Modelling with R

Exercises – Set 1

1. For the `Birthwt` data set, fit a suitable linear model and state your conclusions.

   Examine the residuals and report if there is any concern about the assumptions underlying your analysis.

   Submit your answer as a working R script, with your conclusions included as comments in the appropriate places.

   [A solution:

```
> Attach()
> m4 <- aov(wt ~ sex/poly(age, 2), Birthwt)
> m3 <- aov(wt ~ sex/age, Birthwt)
> m2 <- aov(wt ~ age + sex, Birthwt)
> m1 <- aov(wt ~ age, Birthwt)
> m0 <- aov(wt ~ 1, Birthwt)

> anova(m0, m1, m2, m3, m4)

Analysis of Variance Table

Model 1: wt ~ 1
Model 2: wt ~ age
Model 3: wt ~ age + sex
Model 4: wt ~ sex/age
Model 5: wt ~ sex/poly(age, 2)
  Res.Df     RSS Df Sum of Sq       F    Pr(>F)
1     23 1829873
2     22  816074  1   1013799 30.1504 3.249e-05
3     21  658771  1    157304  4.6782   0.04425
4     20  652425  1      6346  0.1887   0.66913
5     18  605246  2     47179  0.7015   0.50888
```

   Model `m2` appears to be the one best supported by the data. Check the terms are independently useful:

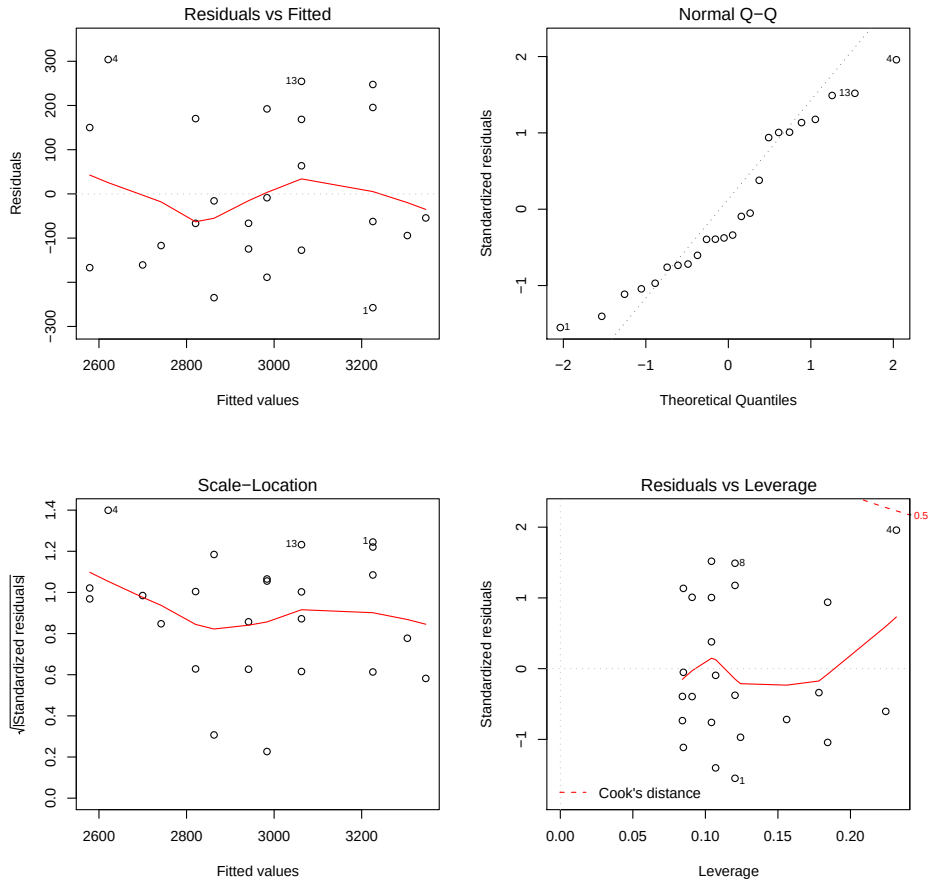
```
> dropterm(m2, test = "F")

Single term deletions

Model:
wt ~ age + sex
       Df Sum of Sq     RSS     AIC F Value     Pr(F)
<none>               658771     251
age     1   1094940 1753711     273      35 7.284e-06
sex     1    157304  816074     254       5   0.03609
```

