Initial analysis

pablo paulsen

15/11/2021

```
data = read.csv('../Ftf Efficiency Dataset/ftf_ev_efficiency_distance_models_20211006_v1.0.csv')[-1]


attach(data)
month_length = c(31,28,31,30,31,30,31,31,30,31,30,31)
head(data)
```

```
##    vehicle                model        region season month year distance
## 1 10bc0a12 Nissan Leaf (30 kWh) Coastal Otago Winter     8 2021     2487
## 2 10bc0a12 Nissan Leaf (30 kWh) Coastal Otago Spring     9 2021     1688
## 3 060ffa8d   Hyundai Ioniq (EV)       Waikato Winter     8 2018      880
## 4 060ffa8d   Hyundai Ioniq (EV)       Waikato Spring     9 2018     1352
## 5 060ffa8d   Hyundai Ioniq (EV)       Waikato Spring    10 2018     1869
## 6 060ffa8d   Hyundai Ioniq (EV)       Waikato Spring    11 2018     1286
##   efficiency   kwh
## 1        6.0 414.5
## 2        5.8 291.0
## 3        7.9 111.4
## 4        8.0 169.0
## 5        7.7 242.7
## 6        7.9 162.8
```

```
str(data)
```

```
## 'data.frame':    23129 obs. of  9 variables:
##  $ vehicle   : chr  "10bc0a12" "10bc0a12" "060ffa8d" "060ffa8d" ...
##  $ model     : chr  "Nissan Leaf (30 kWh)" "Nissan Leaf (30 kWh)" "Hyundai Ioniq (EV)" "Hyundai Ioni
##  $ region    : chr  "Coastal Otago" "Coastal Otago" "Waikato" "Waikato" ...
##  $ season    : chr  "Winter" "Spring" "Winter" "Spring" ...
##  $ month     : int  8 9 8 9 10 11 12 1 2 3 ...
##  $ year      : int  2021 2021 2018 2018 2018 2018 2018 2019 2019 2019 ...
##  $ distance  : num  2487 1688 880 1352 1869 ...
##  $ efficiency: num  6 5.8 7.9 8 7.7 7.9 8 8.26 8.33 8.3 ...
##  $ kwh       : num  414 291 111 169 243 ...
```

```
summary(data)
```

```
##    vehicle              model              region             season
##  Length:23129       Length:23129       Length:23129       Length:23129
##  Class :character   Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##      month             year         distance      efficiency
##  Min.   : 1.000   Min.   :2014   Min.   :   0   Min.   : 2.000
##  1st Qu.: 4.000   1st Qu.:2018   1st Qu.: 564   1st Qu.: 6.300
##  Median : 7.000   Median :2019   Median : 919   Median : 6.900
##  Mean   : 6.497   Mean   :2019   Mean   :1077   Mean   : 6.893
##  3rd Qu.: 9.000   3rd Qu.:2020   3rd Qu.:1415   3rd Qu.: 7.500
##  Max.   :12.000   Max.   :2021   Max.   :8695   Max.   :11.100
##       kwh
##  Min.   :   0.0
##  1st Qu.:  80.5
##  Median : 132.1
##  Mean   : 159.9
##  3rd Qu.: 208.3
##  Max.   :3105.4
```

```
data %>%
  group_by(year) %>%
  summarise(count = n_distinct(vehicle))
```

```
## # A tibble: 8 x 2
##    year count
##   <int> <int>
## 1  2014     1
## 2  2015     5
## 3  2016    23
## 4  2017   287
## 5  2018   755
## 6  2019   850
## 7  2020   728
## 8  2021   637
```

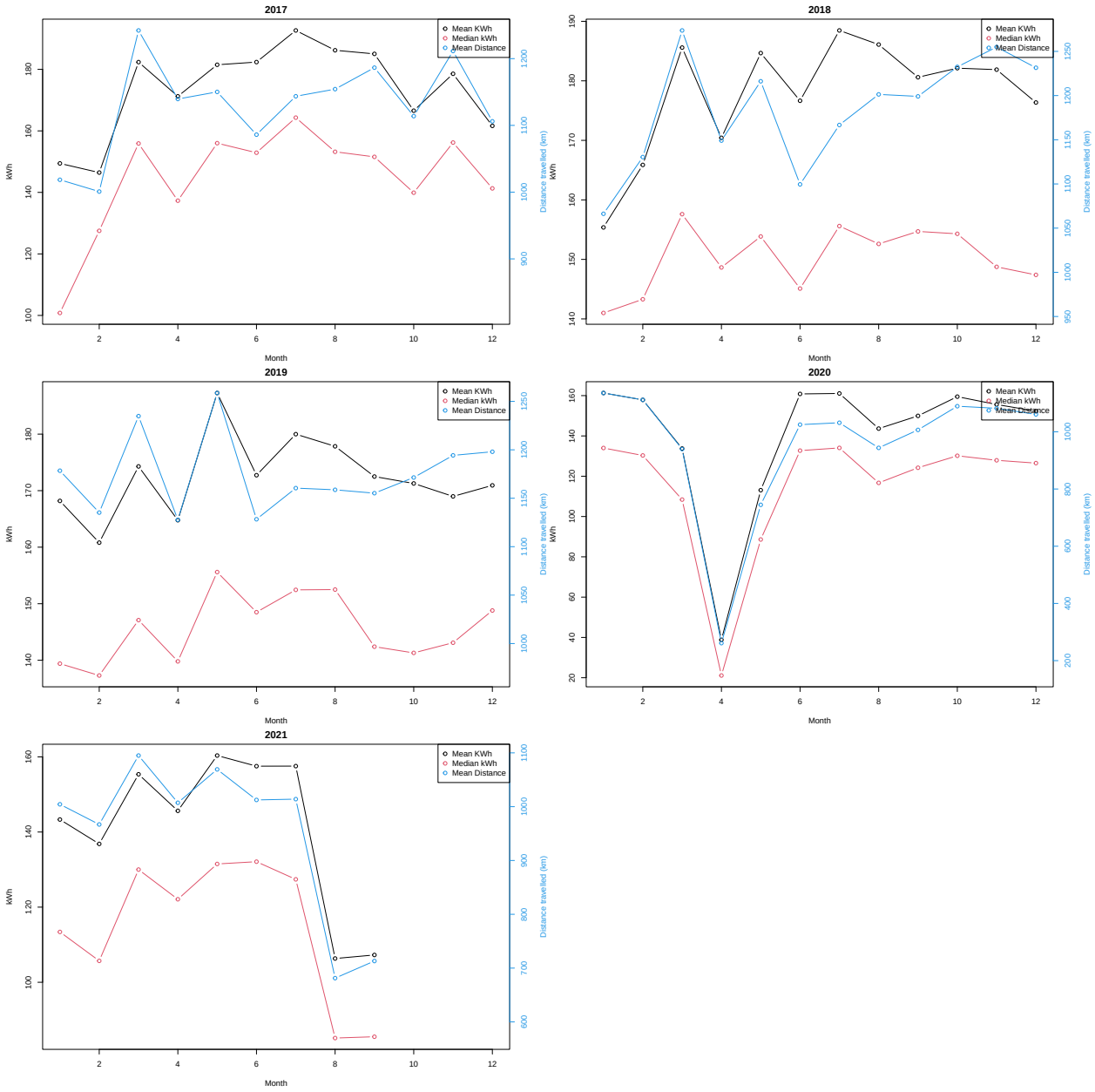
```
data %>%
  group_by(region) %>%
  summarise(count =  n_distinct(vehicle)) %>%
  arrange(-count)
```

```
## # A tibble: 25 x 2
##    region           count
##    <chr>            <int>
##  1 Auckland           334
##  2 Wellington         238
##  3 Christchurch       146
##  4 Coastal Otago      133
##  5 Waikato             65
##  6 Bay of Plenty       53
##  7 North Canterbury    35
##  8 Central Otago       31
##  9 Mid Canterbury      31
## 10 Nelson              31
## # ... with 15 more rows
```

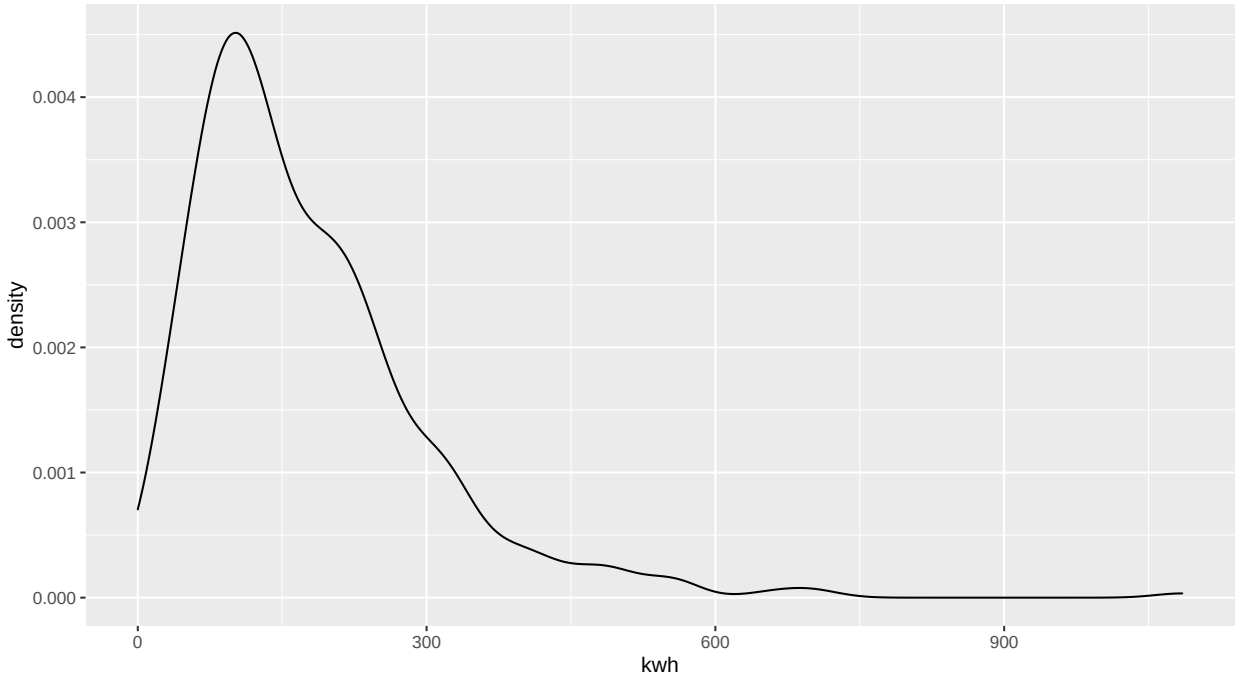
can't really use before 2017

```
mean_data = list()
for (i in 2017:2021) {
  mean_data[[i-2016]] = data[year == i,] %>%
    group_by(month) %>%
    summarise(mean_kwh = mean(kwh), mean_dist = mean(distance), median_kwh = median(kwh), median_dist =
}
```

```
par(mfrow=c(3,2))
for (i in 2017:2021) {
  par(mar = c(4, 4, 2, 4))
  cur_year = data.frame(mean_data[[i-2016]])
  plot(cur_year$month, cur_year$mean_kwh, type = 'b', main = i, xlab = "Month", ylab = "kWh", xlim = c(
       ylim = c(min(c(cur_year$mean_kwh,cur_year$median_kwh)),max(c(cur_year$mean_kwh,cur_year$median_ku
  points(cur_year$month, cur_year$median_kwh, type = 'b', col = 2)


  par(new=TRUE)
  plot(cur_year$month, cur_year$mean_dist, xlab="", ylab="", xlim = c(1,12),
      ylim = c(min(c(cur_year$mean_dist,cur_year$median_dist)),max(c(cur_year$mean_dist,cur_year$median_
      axes=FALSE, type="b", col=4)
  mtext("Distance travelled (km)",side=4,col=4,line=3, cex = 0.7)
  axis(4, ylim=c(min(c(cur_year$mean_dist,cur_year$median_dist)),max(c(cur_year$mean_dist,cur_year$media
      col=4,col.axis=4)
  legend("topright", legend = c("Mean KWh", "Median kWh", "Mean Distance"), col = c(1:2,4), pch = 1)

}
```

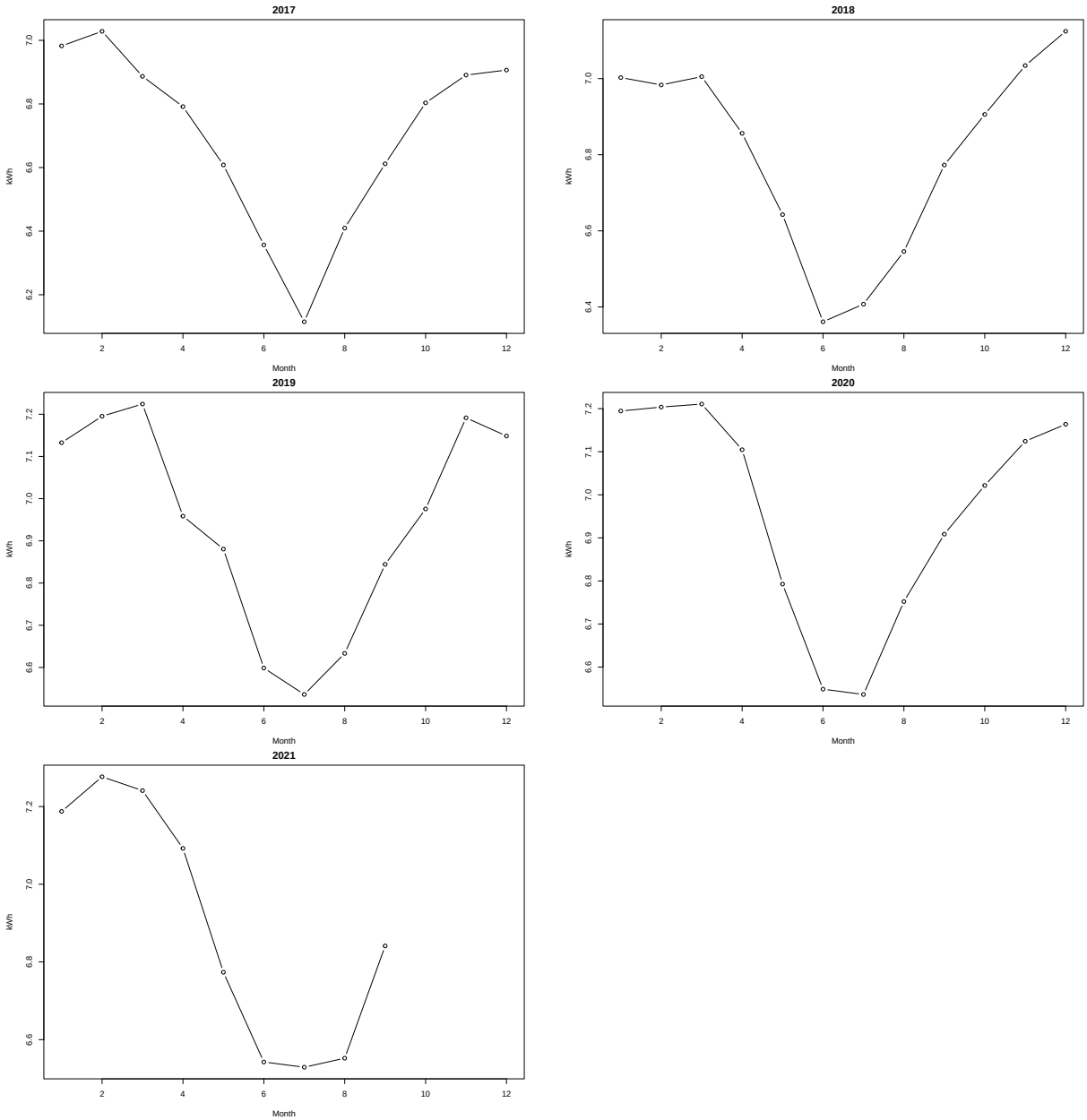



```
ggplot(data = data[year == 2018 & month == 6,], aes(x = kwh)) + geom_density(alpha = 0.5, adjust = 1)
```




```
#ggplot(data = patient, aes(x=age))+ geom_histogram(binwidth =10, color = "peachpuff3", fill = "hotpink
```

```
par(mfrow=c(3,2))
for (i in 2017:2021) {
  par(mar = c(4, 4, 2, 4))
  cur_year = data.frame(mean_data[[i-2016]])
  plot(cur_year$month, cur_year$mean_ef, type = 'b', main = i, xlab = "Month", ylab = "kWh", xlim = c(1
  points(cur_year$month, cur_year$median_kwh, type = 'b', col = 2)
}
```

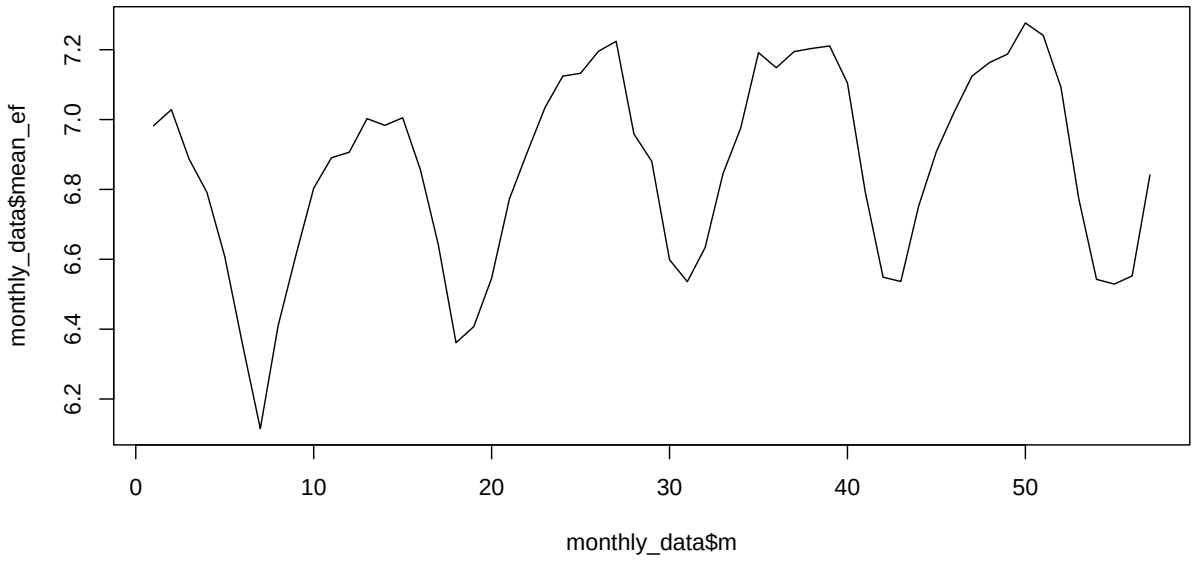



```
monthly_data = data[year >= 2017,] %>%
  group_by(year, month) %>%
  summarise(mean_kwh = mean(kwh), mean_dist = mean(distance), median_kwh = median(kwh), median_dist = me
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

```
monthly_data$m = 1:nrow(monthly_data)
```

```
plot(x = monthly_data$m, y = monthly_data$mean_ef, type = 'l')
```



try a couple of models

```
monthly_eff_lm = lm(mean_ef ~ m+factor(month),data = monthly_data)
summary(monthly_eff_lm)
```

```
##
## Call:
## lm(formula = mean_ef ~ m + factor(month), data = monthly_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.169609 -0.035832 -0.002043  0.041524  0.140991
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.9540070  0.0361299 192.472  < 2e-16 ***
## m                0.0058376  0.0005981   9.760 1.41e-12 ***
## factor(month)2   0.0316996  0.0465181   0.681 0.499161
## factor(month)3   0.0019524  0.0465296   0.042 0.966720
## factor(month)4  -0.1568045  0.0465488  -3.369 0.001579 **
## factor(month)5  -0.3837543  0.0465757  -8.239 1.83e-10 ***
## factor(month)6  -0.6475665  0.0466103 -13.893  < 2e-16 ***
## factor(month)7  -0.7102034  0.0466525 -15.223  < 2e-16 ***
## factor(month)8  -0.5621573  0.0467023 -12.037 1.63e-15 ***
## factor(month)9  -0.3508861  0.0467597  -7.504 2.10e-09 ***
## factor(month)10 -0.1907978  0.0493684  -3.865 0.000362 ***
## factor(month)11 -0.0631082  0.0493938  -1.278 0.208072
## factor(month)12 -0.0433899  0.0494264  -0.878 0.384783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.07355 on 44 degrees of freedom
## Multiple R-squared:  0.9451, Adjusted R-squared:  0.9301
## F-statistic: 63.08 on 12 and 44 DF,  p-value: < 2.2e-16
```

```
monthly_eff_qm = lm(mean_ef ~ m+I(m^2)+factor(month),data = monthly_data)
summary(monthly_eff_qm)
```

```
##
## Call:
## lm(formula = mean_ef ~ m + I(m^2) + factor(month), data = monthly_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.136916 -0.035349 -0.000536  0.029554  0.104565
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.863e+00  3.339e-02 205.498  < 2e-16 ***
## m                1.571e-02  1.936e-03   8.115 3.26e-10 ***
## I(m^2)          -1.703e-04  3.238e-05  -5.259 4.32e-06 ***
## factor(month)2   3.051e-02  3.671e-02   0.831   0.4105
## factor(month)3  -9.092e-05  3.672e-02  -0.002   0.9980
## factor(month)4  -1.594e-01  3.674e-02  -4.338 8.55e-05 ***
## factor(month)5  -3.865e-01  3.676e-02 -10.514 1.84e-13 ***
## factor(month)6  -6.501e-01  3.678e-02 -17.674  < 2e-16 ***
## factor(month)7  -7.122e-01  3.682e-02 -19.346  < 2e-16 ***
## factor(month)8  -5.633e-01  3.686e-02 -15.286  < 2e-16 ***
## factor(month)9  -3.509e-01  3.690e-02  -9.509 3.89e-12 ***
## factor(month)10 -2.117e-01  3.916e-02  -5.407 2.64e-06 ***
## factor(month)11 -8.422e-02  3.918e-02  -2.149   0.0373 *
## factor(month)12 -6.433e-02  3.921e-02  -1.641   0.1081
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05804 on 43 degrees of freedom
## Multiple R-squared:  0.9666, Adjusted R-squared:  0.9565
## F-statistic: 95.63 on 13 and 43 DF,  p-value: < 2.2e-16
```

```
monthly_eff_lm_log = lm(mean_ef ~ m+I(log(m))+factor(month),data = monthly_data)
summary(monthly_eff_lm_log)
```

```
##
## Call:
## lm(formula = mean_ef ~ m + I(log(m)) + factor(month), data = monthly_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.163993 -0.046093  0.001669  0.037024  0.127306
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.880296   0.051541 133.491  < 2e-16 ***
## m                0.003369   0.001392   2.420 0.019840 *
```

```
## I(log(m))        0.050961   0.026137   1.950 0.057746 .
## factor(month)2   0.025470   0.045217   0.563 0.576164
## factor(month)3  -0.007495   0.045375  -0.165 0.869569
## factor(month)4  -0.168201   0.045511  -3.696 0.000616 ***
## factor(month)5  -0.396378   0.045622  -8.688 5.14e-11 ***
## factor(month)6  -0.660944   0.045712 -14.459  < 2e-16 ***
## factor(month)7  -0.723996   0.045784 -15.813  < 2e-16 ***
## factor(month)8  -0.576107   0.045845 -12.567 5.46e-16 ***
## factor(month)9  -0.364788   0.045896  -7.948 5.61e-10 ***
## factor(month)10 -0.210394   0.048912  -4.302 9.59e-05 ***
## factor(month)11 -0.082660   0.048931  -1.689 0.098399 .
## factor(month)12 -0.062752   0.048942  -1.282 0.206658
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07131 on 43 degrees of freedom
## Multiple R-squared:  0.9495, Adjusted R-squared:  0.9343
## F-statistic: 62.23 on 13 and 43 DF,  p-value: < 2.2e-16
```

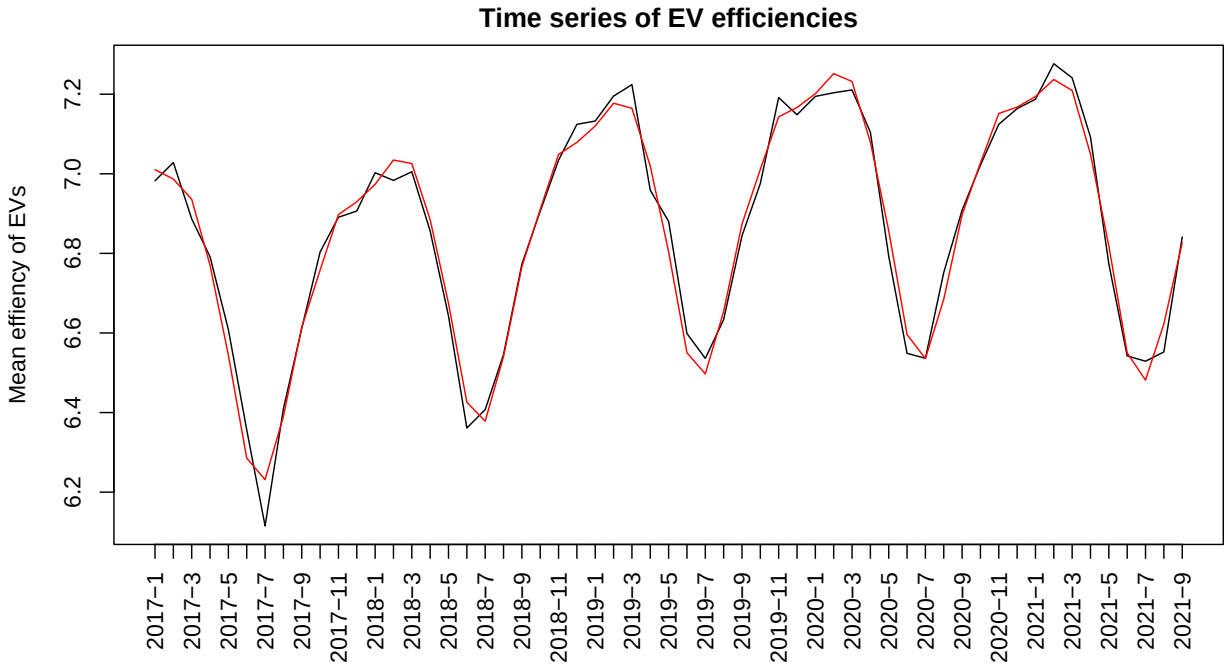
```
monthly_eff_qm_log = lm(mean_ef ~ m+I(log(m))+I(m^2)+factor(month),data = monthly_data)
summary(monthly_eff_qm_log)
```

