Fusing Times-Series Data: Tree Rings and Forest Inventory

In this exercise we will extend the state-space framework to combine multiple data streams with different observation errors and to separate observation error from process error. We will also demonstrate how to add hierarchical random effects to partition the process error into multiple sources.

Specifically, we will be building up to the model presented by Clark et al. 2007 Ecological Applications that combines tree ring data with forest inventory data. Unlike the original model, which was written all in R, we will rewrite this model into JAGS, which makes it easier to see what is going on and to modify the model. In this exercise we will utilize data from a collection of small plots at the Harvard Forest, Petersham, MA.

We will divide this analysis into a number of steps, which we will encapsulate into functions to make them easier to understand and run. Thus we will begin by defining these functions. Specifically, the steps will be:

1. load forest inventory data
2. load tree ring data
3. match the tree core and inventory data for individual trees and merge these data sets into one data frame
4. format this data into a list for input into JAGS
5. run the JAGS model
6. visualize the output

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Warning: package 'coda' was built under R version 4.3.2
```

```
## Linked to JAGS 4.3.2
```

```
## Loaded modules: basemod,bugs
```

```
library(ecoforecastR)
```

Steps 1-4 have already been done for you and leverages functions that are part of the PEcAn system. Specifically, they are within PEcAn's land data R package, which can be downloaded and installed off Github using `remotes` or `devtools`. Because these steps have been done fore you, you don't need to run the following chunk of code.

```
if(!require(PEcAn.data.land)){
  library(remotes)
  install.packages(c("digest","dplR","PeriodicTable"),repos = "https://cloud.r-project.org")
  remotes::install_github("PecanProject/pecan/base/logger")
  remotes::install_github("PecanProject/pecan/base/remote")
  remotes::install_github("PecanProject/pecan/base/utils")
  remotes::install_github("PecanProject/pecan/base/db")
  remotes::install_github("PecanProject/pecan/modules/data.land")
  library(PEcAn.data.land)
}

## 1. Read tree data
trees <- read.csv("data/H2012AdultFieldData.csv")
```

```
## 2. Read tree ring data
rings <- Read_Tucson("data/TUCSON/")

## 3. merge inventory and tree ring data, extract most recent nyears
combined <- matchInventoryRings(trees,rings,nyears=15)

## take a look at the first few rows of data to see the structure
knitr::kable(combined[1:5,])

## 4. organize data into a list
data <- buildJAGSdata_InventoryRings(combined)
```

Instead we'll just load and investigate the final, prepared data object

```
load("data/Activity07.RData")
# y = increment (tree x year)
# z = dbh (tree x year)
# ni = number of individuals
# nt = number of time points
# make sure to take a look at all the priors!
str(data)
```

```
## List of 13
##  $ y     : num [1:80, 1:15] 0.487 0.245 0.188 0.196 0.125 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:80] "H6A1501" "H6A1502" "H6A1505" "H6A1506" ...
##   .. ..$ : chr [1:15] "1998" "1999" "2000" "2001" ...
##  $ z     : num [1:80, 1:15] NA NA NA NA NA NA NA NA NA NA ...
##  $ ni    : int 80
##  $ nt    : int 15
##  $ x_ic  : num 1
##  $ tau_ic: num 1e-04
##  $ a_dbh : num 16
##  $ r_dbh : num 8
##  $ a_inc : num 0.001
##  $ r_inc : num 1
##  $ a_add : num 1
##  $ r_add : num 1
##  $ time  : num [1:15] 1998 1999 2000 2001 2002 ...
```

```
knitr::kable(head(data$y))
```

	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
H6A1501	4866	0.2714	0.2600	0.2602	0.2216	0.2606	0.1740	0.1902	0.2638	0.2862	0.2506	0.2762	0.2214	0.1372	0.1248
H6A1502	2454	0.1162	0.2132	0.1900	0.2774	0.1834	0.2168	0.3648	0.3026	0.4198	0.4132	0.3206	0.1488	0.4208	0.3880
H6A1505	1878	0.1500	0.1734	0.1312	0.0914	0.1644	0.1058	0.1396	0.0762	0.0930	0.0466	0.0866	0.0550	0.0360	0.0466
H6A1506	1958	0.1190	0.1122	0.1916	0.1642	0.1530	0.1608	0.1270	0.2286	0.0970	0.1650	0.0878	0.2056	0.0722	0.1454
H6A1507	1248	0.1072	0.0888	0.0816	0.0544	0.1440	0.1038	0.3684	0.3402	0.3498	0.2688	0.2746	0.2554	0.2212	0.1392
H6A1508	1076	0.0920	0.0920	0.0382	0.0888	0.0888	0.0606	0.0412	0.0792	0.0506	0.0792	0.1396	0.0634	0.0820	0.0626

```
knitr::kable(head(data$z))
```

NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	24.80	24.75
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	24.20	24.50
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	16.75	16.75
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	17.20	17.20
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	30.10	30.40
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	15.95	16.10

Now that we have the data prepped we need to fit the model itself. The bulk of this code is just the same JAGS syntax we've used before, so lets focus on the JAGS code itself. To begin with, lets look back at the JAGS code for the random walk

```
model{

  #### Data Model
  for(i in 1:n){
    y[i] ~ dnorm(x[i],tau_obs)
  }

  #### Process Model
  for(i in 2:n){
    x[i]~dnorm(x[i-1],tau_add)
  }

  #### Priors
  x[1] ~ dnorm(x_ic,tau_ic)
  tau_obs ~ dgamma(a_obs,r_obs)
  tau_add ~ dgamma(a_add,r_add)
}
```

Since we're fusing two data sources, we'll need to add a second data model. We'll also modify our process model to include a mean growth rate term. Finally, we'll need to specify priors on both observation errors, the process error, and the mean.

```
model{

  #### Data Model: DBH
  for(i in 1:n){
    z[i] ~ dnorm(x[i],tau_dbh)
  }

  #### Data Model: growth
  for(i in 2:n){
    inc[i] <- x[i]-x[i-1]
    y[i] ~ dnorm(inc[i],tau_inc)
  }

  #### Process Model
  #### Dnew is the expected new diameter given the previous diameter, x[i-1], and the mean growth rate,
  for(i in 2:n){
```

```
    Dnew[i] <- x[i-1] + mu
    x[i]~dnorm(Dnew[i],tau_add)
  }

  #### Priors
  x[1] ~ dnorm(x_ic,tau_ic)     ## initial DBH
  tau_dbh ~ dgamma(a_dbh,r_dbh) ## observation error: DBH
  tau_inc ~ dgamma(a_inc,r_inc) ## observation error: tree rings
  tau_add ~ dgamma(a_add,r_add) ## process error: growth
  mu ~ dnorm(0.5,0.5)           ## mean growth
}
```

This code would work perfectly if we only had only measured a single tree, but we measured a number of trees so next need to modify the code to work with tree-by-year matrices of DBH and growth.

```
model{

  ### Loop over all individuals
  for(i in 1:ni){

    #### Data Model: DBH
    for(t in 1:nt){
      z[i,t] ~ dnorm(x[i,t],tau_dbh)
    }

    #### Data Model: growth
    for(t in 2:nt){
      inc[i,t] <- x[i,t]-x[i,t-1]
      y[i,t] ~ dnorm(inc[i,t],tau_inc)
    }

    #### Process Model
    for(t in 2:nt){
      Dnew[i,t] <- x[i,t-1] + mu
      x[i,t]~dnorm(Dnew[i,t],tau_add)
    }

    x[i,1] ~ dnorm(x_ic,tau_ic)
  }  ## end loop over individuals

  #### Priors
  tau_dbh ~ dgamma(a_dbh,r_dbh)
  tau_inc ~ dgamma(a_inc,r_inc)
  tau_add ~ dgamma(a_add,r_add)
  mu ~ dnorm(0.5,0.5)
}
```

Finally, since growth is indexed by both tree and year, lets add random effects for both individuals and years. In this case our process model now becomes

```
Dnew[i,t] <- x[i,t-1] + mu + ind[i] + year[t]
```

where `ind` and `year` are the random effects for individual and year respectively. Next, we'll need to specify the distributions that these random effects are drawn from, as well as the priors on the random effect variances

```
model{

### Loop over all individuals
for(i in 1:ni){

  #### Data Model: DBH
  for(t in 1:nt){
    z[i,t] ~ dnorm(x[i,t],tau_dbh)
  }

  #### Data Model: growth
  for(t in 2:nt){
    inc[i,t] <- x[i,t]-x[i,t-1]
    y[i,t] ~ dnorm(inc[i,t],tau_inc)
  }

  #### Process Model
  for(t in 2:nt){
    Dnew[i,t] <- x[i,t-1] + mu + ind[i] + year[t]
    x[i,t]~dnorm(Dnew[i,t],tau_add)
  }

  ## individual effects
  ind[i] ~ dnorm(0,tau_ind)

  ## initial condition
  x[i,1] ~ dnorm(x_ic,tau_ic)

}  ## end loop over individuals

  ## year effects
  for(t in 1:nt){
    year[t] ~ dnorm(0,tau_yr)
  }


  #### Priors
  tau_dbh ~ dgamma(a_dbh,r_dbh)
  tau_inc ~ dgamma(a_inc,r_inc)
  tau_add ~ dgamma(a_add,r_add)
  tau_ind ~ dgamma(1,0.1)
  tau_yr  ~ dgamma(1,0.1)
  mu ~ dnorm(0.5,0.5)

  }
```

Putting this all together gives the following R code for the base case (no random effects)

```
n.iter = 20000                          ## INCREASE THIS NUMBER FOR ACTUAL ANALYSES ******************

## this code fuses forest inventory data with tree growth data (tree ring or dendrometer band)
## for the same plots. Code is a rewrite of Clark et al 2007 Ecol Appl into JAGS
TreeDataFusionMV = "
model{
```

```
  ### Loop over all individuals
  for(i in 1:ni){

    #### Data Model: DBH
    for(t in 1:nt){
      z[i,t] ~ dnorm(x[i,t],tau_dbh)
    }

    #### Data Model: growth
    for(t in 2:nt){
      inc[i,t] <- x[i,t]-x[i,t-1]
      y[i,t] ~ dnorm(inc[i,t],tau_inc)
    }

    #### Process Model
    for(t in 2:nt){
      Dnew[i,t] <- x[i,t-1] + mu
      x[i,t]~dnorm(Dnew[i,t],tau_add)
    }

    x[i,1] ~ dnorm(x_ic,tau_ic)
  }  ## end loop over individuals

  #### Priors
  tau_dbh ~ dgamma(a_dbh,r_dbh)
  tau_inc ~ dgamma(a_inc,r_inc)
  tau_add ~ dgamma(a_add,r_add)
  mu ~ dnorm(0.5,0.5)
}"

