hw2

mitch borgert

February 17, 2018

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Warning: package 'purrr' was built under R version 3.4.3
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 3.4.3
```

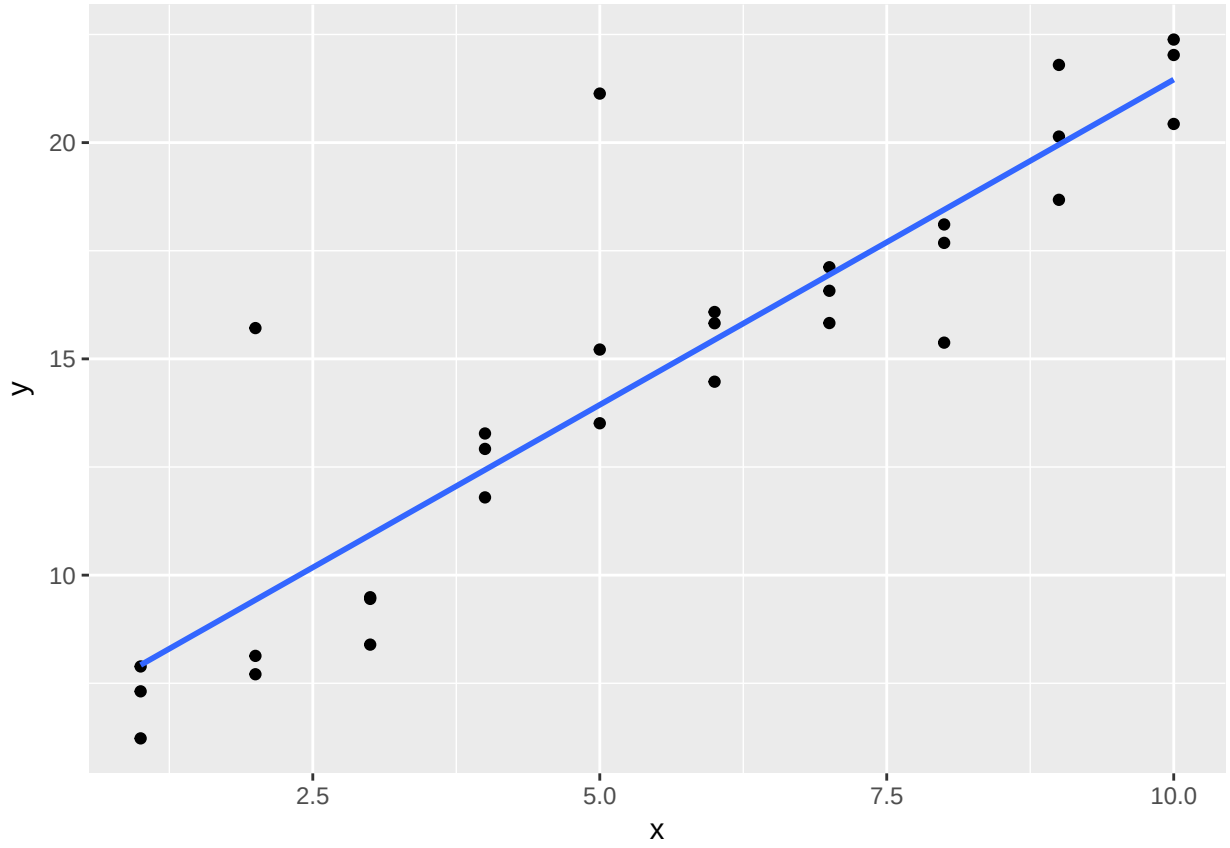
```
library(ggplot2)
library(tidyr)
```

23.2.1 1. One downside of the linear model is that it is sensitive to unusual values because the distance incorporates a squared term. Fit a linear model to the simulated data below, and visualise the results. Rerun a few times to generate different simulated datasets. What do you notice about the model?

```
sim1a <- tibble(
  x = rep(1:10, each = 3),
  y = x * 1.5 + 6 + rt(length(x), df = 2)
)

model1 = lm(x~y,data=sim1a)

