Non-linear Modelling
Solutions to Exercises

April 13, 2017

Contents

Chapter 1

Non-linear Modelling

1.6 Exercises

In these exercises, we use the following colour codes:

- 🟩 **Easy**: make sure you complete some of these before moving on. These exercises will follow examples in the text very closely.
- ♦ **Intermediate**: a bit harder. You will often have to combine functions to solve the exercise in two steps.
- ▲ **Hard**: difficult exercises! These exercises will require multiple steps, and significant departure from examples in the text.

We suggest you complete these exercises in an **R** markdown file. This will allow you to combine code chunks, graphical output, and written answers in a single, easy-to-read file.

1.6.1 Chick Weight

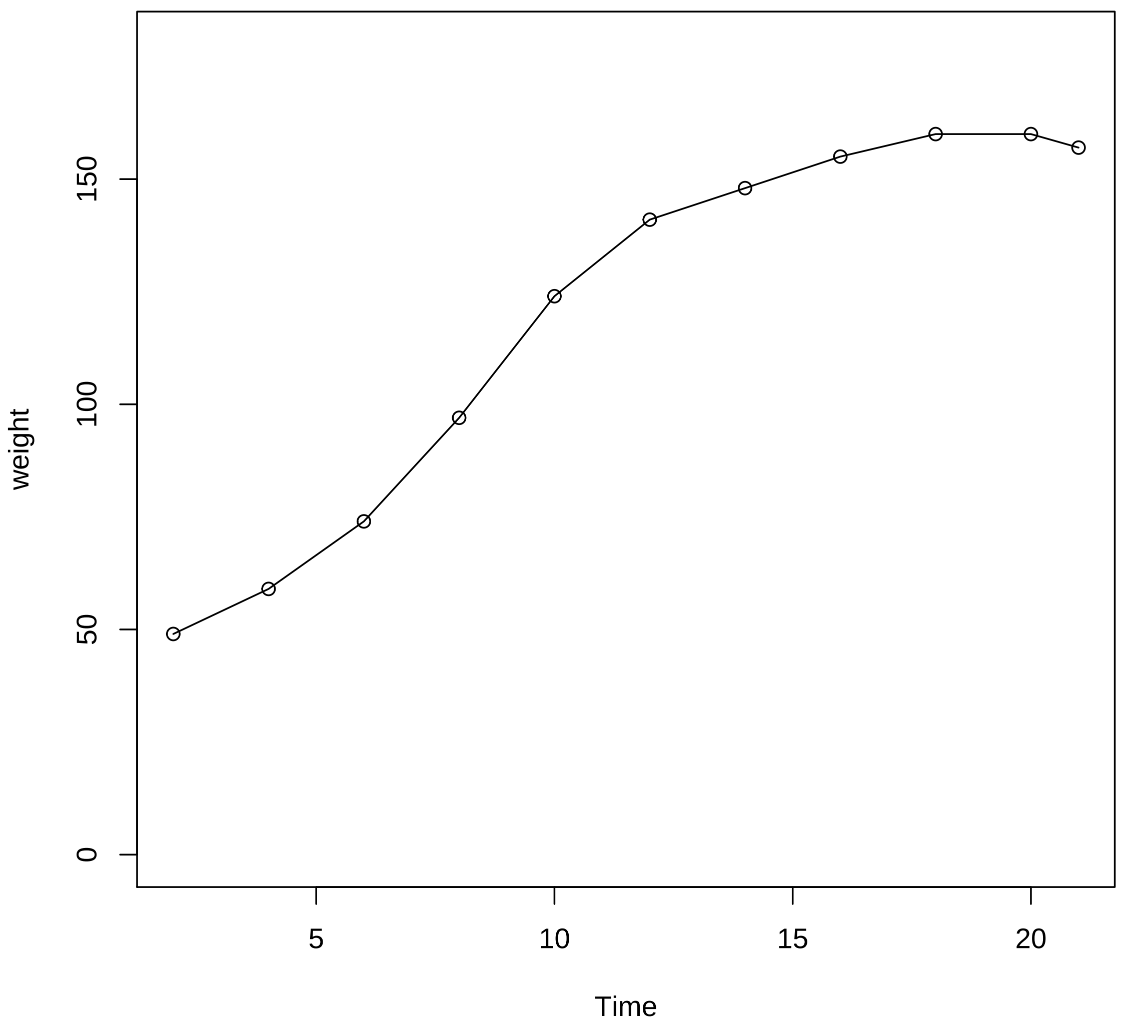
For this exercise, you will be using the `ChickWeight` data, which is a dataset already available and loaded in **R**. See `?ChickWeight` for a description.

Note that the data are in a special `groupedData` format, for the purpose of this exercise, first convert the data to a standard dataframe:

```
chickweight <- as.data.frame(ChickWeight)
```

1. Make a subset of ChickWeight, for Chick number 6, and exclude Time equals zero (i.e. `Time > 0`). Inspect the data, and make a simple line plot of weight over time for this chick.

```
Chick.6 <- subset(ChickWeight, Chick == 6 & Time > 0)

with(Chick.6, plot(Time, weight, type='o', ylim=c(0,180)))
```



2. For this subset, fit the Weibull, logistic, and Gompertz growth models to chick weight against time. Each of these models is defined as a self-starting model, thus no starting values have to be guessed.

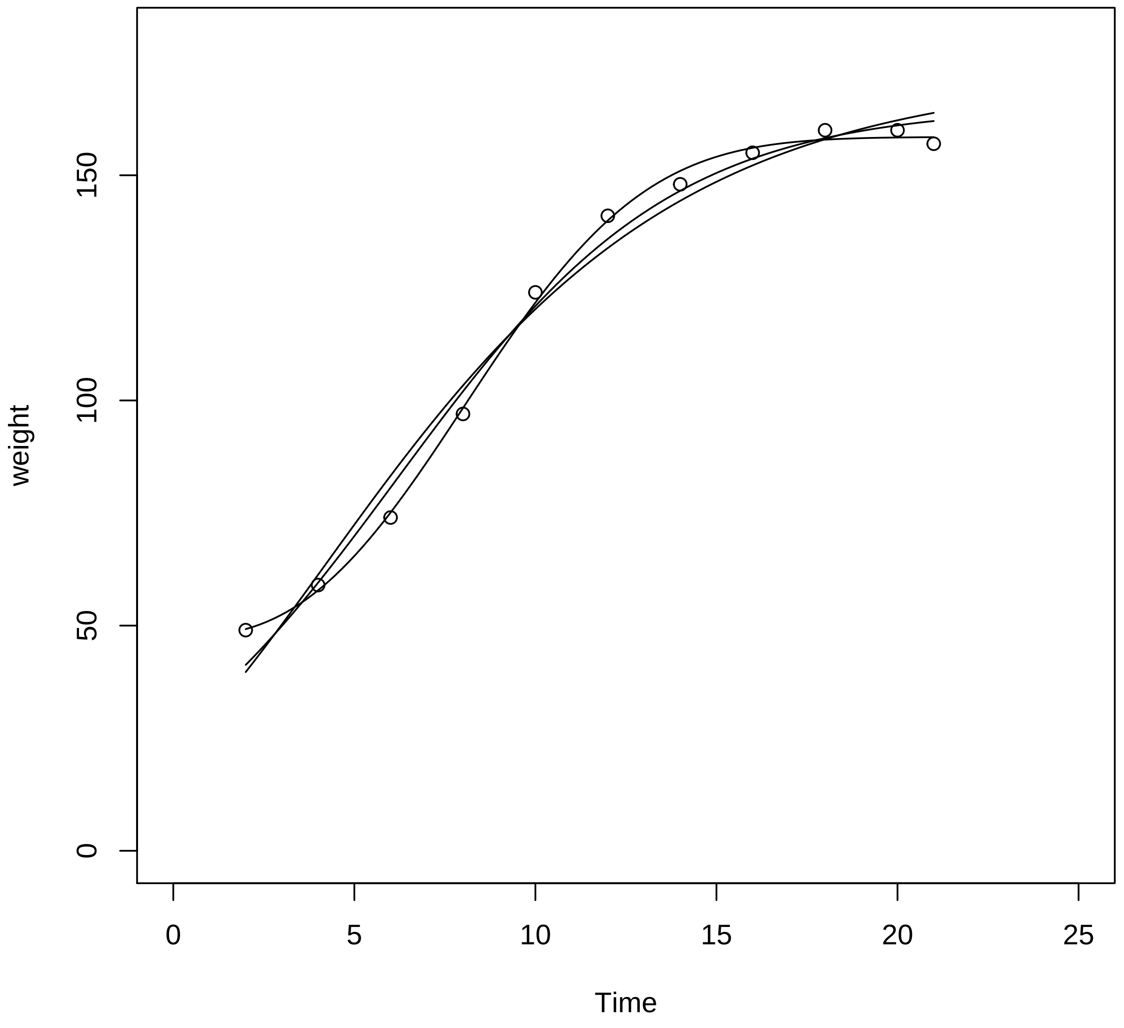
```
fm1 <- nls(weight ~ SSweibull(Time, Asym, Drop, lrc, pwr), data = Chick.6)
fm2 <- nls(weight ~ SSlogis(Time, Asym, xmid, scal), data = Chick.6)
fm3 <- nls(weight ~ SSgompertz(Time, Asym, b2, b3), data = Chick.6)
```

3. Which of the three models fits the data best? Decide based on the AIC. Also make a plot of the fitted curves on the data, either as separate panels, or all added to the same plot with the argument `add=TRUE` to the `plot_nls` function.

```
AIC(fm1, fm2, fm3)
```

```
##     df      AIC
## fm1  5 52.14991
## fm2  4 71.02595
## fm3  4 77.59143
```

```
library(nlshelper)
plot_nls(fm1, xlim=c(0,25), ylim=c(0,180))
plot_nls(fm2, add=TRUE)
plot_nls(fm3, add=TRUE)
```



