Statistical Modelling - Series 2

Konstantinos Papadakis (DSML 03400149)

5 January 2022

Part A

We start by loading the appropriate libraries and the dataset.

```
library("car")
library("corrplot")
library("olsrr")
library("glmnet")
library("ggplot2")

vehicles_full <- read.table("./data/vehicles.txt", header = TRUE)

# We can convert the "vs" and "am" to factors, but since they only have two levels,
# with value 0 and 1, we can keep them as is.
# factor_columns <- c("vs", "am")
# df_full[factor_columns] <- lapply(df_full[factor_columns], as.factor)

# Exclude the columnn with the vehicle names.
vehicles <- vehicles_full[-1]
```

We then fit a linear model using all the variables. No variable is significant even though R^2 is decent. This leads us to believe that some of them might be redundant due to multicollinearity.

```
mod <- lm(mpg ~ ., vehicles)
summary(mod)
```

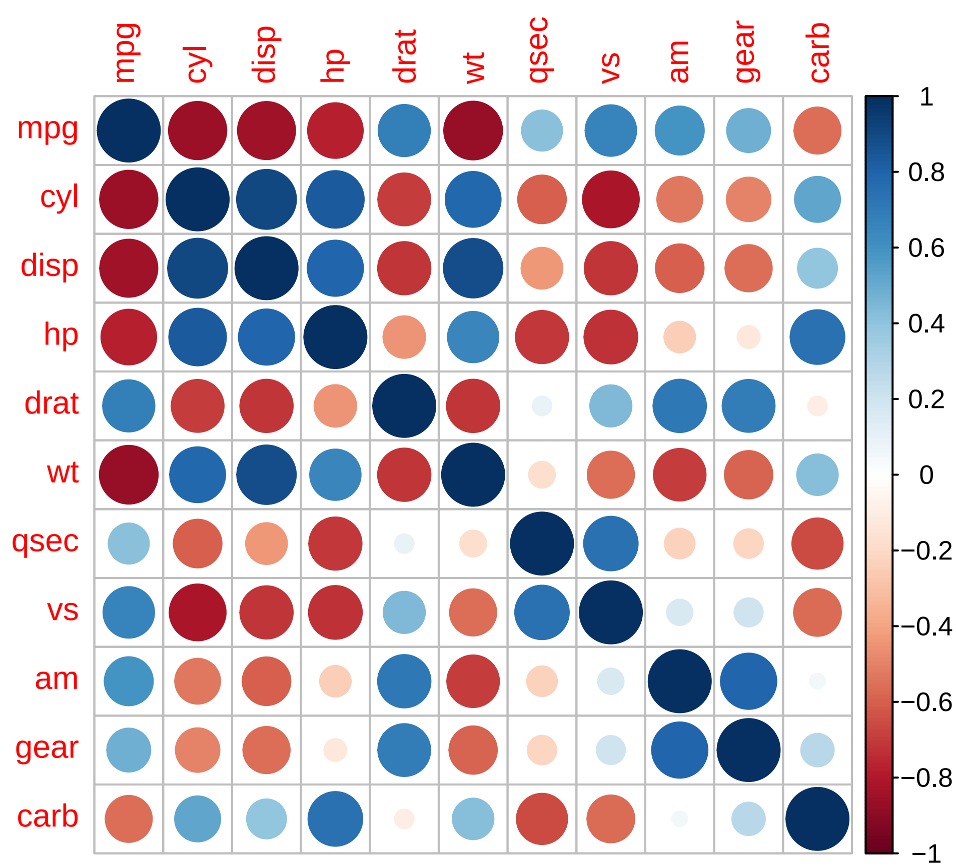
```
##
## Call:
## lm(formula = mpg ~ ., data = vehicles)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4506 -1.6044 -0.1196  1.2193  4.6271
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.30337   18.71788   0.657   0.5181
## cyl         -0.11144    1.04502  -0.107   0.9161
## disp         0.01334    0.01786   0.747   0.4635
## hp          -0.02148    0.02177  -0.987   0.3350
```

```
## drat          0.78711    1.63537   0.481   0.6353
## wt           -3.71530    1.89441  -1.961   0.0633 .
## qsec          0.82104    0.73084   1.123   0.2739
## vs            0.31776    2.10451   0.151   0.8814
## am            2.52023    2.05665   1.225   0.2340
## gear          0.65541    1.49326   0.439   0.6652
## carb         -0.19942    0.82875  -0.241   0.8122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.65 on 21 degrees of freedom
## Multiple R-squared:  0.869,  Adjusted R-squared:  0.8066
## F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07
```

A.1

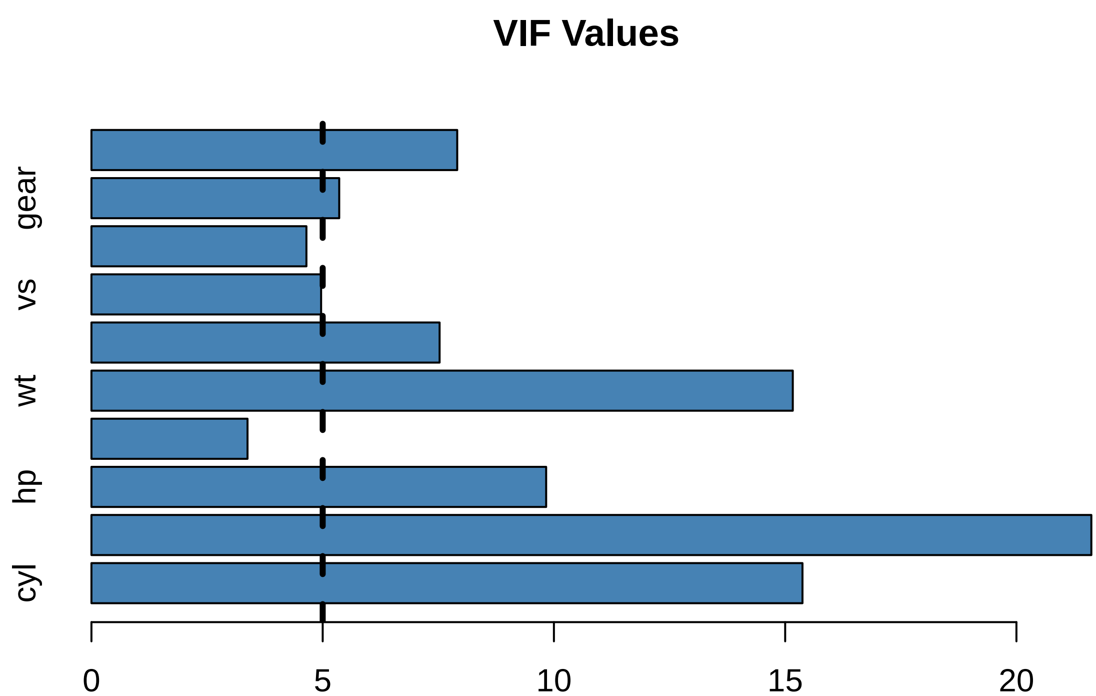
We see that there's large simple collinearity between variables

```
corr_matrix <- cor(vehicles)
corrplot(corr_matrix)
```



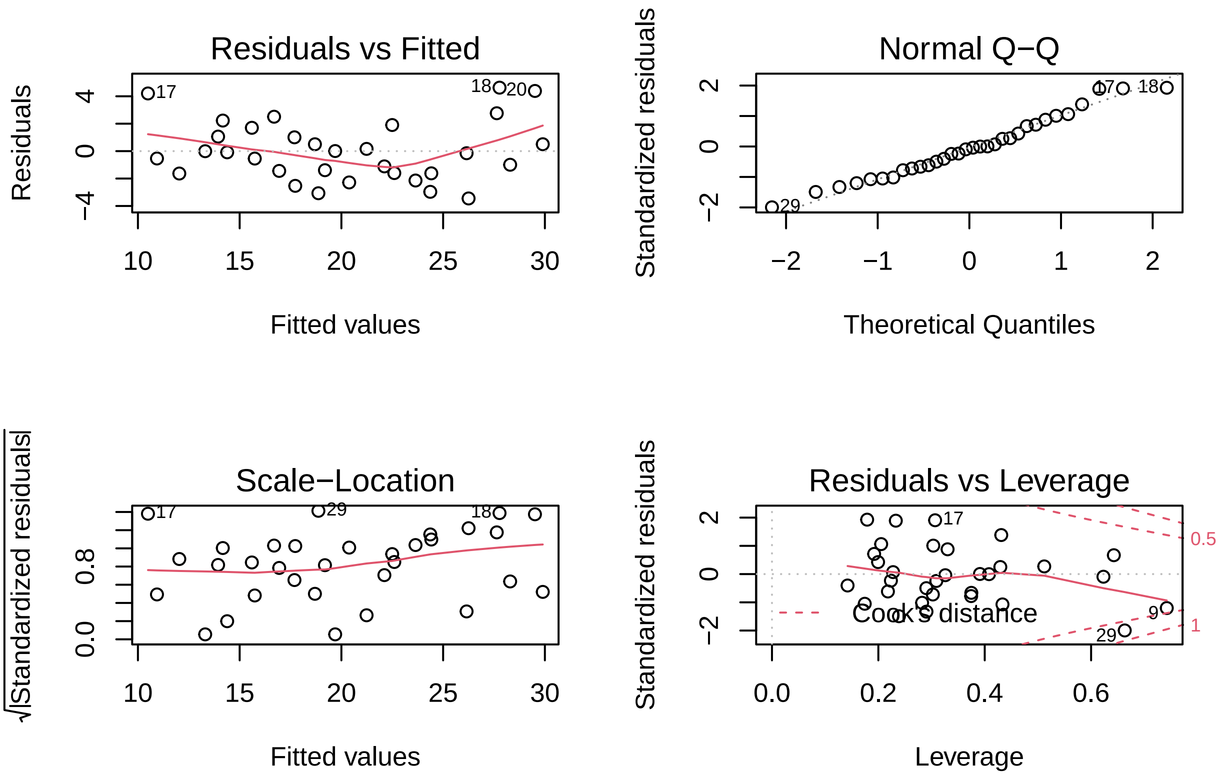
The variance inflation factor for almost all variables is above 5, which means that there's large collinearity among certain subsets of our variables.

```r
vif_values <- vif(mod)
barplot(vif_values, main = "VIF Values", horiz = TRUE, col = "steelblue")
abline(v = 5, lwd = 3, lty = 2)
```



