OSO_Smolt_L_W

Scott Akenhead, Braden Judson

2023-04-22

Table of contents

#Dataset ## Source

For smolts in river, 2006-16, 2018, 2019, 2021; 14 years; 26,230 observations.

establish provenance of data, reference to method.
datasets used in analysis need to be in Zenodo or eq. for paper to be accepted.

import and summarize

overall summary

```
a = read.csv('data/OSO_smolts_tidy.csv')
cat(colnames(a),'\n')
```

```
collect.date year doy BY location ona_id total.age european.age FL_mm weight_g hatchery sex
```

```
a$condition= 1e5*a$weight_g * a$FL_mm^(-3)
Simple(a[, c('FL_mm', 'weight_g','condition')])
```

```
    FL_mm  weight_g condition
n   26230  26083    26083
m   90.3   7.95     1.03
s   11.2   3.21     0.0979
se  0.0692 0.0199   0.000606
cv  0.124  0.404    0.095
med 88     6.95     1.03
mad 11.9   2.8      0.0876
min 73     2.29     0.5
q1  81     5.43     0.97
q3  99     10.3     1.09
max 115    19.3     2.14
```

summaries by year

```
year_simple = by(a[, c('FL_mm', 'weight_g','condition')], a$year,Simple, do.print=F)
year=c(2006:2016,2018, 2019,2021)  # missing 2017, 2020
k = year-2005
year_med_mad=data.frame(year=2006:2021,L_med=NA,L_mad=NA,W_med=NA,W_mad=NA,C_med=NA,C_mad
for(j in 1:14) year_med_mad[ k[j],2:7]= year_simple[[j]][c(6,7, 17,18,28,29)]
# leaves 2 rows, 2017 and 2020, as NA.
print(year_med_mad)
```

```
   year L_med L_mad W_med W_mad C_med  C_mad
1  2006    90  8.90  6.80 2.080 0.945 0.0787
2  2007    94 10.40  8.43 2.550 1.020 0.0672
3  2008    84  8.90  5.74 1.560 0.987 0.1000
4  2009    99  5.93  9.47 1.820 0.973 0.0672
5  2010    80  5.93  4.68 1.040 0.924 0.0957
6  2011   103  7.41 11.50 2.460 1.060 0.0647
7  2012    82  7.41  5.71 1.250 1.020 0.1070
8  2013    89  5.93  7.37 1.560 1.030 0.0668
9  2014    86 11.90  6.66 2.860 1.040 0.0756
10 2015    89  5.93  7.38 1.330 1.080 0.0567
11 2016    83  8.90  6.00 2.020 1.020 0.0642
12 2017    NA    NA    NA    NA    NA     NA
13 2018    83  4.45  5.36 0.919 0.955 0.0460
14 2019   101  4.45 11.10 1.420 1.070 0.0463
15 2020    NA    NA    NA    NA    NA     NA
16 2021    93  4.45  8.00 0.941 1.010 0.0483
```

