Homework 5

Ginevra Carbone

DAAG Chapter 6

Exercise 6

The following investigates the consequences of not using a logarithmic transformation for the `nihills` *data analysis. The second differs from the first in having a* `dist × climb` *interaction term, additional to linear terms in* `dist` *and* `climb`*.*

(a) Fit the two models:

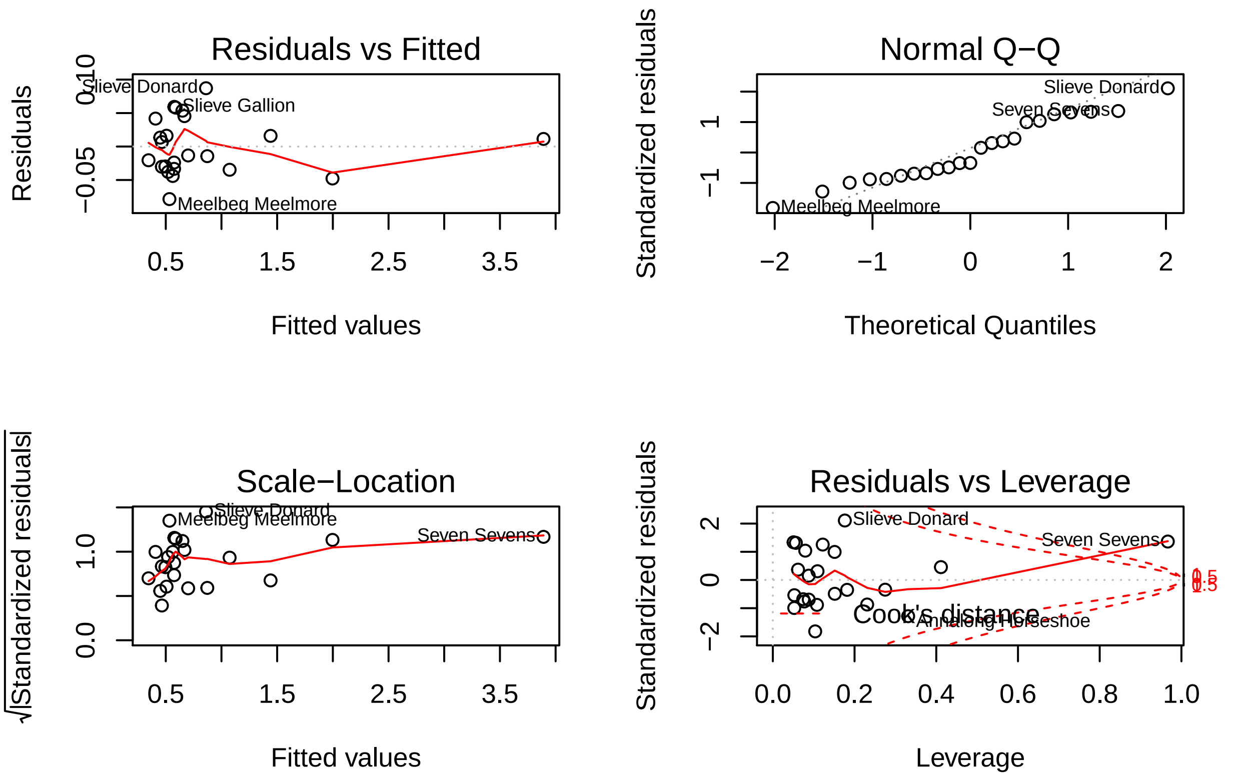
```
nihills.lm <- lm(time ~ dist+climb, data=nihills)
nihills2.lm <- lm(time ~ dist+climb+dist:climb, data=nihills)

# anova only works on nested models
anova(nihills.lm, nihills2.lm)
```

```
## Analysis of Variance Table
##
## Model 1: time ~ dist + climb
## Model 2: time ~ dist + climb + dist:climb
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1     20 0.189361
## 2     19 0.039361  1      0.15 72.406 6.623e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(b) Using the F-test result, make a tentative choice of model, and proceed to examine diagnostic plots. Are there any problematic observations? What happens if these points are removed? Refit both of the above models, and check the diagnostics again.

The small p-value suggests that model 2 is better than model 1 (we reject the hypothesis of the coefficient for `dist x climb` being null).

```
# diagnostic plot for model 2
par(mfrow=c(2,2))
plot(nihills2.lm)
```



```
par(mfrow=c(1,1))
```

Slieve Donard and Meelbeg Meelmore have high residuals, while Seven Sevens is an evident outlier, since it has a very high Cook distance. Let's check the contribution of these observations to the model.

```
# removing Slieve Donard
nihillsSD <- nihills %>%
  subset(rownames(nihills) != "Slieve Donard")
nihills2.lmSD <- lm(time ~ dist+climb+dist:climb, data=nihillsSD)

summary(nihills2.lm)
```

```
##
## Call:
## lm(formula = time ~ dist + climb + dist:climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07854 -0.03182 -0.01334  0.02894  0.08711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.677e-02  3.744e-02   1.249   0.2267
## dist        6.962e-02  7.427e-03   9.374 1.48e-08 ***
## climb       9.988e-05  2.040e-05   4.896   0.0001 ***
## dist:climb  9.964e-06  1.171e-06   8.509 6.62e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04552 on 19 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9964
## F-statistic:  2058 on 3 and 19 DF,  p-value: < 2.2e-16
```

```
summary(nihills2.lmSD)
```

```
##
## Call:
## lm(formula = time ~ dist + climb + dist:climb, data = nihillsSD)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06809 -0.03076 -0.00475  0.02014  0.06529
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.887e-02  3.405e-02   1.729 0.101003
## dist        7.072e-02  6.695e-03  10.563 3.82e-09 ***
## climb       8.262e-05  1.976e-05   4.180 0.000562 ***
## dist:climb  1.070e-05  1.099e-06   9.738 1.34e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04093 on 18 degrees of freedom
## Multiple R-squared:  0.9976, Adjusted R-squared:  0.9973
## F-statistic:  2545 on 3 and 18 DF,  p-value: < 2.2e-16
```

```
# removing Meelbeg Meelmore
nihillsMM <- nihills %>%
  subset(rownames(nihills) != "Meelbeg Meelmore")
nihills2.lmMM <- lm(time ~ dist+climb+dist:climb, data=nihillsMM)

summary(nihills2.lm)
```

```
##
## Call:
## lm(formula = time ~ dist + climb + dist:climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07854 -0.03182 -0.01334  0.02894  0.08711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.677e-02  3.744e-02   1.249   0.2267
## dist        6.962e-02  7.427e-03   9.374 1.48e-08 ***
## climb       9.988e-05  2.040e-05   4.896   0.0001 ***
## dist:climb  9.964e-06  1.171e-06   8.509 6.62e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04552 on 19 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9964
## F-statistic:  2058 on 3 and 19 DF,  p-value: < 2.2e-16
```

```r
summary(nihills2.lmMM)
```

```
##
## Call:
## lm(formula = time ~ dist + climb + dist:climb, data = nihillsMM)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.049299 -0.033248 -0.003817  0.029186  0.078450
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.596e-02  3.525e-02   1.587     0.13
## dist        6.669e-02  7.092e-03   9.403 2.28e-08 ***
## climb       1.056e-04  1.926e-05   5.482 3.31e-05 ***
## dist:climb  9.936e-06  1.093e-06   9.092 3.78e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04248 on 18 degrees of freedom
## Multiple R-squared:  0.9974, Adjusted R-squared:  0.997
## F-statistic:  2336 on 3 and 18 DF,  p-value: < 2.2e-16
```

The residual standard error is sligthly smaller in both cases, but not enough to justify removing them.