   They appear both to be needed. Now check the diagnostics:

```
> par(mfrow = c(2, 2))
> plot(m2)
```



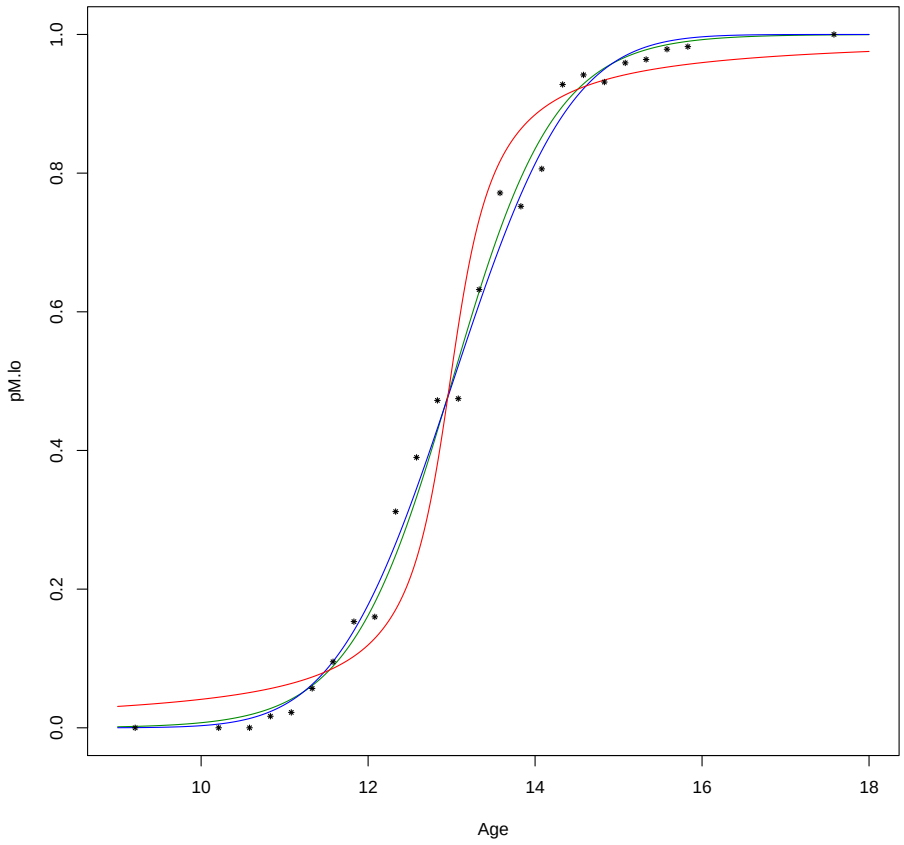
No convincing evidence of any problem. Retain model 2.]

2. With the `menarche` data, fit a binomial model (as done in lectures) but use three link functions, namely the `logistic`, `probit` and `cauchit`. Compare your predictions graphically, including the relative frequencies and fitted lines on the same diagram. Include a legend in the top left hand corner.

