# Logistic Regression

```
require(mosaic)
require(mosaicData)
require(lmtest)
cols <-  trellis.par.get()$superpose.symbol$col
```

Often, we want to model a **response** variable that is **binary**, meaning that it can take on only two possible outcomes. These outcomes could be labeled "Yes" or "No", or "True" of "False", but for all intents and purposes, they can be coded as either 0 or 1. We have seen these types of variables before (as indicator variables), but we always used them as **explanatory** variables. Creating a model for such a variable as the response requires a more sophisticated technique than ordinary least squares regression. It requires the use of a **logistic** model.

The data in the `Whickham` data set (built into `mosaicData`) contains observations about women, and whether they were alive 20 years after their initial observation. Our goal is to determine how being a `smoker` affects the probability of being alive, after controlling for `age`.

```
data(Whickham)
head(Whickham)
```

First, let's plot the data. Already we meet challenges.

```
xyplot(outcome ~ age, groups=smoker, data=Whickham, alpha=0.5, pch=19, cex=2, auto.key = TRUE)
```

1. What do the pink and blue dots represent?

It is very difficult to tell what, if anything, is happening here. Let's add a new variable that is the numeric equivalent of the `outcome` variable.

```
Whickham <- Whickham %>%
  mutate(isAlive = 2 - as.numeric(outcome))
```

2. What is this piece of code doing? Why 2?

Once we have a 0/1 vector for our response, we can try plotting again. This plot makes it a little easier to see by jittering the points in the $y$-direction (something we couldn't have done without numeric data).

```
myplot <- xyplot(jitter(isAlive) ~ age, groups=smoker, data=Whickham, alpha=0.5, pch=19, cex=2, ylab="is
print(myplot)
```

The simplest approach to modeing this would just be to predict the average for every observation.

```
avg <- mean(~isAlive, data=Whickham)
avg #what does this number mean?
ladd(panel.abline(h=avg, col=cols[1], lwd=3))
```

1. What would we predict for a 20-year-old person who smokes? For a 60-year-old person who doesn't smoke? Does this make sense?

Certainly we could improve on this with a linear model, but is it appropriate here? Let's try one.

```
m1 <- lm(isAlive ~ age + smoker, data=Whickham)
summary(m1)
```

4. Interpret the coefficients of this model. Do they makes sense? What sort of predictions does this model make?

```
fit.alive <- makeFun(m1)
plotFun(fit.alive(age=x, smoker="Yes") ~ x, plot=myplot, lwd=3, add=TRUE)
plotFun(fit.alive(age=x, smoker="No") ~ x, col=cols[2], lwd=3, add=TRUE)
```

5. How can we interpret this plot? What would we predict for a 20-year-old who smokes? For a 60-year-old who doesn't smoke? How about for a 15-year-old who doesn't smoke? An 85-year-old who smokes?

### Fitting a Logistic model

So, we do not think that a linear model is the way to go. The predicted values aren't interpetable (what does 0.65 `isAlive` mean?) and they go outside the realm of possibility (what does 1.24 `isAlive` mean?).

So, instead of modeling $\pi$ (the response variable) like this,

$$\pi = \beta_0 + \beta_1 X$$

suppose we modeled it like this,

$$\log\left(\frac{\pi}{1-\pi}\right) = logit(\pi) = \beta_0 + \beta_1 X$$

This transformation is called the **logit** function. What are the properties of this function? Note that this implies that

$$\pi = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \in (0,1)$$

The logit function constrains the fitted values to line within $(0,1)$, which helps to give a natural interpretation as the probability of the response actually being 1.

Fitting a logistic curve is mathematically more complicated than fitting a least squares regression, but the syntax in R is similar, as is the output. The procedure for fitting is called *maximum likelihood estimation*, and the usual machinery for the sum of squares breaks down. Consequently, there is no notion of $R^2$, etc.

```
logm <- glm(isAlive ~ age + smoker, data=Whickham, family=binomial(logit))
# Note that you can also just say "family=binomial" since logit is the default option
summary(logm)
```

1. How can we interpret the coefficients of this model in the context of the problem?

The procedure for adding the logistic curve to the plot is the same as it was before.

```
fit.outcome <- makeFun(logm)
plotFun(fit.outcome(age=x, smoker="Yes") ~ x, lwd=3, plot=myplot, add=TRUE)
plotFun(fit.outcome(age=x, smoker="No") ~ x, col=cols[2], lwd=3, add=TRUE)
```

2. How does the `smoker` term affect the model?

### Plotting the model in space

We can think of logistic regression models in 3 different "spaces." For this example, lets just think about one predictor.

```
logm2 <- glm(isAlive ~ age , data=Whickham, family=binomial(logit))
```

