FAS6337C - Final Exam

Marcel Gietzmann-Sanders

```r
library(lme4)
```

```
## Loading required package: Matrix
```

```r
col2rgbA<-function(color,transparency)
{
  rgb(t(col2rgb(color))/255,alpha=transparency)
}
```

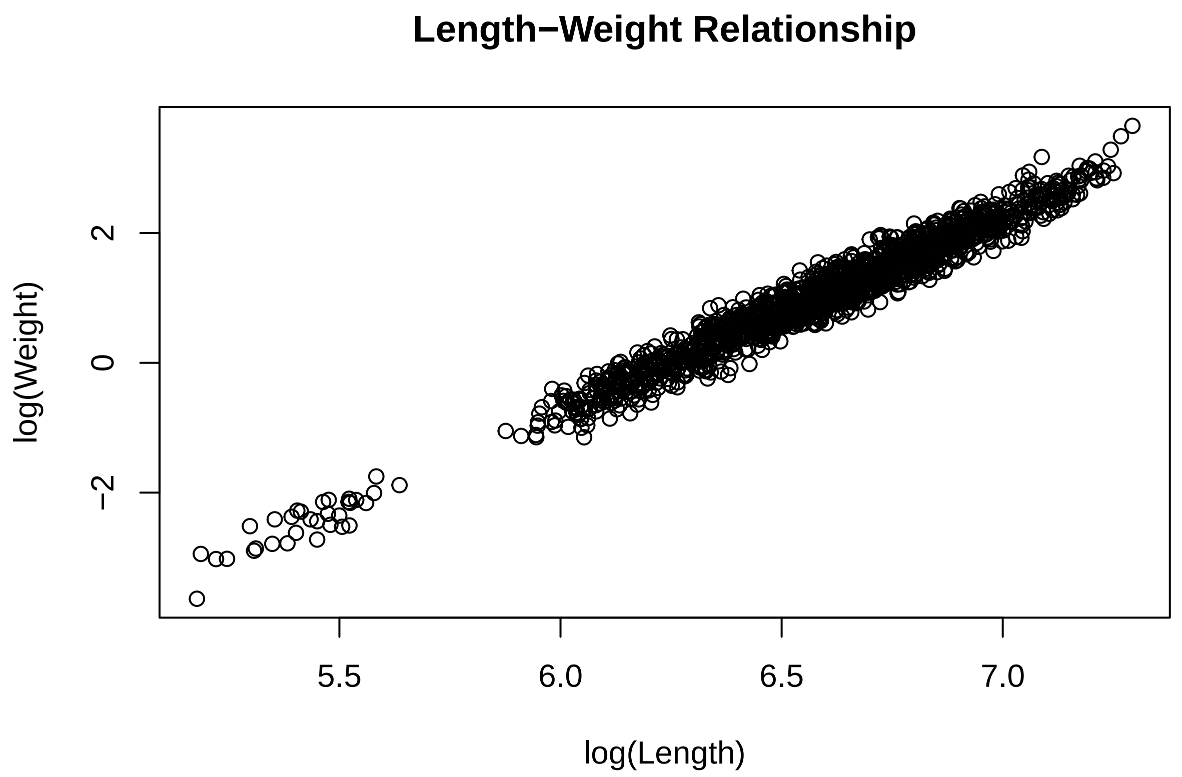
```r
setwd("/workspaces/schooling/population_dynamics/final/")
data <- read.table("data/King_Mackerel_sim-1.csv", header=T, sep=",")
data$Age <- as.numeric(data$Age)
head(data)
```

```
##   Age Sex        TL        Wt Mat Fishery
## 1  NA   M  524.2847  1.073146   1    TRUE
## 2  17   M  897.2538  8.551549   1   FALSE
## 3   7   F 1161.7468 15.037270  NA   FALSE
## 4  NA   F 1009.6333  8.152633  NA    TRUE
## 5  NA   M  531.5507  1.150809  NA    TRUE
## 6   1   F  544.4634  1.097086  NA    TRUE
```

1. Fit a length-weight relationship to the pooled (regardless of sex) data.

```r
length_weight <- na.omit(data[,c('TL', 'Wt')])
length_weight$lTL <- log(length_weight$TL)
length_weight$lWt <- log(length_weight$Wt)
```

```r
plot(
    length_weight$lTL,
    length_weight$lWt,
    xlab="log(Length)",
    ylab="log(Weight)",
    main="Length-Weight Relationship"
)
```



```r
(fit <- lm(lWt ~ lTL, data=length_weight))
```

```
##
## Call:
## lm(formula = lWt ~ lTL, data = length_weight)
##
## Coefficients:
## (Intercept)          lTL
##     -18.775        3.005
```

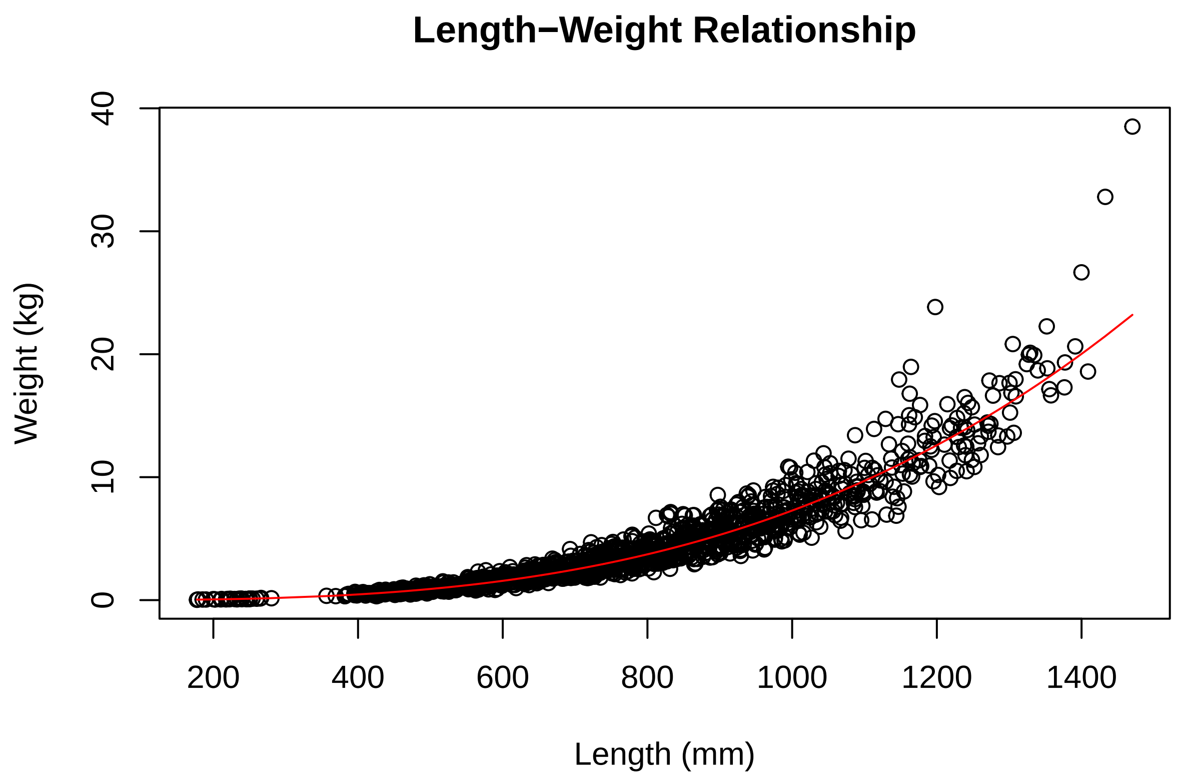
a. Report the length-weight parameters

```r
a <- exp(coef(fit)[1])
b <- coef(fit)[2]
c(a, b)
```

```
##  (Intercept)          lTL
## 7.014685e-09 3.005456e+00
```

b. Plot the mean weight at age against the data

```r
length_weight$predWt <- a * length_weight$TL^b
plot(
    length_weight$TL,
    length_weight$Wt,
    xlab="Length (mm)",
    ylab="Weight (kg)",
    main="Length-Weight Relationship"
)
lines(length_weight$TL[order(length_weight$TL)], length_weight$predWt[order(length_weight$TL)], col="re
```

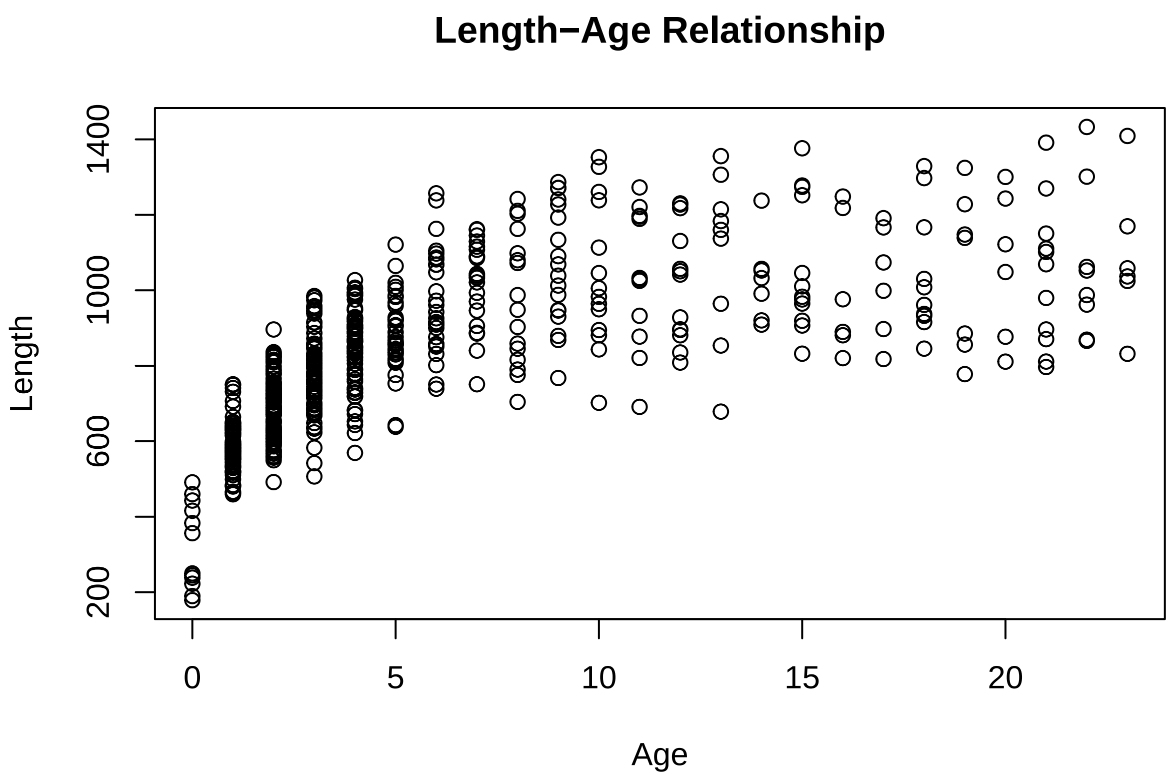


2. Fit a series of von Bertalanffy growth models to the King Mackerel data. Determine if males and females exihibit secually dimorphic growth.

```
filtered_data <- na.omit(data[, c('Age', 'TL', 'Sex')])
head(filtered_data)
```

```
##    Age        TL Sex
## 2   17  897.2538   M
## 3    7 1161.7468   F
## 6    1  544.4634   F
## 7    2  737.3520   F
## 8   15  919.3309   F
## 10   9  987.9968   F
```

```
plot(
    filtered_data$Age,
    filtered_data$TL,
    xlab="Age",
    ylab="Length",
    main="Length-Age Relationship"
)
```



```
predict_length <- function(yearsold, Linf, vbk, tknot) {
  pred_tl <- Linf * (1 - exp(-vbk * (yearsold - tknot)))
  return(pred_tl)
}

get_likelihood <- function(yearsold, tl, Linf, vbk, tknot, sig) {
  pred_tl <- predict_length(yearsold, Linf, vbk, tknot)
  NLL <- -1 * sum(dnorm(tl, pred_tl, sig, log=T), na.rm=T)
  return(NLL)
}
```

```
map_columns <- function(v, cols) {
    c <- rep(0, 8)
    i <- 0
    for (col in cols) {
        i <- i + 1
        if (endsWith(col, 'Linf')) {
            j <- 1
        } else if (endsWith(col, 'vbk')) {
            j <- 3
        } else if (endsWith(col, 'tknot')) {
            j <- 5
        } else {
            j <- 7
        }

        if (startsWith(col, 'f_')) {
            c[j] <- v[i]
        } else if (startsWith(col, 'm_')) {
            c[j+1] <- v[i]
        } else {
            c[j] <- v[i]
            c[j+1] <- v[i]
        }
    }
    return(c)
}

map_columns(c(1, 3, 2, 3, 4), c('m_Linf', 'f_Linf', 'vbk', 'tknot', 'sig'))
```

```
## [1] 3 1 2 2 3 3 4 4
```

```
male_data <- filtered_data[filtered_data$Sex == 'M',]
female_data <- filtered_data[filtered_data$Sex == 'F',]
```

```
do_likelihood_fit <- function(v, cols, runs) {
  objective <- function(v) {
    c <- map_columns(v, cols)

    f_Linf <- exp(c[1])
    f_vbk <- c[3]
    f_tknot <- c[5]
    f_sig <- exp(c[7])

    m_Linf <- exp(c[2])
    m_vbk <- c[4]
    m_tknot <- c[6]
    m_sig <- exp(c[8])

    f_NLL <- get_likelihood(female_data$Age, female_data$TL, f_Linf, f_vbk, f_tknot, f_sig)
    m_NLL <- get_likelihood(male_data$Age, male_data$TL, m_Linf, m_vbk, m_tknot, m_sig)

    NLL <- f_NLL + m_NLL
    return(NLL)
  }

  for (i in 1:runs) {
    fit <- optim(v, objective, hessian=T)
    v <- fit$par
  }
  return(fit)
}

summarize_aic <- function(fit, cols) {
  dof <- length(cols)
  nll <- fit$value
  aic <- 2*nll + 2*dof
  row <- c(paste(cols, collapse=','), dof, nll, aic)
  return(row)
}
```

a. Use AIC model comparison to determine if growth rates are sexually dimorphic.

