1.9 Answers to all the Exercises

In the introduction we asked the reader to be an active participant by attempting all the exercises. Reading an exercise and looking at its solution immediately afterwards does not count as active participation! You should give each exercise an honest attempt and then read the proposed solution. Perhaps you and the author had a different way of approaching the problem or used a different set of R commands to achieve the same result. In any case, reading someone else's proposed solution after thinking about yours will lead to better understanding.

1. We typically organize each project inside a master folder that we keep under version control. Our subfolder structure usually looks like this:

```
Master folder
├── data
├── figs
├── report
└── src
```

The folder `data` contains our original dataset and all intermediate datasets that we create during our analysis. The folder `figs` holds all the figures that we create to include in presentations or reports. The folder `report` has all the milestone and interim documentation for the project. The folder `src` has all the scripts we have written as part of our data analysis.

Having placed the dataset `sim-modeling-data.csv` into the `data` folder, we can load it into object `dta` in our R environment as follows:

```
# Define the data path and filename
data.path <- "./data"
data.fn <- "sim-modeling-dataset.csv"
# Define column class for dataset
colCls <- c("integer",          # row id
            "character",        # analysis year
            "numeric",          # exposure
            "character",        # new business / renewal business
            "numeric",          # driver age (continuous)
            "character",        # driver age (categorical)
            "character",        # driver gender
            "character",        # marital status
            "numeric",          # years licensed (continuous)
            "character",        # years licensed (categorical)
            "character",        # ncd level
            "character",        # region
            "character",        # body code
            "numeric",          # vehicle age (continuous)
            "character",        # vehicle age (categorical)
```

```
              "numeric",          # vehicle value
              "character",        # seats
              rep("numeric", 6),  # ccm, hp, weight, length,
                                  #  width, height (all continuous)
              "character",        # fuel type
              rep("numeric", 3)   # prior claims, claim count,
                                  #  claim incurred (all continuous)
)
# read in the data with the appropriate column classes
dta <- read.csv(paste(data.path, data.fn, sep = "/"),
                colClasses = colCls)
```

All columns in `dta` will be either numeric or character. They will need some processing to convert them into factors. See script `pmaas-data.R` for all the code we used to transform the data for this text.

2. We can set our random seed so we can reproduce our answers. In `R` we can do it as follows:

```
set.seed(1092387456)
N <- dim(dta)[1] # number of rows in our dataset
dta$rnd <- runif(N, min = 0, max = 1)
```

and the calculation of the frequency over those records with a random number less than 0.6 would be

```
with(dta[dta$rnd < 0.6,], sum(clm.count) / sum(exposure))
```

To explore the variability of the frequency calculation over the training dataset, we need to repeat these calculations. Figure 1.17 shows a histogram of the frequency over the training dataset. Note that most frequencies are within 0.5% points. The calculations are as follows:

```
M <- 1000 # do calcs this many times
fq <- numeric(M) # store frequency
for(i in 1:M) {
  u <- runif(40760, min = 0, max = 1)
  fq[i] <- with(dta[u < 0.6, ], sum(clm.count) / sum(exposure))
}
library(MASS)
truehist(fq, xlab = "Frequency over re-sampled training dataset")
```

3. The frequency of each region across the entire dataset is given in Table 1.18. In `R` you can calculate the frequencies as

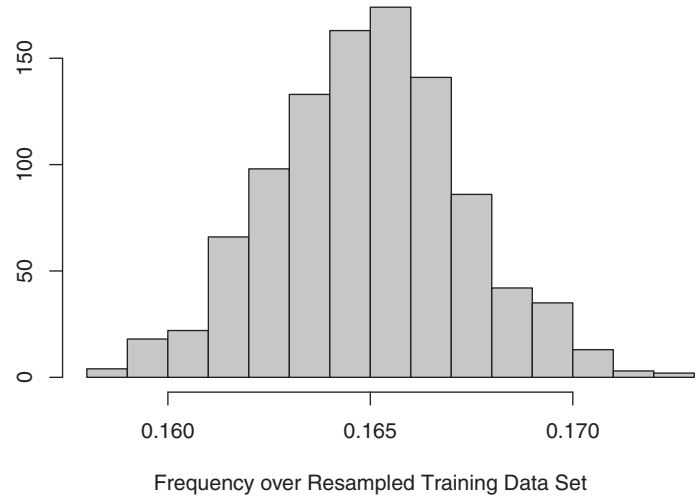

Fig. 1.17. Distribution of frequency over the training dataset as we vary the random numbers used to define the training dataset.

```
expo.region <- with(dta, tapply(exposure, region, sum))
counts.region <- with(dta, tapply(clm.count, region, sum))
fq.region <- counts.region / expo.region
```

where `dta` is the name of the entire dataset. Figure 1.18 shows the frequencies by region and by calendar year. The regions are not labeled but they have been ordered from smallest to largest frequency for the 2013 calendar year.

4. Whatever models we create *we are responsible* for making sure that they can deal sensibly with anything that the real world throws at them. In this case, if we need to quote a policy for an 88-year-old person, our model needs to respond gracefully.

Table 1.18. *Claim Frequencies (in Percentage Points) by Region Sorted from Lowest to Highest for the Entire Dataset*

Region	Freq.	Region	Freq.	Region	Freq.	Region	Freq.
7	5.4	16	13.8	11	15.5	38	18.3
2	7.4	5	13.9	30	15.8	21	18.4
27	10.9	34	14.0	19	15.8	33	18.8
22	11.2	4	14.1	1	16.3	17	19.5
20	12.0	3	14.5	10	16.5	9	20.2
29	12.2	12	14.5	24	16.6	26	21.8
28	12.8	23	14.5	36	16.7	18	22.9
35	13.7	14	14.5	6	16.8	8	23.7
13	13.8	31	14.6	25	17.4		
15	13.8	37	15.3	32	17.7		

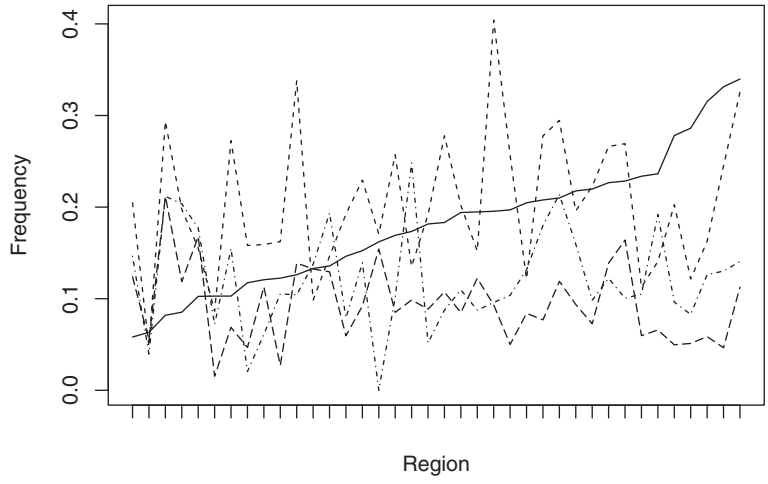

Fig. 1.18. Frequency by region and calendar year. The regions have not been labeled but are ordered by increasing frequency for the 2013 (solid line) calendar year. The dashed line is calendar year 2010, the dash-dotted line is 2011, and the long dashed line is 2012.

Three general ideas come to mind. The first idea would be to map unseen ages to known ages. For example, map $88 \to 87$. The second idea would be to augment our models to include parameters for unseen ages. And the third idea would be to bin high ages together to make a more representative group, say, bin all ages greater than 74 into bin 75+.

5. To get the exposure and number of claims from the training dataset, we use the following `R` code:

```
tmp <- subset(dta, subset = train & (drv.age == "76"))
tmp[, c("row.id", "exposure", "clm.count")]
apply(tmp[, c("row.id", "exposure", "clm.count")], 2, sum)
rm(tmp) # clean your workspace
```

where `dta` is the name of the entire dataset and `train` is the column name that indicates whether a row belongs to the training dataset.

6. See Figure 1.19. Nonetheless, these calendar year frequencies still exhibit the same overall decreasing tendency with an increasing hump around age 45. The pattern for calendar year 2013 is troubling as it differs from the other years significantly. Will the pattern in calendar year 2013 persist into 2014, or will it be more like the years 2010 to 2012?