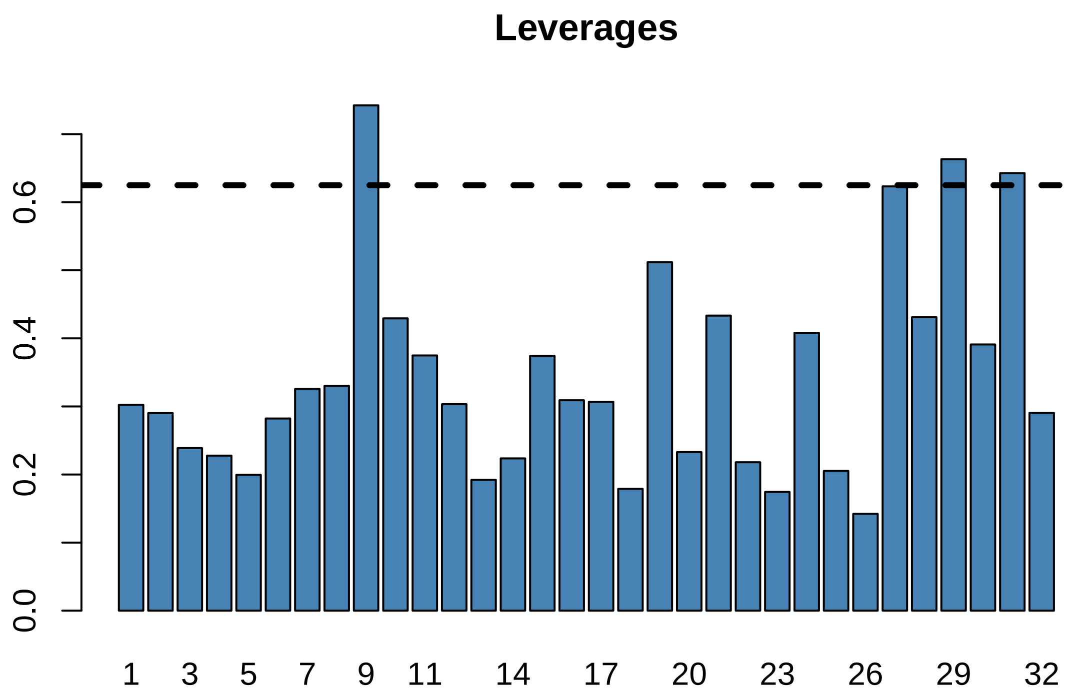
In the Residuals vs Fitted plot, we see that the line is not far from horizontal, and that the spread around it is roughly homogenous. Thus, homoscedacity seems to hold. The QQ plot doesn't look great, so we might need to transform the predicted variable. In the Residuals vs Leverage plot, we see that the two most influential samples are 29 and 9.

```r
par(mfrow = c(2,2))
plot(mod)
```



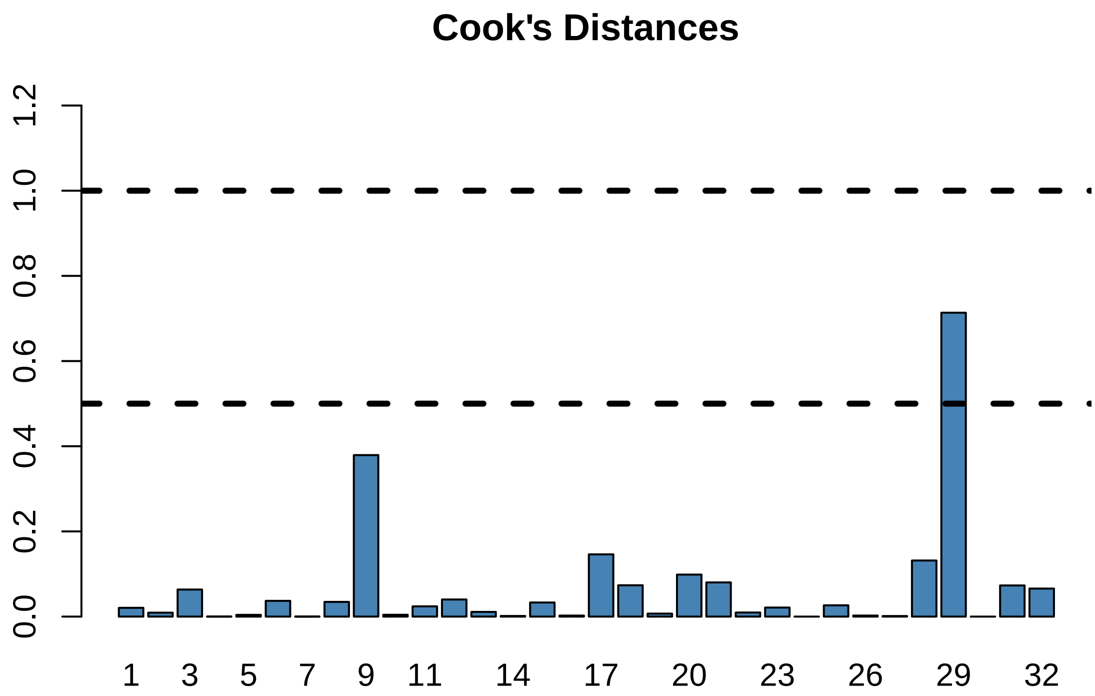
We notice that according to the $\frac{2p}{n}$ criterion for leverages, samples 9, 29 and 31 can be considered influential.

```
leverages <- hatvalues(mod)
influential_threshold <- 2 * (ncol(vehicles) - 1) / nrow(vehicles)
barplot(leverages, main = "Leverages", col = "steelblue")
abline(h = influential_threshold, lwd = 3, lty = 2)
```



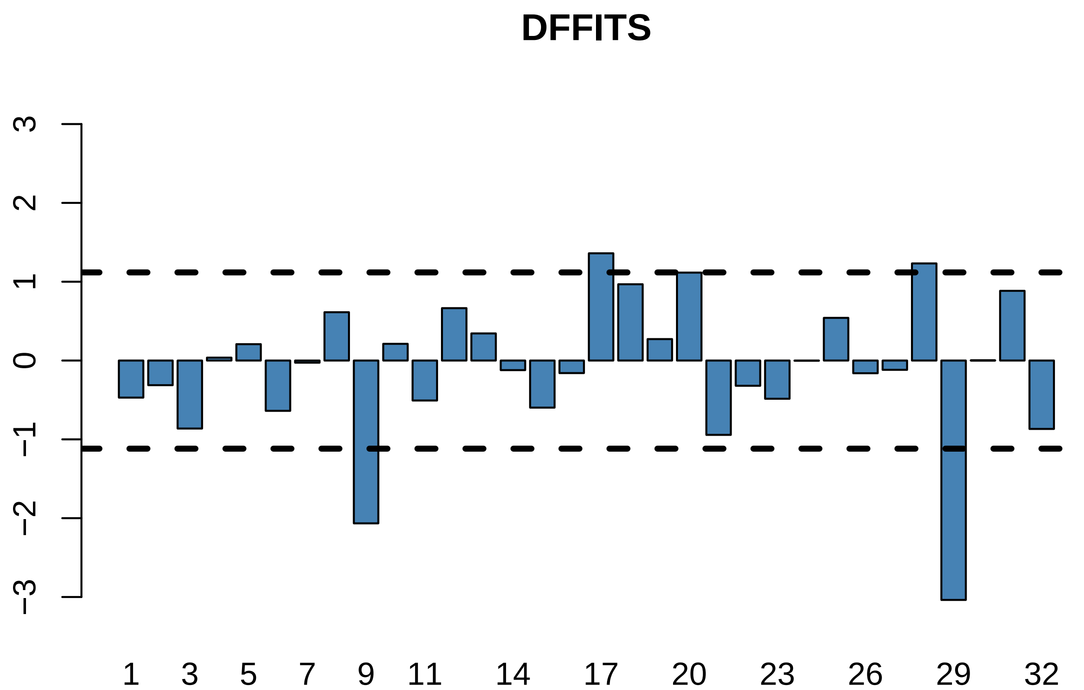
According to Cook's distance, sample 29 is relatively influential, although its corresponding value is not greater than 1.

```
cooks <- cooks.distance(mod)
cooks_thresshold <- c(0.5, 1)
barplot(cooks, main = "Cook's Distances", col = "steelblue", ylim = c(0, max(cooks, 1.2)))
abline(h = cooks_thresshold, lwd = 3, lty = 2)
```