All Parameters Free

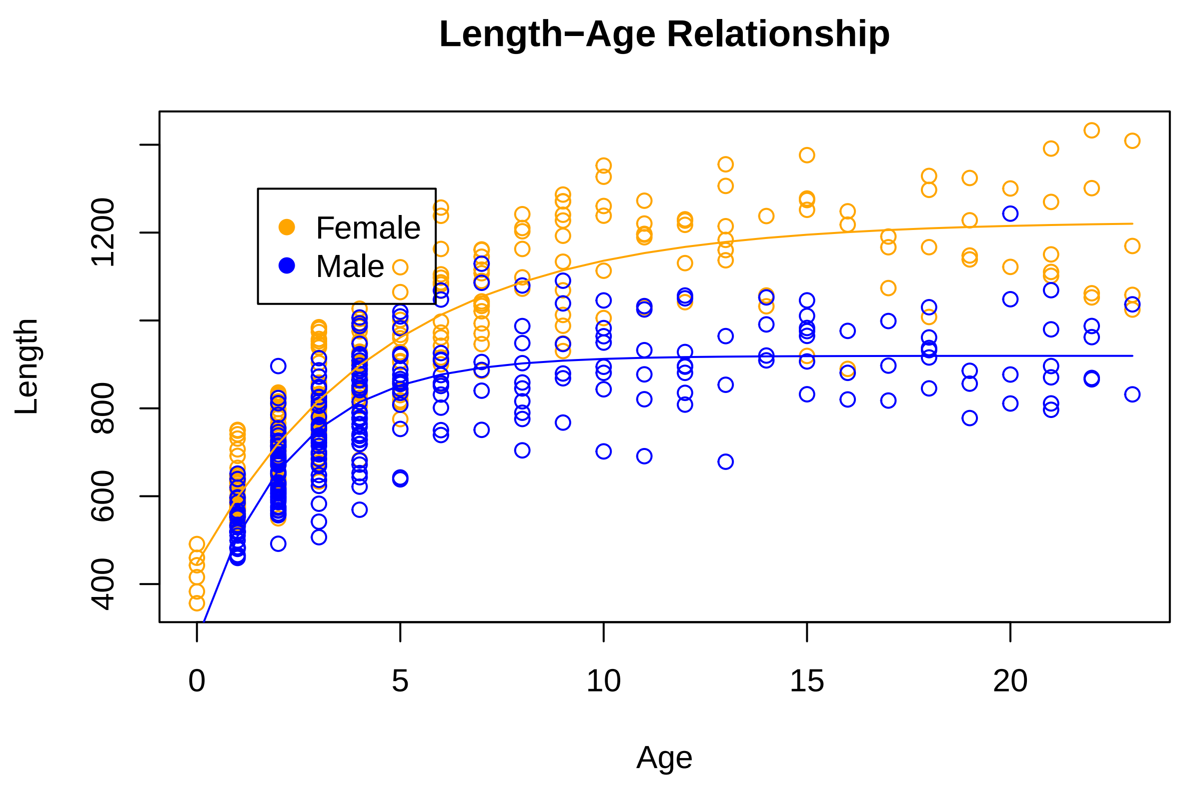
```
v <- c(
  6.7, 0.2, -1.5,  3.7,
  6.9,  0.2, -1.2,  4.1
)
cols <- c(
  'f_Linf', 'f_vbk', 'f_tknot', 'f_sig',
  'm_Linf', 'm_vbk', 'm_tknot', 'm_sig'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.1112054  0.2161858 -2.0997505  4.5126551  6.8238434  0.4530051 -0.7810601
## [8]  4.5217401
##
## $value
## [1] 3614.959
##
## $counts
## function gradient
##      205       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  3.121784e+04  5.166704e+04 -2.166257e+03  4.767381e-01     0.0000000
## [2,]  5.166704e+04  1.185625e+05 -5.663479e+03  2.337851e+00     0.0000000
## [3,] -2.166257e+03 -5.663479e+03  3.075424e+02 -4.495246e-02     0.0000000
## [4,]  4.767381e-01  2.337851e+00 -4.495246e-02  6.401699e+02     0.0000000
## [5,]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 20903.4625066
## [6,] -1.136868e-07  0.000000e+00  0.000000e+00  0.000000e+00 10238.6636161
## [7,] -1.136868e-07 -1.136868e-07  0.000000e+00  5.684342e-08 -1456.8347946
## [8,]  0.000000e+00  1.136868e-07  0.000000e+00 -5.684342e-08    -0.4572431
##               [,6]          [,7]          [,8]
## [1,] -1.136868e-07 -1.136868e-07  0.000000e+00
## [2,]  0.000000e+00 -1.136868e-07  1.136868e-07
## [3,]  0.000000e+00  0.000000e+00  0.000000e+00
## [4,]  0.000000e+00  5.684342e-08 -5.684342e-08
## [5,]  1.023866e+04 -1.456835e+03 -4.572431e-01
## [6,]  9.028189e+03 -1.526505e+03 -2.379298e-01
## [7,] -1.526505e+03  3.154285e+02 -2.883826e-02
## [8,] -2.379298e-01 -2.883826e-02  5.778027e+02
```

```r
(all_free <- summarize_aic(fit, cols))
```

```
## [1] "f_Linf,f_vbk,f_tknot,f_sig,m_Linf,m_vbk,m_tknot,m_sig"
## [2] "8"
## [3] "3614.95883788235"
## [4] "7245.9176757647"
```

```r
plot(
    female_data$Age,
    female_data$TL,
    xlab="Age",
    ylab="Length",
    main="Length-Age Relationship",
    col="orange"
)
points(
    male_data$Age,
    male_data$TL,
    xlab="Age",
    ylab="Length",
    col="blue"
)
lines(
    female_data$Age[order(female_data$Age)],
    predict_length(
        female_data$Age[order(female_data$Age)],
        exp(fit$par[1]),
        fit$par[2],
        fit$par[3]
    ),
    col="orange"
)
lines(
    male_data$Age[order(male_data$Age)],
    predict_length(
        male_data$Age[order(male_data$Age)],
        exp(fit$par[5]),
        fit$par[6],
        fit$par[7]
    ),
    col="blue"
)
legend(x=1.5, y=1300, pch=19, legend=c("Female", "Male"), col=c("orange", "blue"))
```



One Parameter Shared

```r
v <- c(
  6.7, 0.2, -1.5,  3.7,
  0.2, -1.2,  4.1
)
cols <- c(
  'Linf', 'f_vbk', 'f_tknot', 'f_sig',
  'm_vbk', 'm_tknot', 'm_sig'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.0913526  0.2330283 -1.9461077  4.5168442  0.0752754 -8.5507852  4.8240860
##
## $value
## [1] 3703.631
##
## $counts
## function gradient
##      184       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##            [,1]         [,2]          [,3]          [,4]          [,5]
## [1,] 42241.4744 45313.450383 -2.159058e+03  2.607615e+02  78381.513233
## [2,] 45313.4504 95482.543525 -5.168607e+03  2.637779e+00      0.000000
## [3,] -2159.0579 -5168.606757  3.202974e+02 -1.248527e-01      0.000000
## [4,]   260.7615     2.637779 -1.248527e-01  6.398794e+02      0.000000
## [5,] 78381.5132     0.000000  0.000000e+00  0.000000e+00 559537.122404
## [6,]  -436.5054     0.000000  5.684342e-08 -5.684342e-08  -3392.349968
## [7,]  -260.1310     0.000000  5.684342e-08  5.684342e-08      7.832609
##               [,6]          [,7]
## [1,] -4.365054e+02 -2.601310e+02
## [2,]  0.000000e+00  0.000000e+00
## [3,]  5.684342e-08  5.684342e-08
## [4,] -5.684342e-08  5.684342e-08
## [5,] -3.392350e+03  7.832609e+00
## [6,]  2.144474e+01  4.846356e-03
## [7,]  4.846356e-03  5.779087e+02
```

```
(Linf_shared <- summarize_aic(fit, cols))
```

```
## [1] "Linf,f_vbk,f_tknot,f_sig,m_vbk,m_tknot,m_sig"
## [2] "7"
## [3] "3703.63117171683"
## [4] "7421.26234343366"
```

```
v <- c(
  6.7, 0.2, -1.5,  3.7,
  6.9,  -1.2,  4.1
)
cols <- c(
  'f_Linf', 'vbk', 'f_tknot', 'f_sig',
  'm_Linf', 'm_tknot', 'm_sig'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.0774301  0.2688205 -1.5382608  4.5331102  6.8735688 -2.0874284  4.5882476
##
## $value
## [1] 3640.672
##
## $counts
## function gradient
##      182       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]        [,2]          [,3]          [,4]          [,5]
## [1,] 29954.9983062 36029.9860 -2.235673e+03 -3.386353e-01  0.000000e+00
## [2,] 36029.9860197 96103.4722 -4.546372e+03 -4.469944e+02  1.953863e+04
## [3,] -2235.6726111 -4546.3723  3.834520e+02  3.331172e-02 -1.136868e-07
## [4,]    -0.3386353  -446.9944  3.331172e-02  6.398792e+02  1.136868e-07
## [5,]     0.0000000 19538.6328 -1.136868e-07  1.136868e-07  1.826211e+04
## [6,]     0.0000000 -1998.0140  0.000000e+00  0.000000e+00 -1.056002e+03
## [7,]     0.0000000   448.2329  1.136868e-07 -1.136868e-07  8.273706e-01
##               [,6]          [,7]
## [1,]     0.0000000  0.000000e+00
## [2,] -1998.0140360  4.482329e+02
## [3,]     0.0000000  1.136868e-07
## [4,]     0.0000000 -1.136868e-07
## [5,] -1056.0020204  8.273706e-01
## [6,]   130.0742837 -8.116410e-02
## [7,]    -0.0811641  5.778795e+02
```

```
(vbk_shared <- summarize_aic(fit, cols))
```

```
## [1] "f_Linf,vbk,f_tknot,f_sig,m_Linf,m_tknot,m_sig"
## [2] "7"
## [3] "3640.67165887948"
## [4] "7295.34331775897"
```

```
v <- c(
  6.7, 0.2, -1.5,  3.7,
  6.9,  0.2, 4.1
)
cols <- c(
  'f_Linf', 'f_vbk', 'tknot', 'f_sig',
  'm_Linf', 'm_vbk', 'm_sig'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.0944387  0.2497356 -1.6354825  4.5236146  6.8444502  0.3298738  4.5546155
##
## $value
## [1] 3627.938
##
## $counts
## function gradient
##      190       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]        [,3]        [,4]          [,5]
## [1,]  3.053410e+04 40939.7761298 -2249.92979   0.5464788 -1.136868e-07
## [2,]  4.093978e+04 79319.2481551 -5044.54097   0.7110081  0.000000e+00
## [3,] -2.249930e+03 -5044.5409719   540.46346 -32.4875024 -1.187305e+03
## [4,]  5.464788e-01     0.7110081   -32.48750 639.8930564  0.000000e+00
## [5,] -1.136868e-07     0.0000000 -1187.30510   0.0000000  1.955300e+04
## [6,] -2.273737e-07     0.0000000 -1732.12284   0.0000000  1.534938e+04
## [7,] -1.136868e-07     0.0000000    32.54198   0.0000000  7.616635e-01
##               [,6]          [,7]
## [1,] -2.273737e-07 -1.136868e-07
## [2,]  0.000000e+00  0.000000e+00
## [3,] -1.732123e+03  3.254198e+01
## [4,]  0.000000e+00  0.000000e+00
## [5,]  1.534938e+04  7.616635e-01
## [6,]  1.980984e+04  4.386407e-01
## [7,]  4.386407e-01  5.779626e+02
```

```
(tknot_shared <- summarize_aic(fit, cols))
```

```
## [1] "f_Linf,f_vbk,tknot,f_sig,m_Linf,m_vbk,m_sig"
## [2] "7"
## [3] "3627.93765114933"
## [4] "7269.87530229866"
```

```
v <- c(
  6.7, 0.2, -1.5,  3.7,
  6.9,  0.2, -1.2
)
cols <- c(
  'f_Linf', 'f_vbk', 'f_tknot', 'sig',
  'm_Linf', 'm_vbk', 'm_tknot'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.1111712  0.2162425 -2.0988986  4.5169029  6.8238213  0.4529652 -0.7813596
##
## $value
## [1] 3614.971
##
## $counts
## function gradient
##      182       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  3.095379e+04  5.121052e+04 -2.148158e+03    0.44414833  5.684342e-08
## [2,]  5.121052e+04  1.174782e+05 -5.614558e+03    2.06822108 -1.136868e-07
## [3,] -2.148158e+03 -5.614558e+03  3.050577e+02   -0.07467463  1.136868e-07
## [4,]  4.441483e-01  2.068221e+00 -7.467463e-02 1218.17394734  4.131224e-01
## [5,]  5.684342e-08 -1.136868e-07  1.136868e-07    0.41312239  2.110536e+04
## [6,]  0.000000e+00 -1.136868e-07  0.000000e+00    0.01988911  1.033889e+04
## [7,]  0.000000e+00  0.000000e+00  0.000000e+00   -0.02622176 -1.470825e+03
##               [,6]          [,7]
## [1,]  0.000000e+00  0.000000e+00
## [2,] -1.136868e-07  0.000000e+00
## [3,]  0.000000e+00  0.000000e+00
## [4,]  1.988911e-02 -2.622176e-02
## [5,]  1.033889e+04 -1.470825e+03
## [6,]  9.118237e+03 -1.541328e+03
## [7,] -1.541328e+03  3.183934e+02
```

```
(sig_shared <- summarize_aic(fit, cols))
```

```
## [1] "f_Linf,f_vbk,f_tknot,sig,m_Linf,m_vbk,m_tknot"
## [2] "7"
## [3] "3614.97052281043"
## [4] "7243.94104562085"
```

