Computer practical 1: Topics in Statistics III/IV, Term 1

Georgios Karagiannis

Aim

- To become familiar with Iterative Proportional Fitting (IPF) method
- Apply IPF method to produce MLE for the Log Linear models
- To learn how to solve systems of noon-linear equations via Newton method
- Apply Newton method to produce MLE for the Log Linear models
- Extensions of Newton method, and IPF method to to produce MLEs for 4-way and higher order tables

---

Contigency table: data manipulation

Below we load a table where refers to a 1992 survey by the Wright State University School of Medicine and the United Health Services in Dayton, Ohio. The survey asked 2276 students in their final year of high school in a nonurban area near Dayton, Ohio whether they had ever used alcohol, cigarettes, or marijuana. Denote the variables in this $2 \times 2 \times 2$ table by A for alcohol use, C for cigarette use, and M for marijuana use.

Load the observed counts in a data frame `obs.frame` and print the result. Use commands :

- `data.frame()` : as in SC2
- `factor()` : to encode a vector as a factor (aka category)
- `expand.grid()` : to produce all combinations of the supplied vectors or factors.

```
# I will do this for you

## load the data

obs.frame<-data.frame(count=c(911,538,44,456,3,43,2,279),
                      expand.grid(
  marijuana=factor(c("Yes","No"),levels=c("No","Yes")),
  cigarette=factor(c("Yes","No"),levels=c("No","Yes")),
  alcohol=factor(c("Yes","No"),levels=c("No","Yes")))
  )

## orint the obs.frame

obs.frame
```

```
##   count marijuana cigarette alcohol
## 1   911       Yes       Yes     Yes
## 2   538        No       Yes     Yes
## 3    44       Yes        No     Yes
## 4   456        No        No     Yes
```

```
## 5     3       Yes       Yes      No
## 6    43        No       Yes      No
## 7     2       Yes        No      No
## 8   279        No        No      No
```

Create 3 dimentional contingency table from `obs.frame`. Use command:

- `xtabs()`, to create a contingency table from cross-classifying factors in a dara.frame

```
# this is me again
obs.xtabs <- xtabs(count ~ marijuana+cigarette+alcohol, data=obs.frame)
## print
obs.xtabs
```

```
## , , alcohol = No
##
##          cigarette
## marijuana  No Yes
##       No  279  43
##       Yes   2   3
##
## , , alcohol = Yes
##
##          cigarette
## marijuana  No Yes
##       No  456 538
##       Yes  44 911
```

Create a contigency table which contains the row, column, layer, etc. . . margins.

- Use command `addmargins()` with `obs.xtabs`
- Save it in `obs.addmargins`

```
obs.addmargins <- addmargins(obs.xtabs)
obs.addmargins
```

```
## , , alcohol = No
##
##          cigarette
## marijuana   No  Yes  Sum
##       No   279   43  322
##       Yes    2    3    5
##       Sum  281   46  327
##
## , , alcohol = Yes
##
##          cigarette
## marijuana   No  Yes  Sum
##       No   456  538  994
##       Yes   44  911  955
##       Sum  500 1449 1949
##
## , , alcohol = Sum
##
##          cigarette
## marijuana   No  Yes  Sum
##       No   735  581 1316
```

```
##       Yes   46  914  960
##       Sum  781 1495 2276
```

Compute the marginal contigency table of marijuana and cigarette.

- Use command `margin.table( , margin = )` and `obs.xtabs`.
- Save it in `obs.mc.xtabs`.

```
obs.xtabs <- xtabs(count ~ marijuana+cigarette+alcohol, data=obs.frame)
obs.mc.xtabs <- margin.table(obs.xtabs, margin=c(1,2))
obs.mc.xtabs
```

```
##          cigarette
## marijuana  No Yes
##       No  735 581
##       Yes  46 914
```

Compute the (joint) sampling proportions.

- Use command `prop.table()` with `obs.xtabs`.
- Save it in `obs.prop.table`.

```
obs.xtabs <- xtabs(count ~ marijuana+cigarette+alcohol, data=obs.frame)
obs.prop.table <- prop.table(obs.xtabs)
obs.prop.table
```

```
## , , alcohol = No
##
##          cigarette
## marijuana           No          Yes
##       No  0.1225834798 0.0188927944
##       Yes 0.0008787346 0.0013181019
##
## , , alcohol = Yes
##
##          cigarette
## marijuana           No          Yes
##       No  0.2003514938 0.2363796134
##       Yes 0.0193321617 0.4002636204
```

Create a data.frame of proportions.

- Use the command `as.data.frame()` with `obs.prop.table`
- Save it as `obs.prop.frame`

```
obs.prop.frame <- as.data.frame(obs.prop.table)
```

As a homework, you can furhter experiment with commands

- `margin.table` : computing marginal tables
- `prop.table` : computing proportions
- `addmargins` : putting margins on tables;
  - e.g., obs.prop.margin <- addmargins(prop.table(obs.xtabs))

```
#
# Do it later ...
#
```

Odds ratio calculations

Code an R function, named 'odds.ratio' with:

- Inputs:
  - x : a 2 by 2 matrix whose elements are the observed counts of a 2 by 2 contigency table
  - conf.level : with default input value 0.95 representing the confidence level
  - theta0 : with default value 1 representing a null hypothesis value of the odds ratio test
- Outputs:
  - estimator : representing mle of odds satio
  - log.estimator : representing the mle of log odds ratio
  - asympt.SE : representing the standard error / standard deviation of the mle of odds ratio
  - conf.interval: representing confidence interval of mle of odds ratio at sig level conf.level (from the inputs)
  - conf.level =representing confidence level
  - Ztest : representing the test statistic for the odds ratio test (2 tails)
  - p.value : representing the p value of the odds ratio test (2 tails)
  - log.conf.interval : representing in log scale the confidence interval of mle of odds ratio at sig level conf.level (from the inputs)

```
odds.ratio <- function(x,conf.level=0.95,theta0=1)
{
  if (any(x==0)) x <- x+0.5
  theta <- x[1,1] *
    x[2,2]/(x[1,2]  *
              x[2,1])
  SE <- sqrt(sum(1/x))
  Za2 <- qnorm(0.5 *
                 (1+conf.level))
  Low <- exp(log(theta)-Za2 * SE)
  Up <- exp(log(theta)+Za2  *
              SE)
  CI <- c(Low,Up)
  Z=(log(theta)-log(theta0))/SE
  pv=2   *
    pnorm(-abs(Z))

  logCI <- log(CI)

  list (estimator=theta,
        log.estimator=log(theta),
        asympt.SE=SE,
        conf.interval=CI,
        conf.level=conf.level,
        Ztest=Z,
        p.value=pv,
        log.conf.interval=logCI
```

```
        )
}
```

