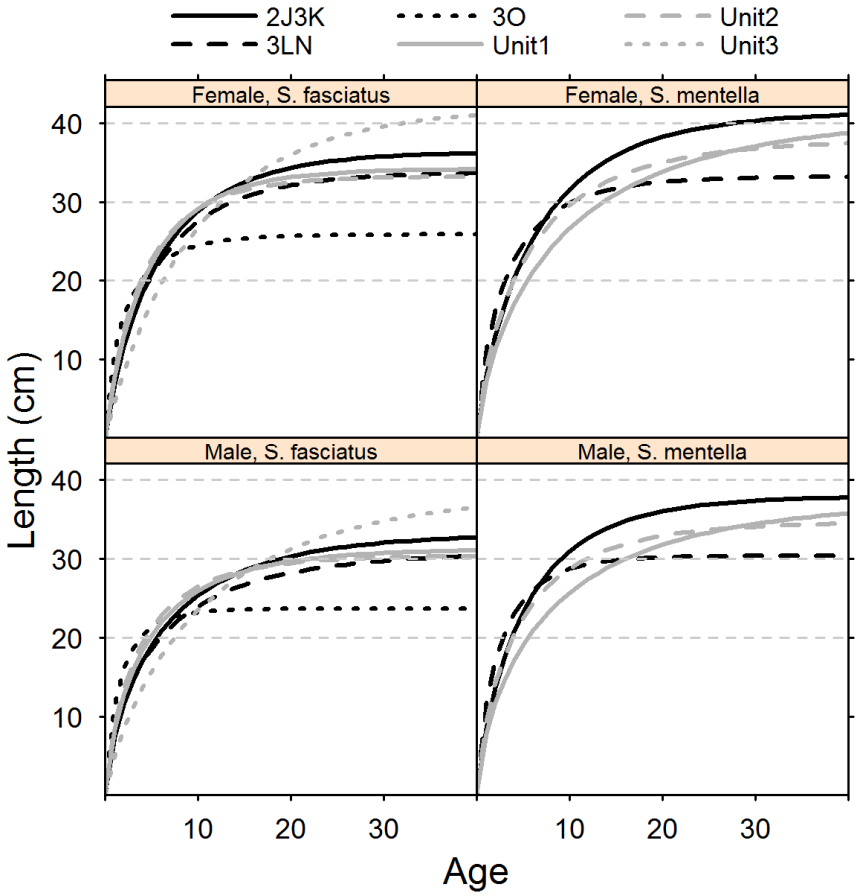
Marine Institute

Lecture 6: Growth of Individual Fish



Noel Cadigan

CFER

Centre for Fisheries Ecosystems Research

MARINE INSTITUTE

MEMORIAL UNIVERSITY

F6004 Lecture 6 Outline

- Growth data – length and weight at age or weight at length, tagging data
- Use of data and problems with extrapolations
- Von Bertalanffy growth model for length at age ($L_t = L_{inf}(1-e^{-K(t-t0)})$)
  - estimation using nls()
  - weighted estimation
  - Regression confidence intervals and bootstrap-t
  - parameter collinearity
  - alternative formulation of the Von Bertalanffy model
- Nonparametric growth model
- Biphasic model
- Growth in Weight ($W_t = a\ L_t^b$)
- Gompertz Growth Model
- Fabens model for growth from tagging data.
- Individual variability in growth
- Non-random samples due to gear selectivity and sampling designs
- Age measurement error
- Several populations

Readings: 1) Haddon Chapter 8.

Growth of individuals

- Stock production (i.e. yield to fishery) primarily involves (1) recruitment of new individuals (see lecture 8) to the stock, (2) growth of individuals in the stock, and (3) natural mortality
- We saw some of this in the yield-per-recruit analysis
- In this section we focus on the mathematical description, and estimation, of growth
- And some of the problems to deal with here!

Growth of individuals

- we focus on growth in terms of how length or weight increases with age
- this is important when estimating stock productivity (i.e. YPR, MSY, etc)
- When computing stock biomass, mature biomass, and fishery catch (in weight) from an age-based stock assessment model
- growth over-fishing – taking too many fish when they are still too small to produce optimum yield.

Growth of individuals

- we use mathematical models to describe growth processes
- these models often involve simple parameters that allow us to describe differences in growth rates for different stocks, species, etc.
- Also, these models allow us to convert stock assessment models of population numbers and catch numbers to population biomass and catch biomass, resp.

the data

- samples of age, size, and weight
- length and weight are easy to get, ages not so easy
- Many agencies (incl. DFO) use length-stratified age sampling; this “informative” sampling design requires some adjustment (e.g Perreault et al, 2019)
- For now we will only use annual estimates of mean length- or weight-at-age (length-stratification adjusted in estimates)
- Occasionally have tagging data that tell us the length when tagged, and the length when captured

Measurement errors

- can be difficult to age using otoliths
- measurement error increases with age
- different readers can give different ages
- cross validations between readers, and with fish of known ages, can help
- lengths and weights reported by fishermen can be inaccurate and have a lot of error
- time at liberty for tagged fish may be inaccurate
- Samples may not be random

Von Bertalanffy (VonB) growth model

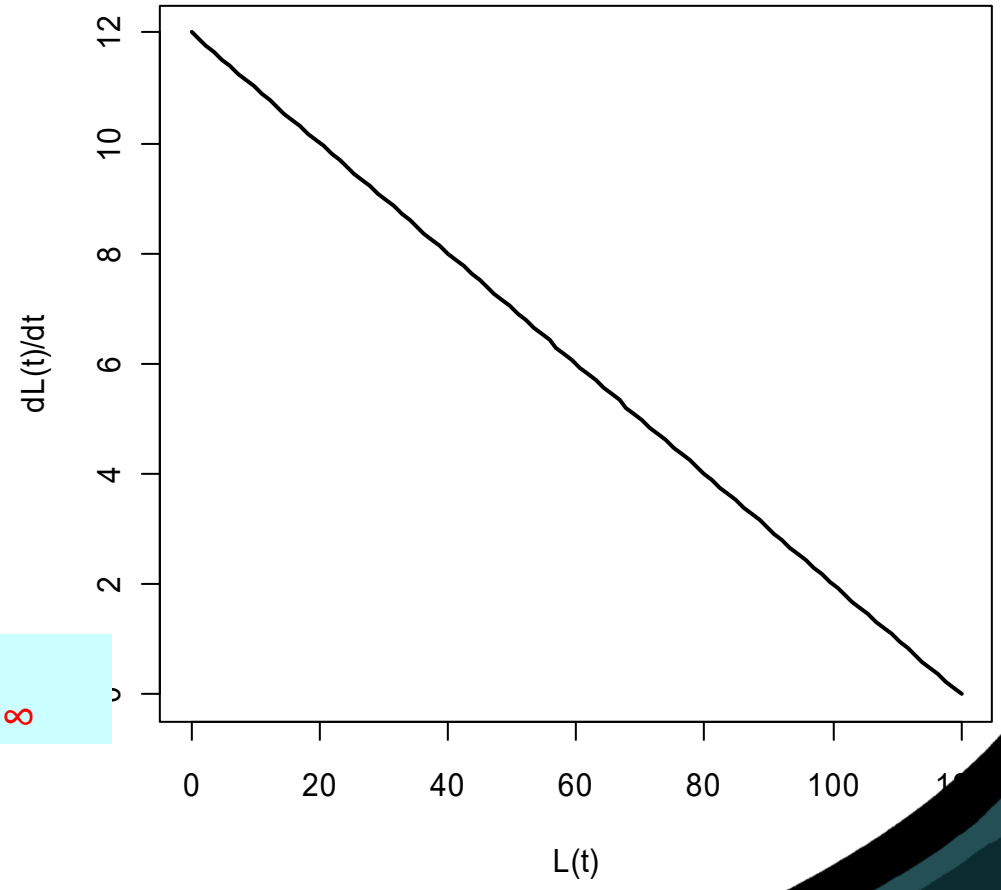
- The basis for the VonB growth model is the assumption that growth rate declines with size;
- that is, the growth rate of large individuals is less than the growth rate of small individuals
- The VonB model says that the growth rate declines linearly with size
- Let *L(t)* denote length at time *t*.

VonB model

the growth rate: change in *L(t)* as *t* changes

$$\frac{dL(t)}{dt} = \kappa\{L_\infty - L(t)\}$$

maximum size

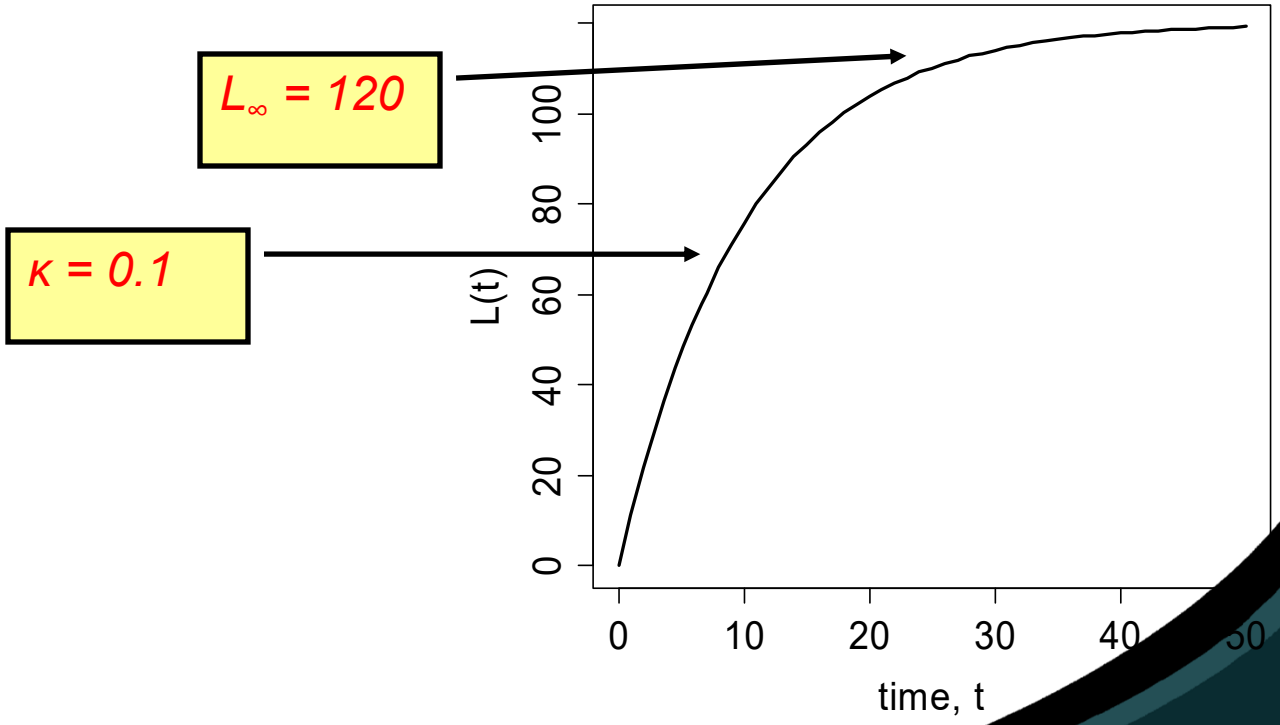


dL(t)/dt = 0 when $L(t)=L_\infty$

VonB model

– Lets assume that *L(t) = 0* when *t = 0*.

– The solution to the VonB differential growth rate equation is $L(t) = L_\infty \left(1 - e^{-\kappa t}\right)$



In Reverse

– Proposition: If $L(t) = L_\infty(1 - e^{-kt})$ then

$$\frac{dL(t)}{dt} = \kappa\{L_\infty - L(t)\}$$

– Proof $$\frac{dL(t)}{dt} = \frac{d}{dt} L_\infty\left(1 - e^{-\kappa t}\right) = L_\infty e^{-\kappa t}\kappa$$

– Note that $$L(t) = L_\infty\left(1 - e^{-\kappa t}\right) \Rightarrow L_\infty - L(t) = L_\infty e^{-\kappa t}$$

– hence $$\frac{dL(t)}{dt} = L_\infty e^{-\kappa t}\kappa = \boxed{\{L_\infty - L(t)\}\kappa}$$

Correct result

Estimation

- Assume we have n independent observations of fish length (denoted as L_i) at ages t_i.
- If the growth rates are the same for all individuals and the only variability is related to the measurement error in length, ε,
- $L_i = L_\infty\left(1 - e^{-kt_i}\right) + \varepsilon_i, \varepsilon_i \sim iid\ N(0, \sigma_\varepsilon^2)$

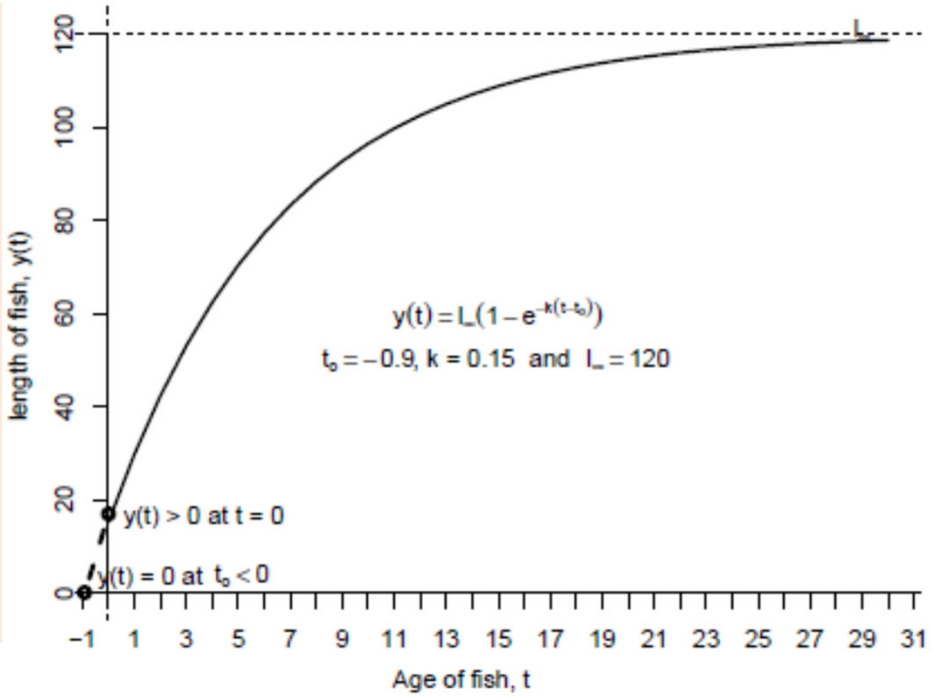
VonB model

- Sometimes $L(t) > 0$ when $t = 0$.
- A common approach is to back-calculate the negative time (t_o) when L would be zero, $L(t_o) = 0$

- This VonB extension is

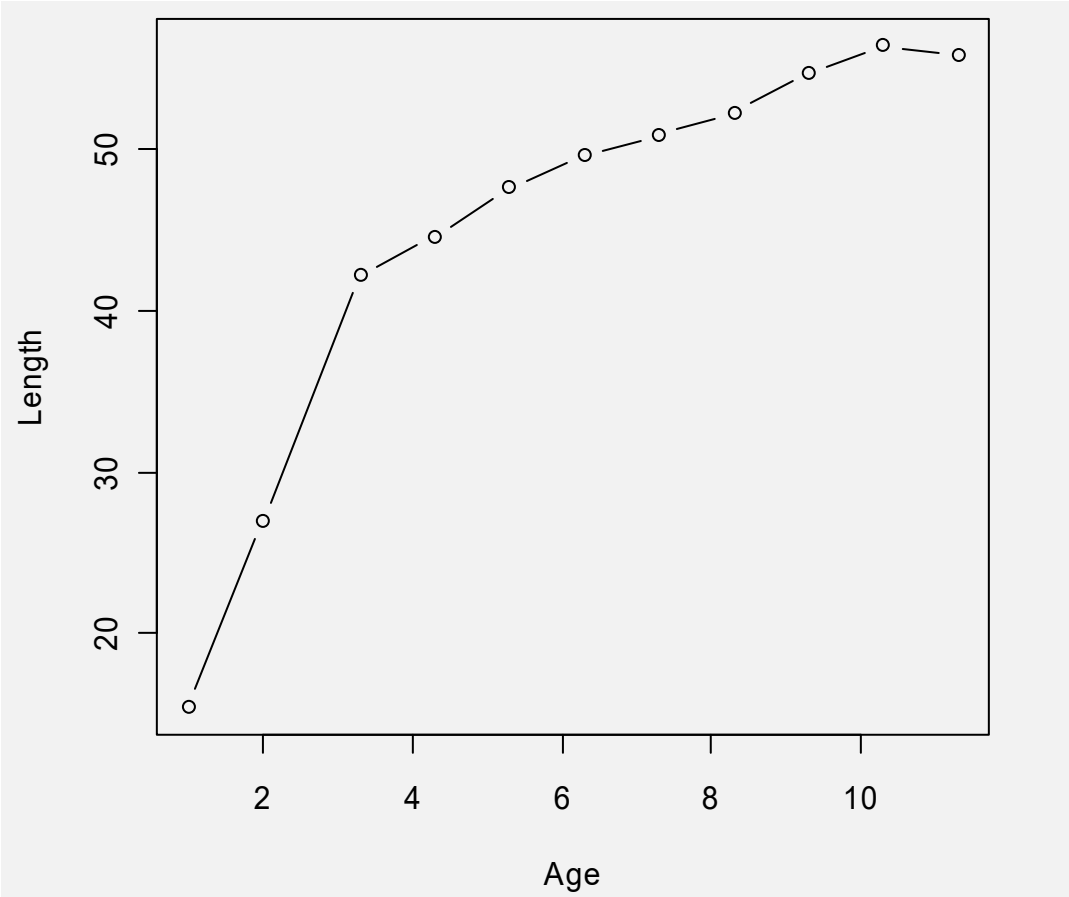
$$L(t) = L_\infty\{1 - e^{-k(t-t_o)}\}$$

- and this is the form we used in the Lecture 4 example



R estimation of vonB for ex 3.6 in Haddon

```
 print(age)
 [1]  1.0  2.0  3.3  4.3  5.3  6.3  7.3  8.3  9.3 10.3 11.3
> print(len)
 [1] 15.40 26.93 42.23 44.59 47.63 49.67 50.87 52.30 54.77 56.43 55.88
```



nls {stats}

Nls() in R

R Documen

Nonlinear Least Squares

Description

Determine the nonlinear (weighted) least-squares estimates of the parameters of a nonlinear model.

Usage

```
nls(formula, data, start, control, algorithm,
    trace, subset, weights, na.action, model,
    lower, upper, ...)
```

Arguments

`formula`
: a nonlinear model formula including variables and parameters. Will be coerced to a formula if necessary.

`data`
: an optional data frame in which to evaluate the variables in `formula` and `weights`. Can also be a list or an environment, but not a matrix.

`start`
: a named list or named numeric vector of starting estimates. When `start` is missing, a very cheap guess for `start` is tried (if `algorithm != "plinear"`).

`control`
: an optional list of control settings. See `nls.control` for the names of the settable control values and their effect.

`algorithm`
: character string specifying the algorithm to use. The default algorithm is a Gauss-Newton algorithm. Other possible values are `"plinear"` for the Golub-Pereyra algorithm for partially linear least-squares models and `"port"` for the 'nl2sol' algorithm from the Port library – see the references. Can be abbreviated.

`trace`
: logical value indicating if a trace of the iteration progress should be printed. Default is `FALSE`. If `TRUE` the residual (weighted) sum-of-squares and the parameter values are printed at the conclusion of each iteration. When the `"plinear"` algorithm is used, the conditional estimates of the linear parameters are printed after the nonlinear parameters. When the `"port"` algorithm is used the objective function value printed is half the residual (weighted) sum-of-squares.

