Ah, so the **family** argument is a function! There are built-in ones we can use, such as the **poisson** one we used above, or a user could define her own custom function.

Well, then, what are the arguments to such a function? A key argument is **link**, which is obviously the link function $q^{-1}()$ discussed above, which we found to be **log()** in the Poisson case.

For a logistic model, as noted earlier, $F_{Y|X}$ is binomial with number of trials m equal to 1. Recall that the variance of a binomial random variable with m trials is $mr(1-r)$, where r is the "success" probability on each trial, Recall too that the mean of a 0-1-valued random variable is the probability of a 1. Putting all this together, we have

$$\sigma^2(t) = \mu(t)[1 - \mu(t)] \tag{4.17}$$

Sure enough, this appears in the code of the built-in function **binomial()**:

```
> binomial
function (link = "logit")
{
...
    variance <- function(mu) mu * (1 - mu)
```

Let's now turn to details of two of the most widely-used models, the logistic and Poisson.

4.3 GLM: the Logistic Model

The logistic regression model, introduced in Section 1.1, is by far the most popular nonlinear regression method. Here we are predicting a response variable Y that takes on the values 1 and 0, indicating which of two classes our unit belongs to. As we saw in Section 1.17.1, this indeed is a regression situation, as $E(Y \mid X = t)$ reduces to $P(Y = 1 \mid X = t)$.

The model, again, is

$$P(Y = 1 \mid X = (t_1, ..., t_p)) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 t_1 + ... + \beta_p t_p)}} \tag{4.18}$$

4.3.1 Motivation

We noted in Section 1.1 that the logistic model is appealing for two reasons: (a) It takes values in [0,1], as a model for probabilities should, and (b) it is monotone in the predictor variables, as in the case of a linear model, a common situation in practice.

But there's even more reason to choose the logistic model. It turns out that the logistic model is implied by many familiar distribution families. In other words, there is often good theoretical justification for using the logit.

To illustrate that, consider a very simple example of text classification, involving Twitter tweets. Suppose we wish to automatically classify tweets into those involving financial issues and all others. We'll do that by having our code check whether a tweet contains words from a list of financial terms we've set up for this purpose, say *bank*, *rate* and so on.

Here Y is 1 or 0, for the financial and nonfinancial classes, and X is the number of occurrences of terms from the list. Suppose that from past data we know that *among financial tweets*, the number of occurrences of words from this list has a Poisson distribution with mean 1.8, while for nonfinancial tweets the mean is 0.2. Mathematically, that says that $F_{X|Y=1}$ is Poisson with mean 1.8, and $F_{X|Y=0}$ is Poisson with mean 0.2. (Be sure to distinguish the situation here, in which $F_{X|Y}$ is a Poisson distribution, from Poisson regression, in which it is assumed that $F_{Y|X}$ is Poisson.) Finally, suppose 5% of all tweets are financial.

Recall once again (Section 1.17.1) that in the classification case, our regression function takes the form

$$\mu(t) = P(Y = 1 \mid X = t) \tag{4.19}$$

Let's calculate this function:

$$\begin{aligned} P(Y = 1 \mid X = t) &= \frac{P(Y = 1 \text{ and } X = t)}{PX = t)} \\ &= \frac{P(Y = 1 \text{ and } X = t)}{P(Y = 1 \text{ and } X = t \text{ or } Y = 0 \text{ and } X = t)} \\ &= \frac{\pi \; P(X = t \mid Y = 1)}{\pi \; P(X = t \mid Y = 1) + (1 - \pi) \; P(X = t \mid Y = 0)} \end{aligned} \tag{4.20}$$

where $\pi = P(Y = 1)$ is the population proportion of individuals in class 1.

The numerator in (4.20) is

$$0.05 \cdot \frac{e^{-1.8}\ 1.8^t}{t!} \tag{4.21}$$

and similarly the denominator is

$$0.05 \cdot \frac{e^{-1.8}\ 1.8^t}{t!} + 0.95 \cdot \frac{e^{-0.2}\ 0.2^t}{t!} \tag{4.22}$$

Putting this into (4.20) and simplifying, we get

$$P(Y = 1 \mid X = t) = \frac{1}{1 + 19e^{1.6(\frac{1}{9})^t}} \tag{4.23}$$

$$= \frac{1}{1 + \exp(\log 19 + 1.6 - t \log 9)} \tag{4.24}$$

That last expression is of the form

$$\frac{1}{1 + \exp[-(\beta_0 + \beta_1 t)]} \tag{4.25}$$

with

$$\beta_0 = -\log 19 - 1.6 \tag{4.26}$$

and

$$\beta_1 = \log 9 \tag{4.27}$$

In other words the setting in which $F_{X|Y}$ is Poisson implies the logistic model!

