Lab 8

Adam Shen

November 11, 2020

Packages

```r
library(EnvStats)
library(tibble)
library(dplyr)
library(broom)
library(ggplot2)
theme_set(theme_bw())
```

Simple example data

```r
newt <- read.table("./simpex.txt", header=TRUE)
```

Fit a NLS model

```r
nls_model <- nls(y ~ exp(-beta1 * x), start=list(beta1=0.25), data=newt)
summary(nls_model)
```

```
##
## Formula: y ~ exp(-beta1 * x)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## beta1 0.203449   0.006002    33.9 0.000869 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01228 on 2 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 1.26e-06
```

Confidence intervals

```r
confint.default(nls_model)
```

```
##           2.5 %   97.5 %
## beta1 0.1916848 0.215213
```

The above confidence interval is calculated using a normal distribution rather than a t distribution, i.e.

$$\hat{\beta}_1 \quad \pm \quad z_{1-\alpha/2} \cdot \text{se}(\hat{\beta}_1)$$

We can verify this:

```
nls_model %>%
  tidy() %>%
  mutate(
    lower = estimate - qnorm(0.975)*std.error,
    upper = estimate + qnorm(0.975)*std.error
  )
```

```
## # A tibble: 1 x 7
##   term  estimate std.error statistic  p.value lower upper
##   <chr>    <dbl>     <dbl>     <dbl>    <dbl> <dbl> <dbl>
## 1 beta1    0.203   0.00600      33.9 0.000869 0.192 0.215
```

The intervals are the same!

Fit a OLS model - Method 1

```
newt <- mutate(
  newt,
  lny = log(y),
  negx = -x
)

ols_model1 <- lm(lny ~ 0 + negx, data=newt)
summary(ols_model1)
```

```
##
## Call:
## lm(formula = lny ~ 0 + negx, data = newt)
##
## Residuals:
##          1          2          3
## -0.0219744  0.0061690 -0.0001689
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## negx 0.2011692  0.0009768   205.9 2.36e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01614 on 2 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:  0.9999
## F-statistic: 4.241e+04 on 1 and 2 DF,  p-value: 2.358e-05
```

Fit a OLS model - Method 2

```
ols_model2 <- lm(log(y) ~ 0 + I(-x), data=newt)
summary(ols_model2)
```

```
##
## Call:
## lm(formula = log(y) ~ 0 + I(-x), data = newt)
##
## Residuals:
##          1          2          3
## -0.0219744  0.0061690 -0.0001689
##
## Coefficients:
```

```
##        Estimate Std. Error t value Pr(>|t|)
## I(-x) 0.2011692  0.0009768   205.9 2.36e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01614 on 2 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:  0.9999
## F-statistic: 4.241e+04 on 1 and 2 DF,  p-value: 2.358e-05
```

What's the difference?

When doing predictions, `ols_model1` will require you to supply **negative** x-values. `ols_model2` will require you to supply **positive** x-values and it will calculate the negative of x for you.

```
predict(ols_model1, newdata=data.frame(negx = -3))
```

```
##          1
## -0.6035076
```

```
predict(ols_model2, newdata=data.frame(x = 3))
```

```
##          1
## -0.6035076
```

Method 2 may be preferable as we can predict using the values on the same scale as the values originally supplied to us.

Confidence intervals

Here, we go back to calculating confidence intervals using the t distribution.

```
confint(ols_model1)
```

```
##          2.5 %   97.5 %
## negx 0.1969664 0.205372
```

```
confint(ols_model2)
```