R estimation of vonB for ex 3.6 in Haddon

Better practise to give nls the data via a data frame

```
 > gdata=data.frame(age=age,len=len)
> VonB.fit <- nls(len ~ Linf*(1 - exp(-k*(age-ao))),algorithm="port",
+           start = list(Linf=50, k = 0.1, ao = 0), lower=c(0,0,-10),
+           data=gdata)

Formula: len ~ Linf * (1 - exp(-k * (age - ao)))

Parameters:
     Estimate    Std. Error  t value   Pr(>|t|)
Linf 55.97801    1.08335  51.671    2.18e-11 ***
k      0.38558    0.03901    9.885    9.25e-06 ***
ao    0.17134    0.14177    1.209    0.261
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.558 on 8 degrees of freedom

Algorithm "port", convergence message: relative convergence (4)
```

```
> confint(VonB.fit)
Waiting for profiling to be done...
           2.5%        97.5%
Linf 53.6535544 58.9365498
k     0.2979468   0.4877251
ao   -0.2328941   0.4571272
```

I had to set algorithm="port", and give lower bounds for confint to work

See +2 slide

R estimation of vonB for ex 3.6 in Haddon

- The R nls function parameter estimates are somewhat different from what we found in lecture 4 using the R “optim” minimizer
- I checked, and the nls parameters result in a lower nll, **so optim did not find the minimum**.
- optim may not give good results when the number of parameters is large
- nls works better – It knows more about the problem, like certain derivatives

Residuals

- Are differences between observed and predicted,

$$e_i = y_i - \hat{y}_i$$

- Let *p* be the number of regression parameters. The **residual standard error** is:

$$\hat{\sigma}_e = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-p}}$$

p parameters involved in the estimate $\hat{y}_i$

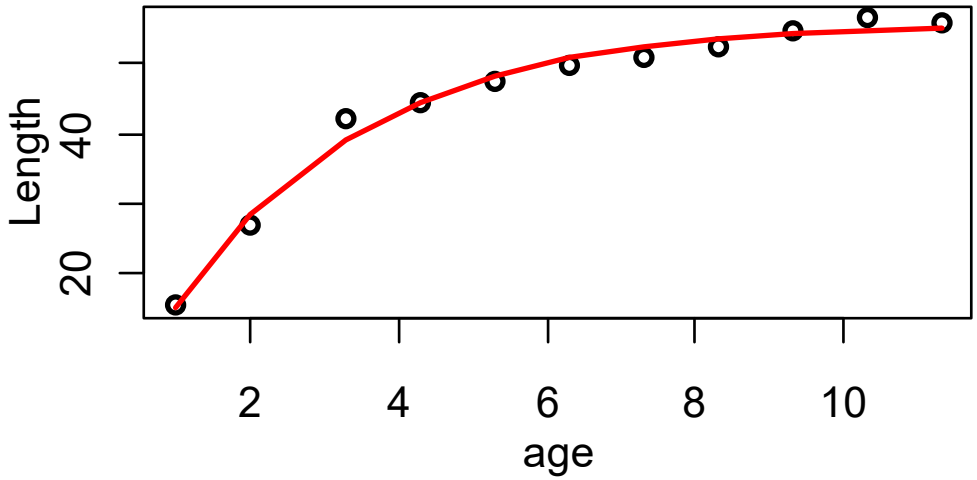
- Standardized residuals, $e_{si} = \left(y_i - \hat{y}_i\right) / \hat{\sigma}_e$
- ~95% of the e_{si}'s should be ±2, and no patterns with time or predicted values $(\hat{y})$

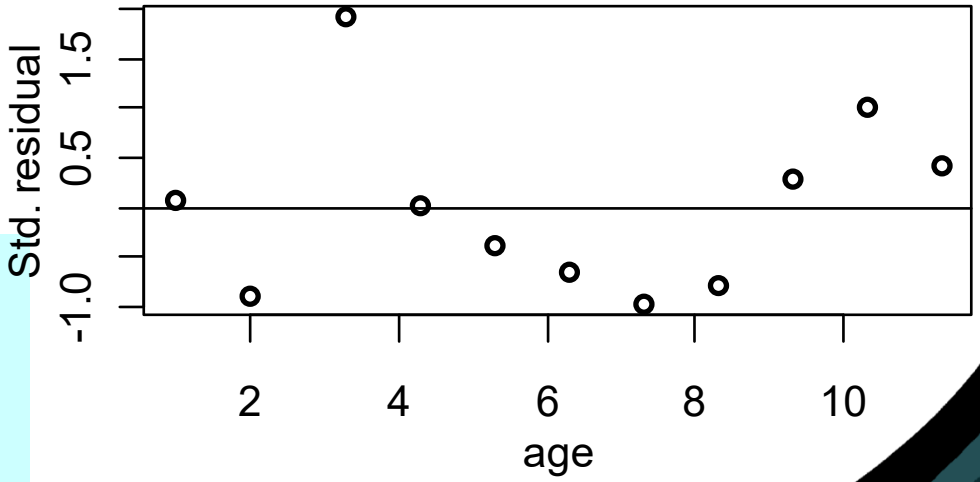
Residuals

- We will use standardized residuals to examine the goodness-of-fit of the model.
- If the model is not appropriate then we will usually see some type of trend in the residuals
- i.e. autocorrelation, trend in covariate, trend in residual variance, etc

Predict and residuals for ex 3.6 in Haddon

```
VonB.predict = predict(VonB.fit)
resid = gdata$len - VonB.predict
se.resid = sqrt(sum(resid**2)/(n - 3))
resid.s = resid/se.resid
```



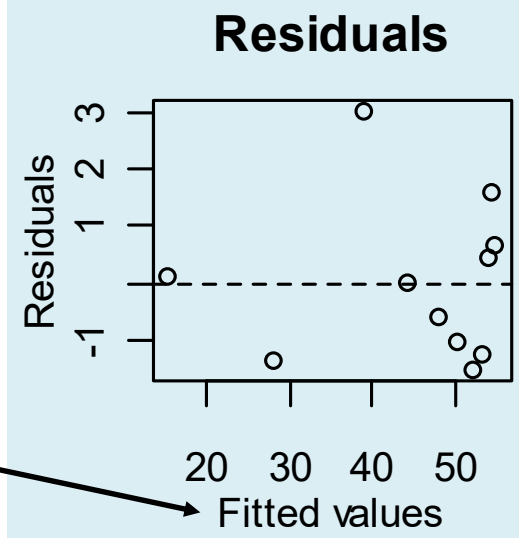


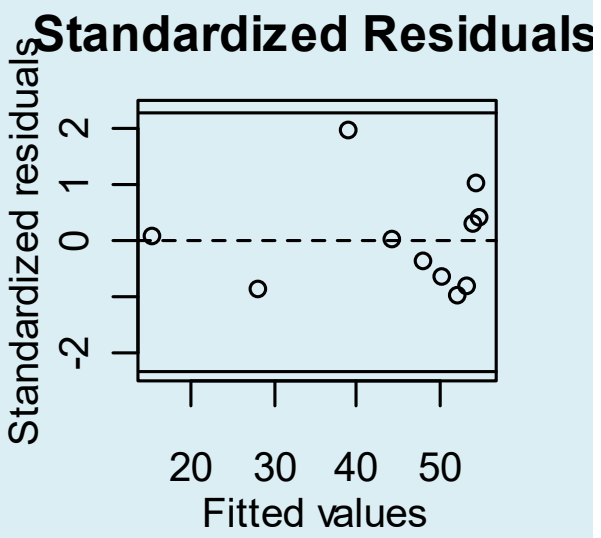

```
par(mfrow=c(2,1),mar=c(3.5,3,0.5,1),mgp=c(2,1,0))
plot(gdata$age,gdata$len,xlab='age',ylab='Length',
    type='p',lwd=2)
lines(gdata$age,VonB.predict,lwd=2,col='red')
plot(gdata$age,resid.s,xlab='age',ylab='Std. residual',
    type='p',lwd=2)
abline(h=0)
```

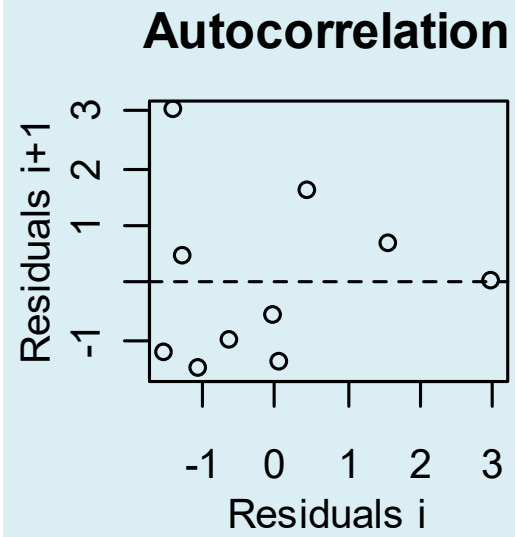
Predict and residuals for ex 3.6 in Haddon

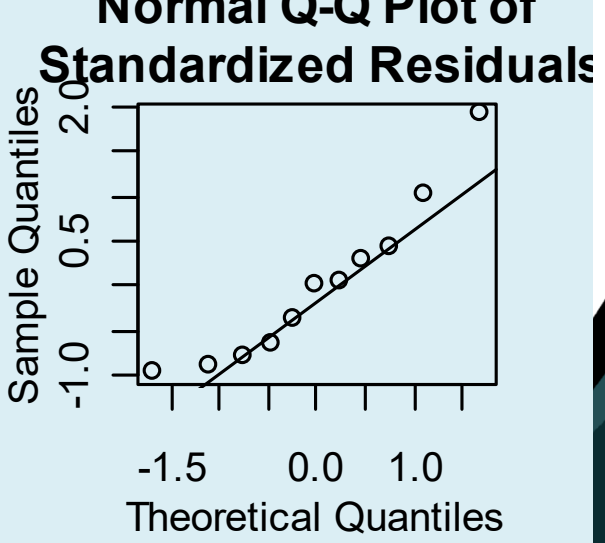
```
nr <- nlsResiduals(VonB.fit)
par(mar=c(3.5,3,3,2),
      mgp=c(2,1,0))
plot(nr, which = 0)
```

Predicted length









Need to install and load nlstools

```
install.packages("nlstools")
library("nlstools")
```

nls residual diagnostics for ex 3.6 in Haddon

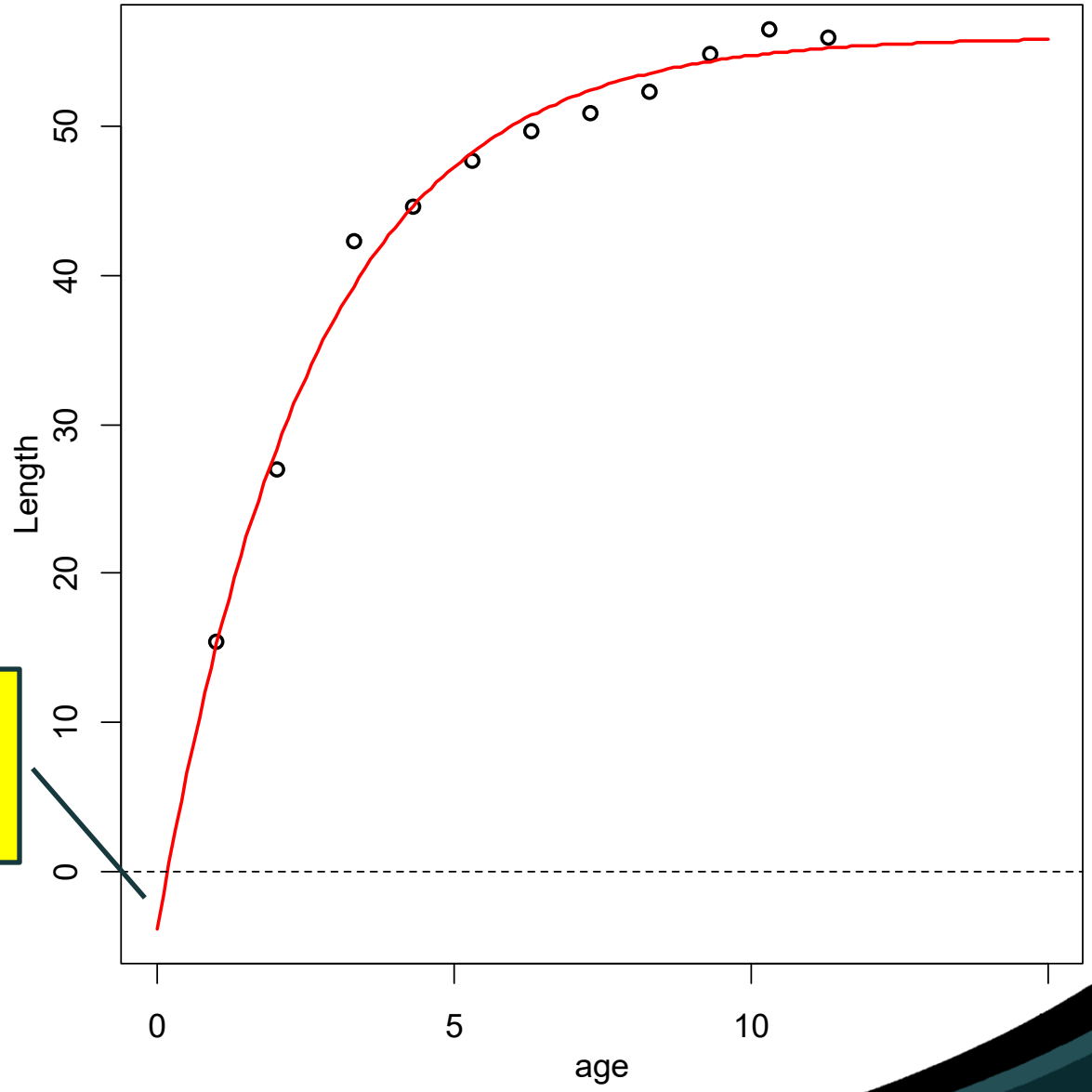
P-value >> 0.05
Std. residuals
seem N(0,1)

The sample size is small (n=11) so there will be limited ability to detect problems

```
> test.nlsResiduals(nr)

------
        Shapiro-Wilk normality test

data:  stdres
W = 0.9145, p-value = 0.2758


------
        Runs Test

data:  as.factor(run)
Standard Normal = -0.9331, p-value = 0.3507
alternative hypothesis: two.sided
```

P-value >> 0.05
no autocorrelation in residuals

Extrapolated growth curve



Not good!

Regression confidence intervals

- We have seen already how to get profile likelihood confidence intervals for the VonB parameter estimates using nls and confint.
- But now we need to think a little about the basis of frequentist statistical inference in the regression setting.
- Assume that we have *n* data pairs
- $s = \{(X_1,y_1),\ldots,(X_n,y_n)\}$

Regression confidence intervals

- The X's could be sets of covariates, and not just one covariate
- The random response variable is Y, and we model its relationship with X as
- $Y = \mu(X) + \varepsilon$
- $\mu(X)$ is the regression function of X, and ε is a random error term
- Eg. $\mu(X)$ could be $\theta_o + \theta_1 X$, the VonB model, etc.

Regression confidence intervals

- For statistical inference (e.g. CI's) we usually treat the X's as fixed, not random.
- There is a strong statistical principal called the "*conditionality principal*" that says that even if the X's are random, we should treat them as fixed for inferences about the regression parameters θ if the distribution of the X's does not depend on the θ's.

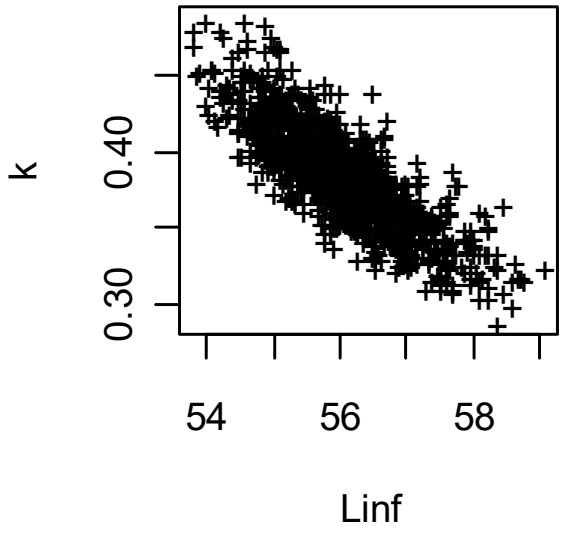
Regression confidence intervals

- So, confint(VonB.fit), the R function that returns profile likelihood confidence intervals for the VonB parameters, treats that *X*'s as fixed.
- They are not changed in confint.
- So how can we get bootstrapped CI's?
- If we resample data pairs then we are changing the *X*'s
- This violates the "conditionality principal"

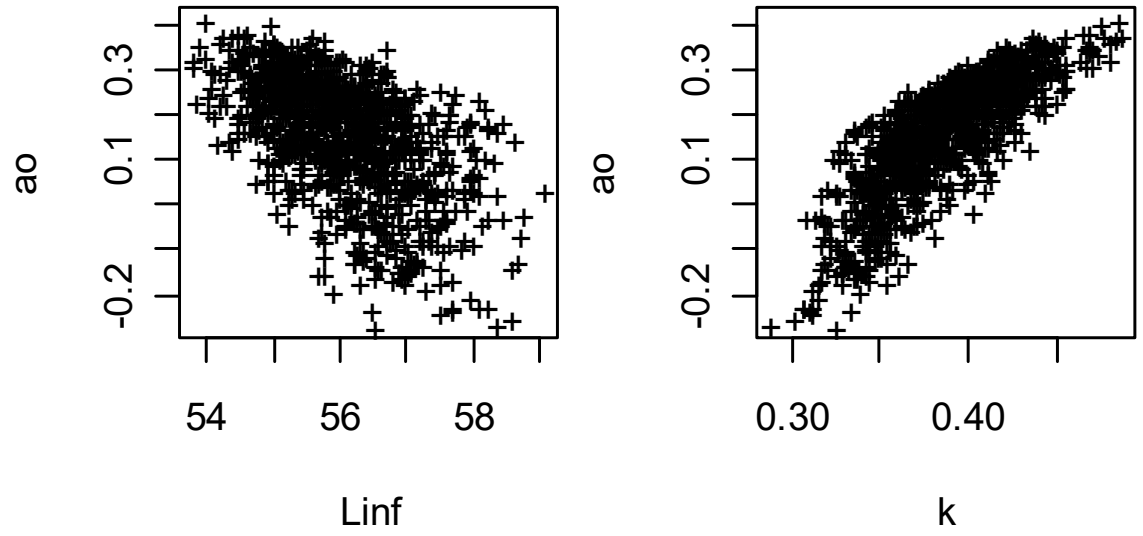
Bootstrap Regression CI's

- The solution is to bootstrap the residuals,
- and add the bootstrapped residuals to the predicted values from the regression model
- To get a bootstrapped sample, $y_{bi} = \hat{y}(X_i) + e_b$
- $s_b = \{(X_1, y_{b1}), \ldots, (X_n, y_{bn})\}$
- We can generate many of these samples, $s_1, \ldots, s_B$.
- Note that the *X*'s are the same in all $s_1, \ldots, s_B$.

VonB bootstrap for ex 3.6 in Haddon



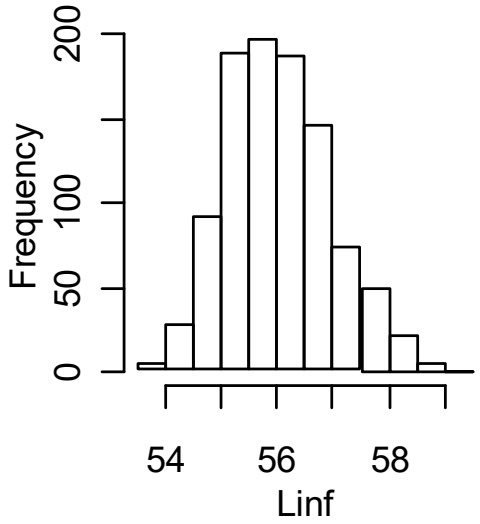

```
 > boo <- nlsBoot(VonB.fit)
Warning message:
In nlsBoot(VonB.fit) :
  The fit did not converge 2 times during bootstrapping
> plot(boo)
```

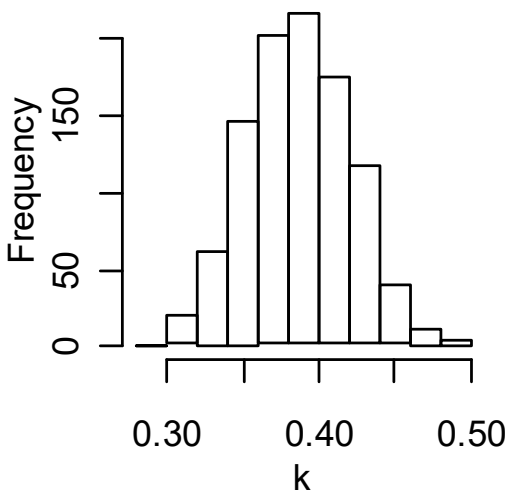


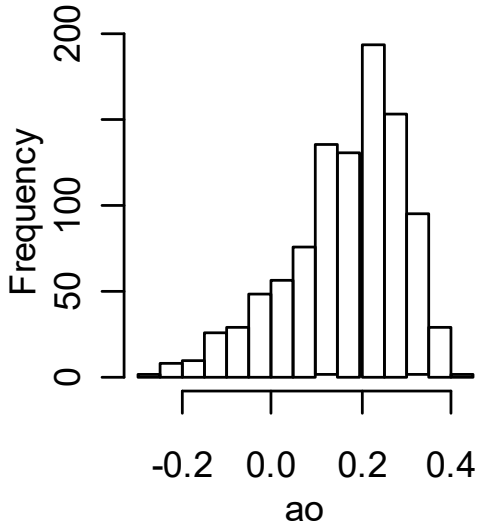
Pair-wise plots of bootstrapped parameter estimates

There is correlation in the estimates

VonB bootstrap for ex 3.6 in Haddon








```
par(mfrow=c(2,2),mar=c(3,3,1,1),mgp
=c(2,1,0))
hist(boo$coef[,1],xlab="Linf",main='')
hist(boo$coef[,2],xlab="k",main='')
hist(boo$coef[,3],xlab="ao",main='')
```

VonB Bootstrap CI's for ex 3.6 in Haddon

```
> summary.nlsBoot(boo)

------
Bootstrap estimates
      Linf         k         ao
55.9663822 0.3863738 0.1917764

------
Bootstrap confidence intervals
           2.5%        97.5%
Linf 54.3978221 58.0676270
k     0.3227703  0.4538977
ao   -0.1393464  0.3545293
```

These are percentile intervals

```
> confint(VonB.fit)
Waiting for profiling to be done...
          2.5%      97.5%
Linf 53.6535544 58.9365498
k     0.2979468  0.4877251
ao   -0.2328941  0.4571272
```

VonB Bootstrap-T CI's for ex 3.6 in Haddon

```
VonB.predict = predict(VonB.fit)
resid = gdata$len - VonB.predict
bootdata=gdata
VonB.summary = summary(VonB.fit)
VonB.parameters = VonB.summary$coefficients[,1]
VonB.parameters.se = VonB.summary$coefficients[,2]
```

