Pondo_Allom_Params

Katie Murenbeeld

3/26/2020

Introduction:

Information about allometric curves and functions used in FATES. https://fates-docs.readthedocs.io/en/latest/fates_tech_note.html#allometry-and-growth-along-allometric-curves

Install the BAAD Package:

Install the package from github:

```
#install.packages("devtools")
#devtools::install_github("richfitz/datastorr")
#devtools::install_github("traitecoevo/baad.data")
```

Review the data:

```
baad <- baad.data::baad_data()
d_baad <- baad$data

# Check the data
#head(d_baad)
```

Refine (subset) the data for Pinus Ponderosa:

```
pipo <- d_baad[ which(d_baad$species == 'Pinus ponderosa'), ]

# Check the data
#head(pipo)
```

Diameter at breast height (dbh) to height - d2h

FATES allows for four different approaches to predicting height from dbh:

- A power function
$$h = p_1 * d^(p_2)$$
- O'Brien et al. (1995)
$$log_{10}Height = log_{10}DBH * slope + intercept$$
- Poorter et al. (2006)
$$h = p_1 * (1 - exp(p_2 * d^(p_3)))$$
- Martinex Cano et al. (2019)
$$h = (p_1 * d^(p_2)/(p_3 + d^(p_2)))$$

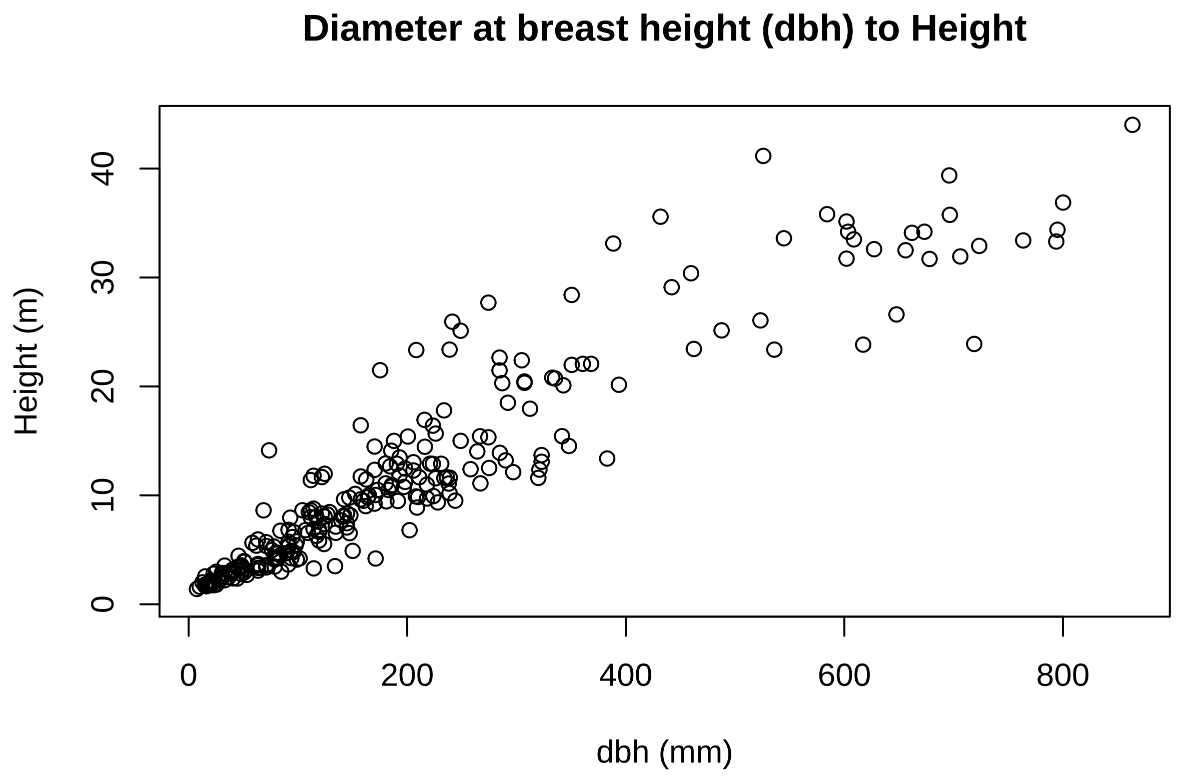
For my research I use the dbh to height relationship from O'Brien et al., 1995. The log10Height (m) is regressed on log10DBH (mm). Where slope is p1 or fates_allom_d2h1 and intercept is p2 or fates_allom_d2h2 in the parameter file.

$$log_{10}Height = log_{10}DBH * slope + intercept$$

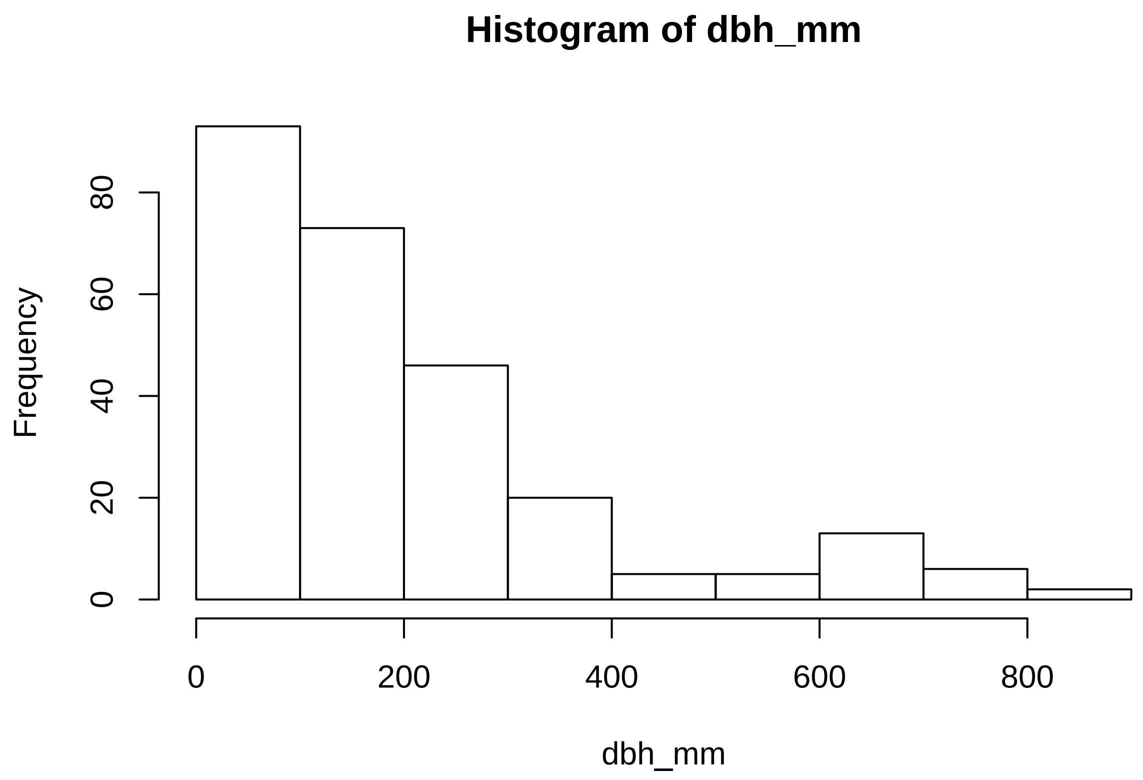
So modeled height would be. I want to solve for the best fit p1 and p2 given height and dbh data from BAAD.

$$Height = 10^{(log10(min(d,dbhmax))*p1+p2)}$$

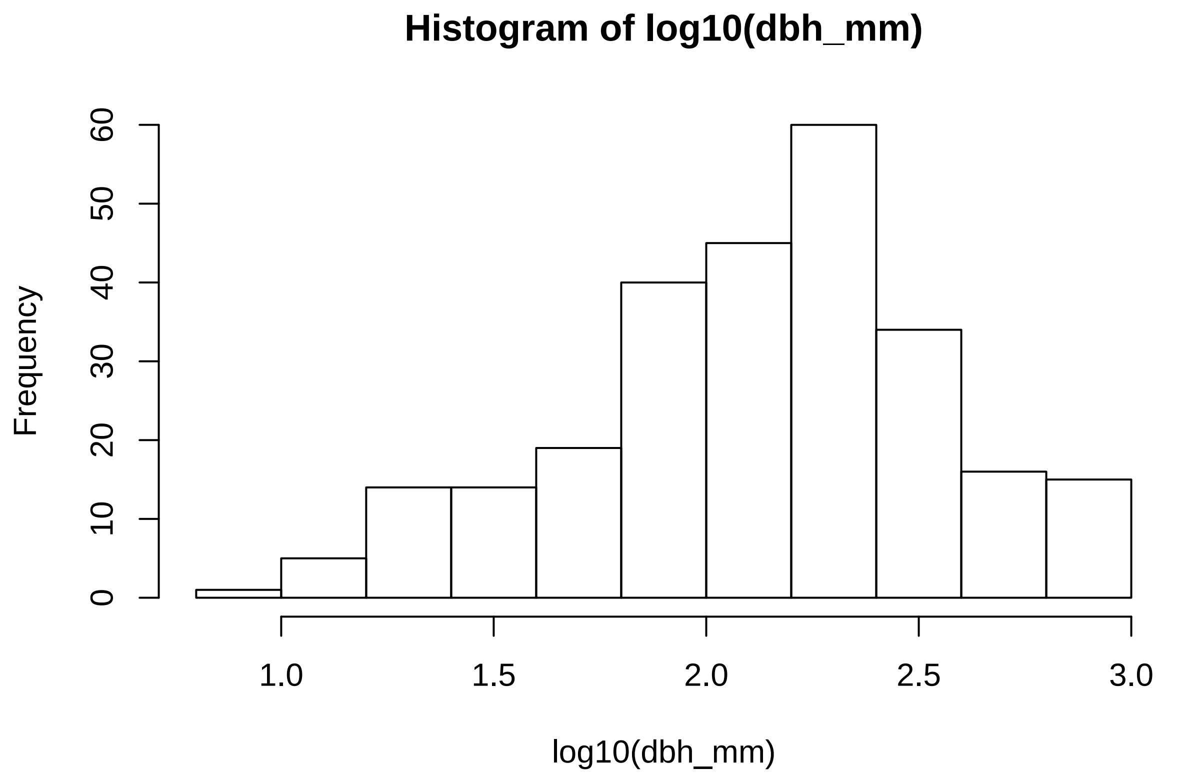
```r
# Set variables from the data.

# Diameter at breat height (dbh) will be used for most allometric calculations
dbh_mm <- (pipo$d.bh)*1000 # O'Brien uses dbh in mm
dbh_cm <- (pipo$d.bh)*100 # FATES needs dbh in cm
# Height
h <- (pipo$h.t)

plot(h~dbh_mm,
     main="Diameter at breast height (dbh) to Height",
     xlab = "dbh (mm)",
     ylab = "Height (m)")
```



```
hist(dbh_mm)
```



```
hist(log10(dbh_mm))
```