This is true too if $F_{X|Y}$ is an exponential distribution. Since this is a continuous distribution family rather than a discrete one, the quantities $P(X = t|Y = i)$ in (4.23) must be replaced by density values:

$$P(Y = 1 \mid X = t) =$$

$$\frac{\pi \ f_1(X = t \mid Y = 1)}{\pi \ f_1(X = t \mid Y = 1) + (1 - \pi) \ f_0(X = t \mid Y = 0)} \tag{4.28}$$

where the within-class densities of X are

$$f_i(w) = \lambda_i e^{-\lambda_i w}, \quad i = 0, 1 \tag{4.29}$$

After simplifying, we again obtain a logistic form.

Most important, consider the multivariate normal case (Section 2.6.2): Say for groups $i = 0, 1$, $F_{X|Y=i}$ is a multivariate normal distribution with mean vector μ_i and covariance matrix Σ, where the latter does *not* have a subscript i. This is a generalization of the classical two-sample t-test setting, in which two (scalar) populations are assumed to have possibly different means but the same variance.[5] Again using (4.28), and going through a lot of algebra, we find that again $P(Y = 1 \mid X = t)$ turns out to have a logistic form,

$$P(Y = 1 \mid X = t) = \frac{1}{1 + e^{-(\beta_0 + \overline{\beta}' t)}} \tag{4.30}$$

with

$$\beta_0 = \log(1 - \pi) - \log \pi + \frac{1}{2}(\mu_1' \mu_1 - \mu_0' \mu_0) \tag{4.31}$$

and

$$\overline{\beta} = (\mu_0 - \mu_1)' \Sigma^{-1} \tag{4.32}$$

where t is the vector of predictor variables, the β vector is broken down into $(\beta_0, \overline{\beta})$, and π is $P(Y = 1)$. The messy form of the coefficients here is not important; instead, the point is that we find that the multivariate normal model implies the logistic model, giving the latter even more justification.

In summary:

> Not only is the logistic model intuitively appealing because it is a monotonic function with values in (0,1), but also because it

[5] It is also the setting for *Fisher's Linear Discriminant Analysis*, to be discussed in Section 5.6.

is implied by various familiar parametric models for the within-class distribution of X.

No wonder the logit model is so popular!

4.3.2 Example: Pima Diabetes Data

Another famous UCI data set is from a study of the Pima tribe of Native Americans, involving factors associated with diabetes. There is data on 768 women.[6] Let's predict diabetes from the other variables:

```
> pima <- read.csv('pima-indians-diabetes.data')
> head(pima)
  NPreg Gluc BP Thick Insul  BMI Genet Age Diab
1     6  148 72    35     0 33.6 0.627  50    1
2     1   85 66    29     0 26.6 0.351  31    0
3     8  183 64     0     0 23.3 0.672  32    1
4     1   89 66    23    94 28.1 0.167  21    0
5     0  137 40    35   168 43.1 2.288  33    1
6     5  116 74     0     0 25.6 0.201  30    0
# Diab = 1 means has diabetes
> logitout <- glm(Diab ~ .,data=pima,family=binomial)
> summary(logitout)
...
Coefficients:
             Estimate Std. Error z value
(Intercept) -8.4046964  0.7166359 -11.728
NPreg        0.1231823  0.0320776   3.840
Gluc         0.0351637  0.0037087   9.481
BP          -0.0132955  0.0052336  -2.540
Thick        0.0006190  0.0068994   0.090
Insul       -0.0011917  0.0009012  -1.322
BMI          0.0897010  0.0150876   5.945
Genet        0.9451797  0.2991475   3.160
Age          0.0148690  0.0093348   1.593
            Pr(>|z|)
(Intercept)  < 2e-16 ***
NPreg       0.000123 ***
Gluc         < 2e-16 ***
BP          0.011072 *
```

[6] The data set is at *https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes*. I have added a header record to the file.

```
Thick        0.928515
Insul        0.186065
BMI          2.76e-09 ***
Genet        0.001580 **
Age          0.111192
...
```

4.3.3 Interpretation of Coefficients

In nonlinear regression models, the parameters β_i do not have the simple marginal interpretation they enjoy in the linear case. Statements like we made in Section 1.9.1.2, "We estimate that, on average, each extra year of age corresponds to almost a pound in extra weight," are not possible here.