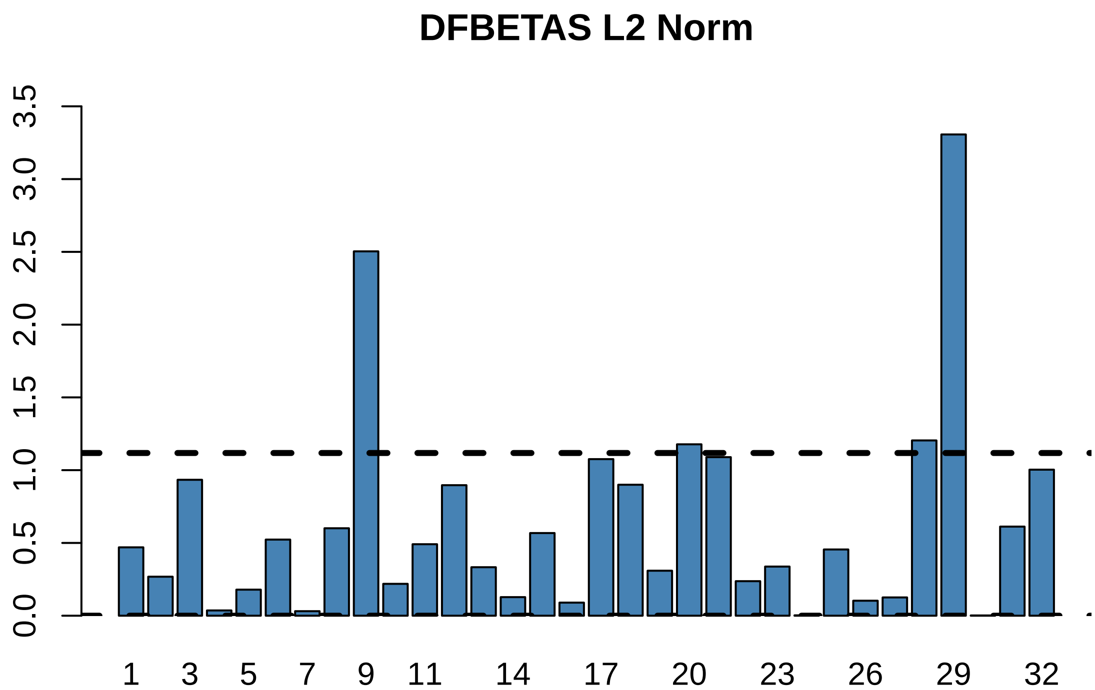
DFFITS also show us that the most influential points are 9 and 29

```
dffits_values <- dffits(mod)
fthresh <- 2 * sqrt((ncol(vehicles) - 1) / nrow(vehicles))
ylim <- max(fthresh, abs(dffits_values)) + 0.2
barplot(dffits_values, main = "DFFITS", col = "steelblue", ylim = c(-ylim, ylim))
abline(h = c(-fthresh, fthresh), lwd = 3, lty = 2)
```



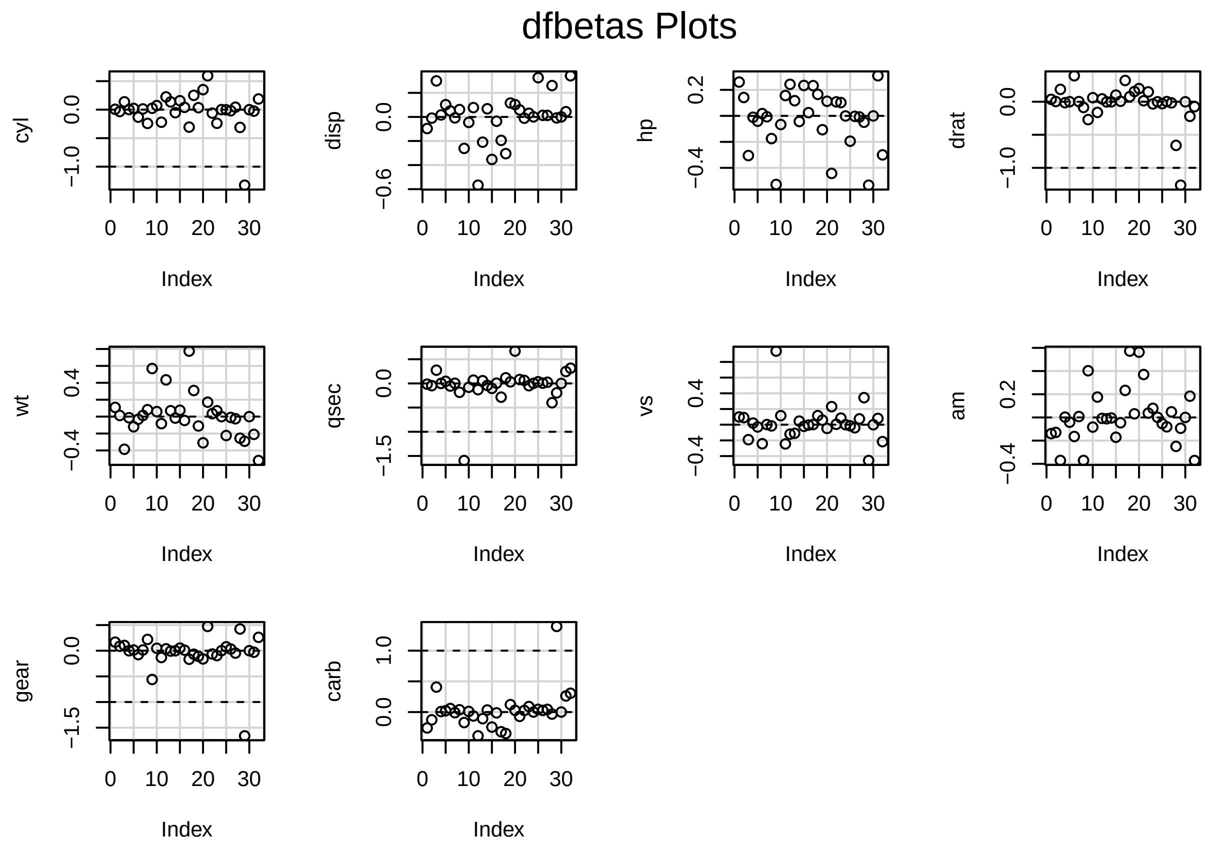
DFBETAS norms also show us that samples 9 and 29 are influential.

```
dfbetas_values <- dfbetas(mod)
dfbetas_norms <- apply(dfbetas_values, 1, function(v){sqrt(sum(v^2))})
bthresh <- 2 * sqrt((ncol(vehicles) - 1) / nrow(vehicles))
ylim <- max(bthresh, abs(dfbetas_norms))
barplot(dfbetas_norms, main = "DFBETAS L2 Norm", col = "steelblue",
        ylim = c(0, max(bthresh, dfbetas_norms)+0.2))
abline(h = c(0, bthresh), lwd = 3, lty = 2)
```



We can also view DFBETAS coordinate by coordinate

```
dfbetasPlots(mod, layout=c(3, 4))
```



A.2

We can use step search ...

```
attach(vehicles)
print("-----FORWARD-----")
mod_forward <- step(
   lm(mpg ~ 1),
   scope = mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb,
   direction = "forward",
   test = "F",
   trace = 100,
)
print("-----BACKWARD-----")
mod_backward <- step(
   lm(mpg ~ ., vehicles),
   scope = NULL,
   direction = "backward",
   test = "F",
   trace = 100,
)
print("-----BOTH-----")
mod_both<- step(
   lm(mpg ~ 1),
   scope = mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb,
```

```r
  direction = "both",
  test = "F",
  trace = 100,
)
detach(vehicles)
```

