

CSSS 510: Lab 3

Logistic Regression

2017-10-13

0. Agenda

1. Deriving a likelihood function for the logistic regression model
2. Fitting a logit model using `optim()` and `glm()`
3. Simulating predicted values and confidence intervals
4. Simulating first differences

1. Deriving a likelihood function for the logistic regression model

Recall from lecture the logit model:

$$y_i \sim \text{Bern}(\pi_i)$$

$$\pi_i = \text{logit}^{-1}(\mathbf{x}_i\boldsymbol{\beta})$$

$$\pi_i = \frac{\exp(\mathbf{x}_i\boldsymbol{\beta})}{1 + \exp(\mathbf{x}_i\boldsymbol{\beta})} = \frac{1}{1 + \exp(-\mathbf{x}_i\boldsymbol{\beta})}$$

1. Deriving a likelihood function for the logistic regression model

In the simple case, this stems from the latent variable model:

$$y^* = \beta_0 + \beta_1 x + \epsilon$$

where the relationship between latent variable y^* and the explanatory variable x is modeled using simple linear regression, and the binary outcome y is a function of the sign of y^* :

$$y = \begin{cases} 1, & \text{if } y^* > 0 \\ 0, & \text{if } y^* \leq 0 \end{cases}$$

1. Deriving a likelihood function for the logistic regression model

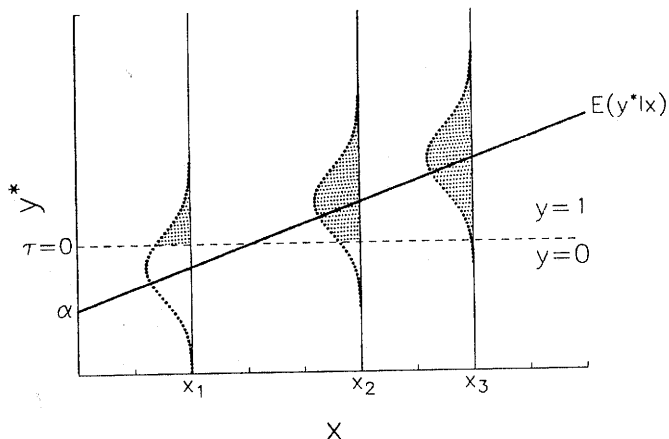


Figure 3.2. The Distribution of y^* Given x in the Binary Response Model

1. Deriving a likelihood function for the logistic regression model

The logistic regression model is obtained if we assume the errors of this latent variable model follow a standard logistic distribution.

Recall that the pdf and cdf of the standard logisitic distribution are as follows:

$$f(t) = \frac{\exp(t)}{(1 + \exp(t))^2}$$

$$F(t) = \frac{\exp(t)}{1 + \exp(t)}$$

1. Deriving a likelihood function for the logistic regression model

We therefore have the following:

$$\begin{aligned}\Pr(y = 1|x) &= \Pr(y^* > 0|x) \\ &= \Pr(\beta_0 + \beta_1 x + \epsilon > 0|x) \\ &= \Pr(\epsilon > -(\beta_0 + \beta_1 x)) \\ &= \Pr(\epsilon < \beta_0 + \beta_1 x) \\ &= F(\beta_0^L + \beta_1^L x)\end{aligned}$$

