REAN protokol

Martin Oharek

12 února 2020

Knihovny

Na začátek načteme doporučené knihovny

```
lbs <- c('car','MASS','tidyverse','ggplot2','ISLR','graphics','effects','leaps','psych',
         'lattice','lmtest','robustbase')

install.lib <- lbs[!lbs %in% installed.packages()]
for(libs in install.lib) install.packages(libs, dependences = TRUE)
sapply(lbs, require, character = TRUE)
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
## Loading required package: MASS
```

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
## Warning: package 'stringr' was built under R version 3.6.2
```

```
## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::recode() masks car::recode()
## x dplyr::select() masks MASS::select()
## x purrr::some()   masks car::some()
```

```
## Loading required package: ISLR
```

```
## Loading required package: effects
```

```
## Warning: package 'effects' was built under R version 3.6.2
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
```

```
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
## Loading required package: leaps
```

```
## Loading required package: psych
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
## Loading required package: lattice
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: robustbase
```

```
##
## Attaching package: 'robustbase'
```

```
## The following object is masked from 'package:psych':
##
##     cushny
```

```
##        car       MASS  tidyverse    ggplot2       ISLR   graphics    effects
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
##      leaps      psych    lattice     lmtest robustbase
##       TRUE       TRUE       TRUE       TRUE       TRUE
```

Zpracování dat, průzkumová a grafická část

Otázka č.1: Zjistěte, zdali data neobsahují chybějící hodnoty (NA), pokud ano tak rozhodněte zdali můžete příslušná pozorování z dat odstranit.

Jako první data načteme z příslušného souboru

```
my_data <- read.table("auto-mpg-01rean.txt",header = TRUE)
my_data = as.data.frame(my_data)
```

Potom se na data podíváme přes summary funkci

```
summary(my_data)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   :10.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.00
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.75
##  Median :23.00   Median :4.000   Median :151.0   Median : 95.00
##  Mean   :23.55   Mean   :5.475   Mean   :194.8   Mean   :105.08
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:302.0   3rd Qu.:130.00
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.00
##  NA's   :9                                       NA's   :6
##      weight      acceleration     model_year        origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2226   1st Qu.:13.70   1st Qu.:73.00   1st Qu.:1.000
##  Median :2822   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2979   Mean   :15.52   Mean   :75.92   Mean   :1.569
##  3rd Qu.:3618   3rd Qu.:17.18   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                 car_name
##  ford pinto         :  6
##  ford maverick      :  5
##  chrysler  matador :  5
##  toyota corolla     :  5
##  volkswagen rabbit  :  5
##  chevrolet chevette:  4
##  (Other)            :376
```

Vidíme, že se vyskytuje 6 NaNs v proměnné horsepower a 9 NaNs v proměnné mpg. Odstranění příslušných řádků závisí na plánovaném využití datasetu. Pokud budeme chtít regresovat mpg, pak jistě budeme muset odstranit řádky s NaNy v mpg. Pokud budeme chtít regresovat navíc pomocí horsepower, pak budeme muset odstranit i řádky s NaNy v horsepower. V našem případě je celkově 15 řádků s Nany, což není vzhledem k celkovému počtu 406 mnoho, navíc budeme téměř ve všech pozdějších úlohách pracovat jak s mpg, tak horsepower dohromady, proto pro větší přehlednost budeme uvažovat pouze data bez NaNů již od začátku pro všechny následující úlohy.

```
data_mpghp = my_data[!is.na(my_data$mpg) & !is.na(my_data$horsepower),]
```

Otázka č.2: Které proměnné jsou kvantitativní a které kvalitativní? Jeli možno některé zařadit do obou skupin, pro kterou se rozhodnete? Které proměnné budete brát jako faktorové a proč?

Za kvalitativní proměnné většinou bereme kategorické proměnné, které nabývají diskrétních hodnot. Kvantitativní jsou naopak spojité, numerické proměnné, které popisují číselnou velikost, množství,..

Zřejmě kvantitativní proměnné v našem případě jsou mpg, displacement, horsepower, weight a acceleration. Faktorové jsou určitě origin a car_name. Do obou skupin lze zařadit cylinders a model_year. Jestli danou proměnnou brát jako faktorovou či nikoliv závisí na zkoumaném problému a plánovaném využití výsledného modelu. Např. pokud budeme chtít používat model i na automobily o jiném počtu válců než máme v datasetu, pak zvolíme cylinders jako numerickou proměnnou. Ale např. model_year bych určitě nebral jako numerickou proměnnou, zde nedává příliš smysl do modelu dosazovat jiné roky, než se kterými se setkáme v datasetu, nebo např. dosazovat neceločíselné hodnoty let. Předpokládat, že spotřeba se bude měnit lineárně i v budoucnu či v minulosti vzhledem k našim datům mi nepřijde správné.

Momentálně zvolím cylinders i model_year jako kategorické proměnné.

```
data_mpghp$origin = as.factor(data_mpghp$origin)
data_mpghp$cylinders = as.factor(data_mpghp$cylinders)
data_mpghp$model_year = as.factor(data_mpghp$model_year)
```

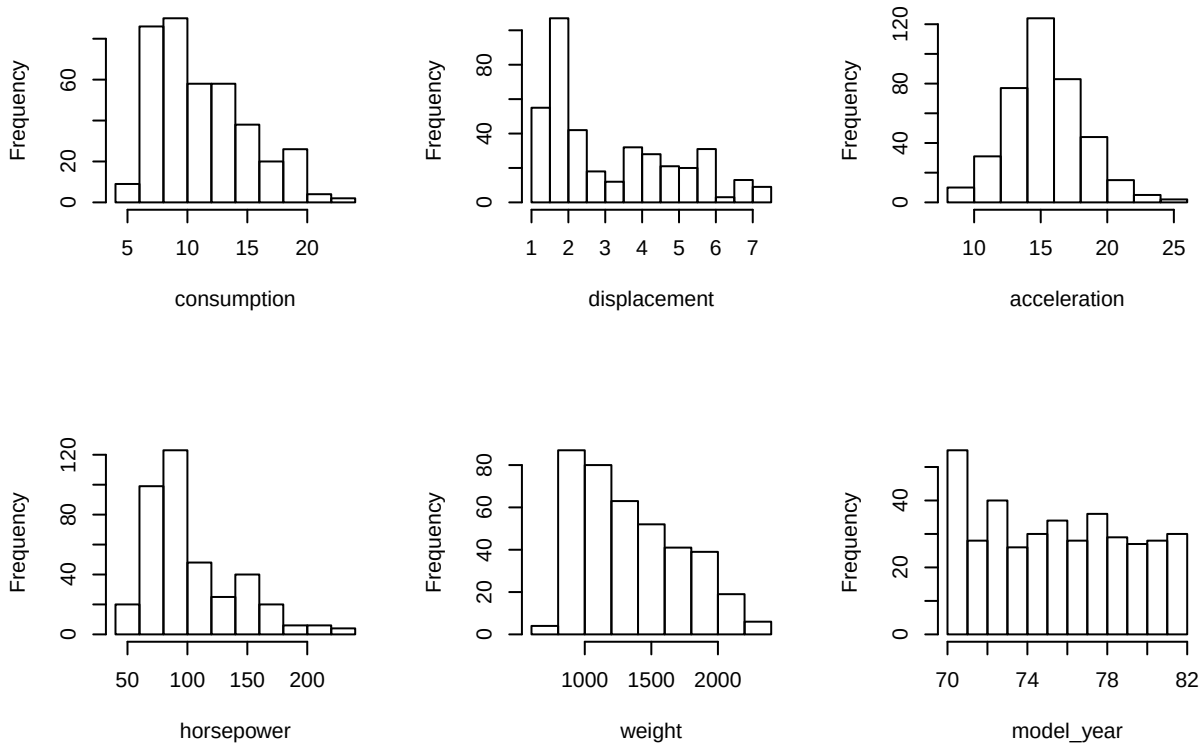
Otázka č.3: Proměnnou mpg nahraďte proměnnou consumption, kde bude místo počtu ujetých mil na galon paliva uvedena hodnota počet litrů na 100km. Jednotky proměnné displacement převěďte z kubických palců na litry a weight z liber na kilogramy.

```
data_mpghp$consumption = (100*3.785411)/(data_mpghp$mpg * 1.609344)
data_mpghp$displacement = data_mpghp$displacement*0.016387064
data_mpghp$weight = data_mpghp$weight*0.45359237
```

Otázka č.4: Vykreslete histogramy a odhady hustot consumption, displacement, acceleration, horsepower, weight, model_year. Proměnnou consumption vykreslete pomocí scatterplotu spolu s ostatními proměnnými - závislost odezvy ("consumption") na vysvětlujících proměnných. Proložte body jak lineárním odhadem, tak vyhlazenou křivkou lokální regrese, buď pomocí LOESS (locally estimated scatterplot smoothing) nebo LOWESS (locally weighted scatterplot smoothing)(lines(lowess(X,Y))). Co lze z tohoto obrázku předpokládat o závislosti spotřeby auta na dalších proměnných.

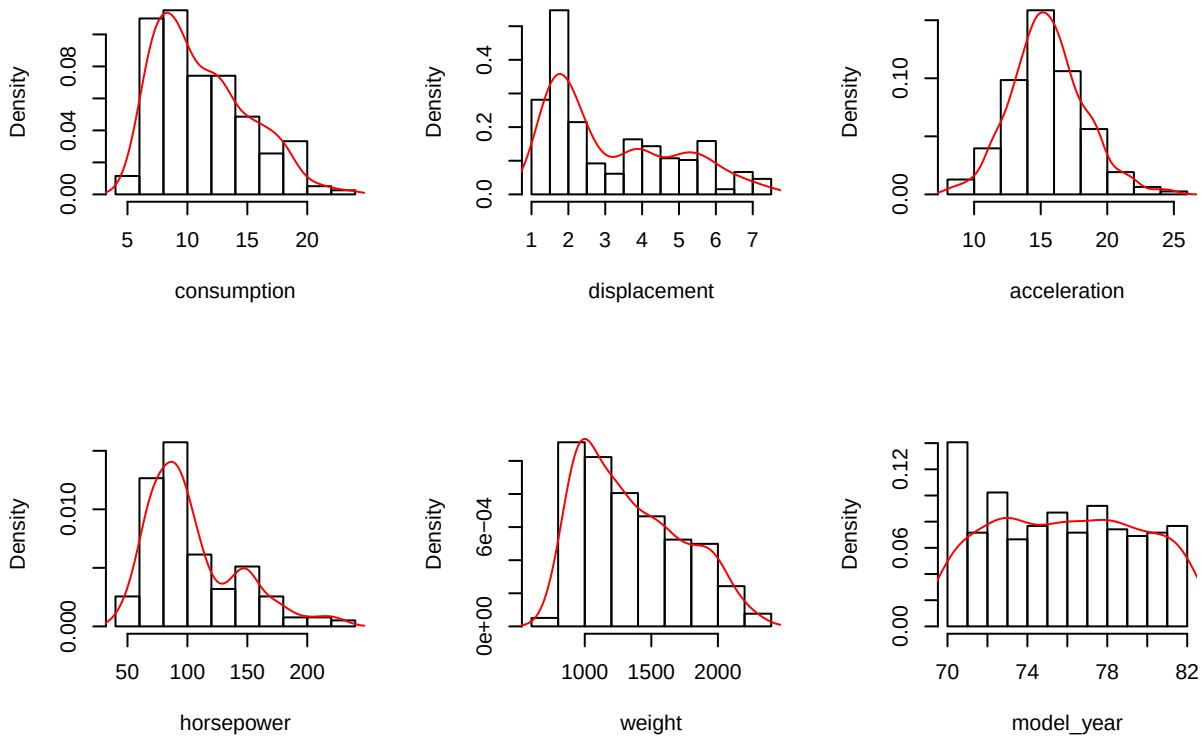
Jako první vykreslíme histogramy.

```
data_mpghp$model_year = as.numeric(as.character(data_mpghp$model_year))# změníme zpět na numerickou kvů

cols = c('consumption', 'displacement', 'acceleration', 'horsepower', 'weight', 'model_year')
par(mfrow = c(2, 3))
for (col in cols)
{
  hist(data_mpghp[,col],xlab = col,main = '')
}
```



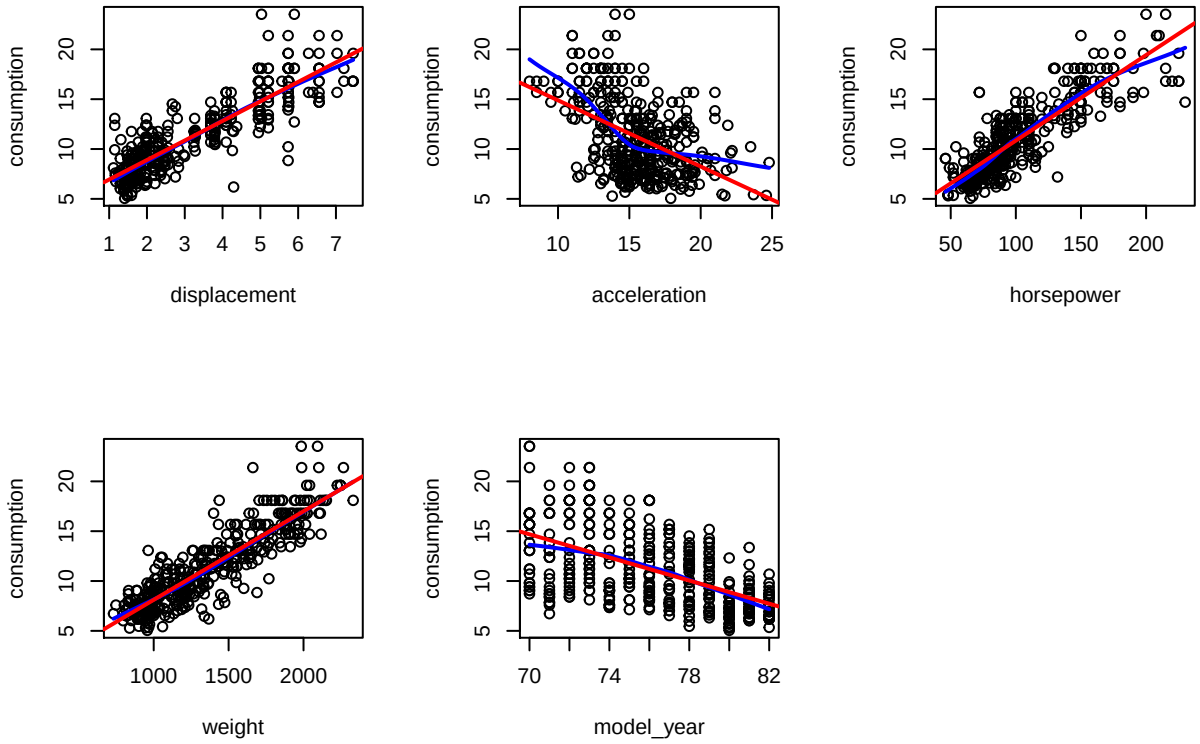
Nyní vykreslíme odhady hustot.

```
par(mfrow = c(2, 3))
for (col in cols)
{
  hist(data_mpghp[,col],xlab = col,freq=FALSE,main = '')
  lines(density(data_mpghp[,col]), col="red", lwd=1)
}
```



A naposledy scatterploty s křivkou lokální regrese.

```
par(mfrow = c(2, 3))
cols = c('displacement', 'acceleration', 'horsepower', 'weight', 'model_year')
for (col in cols)
{
  scatter.smooth(data_mpghp[,col],data_mpghp[,'consumption'],
                 xlab = col,ylab = 'consumption',main = '',lpars = list(col='blue',lwd=2))
  abline(lm(data_mpghp[,'consumption'] ~ data_mpghp[,col],data = data_mpghp),col = 'red', lwd = 2)
}
```



Z obrázku vidíme, že se křivky "přibližně" podobají, z toho lze usoudit, že data jsou vhodná modelovat pomocí lineární regrese. Pokud by se křivky významně lišily, pak by už dopředu mohlo být jasnější, že (jednoduchý) lineární model nebude příliš vhodnou volbou a stálo by za to vybrat jiný (nelineární) model.

Dále si můžeme všimnout poměrně logických trendů, které nám ověřují, že data jsou smysluplná, např. můžeme vidět, že se zvyšující se váhou či výkonem vozidla se zvyšuje i spotřeba, což souhlasí s jakousi empirickou a fyzikální znalostí problému. Také se s přibývajícími roky většinou spotřeba snižovala. Zde by bylo zajímavé si vykreslit (bude rozebráno v úkolu č.8) závislost výkonu či zrychlení v závislosti na model_year a ověřit, že deficit ve spotřebě negativně neovlivnil tyto proměnné.

Otázka č.5: Z proměnné car_name vytvořte proměnnou model podle prvního slova obsaženého v řetězci. Pro proměnné model_year, model, cylinders, origin a jejich vztah k odezvě consumption vykreslete krabicové diagramy (boxploty). Je mezi uvedenými proměnnými některá, pro kterou byste na základě krabicových diagramů navrhli sloučit určité úrovně dohromady? Je z těchto grafů vidět, že některá auta mají jinou, než očekávanou spotřebu?

Ke splnění úlohy využijeme balíček "stringr", který nám usnadní práci.

```
library('stringr')
data_mpghp$producer = word(data_mpghp$car_name,1)
data_mpghp$model_year = factor(data_mpghp$model_year) # model_year převedeme zpět na faktor
```

Na vykreslení boxplotu využijeme ggplot

```
library(gplots)
```

```
## Warning: package 'gplots' was built under R version 3.6.2
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(ggplot2)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 3.6.2
```

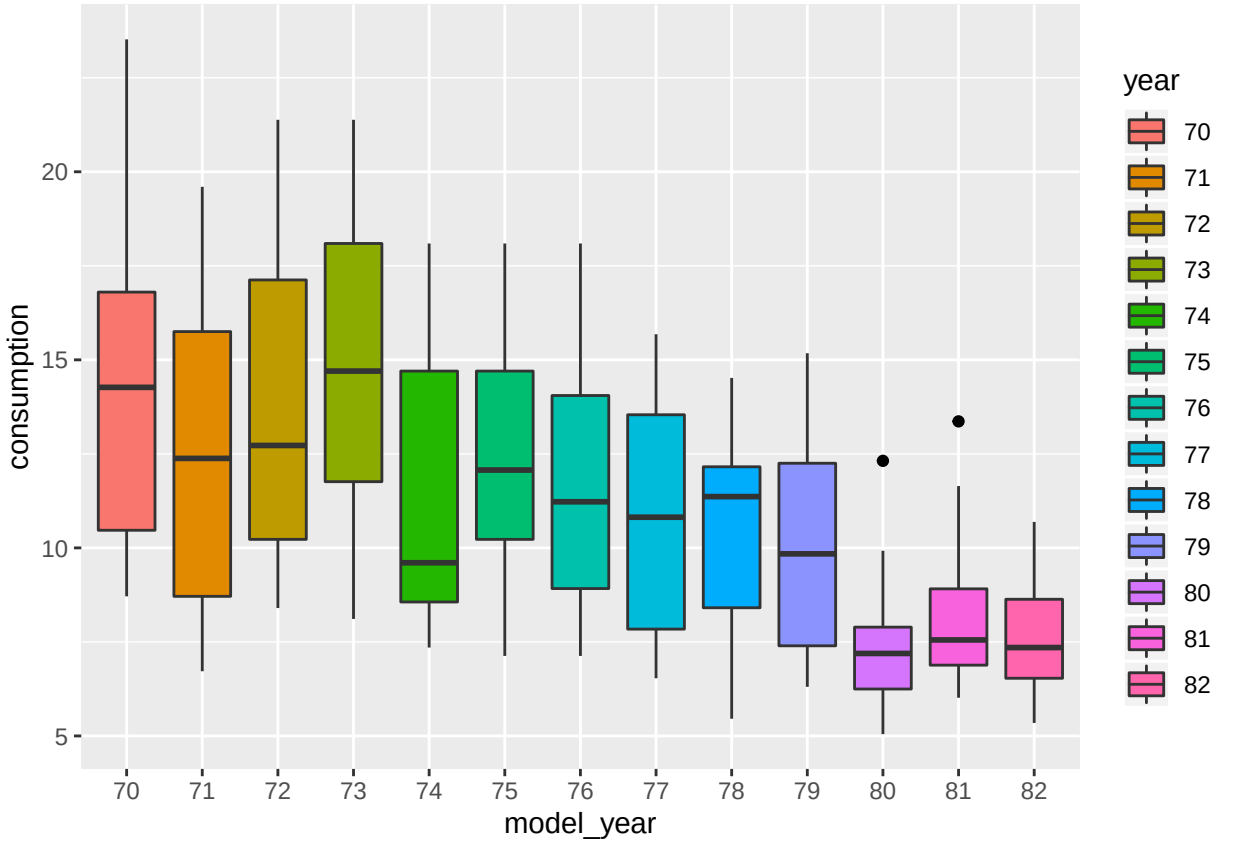
```
## Loading required package: magrittr
```

```
##
## Attaching package: 'magrittr'
```

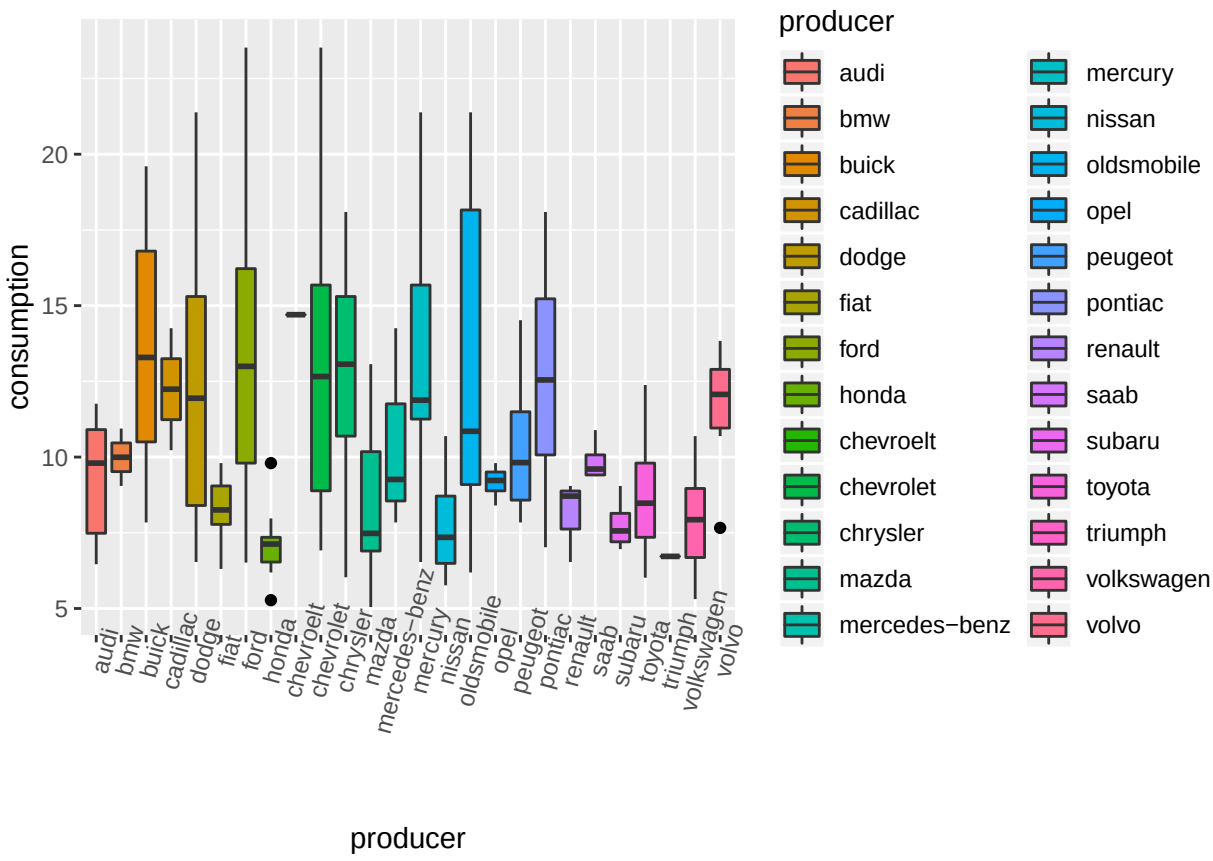
```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

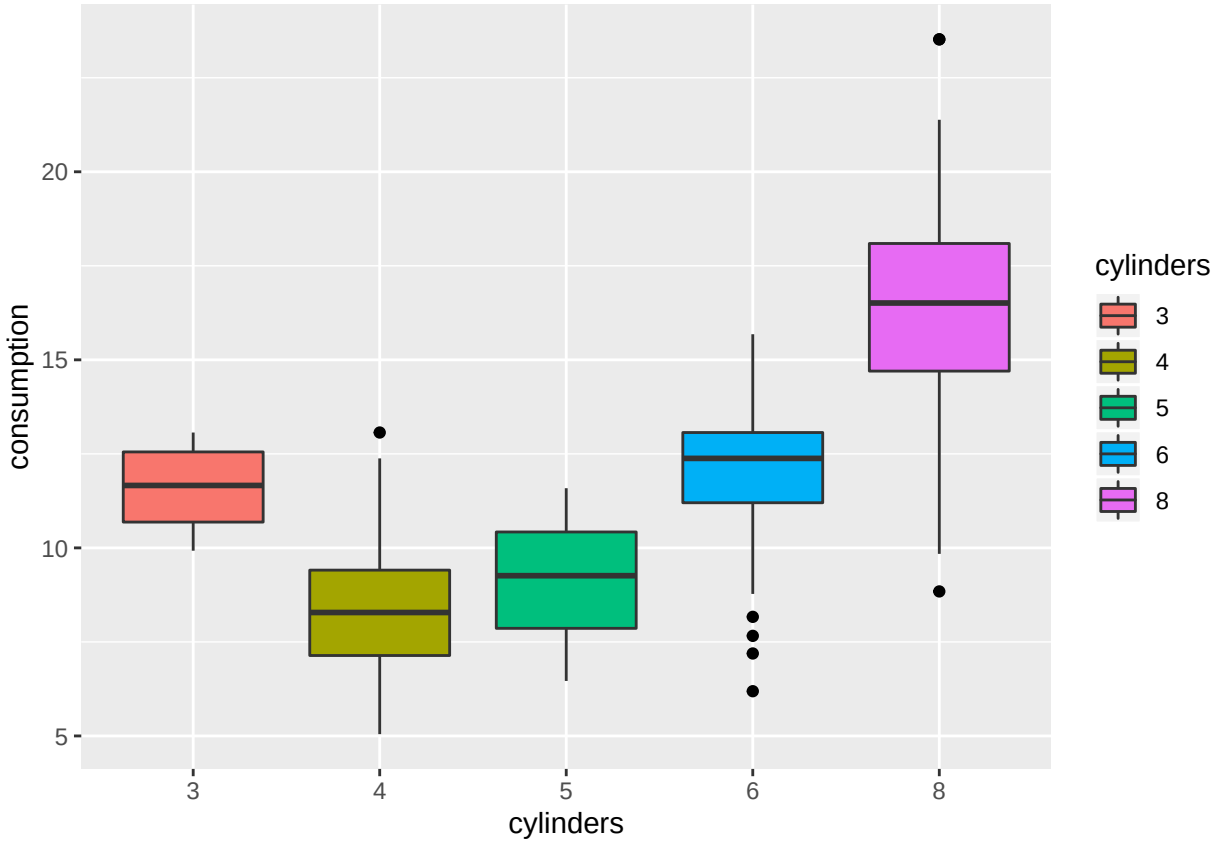
```
cols = c('model_year','producer','cylinders','origin')
p1 <- ggplot(data_mpghp,aes(x=model_year,y=consumption, fill = model_year)) +
    geom_boxplot(notch=FALSE,outlier.color = 'black') + scale_fill_discrete(name='year')
plot(p1)
```



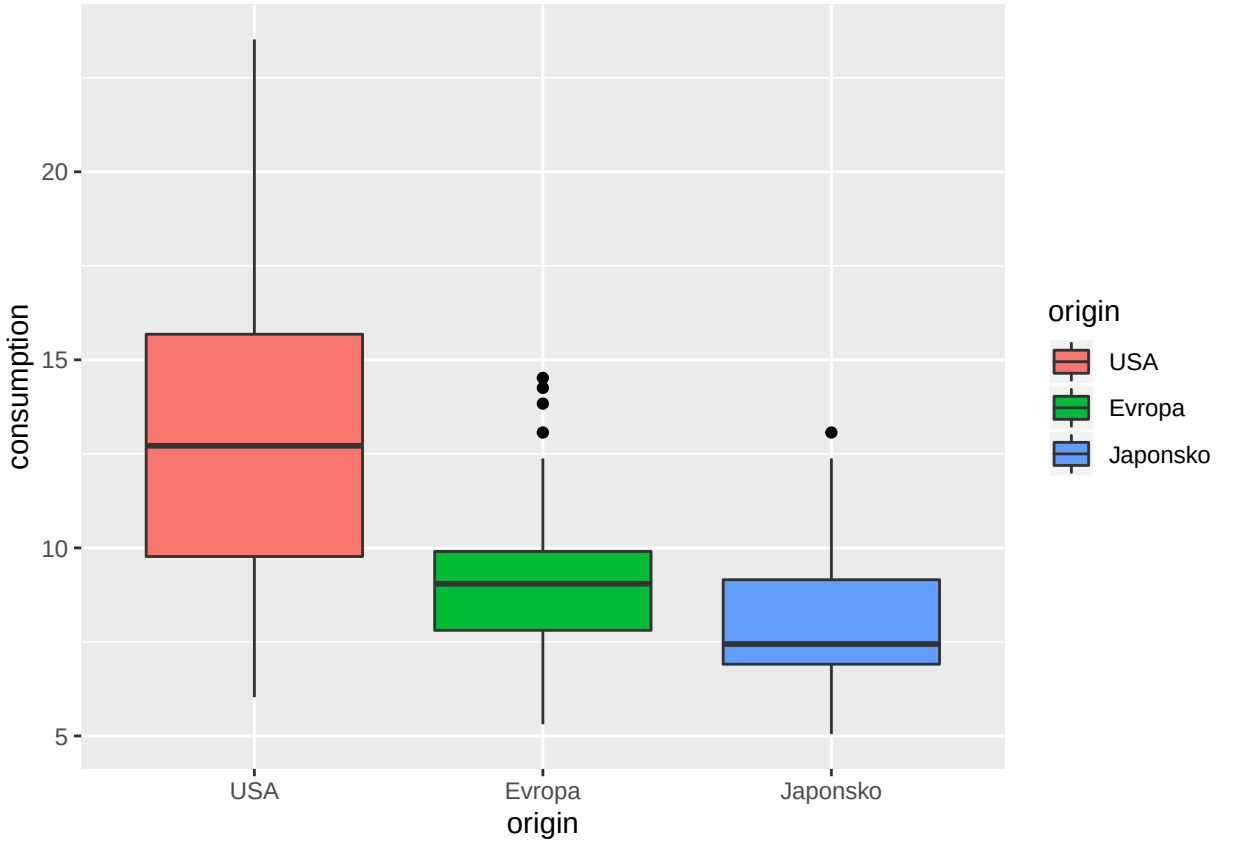

```
p2 <- ggplot(data_mpghp,aes(x=producer,y=consumption, fill = producer)) +
  geom_boxplot(notch=FALSE,outlier.color = 'black') + scale_fill_discrete(name='producer') +
  theme(axis.text.x=element_text(angle=75))
plot(p2)
```




```
p3 <- ggplot(data_mpghp,aes(x=cylinders,y=consumption,fill = cylinders)) +
  geom_boxplot(notch=FALSE,outlier.color = 'black') + scale_fill_discrete(name='cylinders')
plot(p3)
```




```
p4 <- ggplot(data_mpghp,aes(x=origin,y=consumption,fill = origin)) +
  geom_boxplot(notch=FALSE,outlier.color = 'black') + scale_x_discrete(labels=c("1" = "USA", "2" = "Evr
  scale_fill_discrete(name='origin',labels=c("1" = "USA", "2" = "Evropa","3" = "Japonsko"))
plot(p4)
```

