Supplement to notes on linear regression and generalizations

Erik Strumbelj

Example 1: Effect of correlated input variables

We'll use a toy dataset with 2 highly correlated features and a target variable that is a linear combinaton of the two features and some random noise:

```
set.seed(0)
n <- 50
x1 <- rnorm(n)
x2 <- x1 + rnorm(n, 0, 0.2)
y  <- 2 * x1 + x2 - 1 + rnorm(n)
print(cor(x1, x2))
```

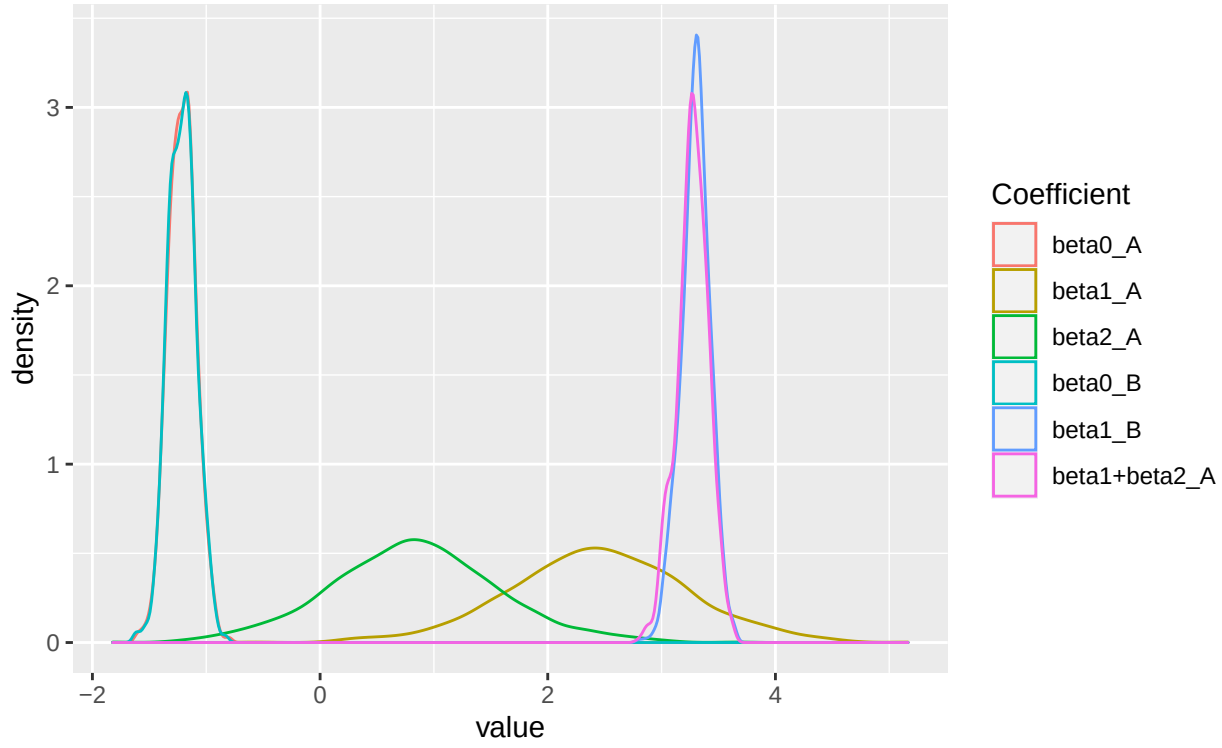
```
## [1] 0.9843066
```

To check the stability of estimated coefficients, we will bootstrap a subset of the dataset and fit a linear regression model using both features (A) and a linear regression model using only the first feature (A). We will repeat this 1000 times and record the coefficients:

```
all_res <- NULL
for (i in 1:1000) {
  idx <- sample(1:n, n, rep = T)
  tmp1 <- (lm(y[idx] ~ x1[idx] + x2[idx]))$coefficients
  tmp2 <- (lm(y[idx] ~ x1[idx] ))$coefficients
  all_res <- rbind(all_res, c(tmp1, tmp2))
}
all_res <- cbind(all_res, all_res[,2] + all_res[,3])
colnames(all_res) <- c("beta0_A", "beta1_A", "beta2_A", "beta0_B", "beta1_B", "beta1+beta2_A")
```

