Binary Dependent Variables

Philip Leifeld

GV903: Advanced Research Methods, Week 16

1. Binary Dependent Variables

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.
- War versus peace per country.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.
- War versus peace per country.
- Survey respondent responds or shuts the door.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.
- War versus peace per country.
- Survey respondent responds or shuts the door.
- Immigrants with dual citizenship vote in Britain or in their country of origin.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.
- War versus peace per country.
- Survey respondent responds or shuts the door.
- Immigrants with dual citizenship vote in Britain or in their country of origin.
- Swing voter or loyal voter.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.
- War versus peace per country.
- Survey respondent responds or shuts the door.
- Immigrants with dual citizenship vote in Britain or in their country of origin.
- Swing voter or loyal voter.
- Prime minister resigns early or stays for the whole term.

Binary Dependent Variables in Political Science

There are thousands of examples of binary data of interest to political scientists, such as:

- Yea or nay votes (e. g., roll-call votes in Congress).
- Individual decisions to vote or not in an election.
- Being in or out of the labour force.
- War versus peace per country.
- Survey respondent responds or shuts the door.
- Immigrants with dual citizenship vote in Britain or in their country of origin.
- Swing voter or loyal voter.
- Prime minister resigns early or stays for the whole term.

How can we analyse these data as dependent variables?

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

vote_obama: DV: Did respondent vote for Obama or Romney?

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem    : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep    : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race      : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income    : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex       : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white     : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black     : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`ft_dem`: Feeling thermometer – Democratic Party

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`ft_rep`: Feeling thermometer – Republican Party

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`income`: 28 ordinal income categories of $5,000 each

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`white`: Is the respondent white?

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`black`: Is the respondent black?

American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`hispanic`: Is the respondent hispanic?

