Chapter 1

Introduction and Overview

The code and explanations for the examples throughout this document are based on R3.4.3 (R Core Team, 2018), SAS9.4 (SAS Institute Inc, 2013), and Stata15 (StataCorp, 2017), although most of the code will run on some earlier versions of these software environments (e.g., the Stata code will run on Stata13). Users also should bear in mind that output from these alternative computing environments may differ slightly, although they should agree fairly closely (typically to at least two places to the right of the decimal-point).

1.3.2 Absolute Bounds Example

1.3.2.1 Analysis in R

The required library is the `MASS` package. The data-file is named `cps0002NoRepeat100.txt`.

```
> library(MASS)
> income <- read.table(file = "cps0002NoRepeat100.txt", header = T)
> attach(income)
```

The graphs of the household income distribution and Q-Q plot and the log-income distribution and Q-Q plot in Figures 1.1 and 1.2 are obtained as follows.

```
> # Income distributions
> par(mfrow = c(1,2))
> truehist(HHINCOME, main = NULL, xlab = "Household Income",
+ breaks = c(seq(0,2210000,10000)), col="gray")
> qqnorm(HHINCOME, ylim = c(-150000,1500000), main = NULL)
> qqline(HHINCOME)
> # log-income distributions
> truehist(log(HHINCOME), main = NULL, xlab = "log Household Income",
+ breaks = c(seq(0,15,0.1)), col="gray")
> qqnorm(log(HHINCOME), main = NULL)
> qqline(log(HHINCOME))
```

We want to fit a "log-normal model". In R there are a few ways of doing this. The lognormal model can be fitted by `glm` with `family = gaussian` and `link = log`. These give identical parameter estimates and standard errors. The linear, log-DV, and log-normal models reported in Chapter 1 can be produced as follows.

```
> # Linear model
> olsmod <- lm(HHINCOME ~ as.factor(FOODSTMP)*as.factor(YEAR));
+ summary(olsmod)
> # Log-DV model
> lgmod <- lm(log(HHINCOME) ~ as.factor(FOODSTMP)*as.factor(YEAR));
+ summary(lgmod)
> # Lognormal model with glm
> lglmod <- glm(HHINCOME ~ as.factor(FOODSTMP)*as.factor(YEAR),
+ family = "gaussian"(link = "log" )); summary(lglmod)
> # Compare log-likelihoods of linear, log-DV, and lognormal models:
> c(logLik(olsmod),logLik(lgmod),logLik(lglmod))
```

Note that the log-likelihoods for the linear and lognormal models are identical, because they both have identical estimates of the means in the original scale of the dependent variable.

Separate Q-Q plots of the residuals of these models for the households with and without food-stamps confirm that the log-DV and log-normal models are more appropriate than the linear model:

```
> par(mfrow = c(1,2))
> # These are residuals plots for the three models.
> # Get the normal distribution Q-Q plots
> qqnorm(residuals(olsmod)[FOODSTMP==1])
> qqline(residuals(olsmod)[FOODSTMP==1])
> qqnorm(residuals(olsmod)[FOODSTMP==2])
> qqline(residuals(olsmod)[FOODSTMP==2])
> # Get the log-DV Q-Q plots
> qqnorm(residuals(lgmod)[FOODSTMP==1])
> qqline(residuals(lgmod)[FOODSTMP==1])
> qqnorm(residuals(lgmod)[FOODSTMP==2])
> qqline(residuals(lgmod)[FOODSTMP==2])
> # Get the lognormal Q-Q plots in the log scale
> qqnorm(log(HHINCOME[FOODSTMP==1])-
+ lglmod$linear.predictors[FOODSTMP==1])
> qqline(log(HHINCOME[FOODSTMP==1])-
+ lglmod$linear.predictors[FOODSTMP==1])
> qqnorm(log(HHINCOME[FOODSTMP==2])-
+ lglmod$linear.predictors[FOODSTMP==2])
> qqline(log(HHINCOME[FOODSTMP==2])-
+ lglmod$linear.predictors[FOODSTMP==2])
```

1.3.2.2 Analysis in SAS

The data-file is named `cps0002NoRepeat100SAS.txt`. We assume that this file has been loaded into a SAS session via the usual data step, and given the name "income". Then a new data-file must be created with the log of income, and dummy variables for the year (2010, 2015) and whether the household received foodstamps or not (1 = yes, 2 = no). The dummy variables are scored 0 and 1, and their product is formed to provide the interaction term for the regression models. The new data-file is named "subincome":

```
data  subincome; set income;
   loghinc = log(HHINCOME);
   yrcat = (YEAR - 2010)/5;
   fdstmp = FOODSTMP - 1;
   yrfdstmp = yrcat*fdstmp;
run;
```

The linear regression model can then be estimated using the REG procedure, with the ods graphics turned on to provide the diagnostics plots:

```
ods graphics on;
proc reg data = subincome  PLOTS(MAXPOINTS= 15000000);
   model HHINCOME = yrcat fdstmp yrfdstmp;
run;
```

The log-DV model can be run either as a linear model or in the `GENMOD` procedure. The linear model version has the advantage of producing the diagnostics plots easily via the ods graphics option. The `GENMOD` procedure's main advantage is to produce outcome tables that are directly comparable with those for the lognormal model. The `GENMOD` syntax is shown next:

```
* OLS model with log-transformed household income;
proc genmod data = subincome;
   model loghinc = yrcat fdstmp yrfdstmp / dist = normal
                   link = identity;
run;
```

Finally, we estimate the lognormal model in `GENMOD`. The code below includes initial values for the coefficients, obtained from a first run where the algorithm did not converge, to aid convergence the second time around.

```
* Generalized linear model with log link;
proc genmod data = subincome;
   model HHINCOME = yrcat fdstmp yrfdstmp / dist = normal
                    link = log  INITIAL = 11.24, 0.124, -1.26, 0.105;
run;
```

1.3.2.3 Analysis in Stata

The data-file is named `cps0002NoRepeat100SAS.dta`. We assume that this file has been loaded into a Stata session. We need the log of household income, and dummy variables for the year (2010, 2015) and whether the household received foodstamps or not (1 = yes, 2 = no).

```
generate loghinc = log(hhincome)
generate yrcat = (year - 2010)/5
generate fdstmp = foodstmp - 1
generate yrfdstmp = yrcat*fdstmp
```

The linear regression model can then be estimated using the `regress` procedure, and residuals saved to produce histogram plots:

```
regress hhincome yrcat##fdstmp
predict e, residual
histogram e, bin(50)
histogram e, bin(50), if foodstmp == 1
histogram e, bin(50), if foodstmp == 2
```

The log-DV model can be run either in `regress` or in the `glm` procedure. Note that the "scale" parameter is presented as the Root MSE in the output. The `regress` syntax is shown next:

```
regress loghinc yrcat##fdstmp
predict f, residual
histogram f, bin(50)
histogram f, bin(50), if foodstmp == 1
histogram f, bin(50), if foodstmp == 2
```

Finally, we estimate the lognormal model using the `glm` procedure:

```
glm hhincome yrcat##fdstmp, family(gaussian) link(log)
```

Note that the AIC and BIC differ from those reported by R and SAS. The `glm` procedure bases its computation of the BIC on deviance instead of the likelihood. On the other hand, `glm` uses the likelihood to compute the AIC, but it is $AIC_{glm} = AIC/N$. Thus, $N * AIC_{glm} = 148076 * 25.37371 = 3757237$, which is close enough to 3757239.0461 if we allow for roundoff error.

1.3.3 Censoring Bounds Example

1.3.3.1 Analysis in R

The required libraries are the `AER` and `MASS` packages. The data-file is named `EthicalSocialRisk.txt`.

```
library(AER)
library(MASS)
ethical <- read.table(file = "EthicalSocialRisk.txt", header = T)
attach(ethical)
```

The linear regression models can be estimated using the `lm` command:

```
> # Null model:
> lmod0 <- lm(ethics ~ 1); summary(lmod0)
> # Model with health as predictor:
> lmod1 <- lm(ethics ~ health); summary(lmod1)
```

The Tobit regression models can be estimated using the `tobit` command:

```
> # Null model:
> tmod0 <- tobit(ethics ~ 1, left = 8, right = Inf,
+ dist = "gaussian"); summary(tmod0)
> # Model with health as predictor:
> tmod1 <- tobit(ethics ~ health, left = 8, right = Inf,
+ dist = "gaussian"); summary(tmod1)
```

The plot of the fitted censored and uncensored normal distributions in Figure 1.3 can be obtained with the following code:

```
> xline <- c(seq(-5,40,0.1))
> ylmod <- c(rep(0,length(xline))); for (i in 1:length(xline))
      {ylmod[i] <- dnorm(xline[i],mean(ethics),sd(ethics))}
> ytmod <- c(rep(0,length(xline))); for (i in 1:length(xline))
      {ytmod[i] <- dnorm(xline[i],tmod0$coefficients,tmod0$scale)}
> truehist(ethics, main = NULL, xlab = "ethical risk taking",
+ breaks = c(seq(7.5,35.5,1)), col="gray", xlim = c(-5,40))
lines(xline,ylmod, lty = 2)
lines(xline,ytmod, lty = 1)
legend("topleft", c("Tobit", "Normal"), lty = c(1,2), horiz = F, bty = "n")
```

1.3.3.2 Analysis in SAS

The data-file is named `EthicalSocialRiskSAS.txt`. We assume that this file has been loaded into a SAS session via the usual data step, and given the name "ethrisk". The linear regression models can be estimated using the REG procedure:

```
* Null model;
proc reg data = ethrisk;
   model ethics =;
run;
* Model with health as a predictor;
proc reg data = ethrisk;
```

```
   model ethics = health;
run;
```

The Tobit models can be estimated in the QLIM procedure. The code below shows the null model and the model with health as predictor, specifying a lower censoring bound of 8.

```
* Null model;
proc qlim data=ethrisk;
   model ethics = ;
   endogenous ethics ~ censored(lb=8);
run;
* Model with health as a predictor;
proc qlim data=ethrisk;
   model ethics = health;
   endogenous ethics ~ censored(lb=8);
run;
```

1.3.3.3 Analysis in Stata

The data-file is named `EthicalSocialRisk.dta`. We assume that this file has been loaded into a Stata session. The linear regression models can then be estimated using the `regress` command:

```
* Null model
regress ethics
* Model with health as a predictor
regress ethics health
```

The Tobit models can be estimated using the `tobit` command, specifying a lower censoring bound of 8:

```
* Null model
tobit ethics, ll(8)
* Model with health as a predictor
tobit ethics health, ll(8)
```

Chapter 2

Models for Singly-Bounded Variables

2.1.2 Time Required to Start a Business by Nation

2.1.2.1 Analysis in R

There is some missing data, so for convenience we eliminate the cases that have missing values. We also need to standardize the two predictors. The predictors are `startprocs`, the average number of processes required to complete prior to starting a business, and `corrup`, the index of corruption described in Chapter 2.

```
> # Load the relevant packages and the data-file
> library(gamlss)
> library(MASS)
> library(lmtest)
> biz <- read.table("time2start.txt", header = T)
> attach(biz)
> # Eliminate the missing-values cases:
> findat <- data.frame(na.omit(cbind(time2start,startprocs,corrup)))
> detach(biz)
> attach(findat)
> # Then standardize the predictor variables, and
> # create a data-set that includes them:
> cstartprocs <- scale(startprocs, center = TRUE, scale = TRUE)
> ccorrup <- scale(corrup, center = TRUE, scale = TRUE)
> findat2 <- data.frame(na.omit(cbind(time2start,cstartprocs, ccorrup)))
> detach(findat); attach(findat2)
```

The linear regression, lognormal, gamma, and Weibull models are fitted next. We use the `gamlss` package to estimate the Weibull and gamma models, and the `glm` procedure to estimate the lognormal and gamma models (the latter as an alternative to compare with the `gamlss` version). The next section of

code estimates each of the relevant models, producing the output summarized in Table 2.1.

```
> # linear regression model
> linmod <- lm(time2start ~ cstartprocs + ccorrup); summary(linmod)
> # gamma model (gamlss version)
> gmod1a <- gamlss(time2start ~ cstartprocs + ccorrup,
+ family = GA); summary(gmod1a)
> # gamma model (glm version)
> glmgmod1a <- glm(time2start ~ cstartprocs + ccorrup,
+ family = Gamma(link = "log")); summary(glmgmod1a)
> # lognormal model
> glmlnmod1a <- glm(time2start ~ cstartprocs + ccorrup,+
+ family = gaussian(link = "log")); summary(glmlnmod1a)
> # Weibull model
> wmod1a <- gamlss(time2start ~ cstartprocs + ccorrup,
+ family = WEI3); summary(wmod1a)
```

The two versions of the gamma model have small but noticeable differences in standard errors and their dispersion parameters. The relevant sections of their outputs are shown next.

```
> # gamlss version
> summary(gmod1a)
Mu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.69404    0.04419   60.97  < 2e-16 ***
cstartprocs  0.64856    0.05223   12.42  < 2e-16 ***
ccorrup     -0.17924    0.04620   -3.88  0.00015 ***
Sigma Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.54561    0.05118  -10.66   <2e-16 ***
> # glm version
> summary(glmgmod1a)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.69404    0.05221  51.597  < 2e-16 ***
cstartprocs  0.64864    0.05609  11.564  < 2e-16 ***
ccorrup     -0.17922    0.05609  -3.195  0.00167 **
---
(Dispersion parameter for Gamma family taken to be 0.4689008)
```

The `gamlss` model in R reports a maximum likelihood estimate that is the square-root of the log of the inverse of the gamma shape parameter: $(1/exp(-0.54561))^2 = 2.977905$. The `glm` model, on the other hand, uses Pearson scaling by default and reports the Pearson estimate of the reciprocal of the shape parameter: $1/0.4689008 = 2.132647$. However, having loaded the `MASS` package, we can get the MLE of the shape parameter from the glm model's fitted object:

```
> gamma.shape(glmgmod1a)
Alpha: 2.9778982
SE:    0.3048454
> # And to get the equivalent dispersion parameter in the output:
> summary(glmgmod1a, dispersion=1/gamma.shape(glmgmod1a)$alpha)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.69404    0.04419  60.971  < 2e-16 ***
cstartprocs  0.64864    0.04747  13.665  < 2e-16 ***
ccorrup     -0.17922    0.04747  -3.776  0.00016 ***
---
(Dispersion parameter for Gamma family taken to be 0.3358073)
```

Checking, we have $1/0.3358073 = 2.977898$, very close to the `gamlss` model estimate. Moreover, the standard errors of the coefficients now are closer to those in the `gamlss` model, due to the different scaling being used. Finally, the comparisons among these models for goodness of fit are provided in the next code chunk (producing the quantiles displayed in Table 2.2).

```
> # Get the log-likelihoods:
> c(logLik(linmod),logLik(gmod1a),logLik(glmgmod1a),
+ logLik(glmlnmod1a),logLik(wmod1a))
> # Compare the models with the data by a 5-number summary:
> quantile(time2start, c(.1,.25,.5,.75,.9))
> quantile(fitted(gmod1a), c(.1,.25,.5,.75,.9))
> quantile(fitted(glmgmod1a), c(.1,.25,.5,.75,.9))
> quantile(fitted(glmlnmod1a), c(.1,.25,.5,.75,.9))
> quantile(fitted(wmod1a), c(.1,.25,.5,.75,.9))
```

We can go beyond modeling the conditional mean in the `gamlss` packaged. It permits the specification of dispersion models with predictors. Accordingly, we estimate models with `cstartprocs` and `ccorrup` in both the location and dispersion submodels, and compare them against their earlier counterparts via likelihood ratio tests. The code chunk below does this for the gamma and Weibull models. In neither of the models did the addition of predictors to the dispersion model significantly improve model fit.

```
> # The gamma and Weibull models now have cstartprocs and
> # corrup in the sigma.formulas.
> # The lrtest command performs the likelihood ratio test.
> gmod3 <- gamlss(time2start ~ cstartprocs + ccorrup, sigma.formula =
+ ~ cstartprocs + ccorrup, family = GA)
> lrtest(gmod1a, gmod3)
> wmod3 <- gamlss(time2start ~ cstartprocs + ccorrup, sigma.formula =
+ ~ cstartprocs + ccorrup, family = WEI3)
> lrtest(wmod1a, wmod3)
```

2.1.2.2 Analysis in SAS

The data-file is named `time2startSAS.txt`. We assume that this file has been loaded into a SAS session via the usual data step, and given the name "time2start". Then we create a data-set with standardized versions of the `startprocs` and `corrup` variables.

```
proc standard data= time2start mean=0 std=1 out=substart;
  var startprocs corrup;
run;
```

The lognormal and gamma models can be estimated in the `GENMOD` procedure, while the Weibull model can be estimated using the `LIFEREG` procedure.

```
* Lognormal model;
proc genmod data = substart;
   model time2start = startprocs corrup / dist = normal
                    link = log;
run;
* gamma model;
proc genmod data = substart;
   model time2start = startprocs corrup / dist = gamma
                    link = log;
run;
* Weibull model;
proc lifereg data = substart;
    model time2start = startprocs corrup / dist = weibull;
run;
```

The Weibull model estimates agree quite closely with their counterparts in `R`, and the SAS gamma model agrees with the `gamlss` version. Adding a `scale = pearson` option to the SAS gamma model produces output that agrees with the `glm` version in `R`. The Scale parameter estimate in the SAS lognormal model is 15.1882 whereas the dispersion parameter reported in the `glm` version is 234.7816, and its square-root is 15.32258, a bit larger than the SAS estimate, being the Pearson estimate rather than the MLE.