4. For the best-fitting model, predict the weight of the chick at time 25 (see Section **??**).

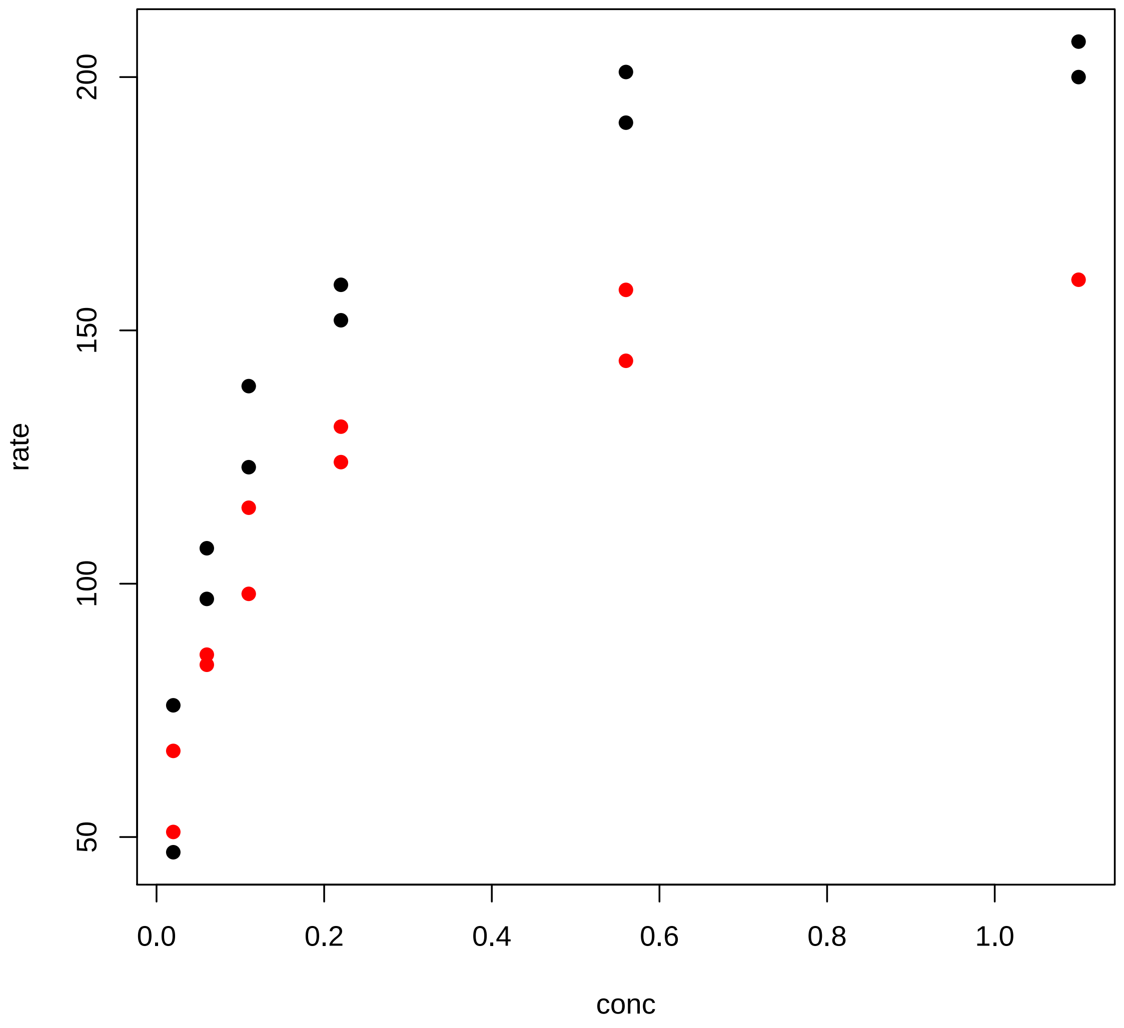
1.6.2 Michaelis-Menten kinetics

For this example we use the built-in dataset `Puromycin`, which can be accessed without loading a package. The data include reaction velocity (`rate`) versus subtrate concentration in an enzymatic reaction for cells treated with the antibiotic Puromycin, or an untreated control.

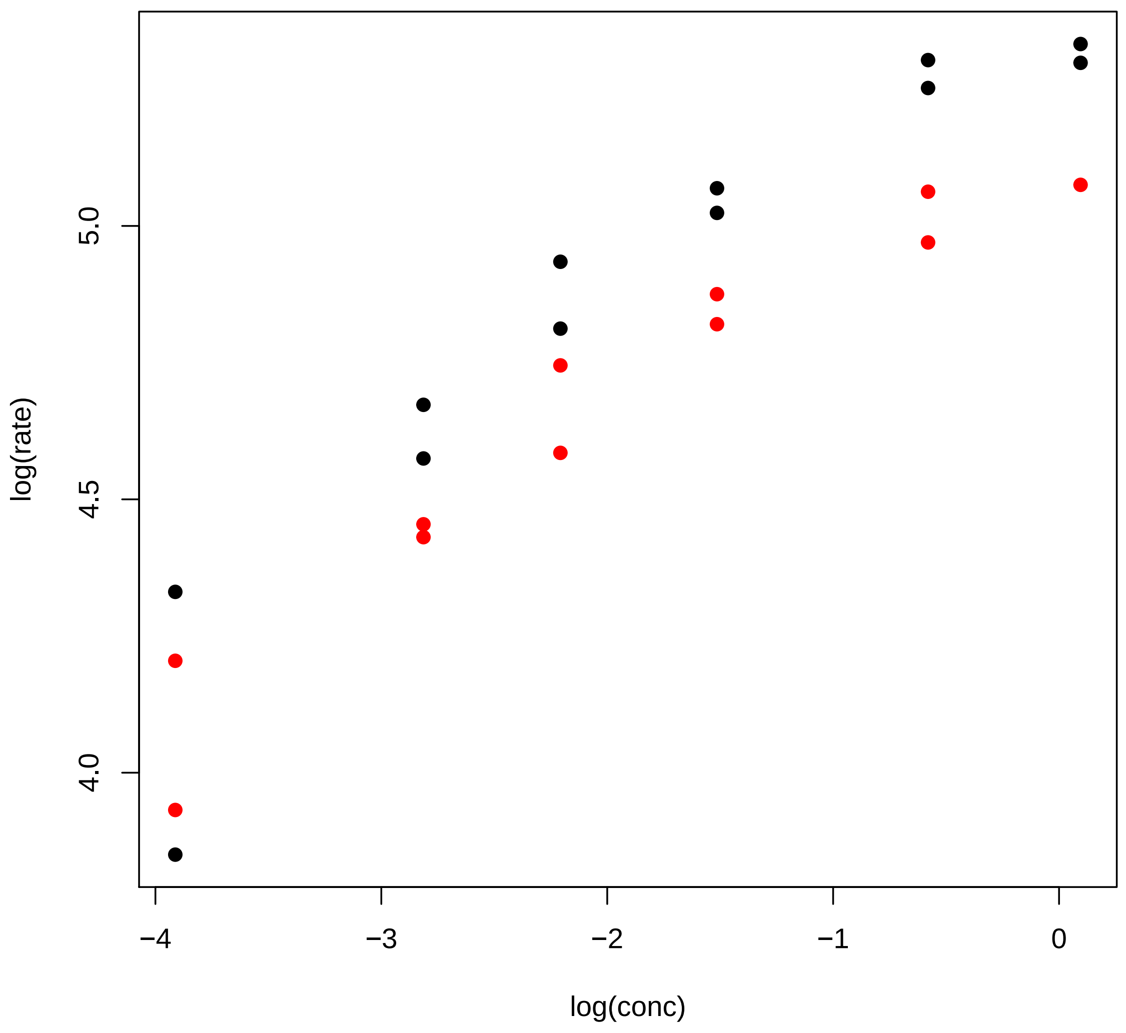
For enzymatic reactions that depend on the concentration of the substrate, the Michaelis-Menten model is often used, and follows from simple assumptions on the reaction rate versus the concentration of the substrate and enzyme.

1. Plot `rate` against `conc`, and colour the symbols by `state` (treated or untreated). Confirm visually that taking the log of both sides does not linearize the relationship.

```
with(Puromycin, plot(conc, rate, pch=19, col=state))
```



```
with(Puromycin, plot(log(conc), log(rate), pch=19, col=state))
```