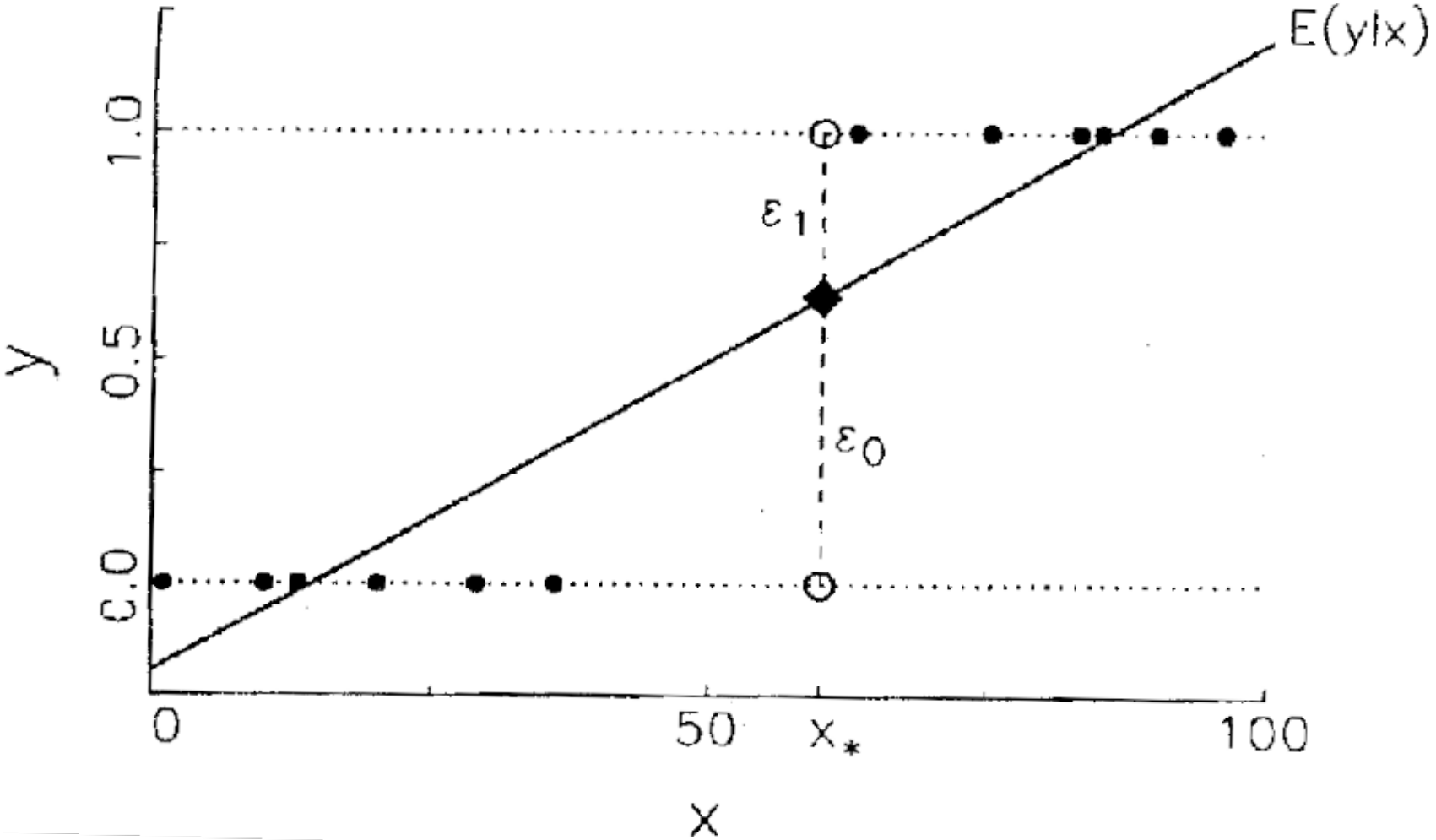
American National Election Survey: Voting for Obama

```
ANES <- read.csv("dataset-anes-2012-subset1.csv")
str(ANES)
## 'data.frame': 3870 obs. of  10 variables:
##  $ ft_dem     : int  100 100 70 100 50 30 70 0 15 85 ...
##  $ ft_rep     : int  0 85 40 15 60 15 30 60 40 30 ...
##  $ race       : int  2 2 2 2 1 1 1 1 1 1 ...
##  $ income     : int  3 2 13 1 9 27 24 25 26 7 ...
##  $ sex        : int  1 1 0 1 0 0 1 1 0 1 ...
##  $ white      : int  0 0 0 0 1 1 1 1 1 1 ...
##  $ black      : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ hispanic   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ other      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vote_obama: int  1 1 1 1 0 0 1 0 0 1 ...
```

