Fisheries-in-R.R

jason.doll

2023-12-06

#R code based on Introductory Fisheries Analysis with R  
#Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.  
  
#########################################  
#######Day 3: Fisheries Analysis in R####  
#########################################  
  
###Topics################################  
# Age-length keys  
# Mortality  
# Individual growth  
# Stock-Recruitment  
# Population Estimates  
# Size structure  
# Weight-length relationships  
# Condition factors  
#########################################  
  
  
#Load packages for today  
library(FSA)

## ## FSA v0.9.5. See citation('FSA') if used in publication.  
## ## Run fishR() for related website and fishR('IFAR') for related book.

library(FSAdata)

## ## FSAdata v0.4.1. See ?FSAdata to find data for specific fisheries analyses.

library(nlstools) #needed for calculating 95% confidence intervals

##   
## 'nlstools' has been loaded.

## IMPORTANT NOTICE: Most nonlinear regression models and data set examples

## related to predictive microbiolgy have been moved to the package 'nlsMicrobio'

library(ggplot2)  
library(ggtext)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

#Age-length keys#######  
#1. Add length intervals  
#2. Separate data into and "age" and "length" data frame  
#3. Create a frequency table  
#4. Create a proportions table  
#5. Apply age-length key to fish without an age  
#6. Summarize length at age.  
  
#Load a bass data set from Florida, name includes an upper case "O" not zero  
RockBassLO2 <- FSAdata::RockBassLO2  
headtail(RockBassLO2)

## age tl  
## 1 6 218  
## 2 5 184  
## 3 7 211  
## 1286 NA 173  
## 1287 NA 214  
## 1288 NA 194

#Add length category  
#Requires column with lengths, the data frame name, a numeric value   
#identifying the starting length measurement and the width of the length   
#measurement category  
RockBassLO2 <- lencat(~tl,data=RockBassLO2,  
 startcat=110,w=10)  
headtail(RockBassLO2)

## age tl LCat  
## 1 6 218 210  
## 2 5 184 180  
## 3 7 211 210  
## 1286 NA 173 170  
## 1287 NA 214 210  
## 1288 NA 194 190

#Data contains missing age values, NA. We want to assign ages based on the ALK.   
  
#First create an "age" data frame with all complete records  
rb.age <- RockBassLO2 %>%  
 dplyr::filter( !is.na(age) )  
  
headtail(rb.age)

## age tl LCat  
## 1 6 218 210  
## 2 5 184 180  
## 3 7 211 210  
## 133 10 274 270  
## 134 8 266 260  
## 135 9 265 260

#then create a "length" data frame with records with missing ages.  
#Note the only difference from above is the !  
rb.len <- RockBassLO2 %>%  
 dplyr::filter( is.na(age) )  
  
headtail(rb.len)

## age tl LCat  
## 1 NA 172 170  
## 2 NA 173 170  
## 3 NA 175 170  
## 1151 NA 173 170  
## 1152 NA 214 210  
## 1153 NA 194 190

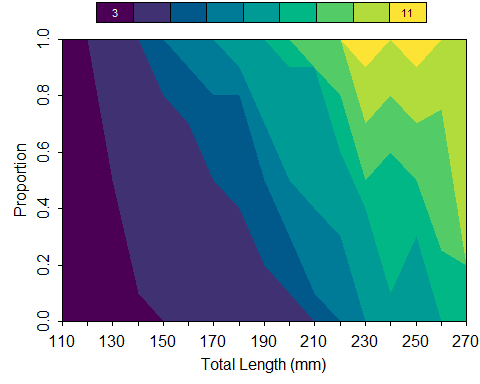
#Construct the age-length key  
  
#Use the xtabs() function to construct a contingency table of the number of   
#fish in each length and age category. The row variable is the first variable   
#and the column variable is the second variable  
rb.raw <- xtabs(~LCat+age,data=rb.age)  
rb.raw

## age  
## LCat 3 4 5 6 7 8 9 10 11  
## 110 1 0 0 0 0 0 0 0 0  
## 120 1 0 0 0 0 0 0 0 0  
## 130 2 2 0 0 0 0 0 0 0  
## 140 1 9 0 0 0 0 0 0 0  
## 150 0 8 2 0 0 0 0 0 0  
## 160 0 7 2 1 0 0 0 0 0  
## 170 0 5 3 2 0 0 0 0 0  
## 180 0 4 4 1 1 0 0 0 0  
## 190 0 2 3 2 3 0 0 0 0  
## 200 0 1 2 2 4 1 0 0 0  
## 210 0 0 1 3 5 0 1 0 0  
## 220 0 0 0 3 3 2 2 0 0  
## 230 0 0 0 0 4 1 2 2 1  
## 240 0 0 0 0 1 5 2 2 0  
## 250 0 0 0 0 3 2 2 2 1  
## 260 0 0 0 0 0 1 2 1 0  
## 270 0 0 0 0 0 1 0 4 0

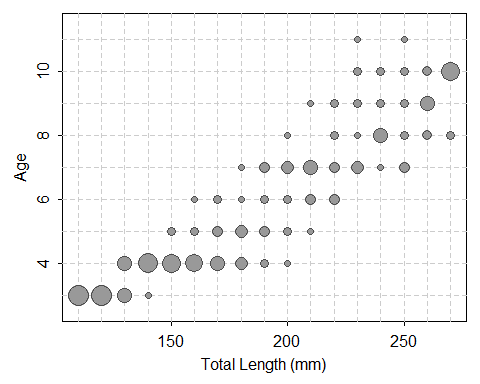
#convert counts to proportions using the prop.table() function. Requires the   
#table created above and margin=1.  
#margin=1 tells R to calculate proportions by row  
#margin=2 tells R to calculate proportions by column  
rb.key <- prop.table(rb.raw,margin=1)  
rb.key

## age  
## LCat 3 4 5 6 7 8 9 10 11  
## 110 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 120 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 130 0.50 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 140 0.10 0.90 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 150 0.00 0.80 0.20 0.00 0.00 0.00 0.00 0.00 0.00  
## 160 0.00 0.70 0.20 0.10 0.00 0.00 0.00 0.00 0.00  
## 170 0.00 0.50 0.30 0.20 0.00 0.00 0.00 0.00 0.00  
## 180 0.00 0.40 0.40 0.10 0.10 0.00 0.00 0.00 0.00  
## 190 0.00 0.20 0.30 0.20 0.30 0.00 0.00 0.00 0.00  
## 200 0.00 0.10 0.20 0.20 0.40 0.10 0.00 0.00 0.00  
## 210 0.00 0.00 0.10 0.30 0.50 0.00 0.10 0.00 0.00  
## 220 0.00 0.00 0.00 0.30 0.30 0.20 0.20 0.00 0.00  
## 230 0.00 0.00 0.00 0.00 0.40 0.10 0.20 0.20 0.10  
## 240 0.00 0.00 0.00 0.00 0.10 0.50 0.20 0.20 0.00  
## 250 0.00 0.00 0.00 0.00 0.30 0.20 0.20 0.20 0.10  
## 260 0.00 0.00 0.00 0.00 0.00 0.25 0.50 0.25 0.00  
## 270 0.00 0.00 0.00 0.00 0.00 0.20 0.00 0.80 0.00

#Construct age-length key in a single block of code using a pipe  
rb.key <- xtabs(~LCat+age,data=rb.age) %>%  
 prop.table(margin=1)  
  
  
  
#Visualizing Age-Length key  
alkPlot(rb.key, type = "area", showLegend = TRUE,  
 leg.cex = 0.7, xlab = "Total Length (mm)")



#Bubble size is proportional to the number of fish   
#in each length interval  
alkPlot(rb.key, type = "bubble",   
 xlab = "Total Length (mm)")



#now we are ready to assign ages to individuals without ages. This example   
#uses a semi-random method of assigning ages to individuals  
#Suppose there are 20 fish in a length category that need assigned an age and  
#the age-length key says that 75% in the length category are age-6 and 25% are age-7.   
#The age assignment for age-6 is (0.75 \* 20) = 15  
#The age assignment for age-7 is (0.25 \* 20) = 5  
  
#What about the fractional assignments for 22 fish?  
#(0.75 \* 22) = 16.5  
#(0.25 \* 22) = 5.5  
#FSA rounds down so that 16 fish are assigned age-6 and 5 fish are assigned age-7.  
  