2.1.2.3 Analysis in Stata

The data-file is named `time2startStata.dta`. We assume that this file has been loaded into a Stata session. To begin, we create versions of the `startprocs` and `corrup` variables.

```
egen cstartprocs = std( startprocs )
egen ccorrup = std( corrup )
```

Next, we estimate the lognormal and gamma models using the `glm` command. These yield similar results to those in `R` and `SAS`. The scale parameters in both models are Pearson estimates, corresponding to their `glm` counterparts in `R`.

```
glm time2start cstartprocs ccorrup , family(gaussian) link(log)
glm time2start cstartprocs ccorrup , family(gamma) link(log)
```

Estimating the Weibull model in Stata requires the `streg` routine, which is akin to the `LIFEREG`. procedure in `SAS`. There are two steps involved: First, telling Stata that the dependent variable is a "time" variable, and then running the model.

```
stset time2start
streg cstartprocs ccorrup , distribution(weibull) nohr
```

The `streg` routine estimates a log proportional-hazards version of the Weibull model, rather than the mean-dispersion model estimated in the `R` and `SAS` routines. In `R` the proportional-hazards Weibull model can be estimated in the `gamlss` package by specifying WEI2 in place of WEI3.

The conversion between the two sets of parameters is as follows. Denote the scale parameter by σ, the proportional-hazards coefficients by β_h, and the scale-shape model coefficients by β_s. Then the conversion is $\beta_h = -\beta_s/\sigma$.

2.1.3 Performance in a Stroop Task Experiment

2.1.3.1 Analysis in R

The analysis begins by subtracting 250 milliseconds from the response-time variable so that it can be treated as a variable with 0 as a lower bound.

```
> # Load the relevant packages and the data-file
> library(gamlss)
> library(MASS)
> library(lmtest)
> stroop <- read.table("stroopdataRT.txt", header = T)
> attach(stroop)
# We will treat the RT variable as having a lower bound of 250 ms.
> rt250 <- StRTOverall - 250
> findat <- na.omit(stroop[, c("rt250", "Condition", "LifeOrientation")])
> attach(findat)
```

The gamma, Weibull, log-normal, and linear regression models containing main and interaction effects for the experimental condition and the LOT-R score are estimated in the next code chunk, with their respective log-likelihoods and outputs displayed in Table 2.3 and the accompanying text. As in the preceding example, the gamma and Weibull models are estimated using the `gamlss` package and the lognormal model is estimated using the `glm` function.

```
> # gamma model
> gmod <- gamlss(findat$rt250 ~ Condition*LifeOrientation,
+ family = GA, data = findat)
> logLik(gmod); summary(gmod)
```

```
> # Weibull model
> wmod <- gamlss(findat$rt250 ~ Condition*LifeOrientation,
+ family = WEI3, data = findat)
> logLik(wmod); summary(wmod)
> # lognormal model
> lnmod <- glm(findat$rt250 ~ Condition*LifeOrientation,
+ family = gaussian(link = "log"))
> logLik(lnmod); summary(lnmod)
> # linear regression model
> linmod <- lm(findat$rt250 ~ Condition*LifeOrientation)
> logLik(linmod); summary(linmod)
```

2.1.3.2 Analysis in SAS

The data-file is named `stroopdataRTSAS.txt`. We assume that this file has been loaded into a SAS session via the usual data step, and given the name "stroop". Then we create a data-set with the rt250 and interaction term variables.

```
data  substroop; set stroop;
   rt250 = StRTOverall - 250;
   conLOT = Condition*LifeOrientation;
run;
```

The lognormal and gamma models can be estimated in the `GENMOD` procedure, while the Weibull model can be estimated using the `LIFEREG` procedure.

```
* Lognormal model;
proc genmod data = substroop;
   model rt250 = Condition LifeOrientation conLOT / dist = normal
                    link = log;
run;
* gamma model;
proc genmod data = substroop;
   model rt250 = Condition LifeOrientation conLOT / dist = gamma
                    link = log;
run;
* Weibull model;
proc lifereg data = substroop;
    model rt250 = Condition LifeOrientation conLOT / dist = weibull;
run;
```

The Weibull model estimates agree fairly closely with their counterparts in `R`, and the SAS gamma model agrees with the `gamlss` version. The Scale parameter estimate in the SAS lognormal model is 117.7923 whereas the dispersion parameter reported in the `glm` version is 14586.65, and its square-root is 120.7752, a bit larger than the SAS estimate, being the Pearson estimate rather than the MLE.

2.1.3.3 Analysis in Stata

The data-file is named `StroopDataRTStata.dta`. We assume that this file has been loaded into a Stata session. To begin, we create the Rt 250 variable and the interaction variable.

```
generate rt250 = strtoverall  250
generate conLOT = condition*lifeorientation
```

Next, we estimate the lognormal and gamma models using the `glm` command. These yield similar results to those in `R` and `SAS`. The scale parameters in both models are Pearson estimates.

```
glm rt250 condition lifeorientation conLOT, family(gaussian) link(log)
glm rt250 condition lifeorientation conLOT, family(gamma) link(log)
```

Estimating the Weibull model in Stata requires the `streg` routine, which is akin to the `LIFEREG`. procedure in `SAS`. There are two steps involved: First, telling Stata that the dependent variable is a "time" variable, and then running the model.

```
stset rt250
streg condition lifeorientation conLOT, distribution(weibull) nohr
```

As mentioned earlier, the `streg` routine estimates a log proportional-hazards version of the Weibull model, rather than the mean-dispersion model estimated in the `R` and `SAS` routines. In `R` the proportional-hazards Weibull model can be estimated in the `gamlss` package by specifying WEI2 in place of WEI3.

2.2 Model Diagnostics

The runs in this subsection build on the models estimated in subsection 2.1.2. The parameter estimate correlations matrices for all four models in R (three of which are tabulated in Table 2.4) are produced by the next code chunk.

```
cov2cor(vcov(gmod1a))
cov2cor(vcov(wmod1a))
cov2cor(vcov(glmlnmod1a))
cov2cor(vcov(linmod))
```

In SAS, the parameter estimates correlation matrix can be obtained by including the CORRB option with the model statement. In Stata, the post-estimation VCE is available.

2.3.1 Ambulance Arrival Times

These data are analyzed with two kinds of models: Hurdle and Tweedie distribution models. In each of the subsections to follow, we present the hurdle models first and the Tweedie model thereafter (if the latter is available).

2.3.1.1 Analysis in R

The analysis begins by subtracting 0.5 minutes from the response-time variable so that it can be treated as a variable with 0 as a lower bound.

```
> # Load the relevant packages and the data-file
> library(gamlss)
> library(MASS)
> library(lmtest)
> library(cplm)
> library(statmod)
> library(tweedie)
> ambolong <- read.table("ambolong.txt", header = T)
> attach(ambolong)
# We treat the RT variable as having a lower bound of 0.
> minut0 <- minut - 0.5
```

The hurdle models begin with a logistic regression model for the zeroes:

```
# First, construct the binary variable identifying the zeroes:
> zeromin <- c(rep(0,length(minut0)))
> for (i in 1:length(minut0))
+ {ifelse(minut0[i] == 0, zeromin[i] <- 1, zeromin[i] <- 0)}
> table(zeromin)
> # Then model the zeroes with logistic regression
> zmod2 <- glm(zeromin ~ as.factor(year)*as.factor(region),
+ family = "binomial")
> summary(zmod2)
## Coefficients:
##                                       Estimate Std. Error z value
## (Intercept)                           -1.66442    0.16079 -10.351
## as.factor(year)2016                   -0.07146    0.22677  -0.315
## as.factor(region)2                    -0.26182    0.22780  -1.149
## as.factor(region)3                     0.15538    0.20995   0.740
## as.factor(region)4                    -0.31048    0.24361  -1.274
## as.factor(region)5                    -0.42125    0.28647  -1.471
## as.factor(region)6                    -0.22012    0.27175  -0.810
## as.factor(region)7                     0.34874    0.37570   0.928
## as.factor(year)2016:as.factor(region)2  0.03995    0.33040   0.121
## as.factor(year)2016:as.factor(region)3  0.29264    0.29257   1.000
## as.factor(year)2016:as.factor(region)4  1.10190    0.31852   3.459
## as.factor(year)2016:as.factor(region)5  0.57199    0.39475   1.449
## as.factor(year)2016:as.factor(region)6  0.40096    0.36748   1.091
## as.factor(year)2016:as.factor(region)7 -0.11694    0.53754  -0.218
> # Region 4 is the big standout in 2016; no others differ strongly
> # from the baseline region in 2015-16.
```

Next, the non-zero response times are analyzed with gamma, Weibull, and lognormal models. All of these give similar results, so only the gamma model output is shown below.

```
> gmod2 <- gamlss(minut0[minut0 > 0] ~ as.factor(year[minut0 > 0])+
+ as.factor(region[minut0 > 0]), family = GA); summary(gmod2)
> wmod2 <- gamlss(minut0[minut0 > 0] ~ as.factor(year[minut0 > 0])+
+ as.factor(region[minut0 > 0]), family = WEI3); summary(wmod2)
> lnmod2 <- glm(minut0[minut0 > 0] ~ as.factor(year[minut0 > 0])+
+ as.factor(region[minut0 > 0]), family = gaussian(link = "log"));
+ summary(lnmod2)
> #
> # Gamma model output:
## Mu link function:  log
## Mu Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      1.89032    0.03215  58.794  < 2e-16 ***
## as.factor(year[minut0 > 0])2016 -0.16197    0.02518  -6.434 1.46e-10 ***
## as.factor(region[minut0 > 0])2   0.06163    0.04091   1.507   0.1320
## as.factor(region[minut0 > 0])3   0.03113    0.04031   0.772   0.4399
## as.factor(region[minut0 > 0])4  -0.07429    0.04289  -1.732   0.0833 .
## as.factor(region[minut0 > 0])5   0.05914    0.04943   1.196   0.2316
## as.factor(region[minut0 > 0])6   0.19291    0.04760   4.053 5.20e-05 ***
## as.factor(region[minut0 > 0])7   0.13092    0.07763   1.687   0.0918 .
##
## -------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4076     0.0125  -32.62   <2e-16 ***
```

Next, the Tweedie model is fitted. This produces the output shown in Table 2.5.

```
> tmod1 <- cpglm(minut0 ~ as.factor(year)+as.factor(region))
> summary(tmod1)
> # Compare the AICs for the Tweedie and hurdle models:
> c(AIC(tmod1),AIC(gmod2)+AIC(zmod2),AIC(wmod2)+AIC(zmod2),
+ AIC(lnmod2)+AIC(zmod2))
```

The next Tweedie model includes the effect of the year, population density, and percentage classified as disabled. This produces the output shown in Table 2.6.

```
> tmod3 <- cpglm(minut0 ~ as.factor(year) + density + pctdisab)
> summary(tmod3)
```

Finally, here is how to get the plot in Figure 2.5 comparing the null-model fitted gamma, Weibull, and Tweedie distributions:

```
> # Estimate the null models:
> tmod0 <- cpglm(minut0 ~ 1)
> gmod0 <- gamlss(minut0[minut0 > 0] ~ 1, family = GA)
> wmod0 <- gamlss(minut0[minut0 > 0] ~ 1, family = WEI3)
> # Compute the probability for inflating the hurdle pdfs:
> zerotab <- table(zeromin)
> probtab <- zerotab[1]/sum(zerotab)
> # Now do the plot:
> xline <- seq(0,29,0.25)
> truehist(minut0, nbins = 29, xlab = "minutes")
> lines(xline,dtweedie(xline,mu=exp(tmod0$coefficients),xi=tmod0$p,
+ phi=tmod0$phi), lty = 1, lwd = 2.5)
> lines(xline,probtab*dGA(xline,mu=exp(gmod0$mu.coefficients),
+ sigma = exp(gmod0$sigma.coefficients)), lty = 2, lwd = 2.0)
> lines(xline,probtab*dWEI3(xline,mu=exp(wmod0$mu.coefficients),
+ sigma = exp(wmod0$sigma.coefficients)), lty = 3, lwd = 2.0)
> legend("topright", c("Gamma", "Tweedie", "Weibull"), lty = c(2,1,3),
+ lwd = c(2.5,2.0,2.0), horiz = F, bty = "n")
```

2.3.1.2 Analysis in SAS

The hurdle models begin with a logistic regression model for the zeroes. The data-file is named `ambolongSAS.txt`. We assume that this file has been loaded into a SAS session via the usual data step, and given the name "ambo". Then we create a data-set with the zeroed call-out times (minut0) and the binary variable identifying the zeroes (zeromin).

```
data  subambo; set ambo;
   minut0 = minut - 0.5;
   zeromin = 1;
   if minut0 = 0 then zeromin = 0;
run;
* Verify that zeromin has found the zeroes;
proc freq data=subambo;
  tables zeromin;
run;
```

Next, we run the logistic regression model. Note that we have to use the ref = first option to ensure that the first category is the reference category in the region and year variables (SAS defaults to the last category as the reference).

```
proc logistic data = subambo;
  class region  / param=ref ref=first;
  class year /param=ref ref=first;
```

```
  model zeromin = region|year;
run;
```

Now the non-zero response times are analyzed with gamma, Weibull, and log-normal models. All of these give similar results. Note that the `LIFEREG` procedure doesnt permit the param=ref or ref=first options. So the analysis uses the last category of each categorical variable as the reference category.

```
* Gamma model;
proc genmod data = subambo;
  class region  / param=ref ref=first;
  class year /param=ref ref=first;
  model minut0 = region year / dist = gamma
                   link = log;
  where minut0 NE 0;
run;
* Lognormal model;
proc genmod data = subambo;
  class region  / param=ref ref=first;
  class year /param=ref ref=first;
  model minut0 = region year / dist = normal
                   link = log;
  where minut0 NE 0;
run;
* Weibull model;
proc lifereg data = subambo;
  class region;
  class year;
  model minut0 = region year / dist = weibull;
  where minut0 NE 0;
run;
```

The Tweedie model can be run in SAS 9.4 but not earlier versions, using the `genmod` procedure. The next block of code produces output that closely agrees with the Tweedie model in R (the output in Table 2.5).

```
proc genmod data = subambo;
  class region  / param=ref ref=first;
  class year /param=ref ref=first;
  model minut0 = region year / dist = tweedie
                   link = log;
run;
```

2.3.1.3 Analysis in Stata

The hurdle models begin with a logistic regression model for the zeroes. The data file is named `ambolongStata.dta`. We assume that this file has been

loaded. First we create the zeroed callout time (minut0) and the binary variable identifying the zeroes (zeromin).

```
generate minut0 = minut - 0.5
generate zeromin = 0
replace zeromin = 1 if minut0 == 0
```

Next, we run the logistic regression model. There are a few ways this can be done in Stata, but here we have used the `logit` command. Note that we are treating the year and region variables as factor variables.

```
logit zeromin i.year#i.region
```

The 2015 x region coefficients are in close agreement with those from SAS and R. Likewise, the log-likelihood is the same. However, the 2016 x region coefficients are not those that are given in the SAS and R output. Instead, they are related to them in the following way. Denote the R or SAS 2016 x region coefficients by $\gamma_{16(j)}$ and the Stata coefficients by $\beta_{15(j)}$ and $\beta_{16(j)}$, for $j = 1, \ldots, 7$. Then $\gamma_{16(j)} = \beta_{16(j)} - \beta_{15(j)} + \beta_{16(1)}$.

Next, we estimate the lognormal and gamma models using the `glm` command. The "if" command selects the nonzero cases for the dependent variable. These models yield similar results to those in `R` and `SAS`. The scale parameters in both models are Pearson estimates.

```
glm minut0 i.year i.region if minut0>0, family(gaussian) link(log)
glm minut0 i.year i.region if minut0>0, family(gamma) link(log)
```

As before, estimating the Weibull model in Stata requires the `streg` routine, which is akin to the `LIFEREG`. procedure in `SAS`. There are two steps involved: First, telling Stata that the dependent variable is a "time" variable, and then running the model.

```
stset minut0
streg i.year i.region if minut0>0, distribution(weibull) nohr
```

As mentioned earlier, the `streg` routine estimates a log proportional-hazards version of the Weibull model, rather than the mean-dispersion model estimated in the `R` and `SAS` routines. In `R` the proportional-hazards Weibull model can be estimated in the `gamlss` package by specifying WEI2 in place of WEI3.