`other`: Other ethnicity than white/black/hispanic

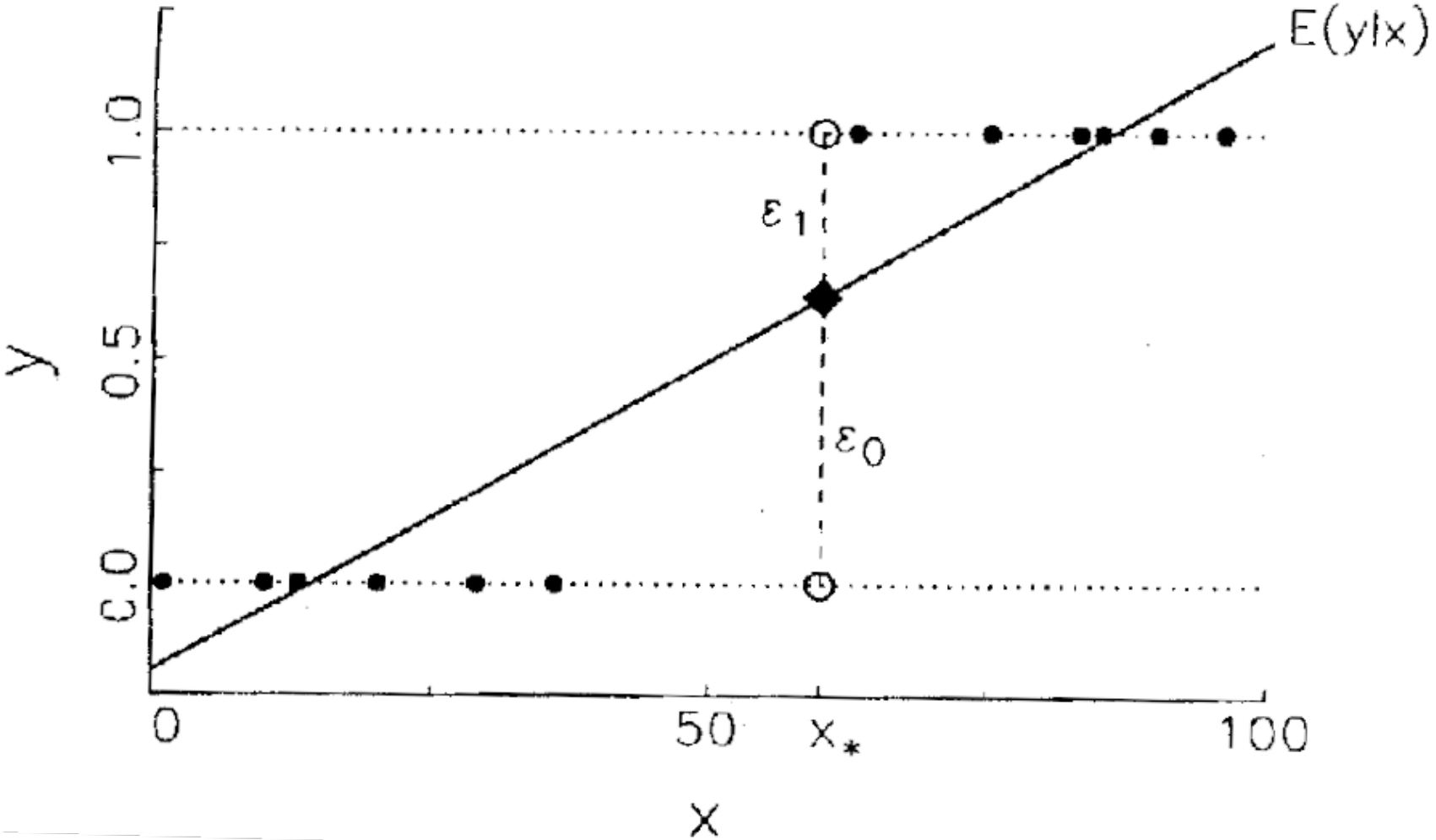
The Linear Probability Model

(= Application of the linear model to binary data. This is usually a bad idea. . .)