Initially I wanted to test out a simple linear function and then a log-log regression (basically O'Brien).

```
# But first I will test out just a linear model
d2hmod_mm <- lm(h~dbh_mm)
coef(d2hmod_mm)
```

```
## (Intercept)      dbh_mm
##  1.88991834  0.04858541
```

```
# Then a linear model with log10(h) regressed on log10(dbh)
d2hmodlog_mm <- lm(log10(h)~log10(dbh_mm))
coef(d2hmodlog_mm)
```

```
##   (Intercept) log10(dbh_mm)
##    -0.8150832     0.8181127
```

```
d2hmodlog_cm <- lm(log10(h)~log10(dbh_cm))
coef(d2hmodlog_cm)
```

```
##   (Intercept) log10(dbh_cm)
##   0.003029536   0.818112721
```

Next, I will create a function to represent the O'Brien calculation used for dbh to height relationships.

```
obrien <- function(dbh, p1, p2){
  height <- 10^((log10(dbh))* p1 + p2)
  return(height)
}

# Remember in O'Brien dbh is in mm.
dbins_mm <- c(100,200,300,400,500,600,700,800,900)
# But FATES uses cm.
dbins_cm <- c(10,20,30,40,50,60,70,80,90)
```

```
# Use default parameters from O'Brien et al 1995
default_mm <- obrien(dbins_mm, 0.7, -0.2)

# Use the parameters from the log10 linear model d2hmodlog
lmfit_mm <- obrien(dbins_mm, 0.82, -0.82) # dbh in mm
lmfit_cm <- obrien(dbins_cm, 0.82, 0.003) # dbh in cm
lmfit_cm2 <- obrien(dbh_cm, 0.82, 0.003)

coef(lm(default_mm~dbins_mm))
```

```
## (Intercept)    dbins_mm
## 11.67774249  0.07124032
```

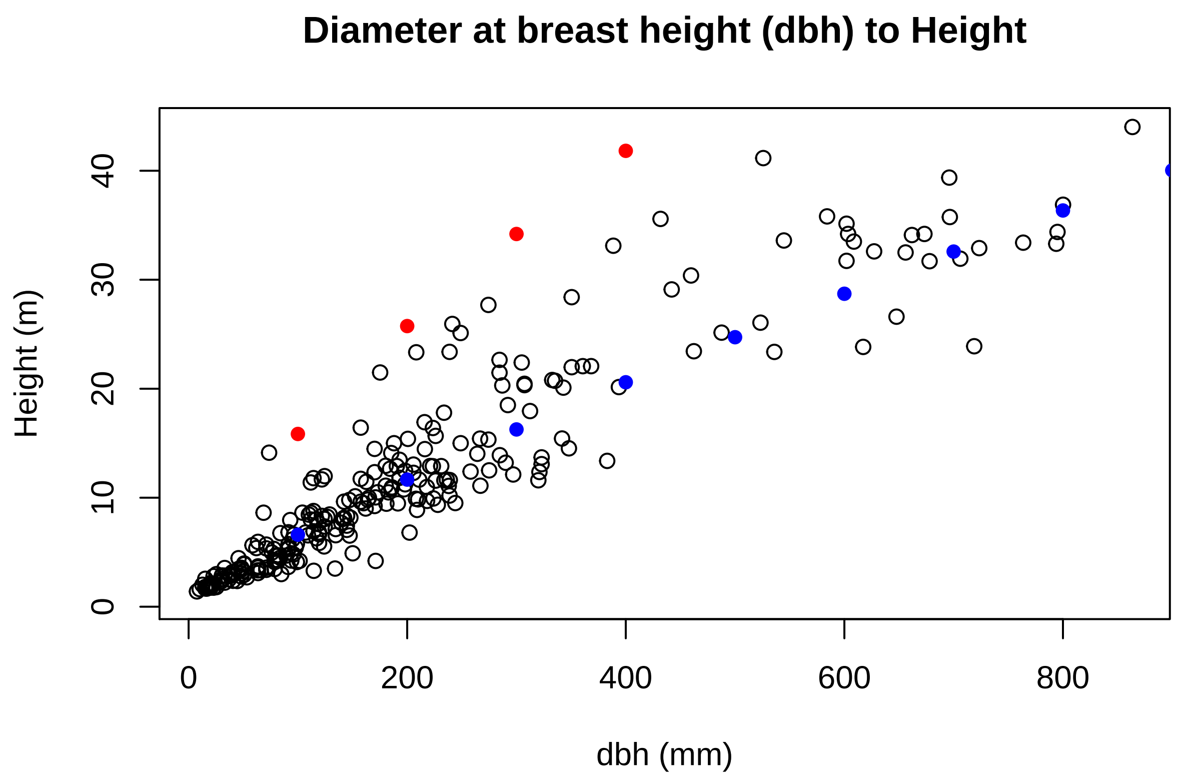
```
coef(lm(lmfit_mm~dbins_mm))
```

```
## (Intercept)    dbins_mm
##  3.45859976  0.04142493
```

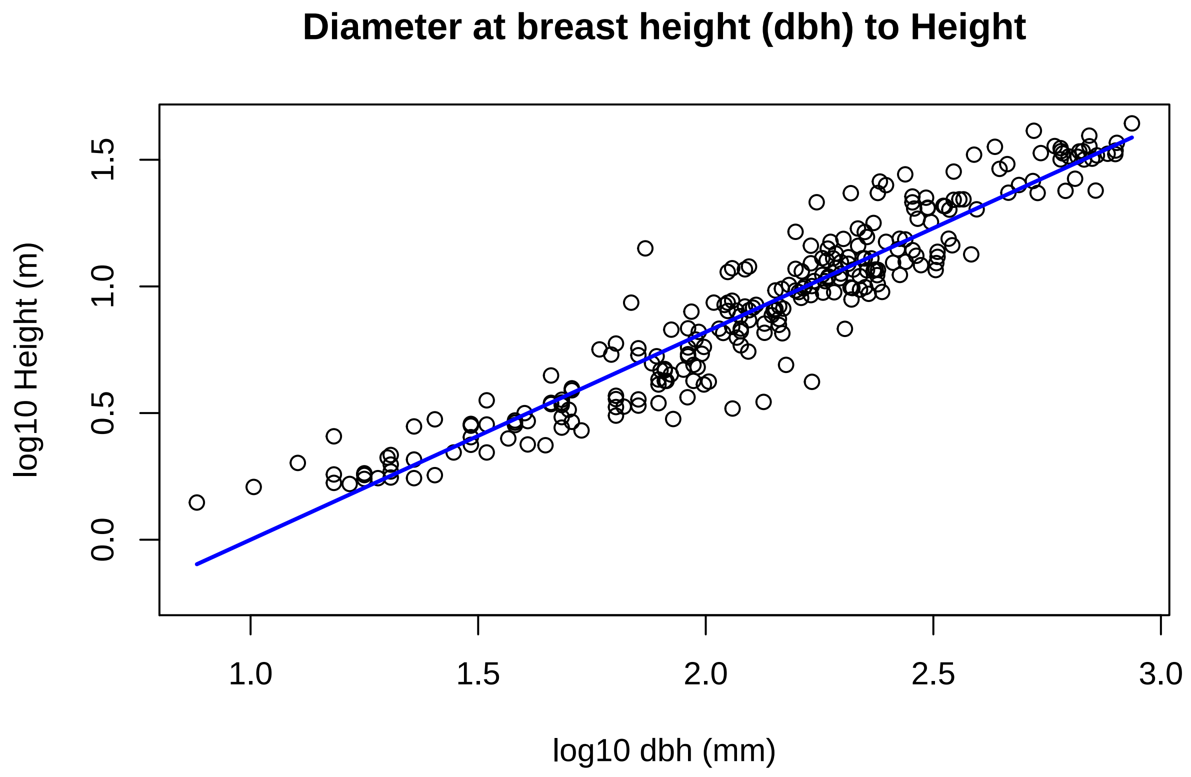
```
coef(lm(lmfit_cm~dbins_cm))
```

```
## (Intercept)    dbins_cm
##   3.4825736   0.4171207
```

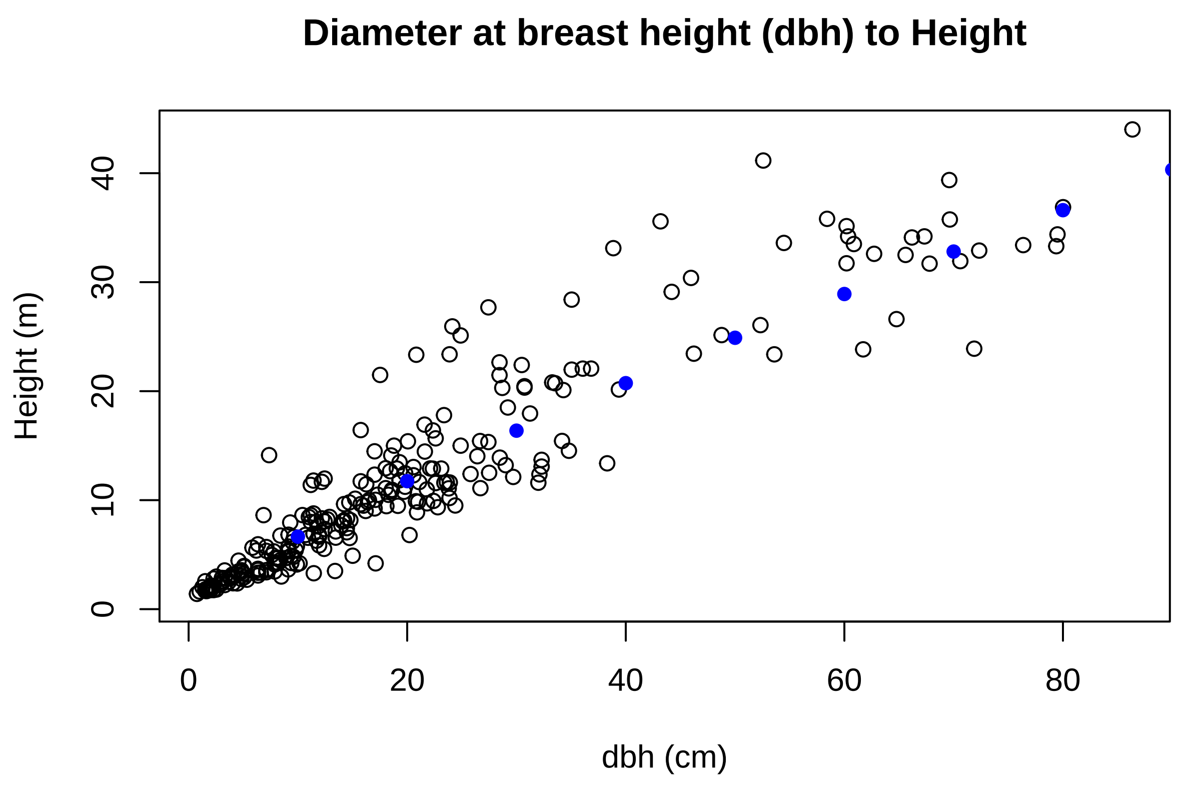
```
plot(h~dbh_mm,
     main="Diameter at breast height (dbh) to Height",
     xlab = "dbh (mm)",
     ylab = "Height (m)")
# default O'Brien slope (0.7) and intercept (-0.2)
points(default_mm~dbins_mm, col="red", pch=16)
# BAAD data slope (0.82) and intercept (-0.82) dbh in mm
points(lmfit_mm~dbins_mm, col="blue", pch=16)
```