However, in the nonlinear case, the regression function is still defined as the conditional mean, which in the logit case reduces to the conditional probability of a 1. Practical interpretation is definitely still possible, if slightly less convenient.

Consider for example the estimated Glucose coefficient in our diabetes data above, 0.035. Let's apply that to the people similar to the first person in the data set:

```
> pima[1,]
  NPreg Gluc BP Thick Insul  BMI Genet Age Diab
1     6  148 72    35     0 33.6 0.627  50    1
```

Ignore the fact that this woman has diabetes. Let's consider the subpopulation of all women with the same characteristics as this one, i.e., all who have had 6 pregnancies, a glucose level of 148 and so on, through an age of 50. The estimated proportion of women with diabetes in this subpopulation is

$$\frac{1}{1+e^{-(8.4047+0.1232 \cdot 6+...+0.0149 \cdot 50)}} \tag{4.33}$$

We don't have to plug these numbers in by hand, of course:

```
> l <- function(t) 1/(1+exp(-t))
> pima1 <- unlist(pima[1,-9])
> l(coef(logitout) %*% c(1,pima1))
          [,1]
[1,] 0.7217266
```

Note that **pima[1,-9]** is actually a data frame (having been derived from a data frame), so in order to multiply it, we needed to make a vector out of it, using **unlist()**.

So, we estimate that about 72% of women in this subpopulation have diabetes. But what about the subpopulation of the same characteristics, but of age 40 instead of 50?

```
> w <- pima1
> w['Age'] <- 40
> l(coef(logitout) %*% c(1,w))
           [,1]
[1,] 0.6909047
```

Only about 69% of the younger women have diabetes.

So, there is an effect of age on developing diabetes, but only a mild one; a 10-year increase in age only increased the chance of diabetes by about 3.1%. However, note carefully that this was for women having a given set of the other factors, e.g., 6 pregnancies. Let's look at a different subpopulation, those with 2 pregnancies and a glucose level of 120, comparing 40- and 50-year-olds:

```
> u <- pima1
> u[1] <- 2
> u[2] <- 100
> v <- u
> v[8] <- 40
> l(coef(logitout) %*% c(1,u))
           [,1]
[1,] 0.2266113
> l(coef(logitout) %*% c(1,v))
           [,1]
[1,] 0.2016143
```

So here, the 10-year age effect was somewhat less, about 2.5%. A more careful analysis would involve calculating standard errors for these numbers, but the chief point here is that the effect of a factor in nonlinear situations depends on the values of the other factors.

$$P(Y = 0 \mid X^{(1)} = t_1, ..., X^{(p))} = t_p) = 1 - \ell(\beta_0 + \beta_1 t_1 + ... + \beta_p t_p) \quad (4.34)$$

Some analysts like to look at the *log-odds ratio*,

$$\log \frac{P(Y = 1 \mid X^{(1)} = t_1, ..., X^{(p))} = t_p)}{P(Y = 0 \mid X^{(1)} = t_1, ..., X^{(p))} = t_p)} \tag{4.35}$$

in this case the logaritihm of the ratio of the probability of having and not having the disease. By Equation (4.8), this simplifies to

$$\beta_0 + \beta_1 t_1 + ... + \beta_p t_p \tag{4.36}$$

a linear function. Thus, in interpreting the coefficients output from a logistic analysis, it is convenient to look at this *log-odds ratio*, as it gives us a single marginal-effect number for each factor. This may be sufficient for the application at hand, but a more thorough analysis should consider the effects of the factors on the probabilities themselves.

