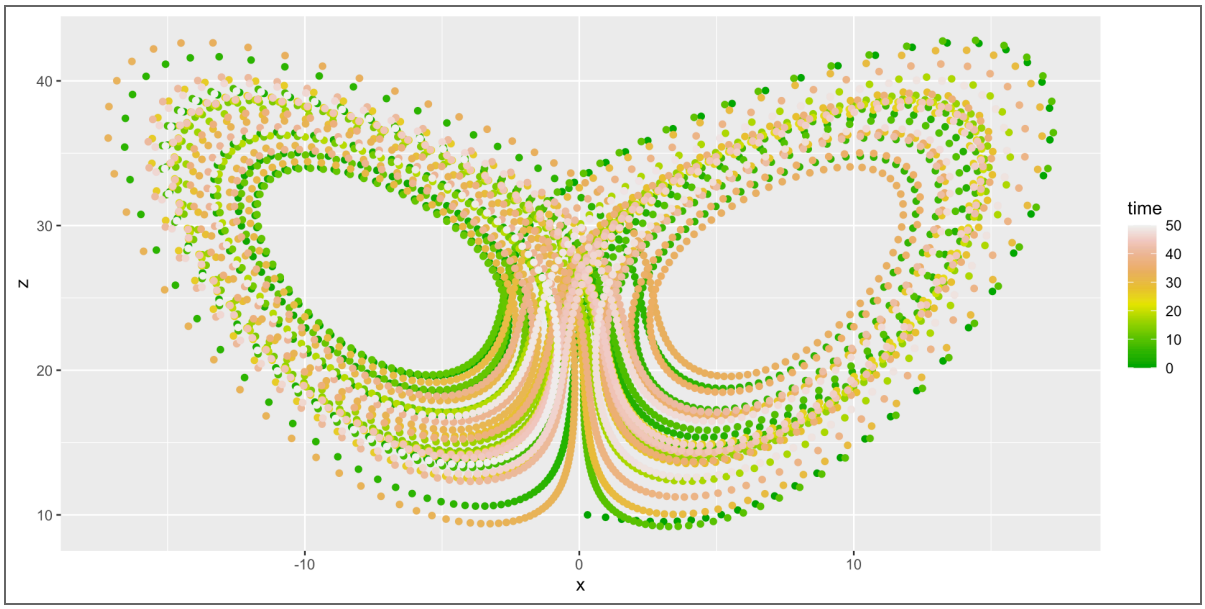


PLOT WITH DIFFERENT INITIAL CONDITIONS

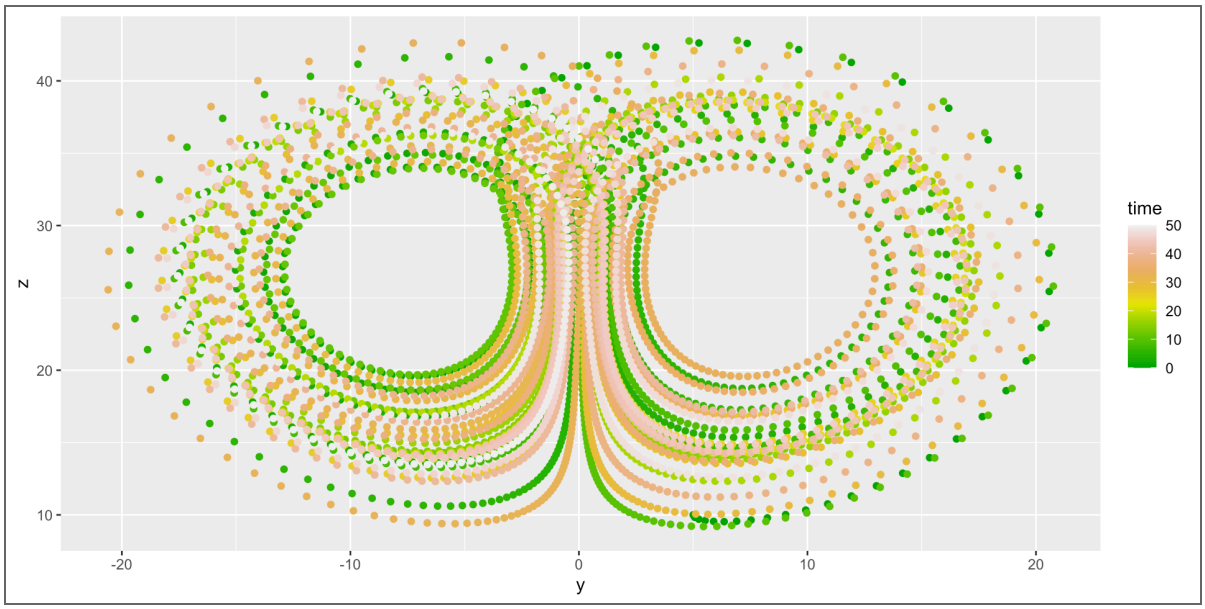
```
# try with different initial conditions
pars <- list(a = 15, b = 28, c = 8 / 4)
res <- ode(func = lorenz, c(x = 0.3, y = 5
ggplot(as.data.frame(res), aes(x, y, col =
  geom_point() +
  scale_colour_gradientn(colours = terrain
```