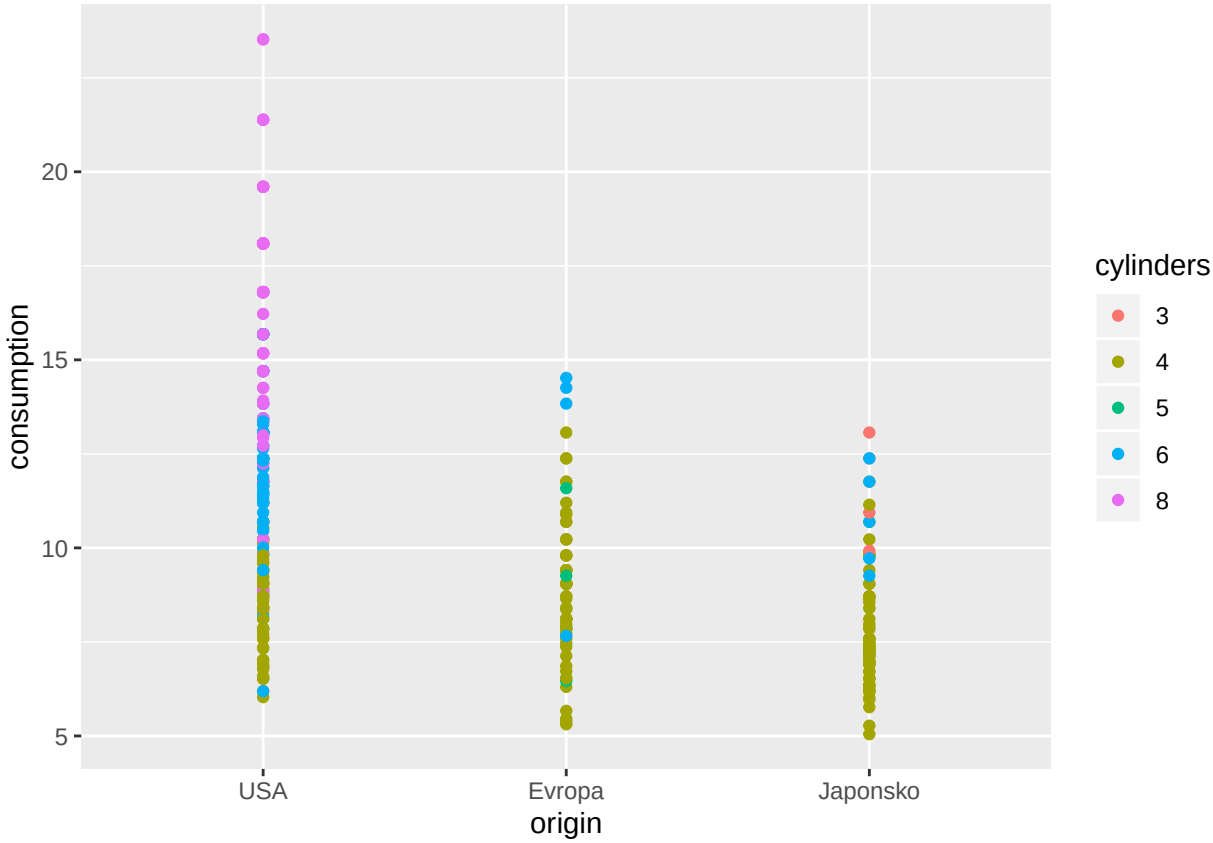


Dle podobnosti středních hodnot a mediánů by se da sloučit model_year na méně faktorů. Daly by se sloučit roky 70-73, 74-79, 80-82. Ostatní proměnné bych neslučoval, jelikož rozdíly středních hodnot a mediánů pro jednotlivé faktory jsou příliš velké.

Jde vidět, že auta se 3 válci mají nižší spotřebu než auta se 4 a 5 válci, což trochu porušuje trend, který vidíme u 4 az 8 válcových aut, kde zpravidla čím větší počet válců, tím nižší spotřeba. Avšak to nemusí být nutně chyba, jelikož spotřebu ovlivňují i další nezávislé proměnné.

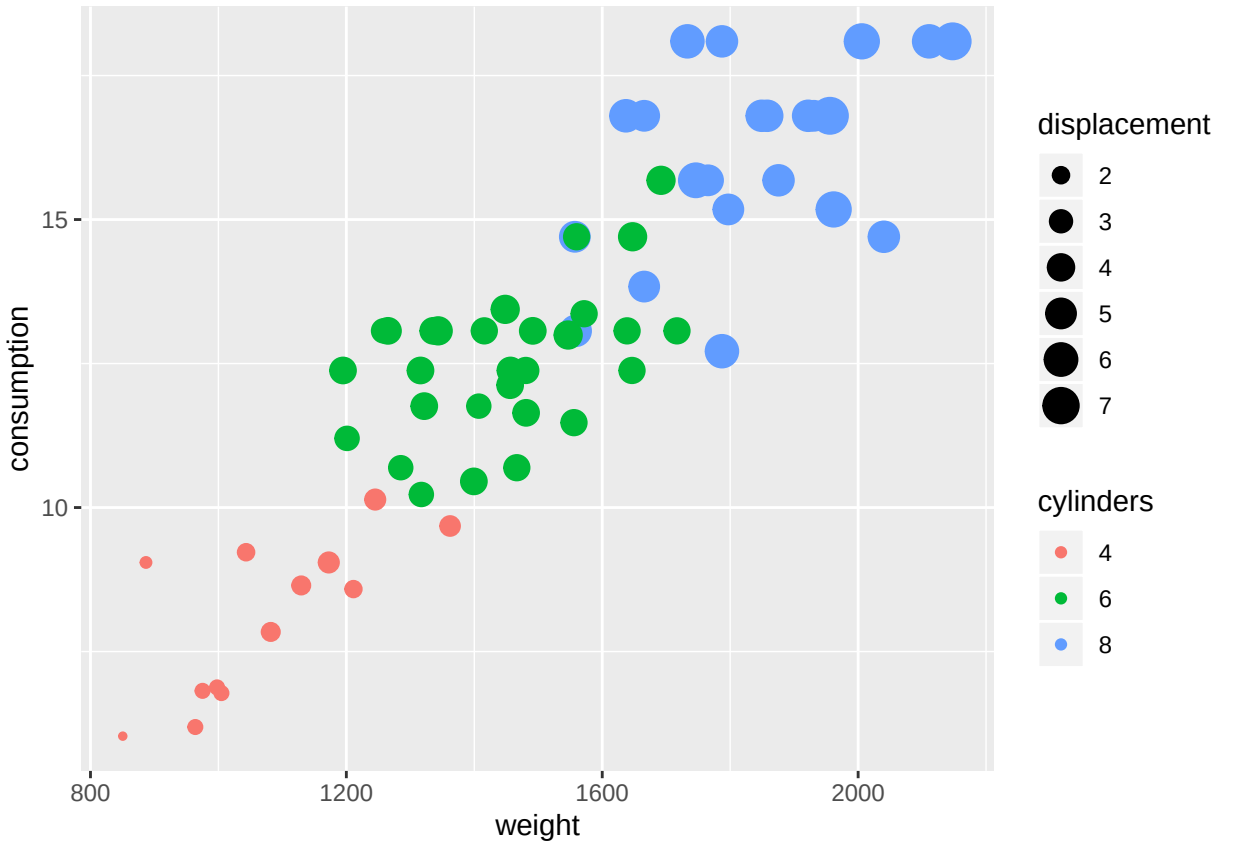
Otázka č.6: Pro kombinaci faktorizovaných proměnných cylinders a origin vykreslete spotřebu aut, aby bylo na obrázku vidět, jestli se liší spotřeba u aut pocházejících z různých kontinentů v závislosti na počtu válců a naopak.

```
p <- ggplot(data_mpghp,aes(y=consumption))+
  geom_point(aes(x = origin,colour=cylinders))+
  scale_x_discrete(labels=c("1" = "USA", "2" = "Evropa","3" = "Japonsko"))
print(p)
```



Otázka č.7: Pro auta výrobce Chrysler vykreslete závislost spotřeby na váze automobilu, kde jednotlivé události označíte barvou podle počtu válců a velikosti bodů v grafu bude odpovídat objemu motoru.

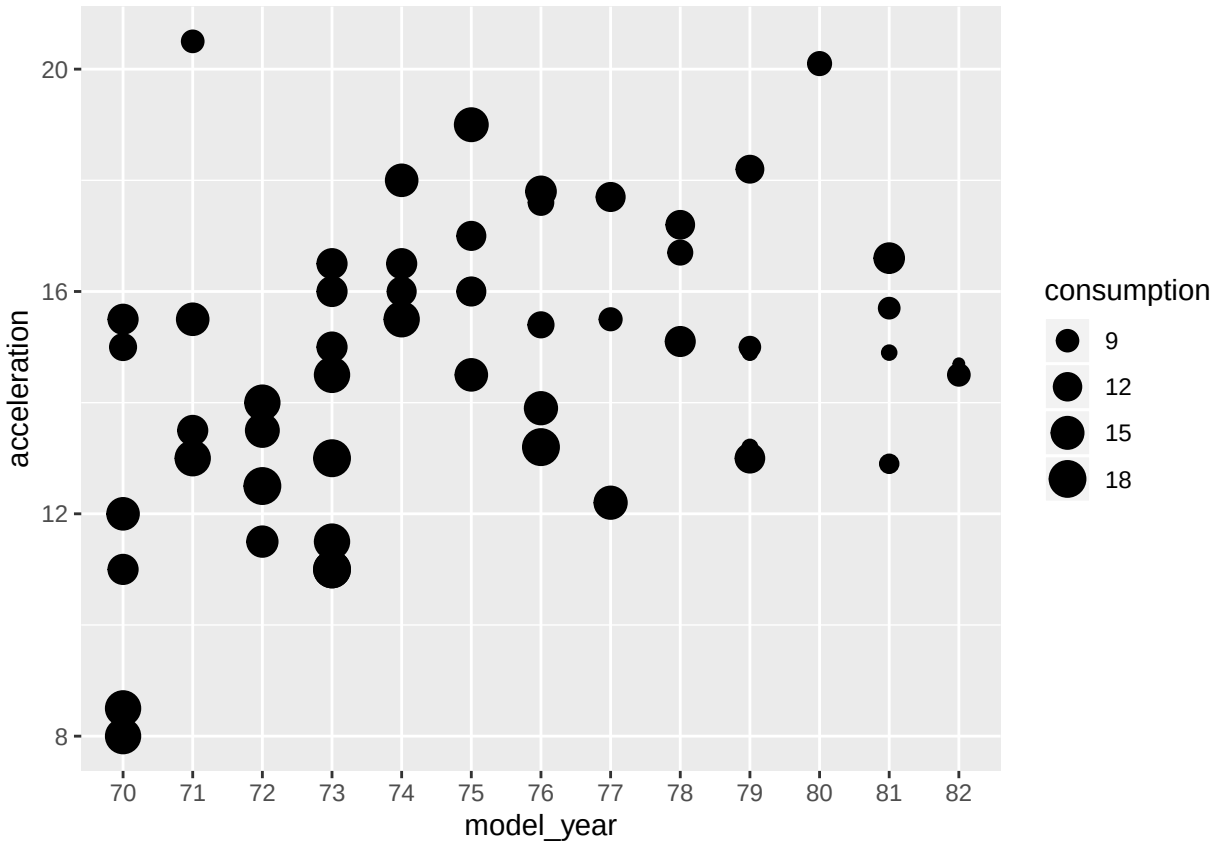
```
data_chrysler = data_mpghp[data_mpghp[,'producer']=='chrysler',]
p <- ggplot(data_chrysler,aes(x=weight,y=consumption))+
  geom_point(aes(size = displacement, colour = cylinders))
print(p)
```



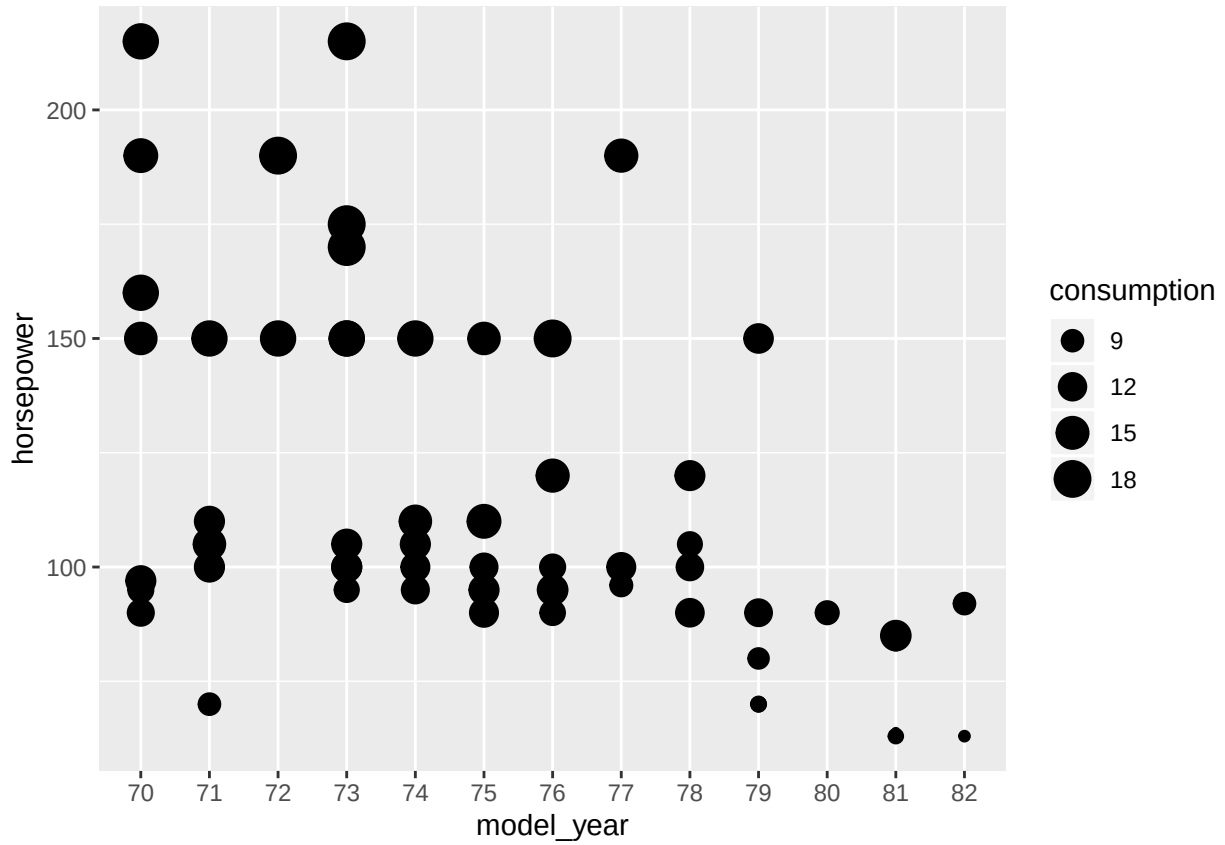
Otázka č.8: Navrhněte další zobrazení datového souboru. Proveďte ho a popište jeho účel.

Už bylo zmíněno v otázce č.4, že z grafů bylo vidět snižování spotřeby v průběhu let. Bylo by proto zajímavé vykreslit jiné proměnné, např. výkon, váhu či zrychlení a podívat se, jak se tyto veličiny měnily s přibývajícím časem. Podíváme se tak na to, jestli snižování spotřeby v důsledky lidského rozvoje nemělo negativní vliv na ostatní klíčové parametry vozidla (jinak by takový vývoj byl poněkud kontraproduktivní, pokud samozřejmě nebylo cílem pouze snížit spotřebu i přes fakt, že získáme pomalejší, méně schopné vozidlo). Tyto veličiny bychom měli vykreslovat pro jednotlivé producery zvlášť. Zkusíme např. auta značky chrysler.

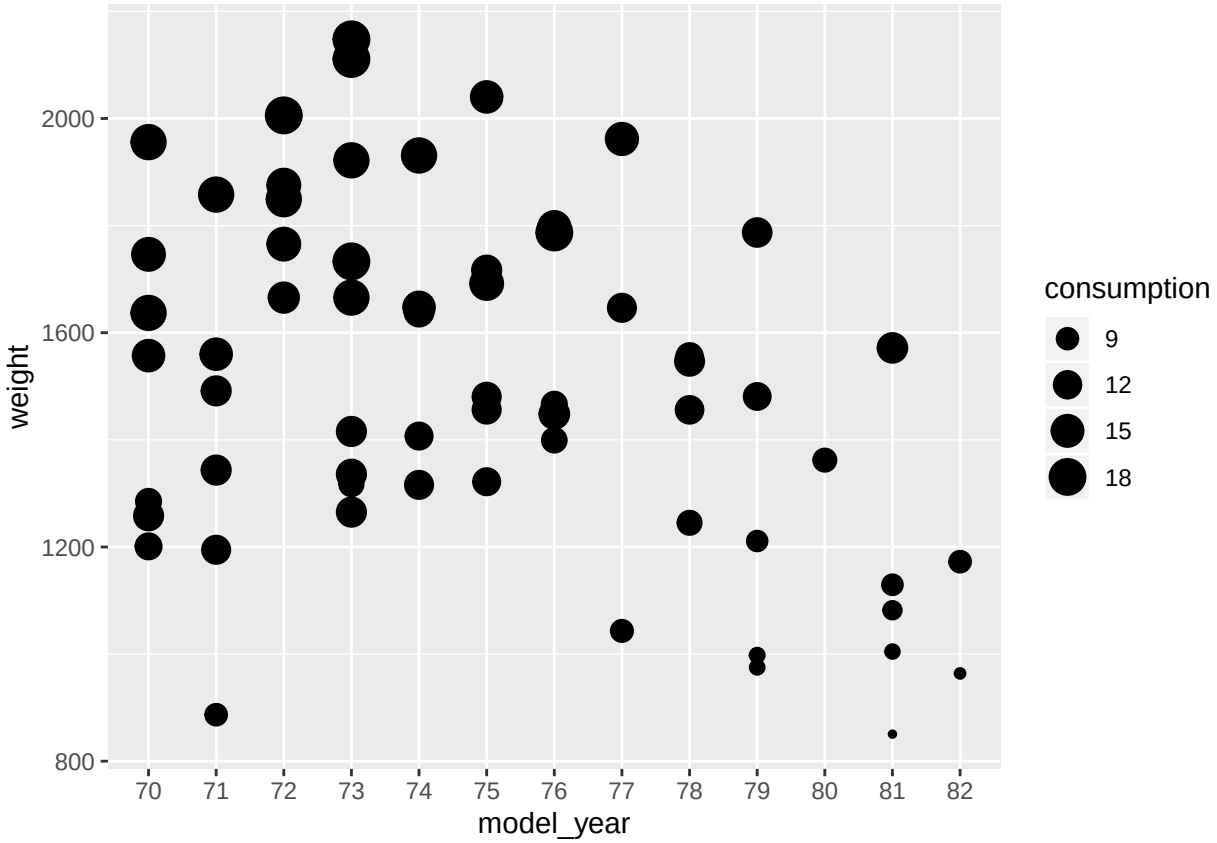
```
p1 <- ggplot(data_chrysler,aes(x=model_year,y=acceleration))+
  geom_point(aes(size = consumption))
print(p1)
```




```
p2 <- ggplot(data_chrysler,aes(x=model_year,y=horsepower))+
  geom_point(aes(size = consumption))
print(p2)
```




```
p3 <- ggplot(data_chrysler,aes(x=model_year,y=weight))+
  geom_point(aes(size = consumption))
print(p3)
```



Vidíme, že spotřeba aut se v průběhu let u této značky snižovala a s ní většinou i výkon. Naopak zrychlení se zvyšovalo. Tohle souvisí s tím, že byla tendence značky snižovat hmotnost a tím pádem i se snížením výkonu motoru zvětšit zrychlení auta.

Otázka č.9: Sestavte jednoduchý regresní model, kde vysvětlovaná proměnná bude spotřeba automobilu. Na jeho základech zjistěte, zdali spotřeba automobilu závisí na hmotnosti automobilu. Pokud ano, o kolik se změní spotřeba automobilu pokud se hmotnost zvýší o 1000kg?

Připravíme si jednoduché modely s interceptem i bez interceptu. Model bez interceptu má v tomto případě určitě smysl uvažovat (nejspíš i lepší smysl než model s interceptem), protože intuice a zkušenost říká, že auto vážící 0kg by mělo spotřebovat 0l paliva. Co se týká závislosti spotřeby na hmotnosti, je z předchozích scatterplotů zřejmé, že spotřeba na hmotnosti závisí. Ověříme si to i z hodnot t-statistik koeficientů z obou modelů.

```
lm_intercept = lm(consumption ~ weight, data_mpghp)
lm_nointercept = lm (consumption ~ weight - 1, data_mpghp)
```

Podíváme se na summary funkce obou modelů

```
summary(lm_intercept)
```

```
##
## Call:
## lm(formula = consumption ~ weight, data = data_mpghp)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -5.3975 -1.1803 -0.0406  1.0157  7.3888
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.7692790  0.3299836  -2.331   0.0202 *
## weight       0.0088833  0.0002354  37.741   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.783 on 389 degrees of freedom
## Multiple R-squared:  0.7855, Adjusted R-squared:  0.7849
## F-statistic:  1424 on 1 and 389 DF,  p-value: < 2.2e-16
```

```
summary(lm_nointercept)
```

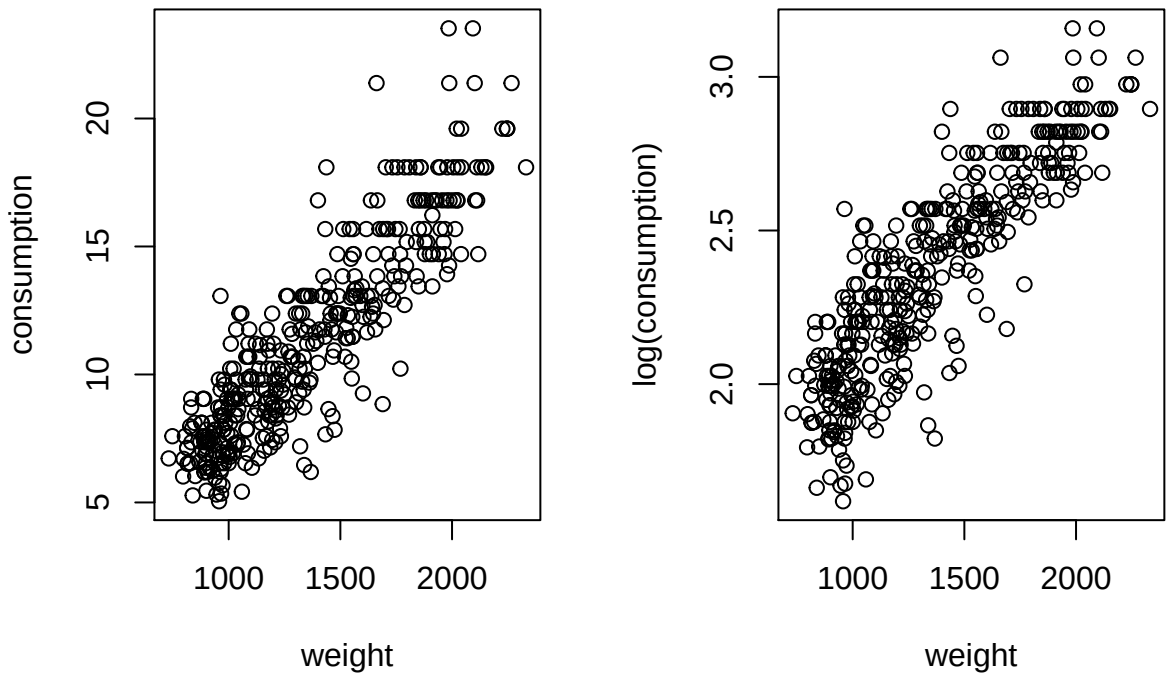
```
##
## Call:
## lm(formula = consumption ~ weight - 1, data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2750 -1.2709 -0.0896  0.9502  7.4967
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## weight 0.0083554  0.0000647   129.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.793 on 390 degrees of freedom
## Multiple R-squared:  0.9772, Adjusted R-squared:  0.9771
## F-statistic: 1.668e+04 on 1 and 390 DF,  p-value: < 2.2e-16
```

Hodnoty t-statistik jsou extrémně malé, což jakousi závislost spotřeby na váze potvrzuje. Abychom to mohli říct s větší jistotou, je potřeba první model validovat. Při změně váhy o 1000kg se v modelu s interceptem změní spotřeba (po zaokrouhlení) o 8,9. V modelu bez interceptu o 8,4.

Otázka č.10: Dá se předešlý jednoduchý regresní model zlepšit pomocí logaritmické transformace odezvy? Jak se poté změní (navýší/poklesne) spotřeba automobilu při změně hmotnosti o 1000kg? Zdůvodněte, proč případné transformace je přínosná, nebo naopak nepřínosná.

Jako první se vykreslíme scatterplot bez i s log transformovanou spotřebou a podíváme se, jestli dostaneme "hezčí" graf.

```
par(mfrow = c(1,2))
plot(data_mpghp$weight,data_mpghp$consumption,xlab = 'weight',ylab='consumption')
plot(data_mpghp$weight,log(data_mpghp$consumption),xlab = 'weight',ylab='log(consumption)')
```



Z grafů není patrné, že by log transformace měla vést k vyšší přesnosti modelu. Nastavíme lineární model. Zvolíme model s interceptem, zde není na místě používat argument (0,0), protože po získání hodnoty z modelu musíme provést zpětnou transformaci a exp(0) = 1.

```
lm_logcons = lm(log(consumption)~weight,data_mpghp)
```

podíváme se na výstup ze summary funkce pro tento model

```
summary(lm_logcons)
```

```
##
## Call:
## lm(formula = log(consumption) ~ weight, data = data_mpghp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55158 -0.09868  0.00584  0.09917  0.50637
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.323e+00  3.034e-02   43.60   <2e-16 ***
## weight      7.688e-04  2.164e-05   35.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.164 on 389 degrees of freedom
## Multiple R-squared:  0.7644, Adjusted R-squared:  0.7638
## F-statistic:  1262 on 1 and 389 DF,  p-value: < 2.2e-16
```

Z výstupu vidíme, že z hodnot R^2 statistiky můžeme usoudit, že model fituje transformovaná data přibližně stejně kvalitně jako model bez interceptu původní data. V čem tahle transformace ale mohla pomoci je zajištění validace OLS požadavků, k čemuž se často takové transformace v praxi využívají. Tímto se budeme zabývat v pozdějších otázkách. Nevýhoda této transformace je zhoršení interpretovatelnosti parametrů modelu vzhledem k původním hodnotám, jelikož se jedná o nelineární transformaci. Navíc v tomto případě nejde říct přesně, jaká bude změna spotřeby při změně váhy o 1000kg, protože ta se mění nekonstantně a závisí na počáteční hmotnosti, od které změnu měříme. Můžeme uvést např. změnu spotřeby při změne váhy z 1000kg na 2000kg, tj.

```
c <- exp(predict(lm_logcons,data.frame(weight = 2000))) - exp(predict(lm_logcons,data.frame(weight = 100
print(c)
```

```
##        1
## 9.374201
```

Z vykreslených grafů se zdá, že by mohla navíc k log transformaci pomoci ještě nějaká mocninná transformace, tím bychom mohli graf lépe "vyrovnat", pro zajímavost zkusíme druhou mocninnu.

```
lm_logcons_squared = lm((log(consumption))^2~weight,data_mpghp)
```

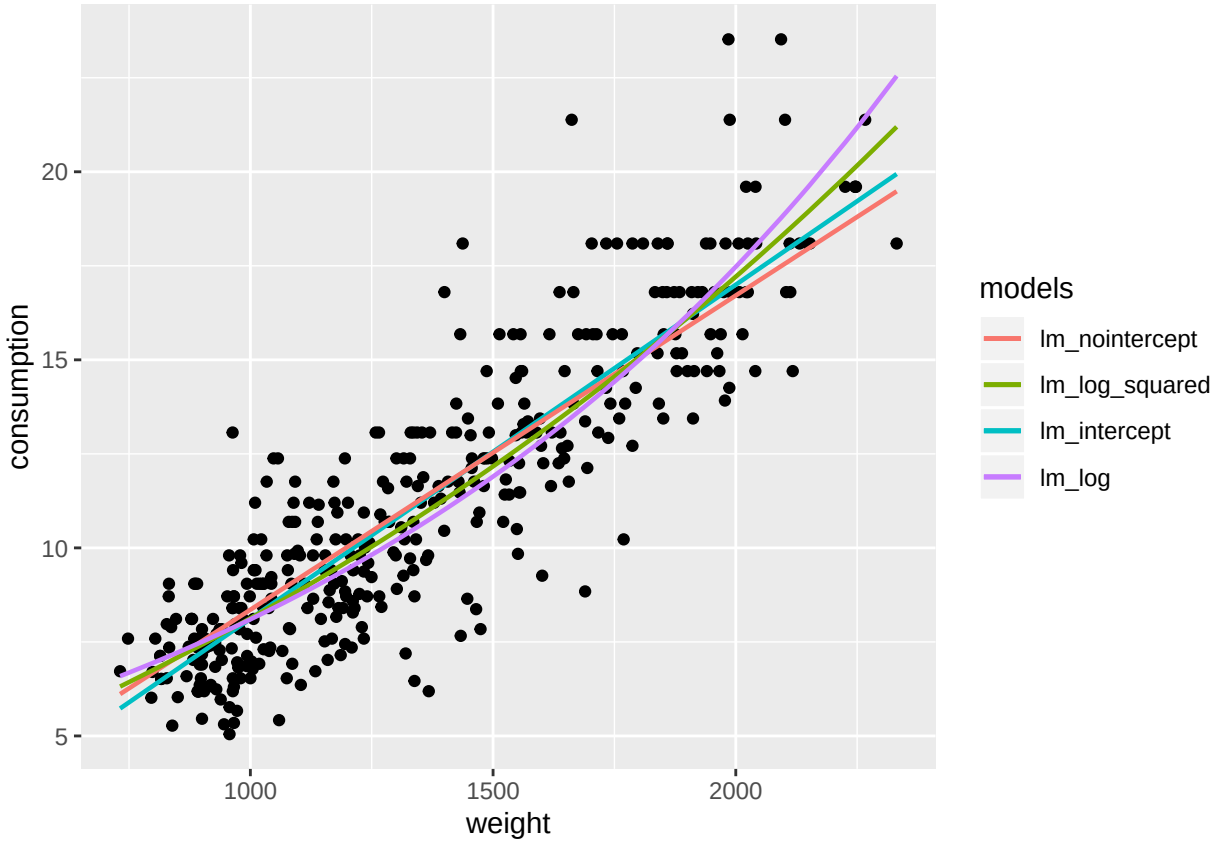
Podíváme se na summary funkci.

```
summary(lm_logcons_squared)
```

```
##
## Call:
## lm(formula = (log(consumption))^2 ~ weight, data = data_mpghp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43002 -0.50025  0.00745  0.45431  2.53535
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.839e-01  1.394e-01   4.905 1.37e-06 ***
## weight      3.707e-03  9.946e-05  37.268  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7536 on 389 degrees of freedom
## Multiple R-squared:  0.7812, Adjusted R-squared:  0.7806
## F-statistic:  1389 on 1 and 389 DF,  p-value: < 2.2e-16
```

Proti modelu s log transformací vyšla vyšší R^2 statistika, což je pozitivní. Validace modelu nechám na později. Ještě si křivky vykreslíme spolu se scatterplotem.

```
p <- ggplot(data_mpghp, aes(x=weight, y=consumption)) + geom_point() +
  geom_line(aes(x = weight, y = lm_intercept$fitted.values, col="red"),size=0.8)+
    geom_line(aes(x = weight, y = lm_nointercept$fitted.values, col="blue"),size=0.8)+
  geom_line(aes(x = weight, y = exp(sqrt(lm_logcons_squared$fitted.values)),col="green") ,size = 0.8)+
  geom_line(aes(x = weight, y = exp(lm_logcons$fitted.values),col="yellow"),size = 0.8)+
   scale_color_discrete(name = 'models',labels = c('lm_nointercept','lm_log_squared','lm_intercept','lm
plot(p)
```



Otázka č.11: Vyzkoušejte transformovat nezávislou proměnnou hmotnost. Vyzkoušejte např. po částech konstantní transformaci a polynomiální transformace (kvadratickou a kubickou).

