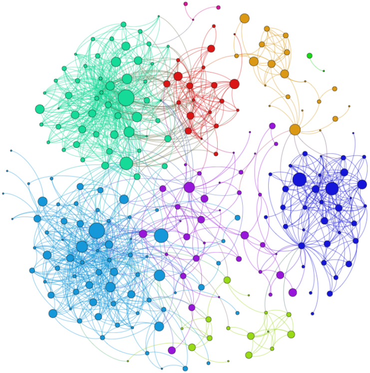
Classification and Discrete Choice Models

Jiaming Mao
Xiamen University



This version: Fall 2021

Contact: jmao@xmu.edu.cn

Course homepage: jiamingmao.github.io/data-analysis

Classification

Classification is a predictive task in which the response variable y is discrete or categorical[1].

Examples:

- Is a credit card user going to default?
- Is a project going to be successful?
- Which product will a consumer buy?
- Which market will a firm enter?
- Which political candidate will an individual vote for?

[1] y is **discrete** if it takes on a set of discrete numerical values. y is **categorical** if it belongs to a set of **categories** (also called **classes**).

Binary Classification

For binary classification problems, let y be coded as $\{0, 1\}$.

We can try to model y using the following linear regression model:

$$y = x'\beta + e \tag{1}$$

Estimating (1) $\Rightarrow \widehat{\beta}$. Then given a data point x_0, we would classify y_0 as

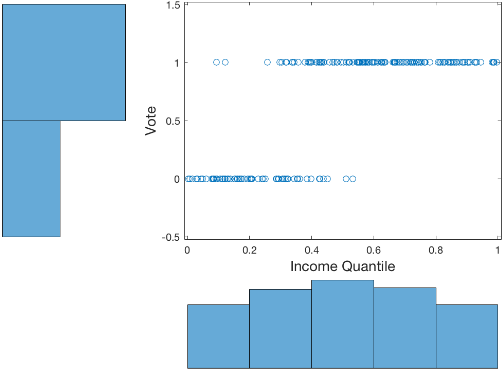
$$\widehat{y}_0 = \begin{cases} 1 & \text{if } x_0'\widehat{\beta} > \frac{1}{2} \\ 0 & \text{o.w.} \end{cases}$$

, which yields the decision boundary: $x'\widehat{\beta} - \frac{1}{2} = 0$.

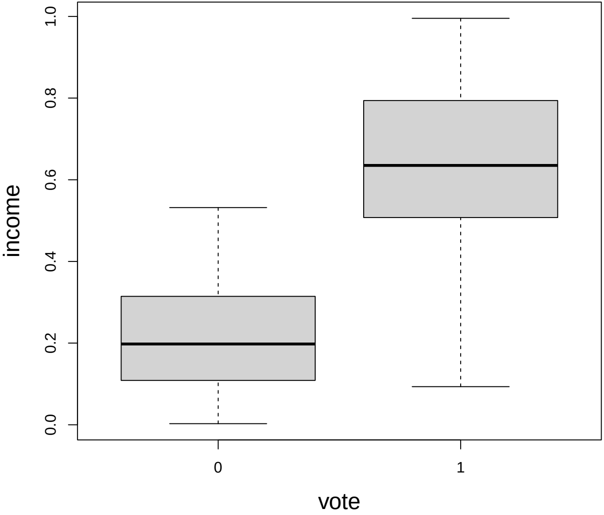
Income and Voting

Data: income and voting records of 200 voters

- income: income quantile
- vote: whether voted in the last election



Income and Voting



Income and Voting

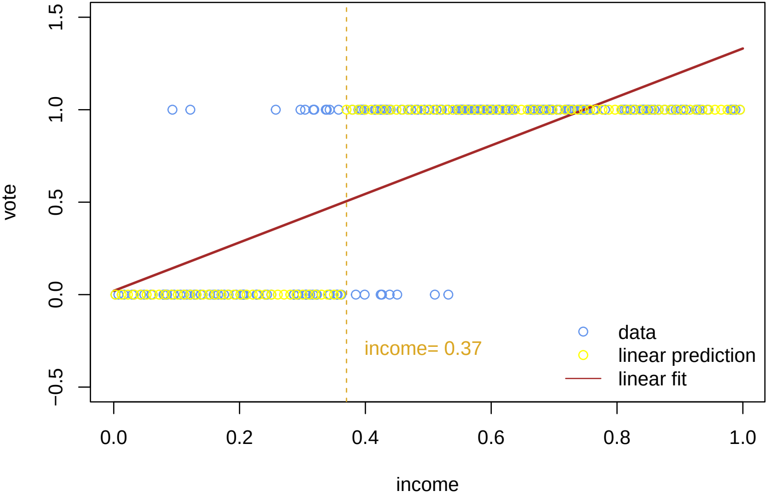
```r
require(AER)
attach(read.csv("voting.txt"))
coeftest(lm(vote ~ income))

##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.020419   0.047237  0.4323    0.666
## income      1.310588   0.083000 15.7902   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Income and Voting

To predict vote at income $= 0.5$:

```
x0 <- data.frame(income=.5)
f_hat <- predict(lm(vote ~ income),x0)
f_hat

##         1
## 0.6757133

vote_hat <- as.numeric(f_hat>.5)
vote_hat

## [1] 1
```

Income and Voting



The least squares linear regression method is not a probabilistic model[2]. The probabilistic approach to classify y is to first estimate $p(y|x)$ and then let

$$\widehat{y}(x) = \arg\max_{c\in\{0,1\}} \{\widehat{p}(y=c|x)\} = \begin{cases} 1 & \text{if } \widehat{p}(y=1|x) > \frac{1}{2} \\ 0 & \text{o.w.} \end{cases} \tag{2}$$

(2) is the Bayes classifier with decision boundary given by $\widehat{p}(y=1|x) - \frac{1}{2} = 0$.

[2]Although it is possible to give (1) a probabilistic reading: notice that when $y \in \{0,1\}$, $E(y|x) = 1 \cdot \Pr(y=1|x) + 0 \cdot \Pr(y=0|x) = \Pr(y=1|x)$. Hence one can interpret the least squares linear regression estimate $x'\widehat{\beta}$ as an estimate of $\Pr(y=1|x)$. However, since $x'\widehat{\beta}$ is not bounded by $[0,1]$, it is not a proper probabilistic model.

Logistic Regression

The logistic regression model assumes[3]

$$\Pr(y=1|x) = \sigma(x'\beta) = \frac{\exp(x'\beta)}{1+\exp(x'\beta)} \tag{3}$$

, where $\sigma(z) \equiv (1+e^{-z})^{-1}$ is called the **logistic function** or **sigmoid function**[4].

[3] More precisely, the logistic regression model is a discriminative probabilistic model with $p(y|x)$ as the target function and $\mathcal{H} = \{q(y|x) : q(y=1|x) = \sigma(x'\beta)\}$, i.e.,

$$\Pr(y|x) = p(y|x) \qquad \text{true distribution}$$

$$\Pr(y|x) = q(y|x) = \begin{cases} \sigma(x'\beta) & y=1 \\ 1-\sigma(x'\beta) & y=0 \end{cases} \qquad \text{hypothesis}$$

[4] The logistic function defines the CDF of the **standard logistic distribution**:

$$\mathcal{F}(x) = \frac{\exp(x)}{1+\exp(x)}$$

Logistic Regression

(3) $\Rightarrow$

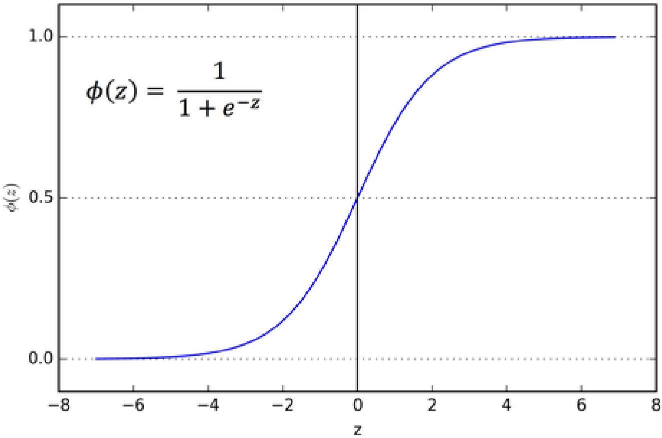
$$\log \frac{\Pr(y=1|x)}{\Pr(y=0|x)} = x'\beta$$

- Logistic regression assumes that the log odds is a linear function[5,6].

[5] If p denotes the probability of "success", then $\frac{p}{1-p}$ is the *odds* of success.

[6] The function $g(p) = \log \frac{p}{1-p}$ – inverse of the sigmoid – is called the **logit function**.

Logistic Regression



Sigmoid Function

Logistic Regression

The logistic regression model can be estimated by maximum likelihood. Given data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$,

$$\widehat{\beta} = \arg\max_{\beta} \log \mathcal{L}(\beta) \tag{4}$$

, where

$$\begin{aligned}
\log \mathcal{L}(\beta) &= \sum_{i=1}^{N} \log \Pr(y_i | x_i; \beta) \\
&= \sum_{i=1}^{N} \left[y_i \log \sigma(x_i'\beta) + (1 - y_i) \log(1 - \sigma(x_i'\beta)) \right]
\end{aligned}$$

Logistic Regression

Equivalently, logistic regression minimizes the cross-entropy error[7]:

$$E_{in}(\beta) = -\frac{1}{N}\sum_{i=1}^{N} \log \Pr\left(y_i \middle| x_i; \beta\right) \tag{5}$$

Note that $E_{in}(\beta)$ is *convex* and *differentiable*,

$$\nabla E_{in}(\beta) = \frac{1}{N}\sum_{i=1}^{N} \left(\sigma\left(x_i'\beta\right) - y_i\right) x_i$$

[7] Recall that given true distribution $p(y|x)$ and hypothesis $q(y|x)$, cross-entropy

$$\mathbb{H}(p, q) = -\sum_{x} p\left(y \middle| x\right) \log q\left(y \middle| x\right)$$

, with the in-sample expression being $-\frac{1}{N}\sum_{i=1}^{N} \log q\left(y_i \middle| x_i\right)$.

[8]

[8] Let $y \in \{-1, 1\}$, then

$$\Pr\left(y|\, x; \beta\right) = \begin{cases} \sigma\left(x'\beta\right) & y = 1 \\ 1 - \sigma\left(x'\beta\right) & y = -1 \end{cases} = \sigma\left(y \cdot x'\beta\right)$$

, where we use the fact that $\sigma(-z) = 1 - \sigma(z)$. Therefore, (5) can be written as

$$\begin{aligned} E_{in}(\beta) &= -\frac{1}{N}\sum_{i=1}^{N} \log \sigma\left(y_i \cdot x_i'\beta\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \underbrace{\log\left(1 + \exp\left(-y_i \cdot x_i'\beta\right)\right)}_{\textbf{binomial cross-entropy loss}} \end{aligned} \tag{6}$$

Logistic Regression

```r
# generate data
require(sigmoid)
n <- 1000
x <- rnorm(n)
p <- sigmoid(x) # true beta = 1
y <- rbinom(n,1,p) # y = {0,1} with probability p
```

Logistic Regression

```
require(AER)
fit <- glm(y ~ x,family=binomial)
coeftest(fit)

##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.109388   0.069492 -1.5741   0.1155
## x            0.989909   0.083548 11.8484   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Logistic Regression

```r
##################################
## Maximum Likelihood Estimation #
##################################
# We can obtain the solution by manually defining
# the negative log likelihood function and minimizing it

# negative log likelihood function
nll <- function(beta){
  h <- sigmoid(x*beta)
  nll <- -sum((y*log(h)) + ((1-y)*log(1-h)))
}

# perform optimization
betahat <- optim(c(0),nll)$par
print(betahat)

## [1] 0.9867188
```

Logistic Regression

```
#####################
## Gradient Descent #
#####################
# We can manually optimize by gradient descent

# cost function (= nll)
cost <- function(X,y,beta){
  N <- length(y)
  h <- sigmoid(X%*%beta)
  cost <- -sum((y*log(h)) + ((1-y)*log(1-h)))/N
}

# gradient function
grad <- function(X,y,beta){
  N <- length(y)
  h <- sigmoid(X%*%beta)
  grad = (t(X)%*%(h-y))/N
}
```

Logistic Regression

```r
# Gradient descent algorithm
## eta: learning rate
## niter: number of iterations
gradientDescent <- function(X,y,beta0,eta,niter){
  beta <- beta0
  cost_hist <- rep(0,niter)
  beta_hist <- list(niter)
  for (i in 1:niter){
    beta_hist[[i]] <- beta
    cost_hist[i] <- cost(X,y,beta)
    beta <- beta - eta*grad(X,y,beta) # update
  }
  result <- list("beta"=beta,"cost_hist"=cost_hist,"beta_hist"=beta_hist)
  return(result)
}
```

Logistic Regression

```r
# estimation
## initial guess: 0; learning rate: 0.1; iteration: 500
X <- cbind(x) # make x column vector
result <- gradientDescent(X,y,0,0.1,500)
print(result$beta)

##         [,1]
## x 0.9858559
```

Logistic Regression

Given an estimated logistic regression model, at any data point x_0, we classify y_0 to be

$$\widehat{y}_0 = \begin{cases} 1 & \text{if } \widehat{p}\left(y_0 = 1 | x_0\right) = \sigma\left(x_0'\widehat{\beta}\right) > \frac{1}{2} \\ 0 & \text{o.w.} \end{cases}$$

Note that this is equivalent to the decision rule:

$$\widehat{y}_0 = \begin{cases} 1 & \text{if } \log \frac{\widehat{p}(y_0 = 1|x_0)}{\widehat{p}(y_0 = 0|x_0)} = x_0'\widehat{\beta} > 0 \\ 0 & \text{o.w.} \end{cases}$$

, i.e., logistic regression yields the decision boundary: $x'\widehat{\beta} = 0$[9].

[9] For this reason, logistic regression is considered a *linear* classification model.

Income and Voting

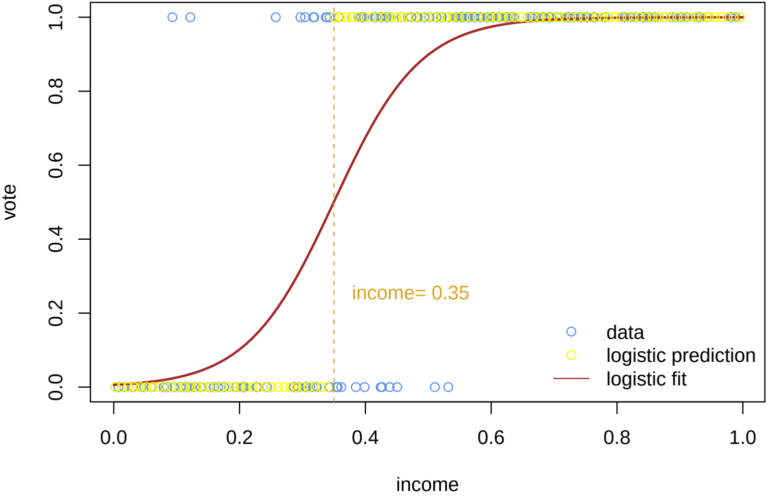
```
logitfit <- glm(vote ~ income, family=binomial)
coeftest(logitfit)

##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -5.08565    0.86061 -5.9093 3.435e-09 ***
## income      14.53879    2.24278  6.4825 9.023e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Income and Voting

To predict vote at income $= 0.5$:

```
x0 <- data.frame(income=.5)
p_hat <- predict(logitfit,x0,type="response")
p_hat