   [A solution:

```
> m.lo <- glm(Menarche/Total ~ Age, binomial, menarche, weights = Total)
> m.pr <- update(m.lo, family = binomial(link = probit))
> m.ca <- update(m.lo, family = binomial(link = cauchit))
> pMen <- data.frame(Age = seq(9, 18, len = 1000))
> pMen <- cbind(pMen, pM.lo = predict(m.lo, pMen, type = "resp"),
        pM.pr = predict(m.pr, pMen, type = "resp"), pM.ca = predict(m.ca,
            pMen, type = "resp"))

> with(pMen, {
        plot(Age, pM.lo, type = "l", col = "green4", ylim = 0:1)
        lines(Age, pM.pr, col = "blue")
        lines(Age, pM.ca, col = "red")
    })
> with(menarche, points(Age, Menarche/Total, pch = 8, cex = 0.5))
```



It is clear that the probit and logit links give very similar fitting models but the cauchit link appears to give a much worse fit. We can check this by looking at the summaries of the fitted models, (suppressing some details):

```
> summary(m.lo, corr = FALSE)

Call:
glm(formula = Menarche/Total ~ Age, family = binomial, data = menarche,
    weights = Total)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0363  -0.9953  -0.4900   0.7780   1.3675

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -21.22639    0.77068  -27.54   <2e-16
Age           1.63197    0.05895   27.68   <2e-16

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3693.884  on 24  degrees of freedom
Residual deviance:   26.703  on 23  degrees of freedom
AIC: 114.76

Number of Fisher Scoring iterations: 4
```

```
> summary(m.pr, corr = FALSE)

Call:
glm(formula = Menarche/Total ~ Age, family = binomial(link = probit),
    data = menarche, weights = Total)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5846  -0.9423  -0.4525   0.4433   1.7539

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.81894    0.38702  -30.54   <2e-16
Age           0.90782    0.02955   30.72   <2e-16

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3693.884  on 24  degrees of freedom
Residual deviance:   22.887  on 23  degrees of freedom
AIC: 110.94

Number of Fisher Scoring iterations: 5

> summary(m.ca, corr = FALSE)

Call:
glm(formula = Menarche/Total ~ Age, family = binomial(link = cauchit),
    data = menarche, weights = Total)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.9879  -2.2135   0.3429   1.3187   7.5396

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -33.5441     2.1691  -15.46   <2e-16
Age           2.5838     0.1668   15.49   <2e-16

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3693.88  on 24  degrees of freedom
Residual deviance:  180.86  on 23  degrees of freedom
AIC: 268.91

Number of Fisher Scoring iterations: 7
```

The large deviance for the cauchit model, (180.86 on 23 d.f.) would suggest that this model by excluded on grounds of fit, but the other two models have acceptable deviances from the fitted model.]

Now consider the analysis with the data presented in *binary* form, that is with one entry for each student in the sample. [Hint: One way to get the data in binary form is as follows:

```
menarche_binary <- with(menarche,
rbind(data.frame(Age = rep(Age, Menarche), Men = T),
      data.frame(Age = rep(Age, Total-Menarche), Men = F)))
```

Then the models may be fitted with `Men` as the binary response and `Age` as the predictor.]

Show computationally that fitting the model in this form,

(a) The estimated coefficients, their standard errors and $t-$statistics are the exactly the same as for the same model fitted with the data in frequency form,

(b) The Deviance is *not* the same, but

(c) If you fit sub-models, *differences* of deviance are the same for the data in both forms.

(d) For the data in binary form, fir the model `Men ~ factor(Age)` and test the straight line model as a sub-model. What do you notice?

(You need only do this with one of the link functions.)

[A solution: Consider the probit link for example.

```
> menarche_binary <- with(menarche, rbind(data.frame(Age = rep(Age,
        Menarche), Men = T), data.frame(Age = rep(Age, Total - Menarche),
        Men = F)))
> bm.pr <- glm(Men ~ Age, binomial(link = probit), menarche_binary)
```