```r
# removing Seven Sevens
nihillsSS <- nihills %>%
  subset(rownames(nihills) != "Seven Sevens")

# fitting the second model again
nihills2.lmSS <- lm(time ~ dist+climb+dist:climb, data=nihillsSS)

# comparing models
summary(nihills2.lm)
```

```
##
## Call:
## lm(formula = time ~ dist + climb + dist:climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07854 -0.03182 -0.01334  0.02894  0.08711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.677e-02  3.744e-02   1.249   0.2267
## dist        6.962e-02  7.427e-03   9.374 1.48e-08 ***
## climb       9.988e-05  2.040e-05   4.896   0.0001 ***
## dist:climb  9.964e-06  1.171e-06   8.509 6.62e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04552 on 19 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9964
## F-statistic:  2058 on 3 and 19 DF,  p-value: < 2.2e-16
```

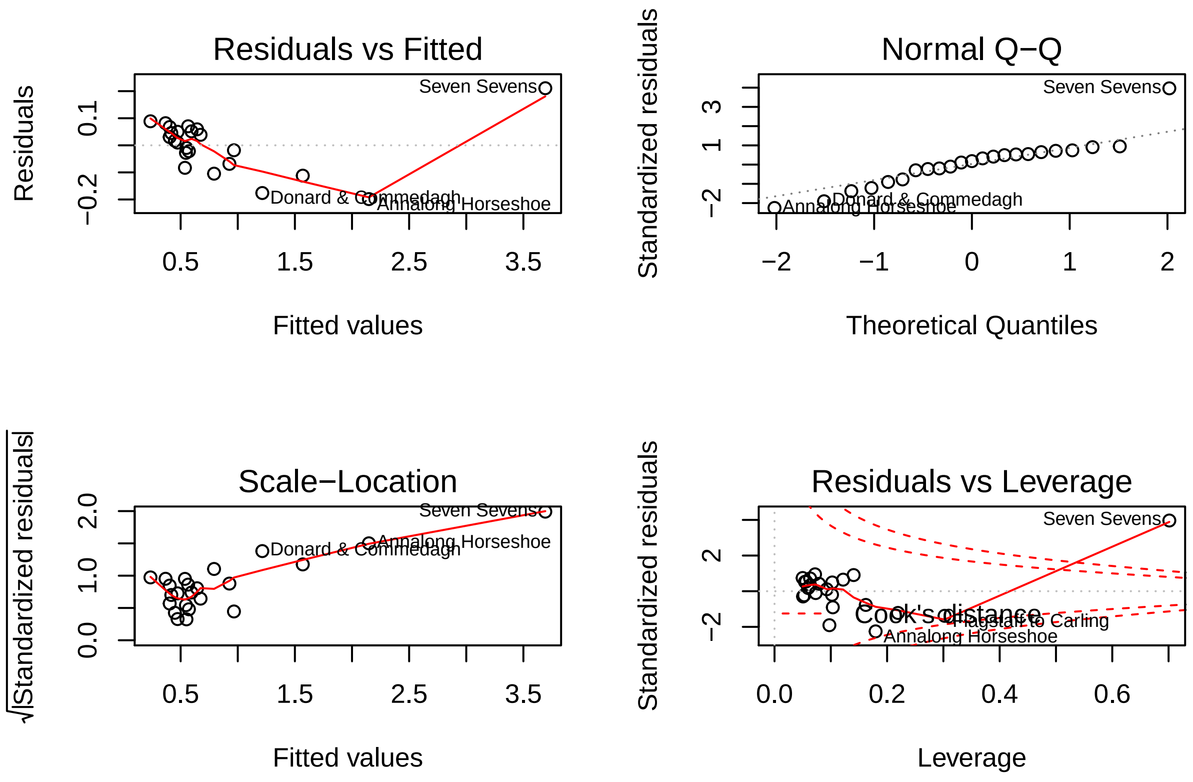
```r
summary(nihills2.lmSS)
```

```
##
## Call:
## lm(formula = time ~ dist + climb + dist:climb, data = nihillsSS)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07999 -0.02947 -0.00124  0.03308  0.07340
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.818e-02  5.911e-02  -0.308 0.761942
## dist         8.126e-02  1.104e-02   7.359 7.89e-07 ***
## climb        1.299e-04  2.929e-05   4.435 0.000319 ***
## dist:climb   5.459e-06  3.420e-06   1.596 0.127898
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04442 on 18 degrees of freedom
## Multiple R-squared:  0.9882, Adjusted R-squared:  0.9862
## F-statistic: 500.4 on 3 and 18 DF,  p-value: < 2.2e-16
```

Removing `Seven Sevens` almost makes no difference in the redisual standard error.

The term `dist x climb` is no longer significant. Let's go back to the first model and analyze its diagnostic plot.

```r
# first model diagnostic plots
par(mfrow=c(2,2))
plot(nihills.lm)
```



```r
par(mfrow=c(1,1))
```

`Seven Sevens` is still a problematic outlier. It also has a high residual, together with `Annalong Horseshoe`.

```r
# remove Annalong Horseshoe
nihillsAH <- nihills %>%
  subset(rownames(nihills) != "Annalong Horseshoe")

nihills.lmAH <- lm(time ~ dist+climb, data=nihillsAH)

# comparing models
summary(nihills.lm)
```

```
##
## Call:
## lm(formula = time ~ dist + climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19857 -0.04824  0.01701  0.05539  0.21083
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.286e-01  4.025e-02  -5.679 1.47e-05 ***
## dist         1.008e-01  1.382e-02   7.293 4.72e-07 ***
## climb        2.298e-04  2.893e-05   7.941 1.31e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0973 on 20 degrees of freedom
## Multiple R-squared:  0.9852, Adjusted R-squared:  0.9838
## F-statistic: 667.6 on 2 and 20 DF,  p-value: < 2.2e-16
```

```r
summary(nihills.lmAH)
```

```
##
## Call:
## lm(formula = time ~ dist + climb, data = nihillsAH)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19623 -0.03187  0.02349  0.05217  0.12856
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.448e-01  3.624e-02  -6.756 1.87e-06 ***
## dist         1.033e-01  1.229e-02   8.408 7.94e-08 ***
## climb        2.355e-04  2.574e-05   9.149 2.16e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08624 on 19 degrees of freedom
## Multiple R-squared:  0.9878, Adjusted R-squared:  0.9865
## F-statistic: 766.1 on 2 and 19 DF,  p-value: < 2.2e-16
```

The residual standard error is almost the same.

