Tehtävät 4

,

2025-11-26

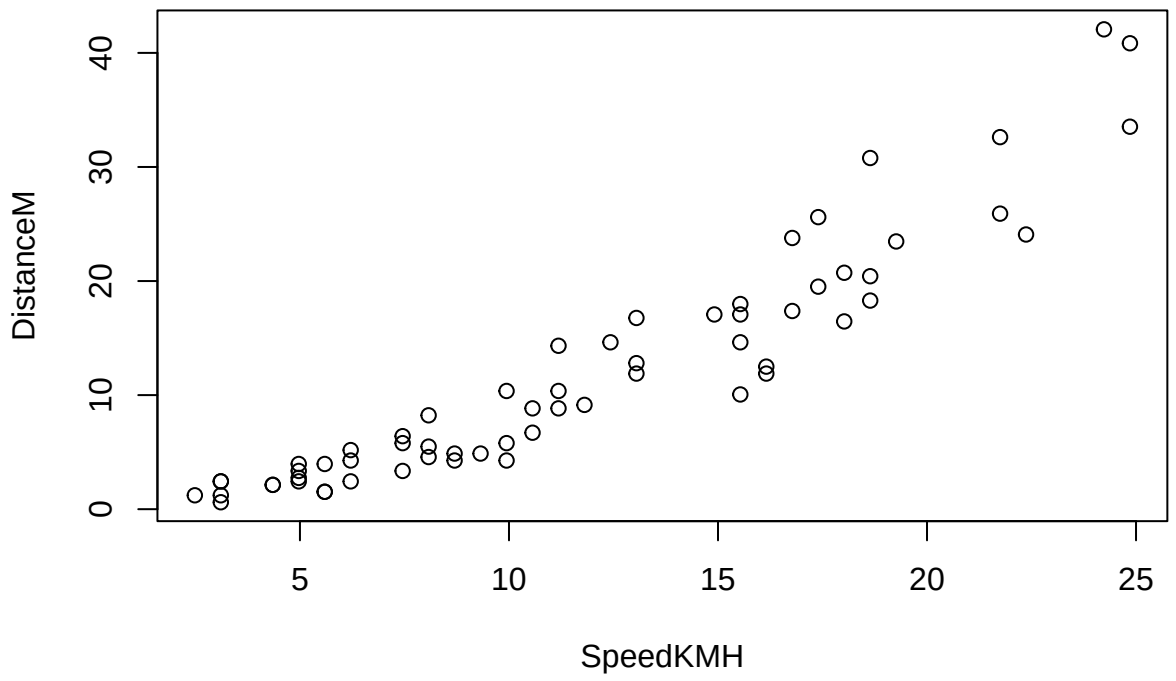
1

```
pysaytys <- read.csv("C:/Users/Omistaja/OneDrive - University of Helsinki/MAISTERI/Lineaariset mallit I

pysaytys["SpeedKMH"] <- pysaytys$Speed / 1.609344

pysaytys["DistanceM"] <- pysaytys$Distance / 3.28084
```

Ärsytttää se, ettei osaa olleenkaan maileja tai jalkoja, joten muutin nuo arvot kilometreiksi tunnissa ja metreiksi. Tämän ei pitäisi muuttaa tehtävää, luulisin.

a)

