Statistical learning og programmering

1. Semesterprojekt. Antal ord:

Af Kenneth Gottfredsen, Eva Rauff og Sanne Sørensen

2022-12-23

Indhold

I første omgang indlæses `pacman::load`:

```
# Inden vi går i gang bruger vi pacman() til at installere og indhente relevante
# pakker på samme tid.
pacman::p_load("tidyverse", "magrittr", "nycflights13", "gapminder",
               "Lahman", "maps", "lubridate", "pryr", "hms", "hexbin",
               "feather", "htmlwidgets", "broom", "pander", "modelr",
               "XML", "httr", "jsonlite", "lubridate", "microbenchmark",
               "splines", "ISLR2", "MASS", "testthat", "leaps", "caret",
               "RSQLite", "class", "babynames", "nasaweather",
               "fueleconomy", "viridis", "readxl", "timeDate", "tinytex",
               "ggbeeswarm", "palmerpenguins", "hms", "RColorBrewer", "boot",
               "openxlsx", "writexl", "PerformanceAnalytics", "car")
```

Import

I første omgang vil vi importere det datasæt vi har fået udleveret til eksamen:

```
# Indlæser datasæt og gemmer det nye datasæt i et objekt.
data1 <- read_excel("data/stud_exam_data.xlsx")
 # Dernæst undersøges strukturen i datasættet.
str(data1)
```

```
## tibble [152 x 4] (S3: tbl_df/tbl/data.frame)
##  $ date             : POSIXct[1:152], format: "2022-04-01" "2022-04-02" ...
##  $ efterspørgsel    : num [1:152] 367 361 376 47 367 402 416 355 283 454 ...
##  $ kammerjunkere    : chr [1:152] "0" "0" "0" "1" ...
##  $ forventet_l_lager: chr [1:152] "3" "3" "3" "3" ...
```

Nu har vi fået indlæst datasættet. Det næste skridt er at transformere de forskellige variabler:

I følgende kode-chunk vil vi rekode og transformere de udvalgte variabler så de stemmer overens med eksamensbesvarelsen. Hele kodestumpen vil blive kædet sammen med 'pipe' funktionen' fra dplyr pakken. Omkodningerne bliver til sidst gemt i en ny dataframe som vi kalder data1.

Derefter bruger vi mutate() til at lave en ny kolonne ud fra data1. Først laver vi en date-variabel, som vi koder til et date objekt med ymd() funktionen fra lubridate pakken.

I den næste del anvendes mutate() til, at lave en kolonne der hedder dag, som bliver omkodet til en faktor. Dernæst koder vi date til et objekt med ymd() funktionen fra lubridate pakken. "lubridate.week.start",1=mandag, istedet for søndag som er standardindstillingerne i R.

Dernæst bruger vi mutate() igen til at danne en ny weekend-variabel der hedder weekend_1. I denne sammenhæng vælger vi at fredag, lørdag, søndag og fire andre helligdage er 1, ellers er de andre værdier 0. Dette kaldes for en dummyvariabel.

Måned, dag, kamjunk, forvent_lager og weekend_1 er alle kategoriske faktorer. For at gøre det nemmere at forstå hvad de forskellige værdier udtrykker, navngiver vi disse med fct_recode funktionen.

```
data1 <- data1 %>%
  mutate(date = ymd(date), måned = factor(month(date)),
         kamjunk = factor(kammerjunkere), forvent_lager =
           factor(forventet_l_lager)) %>%
  mutate(dag = as.factor(wday(date, week_start =
                                getOption("lubridate.week.start", 1)))) %>%
  mutate(weekend_1 = as.integer(dag %in% c("5", "6", "7")| date %in%
                                  ymd("2022-04-14", "2022-04-18", "2022-05-26",
                                      "2022-06-06"))) %>%
  mutate(weekend = factor(weekend_1)) %>%
  mutate(data1, kamjunk = fct_recode(kammerjunkere, "ja" = "0",
                                     "nej" = "1")) %>%
  mutate(data1, forvent_lager = fct_recode(forventet_l_lager, "lav" = "1",
                                           "mellem" = "2", "høj" = "3")) %>%
  mutate(data1, måned = fct_recode(måned, "april" = "4", "maj" = "5",
                                   "juni" = "6", "juli" = "7",
                                   "august" = "8")) %>%
  mutate(data1, dag = fct_recode(dag, "mandag" = "1", "tirsdag" = "2",
                                 "onsdag" = "3", "torsdag" = "4",
                                 "fredag" = "5", "lørdag" = "6",
                                 "søndag" = "7")) %>%
  dplyr::select(date, måned, dag, efterspørgsel, kamjunk, forvent_lager,
                weekend_helligdag = weekend)
```

I denne kodechunk vil vi lave en HTTP GET-anmodning til en API fra DMI. Vi skal bruge adgangen til at få de relevante vejr-variable som vi senere skal bruge i vores analyse. API'en leverer til slut et objekt i JSON format som bliver transformeret om til en dataframe i stedet for en liste.

Først bruger vi base_url og info_url til at anmode om vejrdata fra DMI's API. req_url bruges til at udvælge specifikke parametre fra API´en.

I denne kodechunk vil vi transformere den data vi har hentet fra vores API-kald til nogle mere brugbare data.

Først bruger vi base_url og info_url til at anmode om vejrdata fra DMI's API. req_url bruges til at udvælge specifikke parametre fra API´en.

Derefter bruger vi pivot_wider-funktionen til at sprede variablerne ud i separate kolonner.

Vi bruger derefter Mutate-funktionen til at konvertere kolonnen 'målingstidspunkt' til en datoformat Separate-funktionen bruges til at opdele kolonnen 'målingstidspunkt' i to separate kolonner som vi navngiver 'date' og 'time'.

Filter-funktionen udvælger rækker, der indeholder de første fire characters: "12:0".

```
data2 <- as.data.frame(do.call(cbind, list_dmi))
data2 <- dplyr::select(data2, features.properties.observed,
                       features.properties.value,
                       features.properties.parameterId) %>%
  rename(værdi = features.properties.value, parameter =
           features.properties.parameterId,
         målingstidspunkt = features.properties.observed) %>%
  pivot_wider(names_from = parameter, values_from = værdi) %>%
  mutate(målingstidspunkt = as_datetime(målingstidspunkt)) %>%
  separate(målingstidspunkt, into = c('date', 'time'), sep = " ") %>%
  filter(str_sub(time, 1, 4) == "12:0") %>%
  mutate(date = as_date(date)) %>%
  mutate(time = as_hms(time)) %>%
  dplyr::select(-(temp_max_past12h:temp_min_past12h))
```

I nedestående kode-chunk merger vi data1 og data2 til data3 for at beholde alle observationer i x.

Vi bruger derefter mutate-funktionen til at oprette fire nye variabler i data3 kaldet temp_gt25_3_dage. Lag-funktionen er brugt til at lave variablerne, som har opfanget forsinkede værdier fra temp1, temp2 og temp3. Afslutningsvis dannes variablen 'temp_gt25_3_dage', som måler de dage hvor der har været mere end 3 dage i træk med >= 25 grader. Det er en dummyvariabel fordi vi bruger if_else.

```
data3 <- data1 %>%
left_join(data2, data1, by = c("date" = "date"))
dplyr::select(data3, date, time, weekend_helligdag, everything())
```

```
data3 <- data3 %>%
  mutate(temp1 = lag(temp_max_past1h, 1),
         temp2 = lag(temp_max_past1h, 2),
         temp3 = lag(temp_max_past1h, 3),
         temp1 = if_else(is.na(temp1), 0, temp1),
         temp2 = if_else(is.na(temp2), 0, temp2),
         temp3 = if_else(is.na(temp3), 0, temp3),
         temp_gt25_3_dage = if_else(temp1 >= 25 & temp2 >= 25 & temp3 >= 25,
                                    1, 0))
```

Først identificerer vi outliers i vores dataset data3. Derefter fjerner vi 1 outlier som er 47.

Derefter laver vi en ggplot for at se fordelingen af efterspørgselen af koldskål i form af en histogram.

Vi bruger geom_density til at forstå fordelingen og til at forudsige fordelingen af koldskål i en anden undersøgelse.