For the marginal contigency table of marijuana and cigarette, * compute the mle of the marginal odds ratio of marijuana and cigarette * computet the 95% Confidence Interval of the marginal odds ratio of marijuana and cigarette * perform a statiastical hypothesis test that marijuana and cigarette are independent at sig level 0.05

```
obs.xtabs <- xtabs(count ~ marijuana+cigarette+alcohol, data=obs.frame)

obs.mc.xtabs <- margin.table(obs.xtabs, margin=c(1,2))

odds.ratio.marijuana.cigarette <- odds.ratio(obs.mc.xtabs, conf.level=0.95, theta0=1 )
odds.ratio.marijuana.cigarette
```

```
## $estimator
## [1] 25.1362
##
## $log.estimator
## [1] 3.224309
##
## $asympt.SE
## [1] 0.1609812
##
## $conf.interval
## [1] 18.33463 34.46093
##
## $conf.level
## [1] 0.95
##
## $Ztest
## [1] 20.02911
##
## $p.value
## [1] 3.071215e-89
##
## $log.conf.interval
## [1] 2.908792 3.539826
```

The MLE of the marginal odds ratio of marijuana and cigarette is 25.136197 .

The 95% confidence interval of the marginal odds ratio of marijuana and cigarette is [18.33463, 34.4609298] .

The hypothesis test with $H_0 : \theta = 1$ versus $H_1 : \theta \neq 1$ at sig. level 0.05 has a p-value $3.0712155 \times 10^{-89}$ Hence I reject $H_0 : \theta = 1$ against $H_1 : \theta \neq 1$ at sig. level 0.05.

Fourfold Plots

You can draw Fourfold Plots

Tables 2 x 2

It is a graphical expression visualizing the odds ratio

$$\theta = \frac{n_{11}n_{22}}{n_{12}n_{21}}$$

in 2 x 2 contingency tables.

It shows the departure from independence as measured by the sample odds ratio,

Each cell n_{ij} is represented as a quarter-circle with radius proportional to $\sqrt{n_{ij}}$ and area proportional to n_{ij}.

- If there is no association $\theta = 1$ between classification variables, the quarter-circles should form a circle.
- If there is positive association $\theta > 1$ between classification variables, the diagonal areas are greater than the off-diagonal areas
- If there is negative association $\theta < 1$ between classification variables, the diagonal areas are smaller than the off-diagonal areas

R provides a function to draw this kind of plots by using the function `fourfoldplot' from the package`vcd'

- Install 'vcd' package and load it

```
# install.packages('vcd') # uncomment it if you have not installed package vcd
library(vcd)
```

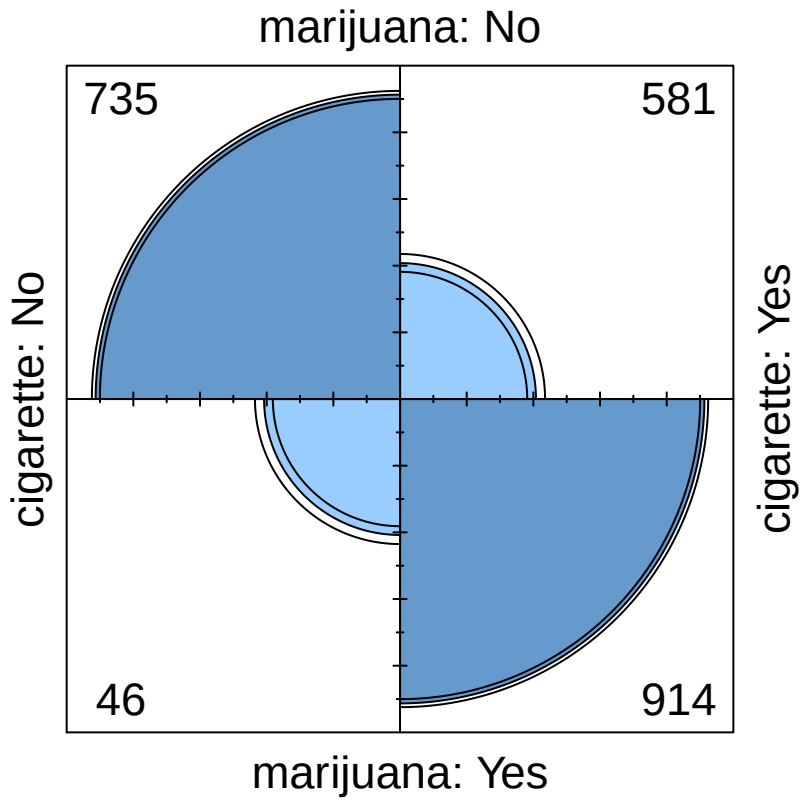
```
## Loading required package: grid
```

Check in help the function fourfoldplot by usign the command '?fourfoldplot'

- Draw a Fourfold Plot for the marginal contigency table of marijuana and cigarette.
- Discuss what you can see

```
obs.mc.xtabs <- margin.table(obs.xtabs, margin=c(1,2))

fourfoldplot(obs.mc.xtabs)
```



Note that:
* The area of each shaded quadrant shows the observed counts.
* Circular arcs show the limits of confidence interval for the odds ratio.

Tables 2 x 2 x K

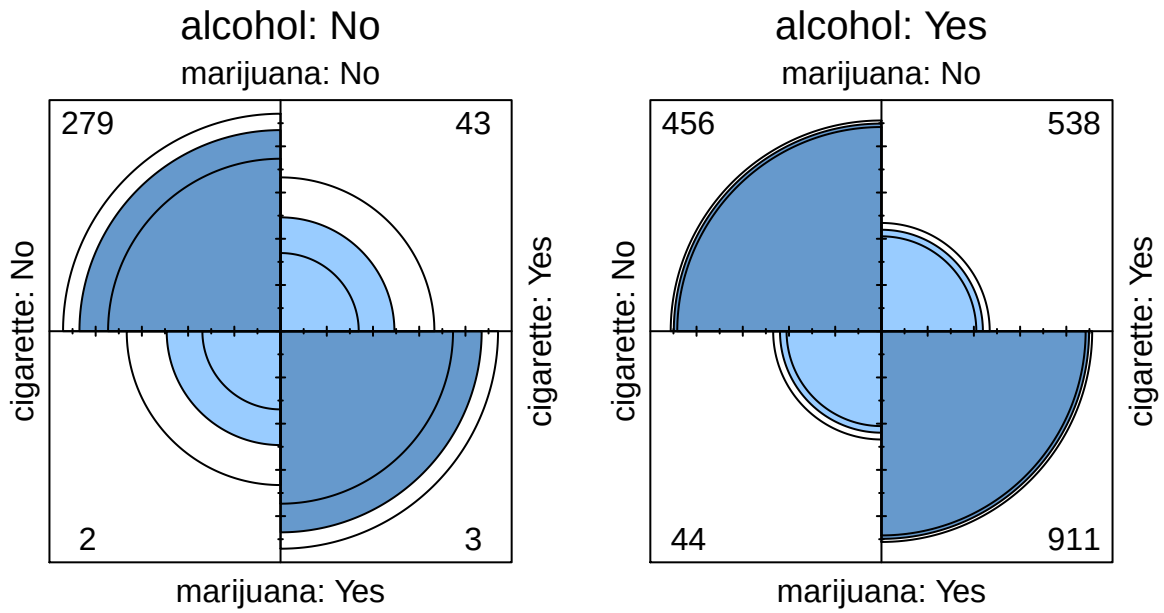
Fourfold Plots can be also used for 2 x 2 x K contigancy tables