ggplot(sim1a, aes(x = x,y = y)) + geom_point() + geom_smooth(method='lm',formula=y~x,se = FALSE)
```



One outlier can shift the line away from where it really should be.

23.2.1 2. One way to make linear models more robust is to use a different distance measure. For example, instead of root-mean-squared distance, you could use mean-absolute distance:

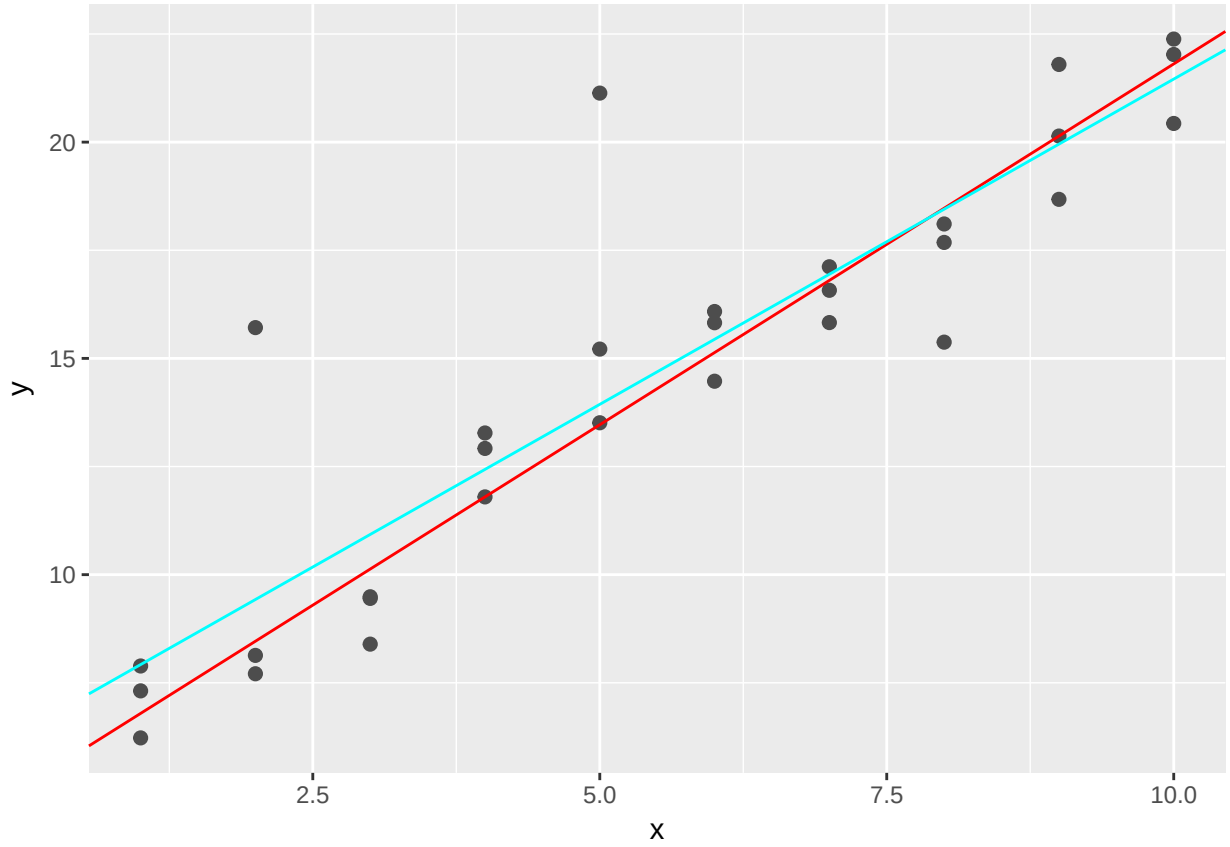
```
model1 <- function(a, data) {
  a[1] + data$x * a[2]
}

measure_distance <- function(mod, data) {
  diff <- data$y - model1(mod, data)
  mean(abs(diff))
}

measure_distance2 <- function(mod, data) {
  diff <- data$y - model1(mod, data)
  sqrt(mean(diff ^ 2))
}

best1 <- optim(c(0,0), measure_distance, data = sim1a)
best2 <- optim(c(0,0), measure_distance2, data = sim1a)
```