##         1
## 0.8987804

vote_hat <- as.numeric(p_hat>.5)
vote_hat

## [1] 1
```

Income and Voting



Linear vs. Logistic Regression

Both linear and logistic regression can be thought of as belonging to a general approach that models a **score function** $\delta_j(x)$[10] for each class j and classify y to be $y = \arg\max_j \{\delta_j(x)\}$.

- Linear regression: $\begin{cases} \delta_0(x) = 1 - x'\beta \\ \delta_1(x) = x'\beta \end{cases}$
- Logistic regression: $\begin{cases} \delta_0(x) = 1 - \sigma(x'\beta) \\ \delta_1(x) = \sigma(x'\beta) \end{cases}$
- Decision boundary: $\{x : \delta_0(x) = \delta_1(x)\}$

The score functions for logistic regression have probabilistic interpretations as models of $\Pr(y = j|x)$.

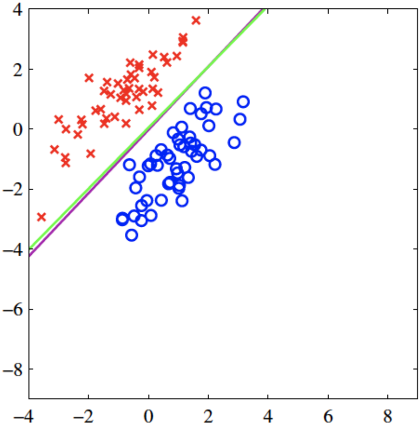
[10] Also called **discriminant function**.

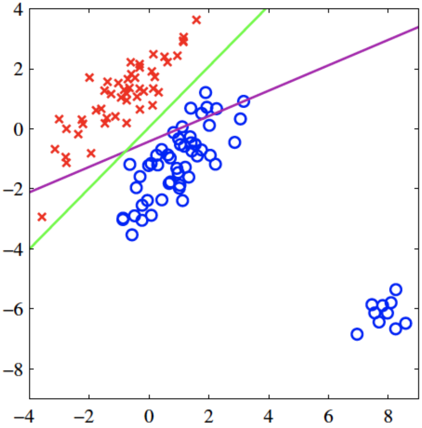
- Compared to logistic regression, linear regression can be less robust due to the L2 loss function that it uses.

- When estimating (1) using least squares, the method seeks to find $\widehat{\beta}$ such that each $x_i'\widehat{\beta}$ is as close to y_i as possible, even though all we need is for $\mathcal{I}\left(x_i'\widehat{\beta} > \frac{1}{2}\right)$ to be the same as y_i.

- In particular, the L2 loss penalizes cases in which $y_i = 1$ and $x_i'\widehat{\beta} \gg 1$, or $y_i = 0$ and $x_i'\widehat{\beta} \ll 0$, i.e. the loss function penalizes predictions that are "too correct".

Linear vs. Logistic Regression





Data from two classes are denoted by red crosses and blue circles, with decision boundaries found by least squares (magenta) and logistic regression (green). Least squares can be highly sensitive to outliers, unlike logistic regression.

Loss Functions for Classification

Let y be coded as $\{-1, 1\}$. The logistic regression can also be thought of as a linear model $\mathcal{H} = \{h(x) = x'\beta\}$ that minimizes an in-sample error based on the binomial cross-entropy loss[11]:

$$\ell^{\mathsf{CE}}(h(x), y) = \log(1 + \exp(-y \cdot h(x))) \tag{7}$$

Least squares linear regression, on the other hand, minimizes the L2 loss:

$$\ell^{\mathsf{L2}}(h(x), y) = (y - h(x))^2 = (y \cdot h(x) - 1)^2 \tag{8}$$

[11] See page 16

Loss Functions for Classification

Now consider a linear model for classification that minimizes the empirical misclassification rate:

$$E_{in} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}\left(\ell^{01}\left(h\left(x_i\right), y_i\right)\right) \tag{9}$$

, where ℓ^{01} is the $0-1$ loss:

$$\ell^{01}\left(h\left(x\right), y\right) = \mathcal{I}\left(y \neq \text{sign}\left(h\left(x\right)\right)\right) = \mathcal{I}\left(y \cdot h\left(x\right) < 0\right) \tag{10}$$

Such a model is called the **perceptron**[12,13].

- Minimizing (9) is NP hard[14].

[12] With $\{-1, 1\}$ target, the perceptron model could also be written as $\mathcal{H} = \{h\left(x\right) = \text{sign}\left(x'\beta\right)\}$ that minimizes the loss function $\mathcal{I}\left(y \neq \left(h\left(x\right)\right)\right)$.

[13] We will formally discuss the perceptron model when we introduce neural networks.

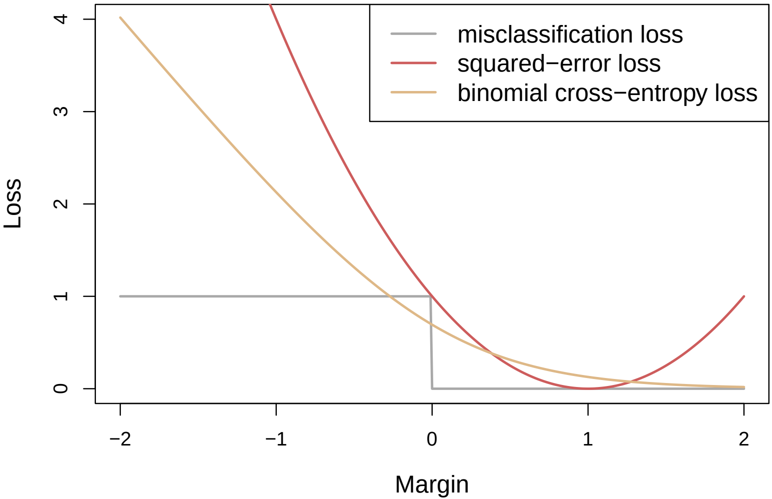
[14] Meaning: there is no efficient algorithm to solve the problem.

Loss Functions for Classification

The loss functions (7), (8), and (10) are all functions of the **margin** $y \cdot h(x)$.

- Positive margin: correct classification ☺. Negative margin: incorrect classification ☹. Decision boundary: $h(x) = 0$.
- The goal of a classification algorithm should be to produce positive margins as frequently as possible.
- Both ℓ^{01} and ℓ^{CE} are *decreasing* functions of the margin. ℓ^{CE} can be viewed as a monotone continuous approximation to ℓ^{01}.
- ℓ^{L2}, however, is *not* a decreasing function of the margin. It penalizes observations with large positive margins and hence is not a suitable loss function for classification.

Loss Functions for Classification



Logistic Regression for Aggregate Outcomes

In addition to binary classification, logistic regression is suitable for regression problems where the response variable is the sum of individual binary outcomes.

The model is[15]:

$$\begin{aligned} y_i &\sim \text{Binomial}\left(n_i, \pi_i\right) \\ \pi_i &= \sigma\left(x_i'\beta\right) \end{aligned} \tag{11}$$

[15]The logistic model for binary classification can be similarly written as:

$$y_i \sim \text{Binomial}\left(1, \sigma\left(x_i'\beta\right)\right) = \text{Bernoulli}\left(\sigma\left(x_i'\beta\right)\right)$$

Logistic Regression for Aggregate Outcomes

The log likelihood function is:

$$\begin{aligned}
\log \mathcal{L}(\beta) &= \sum_{i=1}^{N} \log \left(\left(\begin{array}{c} n_i \\ y_i \end{array} \right) [\pi_i(\beta)]^{y_i} [1 - \pi_i(\beta)]^{n_i - y_i} \right) \\
&\propto \sum_{i=1}^{N} [y_i \log \pi_i(\beta) + (n_i - y_i) \log (1 - \pi_i(\beta))] \\
&= \sum_{i=1}^{N} \left[y_i \log \sigma\left(x_i'\beta\right) + (n_i - y_i) \log \left(1 - \sigma\left(x_i'\beta\right)\right) \right]
\end{aligned}$$

Generalized Linear Models

The logistic regression model belongs to a class of **generalized linear models** (**GLM**). A GLM assumes that the response variable y comes from a known exponential family with mean μ, and

$$g(\mu) = x'\beta$$

, where $g(.)$ is a *monotonic* function called the **link function**.

Generalized Linear Models

- Normal linear model: Normal distribution with the identity link

$$
\begin{aligned}
y &\sim \mathcal{N}\left(\mu, \sigma^2\right) \\
\mu &= x'\beta
\end{aligned}
$$

- Logistic model: Bernoulli/Binomial distribution with the logit link

$$
y \sim \text{Binomial}\left(n, \pi\right)
$$

$$
\log\left(\frac{\pi}{1-\pi}\right) = x'\beta
$$

- Poisson model: Poisson distribution with the log link

$$
y \sim \text{Poisson}\left(\mu\right)
$$

$$
\log \mu = x'\beta
$$

Dose Response

Five groups of animals were exposed to a dangerous substance in varying concentrations. Let n_i be the number of animals and y_i the number that died in group i.

Concentration	$\log_{10}$ conc	n_i	y_i	p_i
1×10^{-5}	-5	6	0	0.000
1×10^{-4}	-4	6	1	0.167
1×10^{-3}	-3	6	4	0.667
1×10^{-2}	-2	6	6	1.000
1×10^{-1}	-1	6	6	1.000

How to model y_i as a function of $\log$ conc?

Dose Response

```
########################
# Logistic Regression #
########################
require(AER)
y <- c(0,1,4,6,6)
n <- c(6,6,6,6,6)
logconc <- c(-5,-4,-3,-2,-1)
logitfit <- glm(cbind(y,n-y) ~ logconc, family=binomial)
coeftest(logitfit)

##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   9.5868     3.7067  2.5864 0.009699 **
## logconc       2.8792     1.1023  2.6121 0.008999 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dose Response

Let $p_i = y_i / n_i$ be the *observed* proportion that died in group i. Can we run linear regression of p_i on $\log \text{conc}$? i.e.,

$$p_i = x_i' \beta + e_i$$

Yes, but the linear model may generate predictions outside the range of $[0, 1]$ ...

Dose Response

Better: let

$$z_i \doteq \log \frac{p_i}{1-p_i}$$

and regress

$$z_i = x_i'\beta + e_i \tag{12}$$

When n_i is large, model (12) $\rightarrow$ the logistic model (11).

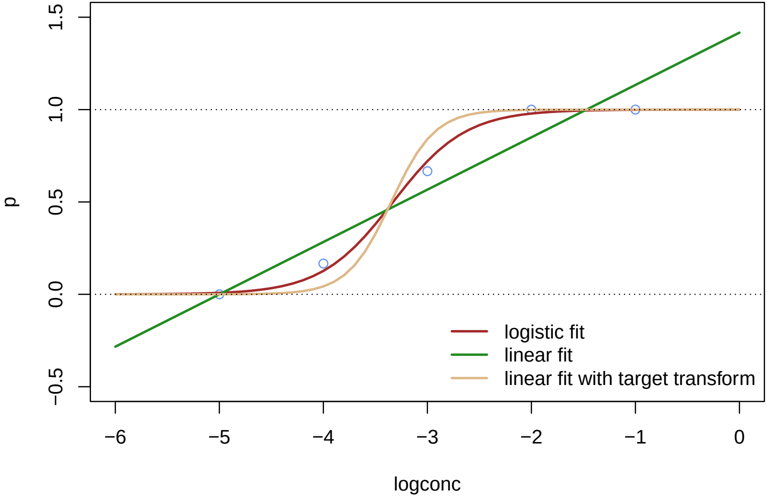
Dose Response

```
########################################
# Linear Regression: p = x'*beta + e #
########################################
p <- y/n
lsfit1 <- lm(p ~ logconc)
coeftest(lsfit1)

##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.416667   0.153055  9.2559 0.002668 **
## logconc     0.283333   0.046148  6.1397 0.008690 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dose Response

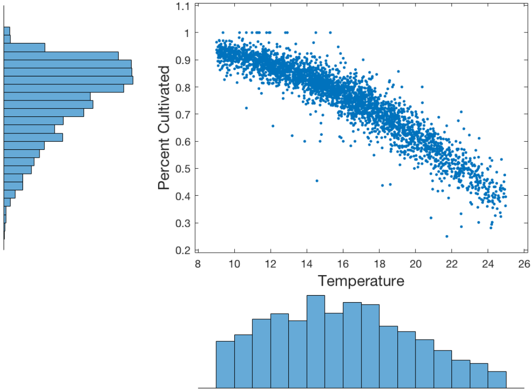
```
############################################
# Linear Regression with Target Transform #
# z = x'*beta + e, where z = log(p/(1-p)) #
############################################
# Since some p=0 and some p=1, we add a small number eps to p=0,
# and subtract eps from p=1, to avoid log(p/(1-p)) being undefined.
# Note: when n is small, regression results are highly sensitive to eps
eps <- 1e-4
p[p==0] <- p[p==0] + eps
p[p==1] <- p[p==1] - eps
z <- log(p/(1-p))
lsfit2 <- lm(z ~ logconc)
coeftest(lsfit2)

##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.95698    2.47044  6.4592 0.007528 **
## logconc      4.76606    0.74487  6.3986 0.007732 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dose Response



Cropland

Data on 3144 counties, including agricultural land (fields) available in each county, the number of fields that are being cultivated, and the annual average temperature of each county.



Cropland

```r
cropland <- read.csv("cropland.txt")
attach(cropland)
head(cropland)

##   temperature fields cultivated percentCultivated
## 1    13.18475     63         49         0.7777778
## 2    12.35680    165        147         0.8909091
## 3    17.57882     38         30         0.7894737
## 4    20.86867    152         95         0.6250000
## 5    13.88084     88         69         0.7840909
## 6    17.18088    191        141         0.7382199
```

Cropland

```r
########################
# Logistic Regression #
########################
require(AER)
logitfit <- glm(cbind(cultivated, fields-cultivated) ~ temperature,
                family=binomial)
coeftest(logitfit)

##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  4.266957   0.017392  245.34 < 2.2e-16 ***
## temperature -0.189233   0.000990 -191.14 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

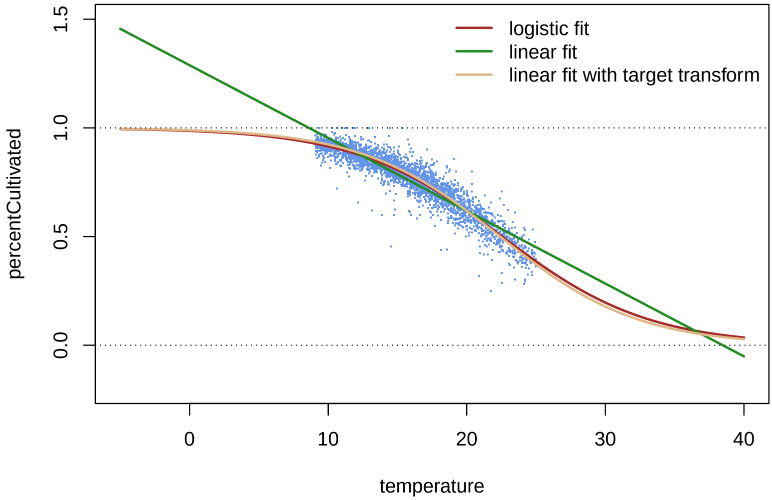
Cropland

```
#####################
# Linear Regression #
#####################
lsfit <- lm(percentCultivated ~ temperature)
coeftest(lsfit)

##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  1.28838143  0.00383395  336.05 < 2.2e-16 ***
## temperature -0.03349385  0.00023385 -143.23 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Cropland

```
########################################
# Linear Regression with Target Transform #
########################################
p <- percentCultivated
eps <- 1e-4
p[p==1] <- p[p==1] - eps
lsfit2 <- lm(log(p/(1-p)) ~ temperature)
coeftest(lsfit2)

##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  4.5086192  0.0430642 104.695 < 2.2e-16 ***
## temperature -0.2012857  0.0026266 -76.632 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Cropland



logistic fit
linear fit
linear fit with target transform
percentCultivated
temperature
0.0
0.5
1.0
1.5
0
10
20
30
40

Classification Error

A binary classifier can make two types of errors:

- **False positive rate** (**FPR**): $\Pr\left(\widehat{y}=1 \middle| y=0\right)$
- **False negative rate** (**FNR**): $\Pr\left(\widehat{y}=0 \middle| y=1\right)$

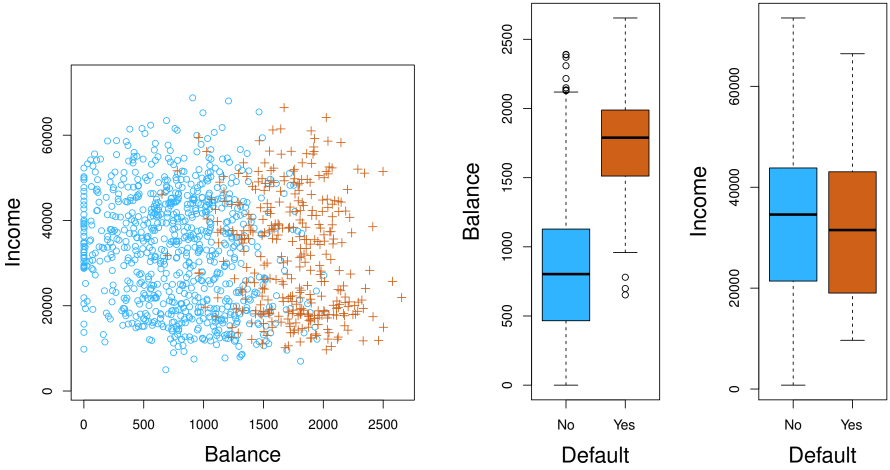
The **sensitivity** of the classifier is $\Pr\left(\widehat{y}=1 \middle| y=1\right)$ and the **specificity** of the classifier is $\Pr\left(\widehat{y}=0 \middle| y=0\right)$.

Classification Error

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

Credit Card Default



Credit Card Default

```r
require(ISLR) # contains the data set 'Default'
attach(Default)
Default <- Default[,-2]
head(Default)
```

```
##   default   balance    income
## 1      No  729.5265 44361.625
## 2      No  817.1804 12106.135
## 3      No 1073.5492 31767.139
## 4      No  529.2506 35704.494
## 5      No  785.6559 38463.496
## 6      No  919.5885  7491.559
```

Credit Card Default

```r
require(AER)
logitfit <- glm(default ~., data=Default, family=binomial)
coeftest(logitfit)

##
## z test of coefficients:
##
##                Estimate  Std. Error  z value  Pr(>|z|)
## (Intercept) -1.1540e+01  4.3476e-01 -26.5447 < 2.2e-16 ***
## balance      5.6471e-03  2.2737e-04  24.8363 < 2.2e-16 ***
## income       2.0809e-05  4.9852e-06   4.1742 2.991e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Credit Card Default

```
cutoff <- .5
logit.p <- logitfit$fit
logit.y <- as.factor(logit.p > cutoff)
levels(logit.y) <- c("No","Yes")
t <- table(logit.y,default,dnn=c("predicted default","true default"))
t

##                    true default
## predicted default    No  Yes
##                No  9629  225
##                Yes   38  108

prop.table(t,2)

##                    true default
## predicted default            No         Yes
##                No  0.996069101 0.675675676
##                Yes 0.003930899 0.324324324
```

Credit Card Default

- Overall error rate: $(225 + 38)/10,000 = 2.63\%$
- FPR: 0.39%. Specificity: 99.61%
- FNR: 67.57%. Sensitivity: 32.43%

- Note that only $333/10,000 = 3.33\%$ individuals defaulted in the data. Hence a simple but useless *null* classifier that always predicts "No" will result in an error rate of 3.33%.
- From the perspective of a credit card company that is trying to identify high-risk individuals, the FNR – not the overall error rate – is what's important.
  - Incorrectly classifying an individual who will not default, though still to be avoided, is less problematic.

Credit Card Default

- In binary classification, the Bayes classifier assigns $\widehat{y} = 1$ if $p(y = 1|x) > 0.5$ – here 0.5 is used as a threshold in order to classify $\widehat{y} = 1$ based on $p(y = 1|x)$.
- Recall that we can use different loss functions[16] to control which type of error we want to minimize: the overall error rate, FPR, or FNR. This is equivalent to changing the threshold for classifying $\widehat{y} = 1$.
- If we are more concerned about FNR, then we can lower this threshold. For example, if we use 0.1 as the threshold, then we assign $\widehat{y} = 1$ if $p(y = 1|x) > 0.1$[17].

[16] other than the $0 - 1$ loss which gives us the Bayes classifier.

[17] This is equivalent to using the loss function: $\ell(y, \widehat{y}) = 9$ if $(y, \widehat{y}) = (1, 0)$, $\ell(y, \widehat{y}) = 1$ if $(y, \widehat{y}) = (0, 1)$, and $\ell(y, \widehat{y}) = 0$ otherwise.