  ## state variable initial condition (subtract observed diameter increments off from the observed diam
  z0 = data$z
  for(i in 1:data$ni){
    z0[i,] = data$z[i,ncol(data$z)] - rev(cumsum(rev(data$y[i,])))
  }

  ## JAGS initial conditions
  nchain = 3
  init <- list()
  for(i in 1:nchain){
    y.samp = sample(data$y,length(data$y),replace=TRUE)
    init[[i]] <- list(x = z0,
                      tau_add=runif(1,1,5)/var(diff(y.samp),na.rm=TRUE), ## process precision
                      tau_dbh=1,   ## DBH obs precision
                      tau_inc=500, ## tree ring obs precision
                      tau_ind=50,  ## individual effect
                      tau_yr=100)  ## year effect
    init[[i]]$year = rnorm(data$nt,0,1/sqrt(init[[i]]$tau_yr)) ## use tau_yr to draw random year effect
    init[[i]]$ind  = rnorm(data$ni,0,1/sqrt(init[[i]]$tau_ind)) ## use tau_ind to draw random individua
  }

  ## compile JAGS model
  j.model   <- jags.model (file = textConnection(TreeDataFusionMV),
```

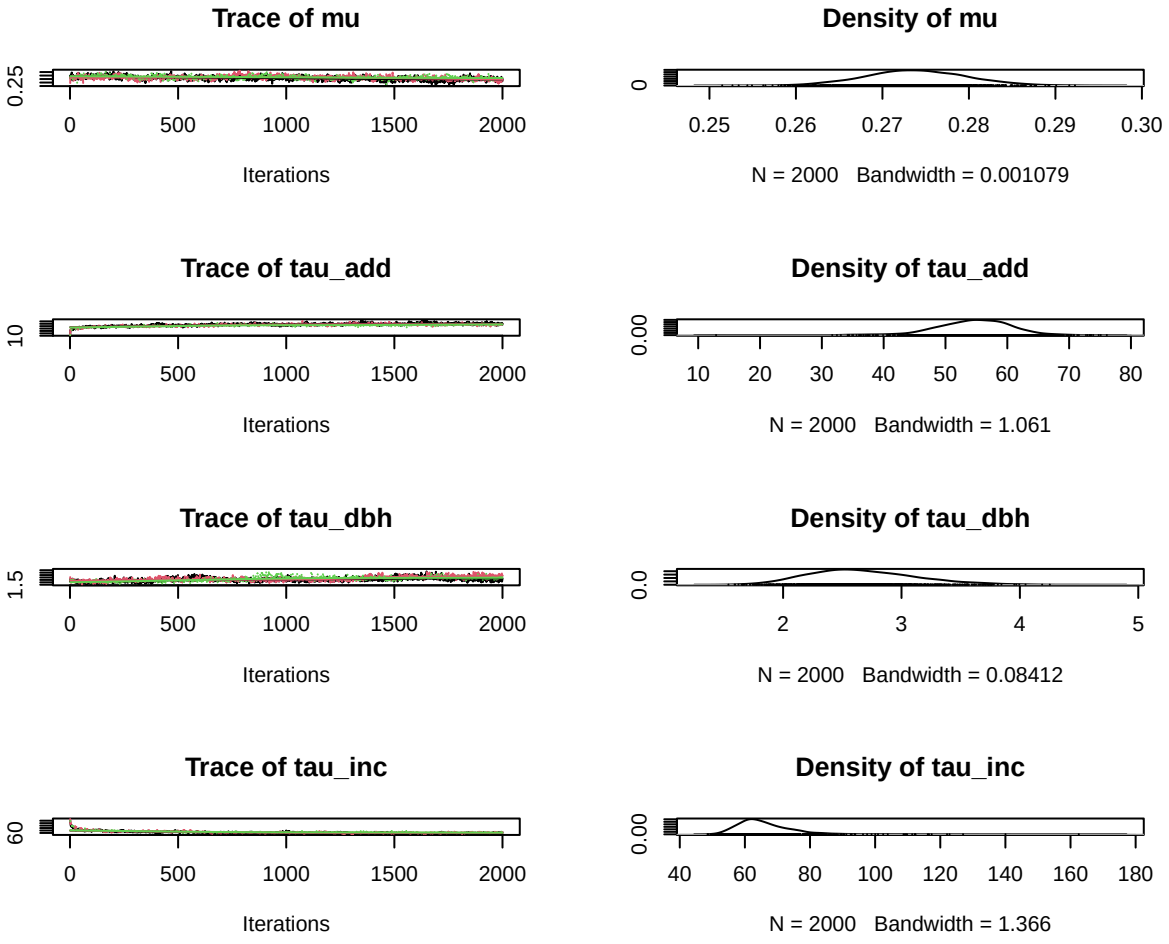
```
                            data = data,
                            inits = init,
                            n.chains = 3)
```

```
## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused variable "time" in data

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1274
##    Unobserved stochastic nodes: 2250
##    Total graph size: 5855

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "tau_ind" in chain 1

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "tau_yr" in chain 1

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "year" in chain 1

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "ind" in chain 1

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "tau_ind" in chain 2

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "tau_yr" in chain 2

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "year" in chain 2

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "ind" in chain 2

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "tau_ind" in chain 3

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "tau_yr" in chain 3

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "year" in chain 3

## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused initial value for "ind" in chain 3

## Initializing model
```

```
## note: this will give a WARNING (not an error) about unused tau_ind, tau_yr, ind, yr
## because we haven't included those to the JAGS code yet


## burn-in
jags.out   <- coda.samples (model = j.model,
                            variable.names = c("tau_add","tau_dbh","tau_inc","mu","tau_ind","tau_yr")
                            n.iter = min(n.iter,2000))
```

```
## Warning in FUN(X[[i]], ...): Failed to set trace monitor for tau_ind
## Variable tau_ind not found

## Warning in FUN(X[[i]], ...): Failed to set trace monitor for tau_yr
## Variable tau_yr not found
```

```
## note: will also give a WARNING about tau_ind, tau_yr
par(mfrow=c(3,3))
plot(jags.out)
```




```
## run MCMC
jags.out   <- coda.samples (model = j.model,
                            variable.names = c("x","tau_add","tau_dbh","tau_inc","mu",
```

```
                                                    "tau_ind","tau_yr","ind","year"),
                                n.iter = n.iter)
```

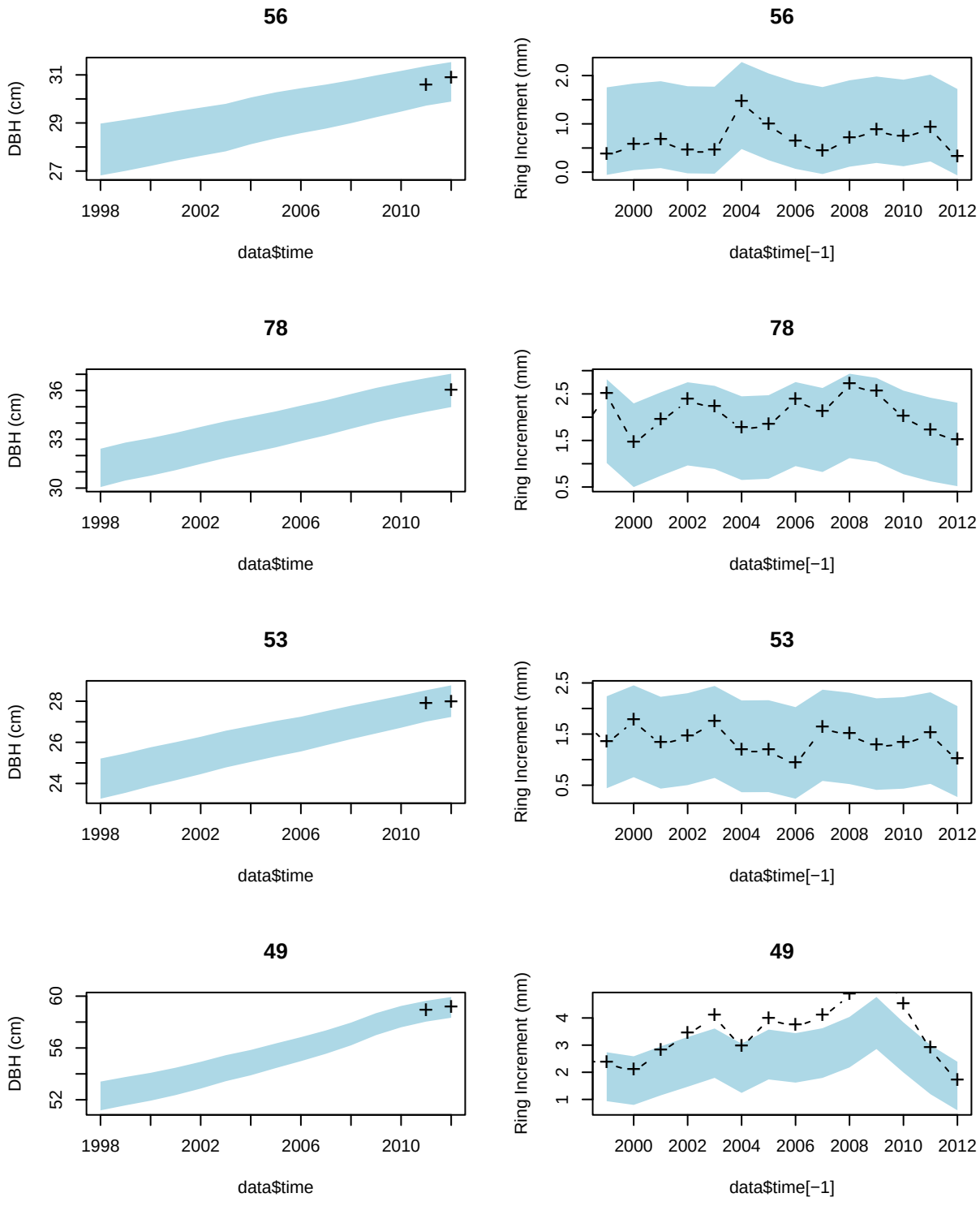
```
## Warning in FUN(X[[i]], ...): Failed to set trace monitor for tau_ind
## Variable tau_ind not found

