# *R* computing for Business Data Analytics

Instructor: Dr. Howard Hao-Chun Chuang
Assistant Professor, Department of Management Information Systems
National Chengchi University, Taipei, Taiwan 116
chuang@nccu.edu.tw

Last Revised: October 2014

In lecture 6 we go through linear regression that accommodates a *continuous response variable y* and assume $y_i \sim N(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}, \sigma^2),\ i = 1, 2, ..., n$. This normality assumption, however, can be violated in many cases where the response variable *y* is neither continuous nor symmetric. So, this lecture introduces to you the *generalized linear modeling* (GLM), which subsumes linear regression and enables us to tackle different types of response variables. In the context of GLM, we no longer need to assume *y* to be normally distributed and continuous.

Remember in linear regression we fit the model below to data

$$y_i = \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} + \varepsilon_i,\ i = 1, 2, ..., n$$

$$\text{where } y_i \in [-\infty, +\infty] \ \&\ \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} \in [-\infty, +\infty]$$

$$E(y_i \mid x_{1i}, ..., x_{ki}) = \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}$$

In GLM, we fit a modified model to data

$$y_i = g(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}) + \varepsilon_i,\ i = 1, 2, ..., n$$

$$\text{where } \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} \in [-\infty, +\infty]$$

where $\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} \in [-\infty, +\infty]$, but the response variable $y_i$ could be a non-negative or strictly positive integer count, a ratio in [0, 1], a binary indicator (0/1), a categorical response (e.g., ratings, voting results, ethnic groups), or a continuous real number (e.g., time-to-event). Thus, we use a *link function* $g(\bullet)$ to map $\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}$ into the relevant domain of $y_i$. Don't worry about the link function too much if you find this idea too abstract. A simple way to think about GLM is that since the normal distribution is NOT valid for those $y_i$ types mentioned above, we must find an appropriate probability distribution to model each type of $y_i$. Therefore, those discrete and continuous probability distributions you have learnt from lectures 4 will come into play during our journey of GLM.

## 7.1 Count data models

- Poisson regression

Suppose we have collected *count data* for an event (e.g., the weekly number of tourists who visit Taipei 101, the daily number of downloads for a song, the hourly number of customers who enter a department store), Poisson regression is almost always the first model to be used by data analysts. Remember in lecture 4 we have seen the Poisson distribution with its *pdf*

$$f(y \mid \lambda) = \frac{\lambda^y e^{-\lambda}}{y!},\ y = 0,1,...,+\infty$$

$$\text{where } \lambda \in (0,+\infty]$$

We want to learn the association between the response $y$ and an array of factors ( $x_1, x_2, ..., x_k$ ) for each individual $i$=1,2,…,$n$. We then assume that

$$E(y_i \mid x_{1i},...,x_{ki}) = \lambda_i = g(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}),\ i = 1,2,...,n$$

In the Poisson regression model, each unit $i$ corresponds to a *setting* (typically a location or a time interval), in which $y_i$ is the number of events. For instance, $i$ could be the index of street intersections in Taipei city and $y_i$ could be the number of traffic accidents at intersection $i$ in a given year. The trick to estimate the regression model is to use a *log link function* such that $\log(\lambda_i) = \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}$ , or equivalently,. $\lambda_i = \exp(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki})$ .

Some of you may wonder, why bother with *glm*( )? Why not just use *lm*( ) ? Similar to what we did in lecture 6, let's perform a simulation experiment.

```
> set.seed(3759)
> b0=0.2
> b1=0.5
> n=1000
> x=runif(n, -1, 1)
> S=1000
> par.est=matrix(NA, nrow=S, ncol=4)
> for(s in 1:S){
> y=rpois(n, exp(b0+b1*x))
> model_glm=glm(y ~ x, family="poisson") #The function glm()
> model_lm=lm(y ~ x)
```

```
> par.est[s, 1]=model_glm$coef[1]
> par.est[s, 2]=model_lm$coef[1]
> par.est[s, 3]=model_glm$coef[2]
> par.est[s, 4]=model_lm$coef[2]}
```

Compare the estimates of *b1* from two regression models.

```
> dev.new(width=12, height=5)
> par(mfrow=c(1,2))
> hist(par.est[,3],main="Poisson Reg b1")
> abline(v=b1,col='red',lwd=2)
> hist(par.est[,4],main="Linear Reg b1")
> abline(v=b1,col='red',lwd=2)
```

Which one is better?