Credit Card Default

```
cutoff <- .1
logit.y <- as.factor(logit.p > cutoff)
levels(logit.y) <- c("No","Yes")
t <- table(logit.y,default,dnn=c("predicted default","true default"))
t

##                   true default
## predicted default   No  Yes
##               No  9105   90
##               Yes  562  243

prop.table(t,2)

##                   true default
## predicted default         No        Yes
##               No  0.94186407 0.27027027
##               Yes 0.05813593 0.72972973
```

Credit Card Default

- Overall error rate: $(90 + 562)\,/10,000 = 6.52\%$
- FPR: 5.81%. Specificity: 94.19%
- FNR: 27.03%. Sensitivity: 72.97%

Credit Card Default

```r
cutoff <- .01
logit.y <- as.factor(logit.p > cutoff)
levels(logit.y) <- c("No","Yes")
t <- table(logit.y,default,dnn=c("predicted default","true default"))
t
```

```
##                    true default
## predicted default    No  Yes
##               No   7134   10
##               Yes  2533  323
```

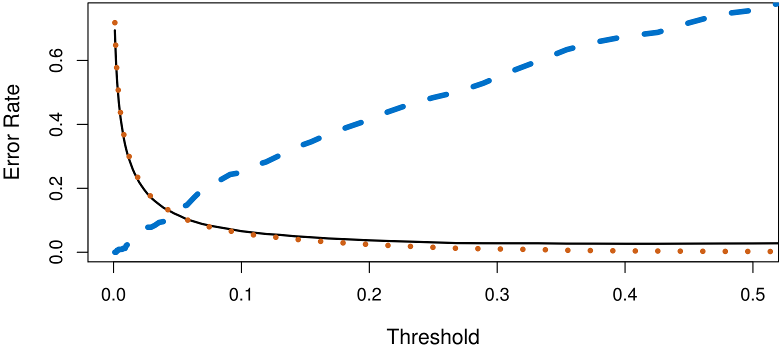
```r
prop.table(t,2)
```

```
##                    true default
## predicted default          No        Yes
##               No   0.73797455 0.03003003
##               Yes  0.26202545 0.96996997
```

Credit Card Default

- Overall error rate: $(10 + 2533)/10,000 = 25.43\%$
- FPR: 26.20%. Specificity: 74.80%
- FNR: 3.00%. Sensitivity: 97.00%

Credit Card Default



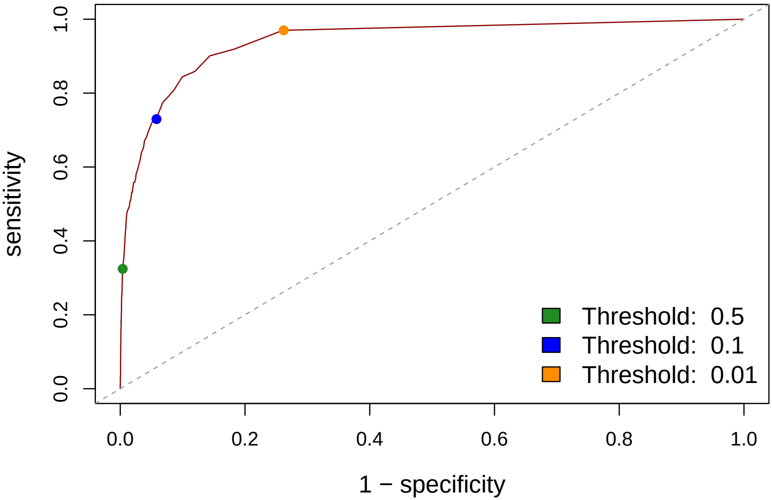
Black solid line: overall error rate; Orange dotted line: FPR;
Blue dashed line: FNR.

The ROC Curve

- The **ROC curve** displays sensitivity (1−FNR) vs 1−specificity (FPR) for *all* possible thresholds.
- The overall performance of a classifier, summarized over all possible thresholds, is given by the **area under the curve** (**AUC**).
- An ideal ROC curve hugs the top left corner (high sensitivity, high specificity): *the larger the AUC the better the classifier*.
- ROC curves are useful for comparing different classifiers[18].

[18] Note that the error rates we have calculated so far are *training* errors. More rigorously, error rates should be calculated and compared on a *test* data set.

Credit Card Default



Similarity-Based Methods

- One way to classify data is to assign a new input the class of the most similar input(s) in the data. This is called the **nearest neighbor** method.

- The nearest neighbor method is a **similarity-based method**. These methods are *model free* and hence *nonparametric*.

KNN

- Given an input x, the **K-nearest neighbors** (**KNN**) classifier finds the K points that are closest in distance to x[19], denoted by $\mathcal{N}_K(x) = \left\{x_{(1)}, \ldots, x_{(K)}\right\}$, and then classify using **majority vote**: let y be the most common class among $\left\{y_{(1)}, \ldots, y_{(K)}\right\}$[20].

- Equivalently, the KNN classifier can be thought of as first estimating

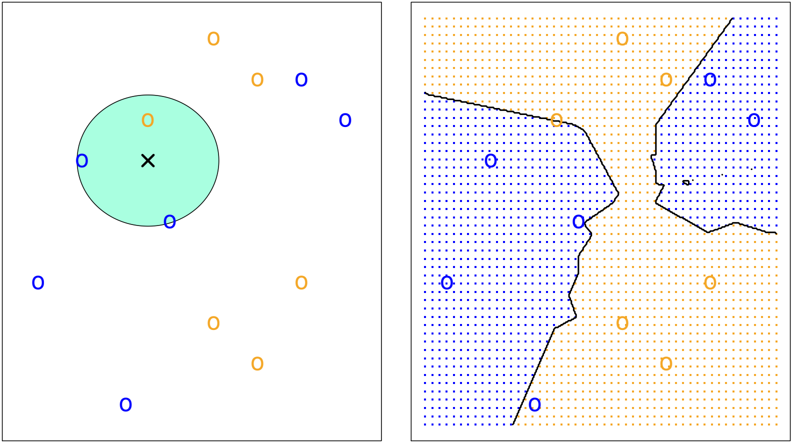
$$\widehat{p}(y = j|x) = \frac{1}{K} \sum_{i \in \mathcal{N}_K(x)} \mathcal{I}(y_i = j)$$

, where $y \in \{1, \ldots, J\}$, and then applying the Bayes classifier.

[19] To do this, we need a **distance measure**, or **similarity measure**. For real-valued inputs, the common choice is to use the Euclidean distance: $d(x, x') = \|x - x'\|$.

[20] Ties are broken at random.

KNN



KNN in two dimensions ($K = 3$)

Credit Card Default

```r
#######
# KNN #
#######
# To perform KNN classification, we first standardize the x variables
# so that all variables have mean zero and standard deviation one.
# Furthermore, let's split our sample into a training data set
# and a test data set, fit the model on the training data,
# and test its performance on the test data.

# standardization
s.balance <- scale(balance)
s.income <- scale(income)
SX <- data.frame(s.balance,s.income) # standardized x variables

# create training and test data
test <- sample(1:nrow(Default),2000) # sample 2000 random indices
TR.SX <- SX[-test,] # training X
TE.SX <- SX[test,] # test X
TR.y <- default[-test] # training y
TE.y <- default[test] # test y
```

Credit Card Default

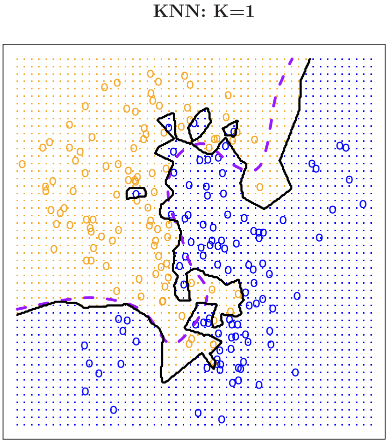
```
require(class)
require(gmodels)
K <- 5 # K value
knn.pred <- knn(TR.SX,TE.SX,TR.y,k=K,prob=TRUE)
r <- table(knn.pred,TE.y,dnn=c("predicted default","true default"))
print(r)

##                    true default
## predicted default   No  Yes
##               No  1918   35
##               Yes   22   25

err <- (r[2,1] + r[1,2])/sum(r) # overall error rate
fpr <- r[2,1]/(r[1,1] + r[2,1]) # false positive rate
fnr <- r[1,2]/(r[1,2] + r[2,2]) # false negative rate
c(err,fpr,fnr)

## [1] 0.02850000 0.01134021 0.58333333
```

KNN

In choosing K, we face a bias-variance tradeoff:

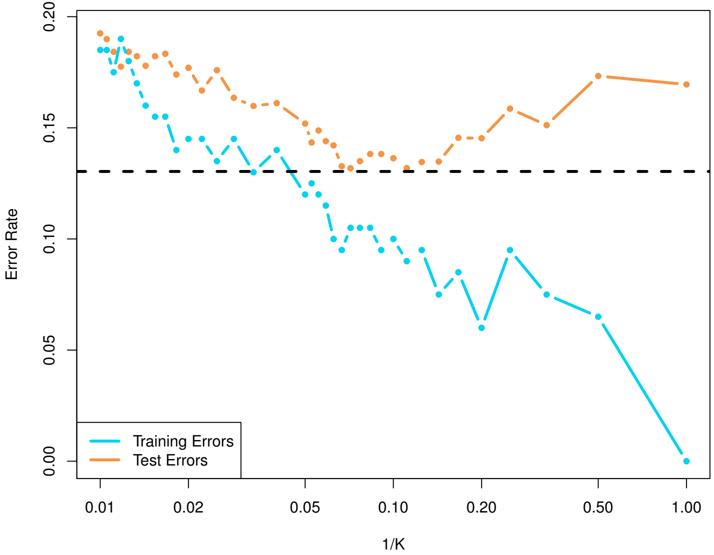
- With K $=$ 1, the KNN training error rate is 0. Bias is low and variance is high.
- As K grows, the method becomes less flexible and produces a decision boundary that is closer to linear, with lower variance and higher bias.

KNN



Black curve: KNN decision boundary. Purple curve: Bayes decision boundary (decision boundary based on the Bayes classifier and the true $p(y|x)$)

KNN



0.20
0.15
0.10
0.05
0.00
Error Rate
Training Errors
Test Errors
0.01
0.02
0.05
0.10
0.20
0.50
1.00
1/K

Parametric vs. Nonparametric Methods

- KNN is a nonparametric (model-free) method. In general, these methods can work well for prediction in a wide variety of situations, since they don't make any real assumptions.

- The downside is that they are essentially a black box and lack interpretability. They are also more *computationally expensive* since they typically need to store the *entire* data and use them whenever predicting on a new point.
  - In contrast, parametric methods summarize the data with a fixed set of parameters, which are sufficient for prediction[21].

- In addition, KNN suffers from the curse of dimensionality: given N, when p is large[22], data become relatively *sparse*. In high dimensions, the neighborhood represented by the K nearest points may not be local.

[21] Fundamentally, a parametric model is a compression of data.

[22] p being the dimension of the input space.

Multiclass Classification

For multiclass problems, let y be coded as $\{1, \ldots, J\}$. The methods of binary classification extends naturally to the multiclass setting.

Let $\delta_j(x)$ be the score function for class j. For linear regression, $\delta_j(x) = x'\beta_j$. Define $y^j = \mathcal{I}(y = j)$. Then we have the following J regression equations:

$$y^j = x'\beta_j + e_j, \quad j = 1, \ldots, J \tag{13}$$

Estimating (13) $\Rightarrow \left\{\widehat{\beta}_j\right\}_{j=1}^{J}$. Given a data point x_0, we classify y_0 to be:

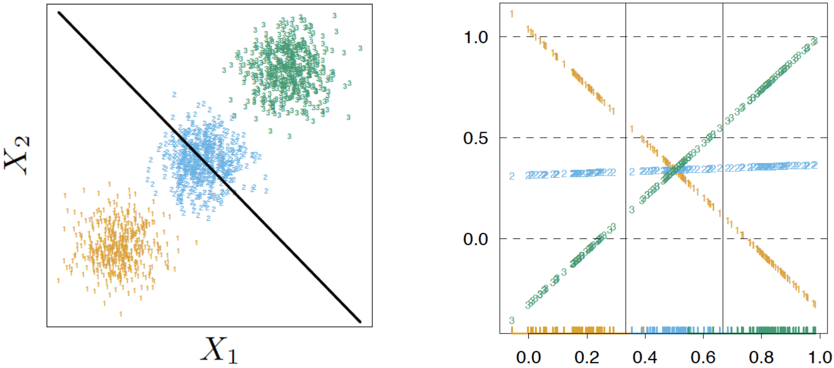
$$y_0 = \arg\max_j \left\{x_0'\widehat{\beta}_j\right\}$$

Linear Regression

- In addition to a lack of robustness, the linear regression approach can have serious problems dealing with multiclass problems ($J \geq 3$). Classes can be masked by others – particularly when J is large and p is small.

- This is not surprising: recall that the least squares estimate corresponds to the estimate of a normal linear model. Binary targets like y^j, however, clearly have a distribution that is far from Gaussian. Hence we obtain better classification results by adopting more appropriate probabilistic models.

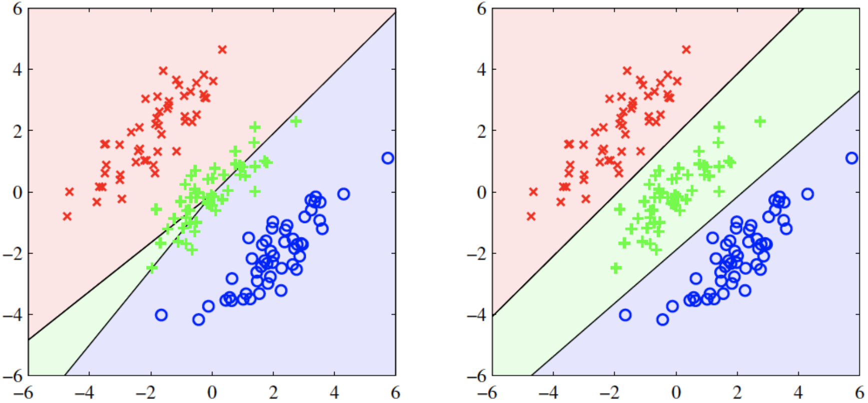
Linear Regression



linear regression decision boundary — the three fitted regression lines

For this particular 3-class problem, the decision boundaries produced by linear regression between 1 and 2 and between 2 and 3 are the same, so we would never predict class 2. This problem is called **masking**. Projecting onto the line joining the three class centroids shows why this happened.

Linear Regression



Left: linear regression; Right: logistic regression

Multinomial Logistic Regression

The multinomial logistic regression model assumes

$$\Pr(y = j|x) = \frac{\exp(x'\beta_j)}{\sum_{\ell=1}^{J} \exp(x'\beta_\ell)} \tag{14}$$

(14) $\Rightarrow$

$$\ln \frac{\Pr(y = j|x)}{\Pr(y = k|x)} = x'(\beta_j - \beta_k)$$

- The function $\sigma_j(z) \equiv \frac{\exp(z_j)}{\sum_{\ell=1}^{J} \exp(z_\ell)}$ [23] is called the **softmax function** – a generalization of the sigmoid.

[23] $z = (z_1, \ldots, z_J)$.

Multinomial Logistic Regression

Note that since $\sum_{j=1}^{J} \Pr(y = j|x) = 1$, we only need to estimate $\Pr(y = j|x)$ for $J - 1$ classes of y. Therefore, we can choose one class of y, say $y = 1$, to be the **reference level** and normalize β_1 to 0.

This implies

$$\Pr(y = 1|x) = \frac{1}{1 + \sum_{\ell=2}^{J} \exp(x'\beta_\ell)}$$
$$\Pr(y = j|x) = \frac{\exp(x'\beta_j)}{1 + \sum_{\ell=2}^{J} \exp(x'\beta_\ell)}, \qquad j = 2, \dots, J$$

, and

$$\ln \frac{\Pr(y = j|x)}{\Pr(y = 1|x)} = x'\beta_j$$

, i.e., $\exp(x'\beta_j)$ becomes the probability of $y = j$ *relative* to $y = 1$.

Mode of Transportation

Modes of transportation: $\{$bus, car, subway$\}$

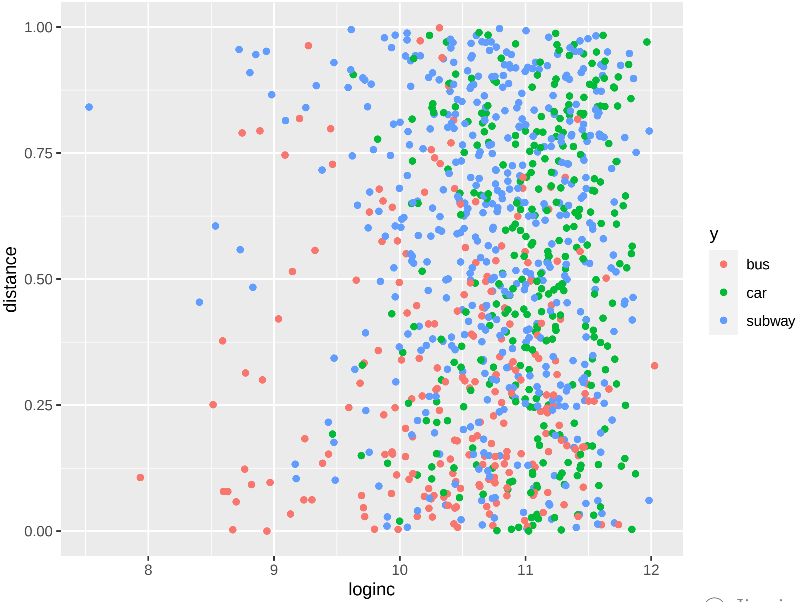
Individual variables: log (annual) income, distance to work (from 0 to 1)

```
transport <- read.csv("Transport.txt")
head(transport,3)

##   LogIncome DistanceToWork ModeOfTransportation
## 1 11.777090      0.6454524                  car
## 2 11.130492      0.5135208               subway
## 3  9.090856      0.8144265               subway

loginc <- transport$LogIncome
distance <- transport$DistanceToWork
y <- transport$ModeOfTransportation
```

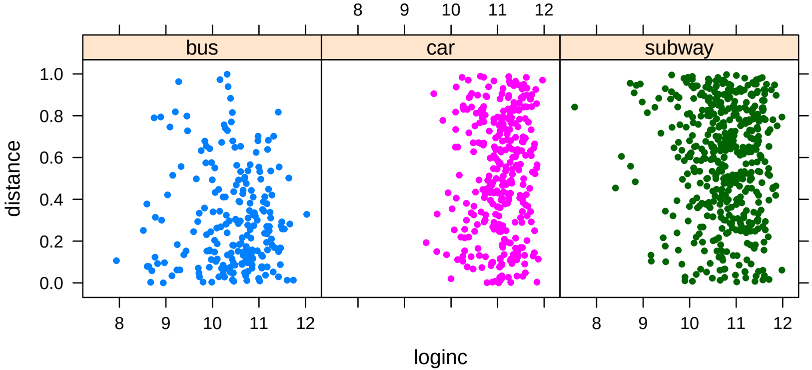
Mode of Transportation

```r
prop.table(table(y))

## y
##    bus    car  subway
##   0.22   0.31   0.47

income <- exp(loginc)
cbind(mean(income[y=="bus"]),mean(income[y=="car"]),
mean(income[y=="subway"]))

##       [,1]     [,2]     [,3]
## [1,] 42792 70006.83 56048.95

cbind(mean(distance[y=="bus"]),mean(distance[y=="car"]),
mean(distance[y=="subway"]))

##           [,1]      [,2]     [,3]
## [1,] 0.3032989 0.5149095 0.580446
```