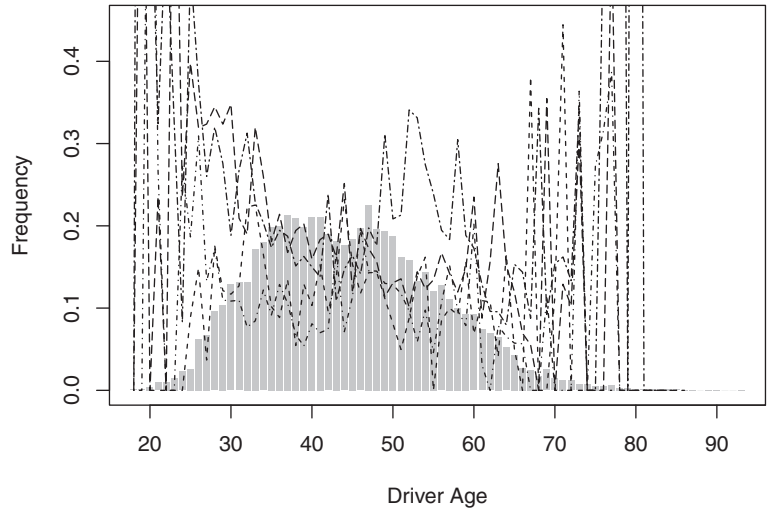

Fig. 1.19. Frequency and exposure by driver age and calendar year for the training dataset. The dashed line corresponds to calendar year 2010, the dash-dotted line is 2011, the long dashed line is 2012, and the double-dashed line is 2013. Note how the 2012 calendar year pattern (long dashed) has much higher frequencies for the younger ages. The 2013 data (double-dashed) seems to shoot up in the age range from 50 to 60.

7. The size of engine (`ccm`) is a continuous variable and is measured in cubic centimeters. The range of values in our entire dataset is [970, 3198]. One way to explore this variable would be to calculate the frequency for various ranges of engine size; in essence, answering the question: What is the frequency when the engine size is between 1560 and 1598? In `R` we could do the following:

```
bk <- unique(quantile(dta$ccm, probs = seq(0, 1, by = 0.05)))
ccm.d <- cut(dta$ccm, breaks = bk, include.lowest = TRUE)
expo <- with(dta, tapply(exposure, ccm.d, sum))
clms <- with(dta, tapply(clm.count, ccm.d, sum))
freq <- clms / expo
rm(bk, ccm.d, expo, clms, freq) # clean your workspace
```

yielding Table 1.19 (formatting not included).

8. Let's see if the variables `driver.gender` and `marital.status` can help us predict the frequency of accidents. The variable `driver.gender` has two levels, `Male` and `Female`. Over the training dataset, male drivers have had a frequency equal to 15.8%, and females have had a frequency equal to 18.4%. This suggests that gender is a variable that could help segment our policyholders. Table 1.20 shows the frequencies by calendar year and driver gender. Note that for almost every year in the training dataset, female drivers have had a higher frequency than male drivers.

Table 1.19. *Frequency by Size of Engine (ccm) for the Entire Dataset*

Size of Engine	Freq. (%)	Size of Engine	Freq. (%)
(969, 1248]	17.45	(1753, 1868]	15.89
(1248, 1398]	19.48	(1868, 1896]	14.41
(1398, 1461]	16.66	(1896, 1997]	15.23
(1461, 1560]	18.51	(1997, 2476]	13.61
(1560, 1598]	16.60	(2476, 2477]	14.18
(1598, 1753]	15.07	(2477, 3198]	17.05

As for marital status, Table 1.21 shows the frequency by calendar year and marital status. Note that the overall difference between "married" and "divorced" is very small,[13] but "widow" is about twice as large. Also there is significant variation from year to year in the individual levels. Widowers in 2012 had a very high frequency.

9. One way to investigate if the difference in frequency between married (15.8%) and widow (27.3%) drivers is significant would be to simulate the possible answers.

Let's restrict our dataset to only married and widow drivers. There are 22,761 married and 279 widow drivers for a total of 23,040 observations. For our simulation, we will randomly assign the label "married" to 22,761 observations. The remaining 279 get the label "widow." Based on this assignment, we'll compute the frequency of each group and take the difference. Repeat this procedure many times and calculate summary statistics for the difference in frequency. In R we could implement these ideas as follows:

Table 1.20. *Frequency by Calendar Year and Driver Gender for the Training Dataset*

	Gender	
Year	Male (%)	Female (%)
2010	11.6	14.5
2011	10.9	10.4
2012	18.2	24.3
2013	20.5	21.6
Total	15.8	18.4

[13] Using two decimal digits, we can see that the overall frequency for "married" is 15.82, and for "divorced" it is 15.85.

Table 1.21. *Frequency by Calendar Year and Marital Status of the Driver*

	Frequency (%) by Marital Status			
Year	Single	Married	Divorced	Widow
2010	6.9	12.1	9.2	16.5
2011	6.8	10.9	11.5	19.3
2012	26.3	18.2	17.3	45.0
2013	28.4	20.3	24.0	13.2
Total	18.7	15.8	15.8	27.3

```
set.seed(1029384756)
N <- 10000
tmp <- subset(dta,
              subset = train &
                       marital.status %in% c("Married", "Widow"),
              select = c("marital.status",
                         "exposure", "clm.count"))
f <- tmp$marital.status == "Married"
d <- numeric(N)
for(i in 1:N) {
  g <- sample(f, length(f))
  married.fq <- sum(tmp$clm.count[g]) / sum(tmp$exposure[g])
  widow.fq <- sum(tmp$clm.count[!g]) / sum(tmp$exposure[!g])
  d[i] <- married.fq - widow.fq
}
quantile(d, c(0.025, 0.05, 0.1, 0.25, 0.5,
              0.75, 0.9, 0.95, 0.975))
rm(N, tmp, f, d, i, g, married.fq, widow.fq) # clean-up
```

The 90% confidence interval for the difference in frequency is $(-0.072, 0.063)$. The actual difference we have observed of -0.115 is clearly outside this interval; therefore, this difference is statistically significant. Figure 1.20 shows a histogram of the simulated differences.

Repeating these calculations for married and single drivers, we get a 95% confidence interval equal to $(-0.039, 0.036)$. The actual difference of -0.029 is well within these bounds.

10. The coefficient for the variable `length` in the model using meters as the unit of measurement would be equal to 10 times the value of the coefficient in the model using decimeters as the unit of measurement.

Fit two models (one using `length` and another using $10 \times$ `length`) and check the estimated coefficients. In `R` we would do the following:

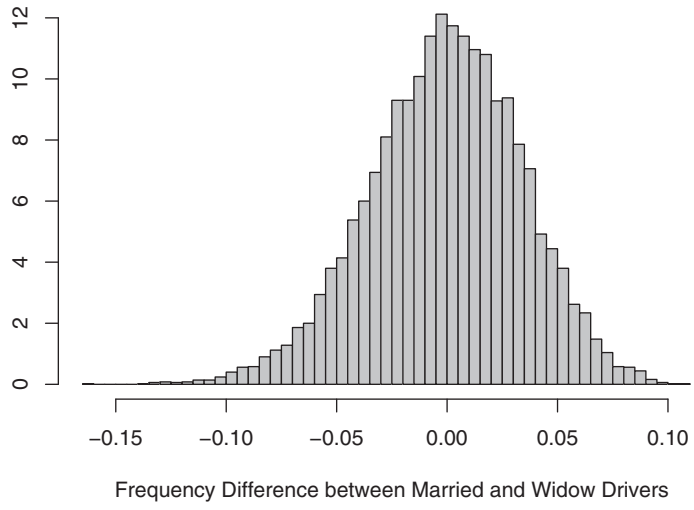

Fig. 1.20. Simulated frequency difference between married and widowed drivers based on the training dataset. The actual difference (−11.5%) between these groups is well outside the bulk of the distribution.

```
summary(glm(clm.count ~ length,
            data = dta,
            subset = train,
            family = poisson(link = "log"),
            offset = log(exposure)))
```

and

```
summary(glm(clm.count ~ I(10 * length),
            data = dta,
            subset = train,
            family = poisson(link = "log"),
            offset = log(exposure)))
```

11. In our training dataset we have 112 distinct lengths of vehicles ranging from 18.05dm to 69.45dm, that is, from about one meter and 80 centimeters to about seven meters in length. The mean length in the training dataset is 43.25dm. Table 1.22 shows selected points in the distribution of this variable. Note that the median 42.97dm of the distribution is very close to the mean 43.25dm. Also there are many vehicles with length equal to 38.64dm. In Figure 1.21 we have the distribution of vehicle length (in decimeters) from the 2.5% percentile to the 97.5 percentile. We can see that about 80% of the vehicle lengths are smaller than or equal to 4.5 meters.