Now compare the coefficients, standard errors and $t-$statistics.

```
> summary(m.pr)$coefficients

              Estimate Std. Error   z value      Pr(>|z|)
(Intercept) -11.818942 0.38701607 -30.53863 8.004674e-205
Age           0.907823 0.02955339  30.71807 3.265395e-207

> summary(bm.pr)$coefficients

               Estimate Std. Error   z value      Pr(>|z|)
(Intercept) -11.8189294 0.38700197 -30.53971 7.744521e-205
Age           0.9078222 0.02955245  30.71902 3.171969e-207

> c(m.pr = deviance(m.pr), bm.pr = deviance(bm.pr))

      m.pr      bm.pr
  22.88743 1635.48872
```

The estimates their standard errors are the same, but the deviances are very different. Now consider, say, a quadratic model in age.

```
> m.pr2 <- update(m.pr, . ~ . + I(Age^2))
> bm.pr2 <- update(bm.pr, . ~ . + I(Age^2))

> anova(m.pr, m.pr2, test = "Chisq")

Analysis of Deviance Table

Model 1: Menarche/Total ~ Age
Model 2: Menarche/Total ~ Age + I(Age^2)
```

```
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        23    22.8874
2        22    15.1488  1   7.7387    0.0054
```

```
> anova(bm.pr, bm.pr2, test = "Chisq")

Analysis of Deviance Table

Model 1: Men ~ Age
Model 2: Men ~ Age + I(Age^2)
  Resid. Df Resid. Dev   Df Deviance P(>|Chi|)
1      3916    1635.49
2      3915    1627.75    1     7.74      0.01
```

The tests are the same (thought with only 2 decimal places this is not immediately obvious, but it can be checked!). Now consider fitting a model with Age as a factor with the binary version of the data and checking the linear version as a sub-model:

```
> bm.prf <- try(update(bm.pr, . ~ factor(Age)))

> anova(bm.pr, bm.prf, test = "Chisq")

Analysis of Deviance Table

Model 1: Men ~ Age
Model 2: Men ~ factor(Age)
  Resid. Df Resid. Dev   Df Deviance P(>|Chi|)
1      3916    1635.49
2      3893    1612.60   23    22.89      0.47

> c(m.pr = deviance(m.pr), "bm.pr within bm.prf" = deviance(bm.pr) -
        deviance(bm.prf))

             m.pr bm.pr within bm.prf
         22.88743            22.88743
```

The factor model can cause some convergence problems, but if it works the deviance is usually accurate. The difference in deviance in the binary data case is the actual deviance in the grouped data case.

If you can, give a theoretical explaination of these results you have observed from the computation.

The deviances are different because the saturated models are different. The model with factor(Age) in the binary case corresponds to the saturated model for the grouped data case, (which partially explains the convergence problems when this model is fitted in the binary data case).

3. For the gamma distribution, defined as having probability densithy function

$$f_Y(y;\alpha,\phi) = \frac{e^{-y/\alpha} y^{\phi-1}}{\alpha^{\phi}\Gamma(\phi)}, \qquad 0 < y < \infty$$

   (a) Show that it belongs to the generalized linear modelling family and find the key functions $\theta(\mu)$ and $b(\theta)$;

(b) Hence write down the Cumuland Generating Function, $K_Y(t)$ and find the mean and variance in terms of the original parameters,

(c) Verify that $\theta(\mu)$ is an *increasing* function of μ.

(d) Find the *natural* link.

A solution: Since φ is used in the in the notation for the general case we will use τ instead of ϕ in this case to avoid confusion. The density may be written as

$$f_Y(y;\alpha,\tau) = \exp\left[\tau\left\{y\left(-\frac{1}{\alpha\tau}\right) - \log(\alpha\tau)\right\} + \tau\log\tau + (\tau-1)\log y - \log\Gamma(\tau)\right]$$

Hence, identifying the correspinding parts of the general form:

$$\begin{aligned}
\varphi &= 1/\tau \quad \text{and} \quad A = 1 \\
\theta &= -\tfrac{1}{\alpha\tau} \\
b(\theta) &= \log(-1/\theta) \quad \text{and so} \\
\mu &= b'(\theta) = -1/\theta = \alpha\tau \\
\text{var}[Y] &= \varphi b''(\theta)/A = 1/\tau \times (\alpha\tau)^2 = \alpha^2\tau
\end{aligned}$$