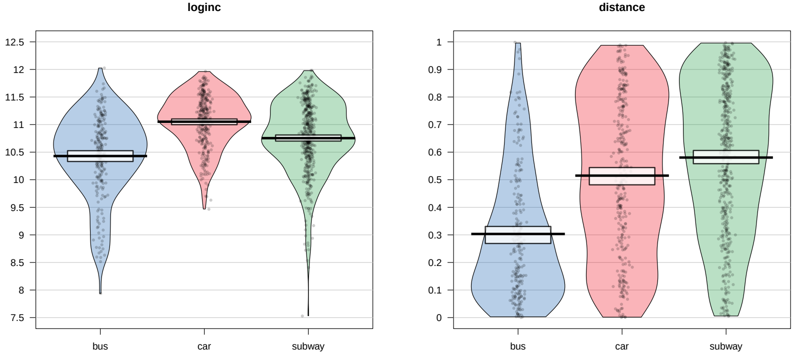
Mode of Transportation



Mode of Transportation



Mode of Transportation



Mode of Transportation

```r
require(nnet)
logitfit <- multinom(y ~ loginc + distance)
```

```r
require(AER)
coeftest(logitfit)

##
## z test of coefficients:
##
##                      Estimate Std. Error  z value  Pr(>|z|)
## car:(Intercept)    -18.60894    1.85544 -10.0294 < 2.2e-16 ***
## car:loginc           1.64705    0.16969   9.7061 < 2.2e-16 ***
## car:distance         2.93996    0.37602   7.8187 5.339e-15 ***
## subway:(Intercept)  -8.55927    1.45952  -5.8645 4.506e-09 ***
## subway:loginc        0.72359    0.13545   5.3421 9.189e-08 ***
## subway:distance      3.75524    0.35014  10.7248 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Mode of Transportation

Estimation results: (reference level: bus)

$$
\begin{aligned}
\log \frac{\widehat{p}\left(\text{car}|x\right)}{\widehat{p}\left(\text{bus}|x\right)} &= -18.61 + 1.65 \times \text{loginc} + 2.94 \times \text{distance} \qquad (15)\\
&= x'\widehat{\beta}_{\text{car}}\\
\log \frac{\widehat{p}\left(\text{subway}|x\right)}{\widehat{p}\left(\text{bus}|x\right)} &= -8.56 + 0.72 \times \text{loginc} + 3.76 \times \text{distance}\\
&= x'\widehat{\beta}_{\text{subway}}
\end{aligned}
$$

, where $x = [1, \text{loginc}, \text{distance}]'$, $\widehat{\beta}_{\text{car}} = [-18.61, 1.65, 2.94]'$, and $\widehat{\beta}_{\text{subway}} = [-8.56, 0.72, 3.76]'$.

Mode of Transportation

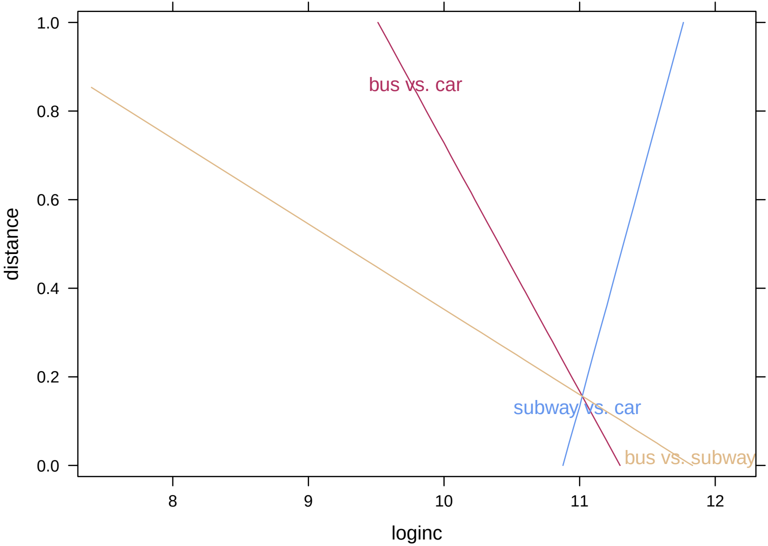
(15) $\Rightarrow$

$$\widehat{p}\left(\text{bus}|x\right) = \frac{1}{1+\exp\left(x'\widehat{\beta}_{\text{car}}\right)+\exp\left(x'\widehat{\beta}_{\text{subway}}\right)} \tag{16}$$
$$\widehat{p}\left(\text{car}|x\right) = \frac{\exp\left(x'\widehat{\beta}_{\text{car}}\right)}{1+\exp\left(x'\widehat{\beta}_{\text{car}}\right)+\exp\left(x'\widehat{\beta}_{\text{subway}}\right)}$$
$$\widehat{p}\left(\text{subway}|x\right) = \frac{\exp\left(x'\widehat{\beta}_{\text{subway}}\right)}{1+\exp\left(x'\widehat{\beta}_{\text{car}}\right)+\exp\left(x'\widehat{\beta}_{\text{subway}}\right)}$$

Mode of Transportation

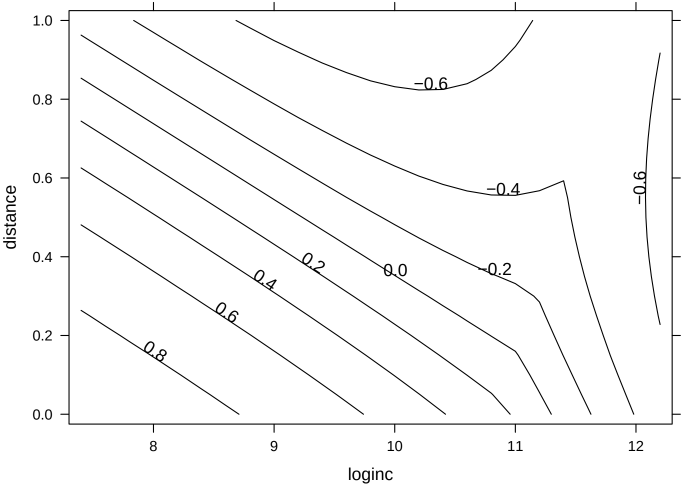
- Decision boundary between bus and car: $x'\widehat{\beta}_{\text{car}} = 0$
- Decision boundary between bus and subway: $x'\widehat{\beta}_{\text{subway}} = 0$
- Decision boundary between car and subway: $x'\left(\widehat{\beta}_{\text{subway}} - \widehat{\beta}_{\text{car}}\right) = 0$

Mode of Transportation



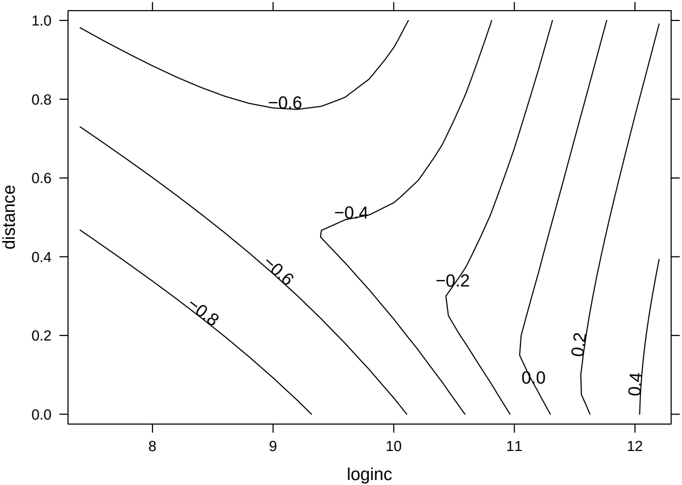
Mode of Transportation

Contour plot of $-\max\left(x'\widehat{\beta}_{\text{car}}, x'\widehat{\beta}_{\text{subway}}\right)$:



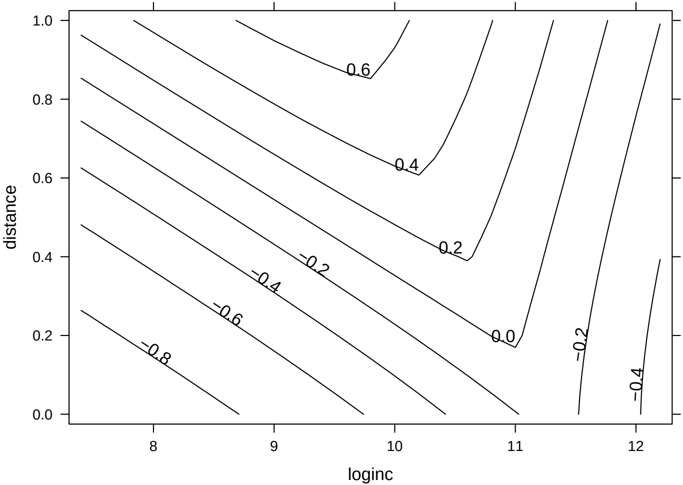
Mode of Transportation

Contour plot of $x'\widehat{\beta}_{\text{car}} - \max\left(0, x'\widehat{\beta}_{\text{subway}}\right)$:



Mode of Transportation

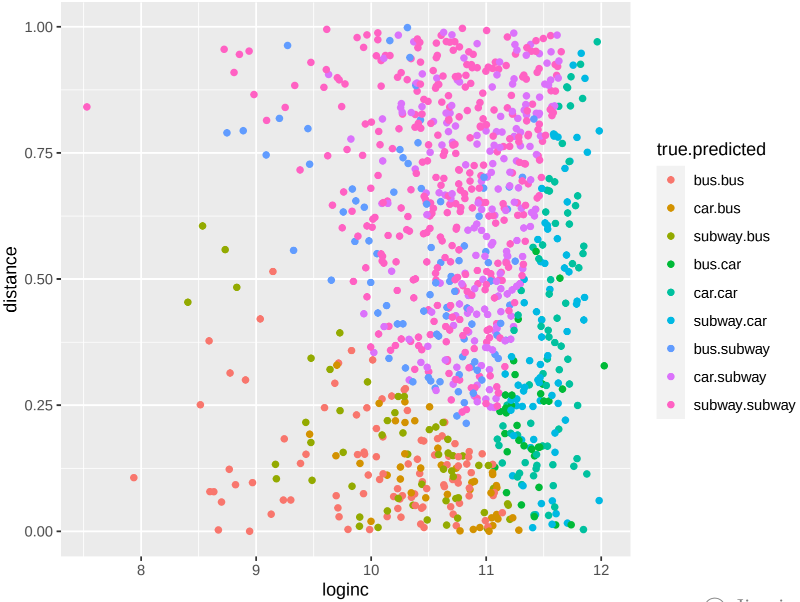
Contour plot of $x'\widehat{\beta}_{\text{subway}} - \max\left(0, x'\widehat{\beta}_{\text{car}}\right)$:



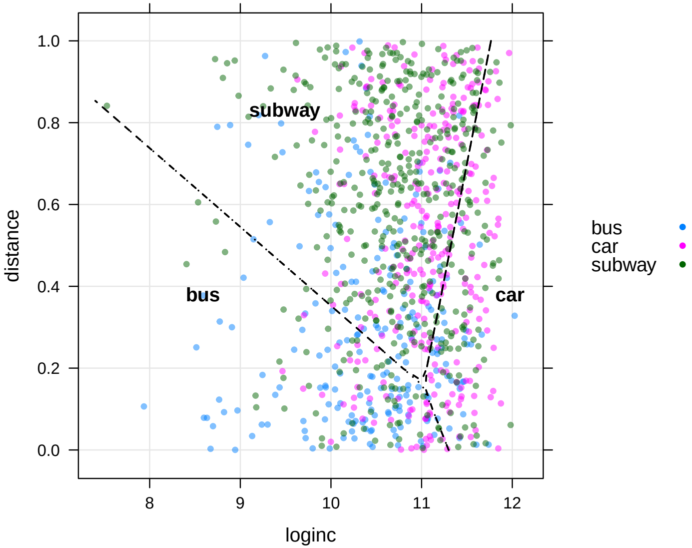
Mode of Transportation

```
logit.yhat <- predict(logitfit)
t <- table(logit.yhat,y,dnn=c("predicted","true"))
t

##            true
## predicted bus car subway
##    bus     101  41     55
##    car      33  78     72
##    subway   86 191    343

1 - sum(diag(t))/sum(t) # training error rate

## [1] 0.478
```

Mode of Transportation



Mode of Transportation



Mode of Transportation

Now suppose there is no subway, what will be the share of bus and car as mode of transportation among the commuters?

From (15), we know that:

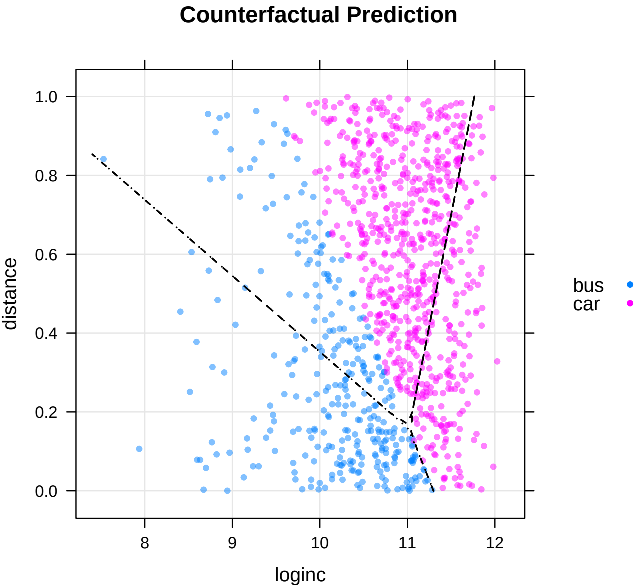
$$\log \frac{\widehat{p}\,(\text{car}|x)}{\widehat{p}\,(\text{bus}|x)} = -18.61 + 1.65 \times \text{loginc} + 2.94 \times \text{distance}$$

The decision boundary between bus and car does *not* change whether there is subway or not.

Mode of Transportation

```r
require(ramify)
logit.phat <- predict(logitfit,type="probs")
counterfactual.p <- logit.phat[,c(1,2)] # no subway
counterfactual.p <- counterfactual.p/rowSums(counterfactual.p)
counterfactual.y <- as.factor(argmax(counterfactual.p))
levels(counterfactual.y) <- c("bus","car")
table(counterfactual.y,logit.yhat)

##                 logit.yhat
## counterfactual.y bus car subway
##              bus 197   0    116
##              car   0 183    504
```

Mode of Transportation



Calculating Market Share

Assume the observed data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is a random sample drawn from the underlying population. Then the "market share" of alternative j – the share of individuals in the population that choose j – is

$$\begin{aligned} \Pr(y_i = j) &= \int \Pr(y_i = j | x_i) f(x_i) \, dx_i \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \Pr(y_i = j | x_i) \end{aligned}$$

, i.e., we can average individual conditional choice probabilities to get an estimate of the market share of each alternative in the population.

Mode of Transportation

```
# note: average choice probabilities estimated by logistic regression
# on the training data always match the observed shares of choices
# (if intercepts are included in the model)