```r
nihills.lmSS <- lm(time ~ dist+climb, data=nihillsSS)
summary(nihills.lm)
```

```
##
## Call:
## lm(formula = time ~ dist + climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19857 -0.04824  0.01701  0.05539  0.21083
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.286e-01  4.025e-02  -5.679 1.47e-05 ***
## dist         1.008e-01  1.382e-02   7.293 4.72e-07 ***
## climb        2.298e-04  2.893e-05   7.941 1.31e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0973 on 20 degrees of freedom
## Multiple R-squared:  0.9852, Adjusted R-squared:  0.9838
## F-statistic: 667.6 on 2 and 20 DF,  p-value: < 2.2e-16
```

```r
summary(nihills.lmSS)
```

```
##
## Call:
## lm(formula = time ~ dist + climb, data = nihillsSS)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.08184 -0.03567  0.01118  0.03540  0.06018
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.049e-01  2.417e-02  -4.342 0.000351 ***
## dist         9.570e-02  6.586e-03  14.530 9.62e-12 ***
## climb        1.701e-04  1.548e-05  10.992 1.12e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04619 on 19 degrees of freedom
## Multiple R-squared:  0.9865, Adjusted R-squared:  0.9851
## F-statistic: 692.9 on 2 and 19 DF,  p-value: < 2.2e-16
```

In this case removing `Seven Sevens` significantly improves the residual standard error (from 0.0973 to 0.04619), but it's still worse than model 2, as shown by the results collected in the following table:

```
table <- data.frame(AIC=c(extractAIC(nihills.lmSS)[2], extractAIC(nihills2.lmSS)[2]),
           Adj_R2=c(summary(nihills.lmSS)$adj.r.squared, summary(nihills2.lmSS)$adj.r.squared),
           RSE = c(summary(nihills.lmSS)$sigma, summary(nihills2.lmSS)$sigma),
           row.names = c("Model 1","Model 2"))

table
```

```
##                AIC    Adj_R2        RSE
## Model 1 -132.5268 0.9850520 0.04618801
## Model 2 -133.4386 0.9861776 0.04441503
```

Exercise 7

Check the variance inflation factors for `bodywt` and `lsize` for the model `brainwt ~ bodywt + lsize`, fitted to the `litters` data set. Comment.

```
vif(lm(brainwt ~ bodywt + lsize, data=litters))
```

```
## bodywt  lsize
##  11.33  11.33
```

VIF values higher than 10 show a problem of high multicollinearity between the explanatory variables `brainwt` and `bodywt`, meaning that there is a linear relashionship between them.

Exercise 8

Apply the `lm.ridge()` function to the litters data, using the generalized cross-validation (GCV) criterion to choose the tuning parameter. (GCV is an approximation to cross-validation.)

(a) In particular, estimate the coefficients of the model relating `brainwt` to `bodywt` and `lsize` and compare with the results obtained using `lm()`.

```
#select lambda in terms of GCV error
select(lm.ridge(brainwt ~ bodywt + lsize, data = litters, lambda = seq(0,0.1,0.001)))
```

```
## modified HKB estimator is 0
## modified L-W estimator is 0
## smallest value of GCV  at 0.1
```

```r
# apply ridge function with the chosen lambda
litters.ridge <- lm.ridge(brainwt ~ bodywt + lsize, data = litters, lambda = 0.1)
litters.ridge
```

```
##                 bodywt      lsize
## 0.19998145 0.02236033 0.00580276
```

```r
# compare to lm coefficients
litters.lm <- summary(lm(brainwt ~ bodywt + lsize, data = litters))$coefficients[,1]
litters.lm
```

```
## (Intercept)      bodywt       lsize
## 0.178246962 0.024306344 0.006690331
```

`bodywt` and `lsize` are penalized in favour of the intercept coefficient.

(b) Using both ridge and ordinary regression, estimate the mean brain weight when litter size is 10 and body weight is 7. Use the bootstrap, with case-resampling, to compute approximate 95% percentile confidence intervals using each method. Compare with the interval obtained using ***`predict.lm()`****.*

```r
paste("ridge estimate: ",
      as.vector(coef(litters.ridge))%*%c(1,7,10))
```

```
## [1] "ridge estimate:  0.414531372401127"
```

```r
paste("lm estimate: ",
      as.vector(litters.lm)%*%c(1,7,10))
```

```
## [1] "lm estimate:  0.415294682299934"
```

Let's start from ordinary regression. These are the bootstrap based confidence intervals obtained using basic method and percentile method.

```r
# linear model
my_lm <- function(data, ind){
  # sample selection
  d <- data[ind,]
  # model fit
  litters.lm <- summary(lm(brainwt ~ bodywt + lsize,
                           data = d))$coefficients[,1]
  # coefficients
  coef <- as.vector(litters.lm)
  # prediction on given values
  return(coef%*%c(1,7,10))
}

litters.boot <- boot(data=litters, statistic=my_lm, R=10^4)
boot.ci(litters.boot, conf=0.95, type=c("basic","perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = litters.boot, conf = 0.95, type = c("basic",
##     "perc"))
##
## Intervals :
## Level      Basic              Percentile
## 95%   ( 0.4077,  0.4254 )   ( 0.4052,  0.4229 )
## Calculations and Intervals on Original Scale
```

Using Ridge regression:

```r
# ridge regression
my_ridge <- function(data, ind){
  # sample selection
  d <- data[ind,]
  # model fit
  litters.ridge <- lm.ridge(brainwt ~ bodywt + lsize,
                            data = d, lambda = 0.1)
  # coefficients
  coef <- as.vector(coef(litters.ridge))
  # prediction on given values
  return(coef%*%c(1,7,10))
}

litters.boot <- boot(data=litters, statistic=my_ridge, R=10^4)
boot.ci(litters.boot, conf=0.95, type=c("basic","perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = litters.boot, conf = 0.95, type = c("basic",
##     "perc"))
##
## Intervals :
## Level      Basic              Percentile
## 95%   ( 0.4070,  0.4243 )   ( 0.4048,  0.4221 )
## Calculations and Intervals on Original Scale
```

Let’s now compute the interval with `predict.lm()` (we cannot use it on a `ridgelm` object.)