Man kan se at fordelingen er størst omkring 500 liter koldskål.

```
boxplot.stats(data3$efterspørgsel)$out
```

```
## [1] 47
```

```
data3 <- data3 %>%
  filter(efterspørgsel > 47)


ggplot(data3, aes(x = efterspørgsel)) +
 geom_histogram(aes(y =..density..), colour = "black",
                fill = "gray", binwidth = 60) +
 geom_density(alpha=0.5, fill="#FF6666", adjust=1.5) +
```

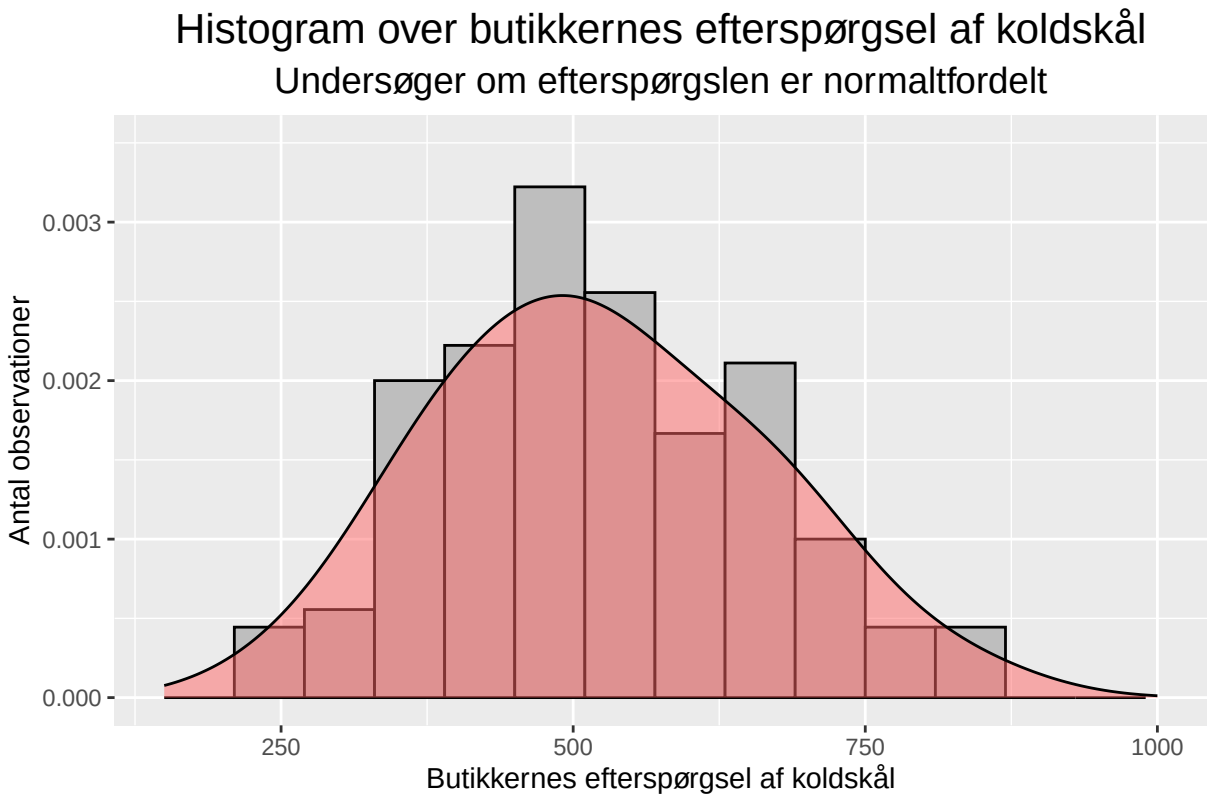
```
  labs(title = "Histogram over butikkernes efterspørgsel af koldskål",
       subtitle = "Undersøger om efterspørgslen er normaltfordelt",
       y = "Antal observationer",
       x = "Butikkernes efterspørgsel af koldskål",
       caption = "Kilde: Thise Mejeri 2022") +
  ggeasy::easy_center_title() + # Centrerer titlen.
  theme( plot.title = element_text(hjust = 0.5, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 1, face = "italic", size = 10)) +
  xlim(150, 1000) + ylim(0, 0.0035)
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_bin()`).
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_density()`).
```

```
## Warning: Removed 1 rows containing missing values (`geom_bar()`).
```

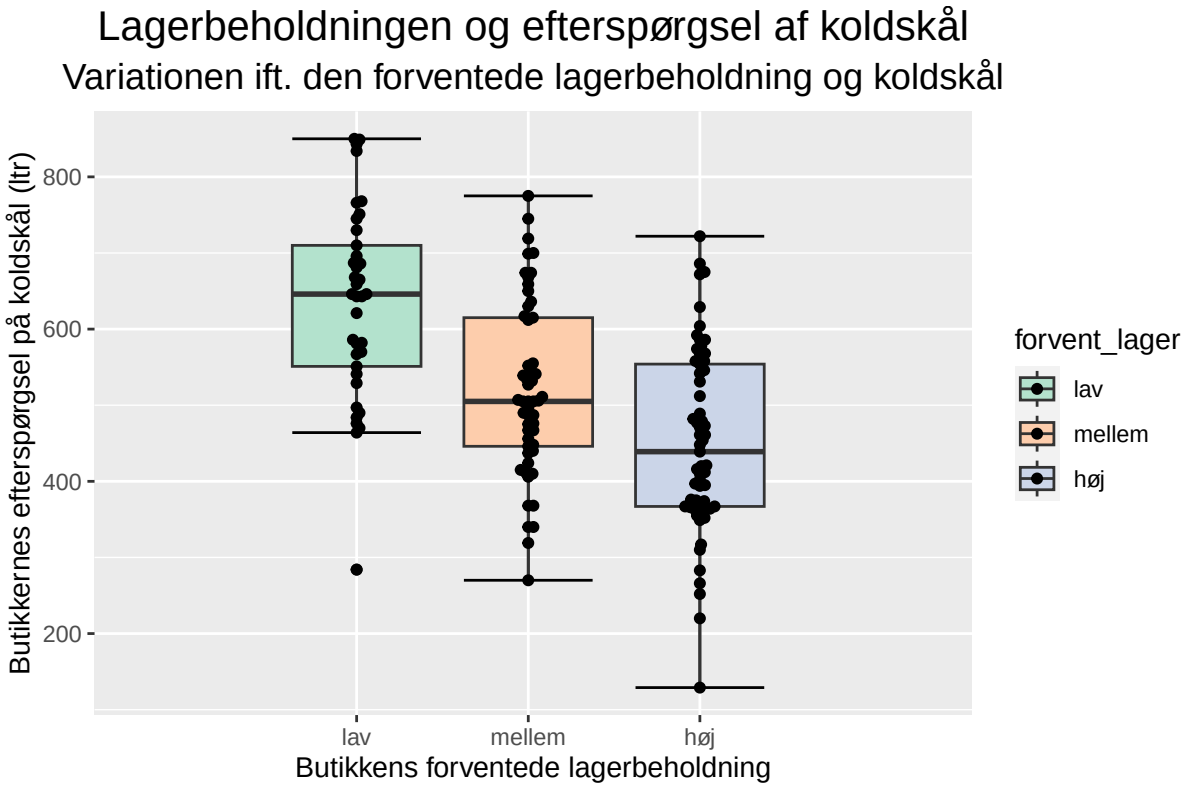


På følgende kode-chunk har vi lavet et boxplot til at vise den statistiske variationen af lagerbeholdningen og efterspørgsel.

Her kan man se at median-efterspørgslen stiger når man går fra høj til lav forventet lagerbeholdning af koldskål. Dette tyder også på at der er en sammenhæng mellem de 2 variabler.

```
ggplot(data = data3, mapping = aes(x = forvent_lager, y = efterspørgsel, fill =
                                     forvent_lager)) +
  stat_boxplot(geom = 'errorbar') + # whiskers.
  geom_boxplot() +
  labs(title = "Lagerbeholdningen og efterspørgsel af koldskål",
       subtitle = "Variationen ift. den forventede lagerbeholdning og koldskål",
       caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22",
       y = "Butikkernes efterspørgsel på koldskål (ltr)",
       x = "Butikkens forventede lagerbeholdning") + # Undgår overplotting
  geom_beeswarm(dodge.width=3, cex =1, color = "black") + # Justerer boksbredden
  ggeasy::easy_center_title() + # Centrerer titlen.
  theme( plot.title = element_text(hjust = 0.5, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 2.3, face = "italic",size = 10 )) +
  scale_fill_brewer(palette = "Pastel2")
```

```
## Warning: `position_dodge()` requires non-overlapping x intervals
```