```
##
## Call:
## lm(formula = mean_ef ~ m + I(log(m)) + I(m^2) + factor(month),
##     data = monthly_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.116381 -0.028026 -0.003808  0.031606  0.076696
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.977e+00  3.700e-02 188.537  < 2e-16 ***
## m                3.423e-02  4.321e-03   7.921 7.21e-10 ***
## I(log(m))       -1.506e-01  3.267e-02  -4.611 3.73e-05 ***
## I(m^2)          -3.637e-04  4.972e-05  -7.315 5.17e-09 ***
## factor(month)2   4.756e-02  3.049e-02   1.560 0.126276
## factor(month)3   2.551e-02  3.078e-02   0.829 0.411873
## factor(month)4  -1.286e-01  3.101e-02  -4.146 0.000161 ***
## factor(month)5  -3.523e-01  3.120e-02 -11.290 2.65e-14 ***
## factor(month)6  -6.135e-01  3.135e-02 -19.568  < 2e-16 ***
## factor(month)7  -6.738e-01  3.148e-02 -21.405  < 2e-16 ***
## factor(month)8  -5.235e-01  3.159e-02 -16.570  < 2e-16 ***
## factor(month)9  -3.098e-01  3.170e-02  -9.773 2.22e-12 ***
## factor(month)10 -1.776e-01  3.312e-02  -5.362 3.26e-06 ***
## factor(month)11 -5.042e-02  3.313e-02  -1.522 0.135513
## factor(month)12 -3.090e-02  3.313e-02  -0.933 0.356286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04785 on 42 degrees of freedom
## Multiple R-squared:  0.9778, Adjusted R-squared:  0.9704
## F-statistic: 132.2 on 14 and 42 DF,  p-value: < 2.2e-16
```

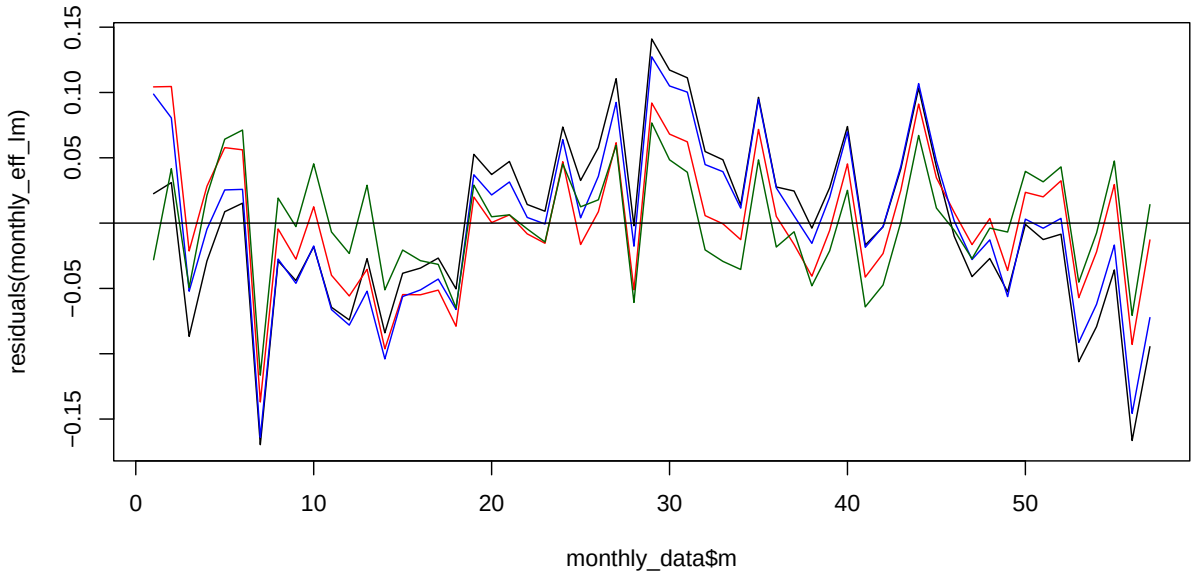
```r
anova(monthly_eff_lm, monthly_eff_lm_log, monthly_eff_qm, monthly_eff_qm_log)
```

```
## Analysis of Variance Table
##
## Model 1: mean_ef ~ m + factor(month)
## Model 2: mean_ef ~ m + I(log(m)) + factor(month)
## Model 3: mean_ef ~ m + I(m^2) + factor(month)
## Model 4: mean_ef ~ m + I(log(m)) + I(m^2) + factor(month)
##   Res.Df      RSS Df Sum of Sq       F    Pr(>F)
## 1     44 0.237993
## 2     43 0.218662  1  0.019331  8.4433  0.005824 **
## 3     43 0.144837  0  0.073824
## 4     42 0.096161  1  0.048676 21.2602 3.726e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
par(mar = c(5, 4, 2, 1))
plot(monthly_data$m, monthly_data$mean_ef, type = 'l', xaxt = "n", xlab = "", ylab = "Mean effiency of E
lines(monthly_data$m, predict(monthly_eff_qm_log), col = 'red')

axis(1,labels = paste(monthly_data$year, monthly_data$month, sep = "-"), at = monthly_data$m, las = 2,si
```




```r
plot(monthly_data$m, residuals(monthly_eff_lm), type = 'l')
points(monthly_data$m, residuals(monthly_eff_qm), type = 'l', col = "red")
points(monthly_data$m, residuals(monthly_eff_lm_log), type = 'l', col = "blue")
points(monthly_data$m, residuals(monthly_eff_qm_log), type = 'l', col = "darkgreen")



abline(h= 0)
```



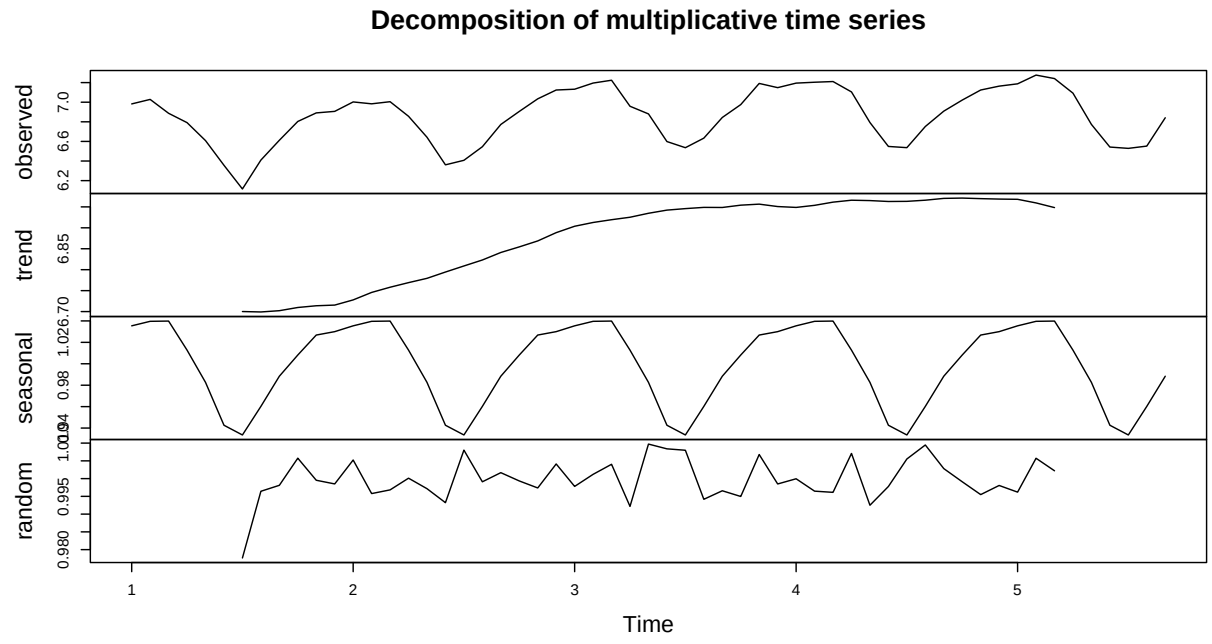

```
eff_series = ts(monthly_data$mean_ef, frequency = 12)
adf.test(eff_series, alternative = "stationary")
```

```
## Warning in adf.test(eff_series, alternative = "stationary"): p-value smaller
## than printed p-value
```

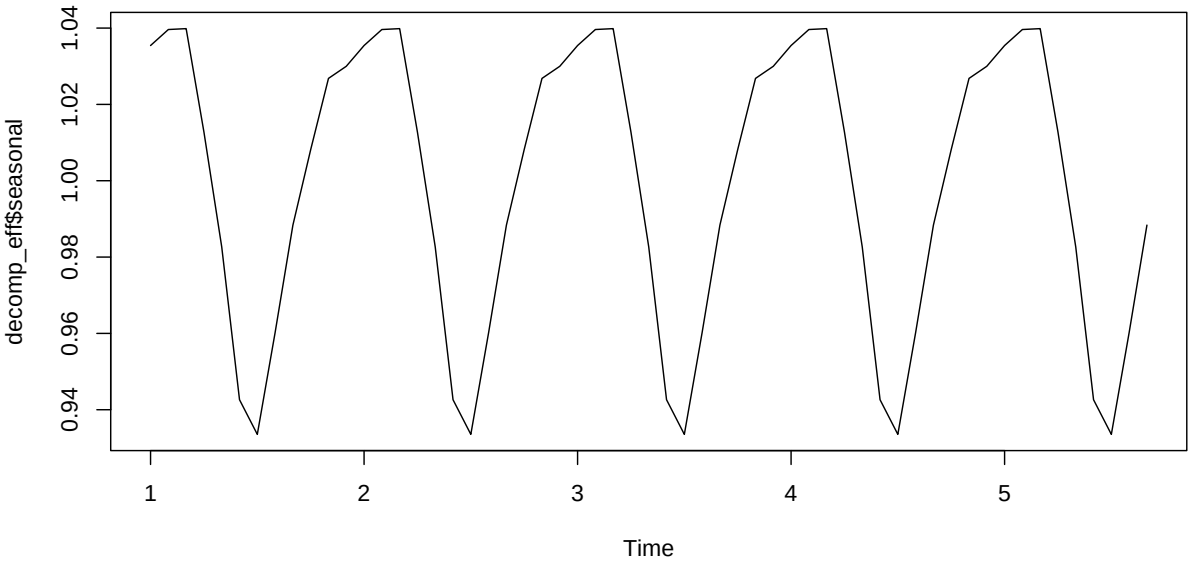
```
##
##  Augmented Dickey-Fuller Test
##
## data:  eff_series
## Dickey-Fuller = -4.7051, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

we can reject null hypothesis that data is not-stationary. this makes sense as average efficiency should not have significantly changed in a couple of years. use multiplicative instead of additive as preferable to know estimated extra power use? or should i know total extra power used in season?

```
#decomp_eff = stl(eff_series, s.window = "periodic")
decomp_eff = decompose(eff_series, "multiplicative")
plot(decomp_eff)
```




```
plot(decomp_eff$seasonal)
```

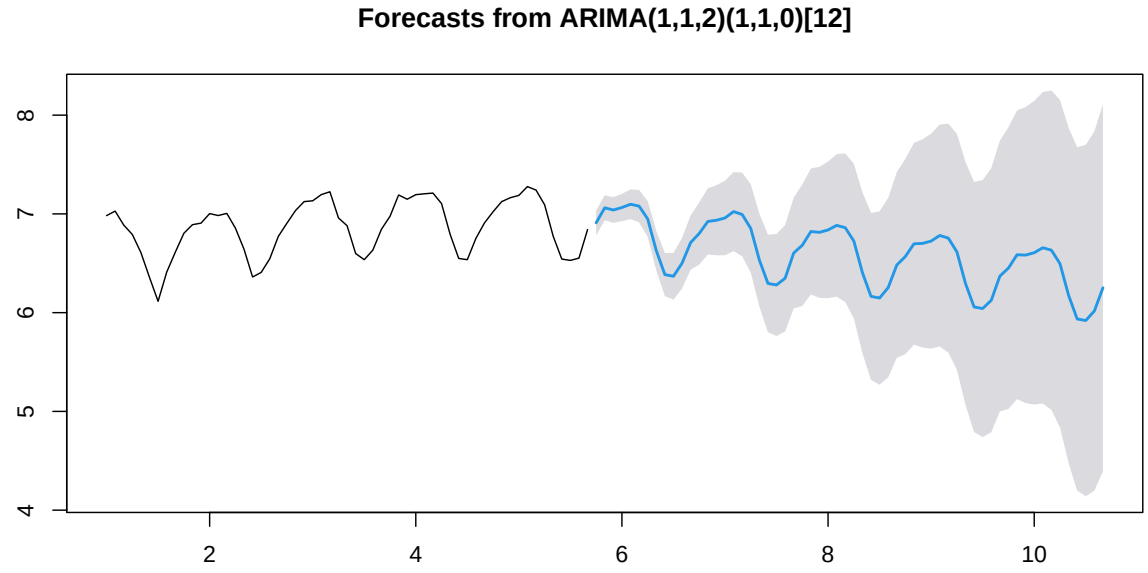



```
eff_model = auto.arima(eff_series) #uses AIC to get best model
summary(eff_model)
```

```
## Series: eff_series
## ARIMA(1,1,2)(1,1,0)[12]
```

```
##
## Coefficients:
##          ar1      ma1     ma2     sar1
##       0.8306  -1.7118  0.8679  -0.5824
## s.e.  0.1549   0.1391  0.1332   0.1348
##
## sigma^2 estimated as 0.004102:  log likelihood=56.94
## AIC=-103.88   AICc=-102.3   BIC=-94.96
##
## Training set error measures:
##                       ME       RMSE        MAE         MPE      MAPE      MASE
## Training set -0.004027656 0.05365432 0.03802298 -0.05547605 0.5542138 0.4152625
##                      ACF1
## Training set -0.02594167
```

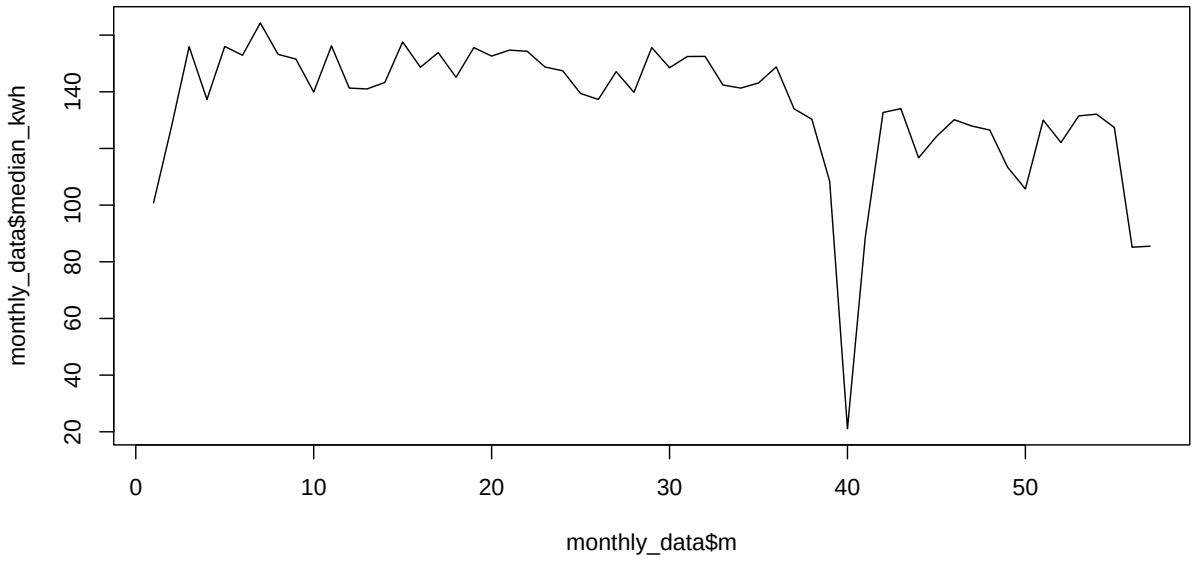
```
efficiency_forcast = forecast(eff_model, level = 95, h = 60)
plot(efficiency_forcast)
```




```
Box.test(efficiency_forcast$resid, lag=6, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  efficiency_forcast$resid
## X-squared = 2.9725, df = 6, p-value = 0.8123
```

```
plot(x = monthly_data$m, y = monthly_data$median_kwh, type = 'l')
```

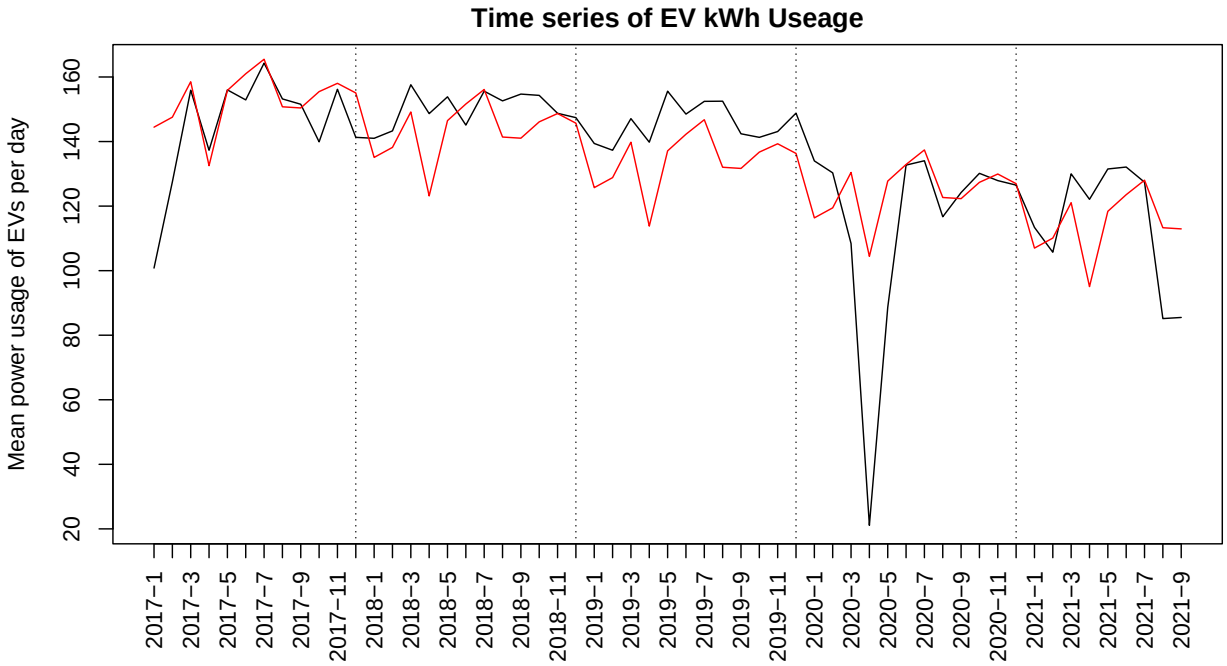



```
monthly_kwh_lm = lm(median_kwh ~ m+factor(month),data = monthly_data)
summary(monthly_kwh_lm)
```

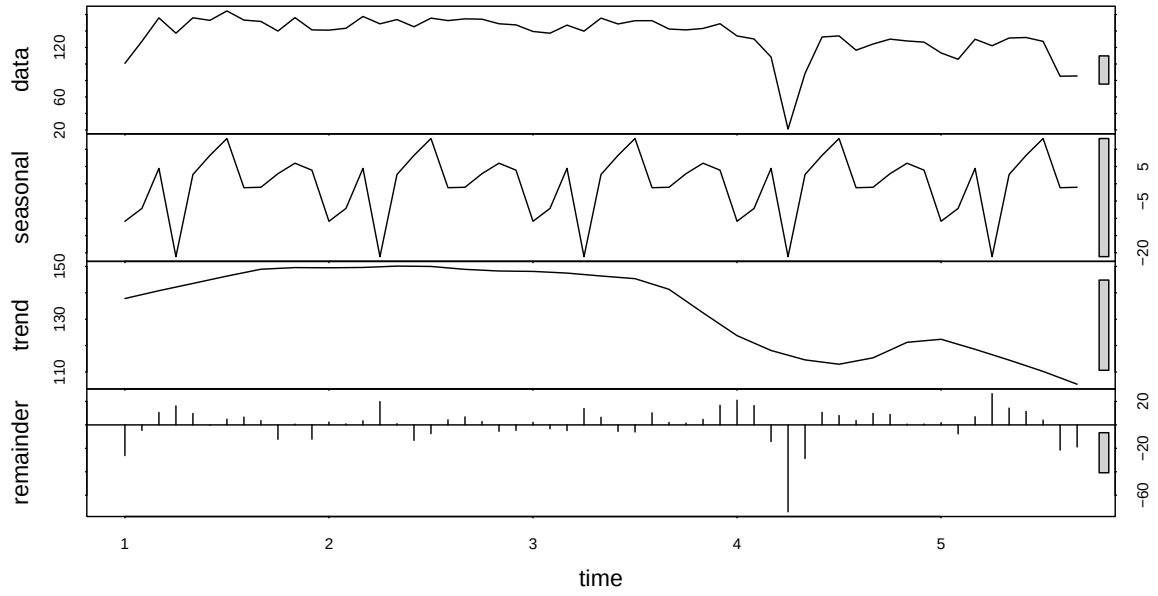
```
##
## Call:
## lm(formula = median_kwh ~ m + factor(month), data = monthly_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -83.324  -2.632   2.787   8.572  27.042
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     145.2329    10.0324  14.476  < 2e-16 ***
## m                -0.7805     0.1661  -4.700 2.58e-05 ***
## factor(month)2    3.8805    12.9169   0.300   0.7653
## factor(month)3   15.6410    12.9201   1.211   0.2325
## factor(month)4   -9.5885    12.9255  -0.742   0.4621
## factor(month)5   14.5121    12.9329   1.122   0.2679
## factor(month)6   20.4426    12.9425   1.579   0.1214
## factor(month)7   25.7231    12.9543   1.986   0.0533 .
## factor(month)8   11.7736    12.9681   0.908   0.3689
## factor(month)9   12.1941    12.9840   0.939   0.3528
## factor(month)10  18.0340    13.7084   1.316   0.1951
## factor(month)11  21.3896    13.7154   1.560   0.1260
## factor(month)12  19.1826    13.7245   1.398   0.1692
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.42 on 44 degrees of freedom
## Multiple R-squared:  0.4311, Adjusted R-squared:  0.276
```

```
## F-statistic: 2.779 on 12 and 44 DF,  p-value: 0.006676
```

```
par(mar = c(5, 4, 2, 1))
plot(monthly_data$m, monthly_data$median_kwh, type = 'l', xaxt = "n", xlab = "", ylab = "Mean power usag
lines(monthly_data$m, predict(monthly_kwh_lm), col = 'red')

axis(1,labels = paste(monthly_data$year, monthly_data$month, sep = "-"), at = monthly_data$m, las = 2)
abline(v = 12, lty = 3)
abline(v = 24, lty = 3)
abline(v = 36, lty = 3)
abline(v = 48, lty = 3)
abline(v = 60, lty = 3)
```




```
kwh_series = ts(monthly_data$median_kwh, frequency = 12)
decomp_kwh = stl(kwh_series, s.window = "periodic")
plot(decomp_kwh)
```




```
adf.test(kwh_series, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  kwh_series
## Dickey-Fuller = -3.5116, Lag order = 3, p-value = 0.04852
## alternative hypothesis: stationary
```

linear model time series but grouped by car model. turns out is basically useless and get more confident on the monthly effiencys by just using the average

```
#monthly_data_model = data[year >= 2017,] %>%
#  group_by(year, month, model) %>%
#  summarise(mean_kwh = mean(kwh), mean_dist = mean(distance), median_kwh = median(kwh), median_dist =
#monthly_data_model$m = as.numeric(factor(paste(monthly_data_model$year,monthly_data_model$month, sep =
#                                       levels = unique(paste(monthly_data_model$year,monthly_data_model
```

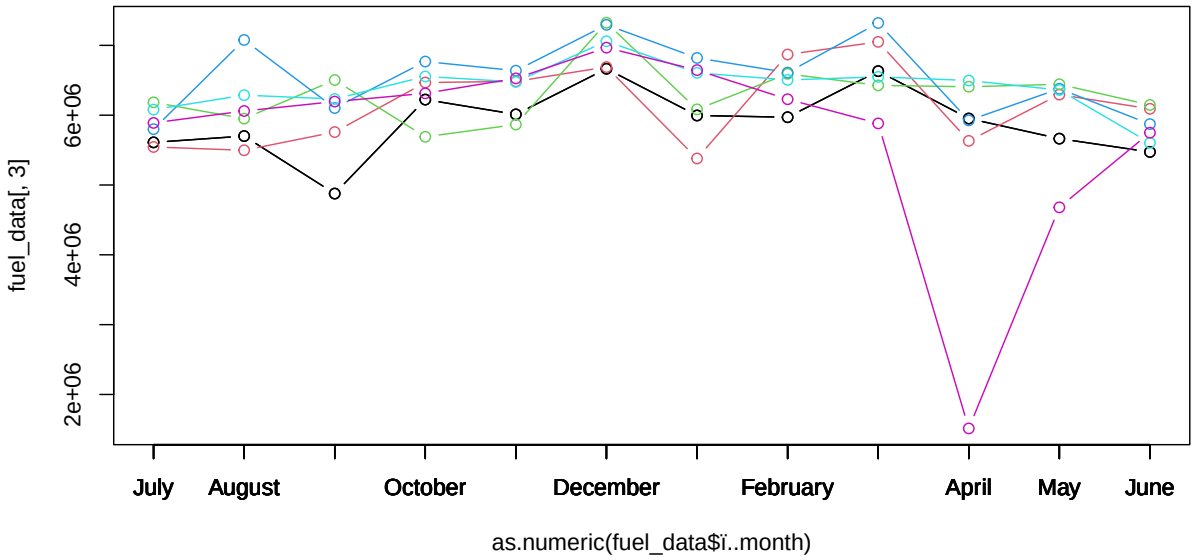
```
#monthly_eff_model_lm = lm(mean_ef ~ m+factor(month)+factor(model),data = monthly_data_model)
#summary(monthly_eff_model_lm)
```

```
fuel_data = read.csv('Dunedin Fuel research/Dunedin fuel usage.csv')
head(fuel_data)
```

```
##    ï..month month_no   X2015   X2016   X2017   X2018   X2019   X2020
## 1       July        7 5610642 5544497 6184078 5799777 6079678 5889484
## 2     August        8 5700740 5497344 5950339 7077168 6286820 6059895
## 3  September        9 4877210 5757994 6503403 6102254 6231719 6194753
## 4    October       10 6223226 6467628 5690018 6768411 6554627 6312884
## 5   November       11 6011595 6484301 5865678 6637329 6476736 6526996
## 6   December       12 6665164 6689692 7324506 7294138 7058879 6967167
```

```
fuel_data$ï..month = factor(fuel_data$ï..month, levels = fuel_data$ï..month)
```

```
plot(as.numeric(fuel_data$ï..month), fuel_data[,3], type = 'b', col = 1 ,xaxt = "n",
     ylim = c(min(fuel_data[,3:8]),max(fuel_data[,3:8])))
for (i in 3:8) {
  points(as.numeric(fuel_data$ï..month), fuel_data[,i], type = 'b', col = i-2 ,xaxt = "n")
  axis(1, labels = as.character(fuel_data$ï..month), at = as.numeric(fuel_data$ï..month))
}
```

