Project 2

Anton Stråhle

15 februari 2020

Exercise 1

```r
library(knitr)
library(readr)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

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

## -- Conflicts ------------------------------------------------------------------------------------ t
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
options(knitr.kable.NA = '')

data <- read.delim("Projekt2_Grupp9.txt", sep = ";") %>%
  mutate(ClaimYear = floor((ClaimDay-1)/365) + 1,
         PaymentYear = floor((PaymentDay-1)/365) + 1) %>%
  mutate(DevelopmentYear = PaymentYear-ClaimYear + 1) %>%
  select(-PaymentYear) %>%
  complete(ClaimType, nesting(ClaimYear, DevelopmentYear), fill = list(PaymentDay = 1, ClaimCost = 0, Cl
  filter(DevelopmentYear < 11)


ct1 <- data %>%
  filter(ClaimType == 1) %>%
  group_by(ClaimYear, DevelopmentYear) %>%
  summarize(Total = sum(ClaimCost)) %>%
  mutate(Total = cumsum(Total)) %>%
  spread(value = Total, key = DevelopmentYear) %>%
  ungroup() %>%
  filter(ClaimYear >= 11) %>%
  select(-ClaimYear)

ct2 <- data %>%
  filter(ClaimType == 2) %>%
  group_by(ClaimYear, DevelopmentYear) %>%
  summarize(Total = sum(ClaimCost)) %>%
  mutate(Total = cumsum(Total)) %>%
```

```
  spread(value = Total, key = DevelopmentYear) %>%
  ungroup() %>%
  filter(ClaimYear >= 11) %>%
  select(-ClaimYear)

#Estimates the incremental factors f based on the observed triangle (not trapezoid!)

estimateF <- function(ct){

  n <- ncol(ct)

  r <- nrow(ct)

  fVec <- c()

  for(i in 1:(n-1)){

    f <- sum(ct[1:(r-i),i+1])/sum(ct[1:(r-i),i])

    fVec[i] <- f

  }

  fVec

}


fillct <- function(ct){

  n <- ncol(ct)

  r <- nrow(ct)

  s <- r - n

  fHat <- estimateF(ct)

  for(i in (2+s):r){

    for(j in (n-i+2+s):n){

      ct[i,j] <- ct[i,j-1]*fHat[j-1]

    }

  }

    ct

}

fHat1 <- estimateF(ct1)
```

```r
fHat2 <- estimateF(ct2)

fullct1 <- fillct(ct1)

fullct2 <- fillct(ct2)

kable(ct1, caption = "Paid claims triangle of type 1", row.names = c(1:10))
```

```
## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used
```

```
## Warning in if (row.names) {: the condition has length > 1 and only the first
## element will be used
```

Table 1: Paid claims triangle of type 1

	1	2	3	4	5	6	7	8	9	10
1	14002196	20146708	21434023	22027415	22151985	22213380	22265789	22265789	22265789	22265789
2	7814759	11756938	12506719	12719773	12846004	12846004	12846004	12846004	12846004	
3	5181897	7401722	7820233	7922290	7922290	7940404	7940404	7940404		
4	5037120	7327216	7944307	8104325	8117003	8117003	8117003			
5	8042298	11453010	12085662	12166515	12212464	12274792				
6	6752949	10210348	10890964	11255347	11302949					
7	3715909	5176779	5580922	5689709						
8	6507705	9460226	10056047							
9	8386236	11910073								
10	6407931									

```r
kable(ct2, caption = "Paid claims triangle of type 2", row.names = c(1:10))
```

```
## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used

## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used
```

Table 2: Paid claims triangle of type 2

	1	2	3	4	5	6	7	8	9	10
1	3380661	11917434	19149141	23136198	25676759	26747939	27363824	27499046	27701642	27860156
2	4125276	14152392	21368586	25851401	28418708	29548445	30201830	30681144	30772972	
3	4388321	16013036	24759810	30271447	32403142	33733300	34205903	34665267		
4	3275256	11368463	16545811	19797983	21403379	22569878	23045976			
5	5981591	19410790	29558338	34979542	38445302	40182223				
6	4140485	13329441	20440916	24176081	26578838					
7	4282806	16150541	23746116	28208154						
8	3958824	13791706	20831932							
9	3775045	15531614								
10	4358136									

```r
kable(fullct1, caption = "Full claims triangle of type 1 predicted with CL", row.names = c(1:10))
```

```
## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used

## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used
```

Table 3: Full claims triangle of type 1 predicted with CL

	1	2	3	4	5	6	7	8	9	10
1	14002196	20146708	21434023	22027415	22151985	22213380	22265789	22265789	22265789	22265789
2	7814759	11756938	12506719	12719773	12846004	12846004	12846004	12846004	12846004	12846004
3	5181897	7401722	7820233	7922290	7922290	7940404	7940404	7940404	7940404	7940404
4	5037120	7327216	7944307	8104325	8117003	8117003	8117003	8117003	8117003	8117003
5	8042298	11453010	12085662	12166515	12212464	12274792	12287377	12287377	12287377	12287377
6	6752949	10210348	10890964	11255347	11302949	11328296	11339910	11339910	11339910	11339910
7	3715909	5176779	5580922	5689709	5717088	5729908	5735783	5735783	5735783	5735783
8	6507705	9460226	10056047	10264529	10313922	10337051	10347649	10347649	10347649	10347649
9	8386236	11910073	12683551	12946506	13008805	13037977	13051345	13051345	13051345	13051345
10	6407931	9286944	9890068	10095108	10143686	10166433	10176857	10176857	10176857	10176857

```r
kable(fullct2, caption = "Full claims triangle of type 2 predicted with CL", row.names = c(1:10))
```

```
## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used

## Warning in if (is.na(row.names)) row.names = has_rownames(x): the condition has
## length > 1 and only the first element will be used
```

Table 4: Full claims triangle of type 2 predicted with CL

	1	2	3	4	5	6	7	8	9	10
1	3380661	11917434	19149141	23136198	25676759	26747939	27363824	27499046	27701642	27860156
2	4125276	14152392	21368586	25851401	28418708	29548445	30201830	30681144	30772972	30949061
3	4388321	16013036	24759810	30271447	32403142	33733300	34205903	34665267	34840692	35040057
4	3275256	11368463	16545811	19797983	21403379	22569878	23045976	23315657	23433647	23567739
5	5981591	19410790	29558338	34979542	38445302	40182223	40973727	41453197	41662973	41901376
6	4140485	13329441	20440916	24176081	26578838	27747438	28294003	28625096	28769955	28934582
7	4282806	16150541	23746116	28208154	30831459	32187035	32821051	33205119	33373155	33564122
8	3958824	13791706	20831932	24963281	27284818	28484458	29045541	29385428	29534135	29703135
9	3775045	15531614	23591639	28270288	30899371	32257933	32893345	33278259	33446666	33638054
10	4358136	15380393	23361944	27995040	30598525	31943860	32573085	32954252	33121019	33310543

Exercise 2

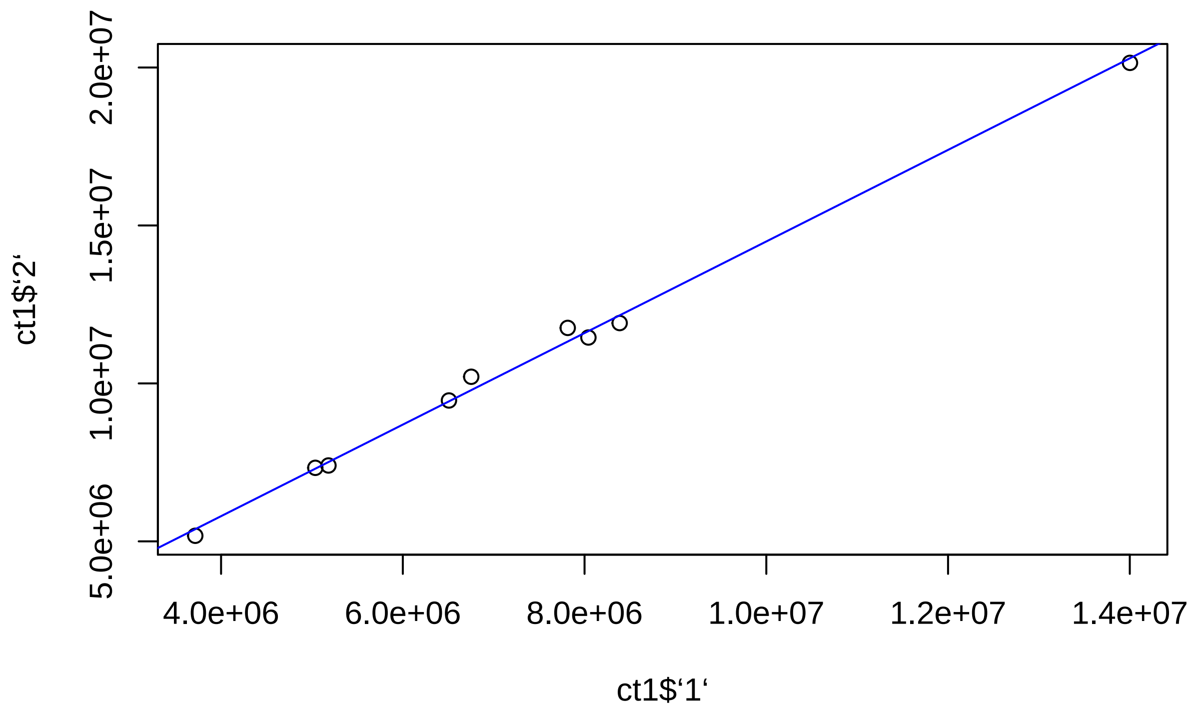
We now want to check whether or not Mack's undrlying assumptions are met in our case. The assumptions are as follows.

