

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

Georgios Karagiannis

Aim

- To become familiar with R Notebook.
 - 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 non-linear equations via Newton method
 - Apply Newton method to produce MLE for the Log Linear models
 - Extensions of Newton method, and IPF method to produce MLEs for 4-way and higher order tables
-

About R Markdown Notebook

What it is

R Markdown Notebook :

- creates documents that contain explanatory text, mathematical equations, live code, and visualizations.
- produces fully interactive documents allowing the readers to change the parameters underlying the analysis, and see the results immediately.
- produces interactive statistical reports that include data analysis, code, and results.

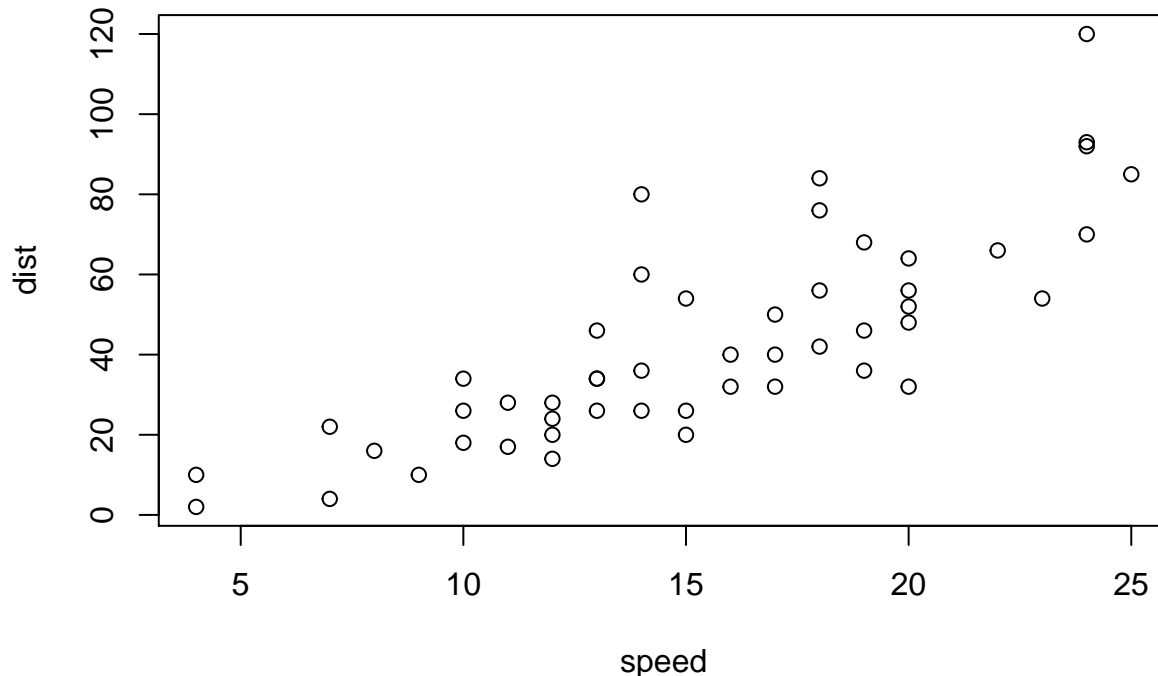
An R Notebook demo:

In what follows, we provide a simple demo to help you familiarise yourselves with the R Markdown Notebook environment.

This is an R Markdown Notebook. When you execute code within the Notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
summary(cars)
##      speed      dist
##  Min.   : 4.0    Min.   : 2.00
##  1st Qu.:12.0    1st Qu.: 26.00
##  Median :15.0    Median : 36.00
##  Mean   :15.4    Mean   : 42.98
##  3rd Qu.:19.0    3rd Qu.: 56.00
##  Max.   :25.0    Max.   :120.00
plot(cars)
```



How it works (Skip it)

Creating documents with R Markdown Notebook, requires the user to create a .Rmd (script) file that contains a combination of Markdown code (that produced the explanatory text) and R code chunks (that produce plots, and data analysis outputs). Then the .Rmd file is processed automatically by the software in order to generate a document in a format such as: HTML (web page), PDF, MS Word document, slide show, handout, book, dashboard, package vignette, etc.

In this practical, we will not go into details about Markdown coding, although it has a super simple syntax. We recommend the interested student to read the nice cheat-sheet available from [\[here\]](#).

How to tune presentation (Skip it)

The user specifies how the chunk will be executed, or how its output will be presented, by setting the appropriate flags inside `{r, ...}`. If time permits, use the following flags in the previous chunk. What is their effect?

- `eval`: (TRUE; logical) whether to evaluate the code chunk; it can also be a numeric vector to select which R expression(s) to evaluate, e.g. `eval=c(1, 3, 4)` or `eval=-(4:5)`
- `echo`: (TRUE; logical or numeric) whether to include R source code in the output file. Besides TRUE/FALSE which completely turns on/off the source code, we can also use a numeric vector to select which R expression(s) to echo in a chunk, e.g. `echo=2:3` means only echo the 2nd and 3rd expressions, and `echo=-4` means to exclude the 4th expression.
- `results`: ('markup'; character) takes these values
 - `hold`: hold all the output pieces and push them to the end of a chunk
 - `hide`: hide results; this option only applies to normal R output (not warnings, messages or errors)
 - `markup`: mark up the results using the output hook, e.g. put results in a special LaTeX environment
- `collapse`: (FALSE; logical; applies to Markdown output only) whether to, if possible, collapse all the source and output blocks from one code chunk into a single block (by default, they are written to separate blocks)

More options can be found at <https://yihui.name/knitr/options/>.

You can use inline R code inside a sentence. For example you can say that the average speed of the car is 15.4, with standard error 0.1057529. Now see the output document.

Do the following:

(...below the enumerated list)

1. Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.
2. Write a simple R command inside this chunk, just to experiment; E.g. write down `hist(cars$speed)`.
3. Run this R chunk as described earlier.

Produce the document:

The document will be saved in the working directory.

- To produce and save the document as a Notebook: click the *Preview* button (or *Knit* button) or press *Ctrl+Shift+K* to preview the HTML file.
 - When you save the notebook, an HTML file containing the code and output will be saved alongside it.
 - To produce and save the document in HTML, PDF or MS Word formats: select the option *Knit to HTML* (or PDF, or Word) available in the menu that appears when you click the little black arrow next to the *Preview* button (or *Knit* button).
 - This will produce the document in more standard / less functional formats.
-

Contingency 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")))
)

## print 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 dimensional contingency table from `obs.frame`. Use command:

- `xtabs()`, to create a contingency table from cross-classifying factors in a `data.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 contingency table which contains the row, column, layer, etc... margins.

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