... but since our variables are only 10, and our samples only 32, we can simply try out all possible $2^{10} - 1 = 1023$ models and pick the best one according to some measure (e.g. AIC).

```r
steps_all <- ols_step_all_possible(mod)
steps_all <- steps_all[order(steps_all$aic), ]
 knitr::kable(steps_all[1:10, c(-1, -2)],
             caption = "Top 10 best results according to AIC", d = 3)
```

Table 1: Top 10 best results according to AIC

	predictors	rsquare	adjr	predrsq	cp	aic	sbic	sbc	msep	fpe	apc	hsp
157	wt qsec am	0.850	0.834	0.795	0.103	154.119	65.714	161.448	193.973	6.802	0.193	0.224
332	hp wt qsec am	0.858	0.837	0.802	0.790	154.327	67.230	163.122	190.464	6.855	0.195	0.228
375	wt qsec am carb	0.857	0.836	0.797	0.958	154.563	67.379	163.358	191.872	6.905	0.196	0.230
528	disp hp wt qsec am	0.864	0.838	0.798	1.846	154.974	69.307	165.234	189.879	7.008	0.199	0.236
65	cyl hp wt	0.843	0.826	0.796	1.147	155.477	66.696	162.805	202.378	7.096	0.202	0.234
297	disp wt qsec am	0.853	0.831	0.786	1.636	155.494	67.970	164.288	197.536	7.109	0.202	0.236
81	cyl wt carb	0.842	0.826	0.797	1.258	155.617	66.798	162.946	203.270	7.128	0.203	0.235
621	drat wt qsec am carb	0.861	0.834	0.789	2.318	155.658	69.675	165.918	193.981	7.159	0.203	0.241
606	hp wt qsec am carb	0.861	0.834	0.794	2.331	155.676	69.685	165.936	194.089	7.163	0.204	0.241
241	cyl wt qsec am	0.851	0.829	0.779	1.889	155.834	68.187	164.629	199.648	7.185	0.204	0.239

We now fit the model using the variable subset with the smallest AIC value.

```r
step_min_aic <- steps_all[1, ]
best_vars <- strsplit(step_min_aic$predictors, " ")[[1]]
best_formula1 <- as.formula(paste("mpg", "~", paste0(best_vars, collapse = "+")))
best_mod1 <- lm(best_formula1, vehicles)
```

The F-test and the t-test results show that all variables are significant. R^2 dropped only by 0.0193522

```r
summary(best_mod1)
```

```
##
## Call:
## lm(formula = best_formula1, data = vehicles)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.4811 -1.5555 -0.7257  1.4110  4.6610
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.6178     6.9596   1.382 0.177915
## wt           -3.9165     0.7112  -5.507 6.95e-06 ***
## qsec          1.2259     0.2887   4.247 0.000216 ***
## am            2.9358     1.4109   2.081 0.046716 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.459 on 28 degrees of freedom
## Multiple R-squared:  0.8497, Adjusted R-squared:  0.8336
## F-statistic: 52.75 on 3 and 28 DF,  p-value: 1.21e-11
```

The p value of the F test H0: `best_mod1` vs H1: `mod` is 0.8636 which means that for any test with significance level lower than 0.8636 (i.e. any reasonable significance level) we can't reject H0. We thus keep the `best_mod1`.

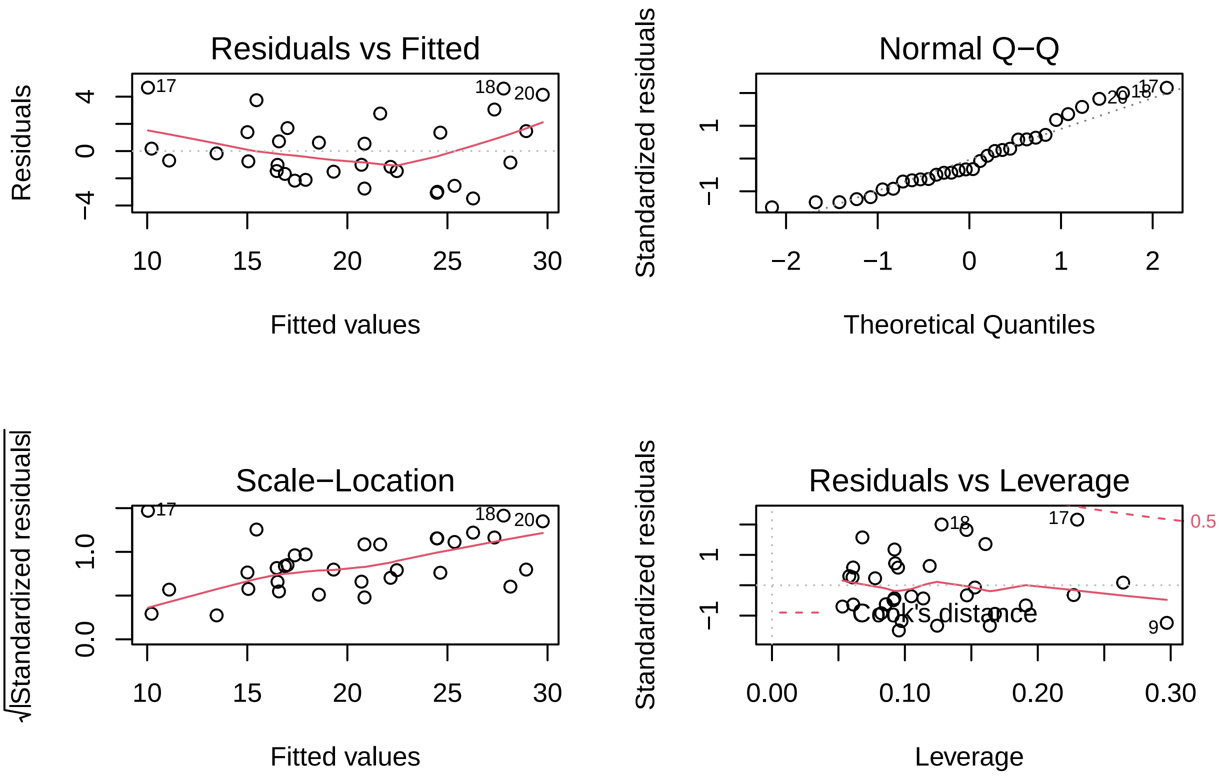
```r
anova(best_mod1, mod, test = "F")
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ wt + qsec + am
## Model 2: mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     28 169.29
## 2     21 147.49  7    21.791 0.4432 0.8636
```

A.3

As we noted before, the qqplot of the regressand doesn't look great. We'll try to take the logarithm of it in case it follows a lognormal distribution.

```r
par(mfrow = c(2,2))
plot(best_mod1)
```

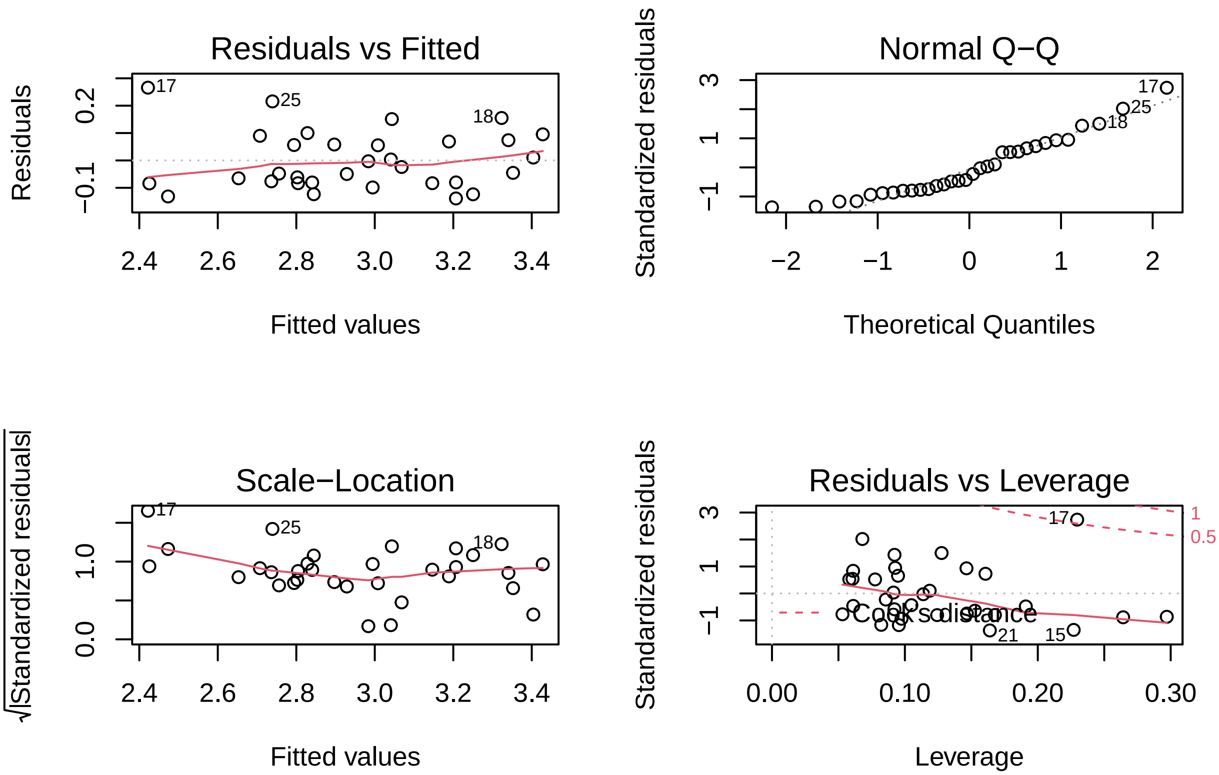


We regress on log(mpg).

```
best_formula2 <- as.formula(paste("log(mpg)", "~", paste0(best_vars, collapse = "+")))
best_mod2 <- lm(best_formula2, vehicles)
```