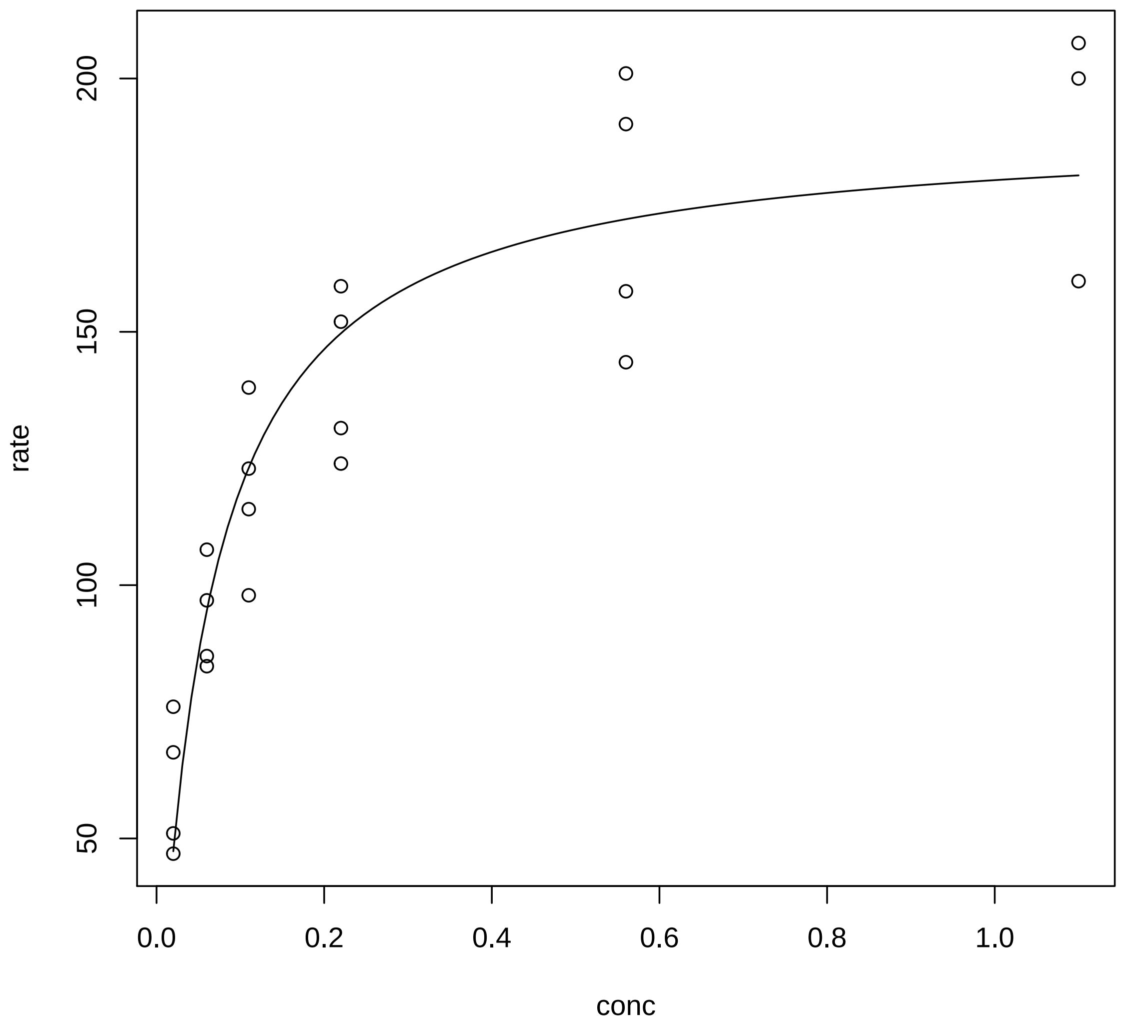


2. Fit the Michaelis-Menten model to the whole dataset, using `nls` and `SSmicmen`. Inspect the summary, make a plot of the data with the fitted curve, and extract the confidence interval of the fitted parameters.

```
pur0 <- nls(rate ~ SSmicmen(conc, Vm, K), data=Puromycin)

summary(pur0)

##
## Formula: rate ~ SSmicmen(conc, Vm, K)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## Vm 190.80667    8.76462  21.770 6.84e-16 ***
## K    0.06039    0.01077   5.608 1.45e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 18.61 on 21 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 6.745e-06
```

```
library(nlshelper)
plot_nls(pur0)
```



```
tidy(pur0, conf.int=TRUE)
```

```
##   term     estimate  std.error statistic      p.value    conf.low
## 1   Vm 190.80666560 8.76461800  21.77011 6.835453e-16 172.6840194
## 2    K   0.06038937 0.01076862   5.60790 1.448648e-05   0.0398126
##      conf.high
## 1 210.97021027
## 2   0.08881351
```

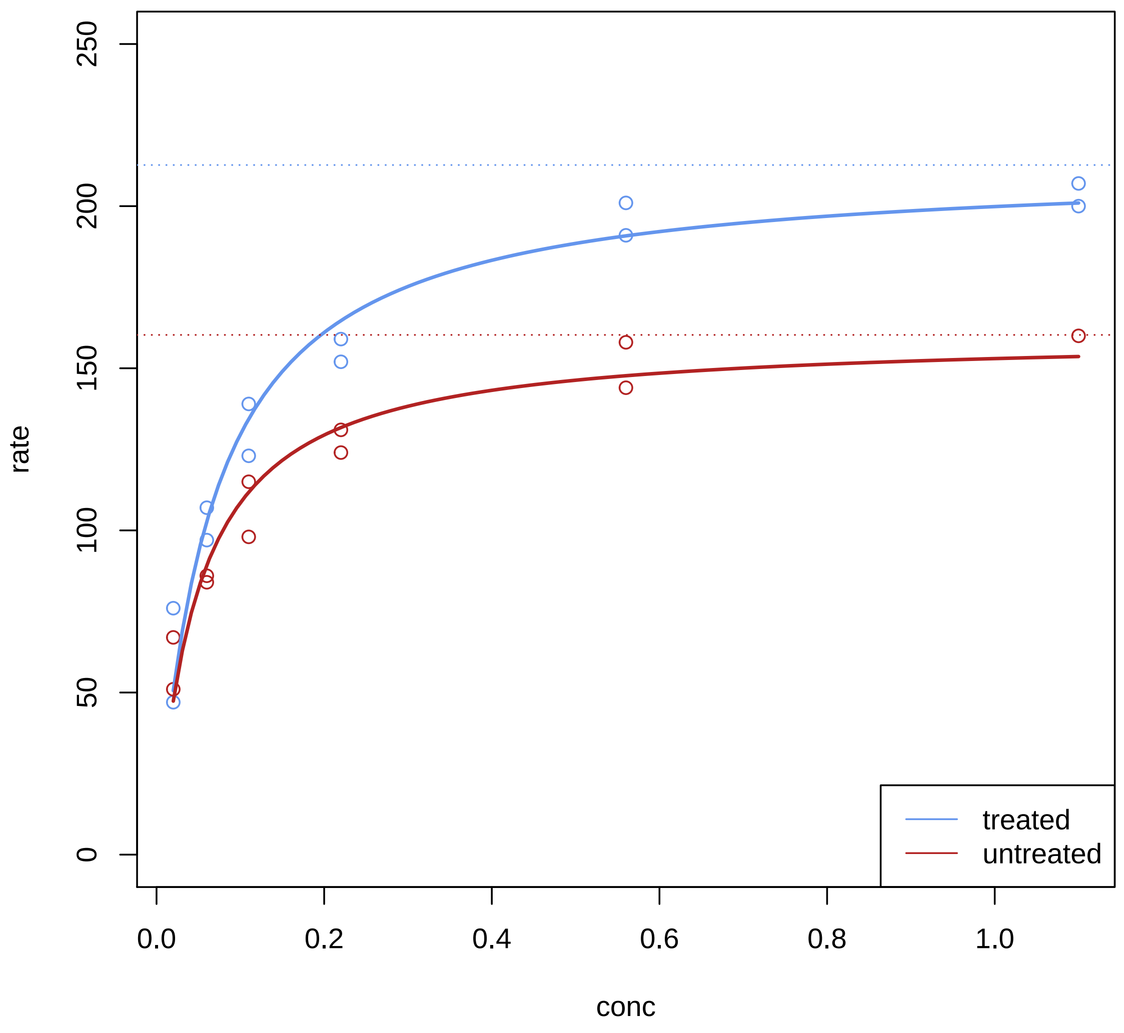
3. Now fit the curve by `state`, using `nlsList`. Perform an F-test comparing the model with and without `state`, using `anova_nlslist` (see Section **??**). Is there evidence for `state` affecting the reaction velocity?

```r
pur1 <- nlsList(rate ~ SSmicmen(conc, Vm, K)|state, data=Puromycin)

# Lots of evidence (F value 24, p < 0.00001)
anova_nlslist(pur1, pur0)
```

```
## Analysis of Variance Table
##
## Model 1: rate ~ SSmicmen(conc, Vm, K)
## Model 2: rate ~ SSmicmen(conc, Vm, K) | state
##   Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
## 1     21     7276.5
## 2     19     2055.1  2 5221.5  24.138 3.608e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. Make a plot of the `nlsList` model with `plot_nls`, make sure the y-axis limit is large enough to include the `Vmax` estimated for each level of `state`. Now add the estimated `Vmax` parameters to the plot as horizontal lines, with `abline(h=...)`.