The Linear Probability Model

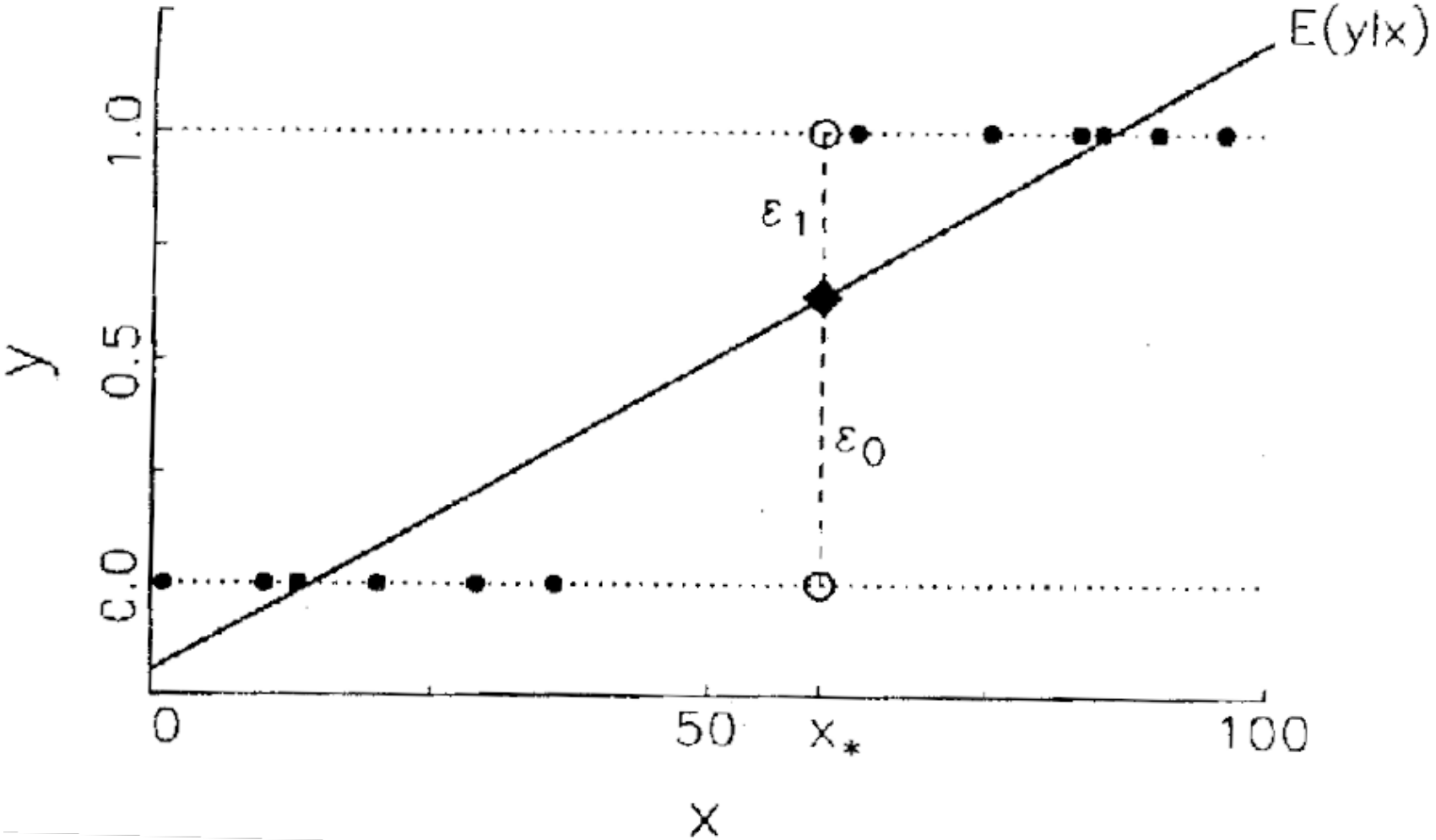
(= Application of the linear model to binary data. This is usually a bad idea. . .)



- Prediction of values outside of $[0; 1]$. . .