1. $E[C_{i,k+1}|C_{i,1},...,Ci,k] = f_kC_{i,k}$

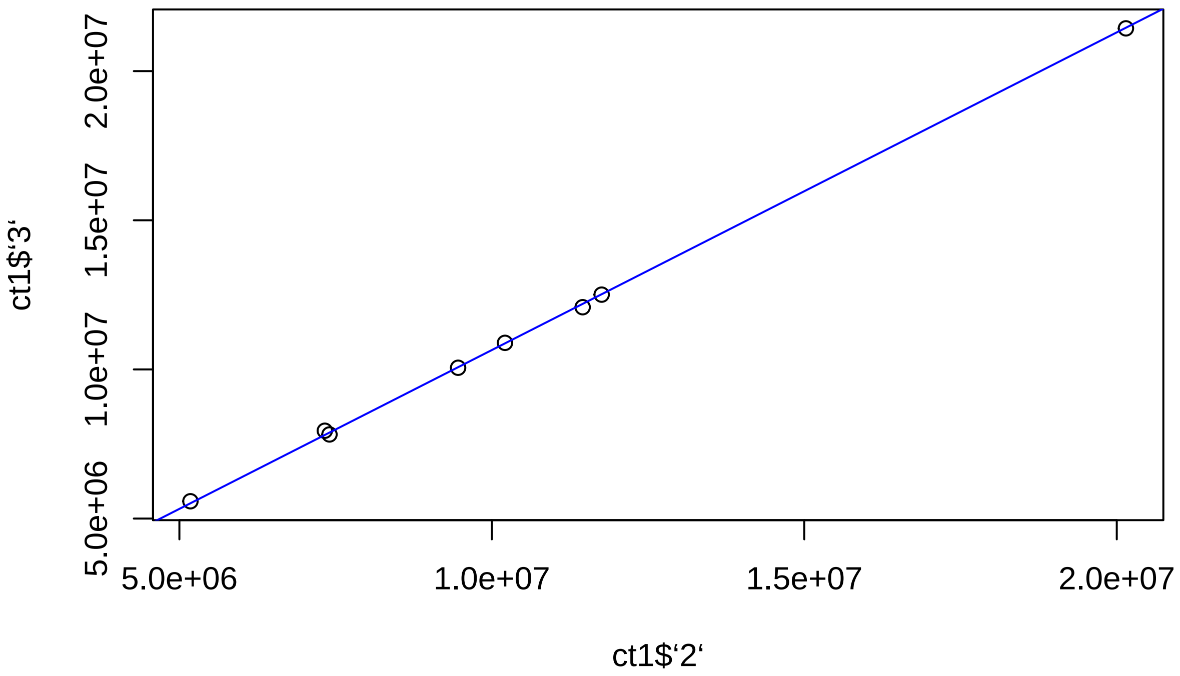
2. Independent accident years
3. $Var(C_{i,k+1}|C_{i,1},...,Ci,k) = \sigma_k^2 C_{i,k}$

We begin by examing whether or not the we have an approximate linear relationship between $C_{i,k}$ and $C_{i,k+1}$ for $i = 1, ..., 10$

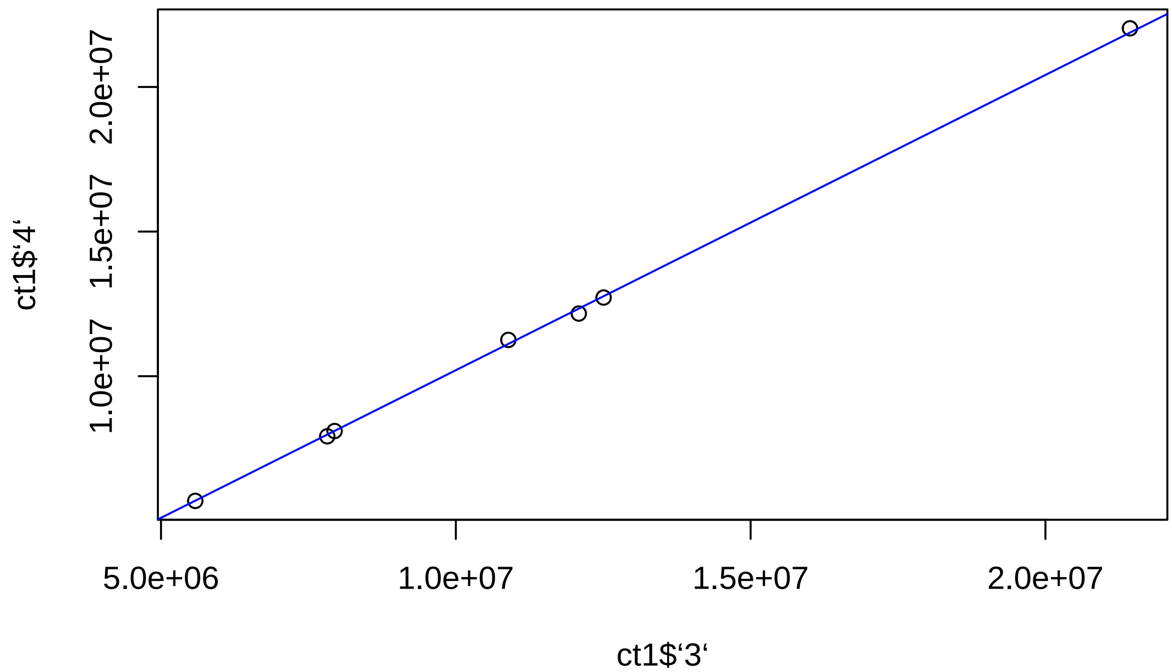
```
#Gör om gör fint!

plot(ct1$`1`, ct1$`2`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[1]}), co
```



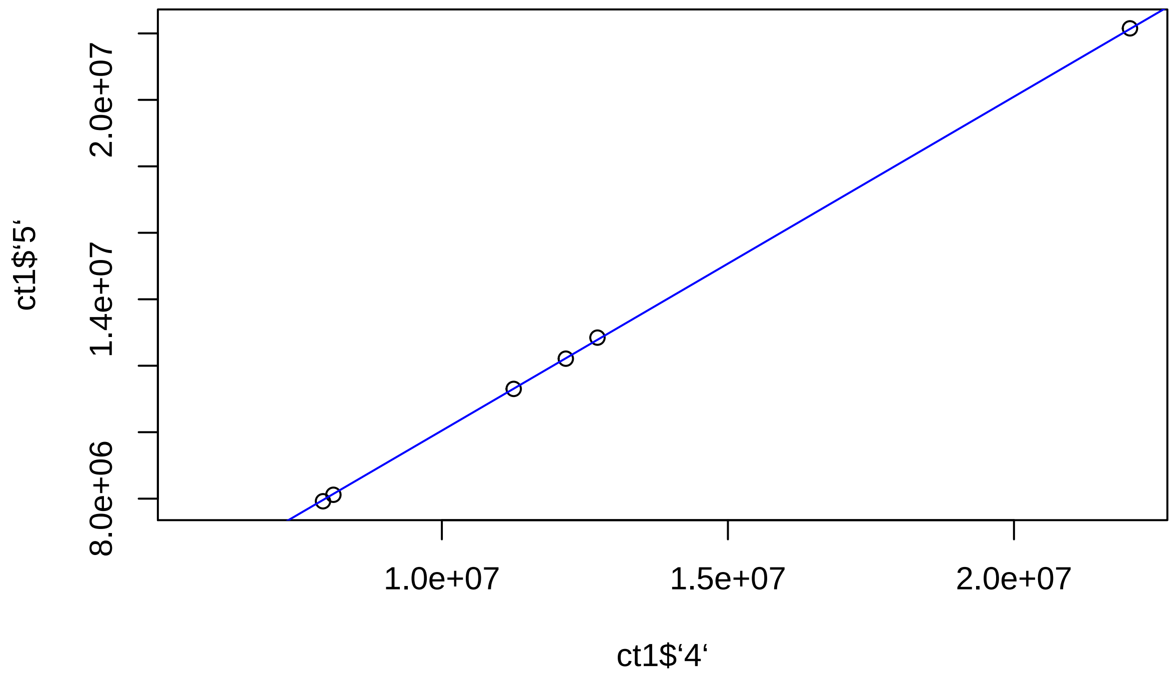

```
plot(ct1$`2`, ct1$`3`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[2]}), co
```