1. Deriving a likelihood function for the logistic regression model

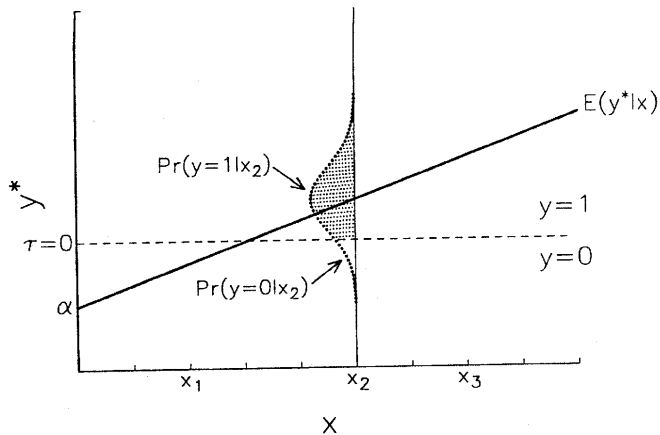


Figure 3.4. Probability of Observed Values in the Binary Response Model

1. Deriving a likelihood function for the logistic regression model

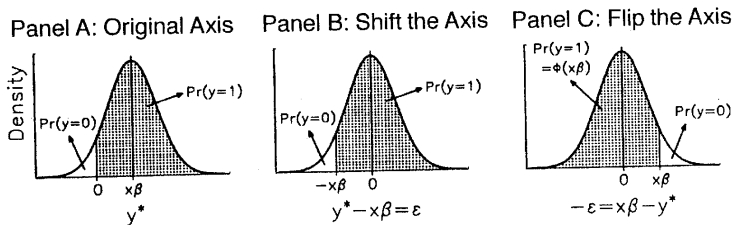


Figure 3.5. Computing $\Pr(y = 1 | x)$ in the Binary Response Model

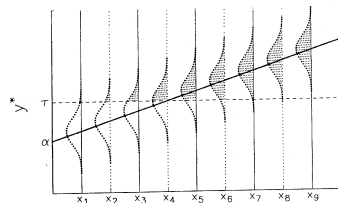
1. Deriving a likelihood function for the logistic regression model

Since we assume the errors follow a standard logistic distribution, we have

$$\begin{aligned}\Pr(y = 1|x) &= F(\beta_0^L + \beta_1^L x) \\ &= \frac{\exp(\beta_0^L + \beta_1^L x)}{1 + \exp(\beta_0^L + \beta_1^L x)}\end{aligned}$$

$$E(\epsilon) = 0 \text{ and } \text{Var}(\epsilon) = \frac{\pi^2}{3}.$$

Panel A: Plot of y^*



Panel B: Plot of $\Pr(y=1|x)$

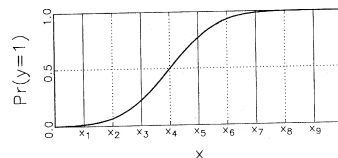


Figure 3.6. Plot of y^* and $\Pr(y = 1|x)$ in the Binary Response Model

1. Deriving a likelihood function for the logistic regression model

The logit function is the inverse of the logistic function:

$$\text{logit}(p) = \log \frac{p}{1-p}$$

or

$$\text{logit}^{-1}(p) = \frac{\exp(x)}{1 + \exp(x)}$$

We therefore have the following

$$\Pr(y = 1|x) = \text{logit}^{-1}(\beta_1^L + \beta_1^L x)$$

or

$$\text{logit}(\Pr(y = 1|x)) = \beta_1^L + \beta_1^L x$$

or

$$\log \frac{\Pr(y = 1|x)}{\Pr(y = 0|x)} = \beta_0^L + \beta_1^L x.$$

1. Deriving a likelihood function for the logistic regression model

Recall from lecture that the Bernoulli distribution has the following pdf:

$$\Pr(y_i = 1|\pi_i) = \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

And the likelihood function can be derived from the joint probability:

$$\mathcal{L}(\boldsymbol{\pi}|\mathbf{y}) \propto \prod_{i=1}^n \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

$$\mathcal{L}(\boldsymbol{\beta}|\mathbf{y}) \propto \prod_{i=1}^n \left(\frac{1}{1 + \exp(-\mathbf{x}_i\boldsymbol{\beta})} \right)^{y_i} \left(1 - \frac{1}{1 + \exp(-\mathbf{x}_i\boldsymbol{\beta})} \right)^{1-y_i}$$

$$\mathcal{L}(\boldsymbol{\beta}|\mathbf{y}) \propto \prod_{i=1}^n (1 + \exp(-\mathbf{x}_i\boldsymbol{\beta}))^{-y_i} (1 + \exp(-\mathbf{x}_i\boldsymbol{\beta}))^{-(1-y_i)}$$

$$\log \mathcal{L}(\boldsymbol{\beta}|\mathbf{y}) \propto \sum_{i=1}^n -y_i \log(1 + \exp(-\mathbf{x}_i\boldsymbol{\beta})) - (1 - y_i) \log(1 + \exp(\mathbf{x}_i\boldsymbol{\beta}))$$

2. Fitting a logit model using optim() and glm()

```
rm(list = ls()) # clear up the memory

#install and load the packages needed
#from CRAN: install.packages("MASS", dependencies = TRUE)
library(MASS)
library(RColorBrewer)

# download simcf and tile packages
# from- http://faculty.washington.edu/cadolph/software
# don't unzip the archive (tar) file
library(simcf)
library(tile)
```

```
## Loading required package: grid
```

```
# Load data
file <- "nes00a.csv"
data <- read.csv(file, header=TRUE)
# attach(data)
```

