SENSITIVITY WITH ODES

DYNAMIC MODELS

- models that have feedbacks (conditions evolve through time)
- numerical integration - usually done with a solver
  - only one independent variable ordinary differential equation (e.g just time)
  - we can use the ODE solver
  - derivative is first order
  - (e.g $\mathrm{dy/dt}$ = ; not $\mathrm{d}^2\mathrm{y/dt}$+dy/dy = f(y,t))
  - there also solvers for high order and partial differential equations

SENSITIVITY ANALYSIS OF A DIFFERENTIAL EQUATION

We can apply sensitivity analysis to a differential equation
A key issue where is sensitivity of what?
Dynamic models often give you many many outputs

- time series (streamflow every day for a year, population for 30 years)
- or output over space (spatially averaged concentration after 10 days?)

So if we are asking 'sensitivity of what' we need to summarize results in some way (reduce their dimensionality)
Ideas?

SOME OPTIONS FOR REDUCING OUTPUT DIMENSIONALITY (SUMMARIZING OUTPUT)

Depends on what is important for your model application

- max
- mean
- min
- total
- variation
- time it takes for something to happen

So a key step in sensitivity analysis with a dynamics model is summarizing results into a few key measures Its useful to turn that summarizing workflow into a function

IN CLASS EXERISE

For the diffusion model that we've been working with

- write a function that takes as input the output of diffusion function and returns a summary
  - the summary should be a single value
    - (or if you are ambitious a list of several single value summary statistics)
  - you can be creative here - what would be an interesting summary
  - include in comments at the top of the function a rationale for your summary choice
- check that it works by applying it to

Upload your function to Canvas - InClass in this weeks Canvas - we will give 10pts for all loaded functions

WORKFLOW FOR SENSITIVITY ANALYSIS

- implement (or identify pre-existing) dynamic model
- obtain parameter sets (from sobel or LHS)
- build a function that will extract the information (metrics) you want from your dynamic model (output of the ode)
- create a data structure to store the metrics for each parameter set - in my example I call it metrics (but could be anything)
- decide on initial conditions and time period over which you will run the model
- run ODE for each parameter sets to fill in this metrics data structure
  - its usually helpful to create a wrapper function
    - runs ODE
    - extracts metrics
  - run wrapper function for each parameter sets
- send the metrics data structure back to the sensitivity analysis object (from sobel or LHS)
- plot and analyze results

EXAMPLE WITH OUR POPULATION ODE

Lets first generate parameter sets and figure out how to run with across uncertainty in just one those parameter sets
Always a good practice to “try” you model on one parameter set *before* trying to run for all parameters

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

dpopgrowth
```

```
function (Time, P, parms)
{
    dP <- parms$r * P * (1 - P/parms$K)
    return(list(dP))
}
```

```r
r <- rnorm(mean = 0.05, sd = 0.01, n = np)

X1 <- cbind.data.frame(r = r, K = K)

# repeat to get our second set of samples
K <- rnorm(mean = 200, sd = 50, n = np)
r <- rnorm(mean = 0.05, sd = 0.01, n = np)
X2 <- cbind.data.frame(r = r, K = K)

# fix any negative values and they are not
X1 <- X1 %>% map_df(pmax, 0.0)
X2 <- X2 %>% map_df(pmax, 0.0)

# create our sobel object and get sets ofp
```

```r
sens_P <- sobolSalt(model = NULL, X1, X2,

# our parameter sets are
head(sens_P$X)
```

```
              [,1]     [,2]
[1,] 0.02827552 186.3854
[2,] 0.05144594 225.5691
[3,] 0.04134681 208.1570
[4,] 0.07108986 250.7854
[5,] 0.05027642 239.3527
[6,] 0.06326704 136.0567
```

```r
# lets add names
colnames(sens_P$X) <- c("r", "K")

head(sens_P$X)
```

```
              r        K
[1,] 0.02827552 186.3854
[2,] 0.05144594 225.5691
[3,] 0.04134681 208.1570
[4,] 0.07108986 250.7854
[5,] 0.05027642 239.3527
[6,] 0.06326704 136.0567
```

RUNNING THE ODE AND SUMMARIZING OUTPUT

- run our differential equation and keep the output

BUT what output do we want to keep?
A couple of options

- how about maximum population if we run the model for 200 years,
- how many years to get to the carrying capacity
- how many year to get to some pre-determined threshold

For illustration lets look at running just one parameter sets and summarizing results