## Warning in FUN(X[[i]], ...): Failed to set trace monitor for tau_yr
## Variable tau_yr not found

## Warning in FUN(X[[i]], ...): Failed to set trace monitor for ind
## Variable ind not found

## Warning in FUN(X[[i]], ...): Failed to set trace monitor for year
## Variable year not found
```

Next, lets generate some diagnostic plots to look at the model. First, lets plot the posterior CI for growth and DBH and compare these to observations. Since we have scores of cores and trees, we'll pick a random subset of trees to check.

One thing that's critical to note is that for the confidence intervals on growth that these are calculated **pathwise** – we're looking at the growth from a whole MCMC iteration – rather than pairwise (i.e. subtracting the posterior distribution for DBH at one point from the posterior distribution of DBH at the next). Because there's high correlations between successive time points, the pathwise uncertainty estimates are considerably lower in uncertainty – essentially saying that we know can know the growth rate of the tree better than we can know the actual size of the tree

```
#### Helper function to parse JAGS variable names that include matrix syntax (e.g. "x[40,13]")
##' @param w mcmc object containing matrix outputs
##' @param pre prefix (variable name) for the matrix variable to be extracted
##' @param numeric boolean, whether to coerce class to numeric
parse.MatrixNames <- function(w, pre = "x", numeric = FALSE) {
  w <- sub(pre, "", w)
  w <- sub("[", "", w, fixed = TRUE)
  w <- sub("]", "", w, fixed = TRUE)
  w <- matrix(unlist(strsplit(w, ",")), nrow = length(w), byrow = TRUE)
  if (numeric) {
    class(w) <- "numeric"
  }
  colnames(w) <- c("row", "col")
  return(as.data.frame(w))
} # parse.MatrixNames

  #### Diagnostic plots

  ### DBH
  layout(matrix(1:8,4,2)) ## arrange plots in a 4 x 2 matrix
  out <- as.matrix(jags.out)
  x.cols = which(substr(colnames(out),1,1)=="x")   ## which columns are the state variable, x
  ci <- apply(out[,x.cols],2,quantile,c(0.025,0.5,0.975))
  ci.names = parse.MatrixNames(colnames(ci),numeric=TRUE)

  smp = c(sample.int(data$ni,3),49)  ## I've rigged the sampling to make sure you see tree 49!
```

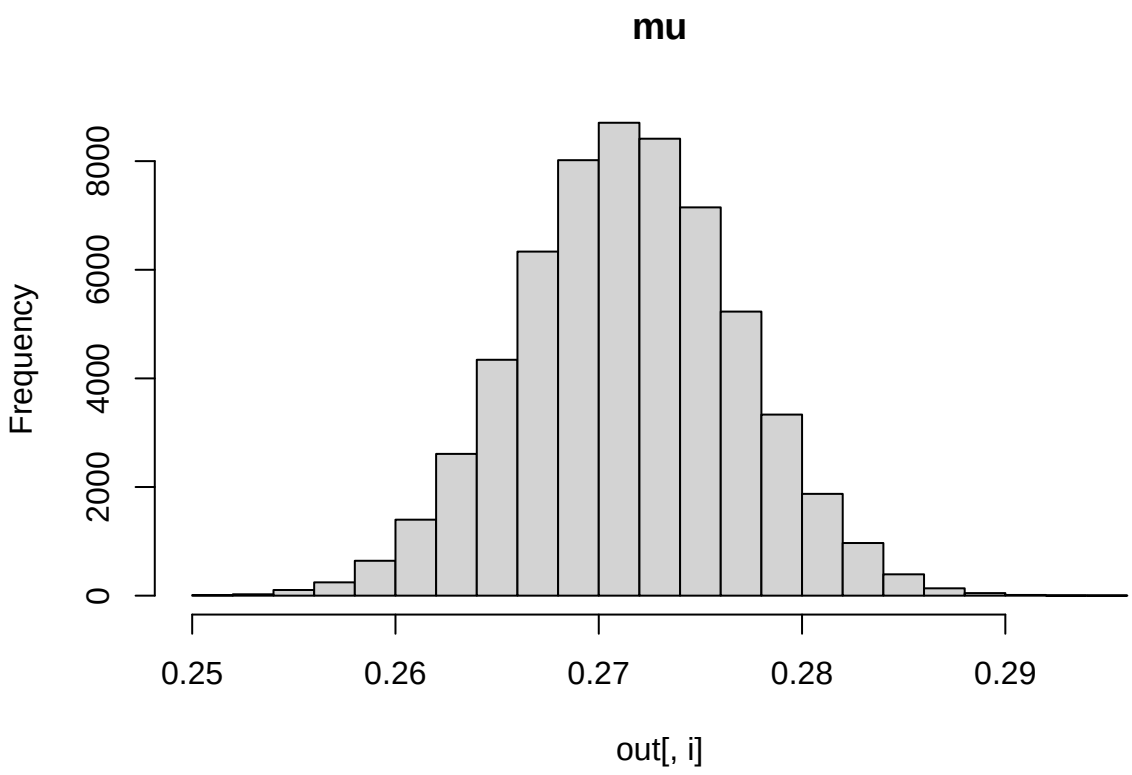
```
## plot sample of DBH timeseries
for(i in smp){
  sel = which(ci.names$row == i)
  plot(data$time,ci[2,sel],type='n',ylim=range(ci[,sel],na.rm=TRUE),ylab="DBH (cm)",main=i)
  ciEnvelope(data$time,ci[1,sel],ci[3,sel],col="lightBlue")
  points(data$time,data$z[i,],pch="+",cex=1.5)
}

## sample of growth timeseries
for(i in smp){
  sel = which(ci.names$row == i)
  inc.mcmc = apply(out[,x.cols[sel]],1,diff)
  inc.ci = apply(inc.mcmc,1,quantile,c(0.025,0.5,0.975))*5

  plot(data$time[-1],inc.ci[2,],type='n',ylim=range(inc.ci,na.rm=TRUE),ylab="Ring Increment (mm)",mai
  ciEnvelope(data$time[-1],inc.ci[1,],inc.ci[3,],col="lightBlue")
  points(data$time,data$y[i,]*5,pch="+",cex=1.5,type='b',lty=2)
}
```



Second, let's look at the histogram of our fixed effect, mu, and the precisions. Let's also convert the precisions to standard deviations to make them easier to interpret

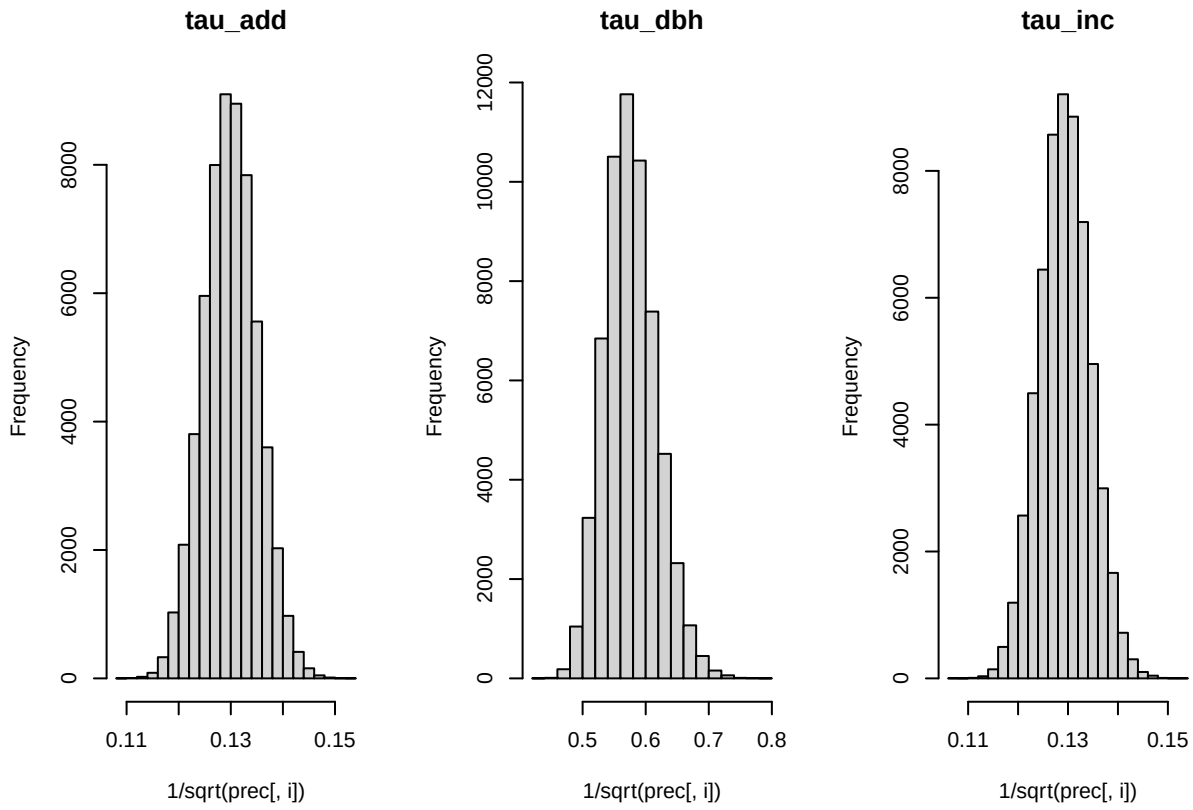
```
## process model
vars = (1:ncol(out))[-c(which(substr(colnames(out),1,1)=="x"),grep("tau",colnames(out)),
                        grep("year",colnames(out)),grep("ind",colnames(out)))] ## remove x, tau's, yr
```