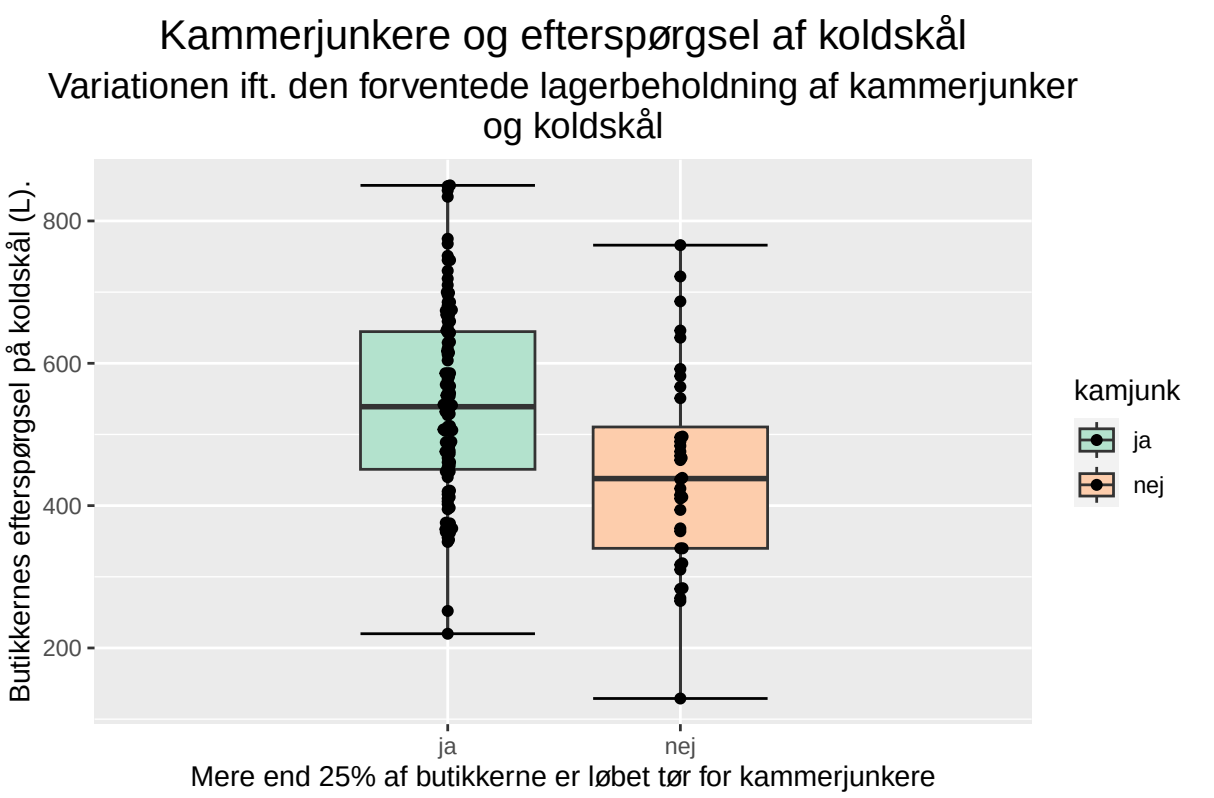
Kilde: Tal fra DMI 2002. Fra perioden 1/4/2

I næste kode-chunk har vi lavet et boxplot som viser fordelingen af efterspørgslen i forhold til om 25% af butikkerne er løbet tør for kammerjunkere eller ej. På baggrund af plottet kan vi se at hvis butikkerne ikke har kammerjunkere på lageret så falder efterspørgslen. Konklusionen er at efterspørgslen på koldskål stiger når de er løbet tør for kammerjunkere.

```
ggplot(data = data3, mapping = aes(x = kamjunk, y = efterspørgsel, fill =
                                     kamjunk)) +
  stat_boxplot(geom = 'errorbar') +
  geom_boxplot() +
  labs(title = "Kammerjunkere og efterspørgsel af koldskål",
  subtitle = "Variationen ift. den forventede lagerbeholdning af kammerjunker
  og koldskål",
  caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22",
  y = "Butikkernes efterspørgsel på koldskål (L).",
  x = "Mere end 25% af butikkerne er løbet tør for kammerjunkere") +
  ggeasy::easy_center_title() + # Centrerer titlen.
  geom_beeswarm(dodge.width=3,cex=0.5, color = "black") + # Justerer boksbredden.
  theme( plot.title = element_text(hjust = 0.5, size = 16),
```

```
        plot.subtitle = element_text(hjust = 0.5, size = 14),
        plot.caption = element_text(hjust = 1.8, face = "italic",
                                    size = 10 )) +
  scale_fill_brewer(palette = "Pastel2")
```

```
## Warning: `position_dodge()` requires non-overlapping x intervals
```