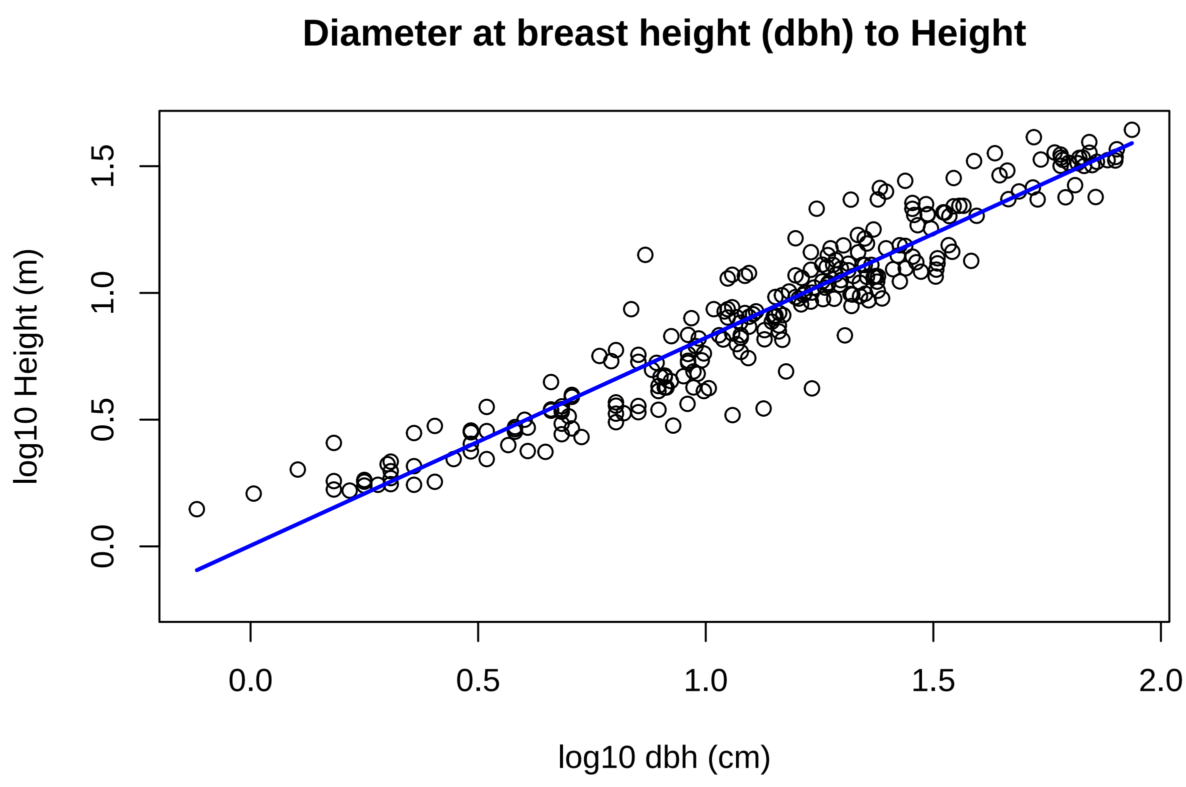
```
plot(log10(h)~log10(dbh_mm),
     main="Diameter at breast height (dbh) to Height",
     xlab = "log10 dbh (mm)",
     ylab = "log10 Height (m)")
curve(-0.82 + (0.82*x), add=T, col="blue", lwd=2) # log regression coefficients from d2hmodlog
```



```
plot(h~dbh_cm,
     main="Diameter at breast height (dbh) to Height",
     xlab = "dbh (cm)",
     ylab = "Height (m)")

points(default_mm~dbins_mm, col="red", pch=16) # default O'Brien slope (0.7) and intercept (-0.2)
points(lmfit_cm~dbins_cm, col="blue", pch=16) # BAAD data slope (0.82) and intercept (-0.82)
```



```
plot(log10(h)~log10(dbh_cm),
     main="Diameter at breast height (dbh) to Height",
     xlab = "log10 dbh (cm)",
     ylab = "log10 Height (m)")
curve(0.003 + (0.82*x), add=T, col="blue", lwd=2) # log regression coefficients from d2hmodlog
```



One

can also test the model fit using the xxx() function in R. Additionally, I want to calculate an r2 to help determine the best fit parameters.

```r
# Predict the fit...these stupid logs get me all turned around



# Set up a function to calculate r^2
r2 <- function(y_hat,y){
  RSS<-sum((((y_hat))-(y))^2)
  TSS<-sum(((y)-(mean(y)))^2)
  return(1-RSS/TSS)}

d2bh_defaultR2 <- r2(default_mm, h)
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

Diameter at breast height (dbh) to above ground biomass (AGB) - d2bagw

FATES provides three different options for calculating AGB from dbh:

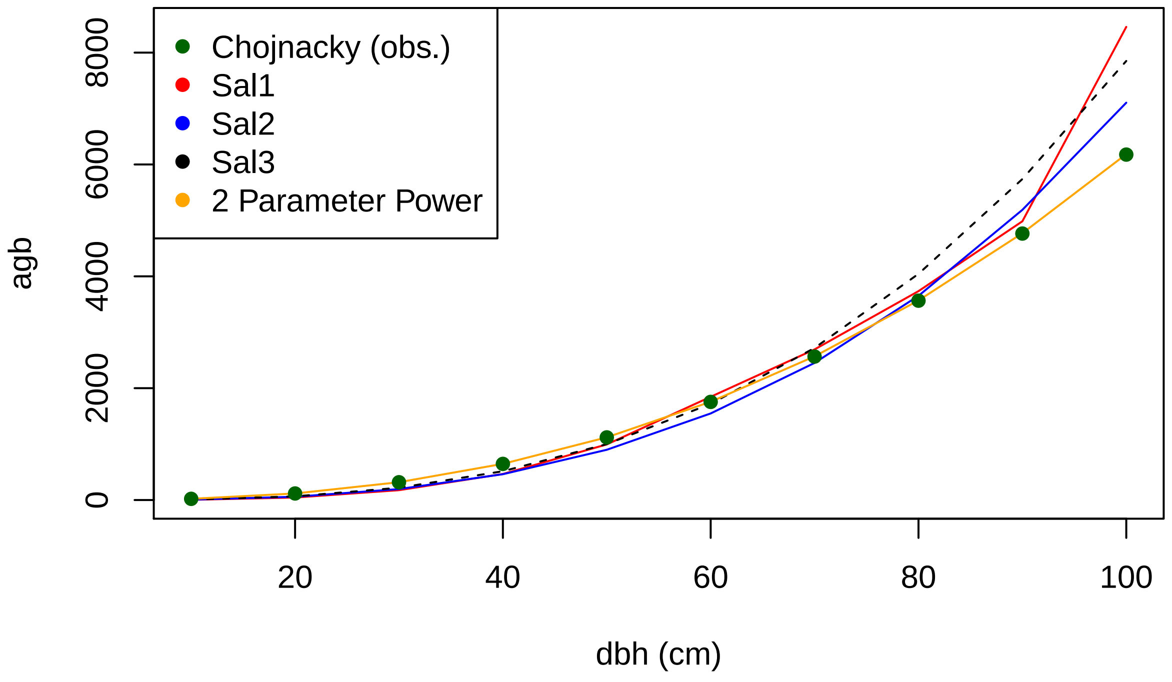
- Saldarriaga et al. (1998)
$$(C_{agb} = f_{agb} * p_1 * h^{p_2} * d^{p_3} * rho^{p_4})$$
- 2 parameter power function
$$(C_{agb} = p_1/c2b * d^{p_2})$$
- Chave et al. (2014)
$$(C_{agb} = p_1/c2b * (rho * d^2 * h)^{p_2})$$

For this project I will test Saldarriaga et al. (1998) as well as the 2 parameter power function.

$$C_{agb} = f_{agb} * p_1 * h^{p_2} * d^{p_3} * rho^{p_4}$$

```r
# In this case dbh and agb calculated from Chojnacky serves as
# the "observed data" to fit the other parameters against.

# Define the variables needed for the Saldarriaga function
rho = 0.367
f_agb = 0.6
d.agb <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
h.agb <- c(2.82, 7.7, 13.9, 21.0, 29.1, 37.9, 37.9, 37.9, 37.9, 57.5)
#use heights from d2h
lmfit.h <- obrien(d.agb, 0.82, 0.003)

# Define the Saldarriaga function
sal <- function(f_agb, p1, h, p2, d, p3, rho, p4){
  agb <- f_agb * p1 * h^p2 * d^p3 * rho^p4
  return(agb)
}

# Define a 2 parameter power function (as a stand-in for Chojnakcy if those param values are used)
par2_pwr <- function(p1,d,p2,c2b){
  bagw <- (p1*(d^p2))/c2b
  return(bagw)
}

choj_eq <- function(b0,b1,d){
  biom <- b0  + (b1*log(d))
  return(biom)
}
choj_p1 = -2.6177
choj_p2 = 2.4638
c2b = 2

test_2par <- par2_pwr(0.146, d.agb, 2.464, c2b)
choj <- choj_eq(choj_p1, choj_p2, d.agb)

# Parameters from earlier Jupyter Notebook.
#p1 = 0.131
#p2 = 0.626
#p3 = 2.46
#p4 = 2.18
```

```r
# New parameters from bet fit below.

p1 = 0.11
p2 = 0.65
p3 = 2.45
p4 = 2.15

sal1 <- sal(f_agb, p1, h.agb, p2, d.agb, p3, rho, p4)
sal2 <- sal(f_agb, p1, lmfit.h, p2, d.agb, p3, rho, p4)
sal3 <- sal(f_agb, 0.131, lmfit.h, 0.626, d.agb, 2.46, rho, 2.18)
#plot(choj.h~d.agb, type="l", col="blue")
plot(sal1~d.agb, col="red", type="l", ylab="agb", xlab="dbh (cm)")
points(sal2~d.agb, col="blue", type="l")
points(sal3~d.agb, col="black", lty=2, type="l")
points(test_2par~d.agb, col="orange", type="l")
points(exp(choj)~d.agb, col="darkgreen", pch=16)
legend("topleft",
  legend = c("Chojnacky (obs.)", "Sal1", "Sal2", "Sal3", "2 Parameter Power"),
  col = c("darkgreen", "red", "blue", "black", "orange"),
  pch = 16)
```



```r
# Here I want to get the r2 or rmse to compare the different allom functions and params to the "observe

rmse <- function(y_hat,y){
  return(sqrt(mean((y-y_hat)^2)))
}

r2 <- function(y_hat,y){
  RSS<-sum((((y_hat))-(y))^2)
  TSS<-sum(((y)-(mean(y)))^2)
  return(1-RSS/TSS)}
```

