UNIVERSITY OF MANNHEIM
School of Social Sciences

Advanced Quantitative Methods in Political Science: Interpretation and Simulation

Thomas Gschwend | Oliver Rittmann | Viktoriia Semenova

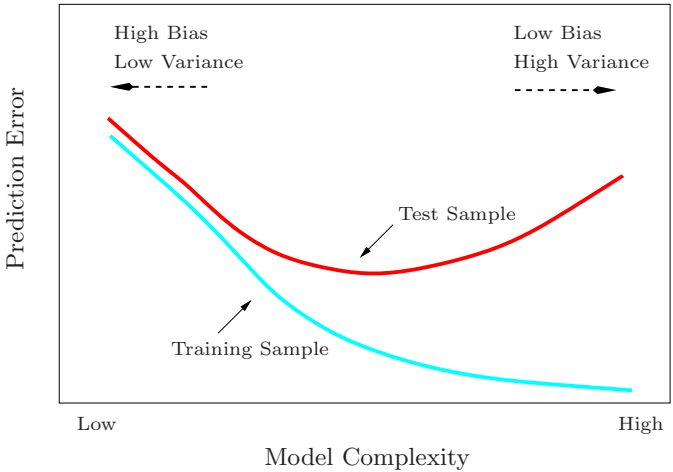
Week 7 - 30 March 2022

Leftovers from last week: Fit Measures for Binary Dependent Variable Models

How to know which model is better?: Out-Of-Sample Forecasts

- Key requirement: Find the *systematic* rather than idiosyncratic features of any one data set (although you only have one draw, i.e., one data set).
- Set aside some (random) parts of the data (aka as *test data*) and fit your model to the rest (aka *training data*)
- Make predictions with training data and compare to the test data.
  - Compare average predictions and also full distribution
  - Say, for a given scenario you predict $Pr(y = 1) = 0.2$ using the *training* data, then 20% of such observations should actually be observed as $y = 1$ in the *test* data.
- Gold standard is test data that is really out-of-sample, i.e. not yet available.
- Of course, you need to assume that test and training data generated from the very same data-generating process. Thus, if the DGP changes between the time training and test data are observed... tough. Even a good model will fail.
- Still, out-of-sample forecast is the right test for any model.

Can We Stop Ourselves? On the Danger of Over-fitting



Taken from: Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2008. *The Elements of Statistical Learning* (2nd edition). Chapter 2: Fig 2.11, p. 38.

Fit Measures for Binary Variable Predictions

- Classify correctly predicted observations (for chosen cut-point at .5)
  - Using $\hat{\beta}$ from your model, generate predicted probabilities $\hat{\pi}_i$.
  - Generate variable of predicted values $\hat{y}_i = 1$ if $\hat{\pi}_i \geq 0.5$, 0 otherwise.
- Generate 2x2 classification table (aka *confusion matrix*).

	Predicted ($\hat{y}_i$)	
Observed (y_i)	0	1
0	n_{00}	n_{01}
1	n_{10}	n_{11}

- From this, we can construct *Percent Correctly Predicted* (PCP):

$$PCP = \frac{n_{00} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}}$$

- If, say, the DV is distributed 70 : 30, then a model (to beat) without independent variables would predict 70% of the cases correctly.
- Problems: (a) Uncertainty? (b) Precision: $\hat{\pi}_i = .51$ and $\hat{\pi}_j = .99$ are counted equally

Other Fit Measures for Binary Variable Predictions

- *Percent Reduction in Error* (PRE)
  - Classify correctly predicted observations relative to a baseline
  - Baseline is the Percent of observations in the Modal Category (PMC) of the dependent variable.

$$PRE = \frac{PCP - PMC}{1 - PMC}$$

  - *PRE* is just a function of *PCP*, thus, still the precision problems.
- *expected Percent Correctly Predicted* (ePCP)
  - Expected percentage of correct model predictions (Herron 1999 - PA article)