- Draw a Fourfold Plot for the contigency table of marijuana cigarette, and alcohol by controlling on the alcohol levels.

- inspect the plots

```
obs.xtabs <- xtabs(count ~ marijuana+cigarette+alcohol, data=obs.frame)

fourfoldplot(obs.xtabs,
             mfrow = c(1,2)
             )
```



Mosaic plot

Mosaic plot display graphically the cells of a contingency table as rectangular areas of size proportional to the corresponding observed frequencies.

When the classification variables are independent the areas tend to be perfectly aligned in rows and columns.

The greater the deviation is, the worse the aforesaid alignment is.

Furthermore, specific locations of the table that deviate from independence the most can be identified and thus the pattern of underlying association can be explained.

The strength of individual cells contribution to divergence from independence as well as the direction of the divergence are reflected in the magnitude and sign of the corresponding independence model's residuals that can be incorporated in a mosaic plot.

R provides a function to draw this kind of plots by using the function `mosaic' from the package`vcd'

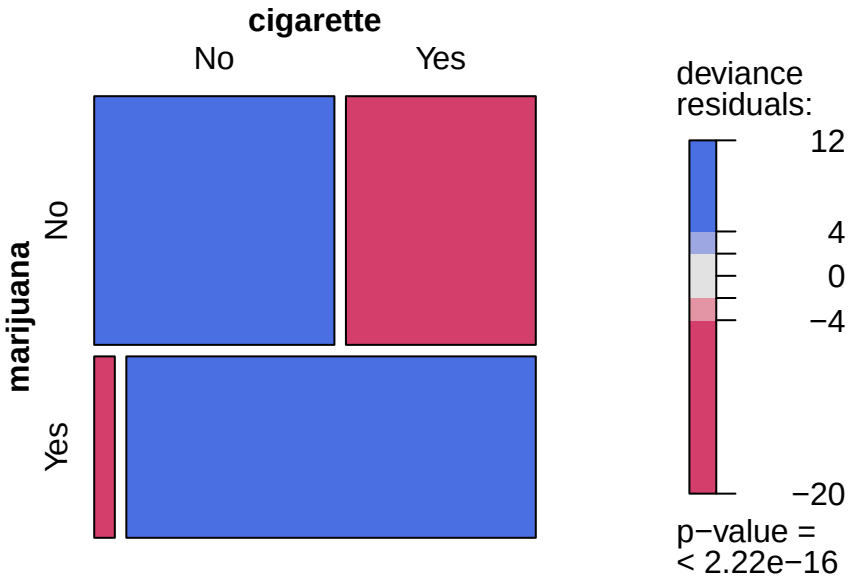
- Install 'vcd' package and load it

```
#install.packages('vcd') # uncomment it if you have not installed package vcd
library(vcd)
```

- Check the command `mosaic' in help by typing`?mosaic'

For the I x J case: * Draw a Mosaic Plot for the marginal contigency table of marijuana cigarette.
* in particular use use mosaic(x,residuals_type="deviance",gp=shading_hcl) where x is the contigency table of interest
* Interpretet the plots

```
mosaic(obs.mc.xtabs,
       residuals_type="deviance",
       gp=shading_hcl)
```

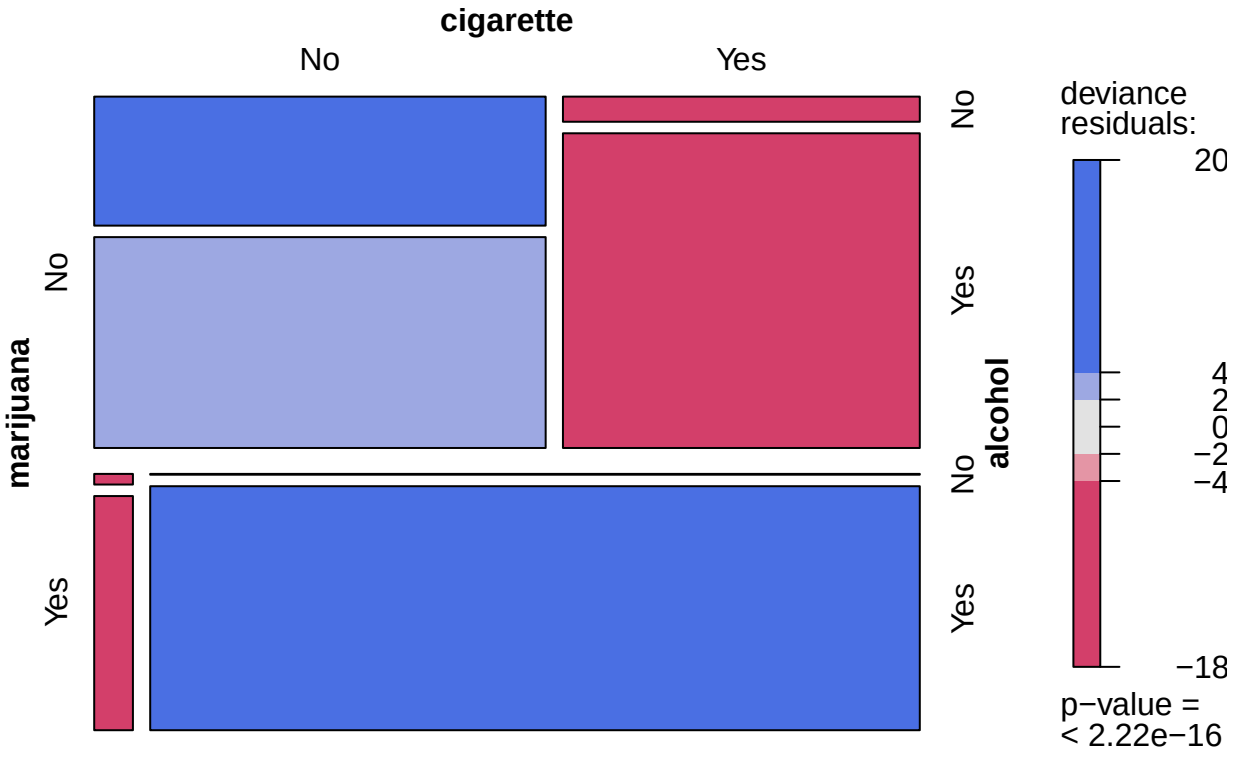


* I observe a substantial deviation from the Independent association.