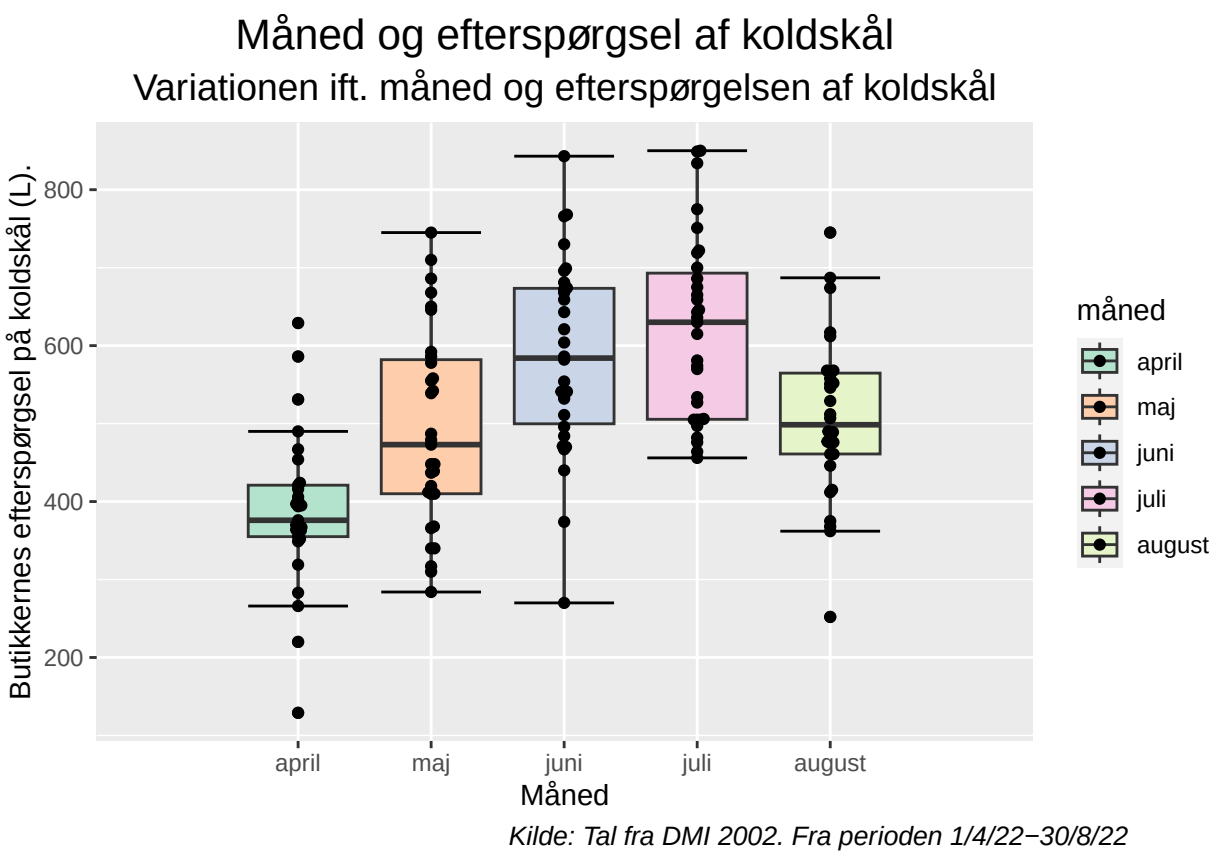


Forneden har vi lavet et boxplot som viser sammenhængen mellem måned og efterspørgslen af koldskål. Det er tydeligt at se at efterspørgslen stiger fra april-juli og derefter falder efterspørgslen i august. Hvilket kan tyde på at efterspørgselen af koldskål hænger sammen med sommerperioden.

```
ggplot(data = data3, mapping = aes(x = måned, y = efterspørgsel,
                                   fill = måned)) +
  stat_boxplot(geom = 'errorbar') +
  geom_boxplot() +
  labs(title = "Måned og efterspørgsel af koldskål",
       subtitle = "Variationen ift. måned og efterspørgelsen af koldskål",
```

```
       caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22",
       y = "Butikkernes efterspørgsel på koldskål (L).",
       x = "Måned") +
  ggeasy::easy_center_title() + # Centrerer titlen.
  geom_beeswarm(dodge.width=3,cex=0.5, color = "black") + # Justerer boksbredden.
  theme( plot.title = element_text(hjust = 0.5, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 1.3, face =
                                       "italic", size = 10 )) +
  scale_fill_brewer(palette = "Pastel2")
```

```
## Warning: `position_dodge()` requires non-overlapping x intervals
```




```
attach(data3)
model1 <- lm(efterspørgsel ~ temp_mean_past1h, data = data3)
predict(model1,data.frame(efterspørgsel=(c(520, 700))), interval = "prediction", leve
```

```
## Warning: 'newdata' had 2 rows but variables found have 151 rows
```

```
##          fit      lwr      upr
## 1   392.3940 130.5485 654.2395
## 2   389.5557 127.4929 651.6185
## 3   416.9928 156.8365 677.1491
## 4   412.2623 151.8083 672.7163
## 5   420.7773 160.8497 680.7048
## 6   411.3162 150.8011 671.8313
## 7   407.5317 146.7670 668.2964
## 8   427.4000 167.8528 686.9472
## 9   429.2922 169.8489 688.7355
## 10  434.0228 174.8301 693.2154
## 11  442.5378 183.7631 701.3124
## 12  451.0527 192.6532 709.4523
## 13  475.6515 218.0933 733.2098
## 14  412.2623 151.8083 672.7163
## 15  463.3521 205.4184 721.2859
## 16  463.3521 205.4184 721.2859
## 17  472.8132 215.1763 730.4501
## 18  480.3821 222.9441 737.8201
## 19  489.8432 232.6055 747.0809
## 20  504.9809 247.9516 762.0103
## 21  508.7653 251.7665 765.7642
## 22  478.4899 221.0054 735.9744
## 23  456.7294 198.5560 714.9028
## 24  455.7833 197.5735 713.9930
## 25  478.4899 221.0054 735.9744
## 26  471.8671 214.2029 729.5313
## 27  475.6515 218.0933 733.2098
## 28  477.5438 220.0352 735.0523
## 29  482.2743 224.8807 739.6679
## 30  487.0049 229.7127 744.2970
## 31  485.1126 227.7815 742.4438
```

```
## 32  478.4899 221.0054 735.9744
## 33  473.7593 216.1492 731.3695
## 34  492.6815 235.4934 749.8696
## 35  497.4120 240.2958 754.5283
## 36  504.0348 246.9965 761.0731
## 37  501.1965 244.1280 758.2650
## 38  492.6815 235.4934 749.8696
## 39  511.6037 254.6220 768.5853
## 40  484.1665 226.8151 741.5179
## 41  491.7354 234.5313 748.9395
## 42  473.7593 216.1492 731.3695
## 43  490.7893 233.5686 748.0099
## 44  505.9270 248.9061 762.9479
## 45  523.9031 266.9398 780.8664
## 46  518.2264 261.2660 775.1869
## 47  499.3043 242.2130 756.3956
## 48  539.9869 282.9099 797.0640
## 49  503.0887 246.0409 760.1365
## 50  476.5977 219.0645 734.1308
## 51  491.7354 234.5313 748.9395
## 52  518.2264 261.2660 775.1869
## 53  489.8432 232.6055 747.0809
## 54  494.5737 237.4160 751.7315
## 55  511.6037 254.6220 768.5853
## 56  463.3521 205.4184 721.2859
## 57  489.8432 232.6055 747.0809
## 58  482.2743 224.8807 739.6679
## 59  487.0049 229.7127 744.2970
## 60  473.7593 216.1492 731.3695
## 61  497.4120 240.2958 754.5283
## 62  473.7593 216.1492 731.3695
## 63  474.7054 217.1215 732.2894
```

```
## 64  511.6037 254.6220 768.5853
## 65  547.5558 290.3712 804.7404
## 66  482.2743 224.8807 739.6679
## 67  509.7115 252.7189 766.7040
## 68  503.0887 246.0409 760.1365
## 69  542.8253 285.7119 799.9386
## 70  534.3103 277.2912 791.3294
## 71  539.9869 282.9099 797.0640
## 72  529.5797 272.5941 786.5654
## 73  517.2803 260.3185 774.2422
## 74  502.1426 245.0847 759.2005
## 75  527.6875 270.7115 784.6635
## 76  534.3103 277.2912 791.3294
## 77  555.1247 297.7980 812.4513
## 78  557.9630 300.5742 815.3518
## 79  459.5677 201.5002 717.6352
## 80  483.2204 225.8482 740.5927
## 81  543.7714 286.6448 800.8979
## 82  511.6037 254.6220 768.5853
## 83  556.0708 298.7240 813.4176
## 84  601.4840 342.5392 860.4288
## 85  571.2085 313.4658 828.9512
## 86  602.4301 343.4390 861.4212
## 87  610.9451 351.5133 870.3768
## 88  549.4480 292.2311 806.6649
## 89  574.0468 316.2147 831.8790
## 90  592.9690 334.4173 851.5207
## 91  552.2863 295.0170 809.5557
## 92  554.1786 296.8716 811.4855
## 93  571.2085 313.4658 828.9512
## 94  545.6636 288.5091 802.8181
## 95  533.3642 276.3528 790.3755
```

```
## 96  517.2803 260.3185 774.2422
## 97  518.2264 261.2660 775.1869
## 98  515.3881 258.4218 772.3544
## 99  518.2264 261.2660 775.1869
## 100 546.6097 289.4404 803.7790
## 101 558.9091 301.4986 816.3196
## 102 576.8852 318.9587 834.8117
## 103 591.0768 332.6067 849.5469
## 104 535.2564 278.2290 792.2838
## 105 522.9570 265.9955 779.9184
## 106 526.7414 269.7694 783.7134
## 107 540.9330 283.8444 798.0217
## 108 540.9330 283.8444 798.0217
## 109 608.1067 348.8266 867.3869
## 110 637.4361 376.3626 898.5096
## 111 603.3762 344.3382 862.4142
## 112 532.4181 275.4140 789.4222
## 113 505.9270 248.9061 762.9479
## 114 542.8253 285.7119 799.9386
## 115 606.2145 347.0329 865.3962
## 116 528.6336 271.6531 785.6142
## 117 511.6037 254.6220 768.5853
## 118 530.5258 273.5346 787.5171
## 119 547.5558 290.3712 804.7404
## 120 581.6157 323.5213 839.7101
## 121 560.8013 303.3456 818.2571
## 122 518.2264 261.2660 775.1869
## 123 540.9330 283.8444 798.0217
## 124 602.4301 343.4390 861.4212
## 125 627.9750 367.5347 888.4154
## 126 523.9031 266.9398 780.8664
## 127 527.6875 270.7115 784.6635
```

```
## 128 496.4659 239.3364 753.5955
## 129 527.6875 270.7115 784.6635
## 130 553.2324 295.9446 810.5203
## 131 570.2624 312.5485 827.9763
## 132 592.9690 334.4173 851.5207
## 133 613.7834 354.1953 873.3715
## 134 616.6217 356.8726 876.3708
## 135 618.5139 358.6548 878.3730
## 136 621.3523 361.3243 881.3803
## 137 577.8313 319.8723 835.7903
## 138 606.2145 347.0329 865.3962
## 139 614.7295 355.0883 874.3707
## 140 572.1546 314.3827 829.9266
## 141 569.3163 311.6306 827.0020
## 142 574.0468 316.2147 831.8790
## 143 549.4480 292.2311 806.6649
## 144 576.8852 318.9587 834.8117
## 145 583.5079 325.3426 841.6732
## 146 581.6157 323.5213 839.7101
## 147 592.0229 333.5123 850.5335
## 148 535.2564 278.2290 792.2838
## 149 539.9869 282.9099 797.0640
## 150 510.6576 253.6707 767.6444
## 151 539.9869 282.9099 797.0640
```

```
prædiktion <- predict(model1, interval = "prediction", level = 0.95)
```

```
## Warning in predict.lm(model1, interval = "prediction", level = 0.95): predictions
```

```
new_df <- cbind(data3, prædiktion)
ggplot(new_df, aes(temp_mean_past1h, efterspørgsel))+
    geom_point() +
    geom_line(aes(y=lwr), color = "red", linetype = "dashed")+
```

```
    geom_line(aes(y=upr), color = "red", linetype = "dashed")+
    geom_smooth(method=lm, se=TRUE) +
    labs(title = "Sammenhængen mellem temperatur og efterspørgsel af koldskål",
    subtitle = "Linelær regression der viser relationen mellem den forventede lagerbe
    caption = "Kilde: tal fra DMI 2002 fra perioden 1/4/22-30/8/22",
    y = "Butikkernes efterspørgsel på koldskål (ltr.)",
    x = "Maks temperatur hver time (celcius)") +
ggeasy::easy_center_title() + # Centrerer titlen.
theme( plot.title = element_text(hjust = 0.5, size = 16),
plot.subtitle = element_text(hjust = 0.5, size = 14),
plot.caption = element_text(hjust = 1, face = "italic", size = 10 ))+
xlim(5, 30.70) + ylim(220, 900)
```