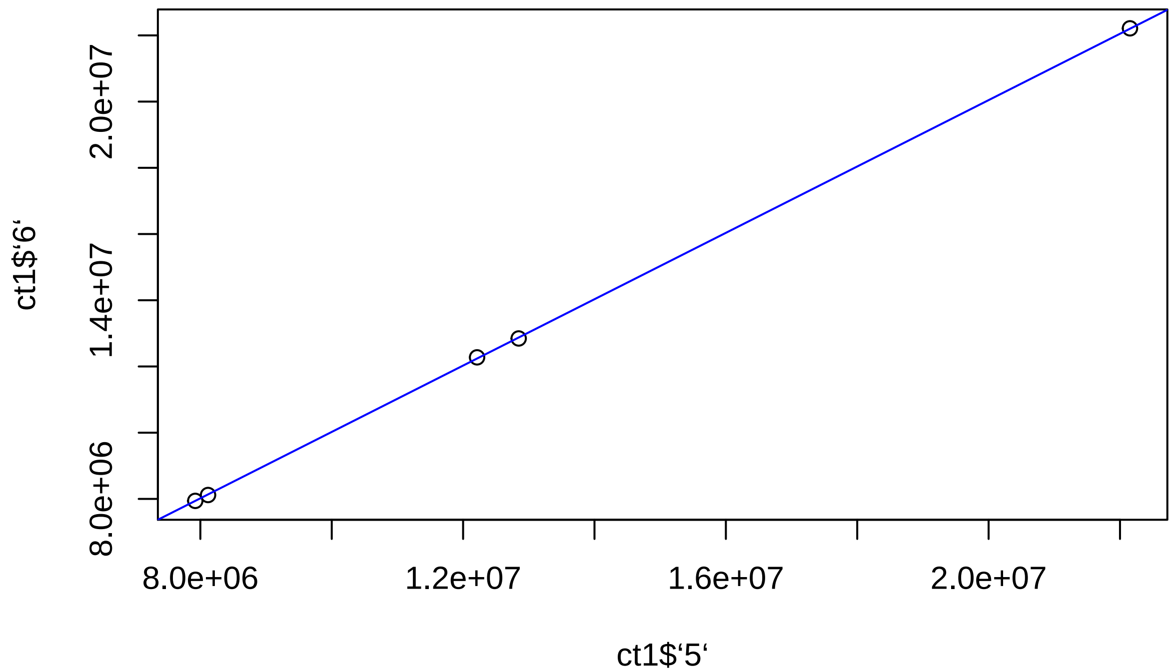

```
plot(ct1$‘3‘, ct1$‘4‘)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[3]}), co
```



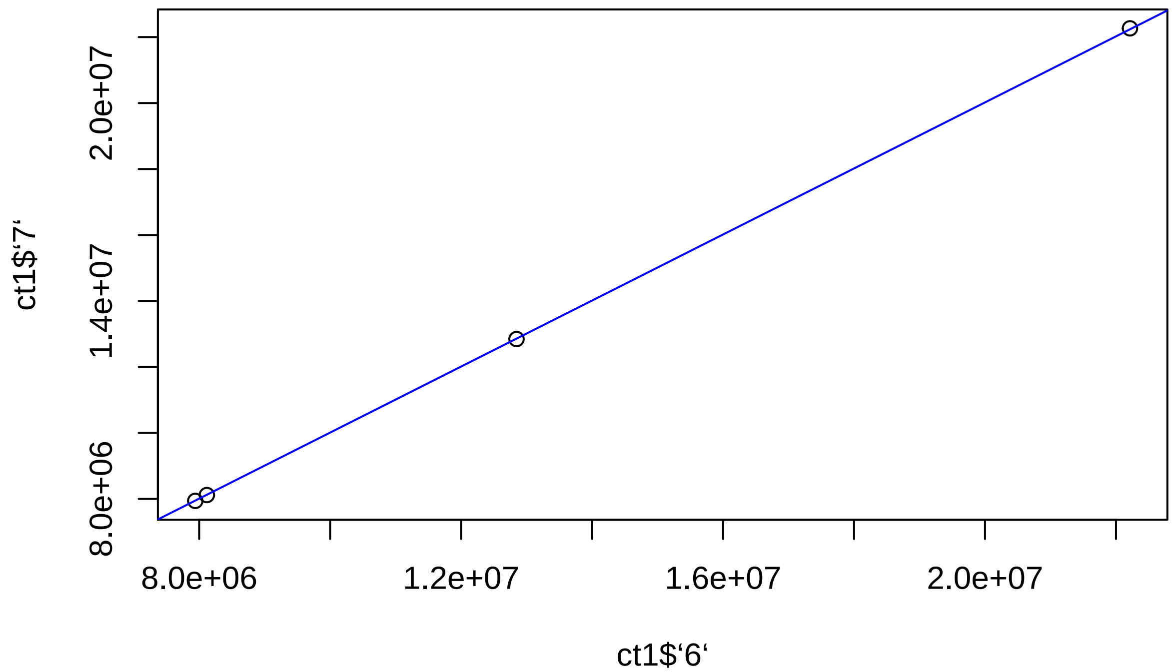

```
plot(ct1$‘4‘, ct1$‘5‘)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[4]}), co
```



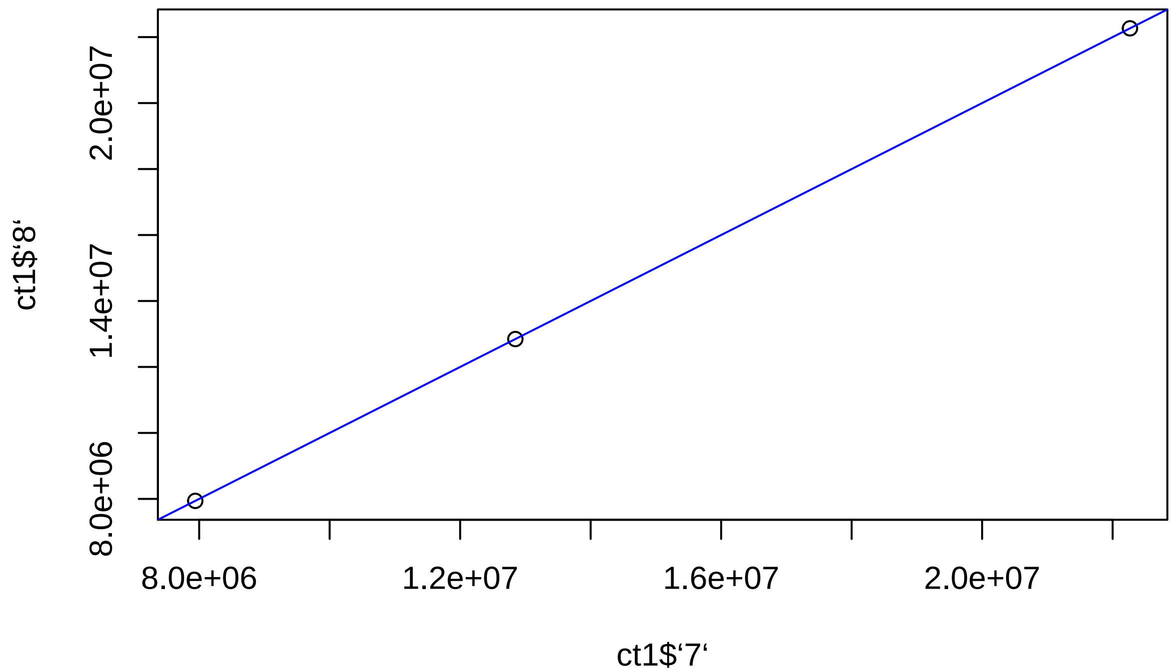

```
plot(ct1$`5`, ct1$`6`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[5]}), co
```