2. Fitting a logit model using optim() and glm()

```
# Estimate logit model using optim()
# Construct variables and model objects
y <- data$vote00
x <- cbind(data$age,data$hsdeg,data$coldeg)

# Likelihood function for logit
llk.logit <- function(param,y,x) {
  os <- rep(1,length(x[,1])) # constant
  x <- cbind(os,x) # constant+covariates
  b <- param[ 1 : ncol(x) ]
  # number of parameters to be estimated equals number of columns in x
  # (i.e., one for constant and one for each covariates : total 4)
  xb <- x%*%b
  sum( y*log(1+exp(-xb)) + (1-y)*log(1+exp(xb)))
  # log-likelihood function for logit model
  # (based on our choice of standard logistic cdf as the systematic component)
  # optim is a minimizer, so use -lnL here
}
```

2. Fitting a logit model using optim() and glm()

```
# Fit logit model using optim
ls.result <- lm(y~x) # use ls estimates as starting values (for convenience)
stval <- ls.result$coefficients # initial guesses
logit.result.opt <- optim(stval,llk.logit,method="BFGS",hessian=T,y=y,x=x)
# call minimizer procedure or max by adding control=list(fnscale=-1)
pe.opt <- logit.result.opt$par # point estimates
vc.opt <- solve(logit.result.opt$hessian) # var-cov matrix
se.opt <- sqrt(diag(vc.opt)) # standard errors
ll.opt <- -logit.result.opt$value # likelihood at maximum

logit.optim<-data.frame(cbind(round(pe.opt,3), round(se.opt,3)))
rownames(logit.optim)<-c("intercept", "age", "highschool", "college")
colnames(logit.optim)<-c("pe", "std.err")
logit.optim
```

##	pe	std.err
## intercept	-2.149	0.257
## age	0.031	0.003
## highschool	1.213	0.179
## college	1.102	0.130

2. Fitting a logit model using `optim()` and `glm()`

```
#p-value based on t-statistics
```

```
2*pt(abs(logit.optim$pe/logit.optim$std.err),  
      df=length(y)-length(pe.opt) , lower.tail = FALSE)
```

```
## [1] 1.229175e-16 2.393414e-24 1.668155e-11 4.780627e-17
```

```
# Estimate logit model using glm()
```

```
# Run logit & extract results using glm.
```

```
# GLM solves the likelihood equations with a common numeric algorithm
```

```
# called iteratively re-weighted least squares (IRWLS).
```

```
logit.result <- glm(vote00~age +hsdeg+coldeg, family=binomial, data=data)
```

```
# family "binomial" calls logit transformation
```

```
# (log(pi/1-pi)) as a "link function"
```

```
# corresponding to logistic distribution.
```

```
# Link function transforms pi so that it follows a linear model.
```

```
# (So although pi itself is dependent on covariates in a non-linear way,
```

```
# logit transformed pi is dependent on covariates in a linear way.)
```


2. Fitting a logit model using `optim()` and `glm()`

```
summary(logit.result)
```

```
##
## Call:
## glm(formula = vote00 ~ age + hsdeg + coldeg, family = binomial,
##      data = data)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.4487  -1.1347   0.6360   0.8973   1.8854
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.147997   0.256608  -8.371 < 2e-16 ***
## age          0.030885   0.003386   9.121 < 2e-16 ***
## hsdeg        1.212882   0.179447   6.759 1.39e-11 ***
## coldeg       1.102465   0.130426   8.453 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 2293.5  on 1782  degrees of freedom
## Residual deviance: 2076.0  on 1779  degrees of freedom
## AIC: 2084
##
## Number of Fisher Scoring iterations: 4
```

2. Fitting a logit model using `optim()` and `glm()`

```
# now a new model adding age^2
model <- vote00 ~ age + I(age^2) + hsdeg + coldeg
mdata <- extractdata(model, data, na.rm=TRUE) # needs library(simcf)

logit.result <- glm(model, family=binomial, data=mdata)

pe <- logit.result$coefficients # point estimates
vc <- vcov(logit.result)       # var-cov matrix

pe
```

```
##      (Intercept)          age      I(age^2)          hsdeg          coldeg
## -3.0193890720   0.0747251922 -0.0004427014   1.1243907749   1.0795702357
```

```
sqrt(diag(vc))
```

```
##      (Intercept)          age      I(age^2)          hsdeg          coldeg
## 0.4181899367   0.0168440337 0.0001655466 0.1800068893 0.1312113265
```