#The remaining fish is assigned to age-6 with a   
#probability of 0.75 or age-7 with a probability of 0.25  
  
rb.len1 <- alkIndivAge(rb.key, age ~ tl,   
 data = rb.len)  
head(rb.len)

## age tl LCat  
## 1 NA 172 170  
## 2 NA 173 170  
## 3 NA 175 170  
## 4 NA 171 170  
## 5 NA 173 170  
## 6 NA 184 180

head(rb.len1)

## age tl LCat  
## 1 4 172 170  
## 2 4 173 170  
## 3 4 175 170  
## 4 4 171 170  
## 5 5 173 170  
## 6 4 184 180

#Combine aged and unaged (but with new ages) samples  
rb.combined <- rbind(rb.age, rb.len1)  
  
# Calculate mean length-at-age assuming fully   
#random selection  
Summarize(tl~age,data=rb.combined,digits=2)

## age n mean sd min Q1 median Q3 max  
## 1 3 5 129.40 12.28 111 125.00 131 137.00 143  
## 2 4 319 175.83 15.74 130 164.00 175 187.00 208  
## 3 5 264 187.32 14.05 153 178.00 187 197.00 217  
## 4 6 215 199.77 17.37 163 187.50 202 214.50 228  
## 5 7 310 209.48 15.61 180 198.00 208 217.00 258  
## 6 8 76 227.09 17.59 200 208.00 224 242.25 270  
## 7 9 62 228.92 13.41 210 220.25 227 235.75 265  
## 8 10 31 245.39 14.40 230 234.50 243 253.00 278  
## 9 11 6 238.17 10.09 230 233.00 236 236.75 258

# age frequency distribution  
af <- xtabs(~age,data=rb.combined)  
# proportional age distribution  
ap <- prop.table(af)  
ap

## age  
## 3 4 5 6 7 8   
## 0.003881988 0.247670807 0.204968944 0.166925466 0.240683230 0.059006211   
## 9 10 11   
## 0.048136646 0.024068323 0.004658385

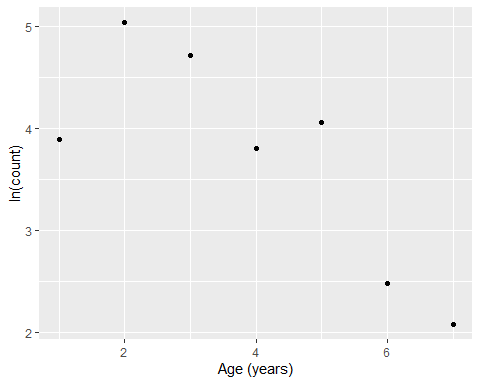
#Calculate mean length-at-age following Bettoli and Miranda 2001.   
#Used when fish are aged with a stratified design to get number of   
#fish in each length interval in the entire sample.  
  
#Generate number per length class  
len.n <- xtabs(~LCat,data=RockBassLO2)  
  
#Calculate mean length at age and SD  
alkMeanVar(rb.key, tl ~ LCat + age,   
 data = rb.age,   
 len.n = len.n)

## age mean sd  
## 1 3 129.6667 12.13072  
## 2 4 177.0340 16.05173  
## 3 5 187.2448 13.77540  
## 4 6 200.1241 17.26455  
## 5 7 208.3250 15.40027  
## 6 8 227.2428 16.58362  
## 7 9 227.4819 14.56039  
## 8 10 246.0548 14.45170  
## 9 11 241.3590 10.11402

#Mortality#######  
#This example will calculate instantaneous total mortality (Z)  
  
#Create a data frame for say, Brook Trout  
bkt <- data.frame(age=c(1:7),  
 ct=c(49,155,112,45,58,12,8))  
  
bkt

## age ct  
## 1 1 49  
## 2 2 155  
## 3 3 112  
## 4 4 45  
## 5 5 58  
## 6 6 12  
## 7 7 8

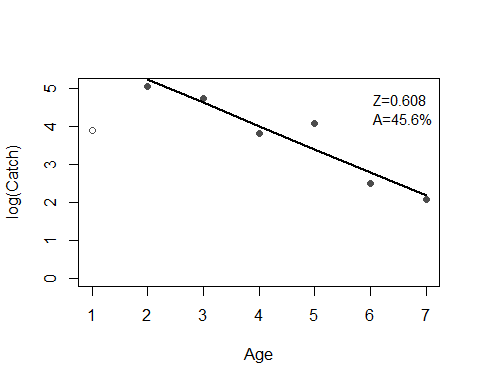
#Create a scatterplot with log(ct) to   
#identify the descending limb of the catch curve  
ggplot(bkt, aes(x = age, y = log(ct))) +  
 geom\_point() +  
 xlab("Age (years)") +   
 ylab("ln(count)")



#The catchCurve() function requires  
#1. Formula in the form of catch ~ age  
#2. A data argument, does not have to contain only the descending limb.  
#3. A required age2use argument that specifies the ages to use  
  
bktcc <- catchCurve(ct ~ age, data = bkt, ages2use=2:7)  
  
#The summary function will return  
#the instantaneous mortality (Z)  
#and annual mortality (A)  
summary(bktcc)

## Estimate Std. Error t value Pr(>|t|)  
## Z 0.6076261 0.0944847 6.430947 0.003006749  
## A 45.5357746 NA NA NA

#the plot() function will create a plot of the   
#catch curve  
plot(bktcc)

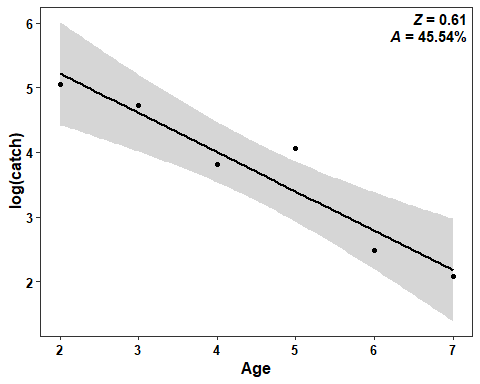


#remove age-1  
bkt\_sub <- bkt %>%   
 dplyr::filter(age > 1)  
#Create a plot using ggplot with confidence bands  
lbl <- paste0("\*Z\* = ",round(coef(bktcc)[["Z"]],2),  
 "<br>\*A\* = ",round(coef(bktcc)[["A"]],2),"%")  
lbl

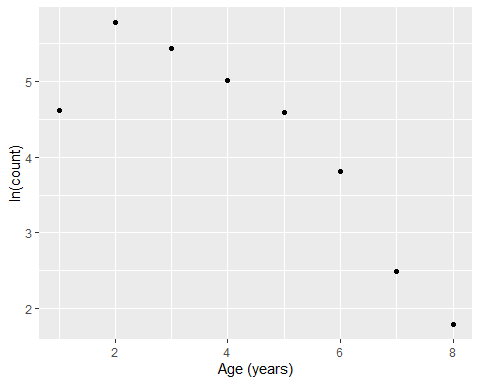
## [1] "\*Z\* = 0.61<br>\*A\* = 45.54%"

ggplot(data=bkt\_sub,mapping=aes(x=age,y=log(ct))) +  
 geom\_smooth(method="lm",color="black") +  
 geom\_point() +  
 ylab("log(catch)")+  
 xlab("Age")+  
 theme\_bw() +  
 theme(panel.grid=element\_blank(),  
 axis.title=element\_text(size=12,face="bold"),  
 axis.text=element\_text(size=10,face="bold",color="black")) +  
 annotate(geom="richtext",x=Inf,y=Inf,vjust=1,hjust=1,label=lbl,  
 label.color=NA,fontface="bold")

## `geom\_smooth()` using formula = 'y ~ x'



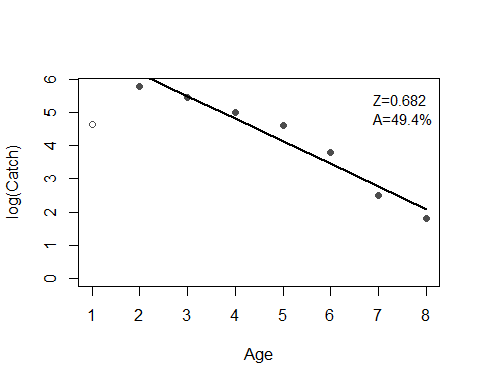
###Mortality Practice#############  
#Create a catch curve plot of the data below.  
#Calculate Z and A from the following data frame  
lmbcatch <- data.frame(age=1:8,  
 ct=c(102, 325, 230, 150, 99, 45, 12, 6))  
  
###Answer########  
ggplot(lmbcatch, aes(x = age, y = log(ct))) +  
 geom\_point() +  
 xlab("Age (years)") +   
 ylab("ln(count)")



lmbcc <- catchCurve(ct ~ age, data = lmbcatch,   
 ages2use=2:8)  
summary(lmbcc)

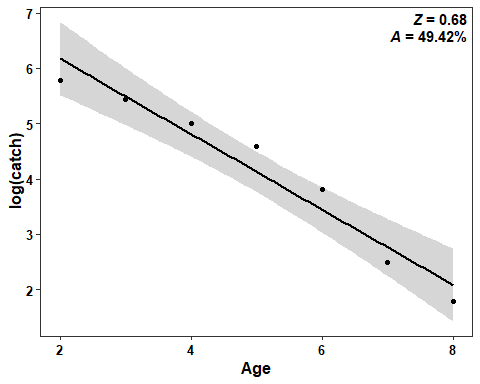
## Estimate Std. Error t value Pr(>|t|)  
## Z 0.6816613 0.07100172 9.600631 0.0002077792  
## A 49.4223931 NA NA NA

plot(lmbcc)