```
par(mfrow=c(1,1))
for(i in vars){
  hist(out[,i],main=colnames(out)[i])
}
```

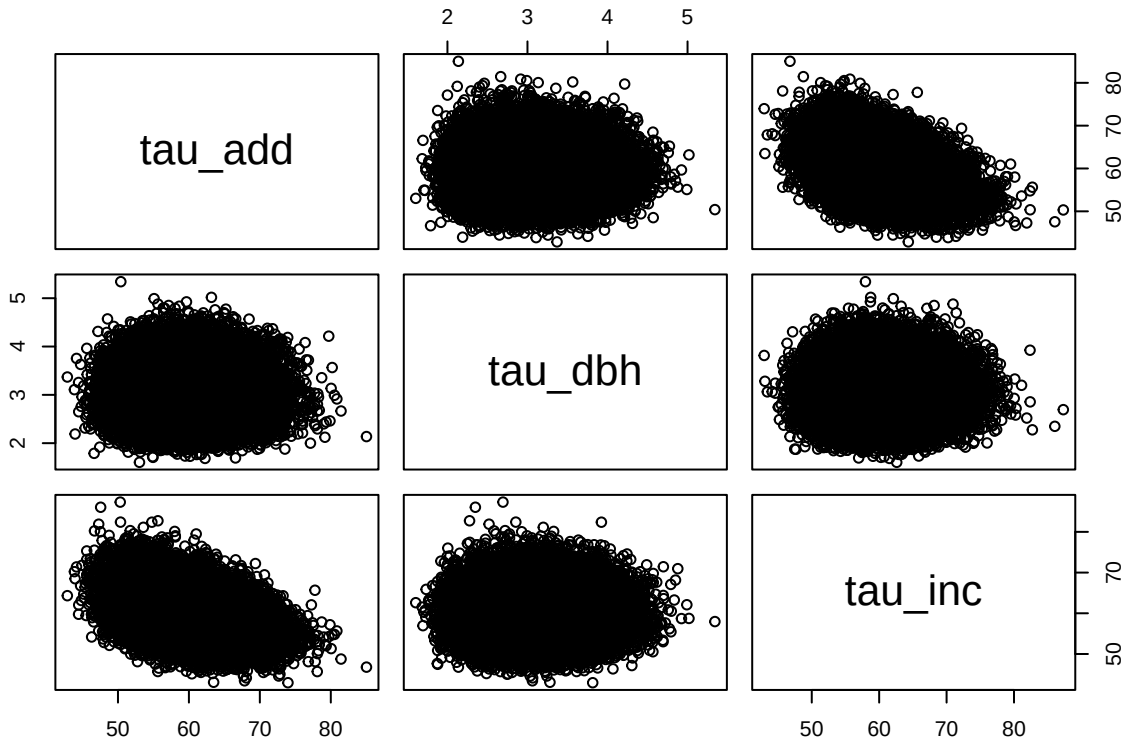



```
if(length(vars)>1) pairs(out[,vars])
```

```
  ## Standard Deviations
par(mfrow=c(1,3))
prec = out[,grep("tau",colnames(out))] ## select just taus to plot
  for(i in 1:ncol(prec)){
  hist(1/sqrt(prec[,i]),main=colnames(prec)[i])
}
```


tau_add
tau_dbh
tau_inc
Frequency
0
2000
4000
6000
8000
10000
12000
0.11
0.13
0.15
0.5
0.6
0.7
0.8
1/sqrt(prec[, i])

```
pairs(prec)
```


tau_add
tau_dbh
tau_inc
2
3
4
5
50
60
70
80

```
cor(prec)
```

```
##             tau_add      tau_dbh      tau_inc
## tau_add  1.00000000 -0.015759456 -0.424327609
## tau_dbh -0.01575946  1.000000000  0.009124899
## tau_inc -0.42432761  0.009124899  1.000000000
```

Third, let's look at the random effects. It is easy enough to plot the year effects by year. For the individual effects we'll plot these twice, first ordering the effects by plot and the second ordering them by species. Note that this chunk of code will not plot ANYTHING until the year and/or individual effects are added to the model.

```
  load("data/combined.07.RData")
  par(mfrow=c(1,1))
  ### YEAR
  year.cols = grep("year",colnames(out))
  if(length(year.cols>0)){
    ci.yr <- apply(out[,year.cols],2,quantile,c(0.025,0.5,0.975))
    plot(data$time,ci.yr[2,],type='n',ylim=range(ci.yr,na.rm=TRUE),main="Year Effect",ylab="cm")
    ciEnvelope(data$time,ci.yr[1,],ci.yr[3,],col="lightBlue")
    lines(data$time,ci.yr[2,],lty=1,lwd=2)
    abline(h=0,lty=2)
  }
```

```
  ### INDIV
  ind.cols= which(substr(colnames(out),1,3)=="ind")

  if(length(ind.cols)>0){
    boxplot(out[,ind.cols],horizontal=TRUE,outline=FALSE,col=combined$PLOT,main="Individual Effects By
    abline(v=0,lty=2)
    ## calculate plot-level means for random effects
    tapply(apply(out[,ind.cols],2,mean),combined$PLOT,mean)
    table(combined$PLOT)

  }
```

```
  if(length(ind.cols)>0){
    unique_species <- unique(combined$SPP)
    colors_for_species <- rainbow(length(unique_species))
    species_colors <- setNames(colors_for_species, unique_species)
    boxplot(out[order(combined$SPP), ind.cols], horizontal=TRUE, outline=FALSE,
            col = species_colors[combined$SPP[order(combined$SPP)]],
            main="Individual Effects By Species", xlab="cm")
    abline(v=0, lty=2)
    legend("bottomright", legend=rev(unique_species),
          col=rev(species_colors[unique_species]), lwd=4)
    tapply(apply(out[,ind.cols], 2, mean), combined$SPP, mean)
  }
```

```
summary_stats <- summary(jags.out)
vars_of_interest <- c("tau_dbh", "tau_inc", "tau_add", "tau_ind", "tau_yr")
for (var in vars_of_interest) {
```

```
  cat("Summary for", var, ":\n")
  print(summary_stats$statistics[rownames(summary_stats$statistics) == var, ])
  cat("\n")
}
```

```
## Summary for tau_dbh :
##           Mean             SD       Naive SE Time-series SE
##    3.056583958    0.429470821    0.001753307    0.019300895
##
## Summary for tau_inc :
##           Mean             SD       Naive SE Time-series SE
##    60.04452003     4.75714784     0.01942097     0.15457284
##
## Summary for tau_add :
##           Mean             SD       Naive SE Time-series SE
##    59.48559633     4.70870690     0.01922322     0.15449077
##
## Summary for tau_ind :
##      Mean SD Naive SE Time-series SE
##
## Summary for tau_yr :
##      Mean SD Naive SE Time-series SE
```

By default this code is set to run with a small number of years (15), and a much too low number of MCMC iterations (500), just so that the code with "knit" quickly initially. **For your analyses you should obviously increase these** – I found that convergence was adequate with around 20,000 samples, though I probably would run 10x longer than that for a publishable analysis. However, such an analysis would take hours to run.

Assignment:

1. Run the model initially with random effects off. [A]
2. Rerun the model with both random effects on.

- Make sure to check convergence, burn-in, and effective sample size. [A]
- Compare model fit and parameter estimates to the previous run.[A]

```
n.iter = 20000                          ## INCREASE THIS NUMBER FOR ACTUAL ANALYSES ******************

## this code fuses forest inventory data with tree growth data (tree ring or dendrometer band)
## for the same plots. Code is a rewrite of Clark et al 2007 Ecol Appl into JAGS
TreeDataFusionMV = "
model{

### Loop over all individuals
for(i in 1:ni){

  #### Data Model: DBH
  for(t in 1:nt){
    z[i,t] ~ dnorm(x[i,t],tau_dbh)
  }
```

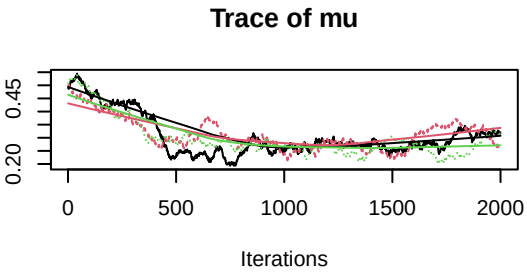
```
  #### Data Model: growth
  for(t in 2:nt){
    inc[i,t] <- x[i,t]-x[i,t-1]
    y[i,t] ~ dnorm(inc[i,t],tau_inc)
  }

  #### Process Model
  for(t in 2:nt){
    Dnew[i,t] <- x[i,t-1] + mu + ind[i] + year[t]
    x[i,t]~dnorm(Dnew[i,t],tau_add)
  }

  ## individual effects
  ind[i] ~ dnorm(0,tau_ind)

  ## initial condition
  x[i,1] ~ dnorm(x_ic,tau_ic)

}  ## end loop over individuals

  ## year effects
  for(t in 1:nt){
    year[t] ~ dnorm(0,tau_yr)
  }


  #### Priors
  tau_dbh ~ dgamma(a_dbh,r_dbh)
  tau_inc ~ dgamma(a_inc,r_inc)
  tau_add ~ dgamma(a_add,r_add)
  tau_ind ~ dgamma(1,0.1)
  tau_yr  ~ dgamma(1,0.1)
  mu ~ dnorm(0.5,0.5)

  }"

  ## state variable initial condition (subtract observed diameter increments off from the observed diam
  z0 = data$z
  for(i in 1:data$ni){
    z0[i,] = data$z[i,ncol(data$z)] - rev(cumsum(rev(data$y[i,])))
  }

  ## JAGS initial conditions
  nchain = 3
  init <- list()
  for(i in 1:nchain){
    y.samp = sample(data$y,length(data$y),replace=TRUE)
    init[[i]] <- list(x = z0,
                      tau_add=runif(1,1,5)/var(diff(y.samp),na.rm=TRUE), ## process precision
                      tau_dbh=1,   ## DBH obs precision
                      tau_inc=500, ## tree ring obs precision
                      tau_ind=50,  ## individual effect
                      tau_yr=100)  ## year effect
```

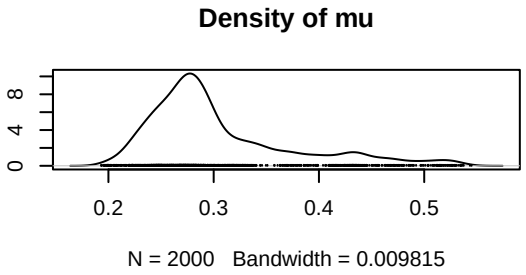
```
  init[[i]]$year = rnorm(data$nt,0,1/sqrt(init[[i]]$tau_yr)) ## use tau_yr to draw random year effect
  init[[i]]$ind  = rnorm(data$ni,0,1/sqrt(init[[i]]$tau_ind)) ## use tau_ind to draw random individua
}

## compile JAGS model
j.model   <- jags.model (file = textConnection(TreeDataFusionMV),
                         data = data,
                         inits = init,
                         n.chains = 3)
```

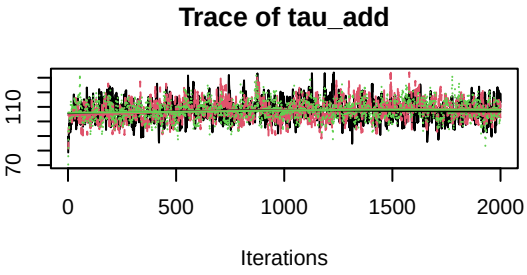
```
## Warning in jags.model(file = textConnection(TreeDataFusionMV), data = data, :
## Unused variable "time" in data
```

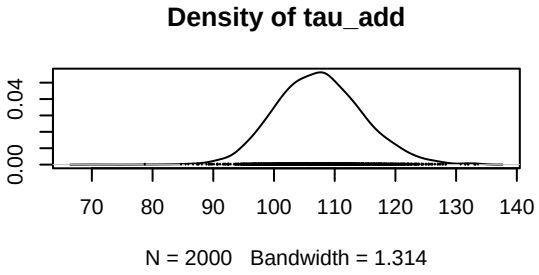
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1274
##    Unobserved stochastic nodes: 2347
##    Total graph size: 5955
##
## Initializing model
```