```r
cols <- c("cornflowerblue","firebrick")
plot_nls(pur1, col=cols, lwd=2, ylim=c(0,250))
p <- coef(pur1)
abline(h = p["treated","Vm"], col=cols[1], lty=3)
abline(h = p["untreated","Vm"], col=cols[2], lty=3)
legend("bottomright", levels(Puromycin$state), col=cols, lty=1)
```



1.6.3 Brunhilda the radioactive baboon

The data used in this exercise were extracted from `http://www.statsci.org/data/general/brunhild.html`. The observed responses are Geiger counter counts (times 10-4) used to measure the amount of radioactively tagged sulfate drug in the blood of a baboon named Brunhilda after an injection of the drug. The variables are `Hours` (time since injection), `Sulfate` (Geiger counter counts).

One researcher claims that a simple exponential decay function should describe the data well:

$$Y = A0 + A * exp(-k * X)$$

where Y is the response variable, X the predictor, and `A0`, `A` and `k` are parameters to be estimated.

A second researcher does not believe her and claims a bi-exponential decay is necessary, which is a mixture of two simple exponential decays (see `?SSbiexp` for details).

1. Follow the link above, and download the data file ('brunhild.txt'), place it in your working directory. Read it in with `read.delim` (it is a TAB delimited file).

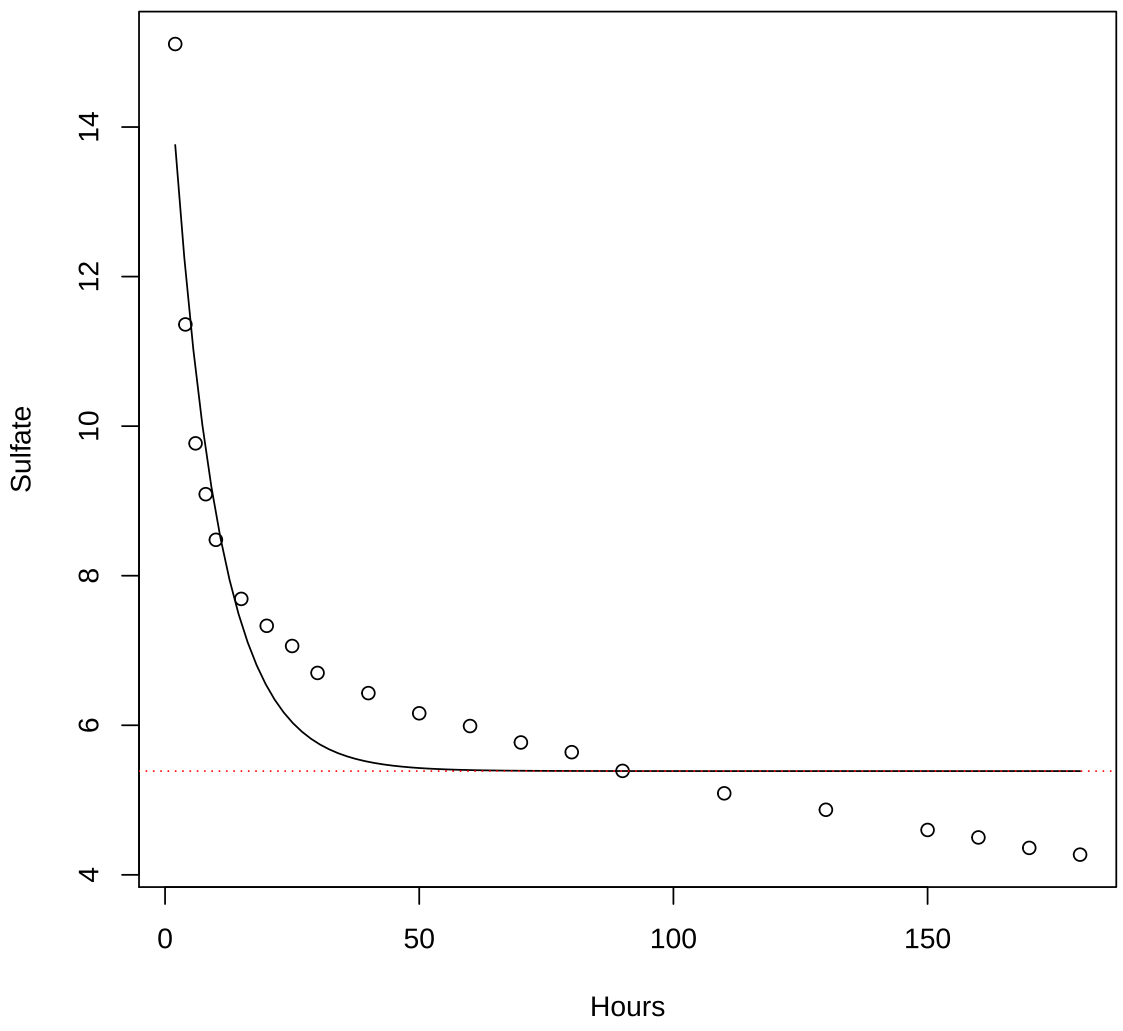
```r
brunhild <- read.delim("brunhild.txt")
```

2. Fit the simple exponential decay function defined above to the data. You may have to set reasonable starting values. When setting the starting values, consider what the `A0` parameter represents. Also don't forget to plot the data. *Hint:* if you have trouble finding starting values that make the model converge, try `A0=4, A=10, k=0.5`.

```r
# A0 is the value of the curve when Hours --> infinity.
# From reading the figure, this is about 4.
# A+A0 is the value when Hours --> 0, which is about 14, so
# A is about 10.
# Finally you have to set k to a smallish value for the model to converge.
fit1 <- nls(Sulfate ~ A0 + A*exp(-k*Hours), data=brunhild,
            start=list(A0=4, A=10, k=0.5))
```

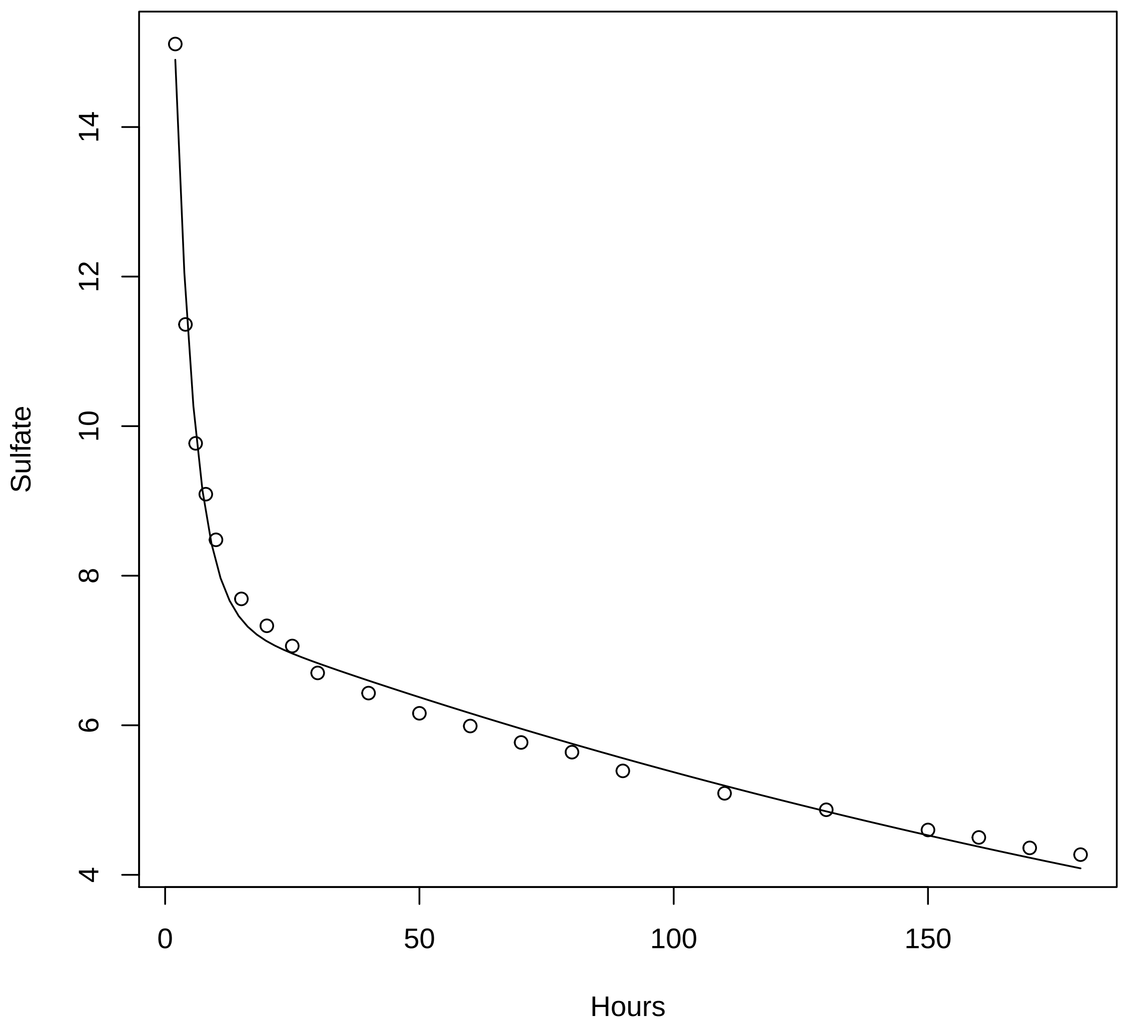
3. Plot the data with the fitted curve. Note that an asymptote is reached, confirm that one of the parameters in the equation gives the value of the asymptote (and add it with a dashed line, using `abline`).

```r
library(nlshelper)
plot_nls(fit1)

p <- coef(fit1)
abline(h=p["A0"], col="red", lty=3)
```



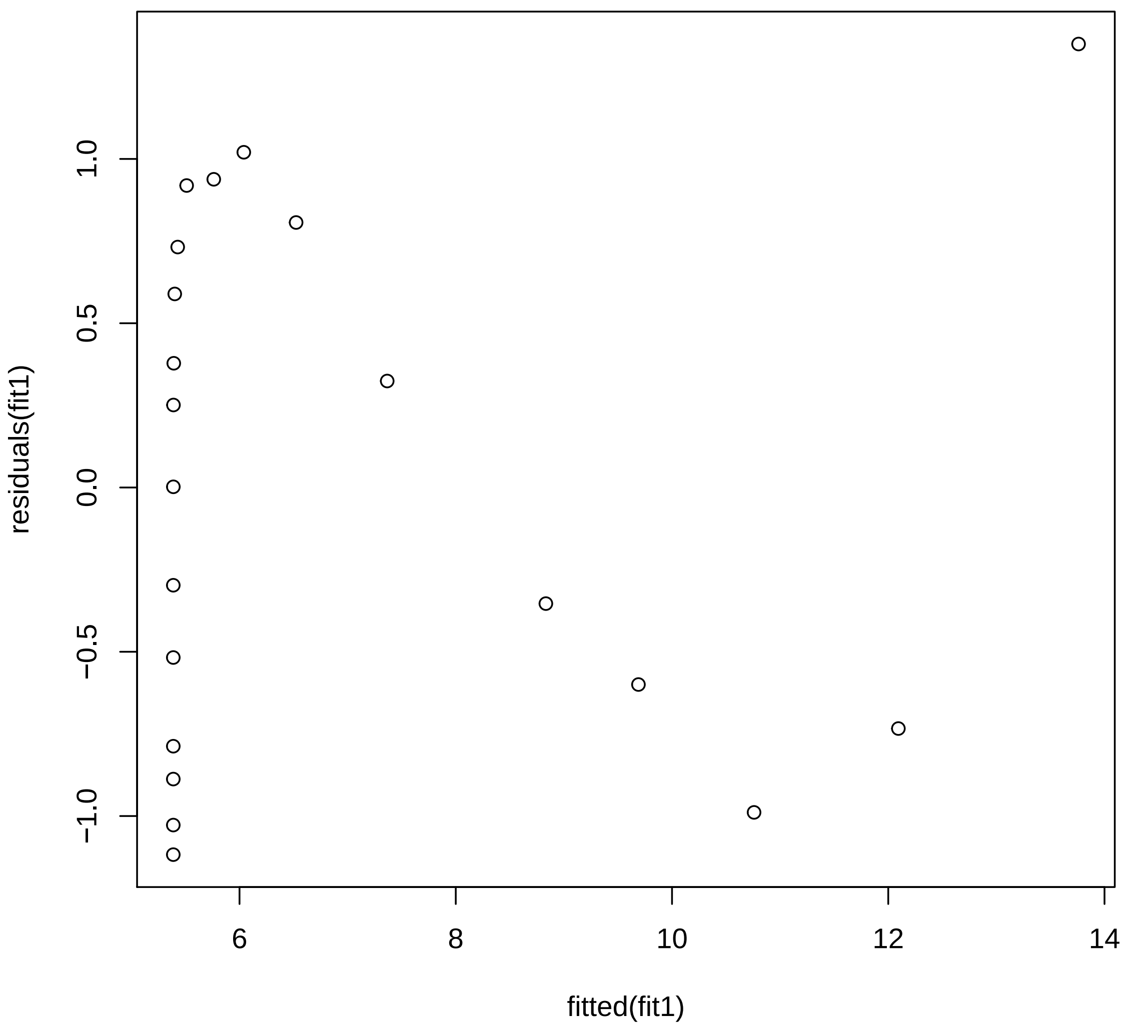
4. Fit the bi-exponential model (`SSbiexp`), plot the data with the curve.