- The p-value of the GoF test based on the deviance for independence model is below the colorbar, and smaller than 0.05; hence I reject the hipothesis at sig. level 5%.

  For the I x J x K case: * Draw a Mosaic Plot for the contigency table of marijuana cigarette and alcohol. * Interpretet the plots

```
obs.xtabs <- xtabs(count ~ marijuana+cigarette+alcohol, data=obs.frame)
mosaic(obs.xtabs,
       residuals_type="deviance",
       gp=shading_hcl)
```



- I observe a substantial deviation from the Independent association.
- The p-value of the GoF test based on the deviance for independence model is below the colorbar, and smaller than 0.05; hence I reject the hypothesis at sig. level 5%.

The Iterative Proportions Fitting method

The iterative proportional fitting (IPF) algorithm is a simple method for calculating μ_{ijk} for log-linear models.

The main idea of the procedure is the following:

- Start with $\mu^{(0)}_{ijk...}$ satisfying a model no more complex than the one being fitted. E.g., $\mu^{(0)}_{ijk...} = 1.0$ should be ok.
- For $t = 1, ...,$
  - adjust $\mu^{(t)}_{ijk}$ to match by multiplying each marginal table in the set of minimal sufficient statistics, by appropriate factors
  - escape the loop, when the maximum difference between the sufficient statistics and their fitted values is sufficiently close to zero.

Illustration:

Consider 3-way, $I \times J \times K$ tables, and with classifiers X, Y,Z.

Given the model (XY, XZ, YZ) design a IPF recursion producing estimates for μ_{ijk}'s

Steps:

- Compute the minimal sufficient statistics are $\{n_{ij+}\}$, $\{n_{i+k}\}$, $\{n_{+jk}\}$.
- Assume that the approximated μ_{ijk}'s from the $(t-1)$-th cycle is $\mu^{(t-1)}_{ijk}$.
- Then the t-th cycle of the IPF algorithm has the following steps:
  - Set $m^{(0)}_{ijk} = \mu^{(t-1)}_{ijk}$
  - Compute

$$m^{(1)}_{ijk} = m^{(0)}_{ijk} \frac{n_{ij+}}{m^{(0)}_{ij+}};\ \forall i, j, k$$
$$m^{(2)}_{ijk} = m^{(1)}_{ijk} \frac{n_{i+k}}{m^{(1)}_{i+k}};\ \forall i, j, k$$
$$m^{(3)}_{ijk} = m^{(2)}_{ijk} \frac{n_{+jk}}{m^{(2)}_{+jk}};\ \forall i, j, k$$

  - Set $\mu^{(t)}_{ijk} = m^{(3)}_{ijk}$, $\forall i, j, k$

... and produces $\mu^{(t)}_{ijk}$ as approximation.

Example

We use the Alcohol, Cigarette, and Marijuana data-set

```
# I will do this for you

## load the data
```

```
obs.frame<-data.frame(count=c(911,538,44,456,3,43,2,279),
                      expand.grid(
  marijuana=factor(c("Yes","No"),levels=c("No","Yes")),
  cigarette=factor(c("Yes","No"),levels=c("No","Yes")),
  alcohol=factor(c("Yes","No"),levels=c("No","Yes")))
  )

## orint the obs.frame

obs.frame
```

```
##   count marijuana cigarette alcohol
## 1   911       Yes       Yes     Yes
## 2   538        No       Yes     Yes
## 3    44       Yes        No     Yes
## 4   456        No        No     Yes
## 5     3       Yes       Yes      No
## 6    43        No       Yes      No
## 7     2       Yes        No      No
## 8   279        No        No      No
```

- Consider the Log-linear model describing a homogeneous association between each pair of variables at each level of the third one; i.e. $[XY, XZ, YZ]$
- Find the fitted values for $\mu_{i,j,k}$, by using your own code
- Check your results with R function 'loglin{stats}'

Solution

```
# Step 1 :  compute and save the minimal statistics

obs.xtab <- xtabs(obs.frame)
n_xy <- margin.table(obs.xtab,c(1,2))
n_xz <- margin.table(obs.xtab,c(1,3))
n_yz <- margin.table(obs.xtab,c(2,3))

# Step 2: Create a seed for the fitted mu's

# seed the mu_opt
mu_opt <- obs.xtab
for(i in 1:2)
  for(j in 1:2)
    for(k in 1:2)
      mu_opt[i,j,k] = 1.0

# Step 3: Perform the loop to approximate the mu's

for (t in 1: 100) {

  mu_xy <- margin.table(mu_opt,c(1,2))
for(i in 1:2)
  for(j in 1:2)
    for(k in 1:2)
      mu_opt[i,j,k] <- mu_opt[i,j,k]*n_xy[i,j]/mu_xy[i,j]
```

```
  mu_xz <- margin.table(mu_opt,c(1,3))

for(i in 1:2)
  for(j in 1:2)
    for(k in 1:2)
      mu_opt[i,j,k] <- mu_opt[i,j,k]*n_xz[i,k]/mu_xz[i,k]

  mu_yz <- margin.table(mu_opt,c(2,3))
for(i in 1:2)
  for(j in 1:2)
    for(k in 1:2)
      mu_opt[i,j,k] <- mu_opt[i,j,k]*n_yz[j,k]/mu_yz[j,k]
}

mu_opt
```

```
## , , alcohol = No
##
##          cigarette
## marijuana        No       Yes
##       No  279.61683  42.38317
##       Yes   1.38317   3.61683
##
## , , alcohol = Yes
##
##          cigarette
## marijuana        No       Yes
##       No  455.38317 538.61683
##       Yes  44.61683 910.38317
```

```
# CHECK WITH COMMAND loglin
obs.xtab <- xtabs(obs.frame)
loglin(obs.xtab,
         list(
           c(1,2),
           c(1,3),
           c(2,3)
           ),
       fit = TRUE)$fit
```

```
## 4 iterations: deviation 0.09033715
```

```
## , , alcohol = No
##
##          cigarette
## marijuana         No        Yes
##       No  279.616949  42.384216
##       Yes   1.383051   3.615784
##
## , , alcohol = Yes
##
##          cigarette
## marijuana         No        Yes
##       No  455.375087 538.626718
##       Yes  44.624913 910.373282
```