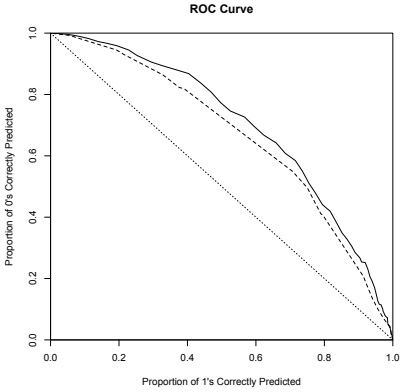
$$ePCP = \frac{1}{N}\left(\sum_{y_i=1}\hat{\pi}_i + \sum_{y_i=0}(1-\hat{\pi}_i)\right)$$

- All such classification-based measures focus on a model's ability to classify observations. No specification test, though (see Esarey and Pierce 2012)!
  - Thus, a good model fit (e.g., high PCP) *does not* imply a correct model specification.

Model Selection using ROC

- *Problem:* Classifications require a normative decision.
  - Let C be the number of times it is more costly classifying a 1 than a 0.
  - C must be chosen independently of the data; from review of literature, (survey of) policy makers
  - $C = 1$ often chosen, but without justification
- *Decision Theory:* Choose $Y = 1$ when $\hat{\pi} > 1/(1 + C)$ and 0 otherwise.
  - If $C = 1$, predict $y = 1$ when $\hat{\pi} > 0.5$ (as for PCP, PRE, ePCP)
  - If $C = 2$, predict $y = 1$ when $\hat{\pi} > 1/3$
  - Increasing C reduces chances of type I error ("false alarm")
  - If $C \to 0$ then $\hat{\pi} \to 1$, and if $C \to \infty$ then $\hat{\pi} \to 0$
- Only with chosen C it makes sense to compute (a) % of 1s and 0s correctly predicted, and (b) error patterns in different subsets of the data (or forecast)
- If you cannot justify *a priori* a value for C, use all of them! Plot ROC (receiver-operator characteristics) curves

ROC Curves



Taken from the `demo(roc)` in the `library(Zelig)` (see `help.zelig(logit)`)

- Compute % 1s and % 0s correctly predicted for every possible value of *C*.
- Plot % 1s by % 0s
- Overlay curve for several model specifications on the same graph.
- Normative decision about *C* does not matter if one curve is above another. We then say that one model *dominates* the other.
- Otherwise, one model (specification) is better than another in specified ranges of *C*.
- In R use e.g, `library(Zelig)` or `library(epicalc)`

Further Model Fit, Specification and Robustness Checks

- *Cross-Validation* (for all types of models)
  - Randomly divide the data set into K equally sized folds (each fold will contain about $\frac{N}{K}$ observations)
  - Train model K-times on all but the k-th fold ($k \in \{1, \ldots, K\}$), then use k-th fold to estimate model on unseen data. Average across the K results.
  - Useful for smaller data sets where one cannot set aside test data
  - What does "average results" imply? Point estimates are the mean of the estimated point estimates of the subsets.
  - Standard errors should account for *within* as well as *across* variance (see King et al. 2001. "Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation". *American Political Science Review* 95: 49 – 69, equation (3))
- *Repeated random sub-sampling validation* (unobs. heterogeneity)
  - Sample 2/3 of data, run model and collect results. Repeat several (about $m = 20$) times for different samples and combine results per King et al 2001 (aka "Rubin Rule", see above).
- Confront (all) *observable implications* with your observations.

How to get “average results” across m data sets?

- Average point estimates of your quantity of interest q across m sets of estimates

$$\bar{q} = \frac{1}{m} \sum_{j=1}^{m} q_j$$

- Standard errors should account for *within* as well as *across* variance

$$SE(\bar{q})^2 = \frac{1}{m} \sum_{j=1}^{m} SE(q_j)^2 + S_q^2 (1 + \frac{1}{m})$$

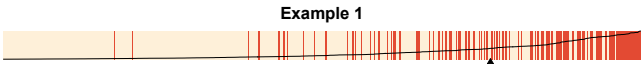
with $S_q^2 = \sum_{j=1}^{m} (q_j - \bar{q})^2 / (m-1)$

Likelihood-Based Approaches

- Evaluation of model fit through any test statistic that is based on a transformation of the log-likelihood will be a *relative* measure of model fit (e.g., LRT)
- Akaike Information Criterion: $AIC = -2 \cdot lnL + 2p$
  - where p is the number of parameters in the statistical model, and L is maximum of the likelihood function for given model.
  - Pick the model among the possible ones with minimum *AIC* value. There is no statistical test of difference in *AIC*.
  - The penalty term ($2p$) does discourage overfitting while rewarding goodness of fit (because of *LL*).
- Bayesian Information Criterion: $BIC = -2 \cdot lnL + p \cdot ln(N)$
  - where N is the number of observations.
  - Larger penalty term ($p \cdot ln(N)$).
- *AIC* and *BIC* work even for non-nested models. Further examples are Vuong test, Bayes factors,....