3. Simulating predicted values and confidence intervals

```
# Simulate parameter distributions
sims <- 10000
simbetas <- mvnrm(sims, pe, vc) #needs library(MASS)
# draw 10000 sets of simulated
# parameter (beta) estimates from a multivariate normal distribution with
# mean pe and variance-covariance vc

# Now let's plan counterfactuals: We will have three
# sets of counterfactuals based on education level
# (less than hs edu, hs edu, college or higher edu), and for each set
# we will make age varies between 18 years old and 97 years old.

# Set up counterfactuals: all ages, each of three educations
xhyp <- seq(18,97,1) # create age vector
nscen <- length(xhyp) # we will have total 80 different age scenarios
#for each education level

nohsScen <- hsScen <- collScen <- cfMake(model, mdata, nscen)
#this is just to initialize 80 scenarios for each education level.
#As default, all covariate values are set at the mean.
```

3. Simulating predicted values and confidence intervals

```
# Create three sets of education counterfactuals
```

```
for (i in 1:nscen) {
```

```
# No High school scenarios (loop over each age, total 80 scenarios)
```

```
nohsScen <- cfChange(nohsScen, "age", x = xhyp[i], scen = i)
```

```
nohsScen <- cfChange(nohsScen, "hsdeg", x = 0, scen = i)#no hs degree
```

```
nohsScen <- cfChange(nohsScen, "coldeg", x = 0, scen = i)#no college degree
```

```
# HS grad scenarios (loop over each age, total 80 scenarios)
```

```
hsScen <- cfChange(hsScen, "age", x = xhyp[i], scen = i)
```

```
hsScen <- cfChange(hsScen, "hsdeg", x = 1, scen = i)#has hs degree
```

```
hsScen <- cfChange(hsScen, "coldeg", x = 0, scen = i)#no college degree
```

```
# College grad scenarios (loop over each age, total 80 scenarios)
```

```
collScen <- cfChange(collScen, "age", x = xhyp[i], scen = i)
```

```
collScen <- cfChange(collScen, "hsdeg", x = 1, scen = i)#has hs degree
```

```
collScen <- cfChange(collScen, "coldeg", x = 1, scen = i)#has college degree
```

```
}
```

3. Simulating predicted values and confidence intervals

```
# Now given the counterfactual covariates (nohsScen/hsScen/collScen)  
# and simulated parameters (simbetas), we can calculate expected value  
# of the response. In this case, expected probability of voting!
```

```
head(nohsScen$x) #we will fit the counterfactual data
```

```
##      vote00 age hsdeg coldeg  
## 1 0.6567583 18      0      0  
## 2 0.6567583 19      0      0  
## 3 0.6567583 20      0      0  
## 4 0.6567583 21      0      0  
## 5 0.6567583 22      0      0  
## 6 0.6567583 23      0      0
```

```
nohsScen$model # in the model specification
```

```
## vote00 ~ age + I(age^2) + hsdeg + coldeg
```

3. Simulating predicted values and confidence intervals

```
nohsSims <- logitsimev(nohsScen, simbetas, ci=0.95) # using simulated  
# betas to get expected values.  
# Built-in function "logitsimev" calculates the expected value  
# for every individual scenario you created.  
# reports lower and upper confidence intervals  
# as well as expected probabilities  
  
# same thing for two other sets of scenarios  
hsSims <- logitsimev(hsScen, simbetas, ci=0.95)  
collSims <- logitsimev(collScen, simbetas, ci=0.95)
```

3. Simulating predicted values and confidence intervals

```
# Get 3 nice colors for traces
col <- brewer.pal(3,"Dark2")

# Set up lineplot traces of expected probabilities

#Traces are elements of tile : lines, labels, legends...

# no hs
nohsTrace <- lineplot(x=xhyp, # age on x-axis
  y=nohsSims$pe, #expected probability on y-axis
  lower=nohsSims$lower, # lower confidence interval
  upper=nohsSims$upper, #upper confidence interval
  col=col[1], #color choice
  extrapolate=list(data=mdata[,2:ncol(mdata)],
    #actual covariates (i.e., values in your data)
    cfact=nohsScen$x[,2:ncol(hsScen$x)],#counterfactual covariates
    omit.extrapolated=FALSE), #don't show extrapolated values
  plot=1)
```