```
# remember y_hat is the model predicted agb
# y observed = test_choj
# y_hat(s) = test (sal), test2 (sal), test_2par, py_test (sal)

sal1_rmse <- rmse(sal1,exp(choj))
sal1_R2 <- r2(sal1, exp(choj))

sal2_rmse <- rmse(sal2,exp(choj))
sal2_R2 <- r2(sal2,exp(choj))

sal3_rmse <- rmse(sal3, exp(choj))
sal3_R2 <- r2(sal3, exp(choj))

test_2par_rmse <- rmse(test_2par, exp(choj))
test_2par_R2 <- r2(test_2par, exp(choj))

agb_rmse <- c(sal1_rmse, sal2_rmse, sal3_rmse, test_2par_rmse)
agb_r2 <- c(sal1_R2, sal2_R2, sal3_R2, test_2par_R2)
agb_names <- c("Sal test 1", "Sal test 2", "Sal test 3", "Two Param Power")

agb_mod_fits <- data.frame(agb_rmse, agb_r2, row.names = agb_names)

agb_mod_fits
```

```
##                    agb_rmse    agb_r2
## Sal test 1      734.098689 0.8675944
## Sal test 2      347.141465 0.9703919
## Sal test 3      636.566290 0.9004400
## Two Param Power   3.794937 0.9999965
```

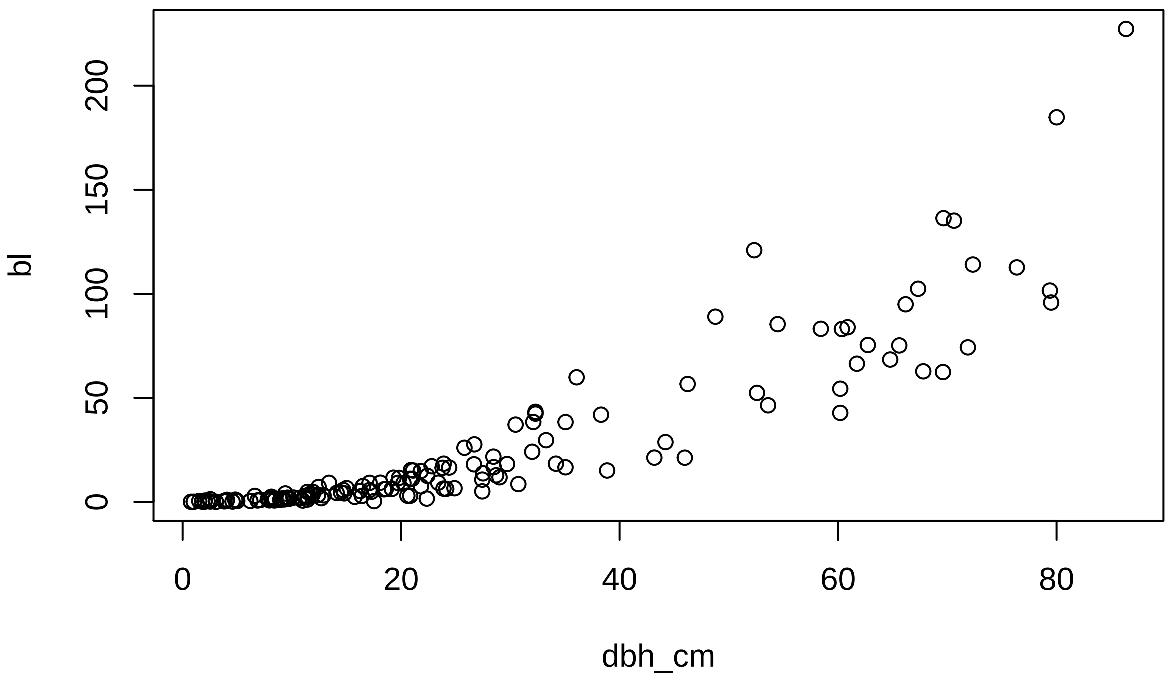
Diamter to Leaf Biomass (d2blmax)

FATES provides three different options for calculating leaf biomass from dbh:

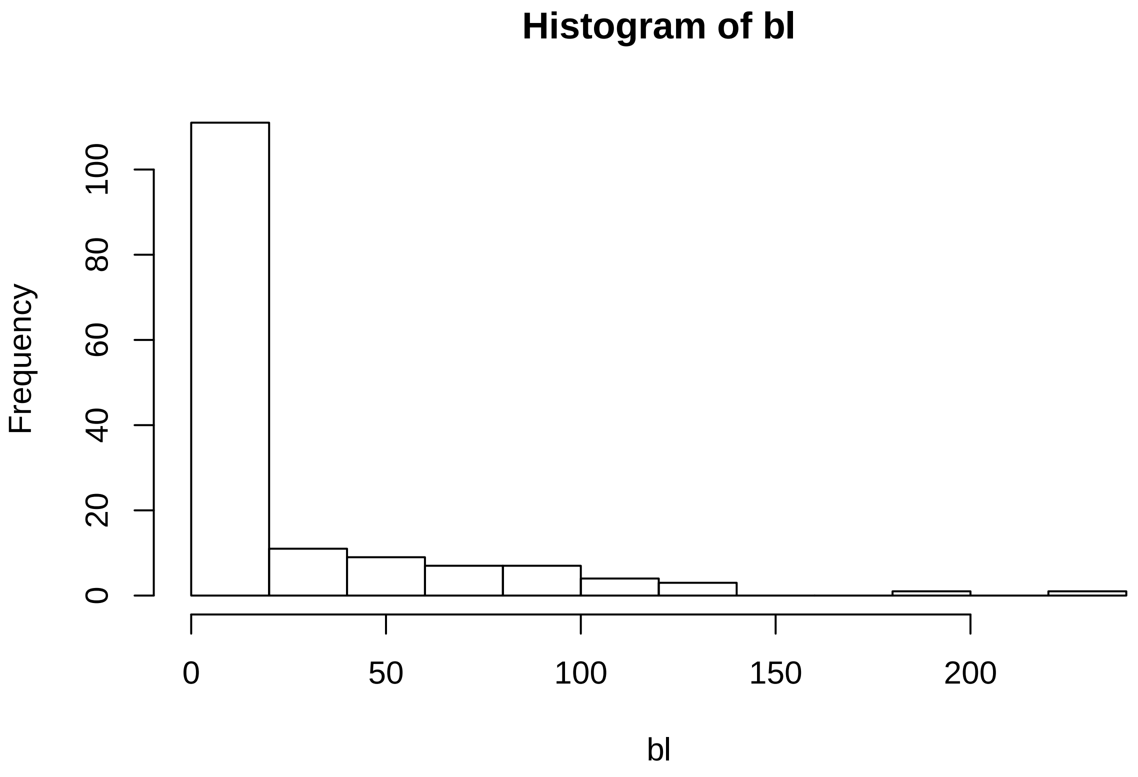
- Saldarriaga et al. (1998)
$$(blmax = p_1 * d^{p_2} * rho^{p_3})$$
- 2 parameter power function
$$(blmax = p_1/c2b * d^{p_2})$$
- 2 parameter power function for large trees
$$(blmax = p_1/c2b * min(d, dbhmax)^{p_2})$$

Here I will test the Saldarriaga and 2 parameter power functions.

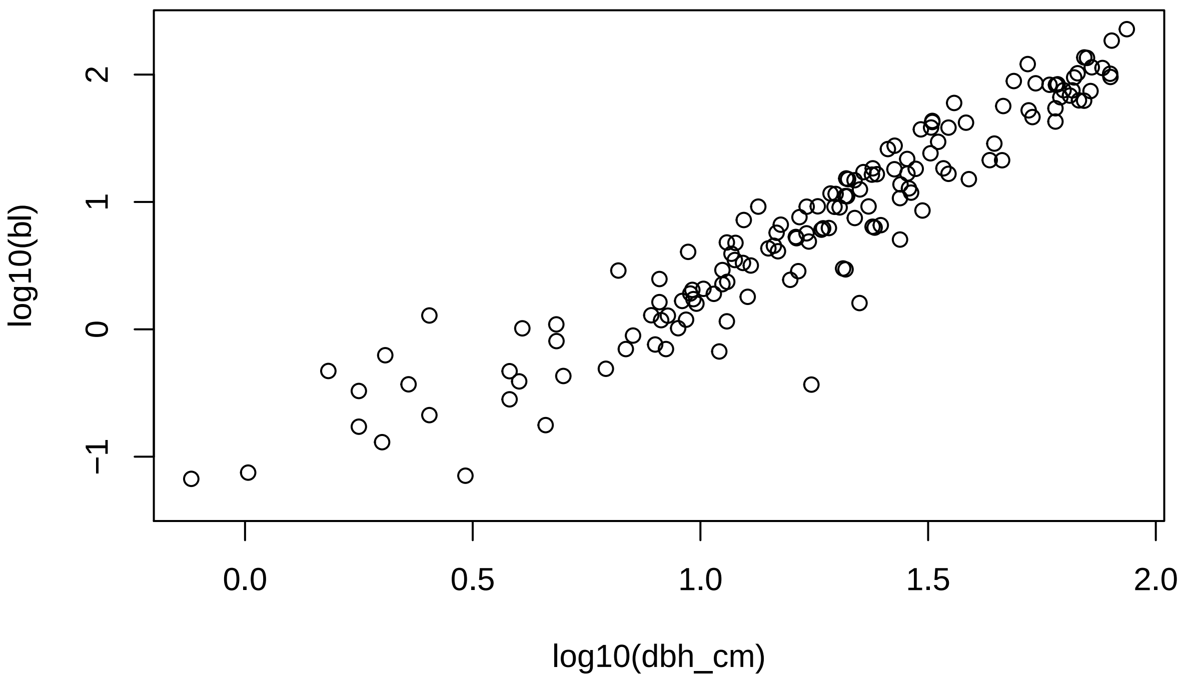
```
bl <- pipo$m.lf

bl_test <-c(0.8, 4.6, 13.2, 30, 36.2, 60, 70.5, 93) # From Polly's sheets
bl_dbh <- c(10, 20, 30, 40, 50, 60, 70, 80) #cm