PDDs

Length

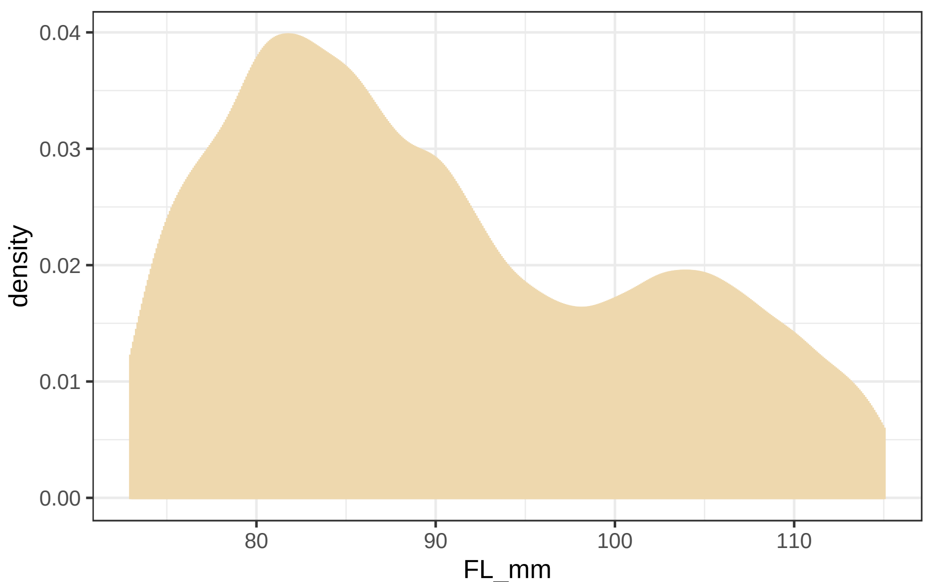
First, probability density distribution (PDD) of lengths (mm), all years combined.

```
Simple(a$FL_mm)
```

```
     n       m       s      se      cv     med     mad     min      q1      q3     max
 26230    90.3    11.2 0.0692   0.124      88    11.9      73      81      99     115
```

```
ggplot(data=a, aes(FL_mm)) + theme_bw()+
    geom_histogram(stat = "density", colour="wheat2")
```

```
Warning in geom_histogram(stat = "density", colour = "wheat2"): Ignoring unknown
parameters: `binwidth`, `bins`, and `pad`
```




```
# breaks=seq(70,120,2.5),
```

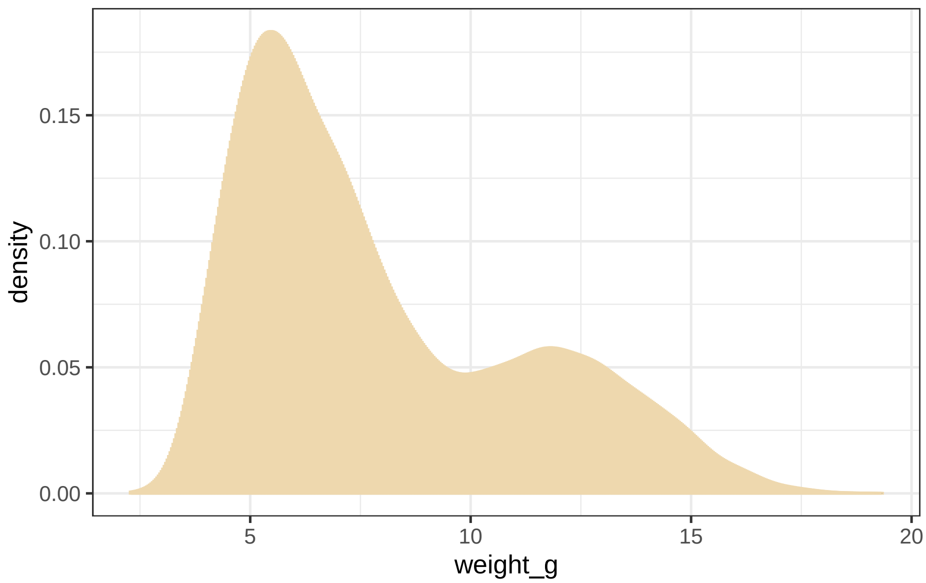
Weight

Second, PDD of weights

```
Simple(a$weight_g)
```

```
    n      m      s      se     cv    med    mad    min     q1     q3    max
26083   7.95   3.21 0.0199  0.404   6.95    2.8   2.29   5.43   10.3   19.3
```