Bootstrapped parameter estimates

```
bootVonB = function(i){
  bootdata$bootlen = VonB.predict + sample(resid,replace=T)
  bVonB.fit <- nls(bootlen ~ Linf*(1 - exp(-k*(age-ao))),algorithm="port",
              start = VonB.parameters,
              lower=c(20,0.1,-5),upper=c(70,0.7,5),data=bootdata)
  stats=summary(bVonB.fit)
  boot.t = (stats$coefficients[,1]-VonB.parameters)/stats$coefficients[,2]
  return(c(stats$coefficients[,1],boot.t))
}
```

Bootstrapped standard errors of the parameter estimates

VonB Bootstrap-T CI's for ex 3.6 in Haddon

```
B=1000
boot.out = sapply(1:B,bootVonB)

qbt = apply(boot.out[4:6,],1,quantile,probs=c(0.025,0.975))
boot.ci = matrix(0,ncol=2,nrow=length(VonB.parameters))
boot.ci[,1] = VonB.parameters - VonB.parameters.se*qbt[2,]
boot.ci[,2] = VonB.parameters - VonB.parameters.se*qbt[1,]
rownames(boot.ci)=names(VonB.parameters)
colnames(boot.ci)=c("2.5%","97.5%")

print(boot.ci)
```

VonB Bootstrap-T CI's for ex 3.6 in Haddon

```
> print(boot.ci)
          2.5%          97.5%
Linf 53.7495582 59.0401565
k     0.2987104   0.4795092
ao   -0.1489242   0.4936472
```

Bootstrap-T

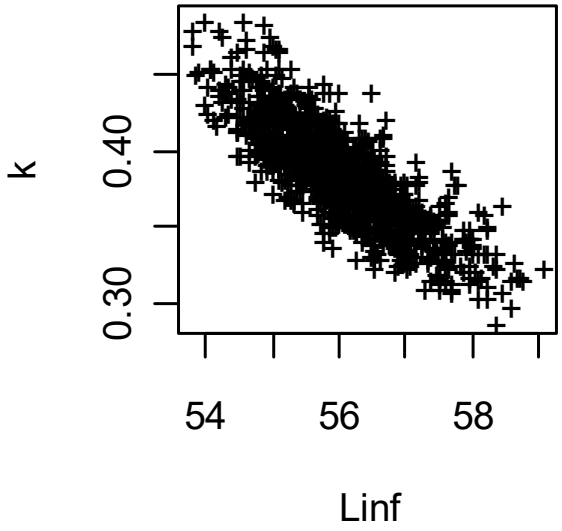
Profile likelihood

```
> confint(VonB.fit)
Waiting for profiling to be
done...
          2.5%      97.5%
Linf 53.6535544 58.9365498
k     0.2979468  0.4877251
ao   -0.2328941  0.4571272
```

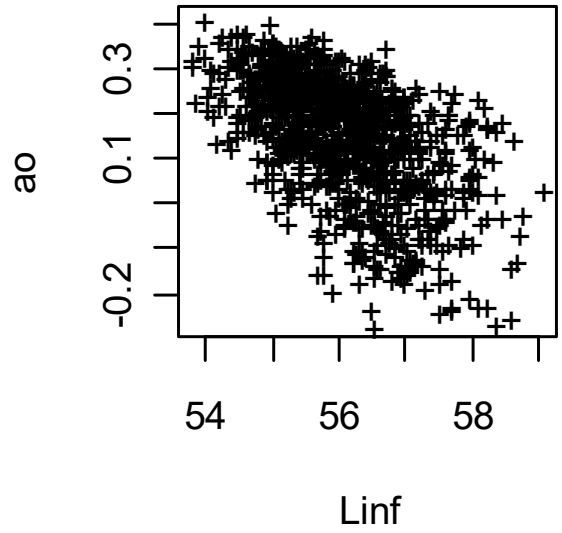
```
Bootstrap confidence intervals
          2.5%          97.5%
Linf 54.3978221 58.0676270
k     0.3227703   0.4538977
ao   -0.1393464   0.3545293
```

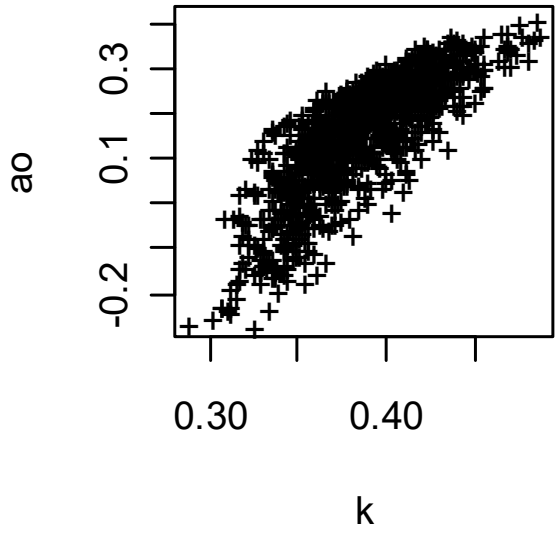
Bootstrap percentile

VonB bootstrap for ex 3.6 in Haddon[35]



- There is correlation in the parameter estimates
- This is a problem
- Different and sometimes strange parameter values can give good fits





Alternative VonB formulation

- Shelton and Mangel (2012) argue that L_∞ and k are inherently correlated, and a better parameterization is $\frac{\partial L(t)}{\partial t} = q - kL(t)$ vs $\frac{\partial L(t)}{\partial t} = k\{L_\infty - L(t)\}$
- They called $q = kL_\infty$ the coefficient of anabolism and k is a coefficient of catabolism

- Shelton, A.O. and Mangel, M., 2012. Estimating von Bertalanffy parameters with individual and environmental variations in growth. *Journal of biological dynamics*, *6*(sup2), pp.3-30. TEAMS

Shelton and Mangel VonB bootstrap

Formula: len ~ lo * exp(-k * age) + q * (1 - exp(-k * age))/k

Parameters:

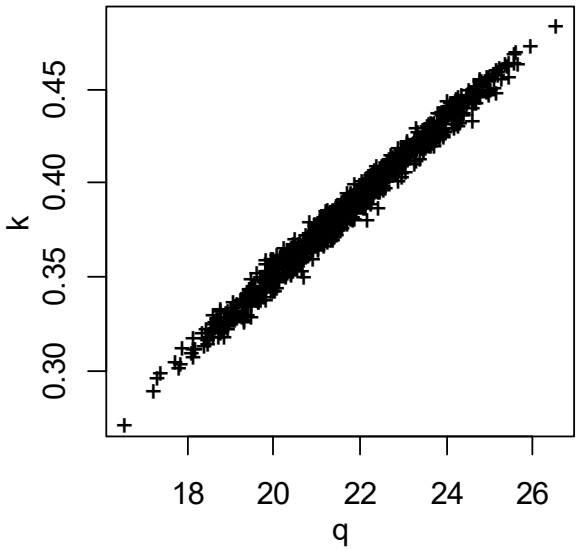
	Estimate	Std. Error	t value	Pr(>|t|)	
q	21.58425	1.85852	11.614	2.75e-06	***
k	0.38558	0.03901	9.885	9.25e-06	***
lo	-3.82312	3.55984	-1.074	0.314	

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.558 on 8 degrees of freedom

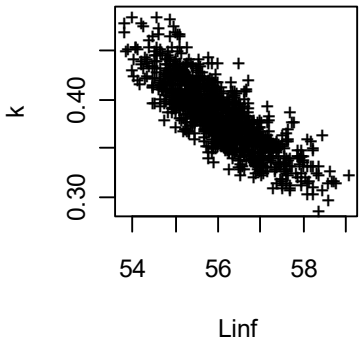
$$x_i(t) = x_{i,0}\,\mathrm{e}^{-k_i t} + \frac{q_i}{k_i}(1 - \mathrm{e}^{-k_i t}).$$

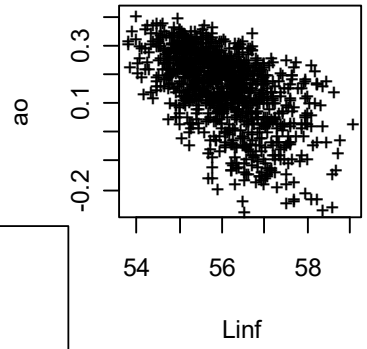
Shelton and Mangel VonB bootstrap

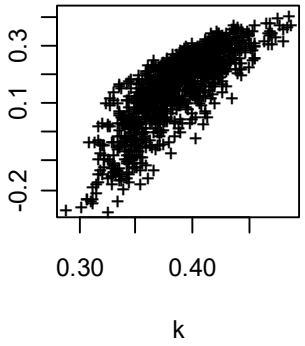


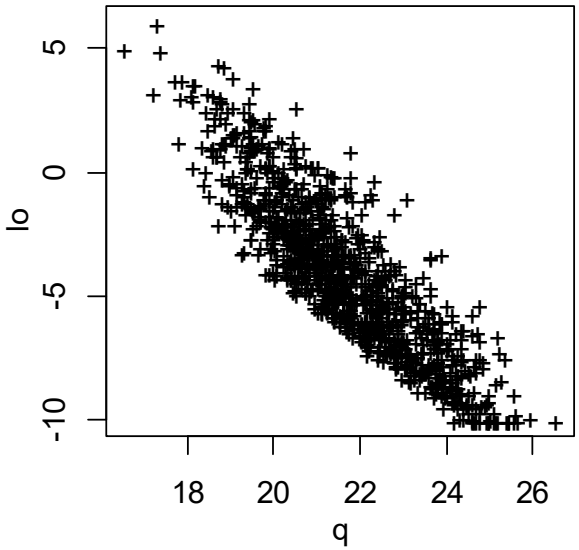
Correlation is Worse!

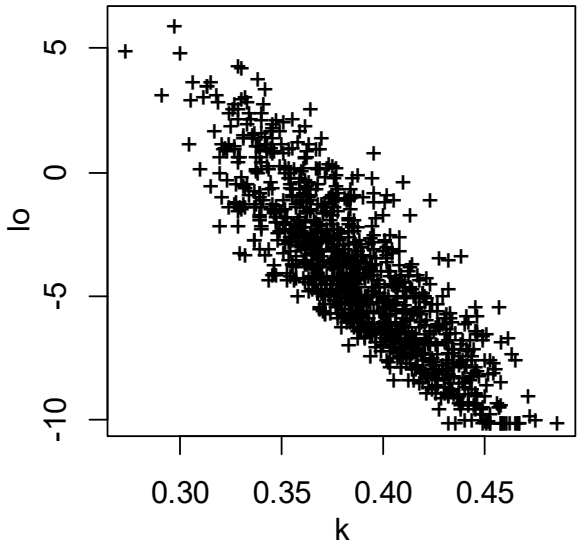
BUNK!











Francis Model

$$L = l_\phi + (l_\psi - l_\phi)(1 - r^{2(T-\phi)/(\psi-\phi)})/(1 - r^2)$$

$$r = (l_\psi - l_\chi)/(l_\chi - l_\phi)$$

In (4) the three usual von Bertalanffy parameters are replaced by l_ϕ, l_χ, and l_ψ, which are mean lengths at ages ϕ, $(\phi+\psi)/2$, and ψ for some arbitrary reference ages ϕ and ψ. (r is used only to simplify the presentation of (4); it is not another parameter.)

$$l_{\infty,a} = l_\phi + (l_\psi - l_\phi)/(1 - r^2)$$

$$k_a = -(2\log_e r)/(\psi - \phi)$$

$$t_0 = \phi + (1/k_a)\log_e((l_{\infty,a} - l_\phi)/l_{\infty,a})$$

Francis, R.I.C.C., 1988. Are growth parameters estimated from tagging and age–length data comparable?. *Canadian Journal of Fisheries and Aquatic Sciences*, *45*(6), pp.936-942. (TEAMS)

Nonparametric

scam {scam} R Documentation

Shape constrained additive models (SCAM) and integrated smoothness selection

Description

This function fits a SCAM to data. Univariate smooths subject to monotonicity, convexity, or monotonicity plus convexity are available as model terms, as well as bivariate smooths with double or single monotonicity. Smoothness selection is estimated as part of the fitting. Confidence/credible intervals are available for each smooth term.

All the shaped constrained smooths have been added to the `gam()` in package `mgcv` setup using the `smooth.construct` function. The routine calls a `gam()` function for the model set up, but there are separate functions for the model fitting, `scam.fit`, and smoothing parameter selection, `bfgs_gcv.ubre`. Any unconstrained smooth available in `gam` can be taken as model terms.

Usage

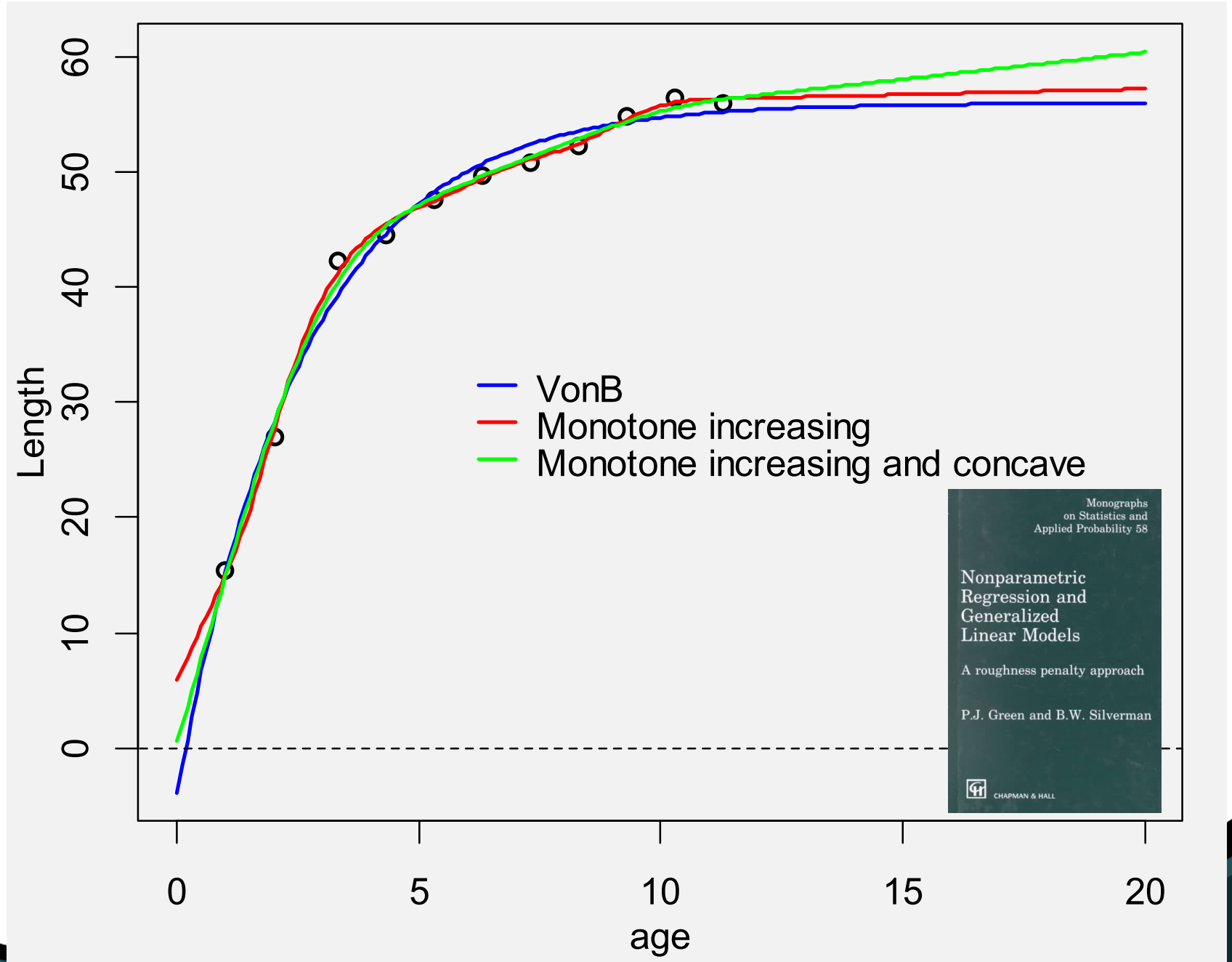
```
scam(formula, family = gaussian(), data = list(), gamma = 1,
      sp = NULL, weights = NULL, offset = NULL,
      optimizer="bfgs", optim.method=c("Nelder-Mead","fd"),
      scale = 0, knots=NULL, devtol = 1e-08, steptol= 1e-8,
      check.analytical=FALSE, del=1e-4, start= NULL, etastart,
      mustart,keepData=FALSE, not.exp=FALSE)
```

- Monotone increasing P-splines `bs="mpi"`. To achieve monotone increasing smooths these reparametrize the coefficients so that they form an increasing sequence. For details see `smooth.construct.mpi.smooth.spec`.
- Monotone decreasing P-splines `bs="mpd"`. To achieve monotone decreasing smooths these reparametrize the coefficients so that they form a decreasing sequence. A first order difference penalty applied to the basis coefficients starting with the second is used for the monotone increasing and decreasing cases.
- Convex P-splines `bs="cx"`. These reparametrize the coefficients so that the second order differences of the basis coefficients are greater than zero. For details see `smooth.construct.cx.smooth.spec`.
- Concave P-splines `bs="cv"`. These reparametrize the coefficients so that the second order differences of the basis coefficients are less than zero. Fo details see `smooth.construct.cv.smooth.spec`.
- Monotone increasing and convex P-splines `bs="micx"`. These reparametrize the coefficients so that the first and the second order differences of the basis coefficients are greater than zero. For details see `smooth.construct.micx.smooth.spec`.

Nonparametric


```
pdata = data.frame(age = seq(0,20,by=0.1))
require("scam")

sfit <- scam(len ~ s(age,bs="mpi"), family=gaussian(link="identity"),data=gdata)
sfit.pred = predict(sfit,newdata=pdata)

sfit1 <- scam(len ~ s(age,bs="micv"), family=gaussian(link="identity"),data=gdata)
sfit1.pred = predict(sfit1,newdata=pdata)

vonb.fit = predict(VonB.fit,newdata=pdata)

ylim = range(sfit.pred,sfit1.pred,vonb.fit)
xlim = range(pdata$age)

par(mar=c(3.5,3,0.5,1),mgp=c(2,1,0))
plot(gdata$age,gdata$len,xlab='age',ylab='Length', type='p',lwd=2,ylim=ylim,xlim=xlim)
lines(pdata$age,vonb.fit,lwd=2,col='blue')
lines(pdata$age,sfit.pred,lwd=2,col='red')
lines(pdata$age,sfit1.pred,lwd=2,col='green')
abline(h=0,lty=2)
legend("center",col=c("blue","red","green"),lty=1,legend=c("VonB","Monotone
increasing","Monotone increasing and concave"),bty='n',lwd=2)
```

Nonparametric



biphasic

Minte-Vera, C.V., Maunder, M.N., Casselman, J.M. and Campana, S.E., 2016. Growth functions that incorporate the cost of reproduction. *Fisheries Research*, *180*, pp.31-44.

As the commonly-used von Bertalanffy growth function (VB) does not explicitly incorporate changes in growth due to allocation of energy to reproduction, a more flexible function could be used when attempting to model juvenile and adult growth simultaneously. Here we review biphasic growth models, with emphasis on those that explicitly incorporate the cost of reproduction, and propose two new models: the von Bertalanffy logistic-L_∞ (VB log-L_∞) and the Cost of Reproduction (CoR) models. We fitted the models to eight data sets from males and females of four unfished or lightly-fished Arctic lake trout (*Salvelinus namaycush*) populations, and compared their fits to those of the commonly-used growth functions. In all

Table 2
Continuous biphasic growth: closed-form models.