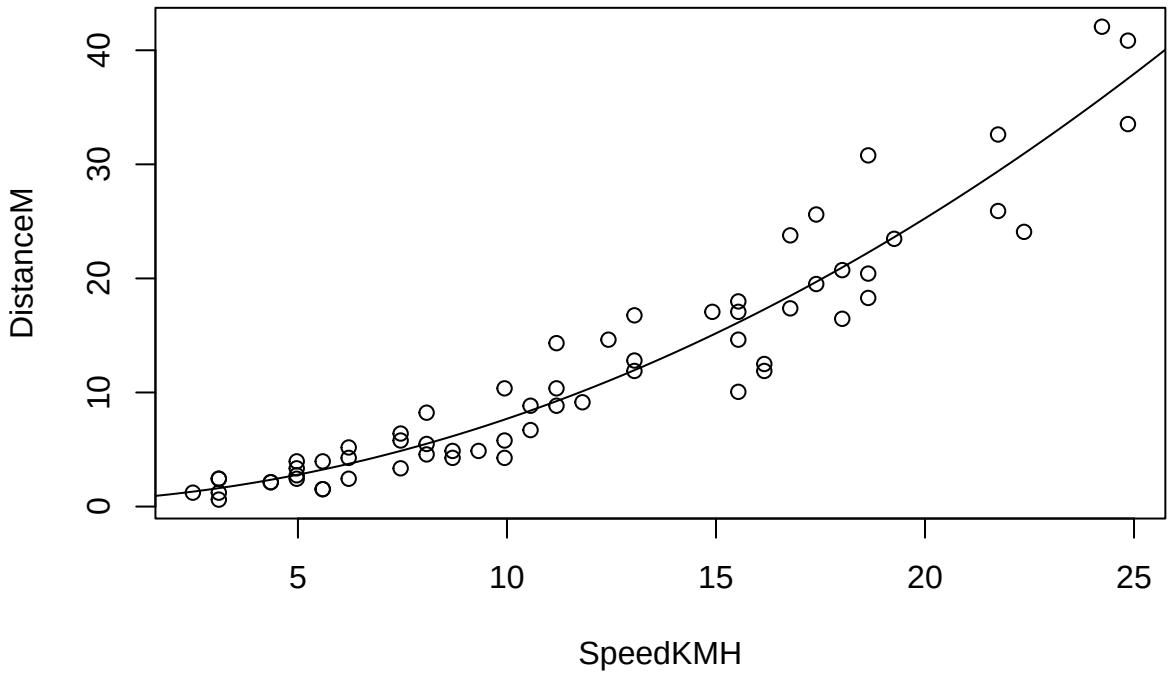
```
plot(DistanceM ~ SpeedKMH, data = pysaytys)
```



Tässä on nähtävissä kevyt nousevan kulmakertoimen trendi, kun nopeus kasvaa. Siksi polynominen sovite voisi olla sopiva.

b)

```
pysmatkat <- lm(DistanceM ~ SpeedKMH + I(SpeedKMH^2), data = pysaytys)
plot(DistanceM ~ SpeedKMH, data = pysaytys)

newdata = data.frame("SpeedKMH" = seq(0,30,0.1), "I(SpeedKMH^2)" = seq(0,30,0.1)^2)
plot(DistanceM ~ SpeedKMH, data = pysaytys)
lines(seq(0,30,0.1), predict(pysmatkat, newdata))
```



Lähdetään sitten testaamaan hypoteesia vakiovarianssista. Seurataan tehtävän 7.5. ohjeita ja verrataan tätä tulosta myös Breusch-Paganin testiin.

```
res <- pysaytys$DistanceM -
coef(pysmatkat)[1] - coef(pysmatkat)[2]*pysaytys$SpeedKMH - coef(pysmatkat)[3]*pysaytys$SpeedKMH^2

u <- (length(pysaytys$SpeedKMH) * res^2) / (sum(res^2))

y <- pysmatkat$fitted.values

lamdareg <- lm(u ~ y)

S <- sum((lamdareg$fitted.values - mean(u))^2)/2

1 - pchisq(S, df=1)
```

```
## [1] 1.645386e-06
```

```
lmtest::bptest(formula = pysmatkat, varformula = ~ pysmatkat$fitted.values)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  pysmatkat
## BP = 17.876, df = 1, p-value = 2.358e-05
```