#Or  
  
lmbcatch\_sub <- lmbcatch %>%   
 dplyr::filter(age > 1)  
  
lbl <- paste0("\*Z\* = ",round(coef(lmbcc)[["Z"]],2),  
 "<br>\*A\* = ",round(coef(lmbcc)[["A"]],2),"%")  
  
ggplot(data=lmbcatch\_sub,mapping=aes(x=age,y=log(ct))) +  
 geom\_smooth(method="lm",color="black") +  
 geom\_point() +  
 ylab("log(catch)")+  
 xlab("Age")+  
 theme\_bw() +  
 theme(panel.grid=element\_blank(),  
 axis.title=element\_text(size=12,face="bold"),  
 axis.text=element\_text(size=10,face="bold",color="black")) +  
 annotate(geom="richtext",x=Inf,y=Inf,vjust=1,hjust=1,label=lbl,  
 label.color=NA,fontface="bold")

## `geom\_smooth()` using formula = 'y ~ x'

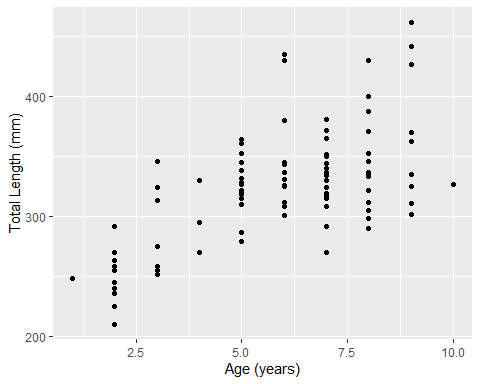


###End Answer###########  
  
#LVB Growth Model workflow############  
#Non-linear models can be challenging to fit using maximum likelihood.  
#This is particularly true for the LVB growth model because of the correlated parameters  
#and when data sets are missing a sufficient number of older/bigger fish.  
  
#The workflow demonstrated here will present a combination of frequentist and Bayesian methods.  
#1. Fit LVB growth model using the maximum likelihood with default settings  
#2. Adjust the Gauss-Newton algorithm settings to increase iterations  
#3. Use a different algorithm (Port) that allows the user to specify the upper and lower bounds of parameters  
#4. Use Bayesian techniques with non-informative priors  
#5. Use Bayesian techniques with informative priors  
  
  
#1. Fit LVB growth model using the default maximum likelihood algorithms with default settings####  
#a. Select data  
#b. Specify the growth model  
#c. Specify starting values and estimate parameters  
#d. Summarize results  
  
#Code for fitting a von Bertalanffy Growth Model  
  
#Load Croaker2 data from the FSAdata package  
Croaker2 <- FSAdata::Croaker2  
  
help("Croaker2")

## starting httpd help server ...

## done

#Subset to only Males  
crm <- subset(Croaker2, sex=="M")  
  
  
#plot the data to visualize trends  
ggplot(crm, aes(x = age, y = tl)) +  
 geom\_point() +  
 xlab("Age (years)") +   
 ylab("Total Length (mm)")



#Select the von Bertalanffy Growth model to use  
#"typical" will use the traditional LVB model  
# Linf \* (1 - exp(-K \* (t - t0)))  
vbT <- vbFuns("typical")  
  
  
#Use the non-linear least squares algorithm to   
#estimate parameters. vbT() arguments must be in   
#the order: age, Linf, K, t0 if using "typical"  
fitCroaker <- nls(tl ~ vbT(age, Linf ,K, t0),  
 data=crm,  
 start=vbStarts(tl ~ age,   
 data = crm,   
 type="typical"))  
  
#Extract summary information and send to a new object  
sumCroaker <- summary(fitCroaker, correlation = TRUE)  
sumCroaker

##   
## Formula: tl ~ vbT(age, Linf, K, t0)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Linf 366.4155 16.7550 21.869 < 2e-16 \*\*\*  
## K 0.3148 0.1076 2.924 0.00419 \*\*   
## t0 -1.7143 1.0486 -1.635 0.10493   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 33.42 on 111 degrees of freedom  
##   
## Correlation of Parameter Estimates:  
## Linf K   
## K -0.95   
## t0 -0.87 0.97  
##   
## Number of iterations to convergence: 4   
## Achieved convergence tolerance: 5.376e-06

#Extract coefficients and send to a new object  
coefCroaker <- coef(fitCroaker)  
coefCroaker

## Linf K t0   
## 366.4155203 0.3147927 -1.7142837

#Calculate 95% confidence intervals for coefficients  
confinCroaker <- confint2(fitCroaker)  
confinCroaker

## 2.5 % 97.5 %  
## Linf 333.2143237 399.6167168  
## K 0.1014851 0.5281003  
## t0 -3.7922328 0.3636653

#2. Adjust algorithm settings to increase iterations####  
#Common error during parameter estimation, number of iterations exceeded maximum of 50  
  
#Occurs if algorithm has trouble finding coefficients for the model  
#Try increasing the algorithms maximum number of iterations  
fitCroaker <- nls(tl ~ vbT(age, Linf ,K, t0),  
 data=crm,  
 start=vbStarts(tl ~ age,   
 data = crm,   
 type="typical"),  
 control=list(maxiter=1000))  
  
summary(fitCroaker, correlation = TRUE)

##   
## Formula: tl ~ vbT(age, Linf, K, t0)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Linf 366.4155 16.7550 21.869 < 2e-16 \*\*\*  
## K 0.3148 0.1076 2.924 0.00419 \*\*   
## t0 -1.7143 1.0486 -1.635 0.10493   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 33.42 on 111 degrees of freedom  
##   
## Correlation of Parameter Estimates:  
## Linf K   
## K -0.95   
## t0 -0.87 0.97  
##   
## Number of iterations to convergence: 4   
## Achieved convergence tolerance: 5.376e-06

#3. Use a different algorithm that allows the user to specify the upper and lower bounds of parameters####  
#Change algorithm to "port" and manually set limits for lower and upper parameters  
#Use historical data to guide minimum and maximum length  
#The max length in the Croaker2 data set is 462, can also seek guidance from the literature  
#If Linf parameter defaults to upper limit, parameter estimates may not be trustworthy  
fitCroaker\_2<-nls(tl~vbT(age,Linf,K,t0),  
 data=crm,  
 start=vbStarts(tl ~ age,   
 data = crm,   
 type="typical"),  
 algorithm = "port",  
 lower = list(Linf=300, k=0.01, t0=-4),  
 upper = list(Linf=600, k=2.00, t0=4))  
  
summary(fitCroaker\_2, correlation = TRUE)

##   
## Formula: tl ~ vbT(age, Linf, K, t0)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Linf 366.4147 16.7545 21.870 < 2e-16 \*\*\*  
## K 0.3148 0.1076 2.924 0.00419 \*\*   
## t0 -1.7142 1.0486 -1.635 0.10493   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 33.42 on 111 degrees of freedom  
##   
## Correlation of Parameter Estimates:  
## Linf K   
## K -0.95   
## t0 -0.87 0.97  
##   
## Algorithm "port", convergence message: relative convergence (4)

####################################################  
###Bayesian method if nls function does not work###  
####################################################  
  
###brms package method; requires Rtools4.2 software download###  
  
#4. Apply Bayesian techniques with non-informative priors####  
###################################  
#brms w/ non-informative priors####  
###################################  
  
library(brms)

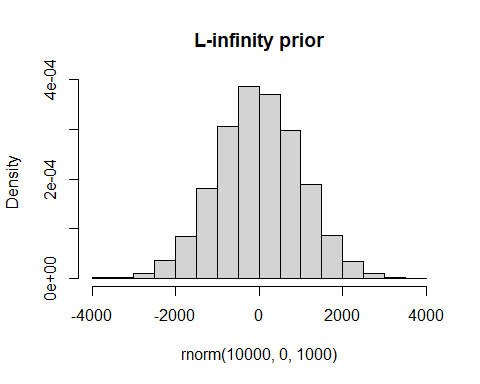
## Loading required package: Rcpp

## Loading 'brms' package (version 2.20.4). Useful instructions  
## can be found by typing help('brms'). A more detailed introduction  
## to the package is available through vignette('brms\_overview').