Model	Parameters	Equation	Reference
VB-hyperL_∞	5 $\phi = \{L_\infty, t_0, k, h, t_h\}$	$L(a) = L_\infty \left[1 - \mathrm{h}\left((\mathrm{a} - t_h)^2 + 1\right)^{-1}\right] \left\{1 - \exp\left[-k(a - t_0)\right]\right\}$	Soriano et al. (1992)
VB-hyperK	5 $\phi = \{L_\infty, t_0, k, h, t_h\}$	$L(a) = L_\infty \left\{1 - \exp\left[-k\left[1 - \mathrm{h}\left((\mathrm{a} - t_h)^2 + 1\right)^{-1}\right](a - t_0)\right]\right\}$	Soriano et al. (1992)
VB-damped	5 $\phi = \{L_\infty, t_0, k_1, k_2, \lambda\}$	$L(a) = L_\infty \left\{1 - \exp\left[(k_2/\lambda)\left(\exp(-\lambda a) - \exp(-\lambda t_0)\right) - k_1(a - t_0)\right]\right\}$	Porch et al. (2002)
VB-S	6 $\phi = \{L_\infty, t_0, k, v, \lambda, t_m\}$	$L(a) =$ $L_\infty \left\{1 - \exp\left[-k\left((1 - v)(a - t_0) - \left(v/\lambda\right)\left\{\ln(1 + \exp(-\lambda(a - t_m))) - ln(1 + \exp(-\lambda(t_0 - t_m))\right\}\right)\right]\right\}$	Ohnishi et al. (2012)
VB-logK	6 $\phi = \{L_\infty, t_0, k_1, k_2, \alpha, \beta\}$	$L(a) = L_\infty \left\{1 - \exp(-k_2(a - t_0))\left[1 + \exp\left(-\beta(a - t_0 - \alpha)\right)\right)/\left(1 + \exp\left(\beta\alpha\right)\right)\right]^{-(k_2 - k_1)/\beta}\right\}$	Laslett et al. (2004)
VB log-L_∞	6 $\phi = \{L_{1,\infty}, L_{2,\infty}, t_0, k, t_{50}, t_{95}\}$	$L(a) =$ $\left\{L_{1,\infty} + (L_{2,\infty} - L_{1,\infty})\left[1 + \exp\left((-\ln(19)(a - t_{50}))/(t_{95} - t_{50})\right)\right]^{-1}\right\}\left\{1 - \exp\left[-k(a - t_0)\right]\right\}$	This study

$$L(a) = \left\{L_{1,\infty} + (L_{2,\infty} - L_{1,\infty})\left[1 + \exp\left((-\ln(19)(a - t_{50}))/(t_{95} - t_{50})\right)\right]^{-1}\right\}\left\{1 - \exp\left[-k(a - t_0)\right]\right\}$$

Growth in weight

- The weight of an animal is often well approximated as a power function, $W = aL^b$ and $\log(W) = \log(a) + b*\log(L)$
- This can be expressed as a function of age or time by substituting the VonB equation for length

$$W(t) = a\{L(t)\}^b$$

$$= a\left[L_\infty\left\{1 - e^{-\kappa(t-t_o)}\right\}\right]^b$$

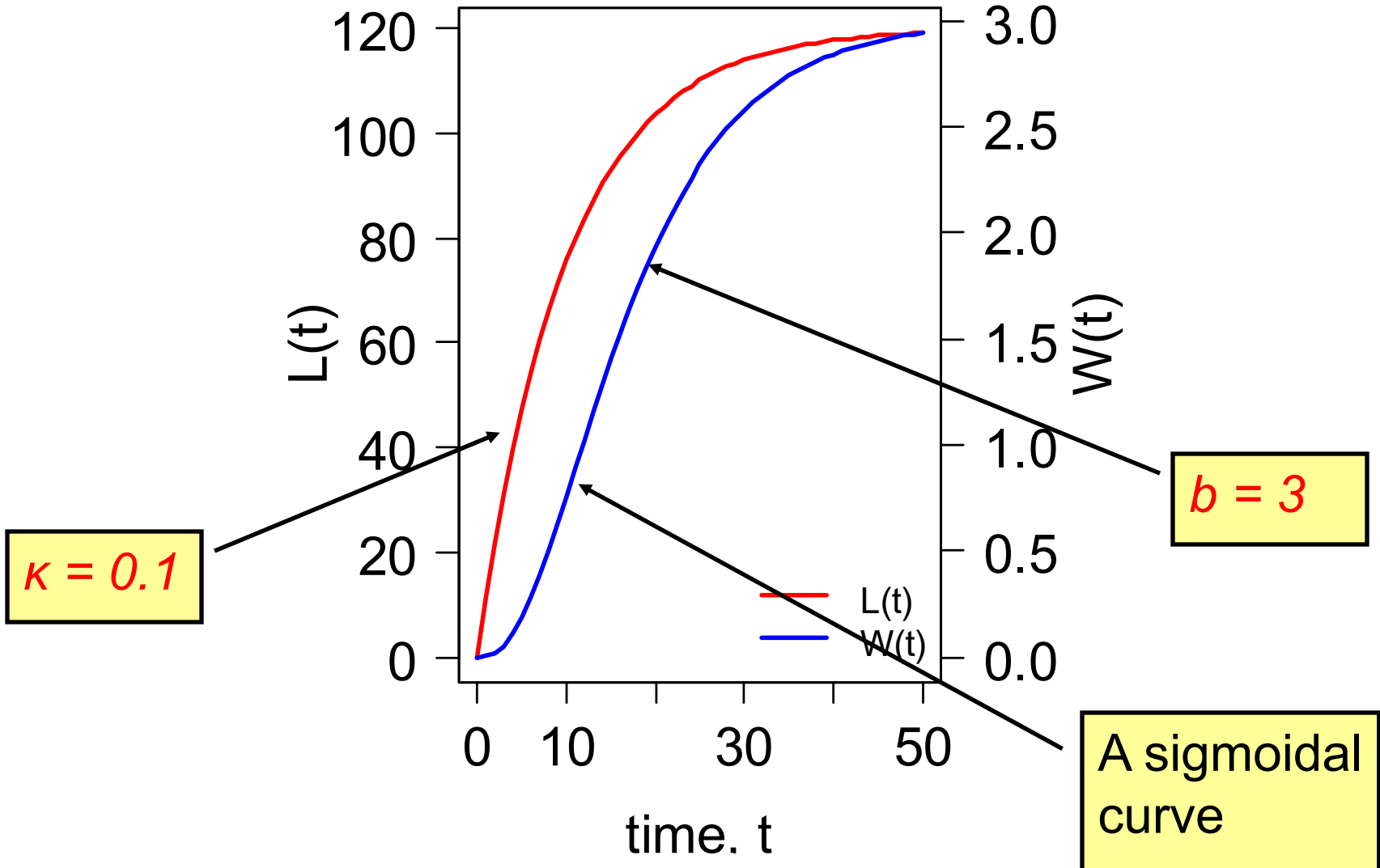
$$= aL_\infty^b\left\{1 - e^{-\kappa(t-t_o)}\right\}^b$$

$$= w_\infty\left\{1 - e^{-\kappa(t-t_o)}\right\}^b$$

b is often set at 3 (Haddon)

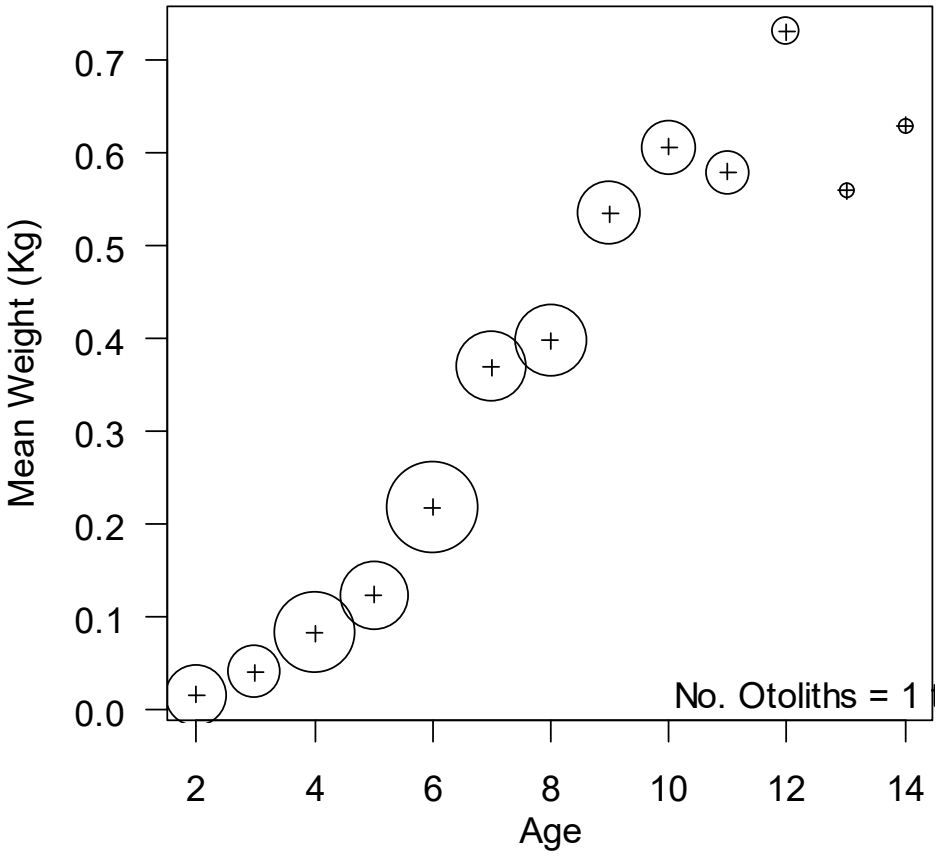
Asymptotic weight

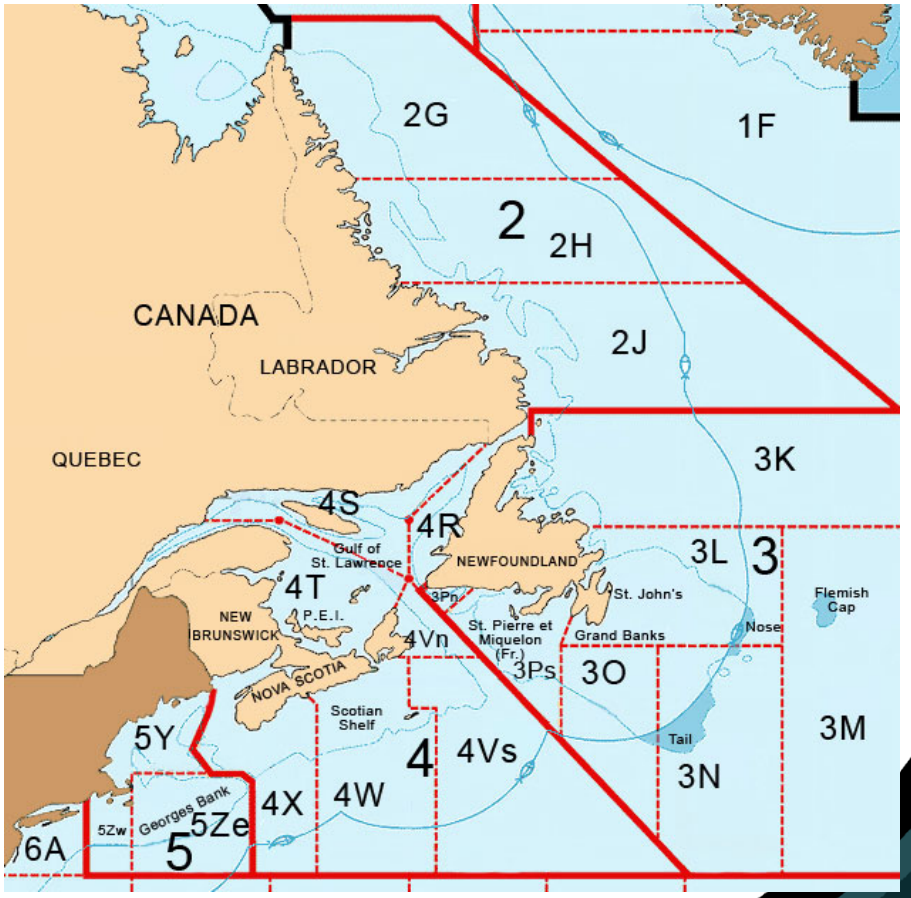
Von Bertalanffy (VonB) growth model



American plaice data

Mean weights at age (W@A) from survey in NAFO Div. 2J in 2010





Weighted Regression

- If the variance of W@A is σ^2 and there are n observations contributing to mean weight-at-age (mW@A), then the variance of the mW@A is σ^2/n.
- Should estimate VonB parameters using weighted regression, to minimize $SSR=\Sigma_i n_i(O_i-P_i)^2$
- Same as MLE with pdf for the i'th observation

$$\frac{1}{\sqrt{2\pi\sigma^2/n_i}}\exp\left\{-\frac{(O_i-P_i)^2}{2\sigma^2/n_i}\right\}$$

Weighted Regression

Can do with using nls() with weights = No. otoliths

```
>   VonB.fit.wt <- nls(meanW ~ Winf*((1 - exp(-k*(age-ao)))**3.0),algorithm="port",
+           start = list(Winf=0.7, k = 0.4,ao=2), lower=c(0,0,-5),upper=c(2,1,5),
+           data=gdata,weights=otolith)
>
> summary(VonB.fit.wt)
```

Formula: meanW ~ Winf * ((1 - exp(-k * (age - ao)))^3)

Parameters:

	Estimate	Std. Error	t value	Pr(>|t|)	
Winf	0.81520	0.10778	7.564	1.92e-05	***
k	0.27424	0.05719	4.795	0.000729	***
ao	2.01957	0.54347	3.716	0.004001	**

--- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: **0.1305** on **10** degrees of freedom

```
Waiting for profiling
2.5% 97.5%
Winf 0.655 1.219
k      0.160 0.391
ao     0.440 2.864
```

Verify using nlminb()


```
> normal.nll <- function(parm){
+   Winf = exp(parm[1])
+   k = exp(parm[2])
+   ao = exp(parm[3])
+   s.hat = exp(parm[4])
+   mu.hat = Winf*((1 - exp(-k*(gdata$age-ao)))**3.0)
+   prob = dnorm(gdata$meanW,mean=mu.hat,sd=s.hat/sqrt(gdata$otolith))
+   nll = -sum(log(prob))
+   return(nll)
+ }
>
> parm.start = log(c(0.8,0.27,2,0.14))
> mle.fit = nlminb(parm.start,normal.nll,control=list(maxit=1000,trace=1),
+ lower=c(-5,-5,-2,-5),upper=c(10,10,10,10))
  0:    -23.372034: -0.223144 -1.30933 0.693147 -1.96611
 15:    -24.376288: -0.204327 -1.29376 0.702891 -2.16785
 16:    -24.376288: -0.204323 -1.29376 0.702886 -2.16785
> print(exp(mle.fit$par),digits=5)
[1] 0.81520 0.27424 2.01957 0.11442
> print(exp(mle.fit$par[4])*sqrt(n/(n-3)),digits=5)
[1] 0.13046
```

	Estimate
Winf	0.81520
k	0.27424
ao	2.01957

Residual standard error: **0.1305** on **10** degrees of freedom

Unweighted Regression


```
>   VonB.fit <- nls(meanW ~ Winf*((1 - exp(-k*(age-ao)))**3.0),algorithm="port",
+         start = list(Winf=0.7, k = 0.4,ao=2), lower=c(0,0,-5),upper=c(2,1,5),
+         data=gdata)
>
> summary(VonB.fit)
```

Formula: meanW ~ Winf * ((1 - exp(-k * (age - ao)))^3)

Parameters:

	Estimate	Std. Error	t value	Pr(>|t|)	
Winf	0.69144	0.05360	12.899	1.48e-07	***
k	0.35791	0.07815	4.580	0.00101	**
ao	2.50053	0.65249	3.832	0.00331	**

--- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: **0.0536** on **10** degrees of freedom

Formula: meanW ~ Winf * ((1 - exp(-k * (age - a

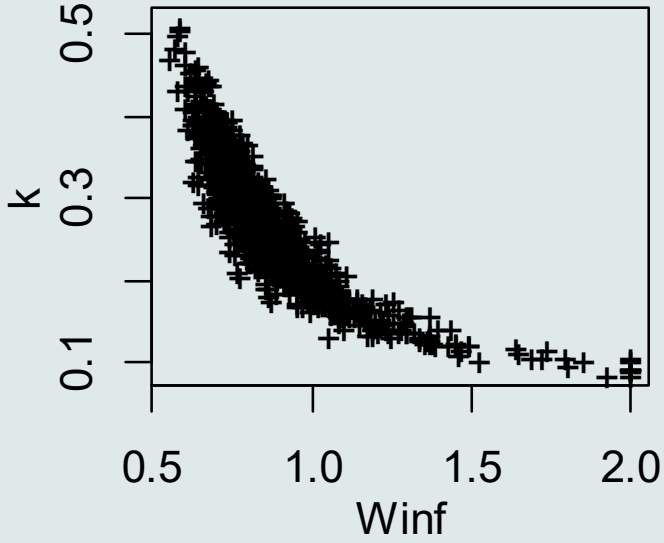
Parameters:

	Estimate	Std. Error	t
Winf	0.81520	0.10778	
k	0.27424	0.05719	
ao	2.01957	0.54347	

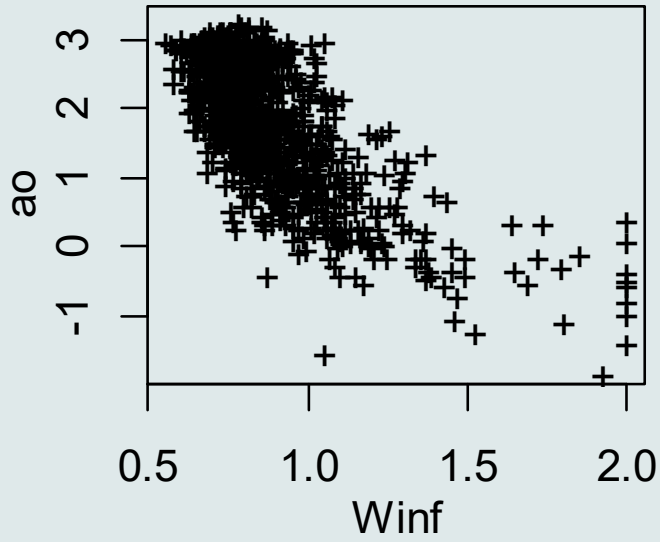
--- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05

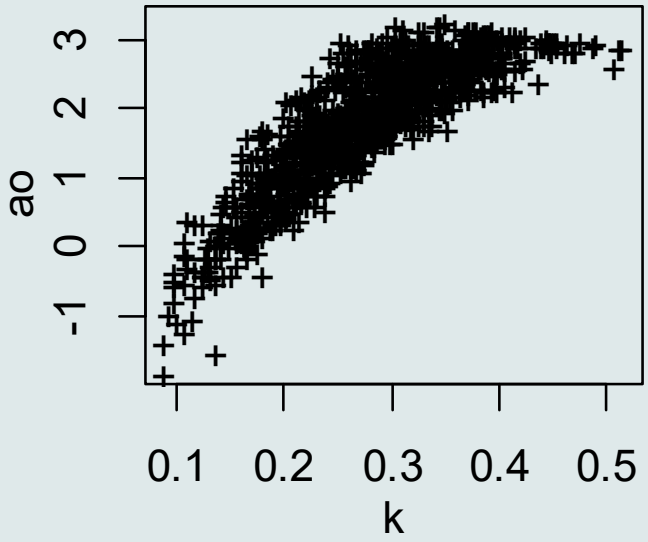
Residual standard error: **0.1305** on **10** degrees