```r
# variables for prediction
new <- data.frame(bodywt = 7, lsize = 10)

# estimated value and CI
predict.lm(lm(brainwt ~ bodywt + lsize,
              data = litters),
           newdata = new,
           interval = "confidence")
```

```
##         fit       lwr       upr
## 1 0.4152947 0.4062582 0.4243312
```

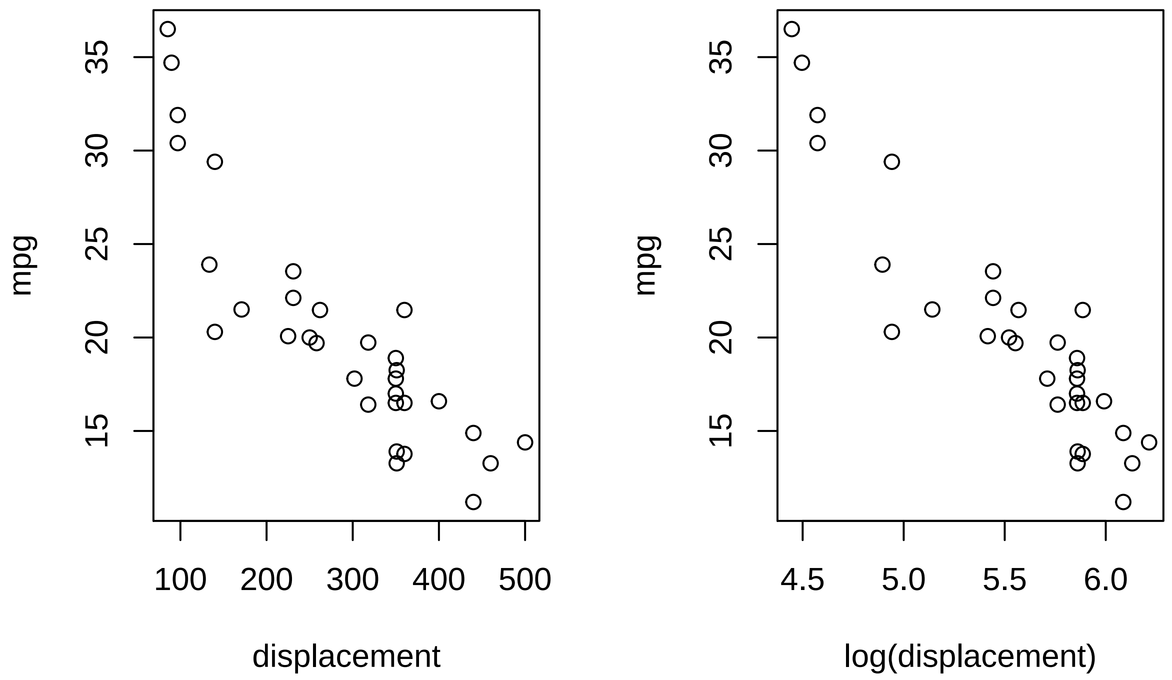
Exercise 10

The data frame `table.b3` in the MPV package contains data on gas mileage and 11 other variables for a sample of 32 automobiles.

(a) Construct a scatterplot of `y` (mpg) versus `x1` (displacement). Is the relationship between these variables non-linear?

```r
data <- table.b3
par(mfrow=c(1,2))
```

```r
plot(y ~ x1, ylab="mpg", xlab = "displacement", data = data)
plot(y ~ log(x1), ylab="mpg", xlab = "log(displacement)", data = data)
```

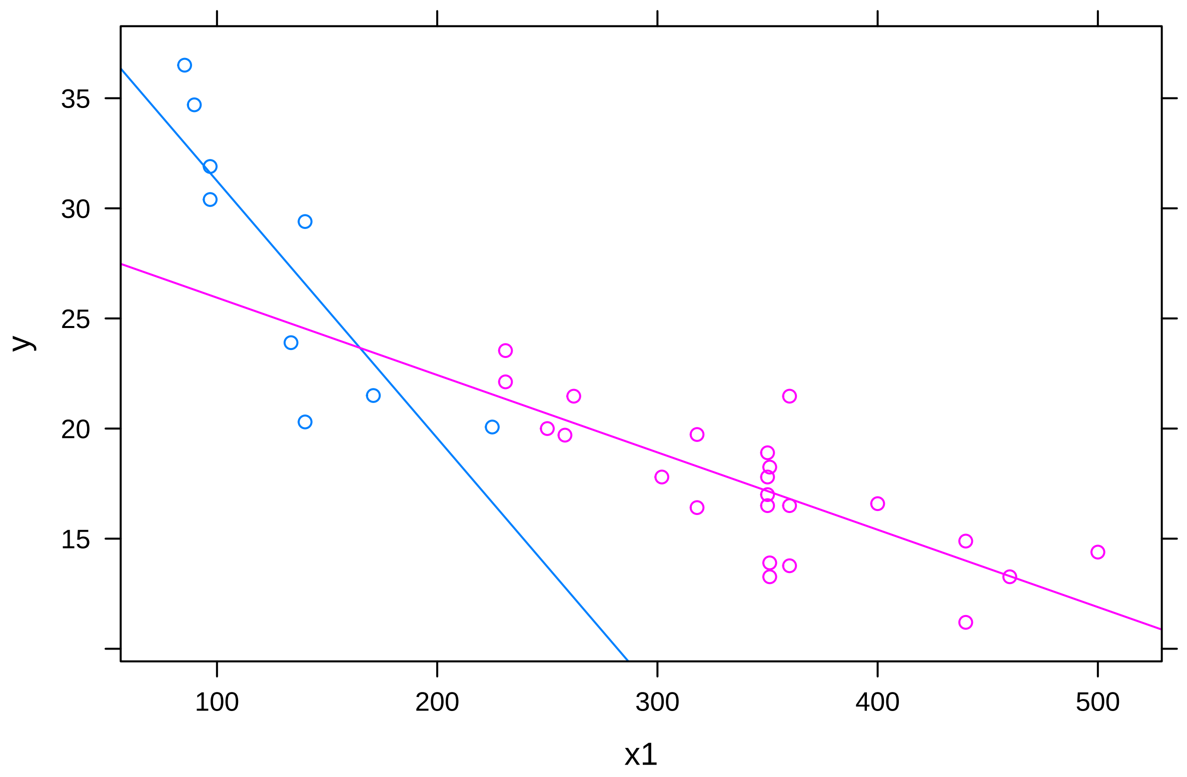