All Parameters Shared

```
v <- c(
  6.7, 0.2, -1.5,  3.7
)
cols <- c(
  'Linf', 'vbk', 'tknot', 'sig'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  6.9696336  0.2925566 -1.5765916  4.7850732
##
## $value
## [1] 3778.24
##
## $counts
## function gradient
##      103       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##                [,1]          [,2]          [,3]          [,4]
## [1,] 30197.0126482 30393.7804352 -2.060128e+03   -0.27322972
## [2,] 30393.7804352 47153.6073625 -3.672394e+03    0.20870669
## [3,] -2060.1282126 -3672.3936988  3.265126e+02    0.02169679
## [4,]    -0.2732297     0.2087067  2.169679e-02 1217.99030853
```

```
(all_shared <- summarize_aic(fit, cols))
```

```
## [1] "Linf,vbk,tknot,sig" "4"                  "3778.24034404852"
## [4] "7564.48068809703"
```

σ and t_0 shared

```
v <- c(
  6.7, 0.2, -1.5,  3.7,
  6.9,  0.2
)
cols <- c(
  'f_Linf', 'f_vbk', 'tknot', 'sig',
  'm_Linf', 'm_vbk'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.0937079  0.2514084 -1.6152783  4.5384581  6.8440475  0.3319652
##
## $value
## [1] 3628.073
##
## $counts
## function gradient
##      161       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  2.964122e+04  3.935482e+04 -2.190114e+03   -0.08224310  1.136868e-07
## [2,]  3.935482e+04  7.564471e+04 -4.875642e+03    0.61480750  0.000000e+00
## [3,] -2.190114e+03 -4.875642e+03  5.407134e+02   -0.01897547 -1.230203e+03
## [4,] -8.224310e-02  6.148075e-01 -1.897547e-02 1218.07695632  6.748701e-01
## [5,]  1.136868e-07  0.000000e+00 -1.230203e+03    0.67487008  2.019602e+04
## [6,]  0.000000e+00  5.684342e-08 -1.783203e+03    0.45434240  1.570887e+04
##               [,6]
## [1,]  0.000000e+00
## [2,]  5.684342e-08
## [3,] -1.783203e+03
## [4,]  4.543424e-01
## [5,]  1.570887e+04
## [6,]  2.012849e+04
```

```
(sig_and_tknot_shared <- summarize_aic(fit, cols))
```

```
## [1] "f_Linf,f_vbk,tknot,sig,m_Linf,m_vbk" "6"
## [3] "3628.07338289556"                    "7268.14676579111"
```

b. Report an AIC table.

```
aic_table <- data.frame(rbind(
    all_free,
    all_shared,
    Linf_shared,
    vbk_shared,
    tknot_shared,
    sig_shared,
    sig_and_tknot_shared
))
colnames(aic_table) <- c('cols', 'dof', 'nll', 'aic')

aic_table$aic <- as.numeric(aic_table$aic)
aic_table$delta <- aic_table$aic - min(aic_table$aic)
(aic_table <- aic_table[order(aic_table$delta),])
```

```
##                                                                        cols dof
## sig_shared                    f_Linf,f_vbk,f_tknot,sig,m_Linf,m_vbk,m_tknot   7
## all_free              f_Linf,f_vbk,f_tknot,f_sig,m_Linf,m_vbk,m_tknot,m_sig   8
## sig_and_tknot_shared                    f_Linf,f_vbk,tknot,sig,m_Linf,m_vbk   6
## tknot_shared                    f_Linf,f_vbk,tknot,f_sig,m_Linf,m_vbk,m_sig   7
## vbk_shared                    f_Linf,vbk,f_tknot,f_sig,m_Linf,m_tknot,m_sig   7
## Linf_shared                    Linf,f_vbk,f_tknot,f_sig,m_vbk,m_tknot,m_sig   7
## all_shared                                             Linf,vbk,tknot,sig   4
##                                   nll      aic     delta
## sig_shared           3614.97052281043 7243.941   0.00000
## all_free             3614.95883788235 7245.918   1.97663
## sig_and_tknot_shared 3628.07338289556 7268.147  24.20572
## tknot_shared         3627.93765114933 7269.875  25.93426
## vbk_shared           3640.67165887948 7295.343  51.40227
## Linf_shared          3703.63117171683 7421.262 177.32130
## all_shared           3778.24034404852 7564.481 320.53964
```

These are definitely sexually dimorphic growth rates.

c. Which VBGF parameters differ between sexes in your top model?

The model with only σ shared has the lowest AIC. Therefore L_∞, t_0, and K all differ between sexes.

d. Report the VBGF parameter means and 95% confidence intervals from your top model

```
v <- c(
  6.7, 0.2, -1.5,  3.7,
  6.9,  0.2, -1.2
)
cols <- c(
  'f_Linf', 'f_vbk', 'f_tknot', 'sig',
  'm_Linf', 'm_vbk', 'm_tknot'
)
(fit <- do_likelihood_fit(v, cols, 25))
```

```
## $par
## [1]  7.1111712  0.2162425 -2.0988986  4.5169029  6.8238213  0.4529652 -0.7813596
##
## $value
## [1] 3614.971
##
```

```
## $counts
## function gradient
##      182       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  3.095379e+04  5.121052e+04 -2.148158e+03    0.44414833  5.684342e-08
## [2,]  5.121052e+04  1.174782e+05 -5.614558e+03    2.06822108 -1.136868e-07
## [3,] -2.148158e+03 -5.614558e+03  3.050577e+02   -0.07467463  1.136868e-07
## [4,]  4.441483e-01  2.068221e+00 -7.467463e-02 1218.17394734  4.131224e-01
## [5,]  5.684342e-08 -1.136868e-07  1.136868e-07    0.41312239  2.110536e+04
## [6,]  0.000000e+00 -1.136868e-07  0.000000e+00    0.01988911  1.033889e+04
## [7,]  0.000000e+00  0.000000e+00  0.000000e+00   -0.02622176 -1.470825e+03
##               [,6]          [,7]
## [1,]  0.000000e+00  0.000000e+00
## [2,] -1.136868e-07  0.000000e+00
## [3,]  0.000000e+00  0.000000e+00
## [4,]  1.988911e-02 -2.622176e-02
## [5,]  1.033889e+04 -1.470825e+03
## [6,]  9.118237e+03 -1.541328e+03
## [7,] -1.541328e+03  3.183934e+02
```

```
(sig_shared <- summarize_aic(fit, cols))
```

```
## [1] "f_Linf,f_vbk,f_tknot,sig,m_Linf,m_vbk,m_tknot"
## [2] "7"
## [3] "3614.97052281043"
## [4] "7243.94104562085"
```

```
stderr <- sqrt(diag(solve(fit$hessian)))
stderr
```

```
## [1] 0.01270934 0.01344477 0.19483086 0.02865137 0.01114028 0.03274172 0.14187294
```

```
ALPHA <- 0.05
P_L95 <- (fit$par - qnorm(1-(ALPHA/2)) * stderr)
P_MEAN <- fit$par
P_U95 <- (fit$par + qnorm(1-(ALPHA/2)) * stderr)
P_U95[1] <- exp(P_U95[1])
P_MEAN[1] <- exp(P_MEAN[1])
P_L95[1] <- exp(P_L95[1])
P_U95[4] <- exp(P_U95[4])
P_MEAN[4] <- exp(P_MEAN[4])
P_L95[4] <- exp(P_L95[4])
P_U95[5] <- exp(P_U95[5])
P_MEAN[5] <- exp(P_MEAN[5])
P_L95[5] <- exp(P_L95[5])
```

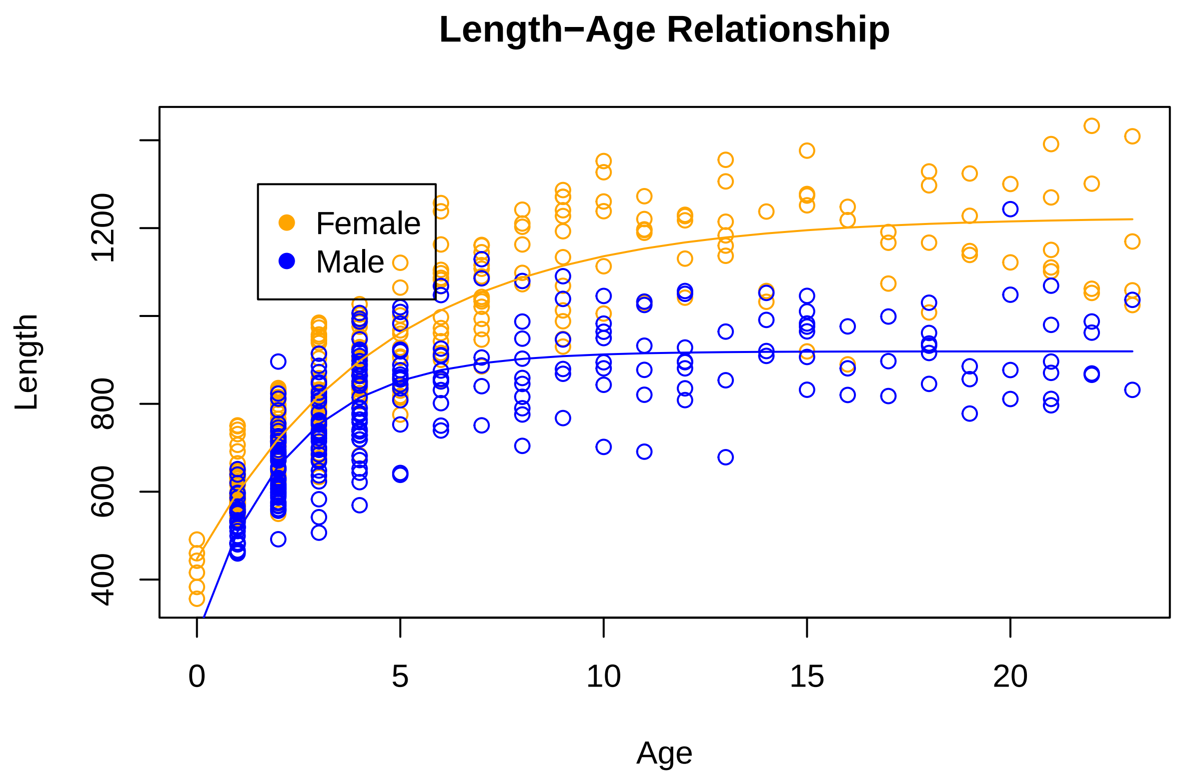
```
tab <- rbind(P_L95, P_MEAN, P_U95)
colnames(tab) <- c('f_Linf', 'f_K', 'f_t0', 'sigma', 'm_Linf', 'm_K', 'm_t0')
```

```
tab
```

```
##          f_Linf       f_K      f_t0    sigma   m_Linf       m_K       m_t0
## P_L95  1195.430 0.1898912 -2.480760 86.55216 899.6329 0.3887926 -1.0594254
## P_MEAN 1225.582 0.2162425 -2.098899 91.55162 919.4920 0.4529652 -0.7813596
## P_U95  1256.495 0.2425938 -1.717037 96.83985 939.7895 0.5171378 -0.5032937
```

e. Plot the resulting male and female predicted length at age from your top model against the data.

```
plot(
    female_data$Age,
    female_data$TL,
    xlab="Age",
    ylab="Length",
    main="Length-Age Relationship",
    col="orange"
)
points(
    male_data$Age,
    male_data$TL,
    xlab="Age",
    ylab="Length",
    col="blue"
)
lines(
    female_data$Age[order(female_data$Age)],
    predict_length(
        female_data$Age[order(female_data$Age)],
        exp(fit$par[1]),
        fit$par[2],
        fit$par[3]
    ),
    col="orange"
)
lines(
    male_data$Age[order(male_data$Age)],
    predict_length(
        male_data$Age[order(male_data$Age)],
        exp(fit$par[5]),
        fit$par[6],
        fit$par[7]
    ),
    col="blue"
)
legend(x=1.5, y=1300, pch=19, legend=c("Female", "Male"), col=c("orange", "blue"))
```



