Universiteit van Amsterdam
Amsterdam School of Economics

Computer class NLIST—Assignment 2

Automobile insurance portfolio; generation and analysis

This assignment helps you study Appendix A.3 in MART, as well as the exploratory data analysis part of the bonus-malus case study of Sec. 9.6.

We study how to use `R` to generate a pseudo-random portfolio of automobile claims data, resembling a real-life portfolio in aspects like frequencies of levels of risk factors, interactions of these, and distribution of numbers of claims. For the claim numbers, we take Poisson random variables with parameters determined by the risk factors, with some policies in force for only part of the year.

We also show how to condense the data so the dataset is easier to handle for purposes of analysis.

Following Sec. 9.5, using a similar aggregated portfolio we will try to reconstruct the way the Poisson parameters depend on the risk factors by the use of GLMs.

Like in Sec. 9.6, a similar aggregated portfolio is studied, resembling the dataset leading to the bonus-malus system of MART, Ch. 6. In a subsequent assignment, we will use GLMs to analyze this dataset further.

Some things you will learn about:

- drawing random samples using the `sample` function
- memory usage in `R`
- producing aggregated data from a portfolio of individual policies
- recoding vectors efficiently
- reading data from keyboard/scriptfile using the `scan()` function
- factor, aka 'category' or 'enumerated type', vs. variate and vector
- constructing factor levels by utilizing their pattern, e.g.:
- using the `gl()` function to generate levels
- using a dataframe to store a matrix with columns of fixed type, but rows with varying types
- producing cross-tables, including a 4-way cross table of the claim frequencies by four factors, and printing it in a readable way
- using `anova()` to do an analysis of deviance for a chain of nested models
- two ways to deal with 'exposure' in a multiplicative GLM: using averages and weights, or using the 'offset' mechanism
- printing the contents of an object produced by a `glm()` call
- viewing part of a dataframe

- the `tapply()` function: `tapply(n,R,f)` means "apply the `f`-function to the `n`-values of the groups sharing the same `R`-level"
- making tables of claim frequencies per policy, broken down by 2 (or more) factors
- producing loss ratios by level for all risk factors
- using a for-loop in which the 'control variable' runs through a list of risk factors
- generating a plot of the average number of claims in each separate BM-class, both on ordinary and log-scale
- plotting regression lines based on a subset of the BM-classes, to show that a particular class is 'special'

1 Simulating an insurance portfolio—App. A3

We are going to generate a simulated automobile insurance portfolio consisting of 10 000 policies, initializing the random number generator to be able to reproduce the stream of random numbers:

```
n.obs <- 10000; set.seed(4)
```

In the population from which we draw, 60% of the policy holders have gender 1. We use the `sample()` function. Its first argument is a vector with the possible values. The second is the number of observations. The `repl=` argument, with default `FALSE`, governs if sampling is done with or without replacement. Evidently, we want to sample with replacement. The `prob=` argument gives the (relative) probabilities of values 1 and 2.

```
sx <- sample(1:2, n.obs, repl=TRUE, prob=c(6,4))
```

As intended, `prob=c(6,4)` means that about 60% is to fall in category 1, 40% in 2.

We want the gender numbers 1 and 2 to act as class labels, so we let `sx` be of type factor, as follows:

```
sx <- as.factor(sx)
```

Analogously, we assign random job classes 1,2,3 to our policies:

```
jb <- as.factor(sample(1:3, n.obs, repl=TRUE, prob=c(3,2,1)))
```

Note that `sx` and `jb` were drawn *independently*, making the probability of, for example, both `sx=1` and `jb=3` equal to $6/10 \times 1/6$. But we want there to be *correlation* between the factors `re` (region of residence) and `tp` (type of car), which means that the conditional probability of `re=r`, given `tp=t`, varies with `t`. We want the joint probabilities of combinations `(re,tp)` to be as follows:

	tp	1	2	3
re	1	.1	.05	.15
	2	.15	.1	.05
	3	.1	.1	.2

To achieve this, we first draw combinations of these factors, with probabilities as given in the table.

```
re.tp <- sample(1:9, n.obs, repl=TRUE, prob=c(.1,.05,.15,.15,.1,.05,.1,.1,.2))
```