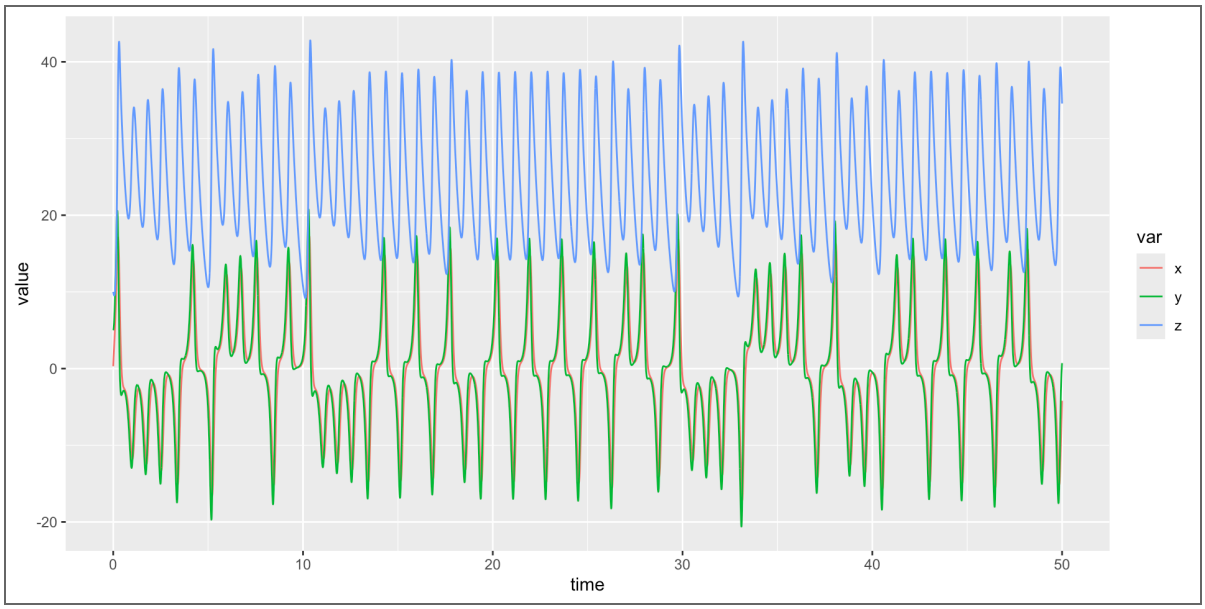

```
ggplot(as.data.frame(res), aes(x, z, col =
  geom_point() +
  scale_colour_gradientn(colours = terrain
```




```
ggplot(as.data.frame(res), aes(y, z, col =
  geom_point() +
  scale_colour_gradientn(colours = terrain
```




```
resl <- as.data.frame(res) %>% gather(key
ggplot(resl, aes(time, value, col = var))
  geom_line()
```


40
20
0
-20
value
0
10
20
30
40
50
time
var
x
y
z

SENSITIVITY ANALYSIS

Consider pred-prey BUT what will be the output - if we want to ‘quantify sensitivity’ useful to look at a single value or set of value
For example

- Max Prey/Predator Population
- Min Prey/Predator Population
- Populations at the end of the time period
- Number of time periods where a population is below a threshold

SENSITIVITY ANALYSIS STEPS

- Generate parameters (LHS, Sobol)
- Metrics function
- Wrapper Function
- Run wrapper function to get metrics for all parameter sets
- Graph and compute sensitivity statistics

EXAMPLE SENSITIVITY ANALYSIS

Given a Predator Prey model with the possible values for parameters

- K - mean of 150, standard deviation 20
- r_{prey} - some where between 0.01 and 0.3
- α - somewhere between 0.1 and 0.4
- eff - mean 0.3 standard deviation 0.01
- $pmort$ - somewhere between 0.01 and 0.45
- Initial Conditions: start with 1 predator and 1 prey
- Look at output for 500 time steps

Questions sensitivity analysis might answer

- Does uncertainty in parameter impact our estimates?
- Which are the more/most important parameters in controlling population dynamics?