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

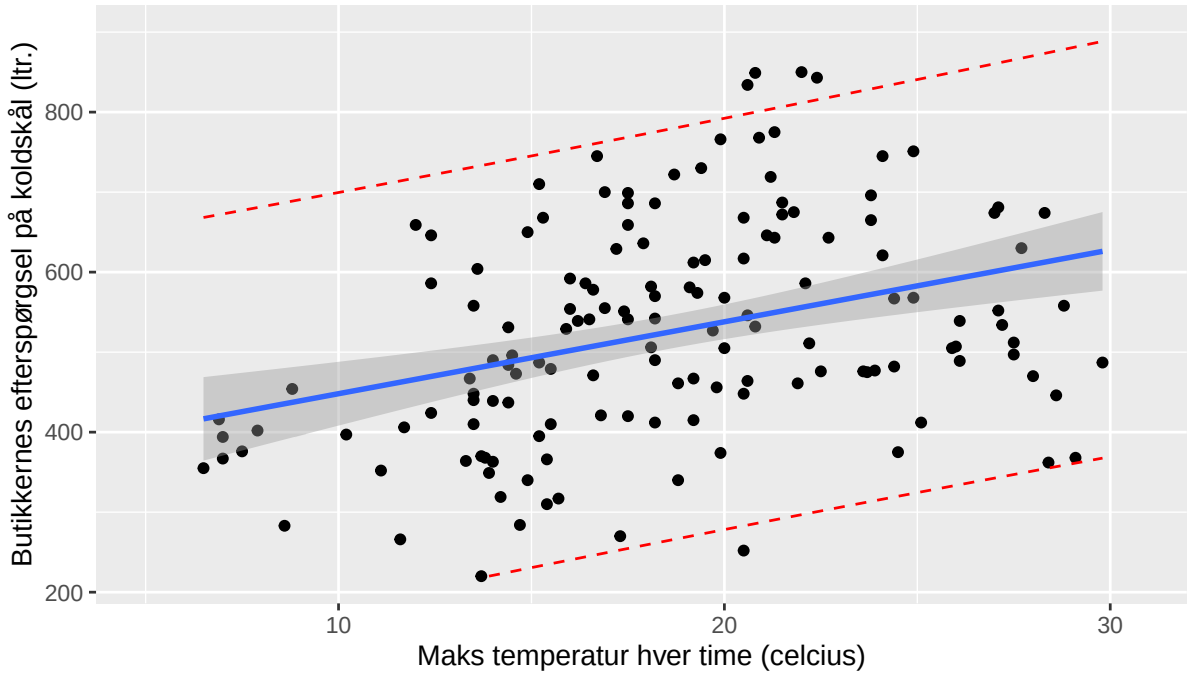
```
## Warning: Removed 4 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 30 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_line()`).
```

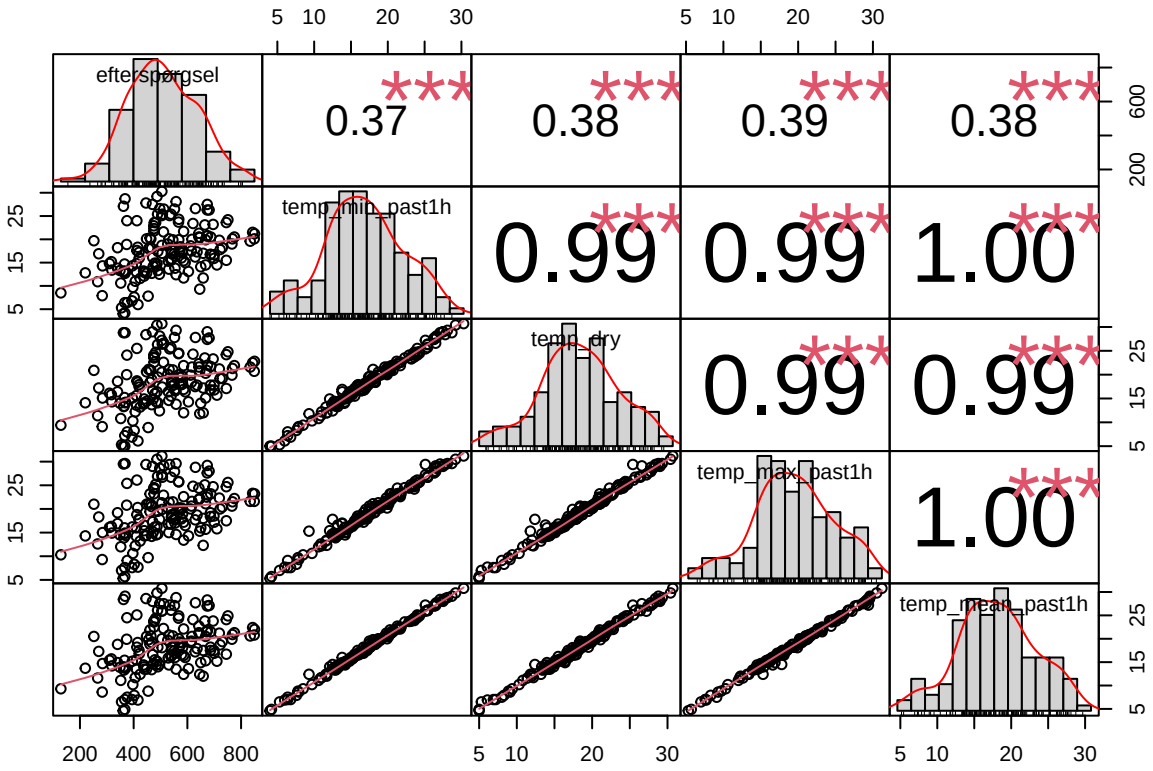
Sammenhængen mellem temperatur og efterspørgsel af koldsk
ær regression der viser relationen mellem den forventede lagerbeholdning o



Kilde: tal fra DMI 2002 fra perioden 1/4/22−30/8/22

I dette afsnit vil vi gå i gang med analysen.

```
# Tester for samvariation og multikolinearitet på de kontinuerte variabler.
cor_matrice <- data3 |>
  dplyr::select(efterspørgsel, temp_min_past1h,
                temp_dry, temp_max_past1h,
chart.Correlation(cor_matrice, histogram = TRUE, method = "pearson")
```




```
# Der er stærk multikolinearitet, det kan være et problem ift. tolkningen af
# vores multible regressionsmodel. Dette har også en negativ indvirkning på
# modellens pålidelighed.
```

```
glimpse(data3)
```

```
## Rows: 151
## Columns: 19
## $ date               <dttm> 2022-04-01, 2022-04-02, 2022-04-03, 2022-04-05, 202~
## $ måned              <fct> april, april, april, april, april, april, april, apr~
## $ dag                <fct> fredag, lørdag, søndag, tirsdag, onsdag, torsdag, fr~
## $ efterspørgsel      <dbl> 367, 361, 376, 367, 402, 416, 355, 283, 454, 129, 39~
## $ kamjunk            <fct> ja, ja, ja, ja, ja, ja, ja, nej, ja, nej, ja, ja, ja~
## $ forvent_lager      <fct> høj, høj, høj, høj, høj, høj, høj, høj, høj, høj, hø~
## $ weekend_helligdag <fct> 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0~
## $ time               <time> 12:00:00, 12:00:00, 12:00:00, 12:00:00, 12:00:00, 1~
```

```
## $ temp_min_past1h   <dbl> 4.2, 4.0, 6.4, 6.4, 7.6, 6.0, 5.3, 7.9, 7.8, 8.5, 9.~
## $ humidity          <dbl> 36, 36, 38, 44, 92, 94, 66, 44, 44, 47, 38, 53, 72, ~
## $ temp_dry          <dbl> 5.0, 5.0, 8.0, 7.2, 8.3, 6.1, 5.3, 9.1, 8.9, 9.4, 10~
## $ temp_dew          <dbl> -8.8, -8.9, -5.4, -4.2, 7.2, 5.2, -0.6, -2.4, -2.6, ~
## $ temp_max_past1h   <dbl> 5.7, 5.3, 9.1, 7.6, 8.3, 7.7, 7.0, 9.3, 9.7, 10.3, 1~
## $ humidity_past1h   <dbl> 38, 37, 41, 45, 94, 91, 59, 45, 47, 49, 37, 52, 74, ~
## $ temp_mean_past1h  <dbl> 4.9, 4.6, 7.5, 7.0, 7.9, 6.9, 6.5, 8.6, 8.8, 9.3, 10~
## $ temp1             <dbl> 0.0, 5.7, 5.3, 4.0, 7.6, 8.3, 7.7, 7.0, 9.3, 9.7, 10~
## $ temp2             <dbl> 0.0, 0.0, 5.7, 9.1, 4.0, 7.6, 8.3, 7.7, 7.0, 9.3, 9.~
## $ temp3             <dbl> 0.0, 0.0, 0.0, 5.3, 9.1, 4.0, 7.6, 8.3, 7.7, 7.0, 9.~
## $ temp_gt25_3_dage  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