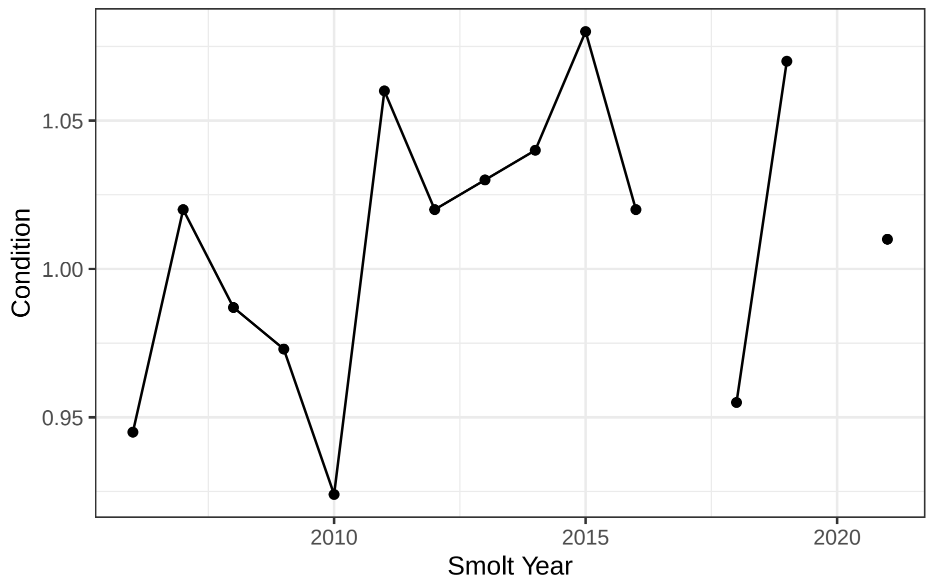
```
ggplot(data=a, aes(weight_g)) + theme_bw()+
    geom_histogram(stat = "density",colour="wheat2")
```



Condition

time series of smolt condition, regardless of age.

```
ggplot(year_med_mad, aes(x=year, y=C_med)) + theme_bw() +
    geom_point() + geom_line() +labs(x='Smolt Year', y='Condition')
```



Fitting Ages to Lengths

fitting two normals to density distributions of length by year.

```
GaussTwice = function (params,dat ){
    prop1  = params[1]
    mean1  = params[2]
    stdev1 = params[3]
    mean2  = params[4]
    stdev2 = params[5]
    bins    = dat[,1] # data as columns
    density = dat[,2]
    # dat is vector bins, vector density
    # sum (density) is 1, so two proportions, prop2 = 1-prop 1
    # prop1 bounded 0 to 1.
    # BUT predicted density range exceeds observed range,
    # so sum(d_hat) < 1. So correct to sum to 1.
    d_hat = prop1  * dnorm(bins, mean1, stdev1) +
         (1-prop1) * dnorm(bins, mean2, stdev2)
    d_hat = d_hat * 1.0/sum(d_hat) # correct for truncated prob. density dists.
    ssq = sum ( (density - d_hat)^2)
    return(ssq)
}
GaussOnce = function (params,dat ){
```

```
    mean1  = params[2]
    stdev1 = params[3]
    bins    = dat[,1] # data as columns
    density = dat[,2]
    # dat is vector bins, vector density
    # BUT predicted density range exceeds observed range,
    # so sum(d_hat) < 1. So correct to sum to 1.
    d_hat = dnorm(bins, mean1, stdev1)
    d_hat = d_hat * 1.0/sum(d_hat) # correct for truncated prob. density dists.
    ssq = sum ( (density - d_hat)^2)
    return(ssq)
}

Stats_SSQ_Hessian = function (fit1, nobs){
    denom = nobs-length(fit1$par)
    sigma2_reg= fit1$value / denom
    sigma_params = sqrt(sigma2_reg*diag(solve(fit1$hessian)) )
    x <- c(stdev_reg=sqrt(sigma2_reg), sigma_params )
    return(x)
}
```

Length Distribution

Typical raw data from manual collection:too many observations at a multiple of 10, alternation of frequency at even and odd values,. Alternative is collection by digital images, automated extraction of lengths.

```
L1= by(a$FL_mm, a$year, hist, breaks=c(72:115), plot=FALSE)
# can't get this to recognize freq=F or probability=T. output varies.
# so might have to calc. density as  n[i] /sum(n)
bins= L1[[1]][[4]] # 72.5  73.5  74.5 ...114.5 114.5 (n=43)
nbins=length(bins)
#
L_dens= as.data.frame(matrix(nrow=14, ncol=nbins))
for (j in 1:14)  L_dens[j,]= L1[[ j ]][[2]]
print('observations per year')
```

