Tree growth

Elmera Azadpour, Grace Lewin and Desik Somasundaram

```
source(here("R", "treegrowthcalc.R"))

treegrowthcalc
```

```
## function (time, Cinitial, parms)
## {
##     if (Cinitial < parms$thresh) {
##         treegrowth = parms$r * Cinitial
##     }
##     else if (Cinitial >= parms$thresh) {
##         treegrowth = parms$g * (1 - (Cinitial/K))
##     }
##     treegrowth = ifelse(Cinitial > parms$K, 0, treegrowth)
##     return(list(treegrowth))
## }
```

```
# lets start with sobel
library(sensitivity)

# come up with first set of sample parameters
# we will assume that we know the initial population,

Cinitial=10
thresh = 50
# want to learn about sensitivity to growth rate (r) and carrying capacity
# set the number of parameters
np=100
K = rnorm(mean=250, sd=25, n=np)
r = rnorm(mean=0.01, sd=0.001, n=np)
g = rnorm(mean=2, sd=0.2, n=np)
thresh = thresh
X1 = cbind.data.frame(r=r, K=K, g=g, thresh=thresh)

# repeat to get our second set of samples
K = rnorm(mean=250, sd=25, n=np)
r = rnorm(mean=0.01, sd=0.001, n=np)
g = rnorm(mean=2, sd=0.2, n=np)
thresh = thresh
X2 = cbind.data.frame(r=r, K=K, g=g, thresh=thresh)

# create our sobel object and get sets ofparameters for running the model

sens_C = sobolSalt(model = NULL,X1, X2, nboot = 300)
```

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

```
##              [,1]     [,2]     [,3] [,4]
## [1,] 0.008205343 244.5798 2.022663   50
## [2,] 0.010869853 255.5264 2.205111   50
## [3,] 0.008078001 239.9610 2.312686   50
## [4,] 0.010696872 277.5965 2.657897   50
## [5,] 0.008568928 271.0918 1.652792   50
## [6,] 0.010504611 187.0313 1.625828   50
```

```
# lets add names
colnames(sens_C$X) = c("r", "K", "g", "thresh")

# run our differential equation and keep the output
# BUT
# what output do we want  to keep
# how about maximum population if we run the model for 200 years, and how many years to get to the carr

# for illustration lets look at running just one parameter sets and summarizing results
sens_C$X[1,]
```

```
##            r            K            g       thresh
## 8.205343e-03 2.445798e+02 2.022663e+00 5.000000e+01
```