```
attach(data3)
```

```
## De følgende objekter er maskerede fra data3 (pos = 3):
##
##     dag, date, efterspørgsel, forvent_lager, humidity, humidity_past1h,
##     kamjunk, måned, temp_dew, temp_dry, temp_gt25_3_dage,
##     temp_max_past1h, temp_mean_past1h, temp_min_past1h, temp1, temp2,
##     temp3, time, weekend_helligdag
```

```
lm.fit1 = lm(efterspørgsel ~ forvent_lager + weekend_helligdag + kamjunk + temp_gt25_
vif(lm.fit1) # VIF > 1 indikerer at der er inflation i variansen på alle variabler.
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## forvent_lager     1.403145  2        1.088368
## weekend_helligdag 1.153516  1        1.074018
## kamjunk           1.131247  1        1.063601
## temp_gt25_3_dage  1.243248  1        1.115010
## temp_mean_past1h  1.486803  1        1.219345
```

```
vif(lm.fit1)
```

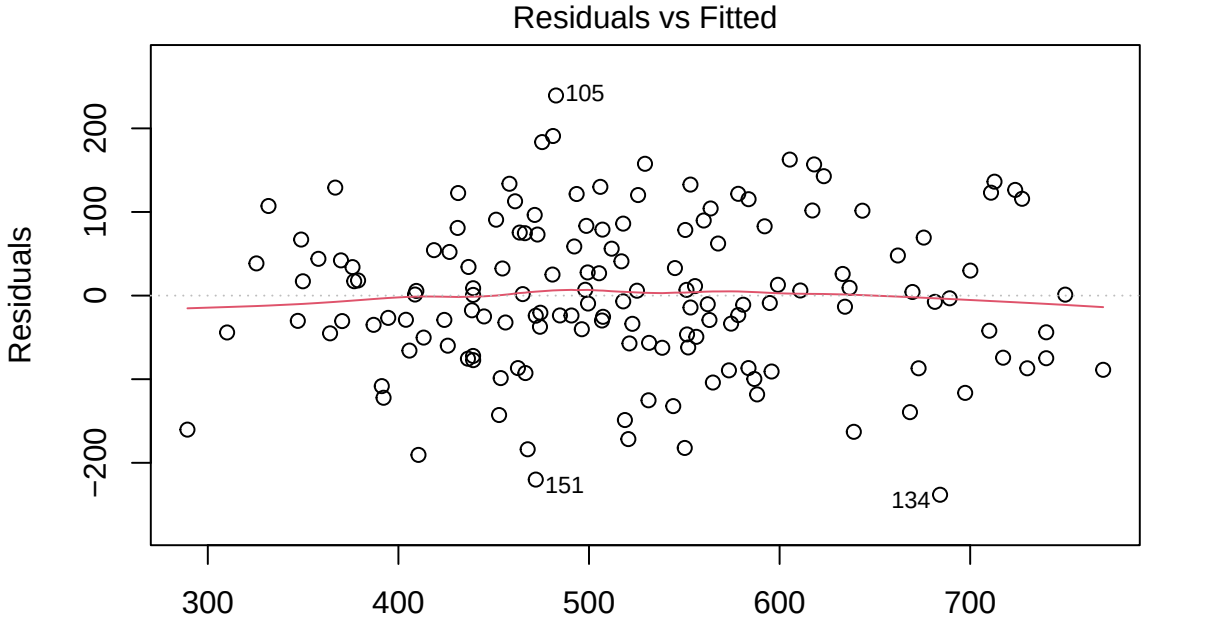
```
##                       GVIF Df GVIF^(1/(2*Df))
## forvent_lager     1.403145  2        1.088368
## weekend_helligdag 1.153516  1        1.074018
## kamjunk           1.131247  1        1.063601
## temp_gt25_3_dage  1.243248  1        1.115010
## temp_mean_past1h  1.486803  1        1.219345
```

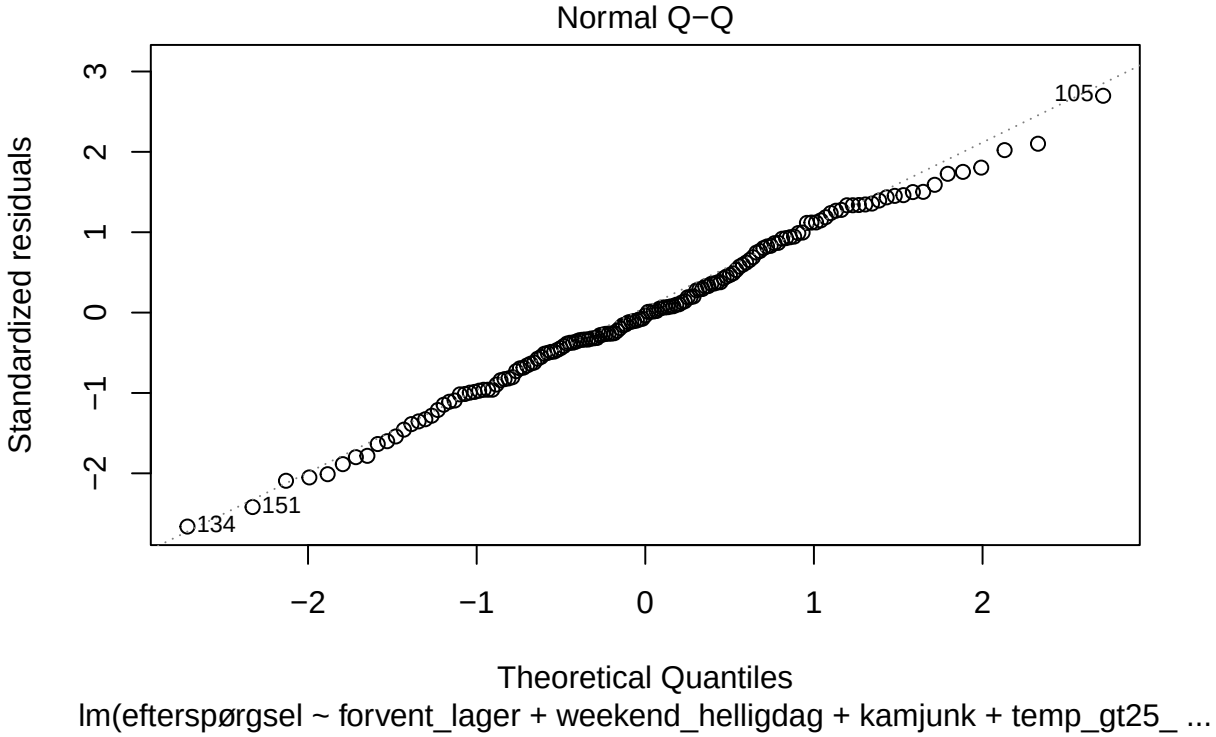
```
summary(lm.fit1)
```

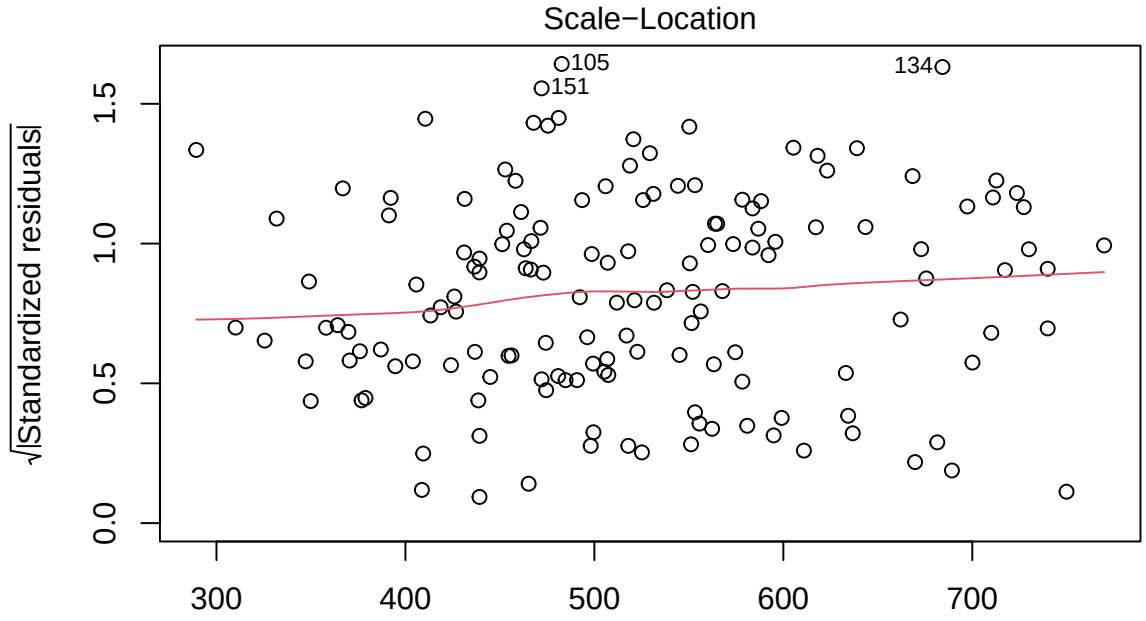
```
##
## Call:
## lm(formula = efterspørgsel ~ forvent_lager + weekend_helligdag +
##     kamjunk + temp_gt25_3_dage + temp_mean_past1h)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -238.194  -56.881   -3.217   68.191  239.302
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          416.173     41.831   9.949  < 2e-16 ***
## forvent_lagermellem  -99.154     20.343  -4.874 2.85e-06 ***
## forvent_lagerhøj    -129.649     22.244  -5.829 3.52e-08 ***
## weekend_helligdag1   108.313     16.180   6.694 4.50e-10 ***
## kamjunknej           -81.395     18.751  -4.341 2.66e-05 ***
## temp_gt25_3_dage    -104.347     35.381  -2.949  0.00372 **
## temp_mean_past1h       9.051      1.638   5.526 1.49e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