Present the fitted μ's as a data frame

```
as.data.frame(mu_opt)
```

```
##   marijuana cigarette alcohol      Freq
## 1        No        No      No 279.61683
## 2       Yes        No      No   1.38317
## 3        No       Yes      No  42.38317
## 4       Yes       Yes      No   3.61683
## 5        No        No     Yes 455.38317
## 6       Yes        No     Yes  44.61683
## 7        No       Yes     Yes 538.61683
## 8       Yes       Yes     Yes 910.38317
```

You can double check your result with the output of the R package

```
obs.xtab <- xtabs(obs.frame)
library(MASS)
fitAC.AM.CM<-loglm( count~alcohol*cigarette+ alcohol*marijuana +cigarette*marijuana,
              data=obs.xtab,
              param=T,
              fit=T)
fit.array<- fitted(fitAC.AM.CM)
fit.array
```

```
## , , alcohol = No
##
##          cigarette
## marijuana        No       Yes
##       No  279.61440 42.383882
##       Yes   1.38316  3.616919
##
## , , alcohol = Yes
##
##          cigarette
## marijuana        No      Yes
##       No  455.38560 538.6161
##       Yes  44.61684 910.3831
```

```
as.data.frame(fit.array)
```

```
##         No.No    Yes.No    No.Yes  Yes.Yes
## No  279.61440 42.383882 455.38560 538.6161
## Yes   1.38316  3.616919  44.61684 910.3831
```

Example

- Consider the Log-linear model describing mutual independence; i.e. $[X, Y, Z]$

1. Write your own code to compute the fitted values for $\mu_{i,j,k}$
2. Check your ewsults with the R command 'loglin{stats}'

Solution

```
# Step 1 :  compute and save the minimal statistics

obs.xtab <- xtabs(obs.frame)
```

```r
n_x <- margin.table(obs.xtab,c(1))
n_y <- margin.table(obs.xtab,c(2))
n_z <- margin.table(obs.xtab,c(3))

# Step 2: Create a seed for the fitted mu's

# seed the mu_opt
mu_opt <- obs.xtab
for(i in 1:2)
  for(j in 1:2)
    for(k in 1:2)
      mu_opt[i,j,k] = 1.0

# Step 3: Perform the loop to approximate the mu's

for (t in 1: 100) {

  mu_x <- margin.table(mu_opt,c(1))
  for(i in 1:2)
    for(j in 1:2)
      for(k in 1:2)
      mu_opt[i,j,k] <- mu_opt[i,j,k]*n_x[i]/mu_x[i]

  mu_y <- margin.table(mu_opt,c(2))
  for(i in 1:2)
    for(j in 1:2)
      for(k in 1:2)
      mu_opt[i,j,k] <- mu_opt[i,j,k]*n_y[j]/mu_y[j]

  mu_z <- margin.table(mu_opt,c(3))
  for(i in 1:2)
    for(j in 1:2)
      for(k in 1:2)
      mu_opt[i,j,k] <- mu_opt[i,j,k]*n_z[k]/mu_z[k]
}


mu_opt
```

```
## , , alcohol = No
##
##          cigarette
## marijuana        No       Yes
##       No   64.87990 124.19392
##       Yes  47.32880  90.59739
##
## , , alcohol = Yes
##
##          cigarette
## marijuana        No       Yes
##       No  386.70007 740.22612
##       Yes 282.09123 539.98258
```

```
# CHECK WITH COMMAND loglin
obs.xtab <- xtabs(obs.frame)
loglin(obs.xtab,
         list(
           c(1),
           c(2),
           c(3)
           ),
       fit = TRUE)$fit
```

```
## 2 iterations: deviation 0
```

```
## , , alcohol = No
##
##          cigarette
## marijuana        No       Yes
##       No   64.87990 124.19392
##       Yes  47.32880  90.59739
##
## , , alcohol = Yes
##
##          cigarette
## marijuana        No       Yes
##       No  386.70007 740.22612
##       Yes 282.09123 539.98258
```

Example (Homework)

The table below, summarises summarizes observations of 68,694 passengers in autos and light trucks involved in accidents in the state of Maine in 1991. The table classifies passengers by gender (G), location of accident (Z), seat-belt use (S), and injury (I). The Table reports the sample proportion of passengers who were injured. For each GL combination, the proportion of injuries was about halved for passengers wearing seat belts.

```
# load dataset
obs.frame.accident<-data.frame(
  count=c(7287,11587,3246,6134,10381,10969,6123, 6693,996, 759, 973, 757, 812, 380, 1084, 513) ,
  expand.grid(
    belt=c("No","Yes"),
    location=c("Urban","Rural"),
    gender=c("Female","Male"),
    injury=c("No","Yes"))
)
#print dataset
obs.frame.accident
```

```
##    count belt location gender injury
## 1   7287   No    Urban Female     No
## 2  11587  Yes    Urban Female     No
## 3   3246   No    Rural Female     No
## 4   6134  Yes    Rural Female     No
## 5  10381   No    Urban   Male     No
## 6  10969  Yes    Urban   Male     No
## 7   6123   No    Rural   Male     No
## 8   6693  Yes    Rural   Male     No
## 9    996   No    Urban Female    Yes
## 10   759  Yes    Urban Female    Yes
```

```
## 11   973   No    Rural Female    Yes
## 12   757  Yes    Rural Female    Yes
## 13   812   No    Urban   Male    Yes
## 14   380  Yes    Urban   Male    Yes
## 15  1084   No    Rural   Male    Yes
## 16   513  Yes    Rural   Male    Yes
```

Consider the Homogeneity association model (GZ, GS, GI, ZS, ZI, SI),

1. Write your own code to compute the fitted values for $\mu_{i,j,k,c}$
2. Check your ewsults with the R command 'loglin{stats}'

