**Introduction**

library(FSAdata) # for data

library(FSA) # for vbFuns(), vbStarts(), confint.bootCase()

library(car) # for Boot()

library(dplyr) # for filter(), mutate()

library(ggplot2)

It was demonstrated how to make a plot that illustrated the fit of a von Bertalanffy growth function (VBGF) to data. In this post, I will demonstrate how to show the VBGF fits for two or more groups (e.g., sexes, locations, years). Here I will again use the lengths and ages of Lake Erie Walleye (*Sander vitreus*) captured during October-November, 2003-2014. These data are available in my [FSAdata package](https://github.com/droglenc/FSAdata) and formed many of the examples in Chapter 12 of the [**Age and Growth of Fishes: Principles and Techniques book**](https://derekogle.com/AGF/). My primary interest is in the tl (total length in mm), age, and sex variables ([see here for more details](https://derekogle.com/fishR/data/data-html/WalleyeErie2.html)). I will focus initially on Walleye from location “1” captured in 2014 (as an example).

data(WalleyeErie2)

w14T <- filter(WalleyeErie2,year==2014,loc==1)

The workflow below requires the predict2() and vb() functions that were created in the previous post.

vb <- vbFuns()

predict2 <- function(x) predict(x,data.frame(age=ages))

**Fitting all von Bertalanffy Growth Functions**

The key to constructing plots with multiple VBGF trajectories is to create a “long format” data.frame of predicted mean lengths-at-age with associated bootstrap confidence intervals. In this format one row corresponds to a single “group” and age with columns (variabls) that identify the “group”, age”, predicted mean length, and the lower and upper values for the confidence interval. There is likely many ways to construct such a data.frame, but a loop over the “groups” (i.e., sexes) is used below

Begin by finding the range of ages for both sexes so that the confidence polygon can be restricted to observed ages.

agesum <- group\_by(w14T,sex) %>%

summarize(minage=min(age),maxage=max(age))

agesum

## # A tibble: 2 x 3

## sex minage maxage

##

## 1 female 0 11

## 2 male 1 11

To simplify coding below, the levels and number of “groups” are saved into objects.

( sexes <- levels(w14T$sex) )

## [1] "female" "male"

( nsexes <- length(sexes) )

## [1] 2

In the loop across sexes, the VBGF will be fit for each sex and parameter estimates will be saved into cfs, confidence intervals for the parameter estimates into cis, predicted mean lengths-at-age for all ages considered in preds1, and predicted mean lengths-at-age for only observed ages in preds2. These objects are initialized with NULL prior to starting the loop.

cfs <- cis <- preds1 <- preds2 <- NULL

The code inside the loop follows the same logic for fitting the VBGF to one group.

for (i in 1:nsexes) {

## Loop notification (for peace of mind)

cat(sexes[i],"Loop\n")

## Isolate sex's data

tmp1 <- filter(w14T,sex==sexes[i])

## Fit von B to that sex

sv1 <- vbStarts(tl~age,data=tmp1)

fit1 <- nls(tl~vb(age,Linf,K,t0),data=tmp1,start=sv1)

## Extract and store parameter estimates and CIs

cfs <- rbind(cfs,coef(fit1))

boot1 <- Boot(fit1)

tmp2 <- confint(boot1)

cis <- rbind(cis,c(tmp2["Linf",],tmp2["K",],tmp2["t0",]))

## Predict mean lengths-at-age with CIs

## preds1 -> across all ages

## preds2 -> across observed ages only

ages <- seq(-1,12,0.2)

boot2 <- Boot(fit1,f=predict2)

tmp2 <- data.frame(sex=sexes[i],age=ages,

predict(fit1,data.frame(age=ages)),

confint(boot2))

preds1 <- rbind(preds1,tmp2)

tmp2 <- filter(tmp2,age>=agesum$minage[i],age<=agesum$maxage[i])

preds2 <- rbind(preds2,tmp2)

}

## female Loop

## male Loop

The cfs, cis, preds1, and preds2 objects will have poorly named rows, columns, or both after the loop. These deficiencies are corrected below.