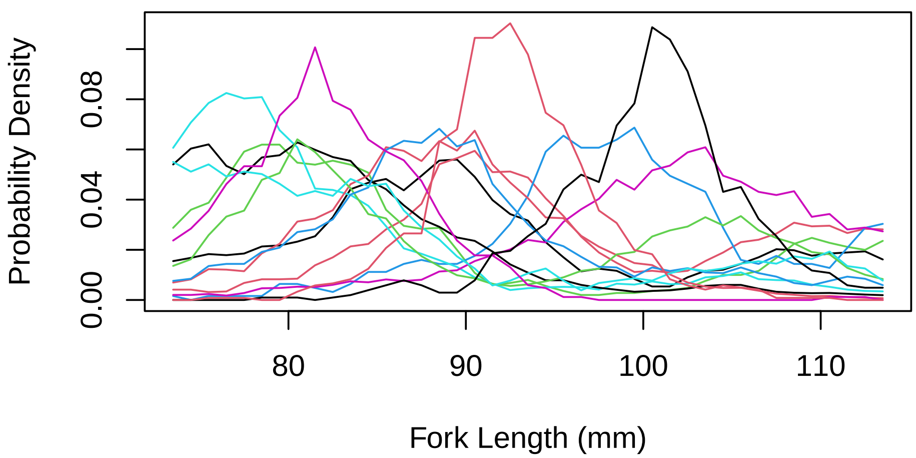
```
[1] "observations per year"
```

```
print(rowSums(L_dens))
```

```
 [1] 1590 1417  880  215 1945 2723 4327 3711 6832 1193  366  287  342  402
```

```r
# smooth with 3-point running mean of history.
# x[t] = 1/3 * (x[t-2] + x[t-1] + x[t])
# remove spikes at 90, 100,; pits 89 91, 99 101,;
# this makes first two bins NA
# but the last bin is the mean of last three bins,
# so the midpoint is the last minus 1 (preceding)
for (j in 1:14) L_dens[j,]=filter(unlist(L_dens[j,]), rep(1.0/3.0,3),method="convolution"
L_dens= L_dens[,-c(1,2)] # remove columns that are NA
bins  = bins[-c(1, nbins)] # drop first, slide left by one
nbins = nbins-2

# convert from frequency to density.
for (j in 1:14)  L_dens[j,] = L_dens[j,]  * (1.0/sum(L_dens[j,]))
matplot(bins, t(L_dens),  type='l', lty=1,
        xlab="Fork Length (mm)", ylab="Probability Density")
```



guess and fit

The fit is via R function optim(), minimizing SSQ (sum of squared deviations) via steepest-decent search (quasi-Newton) within supplied bounds for the parameter estimates (algorithm L-BFGS-B; Byrd *et. al.* 1995, Nocedal and Wright 1999).

Byrd, R. H., Lu, P., Nocedal, J. and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16: 1190-1208. doi:10.1137/0916069.

Nocedal, J. and Wright, S. J. (1999). Numerical Optimization. Springer.

From the minimized SSQ, the standard errors of the regression and of the parameters was calculated as

$$\sigma^2_{\text{reg}} = SSQ/(n-p)$$

$$\sigma_{\text{par}} = (\sigma^2_{\text{reg}}\ \text{diag}(\mathcal{H}^{-1}))^{1/2}$$