```r
par(mfrow=c(1,1))
```

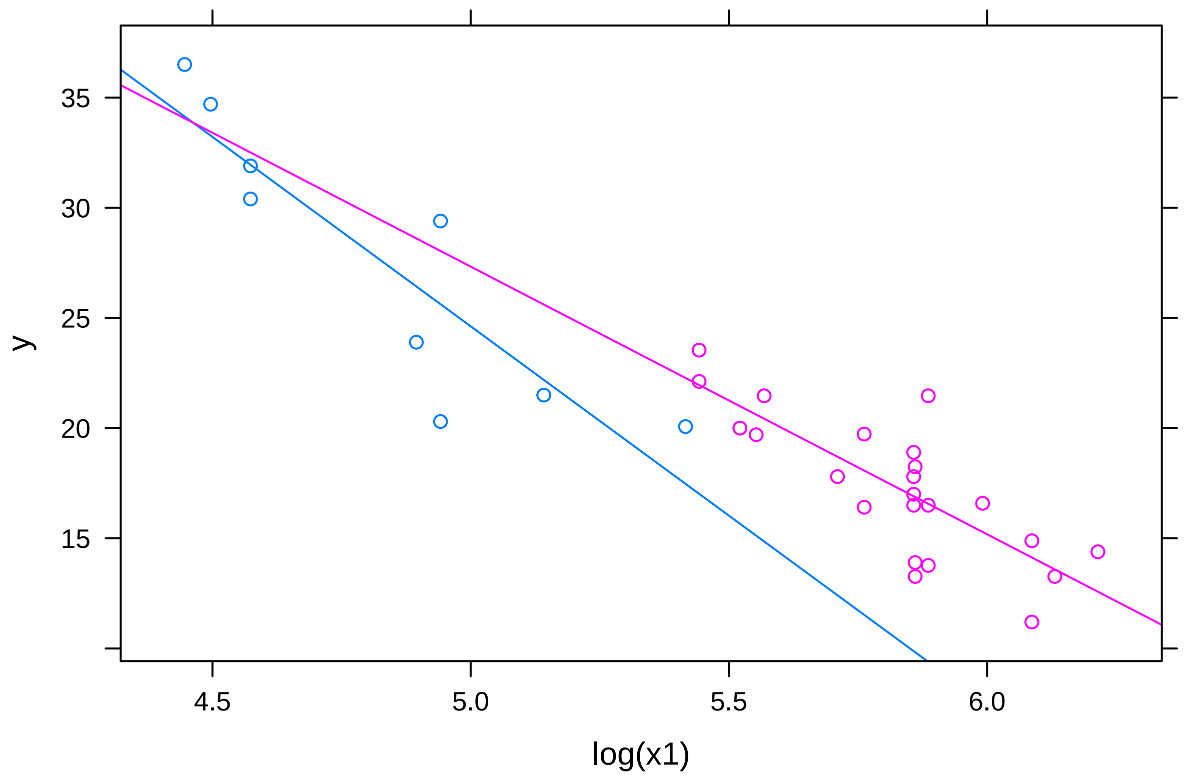
The scatterplot shows a negative non-linear relationship between mpg and displacement. By applying a log transformation to displacement, the relationship almost looks like a negative linear one.

(b) Use the ***xyplot()*** *function, and* ***x11*** *(type of transmission) as a group variable. Is a linear model reasonable for these data?*

```r
par(mfrow=c(1,2))
xyplot( y ~ x1,
        data = data,
        group = x11,
        type = c("p","r"))
```



```
xyplot( y ~ log(x1),
        data = data,
        group = x11,
        type = c("p","r"))
```



```r
par(mfrow=c(1,1))
```

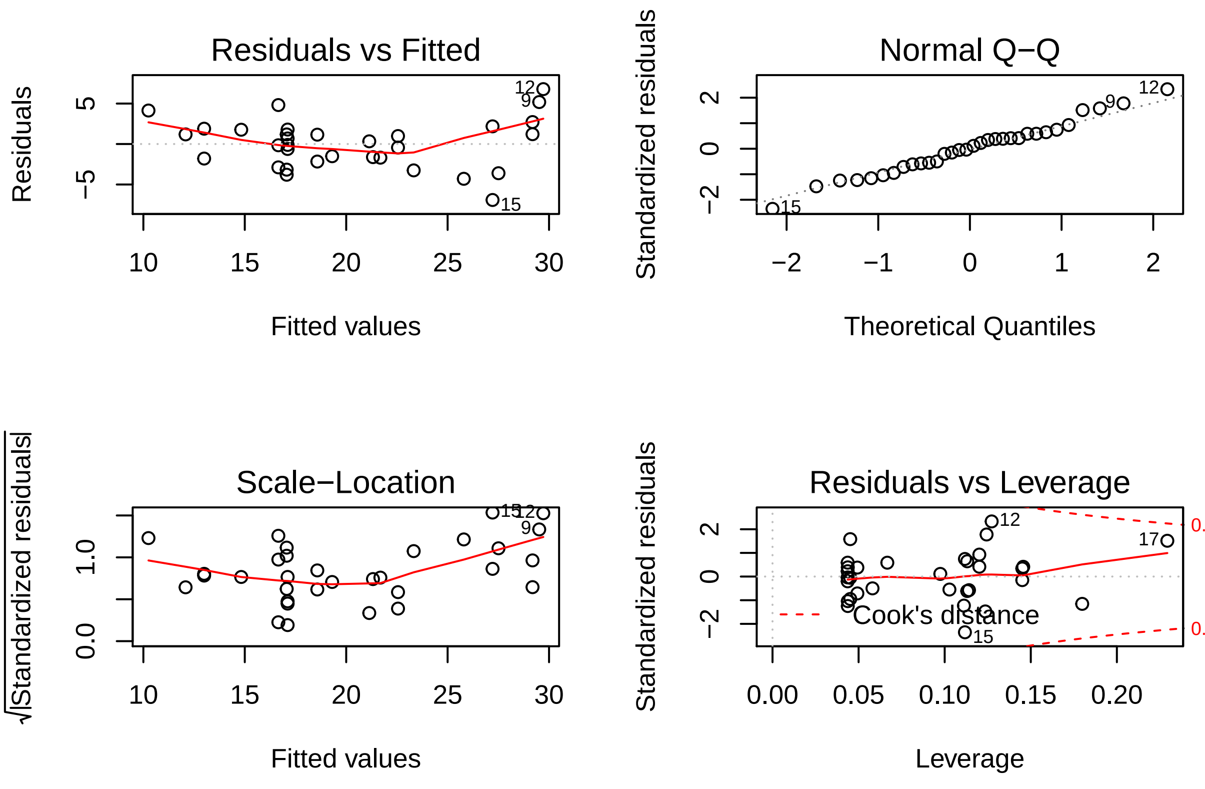
If we consider the grouping of data, a linear model doesn't seem reasonable because we aim at achieving an homogeneity of variance between different groups. The situation assumption becomes reasonable if we apply the log transform first.

(c) Fit the model relating ***y*** *to* ***x1*** *and* ***x11*** *which gives two lines having possibly different slopes and intercepts. Check the diagnostics. Are there any influential observations? Are there any influential outliers?*

```r
lm.fit <- lm(y ~ x1 + x11, data = data)
anova(lm.fit)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x1         1 955.72  955.72 98.5143 7.819e-11 ***
## x11        1   0.49    0.49  0.0501    0.8245
## Residuals 29 281.34    9.70
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
par(mfrow=c(2,2))
plot(lm.fit)
```



```r
par(mfrow=c(1,1))
```

Data points 12 and 15 have higher residual, but removing them from the model gives no improvement.