Weighted Bootstraps



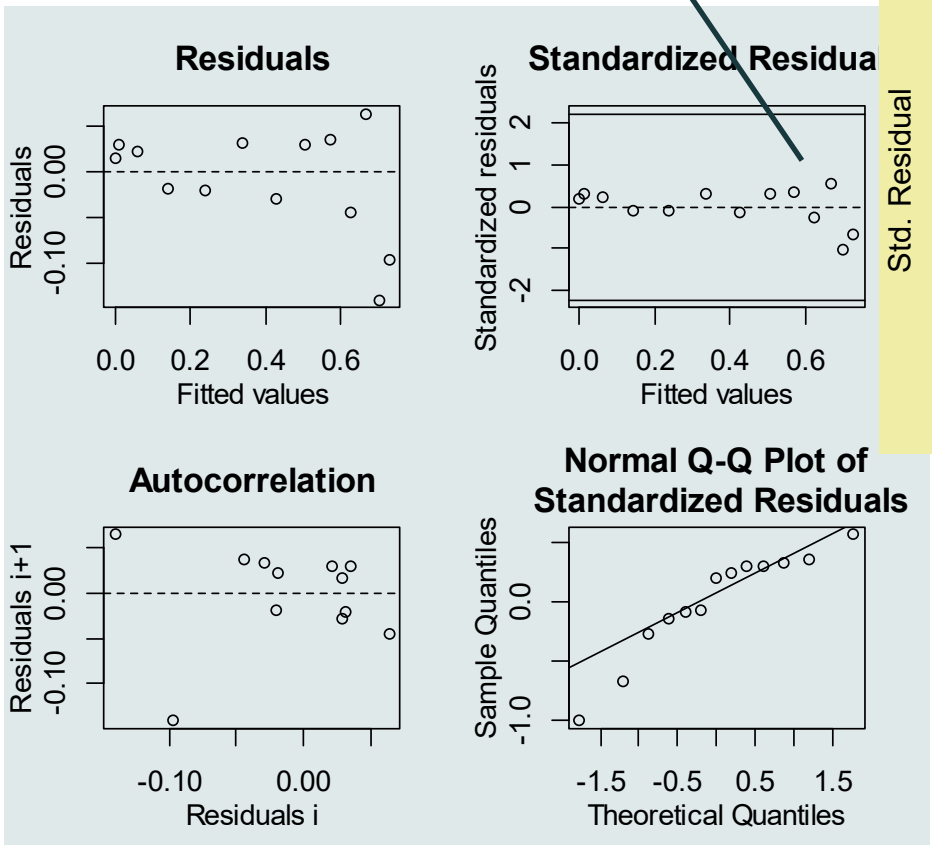
UGHH

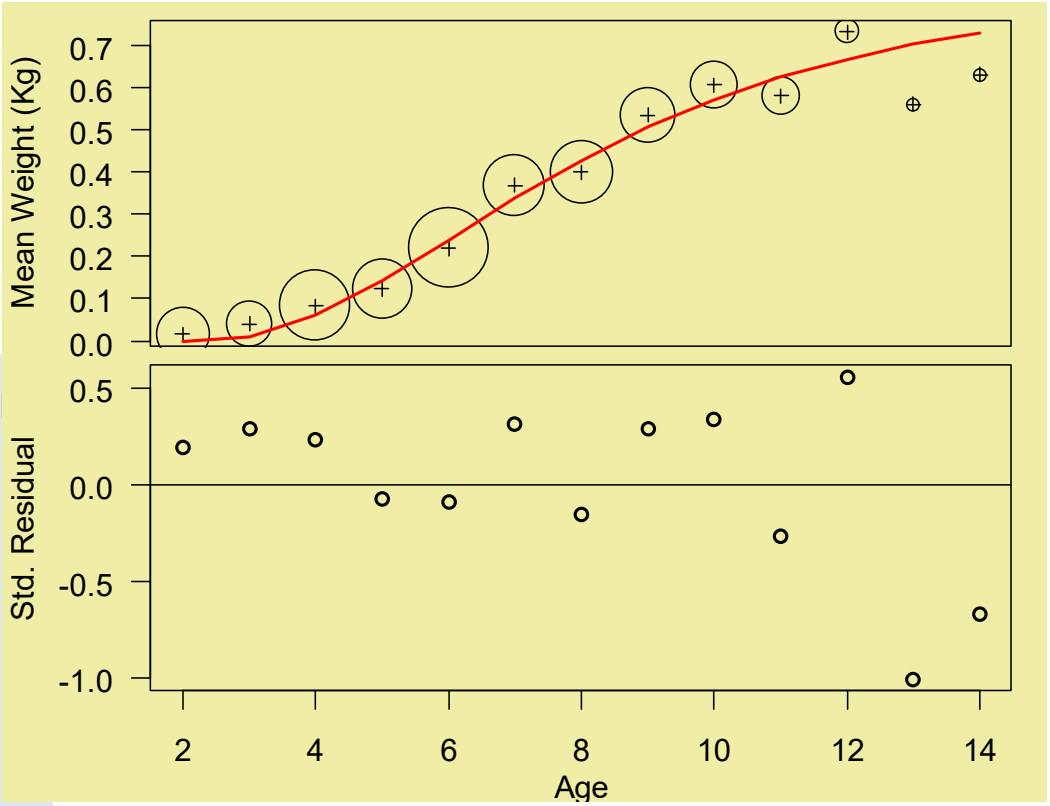




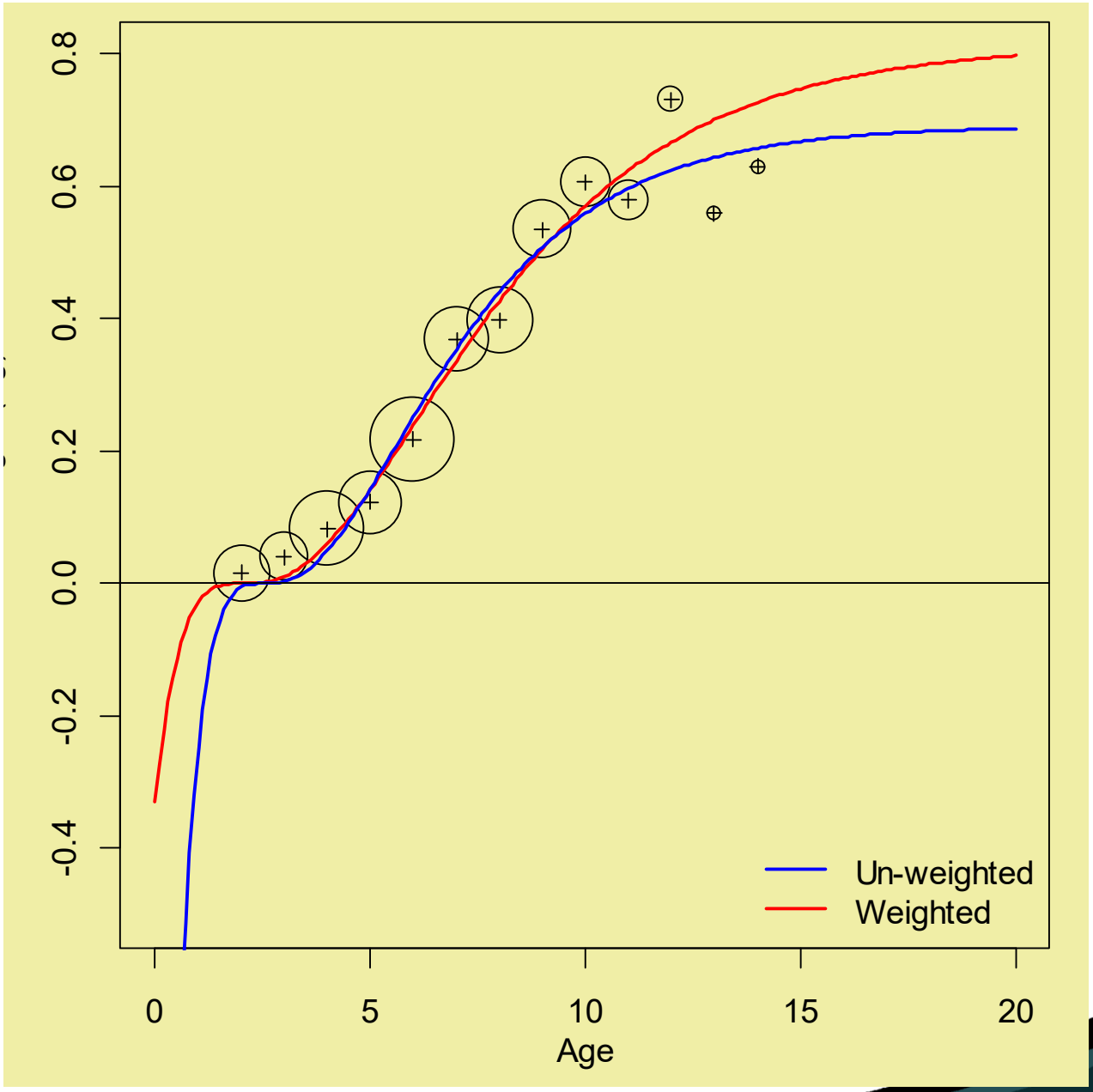
Weighted Fit

Some evidence of increasing variation

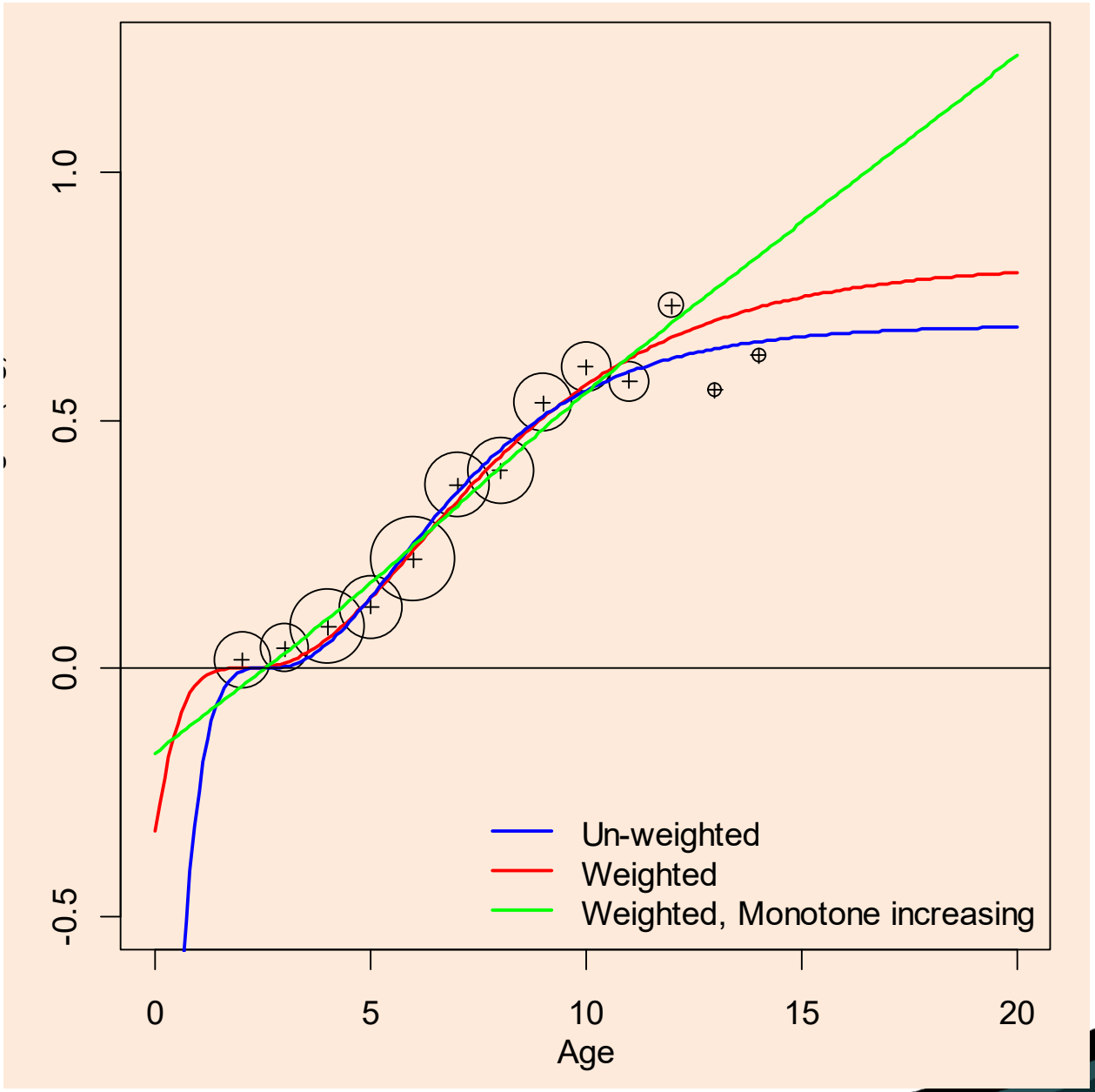




Weighted vs Unweighted fits



+Scam() fits


1.0
0.5
0.0
-0.5
0
5
10
15
20
Age
Un-weighted
Weighted
Weighted, Monotone increasing

Weighted Lognormal Regression


```
VonB.LNfit.wt <- nls(log_meanW ~ log_Winf + 3*log((1 - exp(-k*(age-ao)))),algorithm="port",
        start = list(log_Winf=-0.4, k = 0.3,ao=0), lower=c(-2,-2,-2),upper=c(log(20),1,1.999),
        data=gdata,weights=otolith)
```

Formula: log_meanW ~ log_Winf + 3 * log((1 - exp(-k * (age - ao))))

Parameters:

	Estimate	Std. Error	t value	Pr(>|t|)	
log_Winf	0.83923	0.44362	1.892	0.08781	.
k	0.09986	0.02484	4.020	0.00244	**
ao	-0.06154	0.24561	-0.251	0.80724	

Weighted normal-errors regression

	Estimate	Std. Error	t value
Winf	0.81520	0.10778	7.564
k	0.27424	0.05719	4.795
ao	2.01958	0.54347	3.716

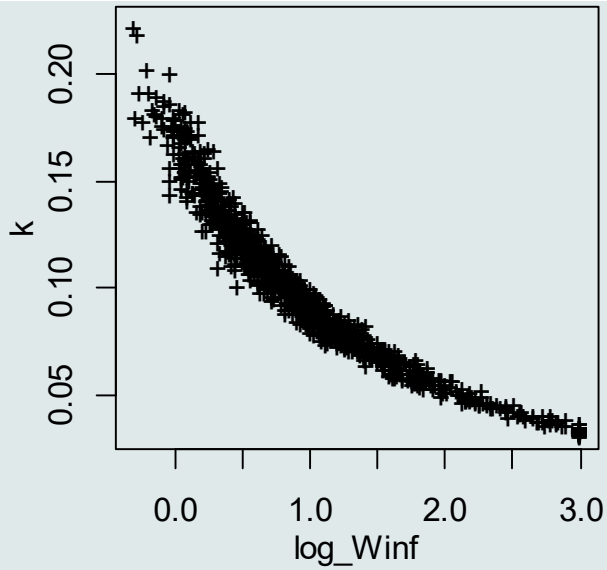
Winf = 2.315

LogNormal Winf estimate much larger than Normal Winf

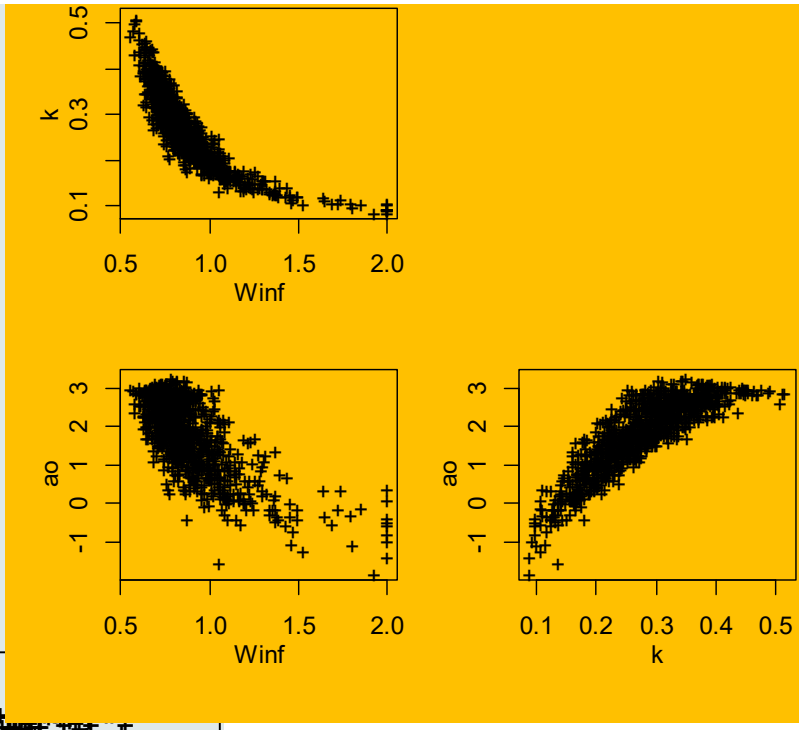
--- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

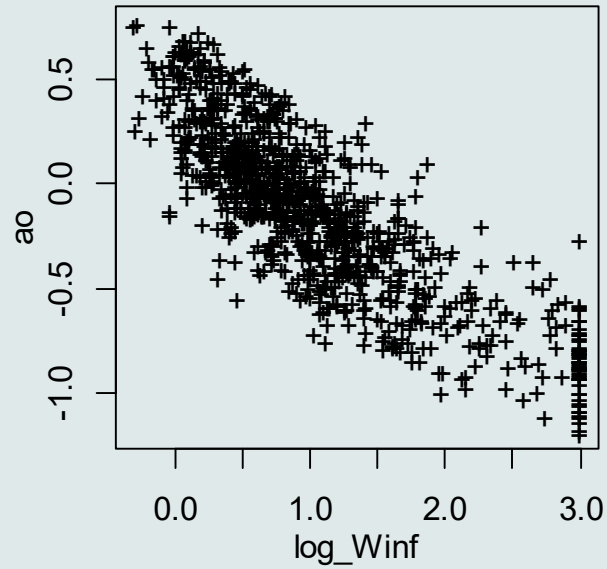
Residual standard error: **0.484** on **10** degrees of freedom