Now we can visualize the distributions:

```
library(ggplot2)
library(reshape2)
df <- melt(all_res)
names(df) <- c("ID", "Coefficient", "value")
ggplot(df, aes(x = value, colour = Coefficient)) + geom_density()
```



The results show that the intercept is relatively stable in both models. When only x1 is used (model B) the coefficient for x1 is also relatively stable. When both x1 and x2 are used (model A), both input variables' coefficients are very unstable. However, the sum of the coefficients is again stable and similar to the coefficient for x1 in model B. Therefore, models A and B would give very similar predictions. To summarize, when we fit a linear regression model using correlated inputs, we must interpret the coefficients with care, because even a slightly different training data set would lead to substantially different coefficients. However, even highly correlated input variables do not affect the quality of the model's predictions.

Example 2: Leverage and influential points

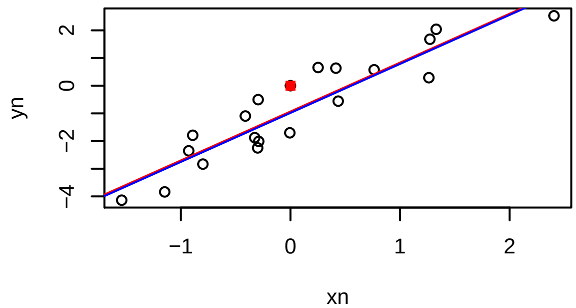
Below is an illustration of the role of leverage and residuals on how influential a point is in linear regression. We use a simple toy dataset with one input variable and 19 points. The blue line is the linear model fit on these 19 points.

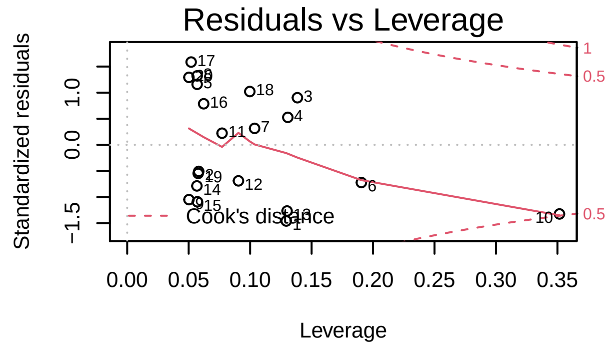
We then add four different points (each separately) and fit the model again. The new fit on these 20 points is shown in red and the larger the difference between the red and blue lines, the larger the effect is of the new point on the model:

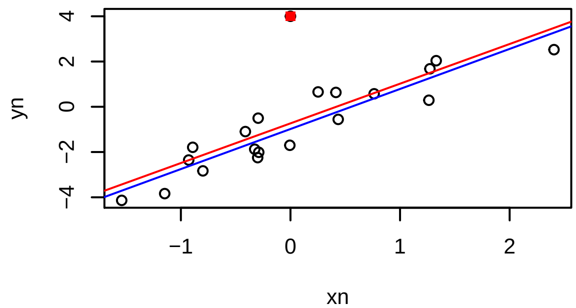
- The first point does not stand out in any way - it has both small leverage and a small residual. Its effect on the model is not noticeable.
- The second point has a large residual but very small leverage. Its effect on the model is barely noticeable.
- The third point has large leverage but a very small residual. Again, the point's effect on the model is barely noticeable.
- The fourth and final point has both a large residual and large leverage. It has a very strong effect on the model.