The qq plot looks better. The residuals plots also look better. The only influential point appears to be 17, whose values look alright, although I am not a car expert.

```
par(mfrow = c(2,2))
plot(best_mod2)
```



We also observe that R^2 increased by 0.0255571.

```
summary(best_mod2)
```

```
##
## Call:
## lm(formula = best_formula2, data = vehicles)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13879 -0.08114 -0.03466  0.07030  0.26575
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.69410    0.31326   8.600 2.40e-09 ***
## wt          -0.22456    0.03201  -7.015 1.25e-07 ***
## qsec         0.05329    0.01299   4.101  0.00032 ***
## am           0.08558    0.06351   1.347  0.18863
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1107 on 28 degrees of freedom
## Multiple R-squared:  0.8752, Adjusted R-squared:  0.8619
## F-statistic: 65.47 on 3 and 28 DF,  p-value: 9.036e-13
```

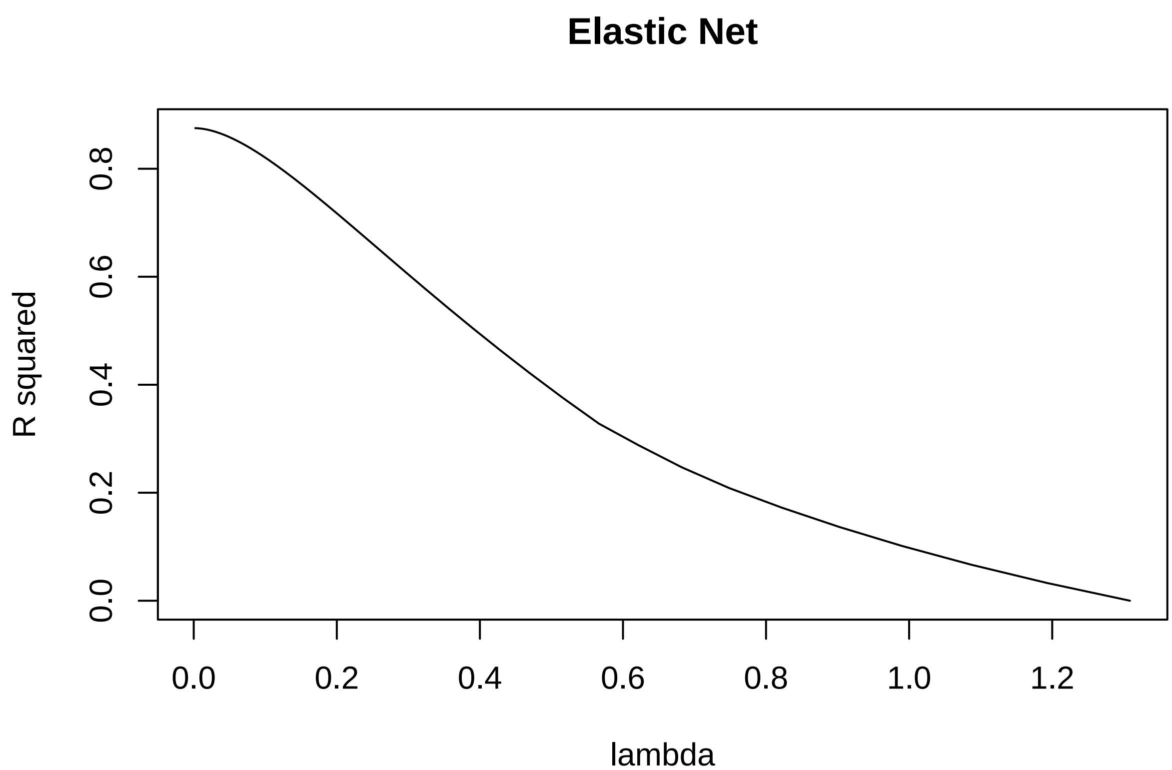
We now attempt $l1$ and $l2$ regularization. We will attempt to use both (i.e. elastic net), with weights $\alpha = 0.2$

for $l1$ and $1 - \alpha = 0.8$ for $l2$. `glmnet` tries multiple values for the regularization parameter λ.

```
y <- log(vehicles$mpg)
x <- vehicles[best_vars]
elastic_mod <- glmnet(x, y, alpha = 0.2, family = "gaussian")
```

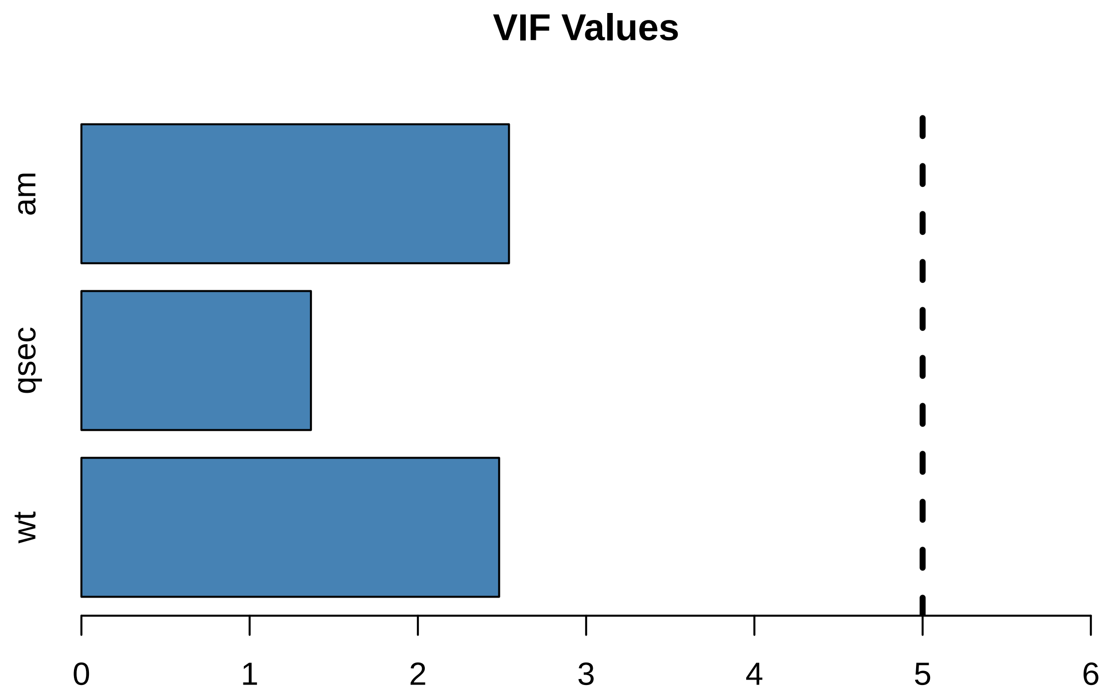
We notice that the smaller λ gets, the better the R^2 value (`%Dev`) with the maximum reached at 0.8751807, which is the same as the original model. This results indicates that our model is simple enough to the point where it doesn't need any regularization.

```
plot(elastic_mod$lambda, elastic_mod$dev.ratio, type = "l", lty = 1,
     main = "Elastic Net", xlab = "lambda", ylab = "R squared")
```