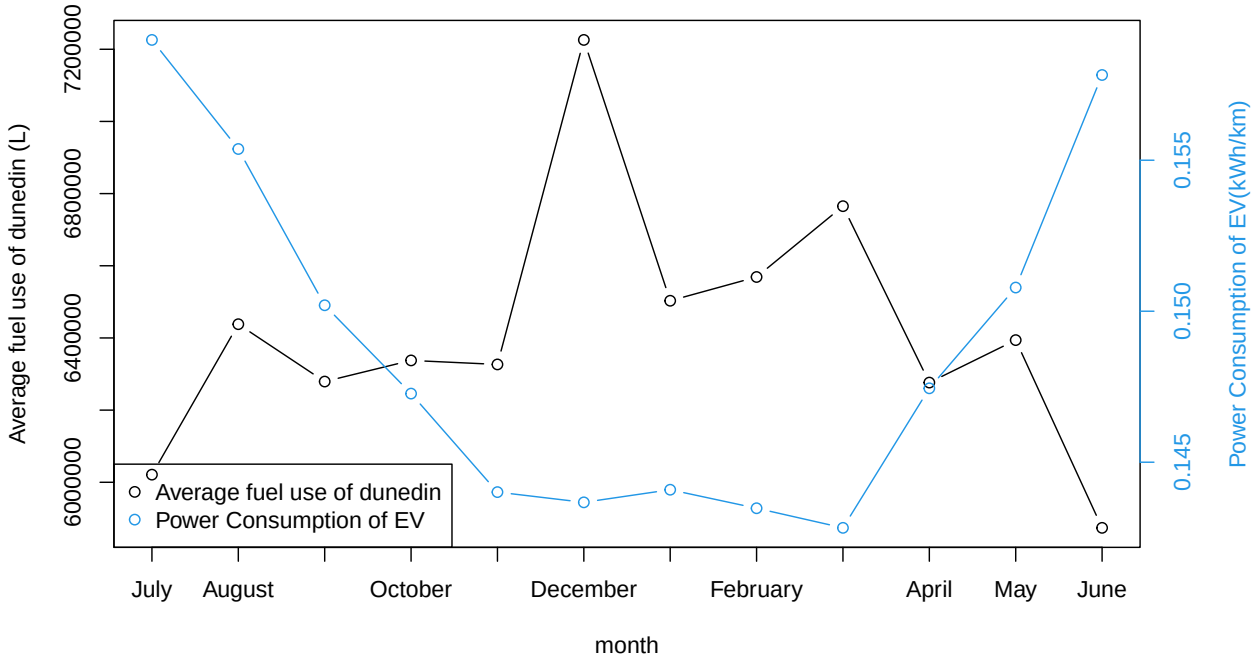



```
mean_fuel = fuel_data %>%
  gather(key = "year", value = "Fuel_used", c("X2017","X2018","X2019")) %>%
  group_by(ï..month) %>%
  summarise(average_fuel_use = mean(Fuel_used))


power = c()
for (i in 1:12) {
  #average power useage (kWh/km)
  power[i] = mean(1/data[month==i & year >=2017 & year <= 2019,]$efficiency, na.rm = T)
}
mean_fuel$power_usage = power[c(7:12,1:6)]
mean_fuel
```

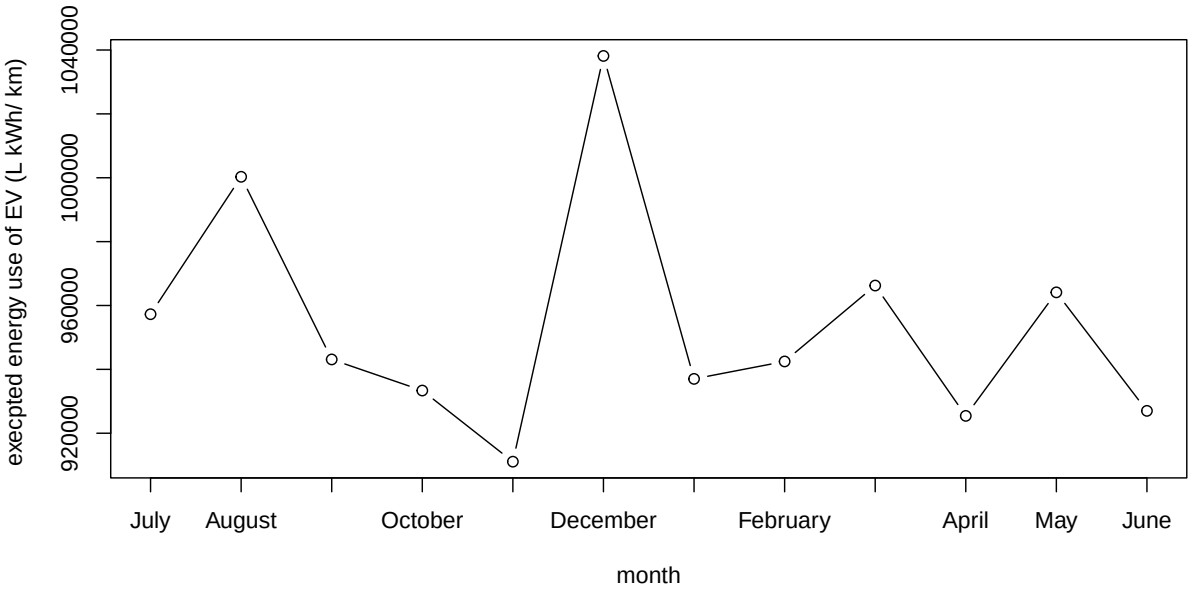
```
## # A tibble: 12 x 3
##    ï..month  average_fuel_use power_usage
##    <fct>                <dbl>       <dbl>
##  1 July              6021178.       0.159
##  2 August            6438109.       0.155
##  3 September         6279125.       0.150
##  4 October           6337685.       0.147
```

```
##  5 November          6326581.       0.144
##  6 December          7225841.       0.144
##  7 January           6503050.       0.144
##  8 February          6568882.       0.143
##  9 March             6765136.       0.143
## 10 April             6276348.       0.147
## 11 May               6394003.       0.151
## 12 June              5873758.       0.158
```

```
par(mar = c(4, 4, 2, 4))
plot(as.numeric(mean_fuel$ï..month), mean_fuel$average_fuel_use, type = 'b', xaxt = "n",
     ylab = "Average fuel use of dunedin (L)", xlab = "month")
axis(1, labels = as.character(fuel_data$ï..month), at = as.numeric(fuel_data$ï..month))

par(new=TRUE)
plot(as.numeric(mean_fuel$ï..month), mean_fuel$power_usage, xlab="", ylab="", xlim = c(1,12),
    ylim=c(min(mean_fuel$power_usage),max(mean_fuel$power_usage)),
    axes=FALSE, type="b", col=4)
mtext("Power Consumption of EV(kWh/km)",side=4,col=4,line=3, cex = 1)
axis(4, ylim=c(min(mean_fuel$power_usage),max(mean_fuel$power_usage)),
    col=4,col.axis=4)
legend("bottomleft", legend = c("Average fuel use of dunedin", "Power Consumption of EV"), col = c(1, 4
```




```
plot(as.numeric(mean_fuel$ï..month), mean_fuel$average_fuel_use * mean_fuel$power_usage, type = 'b', xa
     ylab = "execpted energy use of EV (L kWh/ km) ", xlab = "month")
axis(1, labels = as.character(fuel_data$ï..month), at = as.numeric(fuel_data$ï..month))
```



numbers could probably put to some meaningful scale if use some scaler (L/km) like fuel efficiency of gas car then can just show kwh usage

```
temp_data = read.csv('weather_data/auckland_motat_ews_data.csv',stringsAsFactors = T)
temp_data$Month = factor(temp_data$Month, levels = unique(temp_data$Month))
head(temp_data,10)
```

```
##        City    Station Year Month Day Hour Temp Twet.C. RH... Tdew.C.
## 1  Auckland  Motat Ews 2017   Jan   1    0 15.2    13.7  84.0    12.5
## 2  Auckland  Motat Ews 2017   Jan   1    1 15.1    13.8  86.0    12.8
## 3  Auckland  Motat Ews 2017   Jan   1    2 14.3    13.3  89.0    12.5
## 4  Auckland  Motat Ews 2017   Jan   1    3 14.2    13.4  91.0    12.8
## 5  Auckland  Motat Ews 2017   Jan   1    4 13.2    12.5  92.0    11.9
## 6  Auckland  Motat Ews 2017   Jan   1    5 12.3    11.8  94.0    11.4
## 7  Auckland  Motat Ews 2017   Jan   1    6 13.8    13.3  94.0    12.9
## 8  Auckland  Motat Ews 2017   Jan   1    7 15.5    14.9  93.0    14.4
## 9  Auckland  Motat Ews 2017   Jan   1    8 18.7    16.7  81.0    15.4
## 10 Auckland  Motat Ews 2017   Jan   1    9 21.0    17.1  66.0    14.4
```

```
HDD = function (data, temp = 18) {
  mon_sum = data.frame(Year = NA, Month = NA, HDD = NA)[-1,]
  for (y in min(data$Year):max(data$Year)){
    for (m in unique(data[data$Year == y,]$Month)) {
      #print(paste(y," ", m))
      adj_temps = -data[data$Year == y & data$Month == m,]$Temp+temp
      #print(sum(adj_temps*(adj_temps > 0))/24)
      mon_sum[nrow(mon_sum) + 1,] = c(y,m, sum(adj_temps*(adj_temps > 0))/24)
    }
  }
  mon_sum$Year = as.factor(mon_sum$Year)
  mon_sum$Month = factor(mon_sum$Month, levels = month.abb)
```

```
  mon_sum$HDD = as.numeric(mon_sum$HDD)
  return(mon_sum)
}

CDD = function (data, temp = 16) {
  mon_sum = data.frame(Year = NA, Month = NA, CDD = NA)[-1,]
  for (y in min(data$Year):max(data$Year)){
    for (m in levels(data$Month)) {
      #print(paste(y," ", m))
      adj_temps = data[data$Year == y & data$Month == m,]$Temp-temp
      #print(sum(adj_temps*(adj_temps > 0))/24)
      mon_sum[nrow(mon_sum) + 1,] = c(y,m, sum(adj_temps*(adj_temps > 0))/24)

    }
  }
  mon_sum$Year = as.factor(mon_sum$Year)
  mon_sum$Month = as.factor(mon_sum$Month)
  mon_sum$CDD = as.numeric(mon_sum$CDD)
  return(mon_sum)
}
```

```
HDD_auckland = HDD(temp_data)

HDD_auckland
```

```
##    Year Month        HDD
## 1  2017   Jan  33.479167
## 2  2017   Feb  18.050000
## 3  2017   Mar  23.720833
## 4  2017   Apr  59.337500
## 5  2017   May 151.545833
## 6  2017   Jun 186.275000
## 7  2017   Jul 231.037500
## 8  2017   Aug 177.429167
## 9  2017   Sep 149.337500
## 10 2017   Oct 101.175000
## 11 2017   Nov  62.670833
## 12 2017   Dec  15.516667
## 13 2018   Jan   2.141667
## 14 2018   Feb  13.783333
## 15 2018   Mar  25.395833
## 16 2018   Apr  61.908333
## 17 2018   May 131.304167
## 18 2018   Jun 203.704167
## 19 2018   Jul 226.125000
## 20 2018   Aug 201.520833
## 21 2018   Sep 157.737500
## 22 2018   Oct 133.100000
## 23 2018   Nov  76.700000
## 24 2018   Dec  21.508333
## 25 2019   Jan   6.995833
## 26 2019   Feb  15.508333
## 27 2019   Mar  24.925000
```

```
## 28 2019   Apr  97.870833
## 29 2019   May 113.637500
## 30 2019   Jun 236.512500
## 31 2019   Jul 235.900000
## 32 2019   Aug 231.629167
## 33 2019   Sep 167.445833
## 34 2019   Oct 119.800000
## 35 2019   Nov  60.891667
## 36 2019   Dec  30.129167
## 37 2020   Jan  26.433333
## 38 2020   Feb  15.587500
## 39 2020   Mar  48.300000
## 40 2020   Apr  83.641667
## 41 2020   May 136.325000
## 42 2020   Jun 155.308333
## 43 2020   Jul 204.429167
## 44 2020   Aug 188.341667
## 45 2020   Sep 157.812500
## 46 2020   Oct  99.979167
## 47 2020   Nov  58.900000
## 48 2020   Dec  35.283333
## 49 2021   Jan  20.745833
## 50 2021   Feb  18.687500
## 51 2021   Mar  29.212500
## 52 2021   Apr  70.658333
## 53 2021   May 122.470833
## 54 2021   Jun 141.325000
## 55 2021   Jul 215.991667
## 56 2021   Aug 186.450000
## 57 2021   Sep 155.995833
## 58 2021   Oct  91.454167
## 59 2021   Nov  26.754167
```

```
CDD(temp_data, 20)
```

```
##    Year Month         CDD
## 1  2017   Jan 16.85000000
## 2  2017   Feb 33.18750000
## 3  2017   Mar 19.15833333
## 4  2017   Apr  8.70000000
## 5  2017   May  0.11666667
## 6  2017   Jun  0.00000000
## 7  2017   Jul  0.00000000
## 8  2017   Aug  0.00000000
## 9  2017   Sep  0.01666667
## 10 2017   Oct  0.20000000
## 11 2017   Nov  6.89583333
## 12 2017   Dec 36.88750000
## 13 2018   Jan 65.74166667
## 14 2018   Feb 44.68333333
## 15 2018   Mar 33.15833333
## 16 2018   Apr  7.00000000
## 17 2018   May  1.36666667
## 18 2018   Jun  0.00000000
```

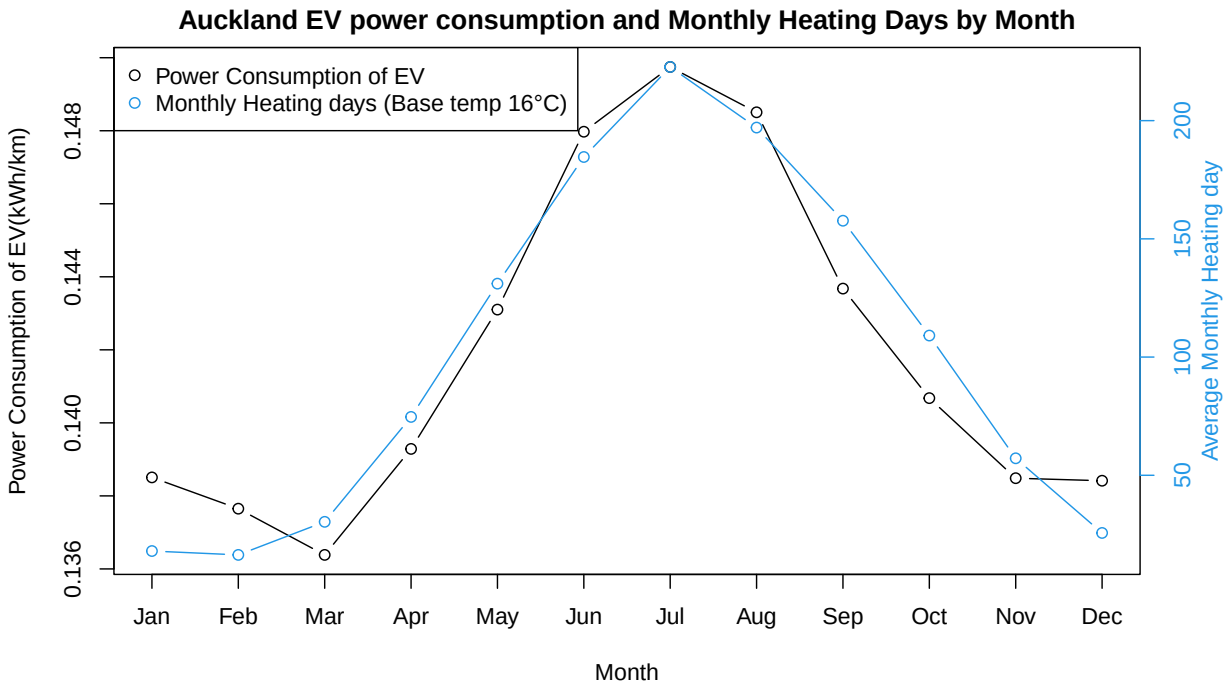
```
## 19 2018   Jul  0.00000000
## 20 2018   Aug  0.00000000
## 21 2018   Sep  0.02500000
## 22 2018   Oct  0.37083333
## 23 2018   Nov  3.75833333
## 24 2018   Dec 14.85416667
## 25 2019   Jan 49.92916667
## 26 2019   Feb 46.15833333
## 27 2019   Mar 36.49583333
## 28 2019   Apr  3.09583333
## 29 2019   May  0.58333333
## 30 2019   Jun  0.00000000
## 31 2019   Jul  0.00000000
## 32 2019   Aug  0.00000000
## 33 2019   Sep  0.00000000
## 34 2019   Oct  0.29166667
## 35 2019   Nov 10.22083333
## 36 2019   Dec 20.06250000
## 37 2020   Jan 33.60833333
## 38 2020   Feb 54.75416667
## 39 2020   Mar 21.21250000
## 40 2020   Apr  5.32083333
## 41 2020   May  0.14166667
## 42 2020   Jun  0.00000000
## 43 2020   Jul  0.00000000
## 44 2020   Aug  0.00000000
## 45 2020   Sep  0.00000000
## 46 2020   Oct  2.52500000
## 47 2020   Nov  4.62500000
## 48 2020   Dec 15.27500000
## 49 2021   Jan 33.09166667
## 50 2021   Feb 33.62083333
## 51 2021   Mar 18.38750000
## 52 2021   Apr  5.70416667
## 53 2021   May  0.15416667
## 54 2021   Jun  0.00000000
## 55 2021   Jul  0.00000000
## 56 2021   Aug  0.00000000
## 57 2021   Sep  0.06666667
## 58 2021   Oct  0.26666667
## 59 2021   Nov  6.37083333
## 60 2021   Dec  0.00000000
```

```r
mean_HDD_auckland = HDD_auckland %>%
  group_by(Month) %>%
  summarise(average_HDD = mean(HDD))

power = c()
eff = c()
kwh = c()
km = c()
for (i in 1:12) {
  #average power useage (kWh/km)
  power[i] = mean(1/data[month==i & region == "Auckland" & year >= 2017,]$efficiency, na.rm = T)
```

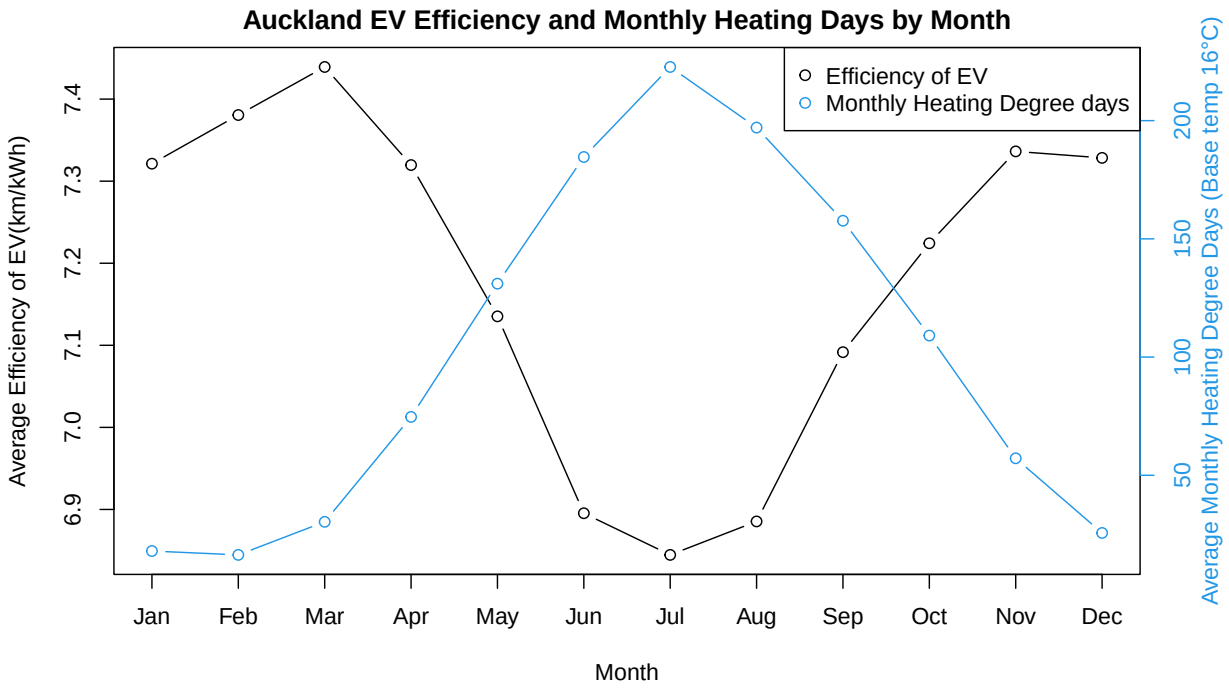
```
  eff[i] = mean(data[month==i & region == "Auckland" & year >= 2017,]$efficiency, na.rm = T)
  kwh[i] = mean(data[month==i & region == "Auckland" & year >= 2017,]$kwh, na.rm = T)
  km[i] = mean(data[month==i & region == "Auckland" & year >= 2017,]$distance, na.rm = T)

}

mean_HDD_auckland$power_usage = power
mean_HDD_auckland$efficiency = eff
mean_HDD_auckland$kwh = kwh
mean_HDD_auckland$km = km
mean_HDD_auckland
```

```
## # A tibble: 12 x 6
##    Month average_HDD power_usage efficiency   kwh    km
##    <fct>       <dbl>       <dbl>      <dbl> <dbl> <dbl>
##  1 Jan          18.0       0.139       7.32  156. 1126.
##  2 Feb          16.3       0.138       7.38  149. 1074.
##  3 Mar          30.3       0.136       7.44  155. 1134.
##  4 Apr          74.7       0.139       7.32  123.  883.
##  5 May         131.        0.143       7.14  154. 1078.
##  6 Jun         185.        0.148       6.90  162. 1089.
##  7 Jul         223.        0.150       6.84  166. 1124.
##  8 Aug         197.        0.149       6.89  146.  983.
##  9 Sep         158.        0.144       7.09  138.  960.
## 10 Oct         109.        0.141       7.22  160. 1137.
## 11 Nov          57.2       0.138       7.34  167. 1202.
## 12 Dec          25.6       0.138       7.33  166. 1193.
```

```
par(mar = c(4, 4, 2, 4))
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$power_usage, type = 'b', xaxt = "n",
     ylab = "Power Consumption of EV(kWh/km)", xlab = "Month", main = "Auckland EV power consumption an
axis(1, labels = as.character(mean_HDD_auckland$Month), at = as.numeric(mean_HDD_auckland$Month))

par(new=TRUE)
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$average_HDD, xlab="", ylab="", xlim = c(1,1
    ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    axes=FALSE, type="b", col=4)
mtext("Average Monthly Heating day",side=4,col=4,line=2, cex = 1)
axis(4, ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    col=4,col.axis=4)
legend("topleft", legend = c( "Power Consumption of EV", "Monthly Heating days (Base temp 16°C)"), col =
```




```
par(mar = c(4, 4, 2, 4))
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$efficiency, type = 'b', xaxt = "n",
     ylab = "Average Efficiency of EV(km/kWh)", xlab = "Month", main = "Auckland EV Efficiency and Montl
axis(1, labels = as.character(mean_HDD_auckland$Month), at = as.numeric(mean_HDD_auckland$Month))

par(new=TRUE)
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$average_HDD, xlab="", ylab="", xlim = c(1,1
    ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    axes=FALSE, type="b", col=4)
mtext("Average Monthly Heating Degree Days (Base temp 16°C)",side=4,col=4,line=2, cex = 1)
axis(4, ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    col=4,col.axis=4)
legend("topright", legend = c( "Efficiency of EV", "Monthly Heating Degree days"), col = c(1, 4), pch =
```



linear model of efficiency and HDD with averages from 2017 to 2021 in Auckland

```
avg_eff_HDD_auckland = data[data$year >=2017 & region == "Auckland",] %>%
  group_by(year, month) %>%
  summarise(efficiency = mean(efficiency), avg_kwh = mean(kwh))
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

```
#avg_eff_HDD_auckland$HDD = HDD_auckland$HDD
HDD_auckland$month = as.numeric(HDD_auckland$Month)
avg_eff_HDD_auckland = merge(avg_eff_HDD_auckland, HDD_auckland, by.x = c("year", "month"), by.y = c("Y
head(avg_eff_HDD_auckland,10)
```

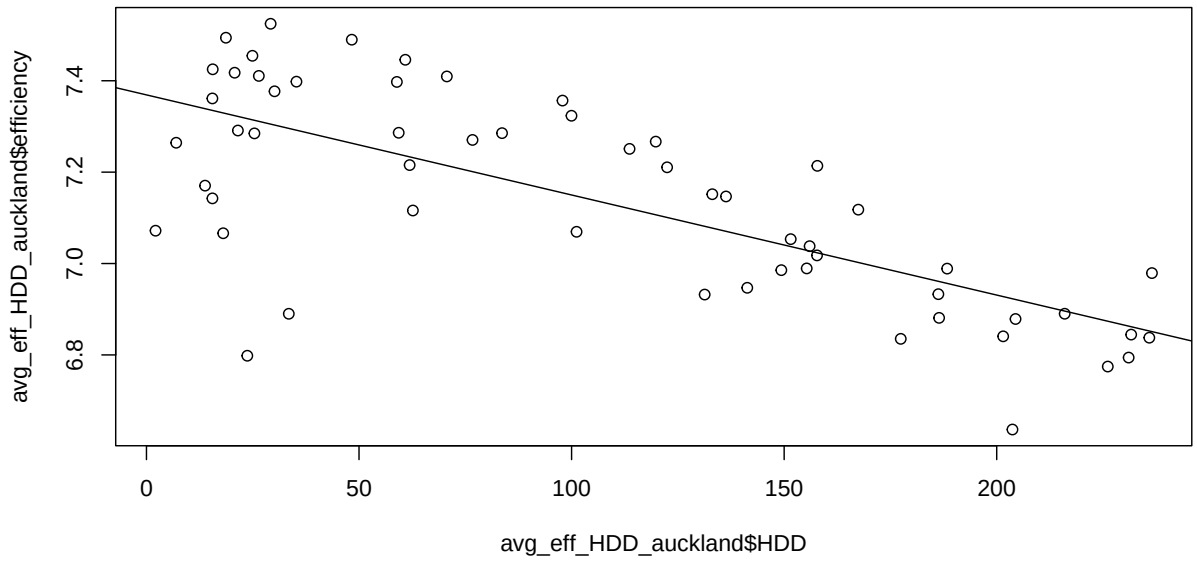
```
##    year month efficiency  avg_kwh Month       HDD
## 1  2017     1   6.890000 127.7750   Jan  33.47917
## 2  2017     2   7.066250 124.9750   Feb  18.05000
## 3  2017     3   6.798182 181.1636   Mar  23.72083
## 4  2017     4   7.286154 193.3000   Apr  59.33750
## 5  2017     5   7.053333 189.0933   May 151.54583
## 6  2017     6   6.932941 175.2000   Jun 186.27500
## 7  2017     7   6.794167 185.2792   Jul 231.03750
## 8  2017     8   6.835128 161.5923   Aug 177.42917
## 9  2017     9   6.985500 159.9525   Sep 149.33750
## 10 2017    10   7.069434 163.4340   Oct 101.17500
```

