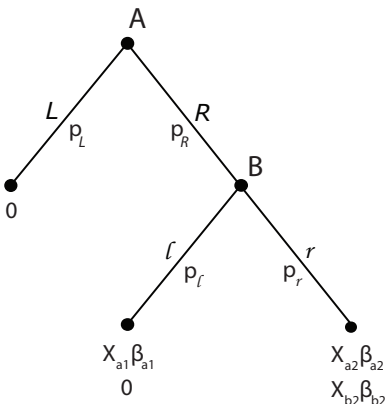


# Statistical Backwards Induction

- **STRAT**: Program for Estimating Statistical Strategic Models: 5-6 game forms in STRAT
- If not in **STRAT**, must program system estimator. Many without tools to do so
- Statistical Backwards Induction (SBI) Estimator (Bas et al (2007)):
  - Extremely simple to implement
  - logit or probit at each stage
  - transform variables
  - bootstrap whole process for standard errors

# The SBI Procedure



- Strategic model as a recursive system of equations

$$y_A^* = p_l X_{a1} \beta_{a1} + p_r X_{a2} \beta_{a2} + \epsilon_A$$

$$y_B^* = X_{b2} \beta_{b2} + \epsilon_B$$

- $p_r = \Pr[y_B^* > 0]$  and  $p_R = \Pr[y_A^* > 0]$  will be logit or probit probabilities

## The Basic Procedure

- Recursive estimation is similar to backwards induction.
- Estimate the system equation-by-equation using standard logit/probit.
- Transform regressors in expected utility calculations into 'new' regressors that can be used in 'higher' information set logit estimations.

## The Basic Procedure

- 1. B's Choice.** Since  $y_B^* = X_{b2}\beta_{b2} + \epsilon_B$  does not require information concerning  $y_A^*$ , logit or probit may be used to estimate  $\hat{\beta}_{b2}$ . Once  $\hat{\beta}_{b2}$  is obtained, calculate  $\hat{p}_r$  and  $\hat{p}_\ell = 1 - \hat{p}_r$ .
- 2. A's Choice.** Substitute  $\hat{p}_\ell$  and  $\hat{p}_r$  into A's equation, giving

$$y_A^* = \hat{p}_\ell X_{a1}\beta_{a1} + \hat{p}_r X_{a2}\beta_{a2} + \epsilon_A$$

- create the transformed regressors

$$Z_{a1} = \hat{p}_\ell X_{a1} \quad \text{and} \quad Z_{a2} = \hat{p}_r X_{a2}$$

- Use logit or probit to estimate

$$y_A^* = Z_{a1}\beta_{a1} + Z_{a2}\beta_{a2} + \epsilon_A$$

# Advantages

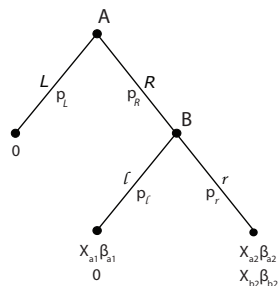
- Properties:
  - Consistent estimates of  $\hat{\beta}$ 's
  - Consistent estimates of std errors of coefficients for terminal node decisions (e.g., B's  $\hat{\beta}$ 's)
- Can be estimated using logit/probit in std stats packages
- Nicer likelihood function, easier convergence
- May make better use of (missing) data
- DISADVANTAGE: std errors of A's parameters are biased.

## Bootstrapped Standard Errors

- Given that you can implement the iterative procedure, write down sequence of commands and 'wrap it' in a bootstrap
- Bootstrap: M iterations
  - Draw sample with replacement from original data.
  - Calculate iterative estimator
  - Save A's  $\hat{\beta}$ 's
  - After the M iterations, s.e. ( $\hat{\beta}$ ) is standard deviation of saved  $\hat{\beta}$ 's
- NOTE: Bootstrap correction only necessary for the estimation of A's standard errors

## Bootstrapped Standard Errors

- Suppose our data set is structured as  
 $\text{Data} = [y_A, y_B, X_{b21}, X_{b22}, X_{a1}, X_{a21}, X_{a22}]$
- To calculate standard errors, we would run  
`boot(Data, SBI, 500)`
- This will print
  - Parameter estimates in A's equation using all the data
  - Estimate of the bootstrap 'bias', where  $\text{bias} = \text{mean}(\text{bootstrap estimates}) - \text{original estimate}$
  - Standard errors of bootstrap estimates



$$\text{Data} = [y_A, y_B, X_{b21}, X_{b22}, X_{a1}, X_{a21}, X_{a22}]$$

```
SBI ← function(Data, ind){
  BSdata ← Data[ind,]      # Draw the bootstrap sample
  YA ← BSdata[,1]         # A's choices
  YB ← BSdata[,2]         # B's choices; unrealized=NA
  XB ← BSdata[,3:4]       # B's regressors

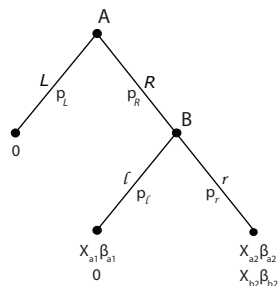
  # Step 1: Logit to estimate B's utilities and calculate  $\hat{p}_r$ 
  logitB ← glm(YB ~ XB, family=binomial(link="logit"))
  bB ← logitB$coefficients
  XB ← cbind(1, XB)
  pr ← exp(XB %*% bB) / (1 + exp(XB %*% bB))

  # Step 2: Transform A's regressors with  $\hat{p}_r$ 
  XA1 ← BSdata[,5]
  XA2 ← BSdata[,6:7]
  ZA1 ← (1-pr)*XA1
  ZA2 ← cbind(pr*XA2[,1], pr*XA2[,2])

  # Step 3: Logit for A's utilities
  logitA ← glm(YA ~ ZA1+ZA2-1, family=binomial(link="logit"))
  logitA$coefficients
}

# Step 4: Calculate standard errors with bootstrapping

boot(DATA, SBI, 500)
```



```
capture program drop sbi
program sbi, rclass
```

\* Step 1: Run a logit to estimate B's utilities and calculate  $\hat{p}_r$

```
logit YB XB21 XB22 if YA
predict pr
```

\* Step 2: Transform A's regressors with  $\hat{p}_r$

```
gen ZA21 = XA21*pr
gen ZA22 = XA22*pr
gen ZA1 = XA1*(1-pr)
```

\* Step 3: Logit for A's utilities

```
logit YA ZA21 ZA22 ZA1, nocons
logitA = e(b)
drop pr ZA1 ZA21 ZA22
```

```
ret scalar XA21 = logitA[1,1]
ret scalar XA22 = logitA[1,2]
ret scalar XA1 = logitA[1,3]
```

```
end
```

\* Step 4: Calculate standard errors with bootstrapping

```
bootstrap "sbi" r(XA21) r(XA22) r(XA1), reps(500) dots
```