rownames(cfs) <- rownames(cis) <- sexes

colnames(cis) <- paste(rep(c("Linf","K","t0"),each=2),

rep(c("LCI","UCI"),times=2),sep=".")

colnames(preds1) <- colnames(preds2) <- c("sex","age","fit","LCI","UCI")

The preds1 and preds2 objects now contain the predicted mean lengths-at-age with associated confidence intervals in the desired long format.

headtail(preds1) # predicted lengths-at-age w/ CIs for ALL ages

## sex age fit LCI UCI

## V1 female -1.0 63.17547 117.1139 200.3062

## V2 female -0.8 103.98483 150.7788 220.9030

## V3 female -0.6 141.94750 182.5517 240.3672

## V641 male 11.6 568.09475 552.3777 582.0790

## V651 male 11.8 568.48780 552.5808 582.6951

## V661 male 12.0 568.85535 552.7899 583.4920

headtail(preds2) # predicted lengths-at-age w/ CIs for OBSERVED ages

## sex age fit LCI UCI

## 1 female 0.0 240.6728 262.9114 293.8324

## 2 female 0.2 269.1007 285.0642 309.3961

## 3 female 0.4 295.5456 305.8965 324.4556

## 105 male 10.6 565.6806 551.1035 578.3509

## 106 male 10.8 566.2303 551.4152 579.1737

## 107 male 11.0 566.7443 551.6928 580.0494

**Multiple VBGFs on One Plot**

Constructing the plot with multiple VBGF trajectories is similar to what was shown for one group. Note, however, that colors will depend on the sex variable for the confidence polygon because of fill=sex in geom\_ribbon(), the points because of color=sex in geom\_point(), and the lines because of color=sex in geom\_line(). The default colors can be changed in a variety of ways but are set manually to two colors for both fill and color aesthetics below with scale\_color\_manual().[^chooseColors] Also note that position\_dodge() is used in geom\_point() to shift the points for the groups slightly left and right to minimize overlap of points between groups. Finally, legend.position= in theme() is used to place the legend inside the plot centered at approximately 80% of the way along the x-axis and 20% of the way up the y-axis, and legend.title= removes the title on the legend (was just the word “sex”).

vbFitPlot1 <- ggplot() +

geom\_ribbon(data=preds2,aes(x=age,ymin=LCI,ymax=UCI,fill=sex),alpha=0.2) +

geom\_point(data=w14T,aes(y=tl,x=age,color=sex),alpha=0.25,size=2,

position=position\_dodge(width=0.2)) +

geom\_line(data=preds1,aes(y=fit,x=age,color=sex),size=1,linetype=2) +

geom\_line(data=preds2,aes(y=fit,x=age,color=sex),size=1) +

scale\_color\_manual(values=c('#00429d', '#93003a'),

aesthetics=c("fill","color")) +

scale\_y\_continuous(name="Total Length (mm)",limits=c(0,700),expand=c(0,0)) +

scale\_x\_continuous(name="Age (years)",expand=c(0,0),

limits=c(-1,12),breaks=seq(0,12,2)) +

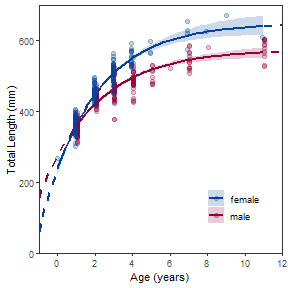
theme\_bw() +

theme(panel.grid=element\_blank(),

legend.position=c(0.8,0.2),

legend.title=element\_blank())

vbFitPlot1



Some people may prefer to just see model fits. If so, then simply omit geom\_point().