```
all_lm = lm(data = avg_eff_HDD_auckland,efficiency~HDD)
summary(all_lm)
```

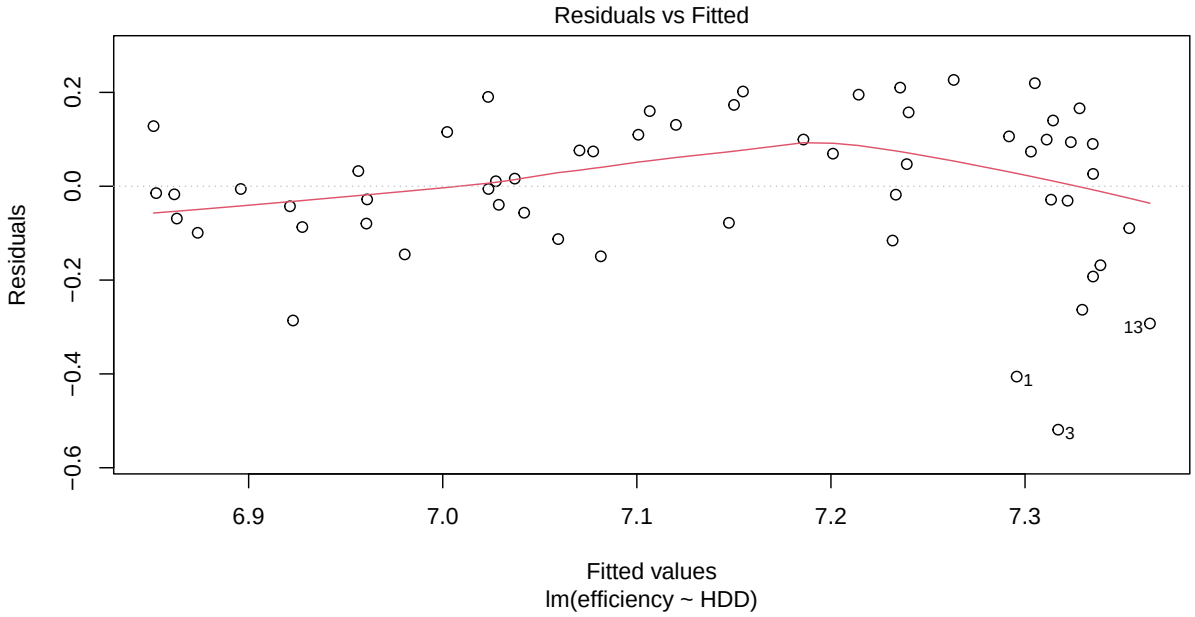
```
##
## Call:
```

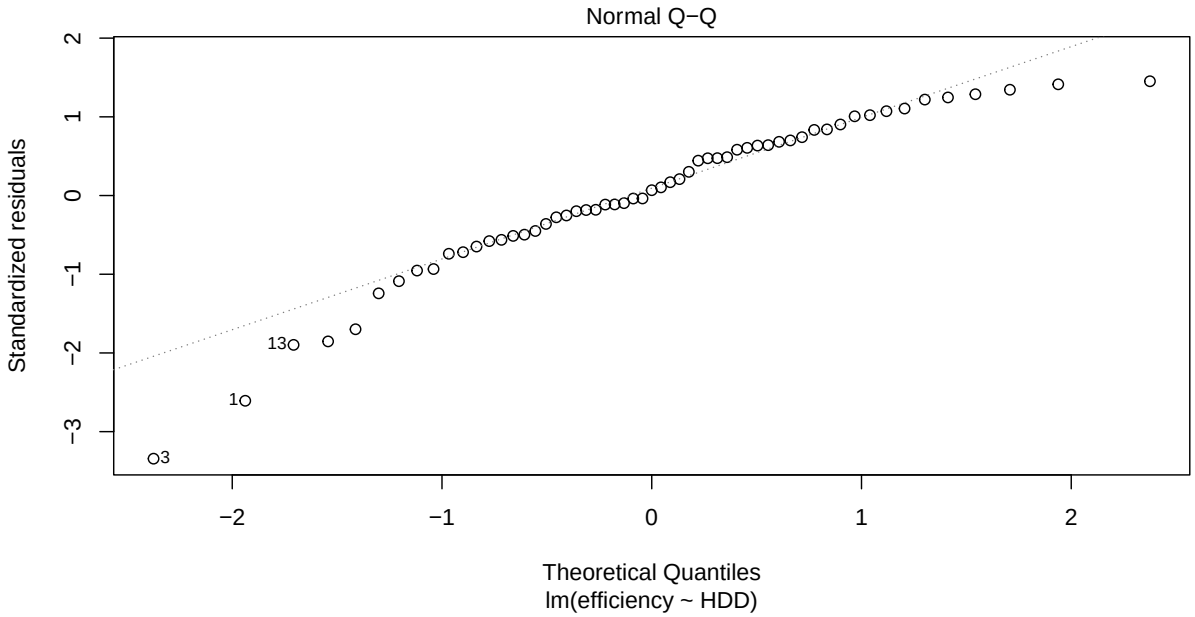
```
## lm(formula = efficiency ~ HDD, data = avg_eff_HDD_auckland)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.51890 -0.07959  0.01060  0.10976  0.22660
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.3690366  0.0362540 203.262  < 2e-16 ***
## HDD         -0.0021903  0.0002821  -7.765 2.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1583 on 55 degrees of freedom
## Multiple R-squared:  0.523,  Adjusted R-squared:  0.5143
## F-statistic:  60.3 on 1 and 55 DF,  p-value: 2.106e-10
```

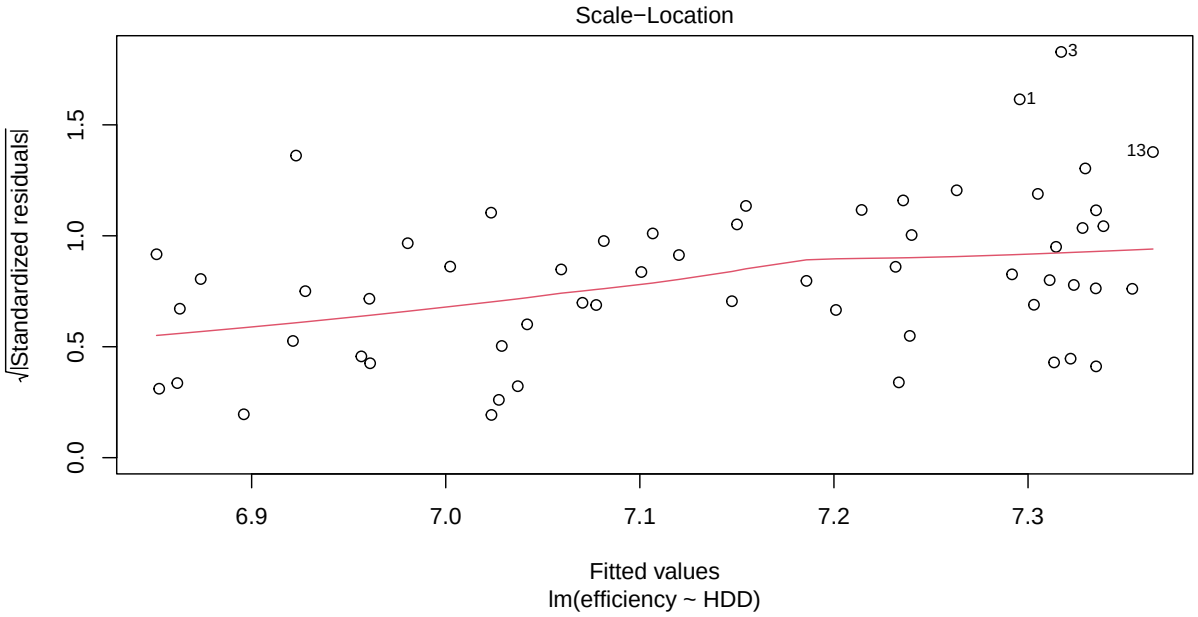
```
plot(avg_eff_HDD_auckland$HDD,avg_eff_HDD_auckland$efficiency)
abline(all_lm)
```

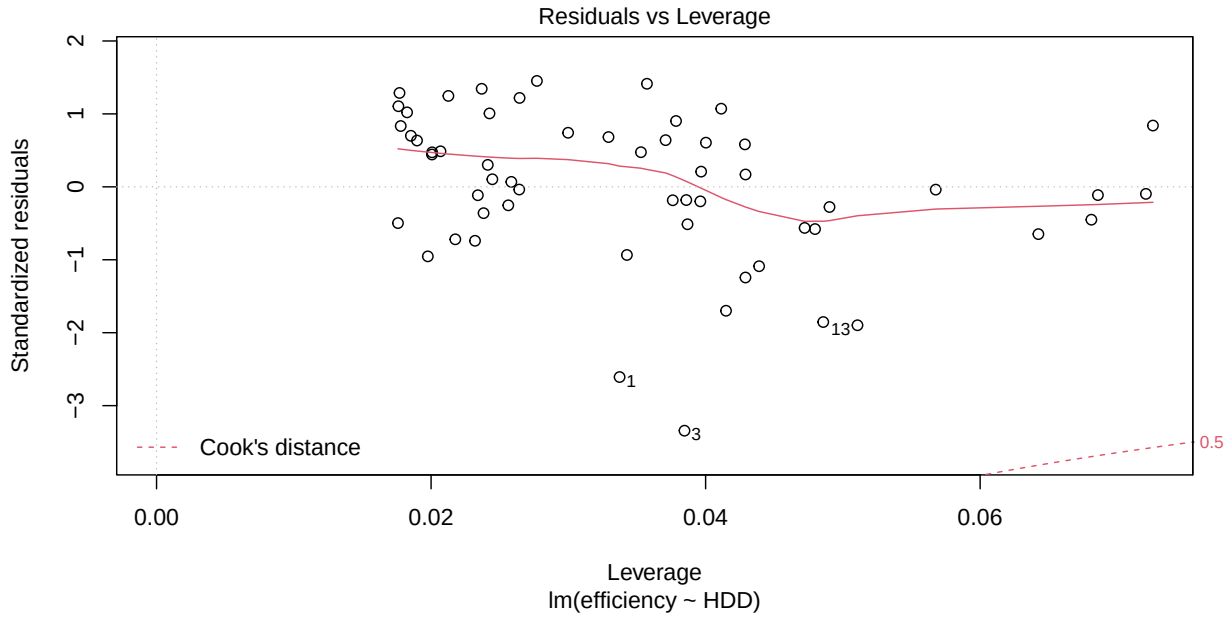



```
plot(all_lm)
```


Residuals vs Fitted
Residuals
0.2
0.0
−0.2
−0.4
−0.6
6.9
7.0
7.1
7.2
7.3
13
1
3
Fitted values
lm(efficiency ~ HDD)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−3
13
1
3
−2
−1
0
1
2
Theoretical Quantiles
lm(efficiency ~ HDD)





linear model with efficiency and HDD of individual cars from 2017 to 2021 in Auckland, categorized by type of car

```
auckland_model_count = data %>%
  group_by(model) %>%
  summarise(count =  n_distinct(vehicle)) %>%
  arrange(-count)
auckland_model_count
```

```
## # A tibble: 26 x 2
```

```
##    model                          count
##    <chr>                          <int>
##  1 Nissan Leaf (24 kWh) 2013-2016   527
##  2 Nissan Leaf (30 kWh)             275
##  3 Nissan Leaf (24 kWh) 2011-2012   204
##  4 Nissan Leaf (40 kWh)              68
##  5 Nissan e-NV200 (24 kWh)           62
##  6 Hyundai Ioniq (EV)                26
##  7 BMW i3                            23
##  8 Hyundai Kona (EV)                 14
##  9 Renault Zoe                       14
## 10 Tesla Model 3                     14
## # ... with 16 more rows
```

```
eff_HDD_auckland = data[data$year >=2017 & region == "Auckland",]
#-2 just to check that values have in inputted correctly
eff_HDD_auckland$HDD = rep(-2, nrow(eff_HDD_auckland))

for (y in min(eff_HDD_auckland$year):max(eff_HDD_auckland$year)){
  for (m in unique(eff_HDD_auckland[eff_HDD_auckland$year == y,]$month)) {
    eff_HDD_auckland[eff_HDD_auckland$year == y & eff_HDD_auckland$month == m,]$HDD = HDD_auckland$HDD[
  }
}
eff_HDD_auckland$model = factor(eff_HDD_auckland$model, levels = auckland_model_count$model)
head(eff_HDD_auckland,10)
```

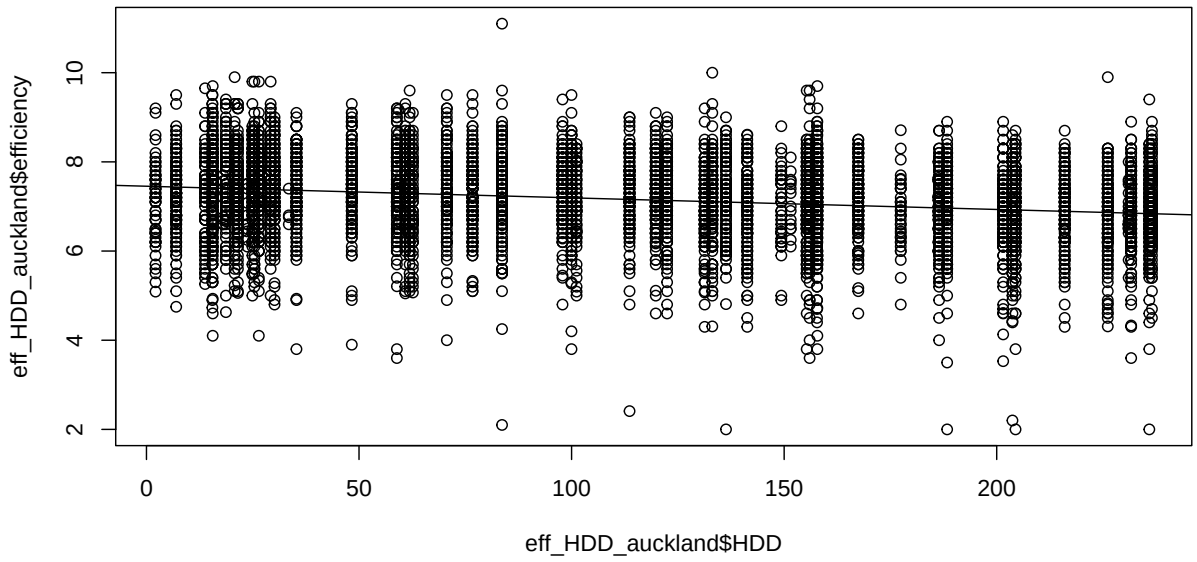
```
##      vehicle              model   region season month year distance efficiency
## 216 14840f85 Hyundai Ioniq (EV) Auckland Summer     2 2020     1371        8.5
## 217 14840f85 Hyundai Ioniq (EV) Auckland Autumn     3 2020      946        8.5
## 218 14840f85 Hyundai Ioniq (EV) Auckland Autumn     4 2020       21        8.5
## 219 14840f85 Hyundai Ioniq (EV) Auckland Autumn     5 2020      368        8.5
## 220 14840f85 Hyundai Ioniq (EV) Auckland Winter     6 2020     1109        8.3
## 221 14840f85 Hyundai Ioniq (EV) Auckland Winter     7 2020     1867        7.8
## 222 14840f85 Hyundai Ioniq (EV) Auckland Winter     8 2020      497        7.8
## 223 14840f85 Hyundai Ioniq (EV) Auckland Spring     9 2020      599        9.7
## 224 14840f85 Hyundai Ioniq (EV) Auckland Spring    10 2020     1201        9.5
## 225 14840f85 Hyundai Ioniq (EV) Auckland Spring    11 2020     1934        9.1
##       kwh       HDD
## 216 161.3  15.58750
## 217 111.3  48.30000
## 218   2.5  83.64167
## 219  43.3 136.32500
## 220 133.6 155.30833
## 221 239.4 204.42917
## 222  63.7 188.34167
## 223  61.8 157.81250
## 224 126.4  99.97917
## 225 212.5  58.90000
```

```
all_lm = lm(data = eff_HDD_auckland,efficiency~HDD)
summary(all_lm)
```

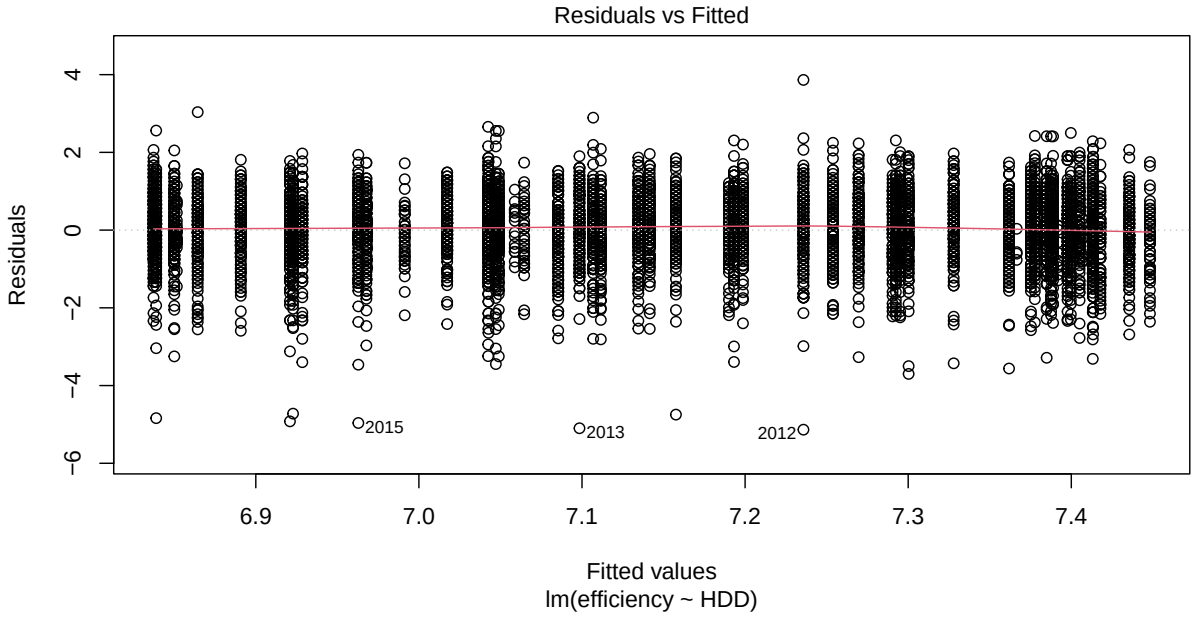
```
##
```

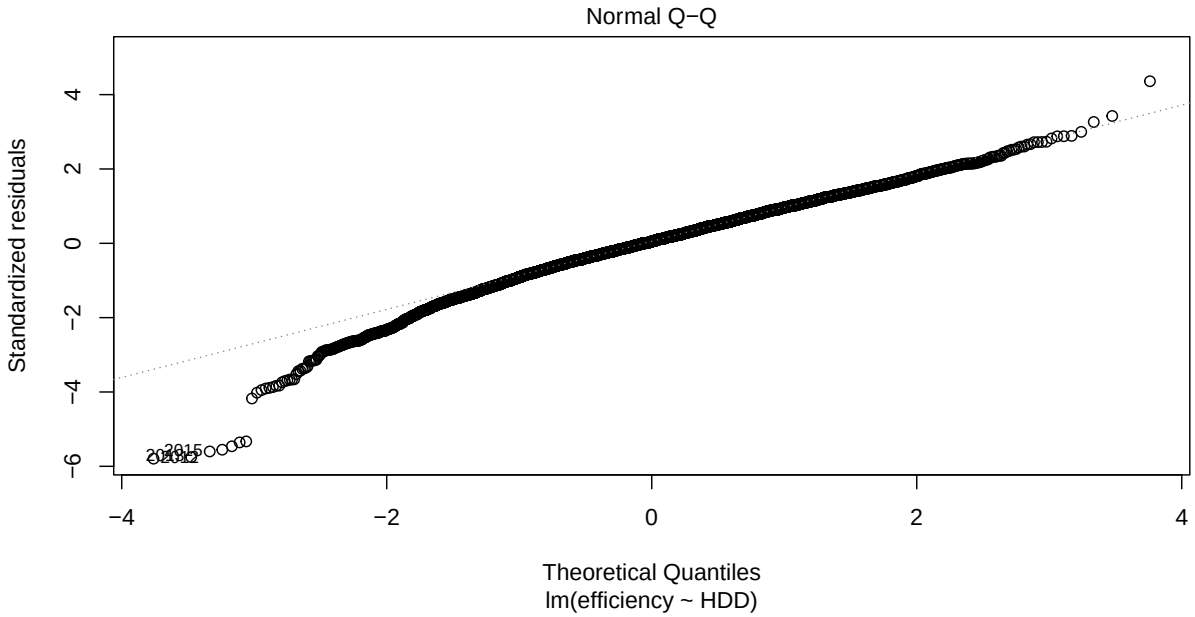
```
## Call:
## lm(formula = efficiency ~ HDD, data = eff_HDD_auckland)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1358 -0.4997  0.0424  0.5949  3.8642
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.4537943  0.0204472  364.54   <2e-16 ***
## HDD         -0.0026065  0.0001566  -16.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8863 on 5878 degrees of freedom
## Multiple R-squared:  0.04504,    Adjusted R-squared:  0.04487
## F-statistic: 277.2 on 1 and 5878 DF,  p-value: < 2.2e-16
```

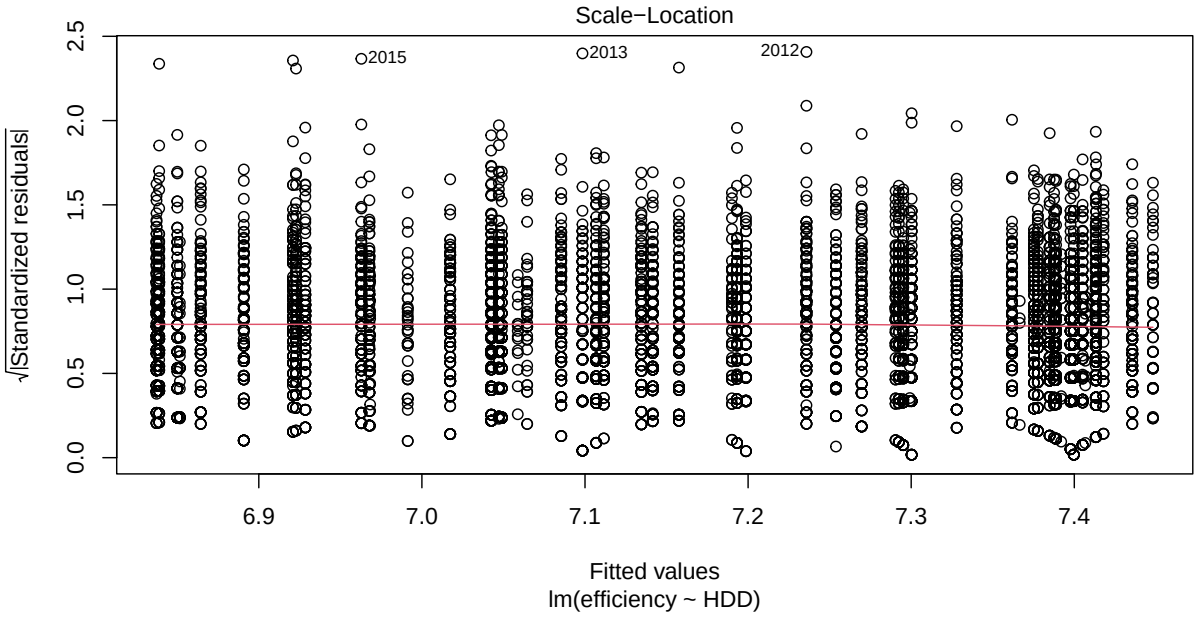
```
plot(eff_HDD_auckland$HDD,eff_HDD_auckland$efficiency)
abline(all_lm)
```

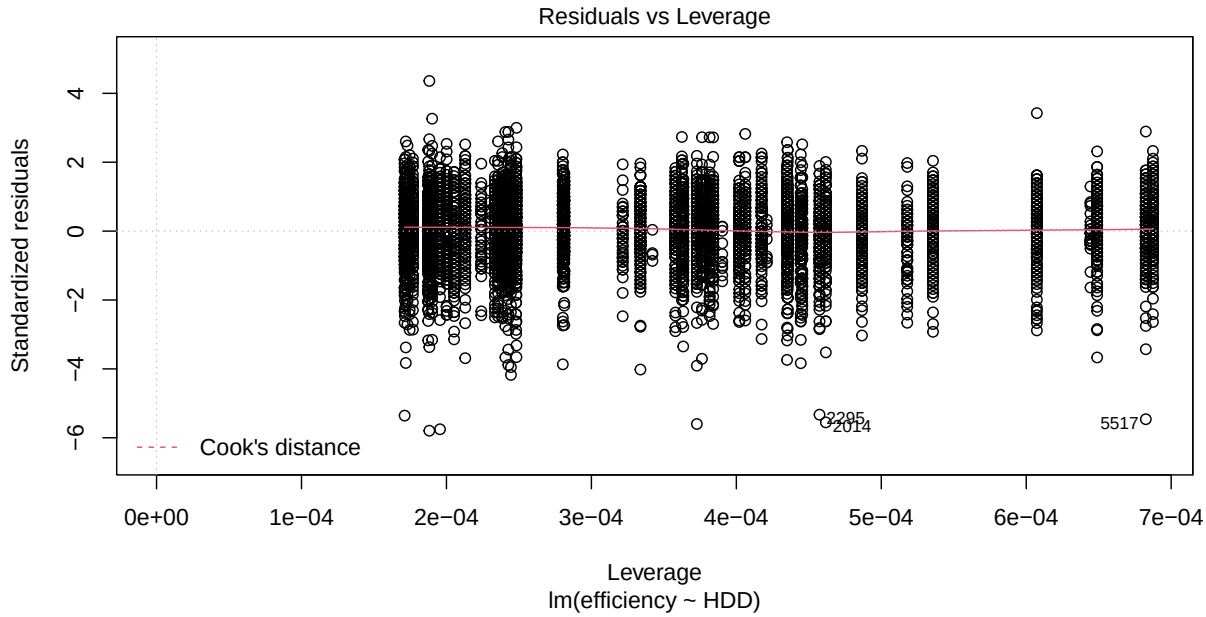