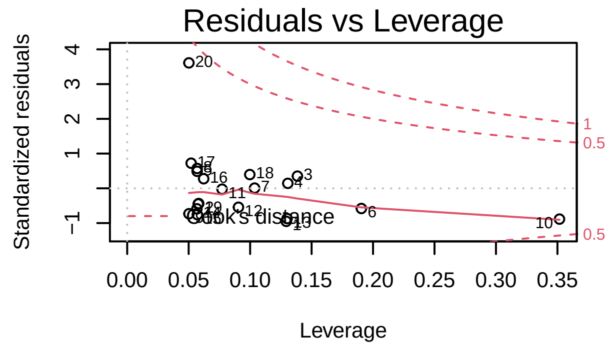
The plots on the right hand side show the leverage and standardized residuals and help us identify points like point 20 in the fourth example. Its Cook's distance substantially exceeds the typically used threshold of 1.

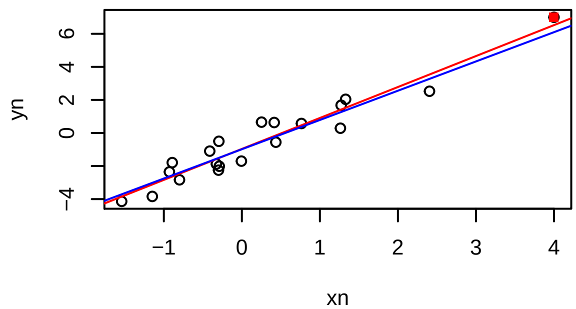
```r
set.seed(0)
n <- 19
x <- rnorm(n)
y  <- 2 * x - 1 + rnorm(n)


df <- data.frame(px = c(0, 0, 4, 4), py = c(0, 4, 7, 0))
par(mfrow=c(4, 2))
for (i in 1:nrow(df)) {
  px <- df$px[i]
  py <- df$py[i]

  xn <- c(x, px)
  yn <- c(y, py)
  plot(xn, yn)
  points(px, py, col = "red", pch = 15)
  abline(lm(yn ~ xn), col = "red")
  abline(lm(y ~ x), col = "blue")
  plot(lm(yn ~ xn), which = 5, id.n = 20)

}
```

