Advanced Quantitative Methods
Bootstrap and simulation

Johan A. Elkink
jos.elkink@ucd.ie

April 13, 2010

If you encounter any mistakes in this text, please inform the author.

1 Bootstrapping confidence intervals

This note is based on the exercise on Slide 39 of the lecture, where the task is to estimate a logistic regression, calculate and plot predicted values, and add uncertainty estimates. We will go step-by-step through answering this exercise.

The data used here is `lisbon08_sample.dta`, which can be opened using the `read.dta()` function:

```
> library(foreign)
> lisbon08 <- read.dta("lisbon08_sample.dta")
```

The model to be estimated is explaining whether a respondent voted or not in the 2008 Lisbon referendum by looking at the respondent's attitude towards immigration (whether or not immigration made Ireland a better place to live), towards neutrality (whether or not Ireland should do everything it can to protect its neutrality) and knowledge about the Treaty. We estimate this model as follows:

```
> vote.model <- glm(VOTED ~ NEUTRALITY + IMMIGRATION + OBJ_KNOW_TR,
+     data = lisbon08, family = binomial(link = "logit"))
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.5894	0.4771	3.33	0.0009
NEUTRALITY	0.0477	0.0583	0.82	0.4130
IMMIGRATION	0.0614	0.0533	1.15	0.2494
OBJ_KNOW_TR	-0.3003	0.0594	-5.06	0.0000

Since this is a logistic regression, predicted values have to be calculated using the logit transformation:

$$\hat{\pi} = \frac{1}{1 + e^{-\mathbf{X}\hat{\boldsymbol{\beta}}}}$$

This is most easily done by writing a little function which takes care of this, so we do not have to type this every time:

```
> invlogit <- function(xb) {
+     1/(1 + exp(-xb))
+ }
```

So now we can calculate a predicted value for any case, for example setting the values for NEUTRALITY and IMMIGRATION at their mean, and that for knowledge at the extremes, we get:

```
> neut.mean <- mean(lisbon08$NEUTRALITY, na.rm = TRUE)
> imm.mean <- mean(lisbon08$IMMIGRATION, na.rm = TRUE)
> x.low <- c(1, neut.mean, imm.mean, 9)
> x.high <- c(1, neut.mean, imm.mean, 0)
> pi.low <- invlogit(x.low %*% coef(vote.model))
> pi.high <- invlogit(x.high %*% coef(vote.model))
> c(pi.low, pi.high)
```

```
[1] 0.3752757 0.8996056
```

Note that on the knowledge scale in this data, a 9 means lack of knowledge about the Lisbon Treaty, while a 0 means high knowledge. x_{low} is thus defined by the constant (1), the mean of the NEUTRALITY scode, the mean of the IMMIGRATION score and the 9 indicating minimal knowledge of the treaty. Given the logistic regression estimates, we predict that, given average scores on the attitude scales, a voter with little knowledge of the treaty has a 38% chance of turning out to vote, while a similar voter with a high level of knowledge has a 90% chance.

Predictions are one thing, but now we would want to have a 95% confidence interval (CI) around these estimates. There are several ways of going about this. Using simulated parameters,[1] we could do the following:

```
> library(MASS)
> niter <- 1000
> b.star <- mvrnorm(niter, coef(vote.model), vcov(vote.model))
> pi.star.low <- invlogit(t(x.low %*% t(b.star)))
> pi.star.high <- invlogit(t(x.high %*% t(b.star)))
```

We have thus drawn niter (i.e. "number of iterations") random draws from the multivariate normal distribution with the mean defined by the estimated β-coefficients, and the variance defined by the variance-covariance matrix of this estimation. We use these random draws to create niter predicted probabilities, which we can subsequently use to see the uncertainty of these estimates. We thus translate the uncertainty of the β-coefficients into uncertainty of the predicted probabilities. There are two ways of calculating confidence intervals from these. The first just uses the standard deviation of these predicted probabilities:

```
> c(pi.low - 1.96 * sd(pi.star.low), pi.low + 1.96 * sd(pi.star.low))
```

```
[1] 0.2585458 0.4920055
```

```
> c(pi.high - 1.96 * sd(pi.star.high), pi.high + 1.96 * sd(pi.star.high))
```

```
[1] 0.8393197 0.9598916
```

[1]This is not a formal term, but one that will do for the moment.

The second approach is more nonparametric and looks at the actual lower 2.5% quantile and the upper 2.5% quantile to get estimates of the 95% confidence interval:

```
> quantile(pi.star.low, c(0.025, 0.975))