Next we reconstruct values of `re` and `tp` from the values $1, \ldots, 9$ of `re.tp`. Values 1, 4 and 7 give `tp=1`, values 2,5,8 give `tp=2`, and 3,6,9 give `tp=3`. Analogously for region `re`. To achieve this, do

```
tp <- c(1,2,3,1,2,3,1,2,3)[re.tp]; tp <- as.factor(tp)
re <- c(1,1,1,2,2,2,3,3,3)[re.tp]; re <- as.factor(re)
```

To explain this, look at the first ten elements of the vector `re.tp`: $3, 8, 2, 4, 4, 9, 7, 8, 6, 9$.
This means that the first ten sample elements will get `tp` $= 3, 2, 2, 1, 1, 3, 1, 2, 3, 3$.
The first ten elements of vector `re` get the values $1, 3, 1, 2, 2, 3, 3, 3, 2, 3$.

To see if we got what we wanted, we do:

```
table(list(region=re, type=tp))
```

This gives a *cross-table* with the numbers of policies for each `(re,tp)` combination (the left panel in the table below). For example, 465 policies had `re` 2 and `tp` 3, the expected number was $10\,000 \times 0.05 = 500$.

Realizations

region	type 1	2	3
1	960	527	1475
2	1496	1038	465
3	991	1002	2046

Expected numbers

region	type 1	2	3
1	1000	500	1500
2	1500	1000	500
3	1000	1000	2000

To get a one-dimensional frequency table, simply do, for example, `table(re)`.

We do some basic tests on memory usage by `R`:

```
hh <- 1:2^20;          object.size(hh)/length(hh) ## 4 + a bit
hh <- rep(0,2^20);     object.size(hh)/length(hh) ## 8 + a bit
hh <- as.integer(hh); object.size(hh)/length(hh) ## 4 + a bit
```

It turns out that with a few dozen bytes overhead, (long) integers occupy 4 bytes each. In `R`, reals take 8 bytes (double precision). Sometimes obvious integers (the 2^{20} zeros) are stored as reals.

Since `R` stores data in *memory*, we remove the vector `re.tp` that is now no longer needed:

```
object.size(re.tp); rm(re.tp); rm(hh)
```

Vector `re.tp` occupied 10000×4 bytes in memory, so a little over 40000 bytes were released.

Q_1 How many bytes does it take to store $1, \ldots, 10, 1\,000, 100\,000$ logical values `TRUE`/`FALSE`? □

The policies had different *exposure* times. Each policy was in force for either 3 (10% probability), 6 (10%) or 12 months (80%), independent of the other characteristics of the policy:

```
mo <- 3 * sample(1:4, n.obs, repl=TRUE, prob=c(1,1,0,8))
```

The number of claims per cell follows a Poisson process with means `mu[1], mu[2], ...` per year. The base claim frequency 0.05 below is the claim frequency in the 'standard' cell (1,1,1,1) with a full year exposure time. An increase of 1 in each level of `sx`, `re` and `tp` leads to 20% more claims; `jb` level is irrelevant for the claim frequency. The Poisson parameters to be fed to `rpois` are multiplied by `mo/12`. To start the analysis, make a table of the number of claims per policy; it reveals that of the 10000 policies, 9276 were claim-free.

```
mu <- 0.05 * c(1,1.2)[sx] *
              c(1,1,1)[jb] *
              c(1,1.2,1.44)[re] *
              1.2^(0:2)[tp] *
              mo/12
y <- rpois(n.obs, mu)
table(y)
##    y
##    0    1    2    3
## 9276  702   20    2
```