Through this example, I hope you have started to understand *the consequence of imposing wrong assumptions* on data analysis.

Below illustrates how to apply Poisson regression to count data analysis in *R*.

```
> library(AER)
> data("RecreationDemand")
> head(RecreationDemand)
> rd_Pois=glm(trips ~ ., data=RecreationDemand, family=poisson)
> summary(rd_Pois)
> logLik(rd_Pois) # Does this look familiar? What is AIC?
```

How can we obtain a measure similar to $R^2$ (i.e., *pseudo-*$R^2$)?

```
>
```

We do not necessarily have to rely on the *glm*( ) function to perform Poisson regression. We can manually estimate the beta parameters by optimizing the log-likelihood function

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \lambda_i = e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}}) \right]$$

Below is how you can do it in *R* using the MLE techniques learnt from lecture 5.

```
> attach(RecreationDemand)
> SKI=ifelse(ski=="yes",1,0)
> USERFEE=ifelse(userfee=="yes",1,0)
```

```
> Poisloglikf=function(par){
> loglik=0
> for(i in 1:nrow(RecreationDemand)){
> lam=exp(1*par[1]+quality[i]*par[2]+SKI[i]*par[3]+ income[i]*par[4]+
> USERFEE[i]*par[5]+costC[i]*par[6]+costS[i]*par[7]+costH[i]*par[8])
> loglik=loglik+dpois(trips[i], lam, log=TRUE)}
> log-lik}
> est=nlminb(c(1,rep(0.001,7)), Poisloglikf,control=list(trace=1))
```

Are the estimated betas identical to those from *glm*( )?

P.S.: You can also derive the standard errors of the beta estimates using *Fisher information* or *Bootstrapping*. However, that is beyond the scope of this course.

- Negative binomial regression

Despite the popularity of Poisson distribution, it has the restricted property of equi-dispersion (i.e., $Var(y_i)/E(y_i) = \lambda_i/\lambda_i = 1$ see lecture 4), which is unlikely to hold in many real datasets.

So, we need the *negative binomial distribution* to tackle over-dispersion.

```
> attach(RecreationDemand)
> var(trips)
> mean(trips)
```

Is the equi-dispersion assumption likely to hold here? Let's do a formal statistical test.

```
> dispersiontest(rd_Pois)
```

Now we need the negative binomial regression model. Here is how you can do it in *R*.

```
> library(MASS)
> rd_NB=glm.nb(trips ~ ., data=RecreationDemand)
> summary(rd_NB)
```

We can compare the estimation results of the two regression models.

```
> coeftest(rd_Pois)
> coeftest(rd_NB)
```

What can you say about the results? Do you notice any differences?

Through this example, I hope you have started to understand the consequence of ignoring the *level of dispersion* – variance-to-mean ratio –on *count data* analysis.

- Zero-inflation and zero-truncation

Here we want to explore some issues related "zero".

> table(trips[1:10])

> table(round(fitted(rd_Pois)))

> table(round(fitted(rd_NB)))

How are we doing in terms of capturing the number of zeros in the data?

When facing *excess zeros* in data, apply modified models such as a zero-inflated Poisson

$$f(y \mid \lambda, \pi) = \begin{cases} \pi + (1-\pi)\dfrac{\lambda^y e^{-\lambda}}{y!}, & y = 0 \\ (1-\pi)\dfrac{\lambda^y e^{-\lambda}}{y!}, & y = 1, ..., +\infty \end{cases} \Rightarrow \begin{matrix} \pi + (1-\pi)e^{-\lambda}, & y = 0 \\ (1-\pi)\dfrac{\lambda^y e^{-\lambda}}{y!}, & y = 1, ..., +\infty \end{matrix}$$

> install.packages("pscl")

> library(pscl)

> rd_ziPois=zeroinfl(trips~. , data=RecreationDemand, dist= "pois")

> rd_ziNB=zeroinfl(trips~. |quality+income, data=RecreationDemand, dist= "negbin")

> round(colSums(predict(rd_ziPois, type= "prob")[,1:10]))

> round(colSums(predict(rd_ziNB, type= "prob")[,1:10]))