3. Simulating predicted values and confidence intervals

```
# hs but no college
hsTrace <- lineplot(x=xhyp,
  y=hsSims$pe,
  lower=hsSims$lower,
  upper=hsSims$upper,
  col=col[2],
  extrapolate=list(data=mdata[,2:ncol(mdata)],
    cfact=hsScen$x[,2:ncol(hsScen$x)],
    omit.extrapolated=FALSE),
  plot=1)
```

```
#college
collTrace <- lineplot(x=xhyp,
  y=collSims$pe,
  lower=collSims$lower,
  upper=collSims$upper,
  col=col[3],
  extrapolate=list(data=mdata[,2:ncol(mdata)],
    cfact=collScen$x[,2:ncol(hsScen$x)],
    omit.extrapolated=FALSE),
  plot=1)
```

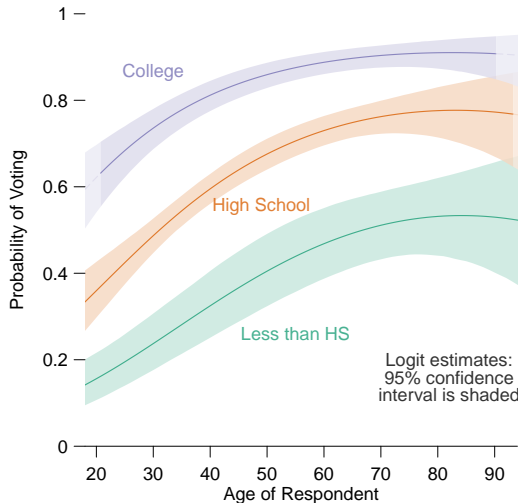

3. Simulating predicted values and confidence intervals

```
# Set up traces with labels
labelTrace <- textTile(labels=c("Less than HS", "High School", "College"),
  x=c( 55,    49,    30),
  y=c( 0.26,  0.56,  0.87),
  col=col,
  plot=1)

# For legend
legendTrace <-
  textTile(labels=c("Logit estimates:", "95% confidence", "interval is shaded"),
    x=c(82, 82, 82),
    y=c(0.2, 0.16, 0.12),
    plot=1)

#options(device="quartz")
# Plot traces using tile
voting<-tile(nohsTrace,
  hsTrace,
  collTrace,
  labelTrace,
  legendTrace,
  limits=c(18,94,0,1),
  xaxis=list(at=c(20,30,40,50,60,70,80,90)),
  yaxis=list(label.loc=-0.5, major=FALSE),
  xaxistitle=list(labels="Age of Respondent"),
  yaxistitle=list(labels="Probability of Voting"),
  width=list(null=5,yaxistitle=4,yaxis.labelspace=-0.5)
  ,output=list(file="educationEV",width=5.5)
)
```

3. Simulating predicted values and confidence intervals



4. Simulating first differences

```
#####  
#  
# Now consider a new specification adding the variable  
# "ever married", or marriedo  
# We will estimate this new model with glm(), then  
# simulate new scenarios for marrieds and non-marrieds  
  
# Estimate logit model using glm()  
# Set up a new model formula and model specific data frame  
  
model2 <- vote00 ~ age + I(age^2) + hsdeg + coldeg + marriedo  
mdata2 <- extractdata(model2, data, na.rm=TRUE)  
  
# Run logit & extract results  
logit.m2 <- glm(model2, family=binomial, data=mdata2)  
pe.m2 <- logit.m2$coefficients # point estimates  
vc.m2 <- vcov(logit.m2)       # var-cov matrix  
  
# Simulate parameter distributions  
sims <- 10000  
simbetas.m2 <- mvrnorm(sims, pe.m2, vc.m2)
```