```
fit2 <- nls(Sulfate ~ SSbiexp(Hours, A1, lrc1, A2, lrc2), data=brunhild)

plot_nls(fit2)
```

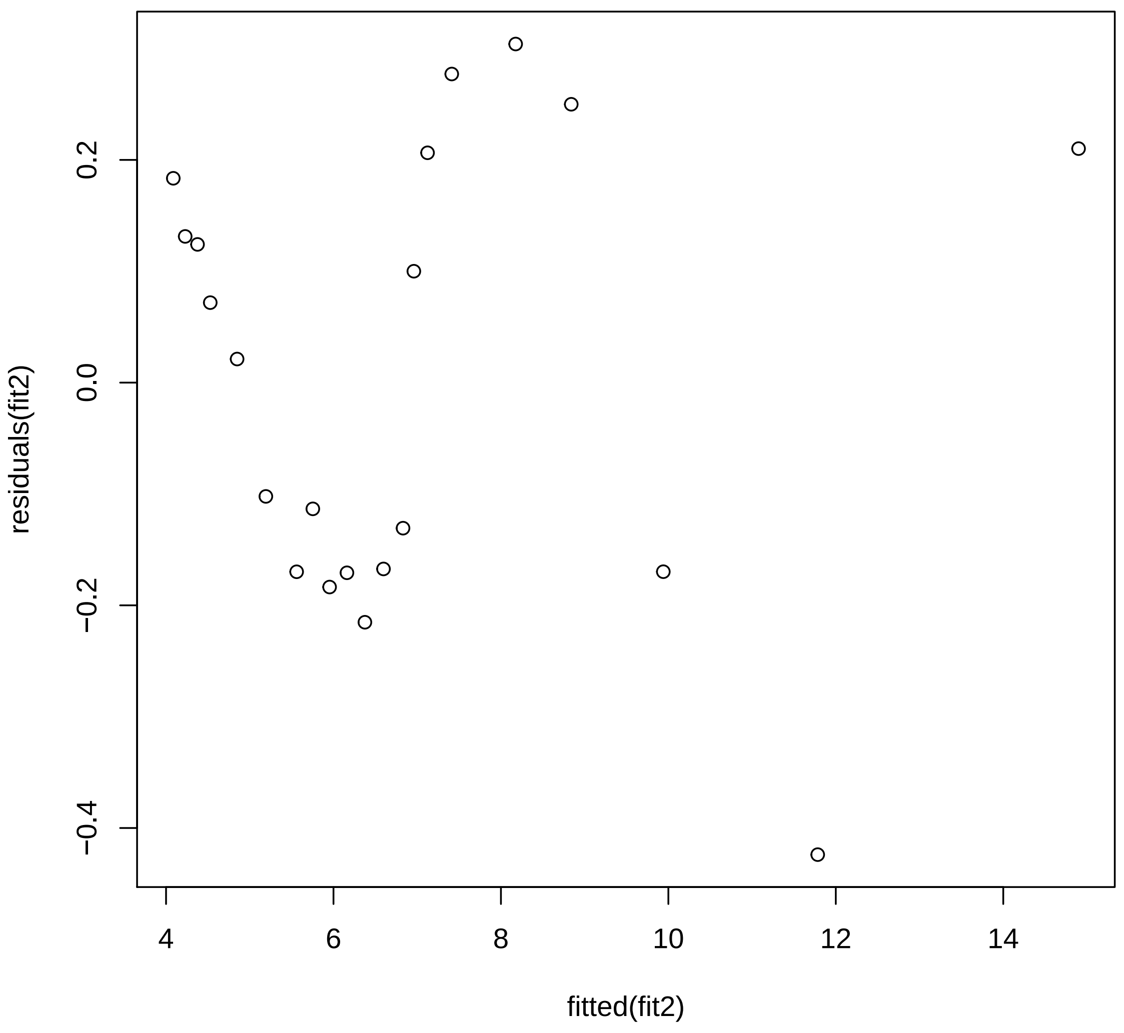


5. Compare residual plots vs. fitted values, RMSE, and AIC of the two models.

```
plot(fitted(fit1), residuals(fit1))
```



```r
plot(fitted(fit2), residuals(fit2))
```



```
glance(fit1)$sigma

## [1] 0.833371

glance(fit2)$sigma

## [1] 0.2190113

AIC(fit1, fit2)

##      df       AIC
## fit1  4 56.702648
## fit2  5  1.375385
```

6. Perform an F-test on the two models to test whether the added complexity of the bi-exponential model is supported by the data. *Hint*: use `anova`.

```
anova(fit1, fit2)

## Analysis of Variance Table
```

```
##
## Model 1: Sulfate ~ A0 + A * exp(-k * Hours)
## Model 2: Sulfate ~ SSbiexp(Hours, A1, lrc1, A2, lrc2)
##   Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
## 1     18    12.5011
## 2     17     0.8154  1 11.686  243.63 1.645e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.6.4 Weight loss

Consider a dataset of sequential measurements of a person's weight while on a diet (the 'weightloss' dataset).

Read the dataset ('weightloss.csv'), call it `weightdat`, and convert the 'Date' variable to the `Date` class, like this:

```
weightdat$Date <- as.Date(weightdat$Date, format="%d/%m/%y")
```

In order to use the `loess` function, you must next convert `Date` to a numeric variable. Run the following code to convert it to days since the start of measurement:

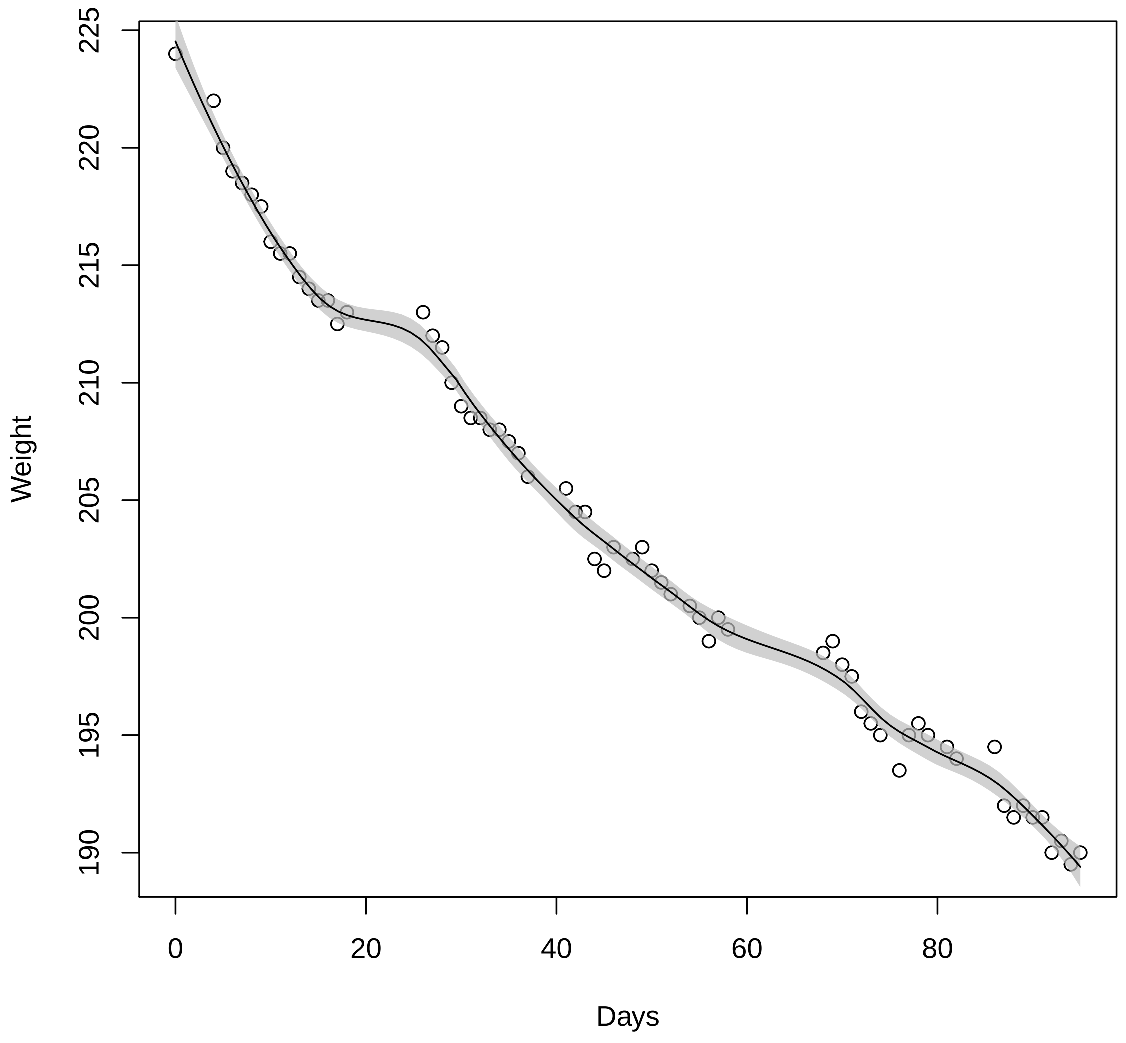
```
weightdat$Days <- with(weightdat, as.numeric(Date - min(Date)))
```