where n is the count of bins in the density distribution, p is the count of parameters fitted, and $\mathcal{H}$ is the Hessian matrix, the curvatures in the SSQ surface at its minimum.

```
fitAll=data.frame(year, prop1=NA, mean1=NA, stdev1=NA, mean2=NA, stdev2=NA, stdev_reg=NA,
for (j in 1:14){
    density= unlist(L_dens[j,])
    dat= data.frame(bins, density)  # data in columns
    params = c(prop1=0.5, mean1=82, stdev1=7.5, mean2=100, stdev2=7.5 )
    # test1 = GaussTwice(params, dat)
    fit1 = optim(params, GaussTwice,method="L-BFGS-B", hessian=T,
       lower=c(0,72, 3, 100, 3), upper=c(1, 95, 8, 120, 8),
       dat=dat)
    if (identical(fit1$convergence,0L)){   # integer zero
        x = Stats_SSQ_Hessian(fit1, 43)
        cat('\nYear',year[j], ' stdev_reg. =', round(x[1],4),  '\n')
        cat('parameters', names(x[-1]),       '\n') # drop first one
        cat('estimates ', round(fit1$par,3), '\n')
        cat('stdev     ', round(x[-1],3),   '\n\n')
        fitAll[j,2:12] = c(fit1$par, x)
    }   else {
        cat('\nYear',year[j], 'did not converge \n')
    }
}
```

```
Year 2006  stdev_reg. = 0.0045
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.772 87.115 6.465 109.91 4.583
stdev      0.019 0.177 0.194 0.561 0.526


Year 2007  stdev_reg. = 0.0033
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.627 90.553 4.945 110.202 5.14
stdev      0.017 0.105 0.117 0.344 0.362


Year 2008  stdev_reg. = 0.003
parameters prop1 mean1 stdev1 mean2 stdev2
```

```
estimates  0.74 80.296 5.885 111.577 5.331
stdev      0.022 0.106 0.116 0.687 0.587
```

```
Warning in sqrt(sigma2_reg * diag(solve(fit1$hessian))): NaNs produced
```

```
Year 2009  stdev_reg. = 0.009
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.512 95 8 100 4.806
stdev      NaN NaN NaN NaN NaN


Year 2010  stdev_reg. = 0.0032
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.875 76.71 5.976 103.749 8
stdev      0.018 0.174 0.235 1.063 1.648
```

```
Warning in sqrt(sigma2_reg * diag(solve(fit1$hessian))): NaNs produced
```

```
Year 2011  stdev_reg. = 0.0028
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.075 81.929 7.435 103.257 7.515
stdev      NaN 1.985 NaN 0.086 0.168
```

```
Warning in sqrt(sigma2_reg * diag(solve(fit1$hessian))): NaNs produced
```

```
Year 2012  stdev_reg. = 0.003
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.944 78.681 8 106.539 7.999
stdev      NaN 0.205 0.188 2.226 NaN
```

```
Warning in sqrt(sigma2_reg * diag(solve(fit1$hessian))): NaNs produced
```

```
Year 2013  stdev_reg. = 0.0028
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.92 88 6.14 100 7.999
stdev      NaN 0.1 0.13 NaN NaN


Year 2014  stdev_reg. = 0.0023
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.578 80.93 4.216 104.274 5.606
stdev      0.006 0.064 0.069 0.136 0.15


Year 2015  stdev_reg. = 0.0042
parameters prop1 mean1 stdev1 mean2 stdev2
```

```
estimates  0.837 87.589 5.381 106.014 5.603
stdev      0.02 0.134 0.154 0.805 0.916


Year 2016  stdev_reg. = 0.0044
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.812 78.059 8 108.003 6.014
stdev      0.018 0.369 0.33 0.669 0.766

Warning in sqrt(sigma2_reg * diag(solve(fit1$hessian))): NaNs produced


Year 2018  stdev_reg. = 0.0042
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.992 81.915 4.995 100 7.512
stdev      0.017 0.091 0.104 NaN 8.868

Warning in sqrt(sigma2_reg * diag(solve(fit1$hessian))): NaNs produced


Year 2019  stdev_reg. = 0.0061
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.255 95 7.429 101.194 3.492
stdev      NaN NaN 1.476 0.191 0.047


Year 2021  stdev_reg. = 0.0046
parameters prop1 mean1 stdev1 mean2 stdev2
estimates  0.887 92.085 3.352 100 3.957
stdev      0.046 0.158 0.121 1.664 1.092
```

```r
kable(fitAll[, 1:6],digits=c(0,2,1,1,1,1) )
```

year	prop1	mean1	stdev1	mean2	stdev2
2006	0.77	87.1	6.5	109.9	4.6
2007	0.63	90.6	4.9	110.2	5.1
2008	0.74	80.3	5.9	111.6	5.3
2009	0.51	95.0	8.0	100.0	4.8
2010	0.87	76.7	6.0	103.7	8.0
2011	0.07	81.9	7.4	103.3	7.5
2012	0.94	78.7	8.0	106.5	8.0
2013	0.92	88.0	6.1	100.0	8.0
2014	0.58	80.9	4.2	104.3	5.6
2015	0.84	87.6	5.4	106.0	5.6
2016	0.81	78.1	8.0	108.0	6.0
2018	0.99	81.9	5.0	100.0	7.5
2019	0.26	95.0	7.4	101.2	3.5