plot(bl~dbh_cm)
```



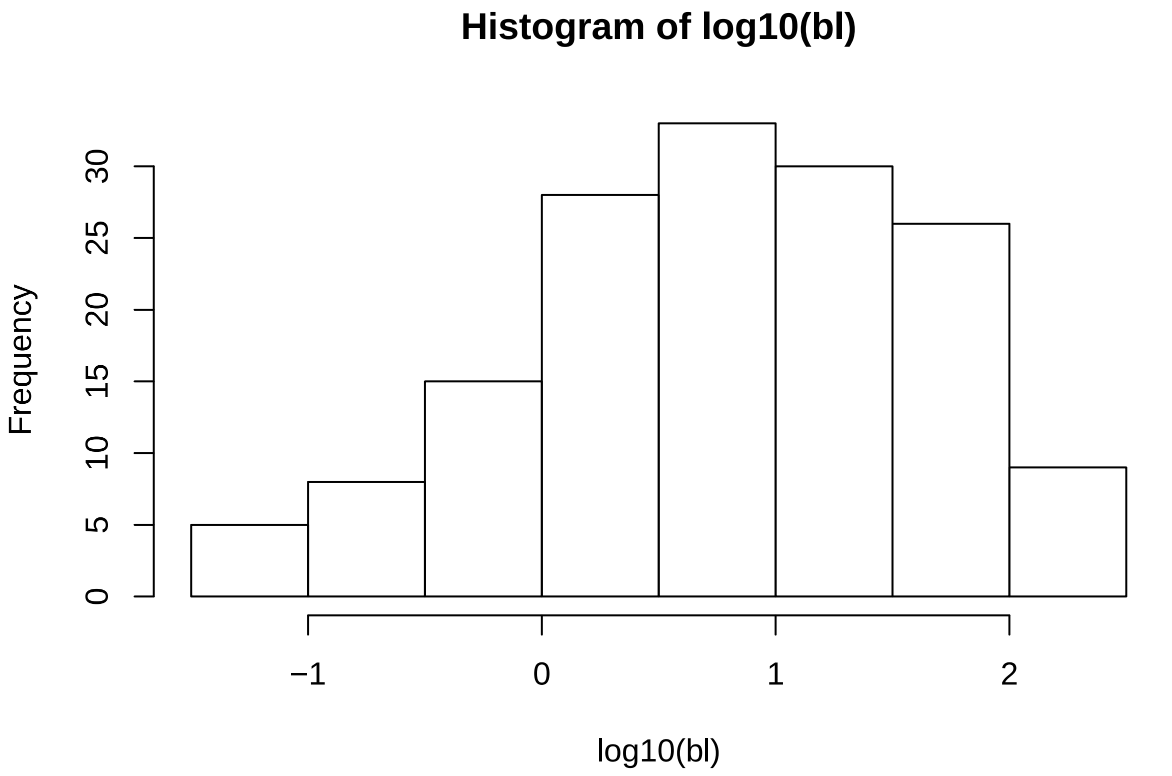
```
hist(bl)
```



```
plot(log10(bl)~log10(dbh_cm))
```



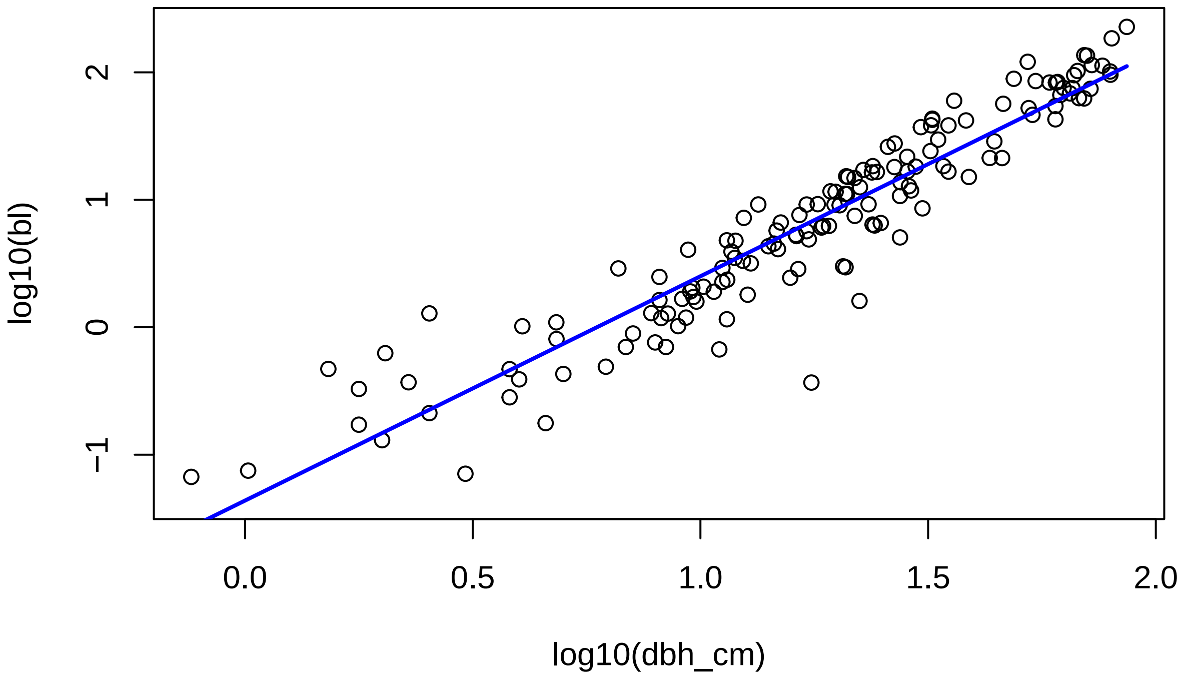
```r
hist(log10(bl))
```



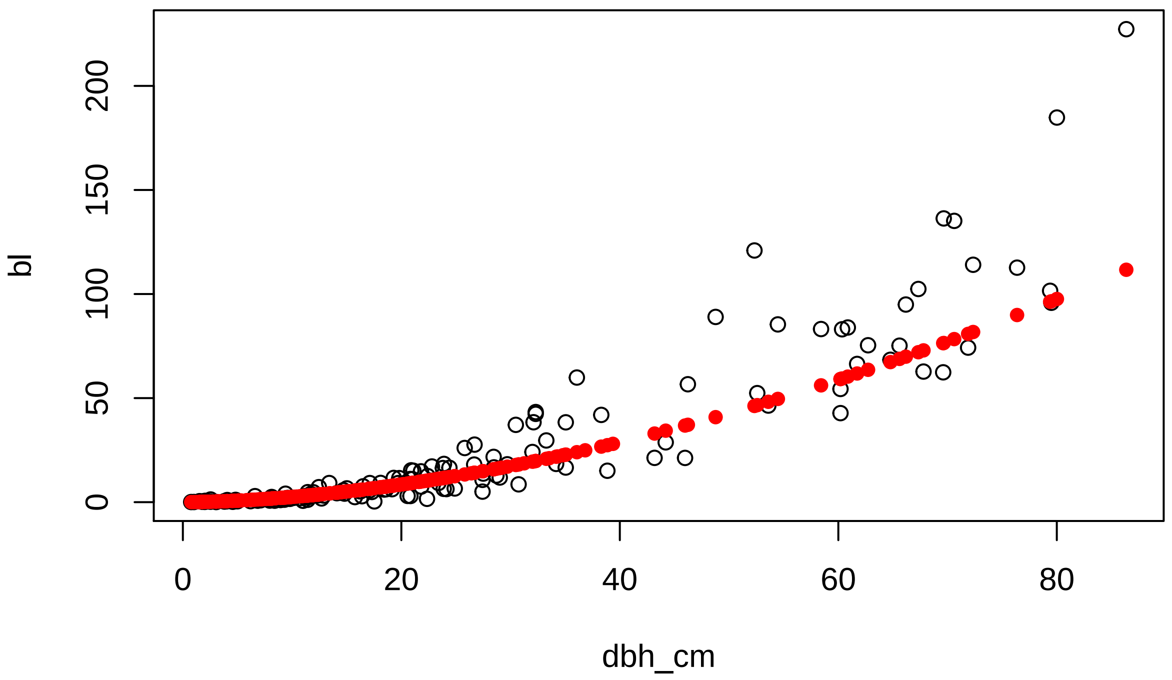
```r
d2blmod <- lm(log10(bl)~log10(dbh_cm))
coef(d2blmod)
```

```
##   (Intercept) log10(dbh_cm)
##     -1.363671      1.762904
```

```
plot(log10(bl)~log10(dbh_cm))
curve((-1.36 + 1.76*x),add=T, col="blue", lwd=2)
```

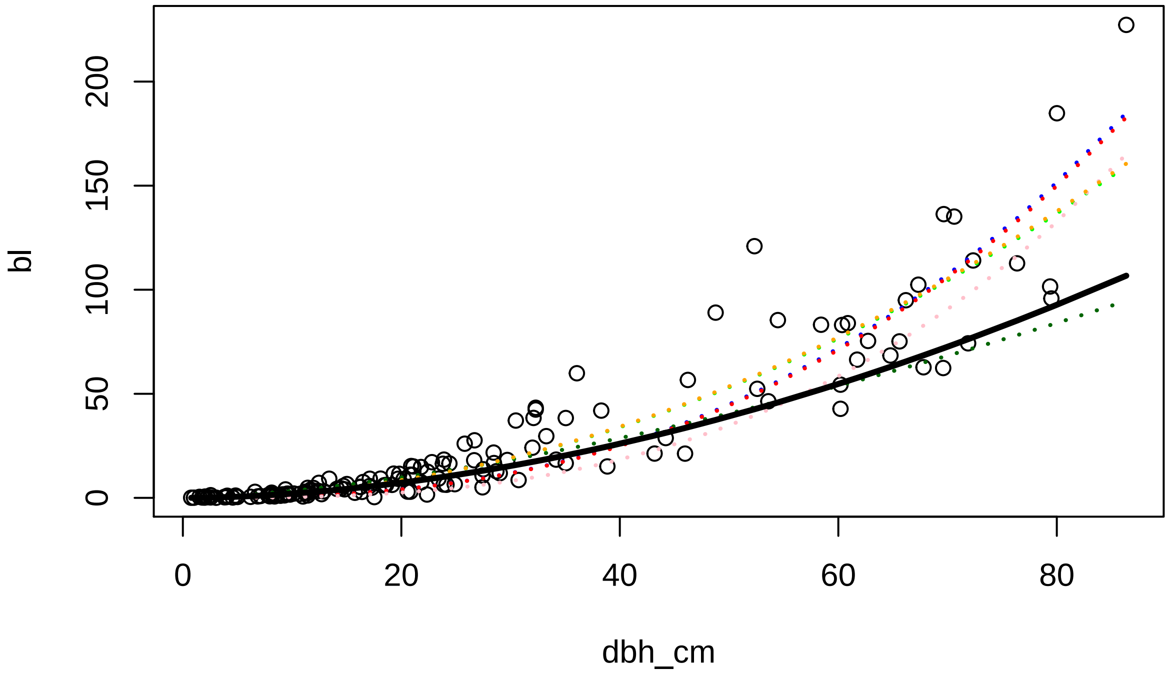


```
bl_lm <- function(dbh, p1, p2){
  blmax <- 10^((log10(dbh))* p1 + p2)
  return(blmax)
}

bl_lm_mod <- bl_lm(sort(dbh_cm), 1.76, -1.36)

plot(bl~dbh_cm)
points(bl_lm_mod~sort(dbh_cm), col="red", pch=16)
```

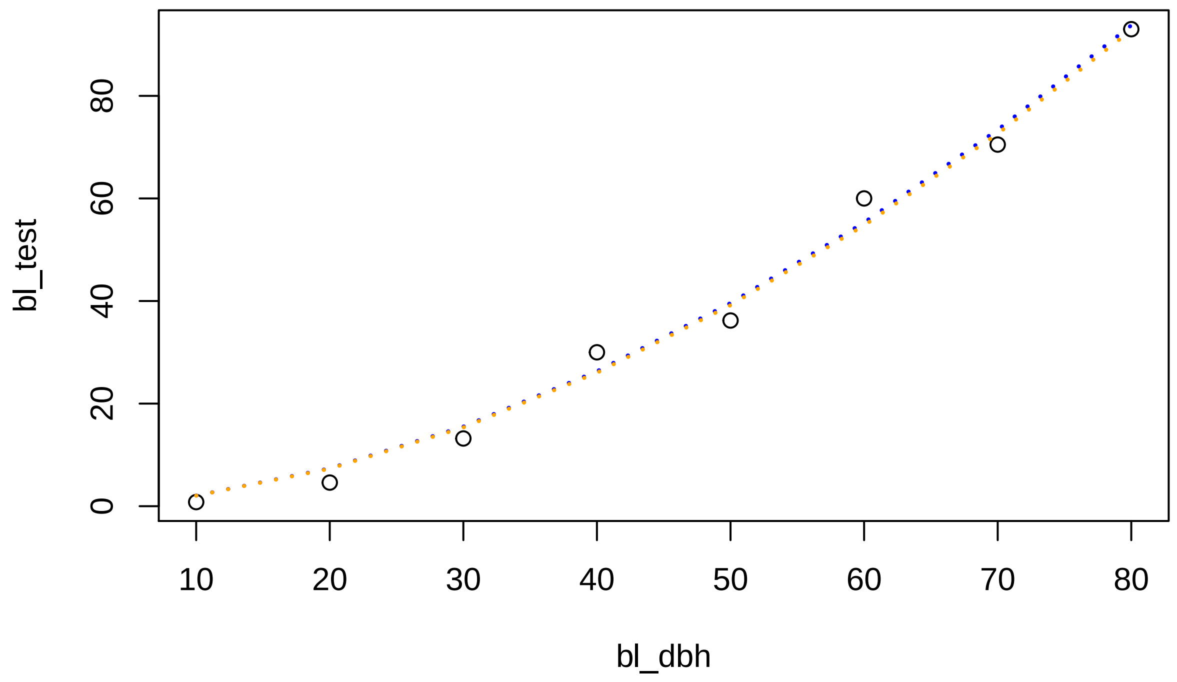