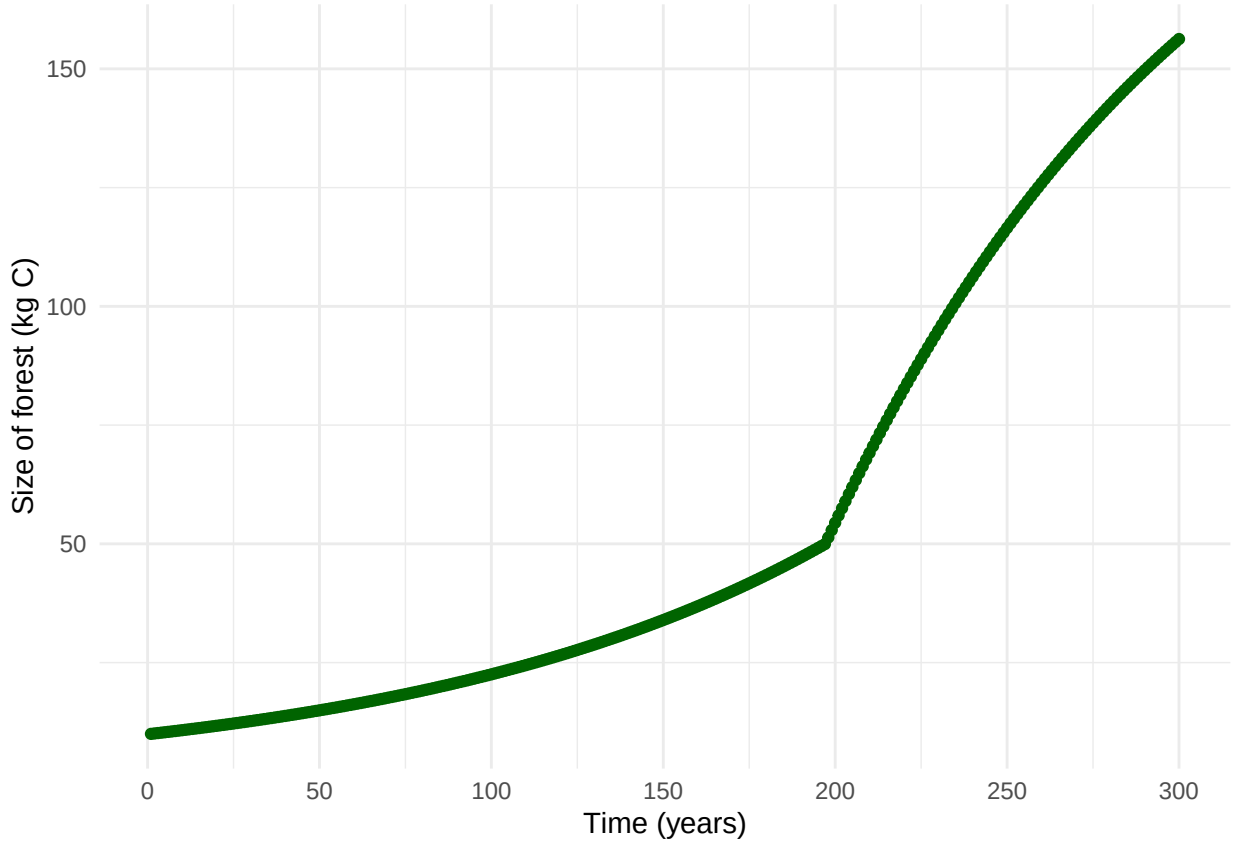
```
# recall ODE needs ALL of our parameters in a single list
# initial population and times for which we want output
Cinitial
```

```
## [1] 10
```

```
# gets results for 200 years (evaluating every year)
simtimes = seq(from=1, to=300)
parms = list(r=sens_C$X[1,"r"], K=sens_C$X[1,"K"], g=sens_C$X[1,"g"], thresh=sens_C$X[1,"thresh"])

# or
#parms = list(r=as.data.frame(sens_C$X)$r[1], K=as.data.frame(sens_C$X)$K[1], g=as.data.frame(sens_C$X)

result = ode(y=Cinitial, times=simtimes, func=treegrowthcalc, parms=parms)

head(result)
```

```
##      time        1
## [1,]    1 10.00000
## [2,]    2 10.08239
## [3,]    3 10.16546
## [4,]    4 10.24922
## [5,]    5 10.33366
## [6,]    6 10.41880
```

```
colnames(result)=c("time","C")
# turn it into a data frame
result = as.data.frame(result)

ggplot(result, aes(time, C))+
  geom_point(color = "dark green") +
  labs(x = "Time (years)", y = "Size of forest (kg C)") +
  theme_minimal() +
  scale_x_continuous(breaks = seq(from = 0, to = 300, by = 50)) +
scale_y_continuous(breaks = seq(from = 0, to = 250, by = 50))
```




```
# extra our metrics of interest  from this
# maximum population it gets to
maxsize = max(result$C)
maxsize
```

```
## [1] 156.2789
```

```
# mean population
meansize = mean(result$C)
meansize
```

```
## [1] 54.63035
```