4. Simulating first differences

```
# Set up counterfactuals: all ages
xhyp <- seq(18,97,1)
nscen <- length(xhyp)
marriedScen <- notmarrScen <- cfMake(model2, mdata2, nscen)
for (i in 1:nscen) {

  # - we will use the marriedScen counterfactuals in FDs and RRs as well as EVs
  # Note below the careful use of before scenarios (xpre) and after scenarios (x)
  # :i.e., use of the same age range (18-97) for both x and xpre, only marriedo values differ.

  # Married (loop over each age)
  marriedScen <- cfChange(marriedScen, "age", x = xhyp[i], xpre= xhyp[i], scen = i)
  marriedScen <- cfChange(marriedScen, "marriedo", x = 1, xpre= 0, scen = i)

  # Not Married (loop over each age)
  notmarrScen <- cfChange(notmarrScen, "age", x = xhyp[i], scen = i)
  notmarrScen <- cfChange(notmarrScen, "marriedo", x = 0, scen = i)
}

# Simulate expected probabilities for all age scenarios for married and not married respectively
marriedSims <- logitsimev(marriedScen, simbetas.m2, ci=0.95)
notmarrSims <- logitsimev(notmarrScen, simbetas.m2, ci=0.95)

# Simulate first difference of voting wrt marriage:  $E(y/\text{married}) - E(y/\text{notmarried})$ 
marriedFD <- logitlimfd(marriedScen, simbetas.m2, ci=0.95)

# Simulate relative risk of voting wrt marriage:  $E(y/\text{married}) / E(y/\text{notmarried})$ 
marriedRR <- logitlimrr(marriedScen, simbetas.m2, ci=0.95)
```

4. Simulating first differences

```
## Make plots using tile

# Get 3 nice colors for traces
col <- brewer.pal(3,"Dark2")

# Set up lineplot traces of expected probabilities
marriedTrace <- lineplot(x=xhyp,
                        y=marriedSims$pe,
                        lower=marriedSims$lower,
                        upper=marriedSims$upper,
                        col=col[1],
                        extrapolate=list(data=mdata2[,2:ncol(mdata2)],
                                         cfact=marriedScen$x[,2:ncol(marriedScen$x)],
                                         omit.extrapolated=TRUE),
                        plot=1)

notmarrTrace <- lineplot(x=xhyp,
                        y=notmarrSims$pe,
                        lower=notmarrSims$lower,
                        upper=notmarrSims$upper,
                        col=col[2],
                        ci = list(mark="dashed"),
                        extrapolate=list(data=mdata2[,2:ncol(mdata2)],
                                         cfact=notmarrScen$x[,2:ncol(notmarrScen$x)],
                                         omit.extrapolated=TRUE),
                        plot=1)
```

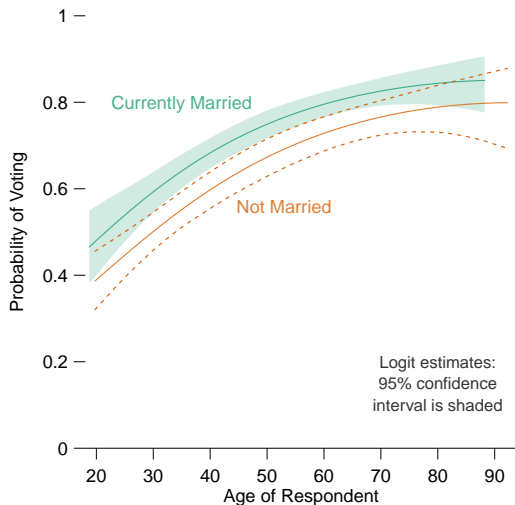
4. Simulating first differences

```
# Set up traces with labels and legend
labelTrace <- textTile(labels=c("Currently Married", "Not Married"),
                      x=c( 35,    53),
                      y=c( 0.8,  0.56),
                      col=col,
                      plot=1)

legendTrace <- textTile(labels=c("Logit estimates:", "95% confidence", "interval is shaded"),
                      x=c(80, 80, 80),
                      y=c(0.2, 0.15, 0.10),
                      cex=0.9,
                      plot=1)

# Plot traces using tile
tile(marriedTrace,
     notmarrTrace,
     labelTrace,
     legendTrace,
     limits=c(18,94,0,1),
     xaxis=list(at=c(20,30,40,50,60,70,80,90)),
     yaxis=list(label.loc=-0.5, major=FALSE),
     xaxis.title=list(labels="Age of Respondent"),
     yaxis.title=list(labels="Probability of Voting"),
     width=list(null=5,yaxis.title=4,yaxis.labelspace=-0.5)
     #, output=list(file="marriedEV", width=5.5)
)
```

