Sensitivity_Analysis_IPM

Esin Ickin

4/4/2023

This code is an example on how to parameterize vital-rate functions using coefficients and covariate inputs to then build a integral projection model and compute sensitivities of the population growth rates to environmental covaraites in vital-rate models.

1) Vital rates

The example is from a paper on yellow-bellied marmots (Paniw et al. 2020, https://doi.org/10.1111/ele.13459)

In the paper, we parameterized vital rates as a function of a latent climatic variable (Q). Because of the way the model was conceptualized, we considered random year variation as a separate covariate.

The vital rates are defined as functions, and we need the coefficients from vital rate models (here mean posterior values from Bayesian GLMs). We would appreciate a similar input on your part. Alternatively, it possible to have vital-rate models as an object, and then just use the "predict" function, instead of building our own functions.

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
library(boot)

# Coefficients (can be supplied as a list; written as R objects; or directly integrated into the functi

# LOAD GLM BAYESIAN MODEL PARAMETERS
load("~/Desktop/outputBay1_latent3.rda")
```

```
## Warning: namespace 'rjags' is not available and has been replaced
## by .GlobalEnv when processing object 'out1J'
```

```
mcmc=out1J$mean # use mean values here

### Parameter are scaled - need to backscale for the vital rate predictions

m_S1=12.8968
sd_S1=2.23326

m_S2=12.62647
sd_S2=1.556634

m_PR=14.89485
sd_PR=0.8104184

m_R=13.74764
sd_R=0.7182092

### Rescale parameters
```

```
# Winter survival
mcmc[["bc.surv1"]]=mcmc[["bc.surv1"]]/sd_S1
mcmc[["a0.surv1"]]=mcmc[["a0.surv1"]]-mcmc[["bc.surv1"]]*m_S1

# Summer survival
mcmc[["bc.surv2"]]=mcmc[["bc.surv2"]]/sd_S2
mcmc[["a0.surv2"]]=mcmc[["a0.surv2"]]-mcmc[["bc.surv2"]]*m_S2

# Transition to reproductive
mcmc[["bc.pr"]]=mcmc[["bc.pr"]]/sd_PR
mcmc[["a0.pr"]]=mcmc[["a0.pr"]]-mcmc[["bc.pr"]]*m_PR

# Number recruits
mcmc[["bc.rec"]]=mcmc[["bc.rec"]]/sd_R
mcmc[["a0.rec"]]=mcmc[["a0.rec"]]-mcmc[["bc.rec"]]*m_R

# Functions

# The marmot model is seasonal, so we have survival and growth in winter and summer,
# and reproduction in summer

# The input covariates are:

# z - mass
# stage - life-history stage: juvenile (J), yearling (Y), non-reproductive adult (NRA),
# reproductive adult (RA)
# year - one of 40 study years (random year effect)
# Q.sim - latent climatic variable

# SURVIVAL: AUGUST JUNE NEXT

S1.fun <- function(z,stage,year,Q.sim) {

  if(year==0){

    mu.surv=exp(mcmc[["a0.surv1"]]+ mcmc[["bc.surv1"]]*z + mcmc[["a1.surv1"]][stage]+
                  mcmc[["bq.surv1"]]*Q.sim)
  }else{
    mu.surv=exp(mcmc[["a0.surv1"]]+ mcmc[["bc.surv1"]]*z + mcmc[["a1.surv1"]][stage]+
                  mcmc[["aY.surv1"]][year]+
                  mcmc[["bq.surv1"]]*Q.sim)
  }


  return(mu.surv/(1+mu.surv))
}

# SURVIVAL: JUNE AUGUST
S2.fun <- function(z,stage,year,Q.sim) {

  if(year==0){

    mu.surv=exp(mcmc[["a0.surv2"]]+ mcmc[["bc.surv2"]]*z + mcmc[["a1.surv2"]][stage]+
```