Table 1.22. *Percentiles of the Distribution of Vehicle* `length` *(in Decimeters)*

Percentile	Value	Statistic	Value
0.0%	18.05		
2.5	38.64		
5.0	38.64		
10.0	38.64		
25.0	40.35		
50.0	42.97	Mean	43.25
75.0	44.05		
90.0	48.90		
95.0	50.75		
97.5	52.20		
100.0	69.45		

Note: The length 38.64 is a very popular length.

12. In Figure 1.22 we have a plot of vehicle length versus frequency. There does not appear to be any systematic effect on the frequency of accidents from the length of the vehicle. So this variable, just like `hp`, does not appear to provide any explanatory power for predicting frequency.

13. First let us create the bins for the ages. In essence we want to categorize the variable `driver.age` into three bins: (0, 34], (34, 64], and (64, 110]. We can do this via

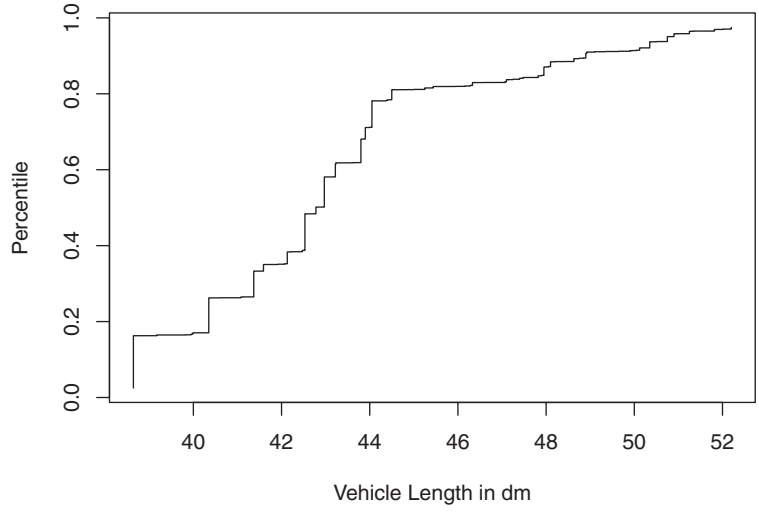

Fig. 1.21. The cummulative distribution of vehicle length from the 2.5 percentile to the 97.5 percentile for the training dataset.

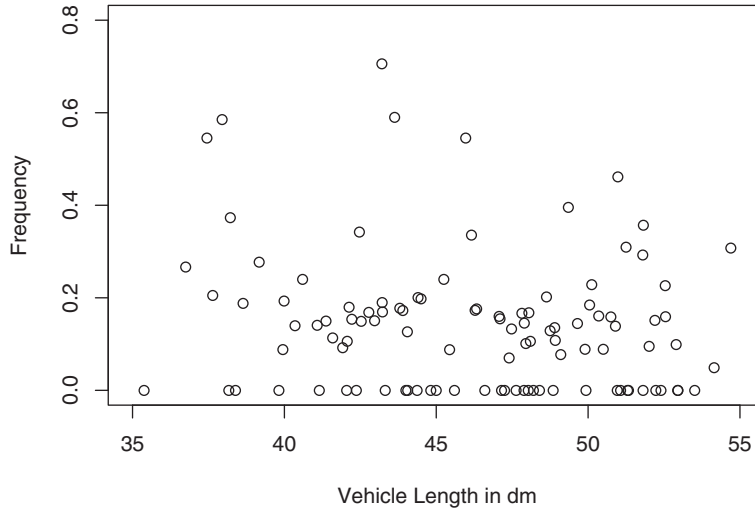

Fig. 1.22. Vehicle length versus frequency for the training dataset. We have omitted six outlying points from the plot to enhance the display of the rest of the data. The omitted points are (18.05, 0.09), (45.85, 0.92), (46.85, 1.20), (49.98, 1.20), (51.35, 1.07), and (69.45, 0.92).

```
age.bins <- cut(dta$driver.age, c(0, 34, 64, 110))
```

Now that we have the age bins, we need to collect all claims and exposure by gender and age bin. This we do as follows:

```
expo <- tapply(dta$expo, list(dta$driver.gender, age.bins), sum)
clms <- tapply(dta$clm.count, list(dta$driver.gender, age.bins), sum)
```

and finally the claim frequencies (in %) are in Table 1.23.

14. See Answers 3 and 7 and replicate.

15. Total claim counts and incurred claims over the training dataset:

```
with(dta[dta$train,], c("Claim Counts" = sum(clm.count),
                        "Claims Incurred" = sum(clm.incurred)))
```

Table 1.23. *A Two-Way Table of Claim Frequencies (in %) by Gender and Driver Age Category*

	[0, 34]	[35, 64]	[65, 110]
Male	21.57	14.86	13.15
Female	23.36	19.01	19.24

16. Looking at Answer 9, we'll simulate the difference between the severity for new versus renewal business. Adapting the R code from that answer, we have the following:

```
N <- 10000
tmp <- subset(dta, subset = train & clm.count > 0)
f <- tmp$nb.rb == "NB"
d <- numeric(N)
for(i in 1:N) {
  g <- sample(f, length(f))
  nb.sv <- with(tmp, sum(clm.incurred[ g]) / sum(clm.count[ g]))
  rb.sv <- with(tmp, sum(clm.incurred[!g]) / sum(clm.count[!g]))
  d[i] <- nb.sv - rb.sv
}
quantile(d, c(0.025, 0.05, 0.1, 0.25, 0.5,
              0.75, 0.9, 0.95, 0.975))
rm(N, tmp, f, d, i, g, nb.sv, rb.sv) # clean-up
```

The actual difference, $812 - 707 = 105$, is between the 95% and the 97.5% of the simulated distribution; so the difference is significant at the 5% level if we consider a one-sided test.

17. We want to estimate the mean for each category of `fuel.type`. Let's use linear regression to get the job done. In R we could do it this way:

```
summary(lm(I(clm.incurred/clm.count) ~ fuel.type - 1,
           data = dta, subset = train & (clm.count > 0),
           weights = clm.count))
```

This yields the following coefficients and standard errors:

```
Coefficients:
                  Estimate Std. Error
fuel.type Diesel     792.4       25.4
fuel.type Gasoline   514.8      230.2
fuel.type LPG        465.8      426.3
```

Note that the estimates for gasoline and LPG are not very precise (the standard error is quite large compared to the mean) and that the confidence intervals overlap with the confidence interval for diesel vehicles.

18. Replicate some of the analyses we have done in the text and modify them to suit your needs.

19. We need to sum up the exposure of all records with zero claims, one claim, two claims, and so forth. We can do this with the `tapply` function:

```
with(dta, tapply(expo, clm.count, sum)
```

yielding the following result:

Number of claims	0	1	2	3	4	5	6+
Total exposure	18,726.42	1,891.83	166.50	10.67	0.67	1.00	0

The mean frequency is equal to the sum of all claims divided by the sum of the exposure. This calculation can be recast as the weighted average of each record's frequency; where the weights are equal to the exposure. Namely, define f_i to be the frequency for each record

$$f_i = \frac{\text{claim count}_i}{\text{exposure}_i}$$

and w_i the weight for each record

$$w_i = \text{exposure}_i$$

Then the mean frequency is given by the expression

$$\frac{\sum_i f_i \cdot w_i}{\sum_i w_i}$$

and the second moment is equal to

$$\frac{\sum_i f_i^2 \cdot w_i}{\sum_i w_i}$$

In R we can calculate as follows:

```
f <- with(dta, clm.count / exposure) # frequency for each record
w <- with(dta, exposure) # weight for each record
mean.f <- sum(f * w) / sum(w) # mean frequency
second.f <- sum(f^2 * w) / sum(w) # second moment
var.f <- second.f - (mean.f)^2
```

The mean frequency is 16.5% and the variance is equal to 33.8%. In this case the variance is larger than the mean, and we have an overdispersed dataset.