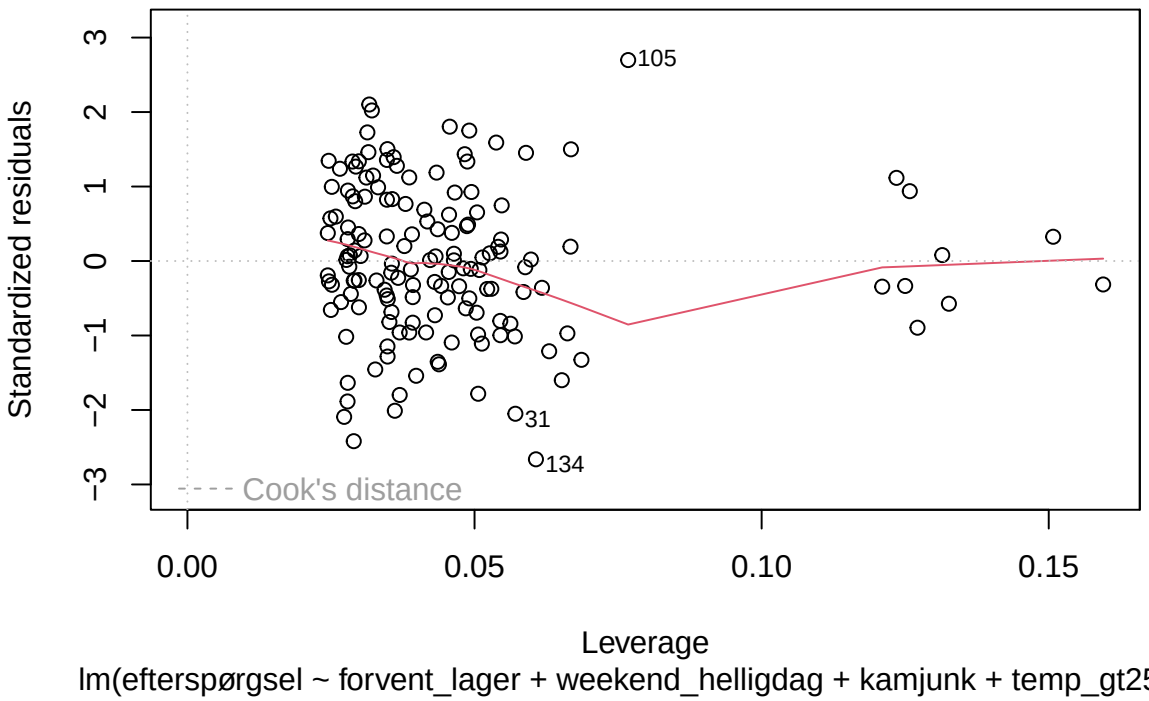
```
## Residual standard error: 92.31 on 144 degrees of freedom
## Multiple R-squared:  0.5806, Adjusted R-squared:  0.5631
## F-statistic: 33.22 on 6 and 144 DF,  p-value: < 2.2e-16
```

```
plot(lm.fit1)
```










```
# R^2 indikerer at de uafhængige variable forklarer 58% af variansens i data.
```

```
lm.fit2 = lm(efterspørgsel ~ 1) # simpel model
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = efterspørgsel ~ 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -391.23 -104.73  -14.23  104.77  329.77
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   520.23      11.37   45.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 139.7 on 150 degrees of freedom
```

```
lm.fit3 = lm(efterspørgsel ~ temp_mean_past1h^2) # melllem model.
summary(lm.fit3)
```

```
##
## Call:
## lm(formula = efterspørgsel ~ temp_mean_past1h^2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -305.02  -92.77  -11.28   84.39  306.18
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       346.035     36.288   9.536  < 2e-16 ***
## temp_mean_past1h    9.461      1.886   5.017 1.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 129.6 on 149 degrees of freedom
## Multiple R-squared:  0.1445, Adjusted R-squared:  0.1388
## F-statistic: 25.17 on 1 and 149 DF,  p-value: 1.476e-06
```

```
lm.fit4 = lm(efterspørgsel ~ temp_mean_past1h + temp_mean_past1h^5) # ekstrem model
summary(lm.fit4)
```

```
##
## Call:
## lm(formula = efterspørgsel ~ temp_mean_past1h + temp_mean_past1h^5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -305.02  -92.77  -11.28   84.39  306.18
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        346.035     36.288   9.536  < 2e-16 ***
## temp_mean_past1h     9.461      1.886   5.017 1.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 129.6 on 149 degrees of freedom
## Multiple R-squared:  0.1445, Adjusted R-squared:  0.1388
## F-statistic: 25.17 on 1 and 149 DF,  p-value: 1.476e-06
```

```
x <- data3$temp_max_past1h
y <- data3$efterspørgsel
data <- data.frame(y, x)
data
```

```r
set.seed(1)
cv.error <- rep(0, 5)
for (i in 1:5) {
  glm.fit <- glm(y~poly(x , i), data = )
  cv.error[i] <- cv.glm(data , glm.fit)$delta [1]
}
cv.error
```

```
## [1] 16916.63 15619.05 14636.34 14742.51 14743.11
```

```r
cv.error1 <- rep(0, 10)
for (i in 1:10) {
  glm.fit1 <- glm(efterspørgsel ~ poly(temp_mean_past1h, i), data = data3)
  cv.error1[i] <- cv.glm(data3, glm.fit1)$delta[1]
}
cv.error1
```

```
##  [1] 16993.90 15645.07 14800.23 14856.35 14804.66 14945.99 15194.21 16097.59
##  [9] 15579.84 15504.53
```

```r
summary(glm.fit1)
```

```
##
## Call:
## glm(formula = efterspørgsel ~ poly(temp_mean_past1h, i), data = data3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -358.80   -86.62     1.12    68.93   246.50
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   520.225      9.962  52.219  < 2e-16 ***
```

```
## poly(temp_mean_past1h, i)1   650.242    122.420   5.312 4.17e-07 ***
## poly(temp_mean_past1h, i)2  -469.047    122.420  -3.831 0.000192 ***
## poly(temp_mean_past1h, i)3  -373.456    122.420  -3.051 0.002732 **
## poly(temp_mean_past1h, i)4   111.787    122.420   0.913 0.362737
## poly(temp_mean_past1h, i)5   112.686    122.420   0.920 0.358903
## poly(temp_mean_past1h, i)6     7.861    122.420   0.064 0.948894
## poly(temp_mean_past1h, i)7    -9.160    122.420  -0.075 0.940462
## poly(temp_mean_past1h, i)8   -52.391    122.420  -0.428 0.669336
## poly(temp_mean_past1h, i)9   111.392    122.420   0.910 0.364429
## poly(temp_mean_past1h, i)10   70.260    122.420   0.574 0.566940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 14986.71)
##
##     Null deviance: 2925858  on 150  degrees of freedom
## Residual deviance: 2098139  on 140  degrees of freedom
## AIC: 1893
##
## Number of Fisher Scoring iterations: 2
```

```r
cv.error1 <- rep(0, 10)
for (i in 1:10) {
  glm.fit2 <- glm(efterspørgsel ~ poly(temp_max_past1h, i), data = data3)
  cv.error1[i] <- cv.glm(data3, glm.fit1)$delta[1]
}
cv.error1
```

```
##  [1] 16993.90 15645.07 14800.23 14856.35 14804.66 14945.99 15194.21 16097.59
##  [9] 15579.84 15504.53
```

```
summary(glm.fit1)
```

```
##
## Call:
## glm(formula = efterspørgsel ~ poly(temp_mean_past1h, i), data = data3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -358.80   -86.62     1.12    68.93   246.50
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  520.225      9.962  52.219  < 2e-16 ***
## poly(temp_mean_past1h, i)1   650.242    122.420   5.312 4.17e-07 ***
## poly(temp_mean_past1h, i)2  -469.047    122.420  -3.831 0.000192 ***
## poly(temp_mean_past1h, i)3  -373.456    122.420  -3.051 0.002732 **
## poly(temp_mean_past1h, i)4   111.787    122.420   0.913 0.362737
## poly(temp_mean_past1h, i)5   112.686    122.420   0.920 0.358903
## poly(temp_mean_past1h, i)6     7.861    122.420   0.064 0.948894
## poly(temp_mean_past1h, i)7    -9.160    122.420  -0.075 0.940462
## poly(temp_mean_past1h, i)8   -52.391    122.420  -0.428 0.669336
## poly(temp_mean_past1h, i)9   111.392    122.420   0.910 0.364429
## poly(temp_mean_past1h, i)10   70.260    122.420   0.574 0.566940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 14986.71)
##
##     Null deviance: 2925858  on 150  degrees of freedom
## Residual deviance: 2098139  on 140  degrees of freedom
## AIC: 1893
##
```

```
## Number of Fisher Scoring iterations: 2
```

```
cv.error1 <- rep(0, 10)
for (i in 1:10) {
  glm.fit3 <- glm(efterspørgsel ~ poly(temp_mean_past1h, i), data = data3)
  cv.error1[i] <- cv.glm(data3, glm.fit1)$delta[1]
}
cv.error1
```

```
##  [1] 16993.90 15645.07 14800.23 14856.35 14804.66 14945.99 15194.21 16097.59
##  [9] 15579.84 15504.53
```