The first space is the linear model space, where we are considering log-odds.

```
xyplot(log(fitted.values(logm2)/(1-fitted.values(logm2)))~age,
       data=Whickham, type=c("l"),ylab="log odds")
```

One is the odds space,

```
xyplot(fitted.values(logm2)/(1-fitted.values(logm2))~age, data=Whickham,
       type="spline", ylab="odds")
```

Another is the probability space,

```
plotModel(logm2, ylab="probability")
```

### Binning

Another way to make sense of a binary response variable is to **bin** the explanatory variable and then compute the average proportion of the response within each bin.

```
Whickham <- Whickham %>%
  mutate(ageGroup = cut(age, breaks=10))
favstats(~isAlive | ageGroup, data=Whickham)
```

Although this is not the preferred method for performing logistic regression, it can be illustrative to see how the logistic curve fits through this series of points.

```
# print(myplot)
binned.y <- mean(~isAlive | ageGroup, data=Whickham)
binned.x <- mean(~age | ageGroup, data=Whickham)
binplot <- xyplot(binned.y ~ binned.x, cex=2, pch=19, col="orange", type=c("p", "r"), lwd=3)
plotFun(fit.outcome(age=x, smoker="Yes") ~ x, lwd=3, add=TRUE, plot=binplot)
plotFun(fit.outcome(age=x, smoker="No") ~ x, col=cols[2], lwd=3, add=TRUE)
```

### The Link Values

Consider now the difference between the fitted values and the link values. Although the fitted values do not follow a linear pattern with respect to the explanatory variable, the link values do. To see this, let's plot them explicitly against the logit of the binned values.

```
xyplot(logit(binned.y) ~ binned.x, pch=19, cex=2, col="orange", type=c("p", "l"))
Whickham <- Whickham %>%
  mutate(logm.link = predict(logm, type="link"))
ladd(with(subset(Whickham, smoker=="Yes"), panel.xyplot(age, logm.link, col=cols[1])))
ladd(with(subset(Whickham, smoker=="No"), panel.xyplot(age, logm.link, col=cols[2])))
```

Note how it is considerably easier for us to assess the quality of the fit visually using the link values, as opposed to the binned probabilities.

5. Why can't we take the logit of the actual responses?
6. Fit a logistic model for *Acceptance* as a function of `GPA`, with `Gender` as a covariate using the **MedGPA** data set.

```
require(Stat2Data)
data(MedGPA)
```

### Odds Ratios and Interpretation of Coefficients

The interpretation of the coefficients in a linear regression model are clear based on an understanding of the geometry of the regression model. We use the terms *intercept* and *slope* because a simple linear regression model is a line. In a simple logistic model, the line is transformed by the logit function. How do the coefficients affect the shape of the curve in a logistic model?

The following **manipulate** function will allow you to experiment with changes to the intercept and slope coefficients in the simple logistic model for `isAlive` as a function of `age`.

```
log.whickham <- function (intercept.offset = 0, slope.multiple=1,...) {
#  data(Whickham)
  Whickham <- Whickham %>%
    mutate(isAlive = 2 - as.numeric(outcome))
  logm <- glm(isAlive ~ age, data=Whickham, family=binomial(logit))
  fit.outcome <- makeFun(logm)
  xyplot(jitter(isAlive) ~ age, groups=smoker, data=Whickham,
         ylab="isAlive",
         panel = function (x,y,...) {
           panel.xyplot(x,y, alpha=0.3, pch=19, cex=2,...)
           panel.curve(fit.outcome(x), col="darkgray", lty=3, lwd=3)
           panel.curve(fit.outcome(x * slope.multiple + intercept.offset), lwd=3)
         }
  )
}
require(manipulate)
manipulate(log.whickham(intercept.offset, slope.multiple),
           intercept.offset = slider(-20, 20, step=1, initial=0),
           slope.multiple = slider(0, 5, step=0.25, initial=1)
)
```

8. How do changes in the intercept term affect the shape of the logistic curve?
9. How do changes in the slope term affect the shape of the logistic curve?