4.3.4 The predict() Function Again

In the previous section, we evaluated the estimated regression function (and thus predicted values as well) the straightforward but messy way, e.g.,

```
> l(coef(logitout) %*% c(1,v))
```

The easy way is to use R's **predict()** function:

```
> predict(object=logitout,newdata=pima[1,-9],
    type='response')
         1
0.7217266
```

We saw that in Section 1.10.3 for objects of **'lm'** class. But in our case here, we invoked it on **logitout**. What is the class of that object?

```
> class(logitout)
[1] "glm" "lm"
```

So, it is an object of class **'glm'**, and, we see, the latter is a subclass of the class **'lm'**. For that subclass, the **predict()** function, i.e., **predict.glm()**, there is an extra argument (actually several), **type**. The value of that argument that we want here is **type = 'response'**, alluding to the fact that we want a prediction on the scale of the response variable, Y.

4.3.5 Overall Prediction Accuracy

How well can we predict in the Pima example above? For the best measure, we should use cross validation or something similar, but we can obtain a quick measure as follows.

The value returned by **glm()** has class **'glm'**, which is actually a subclass of **'lm'**. The latter, and thus the former, includes a component **fitted.values**, the i^{th} of which is

$$\widehat{\mu}(X_i) \tag{4.37}$$

i.e., the estimated value of the regression function at observation i. If we did not know Y_i, we would predict it to be 1 or 0, depending on whether $\widehat{\mu}(X_i)$ is greater than or less than 0.5. In R terms, that predicted value is simply

```
round(logitout$fitted.values[i])
```

Using the fact that the proportion of 1s in a vector of 1s and 0s is simply the mean value in that vector, we have that the overall probability of correct classification is

```
> mean(pima$Diab == round(logitout$fitted.values))
0.7825521
```

That seems pretty good (though again, it is biased upward and cross validation would give us a better estimate), but we must compare it against how well we would do without the covariates. We reason as follows. First,

```
> mean(pima$Diab)
[1] 0.3489583
```

Most of the women do not have diabetes, so our strategy, lacking covariate information, would be to always guess that $Y = 0$. We will be correct a proportion

```
> 1 - 0.3489583
[1] 0.6510417
```

of the time. Thus our 78% accuracy using covariates does seem to be an improvement.

4.3.6 Example: Predicting Spam E-mail

One application of these methods is *text classification*. In our example here, the goal is machine prediction of whether an incoming e-mail message is *span*, i.e., unwanted mail, typically ads.

We'll use the **spam** dataset from the UCI Machine Learning Data Repository. It is also available from the CRAN package **ElemStatLearn** [66], which we will use here, but note that the UCI version includes a word list. It has data on 4601 e-mail messages, with 57 predictors. The first 48 of those predictors consist of frequencies of 48 words. Thus the first column, for instance, consists of the proportions of Word 1 in each of the 4601 messages, with the total number of words in a message as the base in each case. The remaining predictors involve measures such as the numbers of consecutive capital letters in words.

The last column is an R *factor* with levels *spam* and *e-mail*. This R type is explained in Section 4.7.2, and though **glm()** can handle such variables, for pedagogical reasons, let's use dummies for a while. (We will begin using factors directly in Section 5.6.3.)

Let's fit a logistic model.

```
> library(ElemStatLearn)
> data(spam)
> spam$spam <- as.integer(spam$spam == 'spam')
> glmout <- glm(spam ~ .,data=spam,
      family=binomial)
> summary(glmout)
...

Coefficients:
             Estimate Std. Error z value
(Intercept) -1.569e+00  1.420e-01 -11.044
A.1         -3.895e-01  2.315e-01  -1.683
A.2         -1.458e-01  6.928e-02  -2.104
A.3          1.141e-01  1.103e-01   1.035
A.4          2.252e+00  1.507e+00   1.494
A.5          5.624e-01  1.018e-01   5.524
A.6          8.830e-01  2.498e-01   3.534
A.7          2.279e+00  3.328e-01   6.846
A.8          5.696e-01  1.682e-01   3.387
...
            Pr(>|z|)
```

```
(Intercept)  < 2e-16 ***
A.1          0.092388 .
A.2          0.035362 *
A.3          0.300759
A.4          0.135168
A.5          3.31e-08 ***
A.6          0.000409 ***
A.7          7.57e-12 ***
A.8          0.000707 ***
...
```

Let's see how accurately we can predict with this model:

```
> mean(spam$spam == round(glmout$fitted.values))
[1] 0.9313193
```

Not bad at all. But much as we are annoyed by spam, we hope that a genuine message would not be likely to be culled out by our spam filter. Let's check:

```
> spamnot <- which(spam$spam == 0)
> mean(round(glmout$fitted.values[spamnot]) == 0)
[1] 0.956241
```

So if a message is real, it will have a 95% chance of getting past the spam filter.