3. Fit a series of logistic maturity models to the King Mackerel data. Determine if males and females exhibit sexually dimorphic maturity.

```r
filtered_data <- na.omit(data[,c('TL', 'Sex', 'Mat')])
female_data <- filtered_data[filtered_data['Sex'] == 'F',]
male_data <- filtered_data[filtered_data['Sex'] == 'M',]

get_nll <- function(theta) {
    f_lmat50 <- theta[1]
    f_sig <- theta[2]
    m_lmat50 <- theta[3]
    m_sig <- theta[4]

    prob_mat_female <- 1 / (1 + exp(-(female_data$TL - f_lmat50) / f_sig))
    prob_mat_male <- 1 / (1 + exp(-(male_data$TL - m_lmat50) / m_sig))
    nll_female <- -1*sum(dbinom(female_data$Mat, size=1, prob=prob_mat_female, log=T))
    nll_male <- -1*sum(dbinom(male_data$Mat, size=1, prob=prob_mat_male, log=T))
    return(nll_female + nll_male)
}
```

```r
subset_theta <- function(theta, params_to_fit) {
    input <- c()
    for (i in 1:length(params_to_fit)) {
        input <- append(input, theta[params_to_fit[i]])
    }
    return(input)
}

update_theta <- function(theta, input, params_to_fit, index_to_share) {
    v = theta
    for (i in 1:length(params_to_fit)) {
        v[params_to_fit[i]] <- input[i]
    }
    for (i in 1:length(v)) {
        if (!(i %in% params_to_fit)) {
            v[i] <- v[index_to_share[i]]
        }
    }
    return(v)
}

do_fit <- function(theta, params_to_fit, index_to_share) {

    fun <- function(input) {
        v = update_theta(theta, input, params_to_fit, index_to_share)
        return(get_nll(v))
    }

    input <- subset_theta(theta, params_to_fit)
    for (i in 1:10) {
        fit <- optim(input, fun, hessian=T)
        input <- fit$par
    }
    theta <- update_theta(theta, input, params_to_fit, index_to_share)
    return(theta)
}

get_aic <- function(theta, params_to_fit) {
    nll <- get_nll(theta)
    k <- length(params_to_fit)
    aic <- 2*k + 2*nll
    return(c(nll, k, aic))
}
```

a. Use AIC model comparison to determine if maturity is different between sexes.

```
starting_guess <- c(200, 20, 200, 20)
index_to_share <- c(3, 4, 1, 2)
row_names <- c()
col_names <- c("f_lmat50", "f_sig", "m_lmat50", "m_sig", "nll", "params", "AIC")

params_to_fit <- c(1, 2, 3)
theta <- do_fit(starting_guess, params_to_fit, index_to_share)
aic_info <- get_aic(theta, params_to_fit)
row1 <- c(theta, aic_info)
row_names <- append(row_names, "L(h)sig(.)")
```

```
params_to_fit <- c(1, 2, 4)
theta <- do_fit(starting_guess, params_to_fit, index_to_share)
aic_info <- get_aic(theta, params_to_fit)
row2 <- c(theta, aic_info)
row_names <- append(row_names, "L(.)sig(h)")

params_to_fit <- c(1, 2)
theta <- do_fit(starting_guess, params_to_fit, index_to_share)
aic_info <- get_aic(theta, params_to_fit)
row3 <- c(theta, aic_info)
row_names <- append(row_names, "L(.)sig(.)")

params_to_fit <- c(1, 2, 3, 4)
theta <- do_fit(starting_guess, params_to_fit, index_to_share)
aic_info <- get_aic(theta, params_to_fit)
row4 <- c(theta, aic_info)
row_names <- append(row_names, "L(h)sig(h)")
```

b. Report an AIC table.

```
results <- rbind(row1, row2, row3, row4)
colnames(results) <- col_names
rownames(results) <- row_names
results <- data.frame(results)

results <- results[order(results$AIC),]
results$delta_aic <- results$AIC - results$AIC[1]
results
```

```
##            f_lmat50     f_sig m_lmat50    m_sig      nll params      AIC
## L(h)sig(.) 575.7665  60.36917 453.3458 60.36917 105.1637      3 216.3274
## L(h)sig(h) 571.4993  66.90611 459.9098 51.97033 104.6020      4 217.2040
## L(.)sig(h) 483.1750 109.70903 483.1750 48.48273 113.1332      3 232.2664
## L(.)sig(.) 505.5244  73.94905 505.5244 73.94905 120.0561      2 244.1121
##             delta_aic
## L(h)sig(.)  0.0000000
## L(h)sig(h)  0.8765145
## L(.)sig(h) 15.9389482
## L(.)sig(.) 27.7846937
```

c. Which maturity parameters differ between sexes in your top model?

Technically the top model differs only in σ but honestly the top two models are barely distuinguishable on the basis of AIC alone (I remember in class a difference of less than 5 being considered insignificant and this is much les than that). I'm going to go, in this case, with the model with the most degrees of freedom where both L_{mat} and σ are free. That way if there are biological differences we are capturing them.

d. Report the logistic maturity model parameter means and 95% confidence intervals from your top model.

```
theta <- results[2, 1:4]
for (i in 1:10) {
    fit <- optim(theta, get_nll, hessian=T)
    theta <- fit$par
}
theta
```

```
##  f_lmat50     f_sig  m_lmat50     m_sig
## 571.49928  66.90611 459.90977  51.97033
```

```
stderr <- sqrt(diag(solve(fit$hessian)))
stderr
```

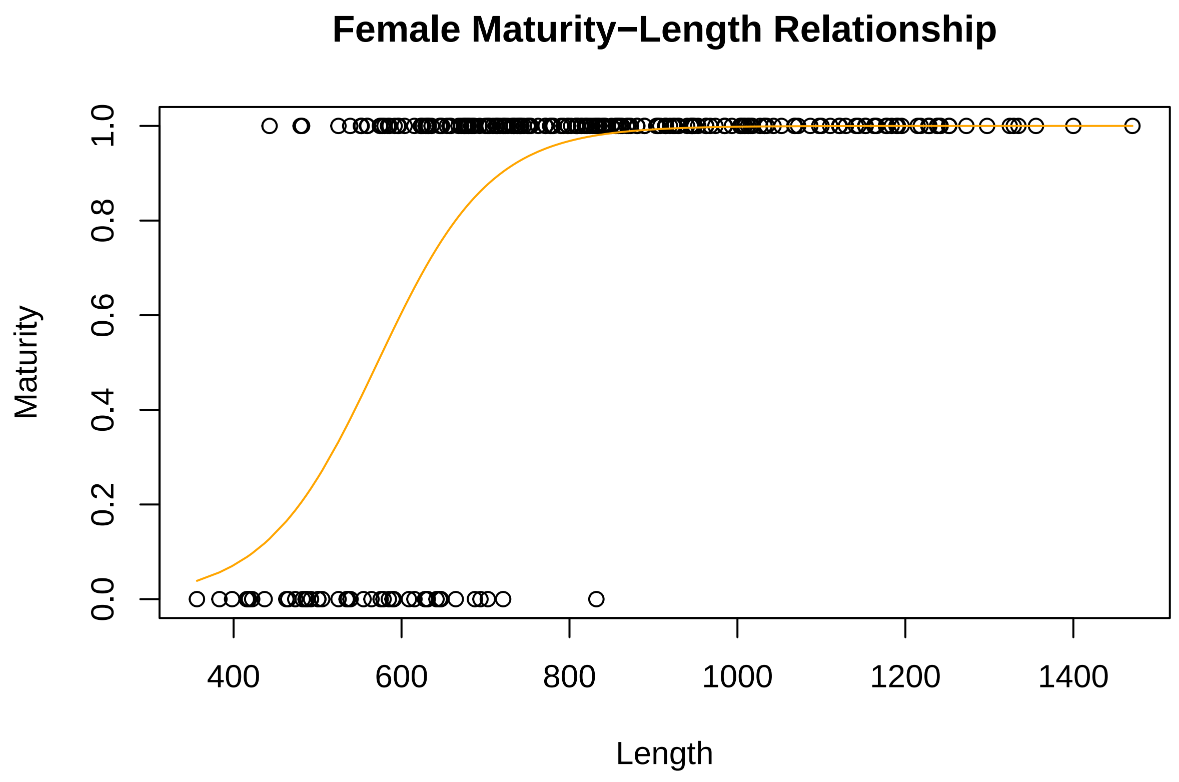
```
##  f_lmat50     f_sig  m_lmat50     m_sig
## 16.470306 10.408096 16.205611  9.397973
```

```
ALPHA <- 0.05
P_L95 <- (fit$par - qnorm(1-(ALPHA/2)) * stderr)
P_MEAN <- fit$par
P_U95 <- (fit$par + qnorm(1-(ALPHA/2)) * stderr)

tab <- rbind(P_L95, P_MEAN, P_U95)
colnames(tab) <- c('f_lmat50', 'f_sig', 'm_lmat50', 'm_sig')
tab
```

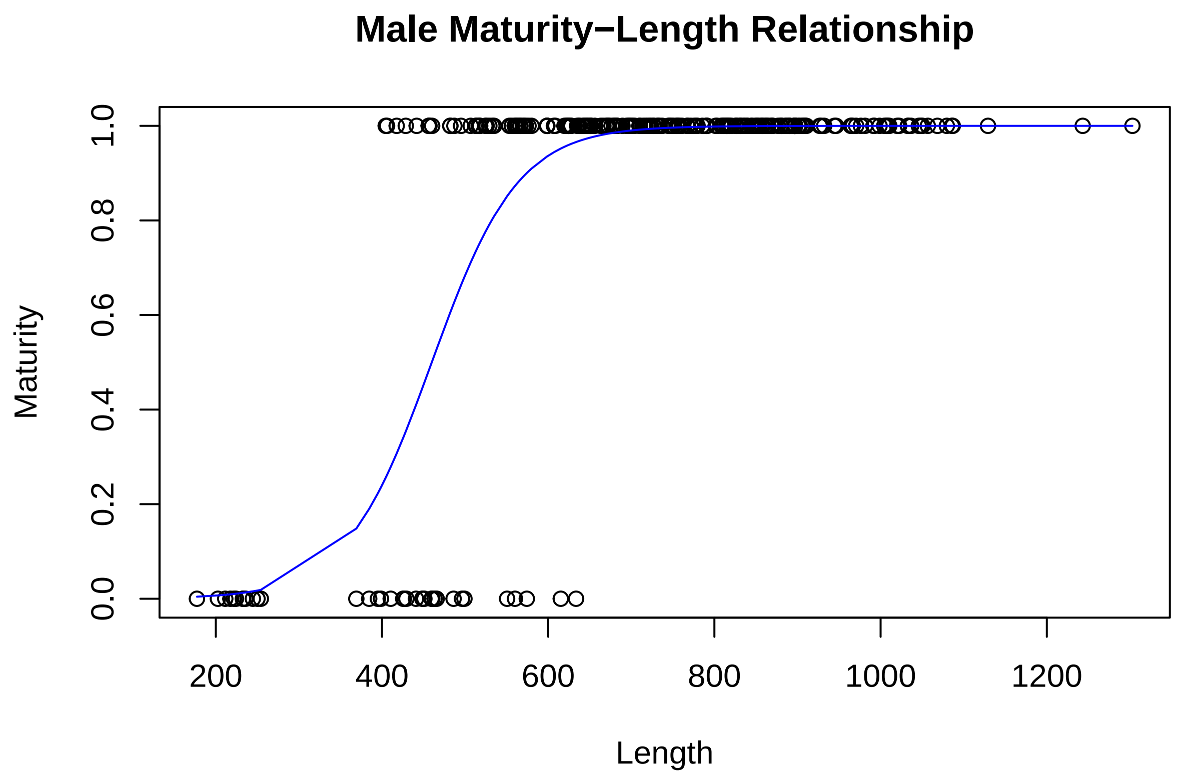
```
##        f_lmat50    f_sig m_lmat50    m_sig
## P_L95  539.2181 46.50661 428.1474 33.55064
## P_MEAN 571.4993 66.90611 459.9098 51.97033
## P_U95  603.7805 87.30560 491.6722 70.39001
```

e. Plot the resulting male and female predicted maturity at length from your top model against the data.

```
plot(
    female_data$TL,
    female_data$Mat,
    xlab="Length",
    ylab="Maturity",
    main="Female Maturity-Length Relationship",
)
lines(
    female_data$TL[order(female_data$TL)],
    1 / (1 + exp(-(female_data$TL[order(female_data$TL)] - fit$par[1]) / fit$par[2])),
    col="orange"
)
```



```r
plot(
    male_data$TL,
    male_data$Mat,
    xlab="Length",
    ylab="Maturity",
    main="Male Maturity-Length Relationship",
)
lines(
    male_data$TL[order(male_data$TL)],
    1 / (1 + exp(-(male_data$TL[order(male_data$TL)] - fit$par[3]) / fit$par[4])),
    col="blue"
)
```