```
                  mcmc[["bq.surv2"]]*Q.sim)
  }else{
    mu.surv=exp(mcmc[["a0.surv2"]]+ mcmc[["bc.surv2"]]*z + mcmc[["a1.surv2"]][stage]+
                  mcmc[["aY.surv2"]][year]+
                  mcmc[["bq.surv2"]]*Q.sim)
  }


  return(mu.surv/(1+mu.surv))
}

# GROWTH AUGUST TO JUNE (we assume a contant variance)

GR1.fun <- function(z,zz,stage,year,Q.sim){

  if(year==0){

    growth.mu=(mcmc[["a0.gr"]]+ (mcmc[["bc.gr"]]+mcmc[["bcstage.gr"]][stage])*z +
                 mcmc[["a1.gr"]][stage]+
                 mcmc[["bq.gr"]]*Q.sim)
  }else{
    growth.mu=(mcmc[["a0.gr"]]+ (mcmc[["bc.gr"]]+mcmc[["bcstage.gr"]][stage])*z +
                 mcmc[["a1.gr"]][stage]+
                 mcmc[["aY.gr"]][year]+
                 mcmc[["bq.gr"]]*Q.sim)
  }


  # Get residual variance
  var.res= (mcmc[["sigma.gr"]])^2
  # Density distribution function of the normal distribution
  gr1 = sqrt(2*pi*var.res)
  gr2 = ((zz-growth.mu)^2)/(2*var.res)

  return(exp(-gr2)/gr1)

}

# GROWTH JUNE TO AUGUST (we assume a contant variance)

GR2.fun <- function(z,zz,stage,year,Q.sim){

  if(year==0){

    growth.mu=(mcmc[["a0.gr2"]]+ (mcmc[["bc.gr2"]]+mcmc[["bcstage.gr2"]][stage])*z +
                 mcmc[["a1.gr2"]][stage]+
                 mcmc[["bq.gr2"]]*Q.sim)

  }else{
    growth.mu=(mcmc[["a0.gr2"]]+ (mcmc[["bc.gr2"]]+mcmc[["bcstage.gr2"]][stage])*z +
                 mcmc[["a1.gr2"]][stage]+
                 mcmc[["aY.gr2"]][year]+
                 mcmc[["bq.gr2"]]*Q.sim)
```

```
  }


  # Get residual variance
  var.res= (mcmc[["sigma.gr2"]])^2
  # Density distribution function of the normal distribution
  gr1 = sqrt(2*pi*var.res)
  gr2 = ((zz-growth.mu)^2)/(2*var.res)

  return(exp(-gr2)/gr1)

}


# PROBABILITY OF REPRODUCING:

PR.fun <- function(z,stage,year,Q.sim) {

  if(year==0){

    mu.rep=exp(mcmc[["a0.pr"]]+ mcmc[["bc.pr"]]*z+ mcmc[["a1.pr"]][stage]+
                 mcmc[["bq.pr"]]*Q.sim)

  }else{
    mu.rep=exp(mcmc[["a0.pr"]]+ mcmc[["bc.pr"]]*z+ mcmc[["a1.pr"]][stage]+mcmc[["aY.pr"]][year]+
                 mcmc[["bq.pr"]]*Q.sim)

  }


  return(mu.rep/(1+mu.rep))
}

# Number of recruits:

R.fun <- function(z,year,Q.sim) {

  if(year==0){

    mu.rec=exp(mcmc[["a0.rec"]]+ mcmc[["bc.rec"]]*z+
                 mcmc[["bq.rec"]]*Q.sim)

  }else{
    mu.rec=exp(mcmc[["a0.rec"]]+ mcmc[["bc.rec"]]*z+  mcmc[["aY.rec"]][year]+
                 mcmc[["bq.rec"]]*Q.sim)

  }




  return(mu.rec)
}
```

```
## OFFSPRING MASS

OffMass.fun <- function(z,zz,year,Q.sim){

  if(year==0){

    growth.mu=(mcmc[["a0.off"]]+ mcmc[["bc.off"]]*z+
                 mcmc[["bq.off"]]*Q.sim)

  }else{
    growth.mu=(mcmc[["a0.off"]]+ mcmc[["bc.off"]]*z+  mcmc[["aY.off"]][year]+
                 mcmc[["bq.off"]]*Q.sim)

  }


  # Get residual variance
  var.res= (mcmc[["sigma.off"]])^2
  # Density distribution function of the normal distribution
  gr1 = sqrt(2*pi*var.res)
  gr2 = ((zz-growth.mu)^2)/(2*var.res)

  return(exp(-gr2)/gr1)

}
```

2) Covariates

There are the input data for the vital rate functions. Here, we have a time series of data. This is the best format because it allows us to calculate not only means and variances, but also covariances, the latter being important to calculate scaled sensitivities.

However, it is also possible to send us just the values we are interested in.