```
bl_sal <- function(p1,d,p2,rho,p3){
  blmax <- (p1 * (d^p2) * (rho^p3))
  return(blmax)
}

bl_2par <- function(p1,d,p2,c2b){
  blmax <- (p1 * (d^p2))/c2b
  return(blmax)
}

# Test difference params

# Sal1: p1= 0.0169, p2= 2.592, p3= 2.251 (from Jupyter notebook curve_fit)
# Sal2: p1= 0.2, p2=1.55, p3=0.75 (from Polly)
# Sal3: p1= 0.02, p2= 2.011, p3= -0.01665 (from nls but setting p1=0.2)
# Sal4: p1= 0.115, p2= 1.834, p3= 1.33 (from nls below, p1=0.115, and using only mean bl for binned dbh

bl_sal1_mod <- bl_sal(0.0169, sort(dbh_cm), 2.592, rho, 2.251)
#bl_sal_mod
bl_sal2_mod <- bl_sal(0.2, sort(dbh_cm), 1.55, rho, 0.75)
#bl_sal2_mod
bl_sal3_mod <- bl_sal(0.2, sort(dbh_cm), 2.011, rho, 2.28)
#bl_sal3_mod
bl_sal4_mod <- bl_sal(0.115, bl_dbh, 1.834, rho, 1.33)

# 2 pwr1: p1= 0.0035 , p2= 2.592 (from jupyter notebook curve_fit)
# 2 pwr2: p1= 0.005, p2= 2
# 2 pwr3: p1= 0.04, p2= 2.01 (from r nls below using full bl and dbh data from BAAD)
# 2 pwr4: p1= 0.06, p2= 1.834 (from nls below using mean BL for binned dbh)

bl_2par_mod <- bl_2par(0.0035, sort(dbh_cm), 2.592, 2)
#bl_2par_mod=
bl_2par2_mod <- bl_2par(0.001, sort(dbh_cm), 2.85, 2) # Brute force
#bl_2par2_mod
bl_2par3_mod <- bl_2par(0.041, sort(dbh_cm), 2.011, 2)
#bl_2par3_mod
bl_2par4_mod <- bl_2par(0.06, bl_dbh, 1.834, 2)

bl_2par5_mod <- bl_2par(0.06, sort(dbh_cm), 1.834, 2)
```

```
plot(bl~dbh_cm)
points(bl_sal1_mod~sort(dbh_cm), type="l", col="blue", lwd=2, lty=3)
points(bl_sal2_mod~sort(dbh_cm), type="l", col="darkgreen", lwd=2, lty=3)
points(bl_sal3_mod~sort(dbh_cm), type="l", col="green", lwd=2, lty=3)
points(bl_2par_mod~sort(dbh_cm), type="l", col="red", lwd=2, lty=3)
points(bl_2par2_mod~sort(dbh_cm), type="l", col="pink", lwd=2, lty=3)
points(bl_2par3_mod~sort(dbh_cm), type="l", col="orange", lwd=2, lty=3)
points(bl_2par5_mod~sort(dbh_cm), type="l", col="black", lwd=3)
```



```
plot(bl_test~bl_dbh)
points(bl_sal4_mod~bl_dbh, type="l", col="blue", lwd=2, lty=3)
points(bl_2par4_mod~bl_dbh, type="l", col="orange", lwd=2, lty=3)
```



```
bl_sal1_rmse <- rmse(bl_sal1_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_sal1_R2 <- r2(bl_sal1_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_2par_rmse <- rmse(bl_2par_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_2par_R2 <- r2(bl_2par_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_sal2_rmse <- rmse(bl_sal2_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_sal2_R2 <- r2(bl_sal2_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_sal3_rmse <- rmse(bl_sal3_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_sal3_R2 <- r2(bl_sal3_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_sal4_rmse <- rmse(bl_sal4_mod, bl_dbh)
bl_sal4_R2 <- r2(bl_sal4_mod, bl_dbh)

bl_2par2_rmse <- rmse(bl_2par2_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_2par2_R2 <- r2(bl_2par2_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_2par3_rmse <- rmse(bl_2par3_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_2par3_R2 <- r2(bl_2par3_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_2par4_rmse <- rmse(bl_2par4_mod, bl_test)
bl_2par4_R2 <- r2(bl_2par4_mod, bl_test)

bl_2par5_rmse <- rmse(bl_2par5_mod, na.omit(bl))
```

```
## Warning in y - y_hat: longer object length is not a multiple of shorter
## object length
```

```
bl_2par5_R2 <- r2(bl_2par5_mod, na.omit(bl))
```

```
## Warning in ((y_hat)) - (y): longer object length is not a multiple of
## shorter object length
```

```
bl_rmse <- c(bl_sal1_rmse, bl_sal2_rmse, bl_sal3_rmse, bl_2par_rmse, bl_2par2_rmse, bl_2par3_rmse, bl_2p
bl_r2 <- c(bl_sal1_R2, bl_sal2_R2, bl_sal3_R2, bl_2par_R2, bl_2par2_R2, bl_2par3_R2, bl_2par5_R2)
bl_names <- c("Sal test 1: BL", "Sal test 2: BL", "Sal test 3:BL","Two Param Power: BL", "Two Param Pow

bl_mod_fits <- data.frame(bl_rmse, bl_r2, row.names = bl_names)

bl_mod_fits
```

```
##                        bl_rmse     bl_r2
## Sal test 1: BL        53.88777 -2.445608
## Sal test 2: BL        46.79779 -1.598580
## Sal test 3:BL         52.31379 -2.247265
## Two Param Power: BL   53.69626 -2.421160
## Two Param Power 2: BL 51.83128 -2.187640
## Two Param Power 3:BL  52.43524 -2.262360
## Two Param Power 5: BL 47.89558 -1.721926
```

```
bl2par.fit <- nls(bl_test ~ bl_2par(p1,bl_dbh, p2, 2), start=list(p1=0.0035, p2=2.5))
bl2par.fit
```

```
## Nonlinear regression model
##   model: bl_test ~ bl_2par(p1, bl_dbh, p2, 2)
##    data: parent.frame()
##      p1      p2
## 0.06039 1.83353
##  residual sum-of-squares: 70.04
##
## Number of iterations to convergence: 20
## Achieved convergence tolerance: 6.307e-06
```

```
blsalda.fit <- nls(bl_test ~ bl_sal(0.12, bl_dbh, p2, 0.367, p3), start=list(p2=1,p3=0.5))
blsalda.fit
```

