POLS6382 Quantitative Method III
Maximum Likelihood Estimation
Lab 2: Normal Linear Model

Ling Zhu and Emily Lee
Department of Political Science
University of Houston

2025/09/17

1. Learning Objectives

- Learn how to implement grid search in R.
- Compare OLS and ML estimators: normal regression model.
- Consider heteroskedasticity in ML estimation (Appendix)

1.1 Grid Search

The purpose of a grid search is to calculate the likelihood for a range of possible parameter values. The two primary considerations are the range from low to high parameter values and the mesh which determines how many values will be evaluated within this range. A coarse mesh will produce a fast but jagged plot of the likelihood, while a fine mesh will take longer to calculate but can produce a maximum likelihood estimate of high precision. Often a grid search is set up to be run several times with a variable mesh, starting with a coarse mesh to narrow the range in which the maximum of the likelihood falls, then an increasingly fine mesh to find the maximum with considerable precision.

While these examples are not likely to be used in practice (where other numerical methods are much quicker and more precise) there are some cases in which grid searches are still used, particularly with "difficult" likelihoods possessing several maxima or with rough likelihood surfaces. The purpose of doing a grid search for elementary problems is to get a feel for how one might find the maximum likelihood estimator to as high a precision as desired using numerical methods alone. Later we will apply analytic techniques and more sophisticated maximization methods.

```
> rm(list=ls())
> # set up the work directory
> setwd("/Users/lingzhu/Dropbox/UH Teaching/POLS6382_2025/2025 Labs/Lab 2")
> require(maxLik)
```

1.2 Grid Search: A Bernoulli Example}

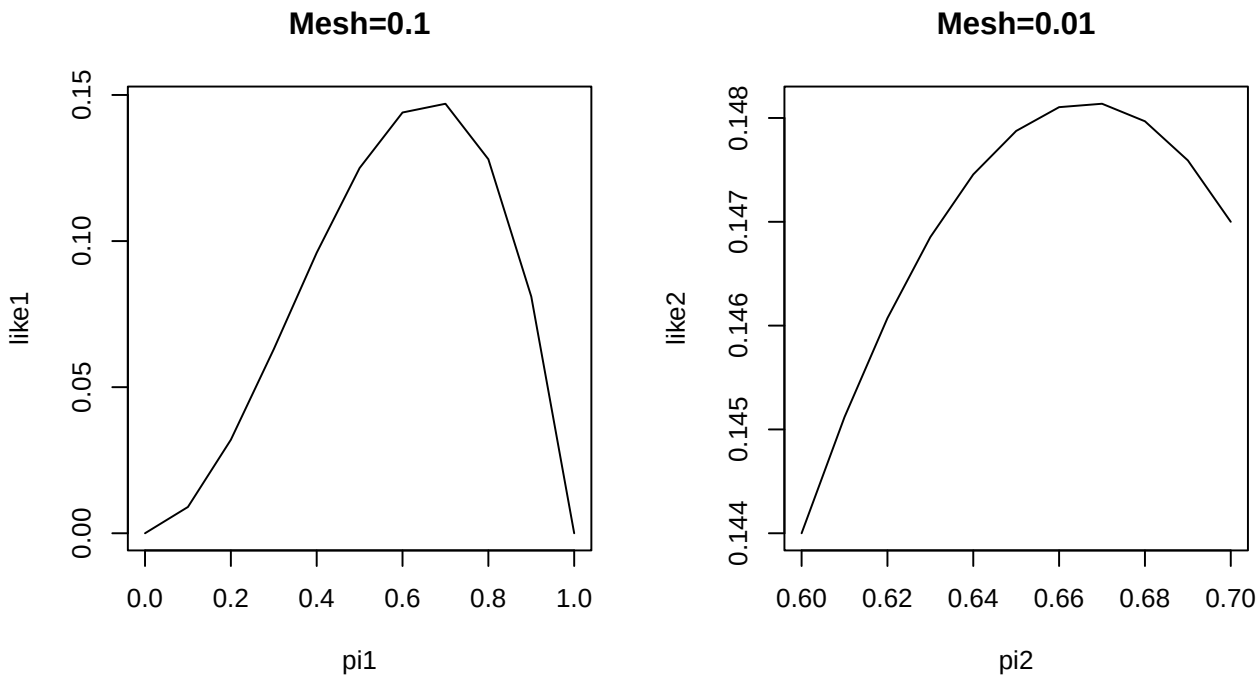
Imagine that you observe three independent outcomes distributed as a Bernoulli process in which two outcomes are successes (the respondents voted in the last election) and the third is a failure (the respondent did not vote.) If $\pi$is the probability of success, and hence $1-\pi$ is the probability of failure, and assuming the observations are independent, then our sample can be represented as $Y = 1, 1, 0$ and the likelihood of the sample is:

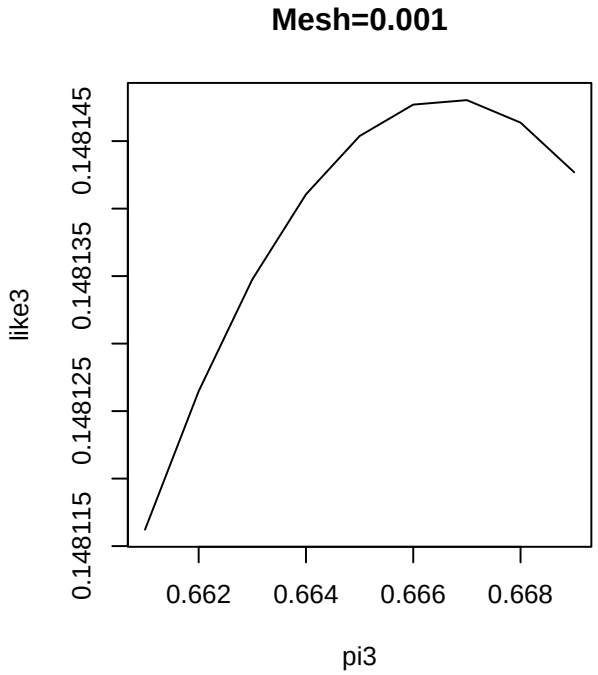
$$L(\pi|y) = \pi \times \pi \times (1-\pi) \tag{1}$$

Since this is a Bernoulli distribution the parameter π represents the probability of a success and so must range between 0 and 1. We start the grid search by setting the range to (0, 1) and a coarse grid of .1. This search shows that the maximum of the likelihood lies between values of π between about .6 and .7. We narrow the range to these values and set the mesh to .01 and repeat the grid search over these values. We narrow the range to these values and set the mesh to .01 and repeat the grid search over these values. Finally, we find that the maximum appears to fall between .661 and .669. We search this range with a mesh of .001. The final estimate of the value of πthat maximizes the likelihood is found to be .667 using the final mesh of .001. If we plot the three searches together, we will see that the first iteration explores a large parameter space of π. The subsequent search just focus on the high-density territory, while the last search gets us even closer to the top of the "hill".