```
ggplot(sim1a, aes(x, y)) + geom_point(size = 2, colour = "grey30") + geom_abline(intercept = best1$par[
```



The model that uses the mean of the absolute difference is less effected by outliers than the model that squares the differences.

23.3.3 1.Instead of using lm() to fit a straight line, you can use loess() to fit a smooth curve. Repeat the process of model fitting, grid generation, predictions, and visualisation on sim1 using loess() instead of lm(). How does the result compare to geom_smooth()?

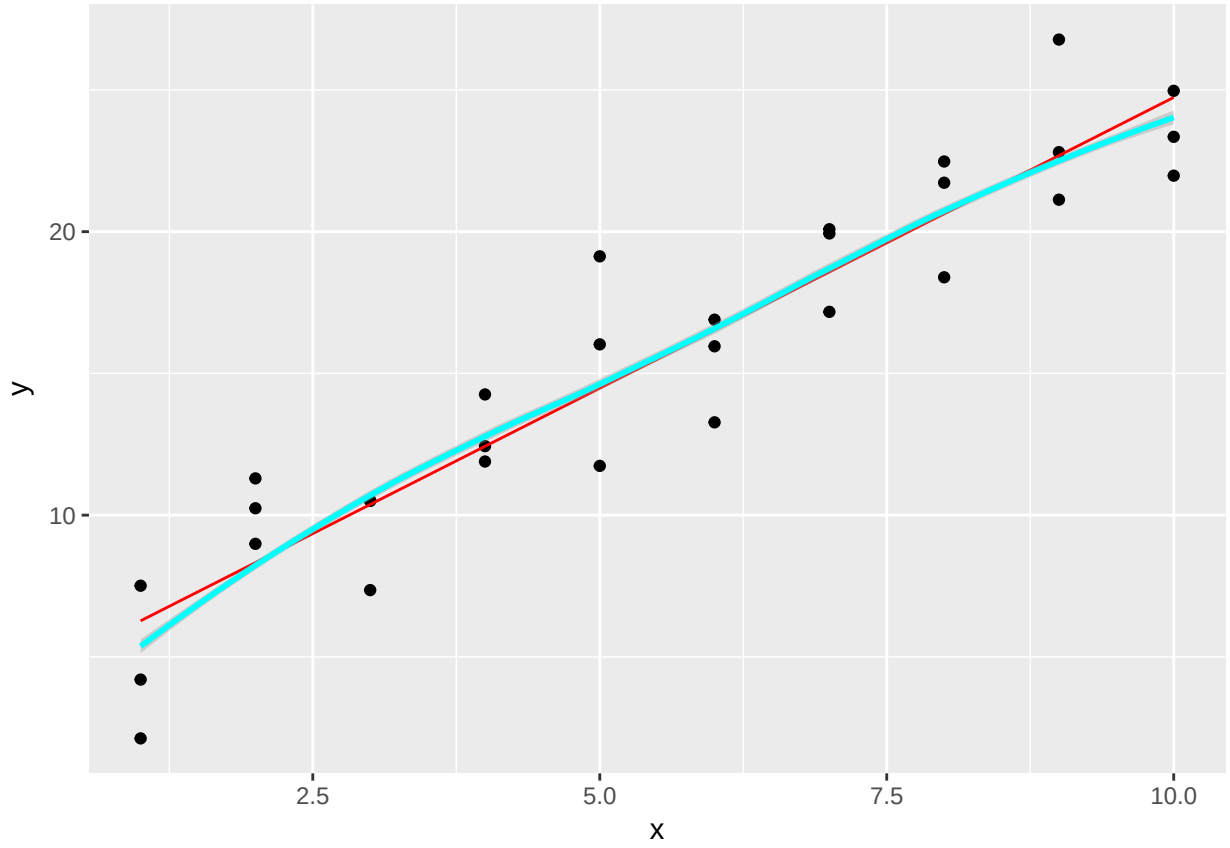
```
fit1 <- lm(y~x, data = sim1)
fit2 <- loess(y~x, data = sim1,degree = 2)

grid <- sim1 %>% data_grid(x)
grid1 <- grid %>% add_predictions(fit1)
sim1_1 <- sim1 %>% add_residuals(fit1)

grid2 <- grid %>% add_predictions(fit2)
sim1_2 <- sim1 %>% add_residuals(fit2)

ggplot(sim1,aes(x=x))+geom_point(aes(y=y))+geom_line(data = grid1, aes(y = pred), color = 'red')+geom_s
```

```
## `geom_smooth()` using method = 'loess'
```



The loess line is close to the normal linear line, but it is pulled more toward outliers. If there were extreme outliers in this data the loess line would be less accurate.

23.3.3. 3.