```
max.Q = max(mcmc$Q)
min.Q = min(mcmc$Q)
mean.Q = mean(mcmc$Q)
sd.Q = sd(mcmc$Q)

# Covariation
year_when_Q_max = which(mcmc$Q == max.Q)
year_when_Q_min = which(mcmc$Q == min.Q)

# ...

# This type of information is required for all (a)biotic
# covariates in the model (here we only have Q) For an
# example on how to provide more covariates (if you don't
# provide a time series), please see the MPM code example
```

3) Population model

Here, we use the vital rate function to construct an annual population model that can give us the population growth rate (lambda).

In the following example, the IPM is constructed with mean abiotic (Q) covariate values.

In the perturbations, the covariate values are changed.

Please note that this part is very specific to each study. Here, we create a stage-by-mass IPM for winter and summer, and then join them into an annual model.

```
library(popbio)

### Necessary IPM input parameters

# Define the lower and upper integration limit
L = 7.791458  # minimum observed mass (log grams)
U = 17.07776  # maximum observed mass (log grams)

n = 100  # bins

b <- L + c(0:n) * (U - L)/n  # interval that each cell of the matrix covers
z <- 0.5 * (b[1:n] + b[2:(n + 1)])  # midpoint
h = (U - L)/n  # bin width

n.stage = 4

# We formulate the IPM construction as a function, so that
# we can use it later for the perturbations

winter_ipm <- function(year, env) {

    IPMaj = array(0, c(n * n.stage, n * n.stage))  # winter

    ### AUGUST - JUNE

    year = year
    Q.sim = env

    Sj <- diag(S1.fun(z, stage = 1, year, Q.sim))  # Survival juveniles
    Sy <- diag(S1.fun(z, stage = 2, year, Q.sim))  # Survival yearlings
    Snra <- diag(S1.fun(z, stage = 3, year, Q.sim))  # Survival non-reproductive adults
    Sra <- diag(S1.fun(z, stage = 4, year, Q.sim))  # Survival reproductive adults

    # Transition To RA or NRA
    Ty <- diag(PR.fun(z, stage = 1, year, Q.sim))
    Tnra <- diag(PR.fun(z, stage = 2, year, Q.sim))
    Tra <- diag(PR.fun(z, stage = 3, year, Q.sim))

    # Growth - stage specific like for survival
    G <- h * t(outer(z, z, GR1.fun, stage = 1, year, Q.sim))
    # Control for eviction: this is equivalent to
    # redistributing evicted sizes evenly among existing
    # size classes
    G = G/matrix(as.vector(apply(G, 2, sum)), nrow = n, ncol = n,
        byrow = TRUE)
    # FILL IPM
    IPMaj[(n + 1):(2 * n), 1:n] = G %*% Sj  # Juvenile to Yearling
```

```
    G <- h * t(outer(z, z, GR1.fun, stage = 2, year, Q.sim))
    # Control for eviction: this is equivalent to
    # redistributing evicted sizes evenly among existing
    # size classes
    G = G/matrix(as.vector(apply(G, 2, sum)), nrow = n, ncol = n,
        byrow = TRUE)
    # FILL IPM
    IPMaj[(2 * n + 1):(3 * n), (n + 1):(2 * n)] = G %*% (Sy *
        diag(1 - PR.fun(z, stage = 1, year, Q.sim)))  # Yearling to Non-Reproductive Adult
    IPMaj[(3 * n + 1):(4 * n), (n + 1):(2 * n)] = G %*% (Sy *
        Ty)  # Yearling to Reproductive Adult