     2.5%     97.5%
0.2620857 0.5013203

> quantile(pi.star.high, c(0.025, 0.975))

     2.5%     97.5%
0.8266377 0.9451864
```

As an alternative to simulated parameters, we can use a bootstrap procedure. A bootstrap procedure consists of taking random samples from the actual data, of the same size, drawn with replacement, and then each iteration the model is re-estimated and the predictions calculated. The code might look as follows:

```
> n <- dim(vote.model$model)[1]
> niter <- 1000
> pi.star.low <- NULL
> pi.star.high <- NULL
> for (i in 1:niter) {
+     s <- sample(1:n, n, replace = TRUE)
+     vote.model.s <- glm(VOTED ~ NEUTRALITY + IMMIGRATION + OBJ_KNOW_TR,
+         data = lisbon08[s, ], family = binomial(link = "logit"))
+     pi.star.low <- c(pi.star.low, invlogit(x.low %*% coef(vote.model.s)))
+     pi.star.high <- c(pi.star.high, invlogit(x.high %*% coef(vote.model.s)))
+ }
```

s is here an index variable, indexing the rows in the data we randomly sampled. The `sample()` function here samples from a vector $\begin{bmatrix}1 & 2 & 3 & \ldots & n\end{bmatrix}$, which is basically an index to all rows, taking a sample of size n, with replacement. This s is then later used to select the cases in the `data` option to `glm()`.

We can now use the same two techniques to get confidence intervals:

```
> c(pi.low - 1.96 * sd(pi.star.low), pi.low + 1.96 * sd(pi.star.low))

[1] 0.2568293 0.4937220

> c(pi.high - 1.96 * sd(pi.star.high), pi.high + 1.96 * sd(pi.star.high))

[1] 0.8405257 0.9586856

> quantile(pi.star.low, c(0.025, 0.975))

     2.5%     97.5%
0.2636162 0.4987624

> quantile(pi.star.high, c(0.025, 0.975))

     2.5%     97.5%
0.8334982 0.9523820
```

Note that in the case of a limited dependent variable model, the latter might make much more sense, as it avoids providing confidence intervals that cross the $[0, 1]$ boundaries.

So all together we now have the following estimates for the confidence intervals:[2]

Method	Lower bound	Upper bound
Lack of knowledge		
Simulated parameters, $\hat{\sigma}$	0.256	0.495
Simulated parameters, quantiles	0.267	0.506
Bootstrap, $\hat{\sigma}$	0.257	0.493
Bootstrap, quantiles	0.256	0.492
High knowledge		
Simulated parameters, $\hat{\sigma}$	0.839	0.960
Simulated parameters, quantiles	0.827	0.945
Bootstrap, $\hat{\sigma}$	0.841	0.959
Bootstrap, quantiles	0.835	0.949

The question now is which one to trust, of course. Basically, from top to bottom, we make fewer assumptions about the normality of the predictions and of the estimated coefficients, thus the bootstrapped quantiles might be more reliable. The proof of this is based on asymptotics, however - in smaller samples it is less clear what to trust. The bootstrap has the main disadvantage that the model needs to be re-estimated every iteration - for many complicated models, this is not feasible, in which case simulated parameters are more useful.

The next task is to plot predicted probabilities by knowledge, which can be done as in Figure 1.

Like with the predicted probabilities for particular scenarios, it would be better if we could add some measure of uncertainty. One way would be to simply bootstrap many plots and overlay them, as demonstrated in Figure 1.

Perhaps more interesting would be to calculate 95% confidence intervals for particular points, for example the 10 points on the scale that are actually possible for objective knowledge of the Treaty. For an example of this, see Figure 1, which also uses the bootstrapping technique. The same can be produced with simulated parameters, as in Figure 1.

[2]The numbers copied into the table are not from the exact same code execution as shown above, so might slightly differ.