Jako první provedeme po částech konstantní transformaci hmotnosti, intervaly zvolíme po 100kg a v každém intervalu bude hodnota zvolena jako střední hodnota původních hmotností spadajících do tohoto intervalu.

```
breaks = seq(from = min(data_mpghp$weight), to = max(data_mpghp$weight)+100, by = 100)
data_mpghp$constant_weight2 = cut(data_mpghp$weight, breaks = breaks, include.lowest = TRUE, right = FA
data_mpghp$constant_weight = replicate(length(data_mpghp$constant_weight2),0)
for (i in levels(factor(data_mpghp$constant_weight2)))
{
  temp = mean(data_mpghp[data_mpghp$constant_weight2 == i,'weight'])
  data_mpghp[data_mpghp$constant_weight2 == i,'constant_weight'] = temp
}
```

Ve sloupci constant_weight jsou nyní hodnoty po částech konstantní transformace.

Vytvoříme příslušné modely.

```
lm_const = lm(consumption ~ constant_weight,data_mpghp)
lm_quadratic = lm(consumption ~ I(weight^2),data_mpghp)
lm_cubic = lm(consumption ~ I(weight^3),data_mpghp)
```

Modely si vypíšeme pomocí summary funkce.

```
summary(lm_const)
```

```
##
## Call:
## lm(formula = consumption ~ constant_weight, data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2580 -1.0896  0.0333  1.0209  7.2825
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.7836018  0.3326671  -2.356    0.019 *
## constant_weight  0.0088939  0.0002373  37.474   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.793 on 389 degrees of freedom
## Multiple R-squared:  0.7831, Adjusted R-squared:  0.7825
## F-statistic:  1404 on 1 and 389 DF,  p-value: < 2.2e-16
```

```
summary(lm_quadratic)
```

```
##
## Call:
## lm(formula = consumption ~ I(weight^2), data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0704 -1.1339 -0.1103  1.0064  7.7518
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.238e+00  1.833e-01   28.58   <2e-16 ***
## I(weight^2) 3.039e-06  8.105e-08   37.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.793 on 389 degrees of freedom
## Multiple R-squared:  0.7833, Adjusted R-squared:  0.7827
## F-statistic:  1406 on 1 and 389 DF,  p-value: < 2.2e-16
```

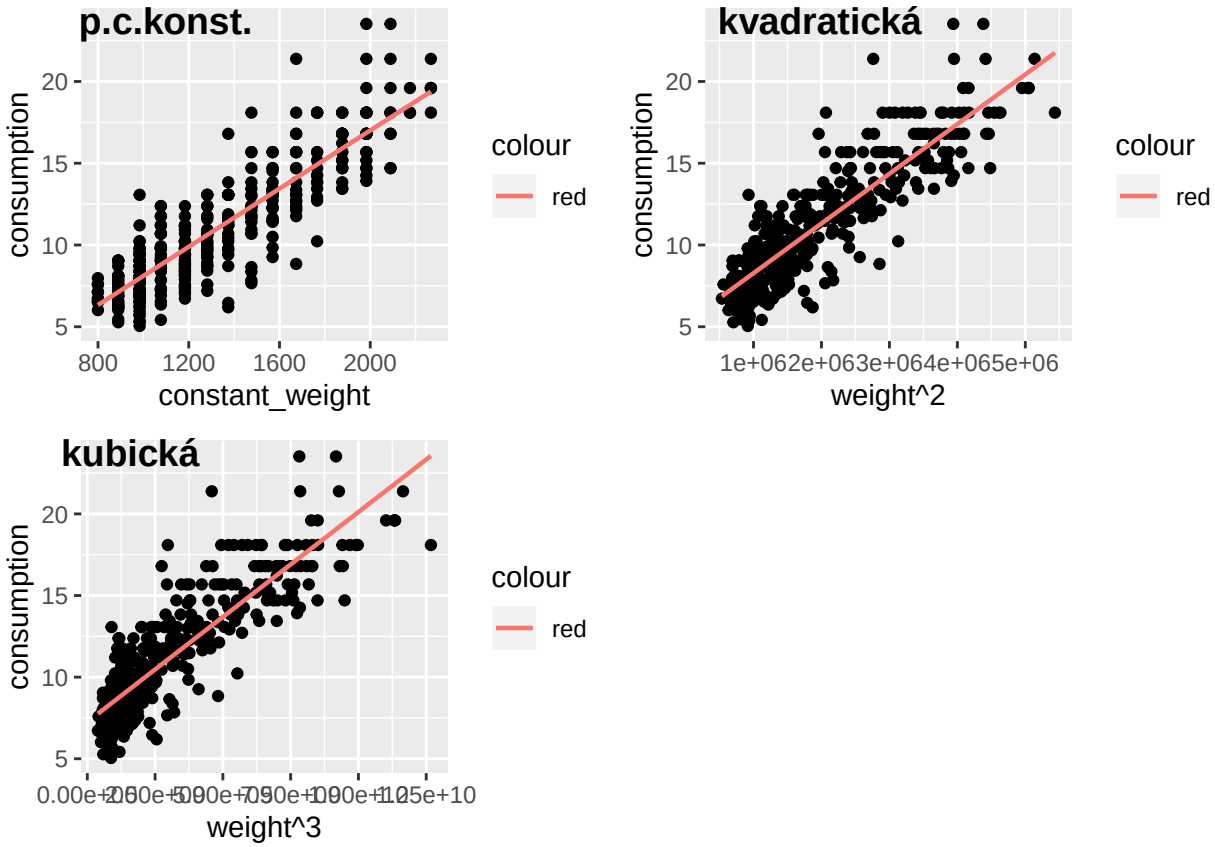
```
summary(lm_cubic)
```

```
##
## Call:
## lm(formula = consumption ~ I(weight^3), data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4590 -1.2584 -0.1314  0.9637  8.2247
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.255e+00  1.486e-01   48.84   <2e-16 ***
## I(weight^3) 1.286e-09  3.689e-11   34.86   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.896 on 389 degrees of freedom
## Multiple R-squared:  0.7575, Adjusted R-squared:  0.7569
## F-statistic:  1215 on 1 and 389 DF,  p-value: < 2.2e-16
```

Nyní si vykreslíme scatterploty s proložením příslušnou křivkou.

```
p_const <- ggplot(data_mpghp, aes(x=constant_weight, y=consumption)) + geom_point()+
  geom_line(aes(x = constant_weight, y = lm_const$fitted.values ,colour = 'red'),size=0.8)


p_quadratic <- ggplot(data_mpghp, aes(x=weight^2, y=consumption)) + geom_point()+
  geom_line(aes(x = weight^2, y = lm_quadratic$fitted.values ,colour = 'red'),size=0.8)


p_cubic <- ggplot(data_mpghp, aes(x=weight^3, y=consumption)) + geom_point()+
  geom_line(aes(x = weight^3, y = lm_cubic$fitted.values ,colour = 'red'),size=0.8)
figure <- ggarrange(p_const, p_quadratic, p_cubic,
                    labels = c("p.č.konst.", "kvadratická", "kubická"),
                    ncol = 2, nrow = 2)
plot(figure)
```

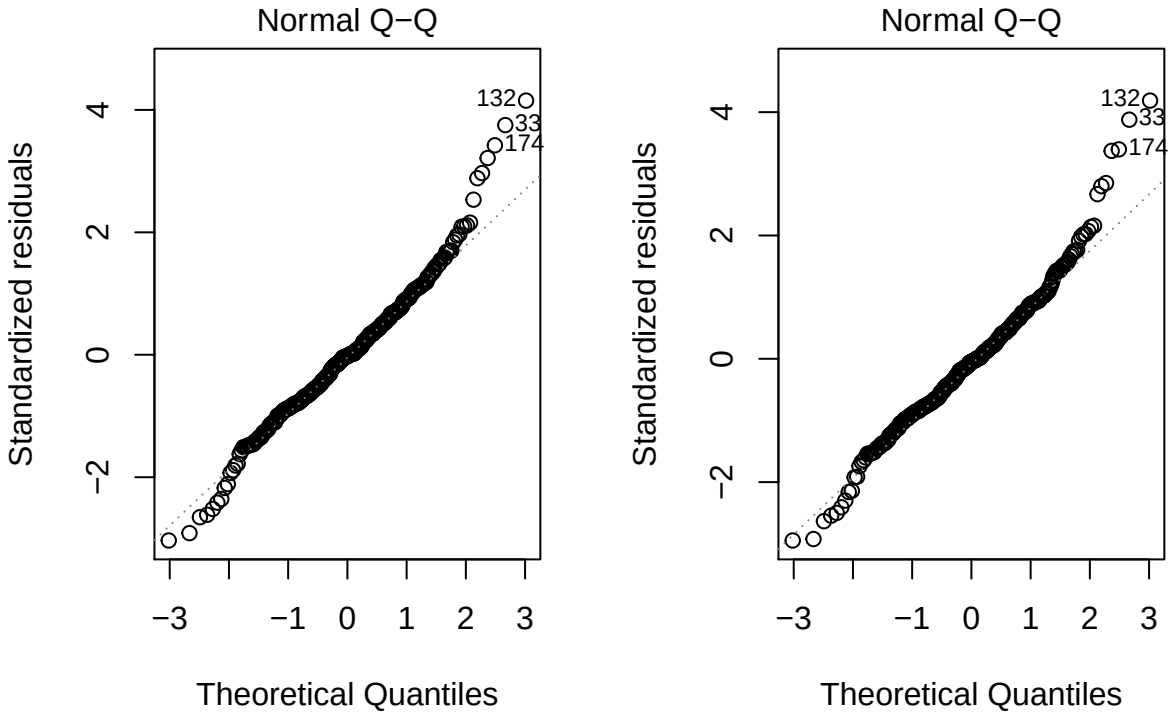


Otázka č.12: Pro vámi vybraný finální model ověřte předpoklady pro použití OLS, diskutujte výstup a model validujte pomocí příslušných testů na rezidua a pomocí příslušných obrázků (QQplot, residua vs fitted, residua vs regressors atd.)

Potřebujeme ověřit: 1. nezávilost a homoskedasticitu reziduí se střední hodnotou 0, což můžeme ověřit např. z residuals vs fitted plotu (nebo scale-location plot), homoskedasticitu můžeme ještě ověřit např. pomocí breusch-paganova testu (bptest) 2. normalitu reziduí pomocí QQ plotu a např. sapirova testu

Jako první ověříme klasický model bez transformace s i bez interceptu. Začneme normalitou. Vykreslíme QQplot.

```
par(mfrow = c(1,2))
plot(lm_intercept,which=2)
plot(lm_nointercept,which=2)
```



Ty chvosty nevypadají příliš dobře, podíváme se na výsledek shapirova testu, uvedeme i model bez interceptu.

```
shapiro.test(residuals(lm_intercept))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_intercept)
## W = 0.97937, p-value = 2.255e-05
```

```
shapiro.test(residuals(lm_nointercept))
```

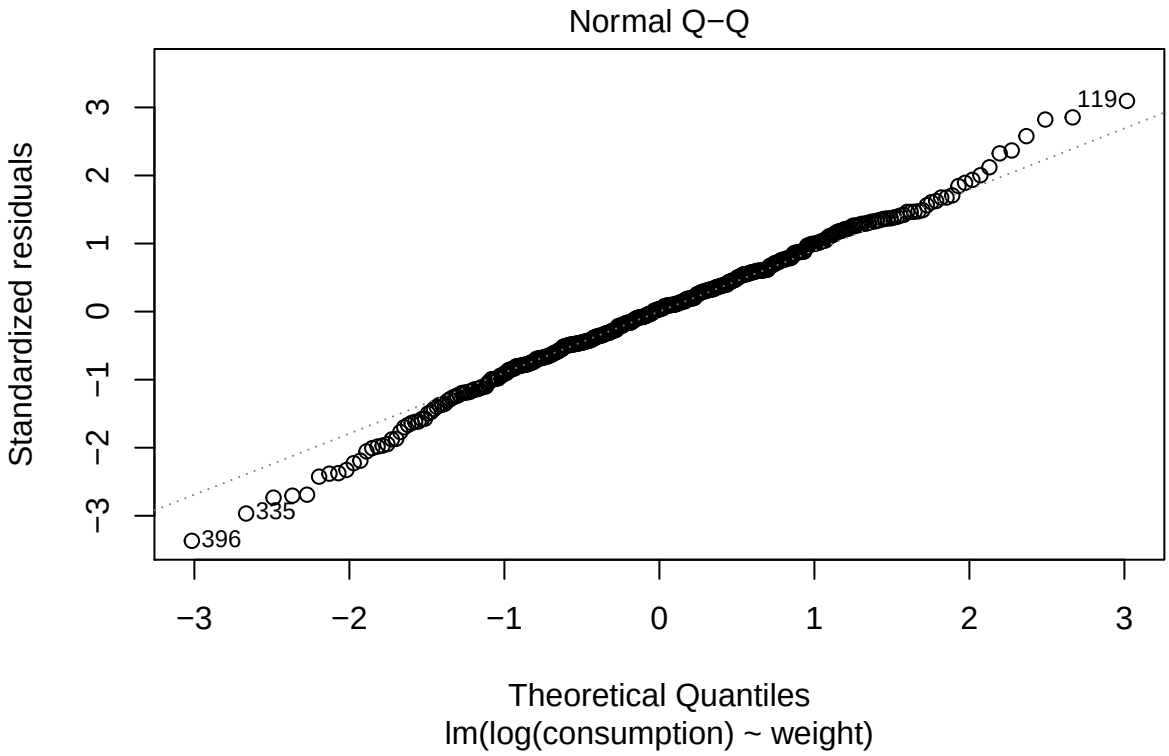
```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_nointercept)
## W = 0.97578, p-value = 4.081e-06
```

Obě p-hodnoty jsou kriticky nízké, tedy test na normalitu neprošel. Zkusíme model s log transformovanou spotřebou.

Začneme normalitou.

```
plot(lm_logcons,which=2)
```



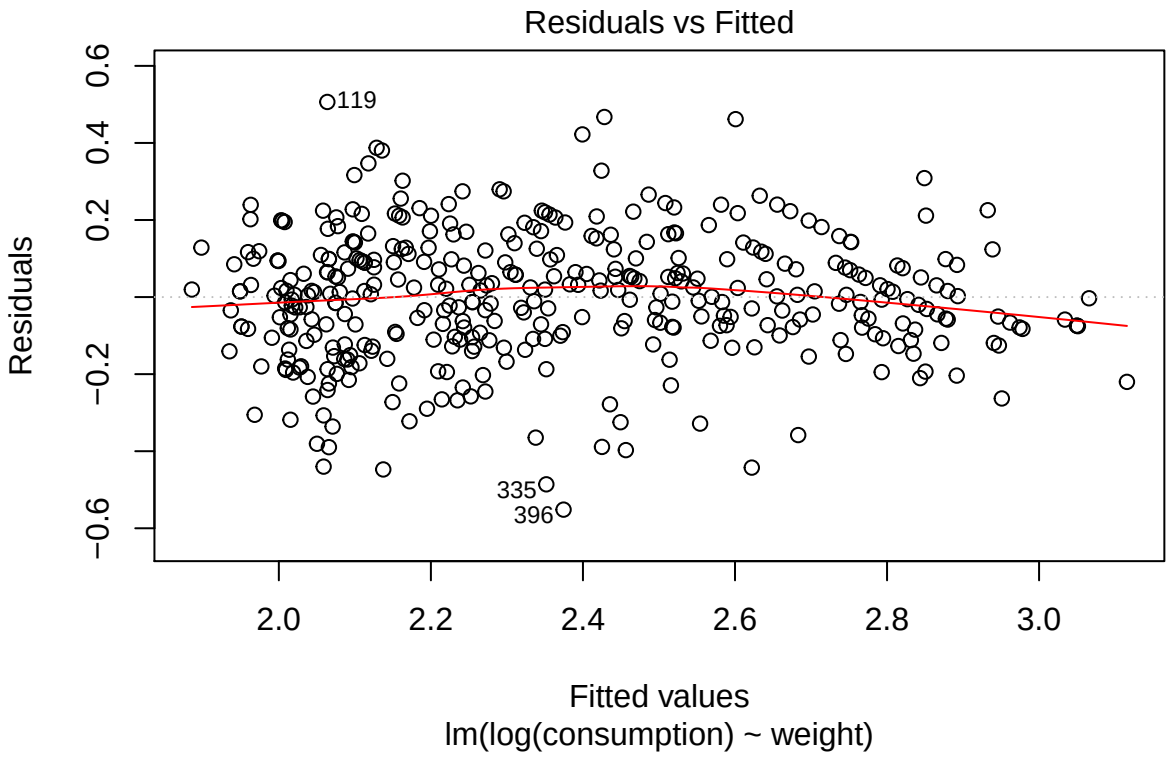
To už vypadá o něco lépe.

```
shapiro.test(residuals(lm_logcons))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_logcons)
## W = 0.99329, p-value = 0.07945
```

Shapirův test normalitu reziduí nezamítá. Podíváme se na homoskedasticitu reziduí.

```
plot(lm_logcons,which=1)
```



Residuals vs fitted plot vypadá hezky. Podíváme se na bptest.

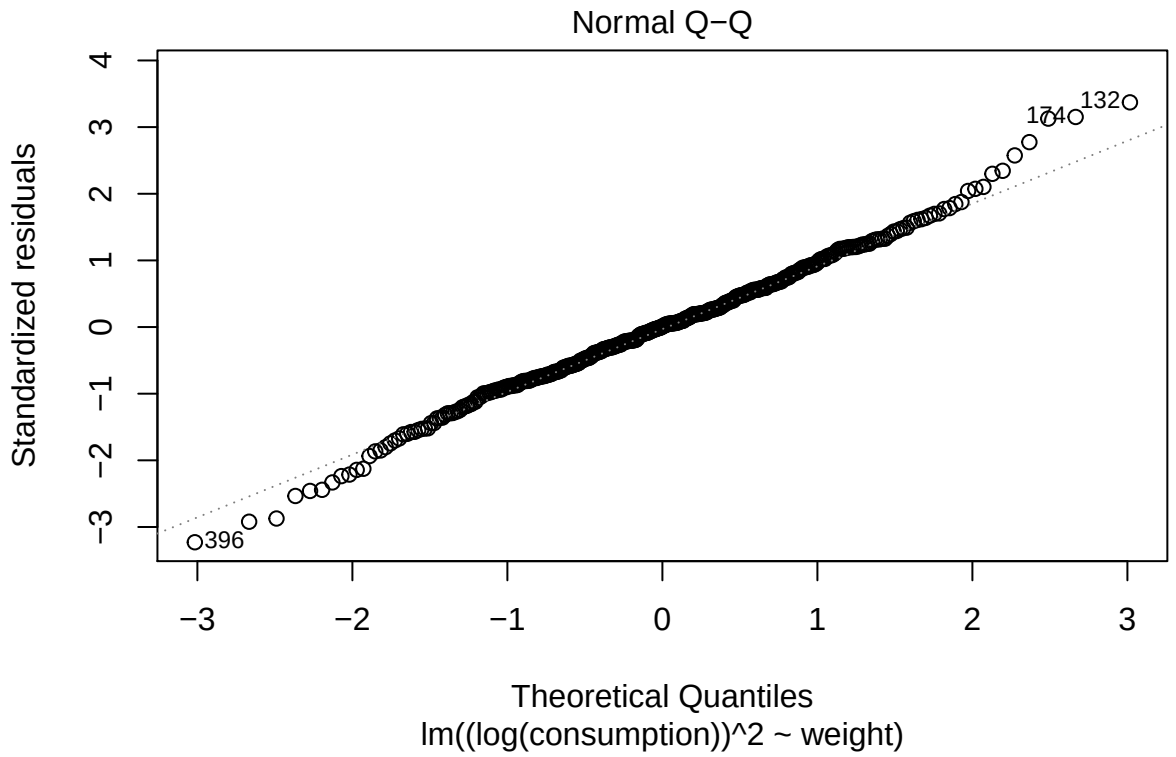
```
bptest(lm_logcons)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_logcons
## BP = 3.2404, df = 1, p-value = 0.07184
```

Zde vyšla p-hodnota nad 0.05, tedy homoskedasticitu nezamítáme. Tento model na OLS požadavky tedy prošel.

Zkusím ještě model lm_logcons_squared.

```
plot(lm_logcons_squared,which=2)
```

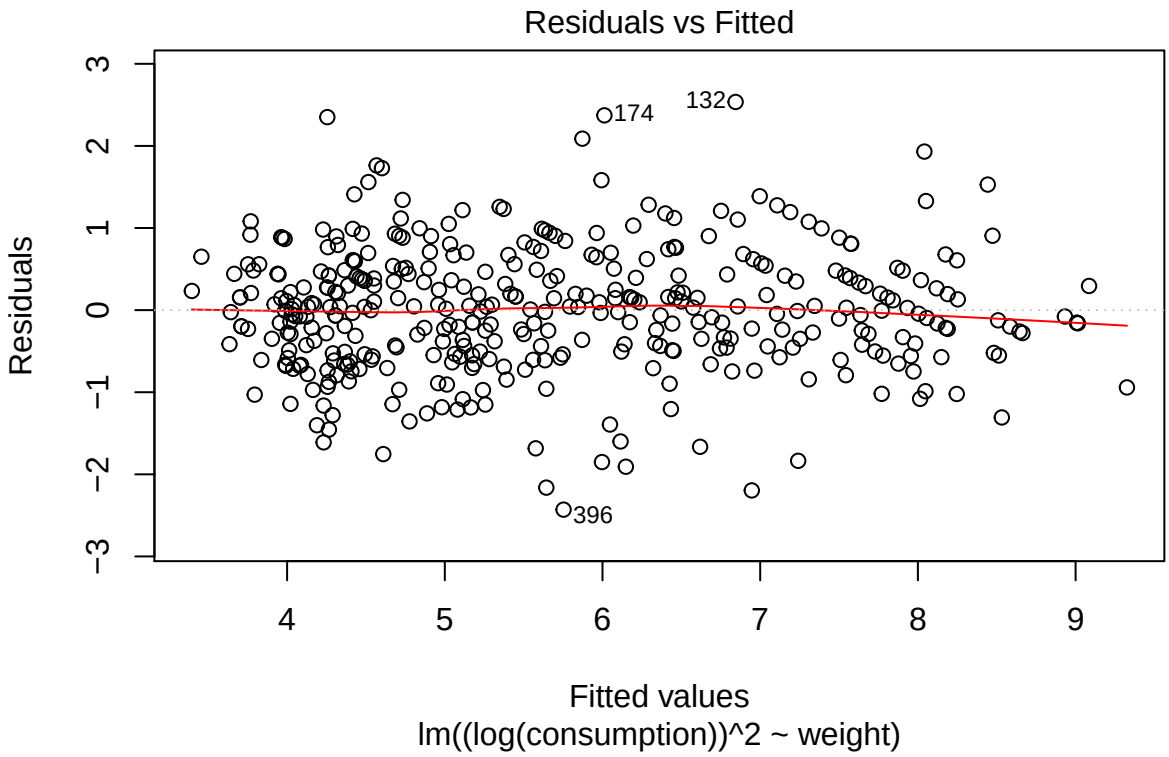



```
shapiro.test(residuals(lm_logcons_squared))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_logcons_squared)
## W = 0.99341, p-value = 0.08565
```

Zde vyšla p-hodnota dokonce o něco vyšší než u lm_logcons.

```
plot(lm_logcons_squared,which=1)
```



A residuals vs fitted plot vypadá taky o něco lépe než v předchozím případě.

```
bptest(lm_logcons_squared)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_logcons_squared
## BP = 0.10966, df = 1, p-value = 0.7405
```

Homoskedasticita je zde daleko lepší než v předchozím případě, což nám hezky potvrzuje i bptest.

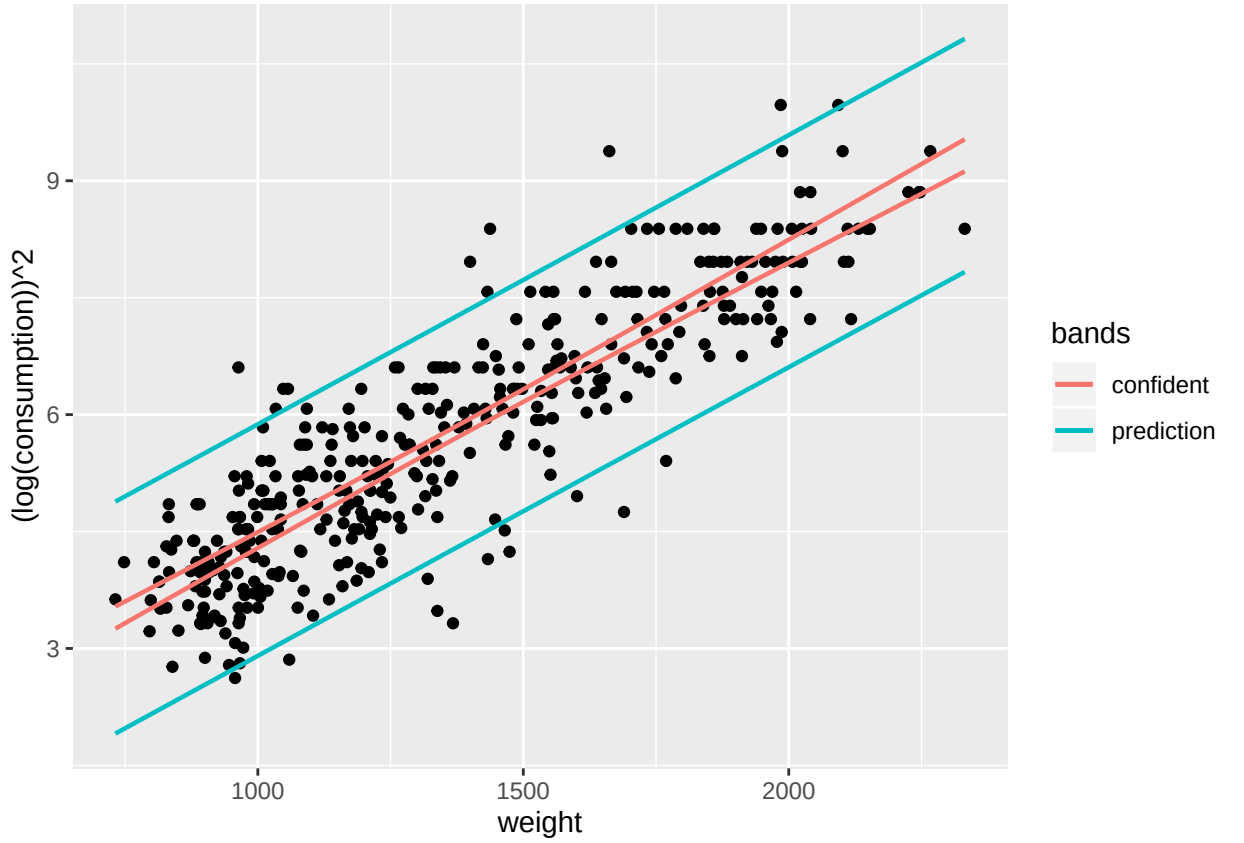
Při porovnání modelů log_cons a log_cons_squared se jeví jako lepší druhá varianta i vzhledem k R^2 statistice.