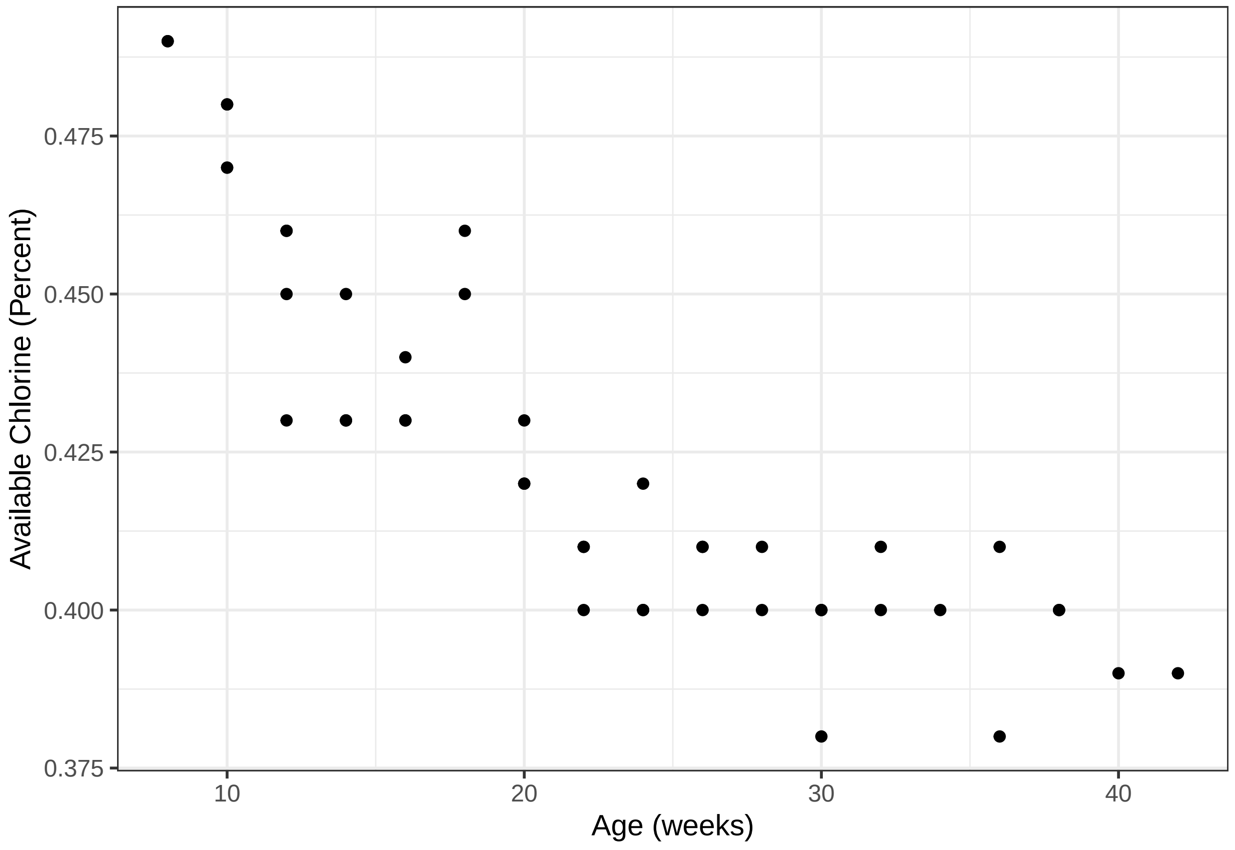
```
##           2.5 %   97.5 %
## I(-x) 0.1969664 0.205372
```

Chlorine data

```
loss <- read.table("./chlorine.txt", header=TRUE)
```

Scatterplot

```
ggplot(loss, aes(x=Age, y=Available))+
  geom_point()+
  labs(x="Age (weeks)", y="Available Chlorine (Percent)")
```



Fit a NLS model

```
nls_model <- nls(Available ~ beta1 + (0.49 - beta1) * exp(-beta2*(Age - 8)),
                 start=list(beta1=0.35, beta2=0.034), data=loss)
summary(nls_model)
```

```
##
## Formula: Available ~ beta1 + (0.49 - beta1) * exp(-beta2 * (Age - 8))
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## beta1 0.390140   0.005045   77.333  < 2e-16 ***
## beta2 0.101633   0.013360    7.607 1.99e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01091 on 42 degrees of freedom
##
```

```
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 1.601e-06
```