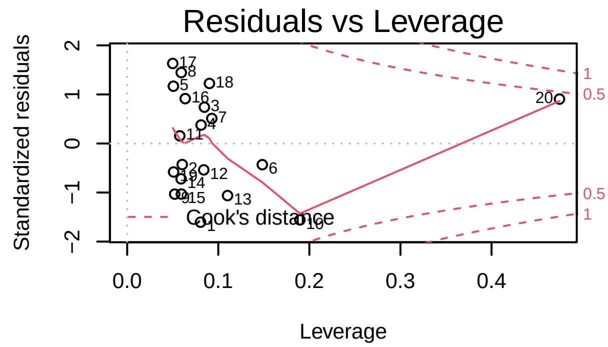

yn
xn

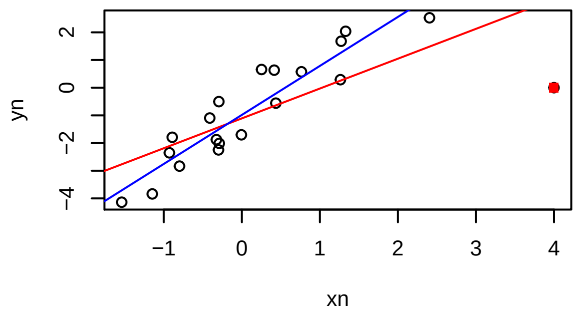

Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance

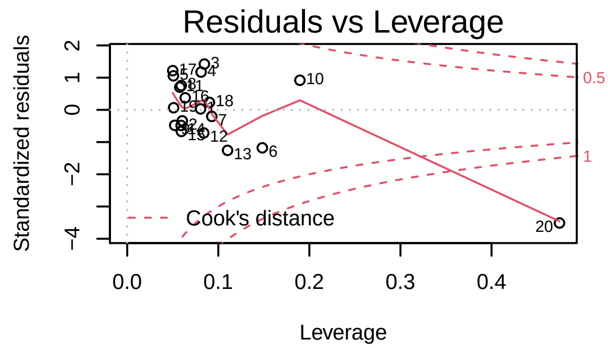

yn
xn


Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance


yn
xn


Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance

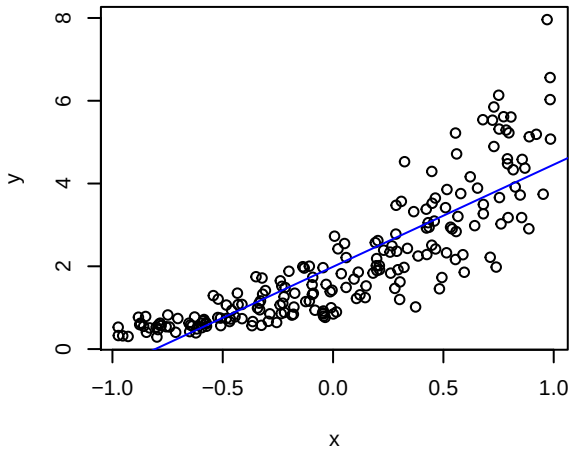

yn
xn

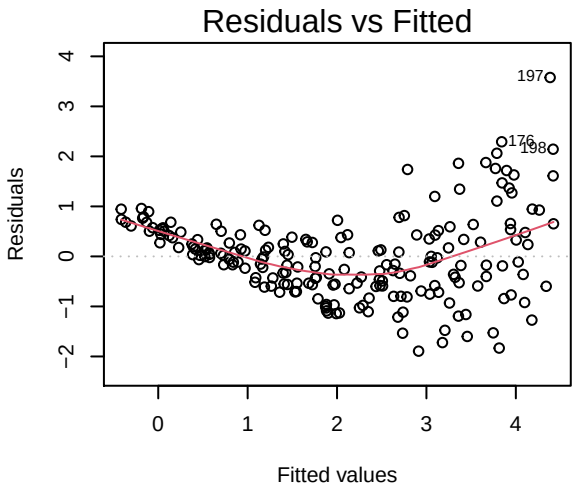

Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance

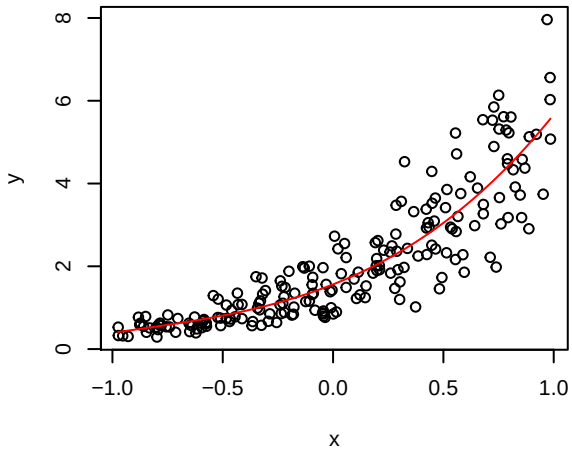
Example 3: Gamma GLM with log link function

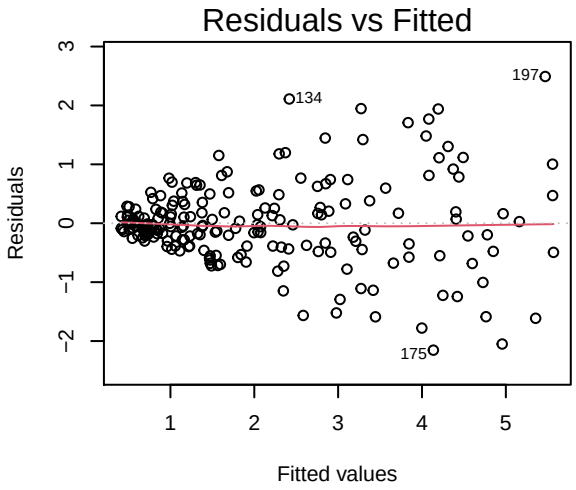
On a toy dataset with heteroskedastic errors we compare linear regression (blue), polynomial regression (red), Gamma GLM with log link function from glm() (orange), and our own implementation of the GLM using standard optimization (green).