```
?geom_ref_line
```

```
## starting httpd help server ... done
```

geom_ref_line is from the modelr package. It adds a reference line to the graph and is usful for visually looking at the trend in the residuals.

23.3.3. 4. Why might you want to look at a frequency polygon of absolute residuals? What are the pros and cons compared to looking at the raw residuals?

You want to check the absolute residuals because it helps to see the overall quality of the prediction but it won't give you the hint about the distribution of residuals with respect to 0.

23.4.5 1.

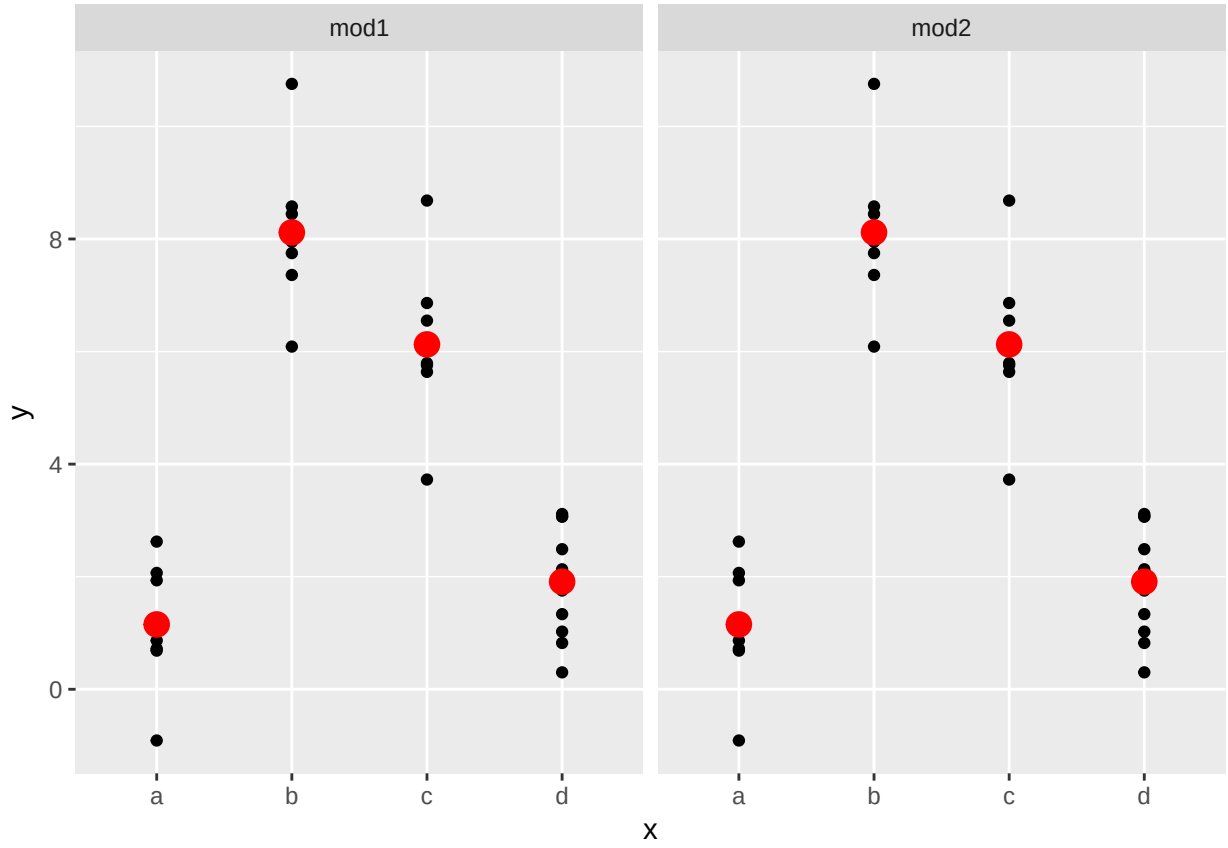
```
mod1 <- lm(y~x - 1, data = sim2)
summary(mod1)
```

```
##
## Call:
## lm(formula = y ~ x - 1, data = sim2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.40131 -0.43996 -0.05776  0.49066  2.63938
##
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## xa   1.1522     0.3475   3.316  0.00209 **
## xb   8.1160     0.3475  23.356  < 2e-16 ***
## xc   6.1272     0.3475  17.633  < 2e-16 ***
## xd   1.9110     0.3475   5.499 3.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.099 on 36 degrees of freedom
## Multiple R-squared:  0.9614, Adjusted R-squared:  0.9572
## F-statistic: 224.4 on 4 and 36 DF,  p-value: < 2.2e-16
```

```
mod2 <- lm(y~x, data = sim2)
summary(mod2)
```

```
##
## Call:
## lm(formula = y ~ x, data = sim2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.40131 -0.43996 -0.05776  0.49066  2.63938
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1522     0.3475   3.316  0.00209 **
## xb            6.9639     0.4914  14.171 2.68e-16 ***
## xc            4.9750     0.4914  10.124 4.47e-12 ***
## xd            0.7588     0.4914   1.544  0.13131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.099 on 36 degrees of freedom
## Multiple R-squared:  0.8852, Adjusted R-squared:  0.8756
## F-statistic: 92.52 on 3 and 36 DF,  p-value: < 2.2e-16
```

```
grid1 <- sim2 %>% data_grid(x) %>% gather_predictions(mod1,mod2)
sim2 %>%ggplot(aes(x))+geom_point(aes(y=y))+geom_point(data = grid1, aes(y = pred),color = "red",size =
```



The equations for the models change but the prediction stays the same.

23.4.5. 3.

