Title

Course

Christian Oppegård Moen

DD MM YYYY

Contents

Libraries

```
library(Rmisc)
library(glmm)
library(ggplot2)
library(tidyr)
library(dplyr)
# library(MASS) library(reshape2) library(reshape)
```

```
defaultMar = c(5.1, 4.1, 4.1, 2.1)
```

Data

Data format is of tab separated values regarding noise on campus. Some variables are shown in the printout below. The response is `spmTriv` and `spmEff` which are "trivsel" and "effektivitet", respectively. The response is evaluated for each building block.

Load data and reformat

The time is on a unfeasible format, so we format it to total seconds used.

```
pathData = "./data"
d = read.delim("./data/data-315297-2023-03-08-1434-utf.txt", header = T)

# Reformat the time to be total time in seconds
formatTime <- function(t) {
    tSplit = strsplit(t, " ")[[1]]
    s = 0
    for (i in seq(1, length(tSplit), 2)) {
        s = s + switch(tSplit[i + 1], dag = strtoi(tSplit[i]) * 24 * 3600, dager = strtoi(tSplit[i]) *
            24 * 3600, time = strtoi(tSplit[i]) * 3600, timer = strtoi(tSplit[i]) *
            3600, minutt = strtoi(tSplit[i]) * 60, minutter = strtoi(tSplit[i]) *
            60, sekund = strtoi(tSplit[i]), sekunder = strtoi(tSplit[i]), 0)
    }
    return(s)
}

ftimes = unlist(lapply(d$Svartid, formatTime))
head(cbind(old = d$Svartid, new = ftimes))
```

```
##      old                       new
## [1,] "2 minutter 32 sekunder" "152"
## [2,] "1 minutt 58 sekunder"   "118"
```

```
## [3,] "2 minutter 32 sekunder" "152"
## [4,] "4 minutter 28 sekunder" "268"
## [5,] "2 minutter 45 sekunder" "165"
## [6,] "2 minutter 41 sekunder" "161"
```

```
d$Svartid = ftimes
```

Anc data someone wanted

```
N = length(d[, 1])
ancPercentage = length(d$spmTiltak_1[d$spmTiltak_1 == "mNc"])/N
ancPercentage
```

```
## [1] 0.7686567
```

Response (empirical mean and standard deviations (SD))

Compute per building means and SDs

```
# All locations/buildings
locs = unique(d$spmHvor)

# initiate per building empirical mean data frame
buildMeans = data.frame(matrix(ncol = length(locs), nrow = 2, dimnames = list(c("trivsel",
    "effektivitet"), locs)))
# initiate per building standard deviation data frame
buildSd = data.frame(matrix(ncol = length(locs), nrow = 2, dimnames = list(c("trivsel",
    "effektivitet"), locs)))
# Compute means and SDs
for (loc in locs) {
    buildMeans[loc] = c(mean(d$spmTriv[d$spmHvor == loc]), mean(d$spmEff[d$spmHvor ==
        loc]))
    buildSd[loc] = c(sd(d$spmTriv[d$spmHvor == loc]), sd(d$spmEff[d$spmHvor == loc]))
}
buildMeans[1:5]
```

```
##               elD      elB     kjel  elE     real
## trivsel      3.75 3.714286 3.666667 4.35 2.600000
## effektivitet 3.50 3.857143 3.000000 3.95 2.666667
```

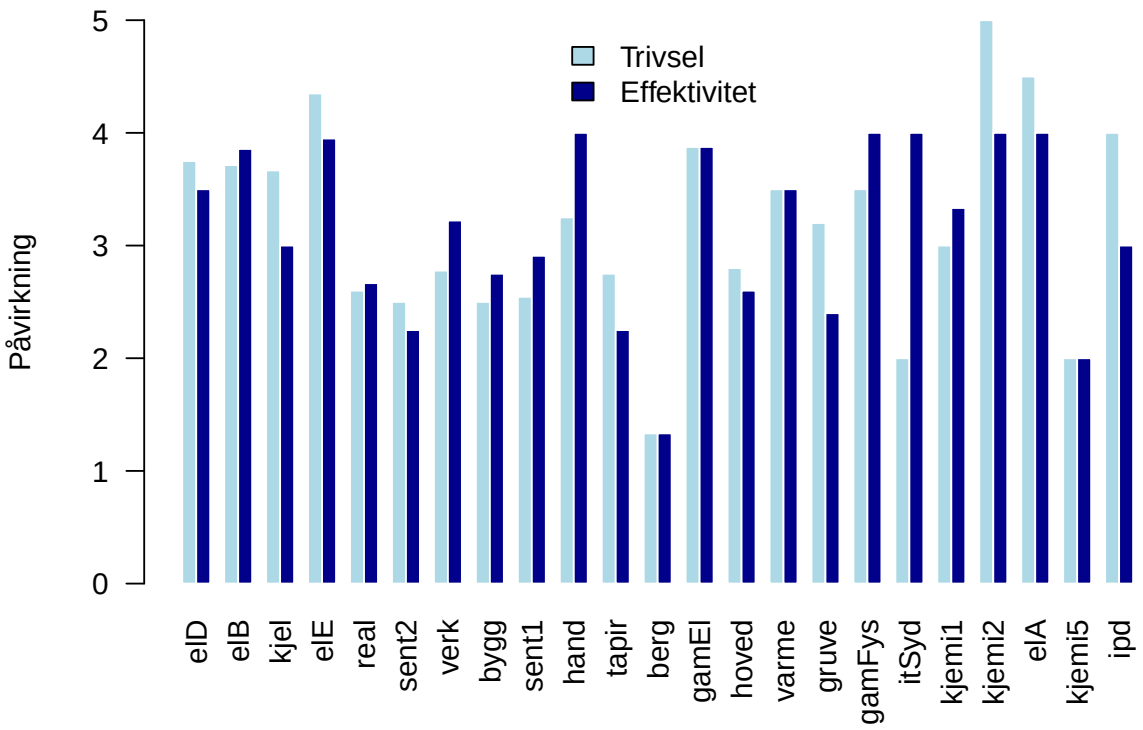
```
buildSd[1:5]
```

```
##              elD       elB     kjel       elE     real
## trivsel      0.5 0.7559289 1.751190 0.8127277 1.328728
## effektivitet 1.0 1.2149858 1.414214 1.0990426 1.321789
```

```
# rbind(buildMeans,buildSd)[1:5]
```