```
plot(all_lm)
```


Residuals vs Fitted
Residuals
Fitted values
lm(efficiency ~ HDD)
2015
2013
2012


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(efficiency ~ HDD)
2015





it is using the most common EV (Nissan Leaf (24 kWh) 2013-2016) as the base line. even with many more predictors (vehicle type categories) the Std. Error of the HDD predictor decreased so t-value has actually increased

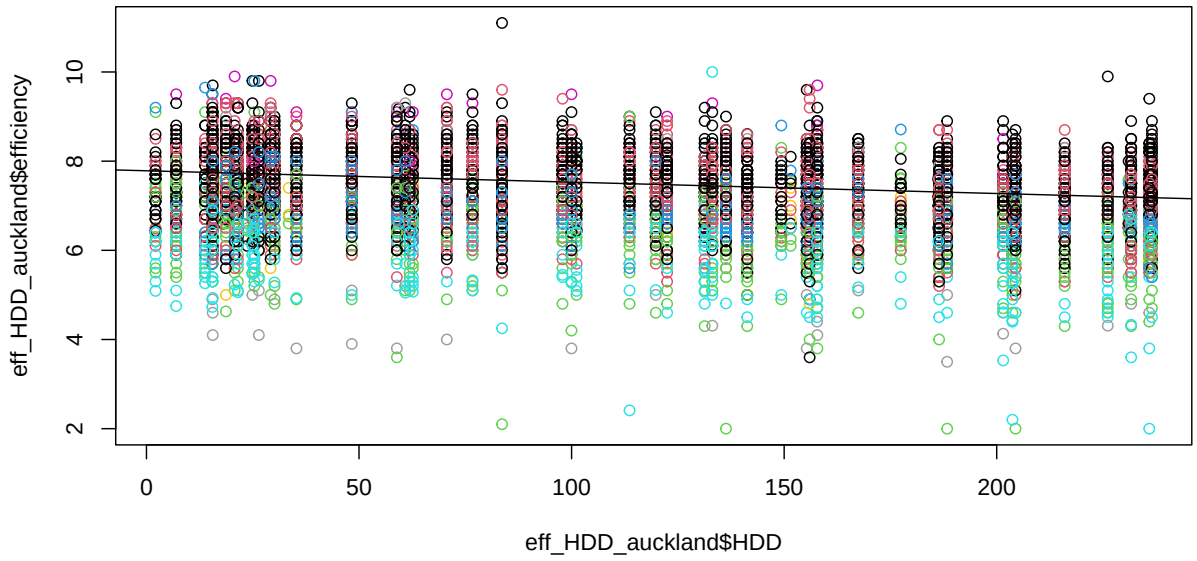
```
all_mdl_lm = lm(data = eff_HDD_auckland, efficiency~HDD+model )
summary(all_mdl_lm)
```

```
##
## Call:
```

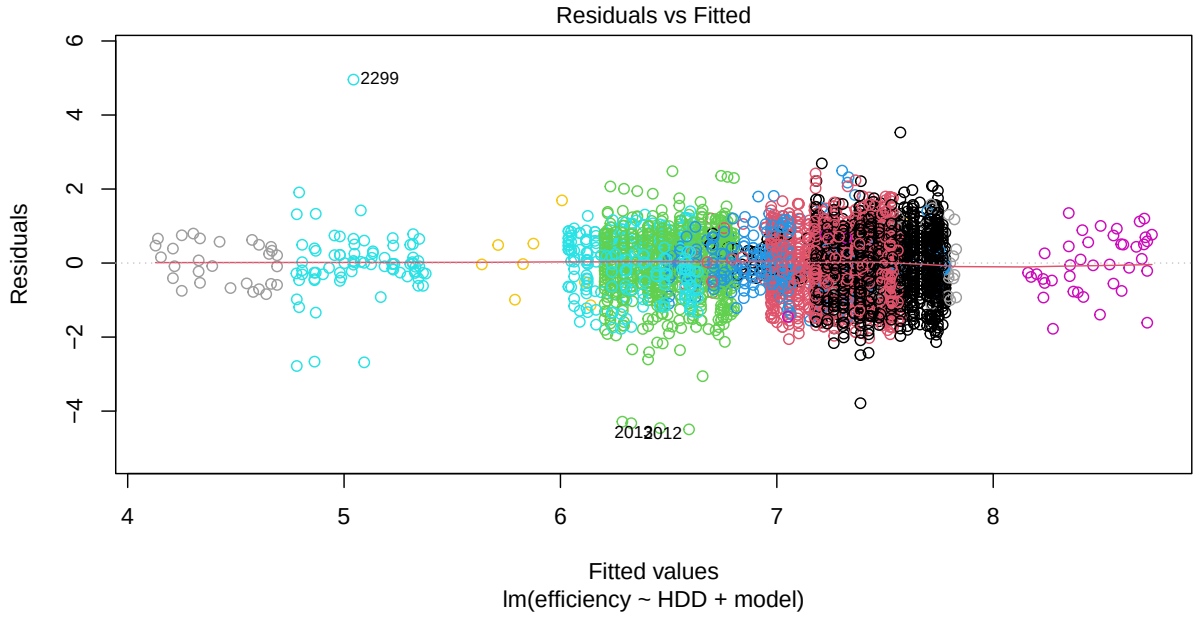
```
## lm(formula = efficiency ~ HDD + model, data = eff_HDD_auckland)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4941 -0.4589  0.0069  0.5004  4.9565
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          7.784172   0.020514 379.465  < 2e-16 ***
## HDD                                 -0.002549   0.000129 -19.760  < 2e-16 ***
## modelNissan Leaf (30 kWh)           -0.207429   0.023246  -8.923  < 2e-16 ***
## modelNissan Leaf (24 kWh) 2011-2012 -0.976933   0.029239 -33.412  < 2e-16 ***
## modelNissan Leaf (40 kWh)           -0.680381   0.059116 -11.509  < 2e-16 ***
## modelNissan e-NV200 (24 kWh)        -1.144450   0.046147 -24.800  < 2e-16 ***
## modelHyundai Ioniq (EV)              0.967094   0.116365   8.311  < 2e-16 ***
## modelBMW i3                         -0.133892   0.084522  -1.584  0.11322
## modelHyundai Kona (EV)               0.084274   0.096196   0.876  0.38103
## modelRenault Zoe                    -0.490255   0.111042  -4.415 1.03e-05 ***
## modelTesla Model 3                  -0.685407   0.117832  -5.817 6.31e-09 ***
## modelNissan Leaf (62 kWh)           -0.747938   0.326801  -2.289  0.02213 *
## modelKia Niro (EV)                   0.034085   0.129881   0.262  0.79300
## modelTesla Model S                  -2.401467   0.084006 -28.587  < 2e-16 ***
## modelTesla Model-X                  -3.054115   0.136349 -22.399  < 2e-16 ***
## modelKia Soul                       -0.425908   0.099548  -4.278 1.91e-05 ***
## modelMG ZS EV                       -0.630101   0.231259  -2.725  0.00646 **
## modelToyota Prius                   -0.417718   0.090421  -4.620 3.93e-06 ***
## modelMitsubishi Outlander           -0.398441   0.421610  -0.945  0.34467
## modelAudi A3 e-tron                 -1.596934   0.258446  -6.179 6.89e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7297 on 5860 degrees of freedom
## Multiple R-squared:  0.3547, Adjusted R-squared:  0.3526
## F-statistic: 169.5 on 19 and 5860 DF,  p-value: < 2.2e-16
```

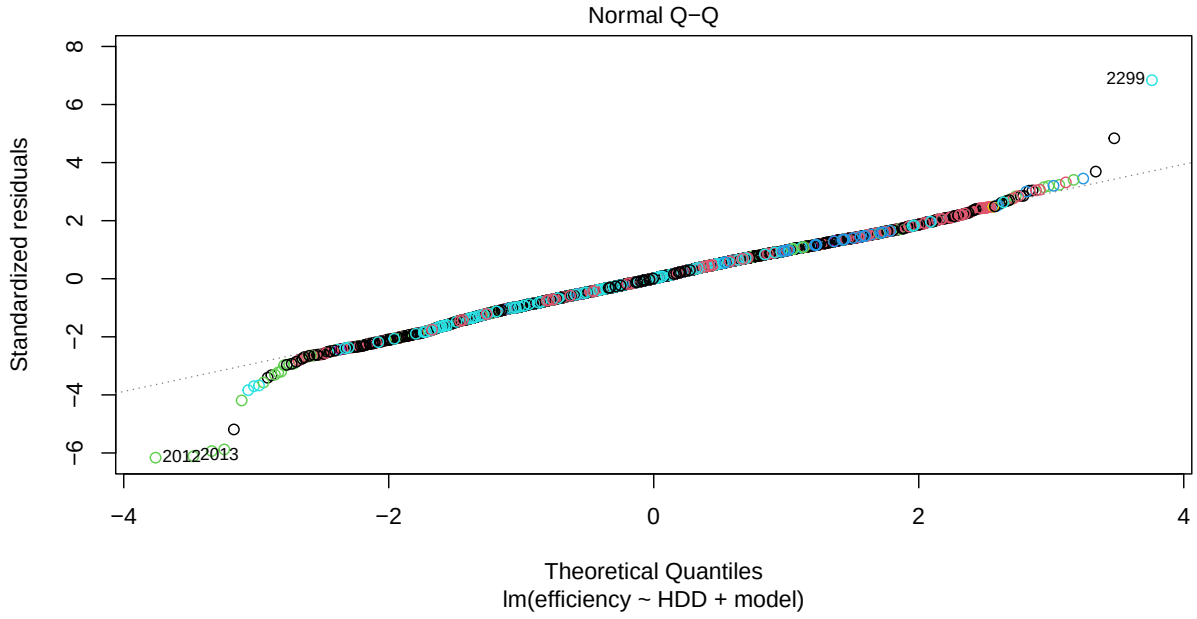
```
plot(eff_HDD_auckland$HDD,eff_HDD_auckland$efficiency, col = eff_HDD_auckland$model)
abline(all_mdl_lm)
```

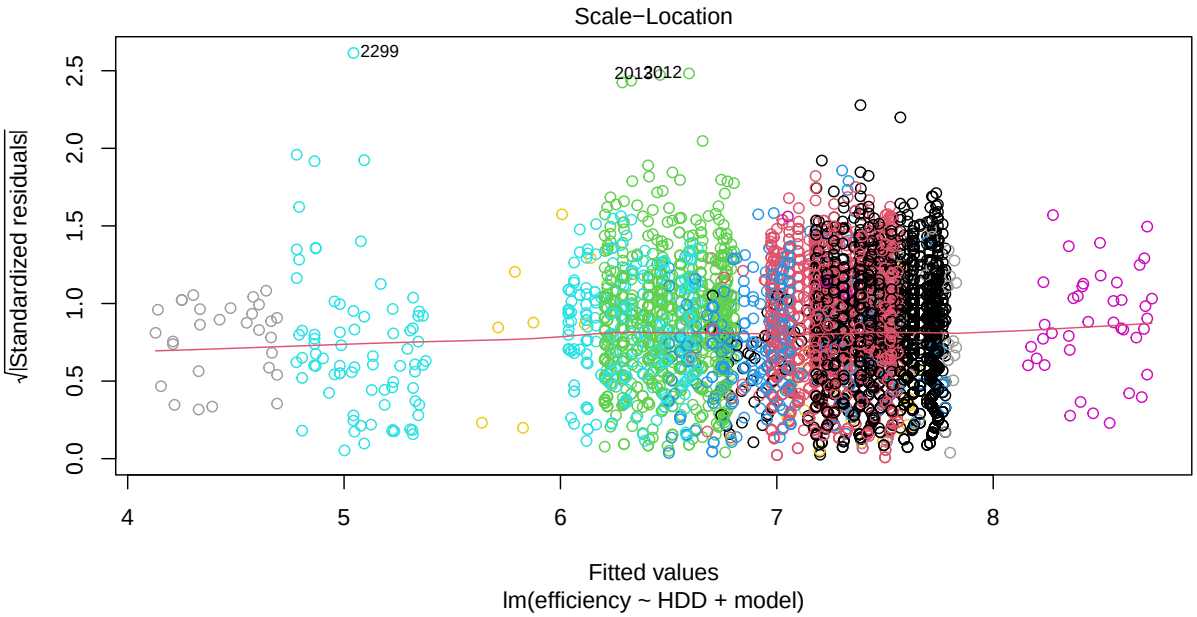
```
## Warning in abline(all_mdl_lm): only using the first two of 20 regression
## coefficients
```

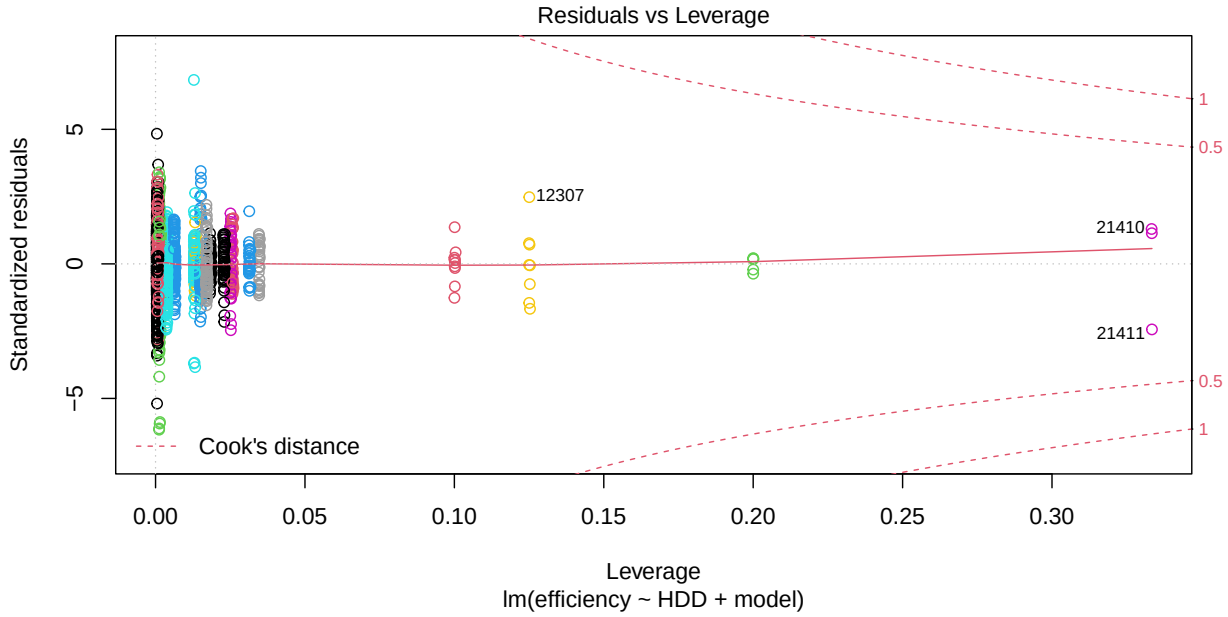



```
plot(all_mdl_lm, col = eff_HDD_auckland$model)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(efficiency ~ HDD + model)
2299
2012
2013


Scale−Location
√|Standardized residuals|
Fitted values
lm(efficiency ~ HDD + model)
2299
2013
2012



residual is well centered for each of the car and approx normal distribution except for a couple that have extremely low efficiency (hills, aggressive driving?)

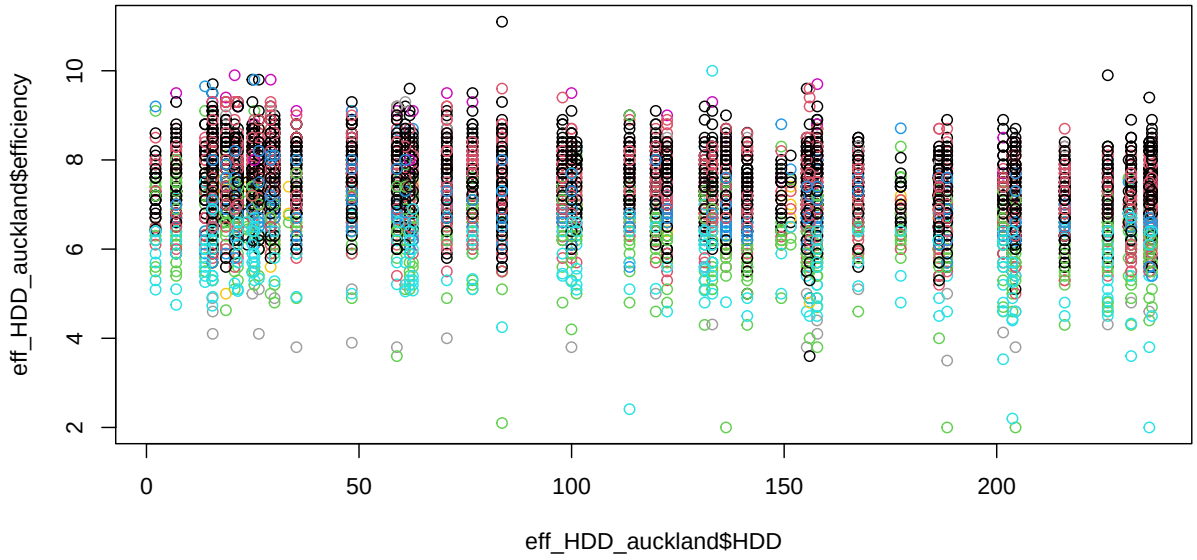
```
inter_mdl_lm = lm(data = eff_HDD_auckland, efficiency~HDD*model )
summary(inter_mdl_lm)
```

```
##
## Call:
## lm(formula = efficiency ~ HDD * model, data = eff_HDD_auckland)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5202 -0.4548  0.0100  0.4969  4.9684
##
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             7.7419247  0.0266540 290.460  < 2e-16
## HDD                                    -0.0021567  0.0002041 -10.567  < 2e-16
## modelNissan Leaf (30 kWh)              -0.1739057  0.0410792  -4.233 2.34e-05
## modelNissan Leaf (24 kWh) 2011-2012    -0.8133035  0.0511355 -15.905  < 2e-16
## modelNissan Leaf (40 kWh)              -0.5827516  0.1072433  -5.434 5.74e-08
## modelNissan e-NV200 (24 kWh)           -1.1724003  0.0808166 -14.507  < 2e-16
## modelHyundai Ioniq (EV)                 1.4366873  0.2150252   6.681 2.58e-11
## modelBMW i3                             0.0633159  0.1486335   0.426 0.670133
## modelHyundai Kona (EV)                  0.1552342  0.1802912   0.861 0.389263
## modelRenault Zoe                       -0.7565049  0.1936976  -3.906 9.51e-05
## modelTesla Model 3                     -0.5561063  0.2092454  -2.658 0.007890
## modelNissan Leaf (62 kWh)              -1.1472231  2.1054653  -0.545 0.585858
## modelKia Niro (EV)                      0.0538757  0.2326336   0.232 0.816864
## modelTesla Model S                     -2.2579304  0.1558622 -14.487  < 2e-16
## modelTesla Model-X                     -3.2253599  0.2445113 -13.191  < 2e-16
## modelKia Soul                          -0.2554847  0.1754437  -1.456 0.145385
```

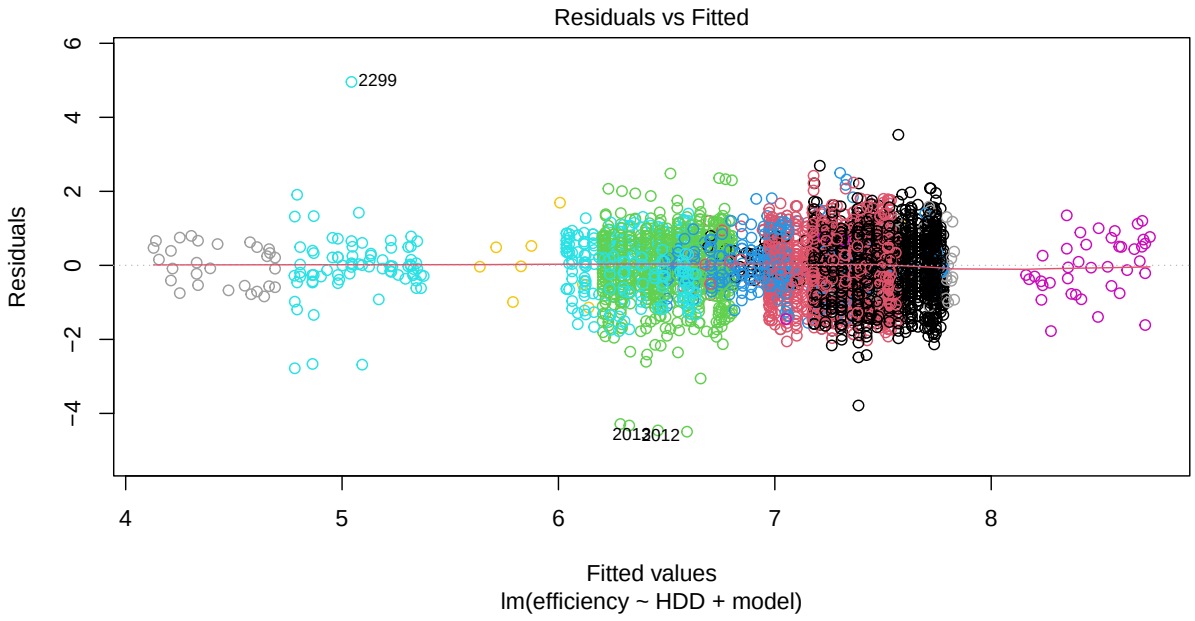
```
## modelMG ZS EV                            -0.9293258  0.4011377  -2.317 0.020553
## modelToyota Prius                        -0.4284212  0.1533694  -2.793 0.005233
## modelMitsubishi Outlander                 1.2235486  0.8151566   1.501 0.133410
## modelAudi A3 e-tron                      -1.7896231  0.5189926  -3.448 0.000568
## HDD:modelNissan Leaf (30 kWh)            -0.0003109  0.0003151  -0.987 0.323864
## HDD:modelNissan Leaf (24 kWh) 2011-2012 -0.0015308  0.0003927  -3.898 9.81e-05
## HDD:modelNissan Leaf (40 kWh)            -0.0009055  0.0008301  -1.091 0.275408
## HDD:modelNissan e-NV200 (24 kWh)          0.0002525  0.0006104   0.414 0.679193
## HDD:modelHyundai Ioniq (EV)              -0.0043135  0.0016606  -2.597 0.009415
## HDD:modelBMW i3                          -0.0017688  0.0010886  -1.625 0.104274
## HDD:modelHyundai Kona (EV)               -0.0006516  0.0013799  -0.472 0.636780
## HDD:modelRenault Zoe                      0.0024832  0.0014803   1.677 0.093501
## HDD:modelTesla Model 3                   -0.0012981  0.0017980  -0.722 0.470348
## HDD:modelNissan Leaf (62 kWh)             0.0021877  0.0121543   0.180 0.857166
## HDD:modelKia Niro (EV)                   -0.0001831  0.0017974  -0.102 0.918853
## HDD:modelTesla Model S                   -0.0012421  0.0011060  -1.123 0.261451
## HDD:modelTesla Model-X                    0.0014817  0.0017840   0.831 0.406266
## HDD:modelKia Soul                        -0.0015817  0.0013419  -1.179 0.238549
## HDD:modelMG ZS EV                         0.0030335  0.0032898   0.922 0.356519
## HDD:modelToyota Prius                     0.0001184  0.0011941   0.099 0.921016
## HDD:modelMitsubishi Outlander            -0.0220712  0.0095775  -2.304 0.021231
## HDD:modelAudi A3 e-tron                   0.0016057  0.0038301   0.419 0.675071
##
## (Intercept)                              ***
## HDD                                      ***
## modelNissan Leaf (30 kWh)                ***
## modelNissan Leaf (24 kWh) 2011-2012      ***
## modelNissan Leaf (40 kWh)                ***
## modelNissan e-NV200 (24 kWh)             ***
## modelHyundai Ioniq (EV)                  ***
## modelBMW i3
## modelHyundai Kona (EV)
## modelRenault Zoe                         ***
## modelTesla Model 3                       **
## modelNissan Leaf (62 kWh)
## modelKia Niro (EV)
## modelTesla Model S                       ***
## modelTesla Model-X                       ***
## modelKia Soul
## modelMG ZS EV                            *
## modelToyota Prius                        **
## modelMitsubishi Outlander
## modelAudi A3 e-tron                      ***
## HDD:modelNissan Leaf (30 kWh)
## HDD:modelNissan Leaf (24 kWh) 2011-2012 ***
## HDD:modelNissan Leaf (40 kWh)
## HDD:modelNissan e-NV200 (24 kWh)
## HDD:modelHyundai Ioniq (EV)              **
## HDD:modelBMW i3
## HDD:modelHyundai Kona (EV)
## HDD:modelRenault Zoe                     .
## HDD:modelTesla Model 3
## HDD:modelNissan Leaf (62 kWh)
## HDD:modelKia Niro (EV)
```

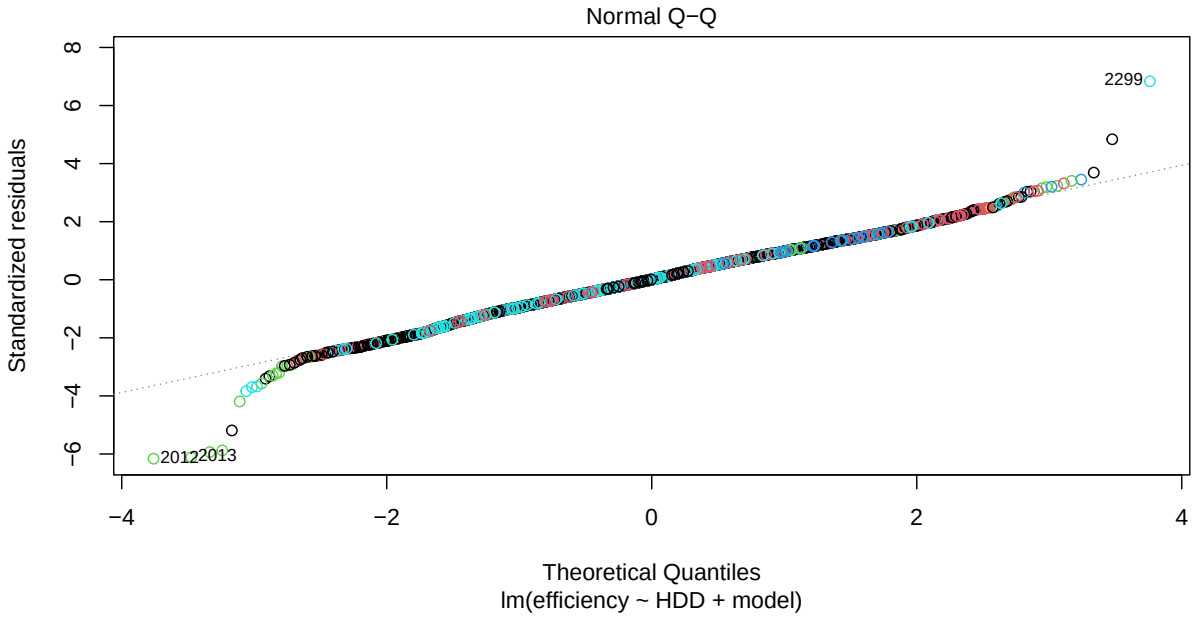
```
## HDD:modelTesla Model S
## HDD:modelTesla Model-X
## HDD:modelKia Soul
## HDD:modelMG ZS EV
## HDD:modelToyota Prius
## HDD:modelMitsubishi Outlander           *
## HDD:modelAudi A3 e-tron
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7285 on 5842 degrees of freedom
## Multiple R-squared:  0.3588, Adjusted R-squared:  0.3547
## F-statistic: 88.35 on 37 and 5842 DF,  p-value: < 2.2e-16
```