4. Using the age composition data from the fishery dependent samples:

```
fishery_data <- data[data$Fishery,]
aged <- na.omit(fishery_data[,c('Age', 'TL')])
aged$cmgrp <- floor(aged$TL / 10)
(length(aged$TL))
```

```
## [1] 400
```

```
(length(fishery_data$TL))
```

```
## [1] 1191
```

```
(max(aged$Age))
```

```
## [1] 13
```

```
head(aged)
```

```
##    Age       TL cmgrp
## 6    1 544.4634    54
## 16   1 565.7178    56
## 19   1 648.6351    64
## 28   1 575.1484    57
## 34   4 901.0916    90
## 35   3 807.5866    80
```

a. Generate an age-length key and project the age of samples with only lengths.

```r
get_ALK <- function(aged) {
    ALK <- data.frame(
        prop.table(
            table(aged$Age, aged$cmgrp),
            margin=2
        )
    )
    names(ALK) <- c("Age","cmgrp","prop")
    ALK$Age <- as.numeric(as.character(ALK$Age))
    ALK$cmgrp <- as.numeric(as.character(ALK$cmgrp))
    ALK$prop <- as.numeric(as.character(ALK$prop))
    ALK$col.prop <- hcl.colors(101,'viridis')[round(ALK$prop,2)*100+1]
    return(ALK)
}

ALK <- get_ALK(aged)
head(ALK)
```

```
##   Age cmgrp prop col.prop
## 1   1    45    1  #FDE333
## 2   2    45    0  #4B0055
## 3   3    45    0  #4B0055
## 4   4    45    0  #4B0055
## 5   5    45    0  #4B0055
## 6   6    45    0  #4B0055
```

```r
get_catch_count_info <- function(catch, ALK) {
    catch_summary <- data.frame(table(catch$cmgrp))
    names(catch_summary) <- c("cmgrp", "total")
    catch_summary$cmgrp <- as.numeric(as.character(catch_summary$cmgrp))
    catch_summary$total <- as.numeric(as.character(catch_summary$total))
    info <- merge(ALK, catch_summary)
    info$count <- info$total * info$prop
    return(info)
}

catch <- na.omit(fishery_data[, c('TL', 'Fishery')])
catch$cmgrp <- floor(catch$TL / 10)
catch_info <- get_catch_count_info(catch, ALK)
head(catch_info)
```

```
##   cmgrp Age prop col.prop total count
## 1    45   1    1  #FDE333    17    17
## 2    45   2    0  #4B0055    17     0
## 3    45   3    0  #4B0055    17     0
## 4    45   4    0  #4B0055    17     0
## 5    45   5    0  #4B0055    17     0
## 6    45   6    0  #4B0055    17     0
```

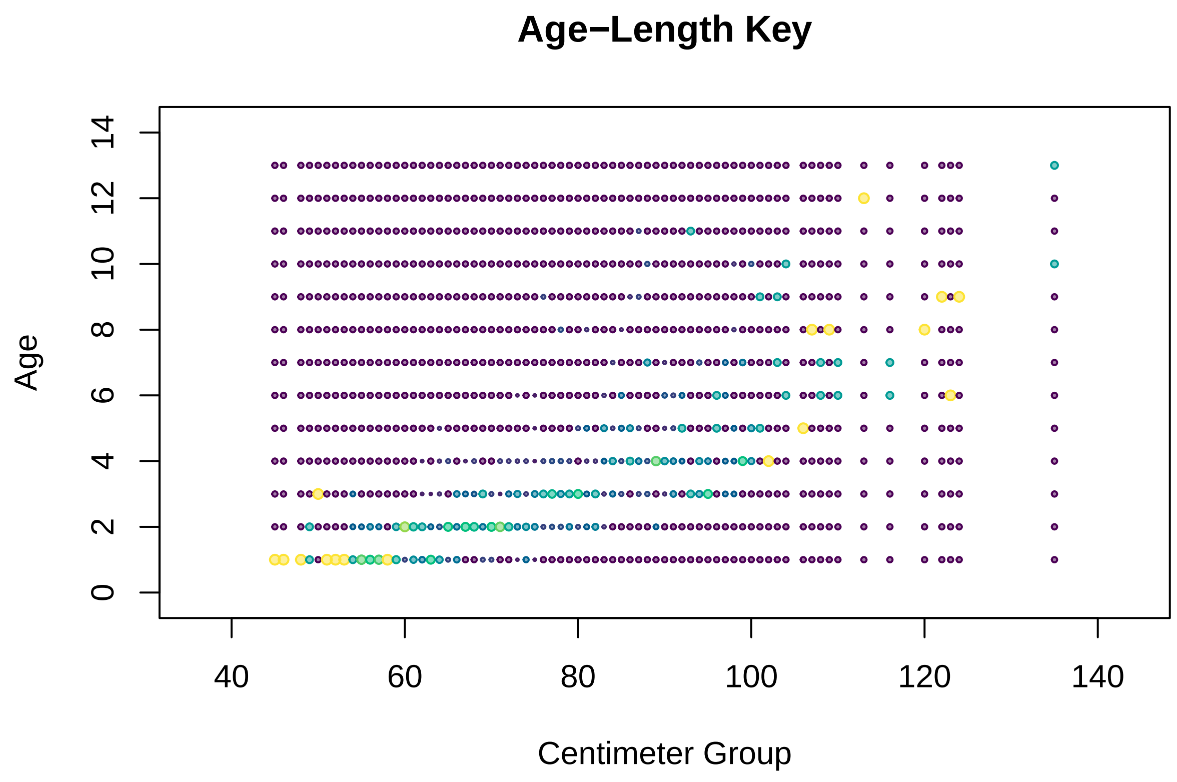
```r
get_CAA <- function(catch_info) {
    CAA <- aggregate(count~Age, data=catch_info, sum)
    return(CAA)
}
```

```r
(CAA <- get_CAA(catch_info))
```

```
##    Age      count
## 1    1 270.311616
## 2    2 271.795893
## 3    3 201.271612
## 4    4 139.356593
## 5    5  67.407143
## 6    6  40.622222
## 7    7  28.968254
## 8    8  17.775000
## 9    9  16.000000
## 10  10  11.325000
## 11  11   4.666667
## 12  12   2.000000
## 13  13   1.500000
```

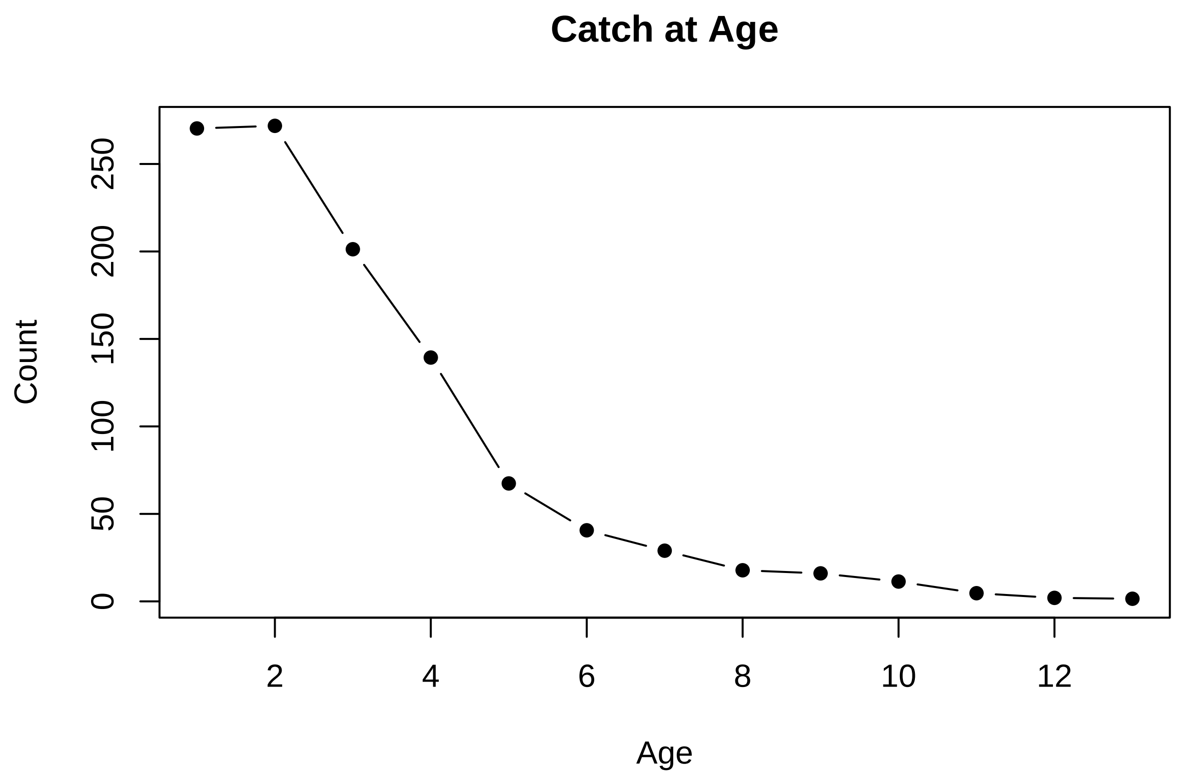
i. Plot a bubble plot of the age-length key.

```r
plot_ALK <- function(ALK, main) {
    # I want the circles to have area equal to
    # my proportion (for easier interpretation)
    # this would mean prop <- 4 * pi * r^2 or
    # r <- sqrt(prop / (4 * pi))
    with(
        ALK, {
            symbols(
                x=cmgrp,
                y=Age,
                circles=sqrt(prop/(4 * pi)) * 2,
                inches=F,
                fg = col.prop,
                bg = col2rgbA(col.prop,0.5),
                xlab="Centimeter Group",
                ylab="Age",
                main=main
            )
        }
    )
}

plot_ALK(ALK, "Age-Length Key")
```



ii. Plot the total catch at age data.

```
with(
    CAA, {
        plot(
            Age,
            count,
            type='b',
            pch=16,
            col="black",
            xlab="Age",
            ylab="Count",
            main='Catch at Age'
        )
    }
)
```



b. Estimate and report the instantaneous (Z) and finite (A) total mortality for this population. Justify the method you used to estimate Z and the data you included in the estimate.

We'll use the Millar model as the random effects allowed for in the model mean we can find the mortality pattern even with variability in the underlying recruitment for each age cohort. Furthermore we're going to only look at the years for which the count per age has already peaked (in order to make sure we've got full recruitment to vulnerability).

```r
reshape_for_millar <- function(CAA) {
    age_at_peak <- CAA$Age[which.max(CAA$count)]
    CAA_lim <- CAA[CAA$Age >= age_at_peak,]

    max_age <- max(CAA_lim$Age)
    CAA_ext <- rbind(
        CAA_lim,
        cbind(
            Age=(max_age+1):(2*max_age),
            count=rep(0,max_age)
        )
    )
    CAA_ext$count <- floor(CAA_ext$count)
    return(CAA_ext)
}

get_millar_fit <- function(CAA) {
    CAA_ext <- reshape_for_millar(CAA)

    fit <- glmer(
        count ~ Age + (1|Age),
        family=poisson,
        data=CAA_ext
    )
    print(summary(fit))
    return(fit)
}

(millar_fit <- get_millar_fit(CAA))
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ Age + (1 | Age)
##    Data: CAA_ext
##
##      AIC      BIC   logLik deviance df.resid
##     83.4     87.1    -38.7     77.4       22
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.02153 -0.52217 -0.19638 -0.08814  1.47695
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  Age    (Intercept) 0.00523  0.07232
## Number of obs: 25, groups:  Age, 25
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  6.65459    0.10051   66.21   <2e-16 ***
## Age         -0.47189    0.02048  -23.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##     (Intr)
## Age -0.882
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ Age + (1 | Age)
##    Data: CAA_ext
##      AIC      BIC   logLik deviance df.resid
##  83.4057  87.0623 -38.7028  77.4057       22
## Random effects:
##  Groups Name        Std.Dev.
##  Age    (Intercept) 0.07232
## Number of obs: 25, groups:  Age, 25
## Fixed Effects:
```

```
## (Intercept)         Age
##      6.6546     -0.4719
```

```
get_params_from_millar <- function(millar_fit, alpha) {
    coef <- summary(millar_fit)$coef
    Z <- coef[2, 1]
    std <- coef[2, 2]
    ZU <- min(Z + qnorm(1-alpha/2)*std, 0)
    ZL <- Z - qnorm(1-alpha/2)*std

    A <- 1 - exp(Z)
    AU <- 1 - exp(ZU)
    AL <- 1 - exp(ZL)

    S <- exp(Z)
    SU <- exp(ZU)
    SL <- exp(ZL)

    m <- matrix(c(ZL, Z, ZU, AL, A, AU, SL, S, SU),3,3)
        colnames(m) <- c("Z", "A", "S")
        rownames(m) <- c("Lower", "MLE", "Upper")
    return(m)
}

get_params_from_millar(millar_fit, 0.05)
```

```
##                Z         A         S
## Lower -0.5120376 0.4007268 0.5992732
## MLE   -0.4718949 0.3761809 0.6238191
## Upper -0.4317522 0.3506297 0.6493703
```

c. Choose a single natural mortality surrogate method and estimate the natural mortality rate (M) for this population using your chosen surrogate method (use female-based parameters where appropriate).