    G <- h * t(outer(z, z, GR1.fun, stage = 3, year, Q.sim))
    G = G/matrix(as.vector(apply(G, 2, sum)), nrow = n, ncol = n,
        byrow = TRUE)
    # FILL IPM
    IPMaj[(2 * n + 1):(3 * n), (2 * n + 1):(3 * n)] = G %*% (Snra *
        diag(1 - PR.fun(z, stage = 2, year, Q.sim)))  # Non-Reproductive Adult to Non-Reproductive Adul
    IPMaj[(3 * n + 1):(4 * n), (2 * n + 1):(3 * n)] = G %*% (Snra *
        Tnra)  # Non-Reproductive Adult to Reproductive Adult

    G <- h * t(outer(z, z, GR1.fun, stage = 4, year, Q.sim))
    G = G/matrix(as.vector(apply(G, 2, sum)), nrow = n, ncol = n,
        byrow = TRUE)
    # FILL IPM
    IPMaj[(2 * n + 1):(3 * n), (3 * n + 1):(4 * n)] = G %*% (Sra *
        diag(1 - PR.fun(z, stage = 3, year, Q.sim)))  # Reproductive Adult to Non-Reproductive Adult
    IPMaj[(3 * n + 1):(4 * n), (3 * n + 1):(4 * n)] = G %*% (Sra *
        Tra)  # Reproductive Adult to Reproductive Adult

    return(IPMaj)
}

#########################################################

summer_ipm <- function(year, env) {
    IPMja = array(0, c(n * n.stage, n * n.stage))  # summer
    year = year
    Q.sim = env

    # SURVIVAL
    S2y <- diag(S2.fun(z, stage = 1, year, Q.sim), nrow = n,
        ncol = n)  # Survival yearlings
    S2nra <- diag(S2.fun(z, stage = 2, year, Q.sim), nrow = n,
        ncol = n)  # Survival non-reproductive adults
    S2ra <- diag(S2.fun(z, stage = 3, year, Q.sim), nrow = n,
        ncol = n)  # Survival reproductive adults

    # GROWTH

    # Yearlings
    Gy <- h * t(outer(z, z, GR2.fun, stage = 1, year, Q.sim))
```

```
    # Reproductive Adults
    Gnra <- h * t(outer(z, z, GR2.fun, stage = 2, year, Q.sim))

    # Non-Reproductive Adults
    Gra <- h * t(outer(z, z, GR2.fun, stage = 3, year, Q.sim))

    # Control for eviction: this is equivalent to
    # redistributing evictd sizes evenly among existing
    # size classes

    Gy = Gy/matrix(as.vector(apply(Gy, 2, sum)), nrow = n, ncol = n,
        byrow = TRUE)
    Gra = Gra/matrix(as.vector(apply(Gra, 2, sum)), nrow = n,
        ncol = n, byrow = TRUE)
    Gnra = Gnra/matrix(as.vector(apply(Gnra, 2, sum)), nrow = n,
        ncol = n, byrow = TRUE)

    # Offspring mass

    OffMass <- h * t(outer(z, z, OffMass.fun, year, Q.sim))
    # Control for eviction: this is equivalent to
    # redistributing evictd sizes evenly among existing
    # size classes
    OffMass = OffMass/matrix(as.vector(apply(OffMass, 2, sum)),
        nrow = n, ncol = n, byrow = TRUE)

    # recruitment (as individuals that die in August still
    # produce offspring, recruitment does not need to be
    # multiplied by S2)

    Rec = (diag(R.fun(z, year, Q.sim)))/2

    Fkernel <- as.matrix(OffMass %*% (Rec * S2ra))

    # FILL IPM

    IPMja[(n + 1):(2 * n), (n + 1):(2 * n)] = Gy %*% S2y  # Yearling stay Yearling
    IPMja[(2 * n + 1):(3 * n), (2 * n + 1):(3 * n)] = Gnra %*%
        S2nra  # Non-Reproductive Adults stay Non-Reproductive Adults
    IPMja[(3 * n + 1):(4 * n), (3 * n + 1):(4 * n)] = Gra %*%
        S2ra  # Reproductive Adults stay Reproductive Adults

