STAT 260 - Nonparametric Statistics

Ying Sun
Statistics (STAT) Program
Computer, Electrical and Mathematical Sciences & Engineering (CEMSE) Division
King Abdullah University of Science and Technology (KAUST)

HOMEWORK III

Henrique Aparecido Laureano

Spring Semester 2018

Contents

Problem 1

Residual checking for non-Gaussian error models is not always as straightforward as it is in the Gaussian case, and the problems are particularly acute in the case of binary data. This question explores this issue.

(a)

The following code fits a GLM to data simulated from a simple binomial model and examines the default residual plots.

```
# <r code> ============================================================================ #
n <- 100 ; m <- 10                                  # defining number of observations
x <- runif(n)           # simulating n uniform observations with parameters 0 and 1
lp <- 3 * x - 1            # lp: linear predictor. \beta_{0} = -1 and \beta_{1} = 3
mu <- binomial()$linkinv(lp)     # generating n values of the inverse link function
y <- rbinom(1:n, m, mu)     # generating n binomial sample with parameters m and mu
par(mfrow = c(2, 2), mar = c(4, 4, 2, 2) + .1)              # graphical definition
              # fitting a glm with response y/m, covariate x and using m as weight
                   # plotting the graphs of goodness of fit (analysis of residues)
plot( glm(y/m ~ x, family = binomial, weights = rep(m, n)) )
# </r code> ============================================================================ #
```

Run the code several times to get a feel for the range of results that are possible even when the model is correct (as it clearly is in this case).

Solution:

```
# <r code> ============================================================================ #
prob1.a <- function(n = 100, m = 10) {
  x = runif(n)
  lp = 3 * x - 1
  mu = binomial()$linkinv(lp)
  y = rbinom(1:n, m, mu)
  plot( glm(y/m ~ x, family = binomial, weights = rep(m, n)) )
}
par(mfrow = c(6, 4), mar = c(4, 4, 2, 2) + .1)
prob1.a() ; prob1.a() ; prob1.a()
prob1.a() ; prob1.a() ; prob1.a()
# </r code> ============================================================================ #
```

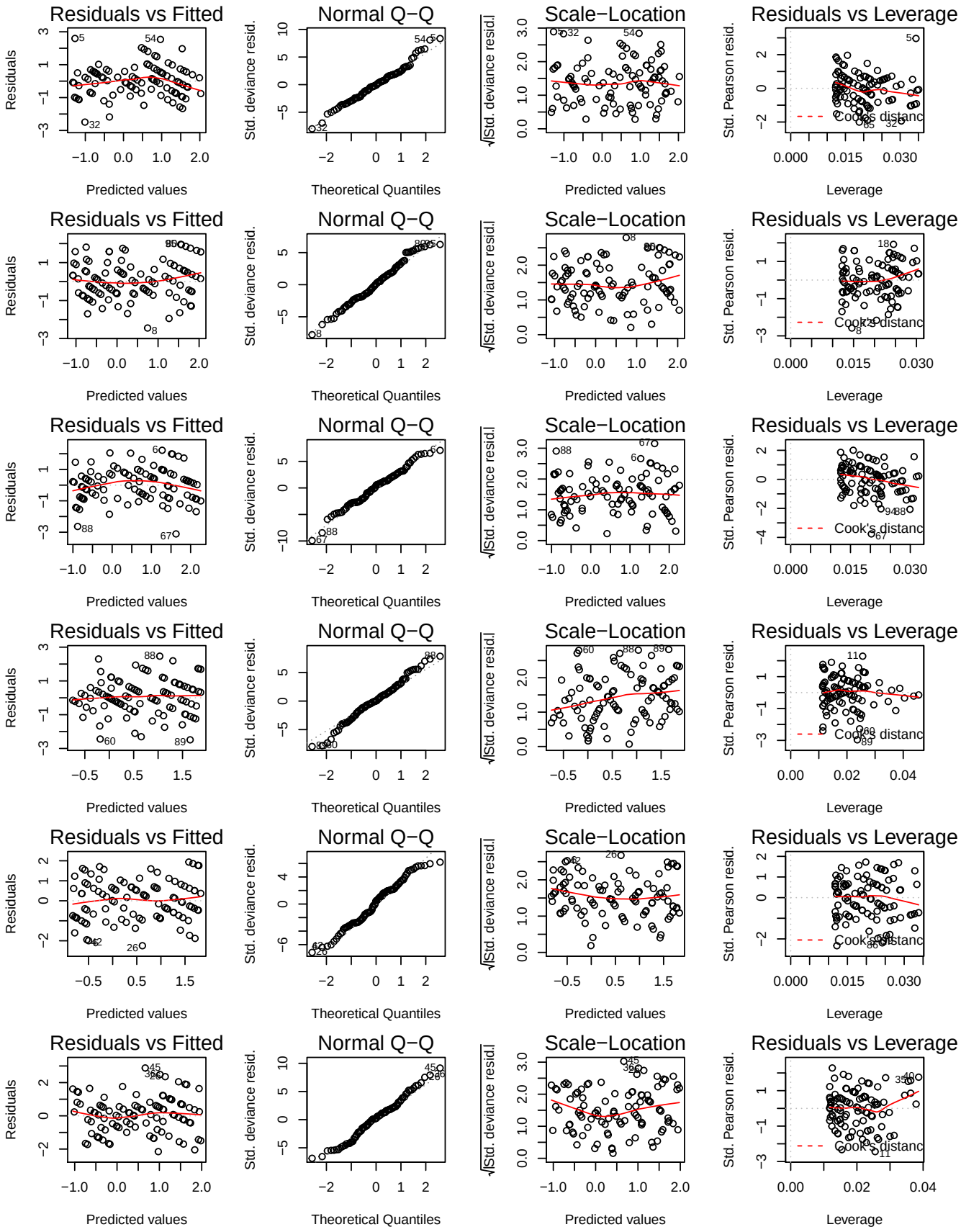

Figure 1: Graphs of goodness of fit (analysis of residues), each line is with a different data sample.

In Figure 1 are presented the results from running the code six times (to see the behavior was runned more, buth to present the behavior in a non-tedious form was choosed to present just six).

General remarks: with different samples we see very few behaviors with a slightly scape of the assumptions, and with a very small scape, e.g. a slightly increase or decrease tendence in the residuals vs fitted values and in the scale-location relationship, some values that scape a little in the Q-Q graph. However, this isn't the common behavior. In general the graphical analysis of residuals looks good, with no considerable scapes of the assumptions (homoscedasticity, scale-location constant relation, residuals normality, non-(high) leverage values). If the model is correct, the results are pretty stables (different samples (from the same distribution) returning very similar behaviors).