```
#
#
```

Compute the marginal contingency table of marijuana and cigarette.

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

```
#
#
```

Compute the (joint) sampling proportions.

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

```
#
#
```

Create a `data.frame` of proportions.

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

```
#
#
```

As a homework, you can further 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))`
-

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 $\mu_{ijk}^{(0)}$ satisfying a model no more complex than the one being fitted. E.g., $\mu_{ijk}^{(0)} = 1.0$ should be ok.
- For $t = 1, \dots$,
 - adjust $\mu_{ijk}^{(t)}$ 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 $\mu_{ijk}^{(t-1)}$.
- Then the t -th cycle of the IPF algorithm has the following steps:
 - Set $m_{ijk}^{(0)} = \mu_{ijk}^{(t-1)}$
 - Compute

$$m_{ijk}^{(1)} = m_{ijk}^{(0)} \frac{n_{ij+}}{m_{ij+}^{(0)}}; \forall i, j, k$$

$$m_{ijk}^{(2)} = m_{ijk}^{(1)} \frac{n_{i+k}}{m_{i+k}^{(1)}}; \forall i, j, k$$

$$m_{ijk}^{(3)} = m_{ijk}^{(2)} \frac{n_{+jk}}{m_{+jk}^{(2)}}; \forall i, j, k$$

- Set $\mu_{ijk}^{(t)} = m_{ijk}^{(3)}, \forall i, j, k$

... and produces $\mu_{ijk}^{(t)}$ 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")))
                      )

## print 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 $\mu_{i,j,k}$

Solution

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

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

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

Present the fitted μ 's as a data frame

```
#
#
```

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. $[X, Y, Z]$
- Find the fitted values for $\mu_{i,j,k}$

Solution

```
#
#
```

Example (Homework)

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

```
##      belt location gender injury count
## 1      No      Urban Female      No 7287
## 2      Yes      Urban Female      No 11587
## 3      No      Rural Female      No 3246
## 4      Yes      Rural Female      No 6134
## 5      No      Urban   Male      No 10381
```

##	6	Yes	Urban	Male	No	10969
##	7	No	Rural	Male	No	6123
##	8	Yes	Rural	Male	No	6693
##	9	No	Urban	Female	Yes	996
##	10	Yes	Urban	Female	Yes	759
##	11	No	Rural	Female	Yes	973
##	12	Yes	Rural	Female	Yes	757
##	13	No	Urban	Male	Yes	812
##	14	Yes	Urban	Male	Yes	380
##	15	No	Rural	Male	Yes	1084
##	16	Yes	Rural	Male	Yes	513

Consider the Homogeneity association model (GL, GS, GI, LS, LI, SI) , and compute the fitted values for $\mu_{i,j,k,c}$

Solution

```
#
#
```

Consider the independent model (G, S, L, I) , and compute the fitted values for $\mu_{i,j,k,c}$

Solution

```
#
#
```

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

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

- Newton'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.

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

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 $x^{(0)} = (x_1^{(0)}, \dots, x_n^{(0)}) \in \mathbb{R}^n$; 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 $J_{i,j} = \frac{d}{dx_j} f_i(x_{\text{opt}})$ for $(i,j) \in \{1, \dots, n\}^2$
 - 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()`

```
#
#
```

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

```
#
#
```

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 $x^{(0)} = (x_1^{(0)}, \dots, x_n^{(0)}) \in \mathbb{R}^n$;
 - 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

```
#
#
```

Solve the equation by using the function the you created

```
#
#
```

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 constraints, 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

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

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

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

since $\mu_i(\beta) = \exp(\sum_j X_{i,j} \beta_j)$.

- Then the Newton's recursion (2) becomes

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

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

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.