    IPMja[1:n, (3 * n + 1):(4 * n)] = Fkernel  # Adults producing juveniles

    return(IPMja)
}

#### Asymptotic lambda at mean environmental values

K = summer_ipm(0, mean.Q) %*% winter_ipm(0, mean.Q)
lambda(K)
```

```
## [1] 1.083627
```

4) Scaled sensitivity analyses

Here, we calculate scaled sensitivities, according to Morris et al. 2020 (DOI: https://doi.org/10.1073/pnas.1918363117)

Note that this is a step that Esin will implement in her MS thesis. With the information given in 1-3, we should be able to run these analyses.

```
library(popbio)

### 1. Sensitivity to Q assuming 0 year effect

K = summer_ipm(0, min.Q) %*% winter_ipm(0, min.Q)
lambda.min.Q = lambda(K)

K = summer_ipm(0, max.Q) %*% winter_ipm(0, max.Q)
lambda.max.Q = lambda(K)

delta = (lambda.max.Q - lambda.min.Q)/((max.Q - min.Q)/sd.Q)


# 2. Sensitivity to rain assuming covariation among
# covariates
K = summer_ipm(year_when_Q_min, min.Q) %*% winter_ipm(year_when_Q_min,
    min.Q)
lambda.min.Q = lambda(K)

K = summer_ipm(year_when_Q_max, max.Q) %*% winter_ipm(year_when_Q_max,
    max.Q)
lambda.max.Q = lambda(K)

delta.V2 = (lambda.max.Q - lambda.min.Q)/((max.Q - min.Q)/sd.Q)

# Do sensitvities decrease when we consider covariation
# among covariates? This example suggests that no

print(paste("Scaled sensitivity no covariation:", round(delta,
    2)))
```

```
## [1] "Scaled sensitivity no covariation: 0.12"
```

```
print(paste("Scaled sensitivity with covariation:", round(delta.V2,
    2)))
```

```
## [1] "Scaled sensitivity with covariation: 0.17"
```

5) Sensitivity analyses at equilibrium dynamics

Here, we perform "classic" sensitivity analyses (see Paniw et al. 2023; https://doi.org/10.1098/rspb.2022.1494)

1. We look for combinations of covariate values where lambda approaches 1

```
# go through different combinations of covariates
cov = expand.grid(Q = seq(min.Q, max.Q, length.out = 20), year = 1:40)