□

(b)

Explore how the plots change as m (the number of binomial trials) is reduced to 1. Also examine the effect of sample size n.

Solution:

Twelve scenarios was explored.
Four different numbers of binomial trials (10, 7, 5, 3 and 1) and
three different sample sizes (100, 50 and 25).

Figures 2, 3 and 4:

```
# <r code> ============================================================================ #
                                                                  # graphical definitions
par(mfrow = c(5, 4), mar = c(4, 4, 2, 2) + .1)

                                                          # number of binomial trials
for (i in c(10, 7, 5, 3, 1))
                                                                      # sample size 100
  prob1.a(n = 100, m = i)
# </r code> ============================================================================ #
```

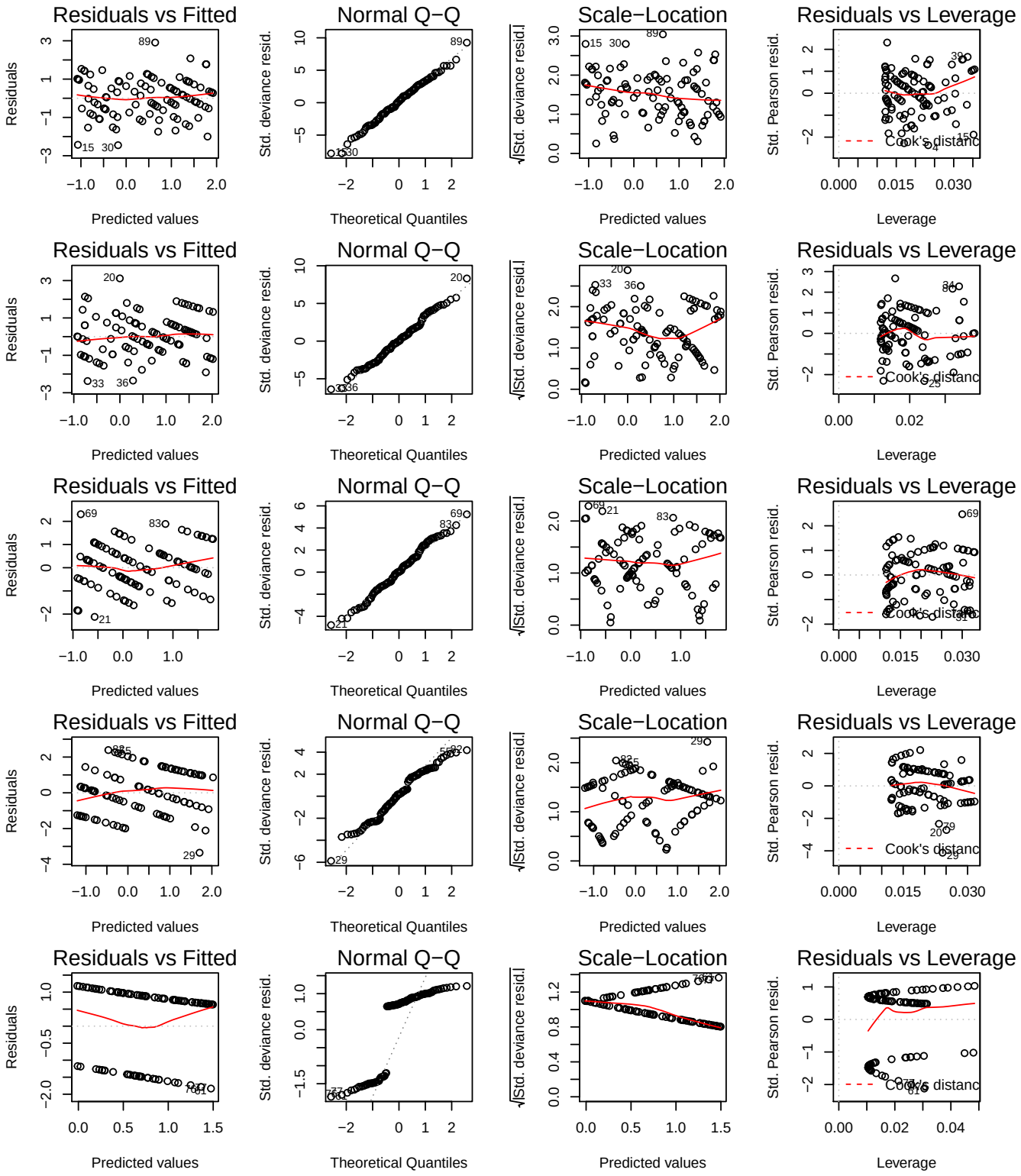

Figure 2: Graphs of goodness of fit for $m = 10$ (first line), 7 (second line), 5 (third), 3 (fourth) and 1 (fifth), with $n = 100$.

```
par(mfrow = c(5, 4), mar = c(4, 4, 2, 2) + .1)                    # <r code>
for (i in c(10, 7, 5, 3, 1)) prob1.a(n = 50, m = i)               # </r code>
```