```
sens_P$X[1, ]
```

```
          r            K 
0.02827552 186.38542567 
```

```
# recall ODE needs ALL of our parameters i
# initial population and times for which w
Pinitial
```

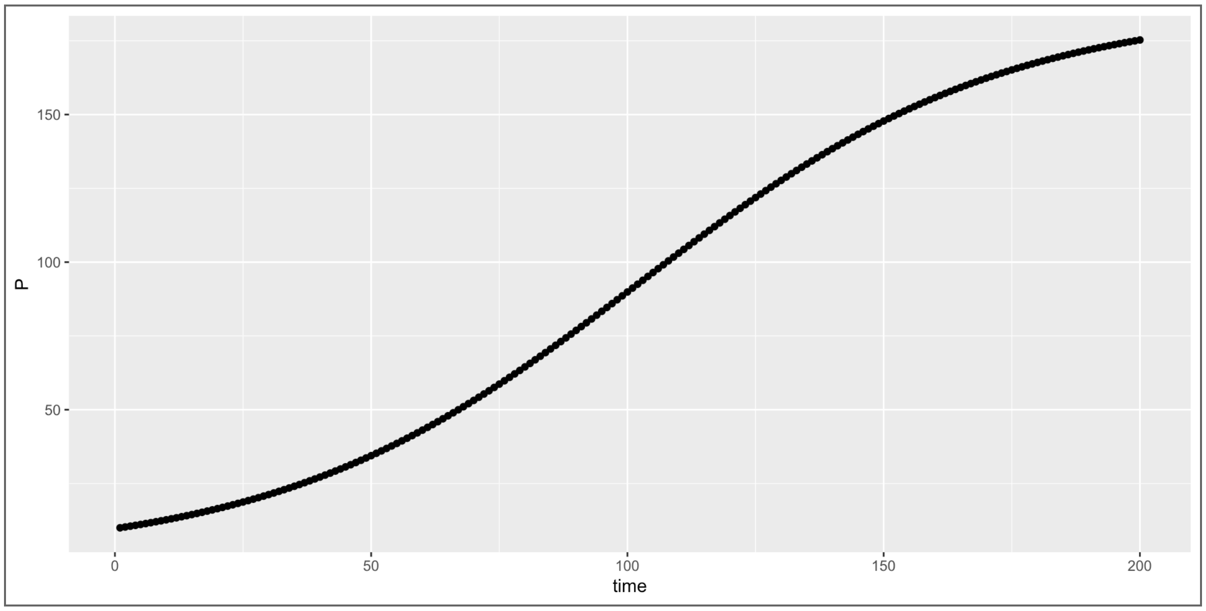
```
[1] 10
```

```
# gets results for 200 years (evaluating e
simtimes <- seq(from = 1, to = 200)
```

```
parms <- list(r = sens_P$X[1, "r"], K = se

# or
parms <- list(r = as.data.frame(sens_P$X)$

result <- ode(y = Pinitial, times = simtim

head(result)
```

```
     time        1
[1,]    1 10.00000
[2,]    2 10.27099
[3,]    3 10.54889
[4,]    4 10.83383
[5,]    5 11.12599
[6,]    6 11.42551
```