20. We can get the total number of claims and exposure from the 29-year-old drivers by the following lines of R code:

```
with(dta, sum(clm.count[train & drv.age == 29]))
with(dta, sum(exposure[train & drv.age == 29]))
```

To estimate a generalized linear model, we can do it via

```
glm(clm.count ~ drv.age,
    data = dta,
```

Table 1.24. *Parameter Estimates for Models Using the Categorical and the Continuous Versions of the Variable Measuring the Number of Years a Driver has been Licensed*

	Categorical	Continuous
1	−1.48491	−1.53922
2	−1.73770	−1.69074
3	−1.95056	−1.84226
4	−1.87459	−1.99378
5	−2.15342	−2.14530
6	−2.48761	−2.29682
7	−2.32516	−2.44834
8+	−2.22175	−2.59987

```
    subset = train,
    family = poisson(link = "log"),
    offset = log(exposure))
```

The estimated frequency for 38-year-olds is $\exp(-2.05993) = 0.1275$, because this is our base level.

If you estimate your model using a different age as the base level for `drv.age`, then your parameter estimates will look very different, but your predictions will be identical.

21. We can estimate a model with an intercept term and a single parameter for the variable `yrs.licensed` via

```
glm(formula = clm.count ~ yrs.licensed,
   family = poisson(link = "log"), data = dta,
   subset = train, offset = log(exposure))
```

yielding an intercept equal to −1.3877 and a slope equal to −0.1515. Table 1.24 shows both sets of parameters side by side.

22. In `R` we can request that the calculation of standard errors adjust for overdispersion by using the `quasipoisson` family as follows:

```
m <- glm(clm.count ~ yrs.lic - 1,
         data = dta, subset = train,
         family = quasipoisson(link = "log"),
         offset = log(exposure))
```

Table 1.25. *Standard Errors Under the Poisson and the Quasi-Poisson Families*

yrs.lic	Estimate	Poisson Std. Error	Quasi Poisson Std. Error
1	−1.48491	0.03965	0.04085
2	−1.73770	0.04527	0.04664
3	−1.95056	0.05689	0.05861
4	−1.87459	0.06178	0.06365
5	−2.15342	0.07931	0.08170
6	−2.48761	0.11396	0.11741
7	−2.32516	0.14142	0.14570
8+	−2.22175	0.21320	0.21965

Note: The parameter estimate under both families is the same.

Fitting a Poisson and a quasi-Poisson model to the same data yields Table 1.25. Note that the parameter estimates under the Poisson and the quasi-Poisson families are the same, and only the standard errors change.

23. To create Figure 1.4, we need to fit a model for every variable we want to plot. The null model will have an intercept and the control variable `year`. The following `R` code will generate the data we need. The last two commands calculate the distance from every variable to the null model and sorts the results. The variables that are furthest away from the null model are very similar:

```
# Variables that are NOT predictors
res.vars <- c("clm.count", "clm.incurred")
mis.vars <- c("row.id", "exposure", "rnd", "train", "valid",
              "sev", "length", "height", "width", "year")

# Null frequency model
fq.null <- glm(clm.count ~ year,
               data = dta,
               subset = train,
               family = poisson(link = "log"),
               offset = log(exposure))
aic.null <- fq.null$aic
dev.null <- fq.null$deviance

# Select predictor variables
vars <- setdiff(colnames(dta),
```

```
                  c(res.vars, mis.vars,
                    "region.g1", "drv.age.gr1", "hp.cat",
                    "veh.val.pl", "driver.age.q35",
                    "driver.age.q49","driver.age.q59",
                    "veh.val.q15", "veh.val.q35"))

aic <- numeric(0)
dev <- numeric(0)

for(v in vars) {
  fmla <- as.formula(paste("clm.count ~ year", v,
                           sep = " + "))
  f <- glm(fmla,
           data = dta[dta$train,],
           family = poisson(link = "log"),
           offset = log(exposure))
  aic[v] <- f$aic
  dev[v] <- f$deviance
}

# sort variables by the distance from null* model
# using aic and dev
sort(sqrt((aic.null - aic)^2 + (dev.null - dev)^2),
      decreasing = TRUE)

# clean up
rm(res.vars,mis.vars,fq.null,aic.null,
   dev.null,vars,aic,dev,v,fmla, f)
```

24. Refitting our frequency model with the newly defined region variable gives the estimated coefficients shown in Table 1.26.

25. The vast majority of parameters are very similar. For the intercept and `driver.age`, the differences are relatively large: on the order of about 0.8 standard deviations. To compute them in `R`, use the following code:

```
m1 <- glm(clm.count ~ year + region.g1 + ncd.level +
                      driver.age + yrs.lic + prior.claims,
          data = dta, subset = train,
          family = poisson(link = "log"),
          offset = log(exposure))
```

Table 1.26. *Parameter Estimates for a Frequency Model that Includes* `region.g1`, `ncd.level`, `yrs.lic`, `driver.age`, `prior.claims`, *and the Control Variable* `year`

Variable	Level	Estimate	Std. Error
Intercept		−0.819003	0.108143
year	2010	−0.546214	0.073514
year	2011	−0.646194	0.067047
year	2012	−0.076704	0.053601
year	2013		
region.g1	R0		
region.g1	R1	−0.548768	0.124089
region.g1	R2	−0.390569	0.138929
region.g1	R3	−0.312587	0.065821
region.g1	R4	−0.246693	0.083775
region.g1	R5	−0.177825	0.086909
region.g1	R6	−0.072967	0.068625
region.g1	R7	−1.037814	0.191165
region.g1	R8	0.121544	0.094122
ncd.level	1		
ncd.level	2	−0.458627	0.206478
ncd.level	3	−0.107076	0.063750
ncd.level	4	−0.289448	0.083941
ncd.level	5	−0.398683	0.097572
ncd.level	6	−0.594338	0.092136
driver.age		−0.005540	0.002183
yrs.lic	1		
yrs.lic	2	−0.210335	0.068554
yrs.lic	3	−0.366530	0.080610
yrs.lic	4	−0.315011	0.087482
yrs.lic	5	−0.470632	0.104743
yrs.lic	6	−0.746499	0.135719
yrs.lic	7	−0.530641	0.163111
yrs.lic	8+	−0.462933	0.230580
prior.claims		0.132649	0.014242

Note: Almost all of the standard errors for `yrs.lic` are small compared to the parameter estimate. Regions R6 and R8 have rather large standard errors.

```
m2 <- update(m1, . ~ . + marital.status +
                         body.code + nb.rb)
m1.s <- summary(m1)
m1.coef <- m1.s$coefficients[,1]
m1.stderr <- m1.s$coefficients[,2]
m2.coef <- coef(m2)[1:26]
round((m1.coef - m2.coef)/m1.stderr, 2)
```

In the last line we are computing how many standard deviations from the corresponding parameter in Model 1 equals the difference in the parameter estimates.

26. We do not get a much better model by adding the variable `seats`. The deviance drops by about 4.9, but we had to estimate an additional four parameters. Not a very good trade-off!