We have learnt how to resolve the issue of *zero-inflation*. What if the dataset has NO zeros at all? We can apply modified models such as a zero-truncated Poisson

$$f(y \mid \lambda) = \frac{\lambda^y e^{-\lambda}}{y!} \Big/ (1 - \frac{\lambda^0 e^{-\lambda}}{0!}) = \frac{\lambda^y e^{-\lambda}}{y!(1-e^{-\lambda})}, \; y = 1, ..., +\infty$$

Let's tweak the RecreationDemand data and pretend we have an issue of *zero-truncation*

> RD_NoZero=RecreationDemand[ which(trips!=0), ]

> detach(RecreationDemand)

> attach(RD_NoZero)

Check the dispersion first

> var(trips)/mean(trips)

> detach(RecreationDemand)

> attach(RD_NoZero)

> table(trips)

We do not want to use the ordinary Poisson/negative binomial model because of the absence of zeros. In *R* we can estimate the zero-truncated regression model for count data easily

```
> install.packages("VGAM")
> library(VGAM)
> rdnz_ztpois=vglm(trips ~ quality + ski + income + userfee + costC + costS + costH,
family=pospoisson, data=RD_NoZero)
```

What is happening? As an estimation method, maximum likelihood is theoretically appealing but the corresponding numerical optimization sometimes can be a daunting task…

```
> rdnz_ztnb=vglm(trips ~ quality + ski + income + userfee + costC + costS + costH,
family= posnegbinomial, data=RD_NoZero)
> summary(rdnz_ztnb)
```

There are no *p*-values readily available, how can you tell the statistical significance of those estimated betas?

- Optional: Under-dispersion and finite range

Before the end of 8.1, let me make things more complicated. In the paper by Chou, Chuang, & Shao (2014, http://onlinelibrary.wiley.com/doi/10.1002/asmb.2037/abstract ), the authors investigate the association between information initiatives made by mobile retailers ($y$) and other factors ($x$). However, the count response variable $y$ exhibits several issues:

Under-dispersion: Var($y$)/E($y$)<<1

Zero-truncation: $y>0$

Finite range: $y \in [1, N]$

The first candidate model is a zero-truncated (ZT) binomial model

$$f(y \mid \pi) = \binom{N}{y} \pi^y (1-\pi)^{N-y} \Big/ \left(1-(1-\pi)^N\right),\ y = 1,\ldots,N$$

where $\pi \in [0,1]$

In *R* we estimate the beta parameters by optimizing the log-likelihood function

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \pi_i = e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}} / (1+e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}})) \right]$$

The second candidate model is a ZT Poisson distribution. Although the Poisson distribution is equi-dispersed (i.e., Var($y$)/E($y$)=1), its ZT version is under-dispersed.

$$f(y \mid \lambda) = \frac{\lambda^y e^{-\lambda}}{y!} \Big/ (1 - e^{-\lambda}),\ y = 1, ..., +\infty$$

$$\text{where } \lambda \in (0, +\infty]$$

In *R*, we estimate the beta parameters by optimizing the log-likelihood function

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \lambda_i = \exp(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki})) \right]$$

The third candidate model is a ZT Conway-Maxwell-Poisson (CMP) distribution.

$$f(y \mid \lambda, \nu) = \frac{\lambda^y}{(y!)^\nu Z(\lambda, \nu)} \Big/ (1 - \frac{1}{Z(\lambda, \nu)}),\ y = 1, ..., +\infty$$

$$\text{where } \lambda \in (0, +\infty],\ \nu \in [0, +\infty],\ \text{and } Z(\lambda, \nu) = \sum_{j=0}^{\infty} \lambda^j / (j!)^\nu$$

The CMP distribution accommodates both under-dispersed and over-dispersed count. In *R*, its computation can be done using packages such as *compoisson* and *COMPoissonReg*. In *R*, we estimate the beta parameters by optimizing the log-likelihood function

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \lambda_i = \exp(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}), \nu) \right]$$

The last candidate model is a ZT Consul's generalized Poisson (GP) distribution.

$$f(y \mid \lambda, \theta) = (\frac{\lambda}{1 + \theta\lambda})^{y_i} (1 + \theta y)^{y-1} e^{-\theta\lambda y/(1+\theta\lambda)} \Big/ y!(e^{\lambda/(1+\theta\lambda)} - 1),\ y = 1, ..., +\infty$$

$$\text{where } \lambda \in (0, +\infty] \text{ and } \theta \in [-\infty, +\infty]$$

Similar to the CMP distribution, the Consul's GP addresses both under-dispersed and over-dispersed count. Again, in *R* we estimate the beta parameters by optimizing the function

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \lambda_i = \exp(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}), \theta) \right]$$

Note that the three Poisson-related distributions all assume $y \in [1, \infty]$ as opposed to the finite range $y \in [1, N]$ exhibited in data. Hence, the three models are *approximations*. Nonetheless, Chou, Chuang, & Shao (2014) find that the ZT Consul's GP model fit the data significantly better than the ZT binomial that exactly addresses the upper limit of $y$.