1. ■ Fit a loess regression model of Weight against Days. Try various values of `span`, and decide on a value that fits the data well without overfitting.

```
weightdat <- read.csv("weightloss.csv")
weightdat$Date <- as.Date(weightdat$Date, format="%d/%m/%y")
weightdat$Days <- with(weightdat, as.numeric(Date - min(Date)))

l <- loess(Weight ~ Days, data=weightdat, span=0.3)
```

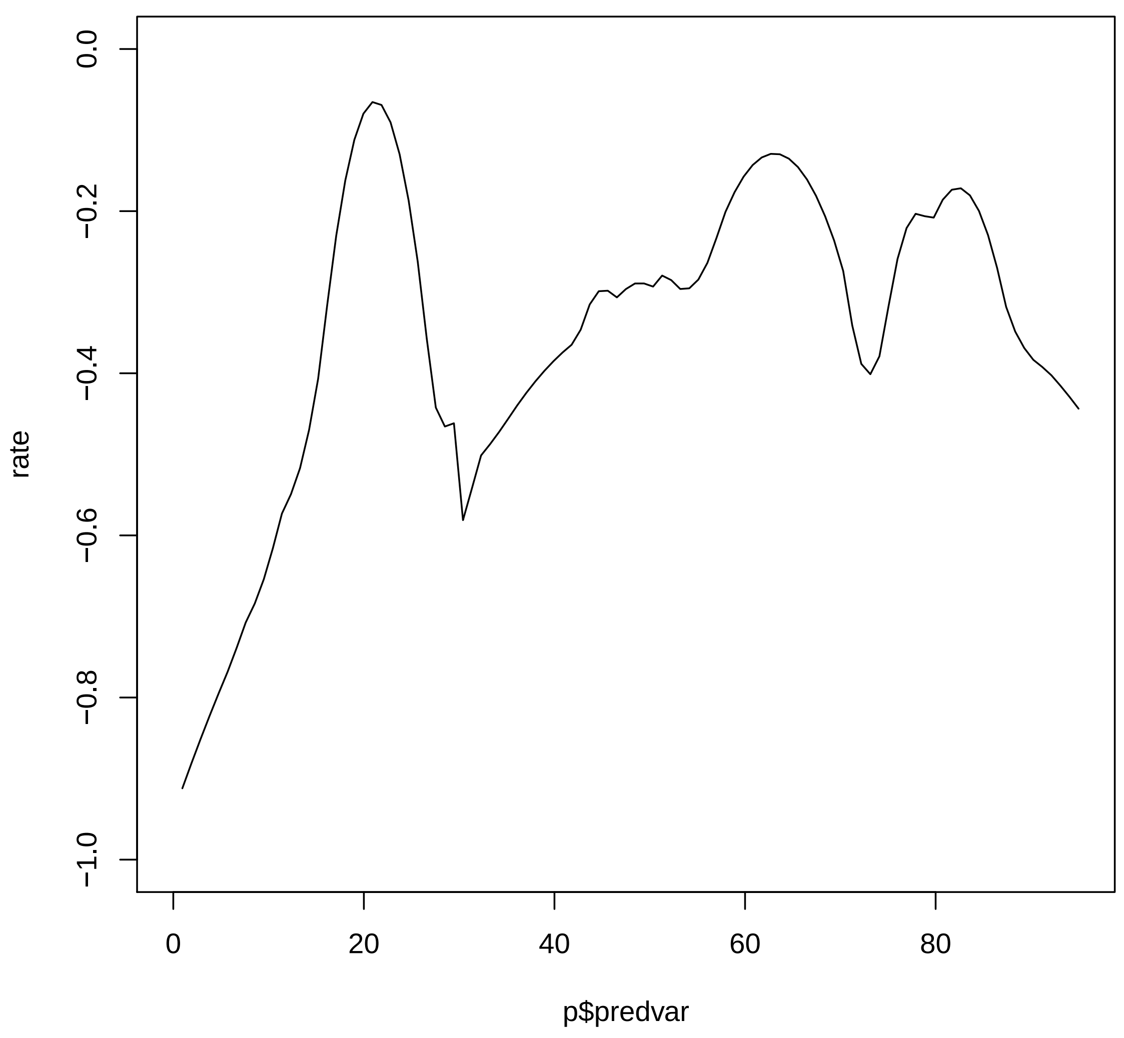
2. ♦ Plot the fitted model, and assign it to an object (see the bottom of Section **??** for details). This object contains the fitted values; inspect the object returned by `plot_loess`.

```
p <- plot_loess(l)
```

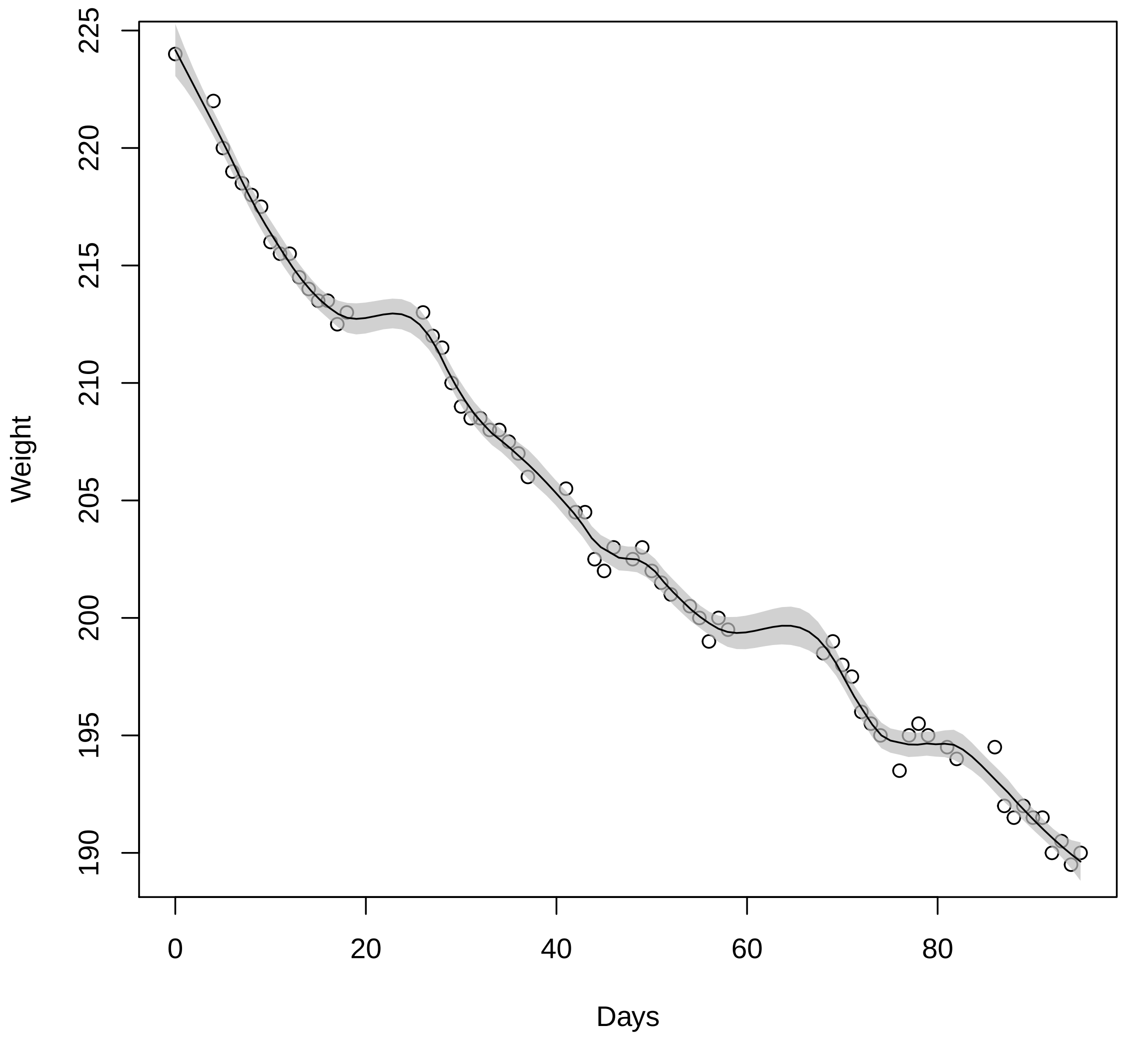


3. ▲ With the object from above, calculate the weight loss rate in pounds per day, using `diff`. Note that `diff` returns a vector that is one element shorter than the original vector (because it calculates successive differences). Make a plot of weight loss rate vs. Days. How sensitive is the calculated weight loss rate to the value of `span` chosen in the above?