The cumulant generating function is therefore:

$$\begin{aligned}
K_Y(t) &= \tfrac{A}{\varphi}\left\{b\left(\theta + \tfrac{t\varphi}{A}\right) - b(\theta)\right\} \\
&= \tau\left\{\log\left(-\tfrac{1}{\theta+t\varphi}\right) - \log\left(-\tfrac{1}{\theta}\right)\right\} \\
&= \cdots = -\tau\log(1-\alpha\tau)
\end{aligned}$$

So, expanding in a powerseries in t we see that the cumulants are $\kappa_r = (r-1)!\alpha^r\tau$. In particular $\mu = \kappa_1 = \alpha\tau$ and $\sigma^2 = \kappa_2 = \alpha^2\tau$, confirming the results shown above.

The natural link funciton is the one for which $\theta\left(\mu(\eta)\right) \equiv \eta$. Since $\theta(\mu) = -1/\mu$ it follows that the *inverse* of the natural link is $\mu = \ell^{-1}(\eta) = -1/\eta$. So the link itself is $\eta = \ell(\mu) = -1/\mu$. This is called the "inverse" (or "reciprocal") link. The negative sign is usually omitted. It is not used very much in practice.

4. The 'credit card' data set `CC` comes from a commercial bank in Switzerland. The response of inerest is the variable `credit.card.owner`, which is a binary response stating whether or not the person has a credit card with the bank. There is also a large set of candidate predictors from which to build a predictive model for the binary response, which was the purpose for which the data were collected.

```
> Attach() # your data sets
> with(CC, table(credit.card.owner))
credit.card.owner
  no  yes
 609 1011
```

(a) Split the data into two parts of about 800 observations each, a 'training' and 'test' set. Build models from the training set and test them on the remainder. [Hint: one way to do this is

```
set.seed(12354)  # choose a suitable seed
ind <- sample(1:nrow(CC), 800)
CCTrain <- CC[ind, ]
CCTest <- CC[-ind, ]
```

and check the sizes of both.]

Before splitting the data, first find the factor predictors and for each, check that each level has a reasonable occupancy (as we did with the birth weight data example)

```
> names(CC)[sapply(CC, is.factor)]
[1] "credit.card.owner" "profession"        "sex"
[4] "nationality"
> with(CC, table(profession))
profession
 business   chemist    doctor  engineer    lawyer      none     nurse  physical
      177       234         4       314       172       276         5         5
   police professor   service   teacher
      159         1         6       267
> with(CC, table(sex))
sex
  F   M
803 817
> with(CC, table(nationality))
nationality
  CH   FR   GB   GE   IT   YU
1381   18   16   37   65  103
> myCC <- CC
> levels(myCC$profession)[c(3, 7, 8, 10, 11)] <- "other"
> levels(myCC$nationality)[2:5] <- "other"
> with(myCC, table(credit.card.owner, profession))
                profession
credit.card.owner business chemist other engineer lawyer none police teacher
              no        66      76     7       93     23  210     48      86
              yes      111     158    14      221    149   66    111     181
> with(myCC, table(credit.card.owner, sex))
                sex
credit.card.owner   F   M
              no  422 187
              yes 381 630
> with(myCC, table(credit.card.owner, nationality))
                nationality
credit.card.owner  CH other  YU
              no  527    48  34
              yes 854    88  69
```

Each factor level now has a reasonable number of trials.

Using the code in the hint, we next split the data and store the results in the `.R_Store` directory to free memory.

```
> set.seed(123541)
> ind <- sample(1:nrow(myCC), 800)
> CCTrain <- myCC[ind, ]
> CCTest <- myCC[-ind, ]
> Store(myCC, CCTrain, CCTest)
```

In the stepwise procedure we need to specify all vaiables as candidates for selection. Rather than type them all out, we put together a formula using R itself.

```
> mform <- as.formula(paste("~", paste(names(myCC)[-1], collapse = "+")))
```

You should print the result to see that it is the complete formula.