Otázka č.13: Vykreslete scatterplot skutečných spotřeb aut a hmotností a na základě vybraného modelu. Proložte skrze data odhadnutou regresní přímku a vykreslete 95% konfidenční intervaly, jak pro predikované hodnoty, tak pro regresní přímku (tzv. Confidence a Prediction band). Porovnejte s výsledkem z funkce plot(allEfects(model)).

Vybral jsem model lm_logcons_squared. První vykreslíme obrázky pro transformovanou spotřebu.

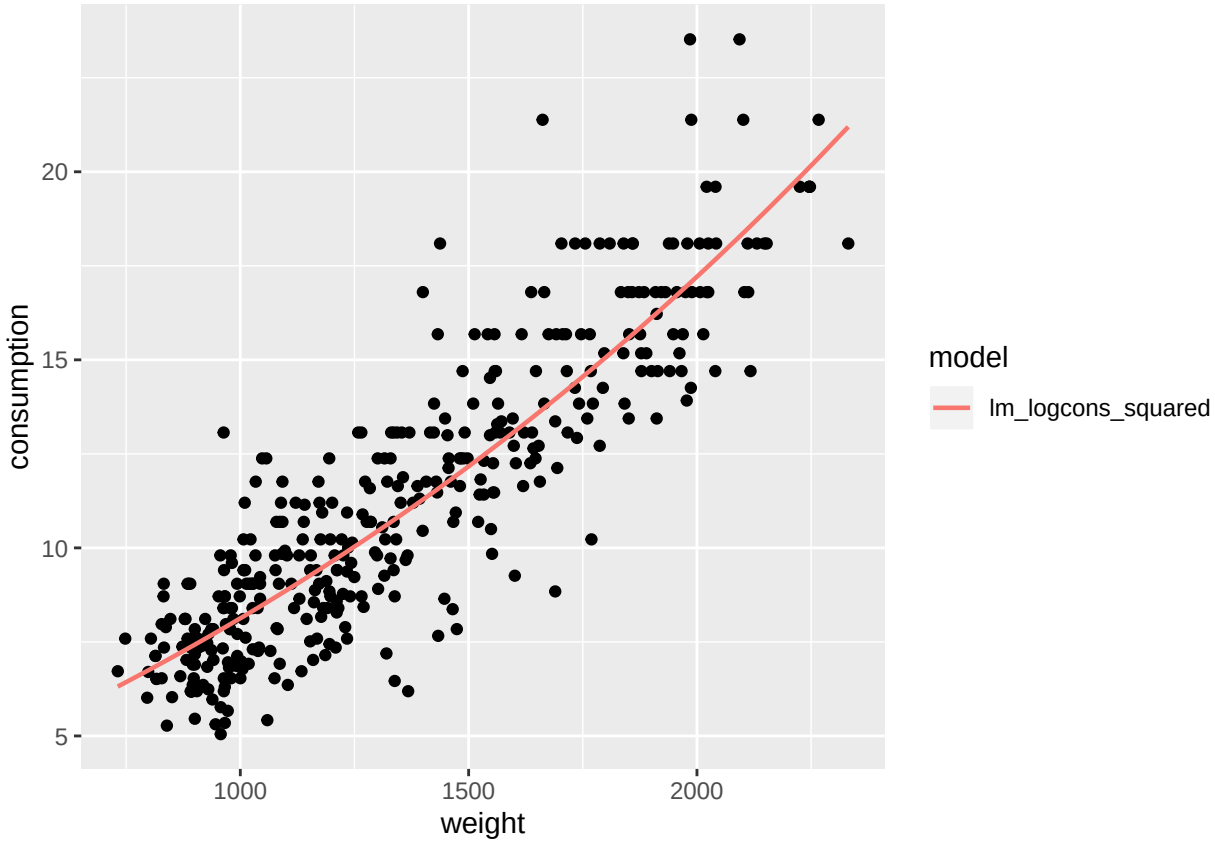
```
pred_transf = predict(lm_logcons_squared,data_mpghp,interval = 'prediction')
conf_transf = predict(lm_logcons_squared,data_mpghp,interval = 'confidence')

data <- cbind(data_mpghp, pred_transf)
conf_transf = data.frame(conf_transf)
```

```
conf_transf$weight = data_mpghp$weight

# vykreslíme
p1 <- ggplot(data, aes(x=weight, y=(log(consumption))^2)) + geom_point()+
  geom_line(aes(y = lwr ,colour = 'red'),size=0.8) +
  geom_line(aes(y = upr ,colour = 'red'),size=0.8) +
  geom_line(data=conf_transf,aes(x=weight,y = lwr ,colour = 'blue'),size=0.8) +
  geom_line(data=conf_transf,aes(x=weight,y = upr ,colour = 'blue'),size=0.8)+
  scale_color_discrete(name = 'bands',labels = c('confident','prediction'))
plot(p1)
```



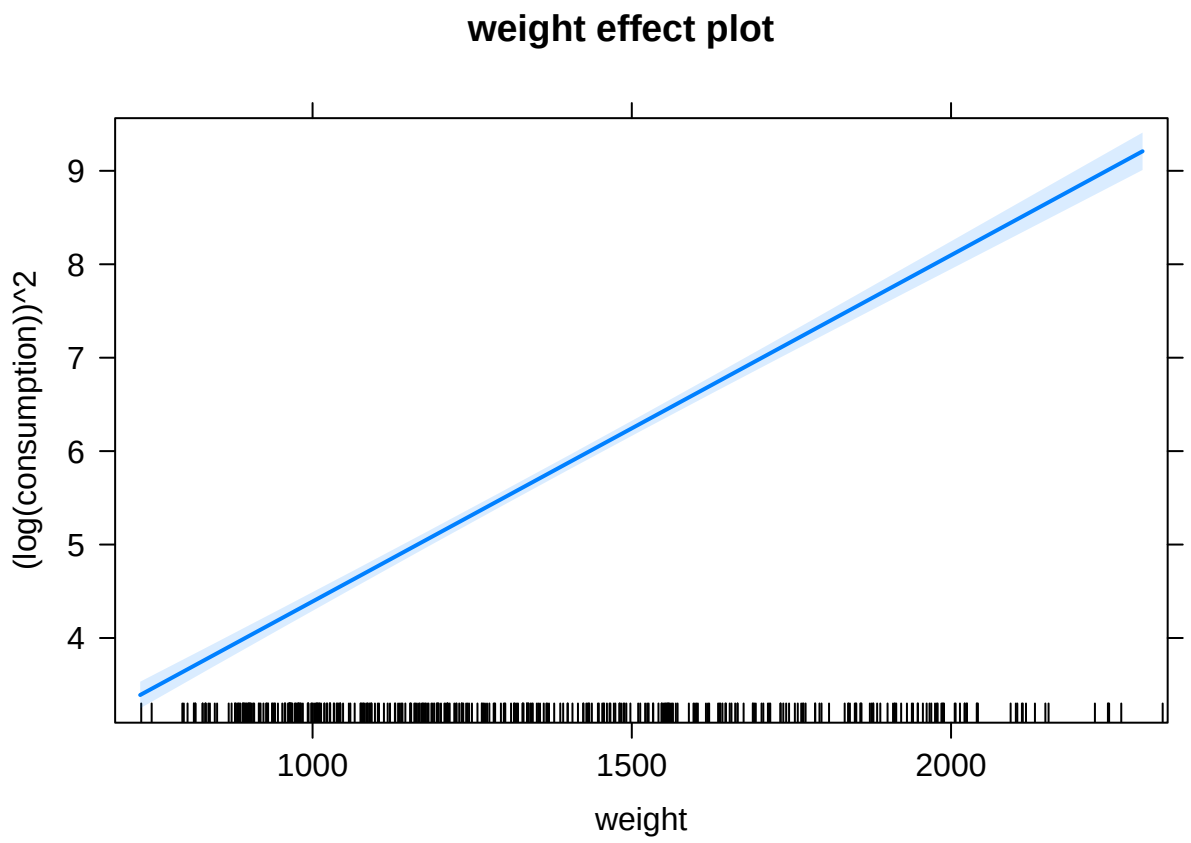
Nyní vykreslíme netransformovanou spotřebu a proložíme křivkou. Nejsem si jistý, jestli je interpretačně korektní transformovat confident a prediction bandy stejně jako hodnoty, raději to dělat nebudu.

```
p <- ggplot(data, aes(x=weight, y=consumption)) + geom_point()+
  geom_line(aes(x=weight,y=exp(sqrt(lm_logcons_squared$fitted.values)), colour = "red"),size = 0.8) +
  scale_color_discrete(name = 'model',labels = c('lm_logcons_squared'))

plot(p)
```



Nyní zkusíme vykreslit výstup z allEffects.

```
library(effects)
plot(allEffects(lm_logcons_squared))
```



Plot vypadá stejně jako náš. Confident band souhlasí.

Otázka č.14: Přidejte k vysvětlujícím proměnným i proměnnou origin, navrhněte aditivní lineární model, a ve scatterplotu vykreslete 3 skupiny různými barvami a data proložte odpovídajícími regresními přímkami.

Vyrobíme aditivní model.

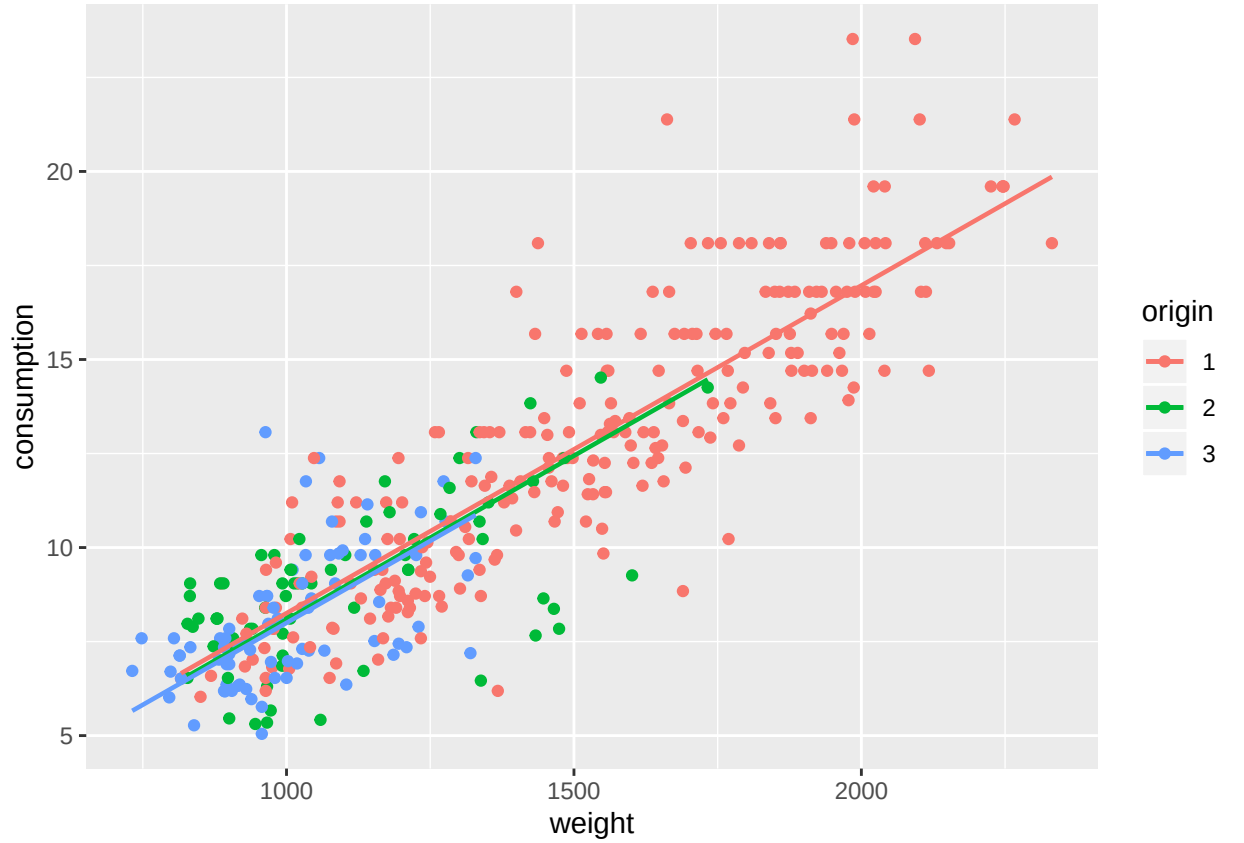
```
lm_weight_origin<-lm(consumption~weight+origin,data_mpghp)
summary(lm_weight_origin)
```

```
##
## Call:
## lm(formula = consumption ~ weight + origin, data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4211 -1.1820 -0.0504  1.0736  7.3605
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.4579487  0.4665404  -0.982    0.327
## weight       0.0087130  0.0002962  29.419   <2e-16 ***
## origin2     -0.1705275  0.2751285  -0.620    0.536
## origin3     -0.2574894  0.2777218  -0.927    0.354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.786 on 387 degrees of freedom
## Multiple R-squared:  0.786,  Adjusted R-squared:  0.7843
## F-statistic: 473.8 on 3 and 387 DF,  p-value: < 2.2e-16
```

Data rozdělíme podle origin a vykreslíme i s regresními přímkami.

```
data_split_origin  <- split(data_mpghp, data_mpghp$origin)
data_split_origin[['1']]$Fit <- predict(lm_weight_origin, data_split_origin[['1']])
data_split_origin[['2']]$Fit  <- predict(lm_weight_origin, data_split_origin[['2']])
data_split_origin[['3']]$Fit  <- predict(lm_weight_origin, data_split_origin[['3']])

p <- ggplot(data_mpghp, aes(x=weight, y=consumption, color = origin))+geom_point()+
  geom_line(data = data_split_origin[['1']], aes(x=weight,y=Fit), size = 0.8)+
  geom_line(data = data_split_origin[['2']], aes(x=weight,y=Fit), size = 0.8)+
  geom_line(data = data_split_origin[['3']], aes(x=weight,y=Fit), size = 0.8)

plot(p)
```



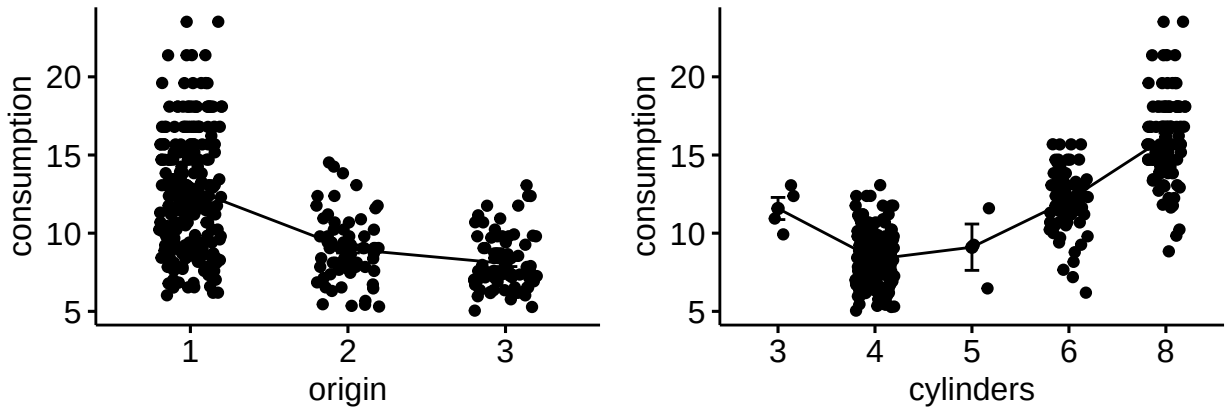
Vidíme, že regresní přímky z aditivního modelu pro jednotlivé země jsou velmi podobné. Jde vidět, že origin v tomto modelu příliš spotřebu neovlivňuje.

Otázka č.15: Porovnejte pomocí vhodného statistického testu shodnost středních hodnot spotřeby cen u automobilů pocházejících z různých kontinentů a podle počtu válců (faktorizovaná proměnná). Zdůvodněte zdali tyto statistické testy jsou vypovídající a zdali lze z nich určit důležitost daných proměnných pro predikci ceny automobilu.

Jako statistický test použijeme rozšíření t-testu, tzv. one-way anova test. Je alternativou t-testu, pokud máme více než 2 faktory. Tento test předpokládá normalitu dat v každém faktoru a stejné rozptyly napříc faktory (mělo by se ověřit). Pokud bychom nesplnili tyto požadavky, lze použít jako alternativu Kruskalův test.

Vykreslíme obrázky zvlášť pro origin a pro cylinders.

```
library("ggpubr")
p1 <- ggline(data_mpghp, x = 'origin', y = 'consumption',
             add = c("mean_se", "jitter"),
             order = c("1", "2", "3"))


p2 <- ggline(data_mpghp, x = 'cylinders', y = 'consumption',
             add = c("mean_se", "jitter"))

figure <- ggarrange(p1, p2,
                    ncol = 2, nrow = 2)
plot(figure)
```



Z obrázku vidíme, že střední hodnoty nabývají na první pohled různých hodnot mezi faktory. Podíváme se

na výsledky testů.

```
summary(aov(consumption~origin,data_mpghp))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## origin        2   1773   886.4   86.11 <2e-16 ***
## Residuals   388   3994    10.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
kruskal.test(consumption~origin,data_mpghp)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  consumption by origin
## Kruskal-Wallis chi-squared = 131.42, df = 2, p-value < 2.2e-16
```

```
summary(aov(consumption~cylinders,data_mpghp))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## cylinders     4   4210    1052   260.8 <2e-16 ***
## Residuals   386   1558       4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
kruskal.test(consumption~cylinders,data_mpghp)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  consumption by cylinders
## Kruskal-Wallis chi-squared = 279.14, df = 4, p-value < 2.2e-16
```

V obou případech vidíme velmi nízkou p-hodnotu, která znamená, že zamítáme nulovou hypotézu, tedy rovnost středních hodnot. Kruskalův test to pro origin i cylinders potvrzuje.

Pokud by nastala situace, že by se střední hodnoty consumption rovnali pro nějakou faktorovou proměnnou pro každý faktor, pak si myslím, že daná faktorová proměnná nebude mít příliš velký vliv na vysvětlovanou proměnnou (zanedbatelný či žádný vliv), kvůli rovnostem středních hodnot se jednotlivé hodnoty v různých faktorech budou 'dost podobat' a tím pádem v podstatě různé faktory nebudou mít vliv na změnu spotřeby.

Můžeme tuto myšlenku demonstrovat na příkladu. Uměle vytvoříme 3-faktorovou proměnnou, kde se v každém faktoru bude rovnat střední hodnota vysvětlované veličiny (generovaná z normálního rozdělení). Následně uděláme stejný experiment, jen s tím rozdílem, že se střední hodnoty rovnat nebudou.

```
x <- seq(-20, 20, by = .1)
x_1 <- rep(1,length(x))
y_1 <- rnorm(x,mean=0,sd=1)


x_2 <- rep(2,length(x))
y_2 <- rnorm(x,mean=0,sd=2)

x_3 <- rep(3,length(x))
y_3 <- rnorm(x,mean=0,sd=0.5)

df1=data.frame(explanatory = y_1,independent = x_1)
df2=data.frame(explanatory = y_2,independent = x_2)
```

```r
df3=data.frame(explanatory = y_3,independent = x_3)

df = rbind(df1,df2,df3)
df$independent = factor(df$independent)

lm = lm(explanatory~independent,df)
summary(lm)
```

```
##
## Call:
## lm(formula = explanatory ~ independent, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6384 -0.5781  0.0023  0.6711  5.3646
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.015033   0.065405   0.230    0.818
## independent2 0.003788   0.092497   0.041    0.967
## independent3 0.006075   0.092497   0.066    0.948
##
## Residual standard error: 1.31 on 1200 degrees of freedom
## Multiple R-squared:  3.667e-06,  Adjusted R-squared:  -0.001663
## F-statistic: 0.0022 on 2 and 1200 DF,  p-value: 0.9978
```

Ze summary funkce vidíme, že žádná z faktorů neovlivňuje vysvětlovanou veličinu a tudíž je zbytečné regresovat podle této proměnné (obrovské p-hodnoty koeficientů).

Nyní uvedeme případ, kdy se střední hodnoty rovnat nebudou.

```r
x <- seq(-20, 20, by = .1)
x_1 <- rep(1,length(x))
y_1 <- rnorm(x,mean=0,sd=1)

x_2 <- rep(2,length(x))
y_2 <- rnorm(x,mean=1,sd=1)

x_3 <- rep(3,length(x))
y_3 <- rnorm(x,mean=0.5,sd=0.5)


df1=data.frame(explanatory = y_1,independent = x_1)
df2=data.frame(explanatory = y_2,independent = x_2)
df3=data.frame(explanatory = y_3,independent = x_3)

df = rbind(df1,df2,df3)
df$independent = factor(df$independent)

lm = lm(explanatory~independent,df)
summary(lm)
```

```
##
## Call:
## lm(formula = explanatory ~ independent, data = df)
```

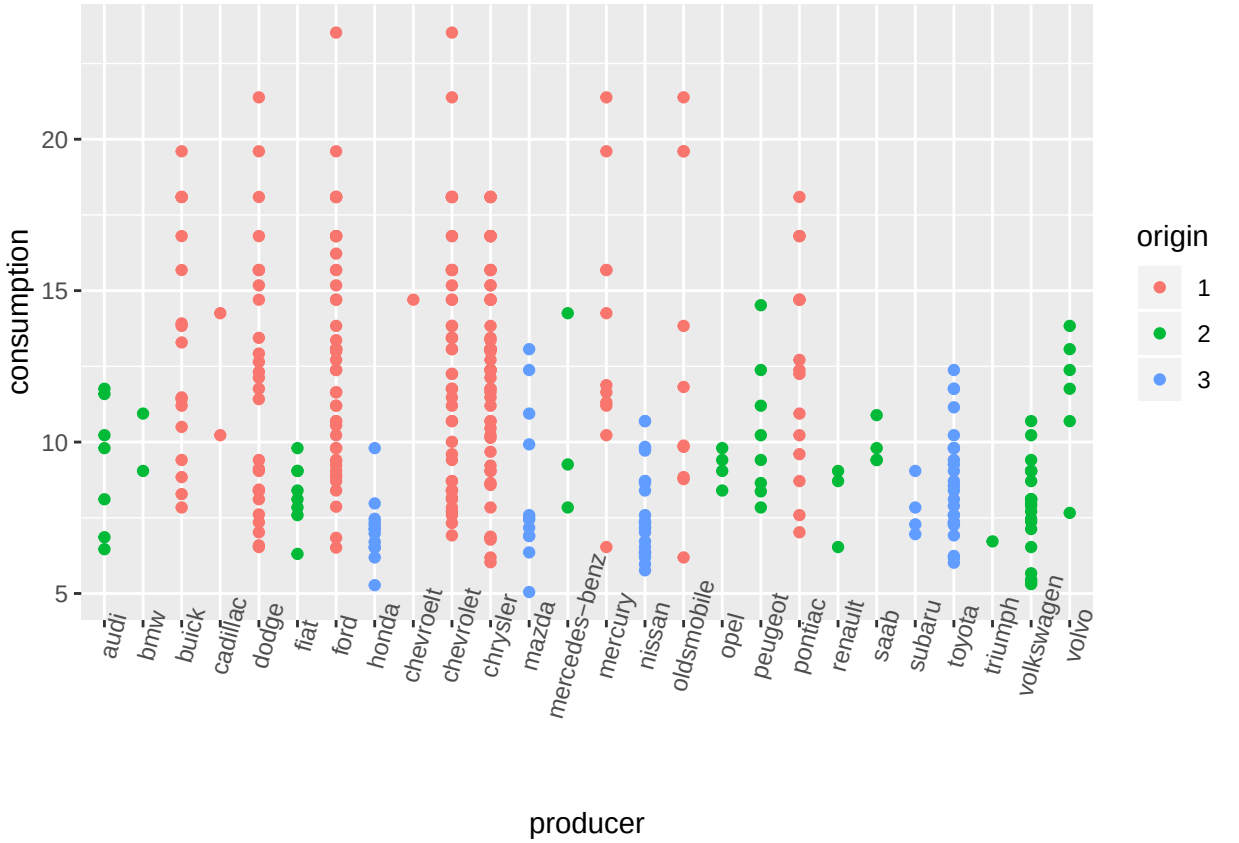
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1181 -0.4909  0.0040  0.5437  3.3799
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.01051    0.04316   0.244    0.808
## independent2  1.05973    0.06104  17.361  < 2e-16 ***
## independent3  0.45073    0.06104   7.384 2.86e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8643 on 1200 degrees of freedom
## Multiple R-squared:  0.2019, Adjusted R-squared:  0.2006
## F-statistic: 151.8 on 2 and 1200 DF,  p-value: < 2.2e-16
```

V tomto případě už je závislost zřejmá (p-hodnoty jsou extrémně nízké).

Otázka č.16: Zkonstruujte lineární model popisující spotřebu automobilu s využitím všech dostupných proměnných, kde bude přítomna interakce nejvýše dvou proměnných. Na základě kritérií jako jsou AIC, BIC, R^2, F, atd. vyberte nejvhodnější model. Ten validujte a okomentujte jeho výběr.

V modelu nebudeme uvažovat všechny pomocné proměnné, co jsme si v průběhu zpracování protokolu vytvořili. Dále nebudeme uvažovat car_name, jelikož bychom dostali extrémní počet faktorů a také proměnnou producer kvůli vysokému počtu faktorů a také zřejmé kolinearitě s origin (mohli bychom sloučit producery z jedné země do jednoho, viz obrázek).

```
p <- ggplot(data_mpghp, aes(x=producer, y=consumption, color = origin))+geom_point()+theme(axis.text.x =
plot(p)
```



Proměnnou cylinders v případě vícerozměrné regrese budeme uvažovat jako numerickou proměnnou, jednak kvůli snížení počtu faktorů v modelu (origin dohromady s model_year už dají 14 faktorů) a také kvůli přesnější detekci multikolinearity mezi numerickými proměnnými.

Nachystáme tedy maximálně veliký model s interakcemi.