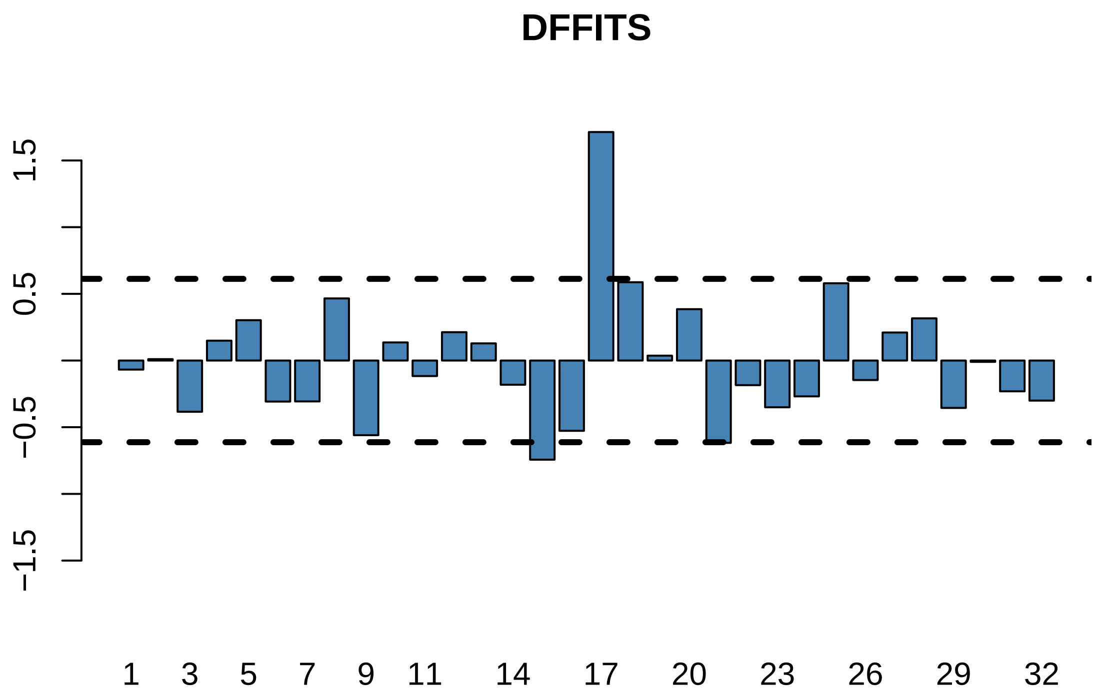
We see that the there's much less multicollinearity now, as expected.

```
barplot(vif(best_mod2), main = "VIF Values",
        horiz = TRUE, col = "steelblue", xlim = c(0, 6))
abline(v = 5, lwd = 3, lty = 2)
```



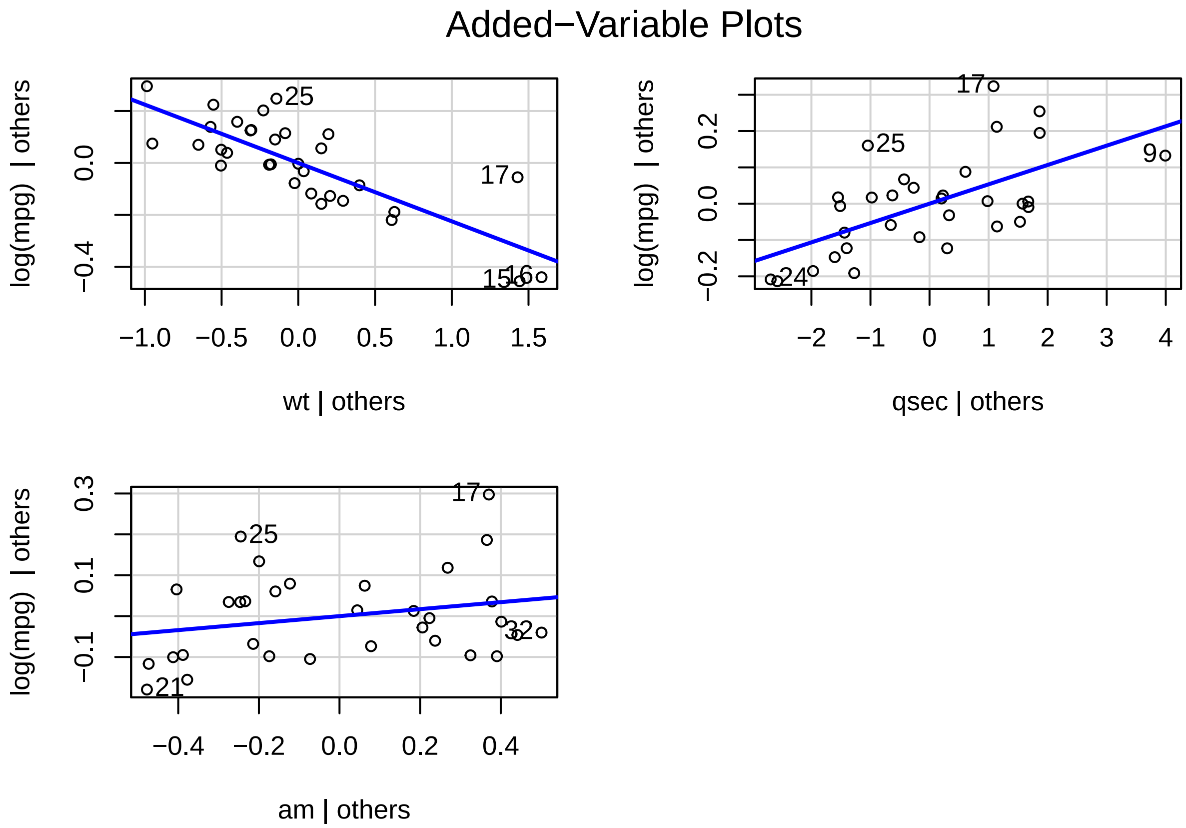
From the DFFITS we confirm that 17 is the only influential point

```
dffits_values <- dffits(best_mod2)
fthresh <- 2 * sqrt((length(best_mod2$coefficients) - 1) / nrow(vehicles))
ylim <- max(fthresh, abs(dffits_values)) + 0.2
barplot(dffits_values, main = "DFFITS", col = "steelblue", ylim = c(-ylim, ylim))
abline(h = c(-fthresh, fthresh), lwd = 3, lty = 2)
```



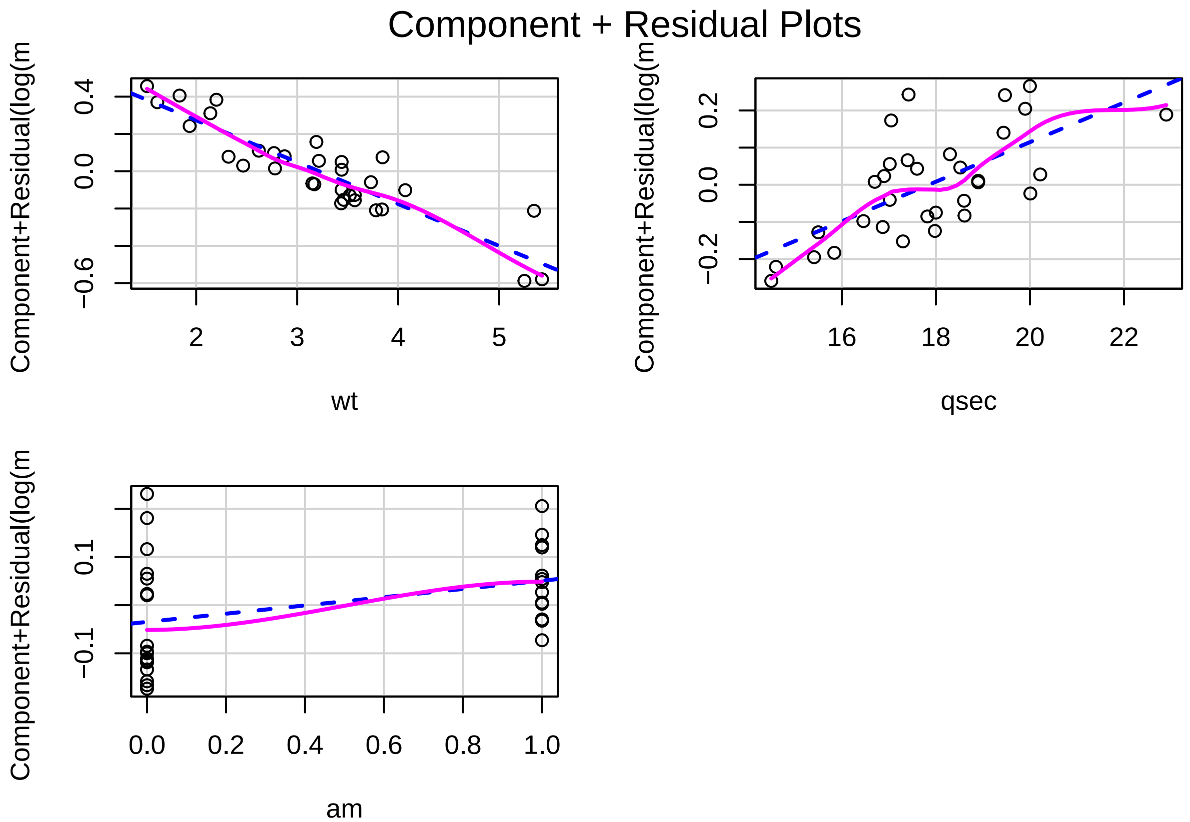
Added variable plots: We see that there is a linear relationship between the predicted variable and each predictor variable, separately. This means that all variables belong to the model.

```
avPlots(best_mod2)
```