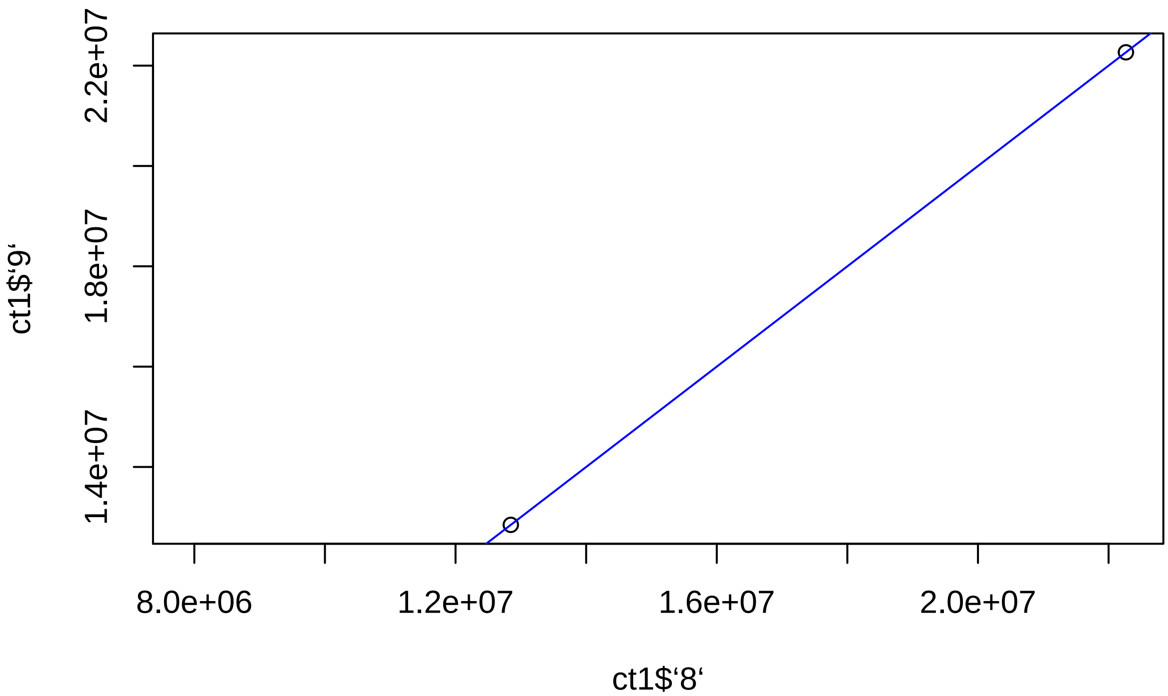

```
plot(ct1$‘6‘, ct1$‘7‘)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[6]}), co
```



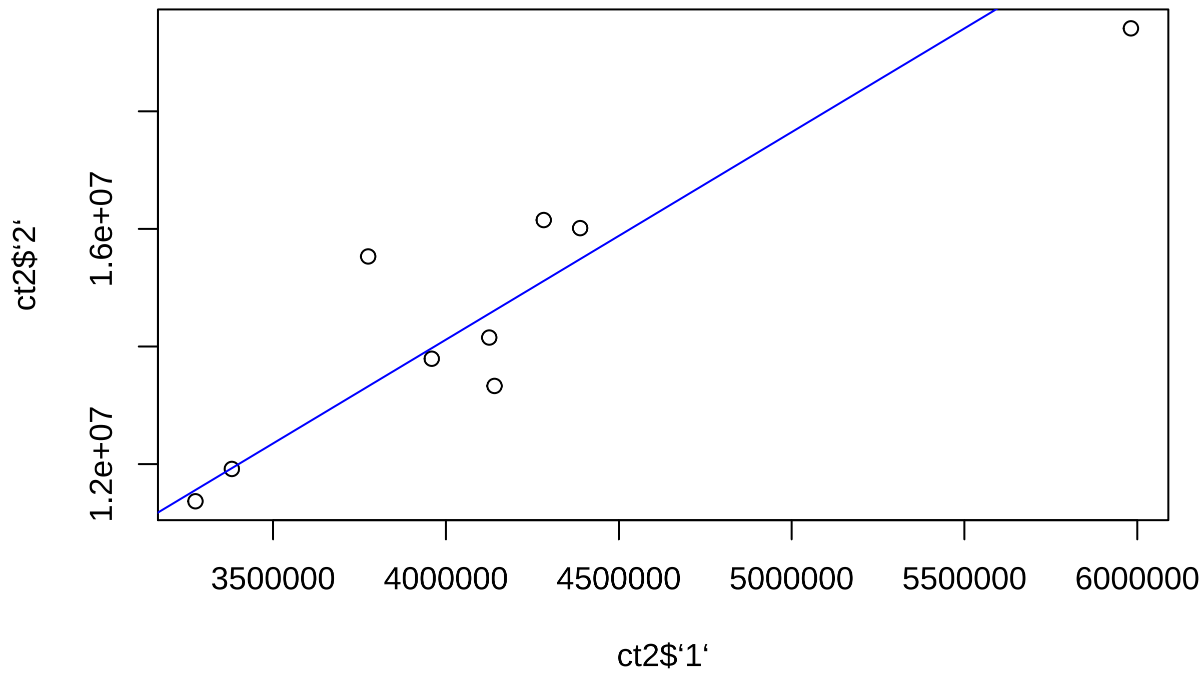

```
plot(ct1$`7`, ct1$`8`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[7]}), c
```



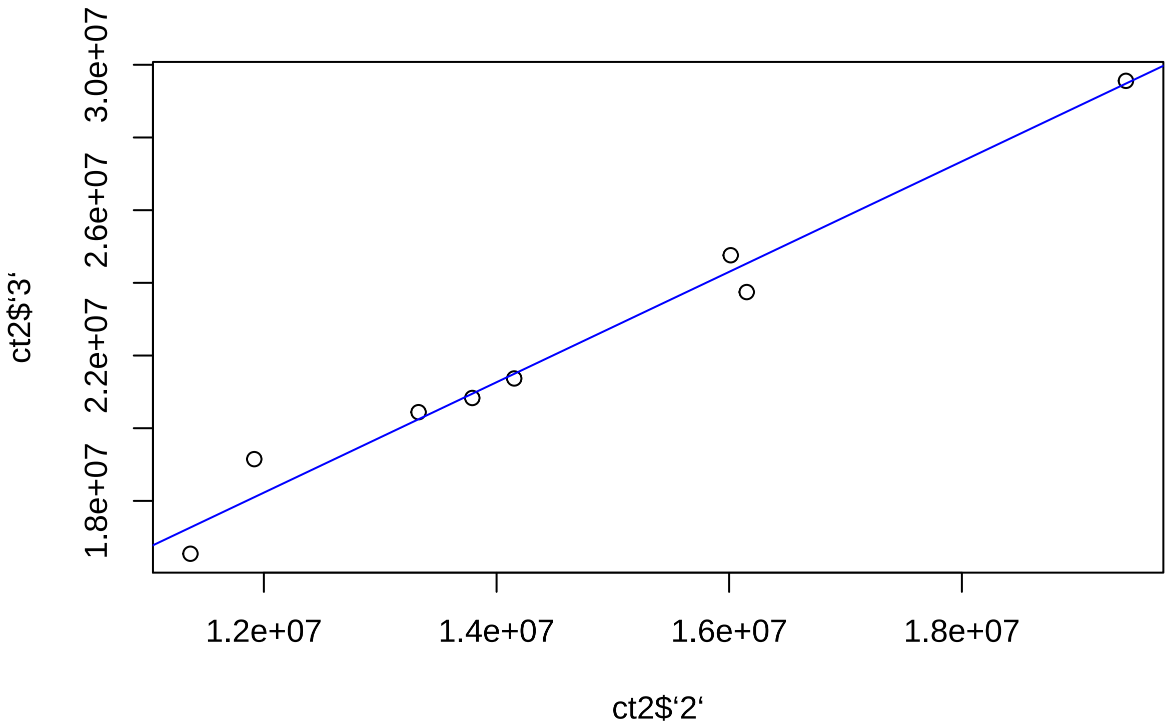

```
plot(ct1$`8`, ct1$`9`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat1[8]}), co
```