There currently is no Tweedie model in Stata, although it is possible that one will be developed in the future.

Chapter 3

Location-Scale Models for Doubly-Bounded Variables

The sections in this Chapter provide methods and code for reproducing the analyses in Chapter 3 for the "Measuring Speakers' Grammaticality Judgments" and "Probability Judgments in Moral Dilemmas" examples.

3.1 Measuring Speakers' Grammaticality Judgments

3.1.1 Analysis in R

The Rdata file `chap3_data.Rdata` has two datasets for the grammar study example.

1. The `grammastudy` file is the original data file named `BNC_Clean_MOP100only.xlsx`.
2. The `grammar1` file is the processed data file for analysis. It contains the following variables:
   - `MOP`: the rating (from 0 to 100) on how grammatical the sentence is.
   - `Language`: the language of the sentences
   - `Text`: the contents of the sentences
   - `Length`: the length of the sentence
   - `rate`: the average rating of that sentence from all raters
   - `ratep`: the average rating of that sentence from all raters as [0,1] scale
   - `Ratings`: same as `ratep`
   - `Language1`: language variable with factor contrast structure, where English is used as the base level

- `Length1`: the centered `Length` value.

We first load the necessary packages and the data file:

```
> # For plotting
> library(ggplot2)
> library(MASS)
> # For analysis
> library(betareg) # main package for beta GLMs
> library(gamlss) # Alternative package for beta GLMs and package
> # for one/zero-inflated beta GLM.
> library(cdfquantreg)
> library(lmtest)
> # Load data
> load("chap3_data.Rdata")
```

We then fit the data using beta regression model function `betareg()`

```
> model_a <- betareg(ratep ~ Language + Length1 | Language  + Length1,
+ data = grammar1)
> summary(model_a)
```

Here are some selected output from R

```
## Coefficients (mean model with logit link):
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.82526    0.11940  15.287  < 2e-16 ***
## LanguageSpanish   -0.88317    0.19449  -4.541 5.60e-06 ***
## LanguageJapanese  -2.19030    0.16968 -12.909  < 2e-16 ***
## LanguageNorwegian -0.89707    0.17407  -5.154 2.56e-07 ***
## LanguageChinese   -1.84278    0.17365 -10.612  < 2e-16 ***
## Length1           -0.02782    0.01092  -2.547   0.0109 *
## Phi coefficients (precision model with log link):
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.40167    0.20408  11.768  < 2e-16 ***
## LanguageSpanish   -1.16886    0.28662  -4.078 4.54e-05 ***
## LanguageJapanese  -1.11557    0.26177  -4.262 2.03e-05 ***
## LanguageNorwegian -0.73767    0.28119  -2.623 0.008705 **
## LanguageChinese   -1.05097    0.27281  -3.852 0.000117 ***
## Length1            0.03937    0.01634   2.410 0.015960 *
```

```
> confint(model_a)
```

```
##                                2.5 %      97.5 %
## (Intercept)              1.591251150  2.059275031
## LanguageSpanish         -1.264360038 -0.501980095
## LanguageJapanese        -2.522863834 -1.857742631
## LanguageNorwegian       -1.238231432 -0.555904108
## LanguageChinese         -2.183121582 -1.502433330
## Length1                 -0.049225989 -0.006412803
## (phi)_(Intercept)        2.001678788  2.801668518
## (phi)_LanguageSpanish   -1.730615962 -0.607101748
## (phi)_LanguageJapanese  -1.628634490 -0.602511967
## (phi)_LanguageNorwegian -1.288783591 -0.186557806
## (phi)_LanguageChinese   -1.585672520 -0.516272447
## (phi)_Length1            0.007349298  0.071388246
```

The following code generates residuals and influence statistics plots such as those displayed in Figure 3.5.

```
 gy1_res <- cbind(
  residuals(model_a, type = "pearson"),
  residuals(model_a, type = "deviance"),
  residuals(model_a, type = "response"),
  residuals(model_a, type = "weighted"),
  residuals(model_a, type = "sweighted"),
  residuals(model_a, type = "sweighted2"),
  cooks.distance(model_a)
)
colnames(gy1_res) <- c("pearson", "deviance", "response",
  "weighted", "sweighted", "sweighted2", "cooks")

par(mfrow = c(2,3), mar = c(4, 2, 1, 1), cex.lab = 1.5)
for (i in 1:ncol(gy1_res)){
  lab <- gy1_res[, i]
  lab1 <- 1:length(gy1_res[, i])
  lab1[which(lab > quantile(gy1_res[, i], c(0.01)) &
  lab<quantile(gy1_res[, i], c(0.99)))] <- NA
  plot(gy1_res[, i], xlab = colnames(gy1_res)[i], type = "p", pch = 3, cex = 0.75)
  text(gy1_res[, i], labels = lab1)
}
```

We then see if there are significant interactions in the location and dispersion submodels using the likelihood ratio test.

```
> model6 <- betareg(ratep ~ Language * Length | Language  + Length, data = grammar1)
> model7 <- betareg(ratep ~ Language * Length |Language * Length, data = grammar1)
> lrtest(model5, model6, model7)
```

```
## Likelihood ratio test
##
## Model 1: ratep ~ Language + Length1 | Language + Length1
## Model 2: ratep ~ Language * Length | Language + Length
## Model 3: ratep ~ Language * Length | Language * Length
##   #Df LogLik Df  Chisq Pr(>Chisq)
## 1  12 100.36
## 2  16 103.02  4 5.3356     0.2546
## 3  20 104.18  4 2.3162     0.6778
```

In chapter 4, we also apply the cdf-quantile regression using logit-logistic distribution to the grammar example data:

```
> library(cdfquantreg)
> model8 <- cdfquantreg(ratep ~ Language + Length1 | Language  + Length1,
+ data = grammar1, fd = "logit", sd = "logistic")
> summary(model8)
```

3.1.2 Analysis in SAS

The data-file is named `grammar1.txt` and the SAS program file for reading the data file is `grammar1.sas`. The categorical variable language was coded into separate columns using base coding where English is treated as the base group. The code for running the beta regression in SAS is:

```
%INCLUDE grammar1;
proc nlmixed data = grammar1 tech = trureg hess cov itdetails;
y = ratep;
xb = b0 + b1 * LanguageSpanish+ b2 * LanguageJapanese + b3 * LanguageNorwegian+
b4 * LanguageChinese + b5 * Length1;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1 * LanguageSpanish+ d2 * LanguageJapanese + d3 * LanguageNorwegian+
d4 * LanguageChinese + d5 * Length1;
phi = exp(1*wd);
p = mu*phi;
q = phi - mu*phi;
ll = lgamma(p+q) - lgamma(p) - lgamma(q) + (p-1)*log(y) + (q-1)*log(1 - y);
model y ~ general(ll);
predict mu out = grammar_pred_het;
run;
```

Users could also implement the `Beta_Regression` macro provided by Swearingen (2011).

```
%INCLUDE SAS_BetaReg_Macro;
/*Usage:
%Beta_Regression(Dataset,tech, details, Vars, Vars2,
```

```
depvar, residuals, gpath);
Argument:
 Dataset  the LIBNAME.DATA file
 tech  allows for different optimization schemes to be used
 details  allows for other options to be specified
 Vars  list of mean covariates or double single quotes [] if none
 Vars2  list of precision covariates or double single quotes [] if none
 depvar  the dependent variable scaled into the (0,1) open interval
 residuals  specify 1 to generate Pearson and Standardized Residual plots
 gpath  specify output directory for residual plots. */

%Beta_Regression(grammar1,trureg,hess cov itdetails,
LanguageSpanish LanguageJapanese LanguageNorwegian LanguageChinese Length1,
LanguageSpanish LanguageJapanese LanguageNorwegian LanguageChinese Length1,
ratep, 1, "\\plot\");
```

In chapter 4, we also apply the cdf-quantile regression using logit-logistic distribution to the grammar example data:

```
%INCLUDE SAS_MACRO;
%cdfquantreg(grammar1, ratep, 'logit', 'logistic',
LMIV=LanguageSpanish LanguageJapanese LanguageNorwegian LanguageChinese Length1,
DMIV= LanguageSpanish LanguageJapanese LanguageNorwegian LanguageChinese Length1,
init=0.1| 0.1 |0.1 |0.1 |0.1 |0.1|0.1 |0.1| 0.1| 0.1 |0.1 |0.1);
```

3.1.3 Analysis in Stata

Stata 14 or newer uses "betareg", whereas Stata 11 or newer has the "betafit" (Buis, Cox, & Jenkins, 2003) user-coded estimation command utility. Initially we will run models using betafit and then show how the same models may be run using betareg.

To use betafit, we need to install it. When installation is complete, we can run the models for this example. The data-file is named `grammar1.dta`. We assume that this data file already has been uploaded. The three models in Stata using betafit give very similar results to those from `R` and `SAS`.

```
* Installing betafit
. ssc install betafit
* First, set up a way of using the string variable language
. encode language, gen(lang)
* Run the linear model with English as the base-group
. betafit ratep, muvar(ib2.lang c.length1) phivar(ib2.lang c.length1)
. estimates store B
* Run a model with an interaction term in the location submodel
. betafit ratep, muvar(ib2.lang##c.length1) phivar(ib2.lang c.length1)
. estimates store C
```

```
* Run model with interaction in both submodels
. betafit ratep, muvar(ib2.lang##c.length1) phivar(ib2.lang##c.length1)
. estimates store D
* Likelihood-ratio tests
. lrtest B C
. lrtest C D
```

Each model's estimates are stored separately via the `estimates store` command. These are available for further analysis, including model comparison via likelihood ratio tests, as demonstrated in the code above. Turning now to diagnostics, betafit includes options for Pearson and standardized weighted residuals, but no influence statistics per se. The next code chunk reruns the main-effects model, generates a case identification variable, and provides Pearson and weighted residuals plots of the kind found in Figure 3.5.

```
* Pearson residuals and weighted residuals 1 (scresid) are available in betafit.
. quietly betafit ratep, muvar(ib2.lang c.length1) phivar(ib2.lang c.length1)
* Obtain the two types of residuals:
. predict residuals, pearson
. predict wresid1, scresidual
* Generate the case identification variable and residuals plots:
. gen caseno = _n
. twoway (scatter wresid1 caseno)
. twoway (scatter residuals caseno)
```

The next code chunk shows how to run the same three models using the `betareg` command in Stata 14, assuming that the `lang` variable already has been generated.

```
* Run the linear model with English as the base-group
. betareg ratep ib2.lang c.length1, scale(ib2.lang c.length1)
. estimates store B
* Run model with interaction in the location submodel
. betareg ratep ib2.lang##c.length1, scale(ib2.lang c.length1)
. estimates store C
* Run model with interaction in both submodels
. betareg ratep ib2.lang##c.length1, scale(ib2.lang##c.length1)
. estimates store D
* Likelihood-ratio tests
. lrtest B C
. lrtest C D
```

3.2 Probability Judgments in Moral Dilemmas

The Rdata file `chap3_data.Rdata` also has two datasets for the moral study example.

1. `Study_Shou_Data` file is the original data file, it has variables inlcuding:
    - `Study`: Shou & Song (2017) reported two experiments, the data file has data from study 1 (`Study = 1`) and from study 2 (`Study = 2`). * `ID`: the subject ID in the experiment
    - `Case`: dilemma type, `car` = car dilemma, `hostage` = hostage dilemma
    - `ActualChoice`: subjects' choices; "KL" = K; "NK" = ˜K
    - `condition`: `twodecisionlate` = two alternatives, probability judgments are made before choices; `twodecisionfirst` = two alternatives, choices are made before probability judgments
    - `thirdgood` = three options; the third is the best choice
    - `PO_KL`: Probability estimate of positive outcome given K
    - `NO_KL`: Probability estimate of negative outcome given K
    - `PO_NK`: Probability estimate of positive outcome given ˜K
    - `NO_NK`: Probability estimate of negative outcome given ˜K
2. `Study_Shou_Data_long` file is a long data format with all four probabilities in a single variable column

3.2.1 Analysis in R

```
> # For data processing
> library(psych)
> library(reshape2)
> # For plotting
> library(ggplot2)
> library(MASS)
> #For analysis
> library(betareg)
> library(gamlss)
> library(cdfquantreg)
> library(lmtest)
> # Load data
> load("chap3_data.Rdata")
```

We then fit the data with a beta regression model, for each of the four probabilities. For the four probabilities,

```
> datamr <- subset(Study_Shou_Data, Study == 1)
> # The probability of five people would survive given kill
> modelmj1 <- betareg(PO_KL ~ Case | Case , data = datamr)
> summary(modelmj1)
>
> # The probability of one person would survive given kill
> Study_Shou_Data$PO_NK = scaleTR(Study_Shou_Data$PO_NK)
> modelmj2 <- betareg(PO_NK ~ Case | Case , data = datamr)
> summary(modelmj2)
```

```
>
> # The probability of one person would survive given kill
> Study_Shou_Data$PO_NK = scaleTR(Study_Shou_Data$PO_NK)
> modelmj2 <- betareg(PO_NK ~ Case | Case , data = datamr)
> summary(modelmj2)
>
> # The probability that one person will die given kill,
> modelmj3 <- betareg(NO_KL ~ Case | Case , data = datamr)
> summary(modelmj3)
>
> # The probability that five persons will die given not kill,
> modelmj4 <- betareg(NO_NK ~ Case | Case , data =datamr)
> summary(modelmj4)
```

The next code chunk reruns the four models, producing the "bias-corrected" parameter estimates and their standard errors, as displayed in the right-most columns of Table 3.2 in Chapter 3.

```
> # For the probability that five persons will survive
> # given that killing has been chosen:
> modelmj1 <- betareg(PO_KL ~ Case | Case , data = datamr, type = "BC")
> summary(modelmj1)
> #
> # For the probability that one person will survive
> # given that not killing has been chosen:
> modelmj2 <- betareg(PO_NK ~ Case | Case , data = datamr, type = "BC")
> summary(modelmj2)
> #
> # For the probability that one person will die
> # given killing has been chosen:
> modelmj3 <- betareg(NO_KL ~ Case | Case , data = datamr, type = "BC")
> summary(modelmj3)
> #
> # For the probability that five persons will die given
> # that not killing has been chosen:
> modelmj4 <- betareg(NO_NK ~ Case | Case , data = datamr, type = "BC")
> summary(modelmj4)
```

The following code generates residuals and influence statistics plots such as those displayed in Figure 3.5.

```
 gy1_res <- cbind(
  residuals(modelmj4, type = "pearson"),
  residuals(modelmj4, type = "deviance"),
  residuals(modelmj4, type = "response"),
```

```
  residuals(modelmj4, type = "weighted"),
  residuals(modelmj4, type = "sweighted"),
  residuals(modelmj4, type = "sweighted2"),
  cooks.distance(modelmj4)
)
colnames(gy1_res) <- c("pearson", "deviance", "response",
  "weighted", "sweighted", "sweighted2", "cooks")

par(mfrow = c(2,3), mar = c(4, 2, 1, 1), cex.lab = 1.5)
for (i in 1:ncol(gy1_res)){
  lab <- gy1_res[, i]
  lab1 <- 1:length(gy1_res[, i])
  lab1[which(lab > quantile(gy1_res[, i], c(0.01)) &
  lab<quantile(gy1_res[, i], c(0.99)))] <- NA
  plot(gy1_res[, i], xlab = colnames(gy1_res)[i], type = "p", pch = 3, cex = 0.75)
  text(gy1_res[, i], labels = lab1)
}
```

The one-inflated beta regression model was implemented in the `gamlss` package.

```
> data_mj <- subset(Study_Shou_Data_long, Study == 1)
> data_mj$prob_est1 <- data_mj$prob_est
> data_mj$prob_est1[which(data_mj$prob_est1!=1)] =
+ scaleTR(data_mj$prob_est1[which(data_mj$prob_est1!=1)])
> model <- gamlss(prob_est1~Case, sigma.fo=~ Case, nu.formula = ~Case,
+ family=BEOI, data=subset(data_mj, OutcomeType=="PO"))
> summary(model)
```

We modified the `confint` function to extract confidence intervals of the model. Here is how we obtained the confidence intervals:

```
> confint1 <- function (object, submodel, level = 0.95)
> {
>     cf <- coef(object, what = submodel)
>     pnames <- names(cf)
>     parm <- pnames
>     a <- (1 - level)/2
>     a <- c(a, 1 - a)
>     pct <- paste(format(100 * a, trim = TRUE, scientific = FALSE, digits = 3), "%")
>     fac <- qnorm(a)
>     ci <- array(NA, dim = c(length(parm), 2L),
>  dimnames = list(parm, pct))
>     if(submodel == "mu") {
>       ind = 1:length(pnames)
```

```
>     }else if (submodel == "sigma"){
>       ind = (1 + length(pnames)):(2*length(pnames))
>     }else if (submodel == "nu"){
>       ind = (1 + 2*length(pnames)):(3*length(pnames))
>     }
>     ses <- sqrt(diag(vcov(object)[ind, ind]))[parm]
>    ci[] <- cf[parm] + ses %o% fac
>     ci
>}
> confint1(model2, "mu")
> confint1(model2, "sigma")
> confint1(model2, "nu")
```

3.2.2 Analysis in SAS

The data-file is named `Study_Shou_Data.txt` and the SAS program file for reading the data file is `Study_Shou_Data.sas`.