marketShare.subway <- colMeans(logit.phat)
marketShare.subway

##       bus       car    subway
## 0.2199985 0.3100005 0.4700010

marketShare.nosubway <- colMeans(counterfactual.p)
marketShare.nosubway

##       bus       car
## 0.3822821 0.6177179
```

Mode of Transportation

predicted share	with subway	without subway
bus	22%	38%
car	31%	62%

Is this reasonable? Many people use subway not because of income or distance considerations, but because they cannot drive or they strongly prefer public transportation. For these people, if there is no subway, they would mostly switch to bus rather than car...

Independence of Irrelevant Alternatives (IIA)

For the multinomial logistic regression model,

$$\log \frac{\Pr(y = j|x)}{\Pr(y = k|x)} = x'(\beta_j - \beta_k)$$

for any two classes j and k.

The probability of $y = j$ relative to $y = k$ depends *only* on $x'\beta_j$ and $x'\beta_k$ – in particular, it is *not* affected by the existence and the properties of other classes.

This is called the **independence of irrelevant alternatives** (**IIA**) property.

Independence of Irrelevant Alternatives (IIA)

As an illustration of the IIA property (and why it can be undesirable in some cases), consider a more extreme example of the transportation problem:

Blue bus, Red bus

A route is currently served by a blue bus. People traveling along this route can either take the blue bus or drive themselves.

Suppose we observe each traveler's transportation choice, but do not observe any other characteristics. Our logistic regression model is then simply:

$$\log \frac{\Pr(\text{blue bus}|x)}{\Pr(\text{car}|x)} = \beta_0 \tag{17}$$

, where $x = 1$. If currently 40% of the travelers take the blue bus, while 60% drive, then $\widehat{\beta}_0 = \log\left(\frac{2}{3}\right)$.

Independence of Irrelevant Alternatives (IIA)

Blue bus, Red bus

Note that (17) predicts the relative share of blue bus riders to car drivers to be 2 : 3 regardless of what other transportation options are available.

What if the government now decides to introduce a red bus to this route, which is identical to the blue bus except the color of the paint?

Suppose people do not care about color, so that $\frac{\Pr(\text{red bus})}{\Pr(\text{blue bus})} = 1$, then the model would predict the rider shares to be
$\Pr(\text{blue bus}) : \Pr(\text{red bus}) : \Pr(\text{car}) = 2 : 2 : 3$
$\Rightarrow \Pr(\text{blue bus}) = \Pr(\text{red bus}) = 28.57\%, \Pr(\text{car}) = 42.86\%$.

This is clearly unreasonable, since we should expect
$\Pr(\text{blue bus}) = \Pr(\text{red bus}) = 20\%, \Pr(\text{car}) = 60\%$, i.e., the bus riders would be split between the blue bus and the red bus, while the car drivers continue to drive.

Independence of Irrelevant Alternatives (IIA)

The problem is due to *unobserved* variables. Suppose the true model is:

$$\Pr(y = j|x, z) = \frac{\exp(x'\beta_j + z'\gamma_j)}{\sum_\ell \exp(x'\beta_\ell + z'\gamma_\ell)}$$

, where z is unobserved[24]. Then

$$\Pr(y = j|x) = \int \frac{\exp(x'\beta_j + z\gamma_j)}{\sum_\ell \exp(x'\beta_\ell + z\gamma_\ell)} f(z)\, dz$$

In this case, $\log \frac{\Pr(y=j|x)}{\Pr(y=k|x)}$ is in general no longer a function of $x'\beta_j$ and $x'\beta_k$ only, hence the IIA no longer holds.

[24] e.g., preference for public transportation.

Multinomial Logistic Regression for Aggregate Outcomes

As in the binary case, multinomial logistic regression can be used for problems where the response variable is the sum of individual discrete outcomes.

The model is:

$$y_i \sim \text{Multinomial}\left(n_i, \pi_i\right) \tag{18}$$

, where $\pi_i = (\pi_{i1}, \ldots, \pi_{iJ})$, $\sum_{j=1}^{J} \pi_{ij} = 1$, and

$$\pi_{ij} = \frac{\exp\left(x_i' \beta_j\right)}{\sum_{\ell=1}^{J} \exp\left(x_i' \beta_\ell\right)}$$

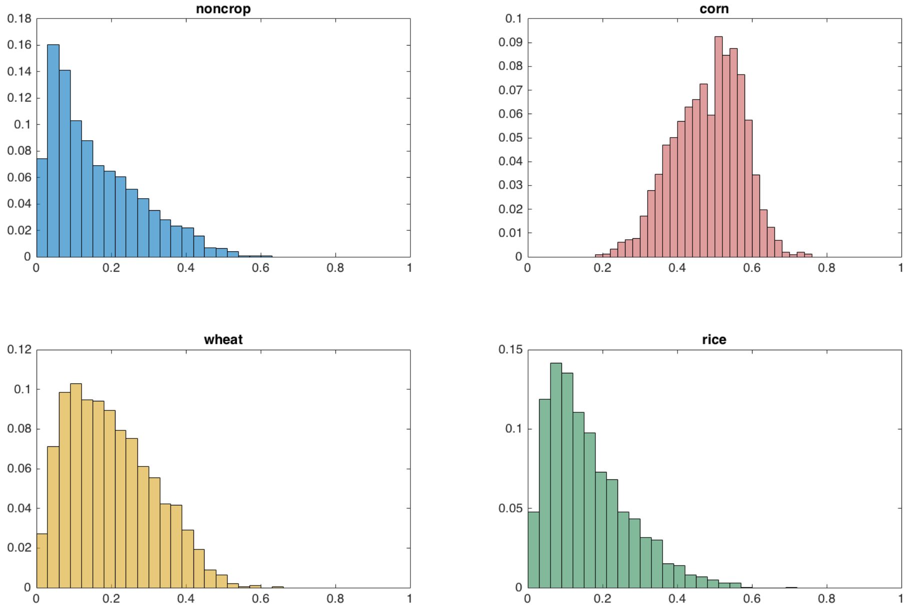
- When $n_i = 1$, (18) becomes the multinomial logistic model for multiclass classification.

Crop Choice

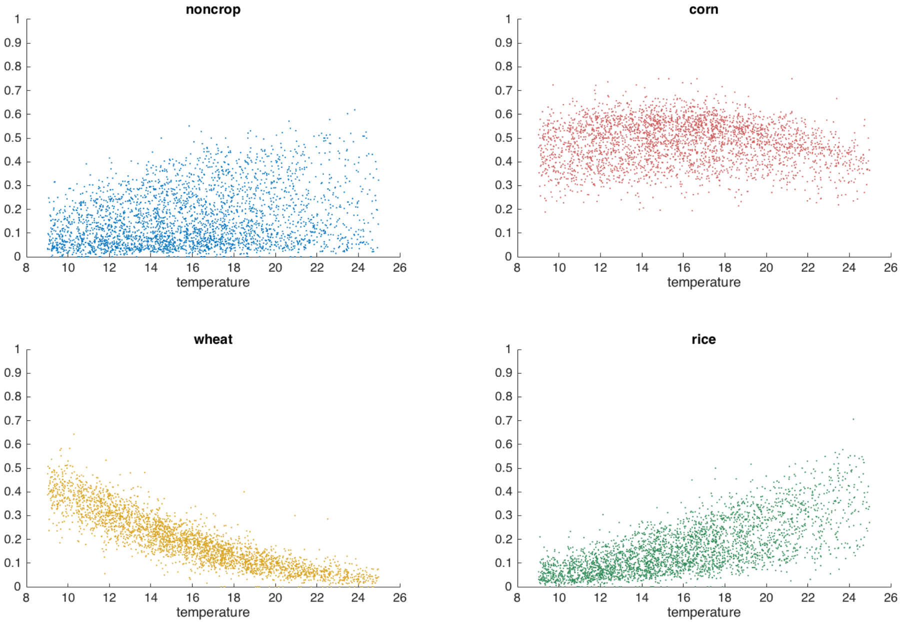
Crops: {corn, wheat, rice}

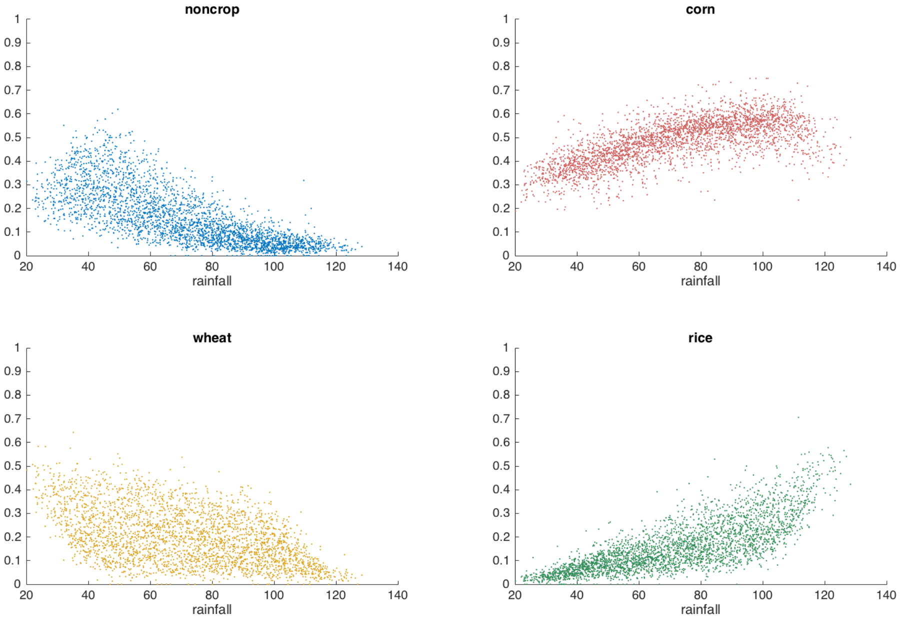
3144 counties, data on each county include number of agricultural land (fields) available, number of fields that are being cultivated for each crop, average temperature, and average monthly rainfall.

```
cropchoice <- read.csv("cropchoice.txt")
attach(cropchoice)
head(cropchoice,5)

##   temperature  rainfall fields noncrop corn wheat rice
## 1    13.18475  75.26666     63       8   31    17    7
## 2    12.35680 102.37572    165       7  100    30   28
## 3    17.57882 101.61363     38       1   26     3    8
## 4    20.86867  64.35788    152      45   78    12   17
## 5    13.88084 107.54101     88       4   54    15   15
```



Distribution of percentage cultivated


noncrop
corn
wheat
rice
temperature
temperature
temperature
temperature
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
8 10 12 14 16 18 20 22 24 26


noncrop
corn
wheat
rice
rainfall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120
140

Crop Choice

```
require(nnet)
crops <- cbind(noncrop,corn,wheat,rice)
logitfit <- multinom(crops ~ temperature + rainfall)
```

```
require(AER)
coeftest(logitfit)

##
## z test of coefficients:
##
##                      Estimate  Std. Error  z value  Pr(>|z|)
## corn:(Intercept)   0.63814409  0.02120175   30.099 < 2.2e-16 ***
## corn:temperature  -0.12877826  0.00128084 -100.542 < 2.2e-16 ***
## corn:rainfall      0.03864995  0.00022141  174.564 < 2.2e-16 ***
## wheat:(Intercept)  2.57310771  0.02427508  105.998 < 2.2e-16 ***
## wheat:temperature -0.25688133  0.00156614 -164.022 < 2.2e-16 ***
## wheat:rainfall     0.02567228  0.00025031  102.563 < 2.2e-16 ***
## rice:(Intercept)  -3.26197982  0.02843702 -114.709 < 2.2e-16 ***
## rice:temperature  -0.02241833  0.00155758  -14.393 < 2.2e-16 ***
## rice:rainfall      0.05132472  0.00026986  190.187 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Can we run linear regression instead?

Yes. Let y_{ij} be the number of fields used for crop j in county i, with $j = 1$ denoting no cultivated crops. Let n_i be the number of fields in county i. Let $p_{ij} = y_{ij}/\, n_i$ and $z_{ij} = \log p_{ij} - \log p_{i1}$. Then we can estimate the following $J-1$ linear regression equations:

$$z_i = x_i' \beta_j + e_j, \quad j = 2, \ldots, J \tag{19}$$

, where $x_i = [1, \text{temperature}_i, \text{rainfall}_i]$.

When n_i is large, (19) $\rightarrow$ the multinomial logistic model (18).

Crop Choice

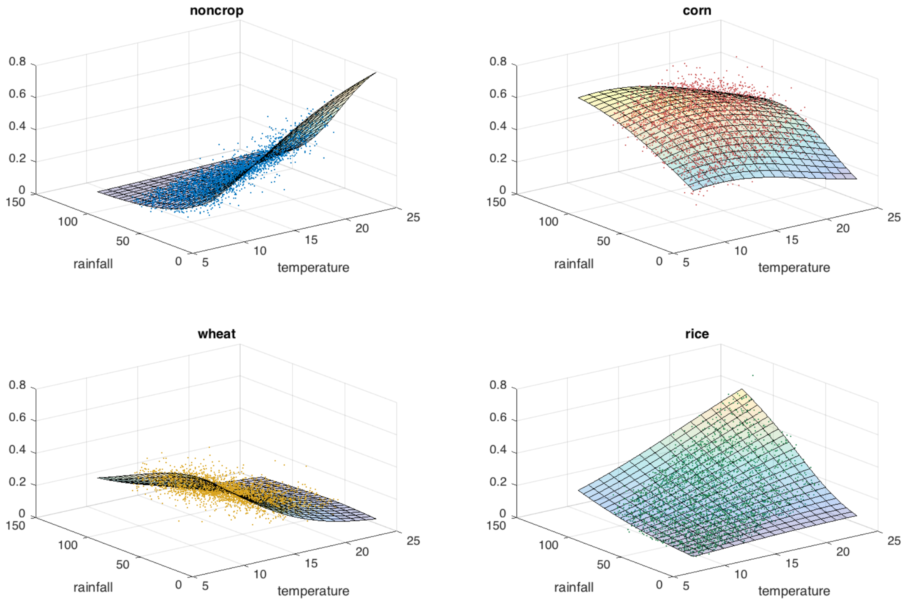
```
p <- crops/fields
eps <- 1e-4
p[p==0] <- p[p==0] + eps
z.corn <- log(p[,2]) - log(p[,1])
lsfit.corn <- lm(z.corn ~ temperature + rainfall)
coeftest(lsfit.corn)

##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  0.64378418  0.06253041  10.296 < 2.2e-16 ***
## temperature -0.14078836  0.00371590 -37.888 < 2.2e-16 ***
## rainfall     0.04268634  0.00059017  72.329 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Crop Choice

```
z.wheat <- log(p[,3]) - log(p[,1])
lsfit.wheat <- lm(z.wheat ~ temperature + rainfall)
coeftest(lsfit.wheat)