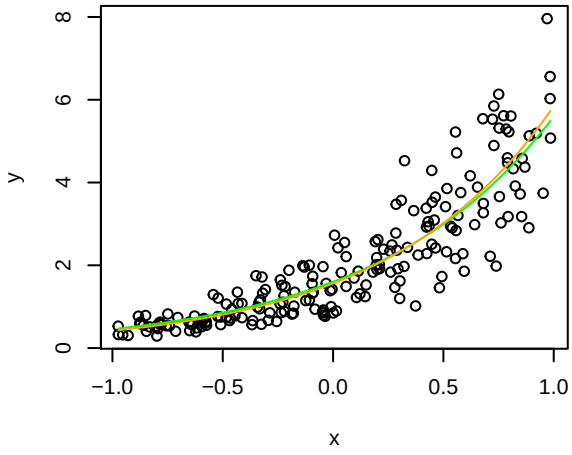
```r
# toy dataset
set.seed(0)
n <- 200
x <- runif(n, -1, 1)
x <- x[order(x)] # to simplify plotting
a <- 0.45
b <- 1.30
y_0 <- exp(a + b * x)
shape <- 11
y <- rgamma(n, rate = shape / y_0, shape = shape)

# fit linear regression
par(mfrow = c(3, 2))
plot(x, y)
abline(lm(y~x),col="blue")
plot(lm(y~x), which = 1)

# fit linear regression with 3rd deg. polynomial
plot(x, y)
lines(x, predict(lm(y~poly(x,3,raw=T))), col = "red")
plot(lm(y~poly(x,3,raw=T)), which = 1)

# fit gamma glm regression with log link function
model <- glm(y ~ x, family = Gamma(link = "log"))

# "do-it-yourself" Gamma GLM
minus_log_lik <- function(x, X, y) {
  beta <- x[3]
  mu   <- exp(X %*% x[1:2])
  -sum(dgamma(y, mu * beta, beta, log = T))
}

X <- cbind(x, 1)
res <- optim(c(0, 0, 0.5), fn = minus_log_lik,
             lower = c(-Inf, -Inf, 0.00001),
             method = "L-BFGS-B", X = X, y = y)
mu   <- exp(X %*% res$par[1:2])
plot(x, y)
lines(x, mu, col = "green")
lines(x, predict(model, type = "response"), col = "orange")
plot(model, which = 1)
```

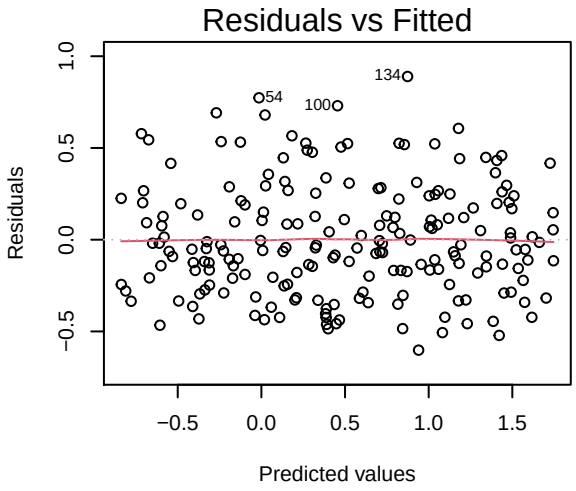

y
x
0
2
4
6
8
−1.0
−0.5
0.0
0.5
1.0


Residuals vs Fitted
Residuals
Fitted values
197
176
198
−2
−1
0
1
2
3
4


y
x
0
2
4
6
8
−1.0
−0.5
0.0
0.5
1.0


Residuals vs Fitted
Residuals
Fitted values
197
134
175
−2
−1
0
1
2
3
1
2
3
4
5


y
x
0
2
4
6
8
−1.0
−0.5
0.0
0.5
1.0


Residuals vs Fitted
Residuals
Predicted values
134
54
100
−0.5
0.0
0.5
1.0
−0.5
0.0
0.5
1.0
1.5

We can see that linear regression is not appropriate. Polynomial regression provides a better fit for the mean, but we do not correctly understand the heteroskedasticit. The Gamma GLM works best and our implementation gives practically the same result as the one from glm().