We first demonstrate the use of the fixed-effects beta regression in SAS via the `nlmixed` procedure. The following shows the basic procedure for running beta regressions of this kind.

For the probability that five people will survive given killing is chosen:

```
%INCLUDE Study_Shou_Data;
proc nlmixed data = Study_Shou_Data tech = trureg hess cov itdetails;
title 'Beta regression predicting the probability that
five people would survive given kill is chosen';
y = PO_KL1;
xb = b0 + b1 * Case1;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1 * Case1;
phi = exp(1*wd);
p = mu*phi;
q = phi - mu*phi;
ll = lgamma(p+q) - lgamma(p) - lgamma(q) + (p-1)*log(y) + (q-1)*log(1 - y);
model y ~ general(ll);
run;
```

For the probability that one person will die given killing is chosen:

```
proc nlmixed data = Study_Shou_Data tech = trureg hess cov itdetails;
title 'Beta regression predicting the probability that
one person would die given kill is chosen';
y = NO_KL1;
xb = b0 + b1 * Case1;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1 * Case1;
```

```
phi = exp(1*wd);
p = mu*phi;
q = phi - mu*phi;
ll = lgamma(p+q) - lgamma(p) - lgamma(q) + (p-1)*log(y) + (q-1)*log(1 - y);
model y ~ general(ll);
run;
```

For the probability that five people will die given not killing is chosen:

```
proc nlmixed data = Study_Shou_Data tech = trureg hess cov itdetails;
title 'Beta regression predicting the probability that
five people would die given not kill is chosen';
y = NO_NK1;
xb = b0 + b1 * Case1;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1 * Case1;
phi = exp(1*wd);
p = mu*phi;
q = phi - mu*phi;
ll = lgamma(p+q) - lgamma(p) - lgamma(q) + (p-1)*log(y) + (q-1)*log(1 - y);
model y ~ general(ll);
run;
```

For the probability that one person will survive given not killing is chosen:

```
proc nlmixed data = Study_Shou_Data tech = trureg hess cov itdetails;
title 'Beta regression predicting the probability that
one person would survive given not kill is chosen';
y = PO_NK1;
xb = b0 + b1 * Case1;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1 * Case1;
phi = exp(1*wd);
p = mu*phi;
q = phi - mu*phi;
ll = lgamma(p+q) - lgamma(p) - lgamma(q) + (p-1)*log(y) + (q-1)*log(1 - y);
model y ~ general(ll);
run;
```

Users could also employ the `Beta_Regression` macro provided by Swearingen (2011).

```
%INCLUDE SAS_BetaReg_Macro;
%Beta_Regression(Study_Shou_Data,trureg,hess cov itdetails,Case1,Case1,PO_KL1, 1,
"\\moraljudgment\plot\");
```

We now demonstrate the implementation of a one-inflated beta regression model in SAS via the `nlmixed` procedure.

```
%INCLUDE SAS_InflatedBeta_Macro;
/*Usage:
%Beta_Regression(Dataset,tech,details,mu_vars,phi_vars,zero_vars,
one_vars,depvar);
 Dataset  the LIBNAME.DATA file
 tech  allows for different optimization schemes to be used
 details  allows for other options to be specified
 mu_vars  variables modeling changes in mean
 phi_vars  variables modeling changes in precision
 zero_vars  variables predicting a response of zero
 one_vars  variables predicting a response of one
 depvar  the dependent variable scaled to a [0,1] interval */

%INCLUDE Study_Shou_Data2;
%Beta_Regression(Study_Shou_Data2,trureg,hess cov itdetails,Case1,Case1,'',
Case1,prob_est1);
```

Fit Statistics

-2 Log Likelihood	88.1
AIC (smaller is better)	100.1
AICC (smaller is better)	100.5
BIC (smaller is better)	120.6

Parameter Estimates

Param.	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
one0	-1.6740	0.2568	224	-6.52	<.0001	-2.1801	-1.1678	9.906E-9
one1	-1.9002	0.6393	224	-2.97	0.0033	-3.1600	-0.6405	9.979E-9
b0	0.3272	0.1240	224	2.64	0.0089	0.08285	0.5716	-15E-15
b1	-0.7423	0.1598	224	-4.64	<.0001	-1.0573	-0.4273	9.99E-16
d0	0.3423	0.1182	224	2.90	0.0041	0.1094	0.5753	2.89E-14
d1	0.6144	0.1680	224	3.66	0.0003	0.2833	0.9455	1.47E-14

3.2.3 Analysis in Stata

First, we present Stata code for the fixed-effects beta regression models, using betafit (for Stata 11 or newer) and then betareg (for Stata 14 or newer). The relevant data-file is named `Study1Shou.dta`, and we assume that it already has been opened in Stata. These models give very similar results to those obtained from R and SAS.

```
* Running the betafit models:
```

```
. betafit po_kl1 , muvar( case1 ) phivar( case1 )
. betafit po_nk1 , muvar( case1 ) phivar( case1 )
. betafit no_kl1 , muvar( case1 ) phivar( case1 )
. betafit no_nk1 , muvar( case1 ) phivar( case1 )
* Running the betareg models:
. betareg po_kl1 case1, scale(case1)
. betareg po_nk1 case1, scale(case1)
. betareg no_kl1 case1, scale(case1)
. betareg no_nk1 case1, scale(case1)
```

Finally, we demonstrate the 1-inflated beta regression example in Stata, using the user-coded `zoib` utility (Buis, 2003). The relevant data-file is named `ShouStudyLong.dta`, and we assume that it already has been opened in Stata. The results are very similar to those from R and SAS.

```
* Installing zoib
. ssc install zoib
* Running the model:
. zoib prob_est1 case1, oneinflate(case1) phivar(case1)
```

Chapter 4

Quantile Models for Bounded Variables

The sections in this Chapter provide methods and code for reproducing the analyses in Chapter 4 for the "Depressive Symptoms among Chinese University students" examples.

4.1 Depressive Symptoms among Chinese University students

Yu et al (2015) conducted a large-scale study that involved 6000 university students in China. The aim of the study was to investigate the role of family environment on the level of depression among the university students.

4.1.1 Analysis in R

The Rdata file `chap4_data.Rdata` has a dataset `yudata` that includes all variables in that study. The variables `E1` to `E21` are the BDI measure variables. The data also has named variables for the family environment traits, and various demographic variables. Because the scales of the four demographic variables are not intuitive, where the higher scores correspond to lower status, we reversed the scores.

```
> # Load the relevant packages and the data-file
> library(psych)
> library(cdfquantreg)
> library(quantreg)
> library(car)
> # Calculate the aggregated BDI scores by summing over all BDI item scores
> yudata$BDI <- rowSums(yudata[, 11:31], na.rm=TRUE)
> yudata$Familyeconomic = 4 - yudata$Familyeconomic
> yudata$Maternaleducation = 26 - yudata$Maternaleducation
```

```
> yudata$Paternaleducation = 26 - yudata$Paternaleducation
> yudata$Parentrelationship = 4 - yudata$Parentrelationship
```

4.1.1.1 Example of quantile regression

This first code chunk demonstrates quantile regression for five quantiles simultaneously.

```
> # Load the relevant packages and the data-file
> modrq <- rq(BDI ~ Cohesion + Conflict + Control + Maternaleducation +
Parentrelationship, tau=c(.1, .25, .5, .75, .9), data = yudata)
> summary(mod2)
> plot(summary(mod2), parm = c(1:6),mfrow = c(2, 3))
```

The `quantreg` package also can conduct tests to compare slopes across different quantiles. The default is a variant of the Wald test described in Koenker and Bassett (1982). It requires separate runs for the quantiles that are to be compared. The demonstration below compares slopes for the 25^{th} and 75^{th} quantiles. The first version simply tests whether any of the slopes differ, whereas the second provides a test for each of the slopes.

```
> # Now to test whether the slopes are equal for tau = c(.25, .75):
> mod25 <- rq(BDI ~ Cohesion + Conflict + Control + Maternaleducation +
+ Parentrelationship, tau= .25, data = yudata)
> mod75 <- rq(BDI ~ Cohesion + Conflict + Control + Maternaleducation +
+ Parentrelationship, tau= .75, data = yudata)
> # Test comparing tau = .25 and .75 for all slopes simultaneously:
> anova(mod25, mod75, joint = TRUE)
## Quantile Regression Analysis of Deviance Table
## Joint Test of Equality of Slopes: tau in {  0.25 0.75  }
##   Df Resid Df F value    Pr(>F)
## 1  5     9153  25.272 < 2.2e-16 ***
> # Test comparing tau = .25 and .75 for individual slopes:
> anova(mod25, mod75, joint = FALSE)
##  Quantile Regression Analysis of Deviance Table
## Tests of Equality of Distinct Slopes: tau in {  0.25 0.75  }
##                     Df Resid Df F value    Pr(>F)
## Cohesion             1     9157 14.2991 0.0001569 ***
## Conflict             1     9157  9.8665 0.0016885 **
## Control              1     9157 26.2127  3.12e-07 ***
## Maternaleducation    1     9157  8.7487 0.0031061 **
## Parentrelationship   1     9157  9.0407 0.0026476 **
```

Utilities such as these for comparing slopes in quantile regression models may differ among alternative quantile regression packages. We describe differences between these utilities in `quantreg` and in Stata's `sqreg` command in the subsection on Stata below.

4.1.1.2 Example of CDF-quantile regression

The following code illustrates the usage of the `cdfquantreg` package to apply cdfquantile regression for analyzing the depressive symptom data. We first linearly transform the BDI scores into the [0, 1] interval using the function `scaleTR` provided in the `cdfquantreg` package. Then we use the `cdfquantreg()` function to fit the data with the logit-logistic distribution. The set up of the formula of the model is similar to the ons used in the `betareg` package in the previous chapter. The left-hand of the "—" specifies the location submodel, while the right-hand side of the "—" specifies the dispersion submodel.

```
> yudata$sBDI <- scaleTR(yudata$BDI)
> mod1 <- cdfquantreg(sBDI ~ Cohesion + Conflict + Control +
Maternaleducation + Parentrelationship | Cohesion + Conflict + Control +
Maternaleducation + Parentrelationship, data = yudata, fd = "logit",
sd = "logistic")
> summary(mod1)
```

Selected output:

```
# Mu coefficients (Location submodel)
#                     Estimate Std. Error z value Pr(>|z|)
# (Intercept)        -2.540740   0.029846 -85.127  < 2e-16 ***
# Cohesion           -0.193782   0.014575 -13.295  < 2e-16 ***
# Conflict            0.114314   0.013179   8.674  < 2e-16 ***
# Control             0.183979   0.014162  12.991  < 2e-16 ***
# Maternaleducation   0.037831   0.006176   6.126 9.04e-10 ***
# Parentrelationship -0.255318   0.041739  -6.117 9.53e-10 ***
#
# Sigma coefficients (Dispersion submodel)
#                     Estimate Std. Error z value Pr(>|z|)
# (Intercept)         0.107214   0.013103   8.182 2.22e-16 ***
# Cohesion            0.131641   0.008416  15.642  < 2e-16 ***
# Conflict           -0.044753   0.007172  -6.240 4.37e-10 ***
# Control            -0.030576   0.008145  -3.754 0.000174 ***
# Maternaleducation  -0.014530   0.003439  -4.225 2.39e-05 ***
# Parentrelationship  0.093581   0.024466   3.825 0.000131 ***
# ---
# Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
#
# Converge:  successful completion
# Log-Likelihood:  5122.789

> confint(mod1)

## $location
##                            2.5%       97.5%
```

```
## (Intercept)        -2.59923727 -2.48224185
## Cohesion           -0.22234925 -0.16521494
## Conflict            0.08848441  0.14014452
## Control             0.15622238  0.21173484
## Maternaleducation   0.02572612  0.04993521
## Parentrelationship -0.33712457 -0.17351106
#
## $dispersion
##                           2.5%        97.5%
## (Intercept)         0.08153137  0.132896007
## Cohesion            0.11514694  0.148135816
## Conflict           -0.05880932 -0.030696452
## Control            -0.04653898 -0.014612727
## Maternaleducation  -0.02127115 -0.007789772
## Parentrelationship  0.04562715  0.141534031
```

To check residuals and influences:

```
> resids1 <- residuals(mod1, type = "raw")
> resids2 <- residuals(mod1, type = "pearson")
> resids3 <- residuals(mod1, type = "deviance")
> infs <- influence(mod1, plot = TRUE)
> par(mfrow = c(1, 4), mar = c(4, 2, 3, 0))
> plot(resids1, xlab = "Raw")
> plot(resids2, xlab = "Pearson's")
> plot(resids3, xlab = "Deviance")
> plot(infs, xlab = "dfBetas")
```

4.1.2 Analysis in SAS

For quantile regression, we use `quantreg` procedure in SAS, and analysis is very straightforward.

```
%INCLUDE yudata;
proc quantreg ci=sparsity data=yudata;
model BDI = Cohesion Conflict Control Maternaleducation Parentrelationship
               / quantile=.1, .25, .5, .75, .9 plot=quantplot;
run;
```

SAS `quantreg` procedure is also able to provide the heterogeneity of the slopes. The test results are the same as the R provides.

```
proc quantreg ci=sparsity data=yudata;
model BDI = Cohesion Conflict Control Maternaleducation Parentrelationship
               / quantile=.25, .75;
test Cohesion/QINTERACT;
test Conflict/QINTERACT;
```

```
test Control/QINTERACT;
test Maternaleducation/QINTERACT;
test Parentrelationship/QINTERACT;
run;
```

For CDF-quantile regression, we use the `cdfquantreg` MACRO in SAS.

```
%INCLUDE SAS_MACRO;
/*-------------------------------------------------------------------------**
 %cdfquantreg(DATA, DV, FD, SD, LMIV= , DMIV= , INIT= );
DATA: A data set including both dependent variables and the independent variables.
The dependent variable should be within (0, 1) interval.
DV:  Specify the variable name for the dependent variable.
FD,SD: The parent and child distribution of the cdfquantile family.
LMIV:  Specify the names of independent variables in the location submodel.
DMIV:  Specify the names of independent variables in the dispersion submodel.
INIT: The starting values for the parameters, such as
the starting values for the intercepts of each submodel.
E.g., the length of INIT should be 2 (1 for
the location submodel and 1 for the dispersion submodel)
for an intercept only model.
*/
%cdfquantreg(yudata, sBDI, 'logit', 'logistic',
LMIV=Cohesion Conflict Control Maternaleducation Parentrelationship,
DMIV= Cohesion Conflict Control Maternaleducation Parentrelationship,
init=0.1| 0.1 |0.1 |0.1 |0.1 |0.1|0.1 |0.1| 0.1| 0.1 |0.1 |0.1);
```

We also apply the model to the grammar example above

```
%INCLUDE grammar1;
%cdfquantreg(grammar1, ratep, 'logit', 'logistic',
LMIV=LanguageSpanish LanguageJapanese LanguageNorwegian LanguageChinese Length1,
DMIV= LanguageSpanish LanguageJapanese LanguageNorwegian LanguageChinese Length1,
init=0.1| 0.1 |0.1 |0.1 |0.1 |0.1|0.1 |0.1| 0.1| 0.1 |0.1 |0.1);
```

4.1.3 Analysis in Stata

Beginning with quantile regression, Stata has four quantile regression utilities:

- `qreg`, which performs quantile regression for a specified quantile;
- `iqreg`, which estimates regression models for differences between quantiles (e.g., the interquantile range);
- `sqreg`, which simultaneously estimates regression models for multiple quantiles using bootstrapped estimates of the variance-covariance matrix; and

- `bsqreg`, which is equivalent to `sqreg` but restricted to estimating a model for one quantile.