Barplot of responses

```
# æ is \u00E6
# ø is \u00F8
# å is \u00E5

barplot(
  ((as.matrix(buildMeans))),
  col = c("lightblue", 'darkblue'),
  border = "white",
  # main="Trivsel",
  ylab="P\u00E5virkning",
  beside = T,
  las=2,
  ylim = c(0,5)
  # space=0.1
)
legend(
  "top",
  legend = c("Trivsel", "Effektivitet"),
  fill = c("lightblue", 'darkblue'), bty = 'n')
```




```
d['Elektro_D.B2',]
```

```
##    NR Opprettet Endret spmFunk spmHvor spmTriv byggestoy1 personstoy1 trafikk1
## NA NA       <NA>   <NA>    <NA>    <NA>      NA          NA           NA       NA
##    vifte1 annet1 spmEff byggestoy2 personstoy2 trafikk2 vifte2 annet2
## NA     NA      NA      NA          NA           NA        NA     NA      NA
##    spmTiltak_1 spmTiltak_2 spmTiltak_3 spmTiltak_4 spmTiltak_5 spmTiltak_6
## NA        <NA>        <NA>        <NA>        <NA>        <NA>        <NA>
##    spmTiltak_7 spmTekst Svartid
## NA        <NA>     <NA>      NA
```

```
head(d)
```

```
##         NR        Opprettet           Endret spmFunk spmHvor spmTriv byggestoy1
## 1 25713468 08.02.2023 10:59 08.02.2023 10:59    stud     elD       4          4
## 2 25713476 08.02.2023 11:00 08.02.2023 11:00    stud     elB       3          4
## 3 25713498 08.02.2023 11:00 08.02.2023 11:00    stud    kjel       1          1
## 4 25713537 08.02.2023 11:02 08.02.2023 11:02    stud     elE       3          1
## 5 25713545 08.02.2023 11:03 08.02.2023 11:03    stud     elB       4          5
## 6 25714321 08.02.2023 11:44 08.02.2023 11:44    stud    real       3          1
##   personstoy1 trafikk1 vifte1 annet1 spmEff byggestoy2 personstoy2 trafikk2
## 1           2        1      2      5      4          5           3        1
## 2           1        1      3      1      2          3           1        1
## 3           2        1      1      1      1          1           3        1
```

```
## 4           2        1      1      1      1          1           1        1
## 5           2        1      1      1      3          5           2        1
## 6           4        1      3      1      4          4           3        4
##   vifte2 annet2 spmTiltak_1 spmTiltak_2 spmTiltak_3 spmTiltak_4 spmTiltak_5
## 1      1      5                                                  ingenting
## 2      3      1                                                  ingenting
## 3      2      1         mNc                                      ingenting
## 4      1      1         mNc                  flytt               ingenting
## 5      1      1                                                  ingenting
## 6      5      3         mNc                  flytt
##   spmTiltak_6 spmTiltak_7 spmTekst Svartid
## 1                                     152
## 2                                     118
## 3     paavirk                         152
## 4                                     268
## 5                                     165
## 6                                     161
```

```
mTriv= max(buildMeans['trivsel',])
mEff = max(buildMeans['effektivitet',])
buildMeans['effektivitet',buildMeans['effektivitet',]==mEff]
```

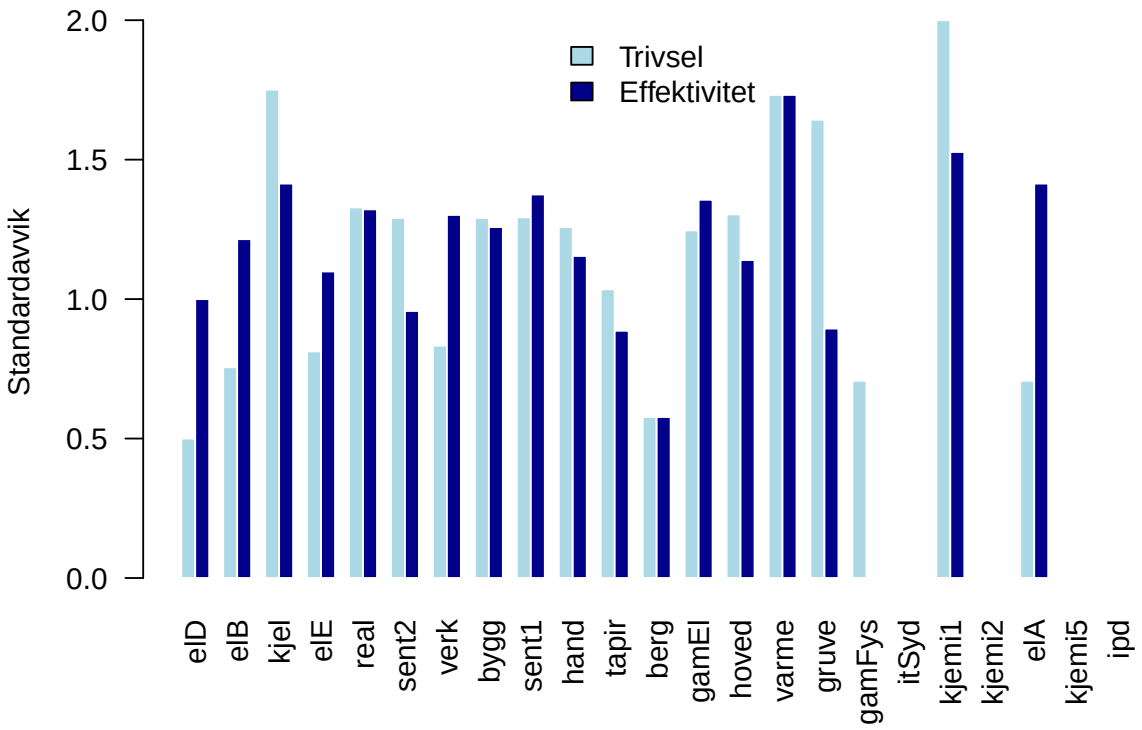
```
##              hand gamFys itSyd kjemi2 elA
## effektivitet    4      4     4      4   4
```

```
buildMeans['trivsel',buildMeans['trivsel',]==mTriv]
```

```
## [1] 5
```

```
# sort(buildMeans['trivsel',], decreasing = T)
```

```
barplot(
  ((as.matrix(buildSd))),
  col = c("lightblue", 'darkblue'),
  border = "white",
  # main="Trivsel",
  ylab="Standardavvik",
  beside = T,
  las=2,
  # ylim = c(0,5)
  # space=0.1
)
legend(
  "top",
  legend = c("Trivsel", "Effektivitet"),
  fill = c("lightblue", 'darkblue'), bty = 'n')
```



Make building data

```
# buildings = split(d, f=d$spmHvor) for (build in buildings){
# print(mean(build$spmTriv)) }
```

GLMM fit on errytin'

```
# set.seed(420) fitAll = glmm(spmTriv~0+Svartid, varcomps.names = c(''), data =
# d, family.glmm = Gaussian) dSummary = summary(d) head(d)
```

GLM fit on Svartid

```
# # Fit fitTidTriv = glm(factor(spmTriv, seq(1,5,1))~Svartid , data = d, family
# = 'binomial') fitTidEff = glm(factor(spmEff, seq(1,5,1))~Svartid , data = d,
# family = 'binomial') summary(fitTidTriv)$coefficients
# summary(fitTidEff)$coefficients summary(fitTidEff)
```