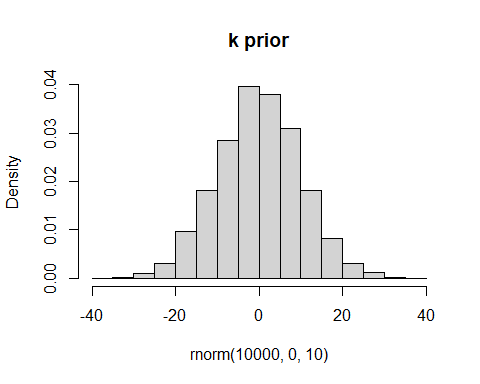
##   
## Attaching package: 'brms'

## The following object is masked from 'package:stats':  
##   
## ar

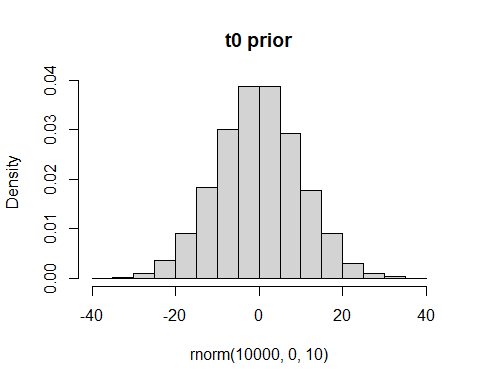
#Set formula  
formula <- brmsformula(tl ~ Linf \* (1 - exp(-K \* (age - t0))),  
 Linf ~ 1, K ~ 1, t0 ~ 1, nl=TRUE)  
  
#Start with non-informative priors to let the data speak for itself  
#If coefficients aren't realistic, proceed with informative priors  
#Added lb=0 to each prior to specify lower bound of 0  
prior1 <- prior(normal(0,1000), nlpar = "Linf", lb=0) +  
 prior(normal(0,10), nlpar = "K", lb=0) +  
 prior(normal( 0,10), nlpar = "t0")  
  
#View the shape of the prior distribution  
hist(rnorm(10000,0,1000), main=c("L-infinity prior"), prob=TRUE)



hist(rnorm(10000,0,10), main=c("k prior"), prob=TRUE)



hist(rnorm(10000,0,10), main=c("t0 prior"), prob=TRUE)



#Set initial values  
#Adjust Linf starting values to something near the maximum length in the data  
inits <- function() list(  
 Linf = runif(1, 200, 800),  
 K = runif(1, 0.05, 3.00),  
 t0 = rnorm(1, 0, 0.5)  
)  
  
fit1 <- brm(formula,  
 family=gaussian(),  
 data = crm,  
 prior = prior1,  
 init = inits,  
 chains=3,  
 cores=3,  
 iter=4000,  
 warmup = 2000,  
 control = list(adapt\_delta = 0.80,max\_treedepth = 15))

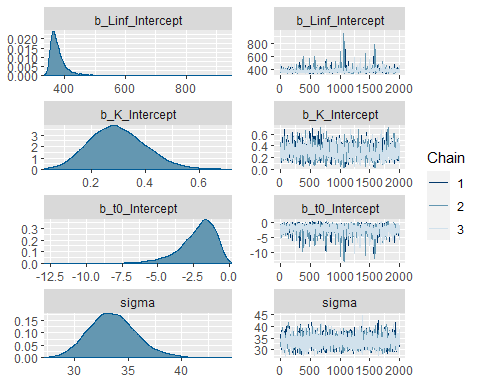
## Compiling Stan program...

## Start sampling

## Warning: There were 2 divergent transitions after warmup. See  
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup  
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems

plot(fit1)



summary(fit1)

## Warning: There were 2 divergent transitions after warmup. Increasing  
## adapt\_delta above 0.8 may help. See  
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: gaussian   
## Links: mu = identity; sigma = identity   
## Formula: tl ~ Linf \* (1 - exp(-K \* (age - t0)))   
## Linf ~ 1  
## K ~ 1  
## t0 ~ 1  
## Data: crm (Number of observations: 114)   
## Draws: 3 chains, each with iter = 4000; warmup = 2000; thin = 1;  
## total post-warmup draws = 6000  
##   
## Population-Level Effects:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## Linf\_Intercept 378.04 34.83 346.33 456.58 1.00 1063 924  
## K\_Intercept 0.30 0.11 0.11 0.53 1.00 980 920  
## t0\_Intercept -2.28 1.45 -5.86 -0.41 1.00 991 933  
##   
## Family Specific Parameters:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## sigma 33.60 2.24 29.52 38.43 1.00 2239 2529  
##   
## Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
## and Tail\_ESS are effective sample size measures, and Rhat is the potential  
## scale reduction factor on split chains (at convergence, Rhat = 1).

#Extract table of effects  
coef <- fixef(fit1)  
#Print coef  
coef

## Estimate Est.Error Q2.5 Q97.5  
## Linf\_Intercept 378.0367017 34.8292643 346.3254990 456.5761310  
## K\_Intercept 0.2976012 0.1065889 0.1089291 0.5259116  
## t0\_Intercept -2.2806107 1.4531116 -5.8550436 -0.4139328

#5. Apply Bayesian techniques with informative priors  
  
###############################  
#brms w/ informative priors####  
###############################  
  
#Set formula  
formula <- bf(tl ~ Linf \* (1 - exp(-K \* (age - t0))),  
 Linf ~ 1, K ~ 1, t0 ~ 1, nl=TRUE)  
  
#Use informative priors by calculating the average and sd of all  
#Linf, K, and t0 from your state and replace the means and sd below.  
#Or pull values from FishBase and calculate the mean and SD  
#Added lb=0 to each prior to specify lower bound of 0  
Summarize(c(182, 310, 390,419, 645))

## n mean sd min Q1 median Q3 max   
## 5.0000 389.2000 169.9462 182.0000 310.0000 390.0000 419.0000 645.0000

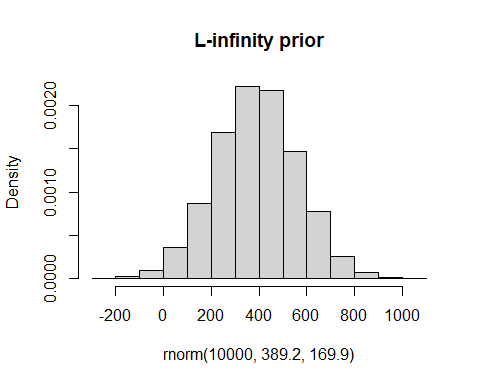
Summarize(c(1.452, 0.360, 0.350, 0.270, 0.200))

## n mean sd min Q1 median Q3 max   
## 5.0000000 0.5264000 0.5214929 0.2000000 0.2700000 0.3500000 0.3600000 1.4520000

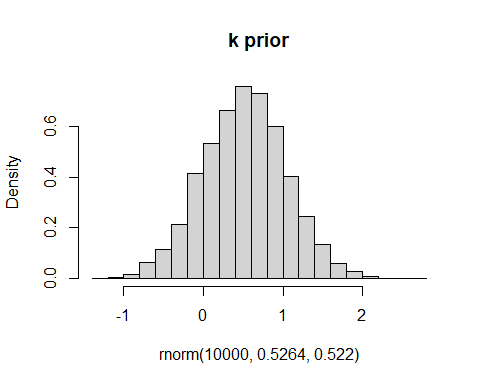
Summarize(c(0.18, -3.26))

## n mean sd min Q1 median Q3 max   
## 2.000000 -1.540000 2.432447 -3.260000 -2.400000 -1.540000 -0.680000 0.180000

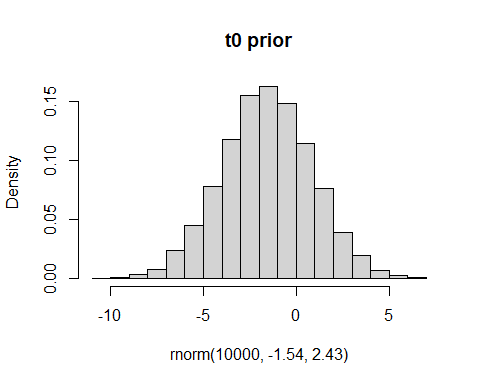
prior2 <- prior(normal(389.2, 169.9), nlpar = "Linf",lb=0) +  
 prior(normal(0.5264, 0.522), nlpar = "K",lb=0) +  
 prior(normal(-1.54, 2.43), nlpar = "t0")  
  
#View the shape of the prior distribution  
hist(rnorm(10000, 389.2, 169.9), main=c("L-infinity prior"), prob=TRUE)



hist(rnorm(10000, 0.5264, 0.522), main=c("k prior"), prob=TRUE)



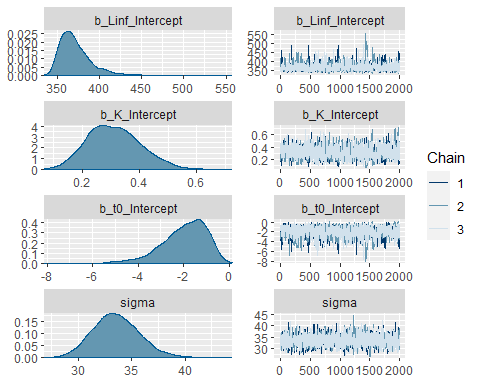
hist(rnorm(10000, -1.54, 2.43), main=c("t0 prior"), prob=TRUE)



#Set initial values  
#Adjust Linf starting values to something near the maximum length in the data  
inits <- function() list(  
 Linf = runif(1, 200, 800),  
 K = runif(1, 0.05, 3.00),  
 t0 = rnorm(1, 0, 0.5)  
)  
  
fit2 <- brm(formula,  
 family=gaussian(),  
 data = crm,  
 prior = prior2,  
 init = inits,  
 chains=3,  
 cores=3,  
 iter=4000,  
 warmup = 2000,  
 control = list(adapt\_delta = 0.80,max\_treedepth = 15))

## Compiling Stan program...  
## Start sampling

plot(fit2)



summary(fit2)

## Family: gaussian   
## Links: mu = identity; sigma = identity   
## Formula: tl ~ Linf \* (1 - exp(-K \* (age - t0)))   
## Linf ~ 1  
## K ~ 1  
## t0 ~ 1  
## Data: crm (Number of observations: 114)   
## Draws: 3 chains, each with iter = 4000; warmup = 2000; thin = 1;  
## total post-warmup draws = 6000  
##   
## Population-Level Effects:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## Linf\_Intercept 372.16 20.69 346.37 423.77 1.00 1041 792  
## K\_Intercept 0.31 0.09 0.15 0.51 1.00 976 827  
## t0\_Intercept -2.00 1.05 -4.54 -0.47 1.00 1071 1013  
##   
## Family Specific Parameters:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## sigma 33.57 2.21 29.50 38.26 1.00 1891 2325  
##   
## Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
## and Tail\_ESS are effective sample size measures, and Rhat is the potential  
## scale reduction factor on split chains (at convergence, Rhat = 1).