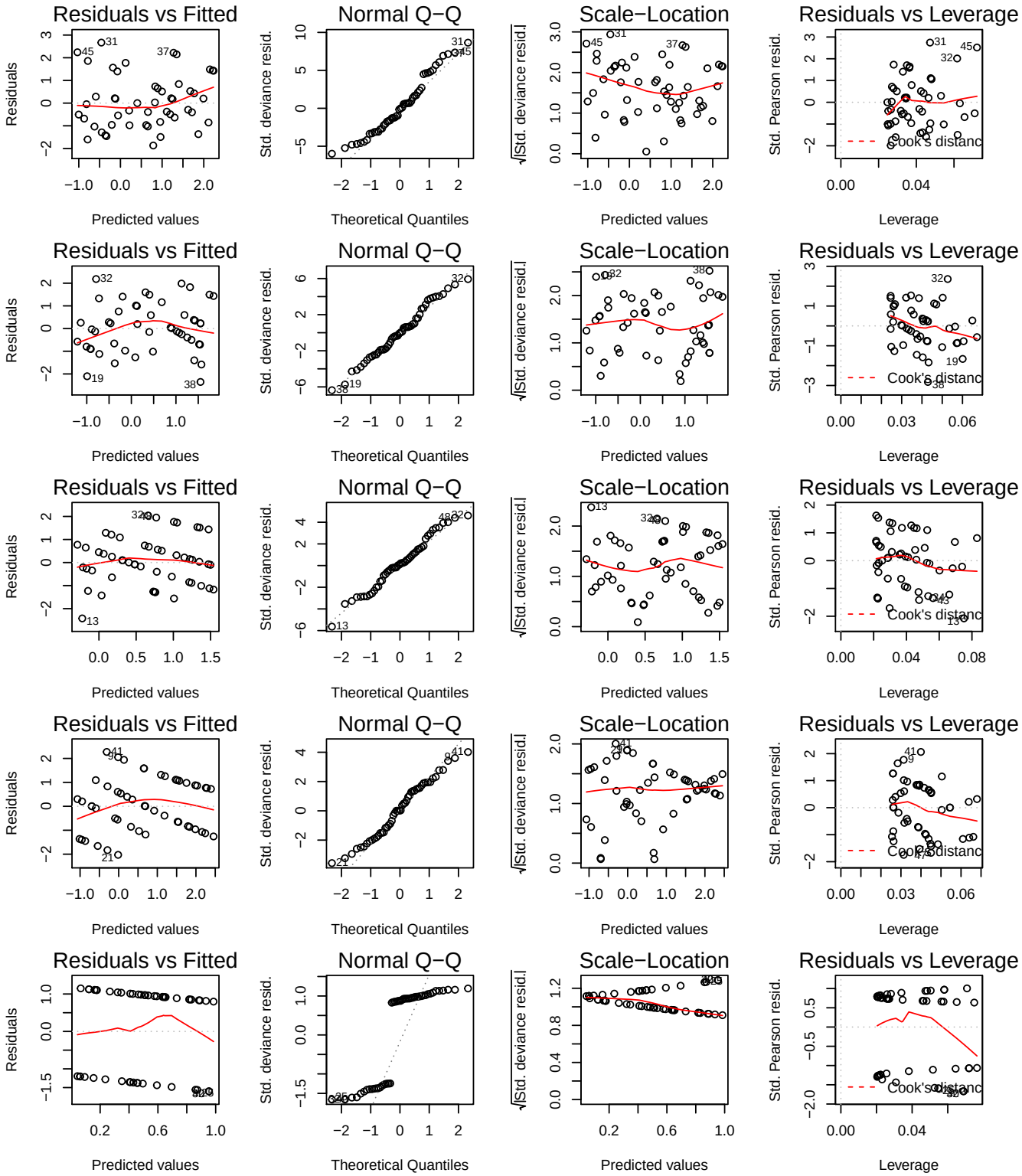

Figure 3: Graphs of goodness of fit for $m = 10$ (first line), 7 (second line), 5 (third), 3 (fourth) and 1 (fifth), with $n = 50$.

```
par(mfrow = c(5, 4), mar = c(4, 4, 2, 2) + .1)                    # <r code>
for (i in c(10, 7, 5, 3, 1)) prob1.a(n = 25, m = i)               # </r code>
```

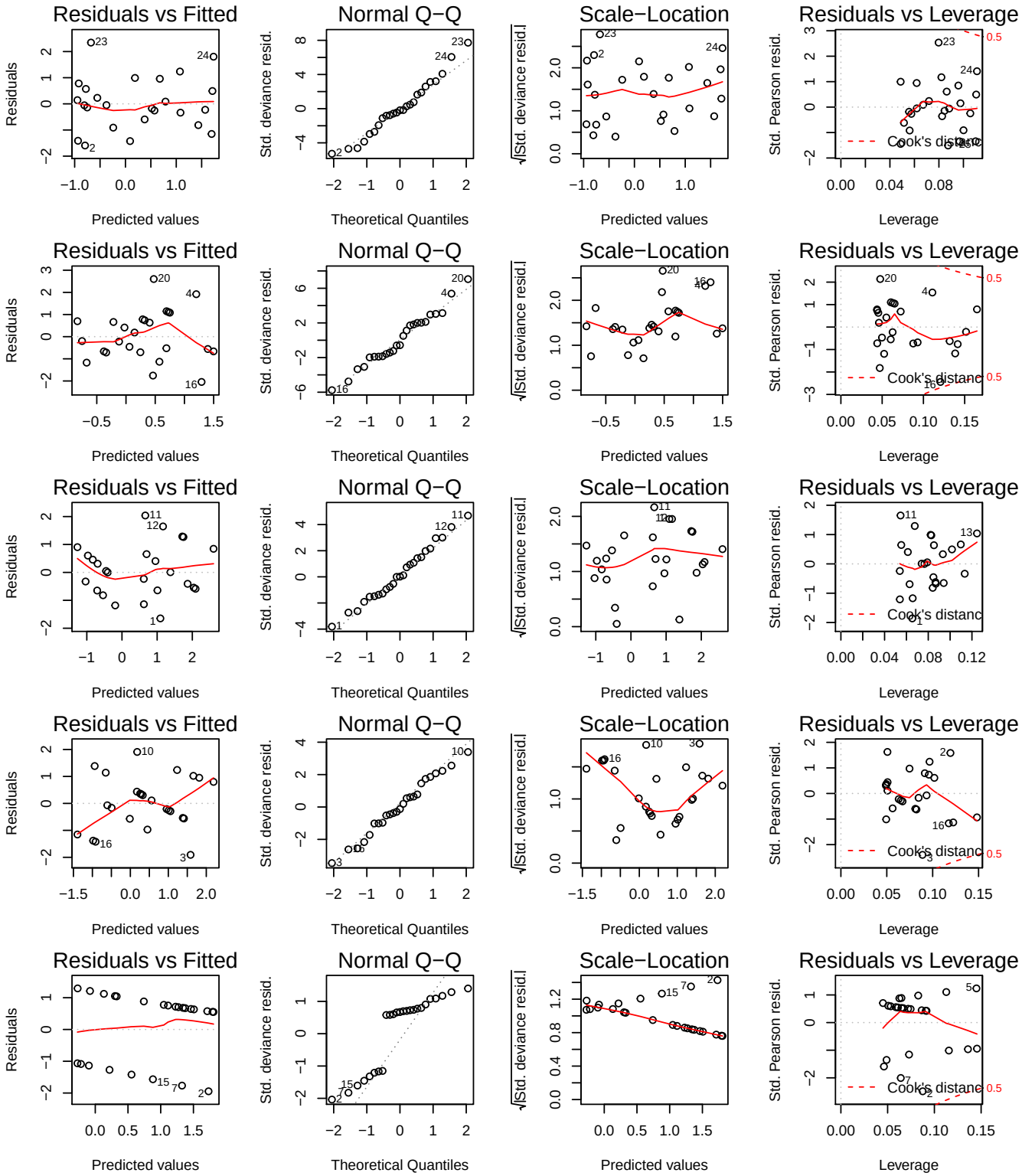

Figure 4: Graphs of goodness of fit for $m = 10$ (first line), 7 (second line), 5 (third), 3 (fourth) and 1 (fifth), with $n = 25$.