SET UP SOBOL SENSITIVITY

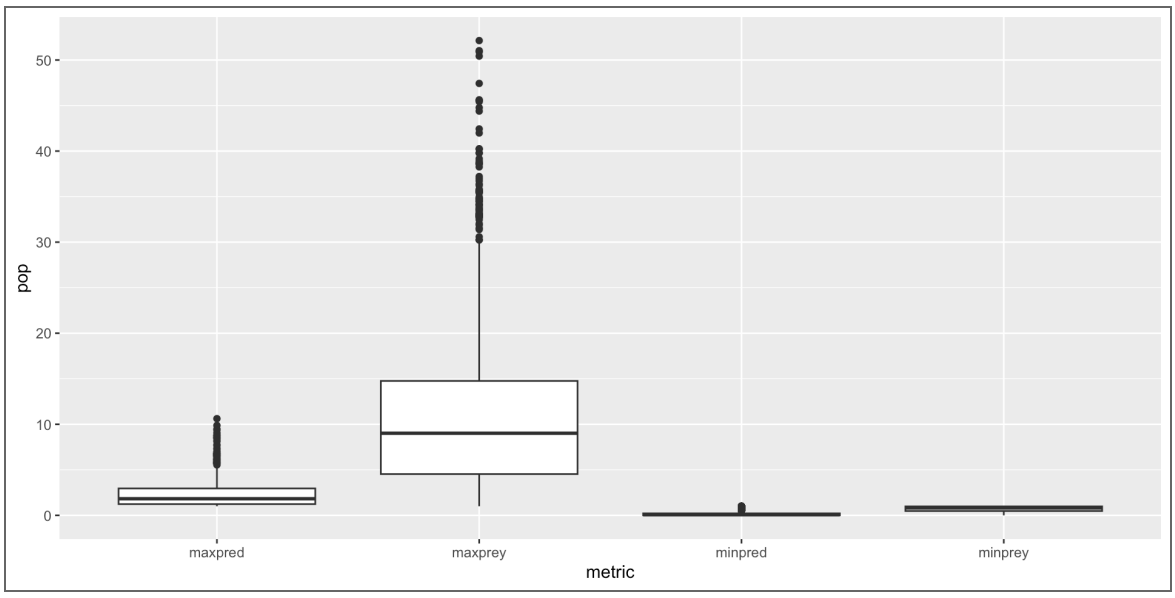
- parameter samples
- metric and wrapper function

```
compute_metrics <- function(result) {
  maxprey <- max(result$prey)
  maxpred <- max(result$pred)
  minprey <- min(result$prey)
  minpred <- min(result$pred)
  return(list(maxprey = maxprey, minprey
}

# build a wrapper function


p_wrapper <- function(rprey, alpha, eff,
  parms <- list(rprey = rprey, alpha = al
  result <- ode(y = currpop, times = days
  colnames(result) <- c("time", "prey", "
  # get metrics
  metrics <- compute_metrics(as.data.fram
  return(metrics)
}
```