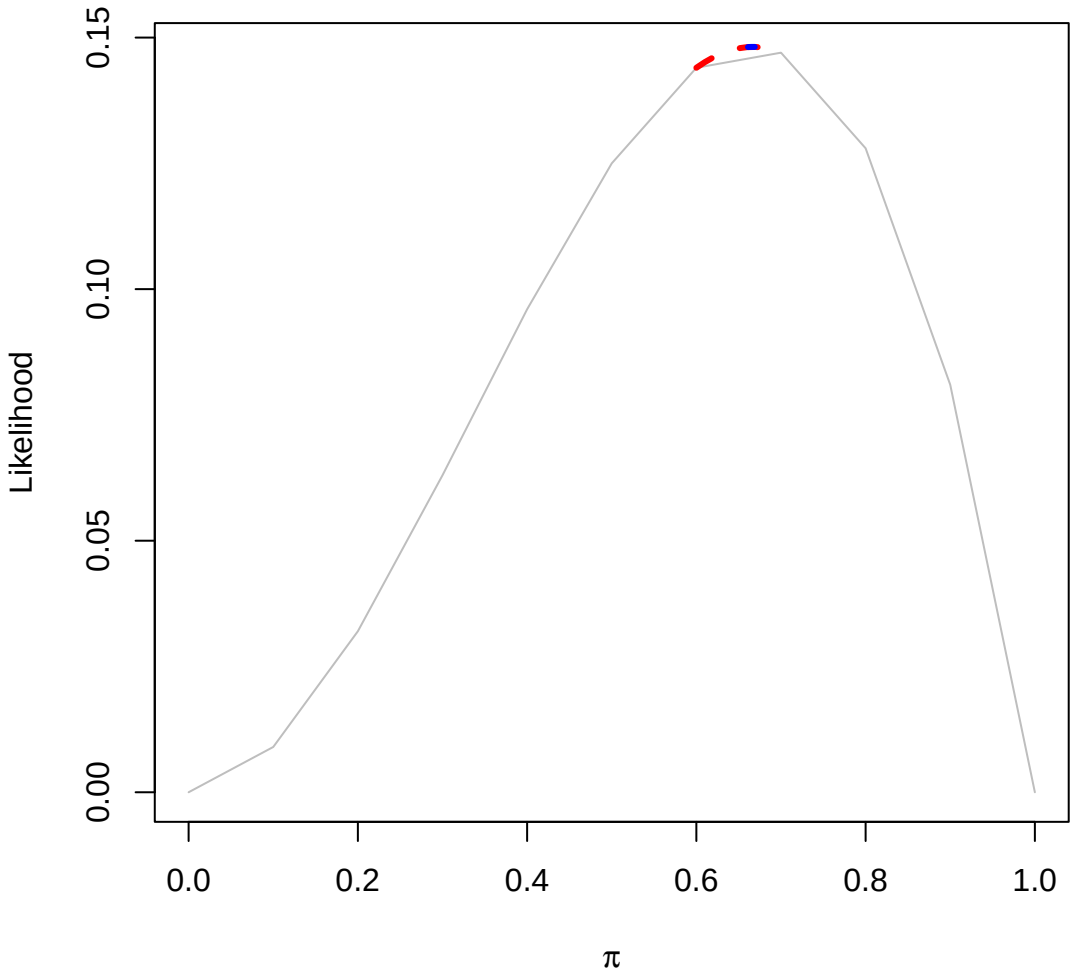
```
> # A Bernoulli Example
> # Search 1
>  pi1<-seq(0,1,.1)
>  like1<-pi1*pi1*(1-pi1)
> # Search 2
>  pi2<-seq(.6,.7,.01)
>  like2<-pi2*pi2*(1-pi2)
> # Search 3
> pi3<-seq(.661,.669,.001)
> like3<-pi3*pi3*(1-pi3)
```

```
> #pdf(file="gridsearch.pdf",height=6,width=6)
> par(mfrow=c(2,2))
> plot(like1~pi1,type="l",main="Mesh=0.1")
> plot(like2~pi2,type="l",main="Mesh=0.01")
> plot(like3~pi3,type="l",main="Mesh=0.001")
> #dev.off()
```






```
> # Plot three figures together
> #pdf(file="gridsearch2.pdf",height=6,width=6)
> par(mfrow=c(1,1))
> plot(like1~pi1,type="l",col="gray",xlab=expression(pi),ylab="Likelihood")
> lines(pi2,like2, type="l",col="red",lty=2,lwd=3)
> lines(pi3,like3, type="l",col="blue",lwd=3)
```




```
> #dev.off()
```

1.3 Finding the Numerical Value for the ML Estimator

As the next step, we find the numerical value of the maximum likelihood estimator using function, `max(0)`. The function `cbind()` combines the vectors $pi3$ and $like3$ as column vectors into an $N \times 2$ matrix. The `print` function prints this matrix. The second `print` function prints only the row of the combined vectors for which $like3$ is equal to its maximum. This is, of course, the maximum of the likelihood function and the corresponding value of $pi3$ is the *maximum likelihood estimator* (for the degree of precision specified in the grid search, .001 in this case). Using grid search, we find that the estimated parameter value is 0.14148.

```
> # Find the numerical value
> print(cbind(pi3,like3))
```

```
       pi3     like3
 [1,] 0.661 0.1481162
 [2,] 0.662 0.1481265
 [3,] 0.663 0.1481348
 [4,] 0.664 0.1481411
 [5,] 0.665 0.1481454
 [6,] 0.666 0.1481477
 [7,] 0.667 0.1481480
 [8,] 0.668 0.1481464
 [9,] 0.669 0.1481427
```

```
> print(cbind(pi3,like3)[like3==max(like3),])
```

```
     pi3    like3
0.667000 0.148148
```

2.Comparing OLS and MLE: The Case of Normal Regression Models

During this week's lecture, we learned how to use analytic solutions to find the ML estimator for a normal regression. We learned that for estimating the mean of the model coefficient, β, the OLS estimator is essentially the ML estimator. However, the two approaches produce different estimation for σ^2. Only with a large sample size, the ML estimator for σ^2 is asymptotically equal to the OLS estimator for σ^2. In this section, we'll use simulations to compare OLS and ML estimators with different samples, with varying sample sizes.