27. To create the graph, we need to select the order in which we want to add the variables and calculate our summary statistic for each fold:

```
# Load the dataset
load("./data/all-data.RData")
# Load some functions
source("./src/fns.R")

# Select variables to inculde (in order of inclusion)
vars <- c("year", "region.g1", "ncd.level", "yrs.lic",
          "prior.claims", "drv.age.gr1", "nb.rb",
          "marital.status", "wid.dm", "vehicle.value",
          "ccm", "hp.cat", "len.dm", "driver.gender",
          "hei.dm", "seats", "veh.age", "fuel.type")
K <- length(vars)
M <- 10 # number of folds
bk <- seq(0,1,length = M+1)

rmsep <- matrix(NA, nrow = M+1, ncol = K)
t.rms <- numeric(0)
for(j in 1:K) {
  predictors <- paste(vars[1:j], sep = "", collapse = " + ")
  fmla <- as.formula(paste("clm.count ~ ", predictors, sep = ""))
  m <- glm(fmla,
           data = dta,
           subset = train,
           family = poisson(link = "log"),
           offset = log(exposure))
  pred <- predict(m, newdata = dta[dta$train, ], type = "response")
  actu <- dta[dta$train, "clm.count"]
  t.rms[vars[j]] <- RMSEP(actu, pred)

  for(i in 2:length(bk)) {
    m <- glm(fmla,
             data = dta,
             subset = train & ((rnd <= bk[i-1]) | (rnd > bk[i])),
             family = poisson(link = "log"),
             offset = log(exposure))
    pred <- predict(m, newdata = dta[dta$train &
```

```
                        ((dta$rnd > bk[i-1]) & (dta$rnd <= bk[i])), ],
                        type = "response")
    actu <- dta[dta$train & ((dta$rnd > bk[i-1]) &
                               (dta$rnd <= bk[i])), "clm.count"]
    rmsep[i-1,j] <- RMSEP(actu, pred)
  }
  rmsep[i,j] <- mean(rmsep[1:M,j])
}

yl <- range(c(rmsep[M+1,], t.rms))

op <- par(mar = c(5,4,0,0)+0.2)

plot(x = 1:K, y = rmsep[M+1,],
     type = "n",
     ylim = yl,
     xlab = "Number of Variables", ylab = "Average Error")
points(x = 1:K, y = t.rms, pch = 3, col = "black")
points(x = 1:K, y = rmsep[M+1,], pch = 16, col = "black")

text(x = 1:2, y = rmsep[11,1:2]-0.0002, labels = vars[1:2],
     adj = c(1,0.5), srt = 90)
text(x = 3:18, y = rmsep[11,3:18]+0.0002, labels = vars[3:18],
     adj = c(0,0.5), srt = 90)

par(op)

# Clean the workspace
rm(vars, K, M, bk, rmsep, t.rms, j, predictors, fmla, m,
   pred, actu, i, yl, op)
```

28. The incurred cost for each of the 17 records with the largest severities can be extracted from our dataset as follows:

```
tmp <- subset(dta, subset = clm.count > 0)
sv <- with(tmp, clm.incurred / clm.count)
o <- order(sv, decreasing = TRUE)[1:17]
tmp[o,"clm.incurred"]
```

29. The first mean was calculated by adding up all the costs and then dividing by all the claims. The second mean was calculated as the average of the record by record severities, that is, claim incurred divided by claim count for each observation.

The second method (average of individual severities) is flawed! We have not taken into account that some of the individual severities arise out of several claims. We need to do a weighted average where the weights are the number of claims.

It's important to understand how your data have been prepared. In our case, one record may represent multiple accidents.

30. In R the glm function to fit the null model would be as follows:

```
m <- glm(clm.incurred ~ 1, data = dta,
         subset = train & clm.count > 0,
         family = Gamma(link = "log"),
         weights = clm.count, offset = log(clm.count))
```

and to get the parameter estimate with its standard error, we would apply the summary function to the object m.

31. You can define a new variable in your dataset, called sev, as the ratio of claims incurred by claim counts. Then you can fit the following model:

```
m <- glm(sev ~ 1, data = dta, subset = train & clm.count >0,
         family = Gamma(link = "log"),
         weights = clm.count)
```

and use the summary function to get parameter estimates along with standard errors.

32. Yes! Go back to the frequency section and include year in all models.

33. We have that for region.g1, AIC $= 30{,}810$ and deviance $= 3{,}940$, and for drv.age.g1, we have AIC $= 30{,}807$ and deviance $= 3{,}941$.

Both values of AIC are still bigger than that of the null model at $30{,}802$ but much closer than the ungrouped versions. The previous numbers you can get with the following R code:

```
m <- glm(clm.incurred ~ year + drv.age.gr1,
         data = dta, subset = train & clm.count > 0,
         family = Gamma(link = "log"),
         weights = clm.count, offset = log(clm.count))
c(aic = m$aic, dev = m$deviance)
```

34. In Figure 1.10, 12 single variable models have been identified. The models furthest away from the null model seem to be yrs.lic, body.code, ncd.level, and the variable driver.gender. Both yrs.lic and body.code have an AIC that is bigger than the null model. Driver gender and no claim discount are equally far apart from the null model.

35. To answer this question, we need to calculate the distance between the null (including the control variable `year`) model and other models. Let's fit a null model and create a pair of vectors to record the AIC and deviance for other models. The following `R` code accomplishes this task:

```
sv.null <- glm(clm.incurred ~ year,
               data = dta[dta$train,],
               subset = clm.count > 0,
               family = Gamma(link = "log"),
               weights = clm.count,
               offset = log(clm.count))

vars <- c("nb.rb","driver.age","drv.age","driver.gender",
          "marital.status","yrs.licensed","yrs.lic","ncd.level",
          "region","body.code","vehicle.age","veh.age",
          "vehicle.value","seats","ccm","hp","weight",
          "fuel.type","prior.claims","len.dm","hei.dm","wid.dm")

aic <- numeric(0)
dev <- numeric(0)
for(v in vars) {
  fmla <- as.formula(paste("clm.incurred ~ year", v, sep = " + "))
  m <- glm(fmla,
           data = dta[dta$train,],
           subset = clm.count > 0,
           family = Gamma(link = "log"),
           weights = clm.count,
           offset = log(clm.count))
  aic[v] <- m$aic
  dev[v] <- m$deviance}
```

Now that we have all the coordinates for each model, we can calculate distances to the null model. Note that the distance from the null model to `driver.gender` and `ncd.level` is about 17.75:

```
dist <- sqrt((sv.null$aic - aic)^2 + (sv.null$deviance - dev)^2)
sort(dist, decreasing = TRUE)
```

Next we can plot the points and add the circle centered at the null model and passing through the two models of interest. There are three points that are very far from the null model: `drv.age`, `region`, and `veh.age`. We'll omit these points from the graph:

```
op <- par(mar = c(4,4,0,0)+0.2)
plot(x = aic, y = dev,
     xlim = c(30792, 30808), ylim = c(3930, 3952),
     xlab = "An Information Criterion", ylab = "Deviance")
```

```
points(x = sv.null$aic, y = sv.null$deviance,
       pch = 16, col = "red")
x <- seq(30790, 30815, length = 100)
y <- sv.null$deviance - sqrt(17.75^2 - (x - sv.null$aic)^2)
lines(x, y, col = "grey")
par(op)
rm(sv.null,vars,aic,dev,v,fmla,m,dist,op,x,y)
```

36. In R we can say the following:

```
summary(glm(clm.incurred ~ year + ncd.level,
            family = Gamma(link = "log"),
            data = dta, weights = clm.count,
            subset = train & clm.count > 0,
            offset = log(clm.count)))
```

The output we get from this model is as follows:

```
Call:
glm(formula = clm.incurred ~ year + ncd.level,
    family = Gamma(link = "log"),
    data = dta, weights = clm.count,
    subset = train & clm.count > 0,
    offset = log(clm.count))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.4446  -1.6703  -0.6898   0.3154   3.9064

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.92737    0.06431 107.722  < 2e-16 ***
year2010    -0.35940    0.10290  -3.493 0.000489 ***
year2011    -0.42138    0.09375  -4.495  7.4e-06 ***
year2012    -0.18263    0.07515  -2.430 0.015183 *
ncd.level2  -0.05055    0.28494  -0.177 0.859212
ncd.level3  -0.09518    0.07878  -1.208 0.227145
ncd.level4  -0.14245    0.10373  -1.373 0.169809
ncd.level5  -0.16950    0.12256  -1.383 0.166811
ncd.level6  -0.29755    0.10486  -2.838 0.004594 **
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Gamma family taken to be 1.968526)

    Null deviance: 3997.8  on 1855  degrees of freedom
```

```
Residual deviance: 3934.3  on 1847  degrees of freedom
AIC: 30800

Number of Fisher Scoring iterations: 8
```

Note that only level 6 is significant. The rest are not. The value of the parameters are decreasing monotonically (from 0 for level 1 down to -0.6 for level 6). This seems reasonable in that more experienced drivers may have less severe accidents.

37. We can fit a generalized linear model with the variable `driver.gender` as follows:

```
m <- glm(clm.incurred ~ year + driver.gender, data = dta,
         subset = train & clm.count > 0,
         family = Gamma(link = "log"),
         weights = clm.count, offset = log(clm.count))
```

This model gives an intercept equal to 6.81151 and a parameter estimate for females equal to 0.26043. Both of these parameters are statitically significant. In fact, the deviance drops by almost 16 points (from 3,952.0 down to 3,936.3) when we introduce the gender of the driver. This variable does help us differentiate severities.

38. Based on Figure 1.10, the next best variable seems to be `marital.status`. An analysis of variance table shows that adding this variable improves the deviance by 14 points, and we have to estimate three additional parameters. This is a significant improvement at the 10% level.