In the Figures 2, 3 and 4 we see that changing the sample size don't cause big differences in the results (graphicals for the goodness of fit). In reality, considering that with a small sample size the variances tend to be bigger, we can say that the behaviors with different sample sizes are pretty the same. By the other hand, when we change the number of binominal trials the results changing a lot, for worst (in all the considered sample sizes). With a number of ten binomial trials the results are good, but when the number decrease the results start to become bad. With seven we already see a difference. The graphs are still ok, acceptable, but we see slight scapes of the model assumptions mencioned in the letter **(a)**. Conform we decrease the number of trials the goodness of fit became worst and with a number of one trial the results are basically unacceptable. With less binomial trials we see that the residuals will becoming more dicotomical (with one trial this is literal).

□

(c)

By repeatedly simulating data from a fitted model, and then refitting to the simulated data, you can build up a picture of how the residuals should behave when the distributional assumption is correct, and the data are really independent. Write code to take a `glm` object fitted to binary data, simulate binary data given the model fitted values, refit the model to the resulting data, and extract residuals from the resulting fits. Functions `fitted` and `rbinom` are useful here.

Solution:

We repeated the process 44 times, the residuals behave are presented in the Figure 5.

```
# <r code> ============================================================================ #
n <- 100 ; m <- 10                            # n observations and m binomial trials
rsd <- matrix(NA, nrow = 45, ncol = n)              # object to keep the residuals
x <- runif(n)                               # generating a random vector of covariates
par(mfrow = c(9, 5), mar = c(2, 4, 2, 1) + .1)             # graphical definitions
for (i in 1:45) {                                                  # 45 repetitions
  lp = 3 * x - 1                                                  # linear predictor
  mu = binomial()$linkinv(lp)                                    # inverse link function
  y = rbinom(1:n, m, mu)                                 # computing the binomial samples
  model = glm(y/m ~ x, family = binomial, weights = rep(m, n))     # model fitting
  rsd[i, ] = residuals(model)                                    # keeping the residuals
  x = fitted(model, type = "response") # generating new data from the fitted model
  plot(rsd[i, ], xlab = NA, ylab = "Residuals", main = paste("Fit:", i-1))  # plot
}
# </r code> ============================================================================ #
```

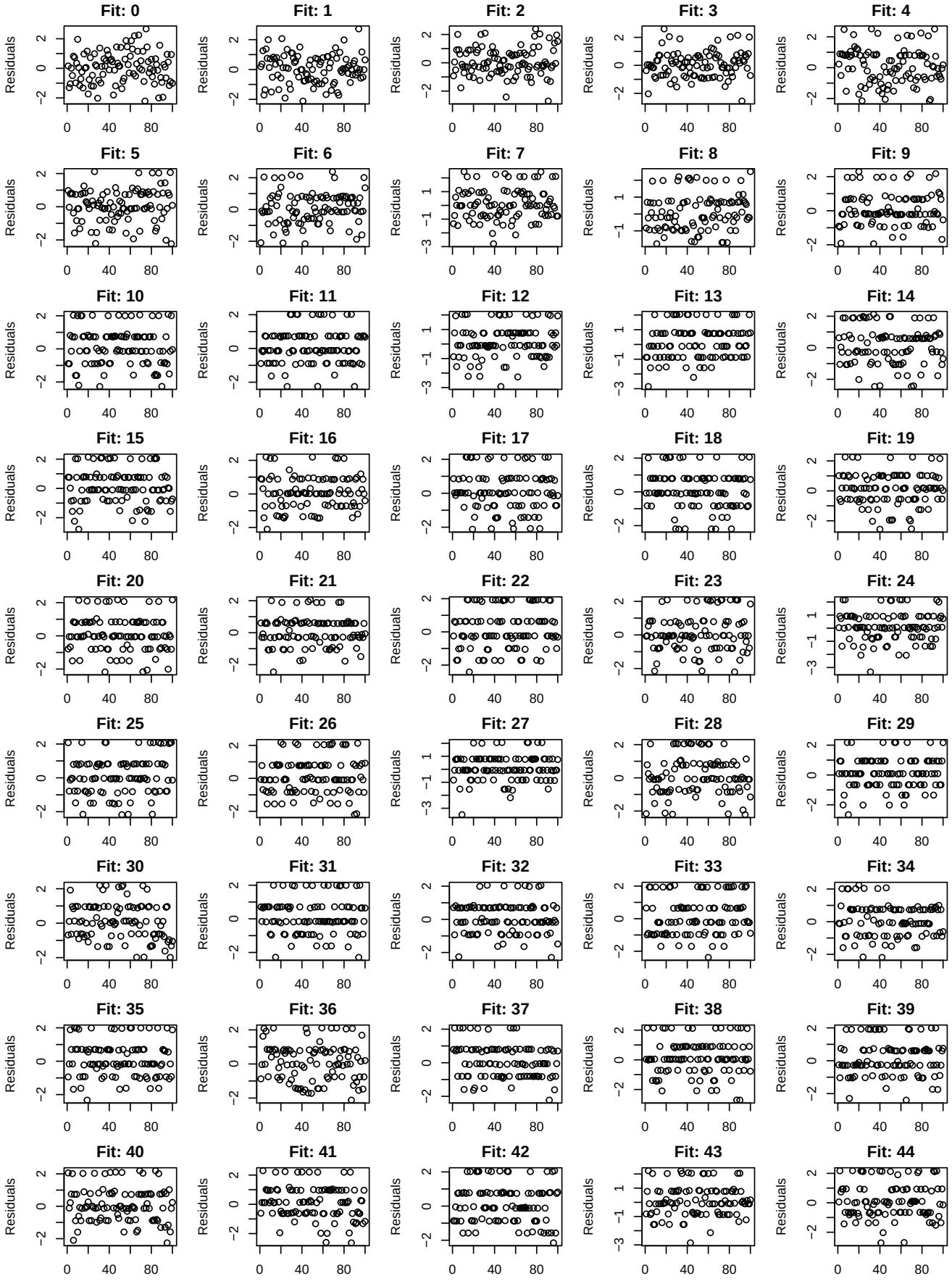

Figure 5: Residues for each (re)fitted model.