```
colnames(result) <- c("time", "P")
# turn it into a data frame
result <- as.data.frame(result)
ggplot(result, aes(time, P)) +
  geom_point()
```



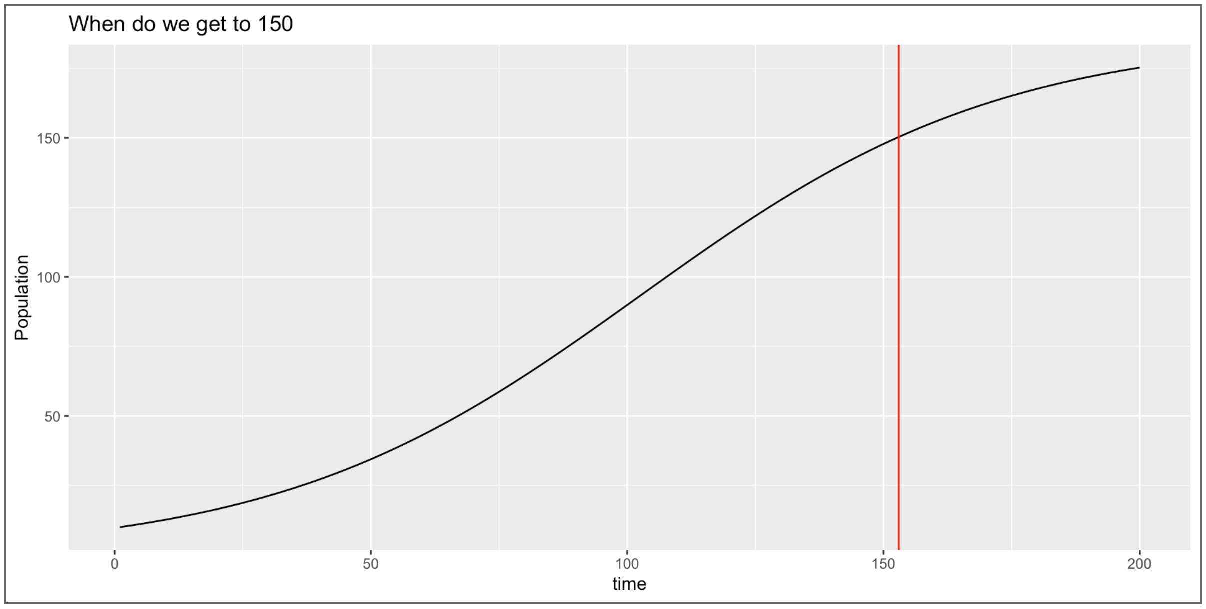
```r
# extra our metrics of interest  from this
# maximum population it gets to
maxpop <- max(result$P)
maxpop
```

```
[1] 175.2569
```

```r
# years required to get to a threshold pop
# which will tell when this occurs - we wi
thresh <- 150
idx <- which(result$P > thresh)[1]

# if it never gets there
idx <- ifelse(is.na(idx), length(result$P)
# turn this index into a year (might be th
threshyear <- result$time[idx]
threshyear
```

```
[1] 153
```

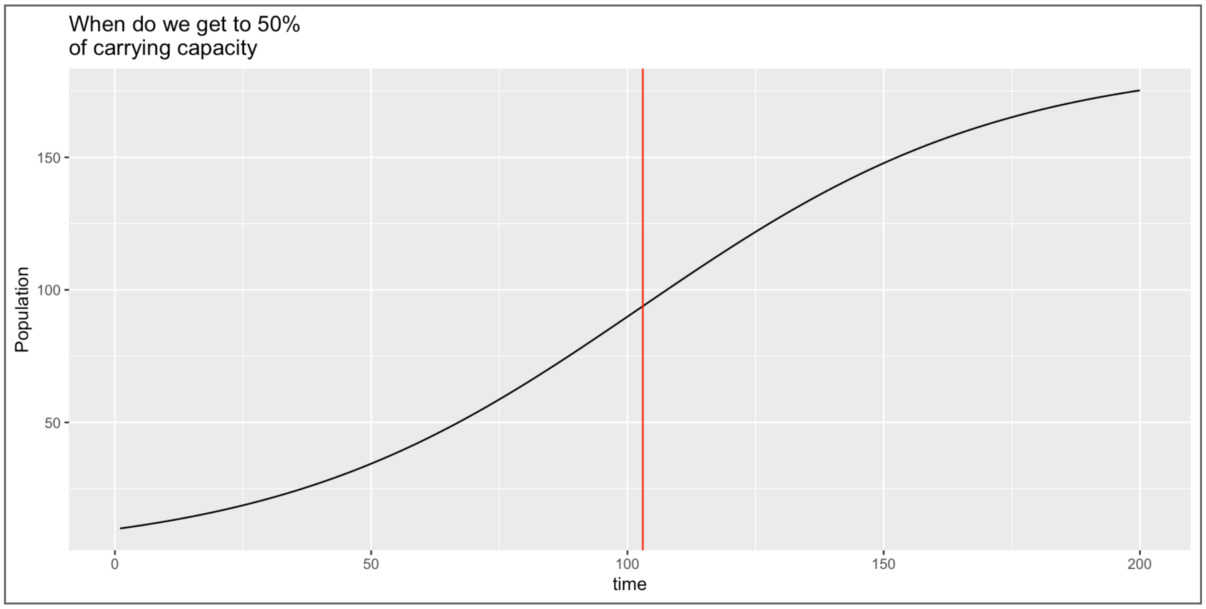
```r
ggplot(result, aes(time, P)) +
  geom_line() +
  geom_vline(xintercept = threshyear, col
  labs(y = "Population", title = "When do
```