# empty object for lambdas
all.l = rep(NA, nrow(cov))
```

```
for (s in 1:nrow(cov)) {

    K = summer_ipm(cov$year[s], cov$Q[s]) %*% winter_ipm(cov$year[s],
        cov$Q[s])
    all.l[s] = lambda(K)
}
stable.l = which(all.l > 0.99 & all.l < 1.01)
```

2. We perturb Q for each vital rate in turn, while maintaining the year effect unperturbed from stable dynamics

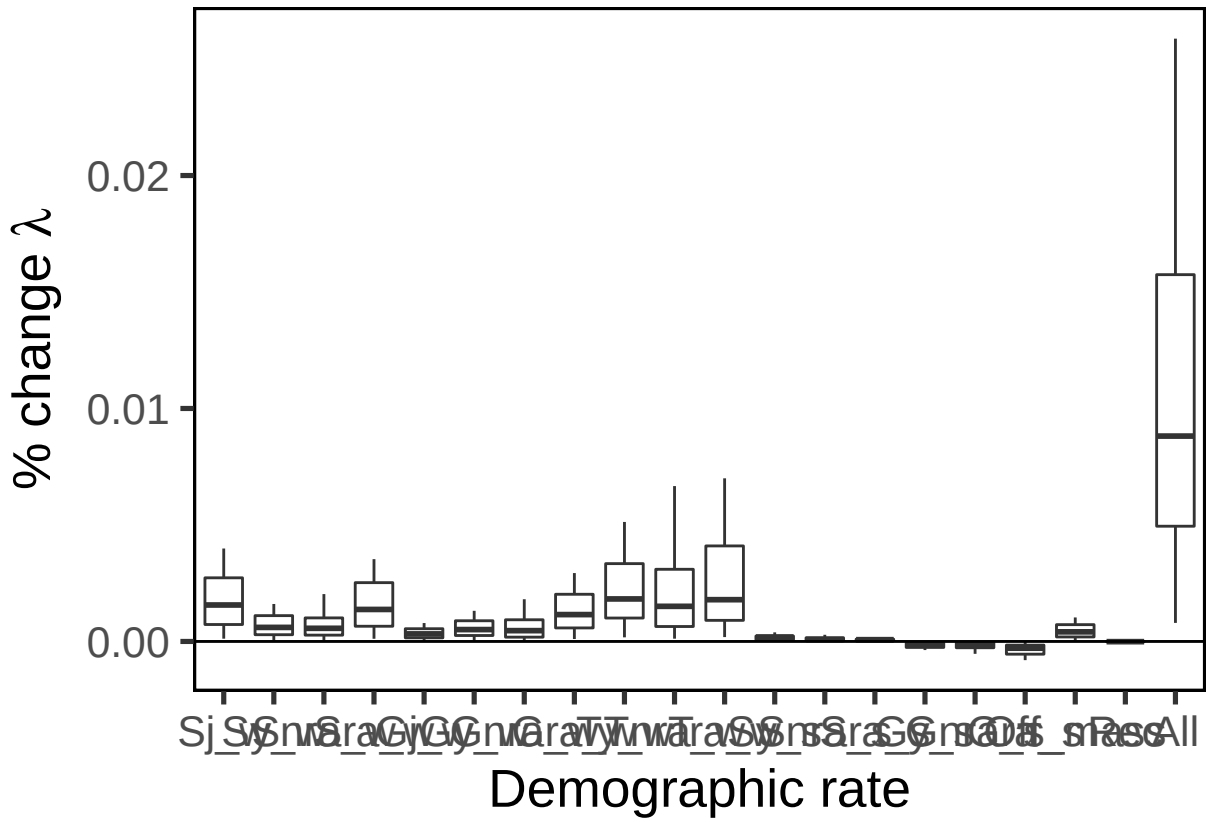
```
dQ = NULL

source("~/Desktop/pert.Sj.winter.R")
source("~/Desktop/pert.Sy.winter.R")
source("~/Desktop/pert.Snra.winter.R")
source("~/Desktop/pert.Sra.winter.R")

source("~/Desktop/pert.Ty.winter.R")
source("~/Desktop/pert.Tnra.winter.R")
source("~/Desktop/pert.Tra.winter.R")

source("~/Desktop/pert.Gj.winter.R")
source("~/Desktop/pert.Gy.winter.R")
source("~/Desktop/pert.Gnra.winter.R")
source("~/Desktop/pert.Gra.winter.R")

source("~/Desktop/pert.Sy.summer.R")
source("~/Desktop/pert.Snra.summer.R")
source("~/Desktop/pert.Sra.summer.R")

source("~/Desktop/pert.Gy.summer.R")
source("~/Desktop/pert.Gnra.summer.R")
source("~/Desktop/pert.Gra.summer.R")

source("~/Desktop/pert.Off.summer.R")
source("~/Desktop/pert.Rec.summer.R")

for (s in 1:length(stable.l)) {

    Q = cov$Q[stable.l[s]]
    Q.pert = cov$Q[stable.l[s]] + 0.1 * abs(cov$Q[stable.l[s]])

    control = summer_ipm(cov$year[stable.l[s]], Q) %*% winter_ipm(cov$year[stable.l[s]],
        Q)

    pert.Sj.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
        winter_pertSj_ipm(cov$year[stable.l[s]], Q, Q.pert)

    pert.Sy.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
        winter_pertSy_ipm(cov$year[stable.l[s]], Q, Q.pert)

    pert.Snra.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
        winter_pertSnra_ipm(cov$year[stable.l[s]], Q, Q.pert)
```