Compute our metric for all the parameter sets

What if we want to run for all parameters

Lets create two additional functions that will help us

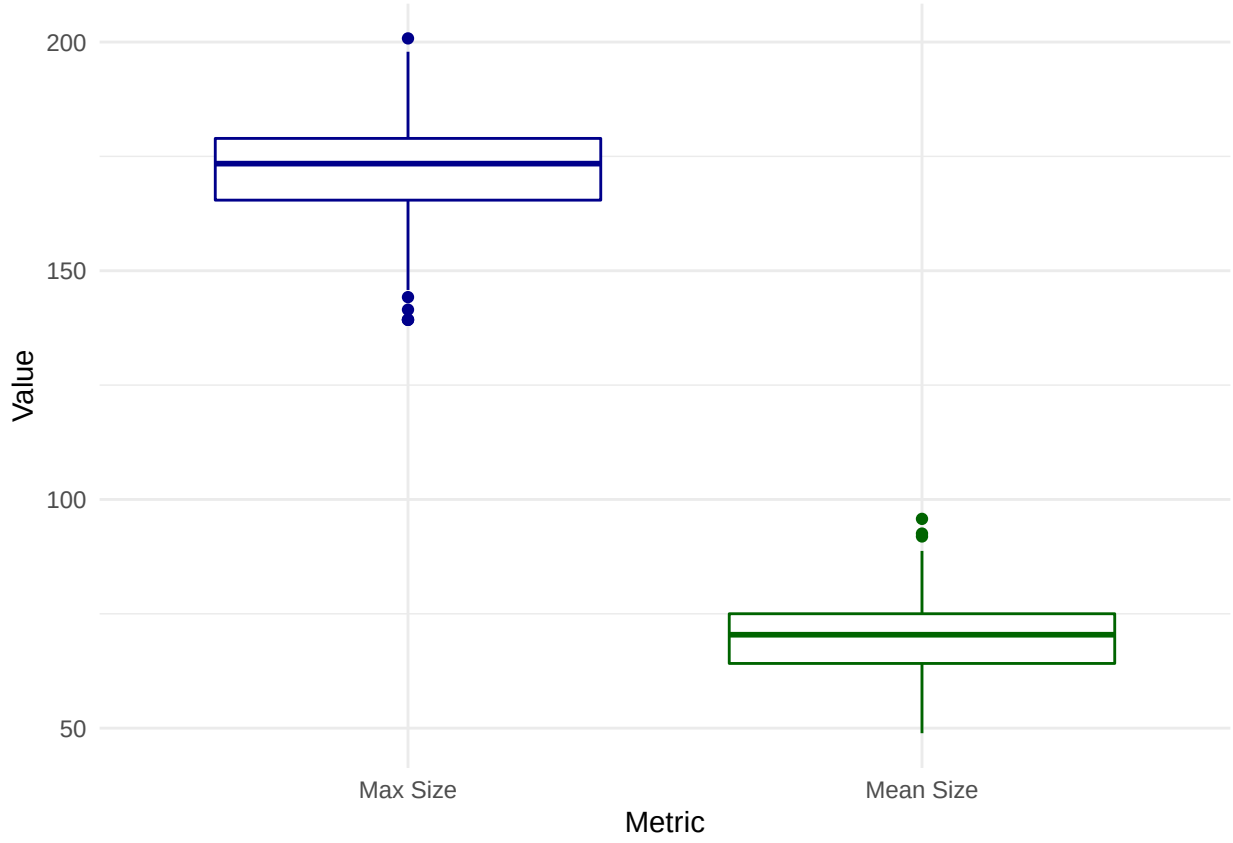
- a function that computes the metrics we want
- a function that runs our ode solver and computes the metrics (I call it a wrapper function as it is really just a workflow/wrapper to call ode solver and then compute metrics)

```
# turn computing our metrics into a function

compute_metrics = function(result) {
  maxsize = max(result$C)
  meansize = mean(result$C)
return(list(maxsize=maxsize, meansize=meansize))}

# try it on our first parameter set
compute_metrics(result)
```

```
## $maxsize
## [1] 156.2789
##
## $meansize
## [1] 54.63035
```

```
# great but we need to apply the ode and this function for all of our parameters



# define a wrapper function to do everything we need - run solver and compute metrics - and send back r

p_wrapper = function(r, K, g, thresh, Cinitial, simtimes, func) {
    parms = list(r=r, K=K, g=g, thresh=thresh)
    result = ode(y=Cinitial, times=simtimes, func=func, parms=parms, method="daspk")
    colnames(result)=c("time","C")
  # get metrics
  metrics=compute_metrics(as.data.frame(result))
  return(metrics)
}

# now use pmap as we did before

allresults = as.data.frame(sens_C$X) %>% pmap(p_wrapper, Cinitial=Cinitial, simtimes=simtimes, func=tre

# extract out results from pmap into a data frame
allres = allresults %>% map_dfr(`[`,c("maxsize","meansize"))


# create boxplots
tmp = allres %>% gather(key="metric", value="value")
```

```
ggplot(tmp, aes(metric, value, col=metric))+
  geom_boxplot() +
  theme_minimal() +
  labs(x = "Metric", y = "Value") +
  scale_x_discrete(labels = c("Max Size", "Mean Size")) +
  theme(legend.position="none") +
  scale_color_manual(values=c("darkblue", "darkgreen"))
```