Solution

```
# Step 1 :  compute and save the minimal statistics

obs.frame <- obs.frame.accident

obs.xtab <- xtabs(obs.frame)
n_ZG <- margin.table(obs.xtab,c(2,3))
n_SG <- margin.table(obs.xtab,c(1,3))
n_GI <- margin.table(obs.xtab,c(3,4))
n_SZ <- margin.table(obs.xtab,c(1,2))
n_ZI <- margin.table(obs.xtab,c(2,4))
n_SI <- margin.table(obs.xtab,c(1,4))


# Step 2: Create a seed for the fitted mu's

# seed the mu_opt
mu_opt <- obs.xtab
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] = 1.0

# Step 3: Perform the loop to approximate the mu's

for (t in 1: 100) {

  mu_ZG <- margin.table(mu_opt,c(2,3))
  for(s in 1:2)
    for(z in 1:2)
      for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_ZG[z,g]/mu_ZG[z,g]

  mu_SG <- margin.table(mu_opt,c(1,3))
  for(s in 1:2)
    for(z in 1:2)
      for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_SG[s,g]/mu_SG[s,g]

  mu_GI <- margin.table(mu_opt,c(3,4))
```

```
  for(s in 1:2)
    for(z in 1:2)
      for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_GI[g,i]/mu_GI[g,i]

  mu_SZ <- margin.table(mu_opt,c(1,2))
  for(s in 1:2)
    for(z in 1:2)
      for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_SZ[s,z]/mu_SZ[s,z]

  mu_ZI <- margin.table(mu_opt,c(2,4))
  for(s in 1:2)
    for(z in 1:2)
      for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_ZI[z,i]/mu_ZI[z,i]

  mu_SI <- margin.table(mu_opt,c(1,4))
  for(s in 1:2)
    for(z in 1:2)
      for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_SI[s,i]/mu_SI[s,i]
}

mu_opt
```

```
## , , gender = Female, injury = No
##
##      location
## belt       Urban      Rural
##   No   7166.3688  3353.8294
##   Yes 11748.3087  5985.4930
##
## , , gender = Male, injury = No
##
##      location
## belt       Urban      Rural
##   No  10471.4955  6045.3062
##   Yes 10837.8269  6811.3714
##
## , , gender = Female, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    993.0169   988.7848
##   Yes   721.3055   781.8927
##
## , , gender = Male, injury = Yes
##
##      location
```

```
## belt       Urban      Rural
##   No    845.1187  1038.0796
##   Yes   387.5588   518.2429
```

```
# CHECK WITH COMMAND loglin
obs.frame <- obs.frame.accident
obs.xtab <- xtabs(obs.frame)
loglin(obs.xtab,
         list(
           c(1,2),
           c(1,3),
           c(1,4),
           c(2,3),
           c(2,4),
           c(3,4)
           ),
       fit = TRUE)$fit
```

```
## 6 iterations: deviation 0.03486629
```

```
## , , gender = Female, injury = No
##
##      location
## belt       Urban      Rural
##   No   7166.3680  3353.8313
##   Yes 11748.3089  5985.4919
##
## , , gender = Male, injury = No
##
##      location
## belt       Urban      Rural
##   No  10471.4936  6045.3073
##   Yes 10837.8295  6811.3696
##
## , , gender = Female, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    993.0167   988.7850
##   Yes   721.3057   781.8926
##
## , , gender = Male, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    845.1186  1038.0795
##   Yes   387.5591   518.2428
```

Consider the 3 way interaction model (GLI, GSI, LSI, GLS).

1. Write your own code to compute the fitted values for $\mu_{i,j,k,c}$
2. Check your ewsults with the R command 'loglin{stats}'

Solution

```
# Step 1 :  compute and save the minimal statistics

obs.frame <- obs.frame.accident
```

```r
obs.xtab <- xtabs(obs.frame)
n_ZGI <- margin.table(obs.xtab,c(2,3,4))
n_SGI <- margin.table(obs.xtab,c(1,3,4))
n_SZI <- margin.table(obs.xtab,c(1,2,4))
n_SZG <- margin.table(obs.xtab,c(1,2,3))


# Step 2: Create a seed for the fitted mu's

# seed the mu_opt
mu_opt <- obs.xtab
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] = 1.0

# Step 3: Perform the loop to approximate the mu's

for (t in 1: 100) {

  mu_ZGI <- margin.table(mu_opt,c(2,3,4))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_ZGI[z,g,i]/mu_ZGI[z,g,i]

  mu_SGI <- margin.table(mu_opt,c(1,3,4))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_SGI[s,g,i]/mu_SGI[s,g,i]

  mu_SZI <- margin.table(mu_opt,c(1,2,4))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_SZI[s,z,i]/mu_SZI[s,z,i]

  mu_SZG <- margin.table(mu_opt,c(1,2,3))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_SZG[s,z,g]/mu_SZG[s,z,g]
}

mu_opt
```

```
## , , gender = Female, injury = No
```

```
##
##      location
## belt      Urban     Rural
##   No   7276.738  3256.262
##   Yes 11597.262  6123.738
##
## , , gender = Male, injury = No
##
##      location
## belt      Urban     Rural
##   No  10391.262  6112.738
##   Yes 10958.738  6703.262
##
## , , gender = Female, injury = Yes
##
##      location
## belt      Urban     Rural
##   No   1006.262   962.738
##   Yes   748.738   767.262
##
## , , gender = Male, injury = Yes
##
##      location
## belt      Urban     Rural
##   No    801.738  1094.262
##   Yes   390.262   502.738
```

```
# CHECK WITH COMMAND loglin
obs.frame <- obs.frame.accident
obs.xtab <- xtabs(obs.frame)
loglin(obs.xtab,
         list(
           c(2,3,4),
           c(1,3,4),
           c(1,2,4),
           c(1,2,3)
           ),
       fit = TRUE)$fit
```

```
## 5 iterations: deviation 0.009604429
```

```
## , , gender = Female, injury = No
##
##      location
## belt       Urban      Rural
##   No   7276.7378  3256.2628
##   Yes 11597.2617  6123.7380
##
## , , gender = Male, injury = No
##
##      location
## belt       Urban      Rural
##   No  10391.2621  6112.7375
##   Yes 10958.7383  6703.2619
##
```

```
## , , gender = Female, injury = Yes
##
##      location
## belt       Urban      Rural
##   No   1006.2622   962.7372
##   Yes   748.7383   767.2620
##
## , , gender = Male, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    801.7379  1094.2625
##   Yes   390.2617   502.7381
```

Consider the independent model (G, S, L, I)

1. compute the fitted values for $\mu_{i,j,k,c}$, by using your own code
2. check your resutls with 'loglin{stats}'