vbFitPlot2 <- ggplot() +

geom\_ribbon(data=preds2,aes(x=age,ymin=LCI,ymax=UCI,fill=sex),alpha=0.2) +

geom\_line(data=preds1,aes(y=fit,x=age,color=sex),size=1,linetype=2) +

geom\_line(data=preds2,aes(y=fit,x=age,color=sex),size=1) +

scale\_color\_manual(values=c('#00429d', '#93003a'),

aesthetics=c("fill","color")) +

scale\_y\_continuous(name="Total Length (mm)",limits=c(0,700),expand=c(0,0)) +

scale\_x\_continuous(name="Age (years)",expand=c(0,0),

limits=c(-1,12),breaks=seq(0,12,2)) +

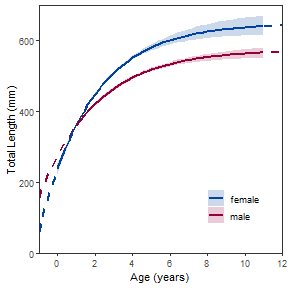
theme\_bw() +

theme(panel.grid=element\_blank(),

legend.position=c(0.8,0.2),

legend.title=element\_blank())

vbFitPlot2



**Multiple VBGFs in Separate Plots**

An alternative to putting multiple VBGF trajectories in one plot is to separate them into individual plots. This is easily handled by including the “grouping” variable name within vars() within facet\_wrap().[3](http://derekogle.com/fishR/2020-01-02-ggplot-vonB-fitPlot-2#fn:removedColors)

vbFitPlot3 <- ggplot() +

geom\_ribbon(data=preds2,aes(x=age,ymin=LCI,ymax=UCI),alpha=0.2) +

geom\_point(data=w14T,aes(y=tl,x=age),alpha=0.25,size=2) +

geom\_line(data=preds1,aes(y=fit,x=age),size=1,linetype=2) +

geom\_line(data=preds2,aes(y=fit,x=age),size=1) +

scale\_y\_continuous(name="Total Length (mm)",limits=c(0,700),expand=c(0,0)) +

scale\_x\_continuous(name="Age (years)",expand=c(0,0),

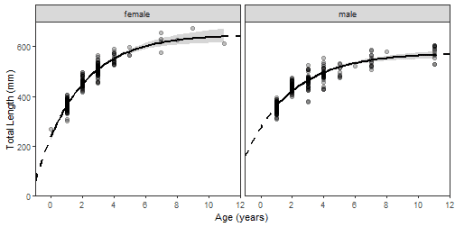
limits=c(-1,12),breaks=seq(0,12,2)) +

facet\_wrap(vars(sex)) +

theme\_bw() +

theme(panel.grid=element\_blank())

vbFitPlot3



Faceting is more interesting when there are more “groups.” The plot below shows different VBGF fits across all available years for female Walleye from location “1.” The code is basically the same as above (i.e., strategically replacing sex with fyear and making sure to use the new data.frame).

wfT <- filter(WalleyeErie2,sex=="female",loc==1)

agesum <- group\_by(wfT,year) %>%

summarize(minage=min(age),maxage=max(age))

years <- unique(wfT$year)

nyears <- length(years)

cfs <- cis <- preds1 <- preds2 <- NULL

for (i in 1:nyears) {

## Loop notification (for peace of mind)

cat(years[i],"Loop\n")

## Isolate year's data

tmp1 <- filter(wfT,year==years[i])

## Fit von B to that year

sv1 <- vbStarts(tl~age,data=tmp1)

fit1 <- nls(tl~vb(age,Linf,K,t0),data=tmp1,start=sv1)

## Extract and store parameter estimates and CIs

cfs <- rbind(cfs,coef(fit1))

boot1 <- Boot(fit1)

tmp2 <- confint(boot1)

cis <- rbind(cis,c(tmp2["Linf",],tmp2["K",],tmp2["t0",]))

## Predict mean lengths-at-age with CIs

## preds1 -> across all ages

## preds2 -> across observed ages only

ages <- seq(-1,16,0.2)

boot2 <- Boot(fit1,f=predict2)

tmp2 <- data.frame(year=years[i],age=ages,

predict(fit1,data.frame(age=ages)),

confint(boot2))

preds1 <- rbind(preds1,tmp2)

tmp2 <- filter(tmp2,age>=agesum$minage[i],age<=agesum$maxage[i])

preds2 <- rbind(preds2,tmp2)