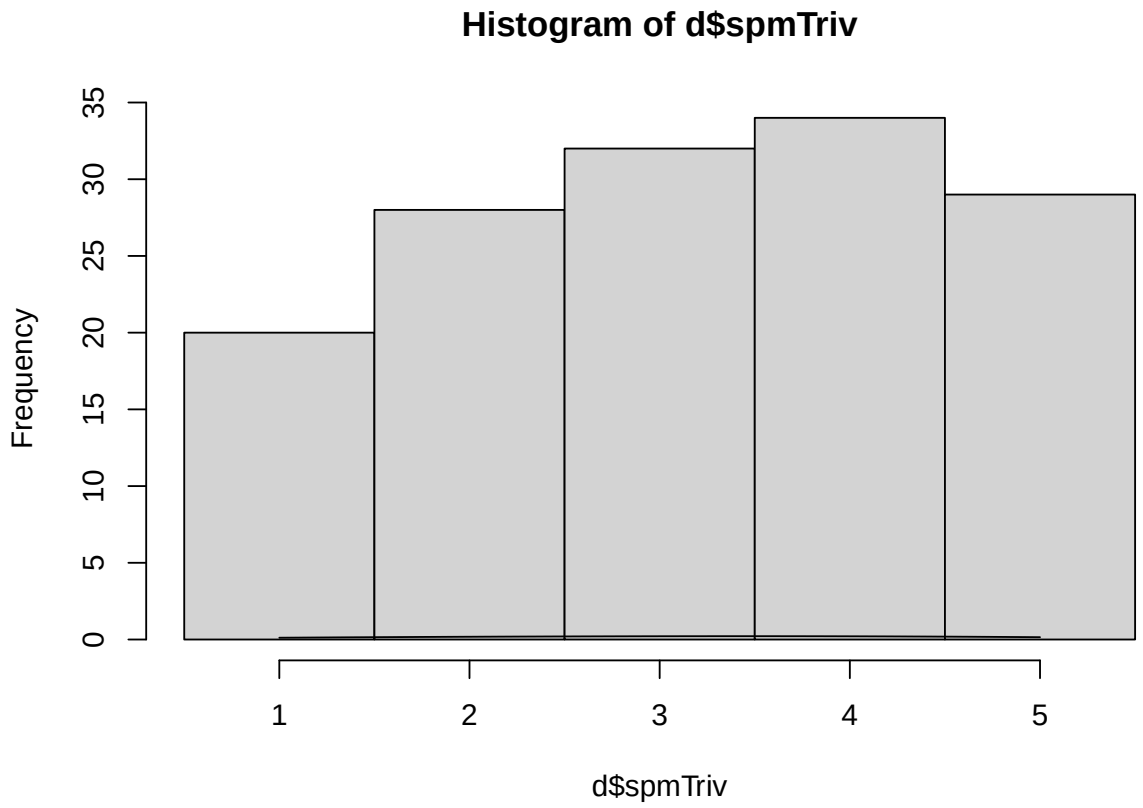
GLM fit on noise types

```
# factors = c('spmTriv', 'spmEff', 'byggestoy1',
# 'personstoy1','trafikk1','vifte1', 'annet1', 'byggestoy2',
# 'personstoy2','trafikk2','vifte2', 'annet2' ) dFactored = d # factorize =
# function() for (f in factors){ dFactored[f,] = factor(d[f,], seq(1,5,1),
# ordered = T) } fitTrivSource = glm( factor(spmTriv, seq(1,5,1), ordered =
# T)~byggestoy1 + personstoy1 + trafikk1 + vifte1 + annet1, data = dFactored,
# family = 'binomial' ) summary(fitTrivSource) cov(d$spmTriv, d$byggestoy1)
# cov(d$spmEff, d$byggestoy2) par(d)
```

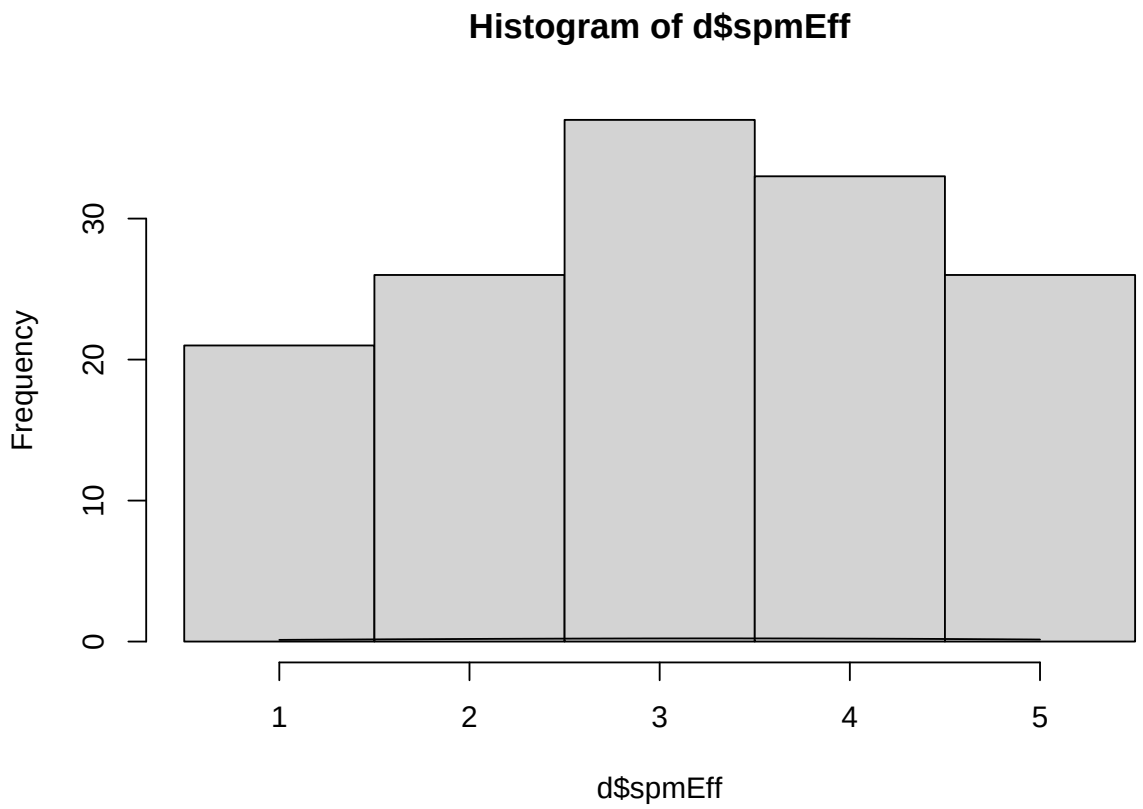
Initial data observations

Histograms of `effektivitet` and `trivsel` showing how many answered 1,2,3,...

```
hist(d$spmTriv, breaks = seq(0.5, 5.5, 1), freq = T)
lines(density(d$spmTriv, bw = 1, from = 1, to = 5))
```



```
hist(d$spmEff, breaks = seq(0.5, 5.5, 1), freq = T)
lines(density(d$spmEff, bw = 1, from = 1, to = 5))
```



Empiric data

```
# head(d) head(d[,18:(18+6)])
dTiltak = d[, 18:(18 + 6)]
hasAns = function(x) {
    return(sum(x != ""))
}
dTiltakSum = apply(dTiltak, FUN = hasAns, MARGIN = 2)
dTiltakSum
```

```
## spmTiltak_1 spmTiltak_2 spmTiltak_3 spmTiltak_4 spmTiltak_5 spmTiltak_6
##         109          26          69          10          66          10
## spmTiltak_7
##           6
```