Solution

```
# Step 1 :  compute and save the minimal statistics

obs.frame <- obs.frame.accident

obs.xtab <- xtabs(obs.frame)
n_S <- margin.table(obs.xtab,c(1))
n_Z <- margin.table(obs.xtab,c(2))
n_G <- margin.table(obs.xtab,c(3))
n_I <- margin.table(obs.xtab,c(4))


# Step 2: Create a seed for the fitted mu's

# seed the mu_opt
mu_opt <- obs.xtab
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
          mu_opt[s,z,g,i] = 1.0

# Step 3: Perform the loop to approximate the mu's

for (t in 1: 100) {

  mu_S <- margin.table(mu_opt,c(1))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_S[s]/mu_S[s]

  mu_Z <- margin.table(mu_opt,c(2))
for(s in 1:2)
```

```
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_Z[z]/mu_Z[z]

  mu_G <- margin.table(mu_opt,c(3))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_G[g]/mu_G[g]

  mu_I <- margin.table(mu_opt,c(4))
for(s in 1:2)
  for(z in 1:2)
    for(g in 1:2)
        for(i in 1:2)
        mu_opt[s,z,g,i] <- mu_opt[s,z,g,i]*n_I[i]/mu_I[i]

}


mu_opt
```

```
## , , gender = Female, injury = No
##
##      location
## belt       Urban      Rural
##   No   8153.4100  4820.3536
##   Yes  9971.3181  5895.1137
##
## , , gender = Male, injury = No
##
##      location
## belt       Urban      Rural
##   No   9493.3447  5612.5324
##   Yes 11610.0085  6863.9190
##
## , , gender = Female, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    819.5209   484.5065
##   Yes  1002.2437   592.5335
##
## , , gender = Male, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    954.2013   564.1305
##   Yes  1166.9528   689.9107
```

```
# CHECK WITH COMMAND loglin
obs.frame <- obs.frame.accident
```

```r
obs.xtab <- xtabs(obs.frame)
loglin(obs.xtab,
         list(
           c(1),
           c(2),
           c(3),
           c(4)
           ),
       fit = TRUE)$fit
```

```
## 2 iterations: deviation 7.275958e-12
```

```
## , , gender = Female, injury = No
##
##      location
## belt       Urban      Rural
##   No   8153.4100  4820.3536
##   Yes  9971.3181  5895.1137
##
## , , gender = Male, injury = No
##
##      location
## belt       Urban      Rural
##   No   9493.3447  5612.5324
##   Yes 11610.0085  6863.9190
##
## , , gender = Female, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    819.5209   484.5065
##   Yes  1002.2437   592.5335
##
## , , gender = Male, injury = Yes
##
##      location
## belt       Urban      Rural
##   No    954.2013   564.1305
##   Yes  1166.9528   689.9107
```

Newton Method

Newton's method is a general purpose procedure to compute numerically the solution of a system of non-linear equations given that a number of assumptions are satisfied.

In general.

- Let function $f : \mathbb{R}^n \to \mathbb{R}^n$.
- Assume you need to find the solution x^* of the equation

$$f(x^*) = 0 \tag{1}$$

- Newtons's method for solving the system (1) is the recursion

$$x^{(t+1)} = x^{(t)} - [\nabla_x f(x^{(t)})]^{-1} f(x^{(t)}) \tag{2}$$

for $t \in \mathbb{N}$ and for a pre-specified seed value $x^{(0)} \in \mathbb{R}^n$.

- In theory, Newton's method converges to the solution quadratically; i.e.

$$\lim_{t\to\infty} \frac{|x^{(t+1)} - x^*|_\infty}{|x^{(t)} - x^*|_\infty^2} = 0$$

  under regularity conditions discussed in (Numerical analysis / R. L. Burden, J. D. Faires.)

- In practice, we run Newton's recursion several times starting from a different seed each time.

An intuitive explanation why it works

- From the Taylor expansion, and assuming that $\nabla_x^2 f(x)$ is continuous, I get

$$f(x_{t+1}) = f(x_t) + \nabla_x f(x_t)(x_{t+1} - x_t) + O(|x_{t+1} - x_t|^2)$$

  and by ignoring the error term and rearranging the quantities I get

$$x_{t+1} \sim x_t + \nabla_x f(x_t)(f(x_{t+1}) - f(x_t))$$

- If x_{t+1} is the solution, or close to that, then $f(x_{t+1}) = 0$, and hence

$$x_{t+1} \sim x_t - \nabla_x f(x_t) f(x_t)$$

- Now, we see that the gradient of f times the values of f at x_t leads the sequence towards locations where f is zero.
- So, eventually, it may work ...

Pseudo-algorithm of Newton's method:

`Aim` Approximate the solution of $f(x) = 0$

`Input` number of equations n; seed $x^{(0)} = (x_1^{(0)}, ..., x_n^{(0)}) \in \mathbb{R}^n$; tolerance τ ; maximum number of iterations T

`Output:` Approximate solution $x^* \in \mathbb{R}^n$; trace of $x^{(t)}$; trace of relative error $\tau^{(t)} = |x^{(t)} - x^{(t-1)}|_\infty$; number of iterations performed t

1. Set $x_{\text{opt}} = x^{(0)}$
2. Set $t = 1$
3. While $(t \leq T)$ do:
   i) Compute $n \times 1$ vector $F \in \mathbb{R}^n$ whose i-th element is $F_i = f(x_{\text{opt},i})$
   ii) Compute $n \times n$ vector $J \in \mathbb{R}^{n\times n}$ whose (i,j)-th element is $J_{i,j} = \frac{\mathrm{d}}{\mathrm{d}x_j} f_i(x_{\text{opt}})$ for $(i,j) \in \{1, ..., n\}^2$
   iii) Solve the $n \times n$ linear system $Jy = -F$ and compute $y \in \mathbb{R}^n$
   iv) Update $x_{\text{opt}} = x_{\text{opt}} + y$
   v) Compute $\epsilon^* = |y|_\infty$
   vi) If $\epsilon^* < \tau$, then escape from the loop
   vii) Increase the time step $t = t + 1$
4. Set $x^* = x_{\text{opt}}$
5. Return as output: Return as output: x^*, t^*, and ϵ^*.

Example

Solve the system of non-linear equations

$$\begin{cases} \cos(x_2 x_3) + \frac{1}{2} & = 3x_1 \\ 81(x_2 + 0.1)^2 & = x_1^2 + (x_3 + 0.1)^2 + \sin(x_3) + 1.06 \\ -\frac{10\pi - 3}{3} & = \exp(-x_1 x_2) + 20x_3 \end{cases}$$

Solution

This is equivalent to solving the system $f(x_1, x_2, x_3) = 0$ where

$$f(x) = \begin{bmatrix} 3x_1 - \cos(x_2 x_3) - \frac{1}{2} \\ x_1^2 - 81(x_2 + 0.1)^2 + \sin(x_3) + 1.06 \\ \exp(-x_1 x_2) + 20x_3 + \frac{10\pi - 3}{3} \end{bmatrix}$$

with Jacobian

$$\nabla_x f(x) = \begin{bmatrix} 3 & x_3 \sin(x_2 x_3) & x_2 \sin(x_2 x_3) \\ 2x_1 & -162(x_2 + 0.1) & \cos(x_3) \\ -x_2 \exp(-x_1 x_2) & -x_1 \exp(-x_1 x_2) & 20 \end{bmatrix}$$

I need to supply the Newton's algorithm with the quantities above, as well as consider a tolerance, e.g. $1e-4$ (meaning 10^{-4}), seed value e.g., $x^{(0)} = (0.1, 0.1, -0.1)^T$, etc...