```r
summary(lm.fit)
```

```
##
```

```
## Call:
## lm(formula = y ~ x1 + x11, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9153 -1.8882  0.1106  1.7706  6.7829
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.618408   1.539505  21.837  < 2e-16 ***
## x1          -0.045736   0.008682  -5.268  1.2e-05 ***
## x11         -0.498689   2.228198  -0.224    0.824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.115 on 29 degrees of freedom
## Multiple R-squared:  0.7727, Adjusted R-squared:  0.757
## F-statistic: 49.28 on 2 and 29 DF,  p-value: 4.696e-10
```

```
data2 <- data %>%
  subset(rownames(data) != "12")

lm.fit2 <- lm(y ~ x1 + x11, data = data2)

summary(lm.fit2)
```

```
##
## Call:
## lm(formula = y ~ x1 + x11, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0767 -2.0734 -0.1195  1.6238  6.1613
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 32.394033   1.492289  21.708  < 2e-16 ***
## x1          -0.042981   0.008038  -5.347 1.07e-05 ***
## x11         -0.225940   2.046984  -0.110    0.913
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.857 on 28 degrees of freedom
## Multiple R-squared:  0.7629, Adjusted R-squared:  0.7459
## F-statistic: 45.04 on 2 and 28 DF,  p-value: 1.779e-09
```

```
data3 <- data %>%
  subset(rownames(data) != "15")

lm.fit3 <- lm(y ~ x1 + x11, data = data3)

summary(lm.fit3)
```

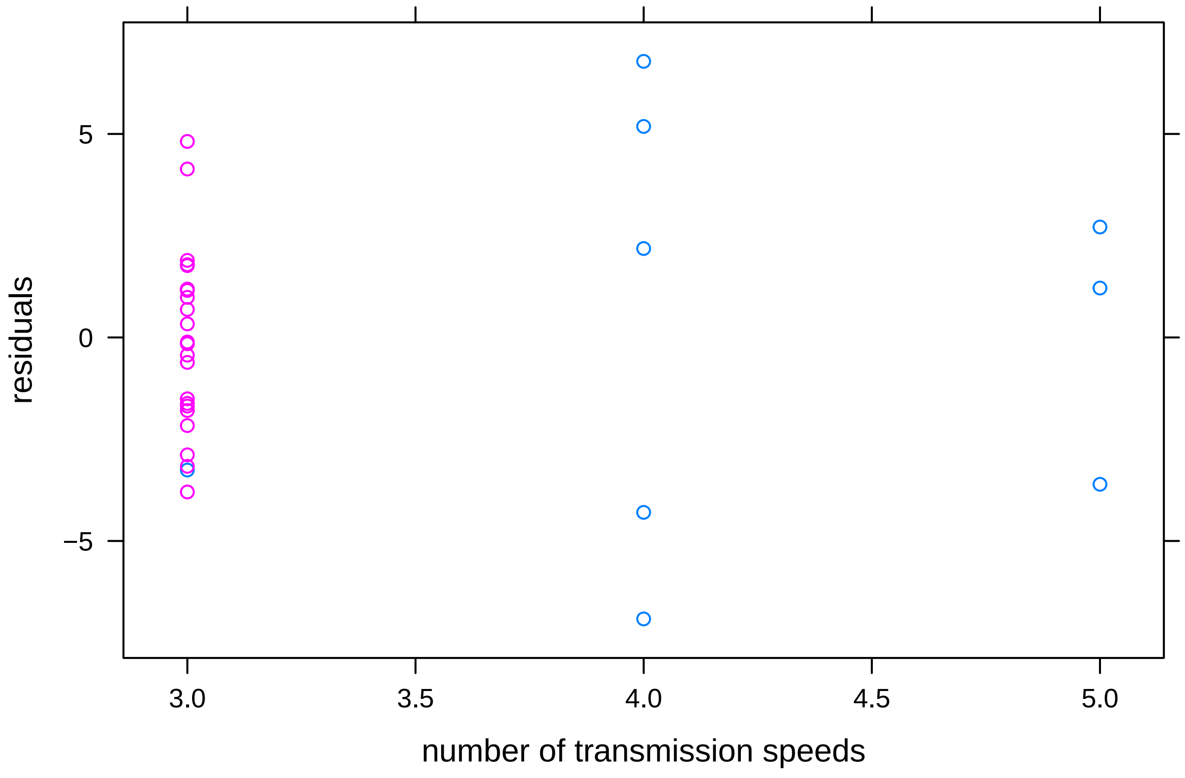
```
##
## Call:
```

```
## lm(formula = y ~ x1 + x11, data = data3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1845 -1.7405  0.3671  1.5248  5.9429
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.411640   1.442198  23.861  < 2e-16 ***
## x1          -0.045188   0.007948  -5.685 4.28e-06 ***
## x11         -1.481314   2.074577  -0.714    0.481
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.85 on 28 degrees of freedom
## Multiple R-squared:  0.8162, Adjusted R-squared:  0.803
## F-statistic: 62.16 on 2 and 28 DF,  p-value: 5.034e-11
```

Moreover, looking at Cook distance, there are no influential outliers.

*(d) Plot the residuals against the variable **x7** (number of transmission speeds), again using **x11** as a group variable. Is there anything striking about this plot?*

```r
xyplot(residuals(lm.fit) ~ x7, data = data,
       groups = x11,
       ylab = "residuals",
       xlab="number of transmission speeds")
```



This plot is distributed along vertical lines because `x7` is a categorical variable. It enhances the different variance of the residual between groups.

DAAG Chapter 8

Exercise 1

The following table shows numbers of occasions when inhibition (i.e., no flow of current across a membrane) occurred within 120 s, for different concentrations of the protein peptide-C (data are used with the permission of Claudia Haarmann, who obtained these data in the course of her PhD research). The outcome yes implies that inhibition has occurred.

```
conc <- c(0.1, 0.5, 1, 10, 20, 30, 50, 70, 80, 100, 150)
no <- c(7, 1, 10, 9, 2, 9, 13, 1, 1, 4, 3)
yes <- c(0, 0, 3, 4, 0, 6, 7, 0, 0, 1, 7)
df <- data.frame("conc"=conc, "no"=no, "yes"=yes)
df
```

```
##     conc no yes
## 1    0.1  7   0
## 2    0.5  1   0
## 3    1.0 10   3
## 4   10.0  9   4
## 5   20.0  2   0
## 6   30.0  9   6
## 7   50.0 13   7
## 8   70.0  1   0
## 9   80.0  1   0
## 10 100.0  4   1
## 11 150.0  3   7
```

Use logistic regression to model the probability of inhibition as a function of protein concentration.

