Hint: the following packages will be useful in solving this prolem set.

In [2]:
```
using Optim
using Statistics
using ForwardDiff
using Plots
using LinearAlgebra
```

Problem Set 2

Due: May 3, 2021 (in class; subject to change if COVID restrictions apply)

A binary response is a variable that takes on only two values, customarily 0 and 1, which can be thought of as codes for whether or not a condisiton is satisfied. For example, 0=drive to work, 1=take the bus. Often the observed binary variable, say y, is related to an unobserved (latent) continuous varable, say y^*. We would like to know the effect of covariates, x, on y. The model can be represented as

$$\begin{aligned} y^* &= g(x) - \varepsilon \\ y &= 1(y^* > 0) \\ Pr(y=1) &= F_\varepsilon[g(x)] \\ &\equiv p(x,\theta) \end{aligned}$$

For the logit model, the probability has the specific form

$$p(x,\theta) = \frac{1}{1+\exp(-x'\theta)}$$

Problem 1 (MLE)

We will consider maximum likelihood estimation of the logit model for binary 0/1 dependent variables. We will use the BFGS algorithm to find the MLE.

The log-likelihood function is

$$s_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln p(x_i,\theta) + (1-y_i)\ln[1-p(x_i,\theta)]\right)$$

The following code generates that follow a logit model with given θ:

In [4]:
```
function LogitDGP(n, theta)
    k = size(theta,1)
    x = ones(n,1)
    if k>1
        x = [x  randn(n,k-1)]
    end
    y = Float64.((1.0 ./ (1.0 .+ exp.(-x*theta)) .> rand(n,1)))
    return y, x
end
```

```
LogitDGP (generic function with 1 method)
```

Out[4]:

In [6]:

```
n=30
theta = [0.75,0.25]
k = size(theta,1)
x = ones(n,1)
if k>1
    x = [x  randn(n,k-1)]
end
```

Out[6]:

```
30×2 Array{Float64,2}:
 1.0   0.168835
 1.0  -2.07858
 1.0  -0.785223
 1.0   0.707707
 1.0  -0.269753
 1.0   0.030736
 1.0   0.773801
 1.0  -2.3563
 1.0  -0.456262
 1.0  -0.40472
 1.0   0.41977
 1.0  -1.28736
 1.0  -0.00203532
 ⋮
 1.0   0.744194
 1.0  -1.80486
 1.0   0.380669
 1.0  -0.165739
 1.0  -1.63143
 1.0  -0.676599
 1.0  -0.419706
 1.0   0.0284142
 1.0   0.830083
 1.0  -0.856151
 1.0   0.641234
 1.0  -0.456509
```

In [8]:

```
1.0 ./ (1.0 .+ exp.(-x*theta))  .>  rand(n,1)
```

Out[8]:

```
30×1 BitArray{2}:
 1
 0
 1
 1
 1
 1
 1
 1
 0
 1
 0
 0
 0
 ⋮
 1
 0
 1
 1
 1
 1
 1
 1
```

```
0
0
1
0
```

In [9]:
```
[1.0 ./ (1.0 .+ exp.(-x*theta)) rand(n,1)]
```

Out[9]:
```
30×2 Array{Float64,2}:
 0.688305  0.698902
 0.557336  0.0716197
 0.634992  0.559135
 0.716451  0.668199
 0.66431   0.993692
 0.680851  0.249511
 0.719796  0.936081
 0.540145  0.0730981
 0.653834  0.785307
 0.656745  0.404449
 0.701603  0.967186
 0.605434  0.113134
 0.679068  0.545709
 ⋮
 0.718301  0.486471
 0.574146  0.286721
 0.699552  0.732547
 0.670084  0.501389
 0.584711  0.897201
 0.641263  0.173494
 0.655899  0.86322
 0.680725  0.498898
 0.722625  0.101069
 0.630873  0.193564
 0.713063  0.30068
 0.65382   0.0584654
```

Let us estimate $\hat{\theta}$ from the dataset with 100 points ($n = 100$) and generated from the true θ (θ_0) value of $[0.5, 0.5]$.