Q_2 Let random variable Y denote the number of accidents of a policy; $Y_1, \ldots, Y_n$ is a sample. For a Poisson random variable, expected value and variance are equal. The distribution of Y, however, is not pure Poisson but *mixed* Poisson as in MART, Example 3.3.1. We have $Y_i \,|\, \Lambda_i = \lambda \sim \text{Poisson}(\lambda)$. The value of the Poisson parameter Λ_i is determined by outcomes of random risk factors and exposure time as `mu <- 0.05 * ... * mo/12` above. By MART (3.14), we have $\mathrm{E}[Y_i] = \mathrm{E}[\mathrm{E}[Y_i|\Lambda_i]] = \mathrm{E}[\Lambda_i]$ and $\mathrm{Var}[Y_i] = \mathrm{Var}[\Lambda_i] + \mathrm{E}[\Lambda_i]$. As a consequence, $\mathrm{Var}[Y_i] - \mathrm{E}[Y_i] = \mathrm{Var}[\Lambda_i] > 0$. It follows that the *overdispersion* factor $\mathrm{Var}[Y]/\,\mathrm{E}[Y] > 1$.

Now compare `mean(y)` and `var(y)`, and comment. □

A *cross-table* of the numbers of accidents by gender shows that, e.g., 3698 persons of gender 2 were claim-free.

```
table(list(nCl=y,gender=sx))
##    gender
## nCl    1    2
##   0 5578 3698
```

```
##   1  400  302
##   2    9   11
##   3    1    1
```

To tabulate the number of claims for each gender-region combination, use either the `":"` interaction operator between factors, or the `interaction()` function:

```
table(list(nCl=y,gender.region=sx:re))
##    gender.region
## nCl  1:1  1:2  1:3  2:1  2:2  2:3
##   0 1683 1686 2209 1103 1119 1476
##   1  102  105  193   70   84  148
##   2    2    0    7    2    4    5
##   3    0    0    1    0    1    0
```

Aggregating data

To reduce time and memory needed, without actually losing vital information, it often makes sense to use an *aggregated* version of a portfolio.

For every combination of risk factors, we need the total numbers of claims, the total exposure times and the numbers of policies. This is achieved by calling the `aggregate()` function, as follows:

```
aggr <- aggregate(list(Expo=mo/12,nCl=y,nPol=1),
                  list(Jb=jb,Tp=tp,Re=re,Sx=sx), sum)
```

The first argument of `aggregate` is a list of the quantities to be combined; the second a list of factors to split up the portfolio. Other operations than totaling by `sum` may be specified.

The resulting object `aggr` is of type *dataframe*, that is, a matrix with rows containing varying types but columns of uniform type. Use `aggr$Jb` to access that particular column, or add an argument `data=aggr` to function calls and access the variable by just `Jb`.

The condensed data have $3 \times 3 \times 3 \times 2 = 54$ rows and 7 columns; the full data had 10000 rows and 6 columns. We list all columns of a *sample* of ten rows of the newly created `aggr` dataframe:

```
aggr[sort(sample(1:54,10)),]
##    Jb Tp Re Sx   Expo nCl nPol
## 6   3  2  1  1  42.75   2   49
## 10  1  1  2  1 376.25  26  436
## 11  2  1  2  1 262.75  13  297
## 27  3  3  3  1 178.00  23  205
## 30  3  1  1  2  68.75   6   78
## 42  3  2  2  2  43.25   4   49
## 43  1  3  2  2  79.00  10   89
## 44  2  3  2  2  55.00   5   66
## 51  3  2  3  2  56.50   3   62
## 54  3  3  3  2 115.75  13  130
```

If the number of policies is large but the number of possible combinations of risk charateristics is small, this gives an enormous reduction in storage size, as well as in computer time. Of course we have to inform our statistical functions, like `(g)lm` for regression, that the values in the dataframe are totals corresponding to different numbers of exposure years.