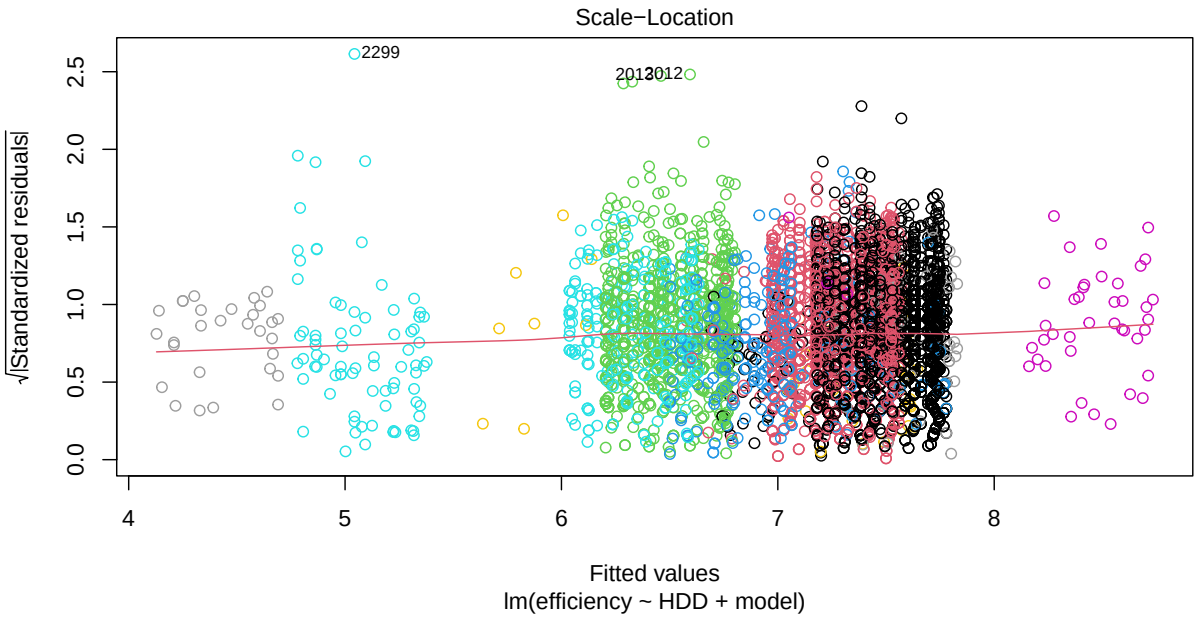
```
plot(eff_HDD_auckland$HDD,eff_HDD_auckland$efficiency, col = eff_HDD_auckland$model)
```

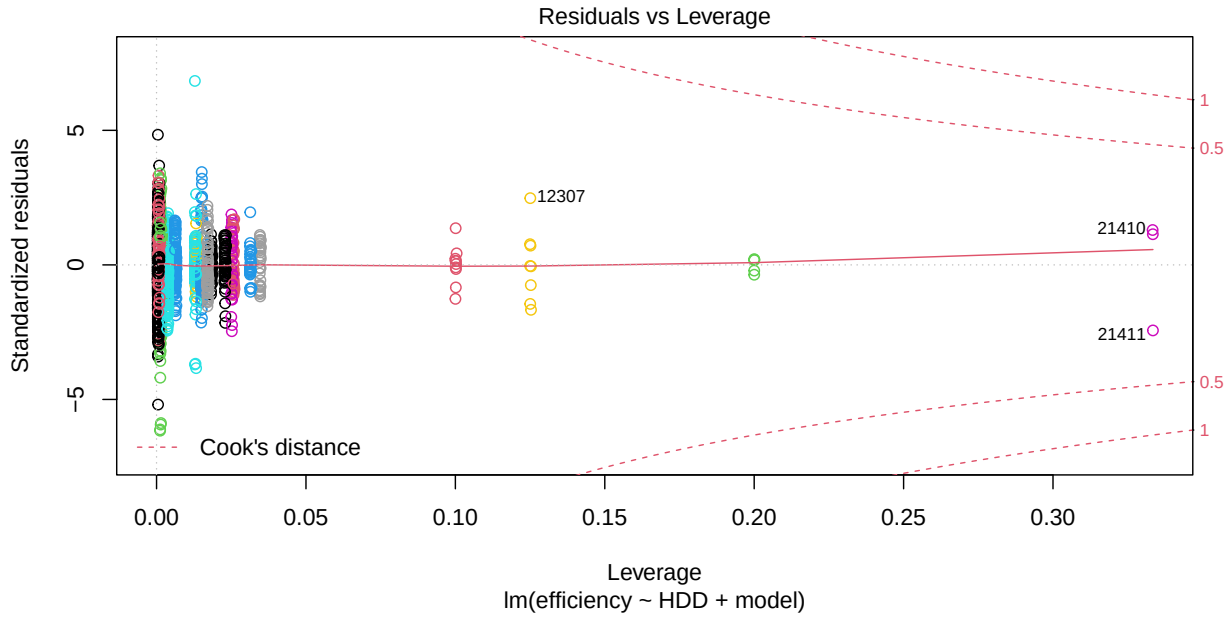



```
plot(all_mdl_lm, col = eff_HDD_auckland$model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(efficiency ~ HDD + model)
2299
2012
2012


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(efficiency ~ HDD + model)
2299
2012
2013





including model of car in model is worth it, interaction term anova says is significant but fit does not really improve and most are not significant. i.e not really worth being concerned about how different car respond to difference in temps

```
anova(all_lm,all_mdl_lm, inter_mdl_lm)
```

```
## Analysis of Variance Table
##
## Model 1: efficiency ~ HDD
```

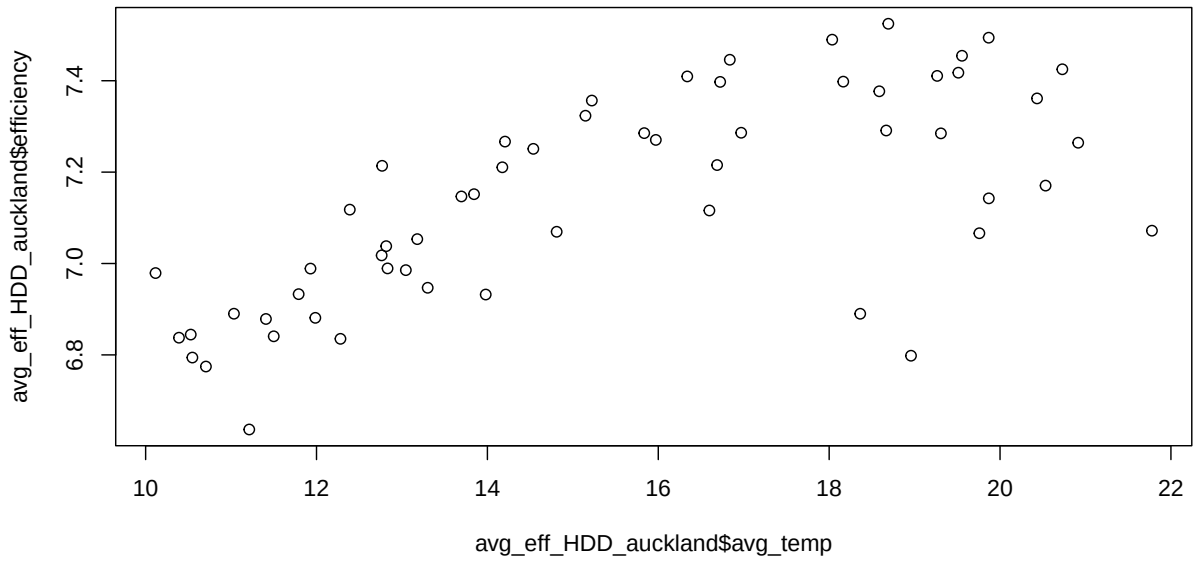
```
## Model 2: efficiency ~ HDD + model
## Model 3: efficiency ~ HDD * model
##   Res.Df    RSS Df Sum of Sq        F    Pr(>F)
## 1   5878 4617.8
## 2   5860 3120.6 18   1497.24 156.7260 < 2.2e-16 ***
## 3   5842 3100.6 18     20.01   2.0951  0.004319 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
auck_avg_temp = temp_data %>%
  group_by(Year, Month) %>%
  summarise(avg_temp = mean(Temp))
```

```
## `summarise()` has grouped output by 'Year'. You can override using the `.groups` argument.
```

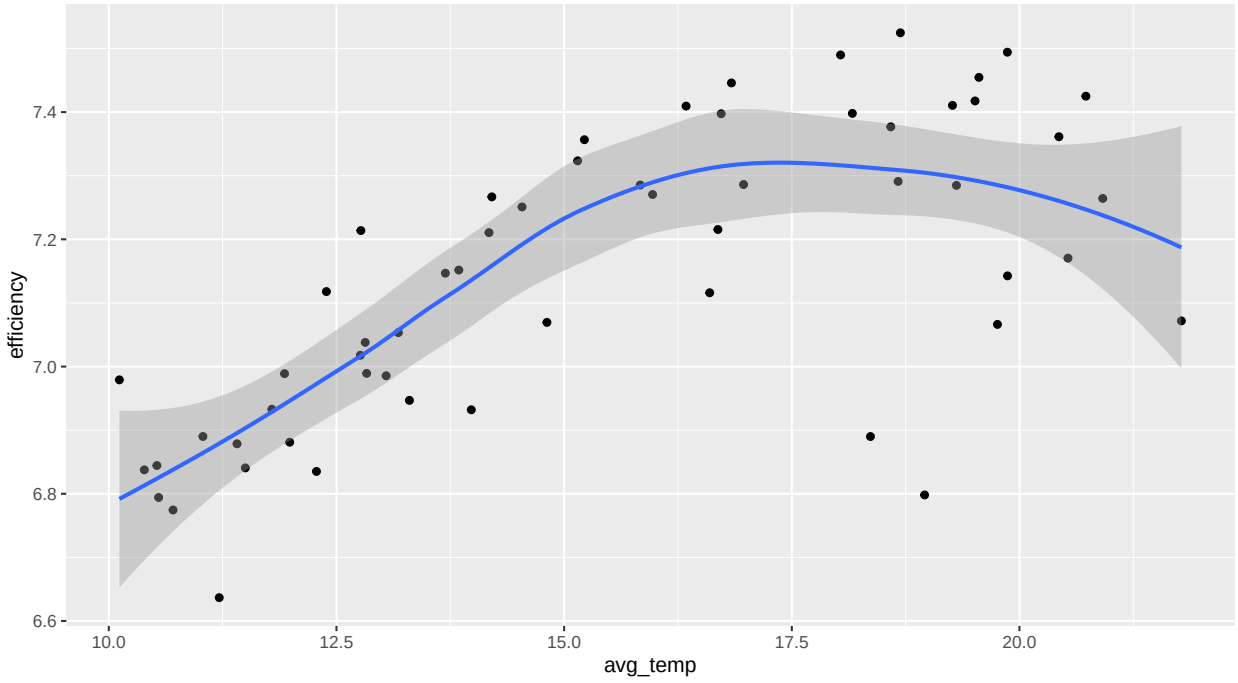
```
avg_eff_HDD_auckland = merge(avg_eff_HDD_auckland, auck_avg_temp, by.x = c('year', 'Month'), by.y = c("
```

```
plot(avg_eff_HDD_auckland$avg_temp,avg_eff_HDD_auckland$efficiency)
```

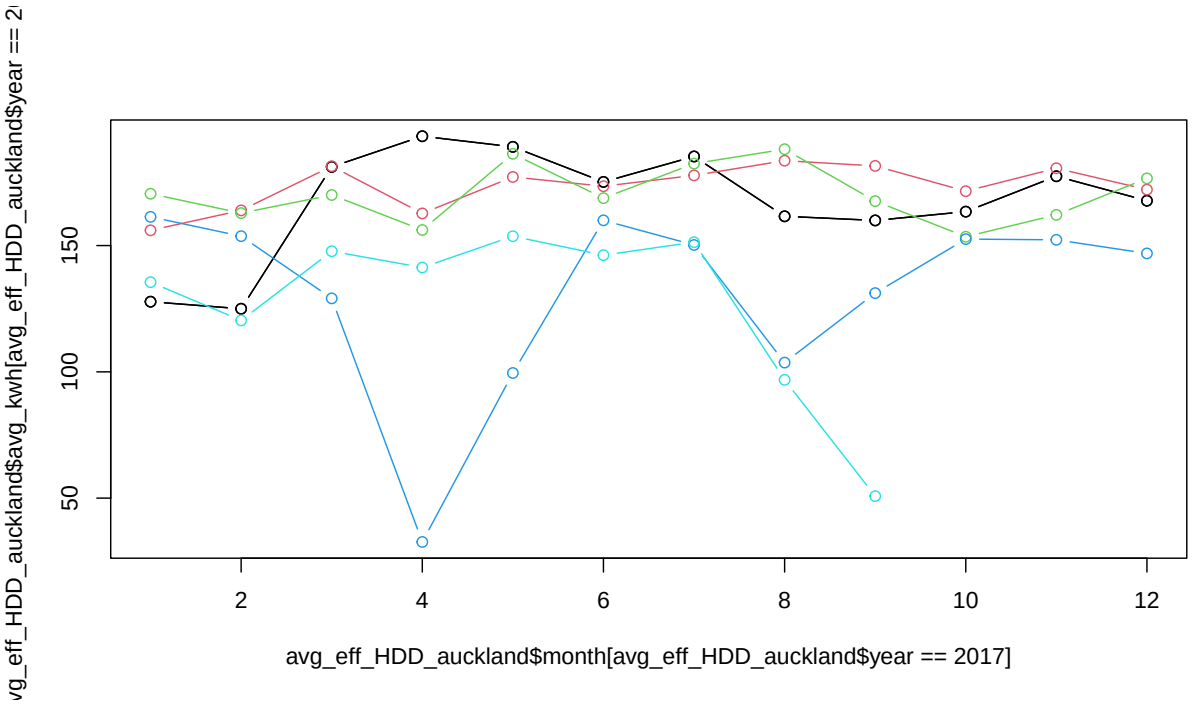



```
avg_eff_HDD_auckland %>% ggplot(aes(avg_temp, efficiency)) + geom_point() + geom_smooth(method = 'loess
```

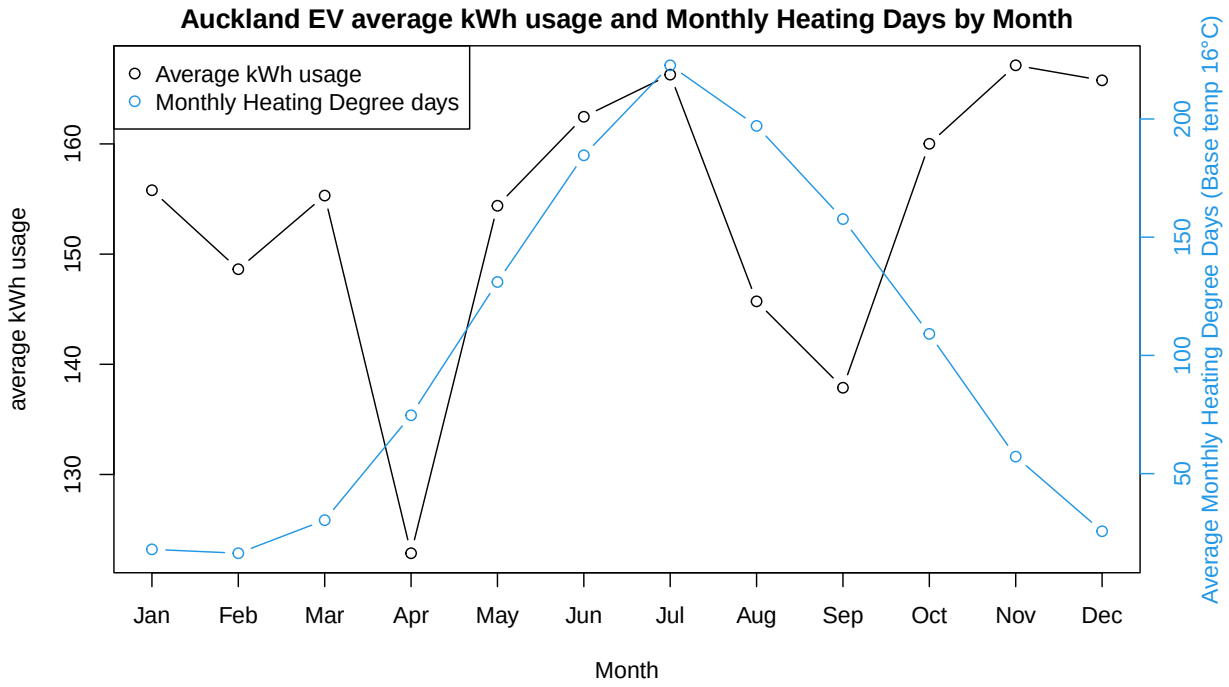
```
## `geom_smooth()` using formula 'y ~ x'
```




```
plot(avg_eff_HDD_auckland$month[avg_eff_HDD_auckland$year == 2017], avg_eff_HDD_auckland$avg_kwh[avg_ef
     type = 'b', col = 1 , ylim = c(min(avg_eff_HDD_auckland$avg_kwh),max(avg_eff_HDD_auckland$avg_kwh)
for (i in 2017:2021) {
  points(avg_eff_HDD_auckland$month[avg_eff_HDD_auckland$year == i], avg_eff_HDD_auckland$avg_kwh[avg_e
      type = 'b', col = i-2016 ,xaxt = "n", ylim = c(min(avg_eff_HDD_auckland$avg_kwh),max(avg_eff_HDD_a

}
```



```r
par(mar = c(4, 4, 2, 4))
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$kwh, type = 'b', xaxt = "n",
     ylab = "average kWh usage", xlab = "Month", main = "Auckland EV average kWh usage and Monthly Heati
axis(1, labels = as.character(mean_HDD_auckland$Month), at = as.numeric(mean_HDD_auckland$Month))

par(new=TRUE)
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$average_HDD, xlab="", ylab="", xlim = c(1,1
    ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    axes=FALSE, type="b", col=4)
mtext("Average Monthly Heating Degree Days (Base temp 16°C)",side=4,col=4,line=2, cex = 1)
axis(4, ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    col=4,col.axis=4)
legend("topleft", legend = c( "Average kWh usage", "Monthly Heating Degree days"), col = c(1, 4), pch =
```



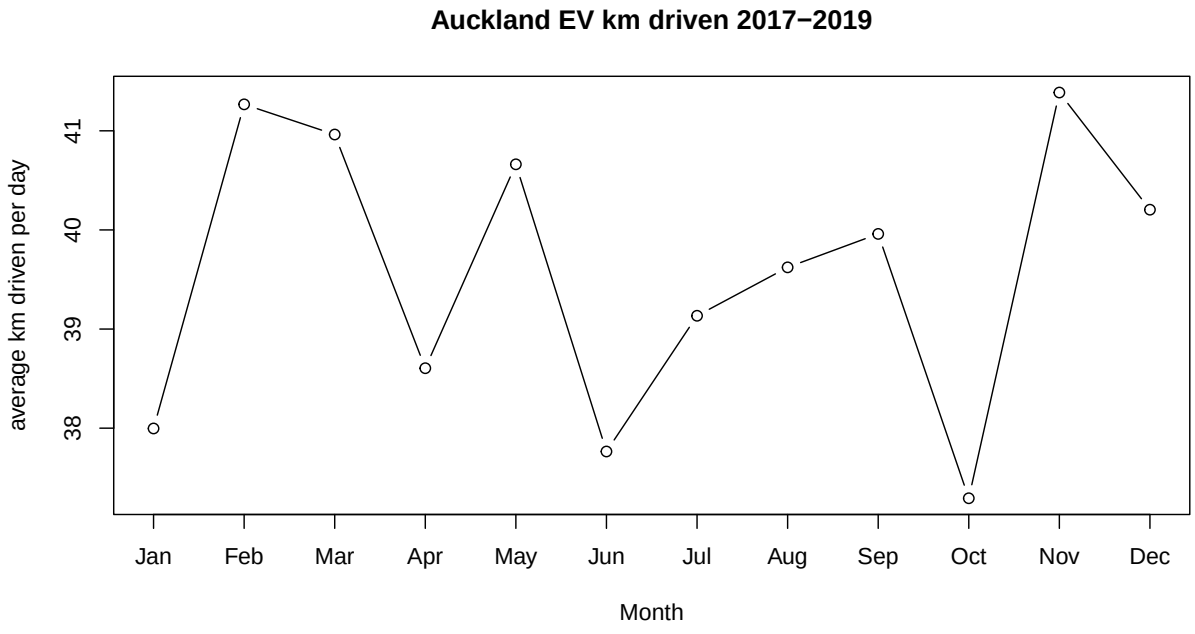
lockdowns make it too hard to see much meaningful pattern

```r
mean_HDD_auckland = HDD_auckland %>%
  group_by(Month) %>%
  summarise(average_HDD = mean(HDD))

power = c()
eff = c()
kwh = c()
km = c()
for (i in 1:12) {
  eff[i] = mean(data[month==i & region == "Auckland" & year >= 2017 & year <= 2019,]$efficiency, na.rm =
  kwh[i] = mean(data[month==i & region == "Auckland" & year >= 2017 & year <= 2019,]$kwh, na.rm = T)
  km[i] = mean(data[month==i & region == "Auckland" & year >= 2017 & year <= 2019,]$distance, na.rm = T
}

mean_HDD_auckland$power_usage = power
```

```
mean_HDD_auckland$efficiency = eff
mean_HDD_auckland$kwh = kwh
mean_HDD_auckland$km = km
mean_HDD_auckland
```

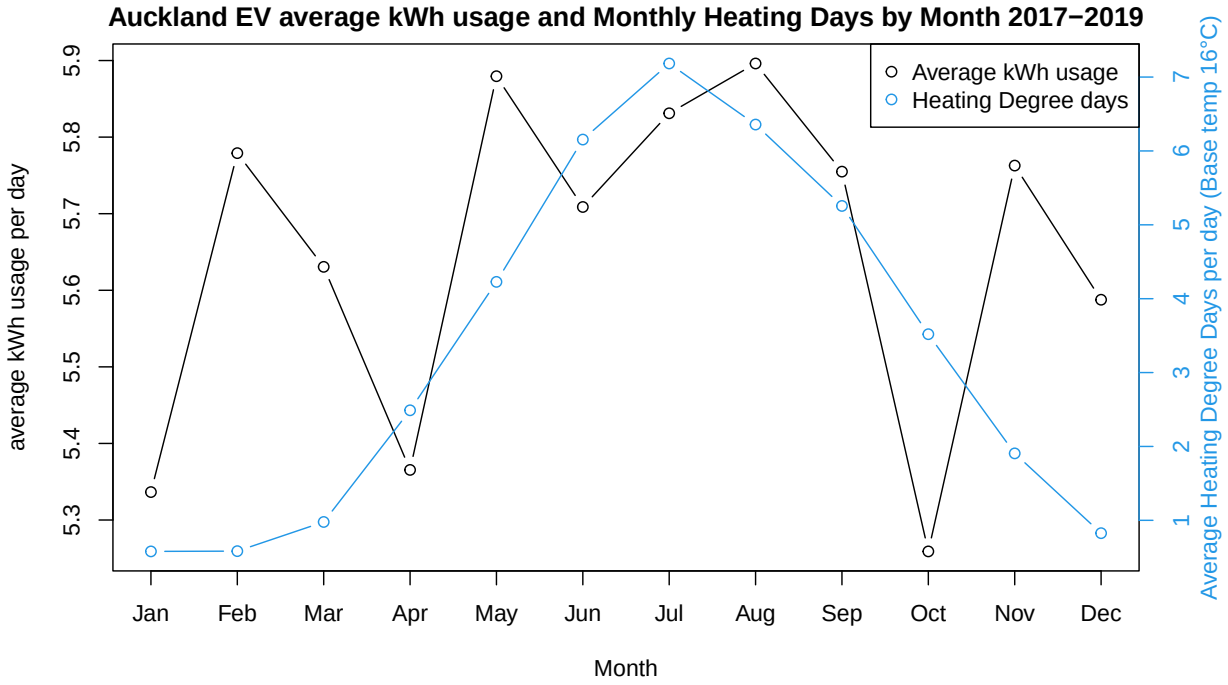
```
## # A tibble: 12 x 5
##    Month average_HDD efficiency   kwh    km
##    <fct>       <dbl>      <dbl> <dbl> <dbl>
##  1 Jan          18.0       7.20  165. 1178.
##  2 Feb          16.3       7.29  162. 1155.
##  3 Mar          30.3       7.36  175. 1270.
##  4 Apr          74.7       7.29  161. 1158.
##  5 May         131.        7.09  182. 1261.
##  6 Jun         185.        6.83  171. 1133.
##  7 Jul         223.        6.81  181. 1213.
##  8 Aug         197.        6.84  183. 1228.
##  9 Sep         158.        7.06  173. 1199.
## 10 Oct         109.        7.18  163. 1156.
## 11 Nov          57.2       7.31  173. 1242.
## 12 Dec          25.6       7.30  173. 1246.
```

```
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$km/month_length, type = 'b', xaxt = "n",
     ylab = "average km driven per day", xlab = "Month", main = "Auckland EV km driven 2017-2019")
axis(1, labels = as.character(mean_HDD_auckland$Month), at = as.numeric(mean_HDD_auckland$Month))
```




```
par(mar = c(4, 4, 2, 4))
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$kwh/month_length, type = 'b', xaxt = "n",
     ylab = "average kWh usage per day", xlab = "Month", main = "Auckland EV average kWh usage and Montl
axis(1, labels = as.character(mean_HDD_auckland$Month), at = as.numeric(mean_HDD_auckland$Month))
```

```
par(new=TRUE)
plot(as.numeric(mean_HDD_auckland$Month), mean_HDD_auckland$average_HDD/month_length, xlab="", ylab="",
    ylim=c(min(mean_HDD_auckland$average_HDD/month_length),max(mean_HDD_auckland$average_HDD/month_leng
    axes=FALSE, type="b", col=4)
mtext("Average Heating Degree Days per day (Base temp 16°C)",side=4,col=4,line=2, cex = 1)
axis(4, ylim=c(min(mean_HDD_auckland$average_HDD),max(mean_HDD_auckland$average_HDD)),
    col=4,col.axis=4)
legend("topright", legend = c( "Average kWh usage", "Heating Degree days"), col = c(1, 4), pch = 1)
```



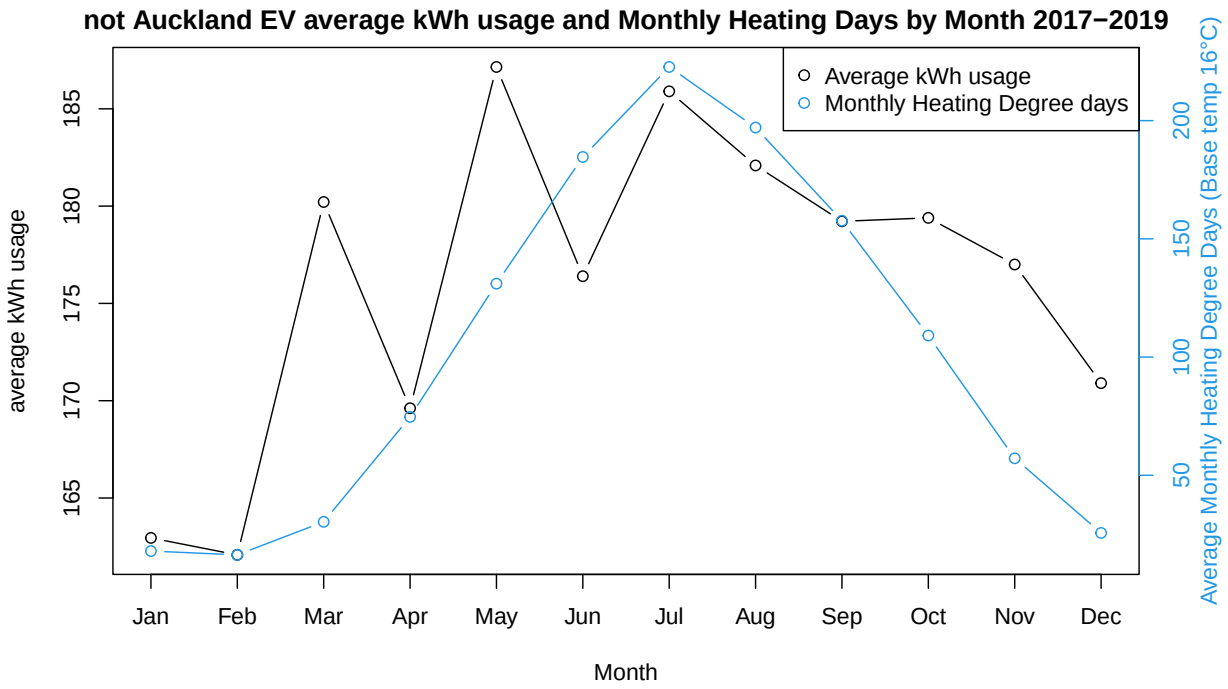
EV power used data goes to the contrary of the fuel usage data. while it is clear that efficiency decreases in the winter months going by fuel usage data total power used should be less in the winter as there is significantly more fuel usage in the summer. is this cause petrol fuel efficiency is lower in summer (possibly due to AC), current EV drivers are different (early adopters or EV commute with 2nd petrol road trip car), or a regional thing?