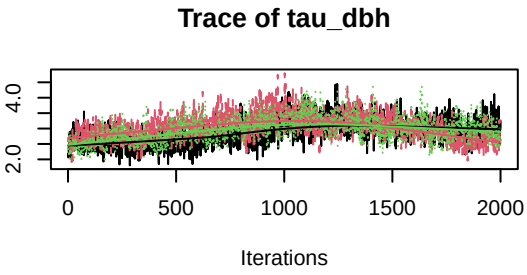
```
## note: this will give a WARNING (not an error) about unused tau_ind, tau_yr, ind, yr
## because we haven't included those to the JAGS code yet


## burn-in
jags.out   <- coda.samples (model = j.model,
                            variable.names = c("tau_add","tau_dbh","tau_inc","mu","tau_ind","tau_yr"),
                            n.iter = min(n.iter,2000))
## note: will also give a WARNING about tau_ind, tau_yr
plot(jags.out)
```

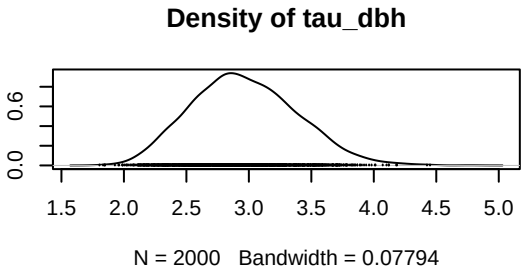

Trace of mu
0.20
0.45
0
500
1000
1500
2000
Iterations

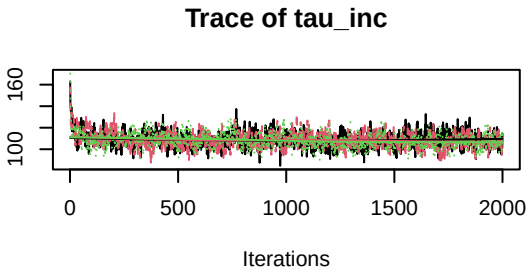

Density of mu
0
4
8
0.2
0.3
0.4
0.5
N = 2000 Bandwidth = 0.009815

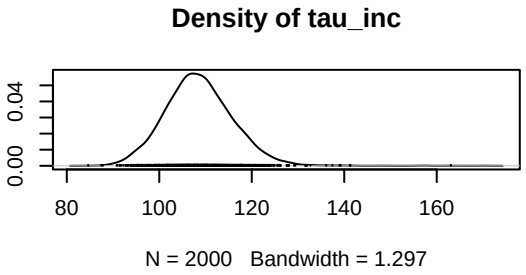

Trace of tau_add
70
110
0
500
1000
1500
2000
Iterations

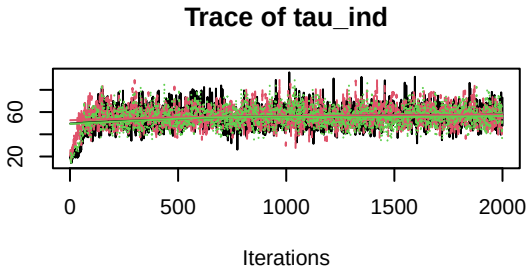

Density of tau_add
0.00
0.04
70
80
90
100
110
120
130
140
N = 2000 Bandwidth = 1.314

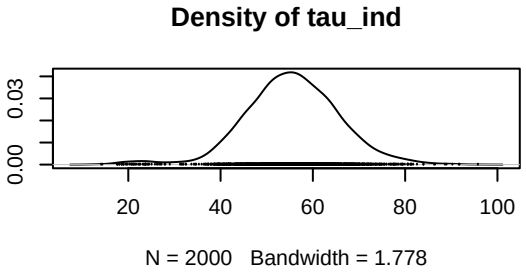

Trace of tau_dbh
2.0
4.0
0
500
1000
1500
2000
Iterations

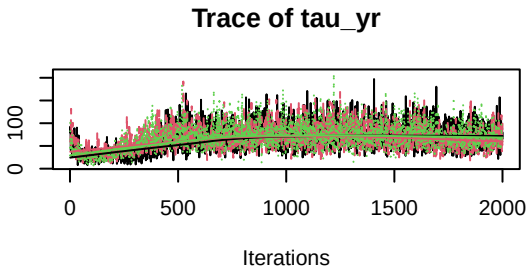

Density of tau_dbh
0.0
0.6
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
N = 2000 Bandwidth = 0.07794

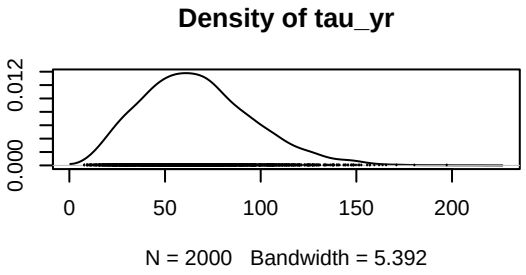








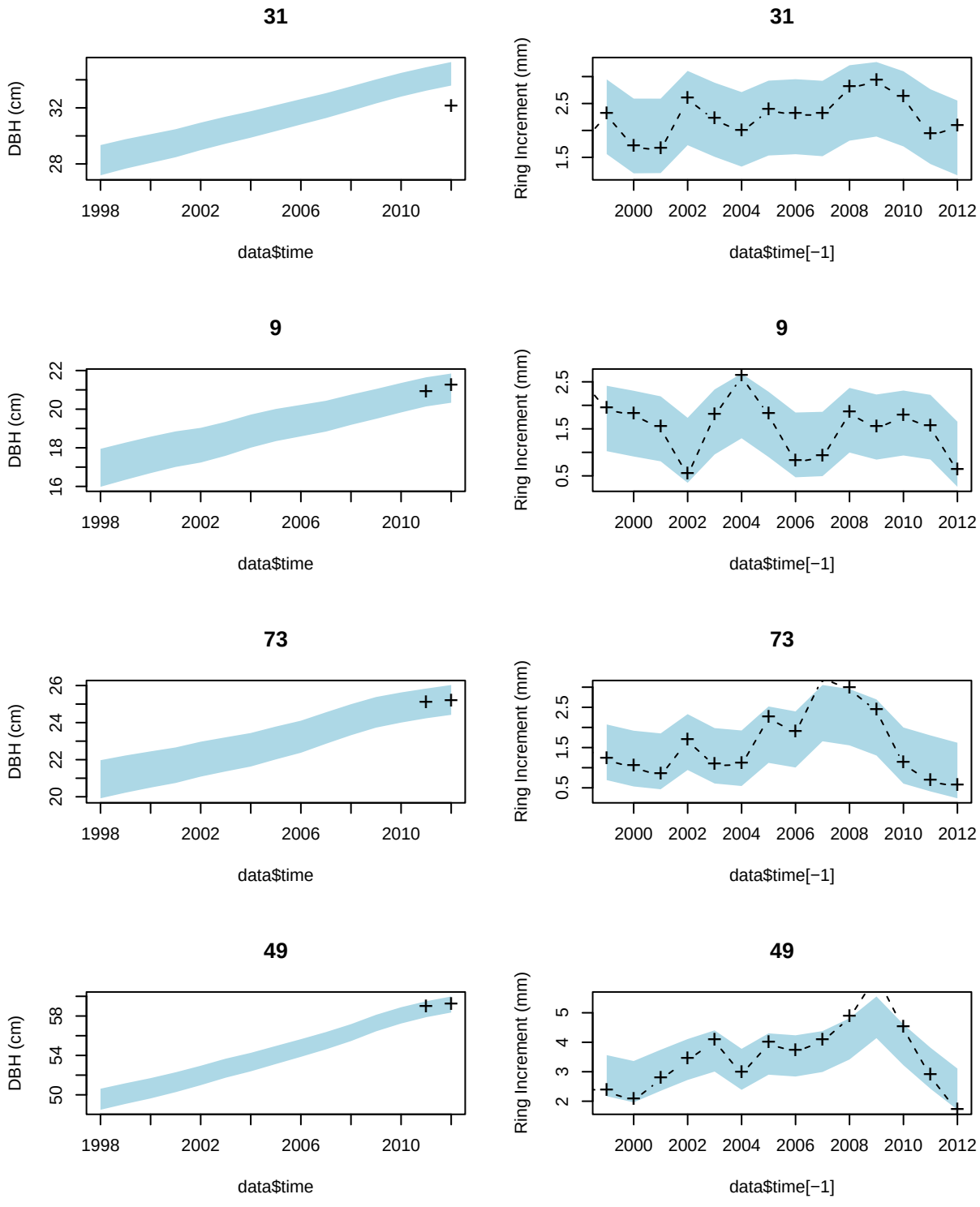





```
  ## run MCMC
  jags.out   <- coda.samples (model = j.model,
                              variable.names = c("x","tau_add","tau_dbh","tau_inc","mu","tau_ind","tau_y
                              n.iter = n.iter)
```