Q_3 Compute how much memory is gained by using the `aggr` dataframe, by doing:

```
object.size(aggr)
object.size(mo)
object.size(y)
object.size(jb) + object.size(tp) + object.size(re) + object.size(sx)
```
□

Q_4 To estimate the Poisson parameter $\lambda_{3,3,3,2}$ per policy year for policies with `jb`=3, `tp`=3, `re`=3 and `sx`=2 (cell 54; last one in the sample above), it can be seen that there were originally 130 policies with these characteristics, having made 13 claims in total. Your statistics book Bain and Engelhardt, Ex. 9.2.6, states that the sample mean is the maximum likelihood estimator for a Poisson random sample. But according to MART Sec. 3.9.3, what is actually the ML-estimate $\widehat{\lambda}_{3,3,3,2}$? □

2 Exploring the automobile portfolio of Sec. 9.5

In the case study in Section 9.5 of MART, a portfolio is studied resembling the aggregated portfolio constructed in Appendix A.3. Here, exposure periods were not fractions of years, but a period of seven years for each policy, so the exposure is simply equal to seven times the number of policies. Instead of using a dataframe, we construct separate vectors containing the data.

First clear the workspace and read the cell totals of the already aggregated automobile portfolio:

```
rm(list=ls(all=TRUE))
n <- scan(n=54) ## read 54 numbers into vector n
 1  8 10  8  5 11 14 12 11 10  5 12 13 12 15 13 12 24
12 11  6  8 16 19 28 11 14  4 12  8 18  3 17  6 11 18
12  3 10 18 10 13 12 31 16 16 13 14  8 19 20  9 23 27
expo <- scan(n=54) ## the number of policies
10 22 30 11 15 20 25 25 23 28 19 22 19 21 19 16 18 29
25 18 20 13 26 21 27 14 16 11 23 26 29 13 26 13 17 27
20 18 20 29 27 24 23 26 18 25 17 29 11 24 16 11 22 29
expo <- 7 * expo ## each policy is in force during a 7-year period
```

Now we reconstruct the corresponding levels of the various risk factors. It is known that the first 27 units have `sex`=1, the others `sex`=2. So the vector `sex` must have numbers 1 and 2 in it, each replicated 27 times; the total length of the vector is 54. This is how to store it as a factor:

```
sex <- rep(1:2, each=27, len=54)
sex <- as.factor(sex)
```

The `gl()` function is a shorter way to generate factor levels; in fact, the two lines above are equivalent to `sex <- gl(2, 27, 54)` or even `sex <- gl(2,27)`, omitting the required length since it is implied. The numbers 1:**3** occur in blocks of length **9** to give **54** `region` levels, and so on:

```
region <- gl(3, 9, 54); type <- gl(3, 3, 54); job <- gl(3, 1, 54)
```

See MART p. 249 for a description of what has been done so far.

Q_5 Comment on the difference between:

```
str(type)
str(rep(1:3, each=3, len=54))
```

□

Define an ordered subset of size 15 of the index set 1:54, and print the data for this random subset of the indices:

```
set.seed(1); subset <- sort(sample(1:54,15))
data.frame(sex, region, type, job, n, expo)[subset,]
```

Taking a sample here is purely for convenience; we could have printed the full dataframe, only then 'old' output would have vanished from sight.

Q_6 For the first two cells listed, check if the covariates have the right value. □

To reproduce Table 9.1, we can make a cross-table of the claim frequency by all four factors as follows:

```
xt <- xtabs(round(1000 * n/expo) ~ sex+region+type+job)
```

We print a 'flat table' with the first two variables as rows, the last two as columns:

```
ftable(xt, row.vars=1:2, col.vars=3:4)
##            type   1           2           3
##            job    1   2   3   1   2   3   1   2   3
## sex region
## 1   1            14  52  48 104  48  79  80  69  68
##     2            51  38  78  98  82 113 116  95 118
##     3            69  87  43  88  88 129 148 112 125
## 2   1            52  75  44  89  33  93  66  92  95
##     2            86  24  71  89  53  77  75 170 127
##     3            91 109  69 104 113 179 117 149 133
```

On the basis of this table, it is hard to judge which variables are important and what their influence is. And in practice, many more factors and levels may be involved. Therefore we turn to a statistical GLM-analysis of the problem.