(b) Starting with any suitable model, use automatic stepwise techniques to arrive at a suitable logistic regression model. You need only consider main effect terms. Compare the result of using AIC and BIC as your selection crierion.

One possibility is to start with a minimal model:

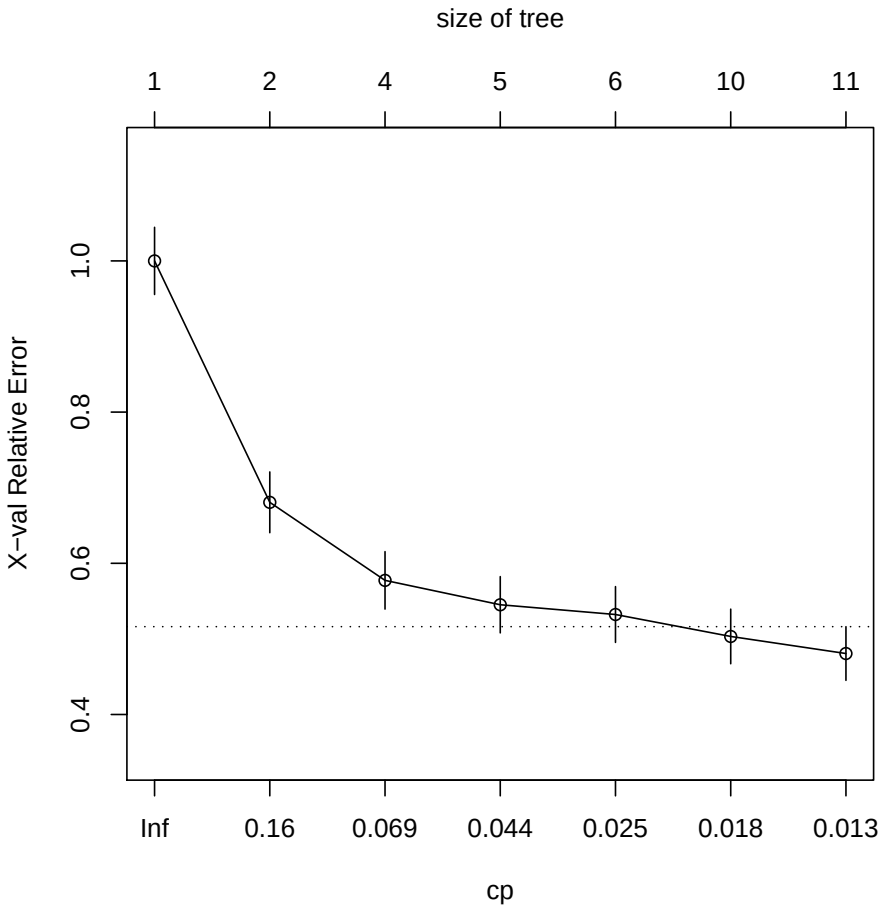
```
> cc.mod0 <- glm(credit.card.owner ~ 1, binomial, CCTrain)
> cc.AIC <- try(stepAIC(cc.mod0, scope = list(lower = ~1, upper = mform),
        trace = FALSE))
> cc.BIC <- try(stepAIC(cc.mod0, scope = list(lower = ~1, upper = mform),
        k = log(800), trace = FALSE))
```

(c) Construct two other predictive models, namely

i. A tree model, fitted by `rpart` from the `rpart` package, and pruned by the 'One standard error' rule,

To fit the tree model:

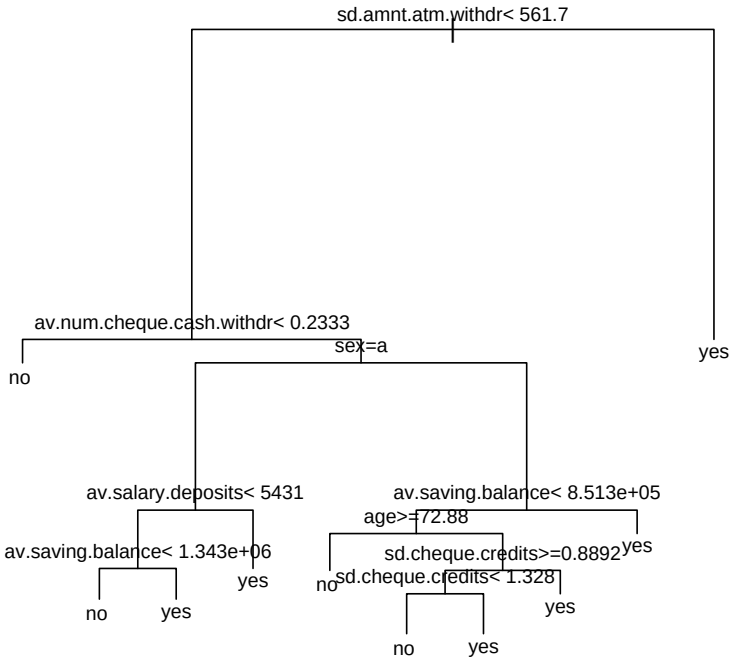
```
> library(rpart)
> cc.rpart <- rpart(credit.card.owner ~ ., CCTrain)
> plotcp(cc.rpart)
```



In this case the 'one SE' rule suggests that pruning shoud be done with complexity parameter set to 0.018. Prune and see what the tree itself looks like:

```
> cc.rpart <- prune(cc.rpart, cp = 0.018)
> plot(cc.rpart)
```

```
> text(cc.rpart, xpd = NA, cex = 0.75)
```



ii. A random forest model, fitted by `randomForest` from the `randomForest` package.

To fit the random forest model:

```
> library(randomForest)
> cc.rf <- randomForest(credit.card.owner ~ ., CCTrain)
```

(d) For each fitted model find the 'confusion matrix' when testing it on the test set and compare each of the models by their crude error rates.

The models you should consider are

i. Your original logistic regression,
ii. The stepwise model got by using AIC,
iii. The stepwise model got by using BIC,
iv. The tree model,
v. The random forest model.

For the regression models, predict '`yes`' if the predicted probability equals or exceeds 0.5. For the tree and random forest models predict with `type = "class"`.

It is useful to have a small function available to show both the confusion matrix and the error rate:

```
> check <- function(pValue) {
        if (is.logical(pValue))
            pValue <- factor(c("no", "yes")[pValue + 1], levels = c("no",
                "yes"))
        confusion <- table(pValue, CCTest$credit.card.owner)
        error_rate <- round(100 * (1 - sum(diag(confusion))/sum(confusion)),
```

```
            2)
        list(confusion = confusion, error_rate = error_rate)
    }
> check(predict(cc.mod0, CCTest) > 0)
$confusion

pValue  no yes
   no    0   0
   yes 299 521

$error_rate
[1] 36.46
> check(predict(cc.AIC, CCTest) > 0)
$confusion

pValue  no yes
   no  207  64
   yes  92 457

$error_rate
[1] 19.02
> check(predict(cc.BIC, CCTest) > 0)
$confusion

pValue  no yes
   no  210  70
   yes  89 451

$error_rate
[1] 19.39
> check(predict(cc.rpart, CCTest, type = "class"))
$confusion

pValue  no yes
   no  214  57
   yes  85 464

$error_rate
[1] 17.32
> check(predict(cc.rf, CCTest, type = "class"))
$confusion

pValue  no yes
   no  227  39
   yes  72 482

$error_rate
[1] 13.54
```

Submit your exercise as before as an annotated working R script.