```
mean_HDD_not_auckland = HDD_auckland %>%
  group_by(Month) %>%
  summarise(average_HDD = mean(HDD))

power = c()
eff = c()
kwh = c()
for (i in 1:12) {
  eff[i] = mean(data[month==i & region != "Auckland" & year >= 2017 & year <= 2019,]$efficiency, na.rm =
  kwh[i] = mean(data[month==i & region != "Auckland" & year >= 2017 & year <= 2019,]$kwh, na.rm = T)

}

mean_HDD_not_auckland$power_usage = power
mean_HDD_not_auckland$efficiency = eff
mean_HDD_not_auckland$kwh = kwh
```

```
mean_HDD_not_auckland
```

```
## # A tibble: 12 x 4
##    Month average_HDD efficiency   kwh
##    <fct>       <dbl>      <dbl> <dbl>
##  1 Jan          18.0       7.05  163.
##  2 Feb          16.3       7.06  162.
##  3 Mar          30.3       7.05  180.
##  4 Apr          74.7       6.77  170.
##  5 May         131.        6.64  187.
##  6 Jun         185.        6.36  176.
##  7 Jul         223.        6.32  186.
##  8 Aug         197.        6.48  182.
##  9 Sep         158.        6.69  179.
## 10 Oct         109.        6.83  179.
## 11 Nov          57.2       6.99  177
## 12 Dec          25.6       7.02  171.
```

```
#not a fair plot as is using Auckland HDD against not auckland kwh
#just used to get an idea for power usage outside auckalnd
par(mar = c(4, 4, 2, 4))
plot(as.numeric(mean_HDD_not_auckland$Month), mean_HDD_not_auckland$kwh, type = 'b', xaxt = "n",
     ylab = "average kWh usage", xlab = "Month", main = "not Auckland EV average kWh usage and Monthly
axis(1, labels = as.character(mean_HDD_not_auckland$Month), at = as.numeric(mean_HDD_not_auckland$Month

par(new=TRUE)
plot(as.numeric(mean_HDD_not_auckland$Month), mean_HDD_not_auckland$average_HDD, xlab="", ylab="", xlim
    ylim=c(min(mean_HDD_not_auckland$average_HDD),max(mean_HDD_not_auckland$average_HDD)),
    axes=FALSE, type="b", col=4)
mtext("Average Monthly Heating Degree Days (Base temp 16°C)",side=4,col=4,line=2, cex = 1)
axis(4, ylim=c(min(mean_HDD_not_auckland$average_HDD),max(mean_HDD_not_auckland$average_HDD)),
    col=4,col.axis=4)
legend("topright", legend = c( "Average kWh usage", "Monthly Heating Degree days"), col = c(1, 4), pch =
```



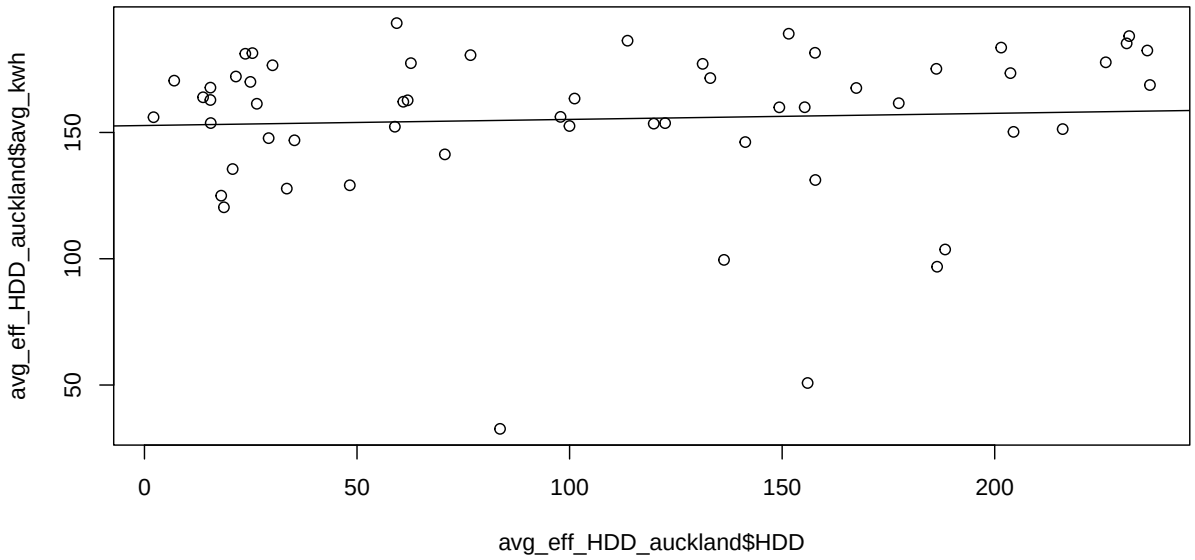
excluding Auckland from the data it appears that it is not a regional thing therefore could be petrol fuel efficiency is lower in summer (possibly) or current EV drivers are different (early adopters or EV commute with 2nd petrol road trip car)? https://www.sciencedirect.com/science/article/pii/S1361920914000923 shows petrol fuel efficiency is lower in the winter

however https://www.proquest.com/docview/2131026877?pq-origsite=gscholar&fromopenview=true cites paper https://www.sae.org/publications/technical-papers/content/2002-01-1957/ that AC accounts for 7% extra power usage

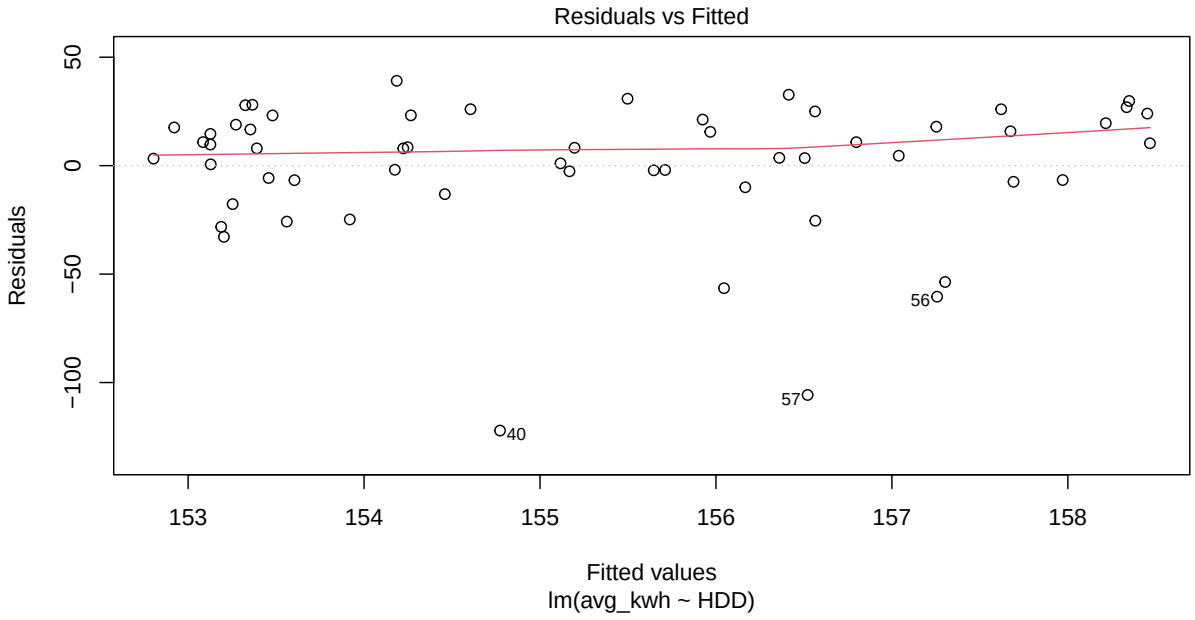
```
all_lm = lm(data = avg_eff_HDD_auckland,avg_kwh~HDD)
summary(all_lm)
```

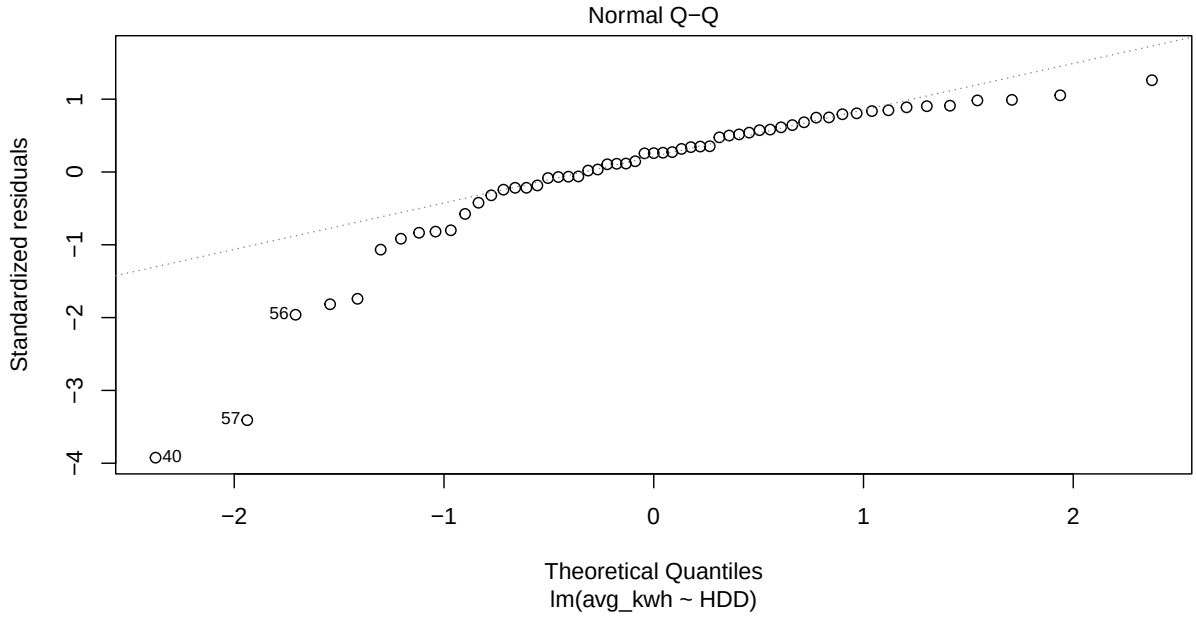
```
##
## Call:
## lm(formula = avg_kwh ~ HDD, data = avg_eff_HDD_auckland)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -122.135   -6.708    7.976   19.572   39.114
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 152.75192    7.19884  21.219   <2e-16 ***
## HDD           0.02416    0.05601   0.431    0.668
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.43 on 55 degrees of freedom
## Multiple R-squared:  0.003372,   Adjusted R-squared:  -0.01475
## F-statistic: 0.1861 on 1 and 55 DF,  p-value: 0.6679
```

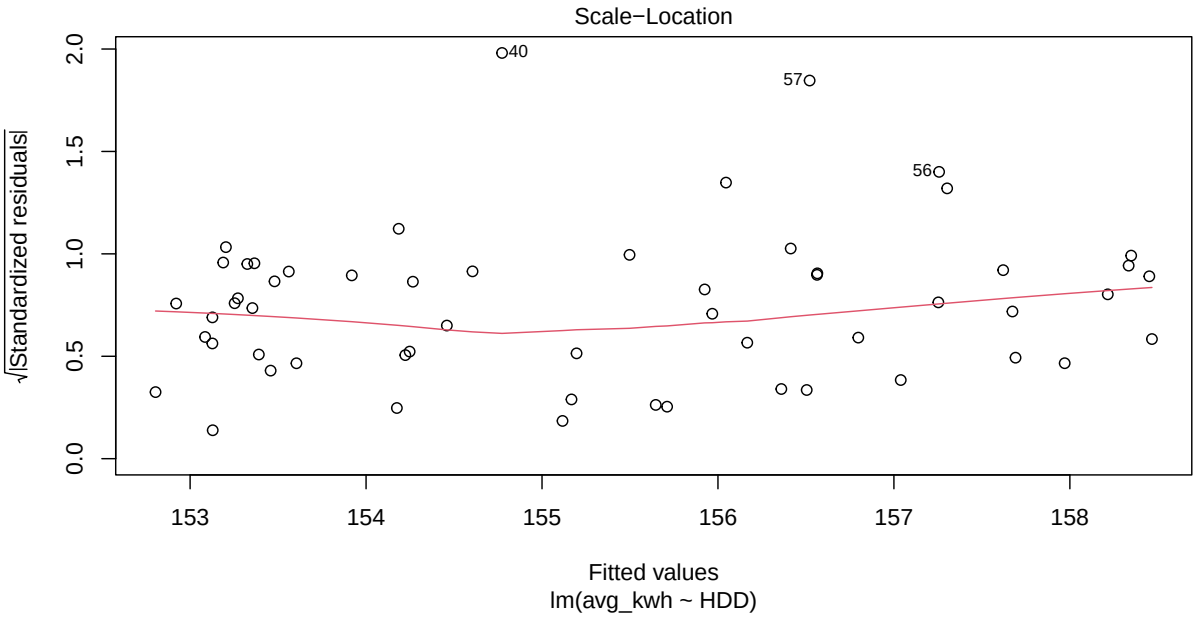
```
plot(avg_eff_HDD_auckland$HDD,avg_eff_HDD_auckland$avg_kwh)
abline(all_lm)
```

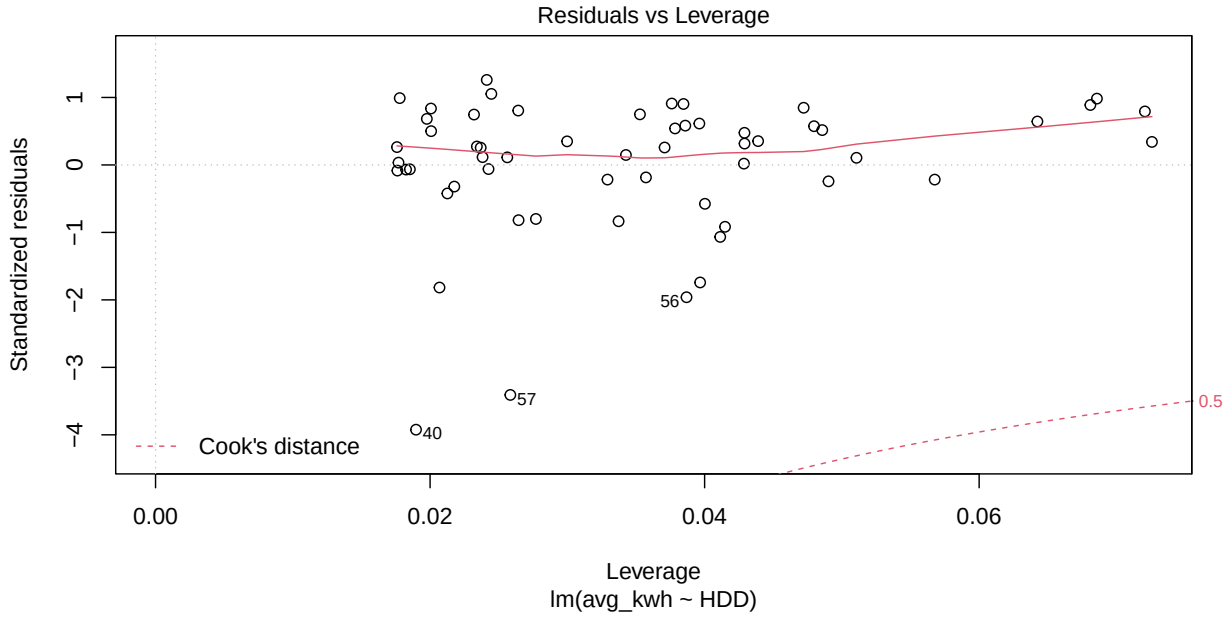



```
plot(all_lm)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(avg_kwh ~ HDD)
40
56
57


Scale−Location
√|Standardized residuals|
Fitted values
lm(avg_kwh ~ HDD)
40
56
57



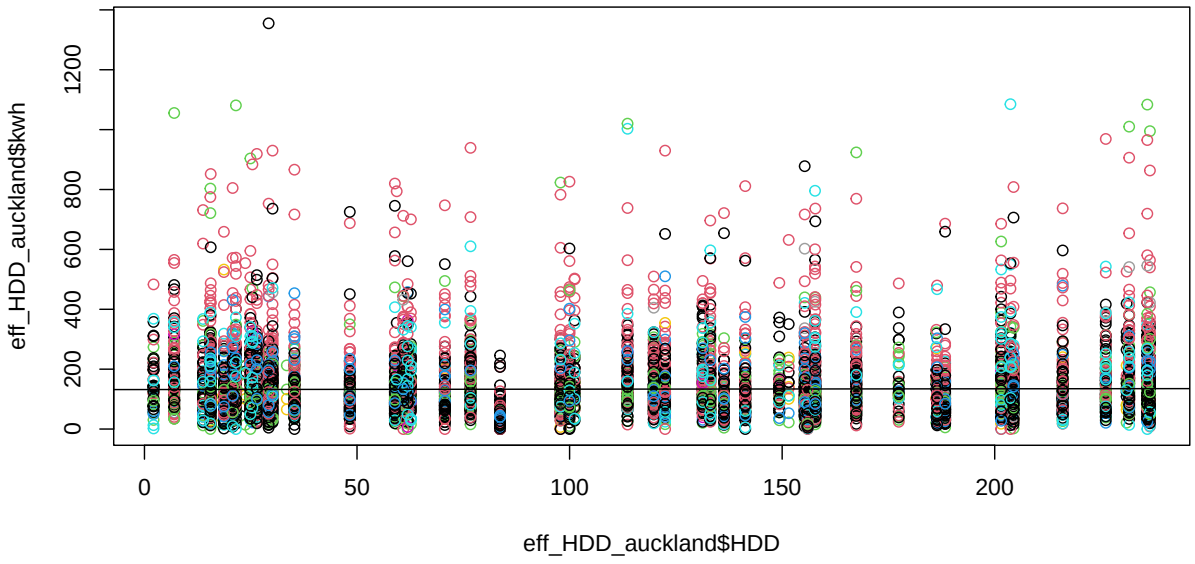

```
all_mdl_lm = lm(data = eff_HDD_auckland, kwh~HDD+model )
summary(all_mdl_lm)
```

```
##
## Call:
## lm(formula = kwh ~ HDD + model, data = eff_HDD_auckland)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -236.04  -72.33  -24.98   38.44 1222.80
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          131.930227   3.365507  39.201  < 2e-16 ***
## HDD                                    0.012631   0.021162   0.597 0.550618
## modelNissan Leaf (30 kWh)             55.081258   3.813767  14.443  < 2e-16 ***
## modelNissan Leaf (24 kWh) 2011-2012    3.595591   4.797043   0.750 0.453560
## modelNissan Leaf (40 kWh)             32.951379   9.698679   3.398 0.000685 ***
## modelNissan e-NV200 (24 kWh)          14.578854   7.571009   1.926 0.054201 .
## modelHyundai Ioniq (EV)               13.273285  19.091074   0.695 0.486919
## modelBMW i3                          -31.437710  13.866842  -2.267 0.023420 *
## modelHyundai Kona (EV)                -6.649758  15.782093  -0.421 0.673516
## modelRenault Zoe                      13.748701  18.217906   0.755 0.450471
## modelTesla Model 3                    53.107347  19.331772   2.747 0.006030 **
## modelNissan Leaf (62 kWh)              0.007987  53.615774   0.000 0.999881
## modelKia Niro (EV)                   -49.965591  21.308532  -2.345 0.019067 *
## modelTesla Model S                    98.177435  13.782264   7.123 1.18e-12 ***
## modelTesla Model-X                    78.572046  22.369796   3.512 0.000447 ***
## modelKia Soul                        114.152473  16.332153   6.989 3.07e-12 ***
## modelMG ZS EV                        -25.499294  37.940987  -0.672 0.501561
## modelToyota Prius                    -62.528363  14.834752  -4.215 2.54e-05 ***
```

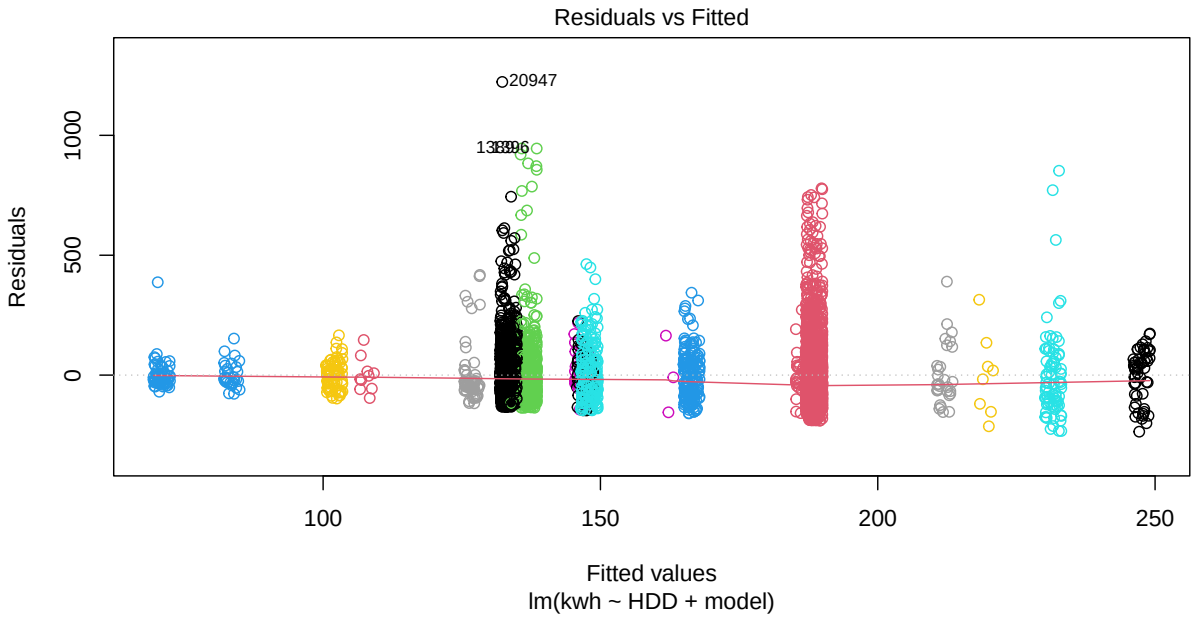
```
## modelMitsubishi Outlander            29.549372  69.170396   0.427 0.669252
## modelAudi A3 e-tron                  86.121902  42.401335   2.031 0.042289 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 119.7 on 5860 degrees of freedom
## Multiple R-squared:  0.05966,    Adjusted R-squared:  0.05661
## F-statistic: 19.57 on 19 and 5860 DF,  p-value: < 2.2e-16
```

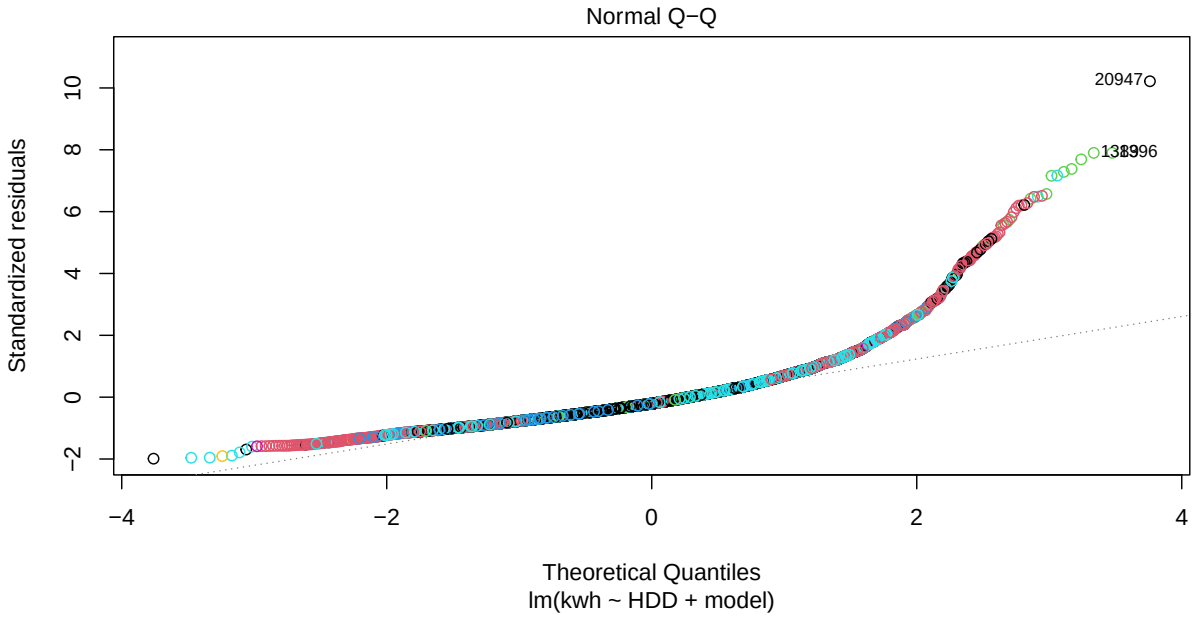
```
plot(eff_HDD_auckland$HDD,eff_HDD_auckland$kwh, col = eff_HDD_auckland$model)
abline(all_mdl_lm)
```