##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  2.85626917  0.07518212  37.991 < 2.2e-16 ***
## temperature -0.28943989  0.00446774 -64.784 < 2.2e-16 ***
## rainfall     0.02933530  0.00070958  41.342 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Crop Choice

```
z.rice <- log(p[,4]) - log(p[,1])
lsfit.rice <- lm(z.rice ~ temperature + rainfall)
coeftest(lsfit.rice)

##
## t test of coefficients:
##
##                 Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept) -3.68724544  0.07945515 -46.4066 < 2.2e-16 ***
## temperature -0.02622848  0.00472166  -5.5549 3.009e-08 ***
## rainfall     0.05834856  0.00074991  77.8074 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



Multinomial Logistic Fit

Discrete Choice Models

- In the econometrics literature, the response variables in classification problems are often individual choices.
  - Here "individuals" can refer to people, firms, governments – any unit of decision making.
- Discrete choice models are a class of econometric models of how individuals make choices.
  - These models can be considered structural models of decision making based on utility maximization.

The Random Utility Maximization (RUM) Framework

- Individual i faces a choice among J alternatives.
- The utility associated with alternative j is U_{ij}.
- The individual chooses the alternative that generates the highest utility, i.e., let $y_i \in \{1, \ldots, J\}$ denote the choice the individual makes, then

$$y_i = \underset{j \in \{1,\ldots,J\}}{\arg\max} \{U_{ij}\} \tag{20}$$

We do not observe U_{ij}. Instead, we observe (x_{ij}, y_i), where x_{ij} are characteristics associated with individual i and alternative j.

In general, x_{ij} may contain two types of variables: s_i and z_{ij}

- s_i : individual-specific variables (e.g., income)
- z_{ij} : alternative-specific variables (e.g., price)[25]

[25] If z_{ij} is the same for all i, then we can denote it by z_j.

The Random Utility Maximization (RUM) Framework

Since we observe x_{ij} but not U_{ij}, we can write:

$$U_{ij} = f_j\left(x_{ij}\right) + e_{ij} \tag{21}$$

, where e_{ij} captures unobserved factors[26] that influence U_{ij}[27].

Let $e_i = \left(e_{i1}, \ldots, e_{iJ}\right)$. We assume

$$e_i \sim^{i.i.d.} \mathcal{F}_e\left(.\right)$$

Different specifications of $f_j\left(x_{ij}\right)$ and $\mathcal{F}_e\left(.\right)$ lead to different discrete choice models.

[26] Unobserved to *us* not to the individual.

[27] One can think of $f_j\left(x_{ij}\right)$ as the systematic component of a decision maker's utility and e_{ij} as the idiosyncratic component.

The Random Utility Maximization (RUM) Framework

Let $x_i = \{x_{ij}\}_{j=1}^{J}$. (20) and (21) $\Rightarrow$

$$\begin{aligned}
\Pr\left(y_i = j \mid x_i\right) &= \Pr\left(U_{ij} > U_{i\ell} \;\; \forall \ell \neq j \mid x_i\right) \\
&= \Pr\left(f_j\left(x_i\right) + e_{ij} > f_\ell\left(x_i\right) + e_{i\ell} \;\; \forall \ell \neq j \mid x_i\right) \\
&= \int \mathcal{I}\left(e_{i\ell} - e_{ij} < f_j\left(x_i\right) - f_\ell\left(x_i\right) \;\; \forall \ell \neq j\right) d\mathcal{F}_e\left(e_i\right)
\end{aligned}$$

, i.e., once we place assumptions on $f_j\left(x_{ij}\right)$ and $\mathcal{F}_e\left(.\right)$, we can calculate $\Pr\left(y_i = j \mid x_i\right)$, which is called the **conditional choice probability** (**CCP**) in discrete choice models[28].

[28] The RUM framework assumes that the individual knows her U_{ij}, so that her decision is *deterministic*. However, since we do not observe U_{ij}, we can only calculate the probability of her choosing each alternative conditional on the variables we observe.

The Random Utility Maximization (RUM) Framework

Discrete choice models derived from the RUM framework has the following features[29]:

1. The absolute level of utility is irrelevant. Only differences in utility matter.
2. The overall scale of utility is irrelevant.

[29] Therefore, we will not be able to learn the level of utility associated with different alternatives, only the scaled differences among them.

Only Differences in Utility Matter

The absolute level of utility is irrelevant. If a constant is added to the utility of all alternatives, then the alternative with the highest utility does not change.

The following models are equivalent:

$$\begin{aligned} \text{Model 1: } & U_{ij} = f_j\left(x_{ij}\right) + e_{ij} \\ \text{Model 2: } & U_{ij} = \alpha + f_j\left(x_{ij}\right) + e_{ij} \end{aligned}$$

, where α is any constant.

Only Differences in Utility Matter

Example

Consider a binary choice problem: $y \in \{A, B\}$. The following models are equivalent:

- Model 1

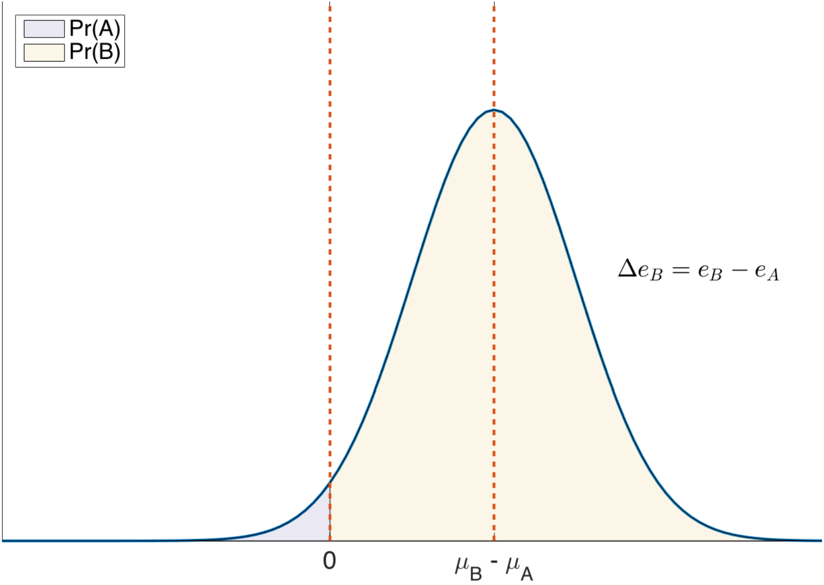
$$U_{iA} = \mu_A + e_{iA}, \;\; e_{iA} \sim \mathcal{N}\left(0, \sigma_A^2\right)$$
$$U_{iB} = \mu_B + e_{iB}, \;\; e_{iB} \sim \mathcal{N}\left(0, \sigma_B^2\right)$$

- Model 2

$$U_{iA} = 0$$
$$U_{iB} = \Delta\mu_B + \Delta e_{iB}, \;\; \Delta e_{iB} \sim \mathcal{N}\left(0, \sigma_A^2 + \sigma_B^2\right)$$

, where $\Delta\mu_B = \mu_B - \mu_A$ and $\Delta e_{iB} = e_{iB} - e_{iA}$.

Only Differences in Utility Matter



The Overall Scale of Utility is Irrelevant

The overall scale of utility is irrelevant. Multiplying the utility of all alternatives does not change individual choice: the alternative with the highest utility is the same irrespective of how utility is scaled.

The following models are equivalent:

$$\text{Model 1: } U_{ij} = f_j\left(x_{ij}\right) + e_{ij}$$
$$\text{Model 2: } U_{ij} = \lambda f_j\left(x_{ij}\right) + \lambda e_{ij}$$

, where λ is any positive constant.

The Overall Scale of Utility is Irrelevant

Example (cont.)

The following models are equivalent to Model 1 and Model 2:

- Model 3

$$U_{iA} = \widetilde{\mu}_A + \widetilde{e}_{iA}, \;\; \widetilde{e}_{iA} \sim \mathcal{N}\left(0, \frac{\sigma_A^2}{\sigma_A^2 + \sigma_B^2}\right)$$

$$U_{iB} = \widetilde{\mu}_B + \widetilde{e}_{iB}, \;\; \widetilde{e}_{iB} \sim \mathcal{N}\left(0, \frac{\sigma_A^2}{\sigma_A^2 + \sigma_B^2}\right)$$

, where $\widetilde{\mu}_j = \lambda \mu_j, \widetilde{e}_{ij} = \lambda e_{ij}$, and $\lambda = 1 \Big/ \sqrt{\sigma_A^2 + \sigma_B^2}$.

The Overall Scale of Utility is Irrelevant

Example (cont.)

- Model 4

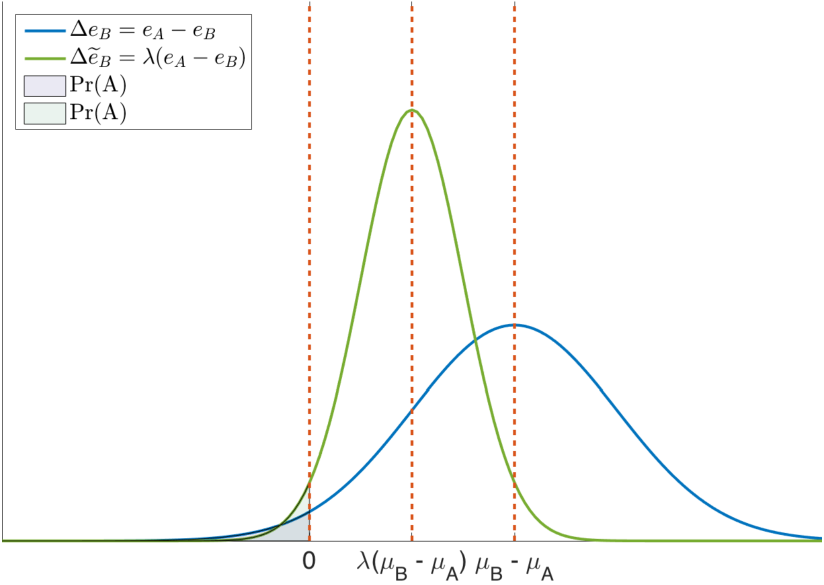
$$U_{iA} = 0$$
$$U_{iB} = \Delta\widetilde{\mu}_B + \Delta\widetilde{e}_{iB}, \;\; \Delta\widetilde{e}_{iB} \sim \mathcal{N}(0,1)$$

, where $\Delta\widetilde{\mu}_B = \widetilde{\mu}_B - \widetilde{\mu}_A$ and $\Delta\widetilde{e}_{iB} = \widetilde{e}_{iB} - \widetilde{e}_{iA}$.

Therefore, in Model 1, the parameters $\mu_A, \mu_B, \sigma_A, \sigma_B$ are not separately *identifiable*, because an infinite number of models (corresponding to different values of α and γ) are *consistent* with the same choice behavior.

To estimate the model, we need to *normalize* the level and scale of utility. What we can estimate as a result is $\Delta\widetilde{\mu}_B = \lambda(\mu_B - \mu_A)$ – the *scaled difference* between μ_A and μ_B.

The Overall Scale of Utility is Irrelevant



Probit

For $j = 1, \ldots, J$,

$$U_{ij} = x_{ij}'\beta_j + e_{ij}$$

, and

$$e_i = \begin{bmatrix} e_{i1} \\ \vdots \\ e_{iJ} \end{bmatrix} \sim \mathcal{N}(0, \Sigma)$$

Probit

For binary discrete choice problems, let $y \in \{A, B\}$. We have:

$$U_{iA} = x'_{iA}\beta_A + e_{iA} \tag{22}$$
$$U_{iB} = x'_{iB}\beta_B + e_{iB}$$

, and

$$e_i = \begin{bmatrix} e_{iA} \\ e_{iB} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_A^2 & \sigma_{AB} \\ . & \sigma_B^2 \end{bmatrix}\right) \tag{23}$$

Probit

Note that (23) $\Rightarrow$

$$e_{iA} - e_{iB} \sim \mathcal{N}\left(0, \sigma_A^2 + \sigma_B^2 - 2\sigma_{AB}\right)$$

Normalizing (22) $\Rightarrow$

$$\begin{aligned} U_{iA} &= 0 \\ U_{iB} &= x_{iB}'\widetilde{\beta}_B - x_{iA}'\widetilde{\beta}_A + \Delta\widetilde{e}_{iB} \end{aligned}$$

, where, let $\lambda = 1 \Big/ \sqrt{\sigma_A^2 + \sigma_B^2 - 2\sigma_{AB}}$, then $\widetilde{\beta}_A = \lambda\beta_A, \widetilde{\beta}_B = \lambda\beta_B$, and $\Delta\widetilde{e}_{iB} = \lambda\left(e_{iB} - e_{iA}\right) \sim \mathcal{N}(0,1)$.

Probit

[30]

[30] Here we have motivated probit using the RUM framework. However, probit can be introduced as a purely statistical classification model just like the logistic model. For binary classification with only individual-specific variables, the probit model is

$$\Pr(y = 1|x) = \Phi\left(x'\beta\right)$$

, where Φ is the CDF of $\mathcal{N}(0, 1)$.

Probit

Example 1

$$U_{iA} = \alpha_A + z'_A \delta_A + e_{iA}$$
$$U_{iB} = \alpha_B + z'_B \delta_B + e_{iB}$$

Here $z'_j \delta_j$ and α_j are both constants and hence cannot be separately identified.

- As long as there is an intercept term, alternative-specific variables z_{ij} *must* vary with i in order to be identified.

Probit

Example 2

$$
\begin{aligned}
U_{iA} &= \alpha_A + s_i'\gamma + e_{iA} \\
U_{iB} &= \alpha_B + s_i'\gamma + e_{iB}
\end{aligned} \tag{24}
$$

(24) $\Rightarrow$

$$U_{iB} - U_{iA} = (\alpha_B - \alpha_A) + (e_{iB} - e_{iA})$$

Since only difference in utility matters, γ cannot be identified.

- The coefficients of individual-specific variables must be alternative-specific in order to be identified.

Probit

Example 3

$$U_{iA} = \alpha_A + s_i' \gamma_A + e_{iA} \tag{25}$$
$$U_{iB} = \alpha_B + s_i' \gamma_B + e_{iB}$$

(25) $\Rightarrow$

$$U_{iB} - U_{iA} = (\alpha_B - \alpha_A) + s_i' (\gamma_B - \gamma_A) + (e_{iB} - e_{iA})$$

- α_A and α_B cannot be separately identified.
- γ_A and γ_B cannot be separately identified.

Probit

Example 3

Normalization of the model:

1. normalize level

$$U_{iA} = 0$$
$$U_{iB} = \Delta\alpha_B + s_i'\Delta\gamma_B + \Delta e_{iB}$$

, where $\Delta\alpha_B = \alpha_B - \alpha_A$, $\Delta\gamma_B = \gamma_B - \gamma_A$, and $\Delta e_{iB} = e_{iB} - e_{iA}$.

2. normalize scale

$$U_{iA} = 0$$
$$U_{iB} = \Delta\tilde{\alpha}_B + s_i'\Delta\tilde{\gamma}_B + \Delta\tilde{e}_{iB}$$

, where we divide $\Delta\alpha_B, \Delta\lambda_B$, and Δe_{iB} by $\sqrt{\sigma_A^2 + \sigma_B^2 - 2\sigma_{AB}}$.

Probit

Example 4

$$U_{iA} = \alpha_A + s_i'\gamma_A + z_{iA}'\delta + e_{iA} \tag{26}$$
$$U_{iB} = \alpha_B + s_i'\gamma_B + z_{iB}'\delta + e_{iB}$$

$$U_{iA} = \alpha_A + s_i'\gamma_A + z_{iA}'\delta_A + e_{iA} \tag{27}$$
$$U_{iB} = \alpha_B + s_i'\gamma_B + z_{iB}'\delta_B + e_{iB}$$

Here we can specify either $z_{ij}'\delta$ or $z_{ij}'\delta_j$.

- Alternative-specific variables can have either **alternative-specific coefficients** or **generic coefficients** that do not change with alternatives.

Probit

Example 4

Normalizing (26) $\Rightarrow$[a]

$$U_{iA} = 0$$
$$U_{iB} = \Delta\widetilde{\alpha}_B + s_i'\Delta\widetilde{\gamma}_B + (z_{iB} - z_{iA})'\,\widetilde{\delta} + \Delta\widetilde{e}_{iB}$$

Normalizing (27) $\Rightarrow$

$$U_{iA} = 0$$
$$U_{iB} = \Delta\widetilde{\alpha}_B + s_i'\Delta\widetilde{\gamma}_B + \left(z_{iB}'\widetilde{\delta}_B - z_{iA}'\widetilde{\delta}_A\right) + \Delta\widetilde{e}_{iB}$$

[a] For both, $\Delta\widetilde{\alpha}_B, \Delta\widetilde{\gamma}_B, \Delta\widetilde{e}_{iB}$ are defined as before.
$\widetilde{\delta} = \lambda\delta, \widetilde{\delta}_A = \lambda\delta_A, \widetilde{\delta}_B = \lambda\delta_B$, and $\lambda = 1 \Big/ \sqrt{\sigma_A^2 + \sigma_B^2 - 2\sigma_{AB}}$.

Probit

Simulation 1:

$$U_{iA} = 5 - 10s_i + e_{iA} \tag{28}$$
$$U_{iB} = -5 + 10s_i + e_{iB}$$
$$e_i = \begin{bmatrix} e_{iA} \\ e_{iB} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \right)$$

Normalizing (28) $\Rightarrow$

$$\begin{aligned} U_{iA} &= 0 \\ U_{iB} &= -\frac{12}{\sqrt{5}} + \frac{20}{\sqrt{5}} s_i + \epsilon_{iB} \\ &= -5.37 + 8.94 s_i + \epsilon_{iB} \end{aligned}$$

, where $\epsilon_{iB} = (e_{iB} - e_{iA})/\sqrt{5} \sim \mathcal{N}(0, 1)$.

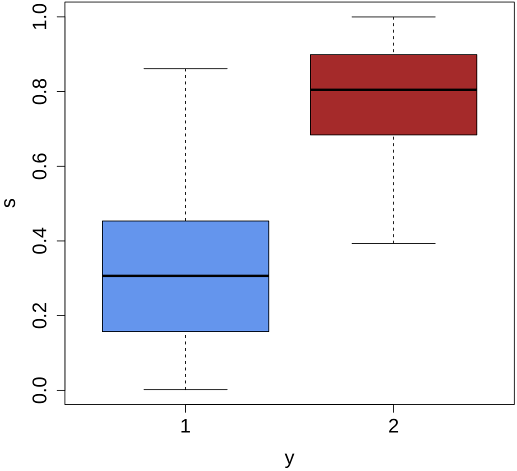
Probit

```
require(ramify)
n <- 1e3
s <- runif(n)
e1 <- rnorm(n,mean=1,sd=1)
e2 <- rnorm(n,mean=-1,sd=2)
u1 <- 5 - 10*s + e1
u2 <- -5 + 10*s + e2
U <- cbind(u1,u2)
y <- as.factor(argmax(U))
mydata <- data.frame(s,y)
```

Probit

```
head(mydata,5)

##           s y
## 1 0.1680415 1
## 2 0.8075164 2
## 3 0.3849424 1
## 4 0.3277343 1
## 5 0.6021007 2

prop.table(table(y))

## y
##     1     2
## 0.586 0.414
```

Probit



Probit

```r
require(AER)
probitfit <- glm(y ~ s, family=binomial(link="probit"))
coeftest(probitfit)

##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -5.40721    0.36303 -14.895 < 2.2e-16 ***
## s            9.07978    0.59449  15.273 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Probit

Simulation 2:

$$
\begin{aligned}
U_{iA} &= 5 - 10s_i + e_{iA} \\
U_{iB} &= -5 + 10s_i + e_{iB} \\
e_i = \begin{bmatrix} e_{iA} \\ e_{iB} \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix} \right)
\end{aligned} \tag{29}
$$

, where we let $\rho\left(e_{iA}, e_{iB}\right) = 0.5$, so that $\sigma_{AB} = \rho\sigma_A\sigma_B = 1$.

Probit

Normalizing (29) $\Rightarrow$

$$\begin{aligned} U_{iA} &= 0 \\ U_{iB} &= -\frac{12}{\sqrt{3}} + \frac{20}{\sqrt{3}} s_i + \epsilon_{iB} \\ &= -6.93 + 11.55 s_i + \epsilon_{iB} \end{aligned}$$

, where $\epsilon_{iB} = (e_{iB} - e_{iA})/\sqrt{3} \sim \mathcal{N}(0, 1)$.

Probit

```
n <- 1e3
s <- runif(n)

# generating e
require(MASS)
mu <- c(1,-1) # mean
sig <- c(1,2) # s.t.d. of each dimension
rho <- .5 # correlation
Sigma <- matrix(c(sig[1]^2,rho*sig[1]*sig[2], # covariance matrix
                  rho*sig[1]*sig[2],sig[2]^2),2,2)
e <- mvrnorm(n,mu,Sigma)

# generating y
e1 <- e[,1]
e2 <- e[,2]
u1 <- 5 - 10*s + e1
u2 <- -5 + 10*s + e2
y <- as.factor(argmax(cbind(u1,u2)))
```

Probit

```r
head(e,4)

##              [,1]       [,2]
## [1,] -0.5750613 -5.1608065
## [2,]  0.1128529 -3.4697423
## [3,]  1.9516721 -1.1075891
## [4,]  0.6012319 -0.4711042

colMeans(e)

## [1]  0.9808862 -1.0736455

var(e)

##             [,1]      [,2]
## [1,] 1.0208563 0.9980833
## [2,] 0.9980833 3.7899603
```

Probit

```
probitfit <- glm(y ~ s, family=binomial(link="probit"))
coeftest(probitfit)

##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -7.22787    0.56184 -12.865 < 2.2e-16 ***
## s           11.96904    0.91906  13.023 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Probit

Simulation 3:

$$U_{iA} = 5 - 10s_i - 0.1z_{iA} + e_{iA} \tag{30}$$
$$U_{iB} = -5 + 10s_i - 0.1z_{iB} + e_{iB}$$
$$e_i = \begin{bmatrix} e_{iA} \\ e_{iB} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \right)$$

Normalizing (30) $\Rightarrow$

$$\begin{aligned} U_{iA} &= 0 \\ U_{iB} &= -\frac{12}{\sqrt{5}} + \frac{20}{\sqrt{5}} s_i - \frac{0.1}{\sqrt{5}} (z_{iB} - z_{iA}) + \epsilon_{iB} \\ &= -5.37 + 8.94 s_i - 0.045 (z_{iB} - z_{iA}) + \epsilon_{iB} \end{aligned}$$

, where $\epsilon_{iB} = (e_{iB} - e_{iA}) / \sqrt{5} \sim \mathcal{N}(0, 1)$.

Probit

```
n <- 1e3
s <- runif(n)
z1 <- 100*runif(n)
z2 <- 50*runif(n)
e1 <- rnorm(n,mean=1,sd=1)
e2 <- rnorm(n,mean=-1,sd=2)
u1 <- 5 - 10*s -0.1*z1 + e1
u2 <- -5 + 10*s -0.1*z2 + e2
y <- as.factor(argmax(cbind(u1,u2)))
mydata <- data.frame(s,z1,z2,y)
```

Probit

```
probitfit <- glm(y ~ s + z1 + z2, family=binomial(link="probit"))
coeftest(probitfit)

##
## z test of coefficients:
##
##              Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -5.571587   0.431299 -12.9182 < 2.2e-16 ***
## s            9.401062   0.621498  15.1265 < 2.2e-16 ***
## z1           0.044736   0.003975  11.2544 < 2.2e-16 ***
## z2          -0.046307   0.005858  -7.9049 2.682e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Probit

Can we estimate the model with a *generic* coefficient for z_{ij} that does not change with j? Yes!

```
dz <- z2 - z1
probitfit <- glm(y ~ s + dz, family=binomial(link="probit"))
coeftest(probitfit)

##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -5.623073   0.388506 -14.474 < 2.2e-16 ***
## s            9.407041   0.621340  15.140 < 2.2e-16 ***
## dz          -0.045071   0.003779 -11.927 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now consider $J = 3$.

$$U_{ij} = x_{ij}'\beta_j + e_{ij}$$

, and

$$e_i = \begin{bmatrix} e_{i1} \\ e_{i2} \\ e_{i3} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ . & \sigma_2^2 & \sigma_{23} \\ . & . & \sigma_3^2 \end{bmatrix}\right)$$

Multinomial Probit

Normalizing level $\Rightarrow$

$$
\begin{aligned}
U_{i1} &= 0 \\
U_{i2} &= \left(x'_{i2}\beta_2 - x'_{i1}\beta_1\right) + \Delta e_{i2} \\
U_{i3} &= \left(x'_{i3}\beta_3 - x'_{i1}\beta_1\right) + \Delta e_{i3}
\end{aligned}
$$

, where $\Delta e_{ij} = e_{ij} - e_{i1}$, and[31]

$$
\begin{bmatrix} \Delta e_{i2} \\ \Delta e_{i3} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_1^2 + \sigma_2^2 - 2\sigma_{12} & \sigma_1^2 + \sigma_{23} - \sigma_{12} - \sigma_{13} \\ . & \sigma_1^2 + \sigma_3^2 - 2\sigma_{13} \end{bmatrix}\right)
$$

---

[31]

$$
\begin{aligned}
Cov\left(\Delta e_{i2}, \Delta e_{i3}\right) &= Cov\left(e_{i2} - e_{i1}, e_{i3} - e_{i1}\right) \\
&= \sigma_{23} - \sigma_{21} - \sigma_{13} + \sigma_1^2
\end{aligned}
$$

Multinomial Probit

Normalizing scale $\Rightarrow$

$$\begin{aligned} U_{i1} &= 0 \\ U_{i2} &= \left(x_{i2}'\widetilde{\beta}_2 - x_{i1}'\widetilde{\beta}_1\right) + \Delta\widetilde{e}_{i2} \\ U_{i3} &= \left(x_{i3}'\widetilde{\beta}_3 - x_{i1}'\widetilde{\beta}_1\right) + \Delta\widetilde{e}_{i3} \end{aligned}$$

, where $\widetilde{\beta}_j = \lambda\beta_j$, $\Delta\widetilde{e}_{ij} = \lambda\Delta e_{ij}$, $\lambda = 1 \Big/ \sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_{12}}$, and

$$\begin{bmatrix} \Delta\widetilde{e}_{i2} \\ \Delta\widetilde{e}_{i3} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & \frac{\sigma_1^2+\sigma_{23}-\sigma_{12}-\sigma_{13}}{\sigma_1^2+\sigma_2^2-2\sigma_{12}} \\ . & \frac{\sigma_1^2+\sigma_3^2-2\sigma_{13}}{\sigma_1^2+\sigma_2^2-2\sigma_{12}} \end{bmatrix}\right)$$

Multinomial Probit

Thus, before normalization, the covariance matrix of the error term has 6 parameters:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ . & \sigma_2^2 & \sigma_{23} \\ . & . & \sigma_3^2 \end{bmatrix}$$

After normalization,

$$\widetilde{\Sigma} = \begin{bmatrix} 1 & \omega_{12} \\ . & \omega_{22} \end{bmatrix}$$

, where $\omega_{12} = \frac{\sigma_1^2+\sigma_{23}-\sigma_{12}-\sigma_{13}}{\sigma_1^2+\sigma_2^2-2\sigma_{12}}, \omega_{22} = \frac{\sigma_1^2+\sigma_3^2-2\sigma_{13}}{\sigma_1^2+\sigma_2^2-2\sigma_{12}}$.

The number of covariance parameters to estimate decreases from 6 to 2 after normalization.

Multinomial Probit

In general, a model with J alternatives has *at most* $\frac{1}{2}J(J-1)-1$ covariance parameters after normalization.

Ketchup

Brands: {Heinz, Hunt's, Del Monte, Store Brand}

Variables: the price of each brand, the income of the buyer (in $1000), the brand purchased