```
data_mpghp$cylinders = as.numeric(as.character(data_mpghp$cylinders))
lm_all  <- lm(consumption ~ (.-car_name-producer-mpg-constant_weight-constant_weight2)*(.-car_name-produ
summary(lm_all)
```

```
##
## Call:
## lm(formula = consumption ~ (. - car_name - producer - mpg - constant_weight -
##     constant_weight2) * (. - car_name - producer - mpg - constant_weight -
##     constant_weight2), data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0539 -0.5546 -0.0272  0.5050  4.2820
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -2.429e+00  9.072e+00  -0.268 0.789125
## cylinders                  -1.179e+00  1.939e+00  -0.608 0.543837
## displacement                5.754e+00  2.799e+00   2.055 0.040808 *
## horsepower                 -1.501e-02  7.500e-02  -0.200 0.841507
## weight                     -5.538e-03  9.857e-03  -0.562 0.574653
```

```
## acceleration               5.649e-01  3.791e-01   1.490 0.137411
## model_year71               4.410e+00  6.291e+00   0.701 0.483911
## model_year72               6.352e+00  7.389e+00   0.860 0.390794
## model_year73               7.769e+00  6.380e+00   1.218 0.224371
## model_year74               9.593e+00  6.854e+00   1.400 0.162809
## model_year75               8.324e+00  7.189e+00   1.158 0.247899
## model_year76              -3.935e-01  6.154e+00  -0.064 0.949062
## model_year77               4.152e+00  8.009e+00   0.518 0.604630
## model_year78               7.806e+00  6.275e+00   1.244 0.214624
## model_year79               6.813e+00  6.345e+00   1.074 0.283908
## model_year80               1.435e+01  7.241e+00   1.982 0.048469 *
## model_year81               1.075e+01  6.516e+00   1.650 0.100221
## model_year82               6.857e+00  6.437e+00   1.065 0.287763
## origin2                    1.413e+00  3.651e+00   0.387 0.699120
## origin3                    2.003e+00  4.024e+00   0.498 0.619118
## cylinders:displacement     1.558e-01  2.620e-01   0.595 0.552641
## cylinders:horsepower       2.439e-02  1.364e-02   1.787 0.074996 .
## cylinders:weight          -1.847e-03  1.241e-03  -1.488 0.137857
## cylinders:acceleration    -8.752e-03  8.729e-02  -0.100 0.920207
## cylinders:model_year71     1.067e+00  1.004e+00   1.062 0.289065
## cylinders:model_year72    -9.878e-01  9.030e-01  -1.094 0.274957
## cylinders:model_year73     1.903e-02  7.534e-01   0.025 0.979867
## cylinders:model_year74     7.242e-01  1.016e+00   0.713 0.476542
## cylinders:model_year75     9.309e-01  9.025e-01   1.032 0.303240
## cylinders:model_year76     1.766e+00  9.668e-01   1.826 0.068942 .
## cylinders:model_year77     1.170e+00  9.460e-01   1.236 0.217389
## cylinders:model_year78     8.079e-01  9.394e-01   0.860 0.390532
## cylinders:model_year79     1.082e+00  9.037e-01   1.197 0.232256
## cylinders:model_year80     5.628e-01  1.066e+00   0.528 0.598072
## cylinders:model_year81     1.420e+00  9.475e-01   1.499 0.135059
## cylinders:model_year82     7.243e-01  1.102e+00   0.657 0.511617
## cylinders:origin2          5.662e-01  5.887e-01   0.962 0.337067
## cylinders:origin3         -8.082e-02  5.407e-01  -0.149 0.881291
## displacement:horsepower   -3.726e-02  1.093e-02  -3.408 0.000756 ***
## displacement:weight        5.584e-04  1.068e-03   0.523 0.601342
## displacement:acceleration -1.555e-01  1.150e-01  -1.353 0.177219
## displacement:model_year71 -1.080e+00  1.156e+00  -0.934 0.351069
## displacement:model_year72  5.209e-01  1.123e+00   0.464 0.643184
## displacement:model_year73  5.180e-01  7.917e-01   0.654 0.513490
## displacement:model_year74  8.590e-01  1.244e+00   0.691 0.490450
## displacement:model_year75  5.277e-01  1.071e+00   0.493 0.622713
## displacement:model_year76 -5.223e-01  1.099e+00  -0.475 0.635012
## displacement:model_year77 -8.690e-01  1.168e+00  -0.744 0.457391
## displacement:model_year78 -1.959e+00  1.225e+00  -1.599 0.110922
## displacement:model_year79 -1.044e+00  1.144e+00  -0.912 0.362500
## displacement:model_year80  2.796e-01  1.701e+00   0.164 0.869557
## displacement:model_year81 -2.014e+00  1.232e+00  -1.635 0.103325
## displacement:model_year82 -1.394e+00  1.318e+00  -1.058 0.290990
## displacement:origin2      -8.058e-01  1.369e+00  -0.589 0.556651
## displacement:origin3      -2.781e+00  1.386e+00  -2.007 0.045759 *
## horsepower:weight          6.054e-05  4.129e-05   1.466 0.143720
## horsepower:acceleration    7.498e-04  2.298e-03   0.326 0.744461
## horsepower:model_year71   -6.375e-02  3.177e-02  -2.007 0.045803 *
## horsepower:model_year72   -8.069e-02  4.065e-02  -1.985 0.048146 *
```

```
## horsepower:model_year73   -7.443e-02  2.263e-02  -3.289 0.001142 **
## horsepower:model_year74   -1.088e-01  4.141e-02  -2.628 0.009095 **
## horsepower:model_year75   -1.158e-01  3.587e-02  -3.228 0.001401 **
## horsepower:model_year76   -4.617e-02  2.982e-02  -1.548 0.122726
## horsepower:model_year77   -7.167e-02  4.154e-02  -1.725 0.085656 .
## horsepower:model_year78   -9.261e-02  3.058e-02  -3.029 0.002696 **
## horsepower:model_year79   -1.144e-01  3.942e-02  -2.903 0.004006 **
## horsepower:model_year80   -1.461e-01  4.711e-02  -3.101 0.002133 **
## horsepower:model_year81   -1.405e-01  4.110e-02  -3.418 0.000729 ***
## horsepower:model_year82   -2.017e-02  4.261e-02  -0.473 0.636416
## horsepower:origin2         3.949e-03  3.004e-02   0.131 0.895519
## horsepower:origin3         1.247e-02  3.294e-02   0.379 0.705287
## weight:acceleration        4.479e-04  3.211e-04   1.395 0.164114
## weight:model_year71        6.114e-03  3.751e-03   1.630 0.104261
## weight:model_year72        1.020e-02  3.865e-03   2.638 0.008827 **
## weight:model_year73        5.149e-03  3.067e-03   1.679 0.094357 .
## weight:model_year74        3.828e-03  4.528e-03   0.846 0.398572
## weight:model_year75        3.318e-03  3.923e-03   0.846 0.398402
## weight:model_year76        7.959e-04  3.583e-03   0.222 0.824373
## weight:model_year77        4.989e-03  3.654e-03   1.365 0.173291
## weight:model_year78        1.114e-02  4.169e-03   2.673 0.007990 **
## weight:model_year79        8.570e-03  4.663e-03   1.838 0.067164 .
## weight:model_year80        5.868e-03  5.108e-03   1.149 0.251741
## weight:model_year81        1.041e-02  4.735e-03   2.198 0.028800 *
## weight:model_year82        1.106e-03  5.318e-03   0.208 0.835490
## weight:origin2            -1.790e-03  3.493e-03  -0.512 0.608728
## weight:origin3             3.236e-03  4.158e-03   0.778 0.437133
## acceleration:model_year71 -5.650e-01  2.450e-01  -2.306 0.021882 *
## acceleration:model_year72 -5.025e-01  2.463e-01  -2.040 0.042310 *
## acceleration:model_year73 -5.614e-01  2.190e-01  -2.564 0.010898 *
## acceleration:model_year74 -7.164e-01  2.695e-01  -2.658 0.008327 **
## acceleration:model_year75 -5.770e-01  2.782e-01  -2.074 0.039054 *
## acceleration:model_year76 -3.521e-01  2.220e-01  -1.586 0.113933
## acceleration:model_year77 -5.786e-01  2.952e-01  -1.960 0.051015 .
## acceleration:model_year78 -8.743e-01  2.434e-01  -3.592 0.000390 ***
## acceleration:model_year79 -7.744e-01  2.482e-01  -3.120 0.002007 **
## acceleration:model_year80 -9.176e-01  2.778e-01  -3.303 0.001086 **
## acceleration:model_year81 -9.619e-01  2.541e-01  -3.786 0.000189 ***
## acceleration:model_year82 -6.293e-01  2.412e-01  -2.609 0.009586 **
## acceleration:origin2      -1.747e-01  1.457e-01  -1.199 0.231577
## acceleration:origin3      -1.806e-01  1.654e-01  -1.092 0.275941
## model_year71:origin2       1.369e+00  1.812e+00   0.756 0.450550
## model_year72:origin2       1.409e+00  1.834e+00   0.768 0.442983
## model_year73:origin2       2.305e+00  1.784e+00   1.292 0.197385
## model_year74:origin2       2.851e+00  1.849e+00   1.542 0.124361
## model_year75:origin2       3.196e+00  1.935e+00   1.651 0.099823 .
## model_year76:origin2       3.495e+00  1.799e+00   1.942 0.053172 .
## model_year77:origin2       2.598e+00  1.822e+00   1.427 0.154880
## model_year78:origin2       2.438e+00  1.785e+00   1.365 0.173281
## model_year79:origin2       1.546e+00  1.788e+00   0.865 0.387940
## model_year80:origin2       1.134e+00  1.837e+00   0.617 0.537598
## model_year81:origin2       5.142e-01  1.866e+00   0.276 0.783084
## model_year82:origin2       2.018e+00  1.947e+00   1.036 0.301001
## model_year71:origin3       1.132e+00  2.065e+00   0.548 0.583893
```

```
## model_year72:origin3       8.608e-01  1.991e+00   0.432 0.665869
## model_year73:origin3       2.956e+00  2.014e+00   1.467 0.143468
## model_year74:origin3       2.354e+00  2.060e+00   1.143 0.254103
## model_year75:origin3       1.977e+00  2.067e+00   0.957 0.339671
## model_year76:origin3       2.527e+00  2.038e+00   1.240 0.216145
## model_year77:origin3       2.192e+00  1.983e+00   1.105 0.269955
## model_year78:origin3       2.228e+00  1.947e+00   1.144 0.253529
## model_year79:origin3       2.521e+00  2.076e+00   1.214 0.225667
## model_year80:origin3       1.121e+00  2.049e+00   0.547 0.584918
## model_year81:origin3       1.317e+00  1.932e+00   0.682 0.496106
## model_year82:origin3       1.867e+00  1.961e+00   0.952 0.341938
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.091 on 267 degrees of freedom
## Multiple R-squared:  0.9449, Adjusted R-squared:  0.9195
## F-statistic: 37.19 on 123 and 267 DF,  p-value: < 2.2e-16
```

Kolinearita v modelu je zřejmá. Vzhledem k následujícím otázkám její řešení odložíme na později a teď ji vynecháme. Použijeme AIC i BIC hodnoty k určení nejvhodnějšího modelu z lm_all. Pak výsledné modely z obou metod porovnáme pomocí anova() a vybereme ten vhodnější.

Vypisovat celou proceduru by zabralo mnoho stran, tak zde vypisování outputu z funkce vynechám a uvedeme jen porovnání a summary výsledného modelu.

```
lm_temp_aic <- stepAIC(lm_all,direction='backward', trace=FALSE) # AIC
lm_temp_bic <- stepAIC(lm_all,direction='backward', k=log(nrow(data_mpghp)), trace=FALSE) # BIC
```

Porovnáme oba modely pomocí anova()

```
anova(lm_temp_bic,lm_temp_aic)
```

```
## Analysis of Variance Table
##
## Model 1: consumption ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + origin + cylinders:displacement +
##     cylinders:horsepower + displacement:horsepower + displacement:origin +
##     acceleration:origin
## Model 2: consumption ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + origin + cylinders:horsepower +
##     displacement:horsepower + displacement:acceleration + displacement:model_year +
##     displacement:origin + horsepower:weight + horsepower:model_year +
##     weight:acceleration + weight:origin + acceleration:model_year +
##     acceleration:origin
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    364 498.64
## 2    324 385.42 40    113.22 2.3795 1.745e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Z nízké hodnoty vychází, že anova() preferuje model získaný metodou AIC. Budeme tedy pokračovat s lm_temp_aic. Detaily modelu vypíšeme pomocí summary funkce.

```
summary(lm_temp_aic)
```

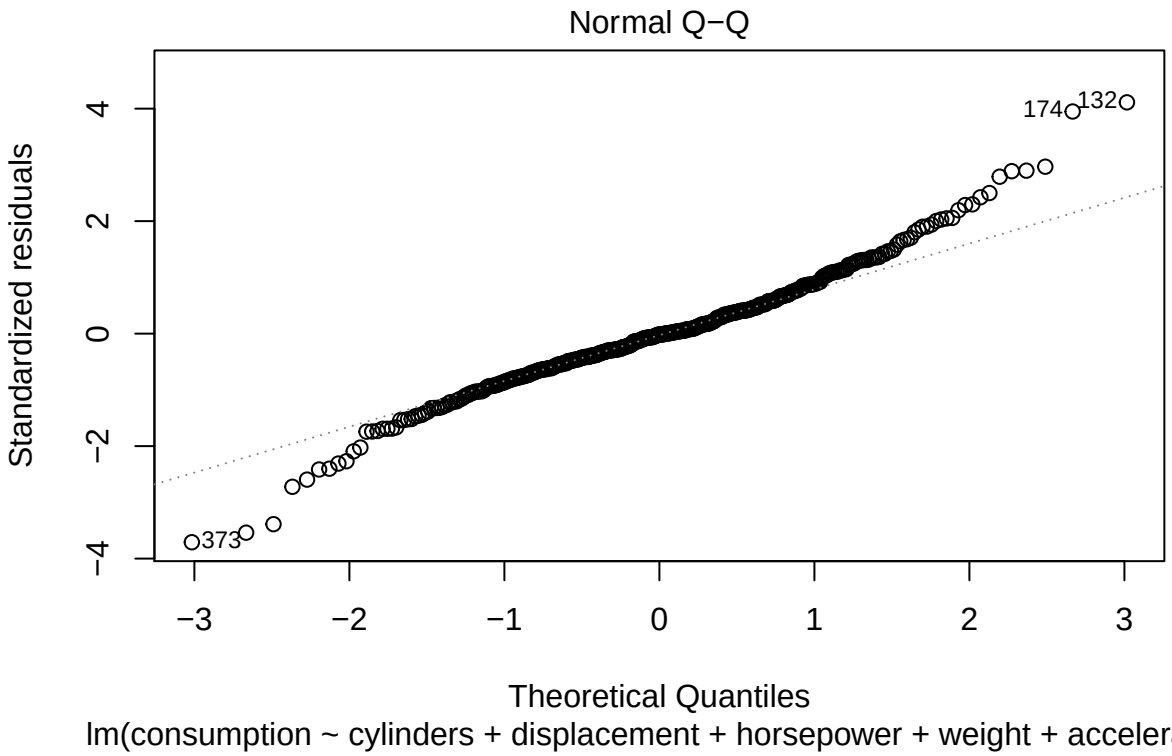
```
##
## Call:
## lm(formula = consumption ~ cylinders + displacement + horsepower +
```

```
##     weight + acceleration + model_year + origin + cylinders:horsepower +
##     displacement:horsepower + displacement:acceleration + displacement:model_year +
##     displacement:origin + horsepower:weight + horsepower:model_year +
##     weight:acceleration + weight:origin + acceleration:model_year +
##     acceleration:origin, data = data_mpghp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3480 -0.5879 -0.0181  0.5323  4.2153
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -9.546e+00  5.746e+00  -1.661 0.097589 .
## cylinders                  -1.042e+00  4.970e-01  -2.097 0.036783 *
## displacement                5.898e+00  1.453e+00   4.061 6.14e-05 ***
## horsepower                  3.747e-02  2.633e-02   1.423 0.155637
## weight                     -7.889e-03  5.715e-03  -1.380 0.168390
## acceleration                6.992e-01  2.516e-01   2.779 0.005771 **
## model_year71                1.193e+01  4.170e+00   2.861 0.004491 **
## model_year72                6.695e+00  4.758e+00   1.407 0.160401
## model_year73                1.217e+01  3.807e+00   3.196 0.001533 **
## model_year74                1.806e+01  4.436e+00   4.071 5.89e-05 ***
## model_year75                1.934e+01  3.915e+00   4.940 1.25e-06 ***
## model_year76                1.134e+01  3.647e+00   3.109 0.002045 **
## model_year77                1.569e+01  5.465e+00   2.870 0.004373 **
## model_year78                1.351e+01  3.868e+00   3.494 0.000543 ***
## model_year79                1.468e+01  4.095e+00   3.585 0.000390 ***
## model_year80                1.839e+01  4.522e+00   4.066 6.01e-05 ***
## model_year81                1.942e+01  4.263e+00   4.556 7.40e-06 ***
## model_year82                1.413e+01  3.962e+00   3.565 0.000418 ***
## origin2                     5.513e+00  1.661e+00   3.320 0.001005 **
## origin3                     4.580e+00  2.639e+00   1.735 0.083618 .
## cylinders:horsepower        1.090e-02  4.287e-03   2.542 0.011491 *
## displacement:horsepower    -2.241e-02  4.401e-03  -5.091 6.04e-07 ***
## displacement:acceleration -1.877e-01  6.803e-02  -2.760 0.006118 **
## displacement:model_year71  3.052e-01  4.389e-01   0.696 0.487215
## displacement:model_year72 -7.195e-02  5.461e-01  -0.132 0.895267
## displacement:model_year73  2.290e-01  3.756e-01   0.610 0.542578
## displacement:model_year74  9.336e-01  5.477e-01   1.704 0.089249 .
## displacement:model_year75  8.446e-01  4.075e-01   2.072 0.039010 *
## displacement:model_year76 -1.702e-01  4.233e-01  -0.402 0.687980
## displacement:model_year77  2.344e-01  5.876e-01   0.399 0.690172
## displacement:model_year78 -7.022e-01  3.985e-01  -1.762 0.078997 .
## displacement:model_year79 -2.272e-02  5.777e-01  -0.039 0.968655
## displacement:model_year80  1.118e+00  7.470e-01   1.496 0.135520
## displacement:model_year81  5.246e-02  4.998e-01   0.105 0.916469
## displacement:model_year82 -1.190e+00  5.565e-01  -2.138 0.033230 *
## displacement:origin2      -9.552e-02  9.254e-01  -0.103 0.917848
## displacement:origin3      -3.766e+00  7.881e-01  -4.779 2.67e-06 ***
## horsepower:weight          3.052e-05  1.828e-05   1.669 0.096006 .
## horsepower:model_year71   -4.358e-02  2.040e-02  -2.136 0.033388 *
## horsepower:model_year72   -1.135e-02  2.890e-02  -0.393 0.694904
## horsepower:model_year73   -3.970e-02  1.604e-02  -2.474 0.013860 *
## horsepower:model_year74   -8.874e-02  2.678e-02  -3.314 0.001025 **
```

```
## horsepower:model_year75   -1.047e-01  2.035e-02  -5.147 4.60e-07 ***
## horsepower:model_year76   -2.397e-02  1.907e-02  -1.257 0.209616
## horsepower:model_year77   -6.327e-02  3.231e-02  -1.958 0.051098 .
## horsepower:model_year78   -2.573e-02  1.791e-02  -1.437 0.151691
## horsepower:model_year79   -5.499e-02  3.082e-02  -1.785 0.075259 .
## horsepower:model_year80   -1.052e-01  3.019e-02  -3.486 0.000559 ***
## horsepower:model_year81   -7.773e-02  2.623e-02  -2.964 0.003266 **
## horsepower:model_year82   -2.047e-02  2.677e-02  -0.765 0.445102
## weight:acceleration        5.170e-04  2.530e-04   2.043 0.041859 *
## weight:origin2            -1.569e-04  1.901e-03  -0.083 0.934284
## weight:origin3             7.189e-03  2.337e-03   3.076 0.002273 **
## acceleration:model_year71 -5.542e-01  2.023e-01  -2.740 0.006490 **
## acceleration:model_year72 -3.326e-01  2.053e-01  -1.621 0.106063
## acceleration:model_year73 -5.216e-01  1.851e-01  -2.818 0.005128 **
## acceleration:model_year74 -7.853e-01  2.169e-01  -3.621 0.000341 ***
## acceleration:model_year75 -7.682e-01  1.847e-01  -4.160 4.08e-05 ***
## acceleration:model_year76 -5.802e-01  1.756e-01  -3.305 0.001057 **
## acceleration:model_year77 -7.231e-01  2.452e-01  -2.949 0.003418 **
## acceleration:model_year78 -6.519e-01  1.845e-01  -3.533 0.000471 ***
## acceleration:model_year79 -7.217e-01  1.803e-01  -4.002 7.78e-05 ***
## acceleration:model_year80 -8.792e-01  2.013e-01  -4.367 1.70e-05 ***
## acceleration:model_year81 -9.187e-01  1.964e-01  -4.679 4.24e-06 ***
## acceleration:model_year82 -7.417e-01  1.769e-01  -4.193 3.56e-05 ***
## acceleration:origin2      -3.140e-01  8.749e-02  -3.589 0.000383 ***
## acceleration:origin3      -3.304e-01  1.128e-01  -2.929 0.003640 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.091 on 324 degrees of freedom
## Multiple R-squared:  0.9332, Adjusted R-squared:  0.9196
## F-statistic: 68.55 on 66 and 324 DF,  p-value: < 2.2e-16
```

Pokusíme se nyní validovat OLS předpoklady. Začneme normalitou.

```
plot(lm_temp_aic,which=2)
```



QQ plot nevypadá příliš dobře, je tam příliš velké odchýlení chvostů. Podíváme se na shapirův test.

```
shapiro.test(residuals(lm_temp_aic))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_temp_aic)
## W = 0.98009, p-value = 3.232e-05
```

p-hodnota je příliš nízká, shapirův test normalitu zamítá. OLS požadavaky tento model nesplňuje. V dalších otázkách budeme zkoušet různé transformace z důvodu zlepšení normality. Problémy s normalitou mohou být také způsobeny multikolinearitou v modelu, kterou budeme také teprve zkoumat.

Otázka č.17: Pro vybraný model z předchozí otázky vyzkoušejte jak logaritmickou transformaci odezvy, tak Box-Coxovu transformaci pro zlepšení normality reziduí. Vykreslete optimální log-věrohodnostní profil u Box-Coxovy transformace a porovnejte navrženou mocninou transformaci s logaritmickou. Pro který model se rozhodnete a proč?

Jako první provedeme logaritmickou transformaci.

```
form = 'log(consumption) ~ cylinders + displacement + horsepower + weight +
acceleration + model_year + origin + cylinders:horsepower +
cylinders:model_year + displacement:horsepower + displacement:acceleration +
displacement:model_year + displacement:origin + horsepower:weight +
horsepower:model_year + weight:acceleration + weight:model_year +
weight:origin + acceleration:model_year + acceleration:origin'
```

```r
lm_final_log = lm(form,data_mpghp)
summary(lm_final_log)
```

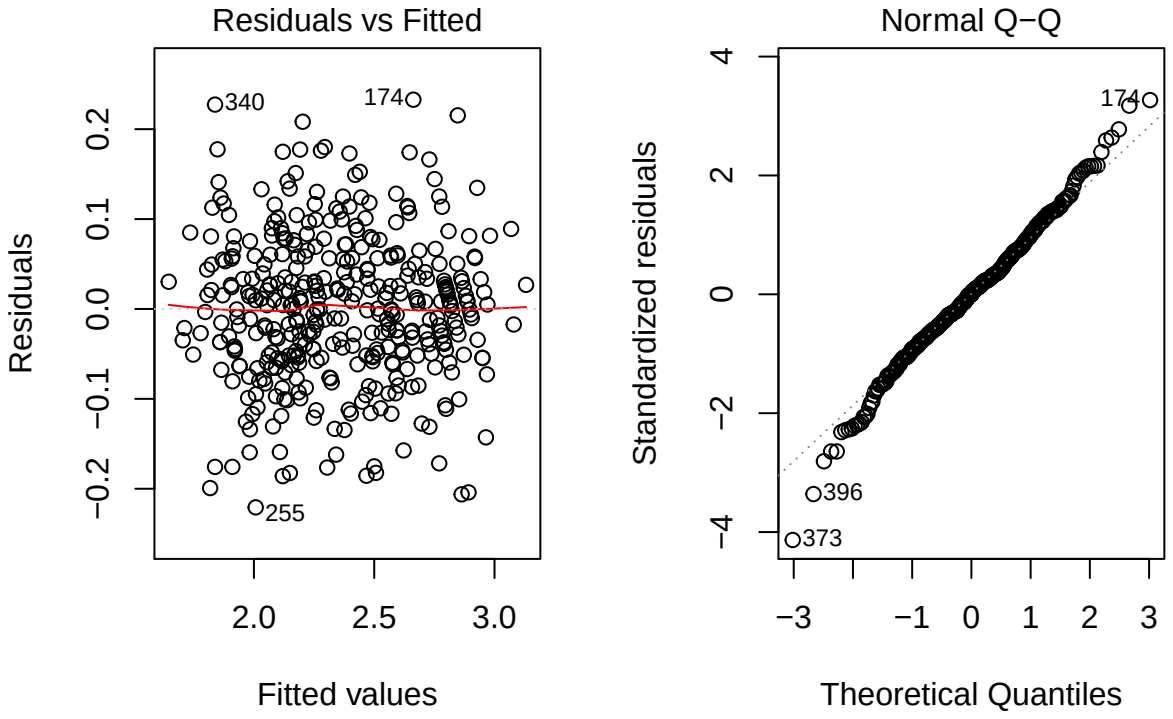
```
##
## Call:
## lm(formula = form, data = data_mpghp)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.220732 -0.051772 -0.000002  0.054068  0.232813
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.151e-01  5.701e-01   0.553 0.580933
## cylinders                  -6.597e-02  7.933e-02  -0.832 0.406349
## displacement                5.967e-01  1.643e-01   3.632 0.000331 ***
## horsepower                  6.718e-03  2.530e-03   2.655 0.008345 **
## weight                     -7.719e-04  6.680e-04  -1.155 0.248810
## acceleration                7.290e-02  2.259e-02   3.227 0.001388 **
## model_year71                8.899e-01  3.955e-01   2.250 0.025161 *
## model_year72                1.074e+00  5.014e-01   2.142 0.032958 *
## model_year73                1.021e+00  3.803e-01   2.686 0.007644 **
## model_year74                1.554e+00  4.058e-01   3.831 0.000156 ***
## model_year75                1.271e+00  4.447e-01   2.858 0.004557 **
## model_year76                5.274e-01  3.551e-01   1.485 0.138582
## model_year77                9.287e-01  5.118e-01   1.814 0.070610 .
## model_year78                1.075e+00  3.783e-01   2.842 0.004796 **
## model_year79                7.042e-01  3.867e-01   1.821 0.069606 .
## model_year80                1.648e+00  4.620e-01   3.568 0.000418 ***
## model_year81                1.310e+00  4.024e-01   3.255 0.001266 **
## model_year82                7.325e-01  4.004e-01   1.829 0.068342 .
## origin2                     5.976e-01  1.511e-01   3.955 9.56e-05 ***
## origin3                     3.564e-01  2.312e-01   1.541 0.124300
## cylinders:horsepower        6.756e-04  4.421e-04   1.528 0.127494
## cylinders:model_year71      9.421e-03  7.124e-02   0.132 0.894889
## cylinders:model_year72     -1.391e-01  5.938e-02  -2.343 0.019788 *
## cylinders:model_year73     -5.479e-02  4.591e-02  -1.193 0.233661
## cylinders:model_year74     -1.123e-02  6.897e-02  -0.163 0.870784
## cylinders:model_year75      6.517e-03  5.915e-02   0.110 0.912350
## cylinders:model_year76      4.128e-02  6.094e-02   0.677 0.498653
## cylinders:model_year77      6.068e-03  5.708e-02   0.106 0.915416
## cylinders:model_year78      1.078e-02  5.848e-02   0.184 0.853826
## cylinders:model_year79      1.329e-02  5.663e-02   0.235 0.814631
## cylinders:model_year80     -1.081e-01  6.642e-02  -1.628 0.104530
## cylinders:model_year81      2.396e-02  6.080e-02   0.394 0.693809
## cylinders:model_year82     -3.544e-03  7.812e-02  -0.045 0.963840
## displacement:horsepower    -2.166e-03  5.824e-04  -3.720 0.000238 ***
## displacement:acceleration -1.742e-02  6.301e-03  -2.764 0.006055 **
## displacement:model_year71 -3.250e-02  8.828e-02  -0.368 0.713042
## displacement:model_year72  8.855e-02  8.711e-02   1.017 0.310159
## displacement:model_year73  1.458e-02  6.149e-02   0.237 0.812712
## displacement:model_year74  7.694e-02  9.272e-02   0.830 0.407340
## displacement:model_year75  6.263e-03  8.008e-02   0.078 0.937713
```

```
## displacement:model_year76 -9.626e-02  7.668e-02  -1.255 0.210305
## displacement:model_year77 -3.916e-02  8.328e-02  -0.470 0.638596
## displacement:model_year78 -2.181e-01  8.468e-02  -2.575 0.010498 *
## displacement:model_year79 -1.149e-01  9.310e-02  -1.234 0.218174
## displacement:model_year80  1.808e-01  1.160e-01   1.558 0.120288
## displacement:model_year81 -1.031e-01  8.745e-02  -1.179 0.239340
## displacement:model_year82 -1.704e-01  1.061e-01  -1.606 0.109409
## displacement:origin2      -1.090e-01  9.042e-02  -1.206 0.228794
## displacement:origin3      -3.146e-01  7.553e-02  -4.166 4.06e-05 ***
## horsepower:weight          1.948e-06  2.219e-06   0.878 0.380595
## horsepower:model_year71   -5.163e-03  2.326e-03  -2.220 0.027200 *
## horsepower:model_year72   -6.186e-03  3.160e-03  -1.958 0.051192 .
## horsepower:model_year73   -2.726e-03  1.747e-03  -1.561 0.119665
## horsepower:model_year74   -7.443e-03  3.112e-03  -2.391 0.017400 *
## horsepower:model_year75   -7.923e-03  2.819e-03  -2.810 0.005273 **
## horsepower:model_year76   -8.603e-04  2.237e-03  -0.385 0.700769
## horsepower:model_year77   -4.362e-03  3.165e-03  -1.378 0.169194
## horsepower:model_year78   -5.505e-03  2.218e-03  -2.482 0.013598 *
## horsepower:model_year79   -5.974e-03  3.109e-03  -1.921 0.055627 .
## horsepower:model_year80   -1.082e-02  3.513e-03  -3.080 0.002265 **
## horsepower:model_year81   -9.011e-03  2.976e-03  -3.028 0.002676 **
## horsepower:model_year82    1.760e-03  3.296e-03   0.534 0.593649
## weight:acceleration        3.617e-05  2.324e-05   1.556 0.120734
## weight:model_year71        3.843e-04  3.039e-04   1.264 0.207053
## weight:model_year72        6.111e-04  2.927e-04   2.088 0.037620 *
## weight:model_year73        2.005e-04  2.300e-04   0.872 0.383915
## weight:model_year74        1.534e-04  3.590e-04   0.427 0.669468
## weight:model_year75        2.852e-04  3.017e-04   0.946 0.345155
## weight:model_year76        1.713e-04  2.717e-04   0.630 0.528903
## weight:model_year77        3.107e-04  2.882e-04   1.078 0.281894
## weight:model_year78        1.047e-03  3.084e-04   3.396 0.000776 ***
## weight:model_year79        7.585e-04  3.632e-04   2.089 0.037591 *
## weight:model_year80        4.419e-04  4.014e-04   1.101 0.271879
## weight:model_year81        7.288e-04  3.340e-04   2.182 0.029897 *
## weight:model_year82        2.263e-04  4.015e-04   0.564 0.573353
## weight:origin2             4.488e-05  1.814e-04   0.247 0.804746
## weight:origin3             5.983e-04  2.197e-04   2.723 0.006850 **
## acceleration:model_year71 -5.342e-02  1.868e-02  -2.859 0.004543 **
## acceleration:model_year72 -4.634e-02  1.922e-02  -2.411 0.016500 *
## acceleration:model_year73 -4.374e-02  1.674e-02  -2.614 0.009407 **
## acceleration:model_year74 -7.572e-02  1.924e-02  -3.936 0.000103 ***
## acceleration:model_year75 -5.970e-02  2.047e-02  -2.916 0.003809 **
## acceleration:model_year76 -4.037e-02  1.619e-02  -2.494 0.013167 *
## acceleration:model_year77 -5.815e-02  2.214e-02  -2.627 0.009065 **
## acceleration:model_year78 -8.941e-02  1.846e-02  -4.845 2.04e-06 ***
## acceleration:model_year79 -6.477e-02  1.769e-02  -3.662 0.000296 ***
## acceleration:model_year80 -8.702e-02  2.113e-02  -4.119 4.92e-05 ***
## acceleration:model_year81 -9.057e-02  1.853e-02  -4.889 1.66e-06 ***
## acceleration:model_year82 -6.130e-02  1.735e-02  -3.533 0.000475 ***
## acceleration:origin2      -2.869e-02  7.938e-03  -3.614 0.000354 ***
## acceleration:origin3      -2.644e-02  1.017e-02  -2.599 0.009798 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.09297 on 300 degrees of freedom
## Multiple R-squared:  0.9416, Adjusted R-squared:  0.9241
## F-statistic: 53.73 on 90 and 300 DF,  p-value: < 2.2e-16
```

Zkusíme tento model validovat.

```
par(mfrow = c(1,2))
plot(lm_final_log,which=1)
plot(lm_final_log,which=2)
```



QQ plot i residuals vs fitted vypadají velmi hezky. Podíváme se, co říká Shapirův test.