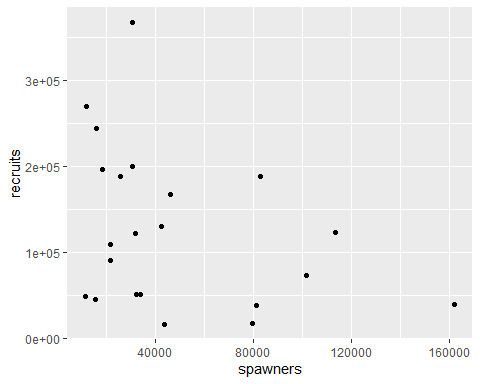
#Extract table of effects  
coef = fixef(fit2)  
#Print coef  
coef

## Estimate Est.Error Q2.5 Q97.5  
## Linf\_Intercept 372.1555933 20.69195418 346.3724530 423.7748048  
## K\_Intercept 0.3116542 0.09388463 0.1463648 0.5063678  
## t0\_Intercept -1.9968823 1.04846383 -4.5375287 -0.4721905

#Stock-Recruitment#######  
#1. Select data  
#2. Specify the growth model  
#3. Specify starting values and estimate parameters  
#4. Summarize results  
  
#Ricker Model####  
#Stock and recruitment data for Klamath River  
#Chinook salmon, 1979-2000  
ChinookKR <- FSAdata::ChinookKR %>%   
 na.omit(ChinookKR) %>%   
 mutate(logrec = log(recruits),  
 logstock = log(spawners))  
  
  
headtail(ChinookKR)

## brood.year spawners recruits logrec logstock  
## 1 1979 30637 200698 12.20956 10.329964  
## 2 1980 21484 109430 11.60304 9.975064  
## 3 1981 33857 50968 10.83895 10.429901  
## 20 1998 42488 130283 11.77746 10.656977  
## 21 1999 18457 196197 12.18687 9.823199  
## 22 2000 82728 188537 12.14705 11.323313

#Plot spawners (stock) vs recruits  
ggplot(ChinookKR, aes(x = spawners, y = recruits)) +  
 geom\_point()



#Ricker function  
#E[R|S] = alpha \* S \* exp(-beta \* S)  
  
#We will use the nls function to fit this non-linear model  
#Requires starting values  
svR <- srStarts(recruits ~ spawners,   
 data = ChinookKR,   
 type = "Ricker")  
svR

## $a  
## [1] 8.527721  
##   
## $b  
## [1] 2.517053e-05

#Obtain Ricker function from FSA  
rckr <- srFuns("Ricker")  
  
#Fit Ricker function to stock and recruitment data  
srR <- nls(logrec ~ log(rckr(spawners,a,b)),   
 data = ChinookKR,   
 start = svR)  
  
summary(srR)

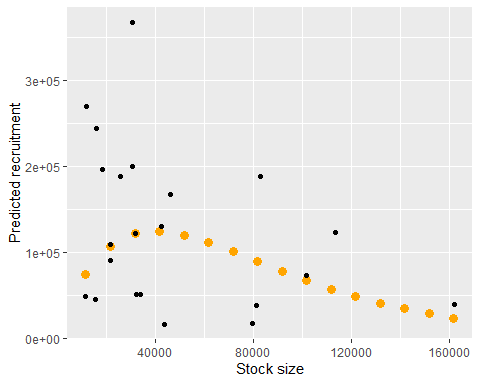
##   
## Formula: logrec ~ log(rckr(spawners, a, b))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## a 8.528e+00 2.625e+00 3.248 0.00403 \*\*   
## b 2.517e-05 5.018e-06 5.016 6.63e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9005 on 20 degrees of freedom  
##   
## Number of iterations to convergence: 0   
## Achieved convergence tolerance: 6.721e-09

#Coefficients with 95% Confidence Intervals  
cbind(estimates = coef(srR), confint(srR))

## Waiting for profiling to be done...

## estimates 2.5% 97.5%  
## a 8.527721e+00 4.491396e+00 16.208865109  
## b 2.517053e-05 1.470305e-05 0.000035638

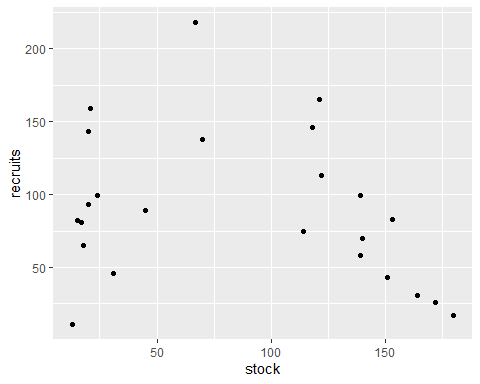
#Visualize the model fit  
#Range of spawning stock  
x <- seq(from = min(ChinookKR$spawners),   
 to = max(ChinookKR$spawners),   
 by= 10000)  
  
#Predict recruitment from model fit above  
pR<- rckr(x, a=coef(srR))  
#combine in a data frame  
CombSR <- data.frame(x = x, pR = pR)  
  
#Plot predictions with raw data  
ggplot() +  
 geom\_point(aes(x = CombSR$x, y = CombSR$pR),   
 color = "orange", size = 3) +  
 geom\_point(aes(x = ChinookKR$spawners,   
 y = ChinookKR$recruits)) +  
 xlab("Stock size") +   
 ylab("Predicted recruitment")



#Beverton-Holt####  
CodNorwegian <- FSA::CodNorwegian %>%   
 mutate(logrec = log(recruits),  
 logstock = log(stock))  
  
headtail(CodNorwegian)

## year recruits stock logrec logstock  
## 1 1937 146 118 4.983607 4.770685  
## 2 1938 31 164 3.433987 5.099866  
## 3 1939 17 180 2.833213 5.192957  
## 22 1958 143 20 4.962845 2.995732  
## 23 1959 65 18 4.174387 2.890372  
## 24 1960 11 13 2.397895 2.564949

#Plot spawners (stock) vs recruits  
ggplot(CodNorwegian, aes(x = stock, y = recruits)) +  
 geom\_point()



#Beverton-Holt function  
#E[R|S] = alpha \* S /(1 + beta \* S)  
  
#We will use the nls function to fit this non-linear model  
#Requires starting values  
bhs <- srStarts(recruits ~ stock,   
 data = CodNorwegian,   
 type = "BevertonHolt")  
bhs

## $a  
## [1] 6.040326  
##   
## $b  
## [1] 0.08789475

#Obtain Beverton-Holt function from FSA  
  
bh1 <- srFuns("BevertonHolt")  
  
srBH <- nls(logrec ~ log(bh1(stock,a,b)),   
 data = CodNorwegian,   
 start = bhs)  
  
summary(srBH)

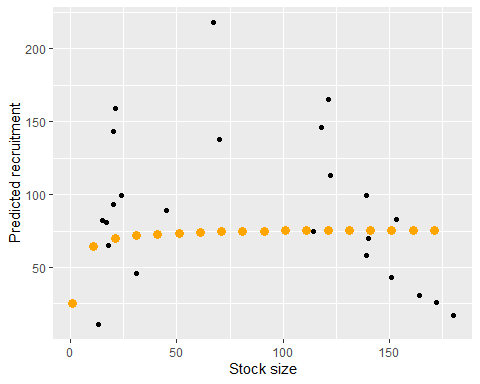
##   
## Formula: logrec ~ log(bh1(stock, a, b))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)  
## a 37.3341 132.0144 0.283 0.78  
## b 0.4876 1.8099 0.269 0.79  
##   
## Residual standard error: 0.7486 on 22 degrees of freedom  
##   
## Number of iterations to convergence: 6   
## Achieved convergence tolerance: 6.77e-06

#Coefficients with 95% Confidence Intervals  
cbind(estimates = coef(srR), confint(srR))

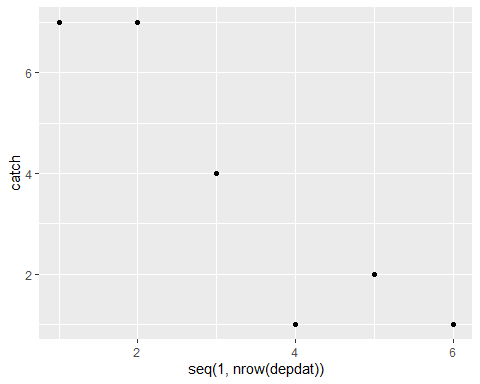
## Waiting for profiling to be done...