```
sim3 = sim3
sim3 <- sim3 %>%mutate(present = 1)%>%spread(x2,present,fill=0) #splits a catagorical column into a bun
mod1 <- lm(y ~ x1 + a + b + c + d, data = sim3)
summary(mod1)
```

```
##
## Call:
## lm(formula = y ~ x1 + a + b + c + d, data = sim3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4674 -0.8524 -0.0729  0.7886  4.3005
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.23125    0.38738  10.923  < 2e-16 ***
## x1          -0.19674    0.04871  -4.039 9.72e-05 ***
## a           -2.35959    0.39571  -5.963 2.79e-08 ***
## b            0.52822    0.39571   1.335    0.185
## c            2.44615    0.39571   6.182 9.93e-09 ***
```

```
## d                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.533 on 115 degrees of freedom
## Multiple R-squared:  0.5911, Adjusted R-squared:  0.5768
## F-statistic: 41.55 on 4 and 115 DF,  p-value: < 2.2e-16
```

```
mod2 <- lm(y ~ x1*a*b*c*d, data = sim3)
summary(mod2)
```

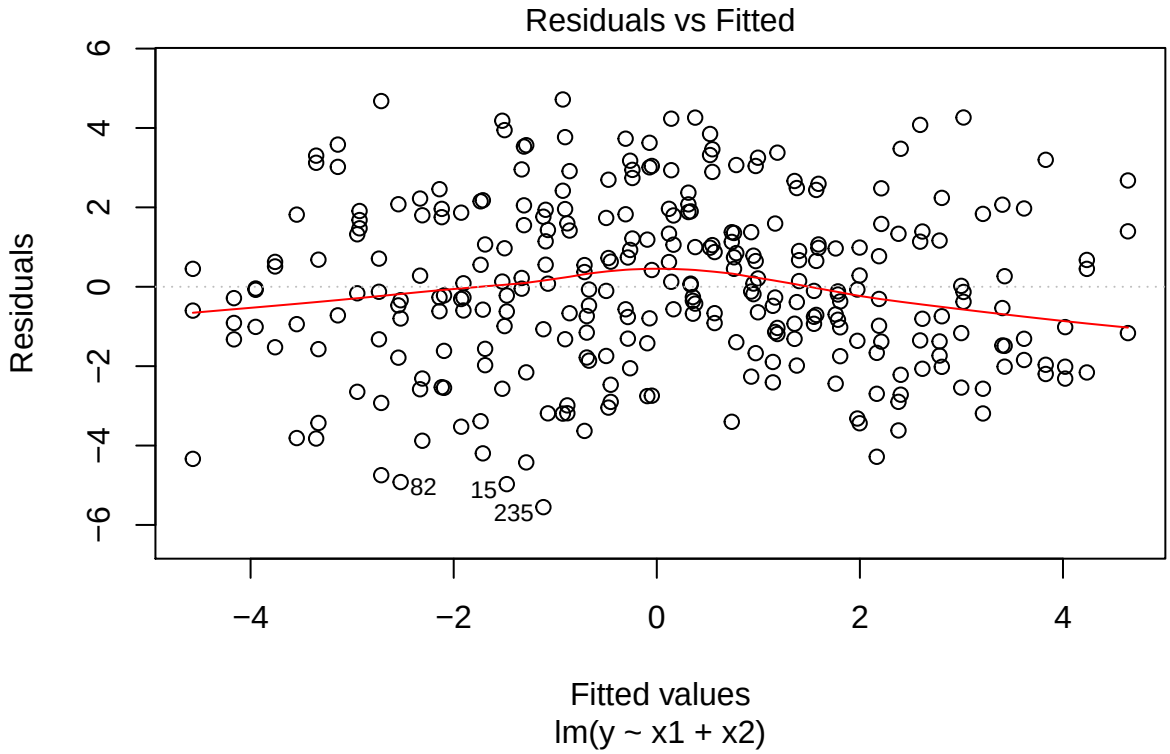
```
##
## Call:
## lm(formula = y ~ x1 * a * b * c * d, data = sim3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.87634 -0.67655  0.04837  0.69963  2.58607
##
## Coefficients: (24 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.13579    0.40400   5.287 6.22e-07 ***
## x1           0.18425    0.06511   2.830  0.00552 **
## a           -0.83455    0.57134  -1.461  0.14690
## b            6.23483    0.57134  10.913  < 2e-16 ***
## c            3.59634    0.57134   6.295 6.18e-09 ***
## d                 NA         NA      NA       NA
## x1:a        -0.27728    0.09208  -3.011  0.00322 **
## x1:b        -1.03756    0.09208 -11.268  < 2e-16 ***
## a:b               NA         NA      NA       NA
## x1:c        -0.20913    0.09208  -2.271  0.02505 *
## a:c               NA         NA      NA       NA
## b:c               NA         NA      NA       NA
## x1:d              NA         NA      NA       NA
## a:d               NA         NA      NA       NA
## b:d               NA         NA      NA       NA
## c:d               NA         NA      NA       NA
## x1:a:b            NA         NA      NA       NA
## x1:a:c            NA         NA      NA       NA
## x1:b:c            NA         NA      NA       NA
## a:b:c             NA         NA      NA       NA
## x1:a:d            NA         NA      NA       NA
## x1:b:d            NA         NA      NA       NA
## a:b:d             NA         NA      NA       NA
## x1:c:d            NA         NA      NA       NA
## a:c:d             NA         NA      NA       NA
## b:c:d             NA         NA      NA       NA
## x1:a:b:c          NA         NA      NA       NA
## x1:a:b:d          NA         NA      NA       NA
## x1:a:c:d          NA         NA      NA       NA
## x1:b:c:d          NA         NA      NA       NA
## a:b:c:d           NA         NA      NA       NA
## x1:a:b:c:d        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