We see in Figure 5 that in the first five replications the residuals still present a random behavior, but always between -3 and 3. After this first replications we see that some patterns are created. Residuals form groups, as if the data were discretized. In fit 42, for example, with the exception of one value, we see as if only five values appear (with the addition of some small random noise).

□

(d)

If `rsd` contains your residual vector, then `plot(sort(rsd), (1:length(rsd) - .5) / length(rsd))` produces a plot of the 'empirical CDF' of the residuals, which can be useful for characterizing their distribution. By repeatedly simulating residuals, as in the previous part, you can produce a 'simulation envelope' showing, e.g., where the middle 95% of these 'empirical CDFs' should lie, if the model assumptions are met: such envelopes provide a guide to whether an observed 'real' residual distribution is reasonable, or not. Based on your answer to the previous part, write code to do this.

Solution:

```
# <r code> ============================================================================ #
n <- 100 ; m <- 10                              # n observations and m binomial trials
x <- runif(n)                                 # generating a random vector of covariates
lp <- 3 * x - 1                                                      # linear predictor
mu <- binomial()$linkinv(lp)                                      # inverse link function
y <- rbinom(1:n, m, mu)                                    # computing the binomial samples
model <- glm(y/m ~ x, family = binomial, weights = rep(m, n))       # model fitting
rsd <- matrix(NA, nrow = 45, ncol = n)                 # object to keep the residuals
rsd[1, ] = sort(residuals(model))                  # keeping and sorting the residuals

for (i in 2:45) {                                                     # 44 repetitions
  x = fitted(model, type = "response") # generating new data from the fitted model
  lp = 3 * x - 1                                                     # linear predictor
  mu = binomial()$linkinv(lp)                                     # inverse link function
  y = rbinom(1:n, m, mu)                                   # computing the binomial samples
  model = glm(y/m ~ x, family = binomial, weights = rep(m, n))      # model fitting
  rsd[i, ] = sort(residuals(model))                # keeping and sorting the residuals
}
x.axes <- (1:n - .5) / n                                        # defining the x axes
lower <- apply(rsd[-1, ], 2, quantile, .025)             # simulations 2.5% quantile
med   <- apply(rsd[-1, ], 2, quantile, .5)                # simulations 50% quantile
upper <- apply(rsd[-1, ], 2, quantile, .975)            # simulations 97.5% quantile
par(mar = c(4, 4, 0, 2) + .1)
plot(rep(x.axes, 2), c(lower, upper), type = "n"                          # plotting
     , xlab = "Probability", ylab = "Residuals")
lines(x.axes, lower) ; lines(x.axes, med, lty = 2) ; lines(x.axes, upper)
points(x.axes, rsd[1, ])
# </r code> ============================================================================ #
```

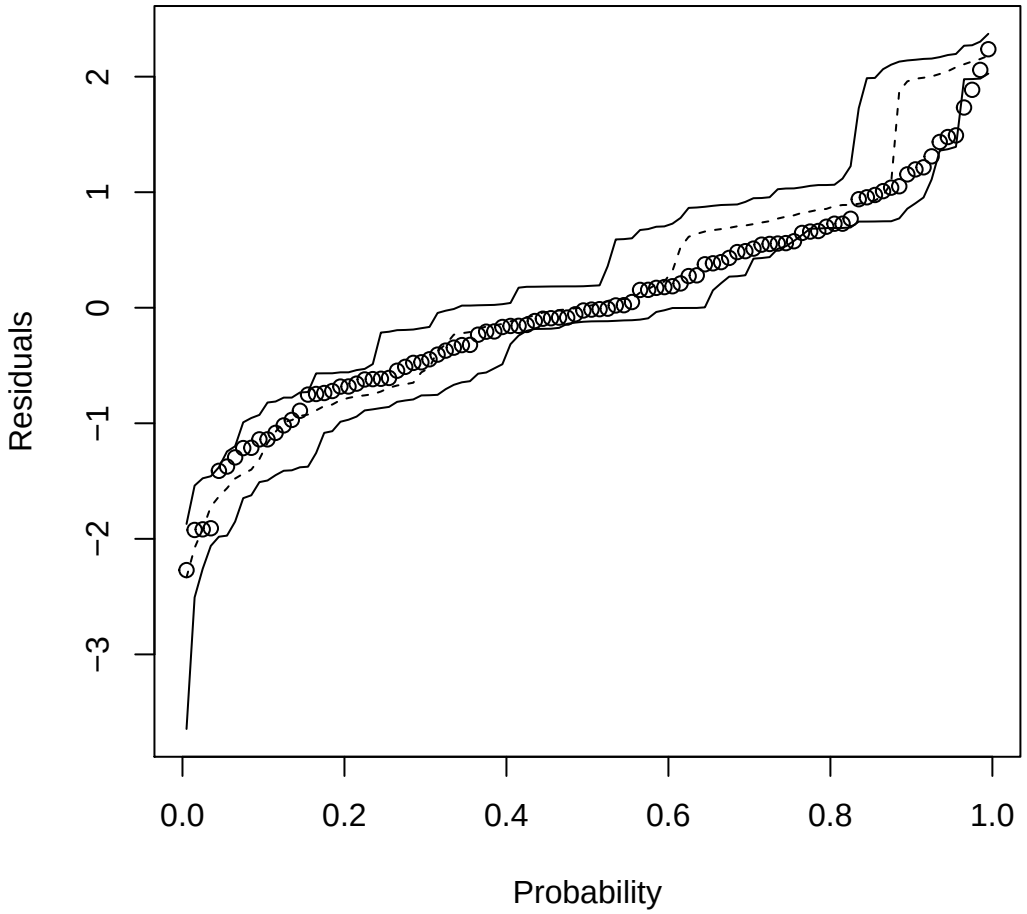

Figure 6: 95% simulated envelope for the residues empirical CDF.

We see in Figure 6 that the simulated envelope isn't so wide, i.e. it is quite precise, and that the residuals of the original model are inside the envelope, scaping a little in the right top part. We expect that 95 in each 100 samples/residuals be inside the envelope.

□

(e)

Plots of the residuals against fitted values, or predictors, are also hard to interpret for models of binary data. A simple check for lack of independence in the residuals can be obtained by ordering the residuals according to the values of the fitted values or a predictor, and checking whether these ordered residuals show fewer (or more) runs of values above and below zero than they should if independent. The command `rsd <- rsd[sort(fv, return.index = TRUE)$ix]` will put `rsd` into the order corresponding to increasing `fv`. It is possible to simulate from the distribution of runs of residuals that should occur, under independence, as part of the simulation loop used in the previous part: modify your code to check whether the residuals appear non-independent with respect to fitted values.