We will focus on illustrating `qreg` and `sqreg`, using the BDI data example in Chapter 4. The next code chunk contains the `qreg` commands for estimating the quantile regression model in this example, for the 50^{th} and 25^{th} quantiles. These give very similar results to those obtained in R and SAS, albeit with some differences in the standard errors.

```
* Estimating the 50th and 25th quantiles
qreg bdi cohesion conflict control momed parentrel, quantile(.50)
qreg bdi cohesion conflict control momed parentrel, quantile(.25)
```

The next code chunk uses `sqreg` to estimate models for three quantiles simultaneously, and then compares coefficients across quantiles in three different ways. This utility is available because the variance-covariance matrix has been estimated during the bootstrap procedure. Selected output has been included from these tests. The first one compares the 25^{th} and 75^{th} quantiles for the Control variable and then presents a confidence interval around their difference. The test after that compares two pairs of quantiles for the Control variable. The final test imitates the simultaneous slopes-comparison test shown earlier via the `quantreg` package. Note that it does not produce the same F statistic that the `quantreg` test does.

```
* Estimating the 25th, 50th, and 75th quantiles (with 100 bootstrap replications)
sqreg bdi cohesion conflict control momed parentrel, q(.25 .5 .75) reps(100)
* Testing the difference between coefficients at the 25th and 75th
* quantiles for Control:
test [q25]control = [q75]control
*  ( 1)  [q25]control - [q75]control = 0
*        F(  1,  4573) =   22.31
*             Prob > F =    0.0000
*
* Getting a 95% CI for the difference:
lincom [q75]control-[q25]control
*  ( 1)  - [q25]control + [q75]control = 0
* ------------------------------------------------------------------------------
         bdi |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
* ------------+----------------------------------------------------------------
         (1) |   .4155094     .08797     4.72   0.000     .2430456    .5879731
* ------------------------------------------------------------------------------
*
* Testing differences between the 25th vs 50th, and 50th vs 75th quantiles:
quietly test [q25]control = [q50]control
test [q50]control = [q75]control, accumulate
*  ( 1)  [q25]control - [q50]control = 0
*  ( 2)  [q50]control - [q75]control = 0
```

```
*        F(  2,  4573) =   12.19
*             Prob > F =    0.0000
*
* Testing whether any of the slopes differ
* between the 25th vs 75th quantiles:
quietly test [q25]cohesion = [q75]cohesion
quietly test [q25]conflict = [q75]conflict, accumulate
quietly test [q25]control = [q75]control, accumulate
quietly test [q25]momed = [q75]momed, accumulate
test [q25]parentrel = [q75]parentrel, accumulate
* ( 1)  [q25]cohesion - [q75]cohesion = 0
* ( 2)  [q25]conflict - [q75]conflict = 0
* ( 3)  [q25]control - [q75]control = 0
* ( 4)  [q25]momed - [q75]momed = 0
* ( 5)  [q25]parentrel - [q75]parentrel = 0
*        F(  5,  4573) =   16.43
*             Prob > F =    0.0000
```

There are several differences between the utilities for comparing slopes for `quantreg` in R and `sqreg` in Stata.

- The `sqreg` command in Stata requires the use of bootstrap estimates, whereas these cannot be used by the `anova` command for `quantreg` in R. Therefore, the standard errors on which their respective tests are based differ.

- For individual slopes tests, Statas `sqreg` uses $F(1, N - v)$, where N is the sample size and v is the number of parameters in the model, whereas `quantreg` uses $F(1, 2N - 1)$. The `sqreg` test standard errors are based on the bootstrapped variance-covariance matrix for the parameters in the model. The `quantreg` test is a variant of the Wald test described in Koenker and Bassett (1982), originally developed for comparing nested quantile regression models.

- Statas `sqreg` command allows individual slopes or subsets of slopes to be tested across multiple quantiles, whereas `quantreg` only can do this for all slopes included in a model.

- Statas `sqreg` has a utility for constructing confidence intervals around slope differences between pairs of quantiles (e.g., for the interquartile range), whereas this does not seem straightforward in `quantreg`.

Finally, Stata can produce graphs of quantiles with confidence intervals like those shown in Figure 4.3 in Chapter 4. The next code chunk uses `sqreg` to estimate the quantiles and then a user-coded utility `grqreg` (Azevedo, 2011) to create an example of such a graph, for the variable Control (after installing `grqreg`). This graph also includes the linear regression coefficient and confidence interval. Note that to keep the code within the margins of the page, we

have had to break a command line. The line-break must be eliminated if this code is used in Stata, because Stata interprets line-breaks as separating one command from another.

```
ssc install grqreg
* Be sure to eliminate the line-break in the next command.
quietly sqreg bdi cohesion conflict control momed parentrel,
 q(.1 .25 .5 .75 .9) reps(100)
grqreg control, ci ols olsci
```

Turning now to CDF-quantile regression, we have provided `.ado` files for the 36 distributions included in the `cdfquantreg` and SAS macro, except for the Burr7-Burr8 distribution. These `.ado` files utilize Stata's `ml` command to fit user-specified models. These files are called by the `cdfquantreg.ado` program. Users may consult the help file `cdfquantreg.sthlp`. The grammaticality data are used here to illustrate how the Stata cdfquantreg package works (the data-file is `grammar1.dta`).

```
* Run the model:
. cdfquantreg ratep ib2.lang c.length1, cdf(logit) quantile(logistic)
zvarlist(ib2.lang c.length1)
                                                Number of obs   =        247
                                                Wald chi2(5)    =     282.46
Log likelihood =  108.24713                     Prob > chi2     =     0.0000
------------------------------------------------------------------------------
       ratep |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
        lang |
  "Chinese"  |  -2.326627    .206805   -11.25   0.000    -2.731957   -1.921296
 "Japanese"  |  -2.860137   .1916156   -14.93   0.000    -3.235696   -2.484577
"Norwegian"  |  -1.082622    .207365    -5.22   0.000     -1.48905    -.676194
  "Spanish"  |  -.9586367   .2357527    -4.07   0.000    -1.420703   -.4965699
             |
     length1 |  -.0423416   .0136622    -3.10   0.002     -.069119   -.0155642
       _cons |   2.261381   .1244935    18.16   0.000     2.017378    2.505384
-------------+----------------------------------------------------------------
eq2          |
        lang |
  "Chinese"  |   .3075083    .168619     1.82   0.068    -.0229789    .6379954
 "Japanese"  |   .2906412   .1625881     1.79   0.074    -.0280256    .6093079
"Norwegian"  |   .2662696    .169619     1.57   0.116    -.0661776    .5987168
  "Spanish"  |   .4714093   .1771736     2.66   0.008     .1241554    .8186632
             |
     length1 |  -.0287045   .0107071    -2.68   0.007      -.04969    -.007719
       _cons |  -.7068469   .1208912    -5.85   0.000    -.9437894   -.4699045
------------------------------------------------------------------------------
```

Post-estimate calculations are available, including the gradient and vce. Moreover, estimates can be stored in the usual fashion. To obtain fitted values and residuals, however, requires an additional file, `cdfquantreg_p.ado`, provided for this purpose. It operates under the `predict` command after the model has been fitted, and it generates the empirical quantiles, fitted values for μ, $\log(\sigma)$, and the quantiles, as well as raw residuals. The next code chunk illustrates its use in our example (without the output). More details are available in the `cdfquantreg_p.sthlp` file.

```
* Run the .ado file that produces fitted values and residuals:
. predict newvar, qtile
* Examine a scatterplot of the dependent variable and fitted values:
. twoway (scatter fitted ratep)
* Examine a scatterplot of the dependent variable and raw residuals:
. twoway (scatter residuals ratep)
. drop newvar xd fitted residuals
* Get fitted values for the median:
. predict newvar, qtile pctle(0.5)
* Plot the estimated medians by sentence length
. twoway (scatter fitted length1)
```

The `margins` facility also is available, and the `cdfquantreg_m.ado` file converts marginal effects for the location and dispersion parameters into effects on a quantile selected by the user. The estimated marginal quantiles in Table 4.4 can be obtained using this utility. This next code chunk illustrates its application to our example, including output. Note that equation(#1) estimates the location parameter μ and equation(#2) estimates the dispersion parameter $\log(\sigma)$. More details are available in the `cdfquantreg_m.sthlp` file.

```
* Run the program to estimate the marginal effects and convert them
* to effects on the 75th percentile:
. cdfquantreg_m lang, pctle(0.75)

Predictive margins                                 Number of obs     =        247
Model VCE    : OIM

Expression   : Linear prediction, predict(equation(#1))

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        lang |
  "Chinese"  |  -.0652456   .1663076    -0.39   0.695    -.3912025    .2607113
  "English"  |   2.261381   .1244935    18.16   0.000     2.017378    2.505384
 "Japanese"  |  -.5987553   .1476184    -4.06   0.000    -.8880821   -.3094284
```

```
"Norwegian"  |   1.178759   .1662584     7.09   0.000     .8528989     1.50462
  "Spanish"  |   1.302745   .2014984     6.47   0.000     .9078149    1.697674
------------------------------------------------------------------------------
(results modresults are active now)

Predictive margins                              Number of obs     =        247
Model VCE    : OIM

Expression   : Linear prediction, predict(equation(#2))

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        lang |
  "Chinese"  |  -.3993387   .1166587    -3.42   0.001    -.6279855   -.1706918
  "English"  |  -.7068469   .1208912    -5.85   0.000    -.9437894   -.4699045
 "Japanese"  |  -.4162057   .1094037    -3.80   0.000     -.630633   -.2017785
"Norwegian"  |  -.4405773   .1181501    -3.73   0.000    -.6721473   -.2090073
  "Spanish"  |  -.2354376   .1282544    -1.84   0.066    -.4868117    .0159364
------------------------------------------------------------------------------
(results modresults are active now)

lang
.75 quantile  factor level
--------------------------
.66187552     1bn.lang
.94284922     2.lang
.53141569     3.lang
.86828703     4.lang
.89760529     5.lang
```

Finally, the first command below executes the logit-logistic model for the BDI data, with output that closely agrees with the results displayed in Table 4.3. The second obtains the parameter correlation matrix.

```
* Remember to eliminate the line-break in this command before using it in Stata.
. cdfquantreg trbdi cohesion conflict control momed parentrel, cdf(logit)
 quantile(logistic) zvarlist(cohesion conflict control momed parentrel)
. estat vce, corr
```

Chapter 5

Censored and Truncated Variables

5.4 Tobit Model Example

This section presents the code for estimating the Tobit model for the doubly-censored gun-ownership attitudes data, as a function of political orientation. It compares this model with the corresponding OLS regression model.

5.4.1 Analysis in R

The dependent variable in the data-file for these analyses is denoted by $SECS\_6$, as a result of it being the sixth item in Everett's (2013) socioeconomic conservatism inventory. This first code chunk loads the required packages, reads and formats the data, and then isolates the subset of the data needed for the subsequent analyses. The graph produces the plot in Figure 5.2.

```
library(AER)
library(MASS)
library(lmtest)
library(car)
conserv <- read.csv("Extreme_Events_Pilot3_covariates.csv",  header=TRUE)
attach(conserv)
political <- NA
political[PoliOrien == "Democrat"] = "Democrat"
political[PoliOrien == "Independent"] = "Independent"
political[PoliOrien == "Republican"] = "Republican"
political[PoliOrien == "No preference"] = "NoPref"
#
# This produces the histogram of the dependent variable:
hist(SECS_6, breaks = 100, main = "", xlab = "gun ownership")
#
# Next, we form the required subset of the data.
```

```
comdat <- cbind.data.frame(political,SECS_6,SECS_E)
comdat <- na.omit(comdat)
```

The OLS and Tobit models are estimated and compared in the code chunk below, which produces the output displayed in Table 5.1. Lower and upper censoring rates are estimated for the four political orientation categories, and the Q-Q plot in Figure 5.3 is reproduced here.

```
> # This is the linear regression model:
> lsmod2 <- lm(comdat$SECS_6 ~ comdat$political);
> summary(lsmod2)
> # This is the Tobit model:
> gmod2 <- tobit(comdat$SECS_6 ~ comdat$political, left = 0, right = 100,dist = "gaussian");
> summary(gmod2)
> # Here are the log-likelihoods for these models:
> logLik(lsmod2); logLik(gmod2);
> #
> # Estimating censoring rates in gmod2 for political groups:
> # Democrats
> 1-pnorm((coef(gmod2)[1])/gmod2$scale)  # lower censoring
> pnorm((coef(gmod2)[1]-100)/gmod2$scale)  # upper censoring
> # Independents
> 1-pnorm((coef(gmod2)[1] + coef(gmod2)[2])/gmod2$scale)  #  lower censoring
> pnorm((coef(gmod2)[1] + coef(gmod2)[2] - 100)/gmod2$scale)  # upper censoring
> # No Preference
> 1-pnorm((coef(gmod2)[1] + coef(gmod2)[3])/gmod2$scale)  # lower censoring
> pnorm((coef(gmod2)[1] + coef(gmod2)[3] - 100)/gmod2$scale)  # upper censoring
> # Republicans
> 1-pnorm((coef(gmod2)[1] + coef(gmod2)[4])/gmod2$scale)  # lower censoring
> pnorm((coef(gmod2)[1] + coef(gmod2)[4] - 100)/gmod2$scale)  # upper censoring
> #
> # This gets an augmented version of the Q-Q plot in Figure 5.3:
> qqPlot(residuals(gmod2), xlab = "Gaussian quantiles", ylab = "Gaussian model residuals")
```

5.4.2 Analysis in SAS

The data file is named `EEPilot3NoMissSAS.txt`. We assume that this file has been loaded. The dependent variable in the data-file for these analyses is denoted by *gunown*. The code chunk below estimates the OLS regression and Tobit models with political orientation as the predictor, and produces the output displayed in Table 5.1. Note that 4 = Democrat, 3 = Independent, 2 = No preference, 1 = Republican. The OLS model is run using the `glm` procedure rather than `reg` because it handles categorical variables adequately. The Tobit model is run using the `qlim` procedure. An alternative is the `lifereg` procedure, and this is employed later in this chapter.

```
# This is the linear regression model:
```

```
ods graphics on;
proc glm data = guns;
   class political;
   model gunown = political / solution;
run;
ods graphics off;
# This is the Tobit model:
proc qlim data=guns;
   class political;
   model gunown = political;
   endogenous gunown ~ censored(lb=0 ub=100);
run;
```

5.4.3 Analysis in Stata

First, we assume that the data are loaded (the filename is `Pilot3NoMiss.dta`). We need to generate a version of the political groups variable that is reverse-coded from the SAS version:

```
generate politrev = 5 - political
```

The dependent variable in the data-file for these analyses is denoted by *gunown*. The code chunk below estimates the OLS regression and Tobit models with *politrev* as the predictor, and produces the output displayed in Table 5.1. Note that 1 = Democrat, 2 = Independent, 3 = No preference, 4 = Republican. The OLS model is run using the `regress` command, whereas the Tobit model is run using the `qlim` procedure.

```
* Linear regression model:
regress gunown i.politrev
* Tobit model:
tobit gunown i.politrev, ll(0) ul (100)
* qqplot of the residuals:
predict double xb if e(sample), xb
gen double residual = gunown - xb
qnorm residual
```

5.5 Heteroskedastic and Non-Gaussian Tobit Models

The heteroskedastic and non-Gaussian Tobit models extend the analysis of the gun ownership attitude data. The code presented here estimates these models and provides additional details to what is presented in Chapter 5.

5.5.1 Analysis in R

As described in Chapter 5, the heteroskedastic models include a submodel of dispersion, with predictors that may or may not be the same as those in

the location submodel. The `crch` library is used here for estimating the heteroskedastic and non-Gaussian Tobit models. Unusually, it also permits fitting t-distributed models with the degrees of freedom as an estimated parameter.

As before, the dependent variable is gun ownership attitude, denoted by $SECS\_6$. Political group and economic conservatism ($SECS\_E$) are the candidate predictors. The first models presented are those that test the mediation hypothesis, namely a model with political group as the predictor, one with political group and economic conservatism as predictors, and one with just economic conservatism.

```
> # Load the required library:
> library(crch)
> # Heteroskedastic model with only political group:
> crmod2a <- crch(comdat$SECS_6 ~ comdat$political|comdat$political, link.scale = "log",
+ dist = "student", left = 0, right = 100, truncated = FALSE)
> summary(crmod2a)
> # Heteroskedastic model with only SECS_E
> crmod4 <- crch(comdat$SECS_6 ~ comdat$SECS_E|comdat$SECS_E, link.scale = "log",
+ dist = "logistic", left = 0, right = 100, truncated = FALSE)
> summary(crmod4)
> # Heteroskedastic model with SECS_E and political group:
> crmod5 <- crch(comdat$SECS_6 ~ comdat$SECS_E + comdat$political|comdat$SECS_E +
+ comdat$political, link.scale = "log", dist = "logistic", left = 0,
+ right = 100, truncated = FALSE)
> summary(crmod5)
> # Likelihood-ratio test comparing these models:
> lrtest(crmod4, crmod5)
```

The next group of models is for comparisons with the heteroskedastic logistic $SECS\_E$-only model (crmod4). The first model is a t-distributed heteroskedastic model that returns an estimated $df = 3.796$ and very similar parameter estimates to its logistic counterpart. The second model compares a homoskedastic logistic model with the heteroskedastic logistic model. The third model compares a heteroskedastic Gaussian against the heteroskedastic logistic. The Q-Q plot produces the graph in Figure 5.5.

```
> # A t-distributed model with df as a free parameter:
> crmod3 <- crch(comdat$SECS_6 ~ comdat$SECS_E|comdat$SECS_E, link.scale = "log",
+ dist = "student", left = 0, right = 100, truncated = FALSE)
> summary(crmod3)
> # Test whether we need heteroskedasticity:
> crmod4a <- crch(comdat$SECS_6 ~ comdat$SECS_E|1, link.scale = "log",
+ dist = "logistic", left = 0, right = 100, truncated = FALSE)
> lrtest(crmod4a, crmod4)
> # Compare the AIC values of these two models as well:
> c(AIC(crmod4a),AIC(crmod4))
> # See whether we need a logistic rather than Gaussian:
```

```
> crmod4b <- crch(comdat$SECS_6 ~ comdat$SECS_E|comdat$SECS_E, link.scale = "log",
+ dist = "gaussian", left = 0, right = 100, truncated = FALSE)
> AIC(crmod4b)
> # Q-Q plot for the heteroskedastic logistic model:
> qqPlot(residuals(crmod4), distribution = "logis", xlab = "logistic quantiles",
+ ylab = "logistic model residuals")
```

5.5.2 Analysis in SAS

The SAS system can run several of the models in this section of Chapter 5, but the heteroskedastic ones require SAS users to "roll their own". We will do this later via the `nlmixed` procedure. First, however, we may estimate the homoskedastic models in the `lifereg` procedure. As before, we assume that the data-file named `EEPilot3NoMissSAS.txt` has been loaded.