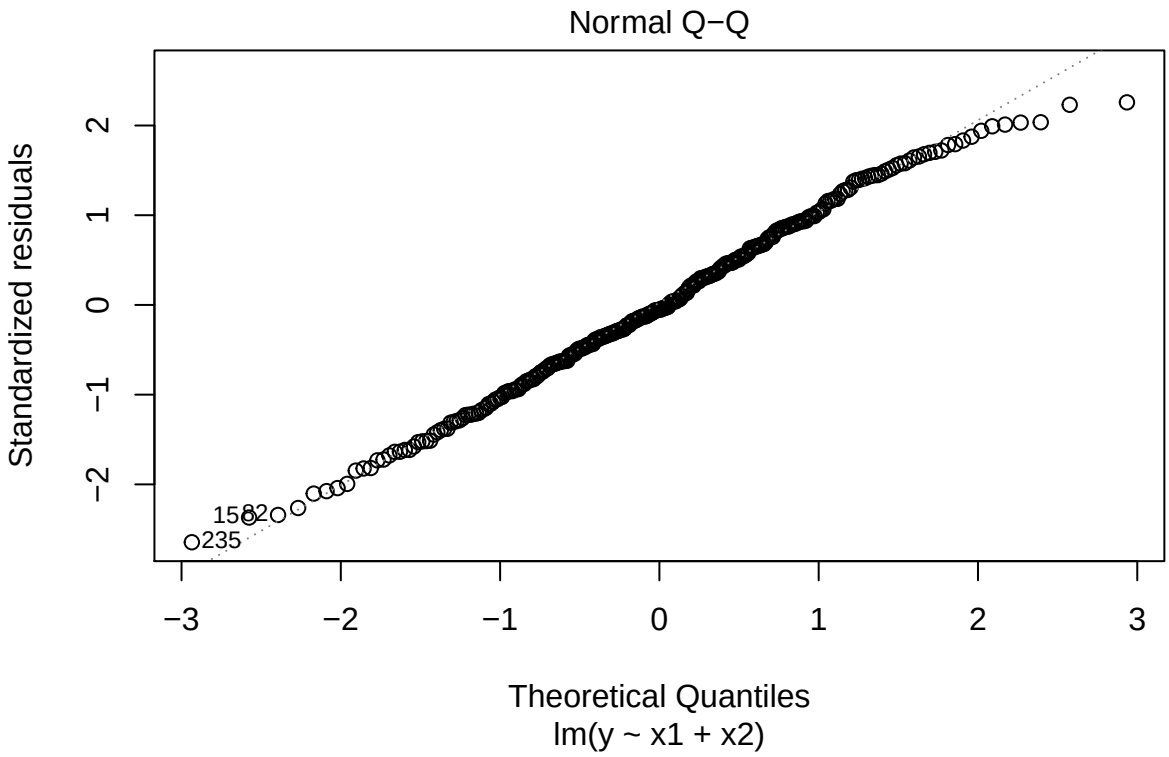
```
##
## Residual standard error: 1.024 on 112 degrees of freedom
## Multiple R-squared:  0.8221, Adjusted R-squared:  0.811
## F-statistic: 73.93 on 7 and 112 DF,  p-value: < 2.2e-16
```