Molemmat testit hylkäävät nollahypoteesin pienellä merkitsevyystasolla. Tehdään sama myös muille vastahypoteeseille. Nyt vain vaihdetaan varianssia selittävien Z-muuttujajoukon määrittelyä.

```
lamdareg <- lm(u ~ pysaytys$SpeedKMH)

S <- sum((lamdareg$fitted.values - mean(u))^2)/2

1 - pchisq(S, df=1)
```

```
## [1] 1.321162e-06
```

```
lmtest::bptest(formula = pysmatkat, varformula = ~ SpeedKMH, data = pysaytys)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  pysmatkat
## BP = 18.204, df = 1, p-value = 1.984e-05
```

Tätäkin vastahypoteesia vasten voimme hylätä nollahypoteesin.

```
lamdareg <- lm(u ~ pysaytys$SpeedKMH + I(pysaytys$SpeedKMH^2))

S <- sum((lamdareg$fitted.values - mean(u))^2)/2

1 - pchisq(S, df=2)
```

```
## [1] 8.026245e-06
```

```
lmtest::bptest(formula = pysmatkat, varformula = ~ SpeedKMH + I(SpeedKMH^2), data = pysaytys)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  pysmatkat
## BP = 18.261, df = 2, p-value = 0.0001083
```

Ja jälleen voimme todeta molempien testien hylkäävän nollahypoteesin vakiovarianssista myös näitä vastahypoteeseja vasten.

Näytetään vielä miten Breusch-Pagan testi on laskettu, esimerkkinä tämä keskimmäinen tapaus.

```
lamdareg <- lm(pysmatkat$residuals^2 ~ pysaytys$SpeedKMH)

S <- summary(lamdareg)$r.squared * length(pysaytys$SpeedKMH)

1 - pchisq(S, df=1)
```

```
## [1] 1.984455e-05
```

```
lmtest::bptest(formula = pysmatkat, varformula = ~ SpeedKMH, data = pysaytys)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  pysmatkat
## BP = 18.204, df = 1, p-value = 1.984e-05
```

Neliöllisen termin lisääminen on hyödyllistä. Se kuvaa mallia paremmin, vaikka se ei teekään varianssista vakiota.

c)

Sovitetaan painotettu malli.

```
pysmatw1 <- lm(DistanceM ~ SpeedKMH + I(SpeedKMH^2), data = pysaytys, weights = 1/SpeedKMH)

#tai sama saataisiin myös seuraavasti:
pysmatw2 <- lm(DistanceM/sqrt(SpeedKMH) ~ -1 + I(1/sqrt(SpeedKMH)) + I(SpeedKMH/sqrt(SpeedKMH)) + I(Spe

#pysmatw1:n kovarianssimatriisi:
Winv <- diag(pysaytys$SpeedKMH, ncol=length(pysaytys$SpeedKMH))
X <- matrix(c(rep(1,62), pysaytys$SpeedKMH, pysaytys$SpeedKMH^2), ncol = 3, byrow=FALSE)
beta <- matrix(coef(pysmatw1), ncol=1)
sigma2 <- (1/59) * t(as.matrix(y - X %*% beta)) %*% solve(Winv) %*% (y - X %*% beta)
covi <-  solve(t(X) %*% solve(Winv) %*% X) * as.numeric(sigma2)
y <- matrix(pysaytys$DistanceM, ncol=1)
beta1 <- solve(t(X) %*% solve(Winv) %*% X) %*% t(X) %*% solve(Winv) %*% y


lmtest::coeftest(pysmatw1)
```

```
##
## t test of coefficients:
##
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   0.4041339  0.9445703  0.4278    0.6703
## SpeedKMH      0.2197629  0.2063405  1.0650    0.2912
## I(SpeedKMH^2) 0.0511451  0.0088538  5.7766 3.027e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#lmtest::coeftest(pysmatw1, vcov. = covi) antaa samantuloksen kuin edellä

lmtest::coeftest(pysmatkat) #ilman painokertoimia
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   0.481695   1.555292  0.3097    0.7579
## SpeedKMH      0.204093   0.272935  0.7478    0.4576
## I(SpeedKMH^2) 0.051752   0.010283  5.0328 4.835e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Estimaatit eivät juurikaan muutu. (Mikä on oletettavaa, koska OLS-pysyy harhattomana ja tarkentuvana, vaikka varianssi ei olisi vakio.) Sen sijaan keskivirheet muuttuvat varsin paljon. Varsinaisesti p-arvojen perusteella tehtävät päätelmät eivät kuitenkaan muutu.

d)

Sitten lasketaankin estimaatit tavallisella OLS- (tai PNS-) tyylillä, mutta estimoimme keskivirheet eri tavalla ottaen huomioon mahdollisesti vaihtelevan virhetermin varianssin.