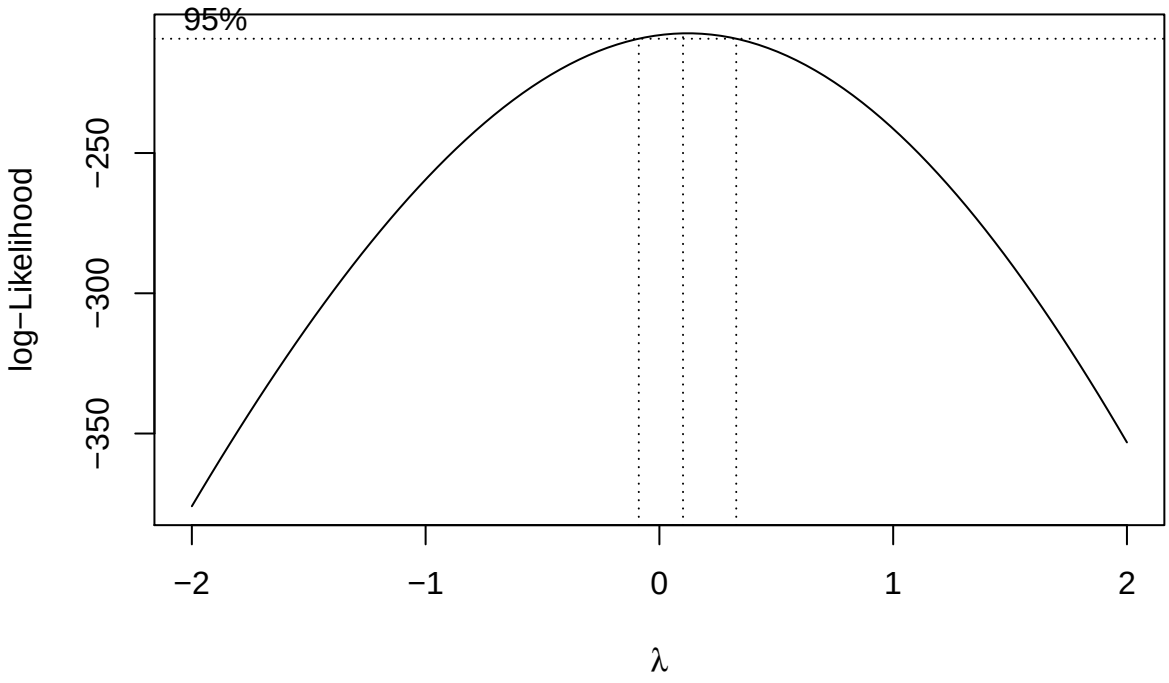
```
shapiro.test(residuals(lm_final_log))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_final_log)
## W = 0.99608, p-value = 0.4466
```

Ten normalitu nezamítá, takže zde logaritmická transformace pomohla k lepšími splnění OLS požadavků.

Zkusíme Box-Coxovu transformaci.

```
library(MASS)
bc <- boxcox(lm_temp_aic)
```




```
lambda <- bc$x[which.max(bc$y)]
```

Nastavíme příslušný model.

```
form_lambda = '(consumption^lambda - 1)/lambda ~ cylinders + displacement + horsepower + weight +
acceleration + model_year + origin + cylinders:horsepower +
cylinders:model_year + displacement:horsepower + displacement:acceleration +
displacement:model_year + displacement:origin + horsepower:weight +
horsepower:model_year + weight:acceleration + weight:model_year +
weight:origin + acceleration:model_year + acceleration:origin'

lm_final_bc = lm(form_lambda,data_mpghp)
summary(lm_final_bc)
```

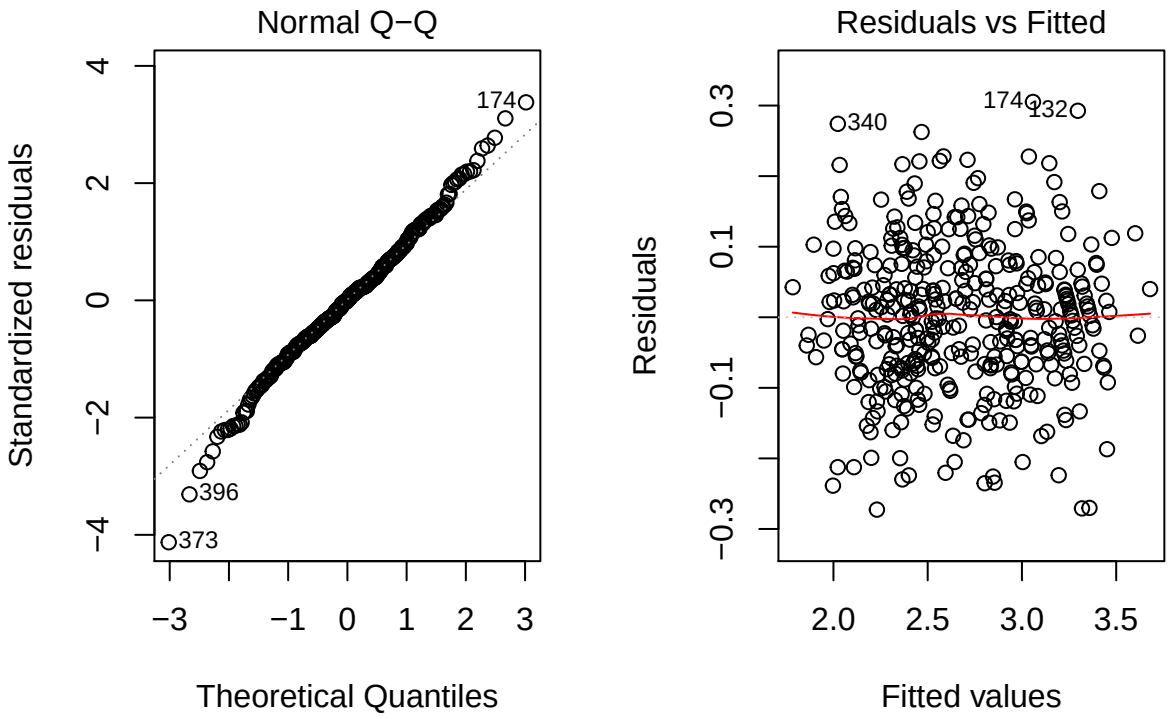
```
##
## Call:
## lm(formula = form_lambda, data = data_mpghp)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.272577 -0.066625  0.000324  0.066627  0.305079
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.336e-01  7.226e-01   0.185 0.853481
## cylinders                  -8.899e-02  1.006e-01  -0.885 0.376889
## displacement                7.692e-01  2.082e-01   3.694 0.000262 ***
```

```
## horsepower                 8.225e-03  3.207e-03   2.565 0.010813 *
## weight                    -1.065e-03  8.467e-04  -1.257 0.209592
## acceleration               9.142e-02  2.863e-02   3.193 0.001558 **
## model_year71               1.133e+00  5.013e-01   2.260 0.024550 *
## model_year72               1.386e+00  6.355e-01   2.180 0.029998 *
## model_year73               1.324e+00  4.820e-01   2.746 0.006395 **
## model_year74               1.980e+00  5.144e-01   3.849 0.000145 ***
## model_year75               1.640e+00  5.637e-01   2.909 0.003898 **
## model_year76               6.925e-01  4.501e-01   1.539 0.124967
## model_year77               1.208e+00  6.487e-01   1.863 0.063495 .
## model_year78               1.379e+00  4.795e-01   2.877 0.004304 **
## model_year79               9.605e-01  4.902e-01   1.959 0.050992 .
## model_year80               2.102e+00  5.855e-01   3.591 0.000385 ***
## model_year81               1.710e+00  5.101e-01   3.352 0.000904 ***
## model_year82               9.853e-01  5.075e-01   1.941 0.053141 .
## origin2                    7.469e-01  1.915e-01   3.899 0.000119 ***
## origin3                    4.525e-01  2.931e-01   1.544 0.123630
## cylinders:horsepower       9.200e-04  5.603e-04   1.642 0.101678
## cylinders:model_year71     1.317e-02  9.031e-02   0.146 0.884143
## cylinders:model_year72    -1.797e-01  7.526e-02  -2.388 0.017547 *
## cylinders:model_year73    -7.090e-02  5.819e-02  -1.218 0.224015
## cylinders:model_year74    -1.639e-02  8.742e-02  -0.187 0.851430
## cylinders:model_year75     7.354e-03  7.498e-02   0.098 0.921936
## cylinders:model_year76     5.497e-02  7.725e-02   0.712 0.477252
## cylinders:model_year77     7.501e-03  7.236e-02   0.104 0.917498
## cylinders:model_year78     1.413e-02  7.412e-02   0.191 0.848889
## cylinders:model_year79     1.528e-02  7.177e-02   0.213 0.831553
## cylinders:model_year80    -1.312e-01  8.419e-02  -1.558 0.120302
## cylinders:model_year81     3.123e-02  7.706e-02   0.405 0.685580
## cylinders:model_year82    -6.006e-03  9.902e-02  -0.061 0.951670
## displacement:horsepower   -2.834e-03  7.382e-04  -3.839 0.000151 ***
## displacement:acceleration -2.224e-02  7.987e-03  -2.785 0.005698 **
## displacement:model_year71 -4.436e-02  1.119e-01  -0.396 0.692079
## displacement:model_year72  1.153e-01  1.104e-01   1.045 0.297059
## displacement:model_year73  2.106e-02  7.795e-02   0.270 0.787244
## displacement:model_year74  9.959e-02  1.175e-01   0.847 0.397480
## displacement:model_year75  1.175e-02  1.015e-01   0.116 0.907944
## displacement:model_year76 -1.204e-01  9.719e-02  -1.238 0.216521
## displacement:model_year77 -4.867e-02  1.056e-01  -0.461 0.645110
## displacement:model_year78 -2.763e-01  1.073e-01  -2.574 0.010520 *
## displacement:model_year79 -1.424e-01  1.180e-01  -1.206 0.228606
## displacement:model_year80  2.230e-01  1.471e-01   1.517 0.130407
## displacement:model_year81 -1.321e-01  1.108e-01  -1.192 0.234253
## displacement:model_year82 -2.113e-01  1.345e-01  -1.571 0.117211
## displacement:origin2      -1.445e-01  1.146e-01  -1.261 0.208373
## displacement:origin3      -4.022e-01  9.574e-02  -4.201 3.50e-05 ***
## horsepower:weight          2.824e-06  2.812e-06   1.004 0.316106
## horsepower:model_year71   -6.703e-03  2.949e-03  -2.273 0.023727 *
## horsepower:model_year72   -8.066e-03  4.005e-03  -2.014 0.044895 *
## horsepower:model_year73   -3.775e-03  2.214e-03  -1.705 0.089275 .
## horsepower:model_year74   -9.686e-03  3.945e-03  -2.455 0.014646 *
## horsepower:model_year75   -1.035e-02  3.574e-03  -2.895 0.004071 **
## horsepower:model_year76   -1.325e-03  2.835e-03  -0.467 0.640594
## horsepower:model_year77   -5.785e-03  4.012e-03  -1.442 0.150357
```

```
## horsepower:model_year78   -7.163e-03  2.811e-03  -2.548 0.011324 *
## horsepower:model_year79   -7.985e-03  3.941e-03  -2.026 0.043627 *
## horsepower:model_year80   -1.378e-02  4.453e-03  -3.094 0.002159 **
## horsepower:model_year81   -1.181e-02  3.772e-03  -3.132 0.001911 **
## horsepower:model_year82    1.726e-03  4.178e-03   0.413 0.679833
## weight:acceleration        4.777e-05  2.946e-05   1.621 0.105993
## weight:model_year71        5.089e-04  3.852e-04   1.321 0.187434
## weight:model_year72        7.976e-04  3.709e-04   2.150 0.032333 *
## weight:model_year73        2.782e-04  2.915e-04   0.954 0.340652
## weight:model_year74        2.135e-04  4.551e-04   0.469 0.639364
## weight:model_year75        3.693e-04  3.824e-04   0.966 0.334884
## weight:model_year76        2.146e-04  3.444e-04   0.623 0.533664
## weight:model_year77        4.009e-04  3.653e-04   1.098 0.273295
## weight:model_year78        1.323e-03  3.909e-04   3.385 0.000808 ***
## weight:model_year79        9.675e-04  4.603e-04   2.102 0.036423 *
## weight:model_year80        5.430e-04  5.088e-04   1.067 0.286736
## weight:model_year81        9.312e-04  4.234e-04   2.199 0.028613 *
## weight:model_year82        2.877e-04  5.089e-04   0.565 0.572290
## weight:origin2             7.139e-05  2.299e-04   0.310 0.756414
## weight:origin3             7.625e-04  2.785e-04   2.738 0.006554 **
## acceleration:model_year71 -6.835e-02  2.368e-02  -2.887 0.004178 **
## acceleration:model_year72 -6.001e-02  2.436e-02  -2.464 0.014310 *
## acceleration:model_year73 -5.704e-02  2.121e-02  -2.689 0.007567 **
## acceleration:model_year74 -9.604e-02  2.439e-02  -3.938 0.000102 ***
## acceleration:model_year75 -7.646e-02  2.595e-02  -2.947 0.003460 **
## acceleration:model_year76 -5.203e-02  2.052e-02  -2.536 0.011731 *
## acceleration:model_year77 -7.458e-02  2.806e-02  -2.658 0.008293 **
## acceleration:model_year78 -1.129e-01  2.339e-02  -4.825 2.23e-06 ***
## acceleration:model_year79 -8.420e-02  2.242e-02  -3.756 0.000207 ***
## acceleration:model_year80 -1.097e-01  2.678e-02  -4.096 5.41e-05 ***
## acceleration:model_year81 -1.160e-01  2.348e-02  -4.939 1.30e-06 ***
## acceleration:model_year82 -7.831e-02  2.199e-02  -3.561 0.000430 ***
## acceleration:origin2      -3.584e-02  1.006e-02  -3.562 0.000428 ***
## acceleration:origin3      -3.335e-02  1.289e-02  -2.586 0.010167 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1178 on 300 degrees of freedom
## Multiple R-squared:  0.942,  Adjusted R-squared:  0.9246
## F-statistic: 54.15 on 90 and 300 DF,  p-value: < 2.2e-16
```

A validujeme tento model.

```
par(mfrow=c(1,2))

plot(lm_final_bc,which=2)
plot(lm_final_bc,which=1)
```



Oba ploty vypadají hezky, podíváme se ještě na normalitu pomocí Shapirova testu.

```
shapiro.test(residuals(lm_final_bc))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_final_bc)
## W = 0.99616, p-value = 0.4678
```

Normalita krásně prochází. Zdá se, že model s box-cox transformací je co se týče R^2 statistiky a splnění OLS požadavků na tom velmi podobně jako model s log transformací nakonec bych se asi rozhodl pro box-cox model, který vykazuje o chloupek lepší R^2 statistiku a vyšší hodnotu p-statistiky Shapirova testu určující normalitu reziduí.

Otázka č.18: Pro model s log-transformovanou spotřebou v bodě 16 vypočtěte procentuální navýšení/pokles spotřeby automobilu při změně váhy o 1000kg. Porovnejte, jak se změnil vliv spotřeby na váze s porovnáním s modelem z bodu 8.

Použili jsme log transformaci a v takovém případě změna spotřeby v závislosti na váze bude záviset na počáteční váze, od které změnu počítáme (změna je nelineární), nelze tedy určit obecně, jaká bude procentuální změna.

Model z otázky 16 obsahuje kromě samotné weight i interakce s weight, takže pak není zřejmé, jaký přesně má vliv určitá změna váhy na spotřebu, každopádně pokud bychom zanedbali interakce s weight , pak má weight opačný vliv než v modelu z otázky 9, tj. při zvyšování váhy spotřeba klesá - bez interakcí a ostatních nezávislých proměnných by ale takový model pak nedával s takovou závislostí příliš smysl.

Otázka č.19: Spočtěte korelace mezi všemi dostupnými proměnnými, kde to dává smysl a u korelovaných proměnných se pokuste zdůvodnit důvod této korelace. Zkoumejte případnou multikolinearitu ve vašem finálním modelu z bodu 16 a pomocí podmíněné matice regresorů, VIF a dalších nástrojů validujte váš výběr.

Na začátek zkoumání kolinearity vyzkoušíme vyzkoušíme kompletní aditivní model bez pomocných proměnných, car_name a producer a necháme si vypsat VIF hodnoty kvůli detekci kolinearity.

```
lmm <- lm(consumption ~ (.)-car_name-producer-constant_weight-constant_weight2-mpg,data_mpghp)
```

```
kappa(lmm)
```

```
## [1] 39176.5
```

```
vif(lmm)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## cylinders    11.341698  1        3.367744
## displacement 24.813437  1        4.981309
## horsepower   11.785566  1        3.433011
## weight       12.720002  1        3.566511
## acceleration  2.779139  1        1.667075
## model_year    2.041548 12        1.030184
## origin        2.266602  2        1.226998
```

Vidíme, že vyšší hodnoty VIF jsou u numerických proměnných cylinders,displacement,horsepowera weight, které by mohly být mezi sebou korelované To jde ostatně vidět i ze scatterplotů v první části protokolu. Podíváme se na výstup z ggpairs pro tyto proměnné.

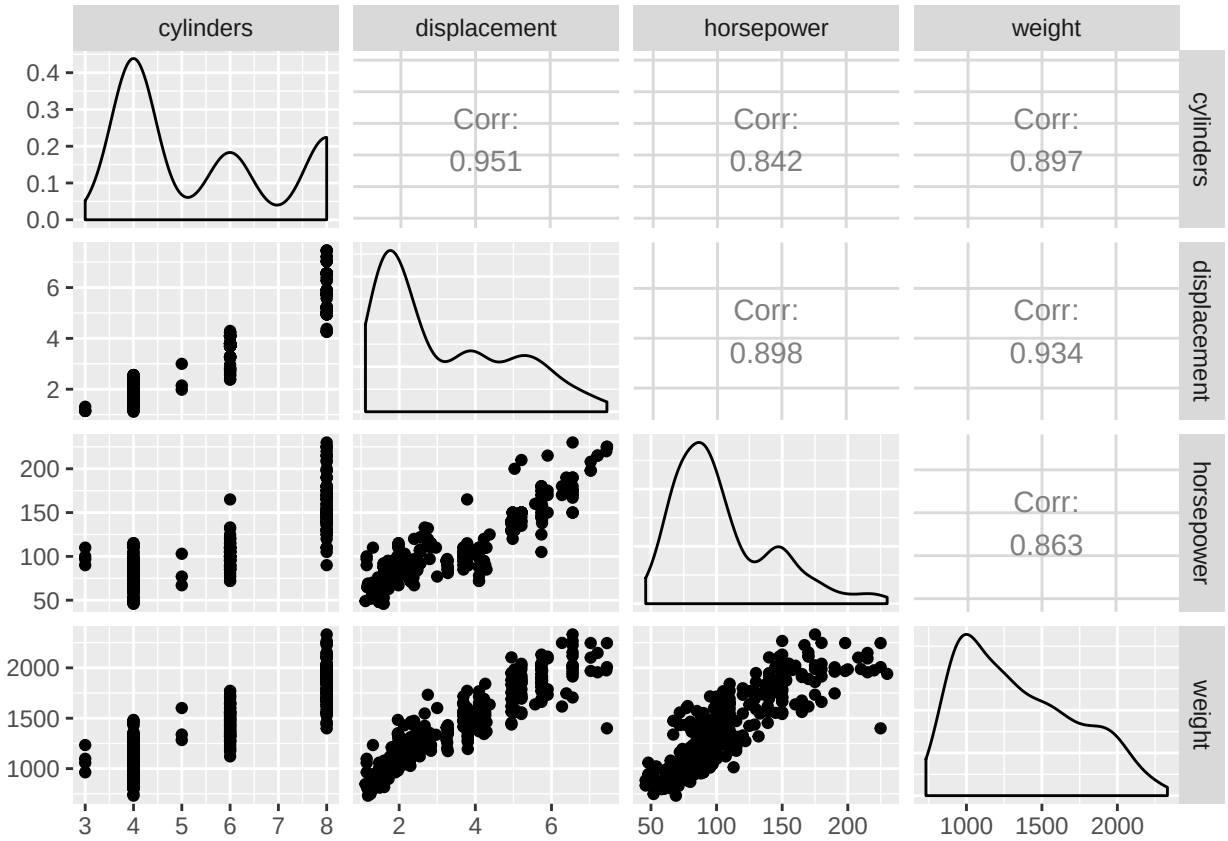
```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
ggpairs(data_mpghp[c("cylinders","displacement","horsepower","weight")])
```



Mezi těmito proměnnými je zřejmá kolinearita. Z grafů jde vidět, že pokud jedna z proměnných "roste", pak "roste" i druhá, což v podstatě sedí i s fyzikálním cítěním úlohy, kde se zdá, že se zvyšující se váhou vozidla musí růst i spotřeba. Totéž, pokud zvyšujeme počet válců, zdvihový objem čí výkon, pak se také zvyšuje spotřeba.

Z těchto proměnných bude tedy stačit uvažovat pouze jednu. Která to bude zjistíme postupným "vyhazováním" proměnných s nejvyšší VIF hodnotou. Tímto způsobem nakonec zůstane weight.

```
lmm1 <- lm(consumption ~ (.)-car_name-producer-displacement-horsepower-cylinders-constant_weight-consta
kappa(lmm1)
```

```
## [1] 30114.88
```

```
vif(lmm1)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## weight       1.969655  1        1.403444
## acceleration 1.377962  1        1.173866
## model_year   1.421003 12        1.014748
## origin       1.774554  2        1.154177
```

Podařilo se nám snížit číslo podmíněnosti kappa a také vidíme, že VIF hodnoty u zbývajících proměnných jsou dostatečně nízké, takže se nám kolinearitu podařilo tímto způsobem významně snížit.

Nyní se zaměříme na multikolinearitu v našem finálním modelu z otázky 16. Vypíšeme si číslo podmíněnosti matice regresorů kappa a VIF hodnoty v modelu.

```
kappa(lm_final_log)
```

```
## [1] 55185551
```

```
vif(lm_final_log)
```

```
##                                     GVIF Df GVIF^(1/(2*Df))
## cylinders                  8.236998e+02  1        28.70017
## displacement               3.579590e+03  1        59.82967
## horsepower                 4.231558e+02  1        20.57075
## weight                     2.963452e+03  1        54.43759
## acceleration               1.751337e+02  1        13.23381
## model_year                 6.070576e+29 12        17.41678
## origin                     4.710469e+04  2        14.73215
## cylinders:horsepower       1.538368e+03  1        39.22204
## cylinders:model_year       4.387903e+27 12        14.18276
## displacement:horsepower    1.923592e+03  1        43.85878
## displacement:acceleration  7.892236e+02  1        28.09312
## displacement:model_year    3.324349e+26 12        12.73712
## displacement:origin        2.119816e+04  2        12.06631
## horsepower:weight          2.183097e+03  1        46.72362
## horsepower:model_year      1.679846e+27 12        13.62656
## weight:acceleration        7.107462e+02  1        26.65982
## weight:model_year          1.100277e+30 12        17.85374
## weight:origin              8.106470e+04  2        16.87361
## acceleration:model_year    3.849834e+26 12        12.81524
## acceleration:origin        1.620149e+04  2        11.28207
```

Všechny hodnoty jsou velmi vysoké, multikolinearita v modelu je jasná a musíme ji co nejvíce snížit. Půjdeme na to tímto způsobem: iterativně budeme odebírat proměnné s nejvyšší VIF hodnotou. Někdy se stalo, že se VIF hodnoty pro více proměnných lišily např. na 3 desetinném míste -> zkusil jsem více možností zahazování a výsledné modely porovnal pomocí anova() threshold VIF hodnoty pro odebrání proměnné jsem nastavil na 3, ale v pozdějších fázích vyhazování se občas stalo, že jedna VIF hodnota byla nápadně vysoká proti ostatním a blízko 3 (2,8...), tak to jsem také zahodil. Celý postup byl velmi zdlouhavý. Uvedu finální model, který z této procedury vyšel nejlépe.

```
form = "log(consumption) ~ cylinders+ origin + acceleration+
horsepower:model_year + weight:acceleration"
lm_final = lm(form,data_mpghp)
vif(lm_final)
```

```
##                            GVIF Df GVIF^(1/(2*Df))
## cylinders                6.014522  1        2.452452
## origin                   1.966112  2        1.184137
## acceleration             7.552697  1        2.748217
## horsepower:model_year   10.781856 13        1.095769
## acceleration:weight      6.464143  1        2.542468
```

```
kappa(lm_final)
```

```
## [1] 233639.5
```

Vidíme, že kolinearitu v modelu se nám podařilo velmi výrazně snížit. Podíváme se na model pomocí summary funkce.

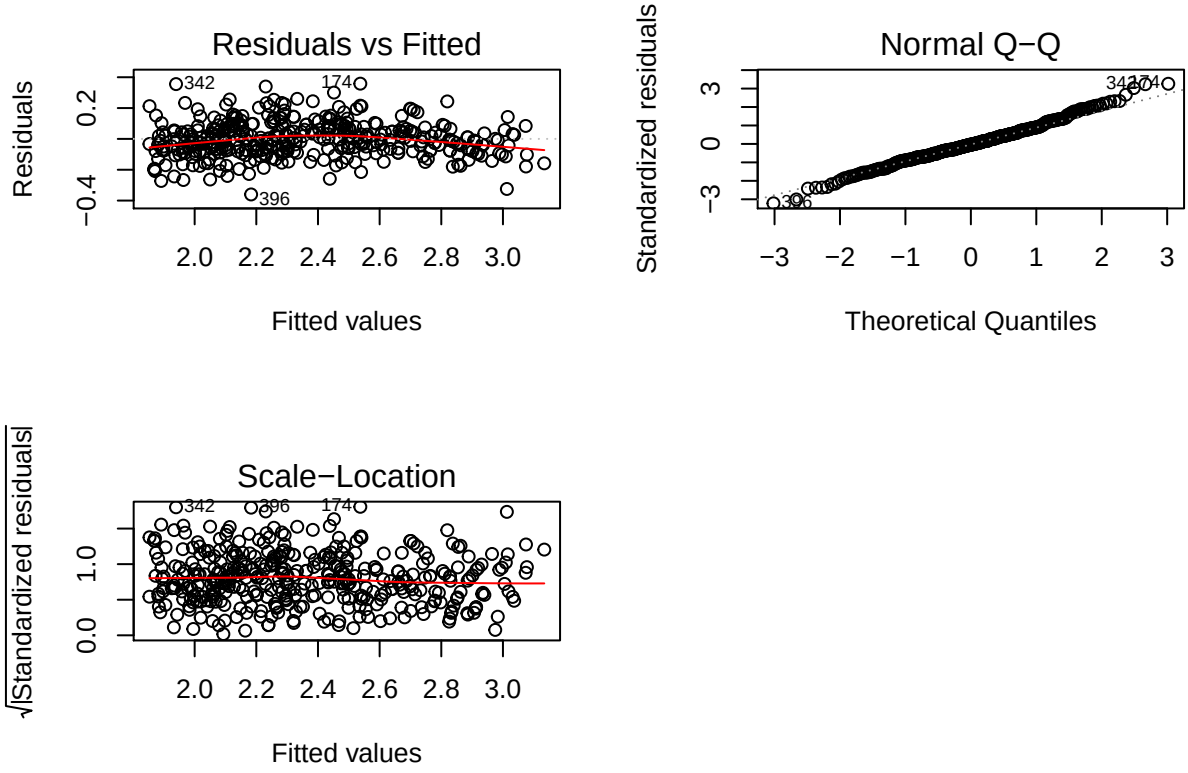
```
summary(lm_final)
```

```
##
## Call:
## lm(formula = form, data = data_mpghp)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36027 -0.07252 -0.00434  0.06539  0.35817
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               2.122e+00  9.746e-02  21.773  < 2e-16 ***
## cylinders                 8.184e-03  8.359e-03   0.979 0.328154
## origin2                  -6.179e-02  1.883e-02  -3.281 0.001131 **
## origin3                  -5.650e-02  1.888e-02  -2.992 0.002955 **
## acceleration             -3.656e-02  5.784e-03  -6.321 7.43e-10 ***
## horsepower:model_year70   2.381e-03  4.211e-04   5.654 3.13e-08 ***
## horsepower:model_year71   2.368e-03  5.070e-04   4.670 4.21e-06 ***
## horsepower:model_year72   2.629e-03  4.723e-04   5.567 4.98e-08 ***
## horsepower:model_year73   2.674e-03  4.500e-04   5.941 6.50e-09 ***
## horsepower:model_year74   2.096e-03  5.461e-04   3.838 0.000146 ***
## horsepower:model_year75   2.069e-03  5.409e-04   3.826 0.000153 ***
## horsepower:model_year76   2.029e-03  5.224e-04   3.884 0.000122 ***
## horsepower:model_year77   1.427e-03  5.055e-04   2.822 0.005027 **
## horsepower:model_year78   1.483e-03  5.050e-04   2.936 0.003535 **
## horsepower:model_year79   4.602e-04  5.359e-04   0.859 0.391015
## horsepower:model_year80  -1.084e-03  5.733e-04  -1.891 0.059465 .
## horsepower:model_year81  -2.409e-04  5.804e-04  -0.415 0.678375
## horsepower:model_year82  -8.181e-04  5.470e-04  -1.496 0.135596
## acceleration:weight       3.024e-05  2.733e-06  11.065  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1146 on 372 degrees of freedom
## Multiple R-squared:  0.8899, Adjusted R-squared:  0.8846
## F-statistic:   167 on 18 and 372 DF,  p-value: < 2.2e-16
```

Zkusíme tento model ještě validovat. Podíváme se na diagnostické ploty.

```
par(mfrow=c(2,2))
plot(lm_final,which=1)
plot(lm_final,which=2)
plot(lm_final,which=3)
```



Ploty vypadají hezky, i když homoskedasticita reziduí v residuals vs fitted plotu je mírně podezřelá.

```
shapiro.test(residuals(lm_final))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_final)
## W = 0.99357, p-value = 0.0954
```

Shapirův test nezamítá normalitu.

```
bptest(lm_final)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_final
## BP = 33.737, df = 18, p-value = 0.01357
```

Bptest těsně zamítá homoskedasticitu.