```
# or how about threshold of 50% of carryin
thresh <- 0.5 * sens_P$X[1, "K"]
idx <- which(result$P > thresh)[1]

# if it never gets there
idx <- ifelse(is.na(idx), length(result$P)
# turn this index into a year (might be th
threshyear <- result$time[idx]
threshyear
```

```
[1] 103
```

```
ggplot(result, aes(time, P)) +
  geom_line() +
  geom_vline(xintercept = threshyear, col
  labs(y = "Population", title = "When do
```


When do we get to 50%
of carrying capacity
Population
150
100
50
0
50
100
150
200
time

MAKE A DIFFERENT METRIC OR ODE

- try a metric that gives you the population at year 5
- change the way carrying capacity is used

TRY IT RUNNING ODE FOR *ALL* PARAMETERS

One issue is the volume of output! you have a time series for *each* parameter set
So just save metrics
Lets create two additional functions that will help us

- a function that computes the metrics we want
- a function that runs our ode solver and computes the metrics (I call it a **wrapper** function as it is really just a workflow/wrapper to call ode solver and then compute metrics)
- wrapper takes as input
  - parameters
  - initial conditions
  - simulation time
  - ode (function name)
  - how to compute metrics (function name)
- always test to make sure it works (good coding practice)

```
# turn computing our metrics into a functi

compute_metrics <- function(result, thresh
  maxpop <- max(result$P)
  idx <- which(result$P > thresh)[1]
```

```r
  idx <- ifelse(is.na(idx), length(result$
  threshyear <- result$time[idx]
  return(list(maxpop = maxpop, threshyear
}

# try it on our first parameter set, and l
compute_metrics(result, 100)
```

```
$maxpop
[1] 175.2569

$threshyear
[1] 108
```

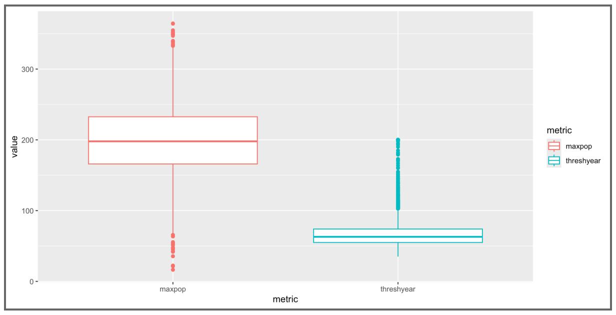
```r

# define a wrapper function to do everythi

# lets make the threshold 90% of carrying

p_wrapper <- function(r, K, Pinitial, simt
  parms <- list(r = r, K = K)
  result <- ode(y = Pinitial, times = simt
  colnames(result) <- c("time", "P")
  # get metrics
  metrics <- metricfunc(as.data.frame(resu
  return(metrics)
}

# test
p_wrapper(
  r = 0.01, K = 150, Pinitial = 3, simtime
  odefunc = dpopgrowth, metricfunc = compu
)
```

```
$maxpop
[1] 3.274979

$threshyear
[1] 10
```

NEXT STEP

Run the wrapper for all parameters and look at results

```
# now use pmap as we did before

allresults <- as.data.frame(sens_P$X) %>%

# extract out results from pmap into a dat
allres <- allresults %>% map_dfr(`[`, c("m


# create boxplots
tmp <- allres %>% pivot_longer(cols = ever
ggplot(tmp, aes(metric, value, col = metri
  geom_boxplot()
```



COMPUTE THE SOBOL INDICIES FOR EACH METRIC

- save the *tell* to different objects so you can keep re-using the original sobel object
- look at total effect and first order sensitivity