```
> plot(jitter(lisbon08$VOTED) ~ jitter(lisbon08$OBJ_KNOW_TR), pch = 19,
+     cex = 0.5, xlim = c(0, 9), ylim = c(0, 1), xlab = "Lack of objective knowledge of the tr
+     ylab = "Probability to vote")
> curve(invlogit(cbind(1, neut.mean, imm.mean, x) %*% coef(vote.model)),
+     from = 0, to = 9, lwd = 2, col = "red", add = TRUE)
```

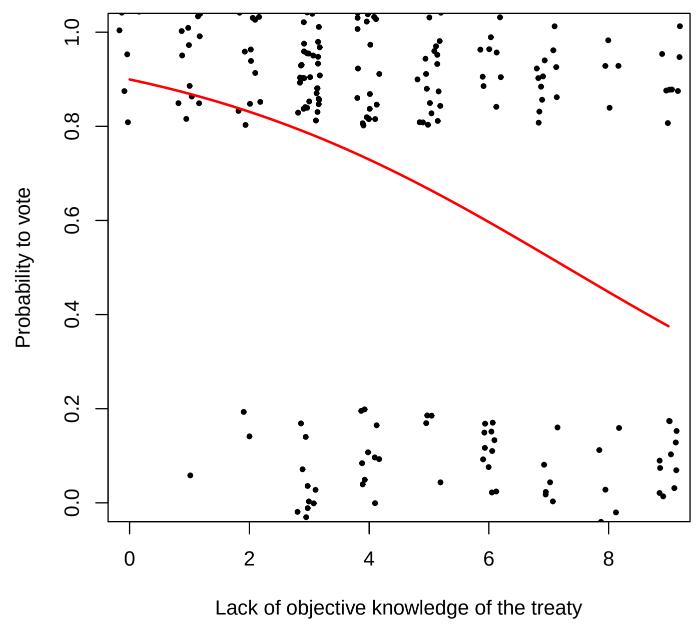

Figure 1: Plain prediction, without uncertainty

```
> plot(jitter(lisbon08$VOTED) ~ jitter(lisbon08$OBJ_KNOW_TR), pch = 19,
+     cex = 0.5, xlim = c(0, 9), ylim = c(0, 1), xlab = "Lack of objective knowledge of the tr
+     ylab = "Probability to vote")
> n <- dim(vote.model$model)[1]
> niter <- 100
> for (i in 1:niter) {
+     s <- sample(1:n, n, replace = TRUE)
+     vote.model.s <- glm(VOTED ~ NEUTRALITY + IMMIGRATION + OBJ_KNOW_TR,
+         data = lisbon08[s, ], family = binomial(link = "logit"))
+     curve(invlogit(cbind(1, neut.mean, imm.mean, x) %*% coef(vote.model.s)),
+         from = 0, to = 9, lwd = 1, col = "lightgray", add = TRUE)
+ }
> curve(invlogit(cbind(1, neut.mean, imm.mean, x) %*% coef(vote.model)),
+     from = 0, to = 9, lwd = 2, col = "red", add = TRUE)
```

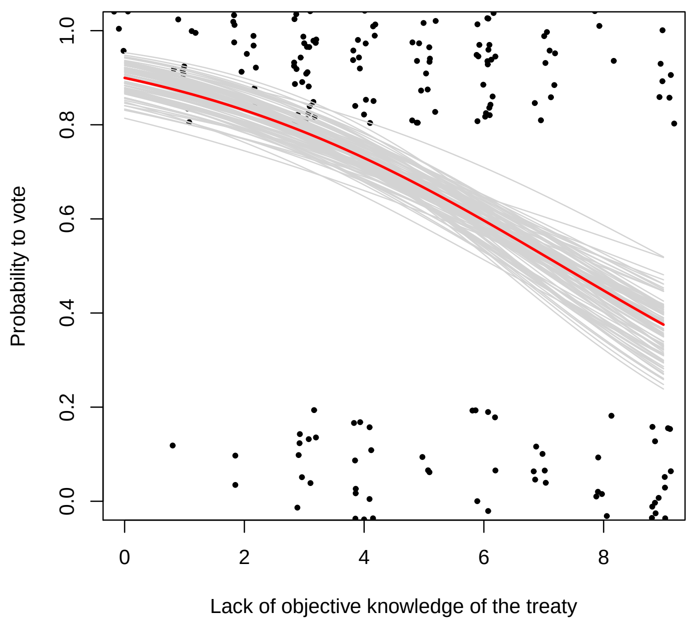

Figure 2: Prediction, with bootstrapped versions