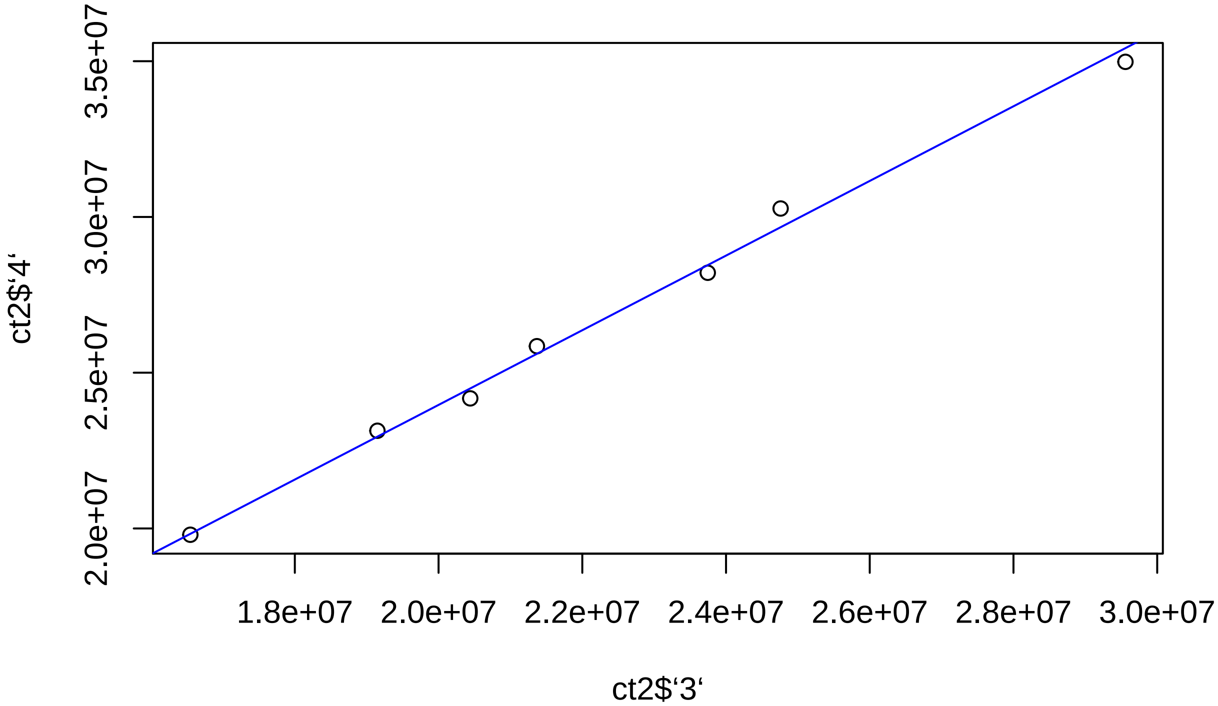

```
#Skoja dem blev faktiskt inte så fula

plot(ct2$`1`, ct2$`2`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[1]}), co
```



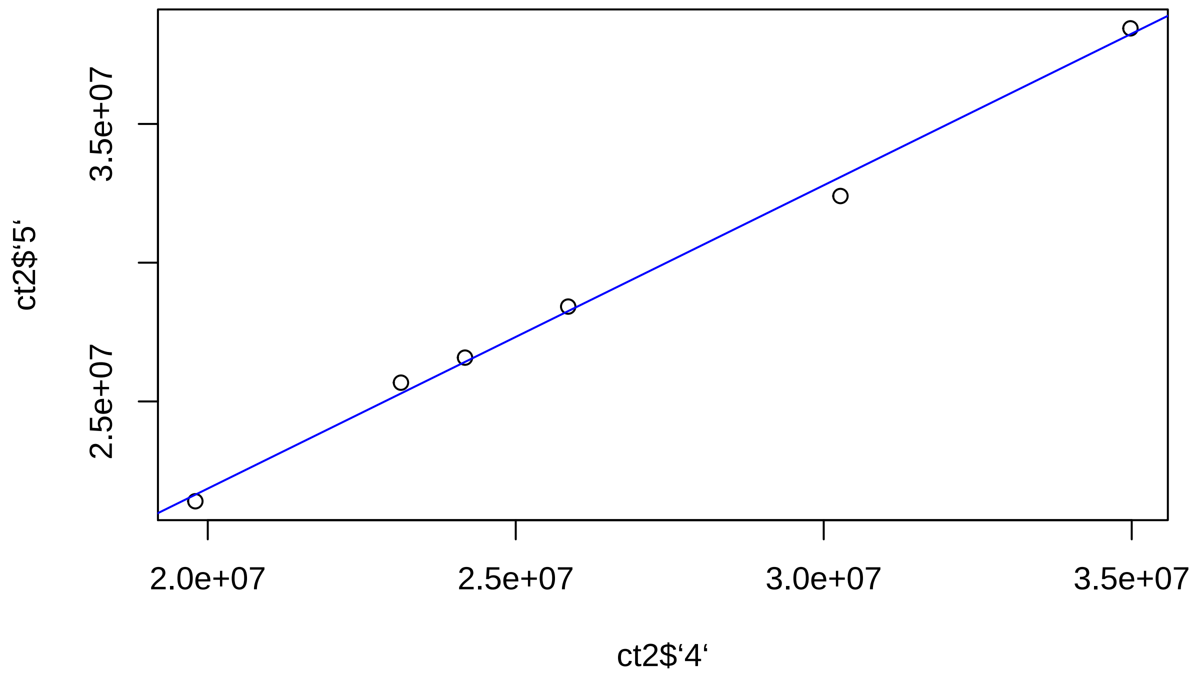

```
plot(ct2$‘2‘, ct2$‘3‘)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[2]}), co
```



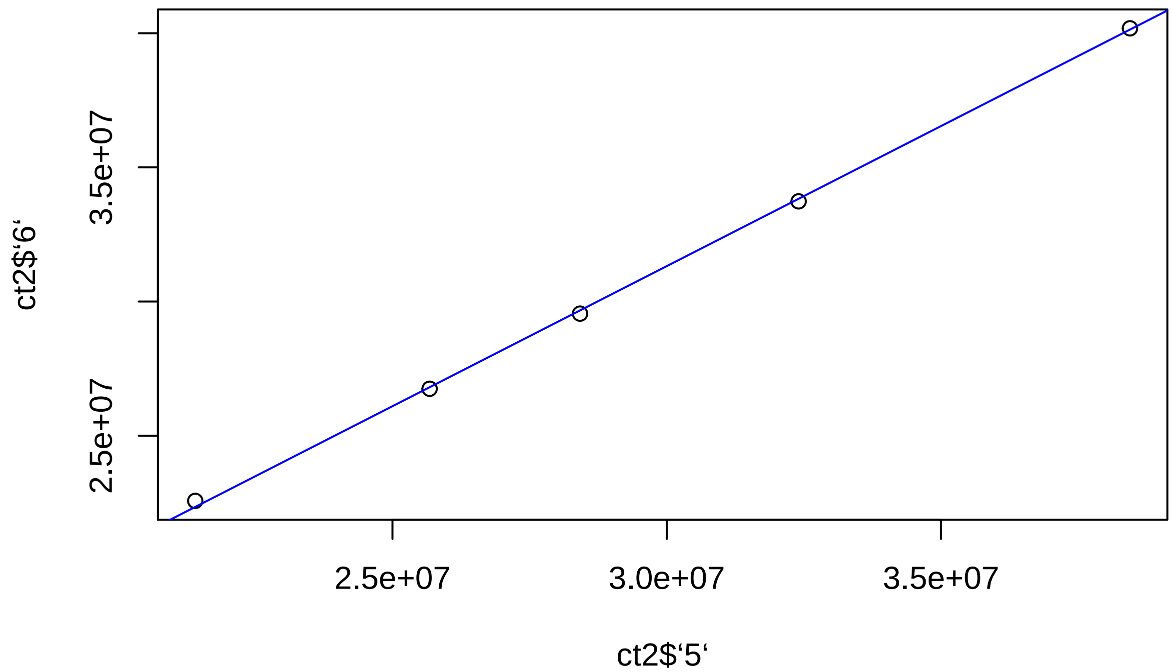

```
plot(ct2$`3`, ct2$`4`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[3]}), co
```