Using the surrogate method from our Yield per Recruit analyses in the previous assignments we have:

$$M = -1.5K$$

```
(M = -1.5 * 0.2257593)
```

```
## [1] -0.3386389
```

d. What is the fishing mortality rate (F) for this population assuming the M you estimated using surrogate methods?

```
Z = -0.4718949
(F = Z - M)
```

```
## [1] -0.133256
```

i. Discuss your level of confidence for this F estimate.

First of all we have to consider the uncertainty in Z itself. Z ranged from -0.51 to -0.43 which is a range of -0.08. This range is itself equivalent to the value of F that we are estimated at present. Furthermore this M was taken from a surrogate method which we know is highly approximate in the first place. So I'd definitely don't have a lot of confidence in this estimate of F.

e. What specific data would you collect to obtain a potentially better estimate of Z,M,and/or F?

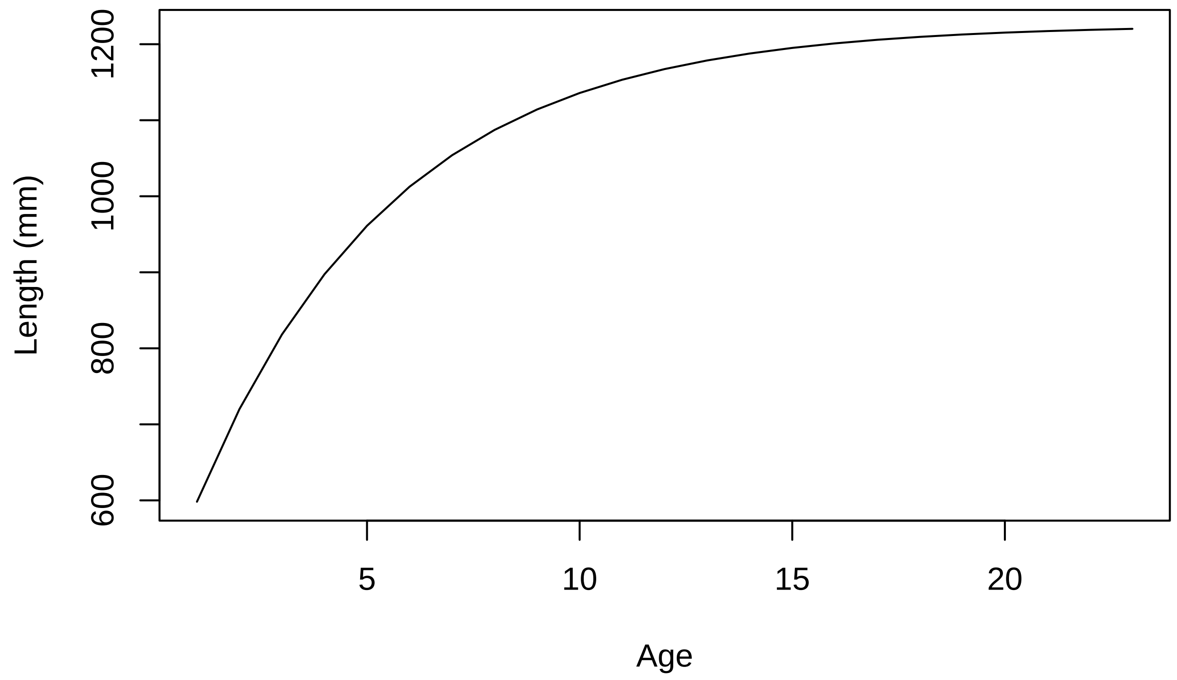
I think we have a reasonable estimate of Z here. The real trick is being able to disentangle F and M. M can be really hard to estimate on its own whereas F is more or less a matter of collecting the right data. Therefore I think it would be interesting to get an abundance estimate of the population in a given year along with a catch estimate in order to get a better sense of F and therefore M.

5. Given the estimates of length-weight, growth, maturity, and mortality you estimated above (use your top models):

a. Conduct an equilibrium Yield-per-Recruit analysis based solely on female King Mackerel.

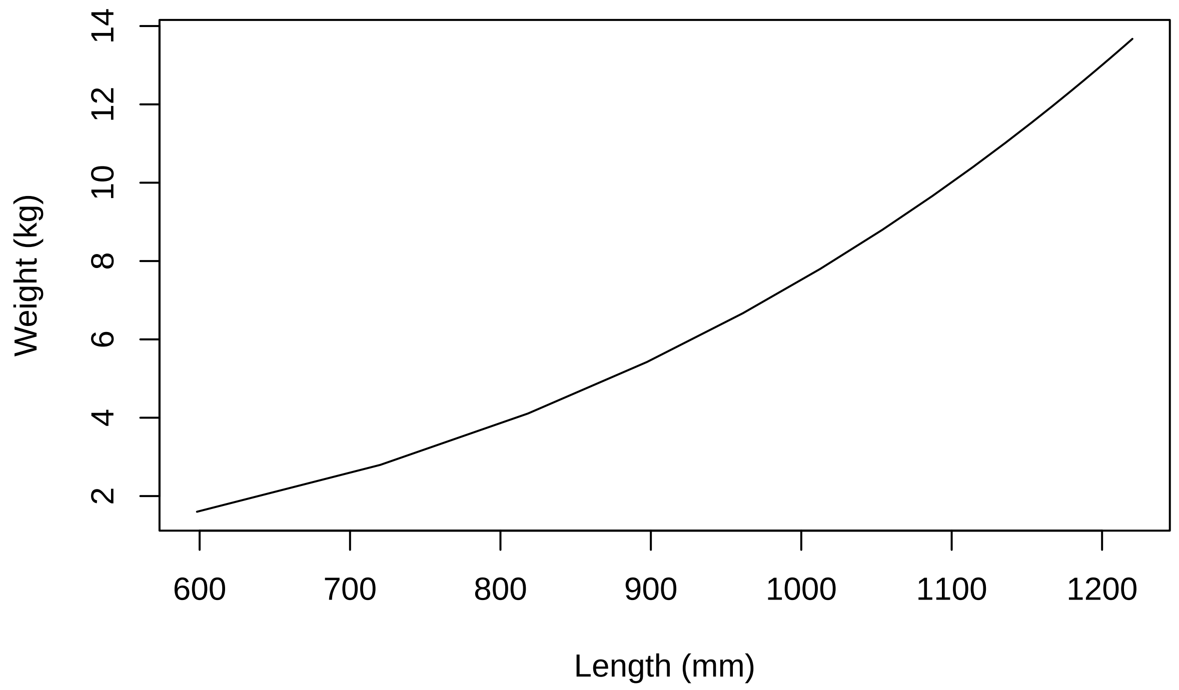
Length at Age

```r
A_max <- 23
ages <- seq(1, A_max, 1)

L_inf <- 1225.6
K <- 0.216
t_0 <- -2.1

get_lengths <- function(ages, L_inf, K, t_0){
  return(L_inf * (1 - exp(-K * (ages - t_0))))
}

lengths <- get_lengths(ages, L_inf, K, t_0)

plot(
  x=ages,
  y=lengths,
  type="l",
  xlab="Age",
  ylab="Length (mm)"
)
```



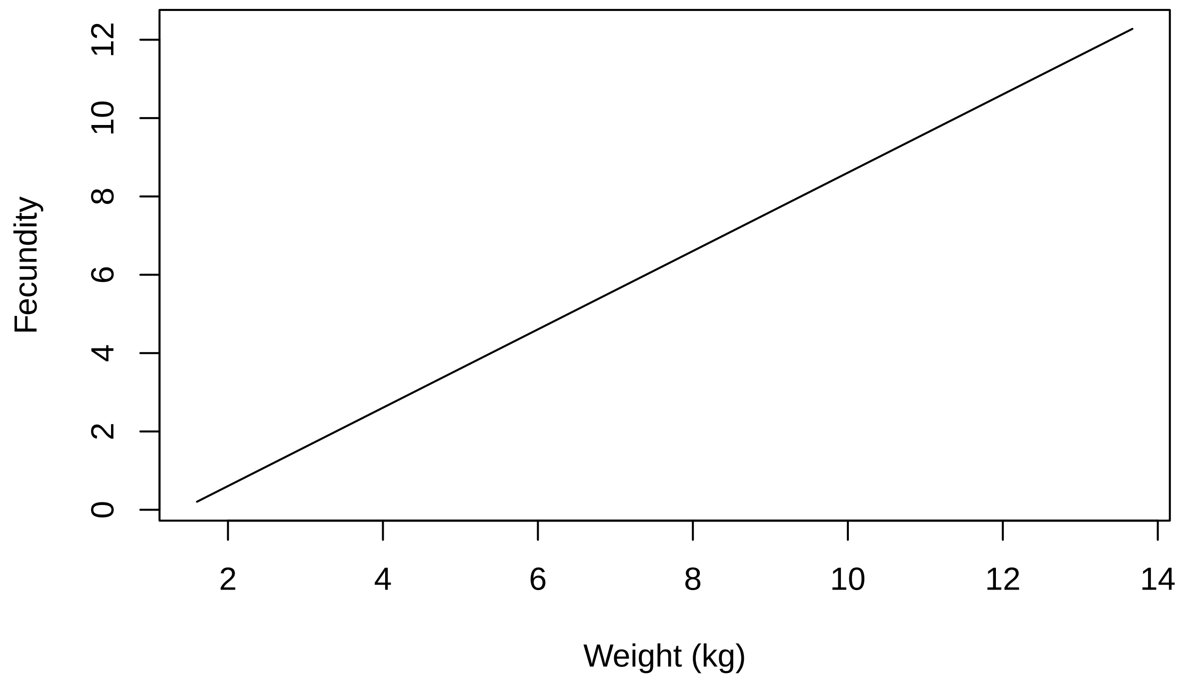
Weight at Length

```
a <- 7.01 * 10 ^ -9
b <- 3.01

get_weights <- function(lengths, a, b){
  return(a * lengths ^ b)
}

weights <- get_weights(lengths, a, b)

plot(
  x=lengths,
  y=weights,
  type="l",
  xlab="Length (mm)",
  ylab="Weight (kg)"
)
```



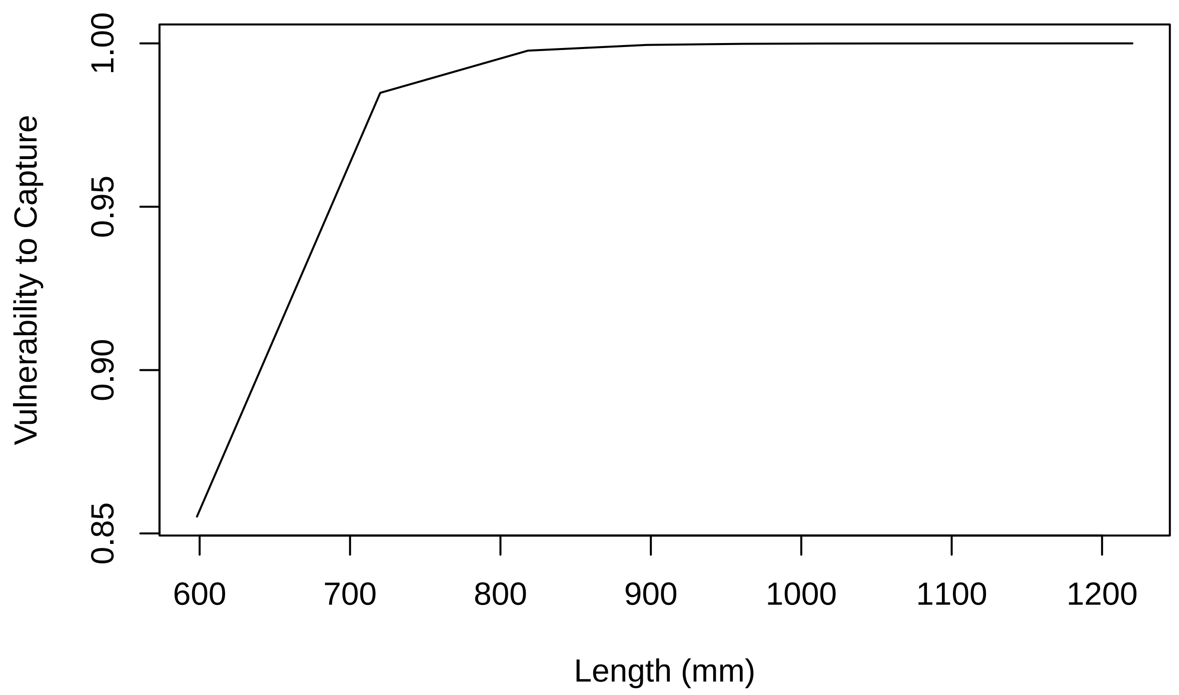
Fecundity at Weight

```
L_mat_50 <- 571.5

get_fecundity <- function(weights, L_mat_50) {
  W_mat_50 <- get_weights(L_mat_50, a, b)
  return(pmax(0, weights - W_mat_50))
}

fecundity <- get_fecundity(weights, L_mat_50)

plot(
  x=weights,
  y=fecundity,
  type="l",
  xlab="Weight (kg)",
  ylab="Fecundity"
)
```