```
## Nonlinear regression model
##   model: bl_test ~ bl_sal(0.12, bl_dbh, p2, 0.367, p3)
##    data: parent.frame()
##    p2    p3
## 1.834 1.376
##  residual sum-of-squares: 70.04
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 6.26e-06
```

Diamter to Crown Area

FATES provides one option for calculating crown area from dbh. There are actually 3 modes to choose from, but they all reference the same equation.

- A 2 power crown area function

$$carea = spreadterm * dbh^{crownareatodbhexponent}$$

where

$$spreadterm = (spread * d2ca_{max}) + ((1 - spread) * d2ca_{min})$$

and

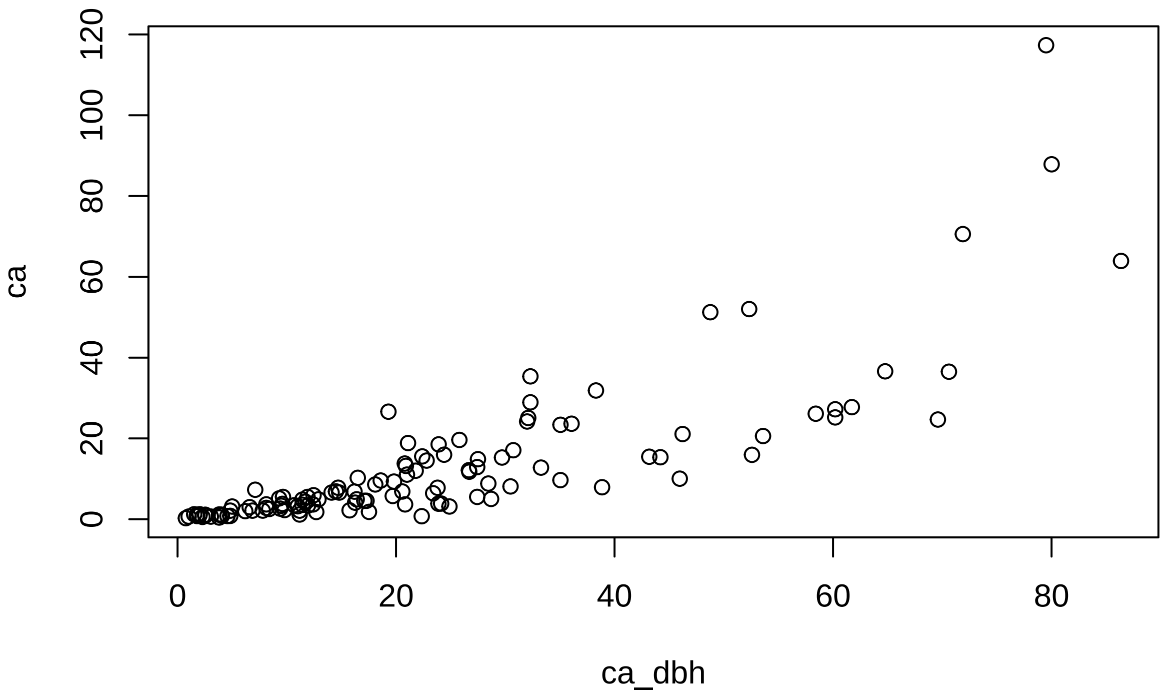
$$crownareatodbhexponent = d2blp2 + d2blediff$$

Combined all together:

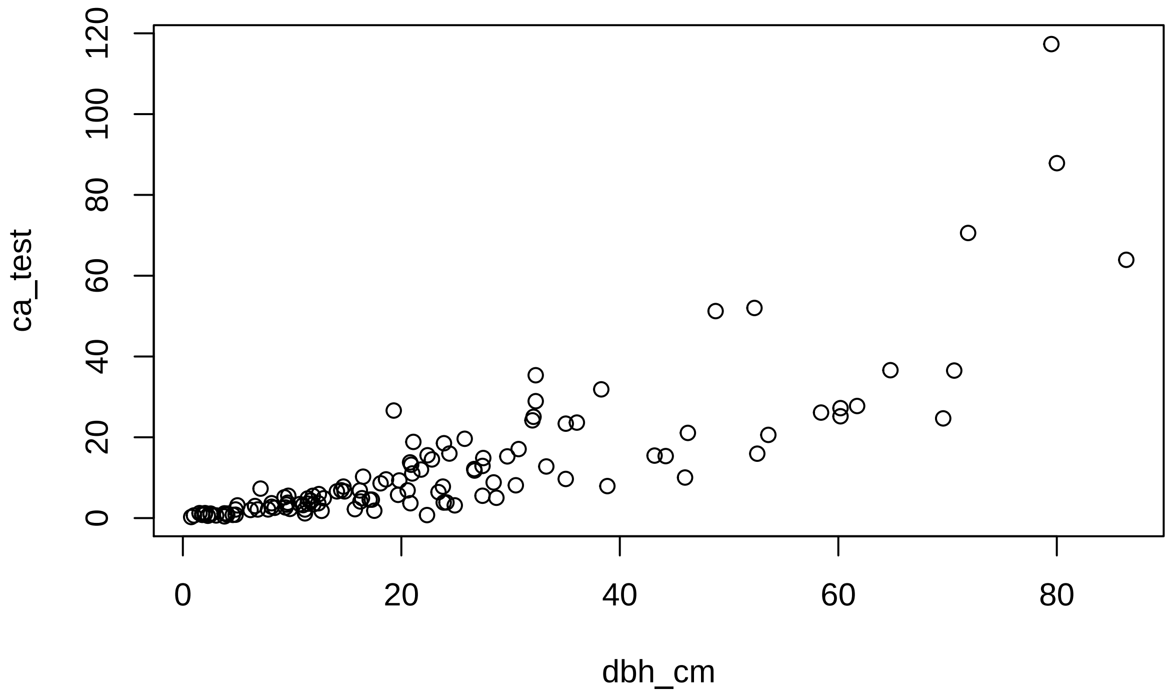
$$carea = (spread * d2ca_{max}) + ((1 - spread) * d2ca_{min}) * dbh^{(d2blp2+d2blediff)}$$

We are given dbh, d2bl p2, d2bl ediff (0), and spread? We need to fit d2camax and min? Or we should know d2ca max and min and only need to fit for spread?

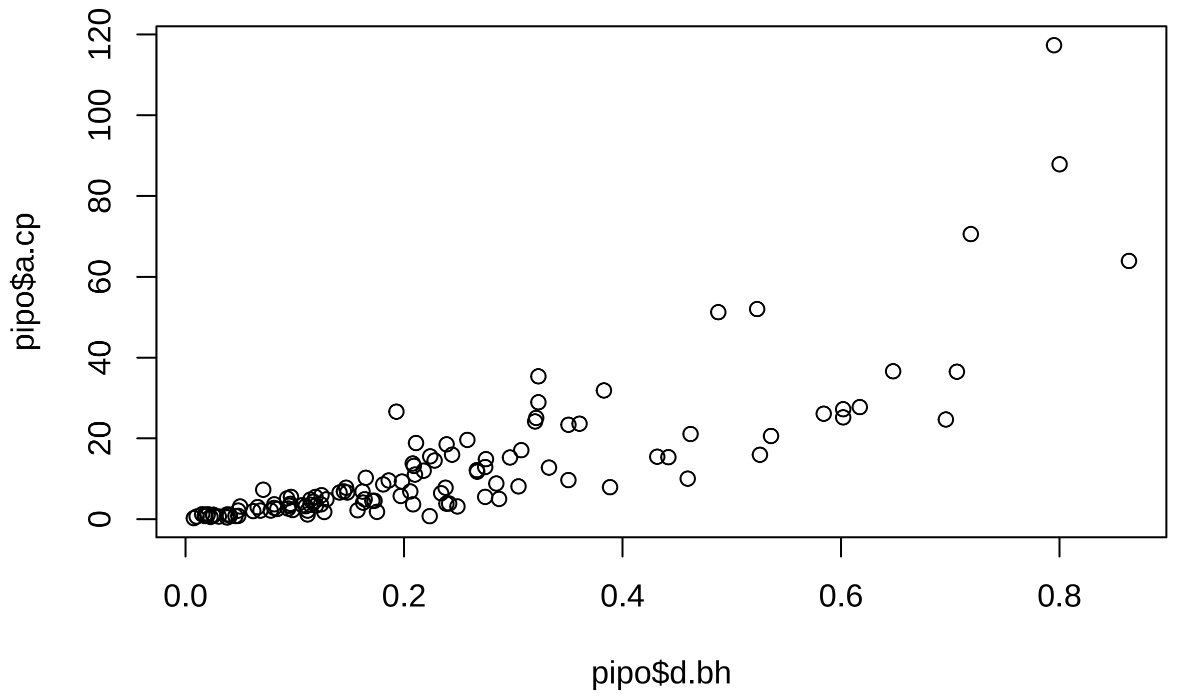
```
ca <- pipo$a.cp[which(pipo$a.cp>0, pipo$dbh>0)]
ca_dbh <- pipo$d.bh[which(pipo$a.cp > 0, pipo$d.bh>0)]*100

plot(ca~ca_dbh)
```



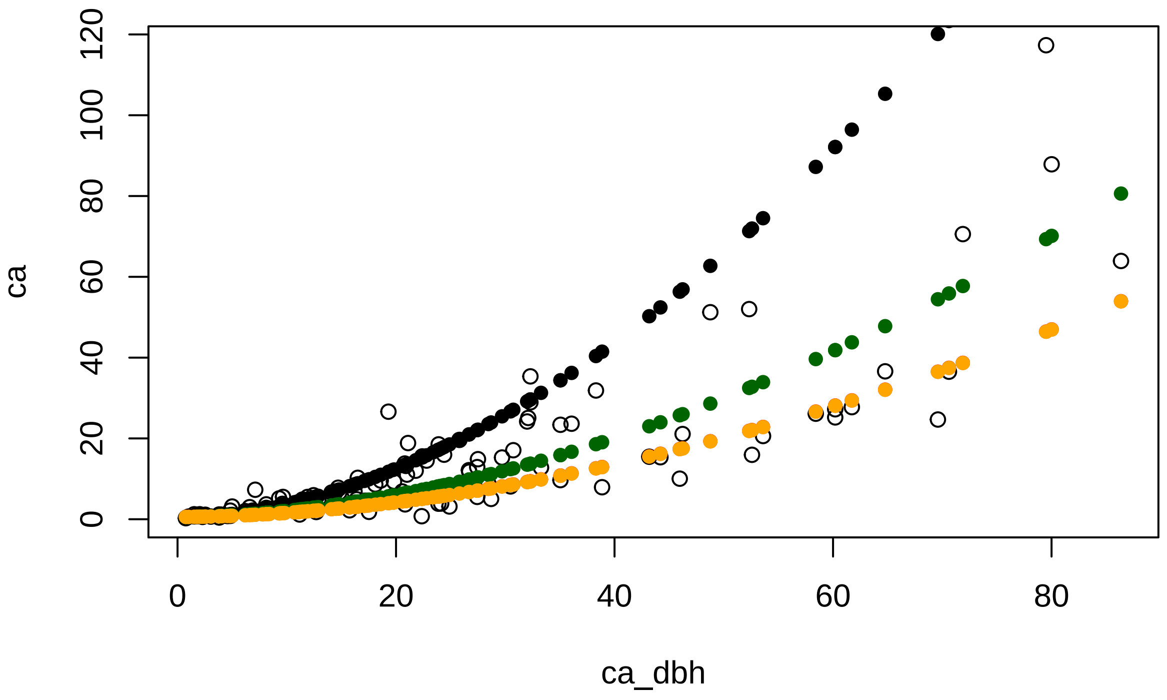
```r
ca_test <- pipo$a.cp

plot(ca_test~dbh_cm)
```



```r
plot(pipo$a.cp ~ pipo$d.bh)
```



```r
d2carea <- function(sprd, d2camax, d2camin, dbh, d2blp2){
  c.area <- ((sprd * d2camax) + ((1-sprd)*d2camin) * dbh^d2blp2)
  return(c.area)
}

ca_mod1 <- d2carea(0.5, 0.7, 0.1, ca_dbh, 1.834)
ca_mod2 <- d2carea(0.85, 0.7, 0.1, ca_dbh, 1.834)
ca_mod3 <- d2carea(0.85, 0.65, 0.15, ca_dbh, 1.834)
ca_mod4 <- d2carea(0.85, 0.65, 0.1, ca_dbh, 1.834)

plot(ca~ca_dbh)
points(ca_mod1 ~ ca_dbh, pch=16, col="black")
points(ca_mod2 ~ ca_dbh, pch=16, col="red")
points(ca_mod3 ~ ca_dbh, pch=16, col="darkgreen")
points(ca_mod4 ~ ca_dbh, pch=16, col="orange")
```



```
# try to fit parameters

#d2ca.fit <- nls(ca ~ d2carea(sprd, d2camax, 0.1, sort(dbh_cm), 0.1834), start=list(sprd=0.4, d2camax=0
#d2ca.fit
```

I want to try something where I step through spread values from 0.1 to 0.9 at intervals of 0.1 and calculate the d2camax.

In the mean time I need to figure out the issue with NAs so that I can calculate some $ r^2 $ values.

```
#ca_mod1[is.na(ca_mod1)] <- min(ca)
#ca_mod2[is.na(ca_mod2)] <- min(ca)
#ca_mod3[is.na(ca_mod3)] <- min(ca)
#ca_mod4[is.na(ca_mod4)] <- min(ca)

ca_mod1_R2 <- r2(ca_mod1[7:125], ca[7:125])
ca_mod2_R2 <- r2(ca_mod2[7:125], ca[7:125])
ca_mod3_R2 <- r2(ca_mod3[7:125], ca[7:125])
ca_mod4_R2 <- r2(ca_mod4[7:125], ca[7:125])

ca_mod1_R2
```

```
## [1] -1.191073
```

```
ca_mod2_R2
```

```
## [1] 0.5956255
```

```
ca_mod3_R2
```

```
## [1] 0.7179147
```

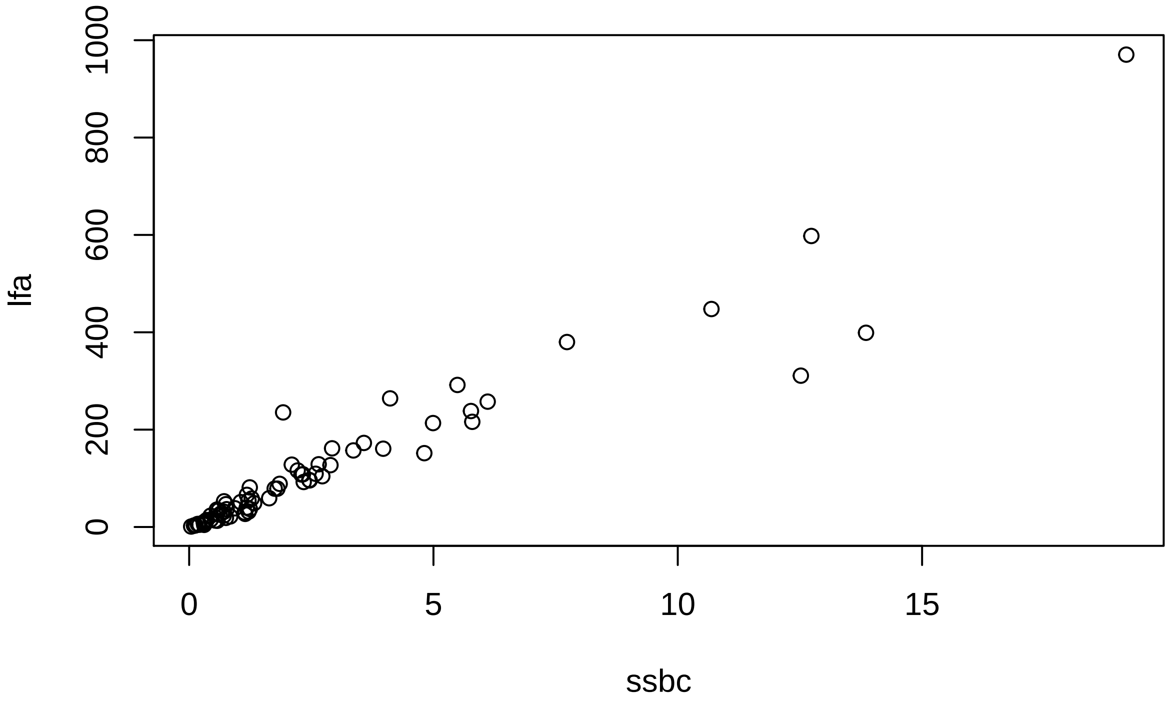
```
ca_mod4_R2
```