Confidence intervals

Using normal distribution here.

```
confint.default(nls_model)
```

```
##             2.5 %    97.5 %
## beta1 0.38025219 0.4000279
## beta2 0.07544721 0.1278185
```

Variance-covariance matrix

```
vcov(nls_model)
```

```
##              beta1        beta2
## beta1 2.545129e-05 5.984317e-05
## beta2 5.984317e-05 1.784969e-04
```

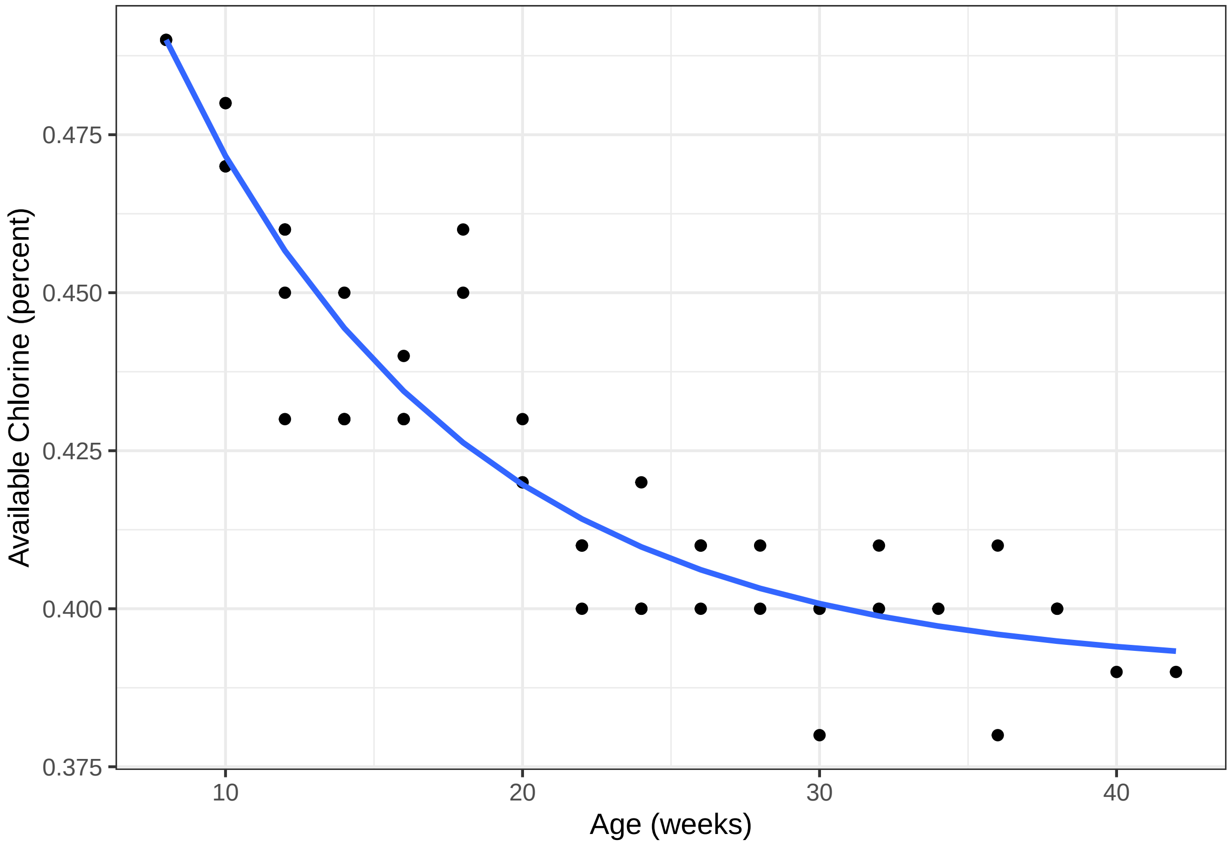
Sums of squares and related quantities

```
(nls_info <- nls_model %>%
   augment() %>%
   summarise(
     n = nrow(loss),
     `p+1` = length(coef(nls_model)),
     SSE = sum(.resid^2),
     df = n - `p+1`,
     MSE = SSE/df,
     s = sqrt(MSE),
     SST = sum(Available^2) - (sum(Available)^2 / n),
     R.sq = 1 - SSE/SST
   ))
```

```
## # A tibble: 1 x 8
##       n `p+1`     SSE    df      MSE      s    SST  R.sq
##   <int> <int>   <dbl> <int>    <dbl>  <dbl>  <dbl> <dbl>
## 1    44     2 0.00500    42 0.000119 0.0109 0.0395 0.873
```