Solution:

In Figure 7 (next page) we see that the sorted residuals by the fitted values form a pattern quickly. After some few simulations we already see that the same values appear several times with the addition of a small noise.

Even after several simulations the residuals still remain distributed around zero and for all the scenarios the residuals still appear independent with respect to the fitted values.

```
# <r code> ============================================================================ #
n <- 100 ; m <- 10                              # n observations and m binomial trials

x <- runif(n)                                 # generating a random vector of covariates
lp <- 3 * x - 1                                                      # linear predictor
mu <- binomial()$linkinv(lp)                                    # inverse link function
y <- rbinom(1:n, m, mu)                                  # computing the binomial samples

model <- glm(y/m ~ x, family = binomial, weights = rep(m, n))       # model fitting

rsd <- matrix(NA, nrow = 45, ncol = n)                 # object to keep the residuals
       # keeping the residuals sorting by the fitted values in an increasing order
rsd[1, ] = residuals(model)[order(fitted(model))]

par(mfrow = c(9, 5), mar = c(2, 4, 2, 1) + .1)                # graphical definitions

plot(rsd[1, ], xlab = NA, ylab = "Residuals", main = "Fit: 1")            # plotting

for (i in 2:45) {                                                    # 44 repetitions
  x = fitted(model, type = "response") # generating new data from the fitted model
  lp = 3 * x - 1                                                     # linear predictor
  mu = binomial()$linkinv(lp)                                    # inverse link function
  y = rbinom(1:n, m, mu)                                  # computing the binomial samples

  model = glm(y/m ~ x, family = binomial, weights = rep(m, n))      # model fitting

       # keeping the residuals sorting by the fitted values in an increasing order
  rsd[i, ] = residuals(model)[order(fitted(model))]
                                                                                # plotting
  plot(rsd[i, ], xlab = NA, ylab = "Residuals", main = paste("Fit:", i))
}
# </r code> ============================================================================ #
```

Figure 7: Sorted residuals by the fitted values in an increasing order.

□

Problem 2

This question relates to the IRLS method, and gives an alternative insight into the distribution of the parameter estimators $\hat{\beta}$. Let y_i be independent random variables with mean μ_i, such that $g(\mu_i) = \eta_i = X_i\beta$, where g is a link function, X a model matrix and β a parameter vector. Let the variance of y_i be $V(\mu_i)\phi$, where V is a known function, and ϕ a scale parameter. Define

$$z_i = g^{'}(\mu_i)(y_i - \mu_i) + \eta_i \quad \text{and} \quad w_i = \{V(\mu_i)g^{'}(\mu_i)^2\}^{-1}.$$

(a)

Show that $\mathbb{E}(z_i) = X_i\beta$.

Solution:

$$\begin{aligned}\mathbb{E}(z_i) = \mathbb{E}\Big[g^{'}(\mu_i)(y_i - \mu_i) + \eta_i\Big] = \mathbb{E}\Big[g^{'}(\mu_i)y_i\Big] - \mathbb{E}\Big[g^{'}(\mu_i)\mu_i\Big] + \mathbb{E}(\eta_i) &= \mathbb{E}(X_iy_i) - \mathbb{E}(X_i\mu_i) + X_i\beta \\ &= X_i\mathbb{E}(y_i) - X_i\mu_i + X_i\beta \\ &= X_i\mu_i - X_i\mu_i + X_i\beta \\ &= \boxed{X_i\beta.}\end{aligned}$$

□

(b)

Show that the covariance matrix of z is $W^{-1}\phi$, where W is a diagonal matrix with $W_{i,i} = w_i$.

Solution:

$$\text{Cov}(z_i, z_j) = 0 \quad \text{for} \quad i \neq j, \quad \text{since } y_i \text{ are independent random variables.}$$

⇒

$$
\begin{aligned}
\mathrm{Var}(z_i) &= \mathbb{E}(z_i^2) - \mathbb{E}(z_i)^2 \\
&= \mathbb{E}\Big[\Big(g^{'}(\mu_i)(y_i-\mu_i)+\eta_i\Big)^2\Big] - \mathbb{E}^2\Big[g^{'}(\mu_i)(y_i-\mu_i)+\eta_i\Big] \\
&= \mathbb{E}\Big[\Big(g^{'}(\mu_i)(y_i-\mu_i)\Big)^2 + 2\eta_i g^{'}(\mu_i)(y_i-\mu_i)+\eta_i^2\Big] - (X_i\beta)^2 \\
&= \mathbb{E}\Big[\Big(g^{'}(\mu_i)(y_i-\mu_i)\Big)^2\Big] + 2\eta_i\mathbb{E}\Big(g^{'}(\mu_i)(y_i-\mu_i)\Big) + (X_i\beta)^2 - (X_i\beta)^2 \\
&= g^{'}(\mu_i)^2\mathbb{E}\Big(y_i^2 - 2\mu_i y_i + \mu_i^2\Big) + 2\eta_i\cdot 0 \\
&= g^{'}(\mu_i)^2\Big[\mathbb{E}(y_i^2) - 2\mu_i^2 + \mu_i^2\Big] \\
&= g^{'}(\mu_i)^2\Big[\mathbb{E}(y_i^2) - \mathbb{E}^2(y_i)\Big] \\
&= g^{'}(\mu_i)^2\mathrm{Var}(y_i) \\
&= g^{'}(\mu_i)^2 V(\mu_i)\phi.
\end{aligned}
$$

$$
\boxed{\mathrm{Var}(\mathbf{z}) = W^{-1}\phi, \quad \text{with } W \text{ being a diagonal matrix with } W_{i,i} = w_i = \{g^{'}(\mu_i)^2 V(\mu_i)\phi\}^{-1}.}
$$

□

(c)

If β is estimated by minimization of $\sum_i w_i(z_i - X_i\beta)^2$ show that the covariance matrix of the resulting estimates, $\hat{\beta}$, is $(X^\top W X)^{-1}\phi$, and find $\mathbb{E}(\hat{\beta})$.