Linear models

Fitting linear model with covariates being noise types.

```
lmTrivSource = lm(spmTriv ~ -1 + byggestoy1 + personstoy1 + trafikk1 + vifte1 + annet1,
    data = d)
lmNTcoefs = summary(lmTrivSource)$coefficients
signCoefs = lmNTcoefs[lmNTcoefs[, 4] < 0.01, ]
lmNTcoefs  # all coefficients
```

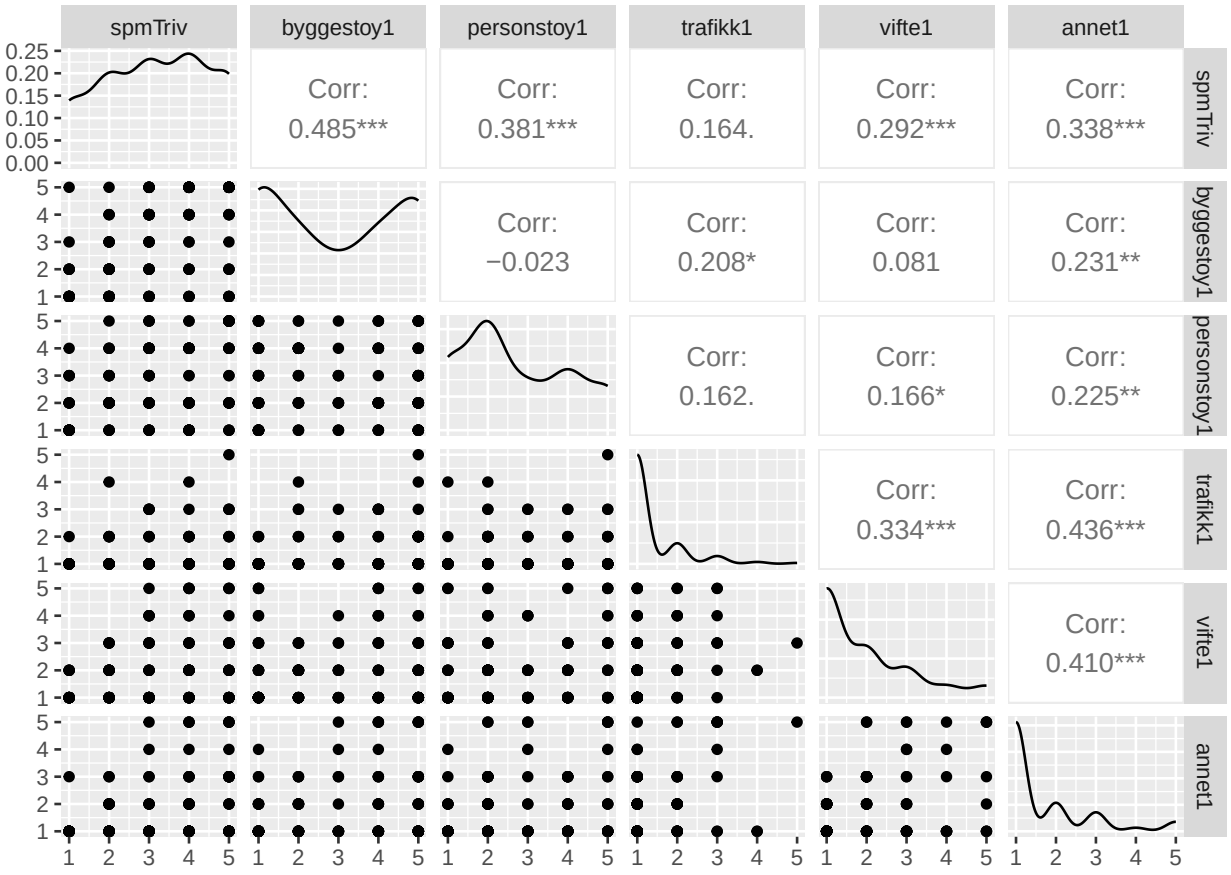
```
##                Estimate Std. Error    t value     Pr(>|t|)
## byggestoy1   0.4537925 0.04918643  9.2259699 4.347854e-16
## personstoy1  0.4367802 0.05741037  7.6080365 3.915891e-12
## trafikk1    -0.1004804 0.13559712 -0.7410217 4.599394e-01
## vifte1       0.2589169 0.08053252  3.2150599 1.624332e-03
## annet1       0.1429574 0.09380863  1.5239257 1.298157e-01
```

```
signCoefs  # significant coefficients
```

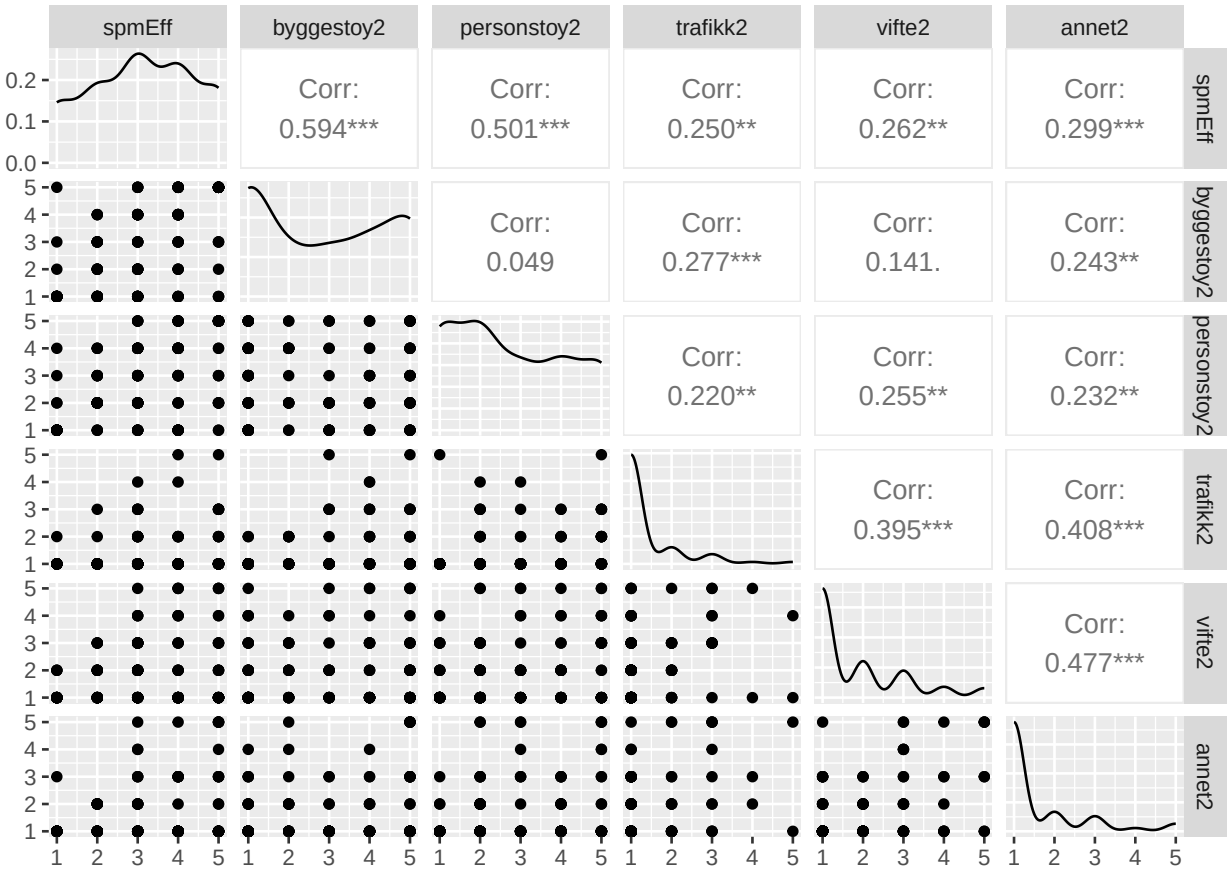
```
##               Estimate Std. Error  t value     Pr(>|t|)
## byggestoy1  0.4537925 0.04918643 9.225970 4.347854e-16
## personstoy1 0.4367802 0.05741037 7.608036 3.915891e-12
## vifte1      0.2589169 0.08053252 3.215060 1.624332e-03
```