Partial residual plots: for each variable we have that the residual and the component curves are relatively close to each other, thus all the variables belong to the model.

```
crPlots(best_mod2)
```



Confidence intervals for the model's coefficients.

```
cbind(
  fit = best_mod2$coefficients,
  confint(best_mod2, level = 0.95)
)
```

```
##                     fit       2.5 %      97.5 %
## (Intercept)  2.69409612  2.05240274  3.33578950
## wt          -0.22455989 -0.29013462 -0.15898517
## qsec         0.05328768  0.02667156  0.07990379
## am           0.08557580 -0.04451343  0.21566503
```

We create a synthetic data point, similar to 19.

```
synthetic_x <- list(wt = 1.6, qsec = 18.5, am = 1)
```

Confidence interval of the expected value of the prediction

```
log_conf <- predict(best_mod2, synthetic_x, interval = "confidence", level = 0.95)
conf <- exp(log_conf)
conf
```

```
##        fit      lwr      upr
## 1 30.1504 27.87311 32.61374
```

Prediction interval of the value of the prediction

```
log_pred <- predict(best_mod2, synthetic_x, interval = "prediction", level = 0.95)
pred <- exp(log_pred)
pred
```

```
##       fit      lwr      upr
## 1 30.1504 23.71881 38.32598
```

Interpretation of the coefficients

- **wt** in -0.2901346, -0.1589852 means that the heavier the vehicle, the fewer miles it can travel given one gallon (-wt miles less)
- **qsec** in 0.0266716, 0.0799038 means that faster vehicles can travel longer distances given one gallon (qsec miles more). This might be due the slow vehicles in the dataset being trucks.
- **am** in -0.0445134, 0.215665 means that manual vehicles tend to be able to travel more miles given one gallon. The confidence intervals are quite large, which might be due to the fact that the skill of manual car drivers varies a lot.
- Now, considering the interval for **E[mpg]** and **mpg**, we see that our synthetic point is similar to sample 19 which has 30.4 mpg. Our prediction for the synthetic point is 30.1503973 which is pretty much what we'd expect, although it is worth noting that the interval lengths are very large.

Part B

We start by loading the data

```
canary <- read.table("./data/canary.txt", header = TRUE)
canary$group <- as.factor(canary$group)
```

We initialize a model, using `pulses`, `group` and `pulses:group` i.e. the interaction between them.

```
mod <- lm(Temp ~ pulses * group, data = canary)
summary(mod)
```

```
##
## Call:
## lm(formula = Temp ~ pulses * group, data = canary)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04192 -0.27068  0.02695  0.34389  1.08949
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.58188    0.98483   3.637  0.00115 **
## pulses         0.25910    0.01141  22.711  < 2e-16 ***
## groupB         1.02615    1.14872   0.893  0.37959
## pulses:groupB  0.02146    0.01471   1.459  0.15603
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.4812 on 27 degrees of freedom
## Multiple R-squared:  0.9857, Adjusted R-squared:  0.9842
## F-statistic: 622.2 on 3 and 27 DF,  p-value: < 2.2e-16
```

Observe that while `pulses` is significant, `group` and `group:pulses` are not when they coexist in the model. This implies that removing at least one of the three might be a good idea.

We can try all subsets of the variables

```r
mod <- lm(Temp ~ pulses * group, data = canary)
steps_all <- ols_step_all_possible(mod)
steps_all <- steps_all[order(steps_all$cp), ]
knitr::kable(steps_all[c(-1, -2)],
             caption = "Best results according to Mallows' Cp", d = 3)
```

Table 2: Best results according to Mallows' Cp

	predictors	rsquare	adjr	predrsq	cp	aic	sbic	sbc	msep	fpe	apc	hsp
3	pulses:group	0.985	0.984	0.982	0.798	47.248	-41.007	52.984	6.883	0.245	0.017	0.008
6	group pulses:group	0.986	0.984	0.981	2.000	48.345	-37.135	55.515	6.933	0.254	0.017	0.009
5	pulses pulses:group	0.985	0.984	0.982	2.798	47.248	-36.258	52.984	7.138	0.252	0.018	0.009
7	pulses group pulses:group	0.986	0.984	0.981	4.000	48.345	-32.246	55.515	7.199	0.261	0.018	0.009
4	pulses group	0.985	0.984	0.981	4.129	48.698	-34.885	54.434	7.479	0.264	0.019	0.009
1	pulses	0.920	0.917	0.912	124.652	97.843	6.972	102.145	37.548	1.289	0.091	0.043
2	group	0.231	0.205	0.123	1428.798	167.956	74.341	172.258	360.450	12.376	0.875	0.415

It is important to note that when we view our data by group, **`group` is essentially part of the intercept**, since it's constant.

We can have 3 types of pairs of lines:

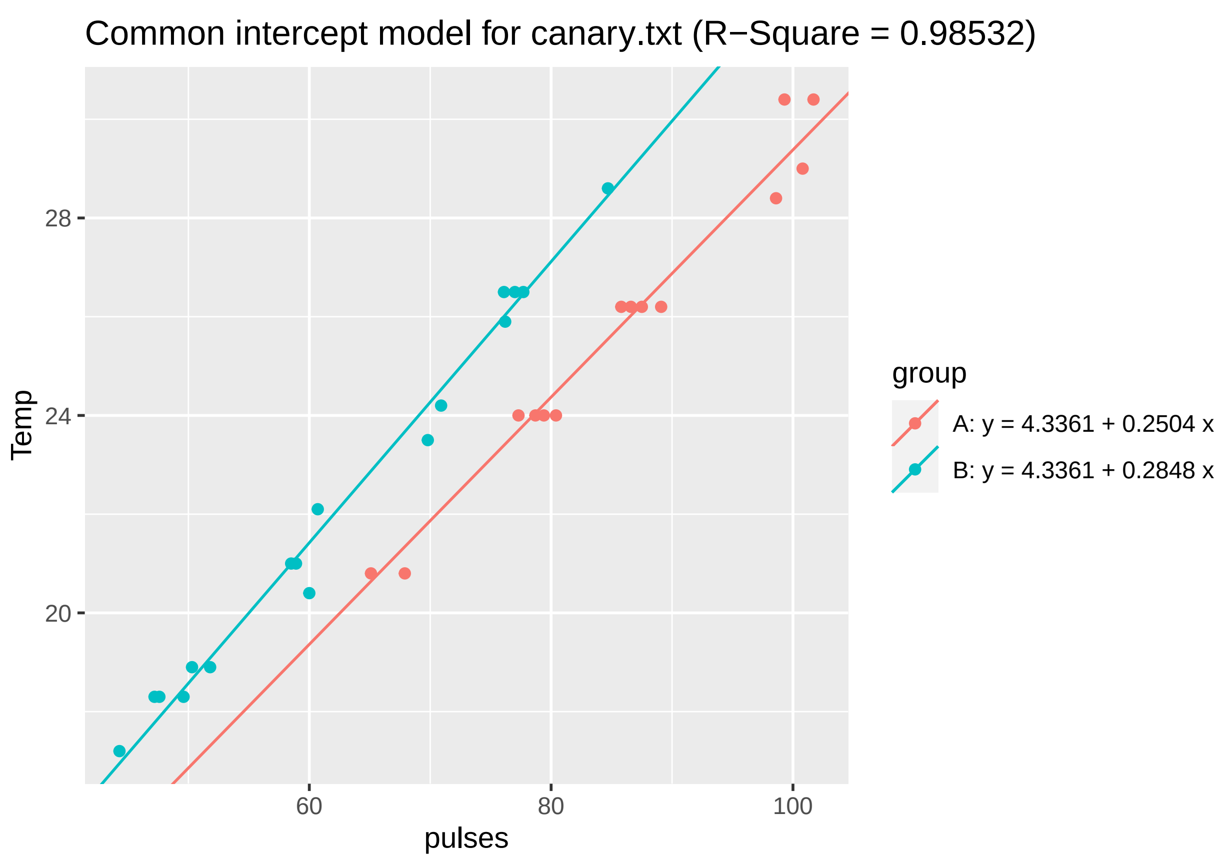
- **Case I**, Two separate lines: When `pulses:group` is part of the model.
- **Case II**, Two parallel lines: When `pulses:group` is not part of the model, but `group` is part of the model.
- **Case III**, One common line: When neither `pulses:group`, nor `group` is part of the model.