```
ketchup <- read.csv("Ketchup.csv")
head(ketchup,3)

##   choice price.heinz price.hunts price.delmonte price.stb   income
## 1    stb        1.46        1.43           1.45      0.99 44.49198
## 2  heinz        0.99        1.39           1.49      0.89 59.26444
## 3    stb        1.19        1.29           1.46      0.95 31.75753

prop.table(table(ketchup$choice))

##
## delmonte    heinz    hunts      stb
##  0.05375  0.51125  0.21375  0.22125
```

Ketchup

Model 1:

$$
\begin{aligned}
U_{ij} &= \alpha_j + \delta \text{price}_{ij} + \gamma_j \text{income}_i + e_{ij} \qquad (31) \\
e_i &\sim \mathcal{N}(0, \Sigma)
\end{aligned}
$$

Ketchup

```
require(mlogit)
ketchup.long <- mlogit.data(ketchup, shape="wide",
                            varying=2:5, choice="choice")
head(ketchup.long,12)

## ~~~~~~~
##  first 12 observations out of 3200
## ~~~~~~~
##    choice   income      alt price chid    idx
## 1   FALSE 44.49198 delmonte  1.45    1 1:onte
## 2   FALSE 44.49198    heinz  1.46    1 1:einz
## 3   FALSE 44.49198    hunts  1.43    1 1:unts
## 4    TRUE 44.49198      stb  0.99    1  1:stb
## 5   FALSE 59.26444 delmonte  1.49    2 2:onte
## 6    TRUE 59.26444    heinz  0.99    2 2:einz
## 7   FALSE 59.26444    hunts  1.39    2 2:unts
## 8   FALSE 59.26444      stb  0.89    2  2:stb
## 9   FALSE 31.75753 delmonte  1.46    3 3:onte
## 10  FALSE 31.75753    heinz  1.19    3 3:einz
## 11  FALSE 31.75753    hunts  1.29    3 3:unts
## 12   TRUE 31.75753      stb  0.95    3  3:stb
```

Ketchup

```
# mlogit(y ~ z|s|w,...)
# - z: alternative-specific vars with generic coeffs
# - s: individual-specific vars
# - w: alternative-specific vars with alternative-specific coeffs
probitfit1 <- mlogit(choice ~ price|income, ketchup.long,
                     reflevel="stb", probit=TRUE)
```

```
require(AER)
coeftest(probitfit1)[1:7,]

##                        Estimate Std. Error    t value     Pr(>|t|)
## delmonte:(intercept) -1.13931111 1.16876911 -0.9747957 3.299608e-01
## heinz:(intercept)    -7.05610040 1.81280583 -3.8923641 1.076714e-04
## hunts:(intercept)    -4.32246680 1.33056061 -3.2486057 1.208861e-03
## price                -3.07882503 0.61797639 -4.9821078 7.733865e-07
## delmonte:income       0.03465121 0.02801584  1.2368435 2.165137e-01
## heinz:income          0.18002372 0.04398663  4.0926917 4.703326e-05
## hunts:income          0.11979359 0.03371002  3.5536490 4.025408e-04
```

Ketchup

```
coeftest(probitfit1)[8:12,]

##                  Estimate Std. Error    t value     Pr(>|t|)
## delmonte.heinz -0.1258684  0.4446334 -0.2830836 0.7771870996
## delmonte.hunts -0.7047540  0.4977681 -1.4158280 0.1572209988
## heinz.heinz     1.2623283  0.3342783  3.7762796 0.0001711608
## heinz.hunts     0.6634783  0.3711713  1.7875259 0.0742367344
## hunts.hunts     0.9704545  0.3640111  2.6660021 0.0078331911
```

So the estimated covariance matrix is ...

```
probitfit1$omega$stb # covariance matrix using "stb" as reference

##            delmonte      heinz      hunts
## delmonte  1.0000000 -0.1258684 -0.7047540
## heinz    -0.1258684  1.6093156  0.9262337
## hunts    -0.7047540  0.9262337  1.8786636
```

Ketchup

$(\widehat{U}_{i,\text{stb}} = 0)$

$$
\begin{aligned}
\widehat{U}_{i,\text{delmonte}} &= -1.14 - 3.08 \times \text{price}_{i,\text{delmonte}} + 0.035 \times \text{income}_i + \epsilon_{i,\text{delmonte}} \\
\widehat{U}_{i,\text{heinz}} &= -7.06 - 3.08 \times \text{price}_{i,\text{heinz}} + 0.18 \times \text{income}_i + \epsilon_{i,\text{heinz}} \\
\widehat{U}_{i,\text{hunts}} &= -4.32 - 3.08 \times \text{price}_{i,\text{hunts}} + 0.12 \times \text{income}_i + \epsilon_{i,\text{hunts}}
\end{aligned}
$$

, where

$$
\begin{bmatrix} \epsilon_{i,\text{delmonte}} \\ \epsilon_{i,\text{heinz}} \\ \epsilon_{i,\text{hunts}} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & -0.13 & -0.70 \\ . & 1.61 & 0.93 \\ . & . & 1.88 \end{bmatrix}\right)
$$

Ketchup

Model 2:

$$U_{ij} = \alpha_j + \delta_j \text{price}_{ij} + \gamma_j \text{income}_i + e_{ij} \tag{32}$$
$$e_i \sim \mathcal{N}(0, \Sigma)$$

Ketchup

```r
probitfit2 <- mlogit(choice ~ 0|income|price, ketchup.long,
                     reflevel="stb", probit=TRUE)
```

```r
coeftest(probitfit2)[1:10,]
```

```
##                          Estimate Std. Error   t value     Pr(>|t|)
## delmonte:(intercept) -2.93786780 2.48851715 -1.180570 2.381313e-01
## heinz:(intercept)    -9.79073108 4.40531214 -2.222483 2.653490e-02
## hunts:(intercept)    -5.50096028 2.64999192 -2.075840 3.823372e-02
## delmonte:income       0.04822031 0.03350156  1.439345 1.504514e-01
## heinz:income          0.25532406 0.13070045  1.953506 5.111457e-02
## hunts:income          0.16927598 0.08526977  1.985182 4.747189e-02
## stb:price            -4.10482188 1.79833571 -2.282567 2.272253e-02
## delmonte:price       -2.85282115 0.64411156 -4.429079 1.080621e-05
## heinz:price          -4.37328318 2.49407340 -1.753470 7.991161e-02
## hunts:price          -4.71107228 2.57769096 -1.827633 6.798415e-02
```

Ketchup

```
coeftest(probitfit2)[11:15,]

##                  Estimate Std. Error    t value   Pr(>|t|)
## delmonte.heinz -0.1424442  0.6506784 -0.2189165 0.82677203
## delmonte.hunts -1.0566066  0.8770324 -1.2047520 0.22866213
## heinz.heinz     1.7969567  0.9806295  1.8324522 0.06726274
## heinz.hunts     0.9872264  0.7128141  1.3849704 0.16645499
## hunts.hunts     1.4535726  0.9021936  1.6111537 0.10754821

probitfit2$omega$stb

##            delmonte      heinz     hunts
## delmonte  1.0000000 -0.1424442 -1.056607
## heinz    -0.1424442  3.2493438  1.924511
## hunts    -1.0566066  1.9245106  4.203907
```

Ketchup

$$\begin{aligned}
\widehat{U}_{i,\text{stb}} &= -4.1 \times \text{price}_{i,\text{stb}} \\
\widehat{U}_{i,\text{delmonte}} &= -2.94 - 2.85 \times \text{price}_{i,\text{delmonte}} + 0.048 \times \text{income}_i + \epsilon_{i,\text{delmonte}} \\
\widehat{U}_{i,\text{heinz}} &= -9.79 - 4.37 \times \text{price}_{i,\text{heinz}} + 0.255 \times \text{income}_i + \epsilon_{i,\text{heinz}} \\
\widehat{U}_{i,\text{hunts}} &= -5.50 - 4.71 \times \text{price}_{i,\text{hunts}} + 0.169 \times \text{income}_i + \epsilon_{i,\text{hunts}}
\end{aligned}$$

, where

$$\begin{bmatrix} \epsilon_{i,\text{delmonte}} \\ \epsilon_{i,\text{heinz}} \\ \epsilon_{i,\text{hunts}} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & -0.14 & -1.06 \\ . & 3.25 & 1.92 \\ . & . & 4.20 \end{bmatrix}\right)$$

Logistic Regression as RUM

Now let's assume the following model:

$$U_{ij} = x'_{ij}\beta_j + e_{ij} \tag{33}$$

, and

$$e_{ij} \sim^{i.i.d.} \text{Gumbel}\left(0, \sigma\right)$$

Extreme Value Distribution

The **Gumbel distribution**, also called the **Type I extreme value distribution**, has the following CDF:

$$\mathcal{F}(e; \mu, \sigma) = \exp\left\{-\exp\left(-\frac{e-\mu}{\sigma}\right)\right\}$$

- μ is the *location* parameter.
- σ is the *scale* parameter

For $e \sim \text{Gumbel}(\mu, \sigma)$,

$$\mathbb{E}(e) = \mu + \sigma\gamma_e$$
$$\mathbb{V}(e) = \frac{\pi^2}{6}\sigma^2$$

, where $\gamma_e \approx 0.577$ is the Euler constant.

Extreme Value Distribution

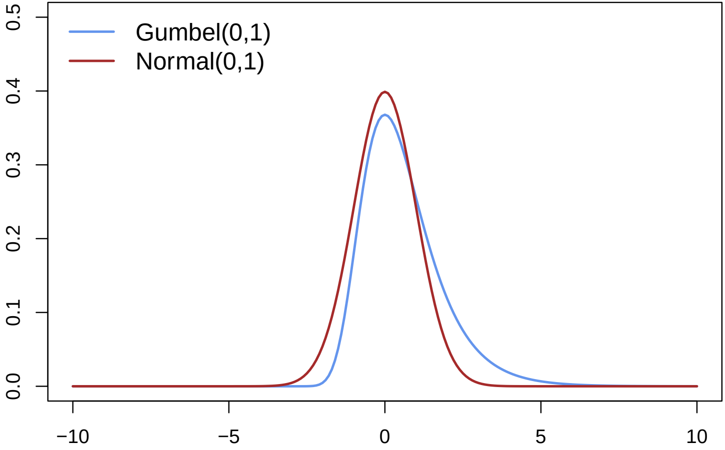
- The difference between two extreme value random variables is distributed as a logistic distribution. Let $e_1, e_2 \sim \text{Gumbel}(0,1)$ and let $\Delta e = e_2 - e_1$. Then the CDF of Δe is[32]:

$$\mathcal{F}(\Delta e) = \frac{\exp(\Delta e)}{1 + \exp(\Delta e)}$$

- In practice, assuming $e_{ij} \sim^{i.i.d.}$ Gumbel is nearly the same as assuming $e_{ij} \sim^{i.i.d.}$ Normal.
  - The extreme value distribution has fatter tails than the normal, but the difference is small empirically.

[32] i.e., the CDF of the logistic distribution is the sigmoid function. See page 11.

Extreme Value Distribution



Logistic Regression as RUM

We can always normalize the scale of (33) so that $\sigma = 1$:

$$U_{ij} = x_{ij}'\beta_j + e_{ij}$$

, where

$$e_{ij} \sim^{i.i.d.} \text{Gumbel}(0, 1)$$

Logistic Regression as RUM

Let $x_i = \{x_{ij}\}_{j=1}^{J}$ and $V_{ij} = x_{ij}'\beta_j$. We have:

$$
\begin{aligned}
\Pr\left(y_i = j \middle| x_i\right) &= \Pr\left(V_{ij} + e_{ij} > V_{i\ell} + e_{i\ell} \;\; \forall \ell \neq j \middle| x_i\right) \\
&= \Pr\left(e_{i\ell} < V_{ij} - V_{i\ell} + e_{ij} \;\; \forall \ell \neq j \middle| x_i\right) \\
&= \int \left[\prod_{\ell \neq j} e^{-e^{-\left(V_{ij} - V_{i\ell} + e_{ij}\right)}}\right] e^{-e_{ij}} e^{-e^{-e_{ij}}} de_{ij} \\
&= \frac{\exp\left(V_{ij}\right)}{\sum_{\ell=1}^{J} \exp\left(V_{i\ell}\right)}
\end{aligned}
$$

- Under the assumption of $e_{ij} \sim^{i.i.d.} \text{Gumbel}\,(0, 1)$, the RUM framework gives rise to the logistic model.

Logistic Regression as RUM

Under the RUM framework, individual utility is given by

$$U_i = \max_j \{U_{ij}\}$$

Let $\overline{U}_i \doteq \mathbb{E}\left[U_i \middle| x_i\right]$ be the expected utility of individual i conditional on x_i. Then under the assumption of $e_{ij} \sim^{i.i.d.} \text{Gumbel}(0,1)$, we have the following closed-form expression for $\overline{U}_i$[33]:

$$\begin{aligned} \overline{U}_i &= \mathbb{E}\left[\max_j \{U_{ij}\} \middle| x_i\right] \\ &= \log\left[\sum_{j=1}^{J} \exp(V_{ij})\right] \end{aligned}$$

[33] Technically, $\overline{U}_i = \log\left[\sum_{j=1}^{J} \exp(V_{ij})\right] + C$, where C is any constant. This is because we can add any C to $(U_{i1}, \ldots, U_{iJ})$ and the model would be the same.

Logistic vs. Probit

- For binary problems, the probit model, after normalization, is

$$\begin{aligned} U_{iA} &= x_{iA}'\beta_A \\ U_{iB} &= x_{iB}'\beta_B + e_{iB} \end{aligned}$$

, where $e_{iB} \sim \mathcal{N}(0,1)$. Therefore, the probit and the logistic model are basically the same for binary problems.

- For multinomial problems, the two types of models are different as probit allows e_i to have an arbitrary covariance structure[34].

[34] In the econometrics literature, logistic and probit models with alternative-specific regressors are called **conditional logit** and **conditional probit models**, so as to be distinguished from logistic and probit models with only individual-specific regressors.

35

[35] For binary problems, if there are only individual-specific variables, then the probit model, after normalization, is

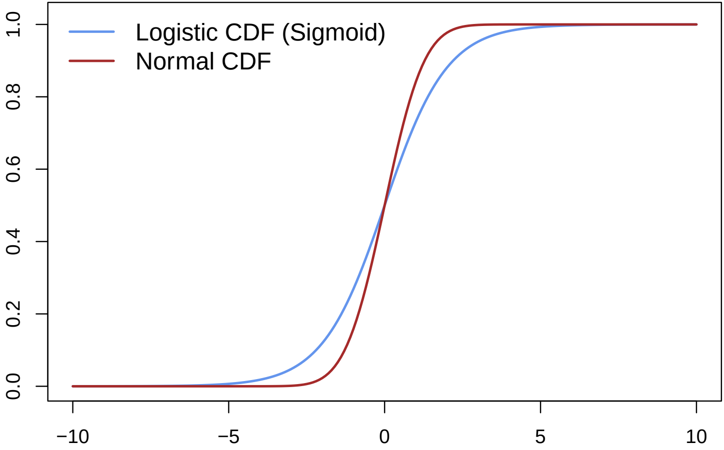
$$\begin{aligned} U_{iA} &= 0 \\ U_{iB} &= x_i'\beta + e_{iB} \end{aligned} \tag{34}$$

, where β is the scaled difference between β_B and β_A. (34) $\Rightarrow$

$$\Pr(y_i = B) = \Phi\left(x_i'\beta\right) \tag{35}$$

Compare (35) with the logistic model, one can see that since $\Phi(.)$ and $\sigma(.)$ are close, the probit and the logistic model are basically the same – they yield very similar conditional choice probability estimates – for binary problems.

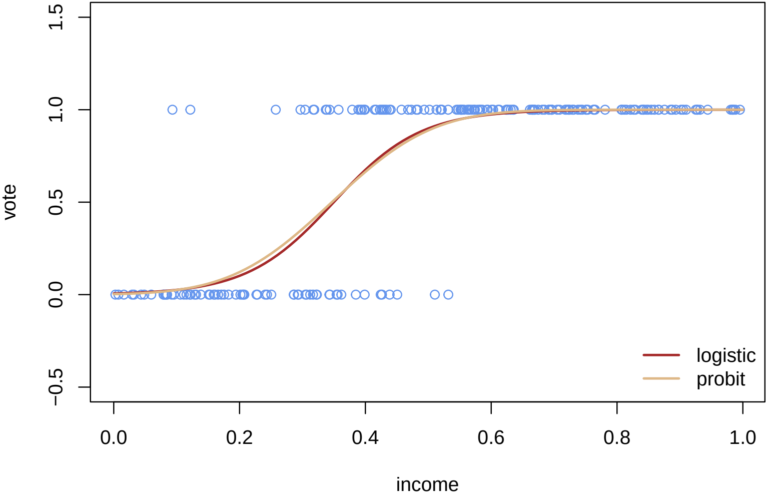
Logistic vs. Probit



Income and Voting

```
probitfit <- glm(vote ~ income, family=binomial(link="probit"))
coeftest(probitfit)

##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -2.75277    0.41935 -6.5644 5.225e-11 ***
## income       7.93916    1.07686  7.3725 1.675e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Income and Voting



Marginal Effects

$$U_{ij} = \alpha_j + \gamma_j s_i + \delta z_{ij} + e_{ij}, \quad e_{ij} \sim^{i.i.d.} \text{Gumbel}(0,1) \tag{36}$$

$\Rightarrow$[36]

$$\frac{\partial \Pr(y_i = j | x_i)}{\partial s_i} = \frac{\partial \left[e^{V_{ij}} \Big/ \sum_\ell e^{V_{i\ell}} \right]}{\partial s_i}$$
$$= \Pr(y_i = j | x_i) \left(\gamma_j - \sum_\ell \gamma_\ell \Pr(y_i = \ell | x_i) \right)$$
$$\frac{\partial \Pr(y_i = j | x_i)}{\partial z_{ij}} = \delta \Pr(y_i = j | x_i) \left(1 - \Pr(y_i = j | x_i)\right)$$

[36] If δ is alternative-specific, i.e. δ_j, then $\partial \Pr(y_i = j | x_i) / \partial z_{ij} = \delta_j \Pr(y_i = j | x_i) (1 - \Pr(y_i = j | x_i))$.