Create a function for $f(x)$, and called `my.f()`

```
my.f <- function(x) {
  f1 <- 3*x[1]-cos(x[2]*x[3])-0.5
  f2 <- x[1]*x[1] -81*(x[2]+0.1)*(x[2]+0.1) +sin(x[3]) +1.06
  f3 <- exp(-x[1]*x[2])+20*x[3]+(10*pi-3)/3
  fvec <- matrix(c(f1,f2,f3),nrow=3)
  return(fvec)
  }
```

Create a function for $\nabla_x f(x)$, and call it `my.Df`

```
my.Df <- function(x) {
  Df11 <- 3
  Df12 <- x[3]*sin(x[2]*x[3])
  Df13 <- x[2]*sin(x[2]*x[3])
  Df21 <- 2*x[1]
  Df22 <- -162*(x[2]+0.1)
  Df23 <- cos(x[3])
  Df31 <- -x[2]*exp(-x[1]*x[2])
  Df32 <- -x[1]*exp(-x[1]*x[2])
  Df33 <- 20
  fmat <- matrix( c(Df11, Df21, Df31,
                    Df12, Df22, Df32,
                    Df13, Df23, Df33 ),
                  nrow=3, byrow=FALSE)
  return(fmat)
  }
```

Create a function called `my.newton.method()` which:

- gets as arguments:
  - the function $f(x)$,

  - the gradient $\nabla_x f(x)$,
  - number of equations n;
  - seed $x^{(0)} = (x_1^{(0)}, ..., x_n^{(0)}) \in \mathbb{R}^n$;
  - tolerance τ ;
  - maximum number of iterations T, etc...
- returns:
  - approximate solution $x^* \in \mathbb{R}^n$;
  - the last relative error τ^*;
  - number of iterations performed t^* , etc...
- use commands
  - `solve()` : to solve the system $Ax = b$
  - `while () {...}`: to perform the loop
  - `break`: to escape from the loop

```
my.newton.method <- function(my.f, my.Df, x0, tol, Tmax) {

  xopt = x0

  t = 1

  while (t <= Tmax) {

    xnow = xopt

    F = my.f( xnow )

    J = my.Df( xnow )

    y = solve(J, -F )

    xnew = xnow + y

    xopt = xnew

    err = max(abs(y))

    if ( err <= tol) break

    t = t +1

  }

  return(list(xopt=xopt, Tmax=t, err=err))
}
```

Solve the equation by using the function the you created

```
#Try me ...

x0  = c(0.1,0.1,-0.1)
tol = 0.00001
Tmax = 200

obj <- my.newton.method(my.f, my.Df, x0, tol, Tmax)

obj$xopt
```

```
##                [,1]
## [1,]  5.000000e-01
## [2,]  4.005740e-18
## [3,] -5.235988e-01
```

```
obj$Tmax
```

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

```
obj$err
```

```
## [1] 7.757857e-10
```

Newton method for the Log linear model

- I wish to solve non-linear equation $X^T n = X^T \hat{\mu}(\beta)$ where matrix X is the design matrix given the non-identifiability constrints, e.g., for the model (XY, XZ, YZ).
- Equivalently, I want to find $\hat{\beta}$ for $f(\hat{\beta}) = 0$, where $f(\hat{\beta}) = X^T(n - \mu(\hat{\beta}))$
- The Jacobian is

$$\begin{aligned}\nabla_\beta f(\beta) = \nabla_\beta X^T(n - \mu(\beta)) \ &= \nabla_\beta [X^T \mu(\beta)] \\ &= X^T \mathrm{diag}(\mu(\beta)) X\end{aligned}$$

  Because the (j,k)th element of $\nabla_\beta [X^T \mu(\beta)]$ is

$$\begin{aligned}\left[\nabla_\beta [X^T \mu(\beta)]\right]_{j,k} &= -\frac{\mathrm{d}}{\mathrm{d}\beta_k} \sum_i X_{i,j} \exp(\sum_j X_{i,j}\beta_j) \\ &= -\sum_i X_{i,j} \exp(\sum_j X_{i,j}\beta_j) X_{i,k}\end{aligned}$$

  since $\mu_i(\beta) = \exp(\sum_j X_{i,j}\beta_j)$.
- Then the Newton's recursion (2) becomes

$$\beta_{t+1} = \beta_t + [X^T \mathrm{diag}(\mu(\beta_t)) X]^{-1} X^T (n - \mu(\beta_t))$$

- It is proven that $\beta_t \to \hat{\beta}$.

Example (For Homework)

Consider the data-set, Alcohol, Cigarette, and Marijuana

```
## load the data

obs.frame<-data.frame(count=c(911,538,44,456,3,43,2,279),
```

```
                      expand.grid(
  marijuana=factor(c("Yes","No"),levels=c("No","Yes")),
  cigarette=factor(c("Yes","No"),levels=c("No","Yes")),
  alcohol=factor(c("Yes","No"),levels=c("No","Yes")))
  )

## orint the obs.frame

obs.frame
```

```
##   count marijuana cigarette alcohol
## 1   911       Yes       Yes     Yes
## 2   538        No       Yes     Yes
## 3    44       Yes        No     Yes
## 4   456        No        No     Yes
## 5     3       Yes       Yes      No
## 6    43        No       Yes      No
## 7     2       Yes        No      No
## 8   279        No        No      No
```

Consider:

- a Log-linear model describing a homogeneous association between each pair of variables at each level of the third one; i.e. $[AC, AM, CM]$
- as identifiability constraints the corner points where the reference levels are the last levels; namely, 2 (yes), 2 (yes), and 2 (yes) for marijuana, cigarette, and alcohol.

Use Newton method in order to compute λ's

Estimate the log linear model coefficients

Solution

```
# This is a homework for further practice.
```

Save me

Generate the document as a Notebook, PDF, Word, or HTML by choosing the relevant option (from the pop-up menu next to the Preview button). Then save your Markdown code by choosing the relevant option (from the task bar menu).

Save the *.Rmd script, so that you can edit it later.