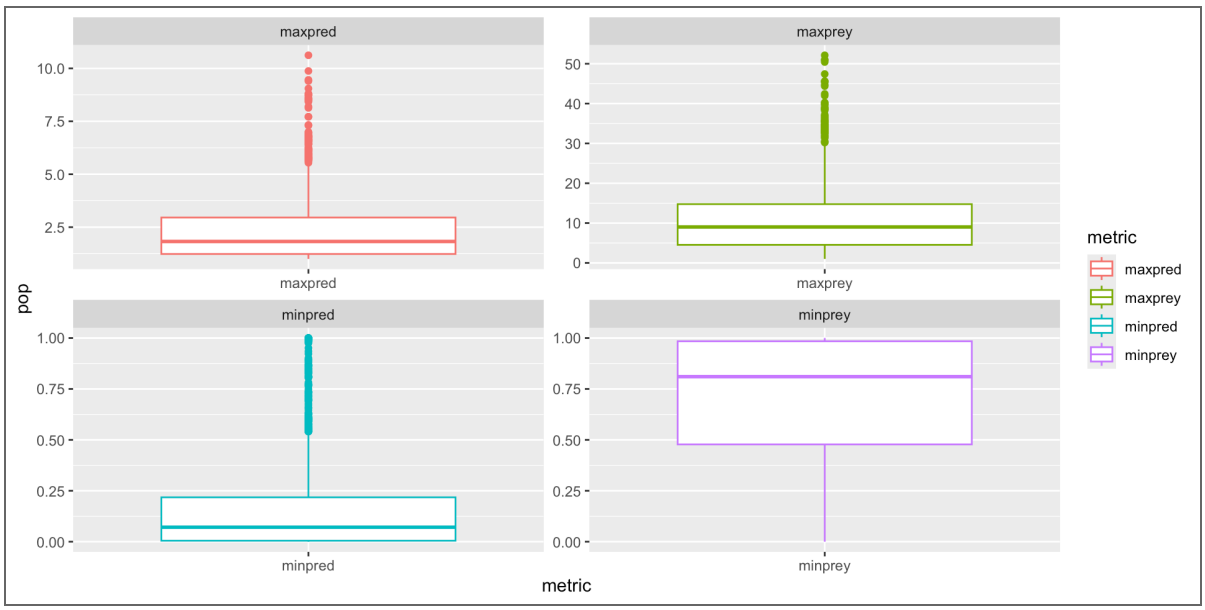
NOW RUN WRAPPER FOR ALL PARAMETERS

- graph
- sobol indices

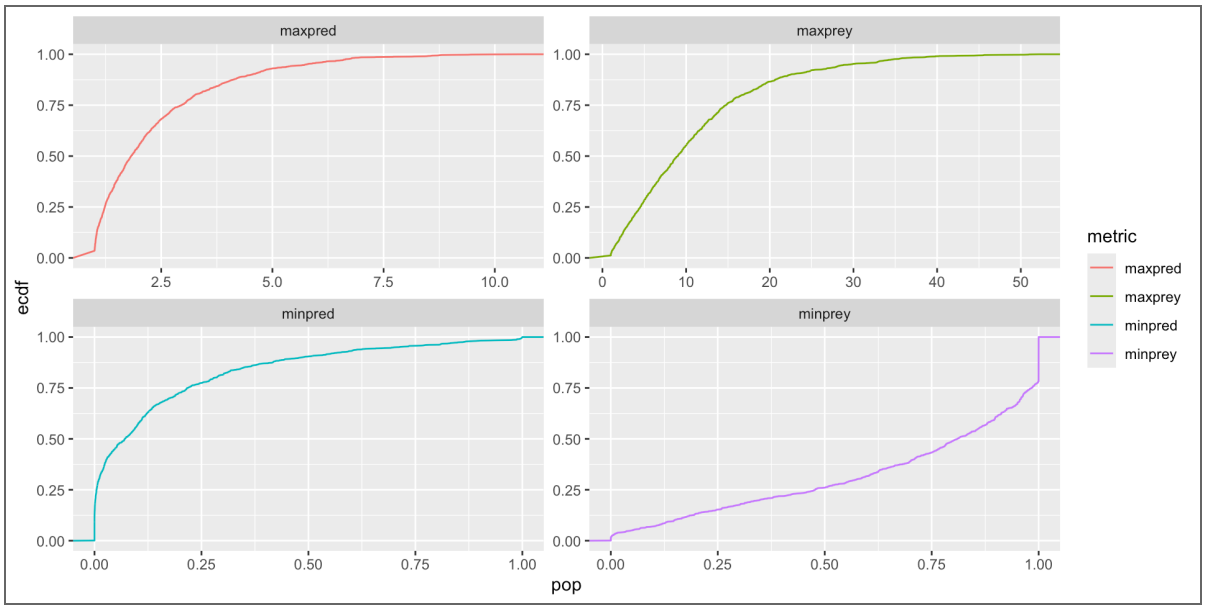
```
# run our model for all parameters and ext
currpop <- c(prey = 1, pred = 1)
days <- seq(from = 1, to = 500)
allresults <- as.data.frame(sens_PP$X) %>%
    pmap(p_wrapper, currpop = currpop, day

# take results back to unlisted form
allres <- allresults %>% map_dfr(`[`, c("m


# range of response across parameter uncer
allresl <- allres %>% gather(key = "metric
ggplot(allresl, aes(metric, pop)) +
  geom_boxplot()
```



```
# dealing with different scales
ggplot(allresl, aes(metric, pop, col = met
  geom_boxplot() +
  facet_wrap(~metric, scales = "free")
```




```
# plot cummulative densities

ggplot(allresl, aes(pop, col = metric)) +
  stat_ecdf(geom = "line") +
  facet_wrap(~metric, scales = "free")
```