Assessing Model Fit graphically - Separation Plot

Brian Greenhill, Michael D. Ward, Audrey Sacks. 2011. “The Separation Plot: A New Visual Method for Evaluating the Fit of Binary Models” *American Journal of Political Science*, 55(4): 991-1002.



- Graph fitted values with different colors for each observed outcome.
- Line indicates the predicted probabilities of the observations
- Helpful for identifying clusters of false negatives and false positives (systematic or coding errors)
- Can be used for models with more than two categorical outcomes!
- In R use e.g, `library(separationplot)`

Intro

What should you take home from this class today?

- We will get to learn tools to improve the interpretation of our results
- Simulations will be our friends
- We will repeat how to simulate *quantities of interest* and apply these tools to a logit example.
- There are two conceptual approaches for defining interesting scenarios: *Average-Case* vs *Observed Value*

Improving Interpretation through Simulating Quantities of Interest

Substantive Interpretation in Non-Linear Models

- Interpreting linear regression coefficients is straightforward because, the effect on Y of a given change in X_j is the same regardless ...
  - ...of the value of that variable
  - ...of the level of all other independent variables in the model
- "Holding all other variables constant, ..."
- This is not so in non-linear models!
  - Neither marginal (or discrete) changes with respect to X_j are constant. They are no longer simply equal to a parameter (β_j).
  - The effect on Y of a given change in X_j depends *on values of all other variables* in the model.
- Thus, we need another strategy for interpretation and need to invest more time in presenting and interpreting our results.

What Grandma doesn't get - The Ugly Table

Table 1: Logistic Regression Predicting Vote Switching

	Vote switching (= 1)
Small Party	0.998
	(0.228)
Ideology	-7.623
	(3.078)
Candidate	0.806
	(0.214)
Coalition	2.845
	(0.340)
Time	-0.395
	(0.133)
Constant	3.474
	(2.487)
Observations	987

Robust standard errors in parentheses

Criteria for Substantive Interpretation

- Convert raw results in quantities of substantive interest
- Convey uncertainty about those quantities
- Avoid statistical jargon

General Principles

1. Communicate Substance, Not Statistics
2. When Performing Inference, Convey Uncertainty
3. Graph Data and Results

Communicate Substance, Not Statistics

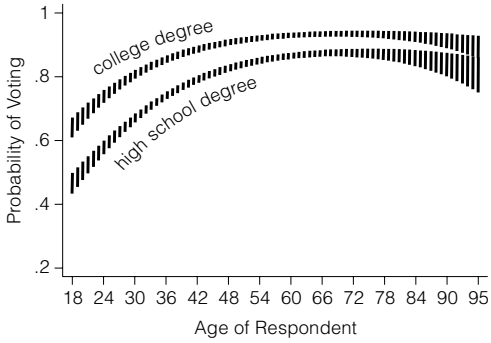
- The coefficient for *Small Party* (.998) is statistically significant
- Other things being equal, for supporters of a small party the likelihood of switching their vote is about 20 percent. That probability decreases to about only 8 percent for large party supporters.

When Performing Inference, Convey Uncertainty

- The coefficient for variable *Small Party* (.998 with a standard error of .23) is statistically significant at the .001 level.
- Other things being equal, for supporters of a small party the likelihood of switching their vote is about 20 (+/- 6) percent. That probability decreases to about only 8 (+/- 2) percent for large party supporters.