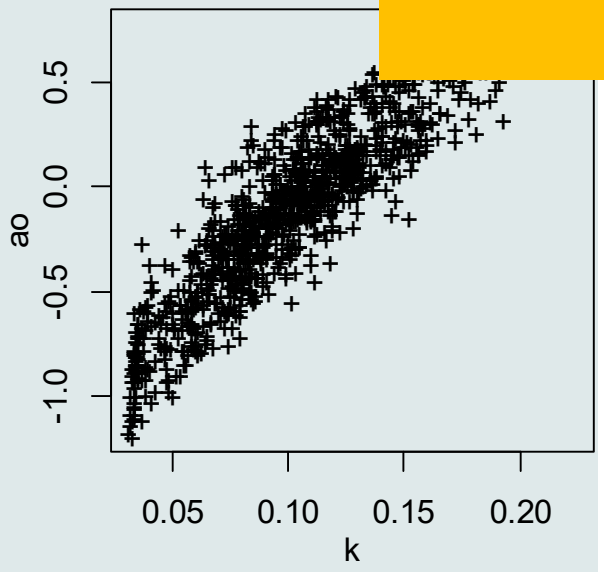
Weighted Lognormal Bootstraps



also UGHH

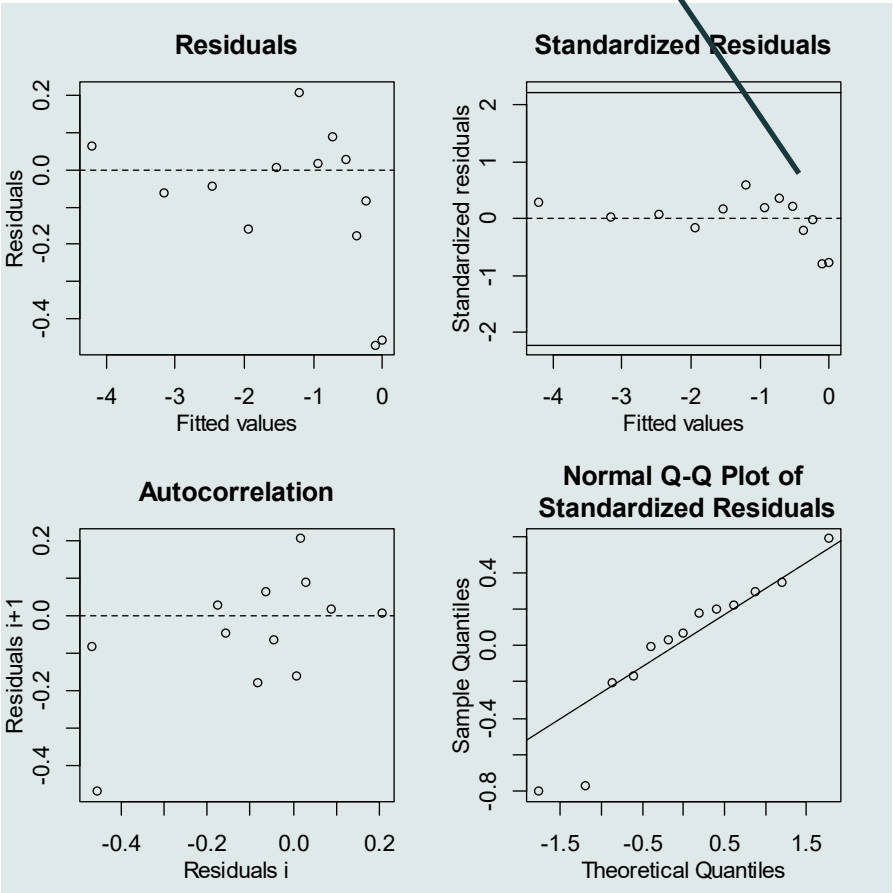


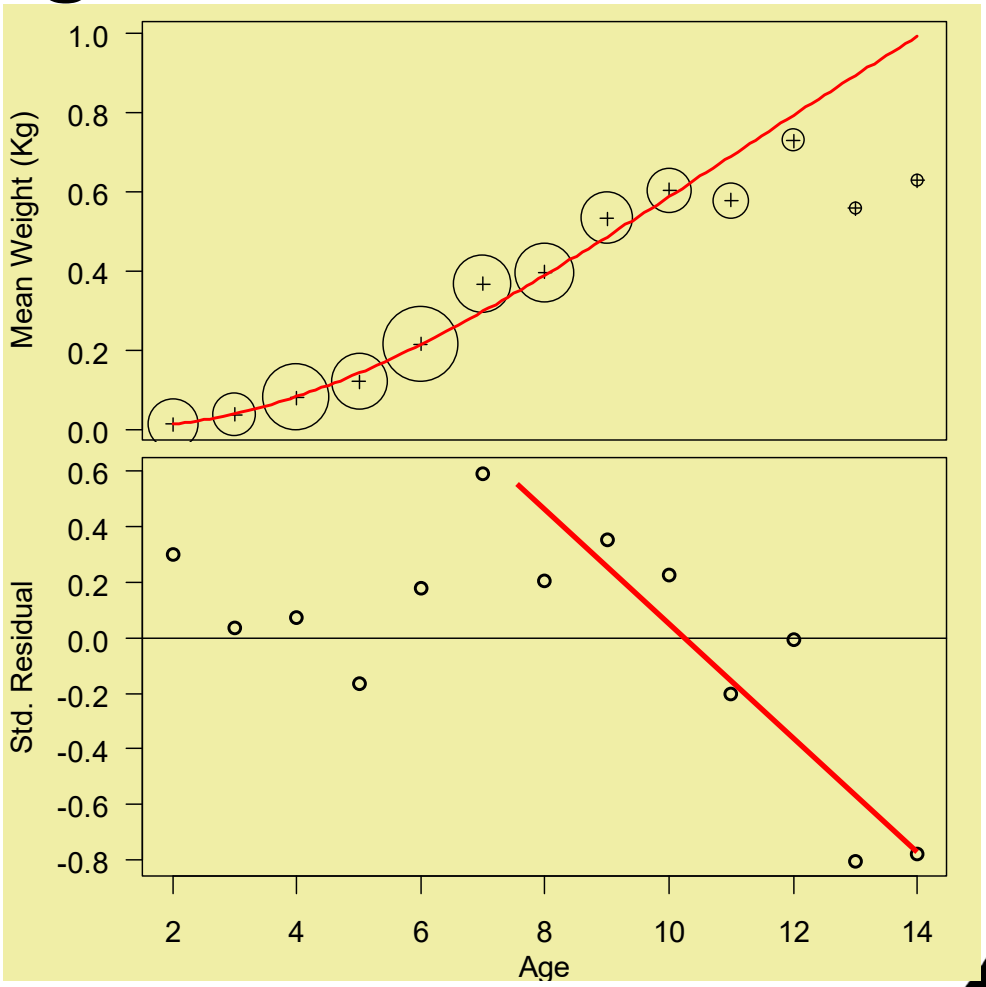




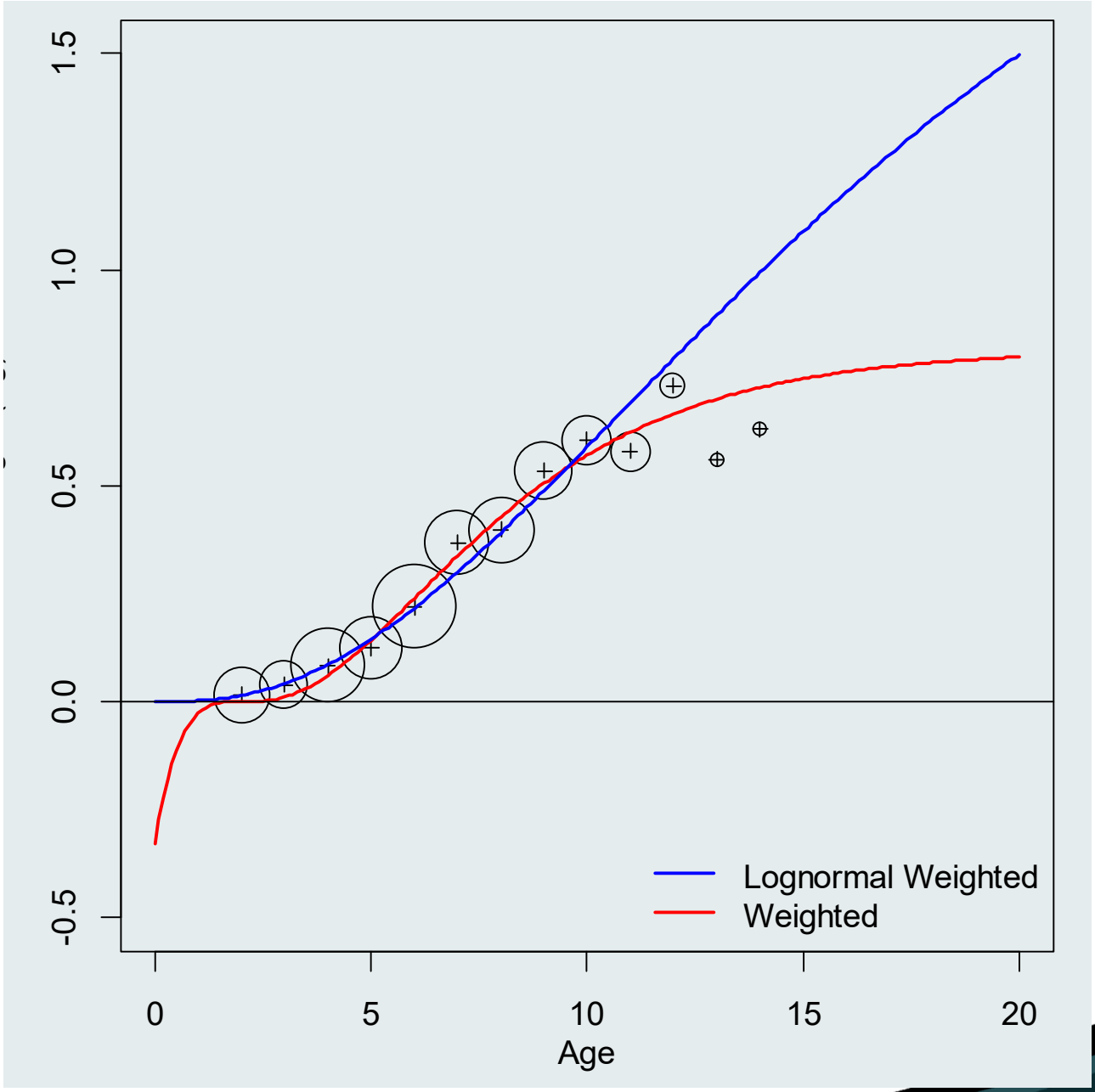
Weighted Lognormal Fit

Not improved





Normal (Least Squares) vs Lognormal fits



Gompertz Growth Model

- Based on $\frac{\partial W(t)}{\partial t} = \lambda e^{-kt} W(t)$
- A Solution:

$$W(t) = W_\infty \exp\{\rho \exp(-ka)\}, \rho = \log\left\{\frac{W(0)}{W_\infty}\right\}$$

- A sigmoidal growth curve
- To my knowledge this is only used for W@A (e.g. Quinn and Deriso, 1995).

Gompertz Results


```
GOMP.fit.wt <- nls(meanW ~ Winf*exp(rho*exp(-k*age)),algorithm="port",
        start = list(Winf=0.7, k = 0.4,rho=-10), lower=c(0,0,-Inf),upper=c(2,1,-2),
        data=gdata,weights=otolith)
```

Formula: meanW ~ Winf * exp(rho * exp(-k * age))

Parameters:

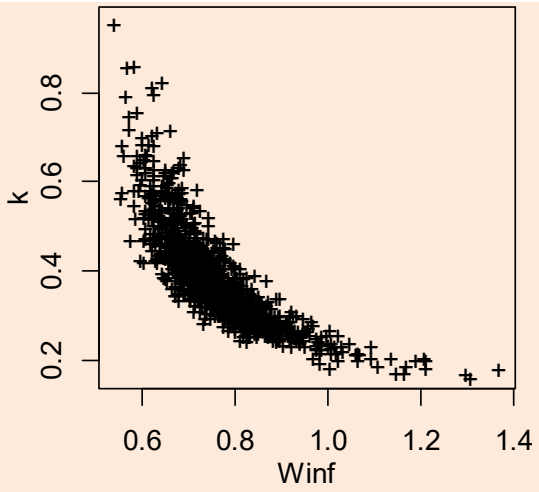
	Estimate	Std. Error	t value	Pr(>|t|)	
Winf	0.74716	0.06973	10.715	8.41e-07	***
k	0.37233	0.05854	6.360	8.24e-05	***
rho	-10.84835	3.09461	-3.506	0.00567	**

	Estimate	Std. Error	t value
Winf	0.81520	0.10778	7.564
k	0.27424	0.05719	4.795
ao	2.01958	0.54347	3.716

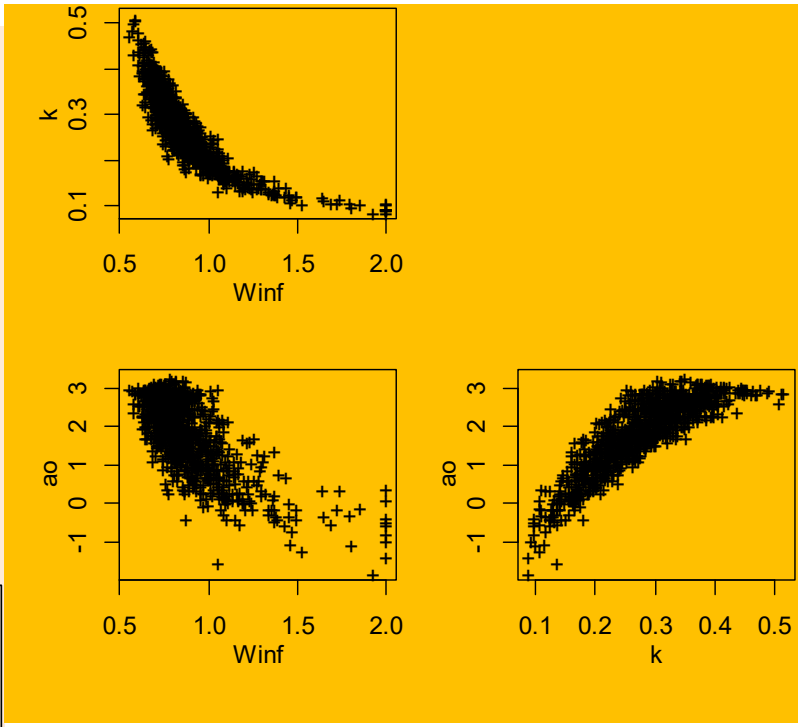
--- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

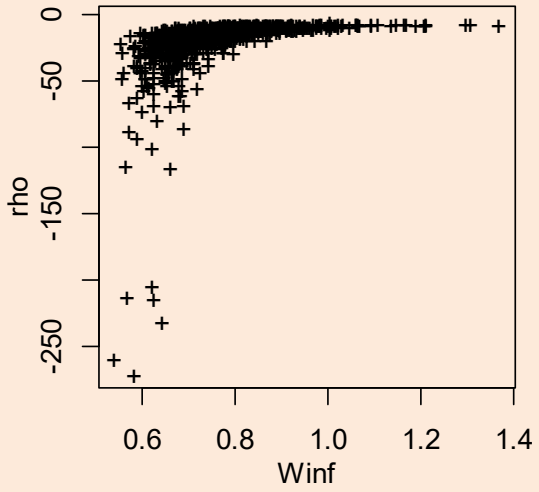
Residual standard error: **0.1192** on **10** degrees of freedom

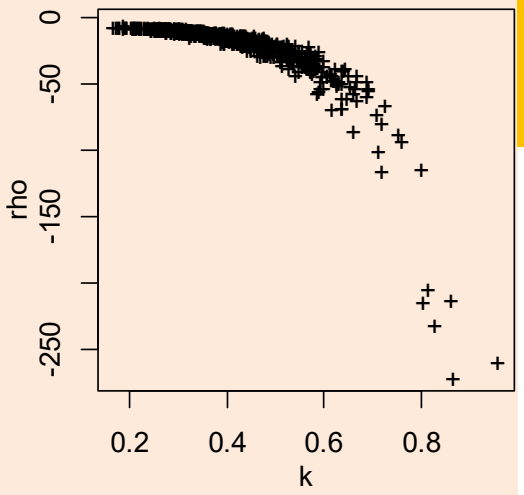
Gompertz Bootstraps



also UGHH

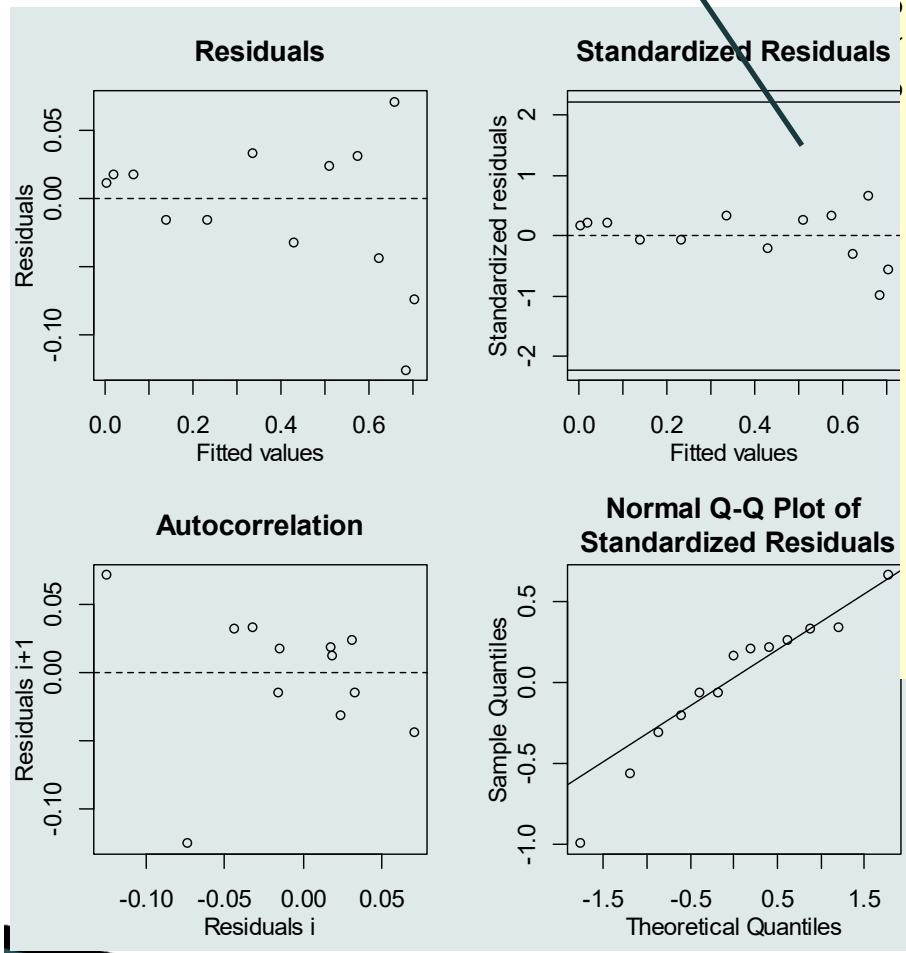


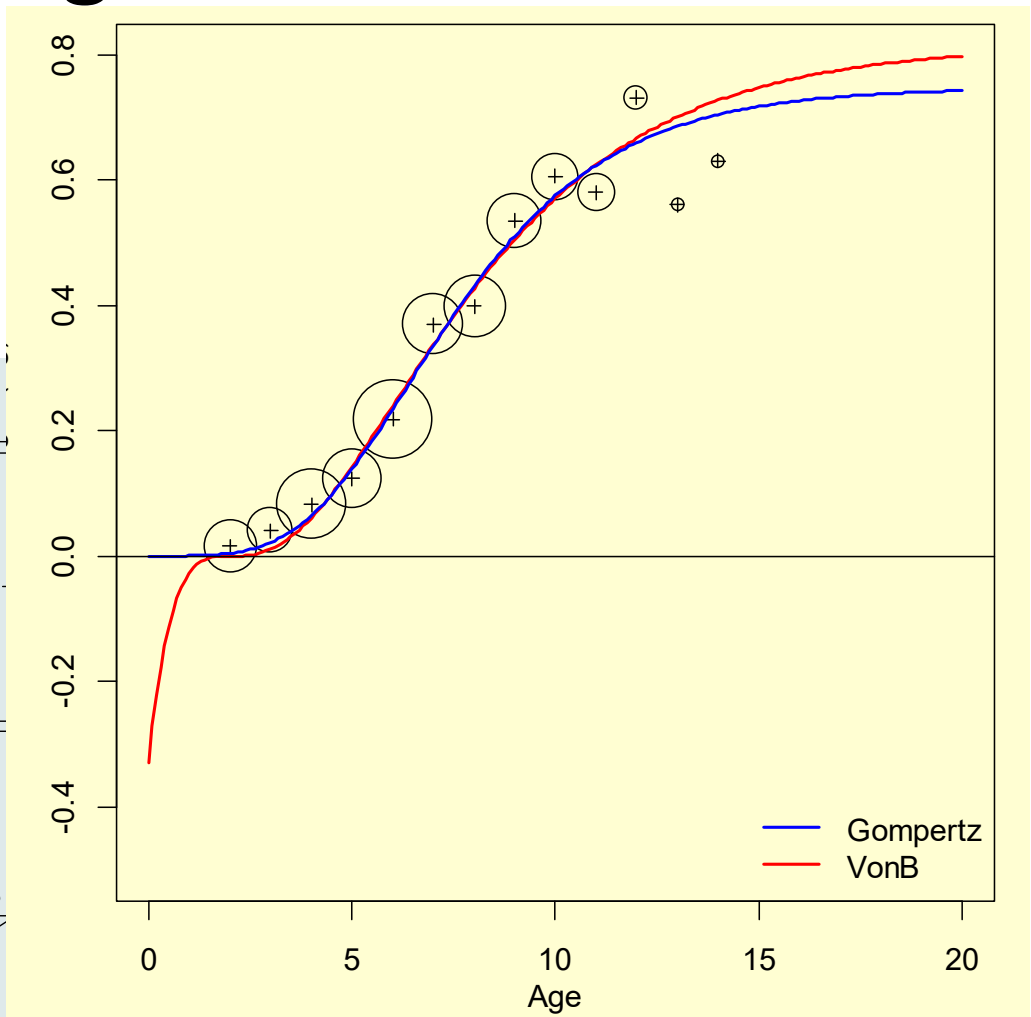




Weighted Lognormal Fit

Similar to VonB





Seasonal Growth

- Common in northern waters (or southern in Southern hemisphere)
- Also common when prey availability varies greatly in different seasons
- Several models have been published that replace the VonB *k* growth rate parameter with some type of oscillating function of time
- See Haddon for examples

Seasonal Growth – why bother

- It can be useful to understand this when deciding the best time to harvest
- The YPR could change during the year
- Also, catch numbers are often inferred from catch weights, and if growth rates vary seasonally then this should be accounted when deriving catch numbers

Seasonal Growth

– Instead of $L(t) = L_\infty \left(1 - e^{-\kappa t}\right)$

– Replace the linear (in time) growth rate model *k x t*

– With a more general function of time, *K(t)*

$$L(t) = L_\infty \left\{1 - e^{-K(t)}\right\}$$

– eg

$$K(t) = C \sin\left(\frac{2\pi(t-s)}{52}\right) + k \times t$$

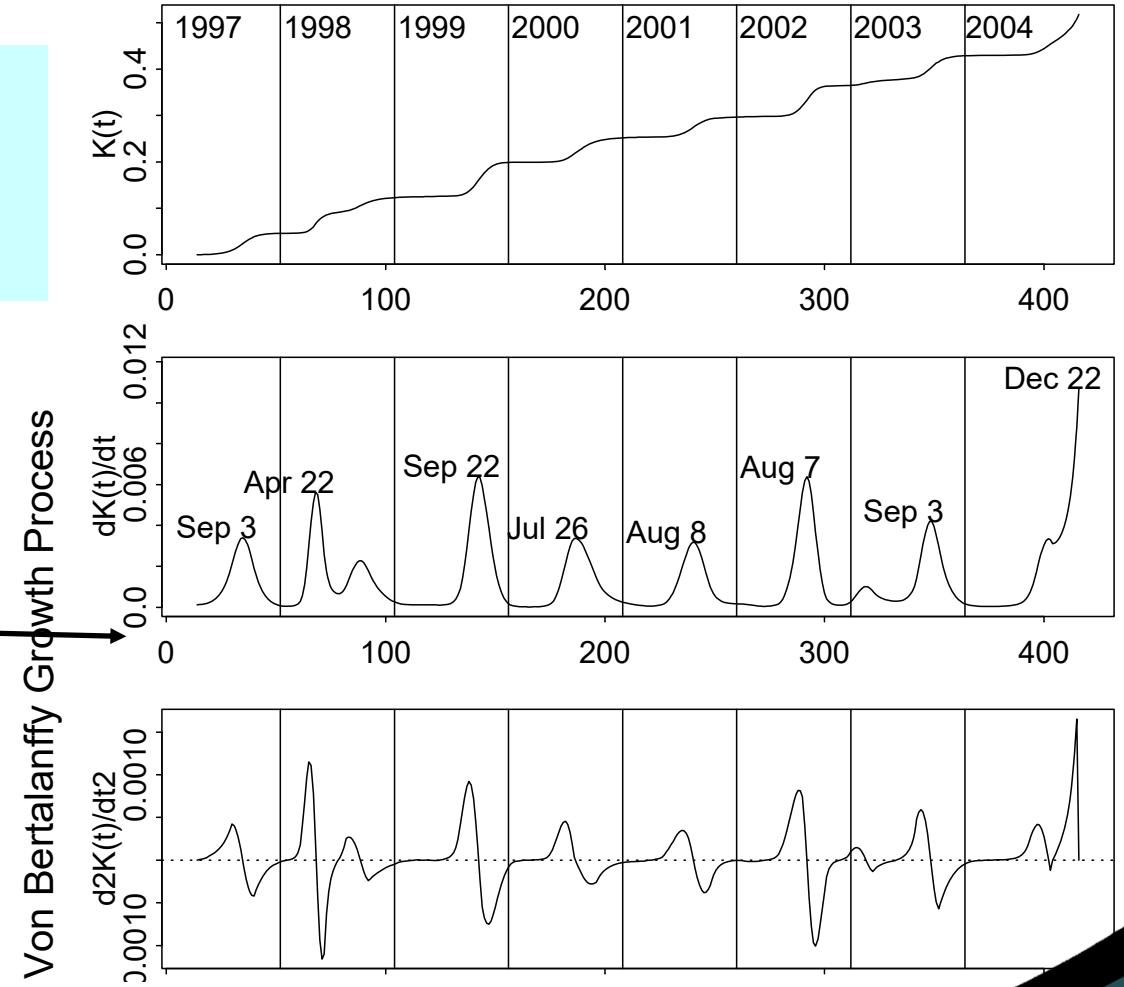
Size of osscillations

Timing of osscillations

Seasonal growth in cod

– A nonparametric approach

Cadigan, N.G. and Brattey, J. 2001. A nonparametric Von Bertallanffy model for estimating growth curves of Atlantic cod. ICES-CM-2001/O:18



Weeks since 1997

Growth curves and tagging data

- The cod work was based on tagging data
- So far we have been looking at the growth of one fish over time, or the average change in size with age for a population
- Fisheries tagging data involves the capture of fish (usually dead) that were previously tagged

Growth curves and tagging data

- With tagging data you get 2 length measurements from many fish:
- length at capture,
- the growth increment (capture length minus release length),
- and time at liberty
- Usually don’t know ages

Faben's growth model for tagging data

- $G(\Delta t) = L(t+\Delta t)-L(t)$ denote the growth increment
- Faben's model is

$$G(\Delta t) = \left(L_\infty - L_t\right)\left(1 - e^{-k\Delta t}\right)$$

- This model can be derived directly from the VonB model (see Appendix 8.1 in Haddon)

Faben's growth model for tagging data

- Tagging growth data is of the form $(G_1,L_1,\Delta t_1),\ldots,(G_n,L_n,\Delta t_n)$, where
- L is the length-at-release, assumed to be known without error (i.e. a covariate)
- Δt is the time between release and capture (time-at-libery)
- G is the growth increment – which is the response variable measured with error.

Faben's growth model for tagging data

- The statistical model is

$$G_i = \left(L_\infty - L_i\right)\left(1 - e^{-k\Delta t}{}_i\right) + \varepsilon_i$$

- Usually it is assumed that $\varepsilon \sim N(0,\sigma^2)$.
- The model parameters and their confidence intervals can be estimated in a similar manner to the VonB model.