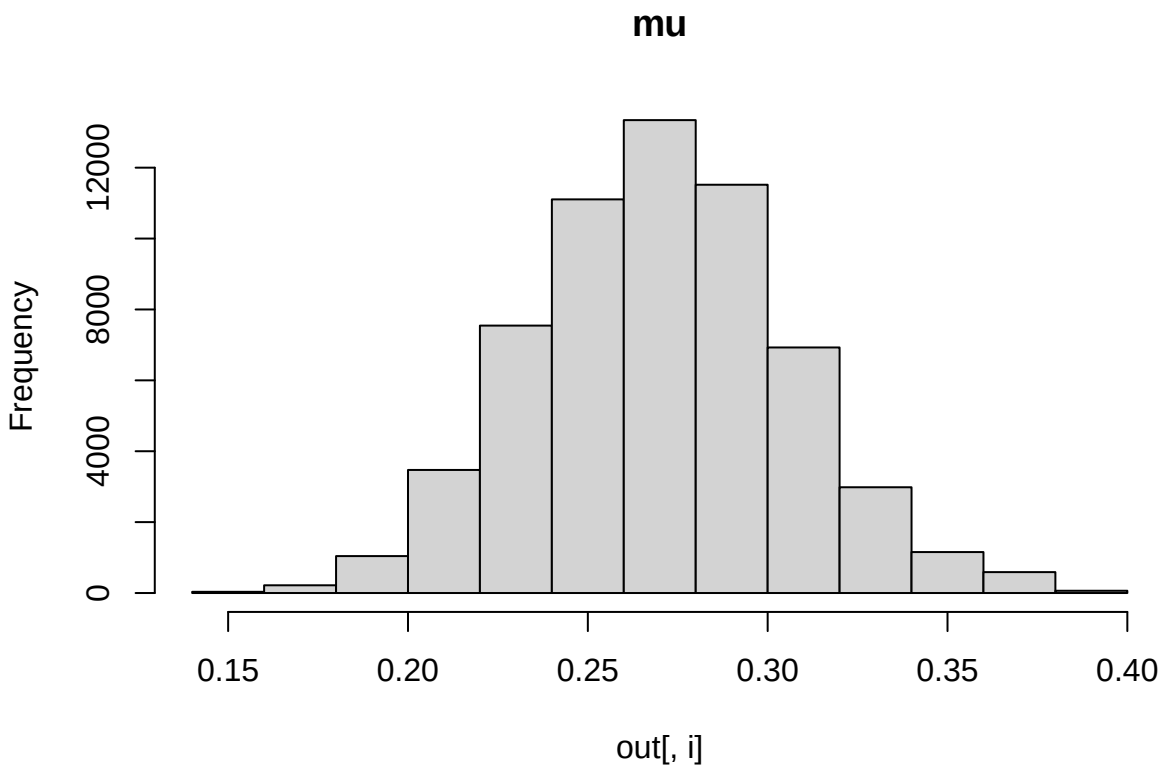
```
#### Helper function to parse JAGS variable names that include matrix syntax (e.g. "x[40,13]")
##' @param w mcmc object containing matrix outputs
##' @param pre prefix (variable name) for the matrix variable to be extracted
##' @param numeric boolean, whether to coerce class to numeric
parse.MatrixNames <- function(w, pre = "x", numeric = FALSE) {
  w <- sub(pre, "", w)
  w <- sub("[", "", w, fixed = TRUE)
  w <- sub("]", "", w, fixed = TRUE)
  w <- matrix(unlist(strsplit(w, ",")), nrow = length(w), byrow = TRUE)
  if (numeric) {
    class(w) <- "numeric"
  }
  colnames(w) <- c("row", "col")
  return(as.data.frame(w))
} # parse.MatrixNames

  #### Diagnostic plots

  ### DBH
```

```
layout(matrix(1:8,4,2)) ## arrange plots in a 4 x 2 matrix
out <- as.matrix(jags.out)
x.cols = which(substr(colnames(out),1,1)=="x")   ## which columns are the state variable, x
ci <- apply(out[,x.cols],2,quantile,c(0.025,0.5,0.975))
ci.names = parse.MatrixNames(colnames(ci),numeric=TRUE)

smp = c(sample.int(data$ni,3),49)  ## I've rigged the sampling to make sure you see tree 49!

## plot sample of DBH timeseries
for(i in smp){
  sel = which(ci.names$row == i)
  plot(data$time,ci[2,sel],type='n',ylim=range(ci[,sel],na.rm=TRUE),ylab="DBH (cm)",main=i)
  ciEnvelope(data$time,ci[1,sel],ci[3,sel],col="lightBlue")
  points(data$time,data$z[i,],pch="+",cex=1.5)
}

## sample of growth timeseries
for(i in smp){
  sel = which(ci.names$row == i)
  inc.mcmc = apply(out[,x.cols[sel]],1,diff)
  inc.ci = apply(inc.mcmc,1,quantile,c(0.025,0.5,0.975))*5

  plot(data$time[-1],inc.ci[2,],type='n',ylim=range(inc.ci,na.rm=TRUE),ylab="Ring Increment (mm)",mai
  ciEnvelope(data$time[-1],inc.ci[1,],inc.ci[3,],col="lightBlue")
  points(data$time,data$y[i,]*5,pch="+",cex=1.5,type='b',lty=2)
}
```



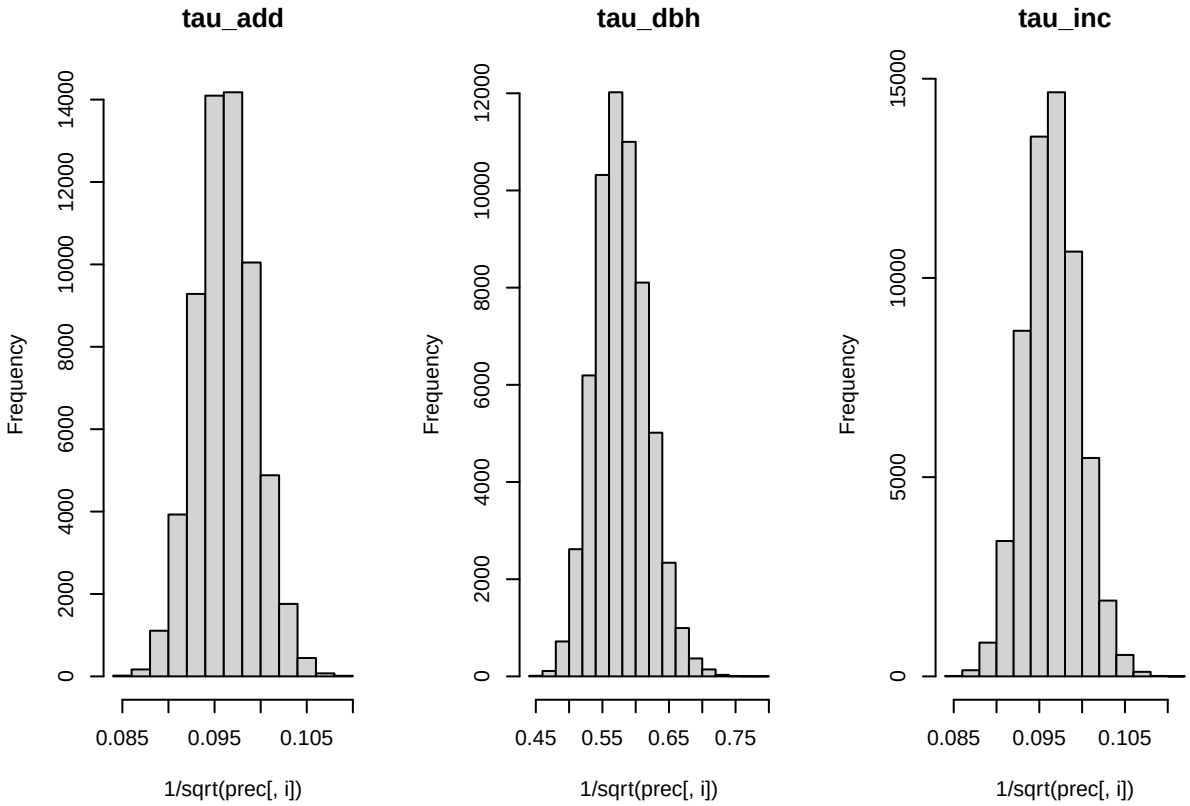

```
## process model
vars = (1:ncol(out))[-c(which(substr(colnames(out),1,1)=="x"),grep("tau",colnames(out)),
                        grep("year",colnames(out)),grep("ind",colnames(out)))] ## remove x, tau's, yr

par(mfrow=c(1,1))
for(i in vars){
  hist(out[,i],main=colnames(out)[i])
```