year	prop1	mean1	stdev1	mean2	stdev2
2021	0.89	92.1	3.4	100.0	4.0

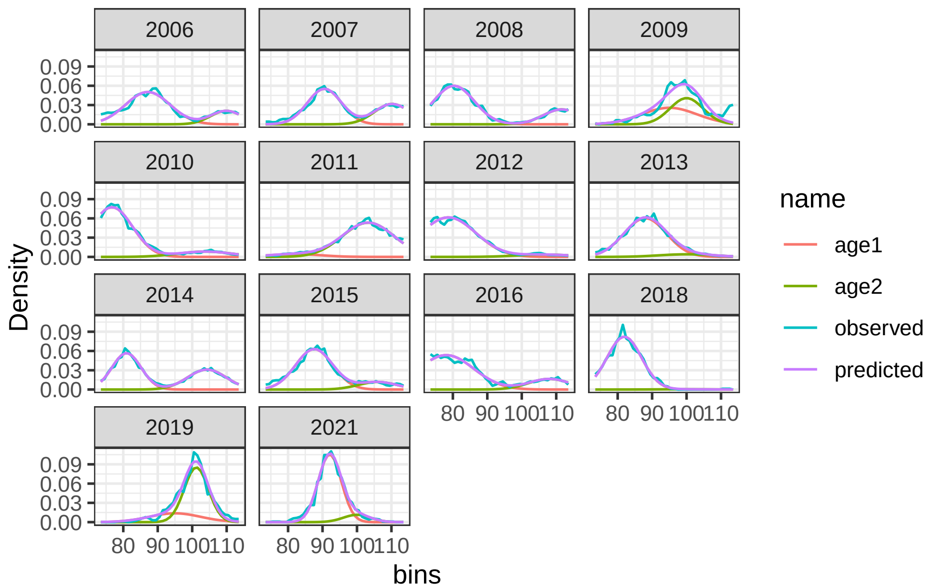
Plot Fit to Length PDD by Year

gather predicted and observed

```
#fitAll[14,]
# year   prop1  mean1 stdev1 mean2 stdev2
# 2021 0.76462 92.914 4.5128 91.68 1.8052
# stdev_reg prop1_SD mean1_SD stdev1_SD mean2_SD stdev2_SD
# 0.016557   0.19346   0.6544   0.71084  0.43318   0.72995
for(j in 1:14){
    gauss1 =     fitAll[j,2] * dnorm(bins, fitAll[j,3],fitAll[j,4])
    gauss2 = (1-fitAll[j,2]) * dnorm(bins, fitAll[j,5],fitAll[j,6])
    gauss = gauss1 + gauss2
    truncation = 1.0/sum(gauss)
    gauss1 = gauss1 * truncation
    gauss2 = gauss2 * truncation
    gauss = gauss * truncation
    density =  unlist(L_dens[j,])
# plot(bins,gauss1,type="l", ylim=c(0,.14));
# lines(bins,gauss2);lines(bins,gauss);lines(bins,density)
    x = data.frame(year=year[j], bins, observed=density, predicted=gauss, age1=gauss1, ag
    if(j == 1) {x1 = x} else {x1 = rbind(x,x1)}
}
```

Plots for each year

```
# colnames(economics_long) date variable value value01
# ggplot(economics_long, aes(date,value01,colour=variable))+geom_line()
# colnames(x1)colnames(x1) year bins observed predicted age1  age2
# colnames(x2) year bins name Density
x2 =pivot_longer(x1,cols=c("observed","predicted","age1","age2"),cols_vary = "slowest", v
ggplot(x2, aes(x=bins, y=Density, colour=name)) + theme_bw()+
    geom_line() + facet_wrap("year")
```


2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2018
2019
2021
0.09
0.06
0.03
0.00
Density
80
90
100
110
bins
name
age1
age2
observed
predicted