4. Simulating first differences



4. Simulating first differences

```
# Plot First Difference  
# Set up lineplot trace of first difference  
  
marriedFDTrace <- lineplot(x=xhyp,  
                           y=marriedFD$pe,  
                           lower=marriedFD$lower,  
                           upper=marriedFD$upper,  
                           col=col[1],  
                           extrapolate=list(data=mdata2[,2:ncol(mdata2)],  
                                             cfact=marriedScen$x[,2:ncol(marriedScen$x)],  
                                             omit.extrapolated=TRUE),  
                           plot=1)  
  
# Set up baseline: for first difference, this is 0  
baseline <- linesTile(x=c(18,94),  
                      y=c(0,0),  
                      plot=1)
```

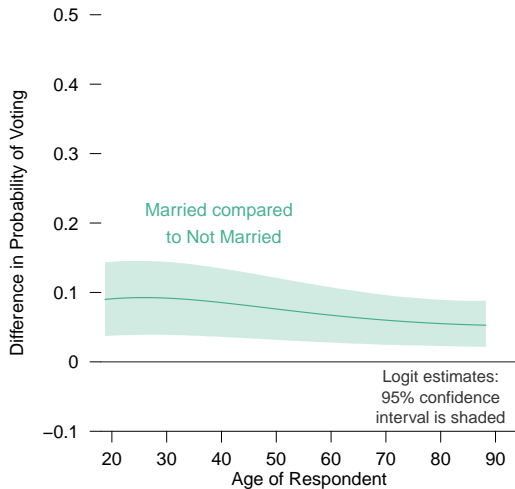

4. Simulating first differences

```
# Set up traces with labels and legend
labelFDTrace <- textTile(labels=c("Married compared \n to Not Married"),
                        x=c( 40),
                        y=c( 0.20),
                        col=col[1],
                        plot=1)

legendFDTrace <- textTile(labels=c("Logit estimates:", "95% confidence", "interval is shaded"),
                        x=c(80, 80, 80),
                        y=c(-0.02, -0.05, -0.08),
                        cex=0.9,
                        plot=1)

# Plot traces using tile
tile(marriedFDTrace,
     labelFDTrace,
     legendFDTrace,
     baseline,
     limits=c(18,94,-0.1,0.5),
     xaxis=list(at=c(20,30,40,50,60,70,80,90)),
     yaxis=list(label.loc=-0.5, major=FALSE),
     xaxis.title=list(labels="Age of Respondent"),
     yaxis.title=list(labels="Difference in Probability of Voting"),
     width=list(null=5,yaxis.title=4,yaxis.labelspace=-0.5)
#, output=list(file="marriedFD", width=5.5)
)
```

4. Simulating first differences



4. Simulating first differences

```
# Plot Relative Risk

# Set up lineplot trace of relative risk
marriedRRTrace <- lineplot(x=xhyp,
  y=marriedRR$pe,
  lower=marriedRR$lower,
  upper=marriedRR$upper,
  col=col[1],
  extrapolate=list(data=mdata2[,2:ncol(mdata2)],
    cfact=marriedScen$x[,2:ncol(marriedScen$x)],
    omit.extrapolated=TRUE),

  plot=1)

# Set up baseline: for relative risk, this is 1

baseline <- linesTile(x=c(18,94),
  y=c(1,1),
  plot=1)
```

4. Simulating first differences

```
# Set up traces with labels and legend
labelRRTrace <- textTile(labels=c("Married compared \n to Not Married"),
                        x=c( 55),
                        y=c( 1.25),
                        col=col[1],
                        plot=1)

legendRRTrace <- textTile(labels=c("Logit estimates:", "95% confidence", "interval is shaded"),
                        x=c(80, 80, 80),
                        y=c(0.98, 0.95, 0.92),
                        cex=0.9,
                        plot=1)

# Plot traces using tile
tile(marriedRRTrace,
     labelRRTrace,
     legendRRTrace,
     baseline,
     limits=c(18,94,0.9,1.5),
     xaxis=list(at=c(20,30,40,50,60,70,80,90)),
     yaxis=list(label.loc=-0.5, major=FALSE),
     xaxis.title=list(labels="Age of Respondent"),
     yaxis.title=list(labels="Relative Risk of Voting"),
     width=list(null=5,yaxis.title=4,yaxis.labelspace=-0.5)
#, output=list(file="marriedRR",width=5.5)
)
```

4. Simulating first differences