```
## [1] 0.5942858
```

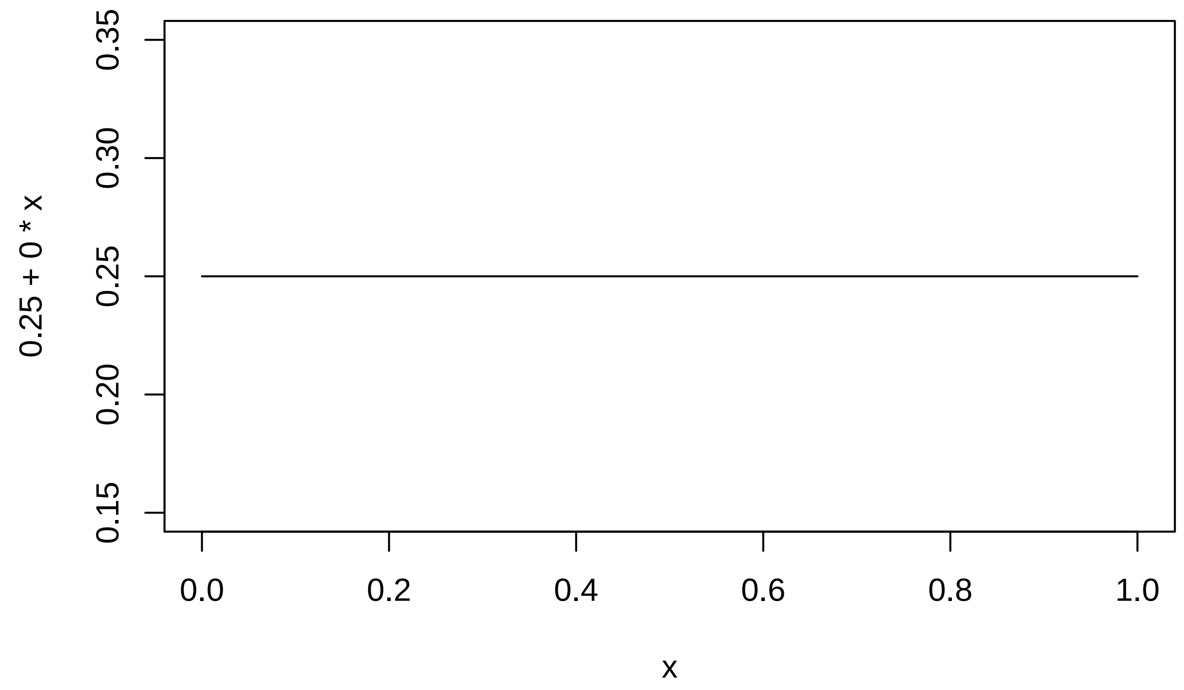
Diameter to sapwood carbon

FATES provides only one sapwood allometry model. The slope of the ratio of leaf area to stem area (la:sa) to diameter line is zero.

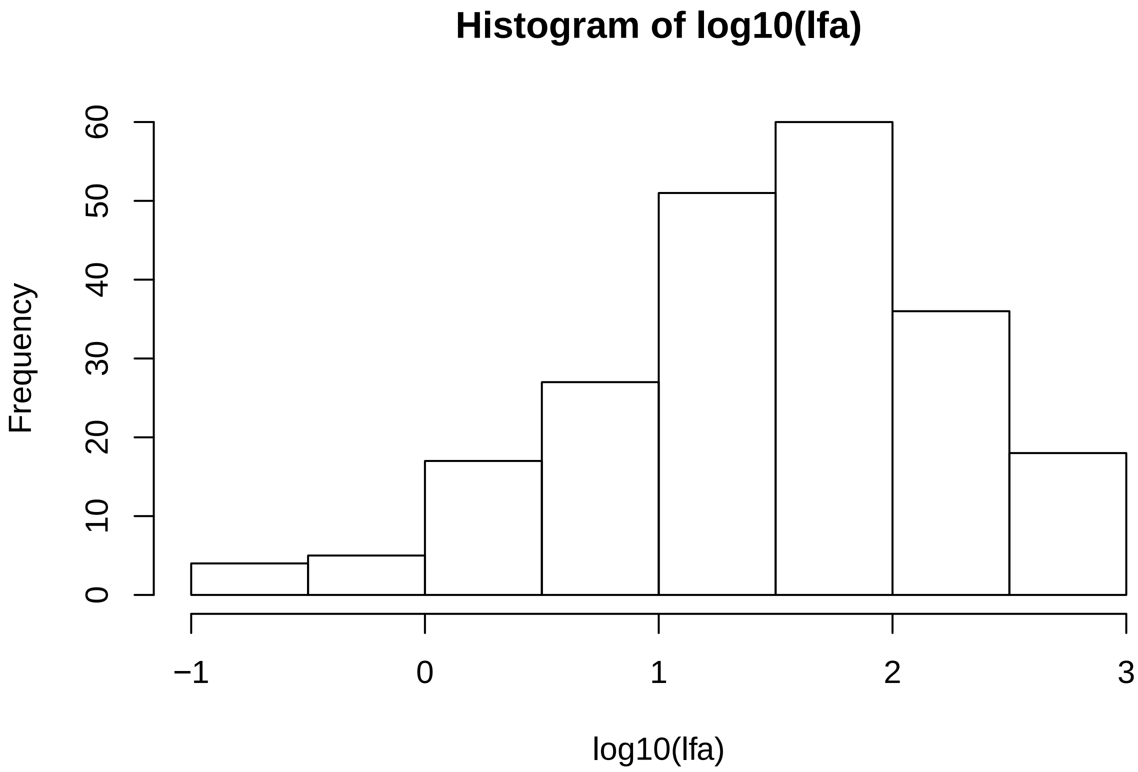
```
# Well I think one really only needs to determine the ratio of leaf area per sap area

# sapwood area

#ssba <- pipo$a.ssba
# ssbbh <= pipo$a.ssbh
ssbc <- pipo$a.ssbc *100 # sapwood area at base of crown

# leaf area

lfa <- pipo$a.lf

plot(lfa~ssbc)
```



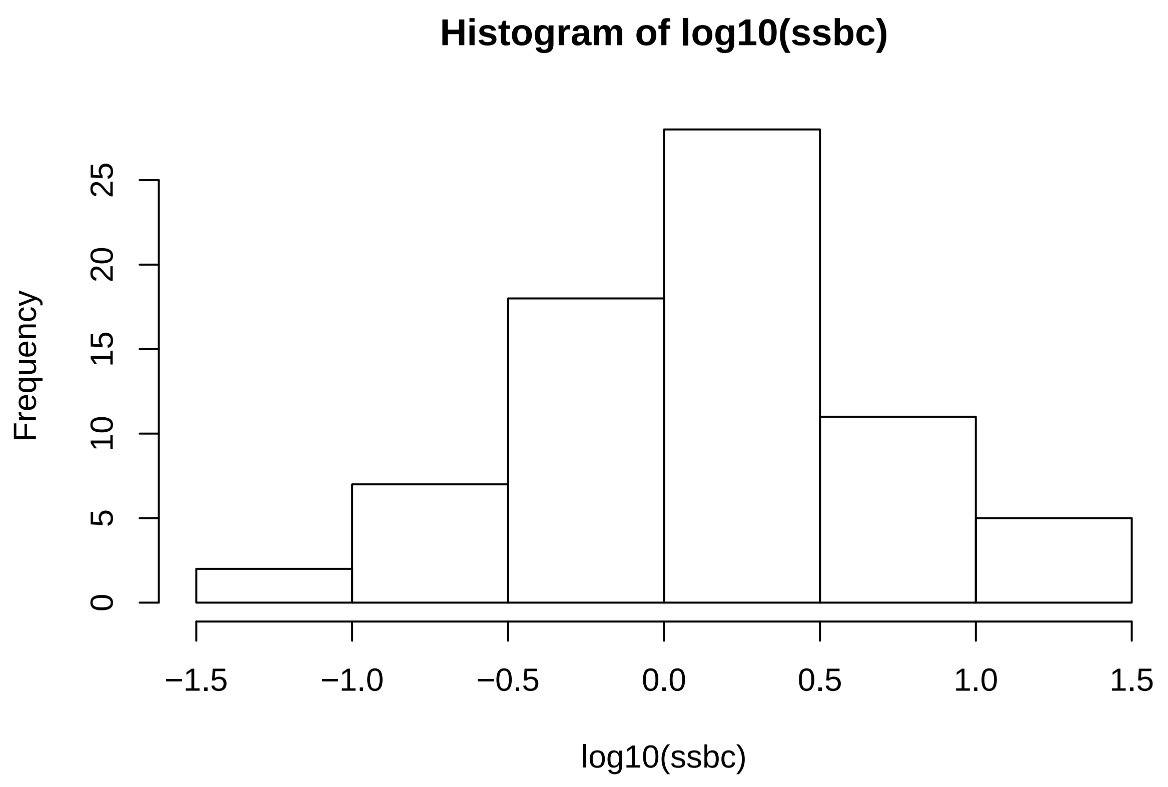
```
curve(0.25 + 0*x)
```



```r
hist(log10(lfa))
```



```r
hist(log10(ssbc))
```



```r
# we know that the slope is set to 0
# so we just need to find the intercept

sap_mod <- lm(log10(lfa) ~ log10(ssbc))
coef(sap_mod)
```

```
## (Intercept) log10(ssbc)
##    1.584270    1.097669
```

```r
# Create a simple linear function to determine the intercept with a slope of 0

sap_lin <- function(int, h, slope){
  lapsa <- int + h * slope
  return(lapsa)
}


sap_lin(-0.8, pipo$h.t, 0)
```

```
##   [1] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [15] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [29] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [43] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [57] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [71] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [85] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
##  [99] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [113] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [127] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [141] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [155] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [169] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [183] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [197] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [211] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [225] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [239] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [253] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
## [267] -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
```

```
#sap_mod.fit <- ( ~ sap_lin(int, pipo$h.t, 0), start=list(int=-1), trace = TRUE)
```