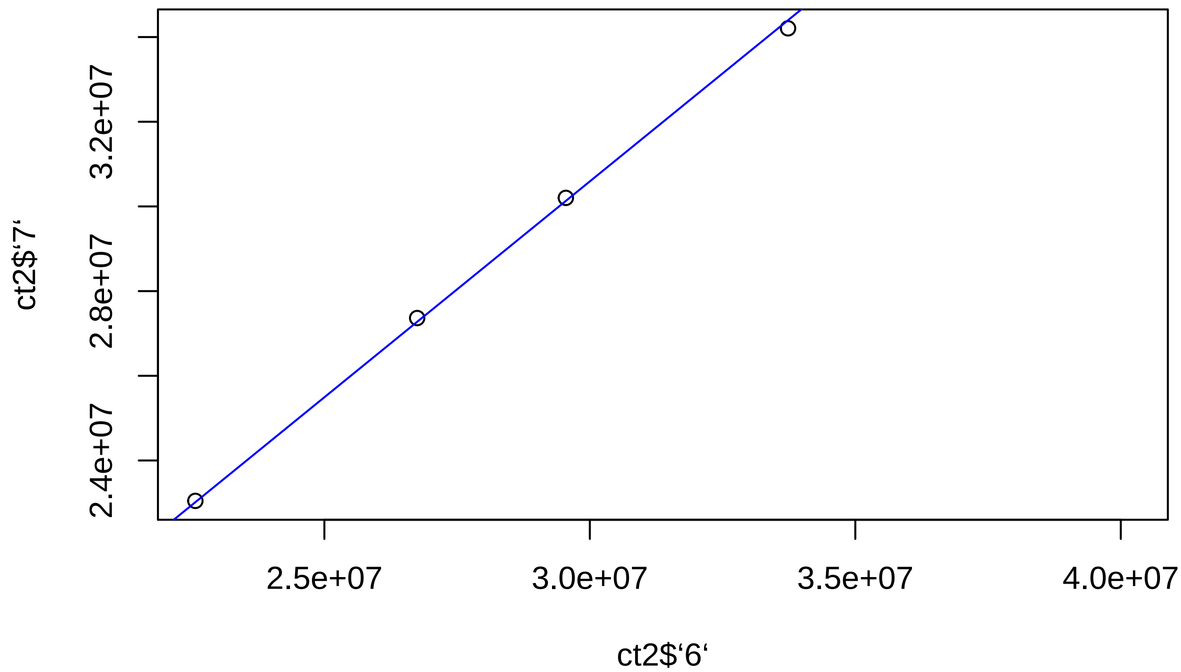

```
plot(ct2$‘4‘, ct2$‘5‘)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[4]}), co
```



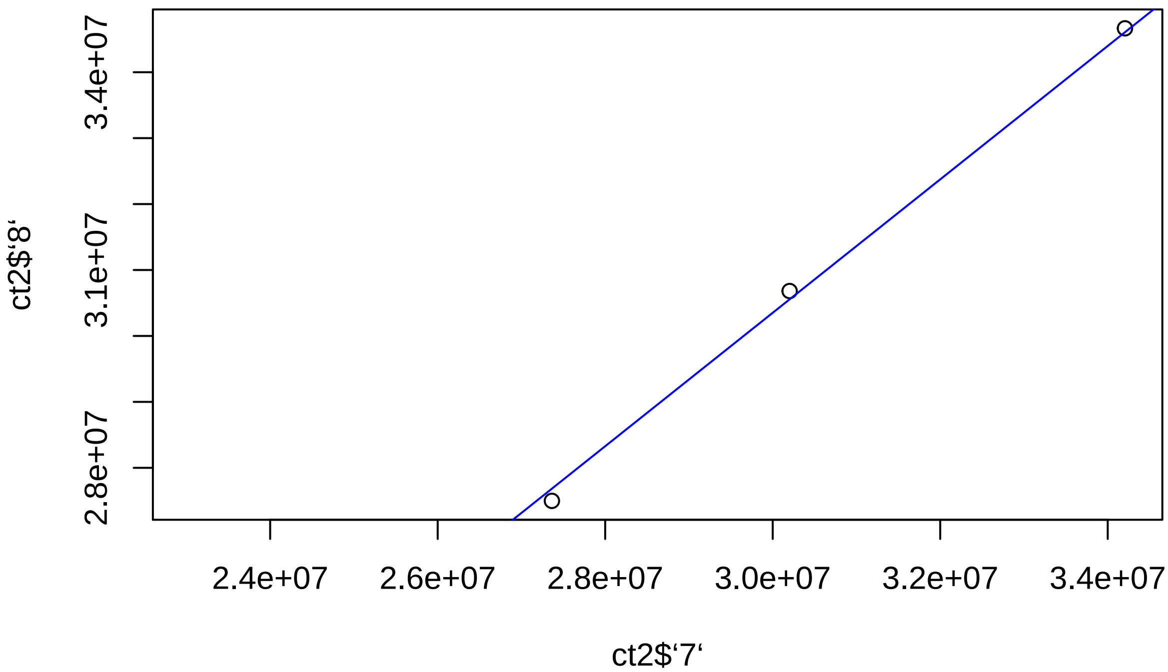

```
plot(ct2$`5`, ct2$`6`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[5]}), co
```



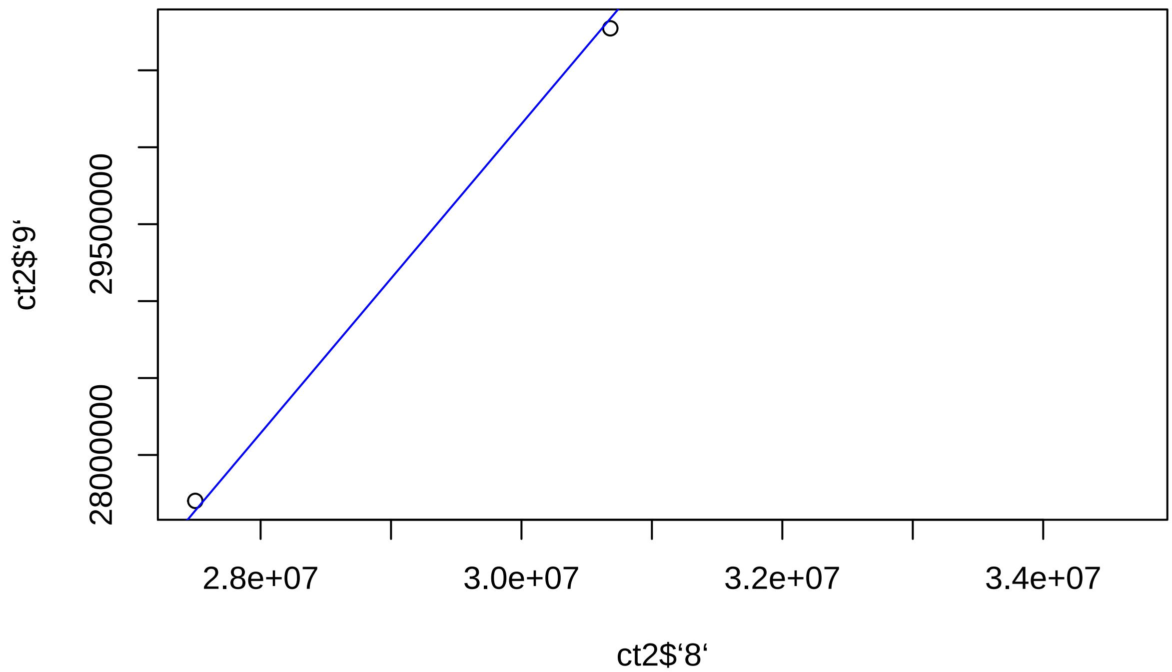

```
plot(ct2$`6`, ct2$`7`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[6]}), co
```




```
plot(ct2$‘7‘, ct2$‘8‘)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[7]}), co
```