Pairs plots

```
# pairs(dNumeric[,2:7])
library(GGally)
dNumeric = select_if(d, is.numeric)
ggpairs(dNumeric[, 2:7])  # Trivsel pairs
```




```
ggpairs(dNumeric[, 8:13])  # Effektivitet pairs
```



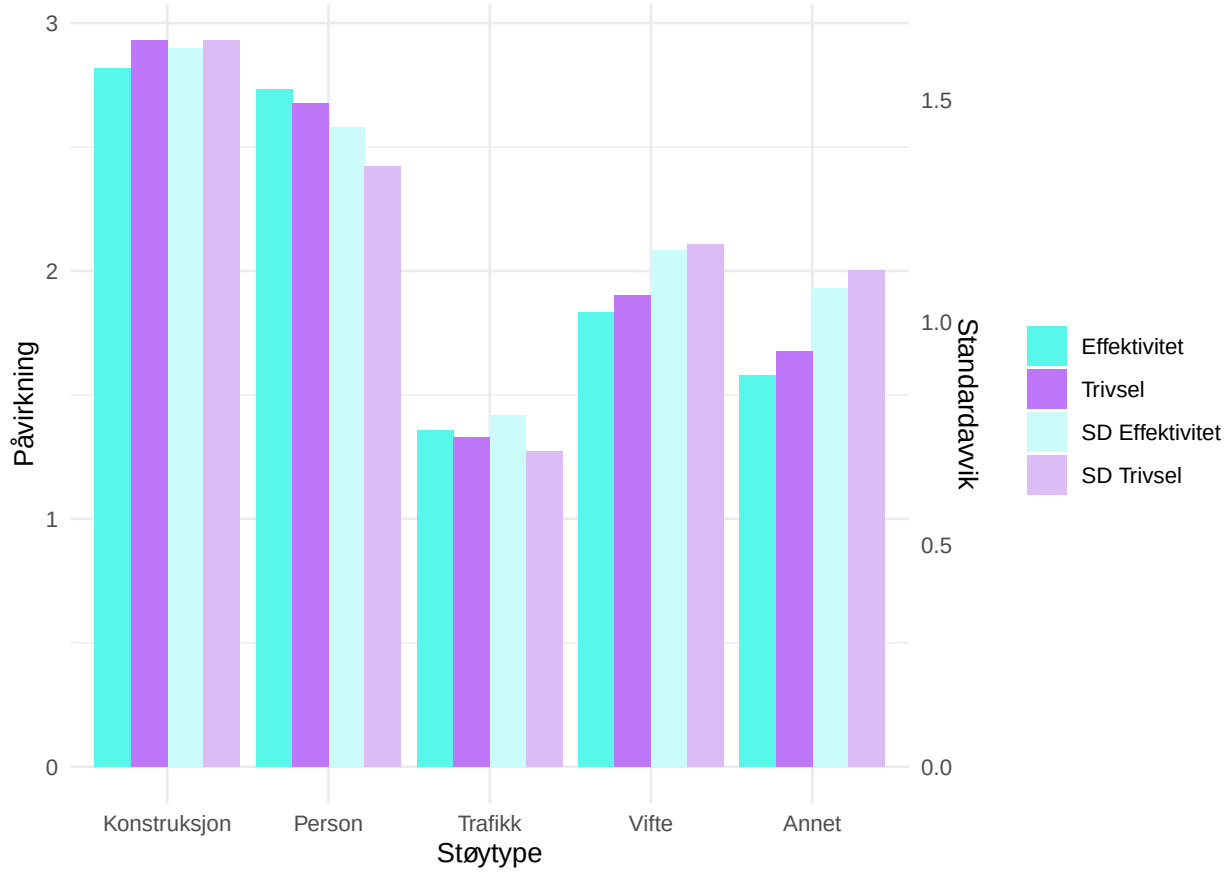
Noise type analysis

In this section we will take a look at each noise type and how much people feel like it influences their efficiency and well-being. First we consider what each individual has set the noise type influence e.g., `byggestoy1`= 3, and then we consider a weighted result where the wights are the overall influence divided by 5, e.g., `effektivitet/5*byggestoy1`.