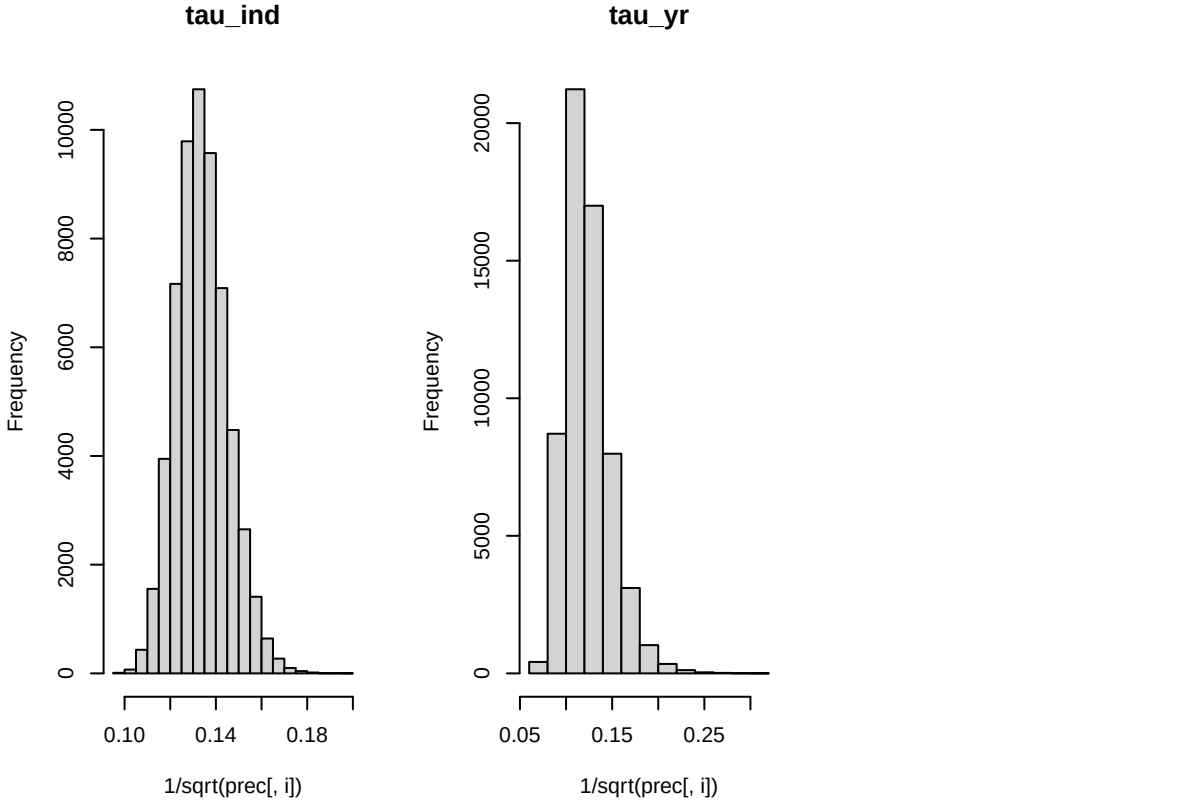
```
}
```



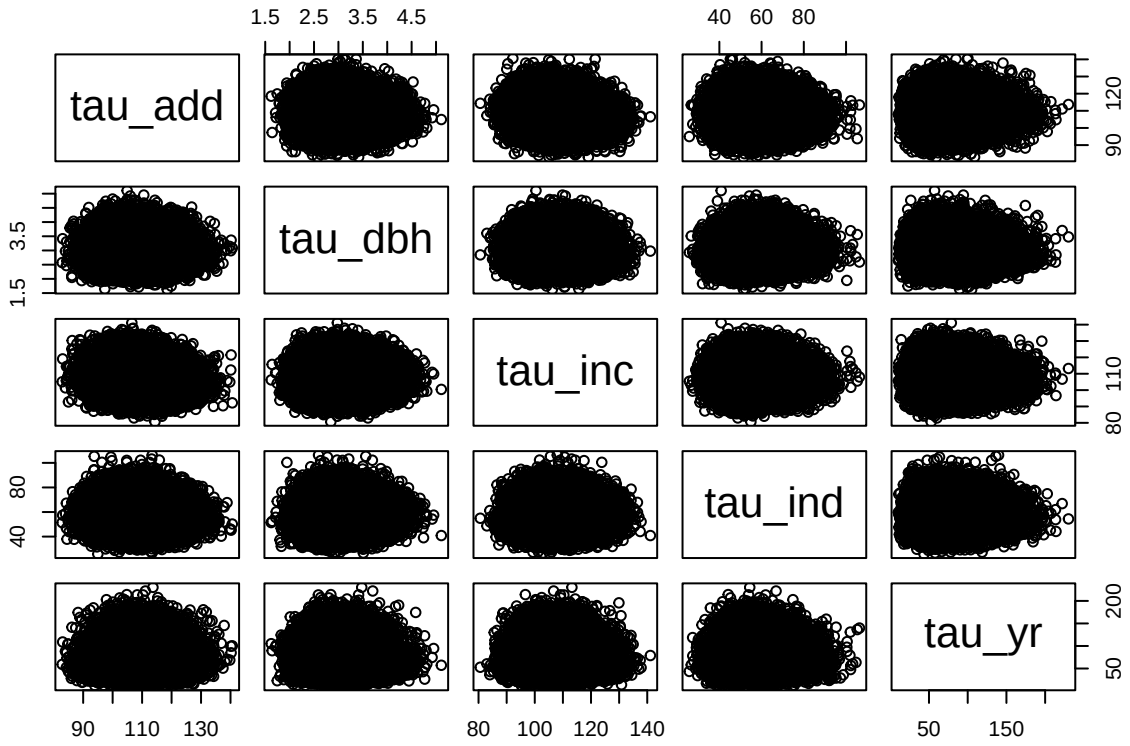

```
if(length(vars)>1) pairs(out[,vars])
```

```
  ## Standard Deviations
par(mfrow=c(1,3))
prec = out[,grep("tau",colnames(out))] ## select just taus to plot
  for(i in 1:ncol(prec)){
  hist(1/sqrt(prec[,i]),main=colnames(prec)[i])
}
```


tau_add
Frequency
0
2000
4000
6000
8000
10000
12000
14000
0.085
0.095
0.105
1/sqrt(prec[, i])
tau_dbh
Frequency
0
2000
4000
6000
8000
10000
12000
0.45
0.55
0.65
0.75
1/sqrt(prec[, i])
tau_inc
Frequency
0
5000
10000
15000
0.085
0.095
0.105
1/sqrt(prec[, i])


tau_ind
Frequency
0
2000
4000
6000
8000
10000
0.10
0.14
0.18
1/sqrt(prec[, i])
tau_yr
Frequency
0
5000
10000
15000
20000
0.05
0.15
0.25
1/sqrt(prec[, i])

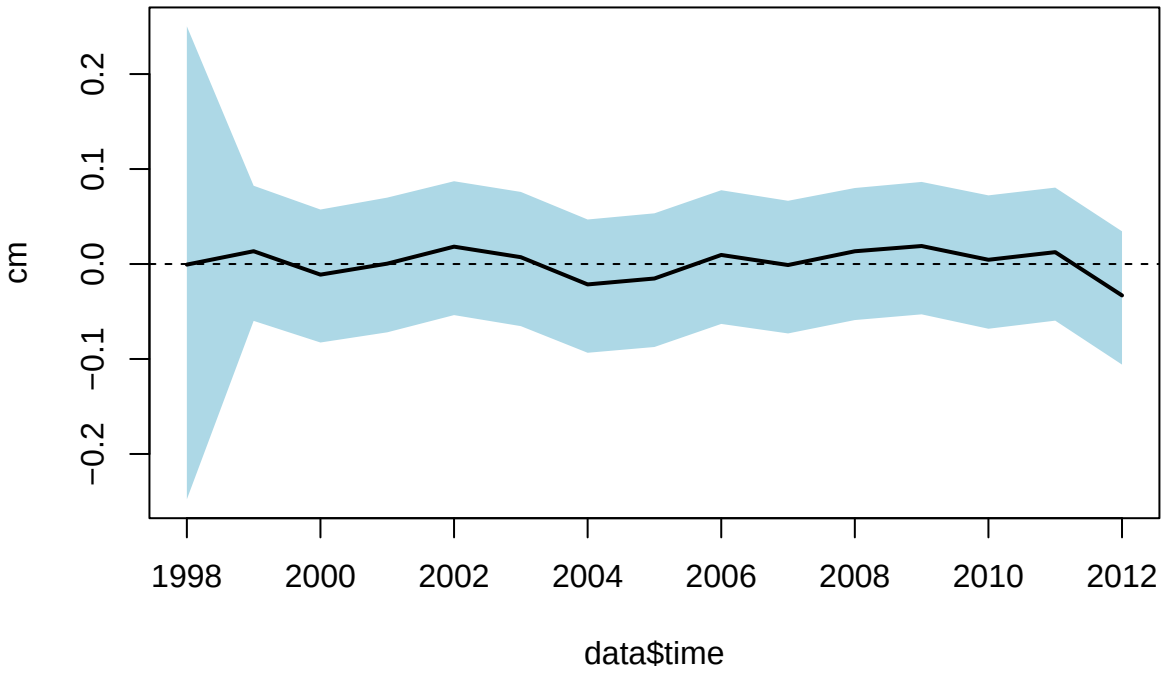
```
pairs(prec)
```



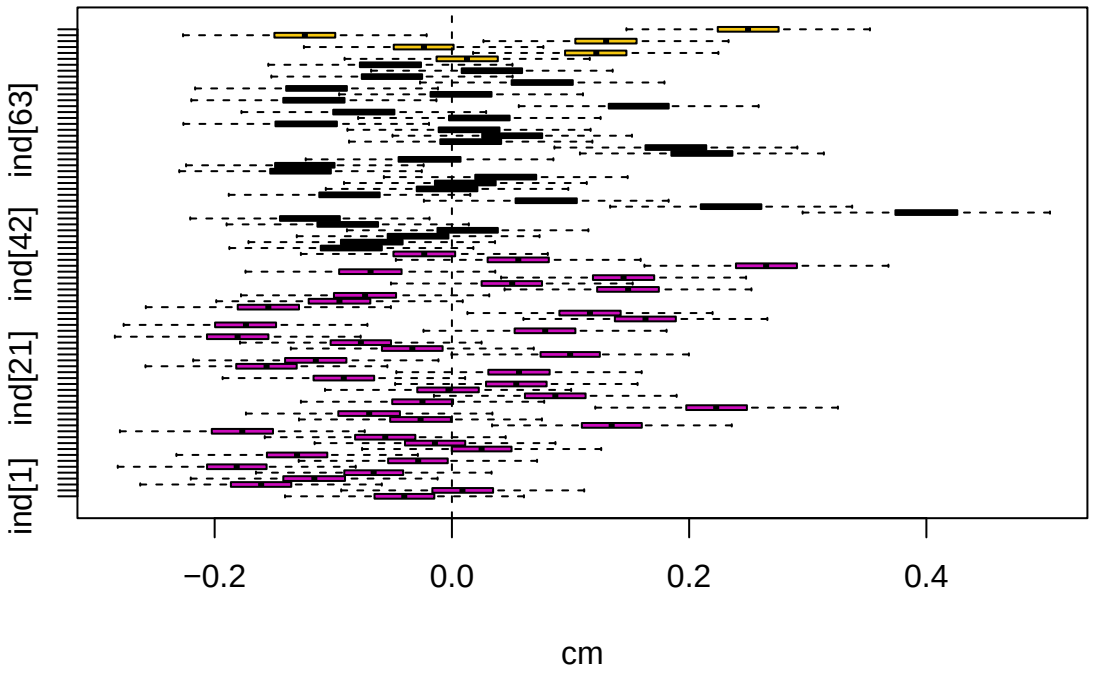

```
cor(prec)
```

```
##              tau_add      tau_dbh      tau_inc      tau_ind       tau_yr
## tau_add  1.000000000 -0.009046066 -0.100655215 -0.006400361  0.002986580
## tau_dbh -0.009046066  1.000000000  0.009680834  0.007260499 -0.011855822
## tau_inc -0.100655215  0.009680834  1.000000000 -0.020391072  0.004173980
## tau_ind -0.006400361  0.007260499 -0.020391072  1.000000000 -0.002502845
## tau_yr   0.002986580 -0.011855822  0.004173980 -0.002502845  1.000000000
```

```
load("data/combined.07.RData")
par(mfrow=c(1,1))
### YEAR
year.cols = grep("year",colnames(out))
if(length(year.cols>0)){
  ci.yr <- apply(out[,year.cols],2,quantile,c(0.025,0.5,0.975))
  plot(data$time,ci.yr[2,],type='n',ylim=range(ci.yr,na.rm=TRUE),main="Year Effect",ylab="cm")
  ciEnvelope(data$time,ci.yr[1,],ci.yr[3,],col="lightBlue")
  lines(data$time,ci.yr[2,],lty=1,lwd=2)
  abline(h=0,lty=2)
}
```