Analysis of deviance

To be able to do an analysis-of-deviance (see Table 9.2) for the string of models `1` → `1+region` → `1+region+type` (main effects only) → `region*type` (interactions as well), do

```
anova(glm(n/expo ~ region*type, quasipoisson, wei=expo))
```

The result is a description of the model, and then the following table:

	Df	Deviance	Resid. Df	Resid. Dev
NULL			53	104.732
region	2	21.597	51	83.135
type	2	38.195	49	44.940
region:type	4	2.529	45	42.412

In this table, the second and third columns are the differences of the last two columns. The null-hypothesis is that the extra parameters have no effect. Under H_0, the deviance differences, divided by the dispersion parameter, which equals 1 here because we have proper Poisson variables, are approximately χ^2 random variables. The number of degrees of freedom is the number of extra parameters estimated, as found in the second column.

The critical value at 95% level for a $\chi^2(2)$ random variable being 6, including `region` and then `type` in our model is meaningful, but adding their interaction as well is not. Apparently there is not a different effect of `region` given `type`, or v.v.

Q_7 Construct the analysis of deviance table when `type` is added before `region`. Are the differences in deviances from adding these terms the same? □

Q_8 There are two different ways to deal with the 'exposure'; see MART p. 250. One is by taking average claim frequencies `n/expo` and using weights `expo`, as is done above, the other is by adding the log of the exposure as an 'offset' to the linear predictor:

```
(g.off <- glm(n ~ 1+region+type+region:type+offset(log(expo)),
              family=poisson(link=log)))
(g.wei <- glm(n/expo ~ region*type, poisson, wei=expo))
```

Explain the similarities and the differences in the resulting output. □

Printing the `glm`-object storing the result of the fit can be done in three ways:

- all elements of the object `g.off` (a *list*) are given by `str(g.off)`
- some basic results of the fit (deviance, parameter estimates, ...) are produced by `print(g.off)`, or just `g.off`, or as above, by `(g.off <- glm(...))`
- do `summary(g.off)` to get in addition the five quartiles of the residuals as well as standard errors and Student test statistics to judge significance of the individual covariates, including all factor levels

The design matrix, or model matrix, or regression matrix, or **X**-matrix, is accessed by

```
X <- model.matrix(g.off)
X[subset,]   ##  print a subset of its rows; all columns
```

Q_9 Explain why multiplying the 0/1 dummies with `region2` and `type3` gives the dummy for `region2:type3`. □

Q_{10} Run:

```
g.main <- glm(n/expo ~ region+type, quasipoisson, wei=expo)
(cc <- g.main$coef) ## or more conveniently:
coef(g.main)
```

In this model, what is the estimated annual number of claims for someone with:

a) `region`=1, `type`=1?
b) the worst `type/region` combination?

Use `R` to do the computations! □

Q_{11} All ingredients of a Generalized Linear Model can be found back in the `glm`-object generated. For example:

- the matrix **X** of regressors is found by `model.matrix(g.off)`;
- the vector $\vec{\beta}$ of parameter estimates is `coef(g.off)`;
- the offset $\vec{o}$ to be added to the linear predictor is `g.off$offset`;
- the inverse of the link function $g(\cdot)$ is `g.off$family$linkinv()`;
- the fitted values $\vec{f}$ can be extracted using `fitted.values(g.off)`.

Using that the vector of linear predictors equals $\mathbf{X}\vec{\beta}$, reconstruct the last vector using the four earlier items. □

Q_{12} Run:

```
g. <- glm(n/expo ~ as.numeric(region)+type, quasipoisson, wei=expo)
summary(g.main); summary(g.)
anova(g., g.main)
```

Analyze the deviances of `g.main` and `g`.

Why is `g.` a restriction of `g.main`?

Does the restriction in `g.` actually make sense? Hint: what does it mean for the difference in premiums for regions 1, 2 and 3, other tariff factors being equal? □

Q_{13} = **9.5.1** in MART.