maxpred
maxprey
minpred
minprey
ecdf
pop
metric
maxpred
maxprey
minpred
minprey
0.00
0.25
0.50
0.75
1.00
2.5
5.0
7.5
10.0
0
10
20
30
40
50

SOBOL INDICES

```
# create sobol indices for Max Prey
sens_PP_maxprey <- sens_PP %>% sensitivity
rownames(sens_PP_maxprey$S) <- c("rprey",
sens_PP_maxprey$S
```

```
        original          bias std. error
min. c.i. max. c.i.
rprey 0.03449832  0.0004211636 0.06709363
-0.09228323 0.1627492
K     0.03144130 -0.0001817130 0.06305458
-0.08839195 0.1504988
alpha 0.37365008 -0.0081786064 0.08131706
0.23744773 0.5362571
eff   0.02882396 -0.0002289820 0.06333170
-0.09171229 0.1474846
pmort 0.45716225 -0.0040115036 0.04620082
0.37141078 0.5537604
```

```
rownames(sens_PP_maxprey$T) <- c("rprey",
sens_PP_maxprey$T
```

```
          original          bias   std.
error    min. c.i.    max. c.i.
rprey 0.0139541131  3.022319e-04
0.0026874308 0.0072638342 0.0181240815
K     0.0005079275 -2.898919e-06
0.0001597233 0.0001666785 0.0007875281
alpha 0.5258168035  3.186012e-03
```

```
0.0471936799 0.4201158126 0.6126495328
eff   0.0035393363  2.335539e-05
0.0008179446 0.0016792294 0.0049246431
pmort 0.6219035243  1.821257e-03
0.0870587147 0.4311560972 0.7681069537
```

INTERPRETATION OF SOBOL INDICES

- what are the most important parameters
  - maximum prey
- What about other metrics
  - minimum prey
  - maximum predator
  - minimum predator