Solution:

$$
W(\mathbf{z}-X\beta)^\top(\mathbf{z}-X\beta) = W\Big(\mathbf{z}^\top\mathbf{z} - 2\beta^\top X^\top\mathbf{z} + \beta^\top X^\top X\beta\Big)
$$

$$
\text{Finding } \hat{\beta}, \qquad \frac{\partial}{\partial\beta}\Big[W\Big(\mathbf{z}^\top\mathbf{z} - 2\beta^\top X^\top\mathbf{z} + \beta^\top X^\top X\beta\Big)\Big] = W\Big(-2X^\top\mathbf{z} + 2X^\top X\hat{\beta}\Big) = 0
$$

$$
\begin{aligned}
\Rightarrow \qquad X^\top W X\hat{\beta} &= X^\top W\mathbf{z} \\
\hat{\beta} &= (X^\top W X)^{-1}X^\top W\mathbf{z}.
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Var}(\hat{\beta}) &= \Big[((X^\top W X)^{-1}X^\top W)^\top(X^\top W X)^{-1}X^\top W\Big]\mathrm{Var}(\mathbf{z}) \\
&= ((X^\top W X)^{-1}X^\top W)^\top(X^\top W X)^{-1}X^\top W W^{-1}\phi \\
&= ((X^\top W X)^{-1}X^\top W X)^\top(X^\top W X)^{-1}\phi \\
&= \boxed{(X^\top W X)^{-1}\phi.}
\end{aligned}
$$

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}\Big[(X^\top W X)^{-1} X^\top W \mathbf{z}\Big] = (X^\top W X)^{-1} X^\top W \mathbb{E}(\mathbf{z}) = (X^\top W X)^{-1} X^\top W X \beta = \boxed{\beta.}$$

□

(d)

The multivariate version of the central limit theorem implies that as the dimension of z tends to infinity, $X^\top W \mathbf{z}$ will tend to multivariate Gaussian. What does this imply about the large sample distribution of $\hat{\beta}$?

Solution:

Implies that, from the general properties of maximum likelihood estimators, in the large sample limit

$$\boxed{\hat{\beta} \sim \text{Normal}(\beta, (X^\top W X)^{-1}\phi).}$$

□

Problem 3

`R` data frame `ldeaths` contains monthly death rates from 3 lung diseases in the UK over a period of several years (see `?ldeaths` for details and reference). One possible model for the data is that they can be treated as Poisson random variables with a seasonal component and a long term trend component, as follows:

$$\mathbb{E}(\text{deaths}_i) = \beta_0 + \beta_1 t_i + \alpha \sin(2\pi \text{toy}_i/12 + \phi),$$

where β_0, β_1, α, and ϕ are parameters, t_i is time since the start of the data, and `toy` is time of year, in months (January being month 1).

(a)

By making use of basic properties of sines and cosines, get this model into a form suitable for fitting using `glm`, and fit it. Use `as.numeric(ldeaths)` to treat `ldeaths` as a regular numeric vector rather than a time series object.

Solution:

Linear predictor:

$$\beta_0 + \beta_1 t_i + \alpha \sin(\pi \text{toy}_i/6 + \phi) = \beta_0 + \beta_1 t_i + \gamma \sin(\pi \text{toy}_i/6) + \delta \cos(\pi \text{toy}_i/6),$$

with $\alpha = \sqrt{\gamma^2 + \delta^2}$ and $\phi = \delta/\sqrt{\gamma^2 + \delta^2}$.

```r
# <r code> ============================================================================ #
y <- as.numeric(ldeaths)                                                  # response vector
t <- 1:length(y)                                 # t_{i}: time since the start of the data
toy <- rep(1:12, 6)                                    # toy_{i}: time of year, in months
formula = y ~ t + sin(toy * pi / 6) + cos(toy * pi / 6)                    # model formula
(fit <- glm(formula, family = "poisson"))                               # fitting the model
# </r code> ============================================================================ #
```

```
Call:  glm(formula = formula, family = "poisson")

Coefficients:
    (Intercept)                t  sin(toy * pi/6)  cos(toy * pi/6)
       7.682867        -0.002496         0.303668         0.216698

Degrees of Freedom: 71 Total (i.e. Null);  68 Residual
Null Deviance:     12330
Residual Deviance: 1514  AIC: 2200
```

□

(b)

Plot the raw data time series on a plot, with the predicted time-series overlaid.

Solution:

```r
# <r code> ============================================================================ #
par(mar = c(4, 4, 1, 1) + .1)                                        # graphical definitions
                                                 # plotting the raw data time series
plot(y ~ t, type = "l", axes = FALSE, xlab = "Time", ylab = "ldeaths", lwd = 2)
abline(v = c(0, 72), h = seq(min(y), max(y), length = 2), col = "gray90")
lines(t, fitted(fit), col = 2, lwd = 2)          # adding the predicted time-series
Axis(side = 1                                                       # plotting x- and y-axes
     , at = 12 * c(0:6), labels = c(1974, 1975, 1976, 1977, 1978, 1979, 1980))
Axis(side = 2, at = seq(1300, 4000, 500))
# </r code> ============================================================================ #
```