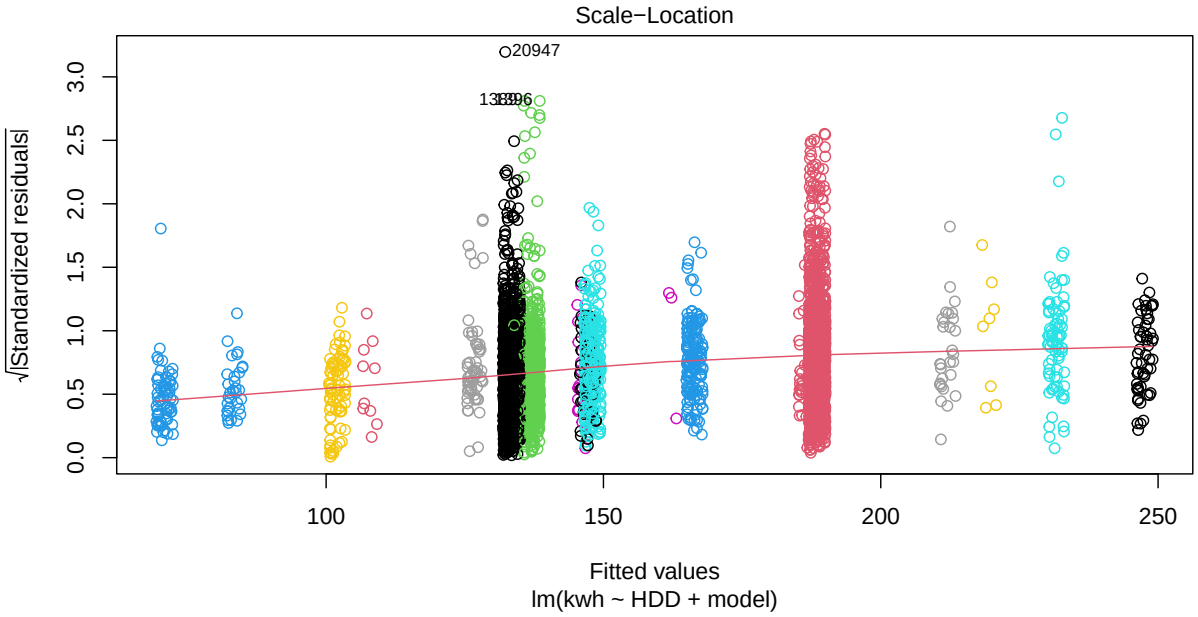
```
## Warning in abline(all_mdl_lm): only using the first two of 20 regression
## coefficients
```

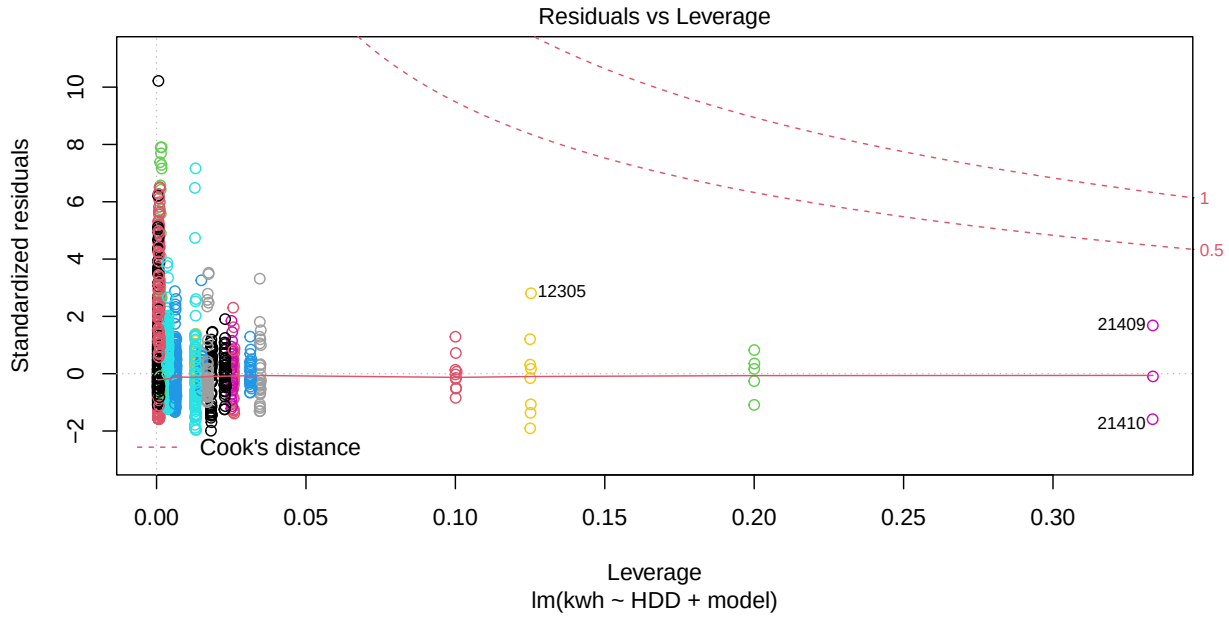



```
plot(all_mdl_lm, col = eff_HDD_auckland$model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(kwh ~ HDD + model)
20947


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(kwh ~ HDD + model)
20947





linear model to try to model kWh used by number of HDD in the month is effectively useless only thing that can be seen is the cars that are driven more or less each month

```
card_fuel = read.csv("Downloaded stats/fuel_usage_card_transactions_data.csv", header = T)
head(card_fuel)
```

```
##   year month fuel_purchased
## 1 2003     1          207.2
## 2 2003     2          206.2
## 3 2003     3          223.5
```

```
## 4 2003     4          198.3
## 5 2003     5          192.6
## 6 2003     6          180.9
```

```
fuel_cost = read.csv("Downloaded stats/MBIE_weekly_fuel_cost.csv")
fuel_cost$Week_ending_Friday = as.Date(fuel_cost$Week_ending_Friday)
fuel_cost$year = as.numeric(format(fuel_cost$Week_ending_Friday, '%Y'))
fuel_cost$month = as.numeric(format(fuel_cost$Week_ending_Friday, '%m'))
head(fuel_cost)
```

```
##   Week_ending_Friday Week_number Status Dubai_crude_USD.p.bbl NZ_US_exrate
## 1         2004-04-23     2004w17  Final                32.10        0.633
## 2         2004-04-30     2004w18  Final                32.65        0.626
## 3         2004-05-07     2004w19  Final                33.38        0.630
## 4         2004-05-14     2004w20  Final                34.41        0.609
## 5         2004-05-21     2004w21  Final                35.77        0.602
## 6         2004-05-28     2004w22  Final                35.03        0.618
##   Dubai_crude_NZD.p.bbl Diesel_importer_cost_NZc.p.l Diesel_ETS_NZc.p.l
## 1                 50.74                        43.4                  0
## 2                 52.16                        45.8                  0
## 3                 52.94                        47.8                  0
## 4                 56.55                        48.1                  0
## 5                 59.40                        46.5                  0
## 6                 56.68                        45.7                  0
##   Diesel_GST_NZc.p.l Diesel_taxes_NZc.p.l Diesel_price_excl_taxes_NZc.p.l
## 1                7.5                 7.9                           59.6
## 2                7.3                 7.7                           58.3
## 3                7.8                 8.2                           62.2
## 4                8.3                 8.7                           66.2
## 5                8.3                 8.6                           65.7
## 6                8.5                 8.9                           68.0
##   Diesel_main_port_price_NZc.p.l Diesel_discounted_retail_price_NZc.p.l
## 1                           67.5                                   67.5
## 2                           66.0                                   66.0
## 3                           70.3                                   70.3
## 4                           74.9                                   74.9
## 5                           74.3                                   74.3
## 6                           76.9                                   76.9
##   Diesel_importer_margin_NZc.p.l Diesel_margin_trend_NZc.p.l
## 1                           18.7                    18.33931
## 2                           15.9                    18.34124
## 3                           18.8                    18.34317
## 4                           20.5                    18.34510
## 5                           17.9                    18.34703
## 6                           19.9                    18.34897
##   Regular_Petrol_importer_cost_NZc.p.l Regular_Petrol_ETS_NZc.p.l
## 1                                 48.6                         0
## 2                                 52.2                         0
## 3                                 53.3                         0
## 4                                 56.4                         0
## 5                                 54.3                         0
## 6                                 53.7                         0
##   Regular_Petrol_GST_NZc.p.l Regular_Petrol_taxes_NZc.p.l
## 1                       12.7                         54.7
```

```
## 2                      12.6                       54.5
## 3                      13.0                       55.0
## 4                      13.5                       55.5
## 5                      13.5                       55.4
## 6                      13.8                       55.7
##   Regular_Petrol_price_excl_taxes_NZc.p.l
## 1                                   59.8
## 2                                   58.5
## 3                                   62.3
## 4                                   66.4
## 5                                   65.9
## 6                                   68.2
##   Regular_Petrol_main_port_price_NZc.p.l
## 1                                  114.5
## 2                                  113.0
## 3                                  117.3
## 4                                  121.9
## 5                                  121.3
## 6                                  123.9
##   Regular_Petrol_discounted_retail_price_NZc.p.l
## 1                                          114.5
## 2                                          113.0
## 3                                          117.3
## 4                                          121.9
## 5                                          121.3
## 6                                          123.9
##   Regular_Petrol_importer_margin_NZc.p.l Regular_Petrol_margin_trend_NZc.p.l
## 1                                   13.9                            13.80219
## 2                                   11.8                            13.78385
## 3                                   13.8                            13.76550
## 4                                   14.2                            13.74716
## 5                                   12.6                            13.72881
## 6                                   11.8                            13.71047
##   Premium_Petrol_95R_importer_cost_NZc.p.l Premium_Petrol_95R_ETS_NZc.p.l
## 1                                 50.54357                              0
## 2                                 52.86514                              0
## 3                                 55.80943                              0
## 4                                 57.75229                              0
## 5                                 60.77043                              0
## 6                                 57.88643                              0
##   Premium_Petrol_95R_GST_NZc.p.l Premium_Petrol_95R_taxes_NZc.p.l
## 1                       13.10000                         55.06500
## 2                       13.43333                         55.39833
## 3                       14.10000                         56.06500
## 4                       14.10000                         56.06500
## 5                       14.10000                         56.06500
## 6                       14.32222                         56.28722
##   Premium_Petrol_95R_price_excl_taxes_NZc.p.l
## 1                                   62.83500
## 2                                   65.50167
## 3                                   70.83500
## 4                                   70.83500
## 5                                   70.83500
## 6                                   72.61278
```

```
##   Premium_Petrol_95R_main_port_price_NZc.p.l
## 1                                      117.9
## 2                                      120.9
## 3                                      126.9
## 4                                      126.9
## 5                                      126.9
## 6                                      128.9
##   Premium_Petrol_95R_discounted_retail_price_NZc.p.l
## 1                                              117.9
## 2                                              120.9
## 3                                              126.9
## 4                                              126.9
## 5                                              126.9
## 6                                              128.9
##   Premium_Petrol_95R_importer_margin_NZc.p.l
## 1                                   12.29143
## 2                                   12.63652
## 3                                   15.02557
## 4                                   13.08271
## 5                                   10.06457
## 6                                   14.72635
##   Premium_Petrol_95R_margin_trend_NZc.p.l
## 1                                13.92260
## 2                                13.91374
## 3                                13.90487
## 4                                13.89600
## 5                                13.88714
## 6                                13.87827
##   Diesel_main_port_price_excl_taxes_NZc.p.l year month
## 1                                      59.6 2004     4
## 2                                      58.3 2004     4
## 3                                      62.1 2004     5
## 4                                      66.2 2004     5
## 5                                      65.7 2004     5
## 6                                      68.0 2004     5
```

```
month_fuel_cost = fuel_cost %>%
  group_by(year, month) %>%
  summarise(petrol = mean(Regular_Petrol_discounted_retail_price_NZc.p.l), diesel = mean(Diesel_discoun
            Premium = mean(Premium_Petrol_95R_discounted_retail_price_NZc.p.l))
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

```
head(month_fuel_cost)
```

```
## # A tibble: 6 x 5
## # Groups:   year [1]
##    year month petrol diesel Premium
##   <dbl> <dbl>  <dbl>  <dbl>   <dbl>
## 1  2004     4   114.   66.8    119.
## 2  2004     5   121.   74.1    127.
## 3  2004     6   120.   72.9    124.
## 4  2004     7   115.   71      120.
```

```
## 5  2004     8   122.   78.6    127.
## 6  2004     9   120.   78.2    125.
```

```
card_fuel = merge(card_fuel, month_fuel_cost, by = c("year", "month"), sort = FALSE)
card_fuel = card_fuel[card_fuel$year != 2004,]
head(card_fuel)
```

```
##    year month fuel_purchased  petrol diesel Premium
## 10 2005     1          279.9 114.875 76.475 121.400
## 11 2005     2          277.0 118.050 76.550 123.150
## 12 2005     3          321.5 121.525 83.250 127.025
## 13 2005     4          300.7 128.540 88.000 133.900
## 14 2005     5          290.2 123.900 82.425 128.150
## 15 2005     6          293.2 126.250 86.250 132.900
```

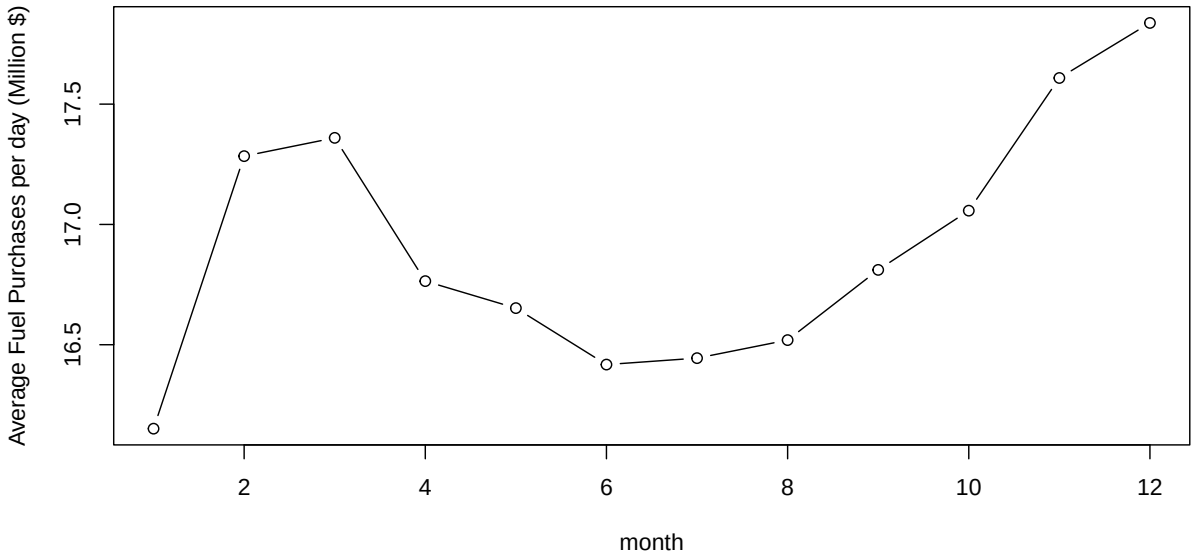
```
#plot(card_fuel$month[card_fuel$year == 2005],card_fuel$fuel_purchased[card_fuel$year == 2005],
#     type = 'b', col = 1 , ylim = c(min(card_fuel$fuel_purchased),max(card_fuel$fuel_purchased)))
#for (i in min(card_fuel$year):max(card_fuel$year)) {
#  points(card_fuel$month[card_fuel$year == i],card_fuel$fuel_purchased[card_fuel$year == i],
#     type = 'b', col = i+1-min(card_fuel$year) )
#
#}
```

effect of lockdown is very obvious so probably have to remove 2020-2021

```
mean_card_fuel = card_fuel[card_fuel$year <=2019,] %>%
  group_by(month) %>%
  summarise(avg_fuel_purchased = mean(fuel_purchased))
head(mean_card_fuel)
```

```
## # A tibble: 6 x 2
##   month avg_fuel_purchased
##   <int>              <dbl>
## 1     1               501.
## 2     2               484.
## 3     3               538.
## 4     4               503.
## 5     5               516.
## 6     6               493.
```

```
plot(mean_card_fuel$month, mean_card_fuel$avg_fuel_purchased/month_length, type = 'b', xlab= 'month', y
```

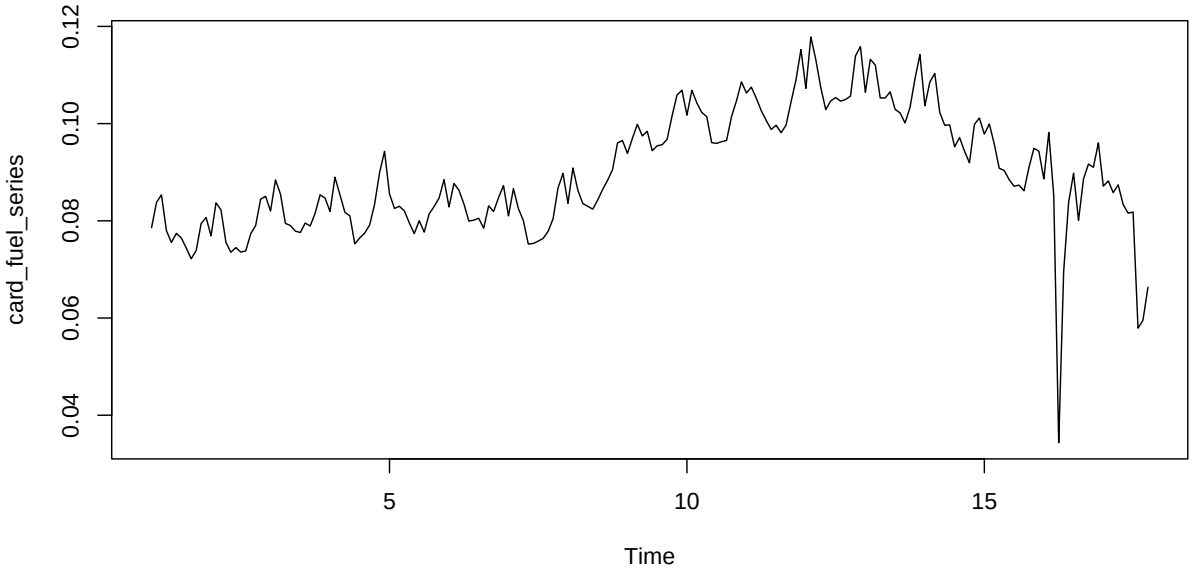


money spent shows a very clear upwards trend towards December (trips). with a valley in the winter. interesting that jan is lowest point. maybe new years resolutions to bike/walk or no school and not as many trips as dec?

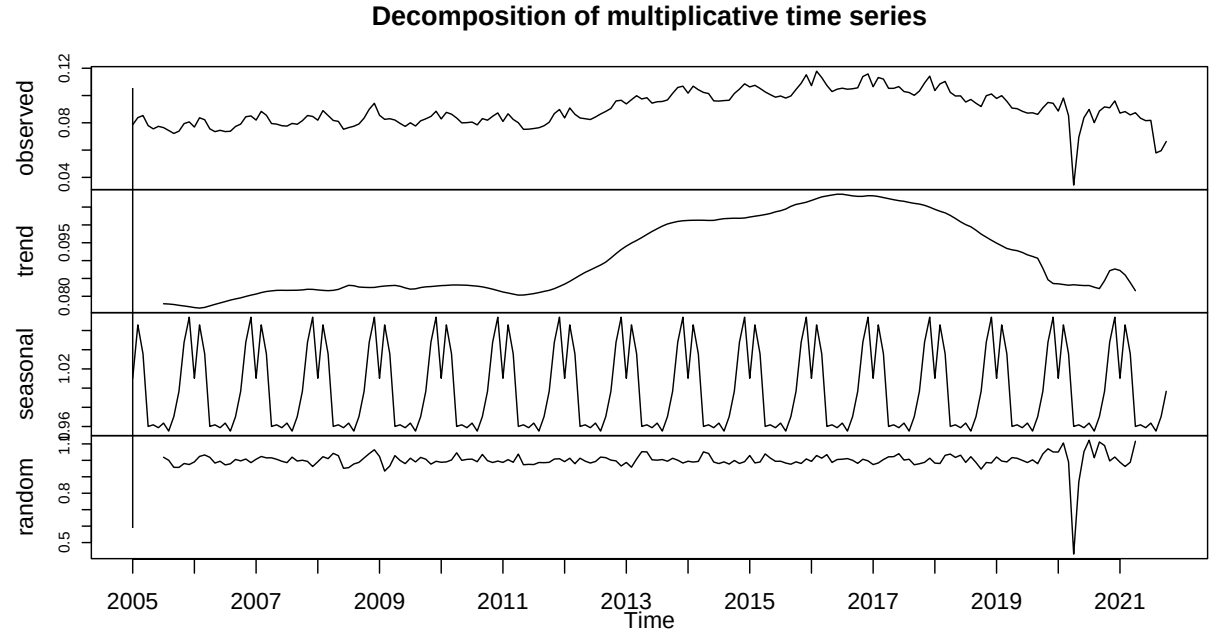
```
card_fuel_series = ts((card_fuel$fuel_purchased/card_fuel$petrol)/month_length, frequency = 12)
```

```
## Warning in (card_fuel$fuel_purchased/card_fuel$petrol)/month_length: longer
## object length is not a multiple of shorter object length
```

```
plot(card_fuel_series)
```



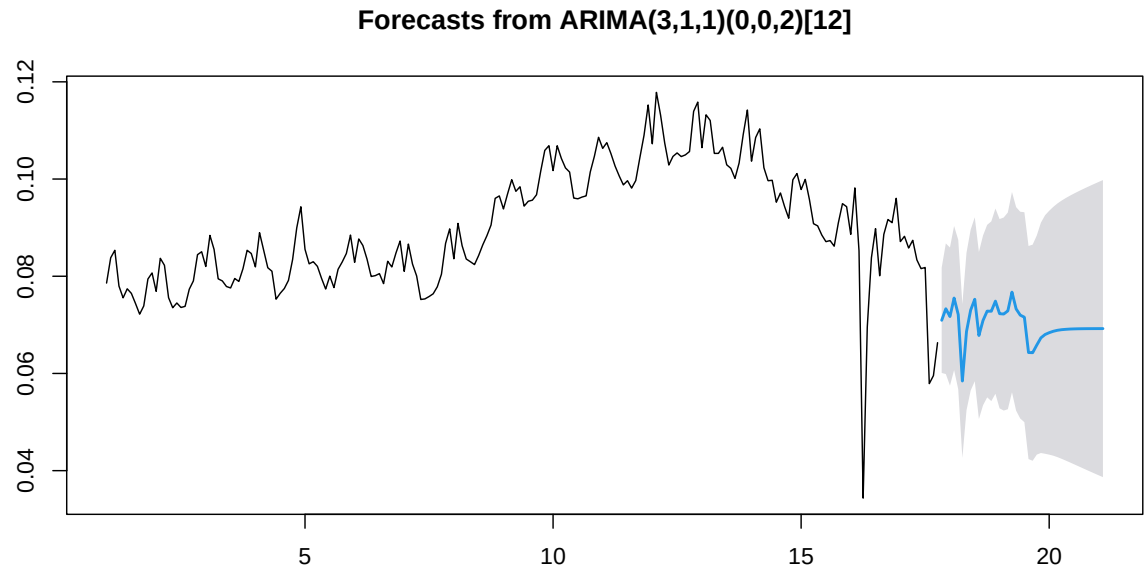

```
card_fuel_decomp = decompose(card_fuel_series, "multiplicative")
plot(card_fuel_decomp, xaxt = "n")
axis(1, labels = 2005:2021, at =1:17, outer = T, pos = 0.520)
abline(v = 1, )
```



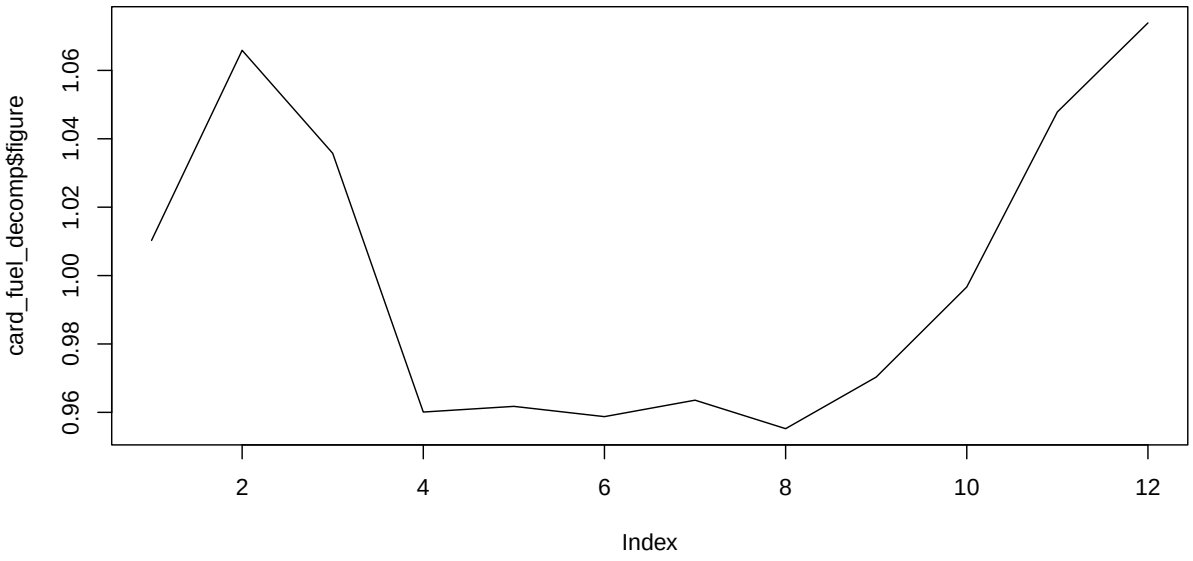

```
card_fuel_model = auto.arima(card_fuel_series) #uses AIC to get best model
summary(card_fuel_model)
```

```
## Series: card_fuel_series
## ARIMA(3,1,1)(0,0,2)[12]
##
## Coefficients:
##          ar1      ar2     ar3      ma1    sma1    sma2
##       0.6110  -0.1421  0.1089  -0.8782  0.1861  0.3320
## s.e.  0.0931   0.0830  0.0819   0.0583  0.0797  0.0986
##
## sigma^2 estimated as 3.056e-05:  log likelihood=760.91
## AIC=-1507.82   AICc=-1507.24   BIC=-1484.7
##
## Training set error measures:
##                         ME        RMSE         MAE        MPE    MAPE      MASE
## Training set -0.0001161465 0.005431042 0.003179015 -0.7363641 4.08468 0.5848447
##                      ACF1
## Training set 0.004041763
```

```
plot(forecast(card_fuel_model, level = 95, h = 40))
```




```
plot(card_fuel_decomp$figure, type = 'l')
```


card_fuel_decomp$figure
0.96
0.98
1.00
1.02
1.04
1.06
2
4
6
8
10
12
Index