Marginal Effects

- For alternative-specific variables, the sign of the coefficient is the sign of the marginal effect: $\gamma > 0 \iff \partial \Pr(y_i = j | x_i) / \partial z_{ij} > 0$.

- For individual-specific variables, the sign of the coefficient is not necessarily the sign of the marginal effect: $\gamma_j > 0$ does not imply $\partial \Pr(y_i = j | x_i) / \partial s_i > 0$.

Let $\mathcal{E}_i^{jj}$ be the **own-elasticity** of the change in $\Pr(y_i = j | x_i)$ given a change in z_{ij}. (36) $\Rightarrow$

$$\begin{aligned} \mathcal{E}_i^{jj} &= \frac{\partial \Pr(y_i = j | x_i)}{\partial z_{ij}} \frac{z_{ij}}{\Pr(y_i = j | x_i)} \\ &= \delta z_{ij} \left[1 - \Pr(y_i = j | x_i)\right] \end{aligned} \tag{37}$$

Similarly, we can calculate the **cross-elasticity** of $\Pr(y_i = j | x_i)$ given a change in z_{ik}, $k \neq j$:

$$\begin{aligned} \mathcal{E}_i^{jk} &= \frac{\partial \Pr(y_i = j | x_i)}{\partial z_{ik}} \frac{z_{ik}}{\Pr(y_i = j | x_i)} \\ &= -\delta z_{ik} \Pr(y_i = k | x_i) \end{aligned} \tag{38}$$

Choice Probability Elasticity

- Note that (38) does *not* depend on j – a percentage change in z_{ik} results in the *same* percentage change in all $\Pr\left(y_i = j \mid x_i\right),\ j \neq k$.

- For example, consider the car market. Suppose the choice set is $\{$Honda, Toyota, Tesla$\}$. Let $z_{ij} = p_{ij}$ be the price of each car to each consumer. Then (38) says that, for each consumer, a 1% decrease in the price of Honda will result in the same percentage decrease in the probability of buying Toyota and the probability of buying Tesla.

- This property, which is called proportional substitution, is a manifestation of the IIA property of the logistic model.

Independence of Irrelevant Alternatives (IIA)

- The IIA property is the result of assuming that errors are independent of each other.
  - Hence IIA holds not only for logistic models with $i.i.d.$ extreme value distributed errors, but holds in general for discrete choice models with independently distributed errors.
- Multinomial probit models, by allowing for correlated errors, do not have the IIA property.

Independence of Irrelevant Alternatives (IIA)

- Note that the IIA property should be a desirable property for well-specified models.
- Under independence, the error for one alternative provides no information about the error for another alternative. This should be the property of a well-specified model such that the unobserved portion of utility is essentially "white noise."
- When a model omits important unobserved variables that explain individual choice patterns, however, the errors can become correlated over alternatives.
- In this sense, the ultimate goal of the researcher is to represent utility so well that the assumption of error independence is appropriate.
- In the absence of that, a discrete choice model that allows for correlated errors, such as the multinomial probit, can be used.

Employment

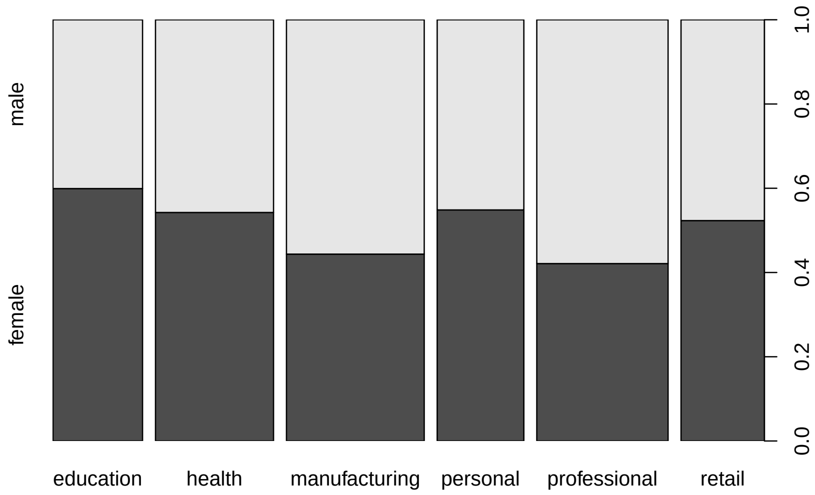
- Sector of employment: Manufacturing, Retail, Education, Health, Personal Service, Professional Service
- Individual variables: sex, education (years of schooling), wage

```
emp <- read.csv("employment.csv")
emp$sex <- factor(emp$sex,labels=c("male","female"))
head(emp,4)

##      sex education     wage        sector
## 1 female        15 32241.35      personal
## 2 female        16 70051.50     education
## 3   male        13 35248.51 manufacturing
## 4 female        12 15535.13        health
```

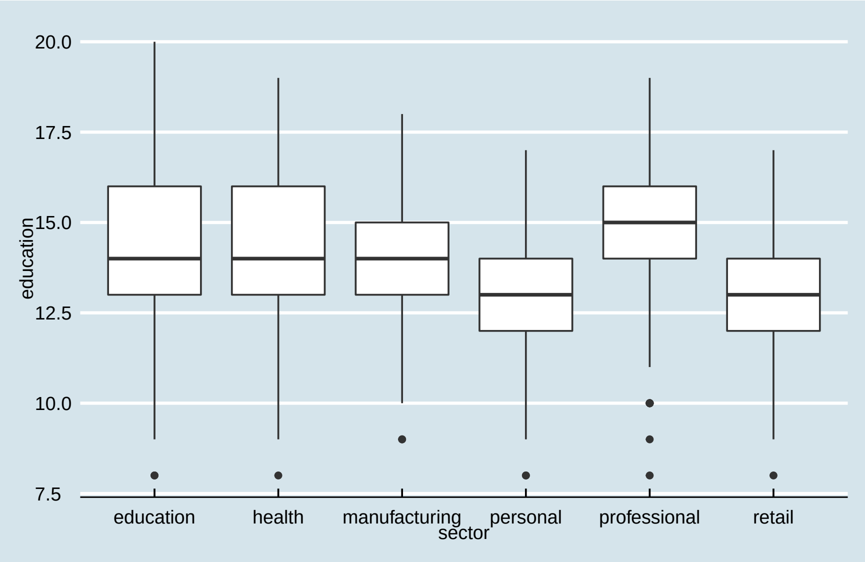
Employment

```
require(descr)
freq(emp$sector,plot=FALSE)

## emp$sector
##               Frequency Percent
## education           277   13.85
## health              365   18.25
## manufacturing       426   21.30
## personal            268   13.40
## professional        406   20.30
## retail              258   12.90
## Total              2000  100.00

aggregate(wage~sector,emp,mean)

##          sector     wage
## 1     education 57134.48
## 2        health 50039.96
## 3 manufacturing 43630.54
## 4      personal 36799.96
## 5  professional 85319.71
## 6        retail 25460.33
```

Employment


male
female
1.0
0.8
0.6
0.4
0.2
0.0
education
health
manufacturing
personal
professional
retail

Employment



20.0
17.5
15.0
12.5
10.0
7.5
education
education
health
manufacturing
personal
professional
retail
sector

Model:

$$U_{ij} = \alpha_j + \beta w_{ij} + e_{ij} \tag{39}$$
$$e_{ij} \sim \text{Gumbel}\,(0, 1)$$

Let y_i be the observed sector of employment of individual i. To estimate the model, we need to construct *counterfactual wages* w_{ij} for each individual i and sector $j \neq y_i$.

Employment

We can predict counterfactual wages by running the following regressions for each sector j:

$$\log w_{ij} = \omega_{0j} + \omega_{1j}\text{Education}_i + \omega_{2j}\text{Female}_i + \omega_{3j}\text{Education}_i \times \text{Female}_i + \xi_{ij} \tag{40}$$

, where Female_i is an indicator variable.

(40) $\Rightarrow \widehat{w}_{ij}$. We then estimate:

$$\begin{aligned} U_{ij} &= \alpha_j + \beta\widehat{w}_{ij} + e_{ij} \\ e_{ij} &\sim \text{Gumbel}\,(0, 1) \end{aligned}$$

Employment

Constructed data set with counterfactual wages:

```
head(emp,4)

##          sector wage.education wage.health wage.manufacturing wage.perso
## 1      personal      36373.753    45757.89           37138.46      45022
## 2     education      60971.110    69129.87           50215.49      50944
## 3 manufacturing      15656.873    21219.96           33982.85      37336
## 4        health       7722.895    13269.87           15023.89      31076
##   wage.professional wage.retail
## 1          54747.97    32333.67
## 2          83152.41    40173.16
## 3          33341.71    24485.34
## 4          15625.99    16858.23
```

Employment

```r
# Estimating the discrete choice model
require(AER)
emp.long <- mlogit.data(emp,shape="wide",varying=2:7,choice="sector")
modelfit <- mlogit(sector ~ wage, emp.long)
coeftest(modelfit)

##
## t test of coefficients:
##
##                               Estimate  Std. Error t value  Pr(>|t|)
## (Intercept):health          8.7959e-02  8.1429e-02  1.0802   0.28019
## (Intercept):manufacturing  1.7359e-01  8.2219e-02  2.1113   0.03487 *
## (Intercept):personal      -3.8266e-01  9.5724e-02 -3.9975 6.634e-05 ***
## (Intercept):professional  -3.9360e-01  9.7211e-02 -4.0489 5.342e-05 ***
## (Intercept):retail          5.5781e-02  8.8256e-02  0.6320   0.52743
## wage                        3.7627e-05  2.6104e-06 14.4142 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Welfare Analysis

The expected utility of individual i is:

$$\overline{U}_i = \log\left[\sum_j \exp\left(\alpha_j + \beta w_{ij}\right)\right] \tag{41}$$

Let $\overline{U}_i^{\$}$ denote the utility of the individual *in monetary terms*. Since in model (39), each dollar in wage adds β to utility, each unit of utility is equivalent to $1/\beta$ dollars. The expected utility of individual i in monetary terms is thus[37]:

$$\overline{U}_i^{\$} = \frac{1}{\beta}\log\left[\sum_j \exp\left(\alpha_j + \beta w_{ij}\right)\right] \tag{42}$$

[37] More precisely, we can add any constant C to (41) and (42).

Welfare Analysis

```r
# Calculating expected utilities
J <- 6 # number of sectors
N <- nrow(emp) # number of individuals
b <- coef(modelfit)["wage"]
X <- model.matrix(modelfit)
V <- X %*% coef(modelfit)
V <- matrix(V,N,J,byrow=TRUE)
U <- log(rowSums(exp(V)))/b
summary(U)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   52366   68246   84799   94882  101307  564822
```

Counterfactual Experiment: 20% decrease in Manufacturing wages

Suppose trade liberalization causes a 20% decrease in the wages of manufacturing workers.

- How does the employment pattern change after trade liberalization?
- What are its welfare consequences?

Counterfactual Experiment:
20% decrease in Manufacturing wages

```
emp2 <- emp
emp2$wage.manufacturing <- emp$wage.manufacturing*0.8
emp2.long <- mlogit.data(emp2,shape="wide",varying=2:7,choice="sector")
colMeans(predict(modelfit,emp2.long))

##     education        health manufacturing      personal  professional
##     0.1464848     0.1937193     0.1657904     0.1406273     0.2176602
##        retail
##     0.1357180
```

Counterfactual Experiment: 20% decrease in Manufacturing wages

Employment Share Before and After Trade Liberalization

Employment Share	Before	After
Manufacturing	21.35	16.63
Retail	12.75	13.41
Education	14.10	14.91
Health	18.40	19.52
Personal Service	12.85	13.49
Professional Service	20.55	22.05

Counterfactual Experiment:
20% decrease in Manufacturing wages

```r
# Calculating expected utilities
X2 <- X
X2[index(emp.long)$alt=="manufacturing","wage"] <-
  X2[index(emp.long)$alt=="manufacturing","wage"]*.8
V2 <- X2 %*% coef(modelfit)
V2 <- matrix(V2,N,J,byrow=TRUE)
U2 <- log(rowSums(exp(V2)))/b
summary(U2)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   52192   67498   82421   93295   97975  564818
```

Counterfactual Experiment: 20% decrease in Manufacturing wages

```
# Change in expected utilities
dU <- U2 - U
summary(dU)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -4026.199 -2377.896 -1161.950 -1587.433  -755.802    -0.146

emp <- data.frame(emp0,U,U2,dU)

# by gender
aggregate(dU ~ sex,emp,mean)

##      sex         dU
## 1   male -2342.3223
## 2 female  -841.5479
```

Counterfactual Experiment: 20% decrease in Manufacturing wages

```
# by education
aggregate(dU ~ education,emp,mean)

##    education          dU
## 1          8  -199.81766
## 2          9  -268.37735
## 3         10  -424.76973
## 4         11  -587.90009
## 5         12  -818.89454
## 6         13 -1228.86410
## 7         14 -1643.87818
## 8         15 -2208.52149
## 9         16 -2637.40772
## 10        17 -2069.31717
## 11        18  -939.75103
## 12        19  -143.24862
## 13        20    -2.87952
```

Ketchup

Let's take model (31) and compare logistic vs. probit counterfactual predictions:

```
logitfit <- mlogit(choice ~ price|income, ketchup.long, reflevel="stb")
coeftest(logitfit)

##
## t test of coefficients:
##
##                        Estimate Std. Error  t value  Pr(>|t|)
## (Intercept):delmonte  -3.831626   1.169149  -3.2773  0.001094 **
## (Intercept):heinz    -10.888985   0.946463 -11.5049 < 2.2e-16 ***
## (Intercept):hunts     -6.305256   0.871547  -7.2346 1.103e-12 ***
## price                 -4.418198   0.329590 -13.4051 < 2.2e-16 ***
## income:delmonte        0.107143   0.025841   4.1462 3.745e-05 ***
## income:heinz           0.276613   0.020943  13.2078 < 2.2e-16 ***
## income:hunts           0.180305   0.019794   9.1091 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Counterfactual Experiment: 20% price increase for Heinz

```
newdata <- ketchup.long
idx <- index(newdata)$alt == "heinz"
newdata[idx,"price"] <- newdata[idx,"price"]*1.2 # 20% price increase

# logistic prediction
logit.phat.new <- predict(logitfit,newdata)
logit.share.new <- colMeans(logit.phat.new)
logit.share.new

##          stb   delmonte      heinz      hunts
## 0.25132916 0.06914047 0.37982532 0.29970505

# probit prediction
probit.phat.new <- predict(probitfit1,newdata)
probit.share.new <- colMeans(probit.phat.new)
probit.share.new

##          stb   delmonte      heinz      hunts
## 0.22741067 0.07871089 0.37283446 0.32164539
```

Counterfactual Experiment: 20% price increase for Heinz

market share	Heinz	Hunts	Del Monte	Store Brand
	51.13%	21.38%	5.38%	22.13%
After Heinz price increase:				
logistic	37.98%	29.97%	6.91%	25.13%
probit	37.28%	32.16%	7.87%	22.74%

Mode of Transportation

```r
## Probit Regression
transport.long <- mlogit.data(transport, shape="wide", choice="y")
probitfit <- mlogit(y ~ 0|loginc+distance, transport.long, probit=TRUE)
```

```r
coeftest(probitfit)

##
## t test of coefficients:
##
##                      Estimate Std. Error t value   Pr(>|t|)
## car:(intercept)    -8.925928   1.389554 -6.4236 2.062e-10 ***
## subway:(intercept) -1.454769   1.609180 -0.9040    0.3662
## car:loginc          0.773128   0.128574  6.0131 2.555e-09 ***
## subway:loginc       0.118611   0.133202  0.8905    0.3734
## car:distance        0.557613   0.532888  1.0464    0.2956
## subway:distance     0.698667   0.772920  0.9039    0.3663
## car.subway         -0.013351   0.153096 -0.0872    0.9305
## subway.subway       0.315844   0.364598  0.8663    0.3865
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Mode of Transportation

```
probitfit$omega

## $bus
##                car      subway
## car     1.00000000 -0.01335131
## subway -0.01335131  0.09993555
##
## $car
##             bus   subway
## bus    1.000000 1.013351
## subway 1.013351 1.126638
##
## $subway
##            bus       car
## bus 0.09993555 0.1132869
## car 0.11328686 1.1266382
```

Counterfactual Experiment: No Subway

```r
# To predict choice probabilities without one alternative,
# one trick is to make the xij associated with that alternative
# extremely large or small so that its predicted prob is always 0
newdata <- transport.long
idx <- index(newdata)$alt == "subway"
newdata[idx,"loginc"] <- -1e10
newdata[idx,"distance"] <- -1e10
probit.phat.new <- predict(probitfit,newdata)
probit.share.new <- colMeans(probit.phat.new)
```

```r
probit.share.new

##       bus       car    subway
## 0.6047072 0.3952928 0.0000000
```

Counterfactual Experiment: No Subway

Observed Market Share

bus	car	subway
22%	31%	47%

Predicted Market Share without Subway

	bus	car
logistic	38%	62%
probit	60%	40%

Acknowledgement

Part of this lecture is based on the following sources:

- Bishop, C. M. 2011. *Pattern Recognition and Machine Learning*. Springer.
- Hastie, T., R. Tibshirani, and J. Friedmand. 2008. *The Elements of Statistical Learning* (2^{nd} ed.). Springer.
- James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. *An Introduction to Statistical Learning: with Applications in R*. Springer.
- Schafer, J. S. *Regression Analysis and Modeling*. Lecture at Penn State University, personal copy.
- Train, K. E. 2009. *Discrete Choice Methods with Simulation* (2^{nd} ed.). Cambridge University Press.