## estimates 2.5% 97.5%  
## a 8.527721e+00 4.491396e+00 16.208865109  
## b 2.517053e-05 1.470305e-05 0.000035638

#Visualize the model fit  
#Range of spawning stock  
x <- seq(from = 1,   
 to = max(CodNorwegian$stock),   
 by= 10)  
  
#Predict recruitment from model fit above  
pR<- bh1(x, a=coef(srBH))  
#combine in a data frame  
CombSR <- data.frame(x = x, pR = pR)  
  
#Plot predictions with raw data  
ggplot() +  
 geom\_point(aes(x = CombSR$x, y = CombSR$pR),   
 color = "orange", size = 3) +  
 geom\_point(aes(x = CodNorwegian$stock,   
 y = CodNorwegian$recruits)) +  
 xlab("Stock size") +   
 ylab("Predicted recruitment")



#Exercise 4  
  
  
#Population Estimates from Depletion Data#######  
#1. Leslie Method  
#2. k-pass removal  
  
  
#Leslie Method  
#C\_i/f\_i = qN0 - q(K\_i-1)  
  
#C\_i=catch for sample i  
#f\_i=fishing effort for sample i  
#q=catchability coefficient  
#N0=initial abundance  
#k\_i-1=cumulative catch prior to sample i  
  
#Essentially a linear regression problem  
  
#Build a data frame with capture data  
depdat <- data.frame(catch = c(7,7,4,1,2,1),  
 effort = c(10,10,10,10,6,10)) %>%  
 mutate(cpe = catch/effort, K = pcumsum(catch))  
  
#Plot catch data  
ggplot(depdat, aes(x = seq(1,nrow(depdat)), y = catch)) +  
 geom\_point()



lm2 <- lm(cpe ~ K, data=depdat)  
summary(lm2)

##   
## Call:  
## lm(formula = cpe ~ K, data = depdat)  
##   
## Residuals:  
## 1 2 3 4 5 6   
## -0.08643 0.12492 0.03627 -0.14296 0.12057 -0.05238   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.786432 0.103558 7.594 0.00161 \*\*  
## K -0.030193 0.006851 -4.407 0.01163 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1246 on 4 degrees of freedom  
## Multiple R-squared: 0.8292, Adjusted R-squared: 0.7865   
## F-statistic: 19.42 on 1 and 4 DF, p-value: 0.01163

#extract coefficients  
(cf1 <- coef(lm2))

## (Intercept) K   
## 0.78643157 -0.03019312

#Calculate N0  
#C\_i/f\_i = qN0 - q(K\_i-1)  
  
#recall intercept = qN0  
#q is also the slope  
#N0 = qNO/-q or intercept/-slope  
(q\_hat <- -cf1[["K"]])

## [1] 0.03019312

(N0\_hat <- cf1[["(Intercept)"]] / q\_hat)

## [1] 26.04672

#k-pass removal estimates require equal catchability and equal effort:  
#removal(catch, method)  
#Carle Strub (default) weighted k-pass estimator  
#Burnham is a likelihood based estimator used in Microfish software (Van Deventer 1989)  
catch <- c(187, 77, 35, 5)  
pr1 <- removal(catch, method = "CarleStrub")  
  
#Extract estimates with 95% confidence intervals  
cbind(summary(pr1), confint(pr1))

## Estimate Std. Error 95% LCI 95% UCI  
## No 310.0000000 3.1830289 303.7613780 316.2386220  
## p 0.6204082 0.0272384 0.5670219 0.6737944

#The previous example can be applied to a single site.  
#Multiple sites can be estimated by using the streamlined code below  
  
#Data from three sites  
catch2 <- data.frame(sta = c("SC10","SC11","SC12"),  
 p1 = c(19,75,20), #Pass 1  
 p2 = c(14,19,11), #Pass 2  
 p3 = c(9,5,3) ) #Pass 3  
catch2

## sta p1 p2 p3  
## 1 SC10 19 14 9  
## 2 SC11 75 19 5  
## 3 SC12 20 11 3

#Use the apply function to generate population estimate at all sites  
#apply(array or matrix, Margin, function,   
# and just.est)  
#MARGIN = 1 indicates function is applied over rows  
#MARGIN = 2 indicates function is applied over columns  
#The data frame has one site for each row  
res <- apply(catch2[,-1],   
 MARGIN = 1,   
 FUN = removal,   
 just.est=TRUE)  
  
(res <- data.frame(sta=catch2$sta, t(res) ) )

## sta No No.se No.LCI No.UCI p p.se p.LCI  
## 1 SC10 55 11.727984 32.01357 77.98643 0.3716814 0.12613965 0.1244522  
## 2 SC11 100 1.470384 97.11810 102.88190 0.7557252 0.04549002 0.6665664  
## 3 SC12 36 2.507203 31.08597 40.91403 0.5964912 0.10295277 0.3947075  
## p.UCI  
## 1 0.6189106  
## 2 0.8448840  
## 3 0.7982749

#Size structure#######  
#1. Length Frequency  
#2. PSD  
  
#Load Largemouth Bass data from the FSAdata package  
LMBassBL <- FSAdata::LMBassBL   
  
#Add a length category column  
LMBassBL$lcat10 <- lencat(LMBassBL$tl , w=10)  
  
#Check data to confirm  
headtail(LMBassBL)

## tl lcat10  
## 1 87 80  
## 2 90 90  
## 3 99 90  
## 445 499 490  
## 446 504 500  
## 447 510 510

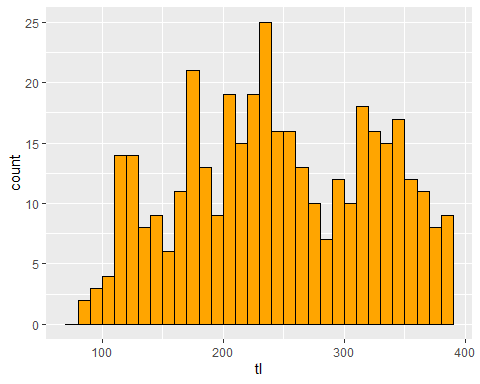
#Create a length frequency table  
#Using the xtabs function, requires the column name   
#with length category and the dataset  
LMBFreq10 <- xtabs(~lcat10, data = LMBassBL)  
#print frequency table  
LMBFreq10

## lcat10  
## 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270   
## 1 4 4 11 16 9 8 6 11 20 13 10 17 15 20 25 18 14 13 12   
## 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470   
## 5 12 11 17 15 16 18 11 11 9 10 13 11 7 9 5 7 5 3 1   
## 490 500 510   
## 2 1 1

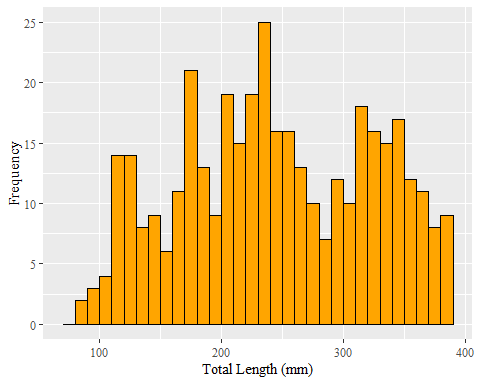
#Percentage of fish in each interval using the   
#prop.table() function  
LMBPer10 <- round( prop.table(LMBFreq10) \* 100 , 1)  
#print proportion table  
LMBPer10

## lcat10  
## 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270   
## 0.2 0.9 0.9 2.5 3.6 2.0 1.8 1.3 2.5 4.5 2.9 2.2 3.8 3.4 4.5 5.6 4.0 3.1 2.9 2.7   
## 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470   
## 1.1 2.7 2.5 3.8 3.4 3.6 4.0 2.5 2.5 2.0 2.2 2.9 2.5 1.6 2.0 1.1 1.6 1.1 0.7 0.2   
## 490 500 510   
## 0.4 0.2 0.2

#A histogram can be useful to visualize   
#length frequencies  
ggplot(LMBassBL, aes(x = tl)) +   
 geom\_histogram(breaks = seq(from = 70 , to = 390, by = 10),  
 fill = "orange", color = "black")



#Add "Total Length (mm)" to the x-axis label  
#Add "Frequency" to the y-axis label  
#Modify font type  
  
ggplot(LMBassBL, aes(x = tl)) +   
 geom\_histogram(breaks = seq(from = 70 , to = 390,   
 by = 10),  
 fill = "orange", color = "black") +  
 xlab("Total Length (mm)") +  
 ylab("Frequency")+  
 theme( text = element\_text(family="serif"))



###PSD####  
#FSA contains a list of Gabelhouse (1984)   
#length categories  
psdVal("Largemouth Bass")