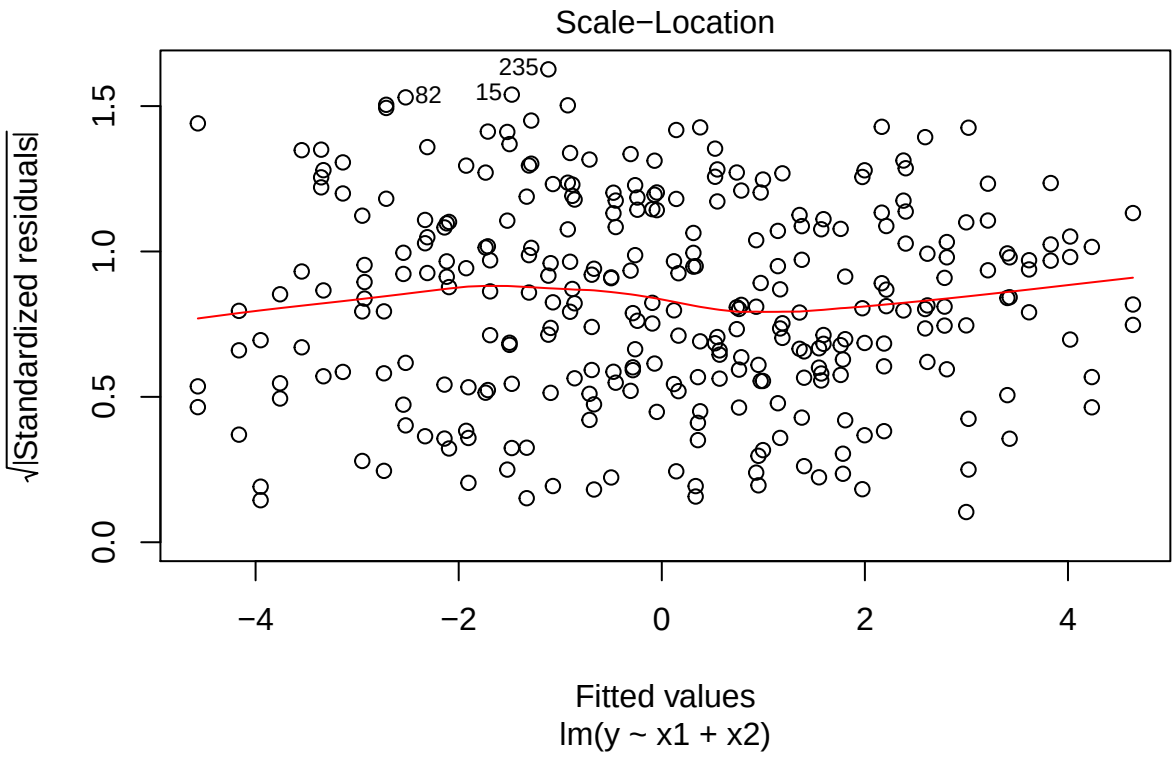
Is this what this problem wants?

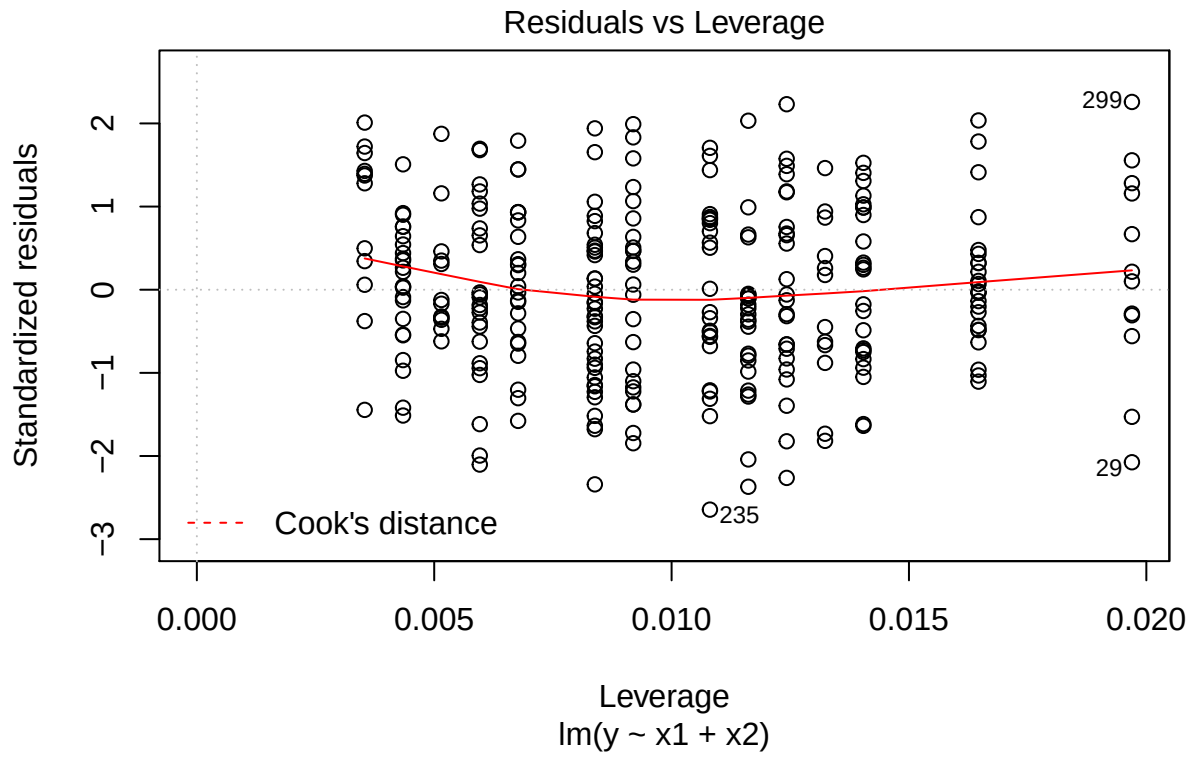
23.4.5. 4.

```
mod1 <- lm(y ~ x1 + x2, data = sim4)
mod2 <- lm(y ~ x1 * x2, data = sim4)
par(plot(mod1), mfrow = c(2,2))
```