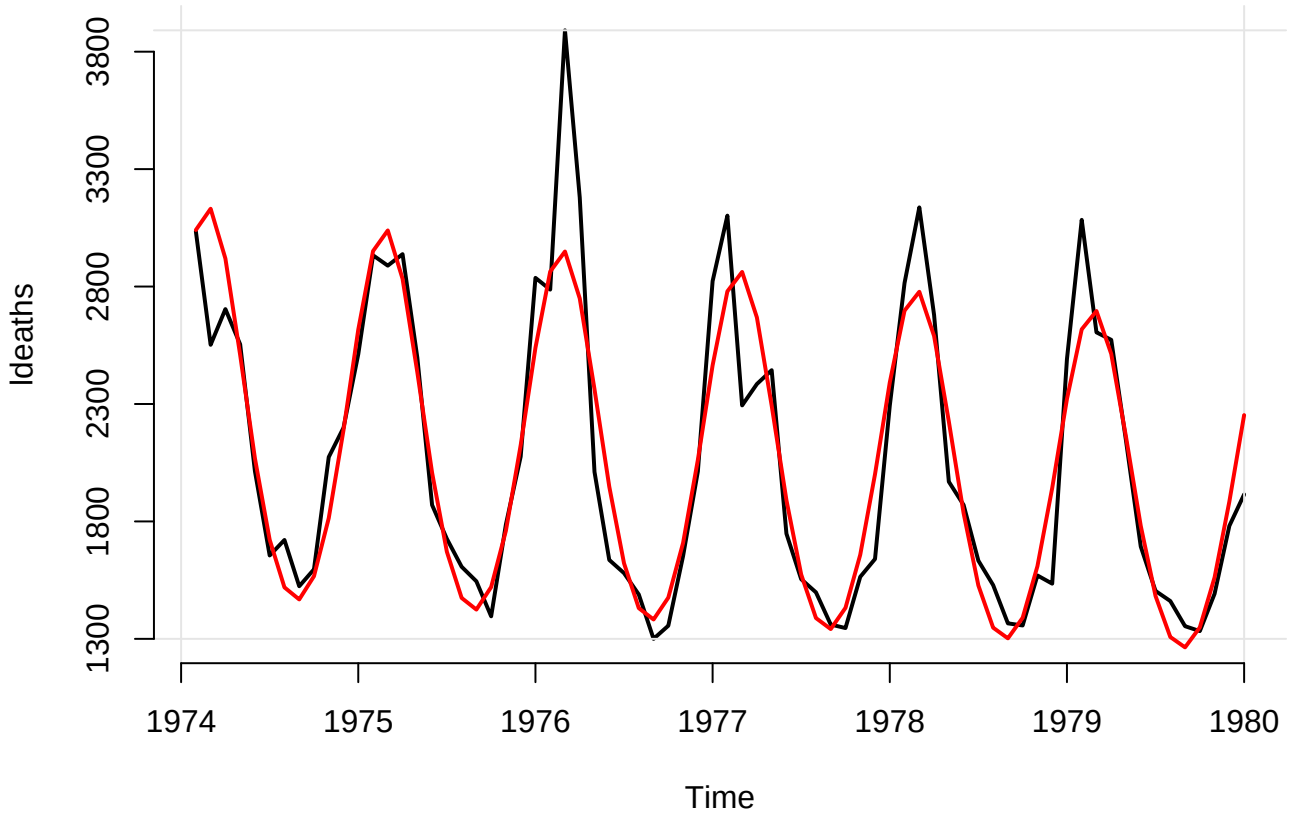

Figure 8: Raw data time series, in black, with the predicted time-series overlaid, in red.

□

(c)

Is the model an adequate fit?

Solution:

In Figure 9 (next page) we see heteroskedasticity in the relation residuals vs fitted, we see that the residues normality isn't attended in the tails, we see a slight increase in the scale-location relation when the predicted values increase, and we see that some few points have a high leverage. In general the model doesn't present an very adequate fit.

In this model we aren't considering the correlation between observations along time, maybe this can justify the non-adequate fit. Others linear predictors can also be considered.

```
# <r code> ============================================================================ #
par(mfrow = c(2, 2))                                          # graphical definitions
plot(fit)                              # plotting the graphs for analysis of residues
# </r code> ============================================================================ #
```

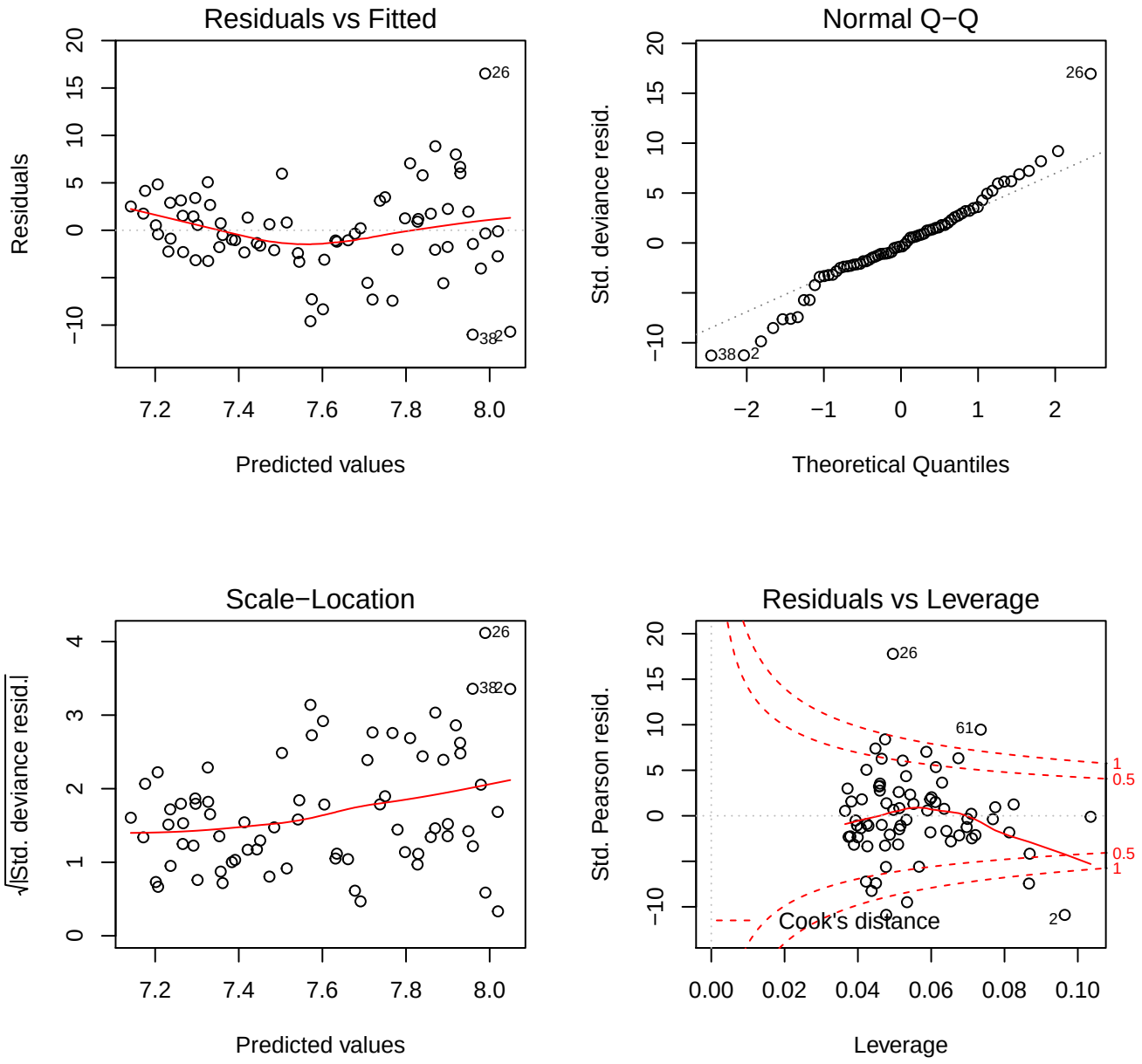

Figure 9: Graphical analysis of residues, goodness-of-fit.

■