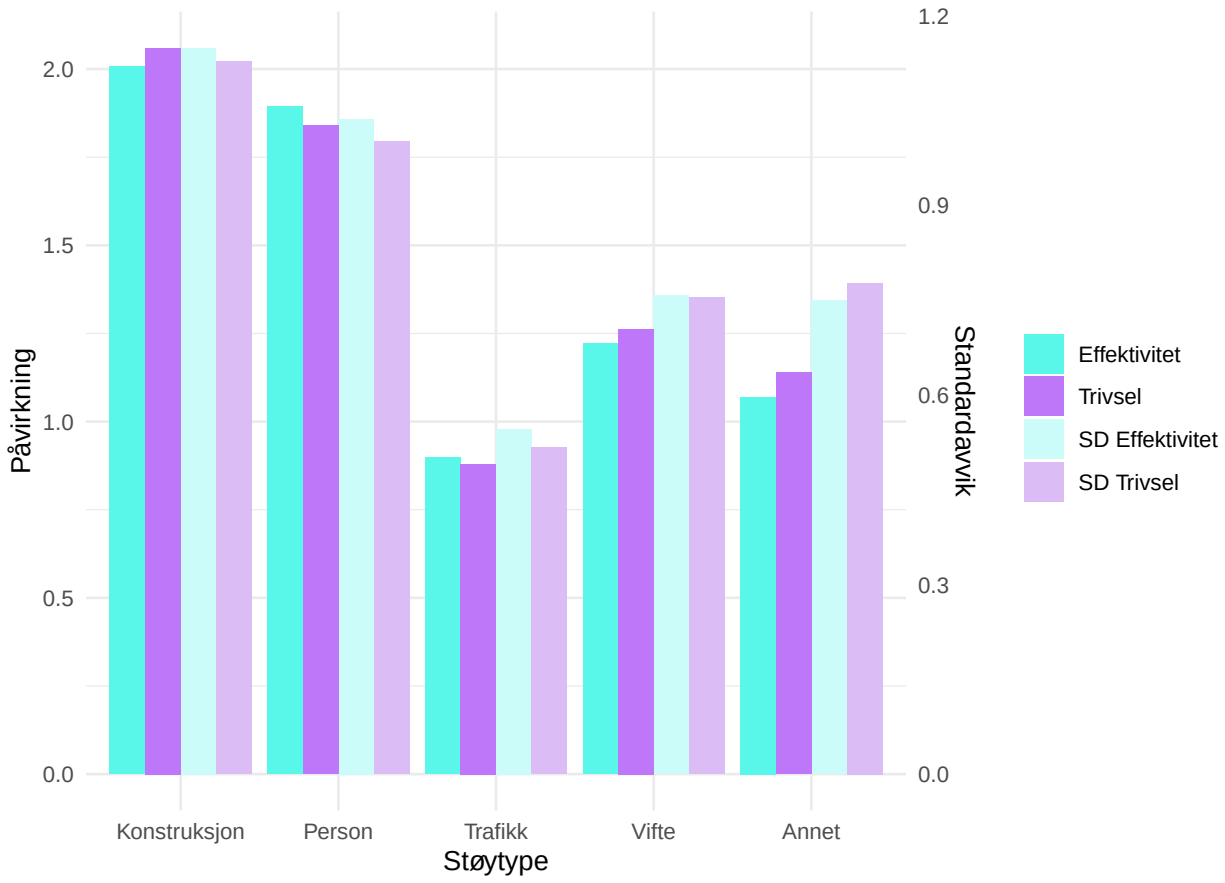
These uncertainties might be wrong!

```
# setup
ntColnames = c("Konstruksjon", "Person", "Trafikk", "Vifte", "Annet")
nts = c("Konstruksjon", "Person", "Trafikk", "Vifte", "Annet")
ntTriv = colnames(d)[7:11]
ntEff = colnames(d)[13:17]

# compute mean and sd unweighted mean(d[, 'vifte1'])
ntDf = data.frame(cbind(nts = 1:length(nts), meanEff = apply(d[, ntEff], FUN = mean,
    MARGIN = 2), sdEff = apply(d[, ntEff], FUN = sd, MARGIN = 2), meanTriv = apply(d[,
    ntTriv], FUN = mean, MARGIN = 2), sdTriv = apply(d[, ntTriv], FUN = sd, MARGIN = 2)))

# compute mean and sd weighted
```

```r
ntDfW = data.frame(cbind(nts = 1:length(nts), meanEff = apply(d[, ntEff] * (d$spmEff/5),
    FUN = mean, MARGIN = 2), sdEff = apply(d[, ntEff] * (d$spmEff/5), FUN = sd, MARGIN = 2),
    meanTriv = apply(d[, ntTriv] * (d$spmEff/5), FUN = mean, MARGIN = 2), sdTriv = apply(d[,
        ntTriv] * (d$spmEff/5), FUN = sd, MARGIN = 2)))

# Plot not weighted
ntDf_maxMean = max(ntDf[, c("meanEff", "meanTriv")])
ntDf_maxSD = max(ntDf[, c("sdEff", "sdTriv")])

ntDfLong <- ntDf |>
    pivot_longer(cols = -nts, names_to = "Type") |>
    mutate(scaled_value = ifelse(Type %in% c("meanEff", "meanTriv"), value, value/ntDf_maxSD *
        ntDf_maxMean))
# head(ntDfLong)

# æ is \u00E6 ø is \u00F8 å is \u00E5
ggplot(ntDfLong, aes(x = nts, y = scaled_value, fill = Type)) + geom_col(position = "dodge") +
    scale_y_continuous(sec.axis = sec_axis(~./ntDf_maxMean * ntDf_maxSD, name = "Standardavvik")) +
    scale_fill_manual(values = c("#59f7ea", "#be77f9", "#ccfcf9", "#dbbcf5"), labels = c("Effektivitet"
        "Trivsel", "SD Effektivitet", "SD Trivsel")) + labs(y = "Påvirkning", x = "Støytype") +
    scale_x_discrete(limit = nts) + theme_minimal() + theme(legend.title = element_blank())
```



```r
# With conf int bars ntDfMean = ntDf[, c('nts','meanEff', 'meanTriv')] ntDfSd =
# ntDf[, c('nts','sdEff', 'sdTriv')] ntDfMeanLong <- ntDfMean |>
# pivot_longer(cols = -nts,names_to = 'Type') ntDfSdLong <- ntDfSd |>
# pivot_longer(cols = -nts,names_to = 'Type') ntDfMeanLong$sd =
# ntDfSdLong$value

# ggplot(ntDfMeanLong, aes(x=nts, y = value,fill= Type)) +
# geom_col(position='dodge') + # scale_y_continuous(sec.axis = sec_axis(~ .
# /ntDf_maxMean*ntDf_maxSD , name = 'Standardavvik'))+ scale_fill_manual(
# values = c('#59f7ea', '#be77f9'), labels = c('Effektivitet', 'Trivsel') )+
# geom_errorbar( aes( # x=nts, ymin = value - 1.96*sd, ymax = value + 1.96*sd),
# color = 'black', position=position_dodge(.9), width=.2 ) + # facet_grid(cols
# = vars(Type))+ labs(y='P\u00E5virkning', x='St\u00F8ytype') +
# scale_x_discrete(limit = nts)+ theme_minimal() + theme(legend.title =
# element_blank())



# Plot Weighted
ntDfW_maxMean = max(ntDfW[, c("meanEff", "meanTriv")])
ntDfW_maxSD = max(ntDfW[, c("sdEff", "sdTriv")])

ntDfWLong <- ntDfW |>
    pivot_longer(cols = -nts, names_to = "Type") |>
    mutate(scaled_value = ifelse(Type %in% c("meanEff", "meanTriv"), value, value/ntDfW_maxSD *
        ntDfW_maxMean))

ggplot(ntDfWLong, aes(x = nts, y = scaled_value, fill = Type)) + geom_col(position = "dodge") +
    scale_y_continuous(sec.axis = sec_axis(~./ntDf_maxMean * ntDf_maxSD, name = "Standardavvik")) +
    scale_fill_manual(values = c("#59f7ea", "#be77f9", "#ccfcf9", "#dbbcf5"), labels = c("Effektivitet"
        "Trivsel", "SD Effektivitet", "SD Trivsel")) + labs(y = "Påvirkning", x = "Støytype") +
    scale_x_discrete(limit = nts) + theme_minimal() + theme(legend.title = element_blank())
```