The Linear Probability Model

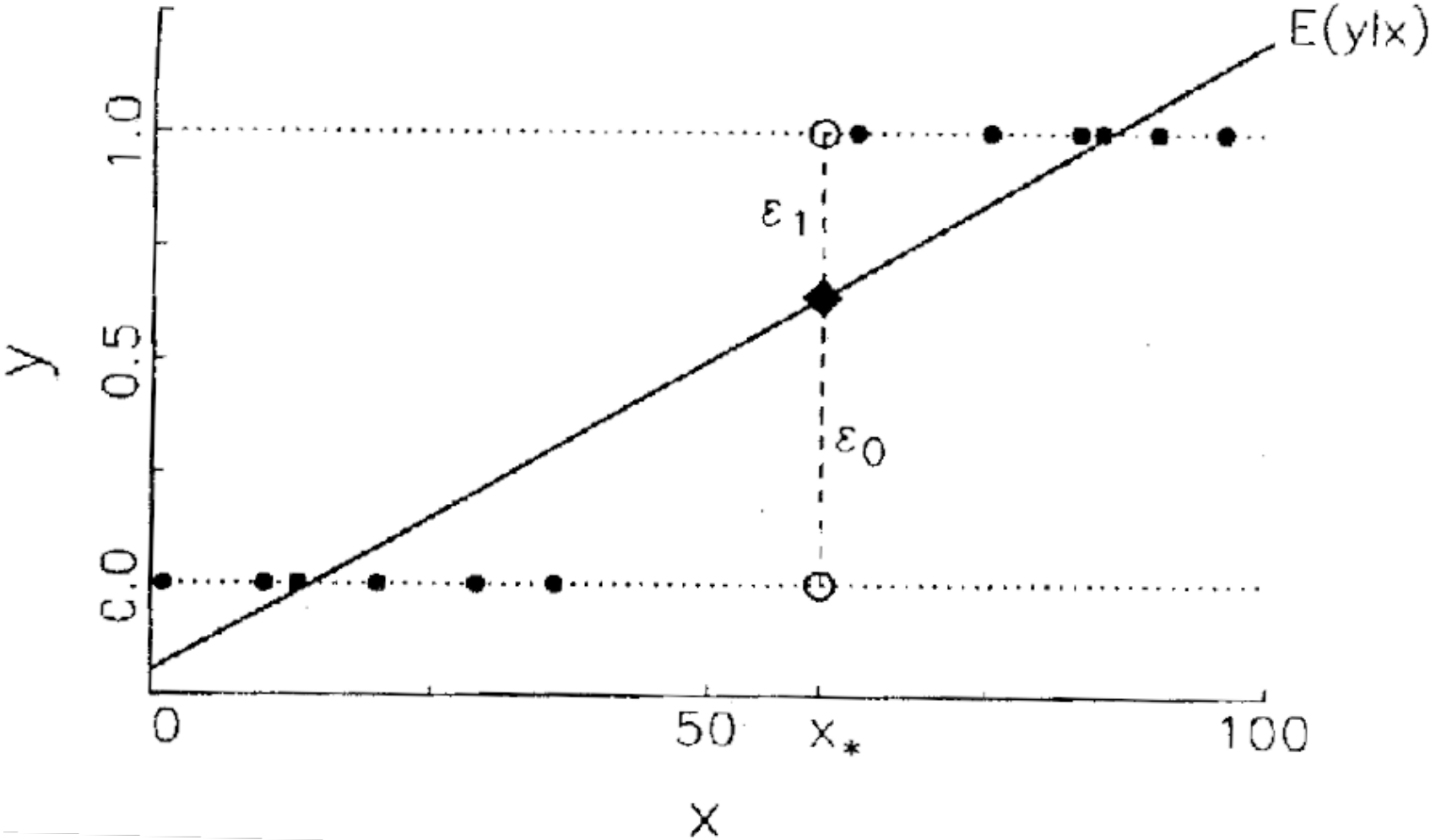
(= Application of the linear model to binary data. This is usually a bad idea. . .)



- Prediction of values outside of $[0; 1]$. . .
- We almost always have heteroskedasticity. . .

The Linear Probability Model

(= Application of the linear model to binary data. This is usually a bad idea. . .)



- Prediction of values outside of $[0; 1]$. . .
- We almost always have heteroskedasticity. . .
- Assuming linear increases in probability. . .

The Generalised Linear Model (GLM)

- The GLM comes to the rescue.

The Generalised Linear Model (GLM)

- The GLM comes to the rescue.
- The idea is that we keep the additive component of the linear model but re-scale it to make it compatible with the outcome distribution of the DV.

The Generalised Linear Model (GLM)

- The GLM comes to the rescue.
- The idea is that we keep the additive component of the linear model but re-scale it to make it compatible with the outcome distribution of the DV.
- That is, instead of using the normal distribution for MLE, we use some other distribution.

The Generalised Linear Model (GLM)

- The GLM comes to the rescue.
- The idea is that we keep the additive component of the linear model but re-scale it to make it compatible with the outcome distribution of the DV.
- That is, instead of using the normal distribution for MLE, we use some other distribution.
- In this case, we use the binomial distribution for binary data.

The Generalised Linear Model (GLM)

- The GLM comes to the rescue.
- The idea is that we keep the additive component of the linear model but re-scale it to make it compatible with the outcome distribution of the DV.
- That is, instead of using the normal distribution for MLE, we use some other distribution.
- In this case, we use the binomial distribution for binary data.
- In `R`, the GLM is implemented in the `glm` function.