## substock stock quality preferred memorable trophy   
## 0 200 300 380 510 630

#Use psdVal() to return list of available species  
psdVal()

##   
## Species name must be one of following. Be careful of spelling and capitalization.

## [1] "Arctic Grayling" "Bighead Carp"   
## [3] "Bigmouth Buffalo" "Black Bullhead"   
## [5] "Black Carp" "Black Crappie"   
## [7] "Blue Catfish" "Bluegill"   
## [9] "Brook Trout" "Brook Trout (lentic)"   
## [11] "Brook Trout (lotic)" "Brown Bullhead"   
## [13] "Brown Trout (lentic)" "Brown Trout (lotic)"   
## [15] "Bull Trout" "Burbot"   
## [17] "Chain Pickerel" "Channel Catfish"   
## [19] "Chinook Salmon (landlocked)" "Common Carp"   
## [21] "Cutthroat Trout" "Flathead Catfish"   
## [23] "Freshwater Drum" "Gizzard Shad"   
## [25] "Golden Trout" "Grass Carp"   
## [27] "Green Sunfish" "Kokanee"   
## [29] "Lake Trout" "Largemouth Bass"   
## [31] "Longnose Gar" "Muskellunge"   
## [33] "Northern Pike" "Paddlefish"   
## [35] "Pallid Sturgeon" "Palmetto Bass"   
## [37] "Palmetto Bass (original)" "Pumpkinseed"   
## [39] "Rainbow Trout" "Redbreast Sunfish"   
## [41] "Redear Sunfish" "River Carpsucker"   
## [43] "Rock Bass" "Ruffe"   
## [45] "Sauger" "Saugeye"   
## [47] "Shoal Bass" "Shorthead Redhorse"   
## [49] "Silver Carp" "Smallmouth Bass"   
## [51] "Smallmouth Buffalo " "Splake"   
## [53] "Spotted Bass" "Spotted Gar"   
## [55] "Spotted Sunfish" "Striped Bass (hybrid)"   
## [57] "Striped Bass (landlocked)" "Striped Bass X White Bass"   
## [59] "Suwannee Bass" "Utah Chub"   
## [61] "Walleye" "Warmouth"   
## [63] "White Bass" "White Catfish"   
## [65] "White Crappie" "White Perch"   
## [67] "White Sucker" "Yellow Bass"   
## [69] "Yellow Bullhead" "Yellow Perch"

#The units returned are in mm but you can request   
#cm, mm, or in  
psdVal("Largemouth Bass", units = "cm")

## substock stock quality preferred memorable trophy   
## 0 20 30 38 51 63

psdVal("Largemouth Bass", units = "mm")

## substock stock quality preferred memorable trophy   
## 0 200 300 380 510 630

psdVal("Largemouth Bass", units = "in")

## substock stock quality preferred memorable trophy   
## 0 8 12 15 20 25

#Assign length category to each fish to the Largemouth Bass data frame  
  
#Load the LMBassBL data set  
LMBassBL <- FSAdata::LMBassBL   
help(LMBassBL)  
  
#First we need to pull out the Largemouth Bass   
#length categories  
lmb.cuts <- psdVal("Largemouth Bass", units = "mm")  
  
#Filter based on tl greater then stock length and add the length category column  
LMB\_SS <- LMBassBL %>%  
 dplyr::filter( tl >= lmb.cuts["stock"]) %>%  
 dplyr::mutate( gcat = lencat(tl, breaks = lmb.cuts,  
 use.names = TRUE))  
  
headtail(LMB\_SS)

## tl gcat  
## 1 200 stock  
## 2 204 stock  
## 3 204 stock  
## 332 499 preferred  
## 333 504 preferred  
## 334 510 memorable

#Calculate a frequency table across size groups  
xtabs(~gcat, data = LMB\_SS)

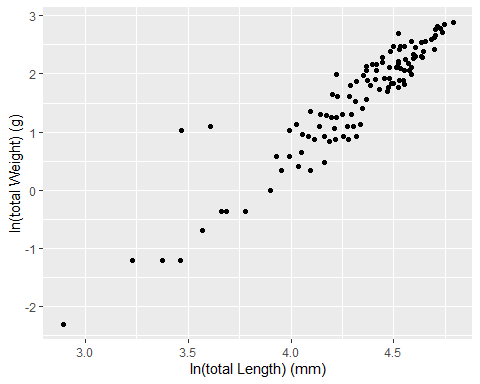
## gcat  
## substock stock quality preferred memorable trophy   
## 0 151 108 74 1 0

#Calculate all PSD-X values and 95% Confidence Intervals. Requires length column, data set,   
#species, and what type. Types available are incremental (e.g., Stock to Quality) or  
#traditional (e.g., Quality, Preferred, Memorable)  
psdCalc(~tl, data=LMB\_SS,   
 species = "Largemouth Bass",   
 what = "traditional")

## Warning: Some category sample size <20, some CI coverage may be  
## lower than 95%.

## Estimate 95% LCI 95% UCI  
## PSD-Q 55 47 62  
## PSD-P 22 16 29

#Weight-length relationships#######  
#Load Chinook data from the FSA package  
ChinookArg <- FSA::ChinookArg  
  
#Plot TL vs TW on the natural log scale  
ggplot(ChinookArg, aes(x = log(tl), y = log(w))) +  
 geom\_point() +  
 xlab("ln(total Length) (mm)") +   
 ylab("ln(total Weight) (g)")



#Estimate parameters of the weight-length model   
#using lm()  
lm1 <- lm(log(w) ~ log(tl), data=ChinookArg)  
  
#Extract summary information and send to a new object  
sumlw <- summary(lm1)  
sumlw

##   
## Call:  
## lm(formula = log(w) ~ log(tl), data = ChinookArg)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.69071 -0.21434 -0.00143 0.17011 1.70527   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -9.86574 0.42048 -23.46 <2e-16 \*\*\*  
## log(tl) 2.64931 0.09707 27.29 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3516 on 110 degrees of freedom  
## Multiple R-squared: 0.8713, Adjusted R-squared: 0.8701   
## F-statistic: 744.8 on 1 and 110 DF, p-value: < 2.2e-16

#Extract coefficients and send to a new object  
coeflw <- coef(lm1)  
coeflw

## (Intercept) log(tl)   
## -9.865737 2.649314

#Extract confidence intervals for coefficients  
confinlw <- confint(lm1)  
confinlw

## 2.5 % 97.5 %  
## (Intercept) -10.699032 -9.032442  
## log(tl) 2.456936 2.841693

#Exercise 3  
  
#Condition factors#######  
#1. Fulton's Condition Factor  
#2. Weight-length residuals  
#3. Relative weight  
  
#Load Bluegill data from the FSA package  
BLG <- FSAdata::BluegillLM   
help(BluegillLM)  
#Select the data and calculate log10 of   
#length and weight  
  
BLGSub <- BLG %>%  
 mutate(logW = log10(wght), logL = log10(tl)) %>% #take log10  
 select( -c(sl, fl, sernum)) #remove excess columns  
  
headtail(BLGSub)

## tl wght logW logL  
## 1 157 92 1.963788 2.195900  
## 2 153 88 1.944483 2.184691  
## 3 149 77 1.886491 2.173186  
## 98 147 64 1.806180 2.167317  
## 99 144 64 1.806180 2.158362  
## 100 133 49 1.690196 2.123852

#Fulton's Condition Factor  
#Describes condition of individual fish  
#Metric: K = W / L^3 \* 100,000  
#English: K = W / L^3 \* 10,000  
  
BLGSub <- BLGSub %>%   
 mutate(K = wght / (tl^3) \* 100000)  
  
headtail(BLGSub)

## tl wght logW logL K  
## 1 157 92 1.963788 2.195900 2.377327  
## 2 153 88 1.944483 2.184691 2.457018  
## 3 149 77 1.886491 2.173186 2.327726  
## 98 147 64 1.806180 2.167317 2.014782  
## 99 144 64 1.806180 2.158362 2.143347  
## 100 133 49 1.690196 2.123852 2.082769

#Weight-Length Residuals  
#Estimate coefficients of weight-length model  
lm1 <- lm(logW ~ logL, data = BLGSub)  
coef(lm1)

## (Intercept) logL   
## -5.524963 3.406255

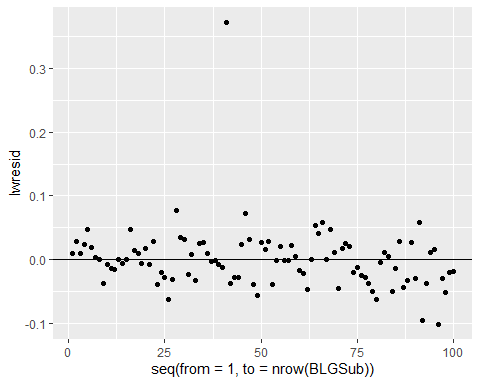
residuals(lm1)