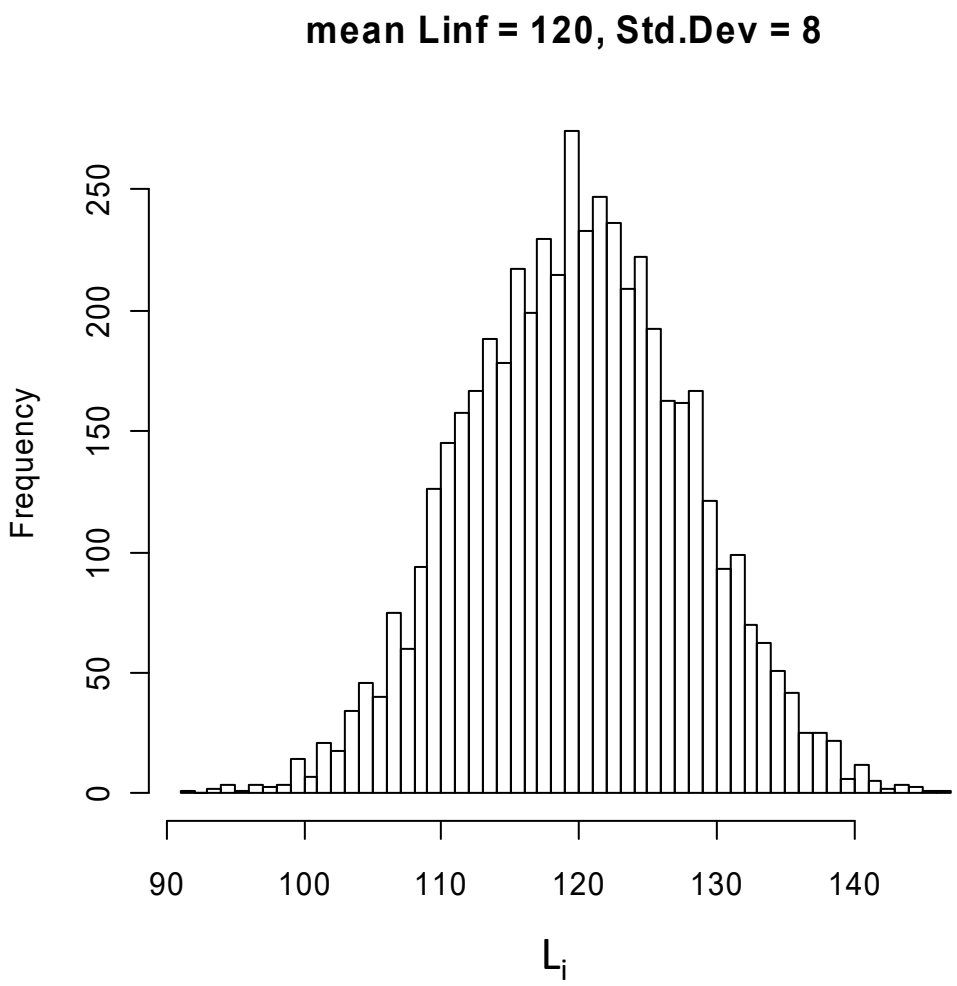
Between individual variation

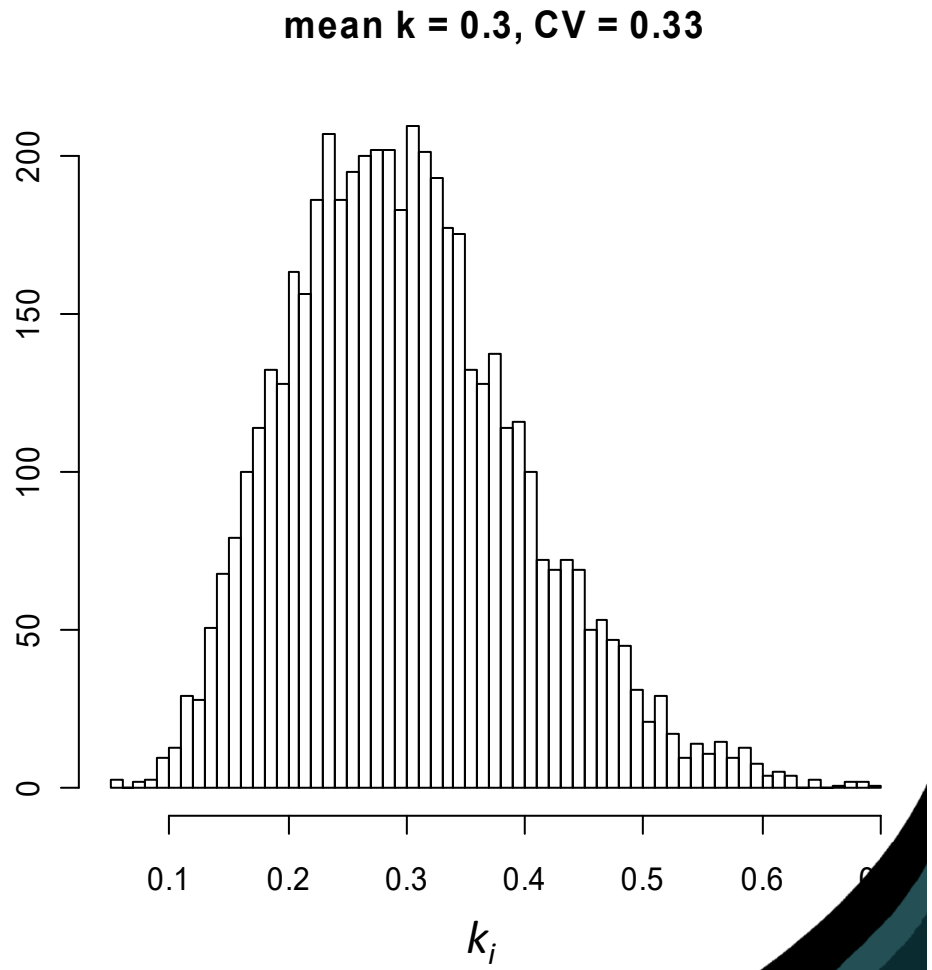
- A problem when applying models to growth data collected from many individuals is between-individual variation in growth rates.
- This is not just an issue of getting the variance right.
- between-individual variation can lead to biased estimates of the average growth rates for the population

Simulation example of bias

- I simulated *n=5000* age and length measurements
- For each individual, $L_{i\infty}$ ~ *N(mean=120,sd=8)* and k_i ~ *Gamma* (see plot)
- Based on a distribution of ages, I generated lengths using the random L_i's and k_i's.

Simulation example of bias



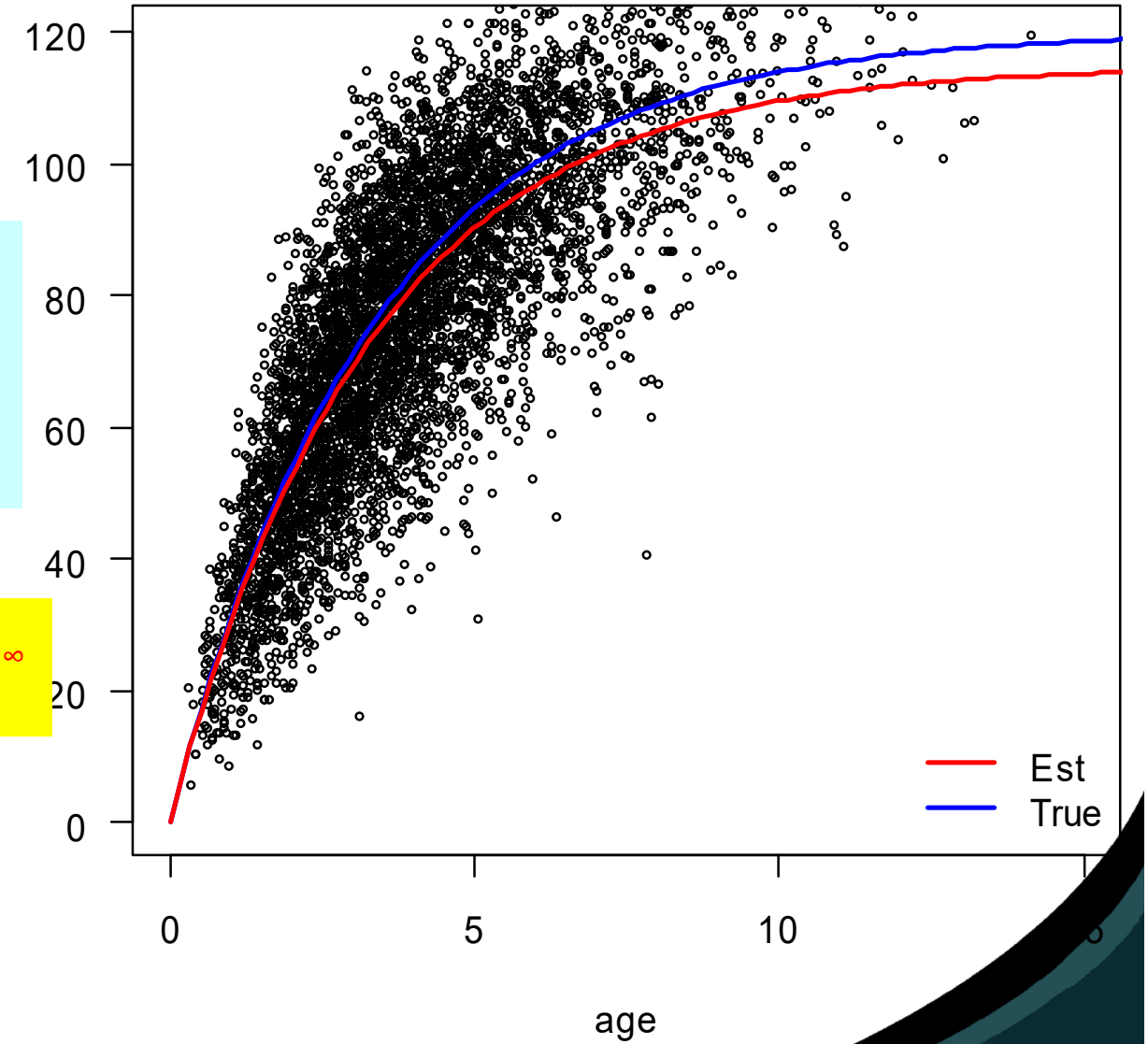


Simulation example of bias

Waiting for profiling to be done...

	2.5%	97.5%
Linf	113.1716959	116.7132172
k	0.2982256	0.3181731

The profile likelihood CI for L_∞ does not cover true value



Simple VonB Case

- Lognormal measurement error in length, and lognormal between-individual variation in

$$L_{i\infty} = L_{\infty} e^{\delta_i},\ \delta_i \sim N(0, \sigma_{\infty}^2)$$

- This VonB extension is

$$L_t = L_{i\infty}\{1 - e^{-k(t-t_o)}\}e^{\varepsilon_i}, \varepsilon_i \sim N(0, \sigma_{\varepsilon}^2)$$

$$L_t = L_{\infty}\{1 - e^{-k(t-t_o)}\}e^{\delta_i + \varepsilon_i}$$

- $Var(\delta_i + \varepsilon_i) = \sigma_{\infty}^2 + \sigma_{\varepsilon}^2$
- Completely confounded sources of variation

Less Simple VonB Case

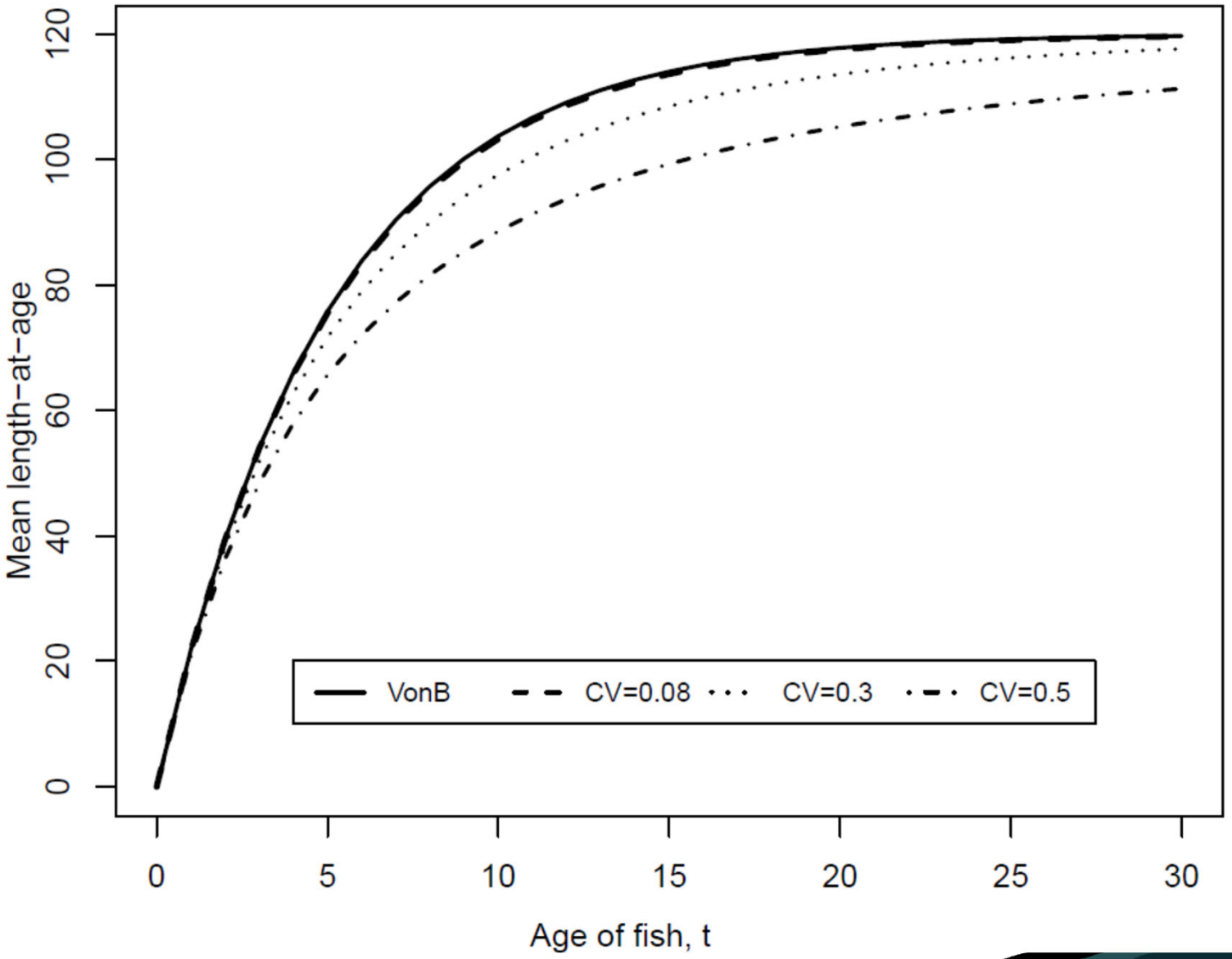
- If k is random: $K_i \sim Gamma$, $E(K_i) = k$, $Var(K_i) = \sigma_k^2$, but L_∞ not random and no measurement error in length. $CV = \sigma_k / k$
- This VonB extension is

$$L(t) = L_\infty\{1 - e^{-K_i\,(t-t_o)}\}$$

$$E_K\{L(t)\} = L_\infty\left[1 - \frac{1}{\{1 + CV^2 k(t - t_o)\}^{CV^{-2}}}\right]$$

$$\lim_{CV \to 0} E\{L(t)\} = VonB?$$

Less Simple VonB Case

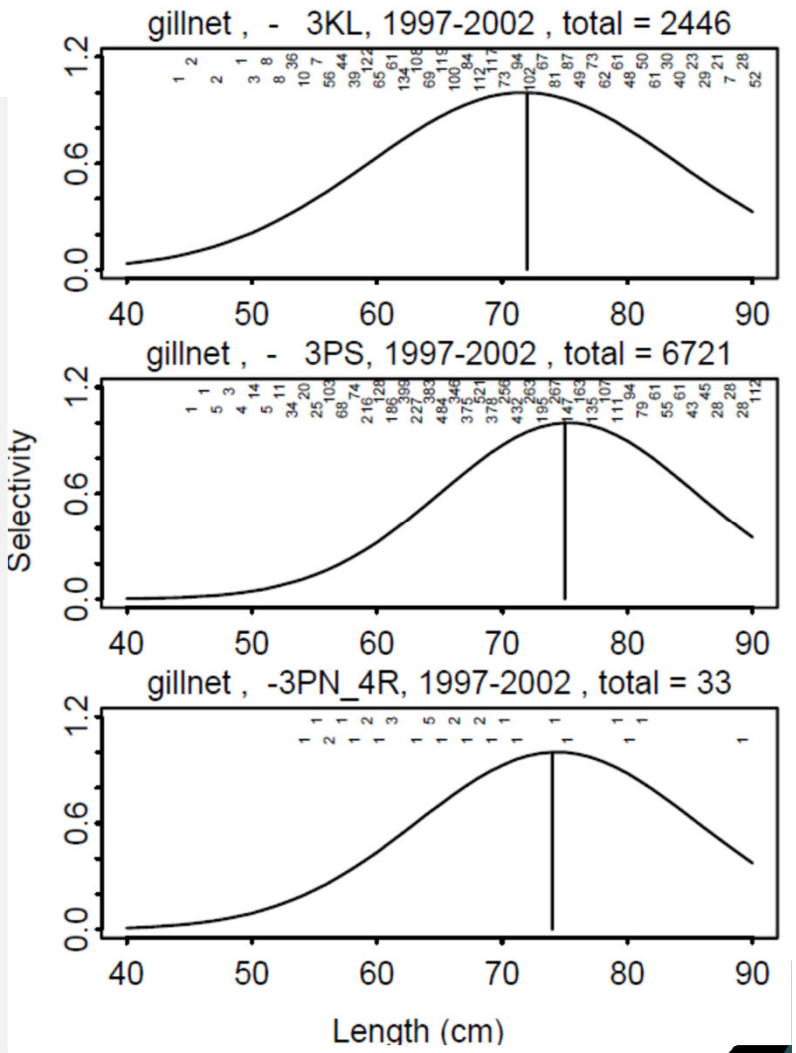


Between individual variation

- Many approaches have been proposed to deal with between-individual variation in growth rates.
- The problem may be worse with tagging data.
- e.g. one does not expect a 10 year old 50cm fish to grow as much in the next year as a 5 year old 50cm fish – but we don't know ages?
- There is no single best method to deal with this problem, but many papers!
- A topic for F6005!