```
> plot(jitter(lisbon08$VOTED) ~ jitter(lisbon08$OBJ_KNOW_TR), pch = 19,
+     cex = 0.5, xlim = c(0, 9), ylim = c(0, 1), xlab = "Lack of objective knowledge of the tr
+     ylab = "Probability to vote")
> n <- dim(vote.model$model)[1]
> niter <- 1000
> pi.star <- NULL
> for (i in 1:niter) {
+     s <- sample(1:n, n, replace = TRUE)
+     vote.model.s <- glm(VOTED ~ NEUTRALITY + IMMIGRATION + OBJ_KNOW_TR,
+         data = lisbon08[s, ], family = binomial(link = "logit"))
+     pi.star <- rbind(pi.star, t(invlogit(cbind(1, neut.mean,
+         imm.mean, c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)) %*% coef(vote.model.s))))
+ }
> for (i in 0:9) {
+     lines(c(i, i), quantile(pi.star[, i + 1], c(0.025, 0.975)),
+         col = "lightgray")
+ }
> curve(invlogit(cbind(1, neut.mean, imm.mean, x) %*% coef(vote.model)),
+     from = 0, to = 9, lwd = 2, col = "red", add = TRUE)
```

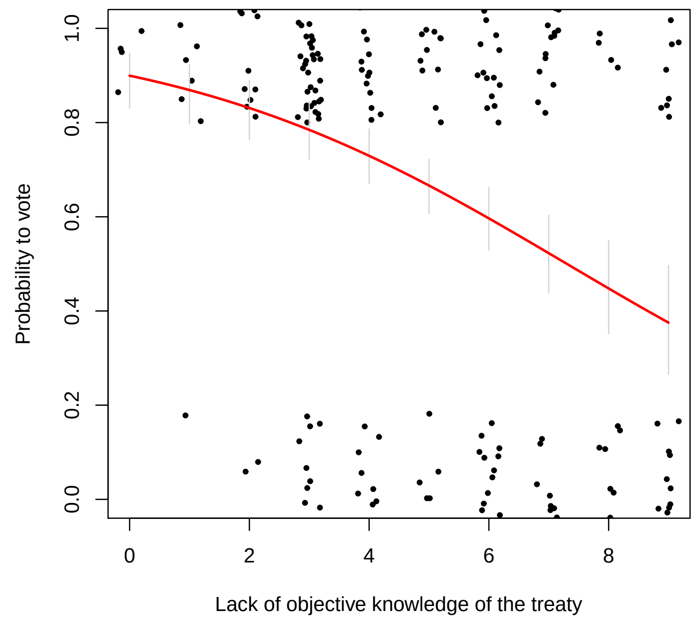

Figure 3: Prediction, with bootstrapped 95% confidence intervals

```
> plot(jitter(lisbon08$VOTED) ~ jitter(lisbon08$OBJ_KNOW_TR), pch = 19,
+     cex = 0.5, xlim = c(0, 9), ylim = c(0, 1), xlab = "Lack of objective knowledge of the t
+     ylab = "Probability to vote")
> n <- dim(vote.model$model)[1]
> niter <- 1000
> b.star <- mvrnorm(niter, coef(vote.model), vcov(vote.model))
> pi.star <- invlogit(cbind(1, neut.mean, imm.mean, c(0, 1, 2,
+     3, 4, 5, 6, 7, 8, 9)) %*% t(b.star))
> dim(pi.star)

[1]   10 1000

> for (i in 0:9) {
+     lines(c(i, i), quantile(pi.star[i + 1, ], c(0.025, 0.975)),
+         col = "lightgray")
+ }
> curve(invlogit(cbind(1, neut.mean, imm.mean, x) %*% coef(vote.model)),
+     from = 0, to = 9, lwd = 2, col = "red", add = TRUE)
```

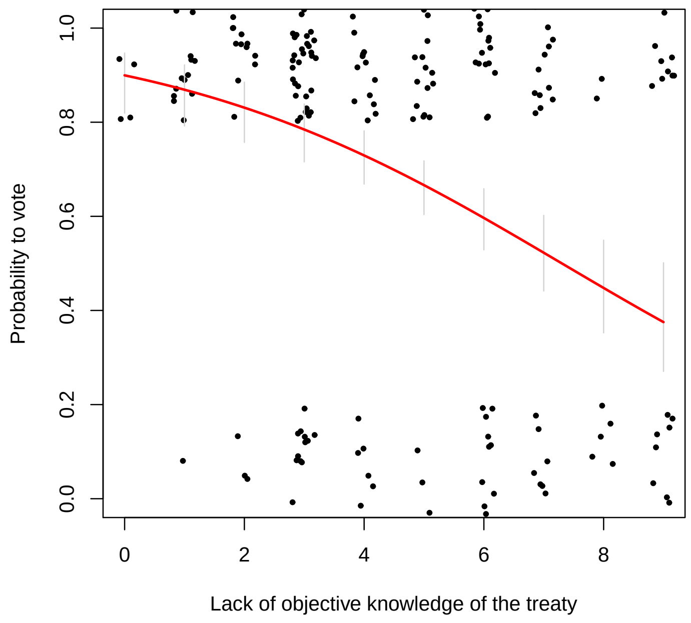

Figure 4: Prediction, with “parameter-simulated” 95% confidence intervals