Comment on the results of the `anova()` call applied to the portfolio of App. A.3 and argue which model should be chosen.

```
rm(list=ls(all=TRUE))
n.obs <- 10000; set.seed(4) ## 10000 obs.; random seed initialized to 4
sx <- sample(1:2, n.obs, repl=TRUE, prob=c(6,4)); sx <- as.factor(sx)
jb <- as.factor(sample(1:3, n.obs, repl=TRUE, prob=c(3,2,1)))
re.tp <- sample(1:9, n.obs, repl=TRUE, prob=c(2,1,3,3,2,1,2,2,4))
tp <- as.factor(c(1,2,3,1,2,3,1,2,3)[re.tp])
re <- as.factor(c(1,1,1,2,2,2,3,3,3)[re.tp])
rm(re.tp)
mo <- 3 * sample(1:4, n.obs, repl=TRUE, prob=c(1,1,0,8))
mu <- 0.05 * c(1,1.2)[sx] * c(1,1,1)[jb] *
            c(1,1.2,1.44)[re] * c(1,1.2,1.44)[tp] * mo/12
y <- rpois(n.obs, mu)
aggr <- aggregate(list(Expo=mo/12,nCl=y,nPol=1),
                  list(Jb=jb,Tp=tp,Re=re,Sx=sx), sum)
anova(glm(nCl~(Tp+Re+Sx+Jb)^4, poisson, offset=log(Expo), data=aggr), test="Chisq")
```

The vectors that are the columns of the dataframe `aggr` can be accessed e.g. by `aggr$Expo`. It is more convenient to add an argument `data=aggr` to each `glm`-call. □

Q_{14} ≈ **9.5.2** To the same portfolio as in the previous exercise, apply

```
g <- glm(nCl~Re*Sx, poisson, offset=log(Expo), data=aggr)
anova(g, test="Chisq")
```

Compare with what happens if we use the full data instead of the aggregated data.

□

3 Analyzing a bonus-malus system—Sec. 9.6

In the case study of Sec. 9.6, a portfolio is studied resembling the portfolio that was the basis for the bonus-malus system developed in Ch. 6. See also the text on MART p. 252–261.

Getting the data

The data needed can be downloaded from the internet. They are on a file with headers on the first line. To read and store it as a dataframe, do (note the forward slashes / in the url):

```
rm(list=ls(all=TRUE)) ## First remove traces of previous sessions
fn <- "http://www1.fee.uva.nl/ke/act/people/kaas/Cars.txt"
Cars <- read.table(fn, header=TRUE)
```

`Cars` is an aggregated portfolio with 8316 cells and 10 variables, of two types:

Cell totals: `Expo, nCl, TotCl, TotPrem` denote exposure, number of claims, total claim amount and total premium paid

Risk factors: `B, WW, R, A, M, U`: bonus-malus class (1–14), weight class (1–11), region of residence (1–3), age class (1–3), mileage class (1–3), usage (1–2)

To see what this dataframe contains, do

```
Cars[1:5,]  ## rows 1,...,5, all columns
##   Expo nCl TotCl TotPrem B WW R A M U
## 1   15   4 10001    6885 1  1 1 1 1 1
## 2    2   1   492     918 1  1 1 1 1 2
## 3   14   2  1136    7140 1  1 1 1 2 1
## 4    6   2  4129    3060 1  1 1 1 2 2
## 5    4   3  4005    2244 1  1 1 1 3 1
```

First we construct some auxiliary variables: `Bminus1` is `B-1`, that is, the numeric index of the BM-class less 1, and `Bis14` is a 0/1 *dummy variable* denoting membership of BM-class 14. Then we make *factors* (classifications) out of the risk variables. In a model matrix for a regression, for each level but the first a 'membership dummy' is created.

```
Bminus1 <- Cars$B - 1; Bis14 <- as.numeric(Cars$B==14)
Cars$A <- as.factor(Cars$A); Cars$R <- as.factor(Cars$R)
Cars$M <- as.factor(Cars$M); Cars$U <- as.factor(Cars$U)
Cars$B <- as.factor(Cars$B); Cars$WW <- as.factor(Cars$WW)
```