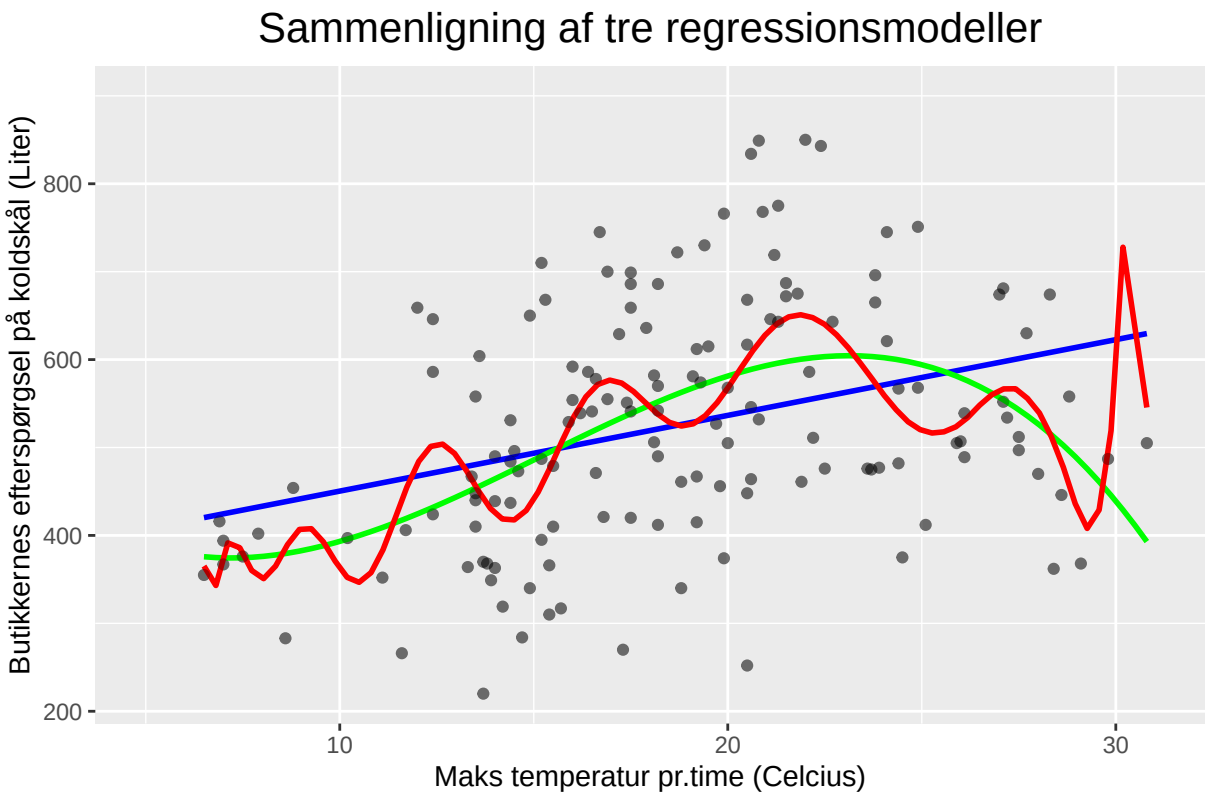
```
summary(glm.fit1)
```

```
##
## Call:
## glm(formula = efterspørgsel ~ poly(temp_mean_past1h, i), data = data3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -358.80   -86.62     1.12    68.93   246.50
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   520.225      9.962  52.219  < 2e-16 ***
## poly(temp_mean_past1h, i)1    650.242    122.420   5.312 4.17e-07 ***
## poly(temp_mean_past1h, i)2   -469.047    122.420  -3.831 0.000192 ***
## poly(temp_mean_past1h, i)3   -373.456    122.420  -3.051 0.002732 **
## poly(temp_mean_past1h, i)4    111.787    122.420   0.913 0.362737
## poly(temp_mean_past1h, i)5    112.686    122.420   0.920 0.358903
## poly(temp_mean_past1h, i)6      7.861    122.420   0.064 0.948894
## poly(temp_mean_past1h, i)7     -9.160    122.420  -0.075 0.940462
## poly(temp_mean_past1h, i)8    -52.391    122.420  -0.428 0.669336
```

```
## poly(temp_mean_past1h, i)9   111.392    122.420   0.910 0.364429
## poly(temp_mean_past1h, i)10   70.260    122.420   0.574 0.566940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 14986.71)
##
##     Null deviance: 2925858  on 150  degrees of freedom
## Residual deviance: 2098139  on 140  degrees of freedom
## AIC: 1893
##
## Number of Fisher Scoring iterations: 2
```

```
x <- data3$temp_mean_past1h
y <- data3$efterspørgsel
data <- data.frame(y, x)

ggplot(data3, mapping = aes(x=x, y=y)) +
  geom_point(alpha=1/3) +
  geom_smooth(method="glm", formula = y ~ poly(x, 1, raw=TRUE), se=FALSE, colour="blu
  geom_smooth(method="glm", formula = y ~ poly(x, 3, raw=TRUE), se=FALSE, colour="gre
  geom_smooth(method="glm", formula = y ~ poly(x, 22, raw=TRUE), se=FALSE, colour="re
geom_point(data=data3, mapping = aes(x=x, y=y), alpha=1/3) +
  labs(title = "Sammenligning af tre regressionsmodeller",
caption = "Kilde: Tal fra DMI 2002 fra perioden 1/4/22-30/8/22",
y = "Butikkernes efterspørgsel på koldskål (Liter)",
x = "Maks temperatur pr.time (Celcius)") +
ggeasy::easy_center_title() + # Centrerer titlen.
theme( plot.title = element_text(hjust = 0.5, size = 16),
plot.subtitle = element_text(hjust = 0.5, size = 14),
plot.caption = element_text(hjust = 1, face = "italic", size = 10 ))+
xlim(5, 31) + ylim(220, 900)
```



Kilde: Tal fra DMI 2002 fra perioden 1/4/22−30/8/22

Tidy

Transformer

Visualiser

Model

Kommunikér/analyse

Sessioninformation

For at højne reproducerbarheden printes der en udskrift om den nuværende R session:

```
SI <- sessionInfo(package = NULL) # Udskriver en liste om denne R session.
```

Litteratur

Bilag