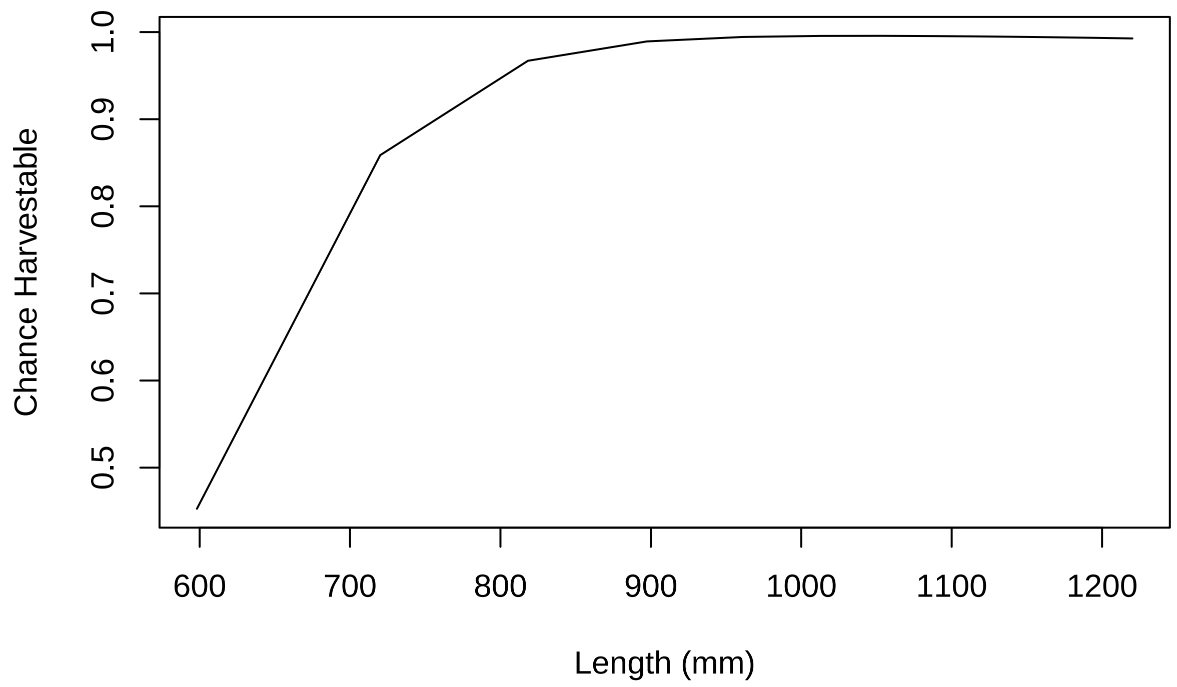
Vulnerability to Capture

```r
length_sigma <- 0.1
L_cap_50 <- 508

get_vul_cap <- function(lengths, L_cap_50, length_sigma) {
  sigma <- L_cap_50 * length_sigma
  return(
    1 / (1 + exp((L_cap_50 - lengths)/sigma))
  )
}

vul_cap <- get_vul_cap(lengths, L_cap_50, length_sigma)

plot(
  x=lengths,
  y=vul_cap,
  type="l",
  xlab="Length (mm)",
  ylab="Vulnerability to Capture"
)
```



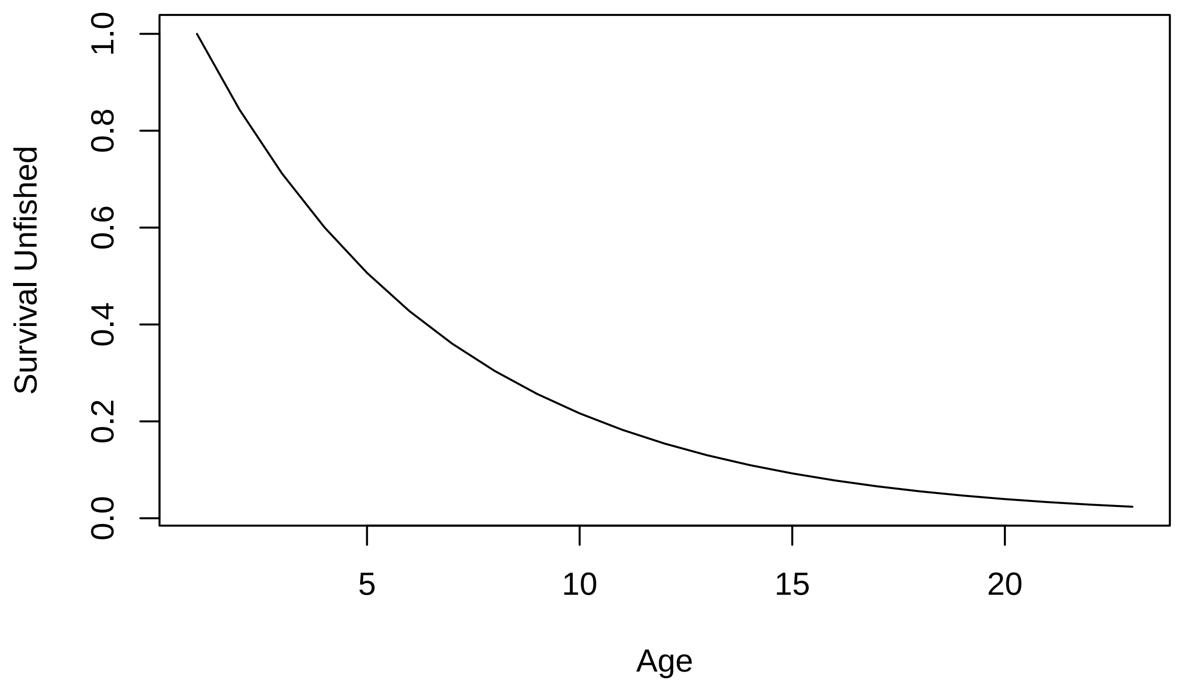
Chance Harvestable

```
minLL <- 609.6
maxLL <- 2400

get_chance_harvestable <- function(lengths, minLL, maxLL, length_sigma) {
  max_sigma <- maxLL * length_sigma
  min_sigma <- minLL * length_sigma
  return(
    1 / (1 + exp((minLL - lengths)/min_sigma))
    - 1 / (1 + exp((maxLL - lengths)/max_sigma))
  )
}

chance_harvestable <- get_chance_harvestable(
  lengths, minLL, maxLL, length_sigma
)

plot(
  x=lengths,
  y=chance_harvestable,
  type="l",
  xlab="Length (mm)",
  ylab="Chance Harvestable"
)
```



Survival Unfished

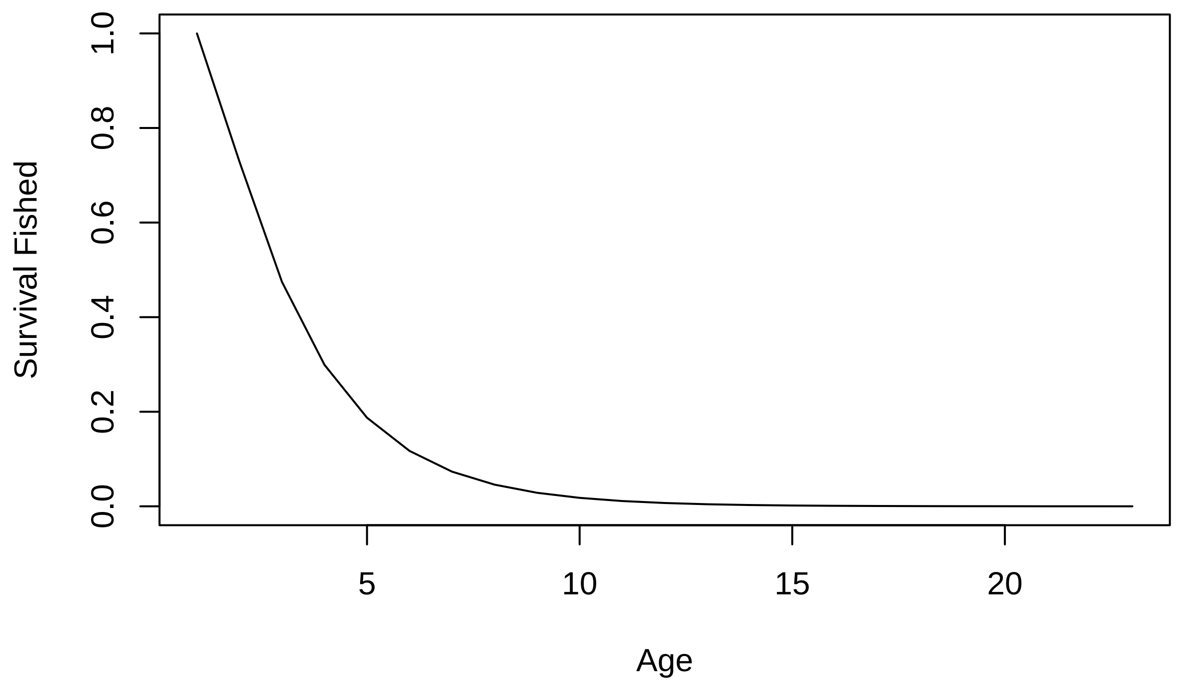
```
get_S <- function(M) {
  return(exp(M))
}

get_natural_factor <- function(A_max, S) {
    return(c(1, rep(S, A_max - 1)))
}

M <- -0.17
natural_factor <- get_natural_factor(A_max, get_S(M))

plot(
  x=ages,
  y=cumprod(natural_factor),
  type="l",
  xlab="Age",
  ylab="Survival Unfished"
)
```



Survival Fished

```
U_harv <- 0.26
U_caught <- 0.0
D <- 0.2

get_harvest_factor <- function(
  U_harv, vul_cap, chance_harvestable
) {
  return(
    c(
      1, 1 - U_harv * chance_harvestable[1:length(chance_harvestable)-1]
    )
  )
}

get_discard_factor <- function(
    U_harv, U_caught, vul_cap, chance_harvestable, D
) {
  U <- U_harv + U_caught
  return(
    c(
      1, 1 - (
        U * vul_cap[1:length(vul_cap)-1]
        - U_harv * chance_harvestable[1:length(chance_harvestable)-1]
```

```r
      ) * D
    )
  )
}

harvest_factor <- get_harvest_factor(
  U_harv, vul_cap, chance_harvestable
)

discard_factor <- get_discard_factor(
  U_harv, U_caught, vul_cap, chance_harvestable, D
)

plot(
  x=ages,
  y=cumprod(natural_factor * harvest_factor * discard_factor),
  type="l",
  xlab="Age",
  ylab="Survival Fished"
)
```



b. Report in a table all the parameters you used to conduct this analysis.