The options `pulses:group` and `group + pulses` have essentially the same R^2. Even though their Mallows' Cp differ, we should view the models as equally complex, since `pulses:group` describes a pair of lines with the intercept fixed, and `group + pulses:group` describes a pair of lines with the slope fixed. Thus, either choice will suffice.

```r
mod_common_intercept <- lm(Temp ~ pulses:group, data = canary)
mod_common_slope <- lm(Temp ~ group + pulses, data = canary)
```

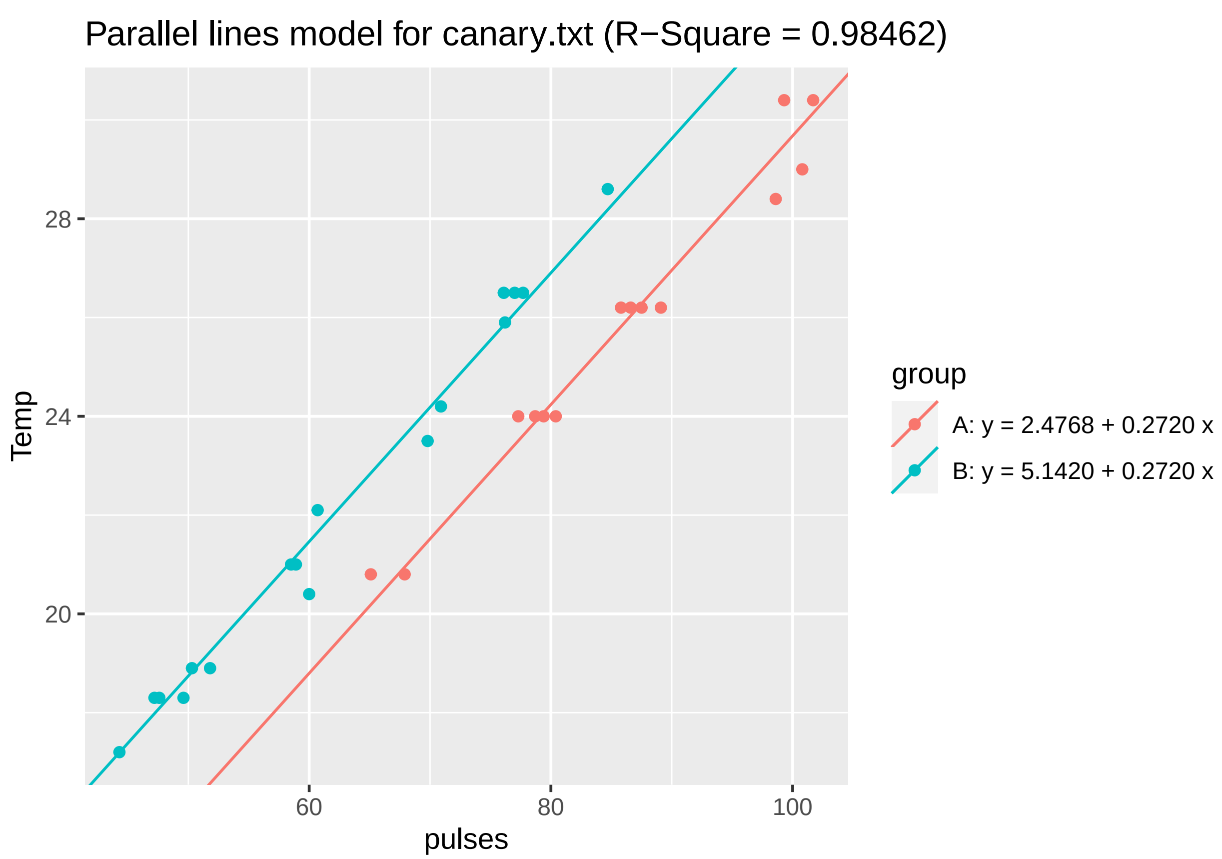
Let us see how the models perform visually.

```r
common_intercept <- mod_common_intercept$coefficients["(Intercept)"]
slopeA <- mod_common_intercept$coefficients["pulses:groupA"]
slopeB <- mod_common_intercept$coefficients["pulses:groupB"]
gg_common_intercept <- ggplot(data = canary, aes(x = pulses, y = Temp)) +
  geom_point(aes(col = group))
# Plot the lines
gg_common_intercept <- gg_common_intercept + geom_abline(aes(intercept = common_intercept,
                                                              slope = slopeA, color = "A"))
gg_common_intercept <- gg_common_intercept + geom_abline(aes(intercept = common_intercept,
                                                              slope = slopeB, color = "B"))
# Add labels
strA <- sprintf("A: y = %.4f + %.4f x", common_intercept, slopeA)
strB <- sprintf("B: y = %.4f + %.4f x", common_intercept, slopeB)
title <- sprintf("Common intercept model for canary.txt (R-Square = %.5f)",
                 summary(mod_common_intercept)$r.square)
gg_common_intercept <- gg_common_intercept + scale_color_discrete(labels=c(strA, strB))
gg_common_intercept <- gg_common_intercept + labs(title = title)
gg_common_intercept
```




```r
interceptA <- mod_common_slope$coefficients["(Intercept)"]
interceptB <- sum(mod_common_slope$coefficients[c("(Intercept)", "groupB")])
common_slope <- mod_common_slope$coefficients["pulses"]
gg_common_slope <- ggplot(data = canary, aes(x = pulses, y = Temp)) +
  geom_point(aes(col = group))
# Plot the lines
```

```r
gg_common_slope <- gg_common_slope + geom_abline(aes(intercept = interceptA,
                                                      slope = common_slope, color = "A"))
gg_common_slope <- gg_common_slope + geom_abline(aes(intercept = interceptB,
                                                      slope = common_slope, color = "B"))
# Add labels
strA <- sprintf("A: y = %.4f + %.4f x", interceptA, common_slope)
strB <- sprintf("B: y = %.4f + %.4f x", interceptB, common_slope)
title <- sprintf("Parallel lines model for canary.txt (R-Square = %.5f)",
                 summary(mod_common_slope)$r.square)
gg_common_slope <- gg_common_slope + scale_color_discrete(labels=c(strA, strB))
gg_common_slope <- gg_common_slope + labs(title = title)
gg_common_slope
```



Summary of the common intercept model

```r
summary(mod_common_intercept)
```

```
##
## Call:
## lm(formula = Temp ~ pulses:group, data = canary)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02116 -0.27291  0.00597  0.27724  1.19542
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)    4.336126   0.505121   8.584  2.5e-09 ***
## pulses:groupA 0.250438   0.005988  41.822  < 2e-16 ***
## pulses:groupB 0.284751   0.007987  35.650  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.4795 on 28 degrees of freedom
## Multiple R-squared:  0.9853, Adjusted R-squared:  0.9843 
## F-statistic: 939.6 on 2 and 28 DF,  p-value: < 2.2e-16
```

Summary of the parallel model

```r
summary(mod_common_slope)
```

```
## 
## Call:
## lm(formula = Temp ~ group + pulses, data = canary)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.06270 -0.27992  0.03071  0.35181  0.91241 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.476831   0.642199   3.857 0.000616 ***
## groupB      2.665171   0.245586  10.852 1.54e-11 ***
## pulses      0.272012   0.007345  37.032  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.4908 on 28 degrees of freedom
## Multiple R-squared:  0.9846, Adjusted R-squared:  0.9835 
## F-statistic:   896 on 2 and 28 DF,  p-value: < 2.2e-16
```

Below we test our simplified models against the full model.

Two lines with common intercept

```r
anova(mod_common_intercept, mod, test = "F")
```

```
## Analysis of Variance Table
## 
## Model 1: Temp ~ pulses:group
## Model 2: Temp ~ pulses * group
##   Res.Df    RSS Df Sum of Sq     F Pr(>F)
## 1     28 6.4377                          
## 2     27 6.2529  1    0.1848 0.798 0.3796
```

Two parallel lines

```r
anova(mod_common_slope, mod, test = "F")
```

```
## Analysis of Variance Table
##
## Model 1: Temp ~ group + pulses
## Model 2: Temp ~ pulses * group
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     28 6.7460
## 2     27 6.2529  1   0.49314 2.1294  0.156
```

Single line (p-value = 7.62e-11 means that we reject this case for all sensible significance levels)

```r
anova(lm(Temp ~ pulses, data = canary), mod)
```

```
## Analysis of Variance Table
##
## Model 1: Temp ~ pulses
## Model 2: Temp ~ pulses * group
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     29 35.121
## 2     27  6.253  2    28.868 62.326 7.62e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- If we accept the common intercept model, then this means that groupB "produces" 0.034313 more degrees per pulse than groupA.
- If we accept the parallel lines model, then this means that groupB always "produces" 2.6651707 more degrees than groupA, regardless of temperature.