2.1 Simulation 1: N=21

In the first simulation, we simulate a vector of x to be a sequence of numbers between -20 and 20, and set the mesh to be 2. This generates 21 observations. Next step, we define the sample size ($n1 = 21$) and the true parameter values for intercept $b0$ and slope $b1$. We set these two parameter values to be 3.5 and 2, respectively. Then, we define the dependent variable $y1$ to be a linear combination of $b0$, $x1$, and an random error,e. We define that $e \sim N(0, 1)$. With this simulated data, we know that $y = 3.5 + 2x + e$ is the true underlying model.

```
> x1<-seq(-20,20,2)
> n1<-21
> b0<-3.5
> b1<-2
> y1<-b0+b1*x1+rnorm(n1,0,1)
```

Next, we will fit both an OLS and an ML normal regression model, then compare results. Fitting an OLS regression in R is simple. We use the `lm()` in R. The function, `summary()` or `print` spell out the model output.

```
> olsmodel<-lm(y1~x1)
> summary(olsmodel)
```

```

Call:
lm(formula = y1 ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7190 -0.4876  0.2656  0.8123  0.9935

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.74268    0.20473   18.28 1.62e-13 ***
x1           1.97586    0.01691  116.88  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9382 on 19 degrees of freedom
Multiple R-squared:  0.9986,    Adjusted R-squared:  0.9985
F-statistic: 1.366e+04 on 1 and 19 DF,  p-value: < 2.2e-16
```

Function stargazer() from the \texttt{stargazer}" package can convert R model outputs in \LaTeX \space table code, following the AJPS style. It can also format a model object into a table, using the option,type="text" ".

```
> require(stargazer)
> stargazer(olsmodel, type="text")
```

```
================================================
                        Dependent variable:
                    ---------------------------
                               y1
------------------------------------------------
x1                          1.976***
                            (0.017)

Constant                    3.743***
                            (0.205)

------------------------------------------------
Observations                  21
R2                           0.999
Adjusted R2                  0.999
Residual Std. Error       0.938 (df = 19)
F Statistic         13,660.270*** (df = 1; 19)
================================================
Note:                *p<0.1; **p<0.05; ***p<0.01
```

Now, we want to estimate a normal regression model using the maximum likelihood method. Estimating an ML normal regression in R is not as easy as estimating an OLS model. It takes several steps. First, we use function() to program the log-likelihood function, which is to be maximized with respect to parameters later. In our example, we need to define three parameters: the intercept (α), the slope (β), and the variance (σ^2). The next line defines the log-likelihood function for a normal regression:

$$lnL = -\frac{1}{2} ln\sigma^2 - \frac{1}{2} \frac{[y - (\alpha + x\beta)]^2}{\sigma^2} \tag{2}$$

Once we write the log-likelihood function, we define data using our simulated sample (y1, x1, and sample size N). We then maximize the log-likelihood function with respect to the three defined parameters using function maxLik(). To start the optimization algorithm, we must set up a sequence of starting values, one for each parameter.

```
> library(maxLik)
> # Define parameters and log-likelihood function
> mlnormal<- function(param) {
+   alpha<- param[1]
+   beta<- param[2]
+   sigma <- param[3]
+   ll <- -0.5*log(sigma^2) -(0.5*((y-(alpha+beta*x))^2/sigma^2))
+   ll
+ }
>
> # Define data
> x<-x1
> y<-y1
> N<-21
```

```
>
> # Maximizing the log-likelihood function
> mlemodel1<-maxLik(mlnormal, start=c(0,0,1))
> summary(mlemodel1)

--------------------------------------------
Maximum Likelihood estimation
Newton-Raphson maximisation, 16 iterations
Return code 1: gradient close to zero (gradtol)
Log-Likelihood: -8.109753
3  free parameters
Estimates:
     Estimate Std. error t value  Pr(> t)
[1,]  3.74268    0.19481  19.212  < 2e-16 ***
[2,]  1.97586    0.01608 122.874  < 2e-16 ***
[3,]  0.89242    0.13771   6.481 9.13e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
--------------------------------------------
```

Now, compare two models (OLS v. MLE), we see that the coefficient estimates are identical. Each model produces an intercept and a slope coefficient, which are quite close to the true parameter values we set. But the OLS model produces slightly greater standard errors than the MLE model.