The next code chunk estimates the Gaussian and logistic models with $SECS\_E$ as the predictor. The *gunupper* and *gunlower* variables identify the lower and upper censored cases. If these variables have the same values, then they are treated as the actual response value. If the lower value is missing, the upper value is used as a left-censored value. If the upper value is missing, the lower value is used as a right-censored value.

```
* This is the Gaussian model;
proc lifereg data=guns;
   model (gunupper,gunlower) = SECS_E / distribution = normal;
run;
* This is the logistic model;
proc lifereg data=guns;
   model (gunupper,gunlower) = SECS_E / distribution = logistic;
run;
```

As mentioned above, the heteroskedastic models must be estimated by submitting their log-likelihood functions to an appropriate optimizer, and this can be done in the `nlmixed` procedure. The first model in the code chunk below verifies that this works by replicating the homoskedastic Gaussian model estimated via the `lifereg` procedure in the preceding code chunk. The second model estimates a heteroskedastic Gaussian model by including $SECS\_E$ in the dispersion submodel. The location and dispersion submodels are specified in these lines:

```
xb = b0 + b1*SECS_E;
mu = xb;
wd = d0 + d1*SECS_E;
sigma = exp(wd);
```

Thus, the location submodel includes an intercept term, $b0$, and a term for the predictor, $b1 * SECS\_E$. Likewise, the dispersion submodel has an intercept, $d0$, a term for the predictor, $d1 * SECS\_E$, and a log link function as specified in

the fourth line. Starting values for the parameters must be supplied, and these are provided in the `parms` statement. The output for these models includes the Hessian and parameter-estimate correlation matrices, in addition to the fit statistics and parameter estimates statistics.

```
* This replicates the preceding homoskedastic Gaussian model;
proc nlmixed data = guns tech = trureg hess corr itdetails;
parms b0 = -55, b1 = 1.9, d0 = 2.7;
title 'homo tobit gaussian';
*The model equations here are substituted into the likelihood below;
y = gunown;
xb = b0 + b1*SECS_E;
mu = xb;
wd = d0;
sigma = exp(wd);
* This next command is the log-likelihood;
ll = t1*t2*log(pdf('normal',y, mu,sigma)) +
(1-t1)*log(1 - cdf('normal',y, mu,sigma)) +
(1-t2)*log(cdf('normal',y, mu,sigma));
model y ~ general(ll);
;
run;
* This is the heteroskedastic Gaussian model;
proc nlmixed data = guns tech = trureg hess corr itdetails;
parms b0 = -52, b1 = 1.8, d0 = 2.7, d1 = 0.1;
title 'hetero tobit gaussian';
*The model equations are constructed here and substituted into the likelihood below;
y = gunown;
xb = b0 + b1*SECS_E;
mu = xb;
wd = d0 + d1*SECS_E;
sigma = exp(wd);
*This next command is the log-likelihood;
ll = t1*t2*log(pdf('normal',y, mu,sigma)) + (1-t1)*log(1 - cdf('normal',y, mu,sigma))
+ (1-t2)*log(cdf('normal',y, mu,sigma));
model y ~ general(ll);
;
run;
```

The homoscedastic and heteroscedastic logistic models are estimated with the code in the next chunk. We do not present the models that include political group along with $SECS\_E$ to test mediation, but setting them up is a straightforward exercise. Dummy variables have to be created to represent the political groups, and these are then included in the location and dispersion submodels.

```
* This replicates the homoskedastic logistic model;
```

```
proc nlmixed data = guns tech = trureg hess corr itdetails;
parms b0 = -55, b1 = 1.9, d0 = 2.7;
title 'homo tobit logistic';
*The model equations are constructed here and substituted into the
likelihood below;
y = gunown;
xb = b0 + b1*SECS_E;
mu = xb;
wd = d0;
sigma = exp(wd);
* This next command is the log-likelihood;
ll = t1*t2*log(pdf('logistic',y, mu,sigma))
+ (1-t1)*log(1 - cdf('logistic',y, mu,sigma))
+ (1-t2)*log(cdf('logistic',y, mu,sigma));
model y ~ general(ll);
;
run;
* This estimates the heteroskedastic logistic model;
proc nlmixed data = guns tech = trureg hess corr itdetails;
parms b0 = -55, b1 = 1.9, d0 = 2.7, d1 = 0.1;
title 'hetero tobit logistic';
* The model equations are constructed here and substituted into the
likelihood below;
y = gunown;
xb = b0 + b1*SECS_E;
mu = xb;
wd = d0 + d1*SECS_E;
sigma = exp(wd);
*This next command is the log-likelihood;
ll = t1*t2*log(pdf('logistic',y, mu,sigma))
+ (1-t1)*log(1 - cdf('logistic',y, mu,sigma))
+ (1-t2)*log(cdf('logistic',y, mu,sigma));
model y ~ general(ll);
;
run;
```

5.5.3 Analysis in Stata

Stata can run several of the models in this section of Chapter 5, but the heteroskedastic ones require Stata users to "roll their own". We will do this later via the `ml` function. First, however, we may estimate the homoskedastic Gaussian Tobit model in the `tobit` procedure. As before, we assume that the datafile, named `Pilot3NoMiss.dta`, has been loaded. First, we need to generate a version of the political groups variable that is reverse-coded from the SAS version because the default reference category in Stata is the opposite to that in SAS. The first line in the next code chunk does this, and then uses it as the

predictor in the linear and Tobit regression models.

```
. generate politrev = 5 - political
* linear regression model:
. regress gunown i.politrev
* Tobit model
. tobit gunown i.politrev, ll(0) ul (100)
```

Next, we estimate the homoskedastic and heteroskedastic Gaussian Tobit models with $SECS\_E$ as a predictor. The log-likelihood program htobit in the chunk below receives a dependent variable y, threshold values $q1$ and $q2$ distinguishing censored from uncensored observations, a vector of predictors Xb for the location submodel, and a vector of predictors Xd for the dispersion submodel.

```
* This is the log-likelihood function:
capture program drop htobit
program define htobit
args lnf Xb Xd
tempvar phi mu
quietly {
gen double ‘phi’ = exp(‘Xd’)
gen double ‘mu’ = ‘Xb’
replace ‘lnf’ = ln(normalden($ML_y,‘mu’,‘phi’)) if $ML_y < 100 & y > 0
replace ‘lnf’ = ln(1 - normal(($ML_y-‘mu’)/‘phi’)) if $ML_y == 100
replace ‘lnf’ = ln(normal(($ML_y-‘mu’)/‘phi’)) if $ML_y == 0
  }
end
```

The next code-chunk runs the two Gaussian Tobit models. The `ml search` command searches for appropriate parameter estimate starting values, and the `ml max` command uses these to initiate the maximum likelihood estimation process. The model results closely match their counterparts from SAS and R.

```
. generate y = gunown
* Homoskedastic model
. ml model lf htobit (muvar: gunown = secs_e) (phivar: )
. ml search
. ml max
* Heteroskedastic model
. ml model lf htobit (muvar: gunown = secs_e) (phivar: secs_e)
. ml search
. ml max
```

The homoscedastic and heteroscedastic logistic models are estimated with the program code in the next chunk. One version is provided for Stata 14 and newer, and another for Stata 13 and older. The difference between them is that the newer versions of Stata have pdf and cdf functions for the logistic

distribution, whereas the older ones do not (and so the older code must provide these functions explicitly).

```
* This is the log-likelihood function for Stata >=14:
capture program drop hltobit
program define hltobit
args lnf Xb Xd
tempvar phi mu
quietly {
gen double ‘phi’ = exp(‘Xd’)
gen double ‘mu’ = ‘Xb’
* Remember to eliminate the line-break in this command before using it in Stata.
replace ‘lnf’ = ln(exp(-($ML_y-‘mu’)/‘phi’)/
(‘phi’*(1 + exp(-($ML_y-‘mu’)/‘phi’))^2)) if $ML_y < 100 & y > 0
replace ‘lnf’ = ln(1 - logistic(($ML_y-‘mu’)/‘phi’)) if $ML_y == 100
replace ‘lnf’ = ln(logistic(($ML_y-‘mu’)/‘phi’)) if $ML_y == 0
  }
end
* This is the log-likelihood function for Stata <=13:
capture program drop hltobit
program define hltobit
args lnf Xb Xd
tempvar phi mu
quietly {
gen double ‘phi’ = exp(‘Xd’)
gen double ‘mu’ = ‘Xb’
* Remember to eliminate the line-break in this command before using it in Stata.
replace ‘lnf’ = ln(exp(-($ML_y-‘mu’)/‘phi’)/
(‘phi’*(1 + exp(-($ML_y-‘mu’)/‘phi’))^2)) if $ML_y < 100 & y > 0
replace ‘lnf’ = ln(1  1/(1+exp(-($ML_y-‘mu’)/‘phi’))) if $ML_y == 100
replace ‘lnf’ = ln(1/(1+exp(-($ML_y-‘mu’)/‘phi’))) if $ML_y == 0
  }
end
```

The next code-chunk runs the two logistic Tobit models. The results closely match those obtained in R and SAS.

```
* Homoskedastic model
ml model lf hltobit (muvar: gunown = secs_e) (phivar: )
ml search
ml max
* Heteroskedastic model
ml model lf hltobit (muvar: gunown = secs_e) (phivar: secs_e)
ml search
ml max
```

The models presented in the next code chunk are those that test the mediation hypothesis, namely a model with political group as the predictor, and one

with political group and economic conservatism as predictors. The models with just economic conservatism were estimated in the preceding chunk, but are re-estimated here for likelihood ratio tests. The last likelihood-ratio test is for determining whether we need a heteroskedastic model.

```
ml model lf hltobit (muvar: gunown = i.politrev) (phivar: i.politrev)
ml search
ml max
estimates store A
ml model lf hltobit (muvar: gunown = i.politrev secs_e) (phivar: i.politrev secs_e)
ml search
ml max
estimates store B
* Likelihood-ratio test comparing the first two models
lrtest A B
ml model lf hltobit (muvar: gunown = secs_e) (phivar: secs_e)
ml search
ml max
estimates store C
* Likelihood-ratio test comparing models C and B
* This is the mediation test
lrtest C B
ml model lf hltobit (muvar: gunown = secs_e) (phivar: )
ml search
ml max
estimates store D
// Likelihood-ratio test comparing models D and C
// This is the heteroskedasticity test
lrtest D C
```

Chapter 6

Extensions and Conclusions

6.2 Absolute Bounds and Censoring

6.2.1 Analysis in R

The `cdfquantreg` package can estimate models using distributions from the CDF-Quantile family with censoring, whereas a censored beta distribution model requires coding a special function to estimate it. The datafile is named `events3.csv`, and it requires some processing before the censored beta model function can be deployed. The next code chunk loads the required packages, processes the data, and provides the censored beta model function.

```
> # Load the libraries
> library(cdfquantreg)
> library(betareg)
> library(lmtest)
> library(MASS)
> #
> #read and process data
> events3 <- read.csv("events3.csv",  header=TRUE)
> attach(events3)
> #
> # Create the censoring variables:
> q1 <- c(rep(0,length(EQ1P))); q2 <- c(rep(0,length(EQ1P)))
> for (i in 1: length(EQ1P)) {ifelse(EQ1P[i]==1e-05, q1[i] <- 1, q1[i] <- 0)}
> for (i in 1: length(EQ1P)) {ifelse(EQ1P[i]==0.1, q2[i] <- 1, q2[i] <- 0)}
> #
> q3 <- c(rep(0,length(EQ3P))); q4 <- c(rep(0,length(EQ3P)))
> for (i in 1: length(EQ3P)) {ifelse(EQ3P[i]==1e-05, q3[i] <- 1, q3[i] <- 0)}
> for (i in 1: length(EQ3P)) {ifelse(EQ3P[i]==0.1, q4[i] <- 1, q4[i] <- 0)}
> #
> t1hat <- 0.00001818; t2hat <- 0.1;
> #
```

```
> # Beta regression function
> betaint <- function(h, y, t1, t2, q1, q2, x, z)
{
   hx = x%*%h[1: length(x[1,])]
   mu = exp(hx)/(1+exp(hx))
   gz = z%*%h[length(x[1,])+1: length(z[1,])]
   phi = exp(gz)
   loglik = log(q1*(pbeta(t1,mu*phi,(1-mu)*phi)) + q2*(1-pbeta(t2,mu*phi,
   (1-mu)*phi)) + pmin(1-q1,1-q2)*(dbeta(y,mu*phi,(1-mu)*phi)))
   -sum(loglik, na.rm = TRUE)
}
```

The following code chunk provides the graphs in Figure 6.1.

```
> par(mfrow = c(1,3))
> truehist(EQ1P[repub==1], nbins = 5000, ylim = c(0,12000))
> truehist(EQ1P[dem==1], nbins = 5000, ylim = c(0,12000))
> truehist(EQ1P[indep==1], nbins = 5000, ylim = c(0,12000))
```

The next code chunk estimates the censored beta regression model.

```
> # Set up the data for the model:
> ydata <- cbind(EQ1P);
> const <- rep(1,length(ydata));
> xdata <- cbind(const, dem, indep, nopref, SECSS);
> zdata <- cbind(const, dem, indep, nopref);
> q1data <- cbind(q1);
> q2data <- cbind(q2);
> # The starting values are based on earlier models not shown here.
> start <- c(-4.4, -0.26, -0.85, 0.01, 0.0, 2.8, 0.7, 1.2, 0.3)
> # Estimate the model first using the Nelder-Mead optimizer:
> musigmaopt90c <- optim(start, betaint, hessian = T, y = ydata,  t1 = t1hat,
+ t2 = t2hat, q1 = q1data, q2 = q2data, x = xdata, z = zdata,
+ method = "Nelder-Mead")
> # Then re-estimate the model using the previous output as starting-values:
> start <- musigmaopt90c$par
> musigmaopt9c <- optim(start,betaint, hessian = T, y = ydata,  t1 = t1hat,
+ t2 = t2hat, q1 = q1data, q2 = q2data, x = xdata, z = zdata,
+ method = "BFGS")
> # Get the estimates, standard errors, z-statistics, and p-values:
> outopt9c <- rbind(estim <- musigmaopt9c$par,
+ serr <- sqrt(diag(solve(musigmaopt9c$hessian))),
+ zstat <- estim/serr, prob <- 1-pnorm(abs(zstat)));
+ row.names(outopt9c) <- c("estim", "serr", "zstat", "prob");
+ colnames(outopt9c) <- c("b0","b1","b2","b3","b4","d0","d1","d2","d3");
+ outopt9c
> # Get the log-likelihood for the model and its convergence status:
```

```
> c(musigmaopt9c$value, musigmaopt9c$convergence)

> # Get the confidence intervals
> fac <- qnorm(c(0.025, 0.975))
> outopt9c[1, ] + outopt9c[2, ] %o% fac
```

The uncensored beta regression model can be estimated using the betareg package. The following code chunk and output presents the details of this model.

```
library(betareg)
# Beta regression model
> betamod <- betareg(EQ1P ~ dem + indep + nopref +
+ SECSS|dem + indep + nopref, data = events3); summary(betamod)
## Coefficients (mean model with logit link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.976148   0.328371 -12.109  < 2e-16 ***
## dem         -0.479167   0.299581  -1.599  0.10972
## indep       -0.821430   0.290909  -2.824  0.00475 **
## nopref      -0.233722   0.369027  -0.633  0.52651
## SECSS       -0.007384   0.003047  -2.423  0.01539 *
##
## Phi coefficients (precision model with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.2201     0.2637  12.213  < 2e-16 ***
## dem           0.6380     0.3397   1.878  0.06037 .
## indep         0.9819     0.3316   2.961  0.00306 **
## nopref        0.2168     0.4205   0.515  0.60624
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood:  1392 on 9 Df
```