We saw earlier that the link values are linear with respect to the explanatory variable. The link values are the log of the **odds**. Note that if an event occurs with proability $\pi$, then

$$odds = \frac{\pi}{1-\pi}, \qquad \pi = \frac{odds}{1+odds}.$$

Note that while $\pi \in [0,1]$, $odds \in (0,\infty)$. Thus, we can interpret $\hat{\beta}_1$ as the typical change in $\log(odds)$ for each one unit increase in the explanatory variable. More naturally, the odds of success are multiplied by $e^{\hat{\beta}_1}$

for each one unit increase in the explanatory variable, since this is the **odds ratio**.

$$odds_X = \frac{\hat{\pi}_X}{1-\hat{\pi}_X} = e^{\hat{\beta}_0+\hat{\beta}_1 X}$$
$$odds_{X+1} = \frac{\hat{\pi}_{X+1}}{1-\hat{\pi}_{X+1}} = e^{\hat{\beta}_0+\hat{\beta}_1 (X+1)}$$
$$\frac{odds_{X+1}}{odds_X} = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 (X+1)}}{e^{\hat{\beta}_0+\hat{\beta}_1 X}} = e^{\hat{\beta}_1}$$

Furthermore, since the *logits* are linear with respect to the explanatory variable, and these changes are constant.

Finding confidence intervals for the odds ratio is easy.

```
exp(confint(logm))
```

9. Find confidence intervals for the odds ratios of the terms in the **MedGPA** model you built earlier.

### Assessing a Logistic Fit

Three of the conditions we require for linear regression have analogs for logistic models:

- Linearity of the *logit* (or $\log(odds)$)
- Independence
- Random

However, the requirements for Constant Variance and Normality are no longer applicable. In the first case, the variability in the response now inherently depends on the value, so we know we won't have constant variance. In the second case, there is no reason to think that the residuals will be normally distributed, since the "residuals" are can only be computed in relation to 0 or 1. So in both cases the properties of a binary response variable break down the assumptions we made previously.

Moreover, since we don't have any sums of squares, we can't use $R^2$, ANOVA, or $F$-tests. Instead, since we fit the model using *maximum likelihood estimation*, we compute the likelihood of our model.

$$L(y_i) = \begin{cases} \hat{\pi} & \text{if } y_i = 1 \\ 1-\hat{\pi} & \text{if } y_i = 0 \end{cases}, \qquad L(model) = \prod_{i=1}^{n} L(y_i)$$

Because these numbers are usually very small (why?), it is more convenient to speak of the log-likelihood $\log(L)$, which is always negative. A larger $\log(L)$ is closer to zero and therefore a better fit.

The log-likelihood is easy to retrieve

```
logLik(logm)
```

but is nearly as easy to calculate directly.

```
pi<- fitted.values(logm)
likelihood <- ifelse(Whickham$isAlive == 1, pi, 1 - pi)
log(prod(likelihood))
```

The closest thing to an analog of the $F$-test is the **Likelihood Ratio Test** (LRT). Here, our goal is to compare the log-likelihoods of two models: the one we build vs. the constant model. This is similar to the

way we compared the sum of the squares explained by a linear regression model to the model that consists solely of the grand mean.

The null hypothesis in the LRT is that $\beta_1 = \beta_2 = \cdots = \beta_k = 0$. The alternative hypothesis is that at least one of these coefficients is non-zero. The test statistic is:

$$G = -2\log(constantmodel) - (-2\log(model)).$$

These two quantities are known as **deviances**. It can be shown that $G$ follows a $\chi^2$ distribution with $k$ degrees of freedom. This allows us to compute a $p$-value for the model.

```
lrtest(logm)
```

In this sense the LRT has obvious similarities to the ANOVA table and $F$-test. In the same way that we previously performed a nested $F$-test to assess the usefulness of a group of predictors, we can perform a nested LRT.

Adding interaction or quadratic terms works in much the same way as it did with linear regression.

```
linteract <- glm(isAlive ~ age + smoker + age*smoker, data=Whickham, family=binomial)
summary(linteract)
```

10. Interpret the coefficients of this model, and find confidence intervals for them.

Suppose now that we suspect that there are diminishing returns to the extent to which being alive is associated with a person's age. We can easily add quadratic terms.

```
lquad <- glm(isAlive ~ age + smoker + age*smoker + I(age^2) + I(age^2):smoker, data=Whickham, family=bi
summary(lquad)
```

11. Interpret the coefficients of this model, and find confidence intervals for them.

How can we assess whether these terms are warranted? Just like the nested $F$-test, the nested LRT gives us information about the incremental contribution of a set of terms to our model.

```
lrtest(logm, linteract, lquad)
```

13. Experiment with different models until you find one that you think fits the data best. Justify your answer.

```
print(myplot)
fit.qalive <- makeFun(lquad)
plotFun(fit.outcome(age=x, smoker="Yes") ~ x, add=TRUE, lty=2)
plotFun(fit.outcome(age=x, smoker="No") ~ x, col=cols[2], add=TRUE, lty=2)
plotFun(fit.qalive(age=x, smoker="Yes") ~ x, add=TRUE)
plotFun(fit.qalive(age=x, smoker="No") ~ x, col=cols[2], add=TRUE)
```