```
# logistic regression fit
logit.fit1 <- glm(as.factor(no) ~ conc,
                  family=binomial(link="logit"),
                  data=df)

summary(logit.fit1)
```

```
##
## Call:
## glm(formula = as.factor(no) ~ conc, family = binomial(link = "logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6603  -0.4085   0.7692   0.7922   0.8803
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.089034   0.971120   1.121    0.262
## conc        -0.002272   0.014280  -0.159    0.874
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 12.891  on 10  degrees of freedom
## Residual deviance: 12.866  on  9  degrees of freedom
## AIC: 16.866
```

```
##
## Number of Fisher Scoring iterations: 4
```

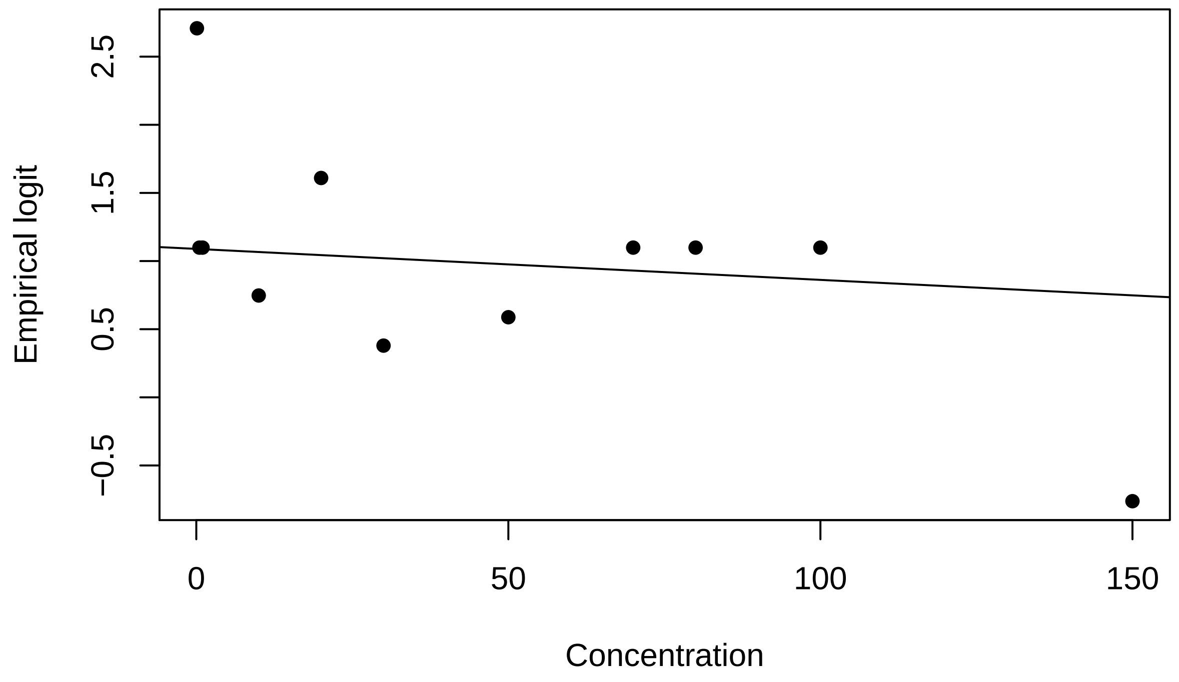
This model assumes that inhibition occurs with a probability that in logistic scale is a linear function of the concentration:

$$logit(p) = log(odds) = \beta_0 + \beta_1 conc$$

As we can see in the summary of this model, p-values are high, due to the small number of observations.

Let's also give a graphical representation of the result.

```
df$emplogit <- log((df$no+0.5)/(df$yes+0.5))

plot(emplogit ~ conc, data=df,
     xlab = "Concentration",
     ylab = "Empirical logit",
     pch = 16)

abline(logit.fit1)
```



Exercise 2

In the data set (an artificial one of 3121 patients, that is similar to a subset of the data analyzed in Stiell et al., 2001) `minor.head.injury`*, obtain a logistic regression model relating* `clinically.important.brain.injury` *to other variables. Patients whose risk is sufficiently high will be sent for CT (computed tomography). Using a risk threshold of 0.025 (2.5%), turn the result into a decision rule for use of CT.*

```
injury.fit <- glm(clinically.important.brain.injury ~ .,
                  family=binomial(link="logit"),
                  data=head.injury)

summary(injury.fit)
```

```
##
```

```
## Call:
## glm(formula = clinically.important.brain.injury ~ ., family = binomial(link = "logit"),
##     data = head.injury)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2774  -0.3511  -0.2095  -0.1489   3.0028
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -4.4972     0.1629 -27.611  < 2e-16 ***
## age.65                     1.3734     0.1827   7.518 5.56e-14 ***
## amnesia.before             0.6893     0.1725   3.996 6.45e-05 ***
## basal.skull.fracture       1.9620     0.2064   9.504  < 2e-16 ***
## GCS.decrease              -0.2688     0.3680  -0.730 0.465152
## GCS.13                     1.0613     0.2820   3.764 0.000168 ***
## GCS.15.2hours              1.9408     0.1663  11.669  < 2e-16 ***
## high.risk                  1.1115     0.1591   6.984 2.86e-12 ***
## loss.of.consciousness      0.9554     0.1959   4.877 1.08e-06 ***
## open.skull.fracture        0.6304     0.3151   2.001 0.045424 *
## vomiting                   1.2334     0.1961   6.290 3.17e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1741.6  on 3120  degrees of freedom
## Residual deviance: 1201.3  on 3110  degrees of freedom
## AIC: 1223.3
##
## Number of Fisher Scoring iterations: 6
```

```r
# divide into train and test sets
size <- nrow(head.injury)
train <- head.injury[1:size*0.6,]
test <- head.injury[(size*0.6+1):size,]

# fit the model
injury.fit <- glm(clinically.important.brain.injury ~ .,
                 family=binomial(link="logit"),
                 data=train)

probabilities <- predict(injury.fit, test)

# from model to binary classification with threshold 0.025
predictions <- ifelse(probabilities > 0.025, 1, 0)

# measuring the accuracy of predictions
misClasificError <- mean(predictions != test$clinically.important.brain.injury)
print(paste('Accuracy', 1-misClasificError))
```

```
## [1] "Accuracy 0.923076923076923"
```

```r
# table of predictions
table <- table(actual=test$clinically.important.brain.injury, predicted=predictions)
```