The following chunk estimates the logit-logistic model using the censored-regression model in the `cdfquantreg` package.

```
> # We have to get rid of missing values before
> # cdf-quantile models can run.
> pdat <- as.data.frame(na.omit(cbind(EQ1P,EQ3P,formatno,orderno,
+ SECSS, SECSE,dem,repub,indep,nopref,q1,q2,q3,q4)))
> detach(events3)
> attach(pdat)
> # Estimate the Chapter 6 logit-logistic model:
> logitlogmod <- cdfquantregC(EQ1P ~ SECSS | SECSS,
+ fd ='logit',sd ='logistic', c1 = 0.00001818, c2 = 0.1, data = pdat)
> summary(logitlogmod)
> #Confidence intervals
> coefs = do.call(rbind, logmod1$coefficients)
```

```
> coefs[, 1] + coefs[, 2] %o% qnorm(c(0.025, 0.975))
```

The next code chunk estimates the uncensored logit-logistic model.

```
# logit-logistic model
> logitmod <- cdfquantreg(EQ1P ~ SECSS|SECSS, fd = 'logit',
+ sd = 'logistic', data = events3); summary(logitmod)
## Mu coefficients (Location submodel)
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.394919   0.477781 -11.292  < 2e-16 ***
## SECSS       -0.024657   0.007514  -3.281  0.00103 **
##
## Sigma coefficients (Dispersion submodel)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.054910   0.156285   0.351   0.7253
## SECSS       0.004901   0.002306   2.126   0.0335 *
##
## Converge:  successful completion
## Log-Likelihood:  1407.204
```

6.2.2 Analysis in SAS

The datafile is named `events3SAS`, and it requires some processing before the censored beta model function can be deployed. We need variables, q1 and q2 to identify the censored cases, t1 and t2 to specify the censoring thresholds, and dummy-variables for the political categories. Note that 4 = Democrat, 3 = Independent, 2 = No preference, 1 = Republican.

```
data subevents3;
   set events3;
   q1 = 0;
   q2 = 0;
   dem = 0; indep = 0; nopref = 0;
   if political = 4 then dem = 1;
   if political = 3 then indep = 1;
   if political = 2 then nopref = 1;
   if EQ1P < 0.0000101 then q1 = 1;
   if EQ1P > 0.09999 then q2 = 1;
   t1 = 0.00001818;
   t2 = 0.1;
run;
```

We must create the models "from scratch" via procedures such as NLMIXED. Fitting the censored beta regression model in SAS has proved tricky. The model in the next code chunk required the optimizer QUANEW to converge properly. The resulting model log-likelihood is the same as the R and Stata models, and the parameter estimates are quite similar to those models. However, the Hessian has a negative eigenvalue and the absence of standard error estimates for

two parameters is worrying. Other optimizers and starting-values dont do any better than this.

```
proc nlmixed data = subevents3 tech = quanew hess corr itdetails;
parms b0 = -3.6, b1 = -0.6, b2 = -1.0, b3 = -0.3, b4 = -0.01,
 d0 = 2.8, d1 = 0.9, d2 = 1.1, d3 = 0.3;
title 'censored beta';
*The model equations are constructed here and substituted
* into the likelihood below;
y = EQ1P;
xb = b0 + b1*dem + b2*indep + b3*nopref + b4*SECSS;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1*dem + d2*indep + d3*nopref;
phi = exp(wd);
w = mu*phi;
t = (1-mu)*phi;
*This next command is the log-likelihood;
ll = (1-q1)*(1-q2)*log(pdf('beta',y, w,t)) + q2*log(1 -
cdf('beta',t2, w,t)) + q1*log(cdf('beta',t1, w,t));
model y ~ general(ll);
;
run;
```

The uncensored beta regression model, on the other hand, is straightforward to estimate and its parameter-estimate correlation matrix doesn't throw up any problematic values. The following model produces results very close to those obtained in R,

```
proc nlmixed data = subevents3 tech = trureg hess corr itdetails;
title 'Uncensored beta regression';
parms b0 = -3.0, b1 = -0.6, b2 = -1.0, b3 = -0.3, b4 = -0.01,
d0 = 3.0, d1 = 1.0, d2 = 1.0, d3 = 0.3;
y = EQ1P;
xb = b0 + b1*dem + b2*indep + b3*nopref + b4*SECSS;
mu = exp(xb)/(1 + exp(xb));
wd = d0 + d1*dem + d2*indep + d3*nopref;
phi = exp(wd);
w = mu*phi;
p = mu*phi;
q = phi - mu*phi;
ll = lgamma(p+q) - lgamma(p) - lgamma(q) + (p-1)*log(y) + (q-1)*log(1 - y);
model y ~ general(ll);
run;
```

The next code chunk estimates the censored logit-logistic regression model. This model converges to values very similar to the Stata estimates, and fairly close to those from R. The log-likelihood is nearly identical to the Stata result.

```
proc nlmixed data = subevents3 tech = trureg hess corr itdetails;
parms b0 = -5.3, b1 = -0.03, d0 = 0.3, d1 = 0.01;
title 'censored logit-logistic';
* The model equations are constructed here and substituted
* into the likelihood below;
y = EQ1P;
xb = b0 + b1*SECSS;
mu = xb;
wd = d0 + d1*SECSS;
sigma = exp(wd);
*This next command is the log-likelihood;
ll = (1-q1)*(1-q2)*log((exp(mu/sigma)*(-1 + 1/y)**(-1 +
1/sigma))/((y + exp(mu/sigma)*(-1 + 1/y)**(1/sigma)* y)**2*sigma))
+ q2*log(1 -1/(1 + exp(-(-log(-1 + 1/y) - mu)/sigma)))
+ q1*log(1/(1 + exp(-(-log(-1 + 1/y) - mu)/sigma)));
model y ~ general(ll);
;
run;
```

The uncensored logit-logistic model can be estimated with the cdfquantreg macro. It yields results that are very close to those obtained in R.

```
%INCLUDE tempdir(SAS_MACRO);
%cdfquantreg(subevents3, EQ1P, 'logit', 'logistic',
LMIV= SECSS,
DMIV= SECSS,
init=0.1| 0.1 |0.1 |0.1 |);
```

6.2.3 Analysis in Stata

The relevant data file is `events3.dta`. We also have to build the censored beta regression model "from scratch" in Stata. The next code chunk estimates the censored beta regression model. Its output is very similar to the results obtained in R.

```
/* beta-distribution model code */
capture program drop hybeta
program define hybeta
args lnf Xb Xd
tempvar phi mu w t
quietly {
gen double `phi' = exp(`Xd')
gen double `mu' = exp(`Xb')/(1 + exp(`Xb'))
gen double `w' = `mu'*`phi'
gen double `t' = (1-`mu')*`phi'
replace `lnf' = ln(betaden(`w',`t', $ML_y)) if $ML_y < 0.1 & $ML_y > 0.00001
replace `lnf' = ln(1 - ibeta(`w',`t',0.1)) if $ML_y > 0.09999
```

```
replace ‘lnf’ = ln(ibeta(‘w’,‘t’, 0.00001818)) if $ML_y < 0.0000101
  }
end
//
ml model lf hybeta (muvar: eq1p = dem indep nopref secss) (phivar: dem indep nopref)
ml search
ml max
```

The uncensored beta regression model can be estimated using the `betareg` function for Stata 14 or later, and betafit for Stata 11 and later. We display the model command below. Its estimates are very close to those from the `betareg` package in R.

```
betareg eq1p dem indep nopref secss, scale(dem indep nopref)
```

The following code chunk estimates the censored logit-logistic regression model. The results are reasonably close, but not identical, to those from R. The log-likelihood for the model in R is 973.96, whereas this model has 963.26, and the coefficients differ slightly from those obtained in the R version of this model. Users copying this code and pasting it into the Stata command window must first eliminate the line-breaks within any of the commands. Otherwise Stata will interpret the command fragments as separate commands.

```
* logit-logistic-distribution model
capture program drop hylogit
program define hylogit
args lnf Xb Xd
tempvar sigma mu
quietly {
gen double ‘sigma’ = exp(‘Xd’)
gen double ‘mu’ = ‘Xb’
* Remember to eliminate the line-break in commands before using them in Stata.
replace ‘lnf’ = ln((exp(‘mu’/‘sigma’)*(-1 + 1/$ML_y)^(-1 + 1/‘sigma’))/
(( $ML_y + exp(‘mu’/‘sigma’)*(-1 + 1/$ML_y)^(1/‘sigma’)* $ML_y)^2*‘sigma’))
if $ML_y < 0.1 & $ML_y > 0.00001
replace ‘lnf’ = ln(1 - 1/(1 + exp(-(-ln(-1 + 1/$ML_y) - ‘mu’)/‘sigma’)))
if $ML_y > 0.09999
replace ‘lnf’ = ln(1/(1 + exp(-(-ln(-1 + 1/$ML_y) - ‘mu’)/‘sigma’)))
if $ML_y < 0.0000101
  }
end
ml model lf hylogit (muvar: eq1p = secss) (phivar: secss)
ml search
ml max
```

Finally, we display the uncensored logit-logistic model and relevant output using the `cdfquantreg` user-defined package. The results are fairly close to those from the `cdfquantreg` package in R.

```
cdfquantreg eq1p secss, cdf(logit) quantile(logistic) zvarlist(secss)
                                                Number of obs     =        297
                                                Wald chi2(1)      =      10.74
Log likelihood =  1407.2066                     Prob > chi2       =     0.0010
------------------------------------------------------------------------------
        eq1p |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
       secss |  -.0246407   .0075178    -3.28   0.001    -.0393754    -.009906
       _cons |  -5.396224   .4786263   -11.27   0.000    -6.334314   -4.458133
-------------+----------------------------------------------------------------
eq2          |
       secss |   .0047399    .002311     2.05   0.040     .0002104    .0092693
       _cons |   .0654552   .1566927     0.42   0.676    -.2416569    .3725672
------------------------------------------------------------------------------
```

6.3 Random-Intercept with Boundary-Inflation Example

The data are from the World Values Survey, the 2011 wave. The questions are from a group of items that ask participants to rate the degree to which a characteristic of a polity is an essential component of a democracy. The rating scale is 1-10, "where 1 means it is 'not at all an essential characteristic of democracy' and 10 means it is definitely 'an essential characteristic of democracy'". The items from the battery of nine selected for our example are the following:

- V132. Religious authorities ultimately interpret the laws.
- V133. People choose their leaders in free elections.
- V135. The army takes over when government is incompetent.
- V136. Civil rights protect people from state oppression.
- V137. The state makes people's incomes equal.
- V139. Women have the same rights as men.

The predictor included in the model is political orientation, as elicited by this question: "In political matters, people talk of 'the left' and 'the right.' How would you place your views on this scale, generally speaking?" The 1-10 scale is anchored with 'Left' at 1 and 'Right' at 10.

V133, 136, and 139 are strongly right-censored because most people agree that these are essential to democracies. V132, 135, and 137 are strongly left-censored because most people agree that these are not essential to democracies. We have reverse-scored the latter three items and divided the scores by 10 so that all of the items are 1-inflated variables in the unit interval. The model to

be estimated is a hurdle beta regression model with a random intercept in the location submodel.

This kind of model turns out to be rather difficult to fit. We have limited this example to being analyzed in R, using the `gamlss` package. Attempts to fit it in Stata and SAS were not successful, and although the `gamlss` model does converge and gives sensible-looking estimates, there is evidence of strong collinearity among the parameter estimates.

6.3.1 Analysis in R

The data-file is named `DemocEssLong.txt`. The package used for the hurdle beta regression model is `gamlss`.

```
> # Load the relevant packages and the data:
> library(gamlss)
> library(MASS)
> library(lmtest)
> democ <- read.table("DemocEssLong.txt", header = TRUE)
> demon <- na.omit(democ)
> attach(demon)
> # The histograms in Figure 6.2 were generated by the following code:
> par(mfrow = c(2,3))
> hist(prop[item == 'X132R'], breaks = c(seq(0.05,1.05,0.1)),
+ main = "X132R, Religious authorities", xlab = "", ylim = c(0,1300), col = "gray")
> hist(prop[item == 'X133'], breaks = c(seq(0.05,1.05,0.1)),
+ main = "X133, Choose leaders", xlab = "", ylim = c(0,1300), col = "gray")
> hist(prop[item == 'X135R'], breaks = c(seq(0.05,1.05,0.1)),
+ main = "X135R, Army takes over", xlab = "", ylim = c(0,1300), col = "gray")
> hist(prop[item == 'X136'], breaks = c(seq(0.05,1.05,0.1)),
+ main = "X136, Civil rights protect", xlab = "", ylim = c(0,1300), col = "gray")
> hist(prop[item == 'X137R'], breaks = c(seq(0.05,1.05,0.1)),
+ main = "X137R, Incomes made equal", xlab = "", ylim = c(0,1300), col = "gray")
> hist(prop[item == 'X139'], breaks = c(seq(0.05,1.05,0.1)),
+ main = "X139, Women same rights as men", xlab = "",
+ ylim = c(0,1300), col = "gray")
```

The next code chunk begins with an intercept-only model and builds the final model by a forward selection process. Because it is difficult to get this kind of model to converge, each earlier model's coefficients are used as starting-values for the model being estimated.

```
> mod0c<- gamlss(prop~random(as.factor(subid)), sigma.fo=~1, nu.fo=~1,
+ family=BEOI, data=demon)
> # Add item as a predictor in all three parameters:
> mod1c<- gamlss(prop~item + random(as.factor(subid)), sigma.fo=~item,
+ nu.fo=~item, family=BEOI, start.from = mod0c, data=demon)
> # It takes two runs to converge, so run it again with the previous
```

```
> # model's coefficients as starting values:
> mod1d<- gamlss(prop~item + random(as.factor(subid)), sigma.fo=~item,
+ nu.fo=~item, family=BEOI, start.from = mod1c, data=demon)
> # Now try adding poliorient as a predictor:
> mod2d<- gamlss(prop~item + poliorient + random(as.factor(subid)),
+ sigma.fo=~item + poliorient, nu.fo=~item + poliorient, family=BEOI,
+ start.from = mod1d, data=demon)
> summary(mod2d)
> # So poliorient doesn't have a significant effect in the one-inflation component.
> # Adding an interaction effect in the location submoddel gives our final model:
> mod3d<- gamlss(prop~item*poliorient + random(as.factor(subid)),
+ sigma.fo=~item + poliorient, nu.fo=~item, family=BEOI,
+ start.from = mod2d, data=demon)
> summary(mod3d)
> confint1 <- function (object, submodel, level = 0.95)
> {
>     cf <- coef(object, what = submodel)
>     pnames <- names(cf)
>     parm <- pnames
>     a <- (1 - level)/2
>     a <- c(a, 1 - a)
>     pct <- paste(format(100 * a, trim = TRUE, scientific = FALSE, digits = 3), "%")
>     fac <- qnorm(a)
>     ci <- array(NA, dim = c(length(parm), 2L),
>  dimnames = list(parm, pct))
>     if(submodel == "mu") {
>       ind = 1:length(pnames)
>     }else if (submodel == "sigma"){
>       ind = (1 + length(pnames)):(2*length(pnames))
>     }else if (submodel == "nu"){
>       ind = (1 + 2*length(pnames)):(3*length(pnames))
>     }
>     ses <- sqrt(diag(vcov(object)[ind, ind]))[parm]
>    ci[] <- cf[parm] + ses %o% fac
>     ci
>}
> confint1(mod3d, "mu")
> confint1(mod3d, "sigma")
> confint1(mod3d, "nu")
```