4.3.7 Linear Boundary

In (4.18), which values of $t = (t_1, ..., t_p)'$ will cause us to gues $Y = 1$ and which will result in a guess of $Y = 0$? The boundary occurs when (4.18) has the value 0.5. In other words, the boundary consists of all t such that

$$\beta_0 + \beta_1 t_1 + ... + \beta_p t_p = 0 \tag{4.38}$$

So, the boundary has linear form, a *hyperplane* in p-dimensional space. This may seem somewhat abstract now, but it will have value later on.

4.4 GLM: the Poisson Regression Model

Since in the Pima data (Section 4.3.2) the number of pregnancies is a count, we might consider predicting it using Poisson regression.[7] Here's how we can do this with **glm()**:

```
> poisout <- glm(NPreg ~ .,data=pima,family=poisson)
> summary(poisout)
...
Coefficients:
             Estimate Std. Error z value
(Intercept)  0.2963661  0.1207149   2.455
Gluc        -0.0015080  0.0006704  -2.249
BP           0.0011986  0.0010512   1.140
Thick        0.0000732  0.0013281   0.055
Insul       -0.0003745  0.0001894  -1.977
BMI         -0.0002781  0.0027335  -0.102
Genet       -0.1664164  0.0606364  -2.744
Age          0.0319994  0.0014650  21.843
Diab         0.2931233  0.0429765   6.821
            Pr(>|z|)
(Intercept)  0.01408 *
Gluc         0.02450 *
BP           0.25419
Thick        0.95604
Insul        0.04801 *
BMI          0.91896
Genet        0.00606 **
Age          < 2e-16 ***
Diab        9.07e-12 ***
...
```

On the other hand, even if we believe that our count data follow a Poisson distribution, there is no law dictating that we use Poisson regression, i.e., the model (4.10). As mentioned following that equation, the main motivation for using exp() in that model is to ensure that our regression function is nonnegative, conforming to the nonnegative nature of Poisson random variables. This is not unreasonable, but as noted in a somewhat different context in Section 3.3.7, transformations — in this case, the use of exp() — can produce distortions. Let's try the "unorthodox" model, (4.11):

[7] It may seem unnatural to predict this, but as noted before, predicting any variable may be useful if data on that variable may be missing.

```
> quasiout <- glm(NPreg ~ .,data=pima,
      family=quasi(variance="mu^2"),start=rep(1,9))
```

This "quasi" family is a catch-all option, specifying a linear model but here allowing us to specify a Poisson variance function:

$$Var(Y \mid X = t) = [\mu(t)]^2 \tag{4.39}$$

with $\mu(t) = t'\beta$. This is (4.11), not the standard Poisson regression model, but worth trying anyway.

Well, then, which model performed better? As a rough, quick look, ignoring issues of overfitting and the like, let's consider R^2. This quantity is not calculated by **glm()**, but recall from Section 2.9.2 that R^2 is the squared correlation between the predicted and actual Y values. This quantity makes sense for any regression situation, so let's calculate it here:

```
> cor(poisout$fitted.values,poisout$y)^2
[1] 0.2314203
> cor(quasiout$fitted.values,quasiout$y)^2
[1] 0.3008466
```

The "unorthodox" model performed better than the "official" one! We cannot generalize from this, but it does show again that one must use transformations carefully.

4.5 Least-Squares Computation

A point made in Section 1.4 was that the regression function, i.e., the conditional mean, is the optimal predictor function, minimizing mean squared prediction error. This still holds in the nonlinear (and even nonparametric) case. The problem is that in the nonlinear setting, the least-squares estimator does not have a nice, closed-form solution like (2.28) for the linear case. Let's see how we can compute the solution through iterative approximation.

4.5.1 The Gauss-Newton Method

Denote the nonlinear model by

$$E(Y \mid X = t) = g(t, \beta) \tag{4.40}$$