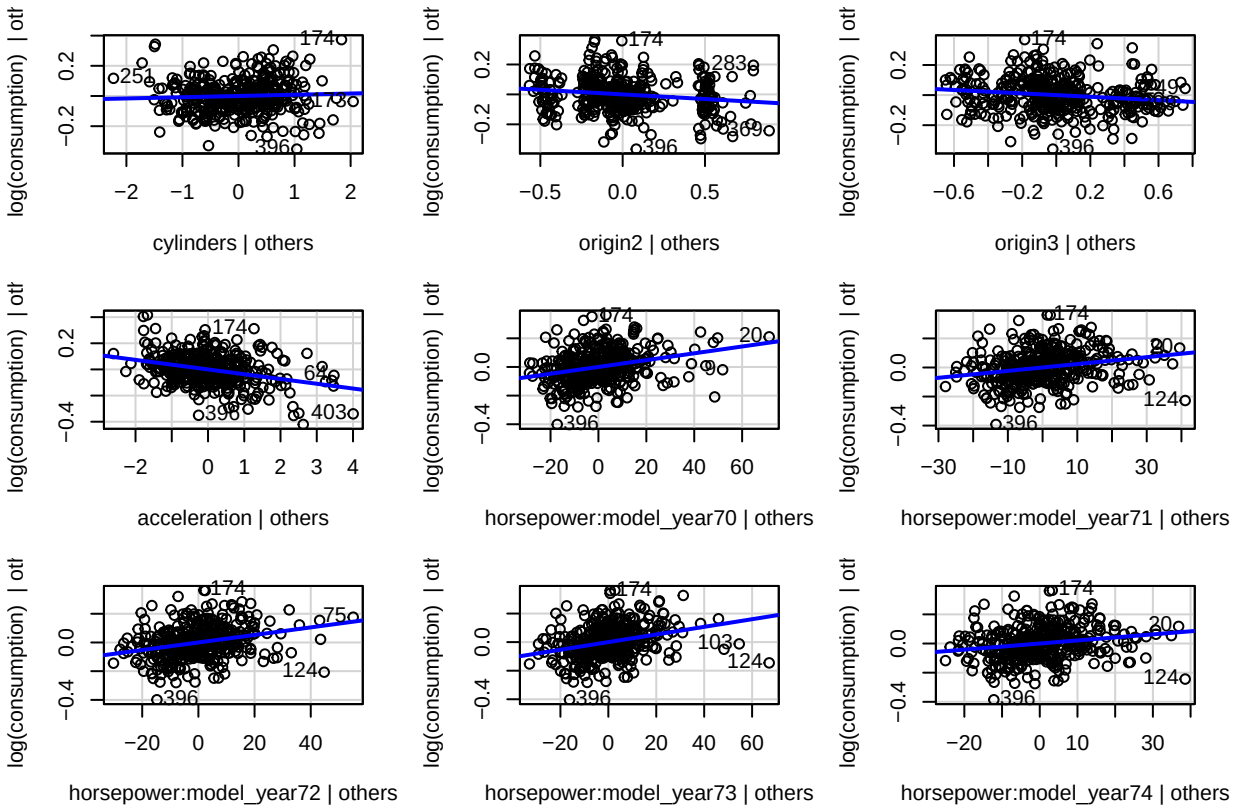
Otázka č.20: Vykreslete tzv. partial regression plot a tzv. partial residual plot pro finální model. Okomentujte, co nám zmíněné grafy říkají o výsledném modelu.

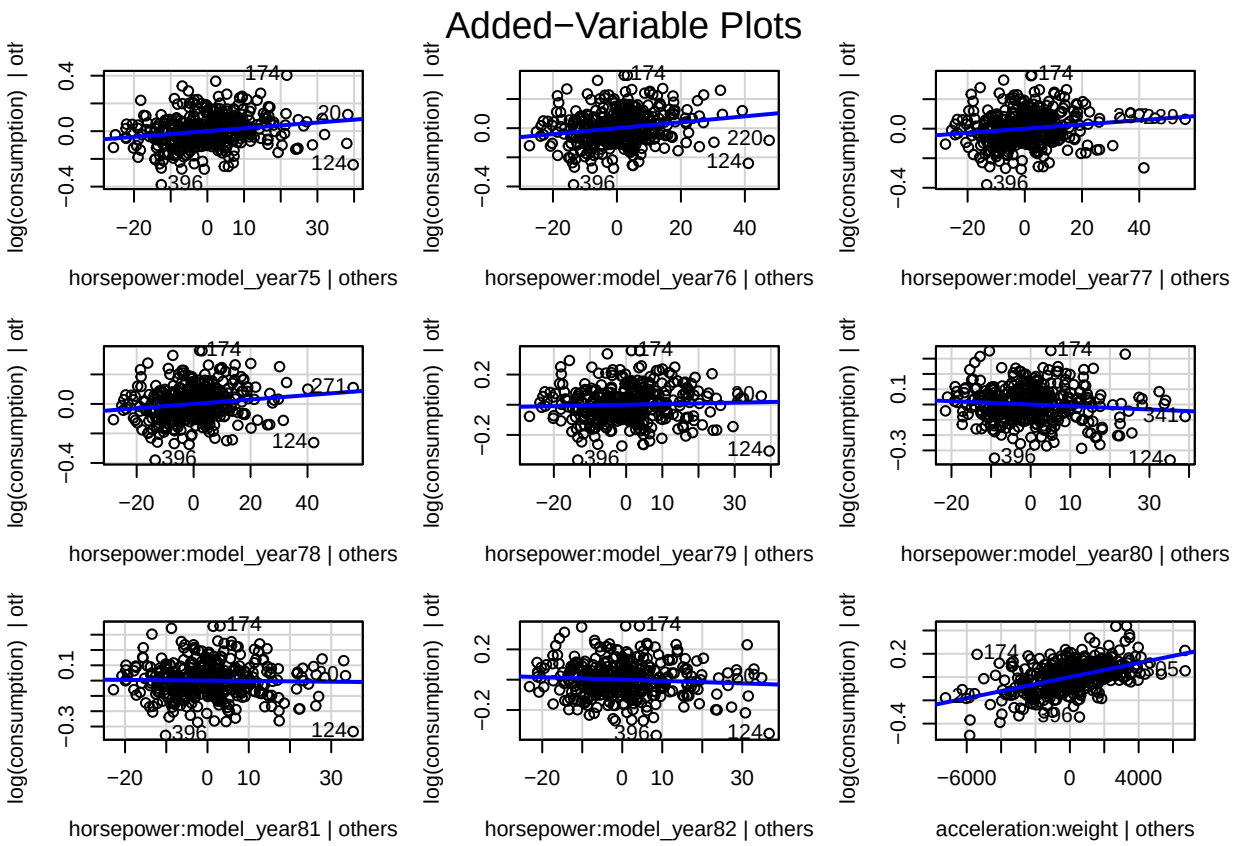
Tyto grafy znázorňují vliv jedné nezávislé proměnné na response variable při uvážení vlivu ostatních proměnných v modelu. Partial regression plot se hodí na detekci influenčních bodů, navíc sklon regresní přímky v

těchto plotech je stejný jako sklon dané proměnné v původním vícerozměrném lineárním modelu. Naproti tomu partial residual plot je vhodný k detekci nelinearity mezi danou proměnnou, může nám odhalit vhodné transformace nezávislých proměnných k zajištění či zlepšení linearity.

Jako první vykreslíme partial regression ploty.

```
library(car)
avPlots(lm_final)
```

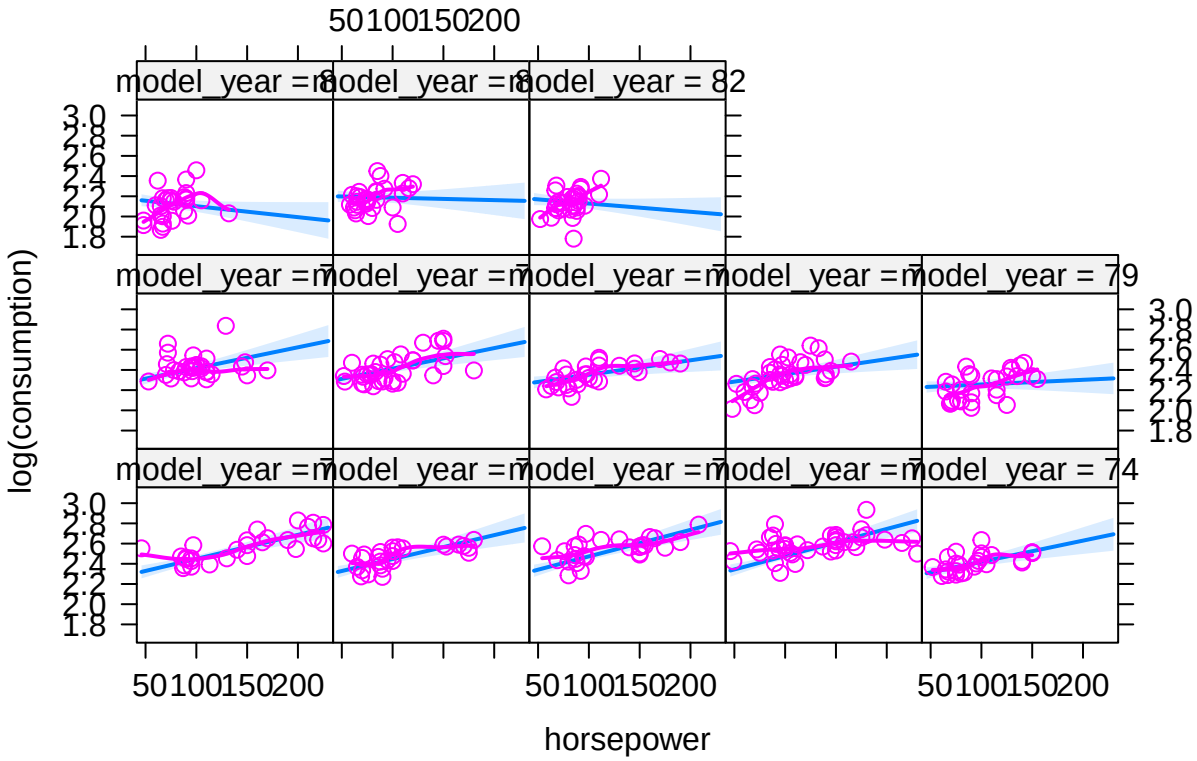




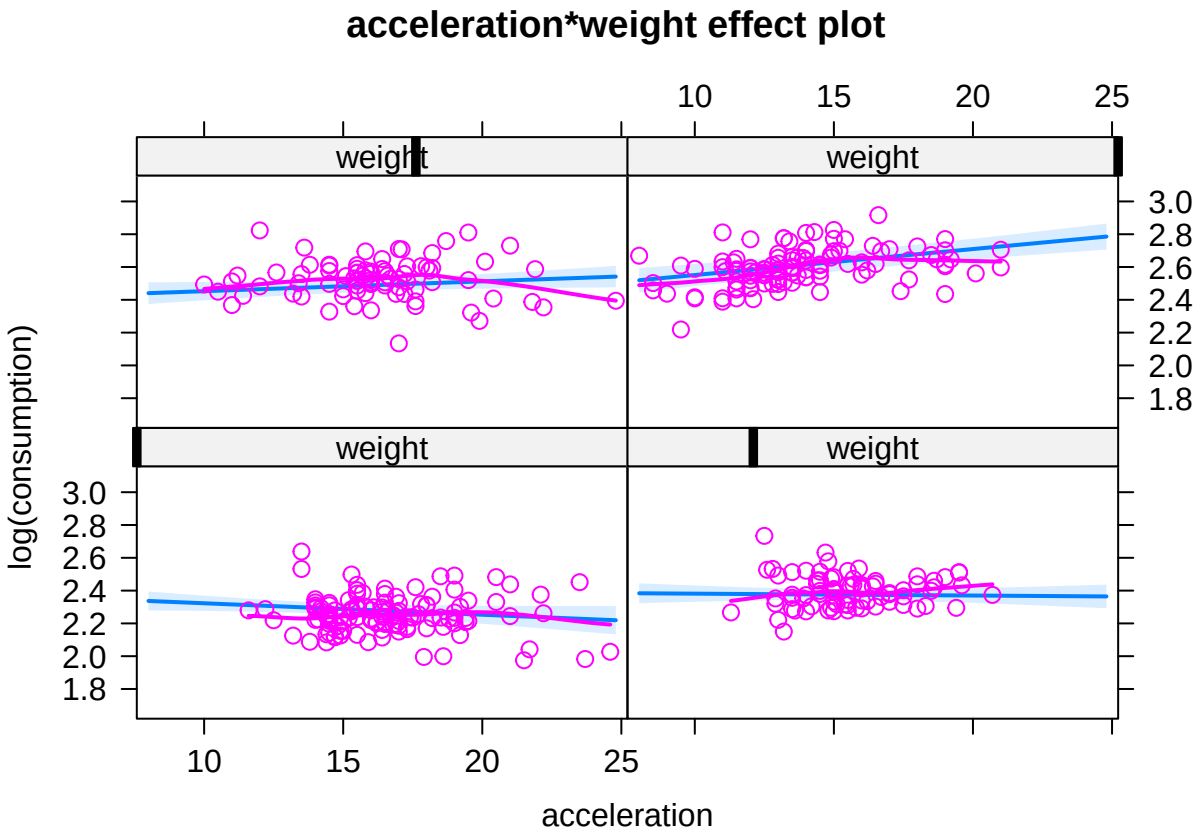
Vidíme, že několik bodů bylo znázorněno jako podezřelé influenční body, budeme se jimi zabývat v dalších otázkách.

Residual plot pomocí podobného balíčku vykreslit nemůžeme (nechce vykreslit pro model s interakcemi). Vykreslíme si interakce a faktorové proměnné alespoň pomocí balíčku effects, i když to není přesně to, co hledáme.

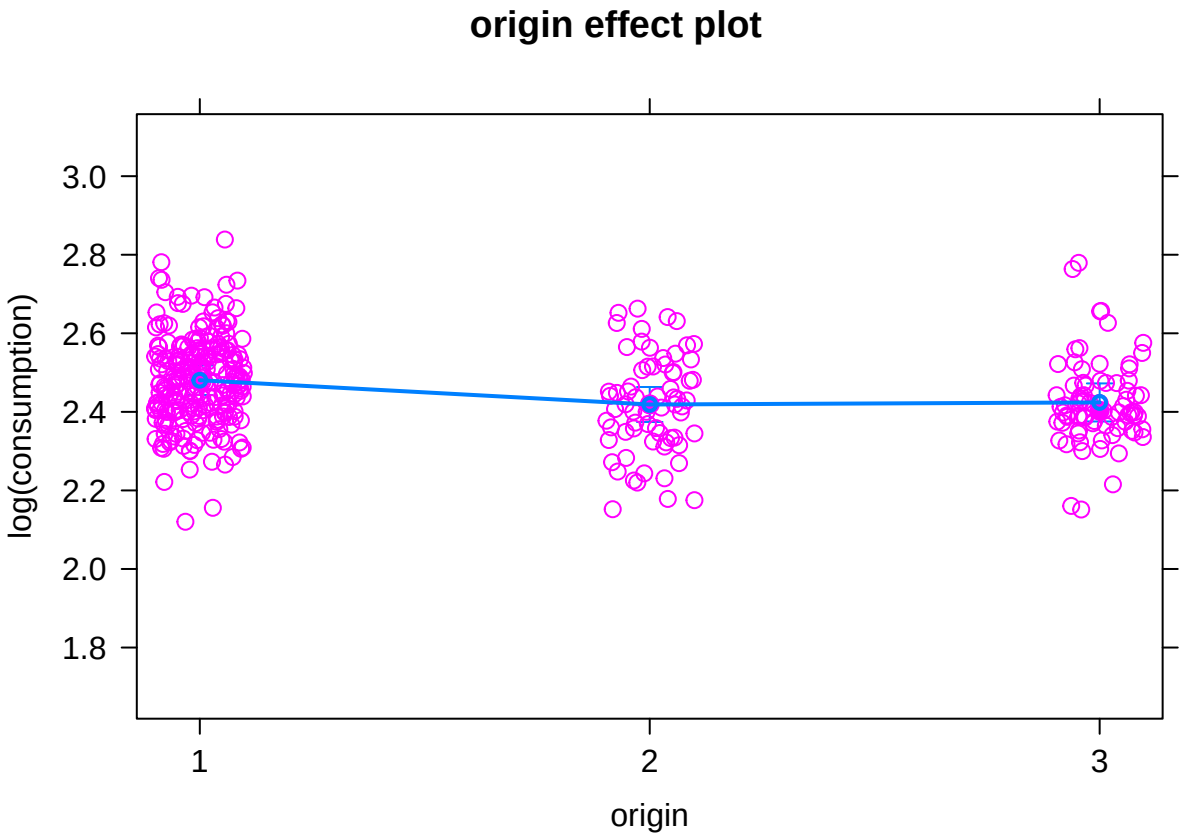
```
library(effects)
plot(Effect(c("horsepower","model_year"), lm_final, partial.residuals = TRUE), nrow = 1)
```




```
plot(Effect(c("acceleration","weight"), lm_final, partial.residuals = TRUE), nrow = 1)
```



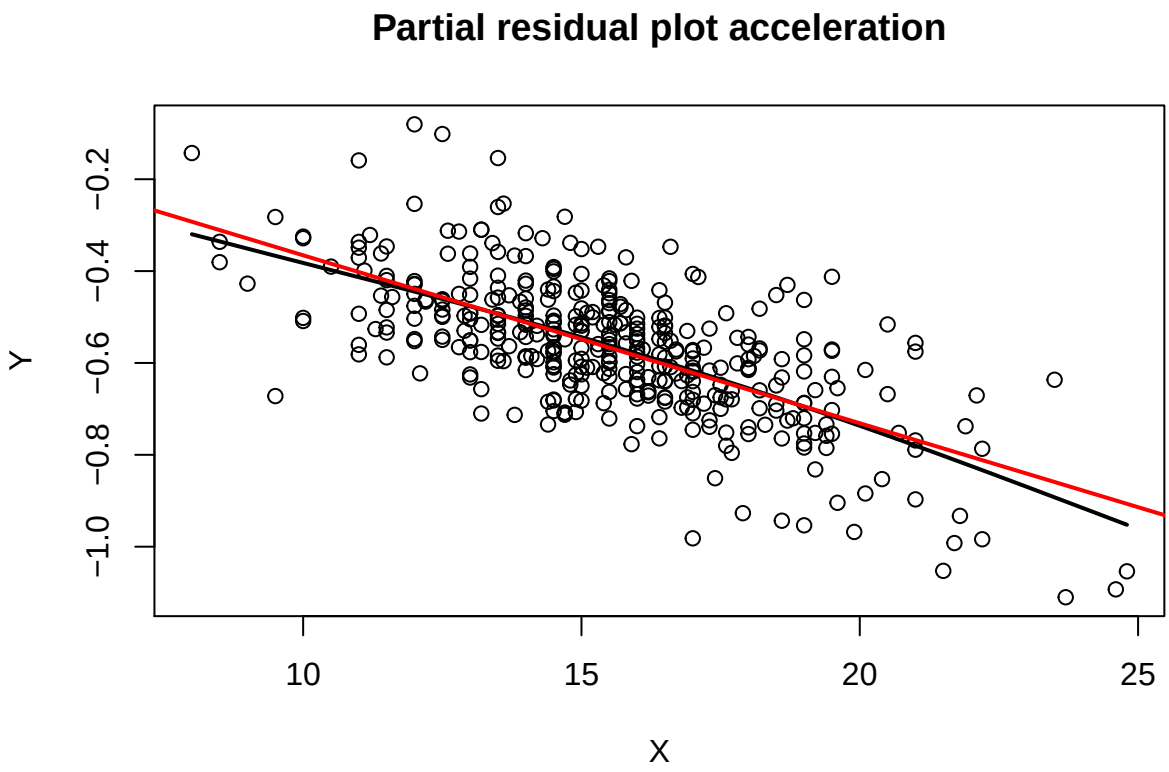

```
plot(Effect(c("origin"), lm_final, partial.residuals = TRUE), nrow = 1)
```



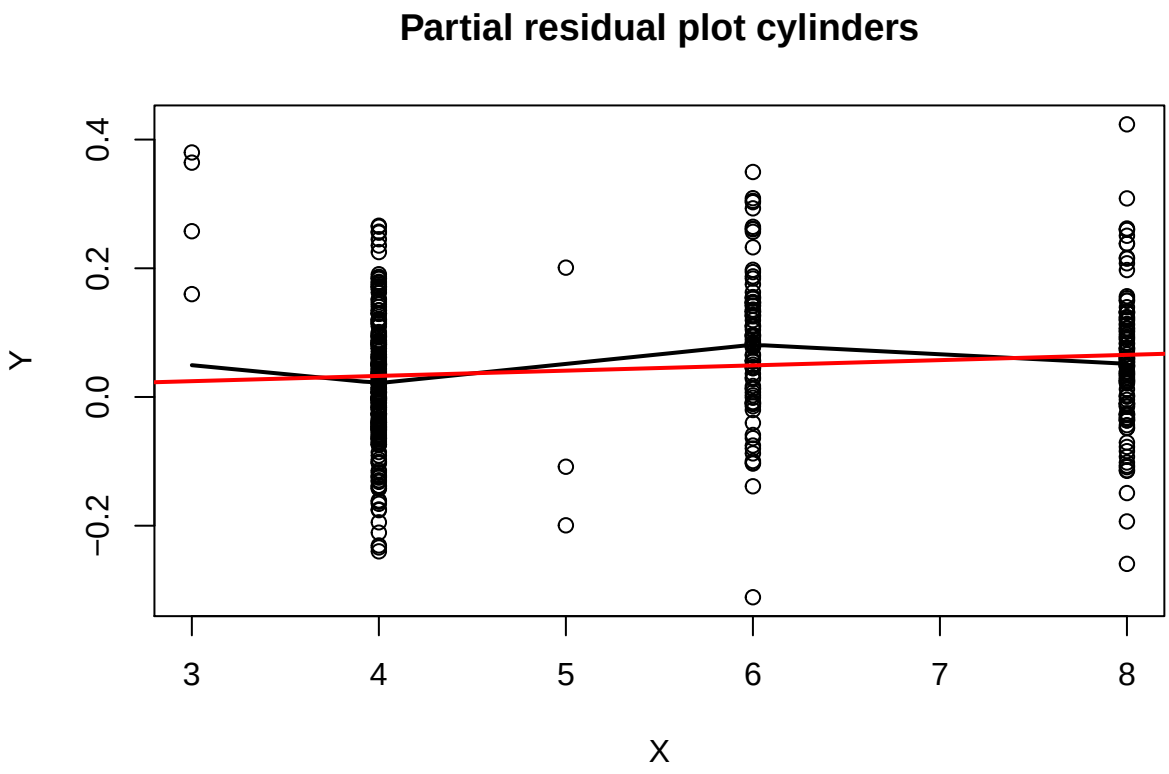
Vidíme, že origin by se teoreticky jako proměnná dala vynechat, jelikož střední hodnoty log(consumption) se pro jednotlivé faktory téměr rovnají a jednotlivé klastry pro každý faktor mají i přibližně stejný rozptyl. Zkusil jsem udělat tuto úpravu a origin z finálního modelu odstranit. Tahle úprava ale vedla k porušení podmínky normality reziduí, proto jsem se tím dál nezabýval.

Pro acceleration a cylinders vykreslíme partial residual ploty ručně.

```
coef_acceleration <- lm_final$coefficients['acceleration']
Y = residuals(lm_final) + coef_acceleration*data_mpghp[,'acceleration']
X = data_mpghp[,'acceleration']
plot(X,Y,main = 'Partial residual plot acceleration')
gamLine(x=X,y=Y) # smooth křivka
abline(lm(Y~X),col='red',lwd=2)
```




```
coef_cylinders <- lm_final$coefficients['cylinders']
Y = residuals(lm_final) + coef_cylinders*data_mpghp[,'cylinders']
X = data_mpghp[,'cylinders']
plot(X,Y,main='Partial residual plot cylinders')
quantregLine(x=X,y=Y)
abline(lm(Y~X),col='red',lwd=2)
```



Pro acceleration i cylinders vypadá linearita dobře. Z toho plyne, že není potřeba aplikovat nějaké transformace vysvětlujících proměnných.

Otázka č.21: Prezentujte váš výsledný model pro predikci consumption, diskutujte výsledné parametry R^2 a sigma tohoto modelu. Pokud jste model nevalidovali v předchozích krocích, tak ho validujte (jak graficky, tak pomocí příslušných testů hypotéz).

Výsledný model pro predikci je

```
summary(lm_final)
```

```
##
## Call:
## lm(formula = form, data = data_mpghp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36027 -0.07252 -0.00434  0.06539  0.35817
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             2.122e+00  9.746e-02  21.773  < 2e-16 ***
## cylinders               8.184e-03  8.359e-03   0.979 0.328154
## origin2                -6.179e-02  1.883e-02  -3.281 0.001131 **
## origin3                -5.650e-02  1.888e-02  -2.992 0.002955 **
## acceleration           -3.656e-02  5.784e-03  -6.321 7.43e-10 ***
```

```
## horsepower:model_year70  2.381e-03  4.211e-04   5.654 3.13e-08 ***
## horsepower:model_year71  2.368e-03  5.070e-04   4.670 4.21e-06 ***
## horsepower:model_year72  2.629e-03  4.723e-04   5.567 4.98e-08 ***
## horsepower:model_year73  2.674e-03  4.500e-04   5.941 6.50e-09 ***
## horsepower:model_year74  2.096e-03  5.461e-04   3.838 0.000146 ***
## horsepower:model_year75  2.069e-03  5.409e-04   3.826 0.000153 ***
## horsepower:model_year76  2.029e-03  5.224e-04   3.884 0.000122 ***
## horsepower:model_year77  1.427e-03  5.055e-04   2.822 0.005027 **
## horsepower:model_year78  1.483e-03  5.050e-04   2.936 0.003535 **
## horsepower:model_year79  4.602e-04  5.359e-04   0.859 0.391015
## horsepower:model_year80 -1.084e-03  5.733e-04  -1.891 0.059465 .
## horsepower:model_year81 -2.409e-04  5.804e-04  -0.415 0.678375
## horsepower:model_year82 -8.181e-04  5.470e-04  -1.496 0.135596
## acceleration:weight      3.024e-05  2.733e-06  11.065  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1146 on 372 degrees of freedom
## Multiple R-squared:  0.8899, Adjusted R-squared:  0.8846
## F-statistic:   167 on 18 and 372 DF,  p-value: < 2.2e-16
```

Model byl v předchozích krocích validován, normalita reziduí je v pořádku, homoskedasticita vypadá z diagnostických grafů dobře, i když residual vs fitted plot obsahuje minimální náznak nějakého "pevného" vzoru, což by mohlo indikovat lehkou heteroskedasticitu. Hodnota R^2 statistiky je 0.8899 (adjusted R^2 je 0.8846), což je považováno za velmi dobrou hodnotu. Hodnota sigma modelu je přibližně 0.114637.

```
sigma(lm_final)
```

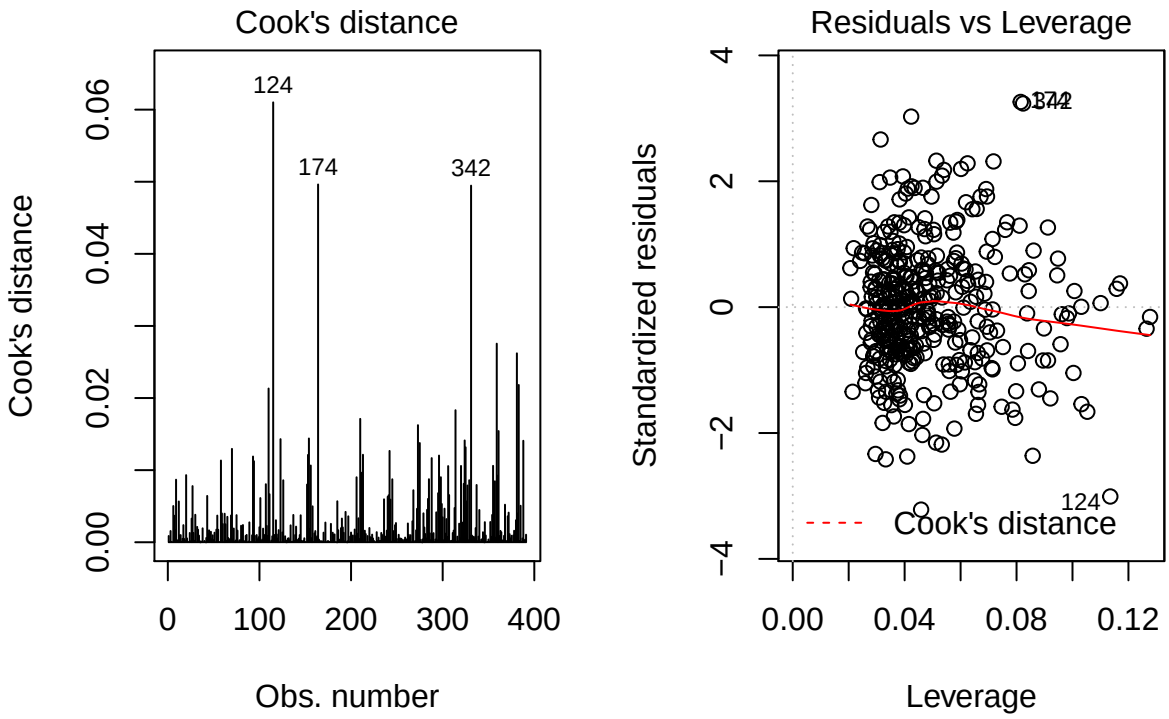
```
## [1] 0.114637
```

Model odhaduje nelineárně transformovanou veličinu log(consumption), což výrazně stěžuje interpretabilitu "kvality" regrese v případě netransformované veličiny consumption. To je bohužel nevýhoda tohoto modelu.

Otázka č.22: Obsahuje Váš model z bodu 19 nějaká vlivná pozorování? Pokuste se detekovat odlehlé a pákové body pomocí různých diagnostických nástrojů tzv. leave-one-out deletion regression.

Podiváme se na cook distance a residuals vs leverage

```
par(mfrow=c(1,2))
plot(lm_final, which = 4)
plot(lm_final, which = 5)
```



Vlivná pozorování jsou v plotech vyznačena 124,174 a 342.