}

## 2003 Loop

## 2004 Loop

## Warning: Starting value for Linf is very different from the observed maximum

## length, which suggests a model fitting problem. See a Walford or

## Chapman plot to examine the problem. Consider either using the mean

## length for several of the largest fish (i.e., use 'oldAge' in

## 'methLinf=') or manually setting Linf in the starting value list

## to the maximum observed length.

## 2005 Loop

## 2006 Loop

## 2007 Loop

## 2008 Loop

## 2009 Loop

## 2010 Loop

## 2011 Loop

## 2012 Loop

## 2013 Loop

## 2014 Loop

rownames(cfs) <- rownames(cis) <- years

colnames(cis) <- paste(rep(c("Linf","K","t0"),each=2),

rep(c("LCI","UCI"),times=2),sep=".")

colnames(preds1) <- colnames(preds2) <- c("year","age","fit","LCI","UCI")

vbFitPlot4 <- ggplot() +

geom\_ribbon(data=preds2,aes(x=age,ymin=LCI,ymax=UCI),alpha=0.2) +

geom\_point(data=wfT,aes(y=tl,x=age),alpha=0.25,size=2) +

geom\_line(data=preds1,aes(y=fit,x=age),size=1,linetype=2) +

geom\_line(data=preds2,aes(y=fit,x=age),size=1) +

scale\_y\_continuous(name="Total Length (mm)",limits=c(0,800),expand=c(0,0)) +

scale\_x\_continuous(name="Age (years)",expand=c(0,0),

limits=c(-1,17),breaks=seq(0,16,2)) +

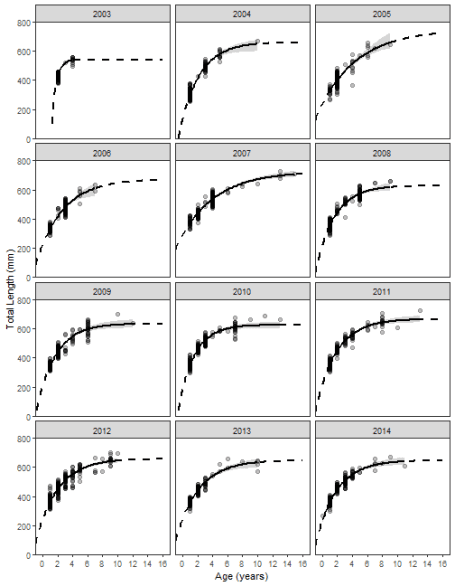
facet\_wrap(vars(year),ncol=3) +

theme\_bw() +

theme(panel.grid=element\_blank())

vbFitPlot4

## Warning: Removed 11 rows containing missing values (geom\_path).



**BONUS – Plots of Parameter Estimates**

A bonus for keeping track of the parameter point and interval estimates through this entire post is to plot the estimates across years. I will leave this up to you to decipher, but note that the years must be added to the cfs and cis data.frames to make the plot shown here.

( cfs <- data.frame(year=years,cfs) )

## year Linf K t0

## 2003 2003 540.1804 1.6843248 1.0770783

## 2004 2004 660.1222 0.3768720 -0.6210112

## 2005 2005 743.4416 0.1972271 -1.9791564

## 2006 2006 673.1494 0.2815995 -1.4618985

## 2007 2007 724.7986 0.2086957 -2.4787931

## 2008 2008 628.1655 0.3978859 -1.1359461

## 2009 2009 633.4864 0.4015328 -0.9812580

## 2010 2010 625.2466 0.4755051 -0.7492581

## 2011 2011 665.2861 0.3639903 -1.1207619

## 2012 2012 657.7450 0.3470683 -1.3561054

## 2013 2013 648.4110 0.3284681 -1.4754777

## 2014 2014 648.2084 0.3615400 -1.2836315

( cis <- data.frame(year=years,cis) )