The aforementioned list of candidate models is by no means exhaustive. My goal is to let you be aware of the importance of distributional assumptions. Under-dispersion and finite range,

unfortunately, are often ignored by analysts and researchers who impose wrong distributional assumptions on observed count data. Such ignorance makes results of analysis misleading.

### 8.2 Ratio response models

- Beta regression

How should we perform regression analysis when the response variable *y* is bounded within [0, 1] or (0, 1)? It is not uncommon for data analysts to encounter such a variable since many response variables of interest could be rates or proportions. Often people tend to use the *logit transformation*, $\tilde{y} = \log\left(y/(1-y)\right)$, and run ordinary linear regression on $\tilde{y}$. However, this sort of Gaussian–based approximations can be quite inaccurate in small samples since rates or proportions are typically *asymmetric* and *heteroskedastic*, that is, the observations exhibit more variation around the mean and less variation as it gets closer to the upper/lower limits. So, what can we do as the normal distribution is not a viable option? It turns out that the *beta distribution* is a flexible alternative. Its pdf is expressed as

$$f(y \mid p,q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} y^{p-1}(1-y)^{q-1},\ 0 < y < 1$$

$$\text{where } p,q \in (0,+\infty]$$

The beta random variable has a theoretical domain in (0, 1). What if the ratio or proportion *y* displays 0/1? A useful transformation is $\left(y \cdot (n-1) + 0.5\right)/n$ where *n* is the sample size. For the sake of estimation, we re-parameterize the pdf above with $\mu = p/(p+q)$ and $\phi = p+q$

$$f(y \mid \mu,\phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1},\ 0 < y < 1$$

$$\text{where } 0 < \mu < 1 \text{ and } \phi \in (0,+\infty]$$

$$E(y) = \mu\ ;\ Var(y) = \mu(1-\mu)/(1+\phi)$$

Accordingly, we assume that

$$E(y_i \mid x_{1i},...,x_{ki}) = \mu_i = g(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}),\ i = 1,2,...,n$$

We first use the logit link function $g(\mu) = \log(\mu/(1-\mu))$ and estimate betas by optimizing

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \mu_i = e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}} / (1+e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}}), \phi) \right]$$

In *R*, you simply need to use the *betareg* package.

> install.packages("betareg")

> library(betareg)

Let's look at Prater's gasoline yield data. Here the response variable of interest is *yield*, the proportion of crude oil converted to gasoline after distillation and fractionation. We assess two explanatory variables – *temp* (the temperature at which all gasoline has vaporized), and *batch* (different batches of crude oil).

> data("GasolineYield", package= "betareg")

> head(GasolineYield)

> gy.logit=betareg(yield ~ batch + temp, data=GasolineYield )

> summary(gy.logit)

Here we set the dispersion parameter $\phi$ to be constant. In fact, we can make it more flexible by optimizing

$$\sum_{i=1}^{n} \log\left[ f(y_i \mid \mu_i = e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}} / (1+e^{\beta_0+\beta_1 x_{1i}+\cdots\beta_k x_{ki}}), \phi_i = \beta_0 + \beta_m x_{mi} + \cdots) \right], m \in \mathbb{N} \text{ in } [1,k]$$

> gy.fit2=betareg(yield ~ batch + temp | temp, data=GasolineYield )

> summary(gy.fit2)

There are several alternative link functions available, including *probit* $g(\mu) = \Phi^{-1}(\mu)$, *log-log* $g(\mu) = -\log(-\log(\mu))$, the *complementary log-log* $g(\mu) = \log(-\log(1-\mu))$, and Cauchy $g(\mu) = \tan(\pi(\mu - 0.5))$. The choice of link functions has substantial impacts on model fit.