Graph Data and Results - Example 1

FIGURE 1 Probability of Voting by Age



Vertical bars indicate 99-percent confidence intervals

Conceptually three Steps

1. Define a *Quantity of Interest*
   - *Predicted values* $Y|X$ of switching a vote for interesting, "typical" (i.e., mean) or observed values of X
   - *Expected values* $E(Y|X)$
   - *First differences* $E(Y|X_{j_1}, X) - E(Y|X_{j_2}, X)$, the difference of two expected values (e.g., size of causal effect)
   - *Average (causal) effect*: compute (causal) effect for every observation and then average across them
   - any other quantity
2. Simulate quantity of interest (QoI) and uncertainty around it
3. Visualize results for interesting scenarios (potentially across all values of a key independent variable)

How to simulate Quantities of Interest?

Strategy for Substantive Interpretation

1. We get QoI as a function of estimated coefficients
2. Where is the uncertainty in a statistical model?
3. Use simulation to account for estimation and fundamental uncertainty
4. Create plots and tables for communicating your results

Where is the Uncertainty?

Recall that we can write any statistical model as

$$
\begin{aligned}
Y_i &\sim f(y_i|\theta_i, \alpha) && \text{stochastic} \\
\theta_i &= g(X_i, \beta) && \text{systematic}
\end{aligned}
$$

1. Estimation Uncertainty: Uncertainty about what the true parameters β and α of the model are. Think of it as caused by small samples. Vanishes if N gets larger.
2. Fundamental Uncertainty: Represented by stochastic component of the model. Exists no matter what (even if model is correct and we would have infinite many observations) because of inherent randomness of the world.

Simulating Estimation Uncertainty

We account for estimation uncertainty by taking random draws from the approximated "sampling distribution" of all parameters

- Step 1: Estimate the model by maximizing the likelihood function (as a canned procedure or using `optim()` in R), record the point estimates $\hat{\gamma}$ of all parameters $\gamma = vec(\beta, \alpha)$ and the estimated variance matrix $\hat{V}(\hat{\gamma})$.
- Step 2: Draw one vector $\tilde{\gamma} = vec(\tilde{\beta}, \tilde{\alpha})$ from the multivariate normal distribution (representing estimation uncertainty because of CLT), which approximates the sampling distribution we do not have

$$\gamma \sim N\left(\hat{\gamma}, \hat{V}(\hat{\gamma})\right)$$

  One draw $\tilde{\gamma}$ is also called *simulated value*.

Logit implementation in R as a running Example

```
ll.logit <- function(theta, y, x) {

# theta consists merely of beta (dim is ncol(X))
  beta   <- theta[1:ncol(X)]
# linear predictor; make sure that X is stored as.matrix
  mu <- X %*% beta
# link function
  p <- 1/(1+exp(-mu))
# individual ll contribution
  ll <- y*log(p) + (1-y)*log(1-p)
# sum
  ll <- sum(ll)
  return(ll)
 }
```

Logit Implementation in R as a running Example

```
# maximize the likelihood function numerically using optim()
res <- optim(c(1,1),                # starting values
         fn=ll.logit,               # the likelihood function
         control=list(fnscale=-1),  # maximize instead of minimize fct
         y =dat$survived, x = X,    # the data
         method = "BFGS",           # optimization method
         hessian=TRUE)              # return numerical Hessian

cat("MLE Betas\n", res$par, "\n")
cat("Hessian\n")
print(res$hessian)
cat("\nMLE Standard Errors \n", sqrt(diag(solve(-1 * res$hessian))), "\n\n")

# compare with canned logit in R (using standard GLM function)
summary(glm(dat$survived ~ dat$sex, family="binomial"))

# Or use Zelig (but version 3.5.5 or below, or version 5 and above)
summary(zelig(survived~sex, model="logit", data=titanic3))
```

1. Predicted Values

We can simulate the distribution of predicted values in the following way:

- To simulate *one* predicted value, follow these steps:
  - Step 1: Draw one vector $\tilde{\gamma} = vec(\tilde{\beta}, \tilde{\alpha})$.
  - Step 2: Decide which scenario you wish to compute (i.e., simulate) and, thus, choose one value for *each* explanatory variable in the model (why?). Denote the vector of such values that defines your scenario with X_c.
  - Step 3: Run those values (from step 1 and 2) through link function $g(X_c, \tilde{\beta})$ of systematic component. Denote it by $\tilde{\theta}_c$.
  - Step 4: Finally, to get a predicted value of the chosen scenario $\tilde{Y}_c$, take one random draw from, $f(\tilde{\theta}_c, \tilde{\alpha})$, the stochastic component of the statistical model, to simulate fundamental uncertainty.
- To simulate say $M = 1000$ predicted values, repeat algorithm 1000 times. Use these to plot distribution of simulated predicted values or compute average, standard deviation, percentile values (or anything you want) to summarize this distribution.