```
table
```

```
##       predicted
## actual    0    1
##      0 1121   20
##      1   76   31
```

Exercise 5

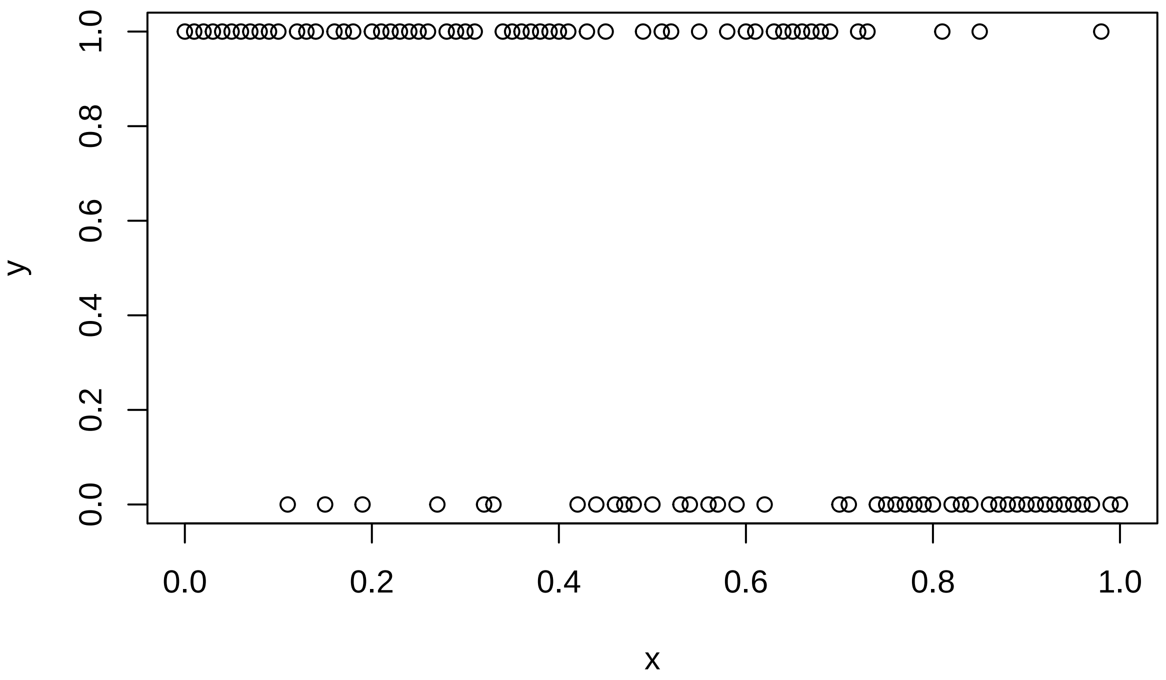
Use the function `logisticsim()` (in the DAAG package) to simulate data from a logistic regression model to study the `glm()` function. For example, you might try experiments such as the following:

(a) Simulate 100 observations from the model logit(x)=2-4x for $x = 0, 0.01, 0.02, ..., 1.0$. [This is the default setting for `logisticsim()`.]

```
x <- seq(0,1,0.01)
sim <- logisticsim(x, a = 2, b = -4)
```

(b) Plot the responses (y) against the "dose" (x). Note how the pattern of 0s and 1s changes as x increases.

```
plot(y ~ x, data = sim)
```



(c) Fit the logistic regression model to the simulated data, using the binomial family. Compare the estimated coefficients with the true coefficients. Are the estimated coefficients within about 2 standard errors of the truth?

```
model.fit <- glm(y ~ x, family=binomial(link="logit"),
                 data = sim)

summary(model.fit)$coeff
```

```
##               Estimate Std. Error  z value     Pr(>|z|)
## (Intercept)  2.672906  0.5739770  4.656818 3.211346e-06
## x           -4.614068  0.9622151 -4.795256 1.624671e-06
```

The estimated coefficients are 1.666875 and -3.382546. They are within about 2 standard errors of the truth, as we can see in the second column.

(d) Compare the estimated logit function with the true logit function. How well do you think the fitted logistic model would predict future observations? For a concrete indication of the difference, simulate a new set of 100 observations at the same x values, using a specified pseudorandom number generator seed and the true model. Then simulate some predicted observations using the estimated model and the same seed.

```r
# simulate observations from the real model
sim1 <- logisticsim(x, a = 2, b = -4, seed=342)

# simulate observations from the estimated model
sim2 <- logisticsim(x, a = 1.763179, b = -3.491443, seed=342)

# comparing predictions
mean(sim1$y==sim2$y)
```

```
## [1] 0.970297
```

The estimated model is very similar to the real one: 97% of predicted observations coincide.

Exercise 6

As in the previous exercise, the function `poissonsim()` allows for experimentation with Poisson regression. In particular, `poissonsim()` can be used to simulate Poisson responses with log-rates equal to $a + bx$, where a and b are fixed values by default.

(a) Simulate 100 Poisson responses using the model

$$log(\lambda) = 2 - 4x$$

for $x = 0, 0.01, 0.02, ..., 1.0$. Fit a Poisson regression model to these data, and compare the estimated coefficients with the true coefficients. How well does the estimated model predict future observations?

```r
# simulate poisson observations
x <- seq(0,1,0.01)
sim3 <- poissonsim(x, a = 2, b = -4, seed=123)

# fit a poisson regression model
model.fit3 <- glm(y ~ x, family=poisson, data = sim3)

# get predicted coefficients
summary(model.fit)$coeff
```

```
##              Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)  2.672906  0.5739770  4.656818 3.211346e-06
## x           -4.614068  0.9622151 -4.795256 1.624671e-06
```

```r
# simulate observations from the estimated model
sim4 <- poissonsim(x, a = 2.171313, b = -4.504560, seed=123)

# comparing predictions
mean(sim3$y==sim4$y)
```

```
## [1] 0.8316832
```

Predictions coincide 83% of the times.

(b) Simulate 100 Poisson responses using the model

$$log(\lambda) = 2 - bx$$

where b is normally distributed with mean 4 and standard deviation 5. [Use the argument `slope.sd=5` *in the* `poissonsim()` *function.] How do the results using the poisson and quasipoisson families differ?*

```
b <- rnorm(100, 4, 5)
sim5 <- poissonsim(seq(0.01,1,0.01), a = 2, b = b, slope.sd = 5)
```

```
## Warning in rpois(n, lambda = rate): NAs produced
```

```
model.fit5_pois <- glm(y ~ x, family=poisson, data = sim5)
model.fit5_quasip <- glm(y ~ x, family=quasipoisson, data = sim5)

summary(model.fit5_pois)
```

```
##
## Call:
## glm(formula = y ~ x, family = poisson, data = sim5)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -965.01  -120.70   -17.38     2.11  1334.69
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.426100   0.005753   74.06   <2e-16 ***
## x           12.752553   0.006223 2049.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 24088508  on 98  degrees of freedom
## Residual deviance: 11562515  on 97  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 11563117
##
## Number of Fisher Scoring iterations: 8
```

```
summary(model.fit5_quasip)
```

```
##
## Call:
## glm(formula = y ~ x, family = quasipoisson, data = sim5)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -965.01  -120.70   -17.38     2.11  1334.69
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4261     3.1337   0.136 0.892122
## x            12.7526     3.3896   3.762 0.000288 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for quasipoisson family taken to be 296676.8)
##
##     Null deviance: 24088508  on 98  degrees of freedom
## Residual deviance: 11562515  on 97  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: NA
##
## Number of Fisher Scoring iterations: 8
```

The two models predict the same coefficients, because the only difference between them is the introduction of the dispersion parameter (which is set to 1 in Poisson model). As a direct consequence the standard errors are scaled by the square root of this parameter, so also confidence intervals and p-values change.