2.3 Simulation 2: N=501

Next, we simulate a second sample, using the same true parameter values, but increasing the sample size from 21 to 501. We repeat the analysis in Section 2.1 to see how OLS compares with MLE with a relatively large sample size. Because we have written the log-likelihood function, in this analysis, we only need to redefine the sample (data). How would you compare the two models with a large sample size?

```
> #2.2 Simulation 2: N=501
> x2<-seq(-250,250,1)
> n2<-501
> y2<-b0+b1*x2+rnorm(n2,0,1)
> # OLS,sample 2
> olsmodel2<-lm(y2~x2)
> # MLE,sample 2
> x<-x2
> y<-y2
> N<- 501
> mlemodel2<-maxLik(mlnormal, start=c(0,0,1))
> summary(olsmodel2)


Call:
lm(formula = y2 ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6635 -0.6049 -0.0326  0.6679  2.7415

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.5936342  0.0434546    82.7   <2e-16 ***
```

```
x2          2.0000789  0.0003005  6656.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9726 on 499 degrees of freedom
Multiple R-squared:      1, Adjusted R-squared:      1
F-statistic: 4.431e+07 on 1 and 499 DF,  p-value: < 2.2e-16
```

```
> summary(mlemodel2)
```

```
--------------------------------------------
Maximum Likelihood estimation
Newton-Raphson maximisation, 21 iterations
Return code 8: successive function values within relative tolerance limit (reltol)
Log-Likelihood: -235.6022
3  free parameters
Estimates:
      Estimate Std. error t value Pr(> t)
[1,] 3.5936342  0.0433707   82.86  <2e-16 ***
[2,] 2.0000789  0.0002999 6670.01  <2e-16 ***
[3,] 0.9707016  0.0306697   31.65  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
--------------------------------------------
```

2.4 Software Issues: R and Stata

In this lab, we use `maxLik()` to estimate normal regression models. When using `maxLik`, we must provide log-likelihood function and choose different optimization algorithms (the default is Newton). There are lots of optimizers in *R*.

- `maxLik` package: options for Newton-Raphson, BHHH, BFGS, others.
- `optima` (in `stats`): quasi-Newton, plus others.
- `newton` (in `stats`): Newton-Raphson solver.
- `solveLP` (in `linprog`): linear programming optimizer.

In `Stata`, `ml` is the command that we can use to implement maximum likelihood estimation. The syntax is:

```
.ml model <method> <progname> <eq>...
.ml maximize
```

A Stata example with logistic likelihood:

$$f(y, xb) = \frac{1}{1 + exp(-xb)} \tag{3}$$

```
sysuse auto.dta
program define mylogit
          args lnf Xb
          replace `lnf' = -ln(1+exp(-`Xb')) if $ML_y1==1
          replace `lnf' = -`Xb' - ln(1+exp(-`Xb')) if $ML_y1==0
  end

ml model if mylogit(foreign=mpg weight)
ml maximize

Iteration 5:   log likelihood = -27.175156

                                                Number of obs   =         74
```

```
                                                Wald chi2(2)    =      17.78
Log likelihood = -27.175156                     Prob > chi2     =     0.0001

------------------------------------------------------------------------------
     foreign |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -.1685869   .0919175    -1.83   0.067    -.3487418     .011568
      weight |  -.0039067   .0010116    -3.86   0.000    -.0058894    -.001924
       _cons |   13.70837   4.518709     3.03   0.002     4.851859    22.56487
------------------------------------------------------------------------------
```

3. Appendix: Considering Heteroskedasticity

3.1 Relax the Assumption of Constant Variance}

When we define a normal regression model using the ML method, we make the assumption that the variance parameter, σ^2, is constant. As such, we focus on the marginal effect of the explanatory variable (the slope coefficient, β) on the dependent variable, y. Yet, there are many potentially interesting phenomena that should lead us directly to a concern with σ^2, rather than with β.

3.2 Example: PACs

Consider PACs contributions for or against incumbent House members. Virtually most of this work in the existing literature treats PACs as homogeneous, full information rational actors. But even the most basic knowledge about organization theory, and qualitative studies of PAC structures, should cast doubt on this facile assumption. PACs may different in many organizational traits: size, revenue sources, internal decision-making rules, internal organizational structure (decentralized v. centralized), etc. How might these organizational features affect donations?

3.3 Task: Modeling Organizational Effects (Heterogeneity)

- Suppose y measures PAC contribution made to a race, with positive values representing donations to the incumbent and negative values support given to the challenger.
- Ignore contribution limits.
- Assume that we take a sample of labor union PACs who are known to be similar in policy and ideological preferences.
- Assume that we only have one explanatory variable to predict contribution: AFL-CIO support score for the incumbent in the previous session of Congress.
- We conceptualize that organizational traits (such as decentralized decision-making or having a research staff) do not affect the contribution level to particular candidates. Instead, these variables should affect how close contributions are to their expected values. That is organizational variables affect the variability of contributions.
  - We assume, PACs with research staff should be more predictable in their support, i.e. small σ^2.
  - We assume, PACs with decentralized decision-making rules should be less predictable in their support, i.e. large σ^2.
  - In other words, organizational effects should show up in different σ^2 not in β.
- Specify the model:

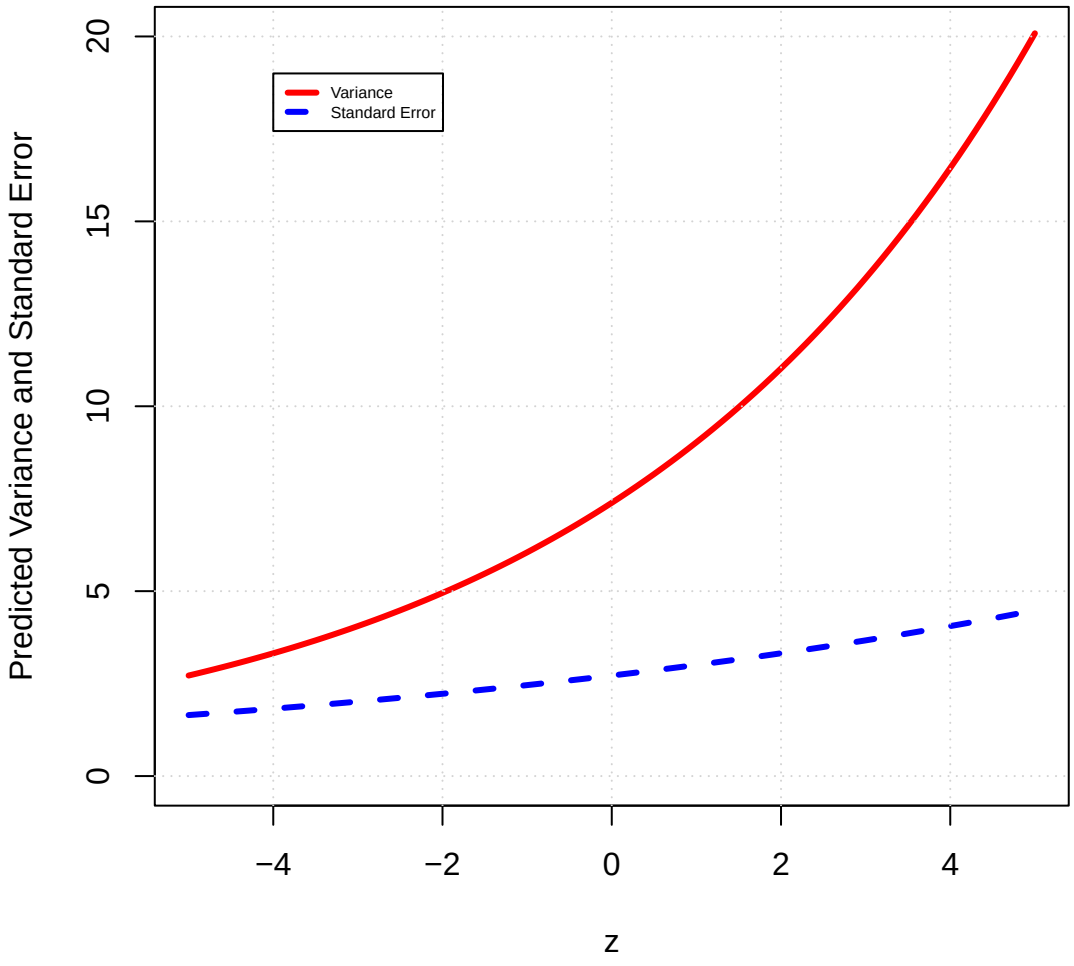
$$y_i \sim \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}[\frac{(y_i - x_i\beta)^2}{\sigma_i^2}]} \tag{4}$$

- In principle, each PAC now has a unique value for σ_i^2. This raises the same identification problem as we confront when estimating u_i.
- What we can do is to re-parameterize σ_i^2, by letting it be a linear function of a set of exogenous factors, z_i. We use the exponential function to make sure that σ_i^2 is always non-negative. This is also technically nice, because we do not have to worry about constraining $z_i\gamma$ to a particular range.

$$\sigma_i^2 = e^{z_i\gamma} \tag{5}$$

The following figure illustrates changing variance and standard errors as a linear function of z.

```
> z<-seq(-5,5,0.1)
> sigma2<-exp(2.0+0.2*z)
> sigma<-sqrt(sigma2)
> plot(sigma2~z,col="red",type="l",lwd=3,
+      ylim=c(0,20),xlim=c(-5,5),
+      ylab="Predicted Variance and Standard Error")
> lines(z,sigma,col="blue",type="l",lwd=3,lty=2 )
> grid()
> legend (-4,19, # The first two arguments define the position of the legend (x,y)
+         lty=c(1,2), # Define the line properties
+         lwd=c(3,3), # Define line width
+         col=c("red","blue"), # Define line colors
+         c("Variance", "Standard Error"), # Define the vector of labels
+         cex=0.5 # Option "cex" define the legend size
+ )
```