## year Linf.LCI Linf.UCI K.LCI K.UCI t0.LCI t0.UCI

## 2003 2003 553.5691 593.7532 0.2806559 0.4049626 -2.426514 -1.569173

## 2004 2004 556.7326 593.1636 0.2812658 0.4006324 -2.389605 -1.582354

## 2005 2005 555.1986 590.8868 0.2822799 0.4049153 -2.427498 -1.583655

## 2006 2006 553.3085 592.0021 0.2773397 0.3996300 -2.441460 -1.620525

## 2007 2007 554.6483 593.0184 0.2798028 0.4045309 -2.416457 -1.584153

## 2008 2008 555.0262 592.0917 0.2840524 0.4124890 -2.396447 -1.562232

## 2009 2009 553.5509 591.2554 0.2848601 0.4068678 -2.388673 -1.555427

## 2010 2010 552.0304 589.9373 0.2844503 0.4069120 -2.421615 -1.577490

## 2011 2011 555.7924 592.1526 0.2778908 0.4008409 -2.439424 -1.589794

## 2012 2012 556.8813 592.1483 0.2826413 0.3972812 -2.391029 -1.599025

## 2013 2013 554.9912 592.0538 0.2843511 0.4075091 -2.368029 -1.559858

## 2014 2014 556.2411 593.6721 0.2781399 0.3990074 -2.452538 -1.590839

p.Linfs <- ggplot() +

geom\_point(data=cfs,aes(x=year,y=Linf)) +

geom\_line(data=cfs,aes(x=year,y=Linf),color="gray80") +

geom\_errorbar(data=cis,aes(x=year,ymin=Linf.LCI,ymax=Linf.UCI),width=0.3) +

scale\_y\_continuous(name=expression(L[infinity])) +

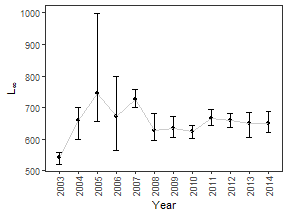
scale\_x\_continuous(name="Year",breaks=years) +

theme\_bw() +

theme(panel.grid=element\_blank(),

axis.text.x=element\_text(angle=90,vjust=0.5))

p.Linfs



This is repeated for the other two parameters.

p.K <- ggplot() +

geom\_point(data=cfs,aes(x=year,y=K)) +

geom\_line(data=cfs,aes(x=year,y=K),color="gray80") +

geom\_errorbar(data=cis,aes(x=year,ymin=K.LCI,ymax=K.UCI),width=0.3) +

scale\_y\_continuous(name="K") +

scale\_x\_continuous(name=" ",breaks=years) +

theme\_bw() +

theme(panel.grid=element\_blank(),

axis.text.x=element\_text(angle=90,vjust=0.5))

p.t0 <- ggplot() +

geom\_point(data=cfs,aes(x=year,y=t0)) +

geom\_line(data=cfs,aes(x=year,y=t0),color="gray80") +

geom\_errorbar(data=cis,aes(x=year,ymin=t0.LCI,ymax=t0.UCI),width=0.3) +

scale\_y\_continuous(name=expression(t[0])) +

scale\_x\_continuous(name=" ",breaks=years) +

theme\_bw() +

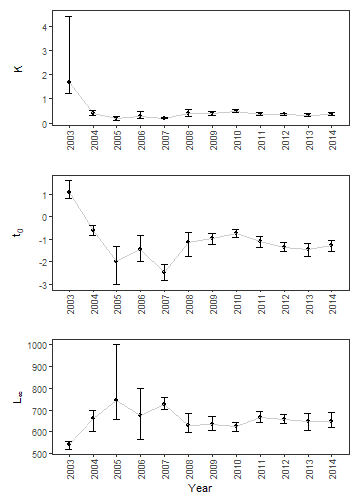
theme(panel.grid=element\_blank(),

axis.text.x=element\_text(angle=90,vjust=0.5))

Which can then be neatly placed on top of each other with the patchwork package.

library(patchwork)

p.K / p.t0 / p.Linfs



**Final Thoughts**

I am trying to post examples here as I learn ggplot2.