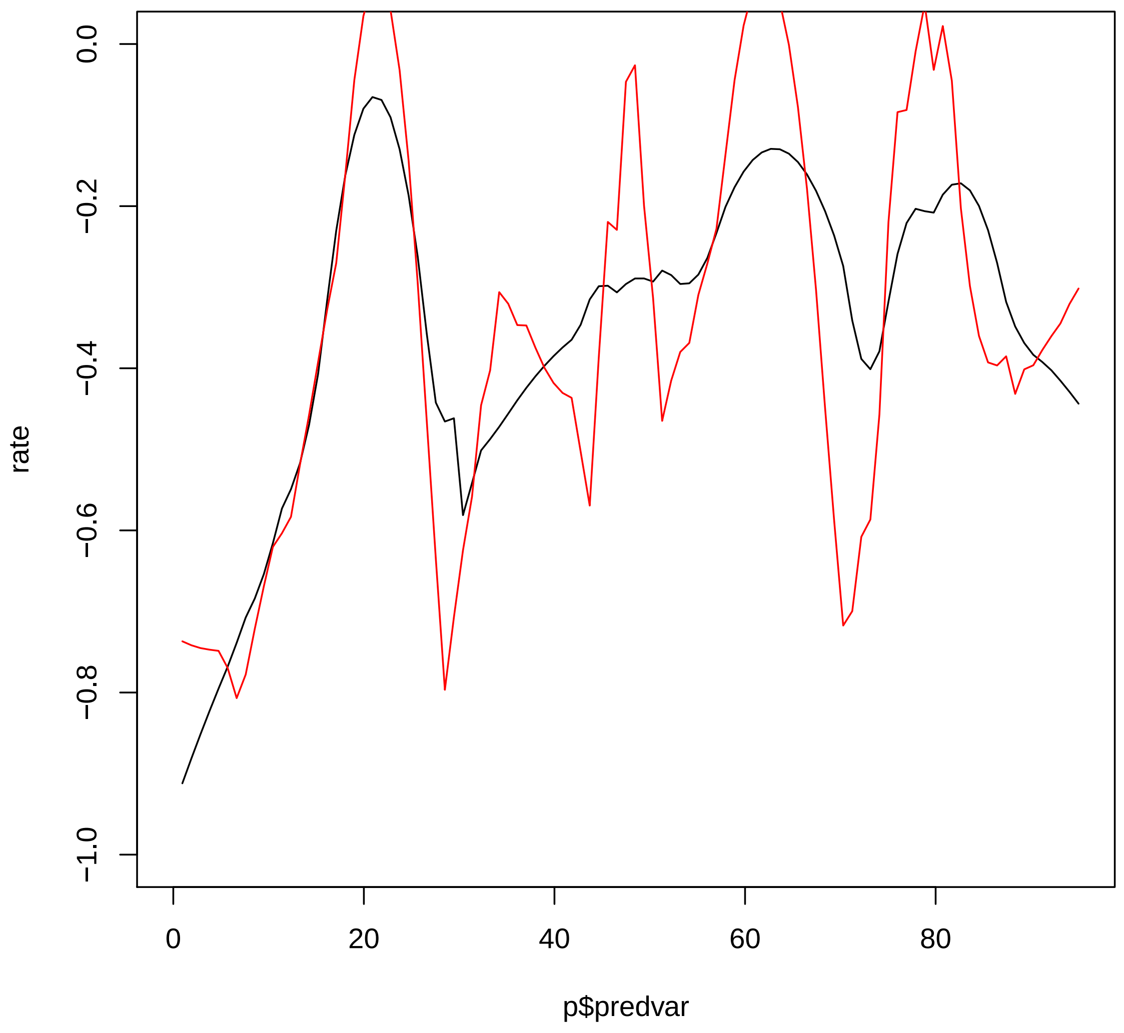
```
rate <- c(NA, diff(p$fit))
plot(p$predvar, rate, type='l', ylim=c(-1,0))
```



```
# refit with lower span
l2 <- loess(Weight ~ Days, data=weightdat, span=0.2)
p2 <- plot_loess(l2)
```



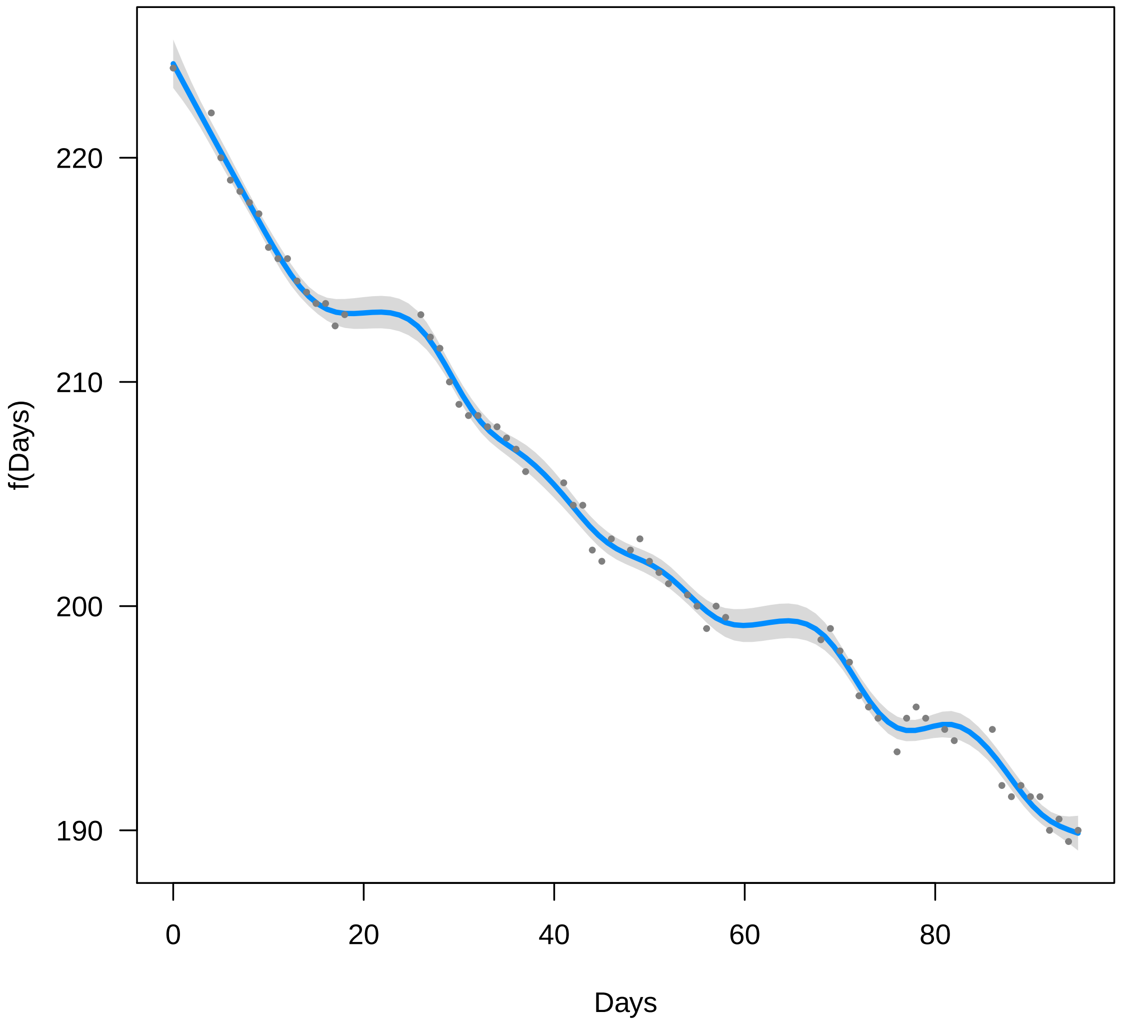
```
# Plot both in one graph
plot(p$predvar, rate, type='l', ylim=c(-1,0))
lines(p2$predvar, c(NA, diff(p2$fit)), col="red")
```



```
# Clearly our method of calculating weight loss rate is quite
# sensitive to the choice of 'span'.
```

4. 🟩 Fit a generalized additive model to the same dataset, plot the resulting object with visreg. Is the fit similar to the loess fit? What happens when you change the `k` parameter?

```
library(mgcv)
g <- gam(Weight ~ s(Days, k=15), data=weightdat)

library(visreg)
visreg(g, "Days")
```



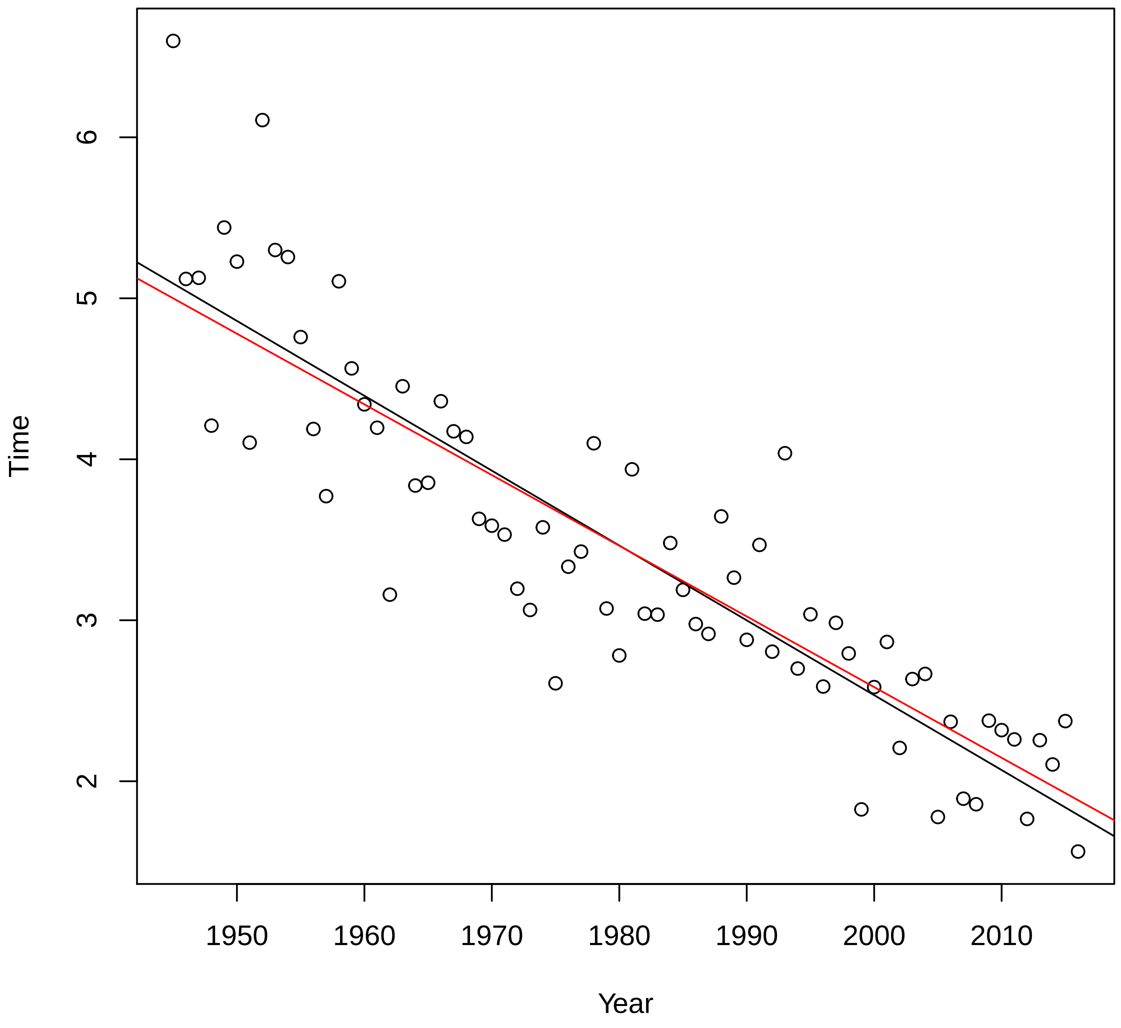
1.6.5 Race quantiles

For this exercise you will use the Sydney to Hobart race finish times, see Section **??**. There is some indication from the data that the spread of finish times has decreased over time. We will test this idea using quantile regression, ignoring the observation that the finish times did not exactly linearly decrease over time

1. ■ Read the data, fit a linear regression model and a median regression model. The latter is achieved via quantile regression with `tau = 0.5`. Compare the fits by plotting the data and adding both fitted models with `abline`.