Vypíšeme influence.measures (leave one out regression) a podíváme se na podezřelé influenční body a jejich vliv na různé uvažované metriky (dffit,..).

```
summary(influence.measures(lm_final))
```

```
## Potentially influential observations of
##   lm(formula = form, data = data_mpghp) :
##
##     dfb.1_ dfb.cyln dfb.org2 dfb.org3 dfb.accl dfb.h:_70 dfb.h:_71 dfb.h:_72
## 20  -0.06   0.00    -0.01    -0.02     0.07     0.10      0.07      0.07
## 50   0.00   0.00     0.00     0.00     0.00     0.00      0.00      0.00
## 75  -0.02  -0.01     0.00    -0.01     0.02     0.03      0.04      0.07
## 119  0.33  -0.34    -0.09     0.23    -0.28    -0.07     -0.10     -0.09
## 124  0.40   0.13     0.06     0.14    -0.43    -0.58     -0.59     -0.60
## 145  0.00   0.00     0.00     0.00     0.00     0.00      0.00      0.00
## 146  0.00   0.00     0.00     0.00     0.00     0.00      0.00      0.00
## 148  0.01   0.00     0.00     0.00    -0.01    -0.01     -0.01     -0.01
## 163  0.02   0.12    -0.08    -0.03    -0.01    -0.17     -0.18     -0.19
## 174 -0.18   0.46     0.00    -0.11     0.22    -0.03      0.02      0.03
## 237  0.00   0.00     0.00     0.00     0.00     0.00      0.01      0.01
## 239  0.02   0.00     0.00     0.00    -0.01    -0.02     -0.02     -0.02
## 252  0.26  -0.21    -0.19     0.05    -0.29    -0.07     -0.07     -0.07
## 255  0.12  -0.04     0.02    -0.17    -0.13    -0.07     -0.06     -0.05
## 271 -0.02  -0.02    -0.01    -0.01     0.02     0.04      0.04      0.04
## 330  0.07  -0.09     0.03    -0.14    -0.07     0.01      0.01      0.01
```

```
## 341  0.01  -0.02    -0.01    -0.02     0.00    -0.01     -0.01     -0.01
## 342  0.29  -0.37    -0.09     0.14    -0.29     0.04      0.03      0.03
## 373  0.18  -0.26    -0.07     0.00    -0.09     0.08      0.07      0.08
## 375  0.15   0.01    -0.03    -0.05    -0.18    -0.19     -0.20     -0.20
## 396  0.00  -0.25    -0.05     0.01     0.04     0.21      0.20      0.20
##     dfb.h:_73 dfb.h:_74 dfb.h:_75 dfb.h:_76 dfb.h:_77 dfb.h:_78 dfb.h:_79
## 20   0.07      0.07      0.07      0.07      0.07      0.07      0.07
## 50   0.00      0.00      0.00      0.00      0.00      0.00      0.00
## 75   0.04      0.04      0.04      0.04      0.04      0.04      0.04
## 119  0.01     -0.10     -0.10     -0.09     -0.10     -0.10     -0.08
## 124 -0.85     -0.59     -0.61     -0.60     -0.59     -0.60     -0.60
## 145  0.00     -0.01      0.00      0.00      0.00      0.00      0.00
## 146  0.00     -0.02      0.00      0.00      0.00      0.00      0.00
## 148 -0.01     -0.03     -0.01     -0.01     -0.01     -0.01     -0.01
## 163 -0.19     -0.19     -0.07     -0.19     -0.19     -0.19     -0.20
## 174  0.01      0.05      0.35      0.04      0.04      0.03      0.02
## 237  0.01      0.01      0.01      0.01      0.01      0.01      0.01
## 239 -0.02     -0.02     -0.02     -0.02     -0.04     -0.02     -0.02
## 252 -0.08     -0.06     -0.07     -0.06     -0.07     -0.12     -0.05
## 255 -0.06     -0.04     -0.05     -0.05     -0.05     -0.14     -0.05
## 271  0.04      0.04      0.04      0.04      0.04      0.07      0.04
## 330  0.01      0.02      0.01      0.01      0.01      0.02      0.01
## 341 -0.01     -0.01     -0.01     -0.01     -0.01     -0.01     -0.01
## 342  0.04      0.04      0.04      0.05      0.03      0.04      0.06
## 373  0.09      0.08      0.08      0.09      0.07      0.07      0.09
## 375 -0.21     -0.20     -0.21     -0.20     -0.20     -0.20     -0.20
## 396  0.21      0.19      0.20      0.21      0.20      0.20      0.21
##     dfb.h:_80 dfb.h:_81 dfb.h:_82 dfb.acc: dffit    cov.r    cook.d hat
## 20   0.07      0.07      0.07     -0.07     0.14     1.18_*   0.00   0.12
## 50   0.00      0.00      0.00      0.00     0.00     1.17_*   0.00   0.10
## 75   0.04      0.04      0.04     -0.02     0.11     1.19_*   0.00   0.12
## 119 -0.16     -0.14     -0.15      0.25     0.64     0.68_*   0.02   0.04
## 124 -0.57     -0.58     -0.57      0.45    -1.09_*   0.74_*   0.06   0.11
## 145  0.00      0.00      0.00     -0.01    -0.03     1.17_*   0.00   0.10
## 146  0.00      0.00      0.00      0.00    -0.04     1.16_*   0.00   0.10
## 148 -0.01     -0.01     -0.01      0.00    -0.06     1.17_*   0.00   0.10
## 163 -0.17     -0.19     -0.19      0.08     0.48     0.75_*   0.01   0.03
## 174  0.09      0.05      0.07     -0.44     0.98_*   0.66_*   0.05   0.08
## 237  0.01      0.01      0.01     -0.01     0.02     1.18_*   0.00   0.11
## 239 -0.02     -0.02     -0.02      0.01    -0.06     1.21_*   0.00   0.13
## 252 -0.05     -0.06     -0.06      0.24    -0.49     0.82_*   0.01   0.04
## 255 -0.01     -0.02     -0.03      0.09    -0.41     0.82_*   0.01   0.03
## 271  0.04      0.04      0.04     -0.02     0.09     1.17_*   0.00   0.10
## 330 -0.16      0.03      0.02      0.06    -0.45     0.81_*   0.01   0.03
## 341 -0.07     -0.01     -0.02      0.02    -0.13     1.20_*   0.00   0.13
## 342  0.41      0.02      0.01      0.22     0.98_*   0.67_*   0.05   0.08
## 373  0.03     -0.22      0.03     -0.01    -0.73_*   0.86     0.03   0.09
## 375 -0.17      0.03     -0.18      0.20     0.54     0.84_*   0.02   0.05
## 396  0.15      0.17     -0.14     -0.04    -0.71_*   0.65_*   0.03   0.05
```

Z tabulky můžeme vypozorovat, že všechny podezřelé body ovlivňují především covariance ratio a hlavní podezřelé body z regression vs leverage plotu také kvalitu fitu (dffits kritérium)

Vzhledem k tabulce se jako nejvíce podezřelé body jeví 124,174,342 , které mají také největší cook distance. Tato 3 pozorování jsou adepty na odstranění.

Otázka č.23: Pokud jste odhalili nějaká vlivná pozorování, jak byste s nimi naložili a proč?

Našli jsme několik influenčních bodů vzhledem k leave one out regression. Po prozkoumání a srovnání podezřelých hodnot s ostatními (např. s podobnou vahou, výkonem,...) jsem se nakonec rozhodl data z finálního modelu neodstraňovat ani nemodifikovat, protože se nezdá, že by tato pozorování byla ovlivněna nějakou systematickou chybou či by se až příliš nepravděpodobně odchylovala od ostatních záznamů.

Otázka č.24: Porovnejte regresní koeficienty, které jste obdrželi z výsledného klasického lineárního modelu (s a bez odhledlých pozorování) s robustními modely. Vyzkoušejte MM odhad (pro dva druhy funkce psi) a LTS odhad při použití (90% a 50% pozorování).

První nastavíme nový dataset bez hlavních influenčních bodů (124,174,342).

```
data_noinfluential = data_mpghp[!(rownames(data_mpghp) %in% c("124","174","342")),]
```

Finální klasický model jsme již v druhé části protokolu vybrali model s logaritmicky škálovanou spotřebou umocněnou na druhou.

```
lm_logsquared_noinf = lm(log(consumption)^2 ~ weight,data_noinfluential)
```

Nachystáme robustní modely. V MM modelech použijeme jako psi ggw a bisquare.

```
library(robustbase)
library(rrcov)
```

```
## Scalable Robust Estimators with High Breakdown Point (version 1.4-9)
```

```
library(robust)
```

```
## Loading required package: fit.models
```

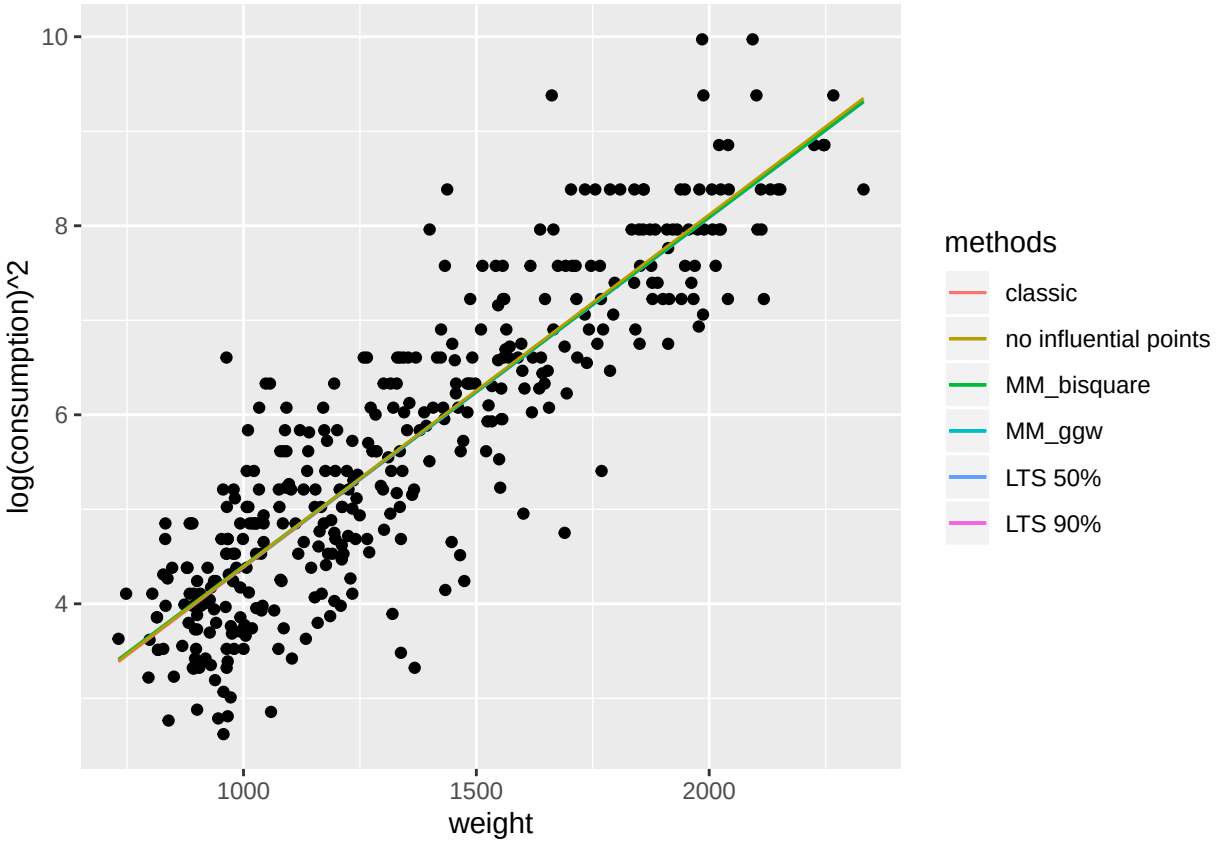
```
MM_bisquare_stars<- rlm(log(consumption)^2 ~ weight, method="MM",psi = psi.bisquare, data = data_mpghp)
MM_ggw   <- rlm(log(consumption)^2 ~ weight, method="MM",psi = psi.ggw, data= data_mpghp)
```

A LTS s 50% a 90% pozorování.

```
LTS_50  <- ltsReg(log(consumption)^2 ~ weight, alpha=0.5, data = data_mpghp)
LTS_90  <- ltsReg(log(consumption)^2 ~ weight, alpha=0.9, data = data_mpghp)
```

Nyní vykreslíme scatter plot a regresní přímky všech modelů.

```
p <- ggplot(data, aes(x=weight, y=log(consumption)^2)) + geom_point()+
  geom_line(aes(x=weight,y=lm_logcons_squared$fitted.values,color = 'red'))+
  geom_line(data = data_noinfluential,aes(x=weight,y=lm_logsquared_noinf$fitted.values,color='blue'))+
  geom_line(aes(x=weight,y=MM_bisquare_stars$fitted.values,color='yellow')) +
  geom_line(aes(x=weight,y=MM_ggw$fitted.values,color='orange')) +
  geom_line(aes(x=weight,y=LTS_50$fitted.values,color='green')) +
  geom_line(aes(x=weight,y=LTS_90$fitted.values,color='brown')) +
  scale_color_discrete(name = 'methods',labels = c('classic','no influential points','MM_bisquare','MM_
plot(p)
```

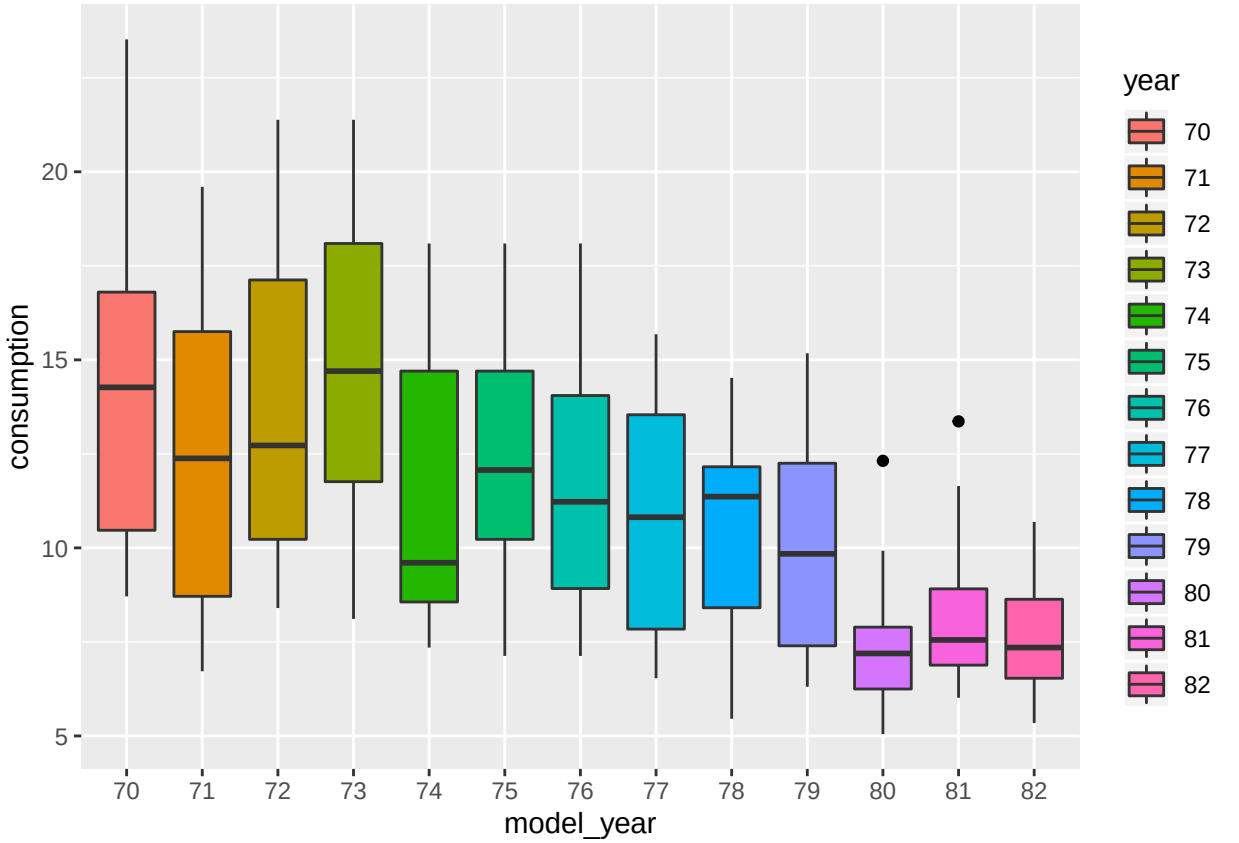


Regresní přímky jsou v podstatě identické. Pokud by existovaly opravdu silné influenční body, které by odchylovaly přímku z klasického modelu, pak by se klasický model a robustní modely více lišily.

Otázka č.25: Diskutujte, jak by šlo případně zlepšit predikci, jaké transformace jednotlivých proměnných by mohli pomoci. Převedli byste některé další spojité proměnné na diskrétní (na faktory)? Jaké další kroky byste při analýze navrhli?

Pokud se podíváme na signifikanci jednotlivých proměnných, vidíme, že horsepower:model_year79 až horsepower:model_year 82 jsou méně signifikantní než ostatní, navíc jsme v předchozích otázkách zjistili (z grafů), že by se proměnná model_year dala sloučit na méně faktorů. V tomto případě by možná mohlo pomoci nechat všechny faktory až na 80-82, které bychom sloučily dohromady. Rok 79 bych tam již nepřidával, hodnoty consumption pro tento faktor (a tedy střední hodnota) se od 80-82 značně liší, viz obrázek.

```
p1 <- ggplot(data_mpghp,aes(x=model_year,y=consumption, fill = model_year)) +
  geom_boxplot(notch=FALSE,outlier.color = 'black') + scale_fill_discrete(name='year')
plot(p1)
```



Co se týče transformací nezávislých proměnných, tak z partial residual plotu jsme nezaznamenali nějaké podezřelé nelineární chování, tím pádem bych transformace nevyužíval. Pokud se podíváme ještě jednou na kolinearitu v modelu

```
vif(lm_final)
```

```
##                              GVIF Df GVIF^(1/(2*Df))
## cylinders                6.014522  1        2.452452
## origin                   1.966112  2        1.184137
## acceleration             7.552697  1        2.748217
## horsepower:model_year   10.781856 13        1.095769
## acceleration:weight      6.464143  1        2.542468
```

Vidíme, že acceleration by mohlo být korelované s acceleration:weight. Zkusíme proto acceleration odstranit (má vyšší VIF než acceleration:weight).

```
lm_no_acceleration = lm(log(consumption)~cylinders + origin+horsepower:model_year+acceleration:weight,da
summary(lm_no_acceleration)
```

```
##
## Call:
## lm(formula = log(consumption) ~ cylinders + origin + horsepower:model_year +
##     acceleration:weight, data = data_mpghp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42081 -0.07311  0.00073  0.06991  0.41646
##
```

```
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.549e+00  3.745e-02  41.354  < 2e-16 ***
## cylinders                 2.442e-02  8.360e-03   2.921 0.003700 **
## origin2                  -7.147e-02  1.972e-02  -3.624 0.000331 ***
## origin3                  -8.080e-02  1.943e-02  -4.159 3.97e-05 ***
## horsepower:model_year70   4.195e-03  3.238e-04  12.957  < 2e-16 ***
## horsepower:model_year71   4.548e-03  3.905e-04  11.647  < 2e-16 ***
## horsepower:model_year72   4.689e-03  3.593e-04  13.049  < 2e-16 ***
## horsepower:model_year73   4.685e-03  3.345e-04  14.008  < 2e-16 ***
## horsepower:model_year74   4.375e-03  4.311e-04  10.147  < 2e-16 ***
## horsepower:model_year75   4.427e-03  4.116e-04  10.756  < 2e-16 ***
## horsepower:model_year76   4.258e-03  4.051e-04  10.512  < 2e-16 ***
## horsepower:model_year77   3.600e-03  3.895e-04   9.241  < 2e-16 ***
## horsepower:model_year78   3.615e-03  3.949e-04   9.155  < 2e-16 ***
## horsepower:model_year79   2.665e-03  4.276e-04   6.231 1.25e-09 ***
## horsepower:model_year80   1.238e-03  4.627e-04   2.675 0.007809 **
## horsepower:model_year81   2.098e-03  4.700e-04   4.463 1.07e-05 ***
## horsepower:model_year82   1.369e-03  4.452e-04   3.076 0.002251 **
## acceleration:weight       1.558e-05  1.519e-06  10.255  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1205 on 373 degrees of freedom
## Multiple R-squared:  0.8781, Adjusted R-squared:  0.8725
## F-statistic:   158 on 17 and 373 DF,  p-value: < 2.2e-16
```

Dostali jsme model, kde je o něco nižší kolinearita než v původním modelu a všechny použité nezávislé proměnné jsou značně signifikantní. Porovnáme pomocí anova() tento model s původním.

```
anova(lm_no_acceleration,lm_final)
```

```
## Analysis of Variance Table
##
## Model 1: log(consumption) ~ cylinders + origin + horsepower:model_year +
##     acceleration:weight
## Model 2: log(consumption) ~ cylinders + origin + acceleration + horsepower:model_year +
##     weight:acceleration
##   Res.Df    RSS Df Sum of Sq     F    Pr(>F)
## 1    373 5.4138
## 2    372 4.8887  1   0.52514 39.96 7.426e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vidíme, že anova preferuje více původní model.

```
shapiro.test(residuals(lm_no_acceleration))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lm_no_acceleration)
## W = 0.98797, p-value = 0.002576
```

Navíc vidíme, že Shapirův test zamítá normalitu, tudíž tento model nesplňuje OLS předpoklady.

Otázka č.26: Představte si, že máte flotilu aut složenou z dostupných dat. Bez ohledu na nosnost auta (předpokládejme, že to zvládnou všechny) potřebujete převést pro známého z Brna hl. nádraží do Prahy do Trojanovy 13, 800kg nákladu. Jaké auto si vyberete a proč? O kolik se vám cesta prodraží a tudíž by vám měl kamarád zaplatit (neuvažujte amortizaci auta a vezměte cenu 30kč za palivo).

Je zřejmé, že nemůžeme jen tak předpokládat pro libovolné auto, že pokud na něj naložíme nějakou zátěž, tak se jeho spotřeba bude chovat lineárně podle námi nalezeného modelu. Musíme na to jít jinak.

Můj postup by byl takovýto:

1. vyrobíme jednoduchý klasický model (lm(consumption ~ weight - 1)- použijeme model bez interceptu, protože auto vážící 0kg logicky nespotřebuje nic, tedy 0l paliva), který nám dá lineární odhad závislosti spotřeby na váze
2. vytvoříme podobné skupiny aut vzhledem k několika nezávislým proměnným určujícím spotřebu (např. pomocí K-Means algoritmu, určitě by to šlo udělat i důkladněji a jinak) , za tyto proměnné zvolíme cylinders,horsepower,acceleration,displacement (zajímají mě především klastry na základě technických parametrů aut, proto origin ani model_year uvažovat nebudeme)
3. jako poslední krok se podíváme na rezidua u jednotlivých skupin a zjistíme, jak moc přesně tyto skupiny opisují regresní přímku danou modelem a vezmeme tu skupinu, která bude mít nejmenší rezidua (suma reziduí^2 normovaná počtem členů ve skupině) a vizuálně bude opisovat nejlépe regresní přímku. U takových skupin lze očekávat velmi podobné hodnoty v ostatních proměnných (cylinders,horsepower,acceleration,displacement) a tedy můžeme předpokládat, že u aut s takovými technickými parametry se závislost spotřeby na váze automobilu chová "přibližně" lineárně - hrubě si to představuji tak, že zafixujeme ostatní proměnné kromě weight a sleduju změnu spotřeby jen na základě přidávání váhy. Nakonec vybereme jedno auto z této skupiny dle uvážení. Měli bychom vybrat nejlepšího reprezentanta dané skupiny (vektor nejblíže ke středu klastru), pokud bude takových aut více, pak vezmeme to nejlehčí, protože pak máme větší jistotu, že přidáním největší váhy se spotřeba změní lineárně (pokud v této skupině máme nejlehčí auto vážící 1000kg a nejtěžší 1500kg, pak u 1000kg auta naložením 500kg ještě stále můžu předpokládat lineární závislost spotřeby).

Rozdělíme do 20 klastrů pomocí KMeans a nastavíme model

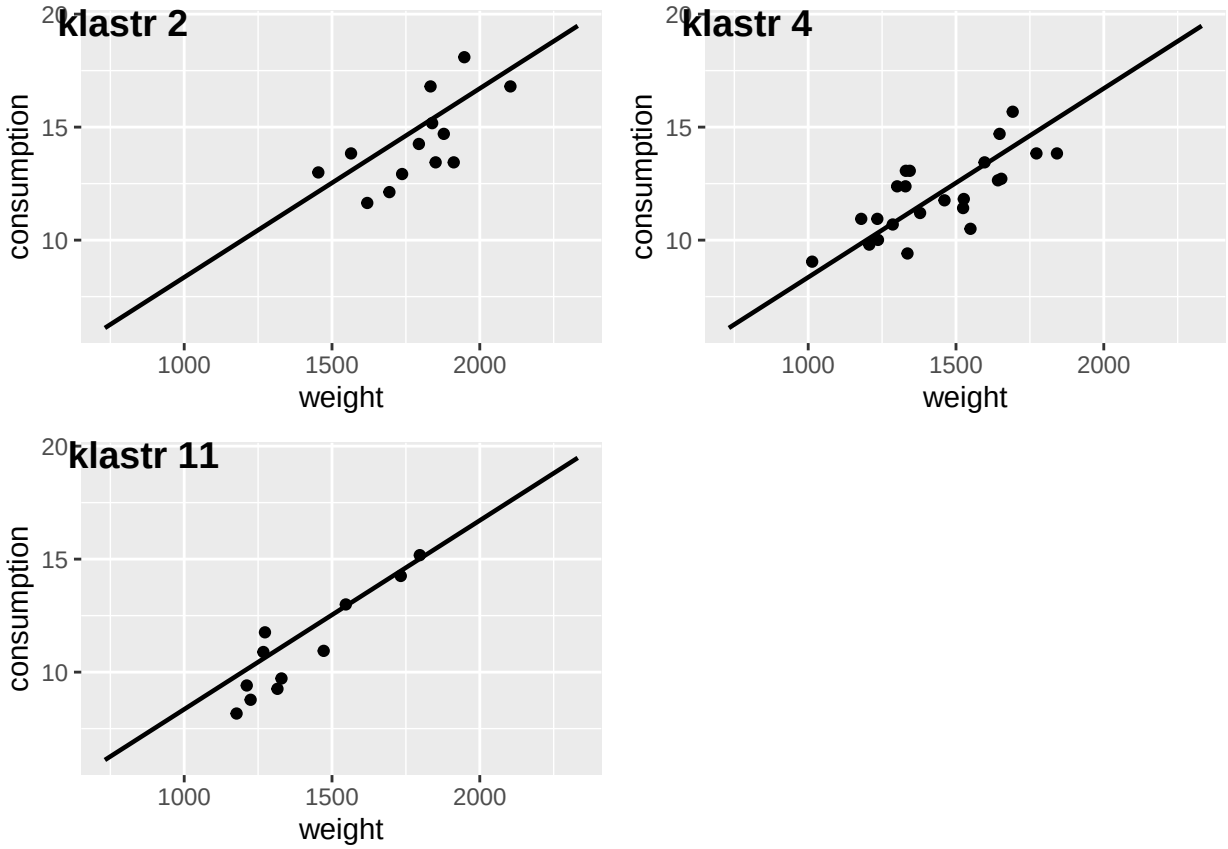
```
data_temp = data_mpghp[,c('cylinders','displacement','horsepower','acceleration')]
# naklastrujeme
set.seed(2)
clusters <- kmeans(data_temp, centers =  20, iter.max = 100)
data_mpghp$clusters = clusters$cluster
lm_classic <- lm(consumption ~ weight - 1, data_mpghp)
```

Jako první si vykreslíme regresní přímky a scatterploty všech 20 skupin, vizuálně jsem zúžil výběr na 3 adepty, které dobře opisují regresní přímku a ve kterých se vyskytují auta s dostatečným váhovým rozdílem (potřebujeme linearitu při naložení 800kg, takže pokud jsou auta ve skupině s přibližně stejnou hmotností, moc nám to nepomůže), grafy zde znázorním

adepti: cluster 2,4,11

```
p1 <- ggplot(data_mpghp[data_mpghp[,'clusters']==2,], aes(weight, consumption))+geom_point()+
  geom_line(data = data_mpghp, aes(x=weight,y=lm_classic$fitted.values), size = 0.8)

p2 <- ggplot(data_mpghp[data_mpghp[,'clusters']==4,], aes(weight, consumption))+geom_point()+
  geom_line(data = data_mpghp, aes(x=weight,y=lm_classic$fitted.values), size = 0.8)

p3 <- ggplot(data_mpghp[data_mpghp[,'clusters']==11,], aes(weight, consumption))+geom_point()+
  geom_line(data = data_mpghp, aes(x=weight,y=lm_classic$fitted.values), size = 0.8)
```

```
figure <- ggarrange(p1, p2, p3,
                    labels = c("klastr 2", "klastr 4", "klastr 11"),
                    ncol = 2, nrow = 2)
plot(figure)
```



Nyní spočítáme a porovnáme sumu normovaných reziduí^2 pro tyto skupiny.

```
res_2 = (lm_classic$residuals[data_mpghp[,'clusters']==2])^2
sum_res_2 = (1/length(res_2))*sum(res_2)
sum_res_2
```

```
## [1] 2.284628
```

```
res_4 = (lm_classic$residuals[data_mpghp[,'clusters']==4])^2
sum_res_4 = (1/length(res_4))*sum(res_4)
sum_res_4
```

```
## [1] 1.483797
```

```
res_11 = (lm_classic$residuals[data_mpghp[,'clusters']==11])^2
sum_res_11 = (1/length(res_11))*sum(res_11)
sum_res_11
```

```
## [1] 1.236121
```

Nejlépe vychází klastr číslo 11. Jako nejlepšího reprezentanta této skupiny vezmeme bod nejblíže středu klastru, tj. bod nejblíže k (vzhledem k euklidovské metrice)

```
center = clusters$centers[11,]
center
```

```
##    cylinders displacement   horsepower acceleration
##     5.818182     3.007771   117.545455    14.036364
```

Nejbližší bod z této skupiny je Toyota Cressida z roku 81 vážící přibližně 1315kg. Volil bych tedy toto auto. Jako alternativu by ještě mohl být druhý nejbližší bod, kterým je auto Oldsmobile Omega Brougham z roku 79, které váží o něco méně, a to přibližně 1225kg.

Vzhledem k předchozí úvaze by se u těchto aut s velmi podobnými technickými parametry měla spotřeba na zvyšující se (pouze) hmotnosti měnit přibližně lineárně.

Kamarád by tedy za převoz 800kg nákladu zaplatil (koeficient weight v lin. modelu)* (800kg) *(vzdálenost Brno hl.n. - Praha Trojanova 13 v jednotkách 100km)*30 =

```
0.0083554*800*2.05*30
```

```
## [1] 411.0857
```

Tedy přibližně 411 Kč.