  What is the most important parameter, Which parameters don’t matter

ORGANIZING YOUR ANALYSIS

```
# keep track of sensitivity from all metri

sens_PP_minprey <- sens_PP %>% sensitivity
rownames(sens_PP_minprey$S) <- c("rprey",
sens_PP_minprey$S
```

```
          original          bias std. error
min. c.i. max. c.i.
rprey  0.163247420 -0.005139186 0.07397344
0.01909218 0.3281960
K     -0.009383569 -0.002356971 0.07284104
-0.14173261 0.1461423
alpha  0.132057108 -0.007412775 0.06732461
0.02208209 0.2814211
eff   -0.007863065 -0.002358240 0.07267084
-0.14103604 0.1460614
pmort  0.399525345 -0.001147800 0.07570112
0.26803774 0.5551299
```

```
rownames(sens_PP_minprey$T) <- c("rprey",
sens_PP_minprey$T
```

```
          original           bias   std.
error     min. c.i.    max. c.i.
rprey 2.732719e-01  8.993745e-04 4.290673e-
02  1.897397e-01 3.456817e-01
K     1.969354e-05  7.196275e-08 1.004207e-
05 -1.551879e-06 3.487192e-05
alpha 3.236745e-01 -2.905742e-06 5.087754e-
```

```
02  2.150573e-01 4.259451e-01
eff   2.545704e-04  3.589427e-07 6.830718e-
05  1.092601e-04 3.768396e-04
pmort 5.307489e-01  6.927800e-03 6.167042e-
02  3.945242e-01 6.400572e-01
```

```
sens_PP_maxpred <- sens_PP %>% sensitivity
rownames(sens_PP_maxpred$S) <- c("rprey",
sens_PP_maxpred$S
```

```
        original         bias std. error
min. c.i. max. c.i.
rprey 0.13233268 -0.011008501 0.07272230
-0.01131634 0.2978213
K     0.02056540 -0.006724132 0.06305028
-0.09836018 0.1616893
alpha 0.50637990 -0.004660672 0.09151969
0.32911685 0.7085967
eff   0.01658104 -0.006645089 0.06319406
-0.09910440 0.1566786
pmort 0.23305406 -0.008309851 0.06644569
0.10838814 0.3662999
```

```
rownames(sens_PP_maxpred$T) <- c("rprey",
sens_PP_maxpred$T
```

```
          original         bias   std. error
min. c.i.   max. c.i.
rprey 0.1846738115 2.687892e-03 0.0268312211
0.1283622331 0.232684386
K     0.0022106801 4.307977e-06 0.0007339483
0.0007158794 0.003498049
alpha 0.7482975725 2.344622e-03 0.0541682009
```

```
0.6404634730 0.852049419
eff   0.0003945996 6.590964e-06 0.0000920557
0.0001546094 0.000538981
pmort 0.3490434008 3.577703e-03 0.0602239146
0.2239664870 0.455808533
```

```
sens_PP_minpred <- sens_PP %>% sensitivity
rownames(sens_PP_minpred$S) <- c("rprey",
sens_PP_minpred$S
```

```
       original          bias std. error
min. c.i. max. c.i.
rprey 0.35237228 -0.0001646473 0.07050449
0.19997119 0.4800522
K     0.06115627 -0.0057556412 0.07868147
-0.09512428 0.2214750
alpha 0.06514740 -0.0060509470 0.08431155
-0.10231755 0.2316867
eff   0.05714566 -0.0058309991 0.07819029
-0.09957600 0.2167077
pmort 0.40764405 -0.0139806443 0.09476154
0.25327775 0.6322451
```

```
rownames(sens_PP_minpred$T) <- c("rprey",
sens_PP_minpred$T
```

```
         original          bias   std.
error    min. c.i.    max. c.i.
rprey 4.523444e-01  1.308888e-02 7.738011e-
02 2.869187e-01 5.833973e-01
K     2.956213e-06  1.319337e-07 6.707537e-
07 1.429358e-06 3.916599e-06
alpha 1.640601e-01  4.512685e-04 4.046761e-
```

```
02 6.966061e-02 2.296588e-01
eff   5.461335e-04  1.559899e-05 1.171166e-
04 2.772833e-04 7.372340e-04
pmort 5.446559e-01 -5.612616e-03 8.787251e-
02 3.660702e-01 7.247044e-01
```

```
# put together in a table
sobol_sumS = as.data.frame(
   rbind(
    sens_PP_maxprey$S$original,
    sens_PP_minprey$S$original,
   sens_PP_maxpred$S$original,
    sens_PP_minpred$S$original
  ))
colnames(sobol_sumS) <- c("rprey", "K", "a
rownames(sobol_sumS) <- c("maxprey", "minp
sobol_sumS
```

```
             rprey             K     alpha
eff     pmort
maxprey 0.03449832  0.031441301 0.3736501
0.028823960 0.4571622
minprey 0.16324742 -0.009383569 0.1320571
-0.007863065 0.3995253
maxpred 0.13233268  0.020565399 0.5063799
0.016581037 0.2330541
minpred 0.35237228  0.061156267 0.0651474
0.057145661 0.4076441
```

STABLITY

- simple mathematical definition - when derivatives are zero
  - algerbra
  - plotting
- more complex definitions - think of metrics that looks at ranges, cycling, return to a population above some threshold after disturbance
- for more complex definitions of stability you'd need to do the integration (e.g run the ode solver) and then compute a metric of stability
  - min or max always above or below a threshold
  - long term values (e.g after X years)

ADDING MORE COMPLEX INPUTS

- What if carrying capacity varied with air temperature

  Conceptual Model
- There is a known optimal carrying capacity (*Kopt*)
  - *Kopt* when ecosystem is at optimal temperature (*Topt*)
  - As temperature increases or decreases from *Topt*, carrying capacity decreases
  - *Ksen* is a sensitivity coefficient that describes how much carrying capacity changes with temperature
    `
- Carrying capacity is a function of temperature
  $K = K_{opt} * (1 - Ksen * (airT(t) - T_{opt}))$
  - airT(t) is the air temperature at time t