The best model is one that has an interaction term in the location submodel. Checking that the model reproduces the location, dispersion, and ones-inflation structures reveals that it is reasonably accurate for the means, somewhat underestimates the variances, and duplicates the ones-inflation probabilities for each of the six items. However, there are some worryingly strong parameter-estimate

correlations, so the model should be treated with some caution.

```
> # Get the parameter-estimate correlation matrix,
> # bearing in mind that it's a fairly large matrix.
> cov2cor(vcov(mod3d))
> #
> # Now start by checking the means:
> expectprop <- mod3d$nu.fv + (1-mod3d$nu.fv)*mod3d$mu.fv
> c(mean(prop), mean(prop[item == 'X132R']), mean(prop[item == 'X133']),
+ mean(prop[item == 'X135R']), mean(prop[item == 'X136']),
+ mean(prop[item == 'X137R']), mean(prop[item == 'X139']))
> c(mean(expectprop), mean(expectprop[item == 'X132R']),
+ mean(expectprop[item == 'X133']), mean(expectprop[item == 'X135R']),
+ mean(expectprop[item == 'X136']), mean(expectprop[item == 'X137R']),
+ mean(expectprop[item == 'X139']))
> #
> # Now check the variances:
> varprop <- mod3d$nu.fv*(1-mod3d$nu.fv)*(1-mod3d$mu.fv)^2 +
+ (1-mod3d$nu.fv)*(1-mod3d$mu.fv)*mod3d$mu.fv/(1+mod3d$sigma.fv)
> c(var(prop), var(prop[item == 'X132R']), var(prop[item == 'X133']),
+ var(prop[item == 'X135R']), var(prop[item == 'X136']),
+ var(prop[item == 'X137R']), var(prop[item == 'X139']))
> c(mean(varprop), mean(varprop[item == 'X132R']), mean(varprop[item == 'X133']),
+ mean(varprop[item == 'X135R']), mean(varprop[item == 'X136']),
+ mean(varprop[item == 'X137R']), mean(varprop[item == 'X139']))
> #
> # Then, check the ones-inflation rates:
> c(table(prop[item == 'X132R'])[10]/sum(table(prop[item == 'X132R'])),
+ table(prop[item == 'X133'])[10]/sum(table(prop[item == 'X133'])),
+ table(prop[item == 'X135R'])[10]/sum(table(prop[item == 'X135R'])),
+ table(prop[item == 'X136'])[10]/sum(table(prop[item == 'X136'])),
+ table(prop[item == 'X137R'])[10]/sum(table(prop[item == 'X137R'])),
+ table(prop[item == 'X139'])[10]/sum(table(prop[item == 'X139'])))
> c(exp(mod3d$nu.coefficients[1])/(1 + exp(mod3d$nu.coefficients[1])),
+ exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[2])/
+ (1 + exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[2])),
+ exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[3])/
+ (1 + exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[3])),
+ exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[4])/
+ (1 + exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[4])),
+ exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[5])/
+ (1 + exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[5])),
+ exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[6])/
+ (1 + exp(mod3d$nu.coefficients[1]+mod3d$nu.coefficients[6])))
```

The next code chunk produces the scatterplot in Figure 6.3.

```
> # First, we need a vector of the subject id numbers:
> idlist <- c(rep(0,length(mod3d$mu.coefSmo[[1]]$coef)))
> for (i in 1:length(idlist)) {idlist[i] <-
+ lapply(dimnames(mod3d$mu.coefSmo[[1]]$coef), '[[', i)}
> # Second, generate a list of means for each subid:
> meanlist <- c(rep(0,length(idlist)))
> for (i in 1:length(idlist)) {meanlist[i] <- mean(prop[subid==idlist[i]])}
> estlist <- c(rep(0,length(idlist)))
> for (i in 1:length(idlist)) {estlist[i] <- mean(expectprop[subid==idlist[i]])}
> par(mfrow = c(1,1))
> plot(meanlist,estlist, xlim = c(0,1), xlab = "data",
+ ylab = "predicted", ylim = c(0.4, 1.0))
```

6.4 Bayesian MCMC Analysis of Grammaticality Judgments Example

We begin by replicating a reduced version of the beta-regression model from Chapter 3, with predictors only in the location submodel. Note that in the betareg model here we have divided sentence-length by 10. This is to prevent the MCMC model from failing because of computational errors. The Bayesian model therefore also employs length/10 as an independent variable.

```
> library(betareg)
> library(R2OpenBUGS)
> # Load the data
> accep <- read.table("acceptab.txt", header = TRUE)
> attach(accep)
> # Replicate the model from Chapter 3:
> leng10 <- length/10
> mod1 <- betareg(rating ~ es+ja+no+zh+leng10|1, data = accep)
> summary(mod1)
```

Table 6.1: Beta-Regression Model Output

	coef.	estim.	s.e.	z	p
	β_0	2.039	0.221		
Spanish	β_1	-0.538	0.195	-2.756	.006
Japanese	β_2	-1.858	0.181	-10.243	$< .001$
Norwegian	β_3	-0.658	0.189	-3.475	.001
Chinese	β_4	-1.548	0.187	-8.283	$< .001$
length/10	β_5	-0.331	0.113	-2.915	.004
	δ	1.460	0.084		

The output from this model is shown in the table above. Comparing it with the output for the MCMC model in Chapter 6, the pattern of the coefficients

is the same but their magnitudes noticeably differ, and of course there is an entirely different model for the variance because the MCMC model is taking the within-sentence variability into account whereas the beta regression model does not.

The MCMC model itself is shown below. The priors for the location and precision submodel coefficients all are "non-informative" Gaussian priors, with means set to 0 and precisions (reciprocals of variances) set to 0.001.

```
# Null submodel for the variance
model {
for (i in 1:N) {
r[i] ~ dbeta(omega[i], tau[i]);
omega[i] <- mu[i]*phi[i];
tau[i] <- (1 - mu[i])*phi[i];
mu[i] <- exp(eta[i])/(1 + exp(eta[i]));
eta[i] <- beta1 + beta2*esp[i] + beta3*jap[i] + beta4*nor[i]
+ beta5*zho[i] + beta6*leng10[i];
phi[i] <- max(-1 + (1 - mu[i])*mu[i]/(pow(se[i] ,2)+ exp(delta1)),0.01);
}
delta1 ~ dnorm(0, 0.001);
beta1 ~ dnorm(0, 0.001);
beta2 ~ dnorm(0, 0.001);
beta3 ~ dnorm(0, 0.001);
beta4 ~ dnorm(0, 0.001);
beta5 ~ dnorm(0, 0.001);
beta6 ~ dnorm(0, 0.001);
}
```

The MCMC model takes a little while to run, several minutes on a typical laptop. The dialog in R is as follows.

```
> # Tell R2OpenBUGS what the model file is:
> model.file <- "ch6BUGSmodel2null.txt"
> # Give it the data, as named in the model code:
> N <- 247
> esp <- es
> jap <- ja
> nor <- no
> zho <- zh
> leng10 <- length/10
> r <- rating
> se <- se
> data <- list ("N", "esp", "jap", "nor", "zho", "leng10", "r", "se")
> # For a 2-chain model, give it two inits:
> inits <- list(
> list(beta1 = 1.6, beta2 = -0.3, beta3 = -1.5, beta4 = -0.4, beta5 = -1.2,
```

```
+ beta6 = -0.3, delta1 = -3.2),
> list(beta1 = 1.8, beta2 = -0.4, beta3 = -1.6, beta4 = -0.5, beta5 = -1.0,
+ beta6 = -0.2,  delta1 = -2.8))
> # Tell it which parameters to estimate:
> parameters <- c("beta1","beta2","beta3","beta4","beta5","beta6", "delta1")
> # Run the model:
> ch6.sim <- bugs(data, inits, parameters, model.file, n.chains=2,
+ n.burnin = 2000, n.iter=8000, debug = TRUE)
> print(ch6.sim, digits = 5)
> plot(ch6.sim)
```

The output in OpenBUGS should look like this (although the times taken for the burn-in and final run may differ):

```
model is syntactically correct
data loaded
model compiled
initial values loaded and chain initialized but another chain contains uninitialized variables
model is initialized
model is already initialized
model is updating
2000 updates took 166 s
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
inference can not be made when sampler is in adaptive phase
DIC can not be monitored when model in adapting phase
model is updating
6000 updates took 504 s
CODA files written
no monitors set
DIC monitor not set
History
```

The print command should produce output like the table below (columns have been removed to stay within the width of the page).

```
Inference for Bugs model at "ch6BUGSmodel2null.txt",
Current: 2 chains, each with 8000 iterations (first 2000 discarded)
Cumulative: n.sims = 12000 iterations saved
               mean      sd       2.5%   ...     97.5%    Rhat n.eff
delta1     -3.18134 0.08178   -3.34103   ...   -3.02300 1.00110  7900
```

```
beta1       1.64742 0.18364    1.28497   ...    2.00602 1.00150  2600
beta2      -0.24942 0.15321   -0.55010   ...    0.04728 1.00111  7700
beta3      -1.52598 0.16148   -1.84100   ...   -1.20898 1.00126  4400
beta4      -0.39130 0.14765   -0.67382   ...   -0.10390 1.00111  7600
beta5      -1.20930 0.16701   -1.53200   ...   -0.87740 1.00113  7100
beta6      -0.28675 0.10316   -0.48670   ...   -0.08170 1.00092 12000
deviance -132.75623 3.78466 -138.10000   ... -123.50000 1.00105 11000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
DIC info (using the rule, pD = var(deviance)/2)
pD = 7.26146 and DIC = -125.48778
DIC is an estimate of expected predictive error (lower deviance is better).
```

There are three commonly used sources of evidence for convergence. One of these, as mentioned in the output above, is the "Rhat" statistic, which should be close to 1 if convergence has been achieved. A second is inspecting graphs of the iteration history for the parameter estimates, to see whether they appear to be stable and whether the chains have mixed well. An example of such a graph is displayed in Figure 6.1.

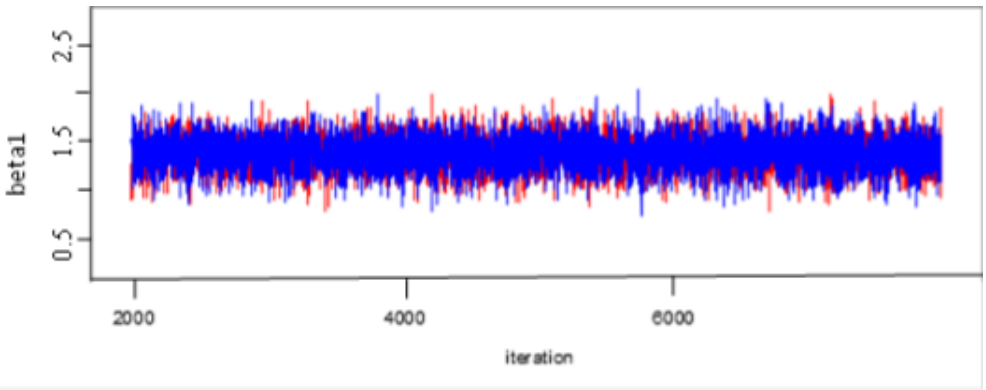

Figure 6.1: Two-chain mixing for the beta1 parameter

The third is the "effective sample size" (ESS), which is denoted by "n.eff" in the output table above. The main concept behind the ESS is the fact that the samples drawn via MCMC processes generally are not independent samples, so parameter estimates at any iteration may be correlated with estimates at earlier iterations. The definition of the ESS is

$$ESS = N \Big/ \left(1 + 2\sum_{j=1}^{\infty} \rho_j\right),$$

where N is the sample size and ρ_j is the correlation between iterations separated by j lags. The smaller the ESS of a parameter, the stronger this autocorrelation, and therefore the poorer the estimate of the posterior distribution

of that parameter. We can see in the output table above that the ESS for beta1 is smaller than the ESS for the other parameters, indicating that the autocorrelation among estimates for this parameter is strongest.

We now present a Bayesian MCMC version of the fixed-effects model for the grammaticality data in Chapter 3. The model syntax is shown below. As before, the priors for the location and precision submodel coefficients all are "non-informative" Gaussian priors, with means set to 0 and precisions (reciprocals of variances) set to 0.001.

```
model {
for (i in 1:N) {
r[i] ~ dbeta(omega[i], tau[i]);
omega[i] <- mu[i]*phi[i];
tau[i] <- (1 - mu[i])*phi[i];
mu[i] <- exp(eta[i])/(1 + exp(eta[i]));
phi[i] <- exp(theta[i]);
eta[i] <- beta1 + beta2*esp[i] + beta3*jap[i] + beta4*nor[i]
+ beta5*zho[i] + beta6*leng[i];
theta[i] <- delta1 + delta2*esp[i] + delta3*jap[i] + delta4*nor[i]
+ delta5*zho[i] + delta6*leng[i];
}
beta1 ~ dnorm(0, 0.001);
delta1 ~ dnorm(0, 0.001);
beta2 ~ dnorm(0, 0.001);
delta2 ~ dnorm(0, 0.001);
beta3 ~ dnorm(0, 0.001);
delta3 ~ dnorm(0, 0.001);
beta4 ~ dnorm(0, 0.001);
delta4 ~ dnorm(0, 0.001);
beta5 ~ dnorm(0, 0.001);
delta5 ~ dnorm(0, 0.001);
beta6 ~ dnorm(0, 0.001);
delta6 ~ dnorm(0, 0.001);
}
```

The dialog in R is as follows.

```
> # Tell R2OpenBUGS what the model file is:
> model.file <- "ch6BUGSmodel.txt"
> # Give it the data, as named in the model code:
> N <- 247
> esp <- es
> jap <- ja
> nor <- no
> zho <- zh
> leng <- length/10
```

```
> r <- rating
> data <- list ("N", "esp", "jap", "nor", "zho", "leng", "r")
> # For a 2-chain model, give it two inits:
> inits <- list(
> list(beta1 = 2.2, beta2 = -0.9, beta3 = -2.2, beta4 = -0.9,
+ beta5 = -1.8, beta6 = -0.3,
+ delta1 = 1.8, delta2 = -1.1, delta3 = -1.0, delta4 = -0.8,
+ delta5 = -1.0, delta6 = 0.4),
> list(beta1 = 1.9, beta2 = -1.0, beta3 = -2.1, beta4 = -1.0,
+ beta5 = -1.6, beta6 = -0.4,
+ delta1 = 1.6, delta2 = -1.2, delta3 = -1.1, delta4 = -0.7,
+ delta5 = -0.9, delta6 = 0.3))
> # Tell it which parameters to estimate:
> parameters <- c("beta1","beta2","beta3","beta4","beta5","beta6",
+ "delta1","delta2","delta3","delta4","delta5","delta6")
> # Run the model:
> ch6.sim <- bugs(data, inits, parameters, model.file, n.chains=2, n.burnin = 2000,
+ n.iter=8000, debug = TRUE)
> print(ch6.sim, digits = 5)
```

Truncated output is shown next (columns have been removed to stay within the width of the page). The evidence for convergence is reasonably favorable, with "Rhat" values close to 1 and graphs of chain mixing similar to Figure 6.1 (not shown here). However, it also is true that the ESS's for some of the parameters (e.g., beta4, delta2, and delta5) are rather small, which suggests that their estimates may be viewed with a bit of caution.

```
Current: 2 chains, each with 8000 iterations (first 2000 discarded)
Cumulative: n.sims = 12000 iterations saved
               mean      sd       2.5%   ...    97.5%    Rhat n.eff
beta1       2.25064 0.22021    1.81797   ...  2.67605 1.00110  8000
beta2      -0.88011 0.20045   -1.28600   ... -0.48409 1.00094 12000
beta3      -2.17915 0.17316   -2.52900   ... -1.84398 1.00254 12000
beta4      -0.88849 0.18546   -1.26402   ... -0.52937 1.00324   660
beta5      -1.83869 0.18364   -2.20300   ... -1.48100 1.00096 12000
beta6      -0.27928 0.11280   -0.49761   ... -0.05426 1.00119  5500
delta1      1.77502 0.33828    1.10000   ...  2.42800 1.00170  1900
delta2     -1.16594 0.28935   -1.73600   ... -0.60490 1.00271   840
delta3     -1.11465 0.26552   -1.62900   ... -0.58426 1.00132  3700
delta4     -0.73858 0.28152   -1.29400   ... -0.19030 1.00183  1600
delta5     -1.04781 0.27459   -1.58502   ... -0.52279 1.00294   740
delta6      0.38150 0.17231    0.04744   ...  0.72480 1.00095 12000
deviance -188.55067 5.02183 -196.30000 ... -176.90000 1.00118  5800

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```

```
DIC info (using the rule, pD = var(deviance)/2)
pD = 12.60828 and DIC = -175.94239
DIC is an estimate of expected predictive error (lower deviance is better).
```

Comparing this model's output with Table 3.1 in Chapter 3, and allowing for the use of sentence length divided by 10 in the MCMC model, we can see that the mean coefficient estimates here are quite similar to those in Table 3.1. However, as expected, the standard deviations in the Bayesian MCMC model are slightly larger than the beta regression standard errors.

References

Azevedo, J. P. (2011). *Grqreg: Stata module to graph the coefficients of a quantile regression.* Retrieved from `https://EconPapers.repec.org/RePEc:boc:bocode:s437001`

Buis, M. L. (2003). zoib: Stata module to fit a zero-one inflated beta distribution [Computer software manual]. Statistical Software Components S435303, Boston College Department of Economics.

Buis, M. L., Cox, N. J., & Jenkins, S. P. (2003). betafit: Stata module to fit a two-parameter beta distribution [Computer software manual]. Statistical Software Components S435303, Boston College Department of Economics. (revised 03 Feb 2012)

Everett, J. A. C. (2013). The 12 item social and economic conservatism scale (secs). *PloS one*, *8*(12), e82131.

R Core Team. (2018). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org/`

SAS Institute Inc. (2013). *Sas 9.4.* Cary, NC: SAS Institute Inc.

StataCorp. (2017). Stata: Release 15 [Computer software manual]. College Station, TX, USA.