Parameter	Value
a_{max}	23
a	7.01e-09
b	3.01
L_∞	1225.6
K	0.216
t_0	-2.1
L_{mat}	571.5
M	-0.17
F	-0.302
U_{harv}	0.26
U_{caught}	0.0
D	0.2
L_{cap}	508
$minLL$	609.6
$maxLL$	2400

c. Report in a table the life history, vulnerability, and survival vectors you used to conduct this analysis.

```r
build_table <- function(
  A_max, # ages to consider
  L_inf, K, t_0, # length params
  a, b, # weight params
  L_mat_50, # fecundity params
  L_cap_50, # vulnerability params
  minLL, maxLL, # harvestability params
  M, U_harv, U_caught, D, # survival params
  length_sigma # variability params
) {
  # build all of the modeling components
  ages <- seq(1, A_max, 1)
  lengths <- get_lengths(ages, L_inf, K, t_0)
  weights <- get_weights(lengths, a, b)
  fecundity <- get_fecundity(weights, L_mat_50)
  vul_cap <- get_vul_cap(lengths, L_cap_50, length_sigma)
  chance_harvestable <- get_chance_harvestable(
    lengths, minLL, maxLL, length_sigma
  )
  S <- get_S(M)
  natural_factor <- get_natural_factor(A_max, S)
  harvest_factor <- get_harvest_factor(
    U_harv, vul_cap, chance_harvestable
  )
  discard_factor <- get_discard_factor(
    U_harv, U_caught, vul_cap, chance_harvestable, D
  )
  tab <- rbind(
    lengths, weights, vul_cap, chance_harvestable, cumprod(natural_factor),
    cumprod(natural_factor * harvest_factor * discard_factor), fecundity
  )
  colnames(tab) <- ages
  rownames(tab) <- c(
    "length", "weight", "vul_cap", "vul_harvest", "survival_unfished",
    "survival_fished", "fecundity"
  )
  return(tab)
}

tab <- build_table(
  A_max, L_inf, K, t_0, a, b, L_mat_50, L_cap_50, minLL, maxLL, M, U_harv,
  U_caught, D, length_sigma
)
round(tab, 2)
```

```
##                         1       2       3       4       5        6        7        8
## length             598.20  720.08  818.29  897.41  961.17  1012.54  1053.93  1087.28
## weight               1.60    2.80    4.11    5.42    6.67     7.80     8.80     9.66
## vul_cap              0.86    0.98    1.00    1.00    1.00     1.00     1.00     1.00
## vul_harvest          0.45    0.86    0.97    0.99    0.99     1.00     1.00     1.00
## survival_unfished    1.00    0.84    0.71    0.60    0.51     0.43     0.36     0.30
## survival_fished      1.00    0.73    0.47    0.30    0.19     0.12     0.07     0.05
## fecundity            0.21    1.40    2.71    4.03    5.27     6.40     7.40     8.27
##                          9       10       11       12       13       14       15
## length             1114.15  1135.80  1153.24  1167.30  1178.63  1187.75  1195.10
## weight               10.40    11.02    11.54    11.97    12.32    12.61    12.84
## vul_cap               1.00     1.00     1.00     1.00     1.00     1.00     1.00
## vul_harvest           1.00     0.99     0.99     0.99     0.99     0.99     0.99
## survival_unfished     0.26     0.22     0.18     0.15     0.13     0.11     0.09
## survival_fished       0.03     0.02     0.01     0.01     0.00     0.00     0.00
## fecundity             9.01     9.63    10.14    10.57    10.92    11.21    11.45
##                         16       17       18       19       20       21       22
## length             1201.03  1205.80  1209.65  1212.75  1215.24  1217.26  1218.88
## weight               13.04    13.19    13.32    13.42    13.51    13.57    13.63
## vul_cap               1.00     1.00     1.00     1.00     1.00     1.00     1.00
## vul_harvest           0.99     0.99     0.99     0.99     0.99     0.99     0.99
## survival_unfished     0.08     0.07     0.06     0.05     0.04     0.03     0.03
## survival_fished       0.00     0.00     0.00     0.00     0.00     0.00     0.00
## fecundity            11.64    11.80    11.93    12.03    12.11    12.18    12.23
##                         23
## length             1220.18
## weight               13.67
## vul_cap               1.00
## vul_harvest           0.99
## survival_unfished     0.02
## survival_fished       0.00
## fecundity            12.28
```

d. Do you believe that this King Mackerel population is undergoing either growth or recruitment overfishing?

```r
get_metrics <- function(tab, U_harv) {
    results <- c(
        sum(tab["survival_fished",] * tab["fecundity",]) / sum(tab["survival_unfished",] * tab["fecundi
        sum(tab['weight',] * tab['vul_harvest',] * tab['survival_fished',]) * U_harv
    )
    return(results)
}

get_metrics(tab, U_harv)
```

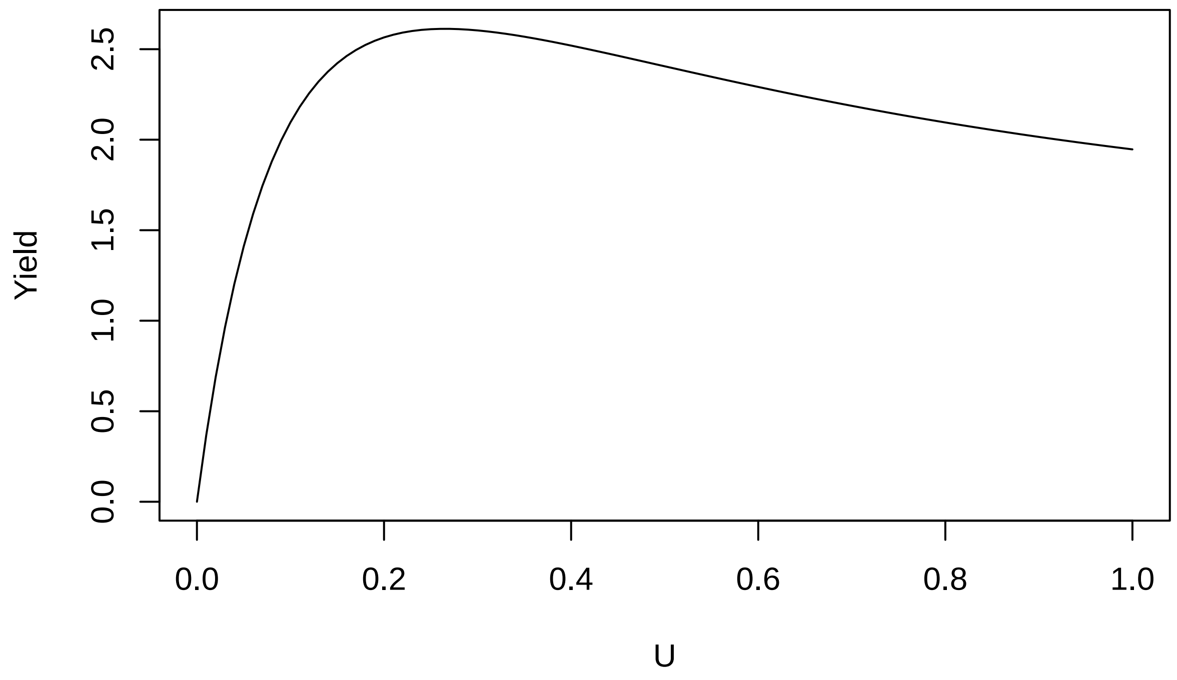
```
## [1] 0.2202678 2.6122947
```

Given SPR is 0.21 and the target is 0.35 we are definitely undergoing recruitment overfishing.

```r
U_vec <- seq(0, 1, 0.01)
yield_vec <- c()
for (U in U_vec) {
    tab <- build_table(
        A_max, L_inf, K, t_0, a, b, L_mat_50, L_cap_50, minLL, maxLL, M, U,
        U_caught, D, length_sigma
    )
    metrics <- get_metrics(tab, U)
    yield_vec <- c(yield_vec, metrics[2])
}

(U_vec[which.max(yield_vec)])
```

```
## [1] 0.27
```

```r
plot(
    x=U_vec,
    y=yield_vec,
    type="l",
    xlab="U",
    ylab="Yield"
)
```



However it seems our yield is maximized at U = 0.27 which is just slightly higher than our U = 0.26. So growth overfishing is not occurring.

e. What would be the minimum length limit necessary to maintain high yields and prevent growth and recruitment overfishing given the U?

```
minLL_vec <- seq(0, 1500, 25)
yield_at_U_vec <- c()
U_at_max_yield_vec <- c()
spr_at_U_vec <- c()
U_vec <- seq(0, 1, 0.01)
for (new_minLL in minLL_vec) {
    yield_vec <- c()
    spr_vec <- c()
    for (U in U_vec) {
        tab <- build_table(
            A_max, L_inf, K, t_0, a, b, L_mat_50, L_cap_50, new_minLL, maxLL, M, U,
            U_caught, D, length_sigma
        )
        metrics <- get_metrics(tab, U)
        yield_vec <- c(yield_vec, metrics[2])
        spr_vec <- c(spr_vec, metrics[1])
    }
    yield_at_U_vec <- c(yield_at_U_vec, yield_vec[which(U_vec == U_harv)])
    spr_at_U_vec <- c(spr_at_U_vec, spr_vec[which(U_vec == U_harv)])
    U_at_max_yield_vec <- c(U_at_max_yield_vec, U_vec[which.max(yield_vec)])
}
```

```
tab <- rbind(
    yield_at_U_vec, U_at_max_yield_vec, spr_at_U_vec
)
colnames(tab) <- minLL_vec
rownames(tab) <- c(
    "yield_at_U", "U_at_max_yield", "spr_at_U"
)
round(tab, 2)
```

```
##                   0   25   50   75  100  125  150  175  200  225  250  275  300
## yield_at_U     2.51 2.51 2.51 2.51 2.51 2.51 2.51 2.51 2.51 2.51 2.51 2.51 2.51
## U_at_max_yield 0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23
## spr_at_U       0.18 0.18 0.18 0.18 0.18 0.18 0.18 0.18 0.18 0.18 0.18 0.18 0.18
##                 325  350  375  400  425  450  475  500  525  550  575  600  625
## yield_at_U     2.51 2.51 2.51 2.51 2.51 2.51 2.52 2.53 2.55 2.57 2.59 2.61 2.62
## U_at_max_yield 0.23 0.23 0.23 0.23 0.23 0.23 0.24 0.24 0.24 0.25 0.26 0.26 0.27
## spr_at_U       0.18 0.18 0.18 0.18 0.18 0.19 0.19 0.19 0.20 0.20 0.21 0.22 0.23
##                 650  675  700  725  750  775  800  825  850  875  900  925  950
## yield_at_U     2.63 2.64 2.64 2.63 2.62 2.59 2.56 2.52 2.48 2.42 2.35 2.28 2.20
## U_at_max_yield 0.28 0.28 0.29 0.30 0.31 0.31 0.32 0.32 0.33 0.33 0.34 0.34 0.35
## spr_at_U       0.24 0.25 0.26 0.27 0.28 0.30 0.31 0.32 0.34 0.35 0.37 0.38 0.40
##                 975 1000 1025 1050 1075 1100 1125 1150 1175 1200 1225 1250 1275
## yield_at_U     2.11 2.02 1.92 1.81 1.70 1.59 1.48 1.37 1.26 1.15 1.05 0.95 0.86
## U_at_max_yield 0.35 0.36 0.36 0.37 0.37 0.38 0.39 0.39 0.40 0.41 0.41 0.42 0.43
## spr_at_U       0.42 0.43 0.45 0.46 0.48 0.49 0.51 0.52 0.54 0.55 0.56 0.57 0.58
##                1300 1325 1350 1375 1400 1425 1450 1475 1500
## yield_at_U     0.78 0.70 0.63 0.56 0.50 0.45 0.40 0.36 0.32
## U_at_max_yield 0.44 0.44 0.45 0.46 0.46 0.47 0.48 0.48 0.49
## spr_at_U       0.59 0.60 0.61 0.61 0.62 0.63 0.63 0.63 0.64
```

In order to prevent growth and recruitment fishing we can simply filter down our data:

```
t_tab <- data.frame(t(tab))
rownames(t_tab) <- minLL_vec
t_tab <- t_tab[t_tab$spr_at_U >= 0.35, ]
t_tab <- t_tab[t_tab$U_at_max_yield >= 0.26, ]
t_tab
```

```
##      yield_at_U U_at_max_yield  spr_at_U
## 875   2.4197729           0.33 0.3526074
## 900   2.3548763           0.34 0.3680005
## 925   2.2817712           0.34 0.3836963
## 950   2.2008676           0.35 0.3996211
## 975   2.1127287           0.35 0.4156917
## 1000  2.0180718           0.36 0.4318158
## 1025  1.9177654           0.36 0.4478931
## 1050  1.8128195           0.37 0.4638159
## 1075  1.7043699           0.37 0.4794716
## 1100  1.5936546           0.38 0.4947453
## 1125  1.4819818           0.39 0.5095231
## 1150  1.3706909           0.39 0.5236968
## 1175  1.2611048           0.40 0.5371686
## 1200  1.1544777           0.41 0.5498561
## 1225  1.0519406            0.41 0.5616963
## 1250  0.9544523            0.42 0.5726486
## 1275  0.8627619            0.43 0.5826959
## 1300  0.7773885            0.44 0.5918432
## 1325  0.6986208            0.44 0.6001154
## 1350  0.6265345            0.45 0.6075531
## 1375  0.5610229            0.46 0.6142086
## 1400  0.5018356            0.46 0.6201414
## 1425  0.4486181            0.47 0.6254145
## 1450  0.4009491            0.48 0.6300915
## 1475  0.3583719            0.48 0.6342343
## 1500  0.3204203            0.49 0.6379013
```

Our highest yield is acheived at a slot limit of 875mm. Therefore the minimum slot limit required to maintain high yields while ensuring we have neither growth or recruitment overfishing is 875mm.