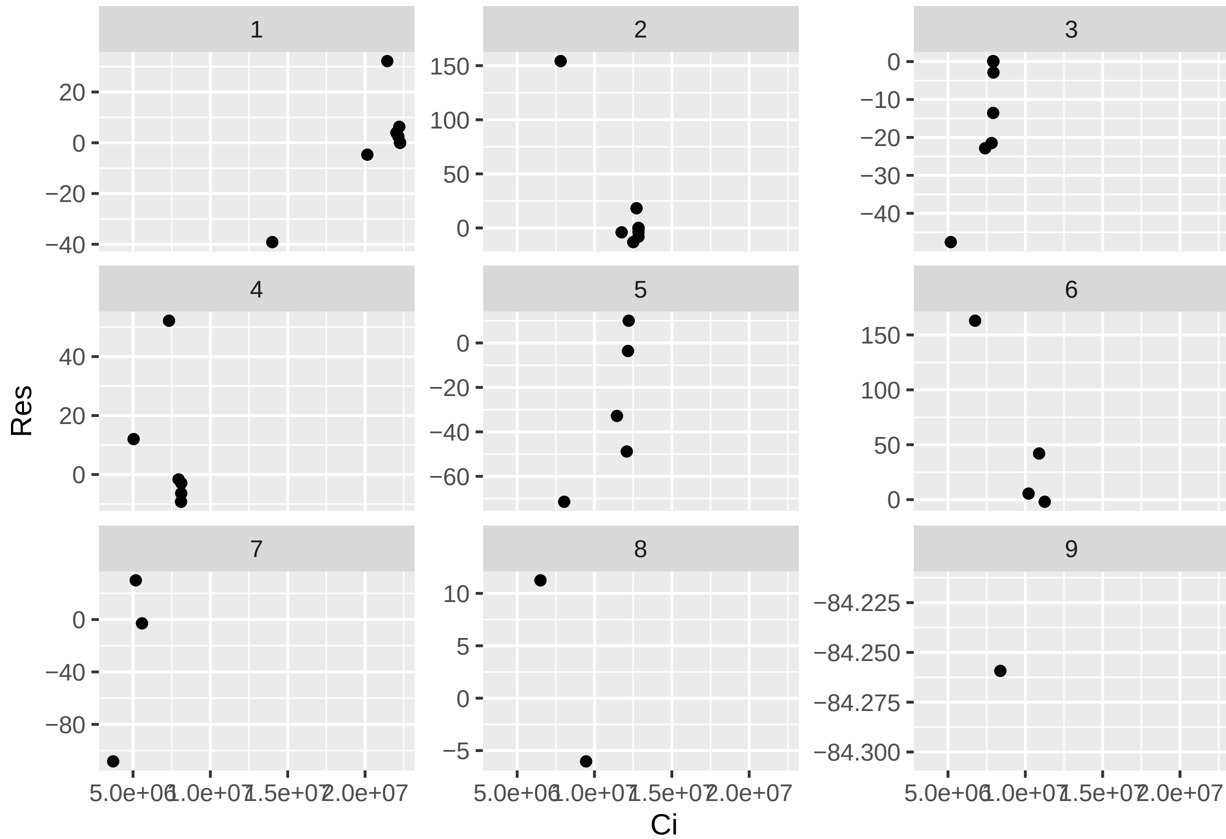

```
plot(ct2$`8`, ct2$`9`)
lines(seq(1e6,7e7, by = 10000), sapply(X = seq(1e6,7e7, by = 10000), FUN = function(x){ x*fHat2[8]}), co
```



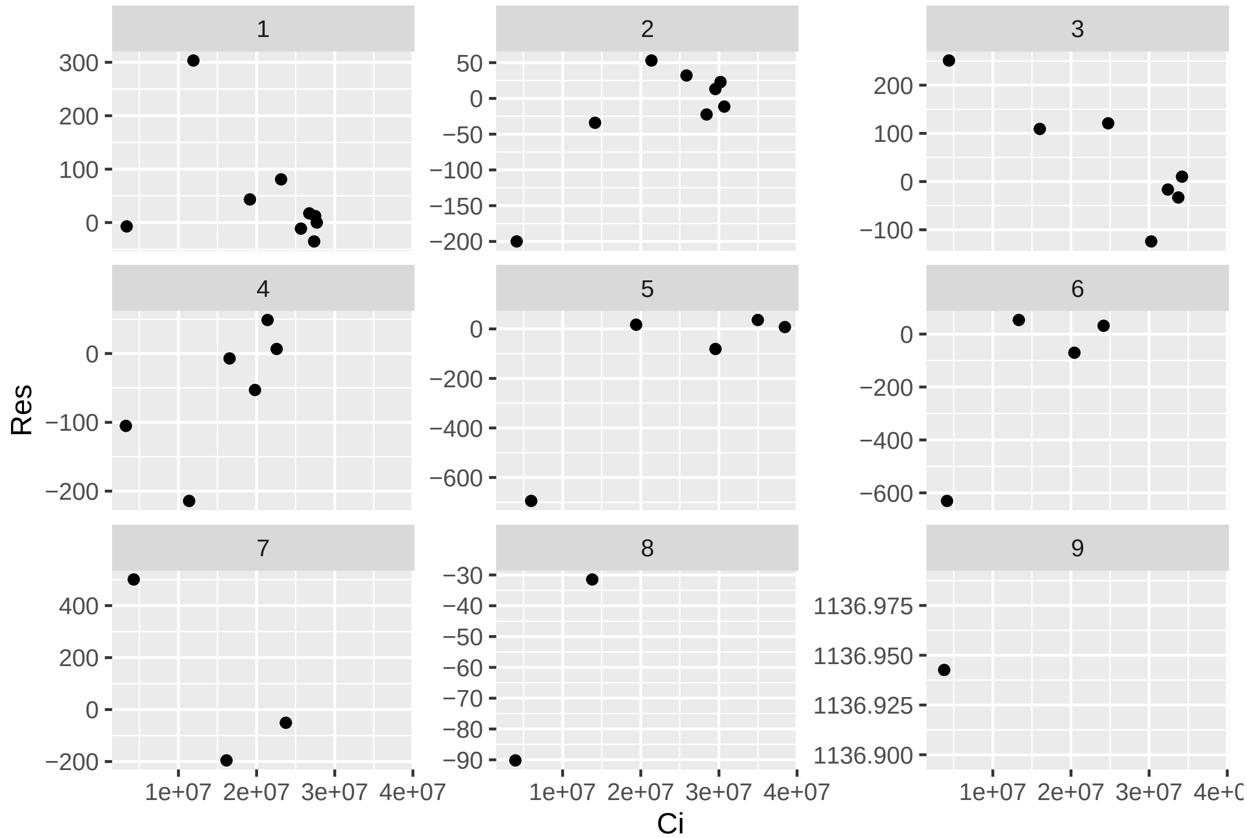
We note that the linear approximation seems to hold. We now want to plot the weighted residuals.

```
residualHelper <- function(ct){

  fHat <- estimateF(ct)

  n <- nrow(ct)

  df <- data.frame(matrix(ncol = 3, nrow = 0))

  for(i in 1:(n-1)){

    for(j in 2:(n-i+1)){

      res <- (ct[i,j] - ct[i,j-1]*fHat[j-1])/sqrt(ct[i,j-1])

      df <- rbind(df, unlist(c(i,unname(res), ct[i,j-1])))

    }

  }

  df

}

resFrame <- residualHelper(ct1) %>%
```

```r
  setNames(c("Year", "Res", "Ci"))

ggplot(resFrame, aes(x = Ci, y = Res)) +
  geom_point() +
  facet_wrap(~Year, scales = "free_y")
```




```r
#Notera att vi bara beaktar år med fler än 6 punkter enl Mack!
#OBS enbart använt vanliga f. Kan vara bra att kolla resterande

resFrame <- residualHelper(ct2) %>%
  setNames(c("Year", "Res", "Ci"))

ggplot(resFrame, aes(x = Ci, y = Res)) +
  geom_point() +
  facet_wrap(~Year, scales = "free_y")
```



All years with more than 6 points, as suggested by Mack, seem to showcase random behaviour and no signs of any systematic deviations.

Lastly we want to examine whether or not we have any calender year effects. An example of a scenario where the assumption of independent years might be violated is if there is an overhaul of the way claims are handled.

```
#Branch 1

n <- nrow(ct1)

fList1 <- list()

for(k in 1:(n-1)){

  fk <- c()

  for(i in 1:(n-k)){

    temp <- unlist(ct1[i, k+1])/unlist(ct1[i,k])

    names(temp) <- k+i-1 #år

    fk <- c(fk, temp)

  }

  fList1[[k]] <- sort(fk)
```

```
}

fList1
```

```
## [[1]]
##        7        9        5        3        1        8        4        2 
## 1.393139 1.420193 1.424097 1.428381 1.438825 1.453696 1.454644 1.504453 
##        6 
## 1.511984 
## 
## [[2]]
##        6        4        9        3        2        7        8        5 
## 1.055239 1.056542 1.062982 1.063773 1.063897 1.066659 1.078068 1.084219 
## 
## [[3]]
##        7        5        4        9        6        3        8 
## 1.006690 1.013050 1.017035 1.019493 1.020142 1.027685 1.033457 
## 
## [[4]]
##        6        7        8        9        4        5 
## 1.000000 1.001564 1.003777 1.004229 1.005655 1.009924 
## 
## [[5]]
##        6        8        7        5        9 
## 1.000000 1.000000 1.002286 1.002772 1.005104 
## 
## [[6]]
##        7        8        9        6 
## 1.000000 1.000000 1.000000 1.002359 
## 
## [[7]]
## 7 8 9 
## 1 1 1 
## 
## [[8]]
## 8 9 
## 1 1 
## 
## [[9]]
## 9 
## 1
```

```
#Branch 2

n <- nrow(ct2)

fList2 <- list()

for(k in 1:(n-1)){

  fk <- c()

  for(i in 1:(n-k)){
```

```r
    temp <- unlist(ct2[i, k+1])/unlist(ct2[i,k])

    names(temp) <- k+i-1 #år

    fk <- c(fk, temp)

  }

  fList2[[k]] <- sort(fk)

}

fList2
```

```
## [[1]]
##        6        5        2        4        8        1        3        7 
## 3.219295 3.245088 3.430653 3.471015 3.483789 3.525179 3.649012 3.771019 
##        9 
## 4.114286 
## 
## [[2]]
##        5        8        3        9        6        7        4        2 
## 1.455413 1.470298 1.509892 1.510468 1.522779 1.533516 1.546228 1.606817 
## 
## [[3]]
##        8        7        9        6        3        4        5 
## 1.182730 1.183407 1.187906 1.196556 1.208211 1.209785 1.222604 
## 
## [[4]]
##        6        7        8        5        9        4 
## 1.070419 1.081089 1.099080 1.099310 1.099386 1.109809 
## 
## [[5]]
##        6        7        5        9        8 
## 1.039753 1.041050 1.041718 1.045179 1.054501 
## 
## [[6]]
##        8        9        7        6 
## 1.014010 1.021094 1.022112 1.023026 
## 
## [[7]]
##        7        9        8 
## 1.004942 1.013429 1.015870 
## 
## [[8]]
##        9        8 
## 1.002993 1.007367 
## 
## [[9]]
##        9 
## 1.005722
```

```r
#Detta blev ju väldigt fult men det får fram budskapet
#Gör en tabell där man räknar kardinaleteten av
```

```r
#sFk och lFk för alla k
```