Noise types with confidence intervals

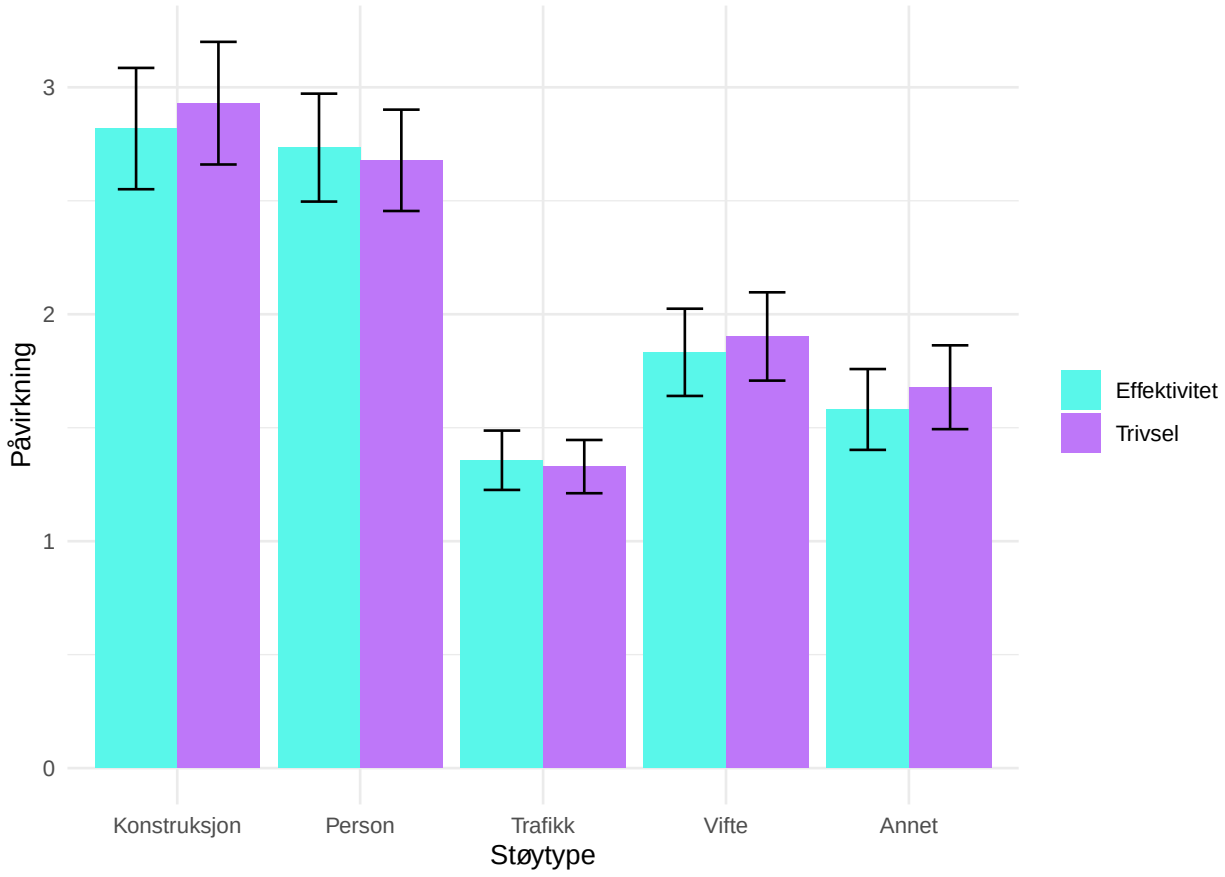
TODO: Need to manually verify the confidence intervals!

```
# With confInt
CI(d[,'byggestoy1'])
```

```
##    upper     mean    lower
## 3.200245 2.930070 2.659895
```

```
ntDfCi = data.frame(
  cbind(
    nts=1:length(nts),
    t(apply(d[,ntEff], FUN = CI, MARGIN=2)),
    t(apply(d[,ntTriv], FUN = CI, MARGIN=2))
    )
)

# colnames(ntDfCi)
meansTemp = c('nts','mean','mean.1')
upperTemp = c('nts','upper','upper.1')
lowerTemp = c('nts','lower','lower.1')
```

```
dfMTemp = ntDfCi[,meansTemp]
dfUTemp = ntDfCi[,upperTemp]
dfLTemp = ntDfCi[,lowerTemp]

ntDfCiLong <- dfMTemp|>
  pivot_longer(cols=-nts, names_to='Type')
ntDfULong = dfUTemp |>
  pivot_longer(cols=-nts, names_to='Type')
ntDfLLong = dfLTemp |>
  pivot_longer(cols=-nts, names_to='Type')

ntDfCiLong$upper=ntDfULong$value
ntDfCiLong$lower=ntDfLLong$value

ntDfCiLong$nts=nts[ntDfCiLong$nts]
# rep(c('eff','triv'), as.integer(length(nts)))
ntDfCiLong$Type=rep(c('eff','triv'), as.integer(length(nts)))
colnames(ntDfCiLong) = c('nts', 'Type', 'mean', 'upper', 'lower')

ggplot(ntDfCiLong, aes(x=nts, y = mean ,fill= Type)) +
  geom_col(position="dodge") +
  # scale_y_continuous(sec.axis = sec_axis(~ . /ntDf_maxMean*ntDf_maxSD , name = "Standardavvik"))+
  scale_fill_manual(
    values = c('#59f7ea', '#be77f9'),
    labels = c('Effektivitet', 'Trivsel')
  )+
  geom_errorbar(
    aes(
      # x=nts,
      ymin = lower,
      ymax = upper),
      color = "black",
    position=position_dodge(.9),
    width=.4
  ) +
  # facet_grid(cols = vars(Type))+
  labs(y="P\u00E5virkning", x="St\u00F8ytype") +
  scale_x_discrete(limit = nts)+
  theme_minimal() +
  theme(legend.title = element_blank())
```



Kjel and varme skit

Combine `kjel` and `varme` because of struggle with polygons in "building colored" map.

```
nKjel = length(d[d$spmHvor == "kjel", 1])
nVarme = length(d[d$spmHvor == "varme", 1])
nKV = nKjel + nVarme
vkTriv = c(d[d$spmHvor == "kjel", "spmTriv"], d[d$spmHvor == "varme", "spmTriv"])
vkEff = c(d[d$spmHvor == "kjel", "spmEff"], d[d$spmHvor == "varme", "spmEff"])
mTriv = mean(vkTriv)
mEff = mean(vkEff)
sdTriv = sd(vkTriv)
sdEff = sd(vkEff)
c(mTriv, mEff, sdTriv, sdEff)
```