We can simulate the distribution of expected values similarly:

- To simulate *one* expected value, follow steps 1-4 to generate m predicted values and, then, "average out" fundamental uncertainty (stochastic component) by taking the expectation of these.
  - Step 1: Draw one vector $\tilde{\gamma} = vec(\tilde{\beta}, \tilde{\alpha})$ to account for estimation uncertainty.
  - Step 2: Decide which scenario you wish to simulate and choose X_c, i.e., fix each explanatory variable at particular values.
  - Step 3: Run those values (from step 1 and 2) through link function $g(X_c, \tilde{\beta})$ to get $\tilde{\theta}_c$.
  - Step 4: Draw m values $\tilde{Y}_c^{(k)}$ (with $k = 1, \ldots, m$) from the stochastic component $f(\tilde{\theta}_c, \tilde{\alpha})$ to simulate fundamental uncertainty.
  - Step 5: Average over fundamental uncertainty (stochastic component) by calculating the mean of those m simulations to get one *simulated expected value* $E(Y_c) = \sum_{k=1}^{m} \tilde{Y}_c^{(k)}/m$.
- To simulate, say, $M = 1000$ expected values, repeat algorithm 1000 times. Use these to plot distribution of simulated predicted values or compute an average, std. dev., certain percentile values (or anything you want) to summarize this distribution.

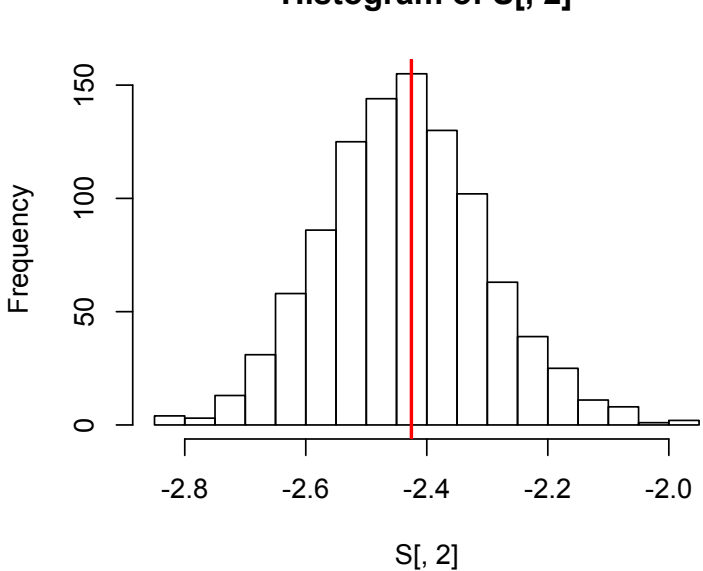
Logit Implementation in R as a running Example

```
# ==========================================
# = Step 1: Simulate Estimation Uncertainty =
# ==========================================

# get gamma and V from optim()
gamma <- res$par
    V <- solve(-res$hessian)

library(MASS)                                # provides mvrnorm
nsim <- 1000                                 # N simulations
set.seed(1234)                               # to replicate draws
S <- mvrnorm(nsim, mu = gamma, Sigma = V) # simulations of gamma

dim(S)
# look at simulated distribution in contrast to
# estimated beta coeff (for men)
hist(S[,2], breaks=20)
abline(v=gamma[2], col="red", lwd=2)
```


Histogram of S[, 2]
Frequency
0
50
100
150
-2.8
-2.6
-2.4
-2.2
-2.0
S[, 2]