We might have some seasonal dependence idk??

Exercise 3

From the paper by Mack we have that

$$Var(C_{i,I}) = C_{i,I}^2 \sum_{k=I+1-i}^{I-1} \frac{\sigma_k^2}{f_k^2} \left(\frac{1}{C_{i,k}} + \frac{1}{\sum_{j=1}^{I-k} C_{j,k}} \right)$$

where we estimate σ_k^2 by

$$\hat{\sigma}_k^2 = \frac{1}{I-k-1} \sum_{j=1}^{I-k} C_{j,k} \left(\frac{C_{j,k+1}}{C_{j,k} - \hat{f}_k} \right)^2$$

which is an unbiased estimator.

```r
#WE now need the last 11 years

ct1Alt <- data %>%
  rbind(c(1,10,10,1,1,0)) %>%
  filter(ClaimType == 1) %>%
  filter(ClaimYear >= 10) %>%
  group_by(ClaimYear, DevelopmentYear) %>%
  summarize(Total = sum(ClaimCost)) %>%
  mutate(Total = cumsum(Total)) %>%
  spread(value = Total, key = DevelopmentYear) %>%
  ungroup() %>%
  select(-ClaimYear)

ct2Alt <- data %>%
  rbind(c(2,10,10,1,1,0)) %>%
  filter(ClaimType == 2) %>%
  filter(ClaimYear >= 10) %>%
  group_by(ClaimYear, DevelopmentYear) %>%
  summarize(Total = sum(ClaimCost)) %>%
  mutate(Total = cumsum(Total)) %>%
  spread(value = Total, key = DevelopmentYear) %>%
  ungroup() %>%
  select(-ClaimYear)

estimateSigma <- function(ct){

  n <- ncol(ct)

  r <- nrow(ct)

  C <- fillct(ct)

  fHat <- estimateF(ct)
```

```
  sigmaHat <- c()
  for(k in 1:(n-1)){
    sum <- 0
    for(i in 1:(r-k)){
      sum = sum + C[i,k]*(C[i,k+1]/(C[i,k]) - fHat[k])^2
    }
    sigmaHat = c(sigmaHat, unlist(sum)/(r-k-1)) #Henning sa att I = r (då I = n ger delning med 0)
  }
  sigmaHat
}
sigmaHat1 <- estimateSigma(ct1Alt) #estimates sigma^2 for type 1
sigmaHat2 <- estimateSigma(ct2Alt) #estimates sigma^2 for type 2
riskCalculation <- function(ct, ctAlt){
  n <- nrow(ct)
  C <- fillct(ct)
  fHat <- estimateF(ctAlt)
  sigmaHat <- estimateSigma(ctAlt) #estimates sigma^2
  risks <- c()
  for(i in 2:n){
    sum <- 0
    for(k in (n+1-i):(n-1)){
      sum = sum + sigmaHat[k]/fHat[k]^2*(1/C[i,k] + 1/sum(C[1:(n-k),k]))
    }
    risks <- c(risks, unlist(sum*C[i,n]^2))
  }
  risks
}
```

```r
risks1 <- cbind(2:10,unname(riskCalculation(ct1, ct1Alt)), fullct1[2:10,10]) %>%
  data.frame() %>%
  setNames(c("Year", "Reserve Risk", "Reserve"))

kable(risks1, caption = "Reserve risk for coming years for branch 1")
```

Table 5: Reserve risk for coming years for branch 1

Year	Reserve Risk	Reserve
2	0	12846004
3	0	7940404
4	0	8117003
5	266918718	12287377
6	840882197	11339910
7	1090433733	5735783
8	12032269331	10347649
9	30534063462	13051345
10	145579553687	10176857

```r
risks2 <- cbind(2:10,unname(riskCalculation(ct2, ct2Alt)), fullct2[2:10,10]) %>%
  data.frame() %>%
  setNames(c("Year", "Reserve Risk", "Reserve"))

kable(risks2, caption = "Reserve risk for coming years for branch 1")
```

Table 6: Reserve risk for coming years for branch 1

Year	Reserve Risk	Reserve
2	3.090748e+10	30949061
3	4.879179e+10	35040057
4	5.772316e+10	23567739
5	1.573477e+11	41901376
6	1.826433e+11	28934582
7	4.063649e+11	33564122
8	5.138441e+11	29703135
9	1.862674e+12	33638054
10	8.343128e+12	33310543

```r
prevct1 <- data %>%
  rbind(c(1,1,10,1,1,0)) %>%
  rbind(c(1,10,10,1,1,0)) %>%
  filter(ClaimType == 1) %>%
  group_by(ClaimYear, DevelopmentYear) %>%
  summarize(Total = sum(ClaimCost)) %>%
  mutate(Total = cumsum(Total)) %>%
  spread(value = Total, key = DevelopmentYear) %>%
  ungroup() %>%
  filter(ClaimYear <= 10) %>%
  select(-ClaimYear)
```

```r
prevct2 <- data %>%
  rbind(c(2,1,10,1,1,0)) %>%
  rbind(c(2,10,10,1,1,0)) %>%
  filter(ClaimType == 2) %>%
  group_by(ClaimYear, DevelopmentYear) %>%
  summarize(Total = sum(ClaimCost)) %>%
  mutate(Total = cumsum(Total)) %>%
  spread(value = Total, key = DevelopmentYear) %>%
  ungroup() %>%
  filter(ClaimYear <= 10) %>%
  select(-ClaimYear)

estimateFAlt <- function(ct, known = 0){

  n <- ncol(ct)

  r <- nrow(ct)

  fVec <- c()

  for(i in 1:(n-1)){

    f <- sum(ct[1:min(r-i+known, r),i+1])/sum(ct[1:min(r-i+known, r),i])

    fVec[i] <- f

  }

  fVec

}


fillAlt <- function(ct, known = 0){

  temp <- ct

  n <- ncol(ct)

  fHat <- estimateFAlt(ct, known)

  if(known < (n-1)){

    for(i in (known+2):n){

      for(j in (n-i+known+2):n){

        temp[i,j] <- temp[i,j-1]*fHat[j-1]

      }

    }
```

```
  }

  temp

}

#Branch 1

uc1 <- c()

for(i in 0:8){

  val <- sum(fillAlt(prevct1, i)[,10])

  uc1 <- c(uc1, val)

}

kable(cbind(c(0:8),uc1), caption = "Ultimate claim amounts over observed years for branch 1",
      col.names = c("Year","UCA"))
```

Table 7: Ultimate claim amounts over observed years for branch 1

Year	UCA
0	149073425
1	149623460
2	149699237
3	149687965
4	149613291
5	149669595
6	149659761
7	149652191
8	149652191

```
#Branch 2

uc2 <- c()

for(i in 0:8){

  val <- sum(fillAlt(prevct2, i)[,10])

  uc2 <- c(uc2, val)

}

kable(cbind(c(0:8),uc2), caption = "Ultimate claim amounts over observed years for branch 2",
      col.names = c("Year","UCA"))
```

Table 8: Ultimate claim amounts over observed years for branch 2

Year	UCA
0	286830634
1	289014336
2	287807470
3	289468788
4	288615446
5	289142567
6	289111674
7	288964958
8	288761436

Exercise 4