The actual mid-weight of the weight classes is 650, 750, ..., 1600 kg. In `W`, we store the log of the weight relative to the lightest cars.

```
ActualWt <- c(650,750,825,875,925,975,1025,1075,1175,1375,1600)
W <- log(ActualWt/650)[Cars$WW]
```

Exploratory data analysis

The next two lines reproduce the numbers in Table 9.3 of MART, being the number of claims (in percent) per exposure year for each region:

```
attach(Cars)  ## to the search path; now nCl means Cars$nCl
options(digits=2) ## no need for many digits
100 * tapply(nCl, R, sum) / tapply(Expo, R, sum) ## 7.6 9.6 12.7
```

The first call to the `tapply` function actually does the following: "Apply the `sum` function to the `nCl` values of the groups with the same level of `R`". Same for `Expo` in the denominator.

Two-dimensional tables like in Table 9.5 (number of claims per 100 exposure years split up by region and age group) are produced by for example

```
100 * tapply(nCl,list(R=R,A=A),sum) / tapply(Expo,list(R=R,A=A),sum)
## A
## R    1    2  3
##   1 21  6.4 11
##   2 27  7.9 14
##   3 34 10.8 18
```

The four-way cross table MART 9.6 can also easily be reproduced, by using the : operation to produce the interaction of two factors:

```
100 * tapply(nCl,list(A:M,R:U),sum) / tapply(Expo,list(A:M,R:U),sum)
```

Loss ratios (see Remark 9.6.2 and Table 9.9) per risk group and in total can for example be produced by:

```
sum(TotCl)/sum(TotPrem)*100 ## the grand total loss ratio in pct
for (rf in list(B,WW,R,M,A,U)) ## for all risk factors, do:
  print(round(tapply(TotCl,rf,sum)/tapply(TotPrem,rf,sum)*100))
```

The control variable of the for-loop above successively takes the value of 6 arrays containing values of the risk factors.

To produce a flat table from the cross-table of the cell totals by the first four risk factors (hence aggregated for weight and BM-class), do

```
options(digits=5)
ftable(xtabs(cbind(Expo, nCl, TotCl, TotPrem) ~ R+A+M+U))
detach(Cars) ## don't forget
```

Problems from MART Ch. 9.6

To construct the portfolio and the other objects referred to, do (repeated from earlier):

```
rm(list=ls(all=TRUE)) ## First remove traces of previous sessions
fn <- "http://www1.fee.uva.nl/ke/act/people/kaas/Cars.txt"
Cars <- read.table(fn, header=TRUE)

Bminus1 <- Cars$B - 1; Bis14 <- as.numeric(Cars$B==14)
Cars$A <- as.factor(Cars$A); Cars$R <- as.factor(Cars$R)
Cars$M <- as.factor(Cars$M); Cars$U <- as.factor(Cars$U)
Cars$B <- as.factor(Cars$B); Cars$WW <- as.factor(Cars$WW)

ActualWt <- c(650,750,825,875,925,975,1025,1075,1175,1375,1600)
W <- log(ActualWt/650)[Cars$WW]
attach(Cars)  ## to the search path; now nCl means Cars$nCl
options(digits=2)
```

In the exercise that follows we want to plot the average number of claims against the bonus-malus class. So we have to count the number of claims in each class, as well as their total number of policies. The percentages of total exposures for each BM-class are found as follows:

```
t2 <- tapply(Expo, B, sum); round(100*t2/sum(Expo),1)
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14
## 3.7 3.7 3.8 3.7 3.7 3.7 3.7 3.8 3.7 3.7 3.4 3.4 3.5 52.2
```

So we apply the function `sum()` to each group of components of the first argument `Expo`, defined by the levels of the second argument, here `B` (BM-class), as if they were separate vector structures. The result is a structure of the same length as the levels attribute of the factor containing the results.