Compute the sobol indicies for each metric

```
# sobol can only handle one output at a time  - so we will need to do them separately

sens_C_maxsize = sensitivity::tell(sens_C,allres$maxsize)

# first-order indices (main effect without co-variance), note: order "r", "K", "g", "thresh" (X1-X4)
sens_C_maxsize$S
```

```
##     original         bias std. error  min. c.i. max. c.i.
## X1 0.6347148 -0.009098655 0.05401106  0.5416485 0.7628170
## X2 0.0638004 -0.010708529 0.09286929 -0.1192533 0.2417569
## X3 0.4485978 -0.009073805 0.07713077  0.3219465 0.6182669
## X4 0.0638004 -0.010708529 0.09286929 -0.1192533 0.2417569
```

```
# total sensitivity index -note that this partitions the output variance - so values sum to 1
sens_C_maxsize$T
```

```
##         original          bias   std. error     min. c.i.    max. c.i.
## X1  5.299965e-01  1.242795e-02 8.624581e-02  3.452955e-01 6.607804e-01
## X2 -2.047251e-13 -1.132061e-13 4.547042e-13 -9.853741e-13 7.863065e-13
## X3  4.555775e-01  1.048174e-02 7.200727e-02  3.035225e-01 5.783270e-01
## X4 -2.047251e-13 -1.132061e-13 4.547042e-13 -9.853741e-13 7.863065e-13
```

```
# create another one for max year
sens_C_meansize = sensitivity::tell(sens_C,allres$meansize)
# first-order indices (main effect without co-variance)
sens_C_meansize$S
```

```
##      original         bias std. error   min. c.i. max. c.i.
## X1 0.93130754 -0.001166484 0.01437809  0.90795466 0.9694954
## X2 0.04356276  0.001972145 0.10449691 -0.15031047 0.2586874
## X3 0.10664659 -0.002076671 0.10696342 -0.09402253 0.3211708
## X4 0.04356276  0.001972145 0.10449691 -0.15031047 0.2586874
```

```
# total sensitivity index -note that this partitions the output variance - so values sum to 1
sens_C_meansize$T
```

```
##        original          bias   std. error    min. c.i.    max. c.i.
## X1 8.703892e-01  6.531283e-04 1.066513e-01 6.293250e-01 1.063247e+00
## X2 1.716405e-13 -1.511276e-13 1.166666e-13 7.046461e-14 5.670301e-13
## X3 9.166402e-02  6.919367e-04 2.045829e-02 4.653976e-02 1.253601e-01
## X4 1.716405e-13 -1.511276e-13 1.166666e-13 7.046461e-14 5.670301e-13
```

Climate change impacts on forest growth

- K is highly sentive under the total sensitivity index for max size. r and K are most sensitive under first-order indices for mean size.
- Climate change will influence the growth rate (r) of forests by influencing forest disturbance events such as wildfires, storms, pest and pathogen outbreaks, drought conditions, and more. Additionally, climate change will influence forest carbon cycling (GPP, NPP, etc.) leading to shifts in the carrying capacity of forest ecosystems. This will ultimately affect the productivity of forests, shifting resource management, economic processes, and forest product harvest.