> gy.probit=betareg(yield ~ batch + temp, data=GasolineYield, link="probit" )

> gy.loglog=betareg(yield ~ batch + temp, data=GasolineYield, link="loglog" )

> gy.cloglog=betareg(yield ~ batch + temp, data=GasolineYield, link="cloglog" )

> gy.cauchy=betareg(yield ~ batch + temp, data=GasolineYield, link="cauchy" )

How do we select the best model? Let's compute the Akaike information criterion (AIC).

> AIC(gy.logit, gy.logit2, gy.probit, gy.loglog, gy.cloglog, gy.Cauchy)

Which one is the model to go with? Do we really need $\phi_i = \beta_0 + \beta_m x_{mi} + \cdots, m \in \mathbb{N}$ in $[1,k]$?

This example gives you the flavor of the Beta regression, which is natural for proportion/rate response and more than a "better lemon squeezer". Now we will shift the gear to categorical response variables.

## 8.3 Binary response models

- Logit and probit models

It is very, very common that data analysts need to deal with binary response variable. The response *y* can be either *true/yes/success* or *false/no/failure*, usually coded as 1 and 0. In many data mining problems, the target is a binary response such as

profit or loss, response greater or less than a certain value, insolvent or not;

thumbs up or down, buy or not buy, become a member or not;

win or lose, sick or healthy;

sentenced to death or not, voting for LienD or KoP.

Remember that we are still interested in estimating the model

$$g(y_i) = \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} + \varepsilon_i,\ i = 1, 2, ..., n$$

Since $y_i$ is either 0 or 1, we can model $y_i$ as a Bernoulli distribution such that

$$f(y_i) = p_i^{y_i}(1 - p_i)^{1-y_i}$$

$$\text{where } p_i = P(y_i = 1) \ \& \ E[y_i] = p_i$$

The first model to be introduced is the famous logistic regression model, which characterizes the probability $p_i$ as

$$p_i = F(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}) = \frac{\exp(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki})}{1 + \exp(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki})}$$

$$\text{where } F(\cdot) \text{ is the logistic CDF}$$

Mathematically we can show that

$$\log(\frac{p_i}{1 - p_i}) = \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki}$$

The function $f(p) = \log(p/1-p)$ is the *logit* and the ratio $p/1-p$ is the *odds of success*.

If we change the function $F(\cdot)$ above to the standard normal CDF (*pnorm*( )), we will have a *probit* model. The logit and probit are the two most commonly used regression models for binary response (0/1) dependent variables.

Again, why bother with *glm*( )? Why not just use *lm*( ) ? Let's run a simulation experiment.

```
> inv.logit=function(p){
> return(exp(p)/(1+exp(p)))}
```

```
> set.seed(32945)
> b0=0.2
> b1=0.5
> n=1000
> x=runif(n, -1, 1)
> S=1000
> par.est=matrix(NA, nrow=S, ncol=2)
> for(s in 1:S){
> y=rbinom(n, 1, inv.logit(b0+b1*x))
> model_glm=glm(y ~ x, family=binomial(link=logit))
> model_lm=lm(y ~ x)
> par.est[s, 1]=model_glm$coef[2]
> par.est[s, 2]=model_lm$coef[2]}
```

Compare the estimates of *b1* from two regression models.

```
> dev.new(width=12, height=5)
> par(mfrow=c(1,2))
> hist(par.est[,1],main="Logistic Reg b1")
> abline(v=b1,col='red',lwd=2)
> hist(par.est[,2],main="Linear Reg b1", xlim=c(0, 0.5))
> abline(v=b1,col='red',lwd=2)
```

Which one is better?

We explore an example in labor economics and consider female labor force participation for a sample of 872 women from Switzerland.

```
> library(AER)
> data("SwissLabor")
> plot(participation ~ age, ylevels=2:1) #this is the spinogram. Try ~ education
```

Fit the logit and probit models

```
> swiss_logit=glm(participation ~ . +I(age^2), data=SwissLabor, family=binomial(link=
"logit"))
> swiss_probit=glm(participation ~ . +I(age^2), data=SwissLabor, family=binomial(link=
"probit"))
```

We can calculate the MacFadden's pseudo-$R^2$

```
> swiss_logit0=update(swiss_logit, formula = . ~ 1) #what are we doing here?
> summary(swiss_logit0)
> 1- as.vector(logLik(swiss_logit)/logLik(swiss_logit0))
```

A Chi-squared test can also be performed to compare two *nested models*

```
> anova(swiss_logit0, swiss_logit, test= "Chisq")
```

We also want to know how good the fitted model is at *classification/prediction*.

```
> table(true=SwissLabor$participation, pred=round(fitted(swiss_logit)))
> table(true=SwissLabor$participation, pred=round(fitted(swiss_probit)))
```

How is our prediction performance using the cutoff 0.5? We care about several metrics when it comes to predictive modeling.