After re-parameterization, the heteroskedastic normal regression model is now written as:

$$y_i \sim \frac{1}{\sqrt{2\pi e^{z_i\gamma}}} e^{-\frac{1}{2}[\frac{(y_i - x_i\beta)^2}{e^{z_i\gamma}}]} \tag{6}$$

We can now write the log-likelihood function as:

$$lnL(y, z, \beta, \gamma) = -\frac{N}{2} ln(2\pi) - \sum_{i=1}^{N} \frac{1}{2} z_i\gamma - \frac{1}{2} \sum_{i=1}^{N} [\frac{(y_i - x_i\beta)^2}{e^{z_i\gamma}}] \tag{7}$$

We maximize the log-likelihood function with respect to β and γ to find the ML estimates. This also allow us to test for the effect of each exogenous organizational variable in z on the variance term. For example, we can test if the coefficient on staff was negative or if the effect of decentralization was positive.

3.4 Estiation using R

Surprisingly, in R, little exiting routine does heteroskedastic normal regression using the ML method. Happily, R and Stata let us write our own likelihood functions and maximize them.[1] While develop a complete R package is rather complicated, getting the basic program working is not too tough. In the next working example, we will use two HetReg functions written by Charles Franklin.

- Data example: pre-election polls and election forecasting. Polls come with margins of error which depend on sample size. But the predictive value of pools depends on when the poll is taken and the dynamics of the campaign. Hence, the practical precision of polls is a function of time. We can model this as a heteroskedastic regression model that accounts for changes in the variability of polls as election day approaches (i.e. time-varying variance).
- Model specification: we posit that the vote margin is a linear function of the following variables: `dempoll, reppoll, deminc,repinc,` and `pollgap`. Heteroskedasticity is attributed to variable `days2go` (the number of days-to-go before the election day), `dempoll,reppoll,deminc`, and `repinc`.

```
> ## Heteroskedastic Normal Regression via ML in R
> # Version 2.
> # Charles H. Franklin, Monday, June 30, 2003 at 19:37
> # Modified: Thursday, July 1, 2004 at 20:19--Make y 1st argument
> # Modified: Saturday, July 17, 2004 at 20:00
>
>   MLhetreg<-function(y,X,Z,method='BFGS',Xnames=colnames(X),Znames=colnames(Z)){
+     X<-cbind(1,X)
+     colnames(X)[1]<-"Constant"
+     nx<-ncol(X)
+     Z<-cbind(1,Z)
+     colnames(Z)[1]<-"ZConstant"
+     nz<-ncol(Z)
+     neglnl<-function(theta,X,Z,y){
+         b<-theta[1:ncol(X)]
+         g<-theta[ncol(X)+1:ncol(Z)]
+         lnl<-as.vector(-.5*(Z%*%g)-(.5/exp(Z%*%g))*(y-X%*%b)^2)
+         -sum(lnl)
+         }
+     result<-c(optim(c(mean(y),rep(0,ncol(X)-1),log(var(y)),rep(0,ncol(Z)-1)),
+                 neglnl, hessian=T, method=method, X=X, Z=Z, y=y),
+                 list(varnames=c(Xnames,Znames),nx=nx,nz=nz))
+     class(result)<-"MLhetreg"
+     return(result)
+     }
>
> ## Print method for MLhetreg
>
>     print.MLhetreg <- function(object){
+         coef<-object$par
+         names(coef)<-object$varnames
+         print(coef)
+         if (object$convergence==0) cat('\n hetreg converged\n')
+         if (!object$convergence==0) cat('\n *** hetreg failed to converge ***\n')
+         invisible(object)
+         }
```