39. We can compute the required information as follows:

```
m <- glm(clm.incurred ~ year + vehicle.value, data = dta,
         subset = train & clm.count > 0,
         family = Gamma(link = "log"),
         weights = clm.count, offset = log(clm.count))
summary(m)
```

The value of the estimated parameter is equal to -0.002177 with a standard error equal to 0.003712. The negative sign seems counterintuitive.

40. The vast majority of people draw a curve that looks very different from the computer generated one. Most people start their hand drawn curve at a severity of about 1,000, and they steadily increase it to about 2,500 when they reach a vehicle value equal to approximately 22. At this point they start decreasing to a value of 1,000 when they reach vehicle values close to 40 and they make an inflection and run their smooth curve down to a value of close to zero when they reach 80.

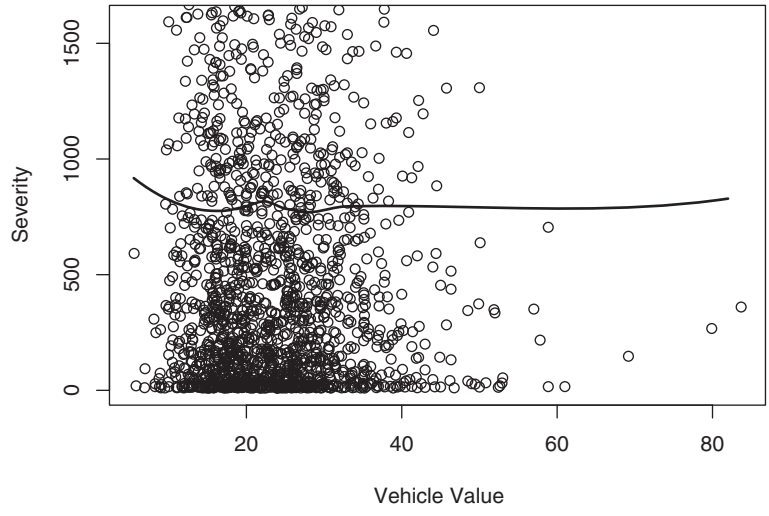

Fig. 1.23. Scatterplot of vehicle value versus severity. A loess smooth has been added to the plot.

When looking at the scatterplot in Figure 1.11, most people do not fully take into account the sheer number of observations that make up the lower portion of the display. They are misled by the observations near the top of the scatterplot.

There are many different kinds of scatterplot smooths. We will use one known as loess. This smooth is based on a set of local regressions. Figure 1.23 shows the same graph as Figure 1.11, but with the loess smooth superimposed. We have restricted the range of the y-axis to values up to 1,500 to enhance the display of the smooth. About 86% of all severities are below 1,500. Note how the smooth sharply decreases for vehicle values in the range $(5, 20)$. Then there is a slight upward hump and decrease in the range $(20, 30)$. For values of 40 or more, there is a steady (near-linear) decline.

The graph with the smooth can be created with the following `R` code:

```
tmp <- subset(dta, subset = train & (clm.count > 0))
lo.fit <- loess(sev ~ vehicle.value, data = tmp)
xs <- seq(5.5, 82, length = 100)
lo.sev <- predict(lo.fit, data.frame(vehicle.value = xs))
plot(x = tmp$vehicle.value, y = tmp$sev,
     ylim = c(0,1600),
     xlab = "Vehicle Value",
     ylab = "Severity")
lines(x = xs, y = lo.sev, col = "red", lwd = 2)
```

41. Let $x_0 = 0, x_1 = 15, x_2 = 35$, and $x_3 = 85$ be a partition of the interval over which we want to have our piecewise linear function. Also define the following four functions: $p_0(x) = 1$, $p_1(x) = x$, $q_1(x) = (x - x_1)_+$, and $q_2(x) = (x - x_2)_+$; where

$$(x - w)_+ = \begin{cases} 0 & \text{if } x \leq w \\ x - w & \text{if } x > w \end{cases}$$

The four functions $p_0(x)$, $p_1(x)$, $q_1(x)$, and $q_2(x)$ will allow us to construct any piecewise linear function over the segments $[0, 15)$, $[15, 35)$, and $[35, 85)$. If we wanted more segments in our piecewise linear function, we would need additional q-functions to match the additional x-points at which to change the slopes of the lines segments.[14]

42. The slope of the first segment is -2 and the y-intercept is 45, the slope of the second segment is 0, and the slope of the last segment is $-1/5$. Therefore, the piecewise linear function, $f(x)$, passing through the given points is

$$f(x) = 45p_0(x) - 2p_1(x) + 2q_1(x) - \frac{1}{5}q_2(x)$$

43. After adding two additional columns to our dataset to represent the functions $q_1(x)$ and $q_2(x)$, we can estimate a severity model with the variables `year`, `vehicle.value`, $q_1(x)$, and $q_2(x)$. The parameter estimates are in Table 1.27. Notice

Table 1.27. *Parameter Estimates for a gamma Log-Link Severity Model with* `year` *and a Piecewise Linear Function of* `vehicle.value` *with Knots at* 15 *and* 35

Variable	Level	Estimate	Std. Error
Intercept		7.120486	0.444910
year2010	2010	−0.362719	0.103465
year2011	2011	−0.407221	0.094350
year2012	2012	−0.175135	0.075633
vehicle.value		−0.019350	0.031154
veh.val.q15		0.020081	0.033691
veh.val.q35		−0.008127	0.014654

Note: The parameter values for `vehicle.value` and $q_1(x)$ (`veh.val.q15`) are of opposite sign and nearly equal in magnitude. The value for $q_2(x)$ (`veh.val.q35`) is about 40% the size of `vehicle.value`.

[14] The points at which the slope of the line segments change are known as knots. We usually work over an interval, and the interval end points are known as *boundary knots*, whereas the other points are *interior knots*.

that the estimated values for `vehicle.value` and $q_1(x)$ (named in our dataset `veh.val.q15`) are almost identical but of opposite sign. This gives us essentially a horizontal segment. Also the estimated parameter for $q_2(x)$ (in our dataset known as `veh.val.q35`) has the same sign as `vehicle.value` but is about 40% the size.

44. From the answer to Exercise 1.43 we can see that none of the parameters for our piecewise linear function of `vehicle.value` have been estimated with a lot of precision (except the intercept term). The standard errors are high compared to the value of the respective parameters. This does not automatically rule out the use of this variable! We need to estimate the combined effect of the three variables that make up our piecewise linear function – similar to what the `anova` function provides. In Table 1.28 we have the summary statistics. Our piecewise linear function is not statistically significant and from this perspective we can remove it from our model.

To generate all the entries, we need to fit three models – the null model, null plus `year`, and null plus `year` plus our piecewise linear function of vehicle value:

```
m0 <- glm(clm.incurred ~ 1,
          data = dta, subset = train & clm.count > 0,
          family = Gamma(link = "log"),
          weights = clm.count,
          offset = log(clm.count))
m1 <- update(m0, . ~ . + year)
m2 <- update(m1, . ~ . + vehicle.value +
                         veh.val.q15 +
                         veh.val.q35)
```

The first row of the table can be filled with the values from applying the `summary` function to `m0`. Similarly, the second row can be filled with the summary of `m1`. For the last row we need to do a bit more work. The residual degrees of freedom and the residual deviance come straight out of the summary of object `m2`. The entries in the

Table 1.28. *Analysis of Deviance Table for Our Piecewise Linear Function of the Variable* `vehicle.value`

Variable	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			1,855	3,997.8	
year	3	45.789	1,852	3,952.0	0
vehicle.age PL	3	1.728	1,849	3,950.3	0.834

Note: We estimated three additional parameters and got a reduction in deviance of 1.7 deviance points. This piecewise linear function is not statistically significant.

column named "Deviance" are the row differences from the column "Resid. Dev." For the last row and last column we need the following calculations:

```
dispersion <- sum(m2$weights * m2$residuals^2)/m2$df.residual
dev <- m1$deviance - m2$deviance
df <- m1$df.residual - m2$df.residual
pchisq(dev/dispersion, df, lower.tail = FALSE)
```

45. In R we can fit the model as follows:

```
m <- glm(clm.incurred ~ year + region,
         data = dta, subset = train & clm.count > 0,
         family = Gamma(link = "log"),
         weights = clm.count, offset = log(clm.count))
```

and using the `summary` function on the object `m` yields a table of parameter estimates where four regions have statistically significant estimates, namely, regions 4, 21, 26, and 30.