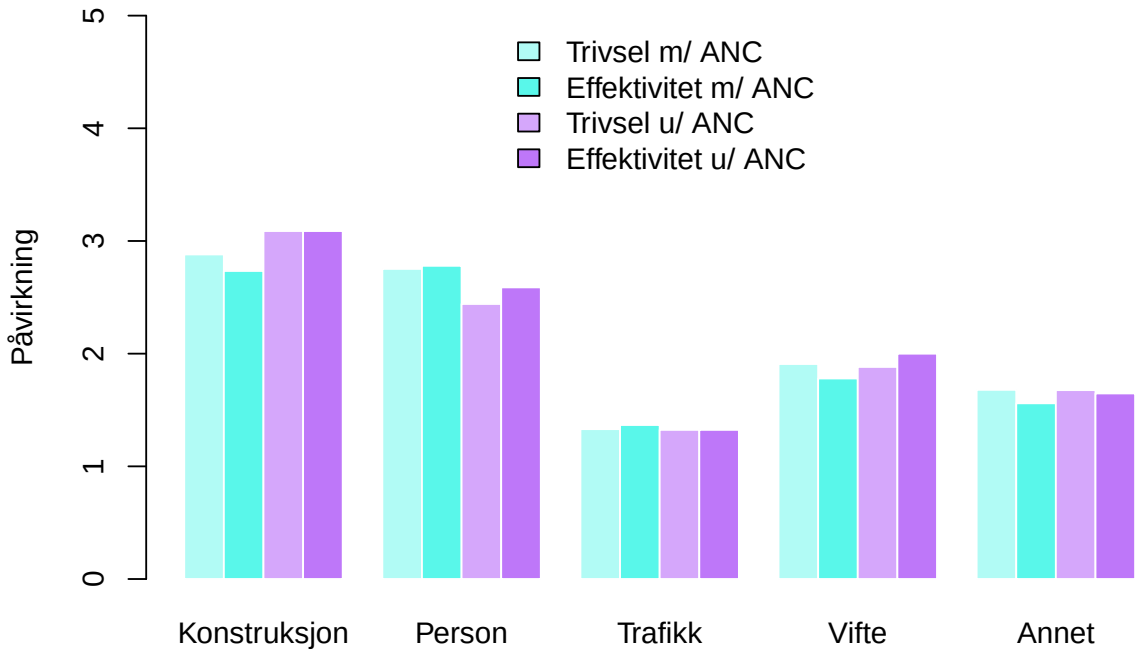
```
## [1] 3.600000 3.200000 1.646545 1.475730
```

ANC and noise type correlation

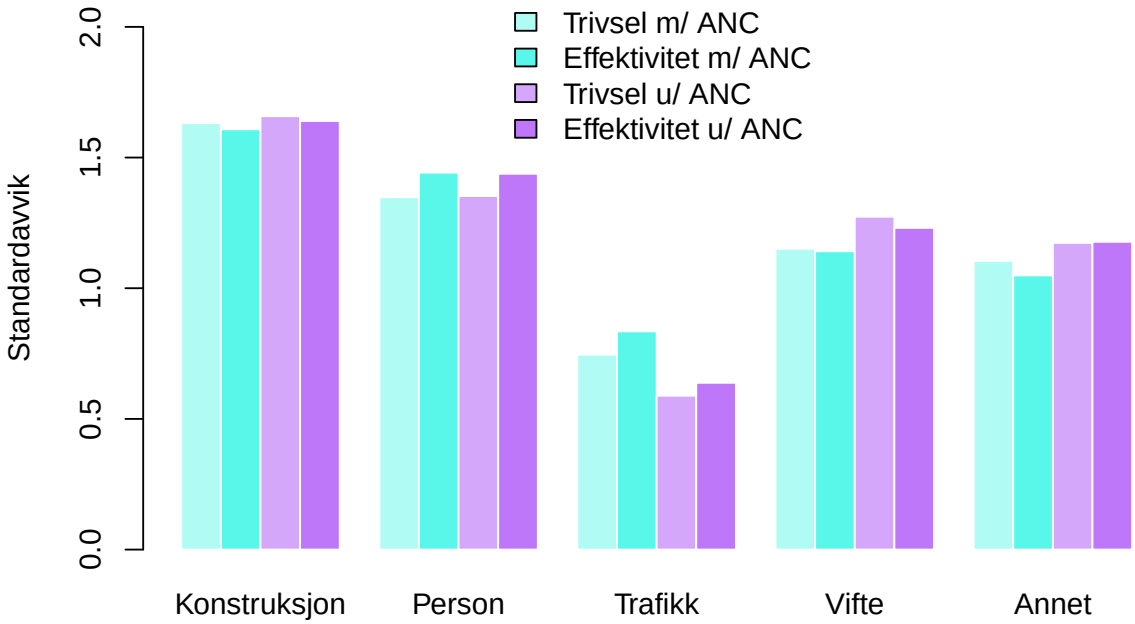
We will look at noise types and how much they disturb when the individual uses ANC compared to when the individual is not.

```
# ntTriv = colnames(d)[7:11]
# ntEff = colnames(d)[13:17]
ntColnames = c(
  'Konstruksjon', 'Person', 'Trafikk', 'Vifte', 'Annet')
ntRownames = c(
  'Trivsel m/ ANC', 'Trivsel u/ ANC', 'Effektivitet m/ ANC',
  'Effektivitet u/ ANC')
ntAnc = data.frame(
  matrix(
    ncol = length(ntTriv), nrow = 4,
    dimnames=list(ntRownames, ntColnames)
  )
)
ntAncSD = data.frame(
  matrix(
    ncol = length(ntTriv), nrow = 4,
    dimnames=list(ntRownames, ntColnames)
  )
)

# Find mean noise type disturbances
ancBool = d[,"spmTiltak_1"] == "mNc"
ntAnc[ntColnames] = rbind(
  apply(d[ancBool, ntTriv], FUN = mean, MARGIN = 2),
  apply(d[!ancBool, ntTriv], FUN = mean, MARGIN = 2),
  apply(d[ancBool,ntEff], FUN = mean, MARGIN = 2),
  apply(d[!ancBool,ntEff], FUN = mean, MARGIN = 2)
)
ntAncSD[ntColnames] = rbind(
  apply(d[ancBool, ntTriv], FUN = sd, MARGIN = 2),
  apply(d[!ancBool, ntTriv], FUN = sd, MARGIN = 2),
  apply(d[ancBool,ntEff], FUN = sd, MARGIN = 2),
  apply(d[!ancBool,ntEff], FUN = sd, MARGIN = 2)
)

# Barplot
reOrder = c(1,3,2,4)
cols = c("#b1fbf5", '#d5a7fb', '#59f7ea', '#be77f9')

# Mean
barplot(
  ((as.matrix(ntAnc[reOrder,]))),
  col = cols[reOrder],
  border = "white",
  # main="Trivsel",
  ylab="P\u00E5virkning",
  beside = T,
  # las=2,
  ylim = c(0,5)
  # space=0.1
)
legend(
  "top",
```

```
  legend = ntRownames[reOrder],
  fill = cols[reOrder], bty = 'n')
```




```
# SD
ntAncSDrange = max(ntAncSD)
barplot(
  ((as.matrix(ntAncSD[reOrder,]))),
  col = cols[reOrder],
  border = "white",
  # main="Trivsel",
  ylab="Standardavvik",
  beside = T,
  # las=2,
  ylim = c(0,ntAncSDrange*(1+0.3))
  # space=0.1
)
legend(
  "top",
  legend = ntRownames[reOrder],
  fill = cols[reOrder], bty = 'n')
```


Trivsel m/ ANC
Effektivitet m/ ANC
Trivsel u/ ANC
Effektivitet u/ ANC
Standardavvik
0.0
0.5
1.0
1.5
2.0
Konstruksjon
Person
Trafikk
Vifte
Annet