[1] The GLLAMM package in Stata can implement this with an HLM (hierarchical linear model) set up. We won't pursue this further in Lab 2.

```
>
>
> ##Summary function for MLhetreg
>
>     summary.MLhetreg<-function(object, covar=FALSE){
+         coef<-object$par
+         names(coef)<-object$varnames
+         nx<-object$nx
+         nz<-object$nz
+         maxl<-object$value
+         vc<-solve(object$hessian)
+         colnames(vc)<-names(coef)
+         rownames(vc)<-names(coef)
+         se<-sqrt(diag(vc))
+         zscore<-coef/se
+         pz<-2*pnorm(-abs(coef/se))
+         dn<-c("Estimate","Std. Error")
+         coef.table<-cbind(coef,se,zscore,pz)
+         dimnames(coef.table)<-list(names(coef),c(dn,"z-value",
+             "Pr(>|z|)"))
+         cat("\nHeteroskedastic Linear Regression\n")
+         cat("\n  Estimated Parameters\n")
+         print(coef.table)
+         cat("\nLog-Likelihood: ",-object$value,"\n")
+         if (covar) {
+             cat("\nVariance-Covariance Matrix for Parameters\n")
+             print(vc) }
+         ghat<-coef[(nx+2):length(coef)]
+         gvc<-vc[(nx+2):length(coef),(nx+2):length(coef)]
+         wald<-t(ghat)%*%solve(gvc)%*%ghat
+         pwald<-1-pchisq(wald,nz-1)
+         cat("\nWald Test for Heteroskedasticity\n")
+         cat("  Wald statistic: ",wald, "with ", nz-1," degrees of freedom\n")
+         cat("                   p=",pwald,"\n")
+         }
```

```
> load("hetpolls.RData")
> attach(govsen)
> names(govsen)
```

```
 [1] "stateab"      "race"         "year"         "n"            "lv"
 [6] "dem"          "rep"          "undec"        "pollster"     "deminc"
[11] "repinc"       "d3pct"        "r3pct"        "pollstercode" "dempoll"
[16] "reppoll"      "acadpoll"     "days2go"      "weeks2go"     "err5"
[21] "pollgap"      "votegap"      "logdays2go"   "demwin"       "abserr5"
```

```
> # Add a new variable into the dataframe
> govsen$anyinc<-deminc+repinc
> attach(govsen)
> hreg1<-MLhetreg(votegap, cbind(dempoll,reppoll,
+       deminc,repinc,pollgap),
+       cbind(days2go,dempoll,reppoll,deminc,repinc))
> summary(hreg1)
```

```
Heteroskedastic Linear Regression

  Estimated Parameters
             Estimate  Std. Error    z-value     Pr(>|z|)
Constant  -0.612171009 0.361605195 -1.6929265 9.046946e-02
```

```
dempoll   -5.457941935 0.842688769 -6.4768182 9.367696e-11
reppoll    3.746292854 0.667026507  5.6164078 1.949682e-08
deminc     0.376559859 0.591230752  0.6369084 5.241845e-01
repinc    -0.229478591 0.533286639 -0.4303100 6.669701e-01
pollgap    0.761652341 0.016286565 46.7656834 0.000000e+00
ZConstant  3.680411989 0.086944987 42.3303529 0.000000e+00
days2go   -0.006553575 0.001442331 -4.5437394 5.526499e-06
dempoll    0.026577312 0.167622811  0.1585543 8.740201e-01
reppoll   -0.354242462 0.159798606 -2.2168057 2.663637e-02
deminc    -0.145487084 0.121359786 -1.1988080 2.306026e-01
repinc    -0.180558044 0.103205405 -1.7495018 8.020432e-02

Log-Likelihood:  -2367.821

Wald Test for Heteroskedasticity
  Wald statistic:  26.81843 with  5  degrees of freedom
                   p= 6.187894e-05
```

We can also specify an alternative model, in which we include fewer explanatory variables.

```
> hreg2<-MLhetreg(votegap, cbind(dempoll,reppoll, pollgap),
+                 cbind(days2go,reppoll,anyinc))
> summary(hreg2)


Heteroskedastic Linear Regression

  Estimated Parameters
              Estimate  Std. Error   z-value     Pr(>|z|)
Constant  -0.609701565 0.232044330 -2.627522 8.600930e-03
dempoll   -5.473179622 0.830495636 -6.590257 4.390661e-11
reppoll    3.785072844 0.659194498  5.741967 9.358324e-09
pollgap    0.769885076 0.012875862 59.792896 0.000000e+00
ZConstant  3.688586547 0.086298759 42.742058 0.000000e+00
days2go   -0.006514894 0.001436769 -4.534408 5.776540e-06
reppoll   -0.362126766 0.158248907 -2.288337 2.211793e-02
anyinc    -0.173191428 0.092461749 -1.873114 6.105260e-02

Log-Likelihood:  -2368.262

Wald Test for Heteroskedasticity
  Wald statistic:  26.70332 with  3  degrees of freedom
                   p= 6.793681e-06
```