IMPLEMENTATION DETAILS

- We would also need a model of carrying capacity (K)
  - determistic (not-dynamic) submodel
  - needs a time series of air temperature as input

  How do you combine these different types of inputs/parameters and different types of models together

DYNAMIC CARRYING CAPACITY EXAMPLE

- Carrying Capacity

$$K = K_{opt} * (1 - Ksen * (airT(t) - T_{opt}))$$

- Prey

$$\frac{\partial prey}{\partial t} = r_{prey} * (1 - \frac{prey}{K}) * prey - \alpha * prey * pred$$

- Predator

$$\frac{\partial pred}{\partial t} = eff * \alpha * pred * prey - mort * pred$$

CODE

```
source(here("R/lotvmodvaryingK.R"))
lotvmodKvar
```

```
function (t, pop, pars)
{
    with(as.list(c(pars, pop)), {
        K = KO * (1 - Ksen * (airT[t] -
Topt))
        K = max(K, 1)
        dprey <- rprey * (1 - prey/K) * prey
- alpha * prey *
            pred
        dpred <- eff * alpha * prey * pred -
pmort * pred
        return(list(c(dprey, dpred)))
    })
}
```

```
# initial conditions
currpop <- c(prey = 10, pred = 1)



# set parameters and inputs
# read in air temperature time series
airT = readRDS(here("Data/Tavg_Rattlesnake
head(airT)
```

```
# A tibble: 6 × 3
# Groups:   month [1]
  month  year  tavg
  <dbl> <dbl> <dbl>
1     1  1989  12.9
2     1  1990  13.8
3     1  1991  14.4
4     1  1992  14.3
5     1  1993  14.8
6     1  1994  14.5
```

```
#note if you were interested in specific p

# time points to see results
days <- seq(from = 1, to = length(airT$tav

# note you have to use a list now because
pars <- list(rprey = 0.5, alpha = 0.3, eff

# run the model
res <- ode(func = lotvmodKvar, y = currpop
# graph the results
head(res)
```

```
     time      prey     pred
[1,]    1 10.000000 1.000000
[2,]    2 11.093271 1.554524
[3,]    3  9.899626 2.420596
[4,]    4  6.752188 3.280867
[5,]    5  3.821711 3.668858
[6,]    6  2.093251 3.566748
```

```
# rearrange for easy plotting
resl <- as.data.frame(res) %>% pivot_longe
p1 <- ggplot(resl, aes(time, pop, col = an
  geom_line()

p1
```

```
p2 <- ggplot(as.data.frame(res), aes(pred,
  geom_point() +
  labs(y = "Prey", x = "Predators")
p2
```

```
# To make this easier to understand - mayb
p2b <- ggplot(as.data.frame(res), aes(pred
  geom_point() +
  labs(y = "Prey", x = "Predators")
p2b
```

```
ggarrange(p1, p2b)
```

ASSIGNMENT

Consider how you might add hunting of prey to the predator prey model that we've been using in class

Part 1:

Build this model (e.g add hunting to the lotvmodK.R),

Some requirements/hints for your model

You should make sure that you don't hunt more prey than exist.

To ensure that you might also add a minimum prey population input that must be met before hunting is allowed.

Note you can make this as simple or as complex as you would like. You could represent hunting in a way that is similar to "harvesting" in the last assignment.

Part 2

Explore how different hunting levels and different minimum prey populations (before hunting is allowed) are likely to effect the stability of the populations of both predator and prey. A key challenge is how you might want to define stability? It is up to you but you will need to write a sentence to explain why you chose the measure that you did. It could be something as simple as maintaining a population above some value 50 years into the future.

Use this exploration to recommend a hunting target that will be sustainable (e.g leave you with a stable

prey and predator population).
It is up to you how you "explore" hunting - you can simply try different values of the parameters in your hunting model or do it more formally by running your model across a range of values. You could think about parameter interactions
You can assume the following are best guesses of key parameters
rprey=0.95, alpha=0.01, eff=0.6,pmort=0.4, K=2000,
Submit - the Quatro document with your analysis AND a pdf/html version, any R files This should include

a. your hunting model
b. your exploration (e.g how you tested different hunting levels and how you defined a stability metric
c. provides you estimated sustainable hunting level and brief explanation of why you chose the metric you did