```
pert.Sra.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertSra_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Ty.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertTy_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Tnra.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertTnra_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Tra.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertTra_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Gj.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertGj_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Gy.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertGy_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Gnra.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertGnra_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Gra.winter = summer_ipm(cov$year[stable.l[s]], Q) %*%
    winter_pertGra_ipm(cov$year[stable.l[s]], Q, Q.pert)

pert.Sy.summer = summer_pertSy_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Snra.summer = summer_pertSnra_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Sra.summer = summer_pertSra_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Gy.summer = summer_pertGy_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Gnra.summer = summer_pertGnra_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Gra.summer = summer_pertGra_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Off.summer = summer_pertOff_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.Rec.summer = summer_pertRec_ipm(cov$year[stable.l[s]],
    Q, Q.pert) %*% winter_ipm(cov$year[stable.l[s]], Q)

pert.all = summer_ipm(cov$year[stable.l[s]], Q.pert) %*%
    winter_ipm(cov$year[stable.l[s]], Q.pert)

temp = data.frame(vr = c("Sj_w", "Sy_w", "Snra_w", "Sra_w",
    "Gj_w", "Gy_w", "Gnra_w", "Gra_w", "Ty_w", "Tnra_w",
```

```
        "Tra_w", "Sy_s", "Snra_s", "Sra_s", "Gy_s", "Gnra_s",
        "Gra_s", "Off_mass", "Rec", "All"), delta = c((lambda(pert.Sj.winter) -
        lambda(control))/lambda(control), (lambda(pert.Sy.winter) -
        lambda(control))/lambda(control), (lambda(pert.Snra.winter) -
        lambda(control))/lambda(control), (lambda(pert.Sra.winter) -
        lambda(control))/lambda(control), (lambda(pert.Gj.winter) -
        lambda(control))/lambda(control), (lambda(pert.Gy.winter) -
        lambda(control))/lambda(control), (lambda(pert.Gnra.winter) -
        lambda(control))/lambda(control), (lambda(pert.Gra.winter) -
        lambda(control))/lambda(control), (lambda(pert.Ty.winter) -
        lambda(control))/lambda(control), (lambda(pert.Tnra.winter) -
        lambda(control))/lambda(control), (lambda(pert.Tra.winter) -
        lambda(control))/lambda(control), (lambda(pert.Sy.summer) -
        lambda(control))/lambda(control), (lambda(pert.Snra.summer) -
        lambda(control))/lambda(control), (lambda(pert.Sra.summer) -
        lambda(control))/lambda(control), (lambda(pert.Gy.summer) -
        lambda(control))/lambda(control), (lambda(pert.Gnra.summer) -
        lambda(control))/lambda(control), (lambda(pert.Gra.summer) -
        lambda(control))/lambda(control), (lambda(pert.Off.summer) -
        lambda(control))/lambda(control), (lambda(pert.Rec.summer) -
        lambda(control))/lambda(control), (lambda(pert.all) -
        lambda(control))/lambda(control)))

    dQ = rbind(dQ, temp)
}


# put all results into a dataframe
dQ$vr = factor(dQ$vr, levels = c("Sj_w", "Sy_w", "Snra_w", "Sra_w",
    "Gj_w", "Gy_w", "Gnra_w", "Gra_w", "Ty_w", "Tnra_w", "Tra_w",
    "Sy_s", "Snra_s", "Sra_s", "Gy_s", "Gnra_s", "Gra_s", "Off_mass",
    "Rec", "All"))
dQ$delta[is.na(dQ$delta)] = 0
```

3. Plot the sensitivities

```
library(ggplot2)

ggplot(dQ,aes(vr,delta))+
  geom_boxplot(outlier.shape = NA)+
  geom_hline(yintercept=0,
             color = "black", size=0.5)+
  ylab(expression(paste("% change ", lambda)))+xlab("Demographic rate")+theme_bw(base_size=20)+
  theme(panel.grid = element_blank())+
  labs(fill='Perturbed',colour='Perturbed')+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        strip.background = element_blank(),
        strip.text = element_text(face = "italic"),
        legend.position = c(0.20,0.70), # might have to change legend position
        legend.background = element_rect(color=NA),
        panel.border = element_rect(colour = "black"))
```


% change λ
0.02
0.01
0.00
Demographic rate