```
#perusmalli
pysmatkat <- lm(DistanceM ~ SpeedKMH + I(SpeedKMH^2), data = pysaytys)
covmat <- sandwich::vcovHC(pysmatkat, type = "HC3")

lmtest::coeftest(pysmatkat, vcov.=covmat)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   0.481695   1.309368  0.3679 0.7142767
## SpeedKMH      0.204093   0.309188  0.6601 0.5117625
## I(SpeedKMH^2) 0.051752   0.013616  3.8008 0.0003439 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Näemme, että keskivirheet yleisesti ottaen isonevat verrattuna painotettuun malliin että malliin, jossa on oletettu vakiovarianssi. Kysessä on siis varsin konservatiivinen tapa laskea keskivirheitä.

e)

Tehdäänpä kunnon bootstrap.

```
kertoimet <- matrix(ncol = 3, nrow = 0)
for(i in 1:5000){
  rivinumerot <- floor(runif(n=62, min = 1, max = 63))
  uusidata <- data.frame(matrix(nrow=0, ncol=5))
  colnames(uusidata) <- c("X", "Speed", "Distance", "SpeedKMH", "DistanceM")
  for(j in rivinumerot) {
    uusidata <- rbind(uusidata, pysaytys[j,])
  }
  kertoimet <- rbind(kertoimet,matrix(coef(lm(DistanceM ~ SpeedKMH + I(SpeedKMH^2),
                                              data = uusidata)), ncol=3, nrow=1))
}

varianssit <- apply(kertoimet, MARGIN = 2, FUN = var)
varianssit
```

```
## [1] 1.5316191258 0.0813899125 0.0001520833
```

Nyt meillä on varianssit kullekin kertoimelle estimoituna. Voimme vielä tehdä näillä keskivirheillä t-testit kertoimille ja verrata niitä aiempiin tuloksiin. Otamme kuitenkin perussovitteen painottamattomasta PNS:stä.

```
lmtest::coeftest(pysmatkat, vcov. = cov(kertoimet))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
```

```
## (Intercept)   0.481695   1.237586  0.3892    0.6985
## SpeedKMH      0.204093   0.285289  0.7154    0.4772
## I(SpeedKMH^2) 0.051752   0.012332  4.1965 9.253e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tällä bootstrap-menetelmällä saamme hiukan suuremmat keskivirheet kuin muilla menetelmillä. Päätelmät eivät kuitenkaan muutu.

Tai bootstrap-funktiolla suoraan:

```
coef_function <- function(formula, data, indices) {
  d <- data[indices,] #allows boot to select sample
  fit <- lm(formula, data=d) #fit regression model
  return(coef(fit)) #return coefficient estimates of model
}

boot::boot(data = pysaytys, statistic = coef_function, R = 5000, formula = DistanceM ~ SpeedKMH + I(Spe
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = pysaytys, statistic = coef_function, R = 5000,
##     formula = DistanceM ~ SpeedKMH + I(SpeedKMH^2))
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 0.48169475 -0.0546986498  1.22394441
## t2* 0.20409324  0.0113801941  0.28481398
## t3* 0.05175164 -0.0005176878  0.01241691
```