Logit Implementation in R as a running Example

```
# ================================
# = Step 2: Choose Scenario X_c  =
# ================================

#  set e.g. all covariates to the sample mean
# mean will be applied over columns of X
# Check ?apply for descriptions of all other arguments.
  X_c <-  apply(X, 2, mean)

# ======================================
# = Step 3: Get lp_c (linear predictor)  =
# ======================================

  lp_c<- S %*% X_c
# run it through link-function
  theta_c <- 1/(1+exp(-lp_c))
# summarize simulation results
  mean(theta_c); sd(theta_c)
  quantile(theta_c, c(.025, .975)
  hist(theta_c, breaks=20)
```

Logit Implementation in R as a running Example

```
# ============================================
# = Step 4: Simulate Fundamental Uncertainty  =
# ============================================

# create empty vector to store random draws
  Y_c<-rep(NA,nsim)

  set.seed(1212)    # to be able to replicate draws
  for (i in 1:nsim){
  Y_c[i]<-rbinom(n = 1, size = 1, prob = theta_c[i])
  }

  table(Y_c)

# ================================================
# = Step 5: Average over Fundamental Uncertainty  =
# ================================================

  mean(Y_c)
```

Difference between Predicted and Expected Values

- Predicted values are draws of Y from the stochastic component that are observed or could be observed (e.g., $\hat{Y} = 1$ or 0)
- Expected values are draws from fixed features of the distribution of Y such as $E(Y)$. Not necessarily observed (e.g., $\pi_i = .6$).
- Predicted values include estimation and fundamental uncertainty
- Expected values average over fundamental uncertainty. The variance of the distribution of expected values (but not of predicted values) go to 0 as N gets large.
- Example use of the distribution of *predicted values*
  - Weather Forecast: Pred. prob. that temperature in Mannheim tomorrow drops below 0^oC.
  - Predicted temperature uncertain because we estimate it and because of natural fluctuations.
- Example use of the distribution of *expected values*
  - Pred. prob. that temperature in Mannheim *on days like tomorrow* drops below 0^oC.
  - Expected temperature is only uncertain because we have to estimate it.
  - We are more certain about expected than actual temperature because we do not care about natural fluctuations.

Further Remarks on Simulating Expected Values

- When $m = 1$ the algorithm to get expected values reduces to the one for predicted values.
- The larger m the better does the algorithm purge away of fundamental uncertainty.
- When $E(Y_c) = \theta_c$, one can skip step 4 and 5. For instance, in the logit model (whenever expected value equals predicted prob. such as for Bernoulli processes), once we have simulated π_i, we do not need to draw Y_c and than average them to get $E(Y_c) = \pi_i$.

```
# results from step 5 (average over fundamental uncertainty)
  mean(Y_c)
  [1] 0.3598

# compare results if we had stopped with step 3
  mean(theta_c)
  [1] 0.3589098
```

3. First Differences

We can simulate the distribution of first differences using the same logic as before for simulating expected (as well as predicted) values:

- To draw *one* simulated first difference, follow the following steps.
  - Step 1: Draw one vector $\tilde{\gamma} = vec(\tilde{\beta}, \tilde{\alpha})$.
  - Step 2: Decide which contrast you wish to simulate and choose X_s and X_c, i.e., fix each explanatory variable at particular values.
  - Step 3: Run those values (from step 1 and 2) through link function $g(X_c, \tilde{\beta})$ to get $\tilde{\theta}_c$. Analogously, run $g(X_s, \tilde{\beta})$ to get $\tilde{\theta}_s$.
  - Step 4: Apply the expected value algorithm twice, once for X_s and X_c (we can reuse random draws).
  - Step 5: Take the difference between the two expected values.
- To simulate say $M = 1000$ first differences, repeat algorithm 1000 times. Use these to plot distribution of simulated first difference values, standard deviation, percentile values (or anything you want) to summarize this distribution.