Accuracy: (TP+TN)/(P+N)

True positive rate (TPR, aka sensitivity): TP/P=TP/(TP+FN)

False positive rate (FPR): FP/N=FP(FP+TN)

True negative rate (TNR, aka specificity): TN/N=TN/(FP+TN)

| | | Actual Value | |
|---|---|---|---|
| | | p | n |
| Predicted Outcome | p' | True Positive (TP) | False Positive (FP) |
| | n' | False Negative (FN) | True Negative (TN) |
| | Total | P | N |

TPR and FPR make the receiver operating characteristic (ROC)={FPR($c$), TPR($c$)|c in [0,1]}. ROC is a useful diagnostic for binary predictive modeling where the ultimate goal is to have a model that achieves FPR=0 and TPR=1. For simplicity I will use exclusively swiss_probit for the rest of this section.

```
> install.packages("ROCR")
> library(ROCR)
> pred=prediction(fitted(swiss_probit), SwissLabor$participation)
```

```
> dev.new(width=12, height=5)
> par(mfrow=c(1, 2))
> plot(performance(pred, "acc"))
> plot(performance(pred, "tpr", "fpr"))
> abline(0, 1, lty=2)
```

Actually, both the logit and probit assume symmetric responses. When the ratio of 1-to-0 is highly asymmetric, you should try the *cloglog* link too.

```
> swiss_cloglog=glm(participation ~ . +I(age^2), data=SwissLabor,
family=binomial(link= "cloglog"))
> pred2=prediction(fitted(swiss_cloglog), SwissLabor$participation)
> plot(performance(pred2, "tpr", "fpr"), col= 'red', lty=2, lwd=2)
```

- Complete separation

This is an occasionally occurred but rarely discussed issue in logit/probit modeling.

```
> data("MurderRates")
> murder_logit=glm(I(executions>0) ~ time + income + noncauc + lfp + southern,
data=MurderRates, family(binomial))
> coeftest(murder_logit)
```

What does the warning message and the big standard error imply?

```
> murder_logit2=glm(I(executions>0) ~ time + income + noncauc + lfp + southern,
data=MurderRates, family(binomial), control=list(epsilon=1e-15, maxit=50, trace=F))
> coeftest(murder_logit2)
```

What is happening to the standard error now?

The numerical issue arises because maximum likelihood estimates do not exist here. Instead,

$$y_i = 0 \text{ whenever } \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} < 0$$
$$y_i = 1 \text{ whenever } \beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki} > 0$$

This data is said to exhibit *complete separation*. What if we exclude the variable *southern*?

P.S.: One can estimate binary response models by optimizing the log-likelihood function

$$\ell(\beta) = \sum_{i=1}^{n} \{ y_i \log(F(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki})) + (1 - y_i) \log(1 - F(\beta_0 + \beta_1 x_{1i} + \cdots \beta_k x_{ki})) \}$$

## 8.4 Categorical response models

- Cumulative link model

Suppose we have ordered categorical responses such as school grades, movie ratings, phases of cancer, etc. Oftentimes we use a cumulative link model for an ordinal variable $y_i$ that can fall into $j$=1,…, $J$ categories (what is the case of $J$=2?) with respective probabilities $p_{ij}$.

We define the cumulative probabilities as

$$r_{ij} = P(Y_i \le j) = p_{i1} + \cdots p_{ij}$$

Again, we will consider the logit link. The cumulative logits are defined as

$$\text{logit}(r_{ij}) = \text{logit}(P(Y_i \le j)) = \log(\frac{P(Y_i \le j)}{1 - P(Y_i \le j)}), j = 1,...,J-1$$

The consequent regression model – ordered logit – is

$$\log(\frac{P(Y_i \le j)}{1 - P(Y_i \le j)}) = \theta_j + \beta_1 x_{1i} + \cdots \beta_k x_{ki} + \varepsilon_i$$

Note that here we have *multiple intercepts*.