Visualize the NLS fit

```
nls_model %>%
  augment() %>%
  ggplot(aes(x=Age))+
    geom_point(aes(y=Available))+
    geom_line(aes(y=.fitted), colour="#3366FF", size=1)+
    labs(x="Age (weeks)", y="Available Chlorine (percent)")
```



Fit OLS model to get SSPE

As mentioned in *Module 8.7*, the SSPE does not change regardless of the type of model we fit. We can obtain the SSPE values by fitting a OLS model.

```
ols_model <- lm(Available ~ Age, data=loss)
summary(ols_model)
```

```
##
## Call:
## lm(formula = Available ~ Age, data = loss)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.025741 -0.012042 -0.001608  0.012034  0.026224
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.4855103  0.0058907   82.42  < 2e-16 ***
## Age         -0.0027168  0.0002431  -11.18 3.67e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01539 on 42 degrees of freedom
## Multiple R-squared:  0.7483, Adjusted R-squared:  0.7423
## F-statistic: 124.9 on 1 and 42 DF,  p-value: 3.675e-14
```

```
anova(ols_model)
```

```
## Analysis of Variance Table
##
## Response: Available
##           Df    Sum Sq   Mean Sq F value    Pr(>F)
## Age        1 0.0295587 0.0295587  124.88 3.675e-14 ***
## Residuals 42 0.0099413 0.0002367
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(ols_model) %>%
  tidy() %>%
  summarise(SST = sum(sumsq))
```

```
## # A tibble: 1 x 1
##      SST
##    <dbl>
## 1 0.0395
```

Side note: we can also see that the SST value obtained from the OLS model is the same as the one obtained from the NLS model.

```
anovaPE(ols_model)
```

```
##                 Df    Sum Sq   Mean Sq  F value    Pr(>F)
## Age              1 0.0295587 0.0295587 324.7290 3.279e-16 ***
## Lack of Fit     16 0.0075747 0.0004734   5.2009 0.0001074 ***
## Pure Error      26 0.0023667 0.0000910
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Create anovaPE table for NLS model

```
pure_error <- anovaPE(ols_model) %>%
  tidy() %>%
  filter(term == "Pure Error")

total_error <- nls_info %>%
  mutate(term = "Total Error") %>%
  select(term, df, sumsq = SSE)

lack_of_fit <- tibble(
  term = "Lack of Fit",
  df = total_error$df - pure_error$df,
  sumsq = total_error$sumsq - pure_error$sumsq,
  meansq = sumsq / df,
  statistic = meansq / pure_error$meansq,
  p.value = pf(statistic, df1=df, df2=pure_error$df, lower.tail=FALSE)
)
```

```r
bind_rows(
  lack_of_fit,
  pure_error,
  total_error
)
```

```
## # A tibble: 3 x 6
##   term           df   sumsq     meansq statistic p.value
##   <chr>       <dbl>   <dbl>      <dbl>     <dbl>   <dbl>
## 1 Lack of Fit    16 0.00264  0.000165       1.81  0.0867
## 2 Pure Error     26 0.00237  0.0000910     NA    NA
## 3 Total Error    42 0.00500 NA             NA    NA
```

The associated hypotheses are:

$$H_0 : \text{functional form is adequate} \quad \text{vs} \quad H_A : \text{functional form is inadequate}$$

The p-value of this test is 0.0867. Since this p-value is greater than 0.05, we fail to reject the null hypothesis. We conclude that there is insufficient evidence that the functional form of the non-linear model is inadequate.