The total numbers of claims in each BM-class are found similarly, so to construct a graph depicting the development of average number of claims against the BM class $1, \ldots, 14$ in `R`, we do:

```
par(mfrow=c(1,2)) ## to get plots next to each other
t1 <- tapply(nCl, B, sum); t2 <- tapply(Expo, B, sum)
t3 <- t1/t2*100
plot(t3, main="Ordinary scale", ylim=c(5,25),
     xlab="BM class", ylab="Av. claims percentage")
lines(fitted(lm(t3[1:13]~I(1:13))),col="darkred")
```

The argument in `plot` giving the x-values is omitted, and then `R` uses the indices, which are just the BM-class numbers $1, \ldots, 14$. The final line adds a plot of the fitted values of a linear model explaining the first 13 elements of `t3` by their index, i.e., the corresponding bonus-malus class. This linear model gives the regression line through the first 13 data points. The formula term `I(1:13)` takes the numbers $1, \ldots, 13$ 'as is' as a covariate.

Q_{15} $\approx$ **9.6.1** Make a plot of the average number of claims against the bonus-malus class, but now on a log-scale. Use the `log=` argument, see `?plot`.
Draw a line through `exp(fitted(lm(log(t3[1:13])~I(1:13))))`. □

The actual weights corresponding to the weight classes were stored in the vector `ActualWt`. We compute the relative weights `relWt` with respect to the lightest cars (650 kg), and, in `s3`, also express the claim percentages relative to the one of this weight class.

To the plot on the ordinary scale, we add a regression line of `s3` by `relWt`.

```
relWt <- ActualWt/ActualWt[1]
s3 <- tapply(nCl,W, sum) / tapply(Expo,W, sum); s3 <- s3 / s3[1]
```

Q_{16} $\approx$ **9.6.2** Make a plot of the average number of claims `s3` by weight class against the car weight `relWt` in that class. The same with both on a logarithmic scale. Also, add regression lines. □

Computing premiums in proportion to the weight means that if c_j is the expected number of claims in weight class $j = 1, \ldots, 11$ and w_j the actual weight, then $c_j = c_1 w_j / w_1$ is assumed to hold.

In the plot on ordinary scale, the regression line is $1.02 + 0.83(w_j - w_1)/w_1 = 0.19 + 0.83 w_j / w_1$. So there is a substantial intercept; it could be treated as fixed policy costs.

In the log-log plot, one sees that the log of the average claims percentage $100 c_j / c_1$ against the log of w_j / w_1 practically equals a line through the origin, which is $(1, 1)$ on log-log scale, with

slope 0.89. So we have $\log(c_j/c_1) \approx 0.89 \log(w_j/w_1)$ or, equivalently, $c_j/c_1 \approx (w_j/w_1)^{0.89}$, and therefore asking premiums proportional to $w_j^{0.89}$ would be more appropriate.

Q_{17} = **9.6.3** Note that there were not 8316 cells in Table 9.7, but only 7524; this is because many cells had weight `Expo == 0`. The starting class for young drivers is one of the classes 2–5. This means that it is impossible to be both young and in class 11–14. Show that this leads to 792 cells that are necessarily empty. □

Q_{18} = **9.6.7** Note that loss ratios over 56% (the portfolio average `sum(TotCl)/sum(TotPrem)`, that is, the loss ratio of the entire portfolio) represent bad risks for the insurer, while under 56% they are good. Discuss if and how the rating system should be changed (consider all risk groups). In your proposal for changes to the rating system, also consider legal and competitive aspects. Explain why the loss ratios evolve with weight `WW` as they do (cf. Q_{15}). □

Q_{19} = **9.6.11** To find the most and least profitable groups of policyholders, run

```
l <- list(Use=U,Age=A,Area=R,Mile=M)
ftable(round(100*tapply(TotCl,l,sum)/tapply(TotPrem,l,sum)),
  row.vars=2, col.vars=c(1,3,4))
detach(Cars) ## don't forget
```

If you were an insurer, on which group(s) of policyholders would you target your marketing instruments? Note that you cannot focus on specific combinations of type, age, usage, region. □