Logit Implementation in R as a running Example

```
# ========================================
# = first differences male vs female     =
# ========================================

# Step 1
# Use simulations of mvrnorm from before

# Step 2
# construct scenario of interest

# set everything to sample mean
  X_1 <- X_2 <- apply(X, 2, mean)

# Now define contrast
  X_1[2] <- 0 # men=0 for female
  X_2[2] <- 1 # men=1 for male
```

Logit Implementation in R as a running Example

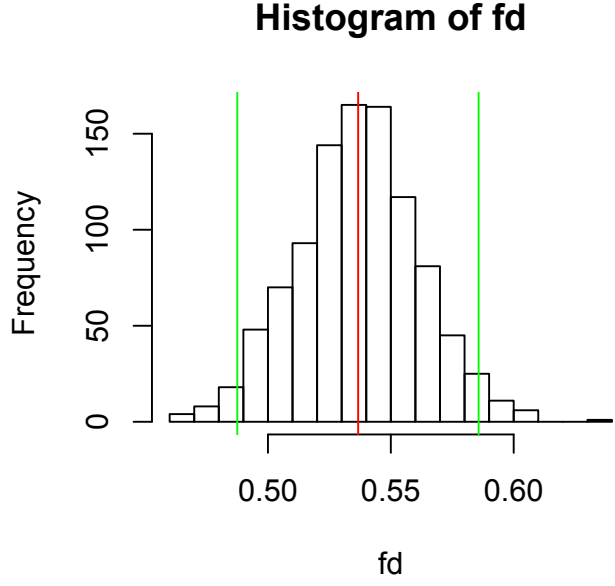
```
# Step 3 (combining step 4 and 5)
# Get theat_c - the linear predictor

   lp_1 <- S %*% X_1
    ev1 <- 1/(1+exp(-lp_1))

   lp_2 <- S %*% X_2
    ev2 <- 1/(1+exp(-lp_2))

# calculate first-difference
# fd = ev(female) - ev(male)
     fd <- ev1 - ev2

mean(fd); sd(fd)
quantile(fd, c(.025, .975))
```


Histogram of fd
Frequency
0
50
100
150
0.50
0.55
0.60
fd

Construction of the Scenario of Interest

There are two conceptual approaches: *Average-Case* vs *Observed Value*

1. So far we have seen examples of the so-called *average-case* approach to define scenario of interest.
   - Set key IV to interesting value, and the other to average (typical) values.
   - Note that those scenarios are hypothetical — they might not actually exist in your data (inference is model dependent!)
   - Nevertheless helpful for theory tests because we often do not hypothesize about an "average case" but about specific conditions under which a relationship holds.
2. The so-called *observed-value* approach is an alternative
   - Provides average marginal effect (AME), i.e. average of the individual marginal effects of the expected change in the probability for a unit change in the key variable of interest
   - Used to calculate average treatment effect (ATE), the average of first-difference when unit gets (hyp.) treatment and when it is not, or more generally the marginal effect of a randomly picked observation.
   - But mean (average) of the marginal effects (AME) $\neq$ marginal effect at the mean (MEMs).

It's problematic to calculate counterfactuals that are "far away from the data".

Simulating Parameters - Some Useful Tips

1. Simulate all parameters including ancillary parameters together.
2. Reparameterize to unbounded scale to ...
   - ...make $\tilde{\gamma}$ converge more quickly in n to a multivariate normal. Thus, more reasonable for smaller sample size.
   - ...make maximization algorithm work faster and without explicit constraints.
3. Ideally, all parameters should be unbounded and logically symmetric, e.g.,
   - for non-negative parameters: $\sigma^2 = e^{\eta}$
   - for $0 - 1$ bounded parameters (e.g., probability): $\pi = \frac{1}{1+e^{-\eta}}$.
   - for $-1 \leq \rho \leq 1$, use $\rho = (e^{2\eta} - 1)/(e^{2\eta} + 1)$ (*Fisher's Z transformation*)

   In all three cases, η is unbounded. Reparameterize back to a scale people care about.

Simulating Parameters - More Useful Tips

- Always compute simulations of Y (i.e. predicted values) and use that as a basis for simulating other quantities (of interest).
- If you are interested in simulating functions of Y, say $ln(Y)$ do the following: Simulate $ln(Y)$ and then apply the inverse function $exp(ln(Y))$ to get Y. Y is probably on a meaningful scale we care about.
- Check *approximation error* of your simulation: Run it twice, check no. of digits of precision that do not change. If *not* enough precision for presenting results in a table, increase M (or m) and try again.
- Analytical calculations (e.g., delta method) and other tricks can speed-up simulation and increase precision.
- `Zelig` does support various models (including Bayesian!) in R.