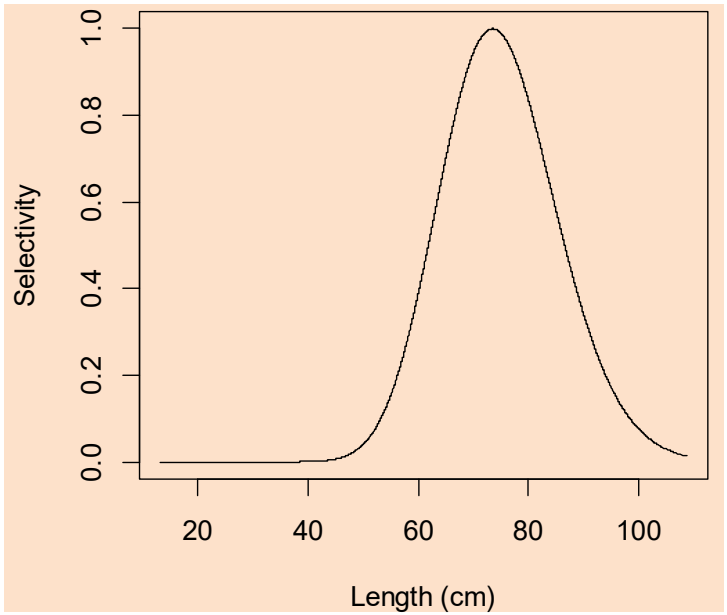
Gear Selectivity

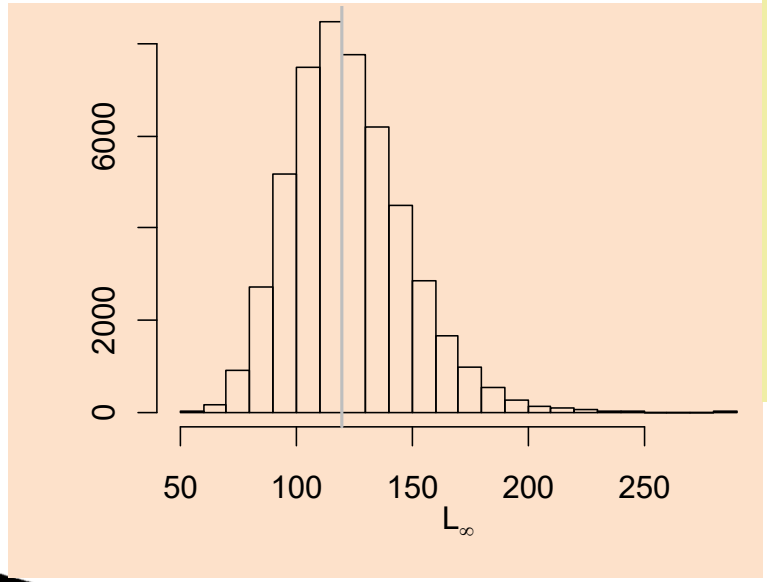
- Causes a complication when there is between individual variation
- For example, if fish are sampled using a gillnet (i.e. cod Sentinel program) with domed-shaped selectivity
- Then you will sample fast growing young fish and slow growing old fish – **a biased sample!!**

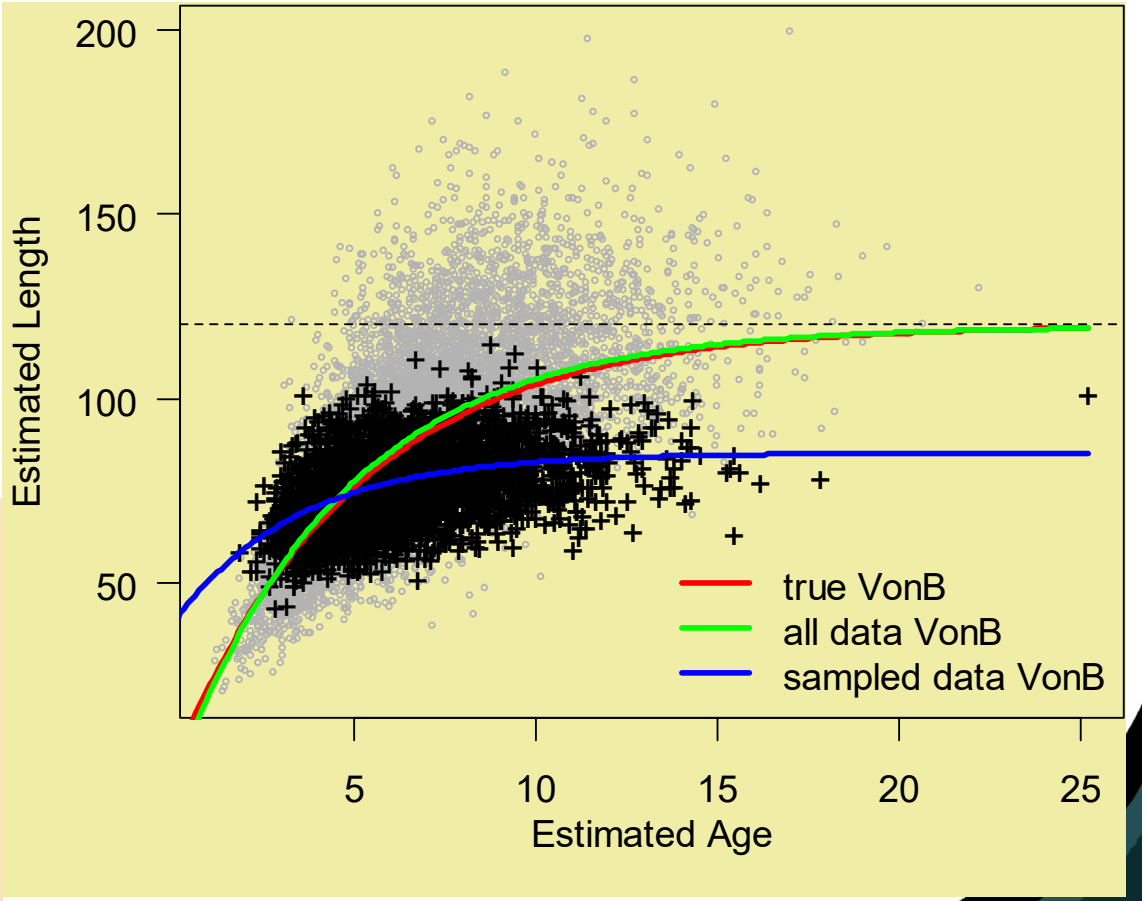


Cadigan, N.G. and Brattey, J., 2003. *Analyses of stock and fishery dynamics for cod in 3Ps and 3KL based on tagging studies in 1997-2002*. CSAS Res. Doc. 2003/037

Gear Selectivity Simulation



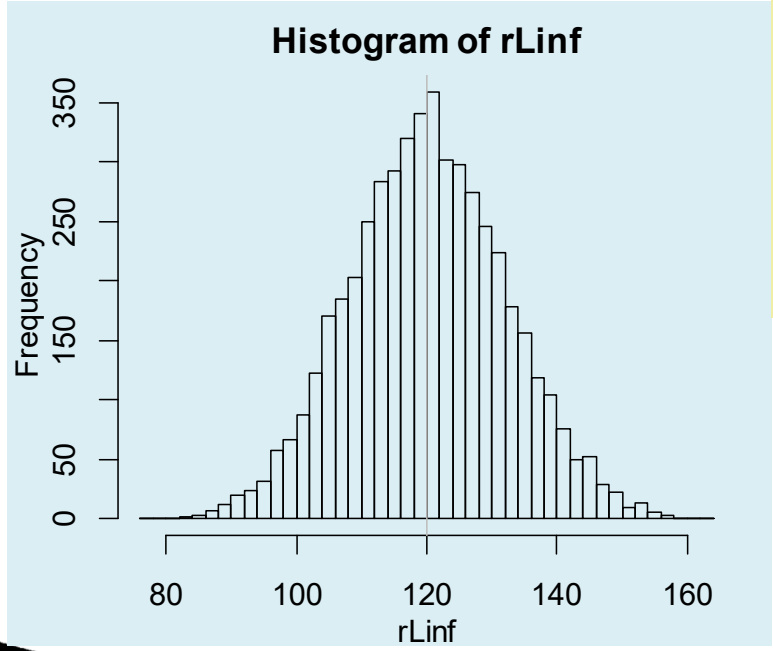


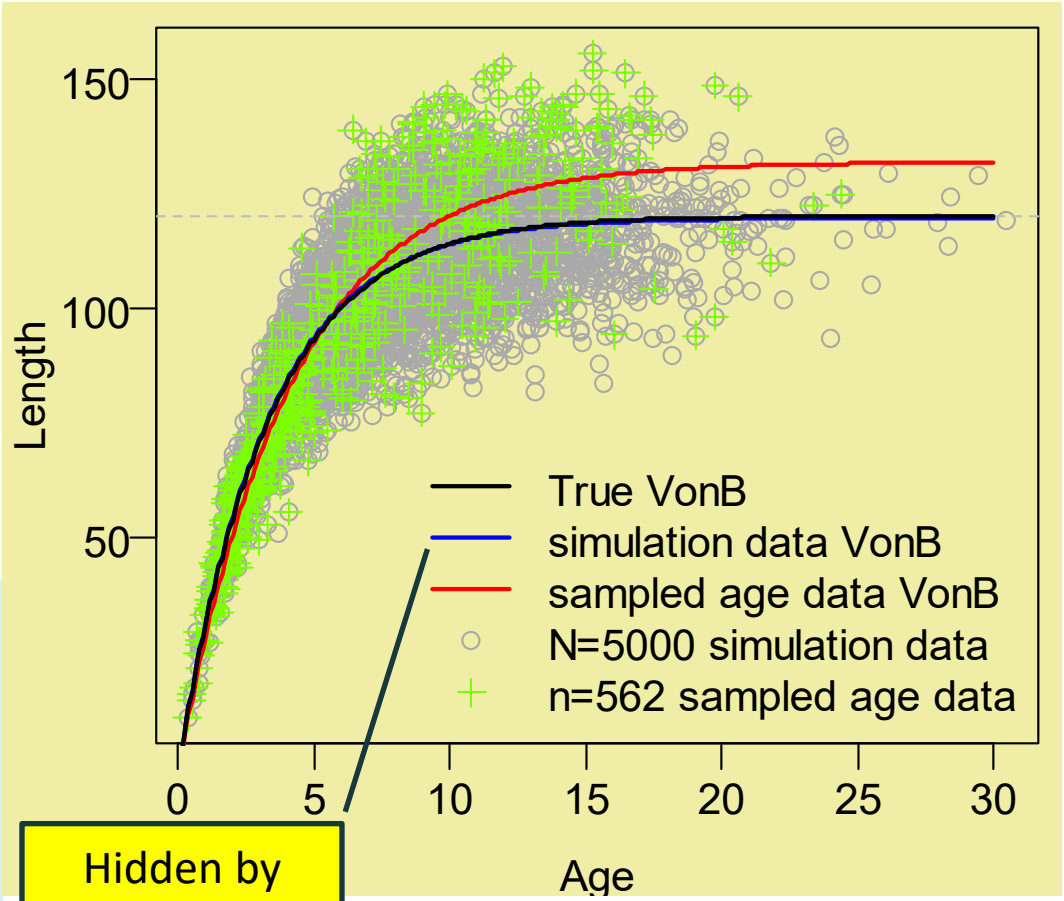


Length-stratified age samples

- Another issue is sampling design
- Length-stratified age sampling: sample a fixed number of fish per length bin for age determination
- To ensure a broad range of ages to estimate growth curves
- But you get a biased sample!

Length-Stratified: 5 per 1cm





Hidden by black curve

Age measurement error

- In addition to between-individual variation in growth rates, selectivity, and length-stratified age sampling
- Measurement error in age is sometimes a problem.
- Under-estimate L_∞ and over-estimate k.
- A topic for F6005!

Alternatives

- The VonB model will not always fit well.
- Haddon discusses some alternative models
- The key is that a growth model has to be monotonic increasing
- and we usually want growth rates to decline with size or age

Testing for differences in growth curves

- Haddon gives a lot of information for this problem.
- The simplest situation is where there are two populations, and you wish to test if their growth rates differ.
- A good strategy is to use a likelihood ratio test

Testing for differences in growth curves

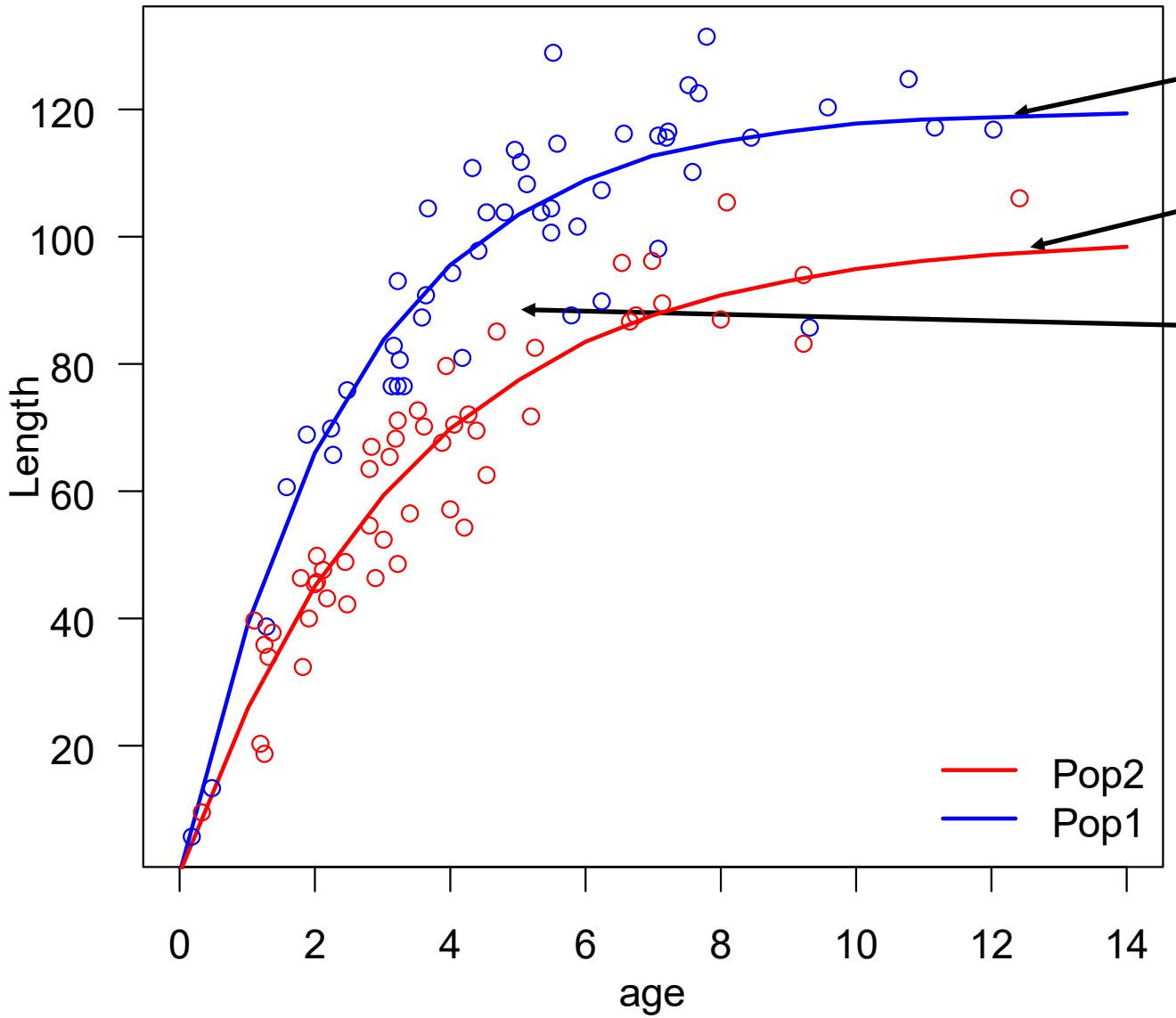
- Find the nll for each population, then get the nll for the combined population (only 1 VonB curve)
- *LR = 2{nll(combined) – nll(1) – nll(2)}* can be used to test the hypothesis that the growth rates differ.
- If *LR* > the upper 1-α quantile from a Chi-square distribution then you reject the null hypothesis that the growth rates are the same for both populations

Degrees of freedom = number of parameters in (1) + (2) – number in combined

Testing for differences in growth curves

- Another useful approach is to use an indicator variable for population, and estimate VonB parameters for Pop1, and the difference in VonB parameters for Pop2 and Pop1

Simulation example



$L_\infty = 120, k = 0.4$

$L_\infty = 100, k = 0.3$

n=50, for each Population

Simulation example

```
gdata=data.frame(age=age,len=len,pop=pop)
gdata$ind=0
gdata$ind[gdata$pop==2]=1
```

```
VonB.fit <- nls(len ~ (Linf1 + dLinf2*ind)*(1-exp(-(k1 + dk2*ind)*age)),
        algorithm="port",data=gdata,
        start = list(Linf1=50,dLinf2=0,k1=0.2,dk2=0), lower=c(0,-60,0,-0.3))
```

dLinf2 is $L_{\infty 2} - L_{\infty 1}$.

dk2 is $k_2 - k_1$.

Simulation example results

Formula: len ~ (Linf1 + dLinf2 * ind) * (1 - exp(-(k1 + dk2 * ind) * age))

Parameters:

	Estimate	Std. Error	t value	Pr(>|t|)	
Linf1	117.32653	2.38538	49.186	< 2e-16	***
dLinf2	-21.29692	3.87173	-5.501	3.13e-07	***
k1	0.46847	0.02897	16.170	< 2e-16	***
dk2	-0.13273	0.03834	-3.462	0.000802	***

--- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: **7.196** on **96** degrees of freedom

$L_{\infty 2} - L_{\infty 1}$ is significantly different from 0

$k_2 - k_1$ is significantly different from 0

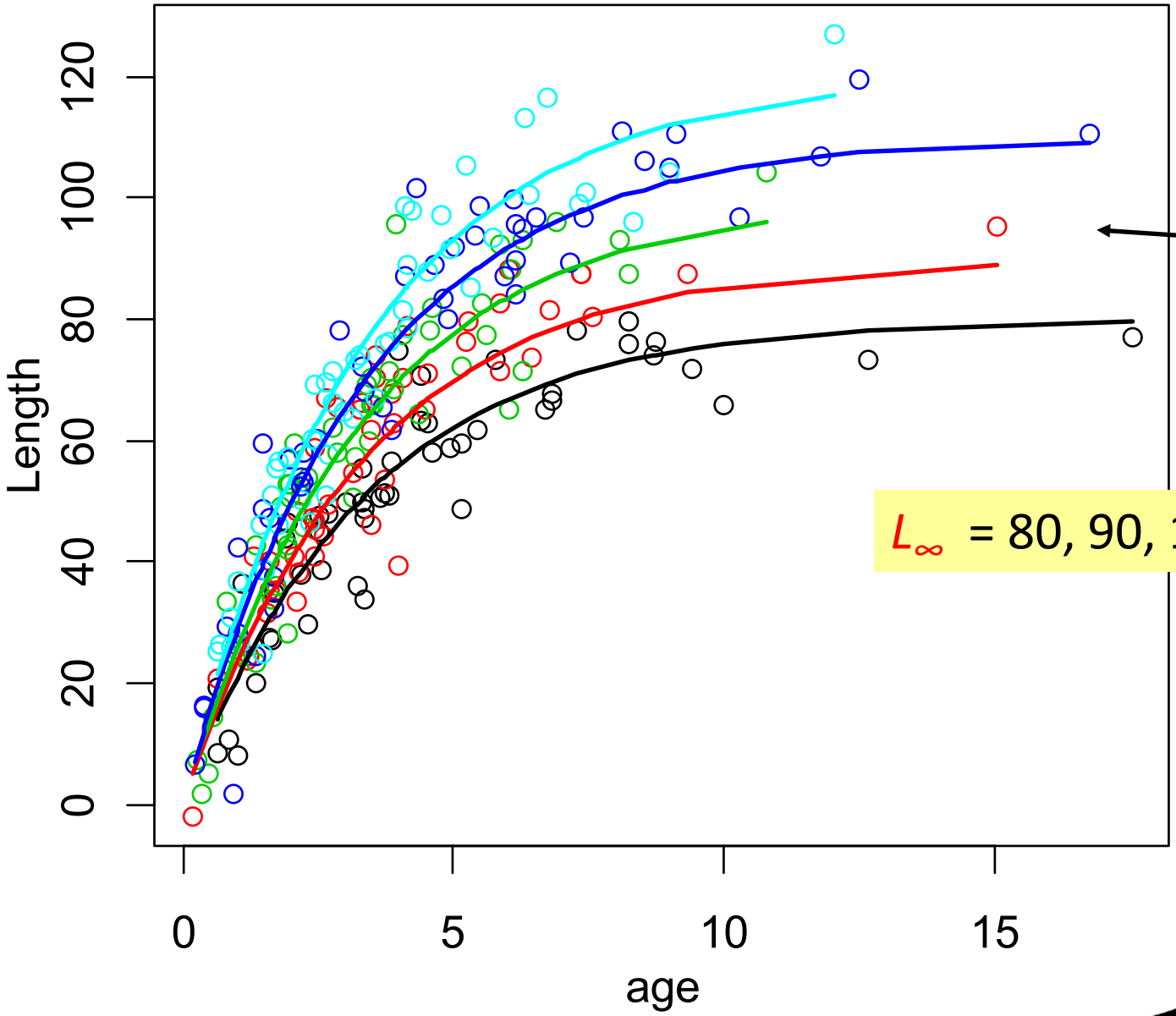
Simulation example results

```
Waiting for profiling to be done...
             2.5%        97.5%
Linf1   112.8526110 122.30378351
dLinf2 -28.9430096  -13.40053780
k1       0.4146473    0.52911145
dk2     -0.2096424   -0.05585173
```

$L_{\infty 2} - L_{\infty 1}$ is not significantly different from 0

$k_2 - k_1$ is significantly different from 0 – because the CI for the difference does not cover 0

5 Population Simulation Example



n=50, for each Population

L_∞ = 80, 90, 100, 110, 120

Nls() code using a residual function + map

```
gfit = function(map,length,age,Linf_main,k_main,Linf_dev,k_dev){
  Linf_dev = c(0,Linf_dev)
  k_dev = c(0,k_dev)
  Linf = Linf_main + Linf_dev
  k = k_main + k_dev
  pred = Linf[map]*(1 - exp(-k[map]*age))
  res = length - pred;
return(res)}

n.pop = length(unique(dat$pop))
dat$map = as.numeric(as.factor(dat$pop))

start.parm = list(Linf_main=100,k_main=0.3,
  Linf_dev=rep(0,n.pop-1),k_dev=rep(0,n.pop-1))

fit <- nls( ~ gfit(map,length,age,Linf_main,k_main,Linf_dev,k_dev),
  data=dat,start = start.parm,  algorithm="port",control=list(maxiter=500),
  lower=c(30,0.1,rep(-Inf,n.pop-1),rep(-0.3,n.pop-1)),
  upper=c(200,0.5,rep(Inf,n.pop-1),rep(0.3,n.pop-1)))
```

Simulation example results

Formula: 0 ~ gfit(map, length, age, Linf_main, k_main, Linf_dev, k_dev)

Parameters:

	Estimate	Std. Error	t value	Pr(>|t|)	
Linf_main	78.408387	3.550369	22.085	< 2e-16	***
k_main	0.307424	0.031771	9.676	< 2e-16	***
Linf_dev1	17.225477	5.986104	2.878	0.004369	**
Linf_dev2	22.412615	6.164201	3.636	0.000339	***
Linf_dev3	33.816352	4.918248	6.876	5.30e-11	***
Linf_dev4	43.078266	6.243714	6.899	4.61e-11	***
k_dev1	-0.015557	0.043131	-0.361	0.718647	
k_dev2	-0.001015	0.044376	-0.023	0.981777	
k_dev3	-0.004537	0.038790	-0.117	0.906986	
k_dev4	-0.024605	0.039449	-0.624	0.533401	

L_∞ deviations are significantly different from 0

k deviations are not significantly different from 0

--- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: **7.906** on **240** degrees of freedom

Simulation profile CI's

	2.5%	97.5%
Linf_main	72.048	85.946
k_main	0.251	0.375
Linf_dev1	5.762	29.516
Linf_dev2	10.696	35.426
Linf_dev3	24.017	43.449
Linf_dev4	31.133	55.959
k_dev1	-0.101	0.069
k_dev2	-0.090	0.086
k_dev3	-0.084	0.069
k_dev4	-0.105	0.050