The base level for the variable `region` is region 17. If you choose a different base level your estimated parameters will look very different, but your predictions will be the same.

46. Fit models for each variable. Check parameter estimates and think about grouping levels together or fitting a piecewise linear function where appropriate.

47. Let us include `drv.age` in our current severity model and look at the estimated parameters. In Figure 1.24 we have a graphical representation of the parameter estimates for driver age. The vast majority of parameters are not statistically significant at the 5% level; that is, their 95% confidence interval crosses the zero line in Figure 1.24. The analysis of variance in Table 1.29 shows that the variable `drv.age` is not statistically significant. But from a business perspective we probably would like to keep this variable in our model. We should look to see if by grouping ages together we can improve our model. A good starting place for the grouping would be in quinquenial ages.

48. We first need to estimate a severity model and compute the residuals. The graph can be generated with the following code:

```
library(statmod)
m <- glm(clm.incurred ~ year + marital.status + drv.age,
         data = dta, subset = train & clm.count > 0,
         family = Gamma(link = "log"),
         weights = clm.count, offset = log(clm.count))
qresid <- qresiduals(m)
```

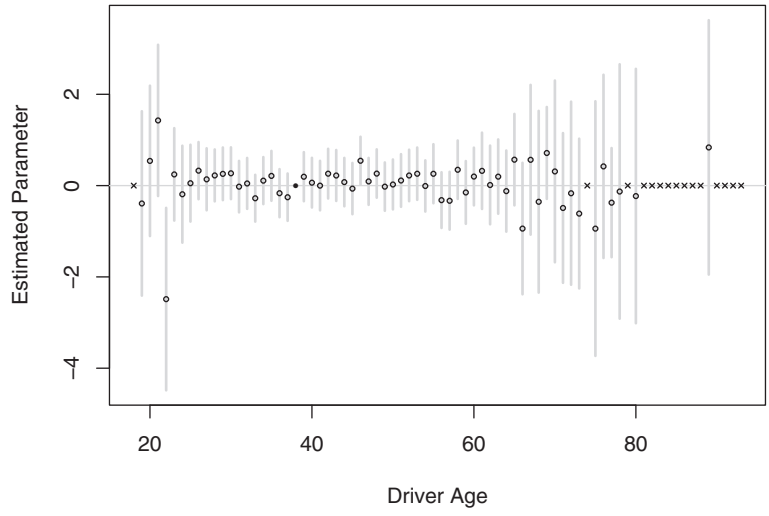

Fig. 1.24. Estimated severity parameters for `drv.age`. Twelve age categories have no reported claims, and the parameters for these levels should be equal to negative infinity. We have displayed them with a cross symbol and at a value of zero. The base level for this variable is at age = 38, and it is shown as a solid dot with a value of zero. The gray lines represent 95% confidence intervals for each parameter.

```
drv <- dta$drv.age[dta$train & dta$clm.count > 0]
plot(x = drv, y = qresid,
     ylim = c(-max(qresid), max(qresid)),
     xlab = "Driver Age", ylab = "Quantile Residuals")
abline(h = 0, col = "gray")
```

If you prefer a scatterplot, then replace `drv.age` with the variable `driver.age` when computing `drv`. There is a very slight increase in the location of the residuals as drivers become older.

I would start with a grouping in quinquennial ages: 18–22, 23–27, and so forth. Re-estimate the model and check residuals again. The main disadvantage of using this

Table 1.29. *Analysis of Deviance for a Single Model*

Variable	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			1,855	3,997.8	
year	3	45.789	1,852	3,952.0	0
marital.status	3	14.062	1,849	3,938.0	0.069
drv.age	60	101.160	1,789	3,836.8	0.790

Note: The variables are added sequentially. Note that for `drv.age` we need to estimate 60 parameters, and we get a reduction in deviance of about 100 points – a very bad trade-off.

type of grouping is that as customers age and transition from one group to the next, there can be a sizable change in their premium. This price change is not intuitive for the customer. To reduce these potential price dislocations, we could use a piecewise linear polynomial with interior knot points at 22, 27, 32, . . .

49. The variable `driver.gender` is categorical with levels "Male" and "Female." The level "Male" is the base level, and the estimated parameter value for "Female" is 0.26043, with a standard error equal to 0.09436.

50. Replicate some the analyses we have done in the text and exercises on the other variables.

51. Perhaps some piecewise linear functions of `weight` might help. Also consider adding other variables. Maybe group some of the levels of a categorical variable. We could also add some interactions between variables.

52. The frequency model can be estimated via

```
fq.m <- glm(clm.count ~ year + ncd.level + drv.age.gr2 +
                        yrs.lic + region.g1 + prior.claims,
            data = dta,
            subset = train,
            family = poisson(link = "log"),
            offset = log(exposure))
```

and we can get a summary of the coefficients by applying the `summary` function to the object `fq.m`.

The severity model can be estimated via

```
sv.m <- glm(clm.incurred ~ year + marital.status +
                           driver.gender + weight +
                           body.code,
            data = dta,
            subset = train & clm.count > 0,
            family = Gamma(link = "log"),
            offset = log(clm.count))
```

and we can extract information on the coefficients by using the `summary` function.

53. To calculate the expected frequency and severity for a particular row in our dataset, we need to know the values of the variables used in the model and the estimated parameters. Table 1.30 shows the calculation of expected frequency for `row.id` 3764. Similar calculations are needed for severity (see Table 1.31).

Table 1.30. *Calculation of Expected Frequency for* `row.id` 3764

Variable	Value	Estimated Coeff.
Intercept		−1.2178
`year`	2013	0.0
`ncd.level`	6	−0.5829
`drv.age.gr2`	48–52	0.1837
`yrs.lic`	3	−0.3634
`region.g1`	R3	−0.3231
`prior.claims`	2	0.2704
`exposure`	7/12	−0.5390
Sum of Coeff.		−2.5721
Expected frequency		0.0764

Note: The expected frequency (last row) is $\exp(-2.5721) = 0.0764$. This record was exposed for seven months, so the offset for exposure is equal to $\log(7/12)$. The estimated coefficient for `prior.claims` is equal to 2×0.1352.

54. In the training dataset we have about 24,500 records, so identifying the extreme 1% on each side of the distribution of pure premiums will yield about 490 records. The 1 percentile is at 3.48, and the 99 percentile is at 296.58.

The distribution of exposures along the `ncd.level` variable for the training dataset and the low and high pure premium records shows significant differences. The

Table 1.31. *Calculation of Expected Severity for* `row.id` 3764

Variable	Value	Estimated Coeff.
Intercept		6.2634
`year`	2013	0.0
`marital.status`	Divorced	0.0019
`driver.gender`	Male	0.0
`weight`	1090	0.3898
`body.code`	D	0.0482
Sum of coeff.		6.7033
Expected severity		815.0912

Note: The expected severity (last row) is $\exp(6.7033) = 815.0912$. The estimated coefficient for `weight` is equal to 1090×0.00035765.

Table 1.32. *Proportion of Exposures on the Training Dataset and the One and Ninety-Nine Percentile Sets of the Pure Premium Distribution*

ncd.level	1	2	3	4	5	6
Training (%)	30.67	1.46	23.03	13.89	10.16	20.80
Low pure premium (%)	4.35	1.19	8.30	7.91	11.86	66.40
High pure premium (%)	82.21	0.93	12.50	2.48	1.08	0.81

low pure records are heavily skewed toward no claim discount level 6, and the high pure premium records are skewed toward level 1. Table 1.32 shows the proportion of exposures for the entire training dataset along with the low and high pure premium records. Apply the same idea to investigate the remaining variables included in the frequency and severity models.