```r
# sobol can only handle one output at a ti

sens_P_maxpop <- sensitivity::tell(sens_P,

# first-order indices (main effect without
rownames(sens_P_maxpop$S) <- c("r", "K")
sens_P_maxpop$S
```

```
     original          bias  std. error
min. c.i.  max. c.i.
r 0.005269032  6.414442e-04 0.023607133
-0.04046049 0.05301377
K 0.992353738 -7.123767e-05 0.001417221
0.98993050 0.99566343
```

```r
# total sensitivity index -note that this
rownames(sens_P_maxpop$T) <- c("r", "K")
sens_P_maxpop$T
```

```
     original          bias  std. error
min. c.i.  max. c.i.
r 0.009343871  0.0001429977 0.001958884
0.004785747 0.01260628
```

```
K 1.010527625 -0.0005473689 0.023583031
0.962976673 1.05431466
```

```
# create another one for max year
sens_P_threshyear <- sensitivity::tell(sen
# first-order indices (main effect without
rownames(sens_P_threshyear$S) <- c("r", "K
sens_P_threshyear$S
```

```
   original           bias std. error min.
c.i. max. c.i.
r 0.3119883  0.0009970806 0.02926924
0.2561973 0.3691037
K 0.6862467 -0.0045270181 0.03022112
0.6351502 0.7540901
```

```
# total sensitivity index -note that this
rownames(sens_P_threshyear$T) <- c("r", "K
sens_P_threshyear$T
```

```
   original           bias std. error min.
c.i. max. c.i.
r 0.3043345  2.380513e-03 0.02750848
0.2410983 0.3543902
K 0.7225270 -9.614289e-05 0.02682866
0.6724758 0.7771277
```

NEGATIVE SOBOL FIRST ORDER INDICES

if confidence interval includes zero - not a problem
if it doesn’t there are numerical issues - try running more samples

ERROR MESSAGES FROM ODE

In lsoda(y, times, func, parms, …) : an excessive amount of work (> maxsteps) was done, but integration was not successful - increase maxsteps
Suggest that the solver (numerical integration) had issues

- increasing maxsteps can help

```
*result = ode(y=Pinitial, times=simtimes, func=func, parms=parms,
maxsteps=100000) *
```

- trying different methods

```
*result = ode(y=Pinitial, times=simtimes, func=func, parms=parms,
method="daspk")*
```

- “stiff” problems are harder for numerical integration to solve - (small changes have big impacts); a threshold carrying capacity does that

ASSIGNMENT

THE MODEL

Consider the following model of forest growth (where forest size in measured in units of carbon (C))

- $dC/dt = r * C$ for forests where C is below a threshold canopy closure
- $dC/dt = g * (1 - C/K)$ for forests where carbon is at or above the threshold canopy closure
- and K is a carrying capacity in units of carbon

The size of the forest (C), Canopy closure threshold and carrying capacity are all in units of carbon
You could think of the canopy closure threshold as the size of the forest at which growth rates change from exponential to linear You can think of r, as early exponential growth rate and g as the linear growth rate once canopy closure has been reached

YOUR TASK

1. Implement this model in R (as a differential equation)
2. Run the model for 300 years (using the ODE solver) starting with an initial forest size of 10 kg/C, and using the following parameters:

- canopy closure threshold of 50 kgC
- K = 250 kg C (carrying capacity)
- r= 0.01 (exponential growth rate before before canopy closure)
- g = 2 kg/year (linear growth rate after canopy closure)

3. Graph the results. Here you are graphing the trajectory with the parameters as given (e.g no uncertainty)
4. Run a sobol global (vary all parameters at the same time) sensitivity analysis that explores how the estimated **maximum forest size** (e.g maximum of C 300 years, varies with these parameters

- pre canopy closure growth rate (r)
- post-canopy closure growth rate (g)
- canopy closure threshold and carrying capacity(K)

Assume that parameters are all normally distributed with means as given above and standard deviation of 10% of mean value

5. Graph the results of the sensitivity analysis as a box plot of maximum forest size and record the two Sobol indices (**S** and **T**).
6. In 2-3 sentences, discuss what the results of your simulation might mean. (For example think about how what parameters climate change might influence).

Submit R markdown with model implementation, graphs and sensitivity analysis and R file with your model
You can work in groups or individually

EXTRA CREDIT

Compute Sobol indices for a second metric: forest size after a 100 years
OR
Try using Sobol for the diffusion model - what would be your metric?