## 1 2 3 4 5   
## 0.0089569461 0.0278298495 0.0090274171 0.0241266996 0.0471400049   
## 6 7 8 9 10   
## 0.0183774990 0.0037777900 -0.0005883718 -0.0382379191 -0.0072120130   
## 11 12 13 14 15   
## -0.0149355830 -0.0153116287 -0.0004355786 -0.0062138759 -0.0005601876   
## 16 17 18 19 20   
## 0.0465523571 0.0133120570 0.0088641694 -0.0070021764 0.0178456285   
## 21 22 23 24 25   
## -0.0080562020 0.0279228169 -0.0388854602 -0.0207908051 -0.0291424595   
## 26 27 28 29 30   
## -0.0633178943 -0.0311505844 0.0769048322 0.0350111829 0.0309456929   
## 31 32 33 34 35   
## -0.0240355862 0.0073918979 -0.0334084823 0.0243955185 0.0272411818   
## 36 37 38 39 40   
## 0.0093439118 -0.0038623573 -0.0013206876 -0.0082315883 -0.0121957661   
## 41 42 43 44 45   
## 0.3729566943 -0.0377992143 -0.0283587734 -0.0288848587 0.0232619911   
## 46 47 48 49 50   
## 0.0717663074 0.0308218247 -0.0389461847 -0.0574178625 0.0262292587   
## 51 52 53 54 55   
## 0.0149940693 0.0284371668 -0.0390686815 -0.0008949461 0.0198646099   
## 56 57 58 59 60   
## -0.0024020447 -0.0023262993 0.0221814999 0.0054112897 -0.0181548160   
## 61 62 63 64 65   
## -0.0222152471 -0.0473854694 0.0004932764 0.0531981476 0.0400979679   
## 66 67 68 69 70   
## 0.0580063434 -0.0008223072 0.0465930554 0.0103549052 -0.0461757988   
## 71 72 73 74 75   
## 0.0170882879 0.0248994122 0.0211269487 -0.0207897749 -0.0129279035   
## 76 77 78 79 80   
## -0.0249633119 -0.0281284037 -0.0382379191 -0.0506712325 -0.0629352961   
## 81 82 83 84 85   
## -0.0050638112 0.0109434745 0.0039385048 -0.0507009921 -0.0149048059   
## 86 87 88 89 90   
## 0.0285100136 -0.0448155927 -0.0325267702 0.0272411818 -0.0300259499   
## 91 92 93 94 95   
## 0.0584003902 -0.0966331399 -0.0371080460 0.0115978706 0.0162535273   
## 96 97 98 99 100   
## -0.1021259199 -0.0300259499 -0.0512922506 -0.0207897749 -0.0192209166

#Calculate weight residuals  
#Weight residuals are the difference between the observed log10 weight and predicted log10 weight  
#Residuals can tell you if the fish is plumper or skinnier than average.  
  
BLGSub <- BLGSub %>%  
 mutate(lwresid = residuals(lm1))  
  
headtail(BLGSub)

## tl wght logW logL K lwresid  
## 1 157 92 1.963788 2.195900 2.377327 0.008956946  
## 2 153 88 1.944483 2.184691 2.457018 0.027829849  
## 3 149 77 1.886491 2.173186 2.327726 0.009027417  
## 98 147 64 1.806180 2.167317 2.014782 -0.051292251  
## 99 144 64 1.806180 2.158362 2.143347 -0.020789775  
## 100 133 49 1.690196 2.123852 2.082769 -0.019220917

#Plot residuals  
ggplot(BLGSub, aes(x = seq(from=1,to=nrow(BLGSub)), y = lwresid)) +  
 geom\_point() +  
 geom\_hline(yintercept=0)



#Relative Weight  
#Wr = W / Ws \* 100 (where Ws is the standard weight given length)  
#Ws = 10 ^ (alpha + beta \* log10(TL))  
  
#FSA contains a list of standard weights from a variety of sources  
#Return list of available species  
wsVal()

##   
## Species name must be one of following. Be careful of spelling and capitalization.

## [1] "Aegean Chub" "African Sharptooth Catfish"   
## [3] "Alabama Bass" "Ankara Nase"   
## [5] "Arctic Grayling" "Bighead Carp"   
## [7] "Bigmouth Buffalo" "Bigmouth Sleepers (all)"   
## [9] "Bigmouth Sleepers (lotic)" "Black Bullhead"   
## [11] "Black Crappie" "Blacktail Redhorse"   
## [13] "Blue Catfish" "Blue Sucker"   
## [15] "Bluegill" "Bridgelip Sucker"   
## [17] "Brook Chub" "Brook Trout"   
## [19] "Brook Trout (appalachia)" "Brown Bullhead"   
## [21] "Brown Trout (lentic)" "Brown Trout (lotic)"   
## [23] "Bull Trout" "Burbot"   
## [25] "Cavedano Chub" "Chain Pickerel"   
## [27] "Channel Catfish" "Chinook Salmon"   
## [29] "Cisco" "Common Carp"   
## [31] "Cutthroat Trout (lentic)" "Cutthroat Trout (lotic)"   
## [33] "European Chub" "European Perch"   
## [35] "Flannelmouth Sucker" "Flathead Catfish"   
## [37] "Fourbarbel Scraper" "Freshwater Drum"   
## [39] "Gizzard Shad" "Golden Shiner"   
## [41] "Golden Trout" "Green Sunfish"   
## [43] "Horse Barbel" "Humpback Chub"   
## [45] "Kokanee" "Lake Herring"   
## [47] "Lake Trout" "Largemouth Bass"   
## [49] "Largescale Sucker" "Longnose Gar"   
## [51] "Marble Trout" "Mountain Mullet"   
## [53] "Mountain Whitefish" "Muskellunge (female)"   
## [55] "Muskellunge (male)" "Muskellunge (overall)"   
## [57] "Nile Tilapia" "Nipple-Lip Scraper"   
## [59] "Northern Pike" "Northern Pikeminnow"   
## [61] "Northern Squawfish" "Paddlefish (female)"   
## [63] "Paddlefish (male)" "Paddlefish (overall)"   
## [65] "Palmetto Bass" "Pejerrey"   
## [67] "Pumpkinseed" "Pursak Chub"   
## [69] "Rainbow Trout (lentic)" "Rainbow Trout (lotic)"   
## [71] "Razorback Sucker" "Redbreast Sunfish"   
## [73] "Redear Sunfish" "Riffle Dace"   
## [75] "River Carpsucker" "River Goby"   
## [77] "Rock Bass" "Roundtail Chub"   
## [79] "Ruffe" "Sardine"   
## [81] "Sauger" "Saugeye"   
## [83] "Shoal Bass" "Shorthead Redhorse"   
## [85] "Shovelnose Sturgeon" "Silver Carp"   
## [87] "Smallmouth Bass" "Smallmouth Buffalo"   
## [89] "South European Roach" "Spotted Bass"   
## [91] "Spotted Bass (alabama subspecies)" "Spotted Gar"   
## [93] "Spotted Sunfish" "Striped Bass"   
## [95] "Striped Bass (hybrid)" "Striped Bass X White Bass"   
## [97] "Suwannee Bass" "Tiger Muskellunge"   
## [99] "Utah Chub" "Walleye"   
## [101] "Walleye (30-149 mm)" "Warmouth"   
## [103] "White Bass" "White Catfish"   
## [105] "White Crappie" "White Perch"   
## [107] "White Sturgeon" "White Sucker"   
## [109] "Yellow Bass" "Yellow Bullhead"   
## [111] "Yellow Perch"

#View the table of coefficients with references  
View(WSlit)  
  
#Return standard weight coefficients for one species  
wsVal("Bluegill", units = "metric")

## species units type ref measure method min.TL int slope source  
## 26 Bluegill metric linear 75 TL Other 80 -5.374 3.316 Hillman (1982)

#Return a simplified object for calculation  
wsBlg <- wsVal("Bluegill", units="metric",   
 simplify = TRUE)  
  
#How to reference the intercept and slope  
wsBlg

## species min.TL int slope  
## 26 Bluegill 80 -5.374 3.316

wsBlg$int

## [1] -5.374

wsBlg$slope

## [1] 3.316

#Add Ws and Wr column  
BLGSub <- BLGSub %>%  
 mutate(Ws = 10 ^ (wsBlg$int + wsBlg$slope \* logL ),  
 Wr = wght/Ws \* 100)  
  
headtail(BLGSub)

## tl wght logW logL K lwresid Ws Wr  
## 1 157 92 1.963788 2.195900 2.377327 0.008956946 80.83571 113.81109  
## 2 153 88 1.944483 2.184691 2.457018 0.027829849 74.20562 118.58940  
## 3 149 77 1.886491 2.173186 2.327726 0.009027417 67.96505 113.29352  
## 98 147 64 1.806180 2.167317 2.014782 -0.051292251 64.98667 98.48174  
## 99 144 64 1.806180 2.158362 2.143347 -0.020789775 60.69180 105.45082  
## 100 133 49 1.690196 2.123852 2.082769 -0.019220917 46.63286 105.07612

#Calculate relative weight summary statistics  
Summarize(BLGSub$Wr,digits=2)

## n mean sd min Q1 median Q3 max   
## 100.00 112.50 17.44 80.17 105.28 111.12 116.89 261.90