55. In R we can score our training dataset and check total pure premiums against claims incurred as follows:

```
fq.m <- glm(clm.count ~ year + ncd.level +
            drv.age.gr2 + yrs.lic + region.g1 +
            prior.claims,
            family = poisson(link = "log"),
            data = dta,
            subset = train,
            offset = log(exposure))
fq <- score(grab.coef(fq.m), newdata = dta[dta$train,],
            offset = log(dta[dta$train, "exposure"]))$mu

sv.m <- glm(clm.incurred ~ year + marital.status +
             driver.gender + weight + body.code,
             family = Gamma(link = "log"),
             data = dta,
             subset = train & clm.count > 0,
             offset = log(clm.count))
sv <- score(grab.coef(sv.m), newdata = dta[dta$train,])$mu

pp <- fq * sv
c(pp = sum(pp), ci = sum(dta[dta$train, "clm.incurred"]),
  ratio = sum(pp)/sum(dta[dta$train, "clm.incurred"]))
```

56. First we need to estimate a frequency model with no restrictions. The parameter estimates for this unrestricted model are given in Table 1.33.

Table 1.33. *Unrestricted Parameter Estimates for a Poisson Log-Link Frequency Model with the Variables* `year`, `ncd.level`, `driver.gender`

Variable	Level	Estimate	Std. Error
(Intercept)		−1.27053	0.04718
year	2010	−0.54683	0.07320
year	2011	−0.64536	0.06674
year	2012	−0.07933	0.05350
year	2013	0.0	
ncd.level	1	0.0	
ncd.level	2	−0.45621	0.20295
ncd.level	3	−0.22177	0.05607
ncd.level	4	−0.50271	0.07392
ncd.level	5	−0.61570	0.08732
ncd.level	6	−0.93100	0.07469
driver.gender	Male	0.0	
driver.gender	Female	0.11997	0.06742

From Table 1.33 we can see that levels 4, 5, and 6 have discounts of 40%, 45%, and 60%, respectively. Level 2 is also way off from the desired percentage.

In our modeling environment we need to create an offset variable that captures these desired discounts (and since we are estimating a frequency model also the exposure of each record). In R we can do this as follows:

```
# create a restricted ncd.level variable
disc <- c("1" = 0.00, "2" = 0.10, "3" = 0.20,
          "4" = 0.25, "5" = 0.30, "6" = 0.35)
# calc discount on the scale of the linear predictor
ncd.level.r <- log(1 - disc)

# create the offset as a new variable in our dataset
dta$ofst <- ncd.level.r[dta$ncd.level] + log(dta$exposure)

# fit a restricted model
fq.mr <- glm(clm.count ~ year + driver.gender,
             family = poisson(link = "log"),
             data = dta,
             subset = train,
             offset = ofst)
summary(fq.mr)
```

Table 1.34. *The Unrestricted and Restricted Coefficients for a Poisson Log-Link Frequency Model*

Variable	Level	Unrestricted Estimate	Restricted Estimate
(Intercept)		−1.27053	−1.39190
year	2010	−0.54683	−0.54991
year	2011	−0.64536	−0.64464
year	2012	−0.07933	−0.08397
year	2013	0.0	0.0
ncd.level	1	0.0	0.0
ncd.level	2	−0.45621	−0.10536
ncd.level	3	−0.22177	−0.22314
ncd.level	4	−0.50271	−0.28768
ncd.level	5	−0.61570	−0.35667
ncd.level	6	−0.93100	−0.43078
driver.gender	Male	0.0	0.0
driver.gender	Female	0.11997	0.13452

Note: The restriction is on the `ncd.level` variable. The restricted discounts are 0%, 10%, 20%, 25%, 30%, and 35%. These values are in the scale of the response variable. The values you see under the restricted column are in the scale of the linear predictor; thus, for no claim discount level 5, we have $1 - e^{-0.35667} = 0.30$.

The unrestricted and restricted coefficients are in Table 1.34. Notice how the variable `year` changed a bit, but the intercept and `driver.gender` picked up most of the changes.

57. We need to calculate actual and predicted claim counts and claims incurred. In `R` we can do it as follows:

```
a.cc <- c("train" = with(dta[dta$train,],
            tapply(clm.count, driver.gender, sum)),
          "valid" = with(dta[dta$valid,],
            tapply(clm.count, driver.gender, sum)))
p.fq <- c("train" = with(dta[dta$train,],
            tapply(efq, driver.gender, sum)),
          "valid" = with(dta[dta$valid,],
            tapply(efq, driver.gender, sum)))
a.ci <- c("train" = with(dta[dta$train,],
            tapply(clm.incurred, driver.gender, sum)),
          "valid" = with(dta[dta$valid,],
            tapply(clm.incurred, driver.gender, sum)))
```

Table 1.35. *Actual Versus Expected Claim Counts and Claims Incurred by the Variable* `driver.gender`

		Training		Validation	
Var		Male	Female	Male	Female
CC	P	1,773.4	229.6	1,178.0	157.3
	A	1,751	252	1,220	208
	R	1.013	0.911	0.966	0.756
CI	P	1,354,910	229,189	904,149	155,933
	A	1,336,294	241,965	954,155	178,774
	R	1.014	0.947	0.948	0.872

Note: The following abbreviations are used: Var, variable; CC, claim counts; CI, claims incurred; P, predicted; A, actual; and R, ratio of predicted to actual.

```
p.pp <- c("train" = with(dta[dta$train,],
            tapply(epp, driver.gender, sum)),
          "valid" = with(dta[dta$valid,],
            tapply(epp, driver.gender, sum)))
```

With these items we can construct Table 1.35.

58. See the answer to Exercise 1.57 for guidance on how to proceed.

59. Suppose we had an ordered set of observations $x_{(1)} \leq \cdots \leq x_{(n)}$; then we would plot the points $(i/n, L(i/n))$ where

$$L(i/n) = \frac{\sum_{j=1}^{i} x_{(j)}}{\sum_{j=1}^{n} x_{(j)}}$$

Figure 1.25 shows a typical Lorenz curve (below the 45° line).

To illustrate these calculations, let 40.3, 6.8, 3.9, 7.5, 7.1, 3.8, 4.6, 5.8, 8.9, 3.9 be a random sample of observations.

First we need to sort these observations and calculate their cumulative proportion. In the following table we have the original data in the first row, then the sorted data, the cumulative data, and finally, in the last row, the cumulative data divided by the sum of all data points:

Original	40.3	6.8	3.9	7.5	7.1	3.8	4.6	5.8	8.9	3.9
Sorted	3.8	3.9	3.9	4.6	5.8	6.8	7.1	7.5	8.9	40.3
Cum	3.8	7.7	11.6	16.2	22.0	28.8	35.9	43.4	52.3	92.6
Prop (%)	4.1	8.3	12.5	17.5	23.8	31.1	38.8	46.9	56.5	100.0

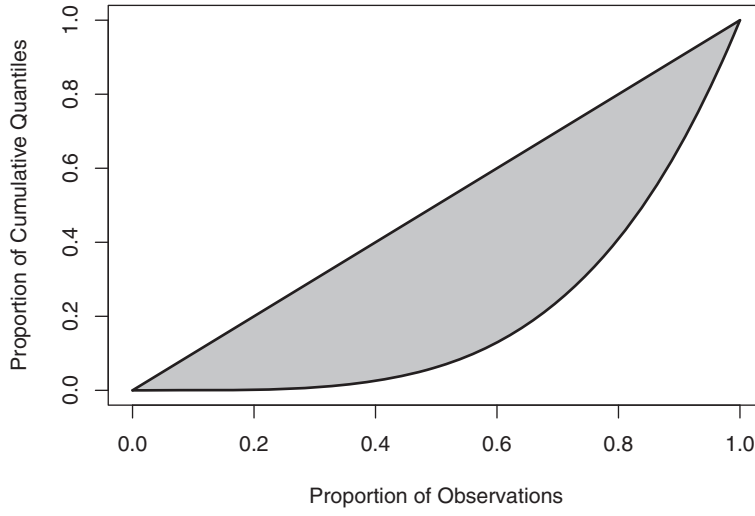

Fig. 1.25. The Lorenz curve for a distribution function. Twice the shaded area is the Gini index, and the 45° line is known as the line of equality.

The Lorenz curve is then the plot of the points $(0, 0)$, $(1/10, 0.041)$, $(2/10, 0.083)$, $\ldots$, $(9/10, 0.565)$, and $(10/10, 1.000)$.

The Gini index is twice the area between the 45° line and the Lorenz curve. We can approximate this area as

$$1 - \sum_{i=1}^{10} (a_i - a_{i-1})\,(b_i + b_{i-1})$$

where $a_i = i/10$ and $b_0 = 0$, $b_1 = 0.041$, $b_2 = 0.083, \ldots, b_9 = 0.565$, and $b_{10} = 1.000$. The Gini index is equal to 0.579.