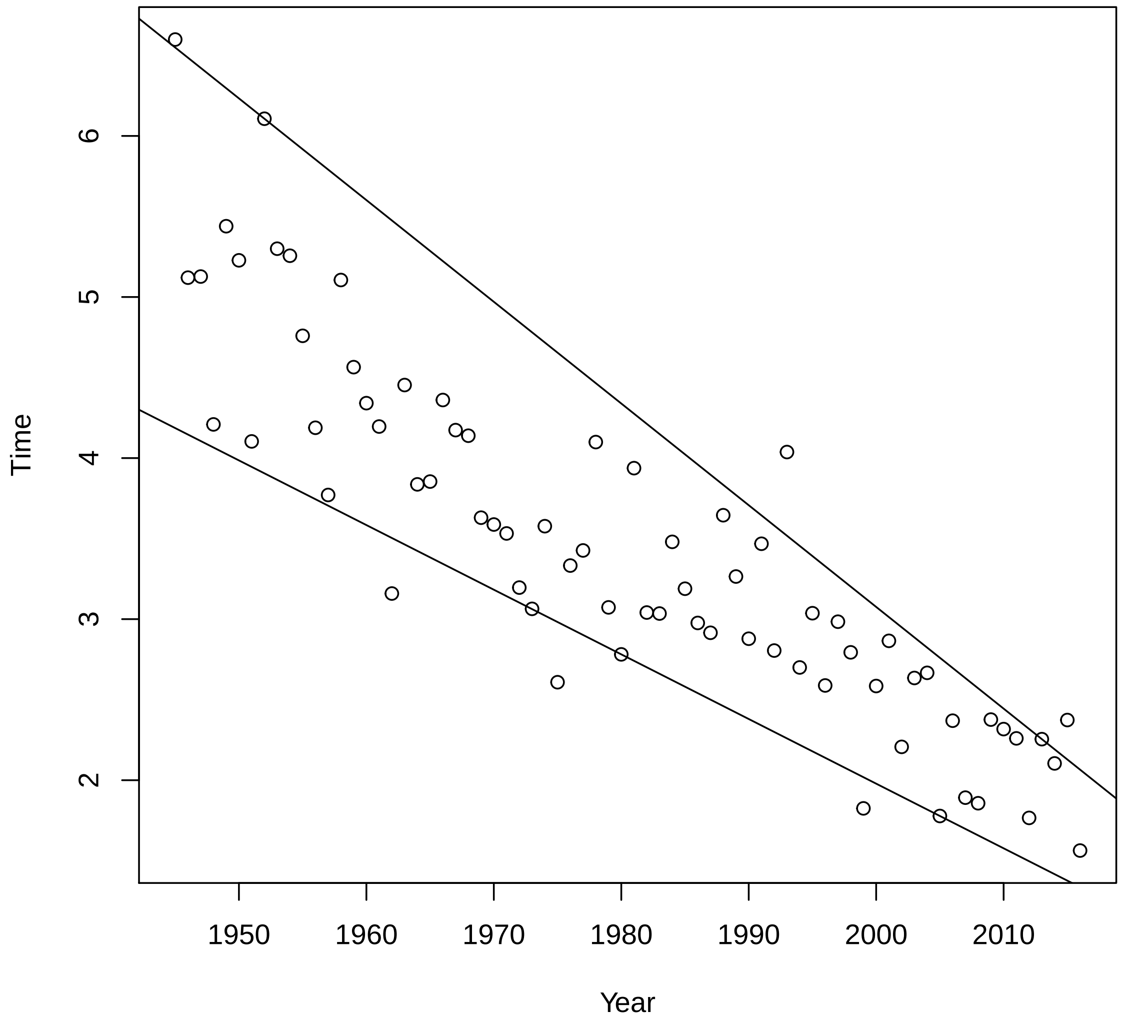
```
library(quantreg)

hobart  <- read.csv("sydney_hobart_times.csv")
```

```r
fit_lm <- lm(Time ~ Year, data=hobart)
fit_rq <- rq(Time ~ Year, tau=0.5, data=hobart)

with(hobart, plot(Year, Time))
abline(fit_lm)
abline(fit_rq, col="red")
```



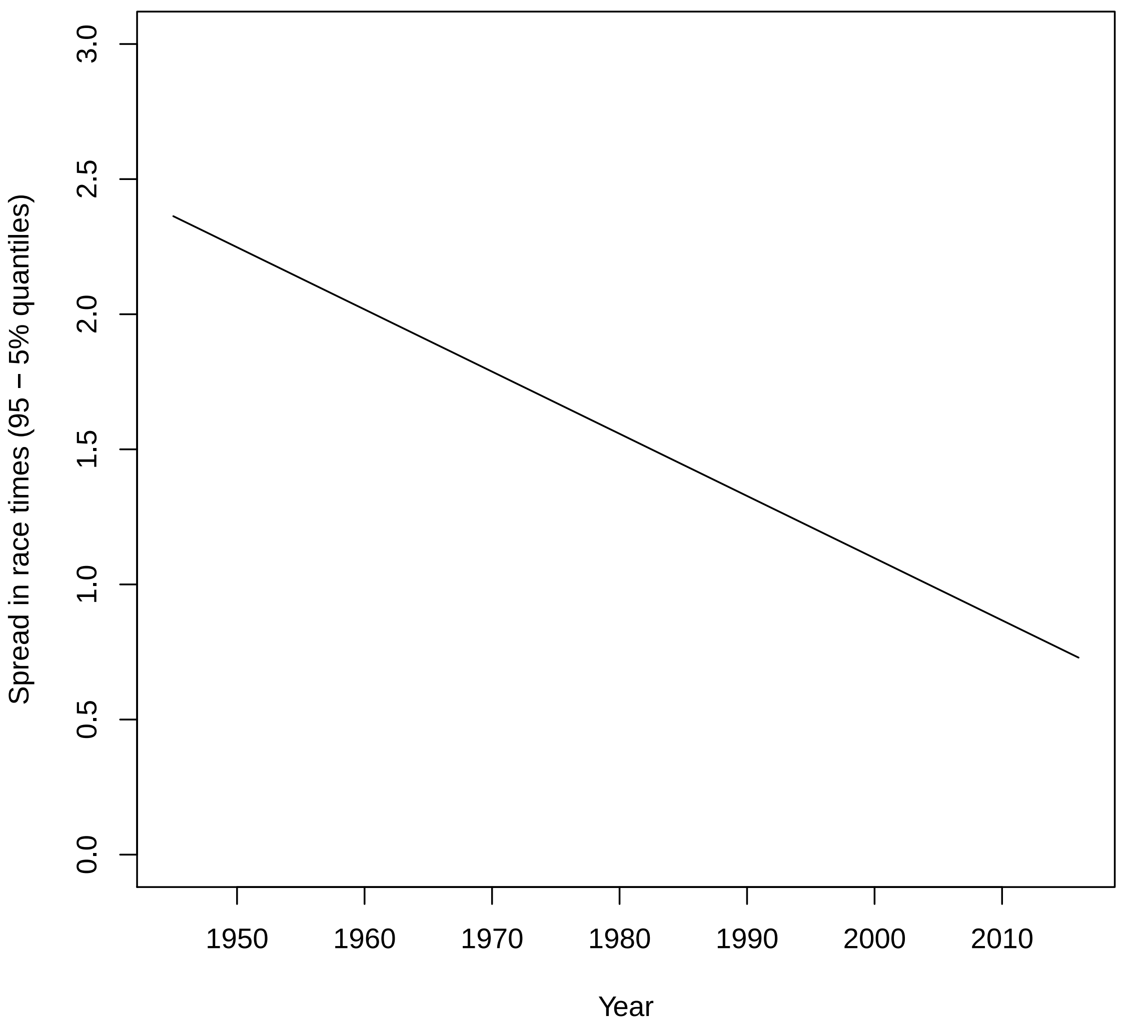
2. 🟩 Now fit two quantile regression models, at 5% and 95%. Plot the data and add both lines.

```r
fit_rq_05 <- rq(Time ~ Year, tau=0.05, data=hobart)
fit_rq_95 <- rq(Time ~ Year, tau=0.95, data=hobart)

with(hobart, plot(Year, Time))
abline(fit_rq_05)
abline(fit_rq_95)
```



3. ▲ Now calculate the 90% range of the data by calculating the difference between the 95% and 5% quantiles for each `Year`. To do this, setup a dataframe with the years for which you want to make the calculations (a sequence from min to max Year in the dataset), and calculate the predictions from both models from the previous exercise. Then plot the difference vs. Year. Inspect Section **??** for using `predict` with fitted quantile regression models (but now you can set `se=FALSE`).

```
preddf <- data.frame(Year=min(hobart$Year):max(hobart$Year))

p05 <- predict(fit_rq_05, preddf)
p95 <- predict(fit_rq_95, preddf)

plot(preddf$Year, p95 - p05, type='l',
     ylim=c(0,3),
     xlab="Year",
     ylab="Spread in race times (95 - 5% quantiles)")
```


Spread in race times (95 – 5% quantiles)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1950
1960
1970
1980
1990
2000
2010
Year