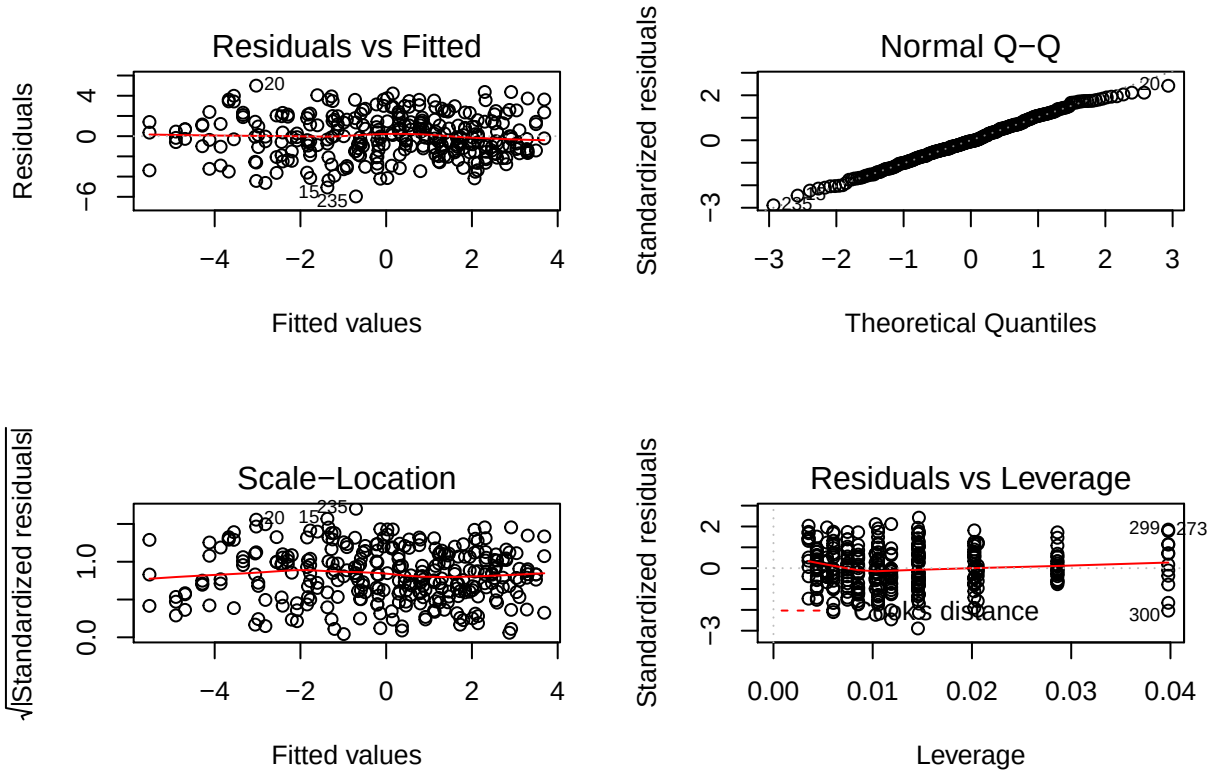



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x1 + x2)
−3
−2
−1
0
1
2
3
−2
−1
0
1
2
235


Scale−Location
235
82
15
√|Standardized residuals|
0.0
0.5
1.0
1.5
−4
−2
0
2
4
Fitted values
lm(y ~ x1 + x2)




```
par(plot(mod2), mfrow = c(2,2))
```



It looks like model 2 is better, the residuals care closer to normal than on model 1.