Once the model is estimated, we can predict the probability of falling into category $j$ by

$$P(Y_i \le j) = \frac{\exp(\hat{\theta}_j + \hat{\beta}_1 x_{1i} + \cdots \hat{\beta}_k x_{ki})}{1 + \exp(\hat{\theta}_j + \hat{\beta}_1 x_{1i} + \cdots \hat{\beta}_k x_{ki})}$$

$$\text{such that } P(Y_i = j) = P(Y_i \le j+1) - P(Y_i \le j)$$

Let's NOT get into technical details. I'll show you an example of how to estimate it in *R*.

```
> data("BankWages")
> edcat=factor(BankWages$education)
> levels(edcat)[3:10]=rep(c("14-15", "16-18", "19-21"), c(2, 3, 3))
> plot(job ~ edcat, data=BankWages, off=0)
```

The response variable *job* can be considered ordinal, with "custodial"< "admin" < "manage".

```
> library(MASS)
> bank_ologit=polr(job ~ education + minority, data=BankWages, Hess=T)
> coeftest(bank_ologit)
```

What is the key finding here?

```
> fitted(bank_ologit)
```

Note that you could estimate the same model using the newly developed *ordinal* library.

What if you do not treat those job categories to be ordinal? That is, the categories are just nominal variables. We can fit a multinomial logit model accordingly.

```
> install.packages("nnet")
> library(nnet)
> bank_nlogit=multinom(job ~ education + minority, data=BankWages)
> coeftest(bank_nlogit)
```

What is the key finding here? Should we use the ordered logit or the multinomial logit?

Note that you could estimate the model using the *mlogit* library.

### 8.5 Quantile regression

The last piece I want to discuss here is *quantile regression*, which has gradually become an attractive alternative to the ordinary least squares (OLS) regression in lecture 7.

The OLS linear regression models the response $y$ as a conditional mean of $x$

$$E(y \mid \boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \cdots \beta_k x_k$$

The quantile regression models the response $y$ as a conditional quantile of $x$

$$Q_y(\tau \mid \boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \cdots \beta_k x_k$$

where $Q_y(\tau \mid \boldsymbol{x})$ denotes the $\tau$-quantile of $y$ conditional on $x$

Suppose we want to estimate the following model in *R*

$$Q_{\log(\text{wage})}(\tau \mid \boldsymbol{x}) = \beta_0 + \beta_1 \text{experience} + \beta_2 \text{experience}^2 + \beta_3 \text{education} + \varepsilon$$

```
> install.packages("quantreg")
> library(quantreg)
> data("CPS1988")
> cps_f=log(wage) ~ experience + I(experience^2) + education
> cps_lad=rq(cps_f, data=CPS1988)
```

The default of *rq*() is $\tau$=0.5, i.e., median or LAD ("least absolute deviations") regression. We can compare it to the OLS regression on mean.

```
> cps_ols=lm(cps_f, data=CPS1988)
> summary(cps_lad)
> summary(cps_ols)
```

The quantile regression is extremely useful when modeling several quantiles of the response $y$ simultaneously. Let's consider the *first* ($\tau$=0.25) and *third* ($\tau$=0.75) quartiles.

```
> cps_rq=rq(cps_f, tau=c(0.25, 0.75), data=CPS1988)
> summary(cps_rq)
```

A natural question is whether the regression lines are parallel. That is, are the effects of the regressors (i.e., education and experience) are uniform across quantiles?

```
> cps_rq25=rq(cps_f, tau=0.25, data=CPS1988)
> cps_rq75=rq(cps_f, tau=0.75, data=CPS1988)
> anova(cps_rq25, cps_rq75)
```

What is the key finding here?

We can further visualize the results from quantile regression modeling.

```
> cps_rqbig=rq(cps_f, tau=seq(0.05, 0.95, 0.05), data=CPS1988)
> cps_rqbigs=summary(cps_rqbig)
> plot(cps_rqbigs)
```

Now, can you see the drawback of relying merely on the OLS regression (*lm*( )) that models the conditional expectation of $y$ on $x$?

Due to time limits, the best I can do is to provide you an overview of GLM as opposed to in-depth discussion of each subsection. In fact, each type of regression models studied here can stand alone for a whole semester course. I do encourage you to pursue further readings if you are highly interested in any particular domain.

The key point is, in addition to the classical linear regression in lecture 6, you need to realize there is whole world out there. Your job as an analyst is to have a deep understanding about the problem domain and data properties. Then you should specify *relevant models* and apply *appropriate techniques* to perform analysis so that you can derive useful insights and reach meaningful conclusions in the end.