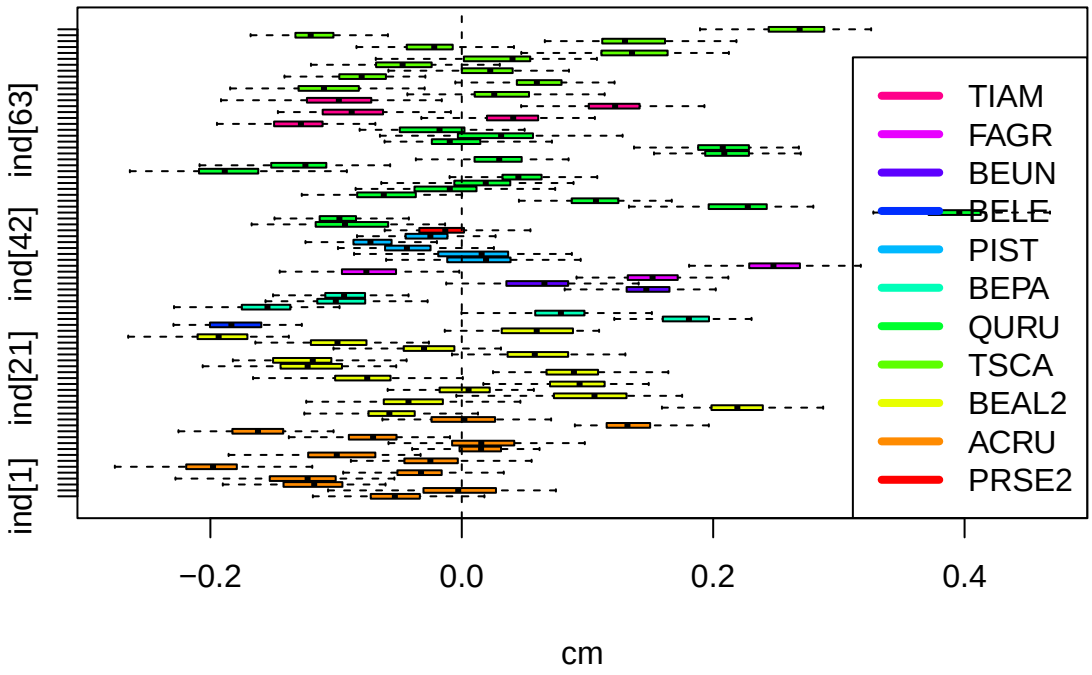



```
### INDIV
ind.cols= which(substr(colnames(out),1,3)=="ind")

if(length(ind.cols)>0){
  boxplot(out[,ind.cols],horizontal=TRUE,outline=FALSE,col=combined$PLOT,main="Individual Effects By
  abline(v=0,lty=2)
  ## calculate plot-level means for random effects
  tapply(apply(out[,ind.cols],2,mean),combined$PLOT,mean)
  table(combined$PLOT)

}
```




```
##
##  1  6  7
## 32 42  6
```

```
  if(length(ind.cols)>0){
    unique_species <- unique(combined$SPP)
    colors_for_species <- rainbow(length(unique_species))
    species_colors <- setNames(colors_for_species, unique_species)
    boxplot(out[order(combined$SPP), ind.cols], horizontal=TRUE, outline=FALSE,
            col = species_colors[combined$SPP[order(combined$SPP)]],
            main="Individual Effects By Species", xlab="cm")
    abline(v=0, lty=2)
    legend("bottomright", legend=rev(unique_species),
          col=rev(species_colors[unique_species]), lwd=4)
    tapply(apply(out[,ind.cols], 2, mean), combined$SPP, mean)
  }
```




```
##        ACRU       BEAL2        BELE        BEPA        BEUN        FAGR
## -0.10219209 -0.02180893 -0.08489626  0.02046105  0.14308660  0.01756925
##        PIST       PRSE2        QURU        TIAM        TSCA
##  0.13491890 -0.04002854  0.02346502 -0.02608221  0.04375634
```

- What is the relative partitioning of uncertainties in the different versions of the model among observation error, process error, and the different random effects? [B]
  - We can partition the uncertainties in the different versions of the model among observation error, process error, and the different random effects as follows:
    - 1. Observation Error:
      - DBH Observation Error (tau_dbh): Precision of diameter at breast height measurements.
      - Growth Observation Error (tau_inc): Precision of growth measurements.
    - 2. Process Error:
      - Process Model Error (tau_add): Precision of the process model that predicts DBH based on previous measurements and the model's process parameters.
    - 3. Random Effects:
      - Individual Effects (tau_ind): Precision of the random effects at the individual tree level.
      - Year Effects (tau_yr): Precision of the random effects at the yearly level.

```
summary_stats <- summary(jags.out)
vars_of_interest <- c("tau_dbh", "tau_inc", "tau_add", "tau_ind", "tau_yr")
for (var in vars_of_interest) {
  cat("Summary for", var, ":\n")
  print(summary_stats$statistics[rownames(summary_stats$statistics) == var, ])
  cat("\n")
}
```

```
## Summary for tau_dbh :
##           Mean             SD       Naive SE Time-series SE
##    3.027200785    0.407677299    0.001664336    0.019000918
##
## Summary for tau_inc :
##           Mean             SD       Naive SE Time-series SE
##   107.74636919     7.05015153     0.02878212     0.10167025
##
## Summary for tau_add :
##           Mean             SD       Naive SE Time-series SE
##   108.28933057     7.13583024     0.02913190     0.07445194
##
## Summary for tau_ind :
##           Mean             SD       Naive SE Time-series SE
##    56.94042313     9.53164208     0.03891277     0.08465314
##
## Summary for tau_yr :
##           Mean             SD       Naive SE Time-series SE
##     73.0473917     26.6723698      0.1088895      0.2416499
```

First Model (Without Random Effects):

Parameter	Mean	SD	Naive SE	Time-series SE
tau_dbh	3.02305503	0.42608480	0.00173948	0.01956820
tau_inc	59.8191216	4.55289110	0.01858710	0.14432680
tau_add	59.7686148	4.59445273	0.01875677	0.14401821

Second Model (With Random Effects):

Parameter	Mean	SD	Naive SE	Time-series SE
tau_dbh	3.01474083	0.40583032	0.00165680	0.01736212
tau_inc	107.7731509	7.08636891	0.02892998	0.10344560
tau_add	108.0529850	7.07601330	0.02888770	0.07075280
tau_ind	56.9715231	9.59167499	0.03915785	0.08801219
tau_yr	72.4433886	26.64060990	0.10875980	0.29069100

- What does the size of these effects suggest about the drivers of uncertainty in tree growth? [B]
  - In this case we observe that the DBH Measurment error dont prsent contributes minimally to overall uncertainty and is relatively stable regardless of the model complexity.
  - In the case of the growth measurement error, we see that the uncertainty increases when we remove the random effects, suggesting that the random effects are capturing some of the uncertainty in the growth measurements.
  - For the process error, we observe a similar behavior to the growth measurement error, where the uncertainty increases when we remove the random effects. This means that the process model error becomes a larger source of uncertainty without the random effects.

1. Based on the diagnostics, propose an additional effect (fixed or random) to add to the model. Such an effect should plausibly chip away at a sizable fraction of the unexplained variability – you wouldn't want to propose an effect that isn't associated with systematic variability. [A]

- Incorporating additional effects such as climate variables, soil characteristics, and light availability into the tree growth model could significantly reduce unexplained variability. Climate effects, as fixed effects, could account for the influence of annual average temperature and total precipitation on growth rates. Soil characteristics, potentially varied as fixed or random effects depending on the availabilty of the data. Lastly, light availability, introduced as a fixed effect, could account for the influence of canopy cover on tree growth rates.

2. Explain any additional exploratory analyses you would perform (e.g. plotting your proposed covariate against one of the random effects). [B]

- Similar to this lab, I would try to run the model with and without the new covariate to see how much of the unexplained variability it explains, and do all the MCMC analysis. I would also plot the new covariate against the random effects to see if there is a relationship between the two. Also if the covariate is random we could make an statistical analysis to see what is the random effect producing with the other error sources.

3. Write the JAGS code that would fit the proposed model (note: you don't have to track down additional covariate data or run this model, just propose the code) [A]

My model proposition will look like this:

```
model {
  ### Loop over all individuals
  for(i in 1:ni){

    #### Data Model: DBH
    for(t in 1:nt){
      z[i,t] ~ dnorm(x[i,t], tau_dbh)
    }

    #### Data Model: growth
    for(t in 2:nt){
      inc[i,t] <- x[i,t] - x[i,t-1]
      y[i,t] ~ dnorm(inc[i,t], tau_inc)
    }

    #### Process Model with additional effects
    for(t in 2:nt){
      Dnew[i,t] <- x[i,t-1] + mu + ind[i] + year[t] + temp[t] + prec[t] + soil[i] + light[i]
      x[i,t] ~ dnorm(Dnew[i,t], tau_add)
    }

    ## initial condition
    x[i,1] ~ dnorm(x_ic, tau_ic)

    ## individual effects
    ind[i] ~ dnorm(0, tau_ind)
  }

  ## year effects
  for(t in 1:nt){
    year[t] ~ dnorm(0, tau_yr)
  }
```

```
  #### Priors
  tau_dbh ~ dgamma(a_dbh, r_dbh)
  tau_inc ~ dgamma(a_inc, r_inc)
  tau_add ~ dgamma(a_add, r_add)
  tau_ind ~ dgamma(1, 0.1)
  tau_yr  ~ dgamma(1, 0.1)
  mu ~ dnorm(0.5, 0.5)

  temp ~ dnorm(0, 0.01)
  prec ~ dnorm(0, 0.01)
  soil ~ dnorm(0, 0.01)
  light ~ dnorm(0, 0.01)

}
```

BECAUSE THE PRODUCTION VERSION OF THIS CODE TAKES A LONG TIME TO RUN, PLEASE SUBMIT THE KNIT HTML NOT THE Rmd

Applying this to your own forecast

- If you already have a state-space model running for your own model (Exercise 06) you can add additional data constraints by adding additional likelihoods to the JAGS code. For simple cases this is relatively straightforward, but you'll want to think very carefully about many of the issues raised in Chapters 6 and 9 when constructing your likelihood (model and data biases, autocorrelation, spatial and temporal aggregation, proxy data, non-Gaussian error, etc.)
- If you are forecasting multiple variables (e.g. different sites, individuals, species, etc.), the code above provides a useful template for how to organize loops over multiple timeseries.
- The code above also provides a useful template for adding different random effects
- While the `fit_dlm` function discussed in Exercise 06 doesn't yet support data fusion models or multivariate models, the `ParseFixed` function in the ecoforecastR package can be used to help expand a basic random walk model to a more complex model.
- When developing your own more sophisticated analyses, make sure to start simple and add complexity incrementally. By only adding one new thing at a time it becomes much easier to debug when something goes wrong and to stop when you push the model beyond the constraints the data are able to provide.