In [10]:
```
n=100
theta = [0.75,0.25]
(y,x) = LogitDGP(n,theta)
```

Out[10]:
```
([1.0; 0.0; … ; 1.0; 0.0], [1.0 0.278557437317092; 1.0 0.09507474449420693; … ; 1.0 -0.9
180350890683213; 1.0 -0.16918013820719])
```

In [11]:
```
y
```

Out[11]:
```
100×1 Array{Float64,2}:
 1.0
 0.0
 1.0
 0.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 ⋮
 0.0
```

```
1.0
1.0
1.0
0.0
1.0
0.0
0.0
0.0
1.0
1.0
0.0
```

In [12]:
```
x
```

Out[12]:
```
100×2 Array{Float64,2}:
 1.0   0.278557
 1.0   0.0950747
 1.0  -0.0839943
 1.0  -0.138736
 1.0   1.35027
 1.0   1.83928
 1.0   1.17331
 1.0   0.403827
 1.0  -0.418006
 1.0   0.26167
 1.0   1.32698
 1.0  -0.48492
 1.0  -2.22801
 ⋮
 1.0  -0.39731
 1.0  -0.234709
 1.0  -0.302407
 1.0   0.859959
 1.0  -1.28018
 1.0  -1.02768
 1.0   0.188731
 1.0   0.927246
 1.0   1.00302
 1.0  -0.108748
 1.0  -0.918035
 1.0  -0.16918
```

(1. a) Estimate $\hat{\theta}$.

Hint:

1. Refer to Nerlove lecture notes for an example code for mle estimation.
2. Code for the log likelihood function

$$s_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i \ln p(x_i,\theta) + (1-y_i)\ln[1-p(x_i,\theta)])$$

   is written as below:

(1. b) Empirically prove consistency of $\hat{\theta}$ by increasing the number of n in DGP and re-estimate.

Hint: Refer to GMM lecture notes for an example code for empirically proving consistency.

(1. c) Empirically prove asymptotic normality of $\hat{\theta}$ by repeatedly generate data.

Hint: Refer to GMM lecture notes for an example code for empirically proving asymptotic normality.

Problem 2 (GMM)

Recall from GMM lecture notes:

Suppose the model is

$$y_t^* = \alpha + \rho y_{t-1}^* + \beta x_t + \epsilon_t$$
$$y_t = y_t^* + \upsilon_t$$

where ϵ_t and υ_t are independent Gaussian white noise errors. Suppose that y_t^* is not observed, and instead we observe y_t. If we estimate the equation

$$y_t = \alpha + \rho y_{t-1} + \beta x_t + \nu_t$$

this the estimator is biased and inconsistent.

What about using the GIV estimator?

Consider using as instruments $Z = [1 \; x_t \; x_{t-1} \; x_{t-2}]$. The lags of x_t are correlated with y_{t-1} as long as β is different from zero, and by assumption x_t and its lags are uncorrelated with ϵ_t and υ_t (and thus they're also uncorrelated with ν_t). Thus, these are legitimate instruments. As we have 4 instruments and 3 parameters, this is an overidentified situation.

In [238…

```
function lag(x::Array{Float64,2},p::Int64)
        n,k = size(x)
        lagged_x = [ones(p,k); x[1:n-p,:]]
end

function lag(x::Array{Float64,1},p::Int64)
        n = size(x,1)
        lagged_x = [ones(p); x[1:n-p]]
end


function  lags(x::Array{Float64,2},p)
        n, k = size(x)
        lagged_x = zeros(eltype(x),n,p*k)
        for i = 1:p
                lagged_x[:,i*k-k+1:i*k] = lag(x,i)
        end
    return lagged_x
end

function  lags(x::Array{Float64,1},p)
        n = size(x,1)
        lagged_x = zeros(eltype(x), n,p)
        for i = 1:p
                lagged_x[:,i] = lag(x,i)
        end
    return lagged_x
end
```

Out[238…

```
lags (generic function with 2 methods)
```

Given $[\alpha_0, \rho_0, \beta_0] = [0, 0.9, 1]$, let us generate data using the pre-defined lag function above:

In [280…

```
n = 100
sig = 1

x = randn(n) # an exogenous regressor
e = randn(n) # the error term
ystar = zeros(n)
# generate the dep var
for t = 2:n
  ystar[t] = 0.0 + 0.9*ystar[t-1] + 1.0*x[t] + e[t]
end
# add measurement error
y = ystar + sig*randn(n)
ylag = lag(y,1)
data = [y ylag x];
data = data[2:end,:] # drop first obs, missing due to lag
theta = [0, 0.9, 1]
```

Out[280…

```
3-element Array{Float64,1}:
 0.0
 0.9
 1.0
```

(2. a) Given the following GIVmoments function, write down the moment conditions for each data point. In other words, write down $m_t(\theta)\ \forall t$ where t is an index for each data points and $\bar{m_n}(\theta)$.

In [281…

```
# moment condition
function GIVmoments(theta, data)
        data = [data lags(data,2)]
    data = data[3:end,:] # get rid of missings
        n = size(data,1)
        y = data[:,1]
        ylag = data[:,2]
        x = data[:,3]
        xlag = data[:,6]
        xlag2 = data[:,9]
        X = [ones(n,1) ylag x]
        e = y - X*theta
        Z = [ones(n,1) x xlag xlag2]
        m = e.*Z
end
```

Out[281…

```
GIVmoments (generic function with 1 method)
```