The Generalised Linear Model (GLM)

- The GLM comes to the rescue.
- The idea is that we keep the additive component of the linear model but re-scale it to make it compatible with the outcome distribution of the DV.
- That is, instead of using the normal distribution for MLE, we use some other distribution.
- In this case, we use the binomial distribution for binary data.
- In `R`, the GLM is implemented in the `glm` function.
- It works almost like the `lm` function.

Estimation of a GLM in R

```
model <- glm(vote_obama ~ ft_dem + ft_rep + black + hispanic + other +
               income, family = binomial, data = ANES)
summary(model)
##
## Call:
## glm(formula = vote_obama ~ ft_dem + ft_rep + black + hispanic +
##     other + income, family = binomial, data = ANES)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2260  -0.1350   0.0140   0.1639   4.1982
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.223619   0.253391  -0.883   0.3775
## ft_dem       0.093786   0.004278  21.923  < 2e-16 ***
## ft_rep      -0.090827   0.004182 -21.716  < 2e-16 ***
## black        3.165810   0.394835   8.018 1.07e-15 ***
## hispanic     0.969995   0.189366   5.122 3.02e-07 ***
## other        0.558628   0.254384   2.196   0.0281 *
## income      -0.025063   0.008530  -2.938   0.0033 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5247.0  on 3869  degrees of freedom
## Residual deviance: 1467.8  on 3863  degrees of freedom
## AIC: 1481.8
##
## Number of Fisher Scoring iterations: 8
```

Estimation of a GLM in R

```
model <- glm(vote_obama ~ ft_dem + ft_rep + black + hispanic + other +
               income, family = binomial, data = ANES)
summary(model)
##
## Call:
## glm(formula = vote_obama ~ ft_dem + ft_rep + black + hispanic +
##     other + income, family = binomial, data = ANES)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2260  -0.1350   0.0140   0.1639   4.1982
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.223619   0.253391  -0.883   0.3775
## ft_dem       0.093786   0.004278  21.923  < 2e-16 ***
## ft_rep      -0.090827   0.004182 -21.716  < 2e-16 ***
## black        3.165810   0.394835   8.018 1.07e-15 ***
## hispanic     0.969995   0.189366   5.122 3.02e-07 ***
## other        0.558628   0.254384   2.196   0.0281 *
## income      -0.025063   0.008530  -2.938   0.0033 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5247.0  on 3869  degrees of freedom
## Residual deviance: 1467.8  on 3863  degrees of freedom
## AIC: 1481.8
##
## Number of Fisher Scoring iterations: 8
```

Now how do we interpret these estimates? 

2. Details of the Model

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

We have encountered link functions before when modelling functions of the DV, such as the log.

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

We have encountered link functions before when modelling functions of the DV, such as the log.

Interpretation of $\log Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$:

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

We have encountered link functions before when modelling functions of the DV, such as the log.

Interpretation of $\log Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$:

Percentage change in the DV. Can be used for skewed DVs.

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

We have encountered link functions before when modelling functions of the DV, such as the log.

Interpretation of $\log Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$:

Percentage change in the DV. Can be used for skewed DVs.

We can rewrite this as: $F(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, where $F(Y) = \log(Y)$.

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

We have encountered link functions before when modelling functions of the DV, such as the log.

Interpretation of $\log Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$:

Percentage change in the DV. Can be used for skewed DVs.

We can rewrite this as: $F(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, where $F(Y) = \log(Y)$.

$F(\cdot)$ is called a link function.

Link Functions

Idea: we can try to map the probability scale defined by 0s and 1s to an unbounded continuous scale.

The function for this mapping is called a link function.

We have encountered link functions before when modelling functions of the DV, such as the log.

Interpretation of $\log Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$:

Percentage change in the DV. Can be used for skewed DVs.

We can rewrite this as: $F(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, where $F(Y) = \log(Y)$.

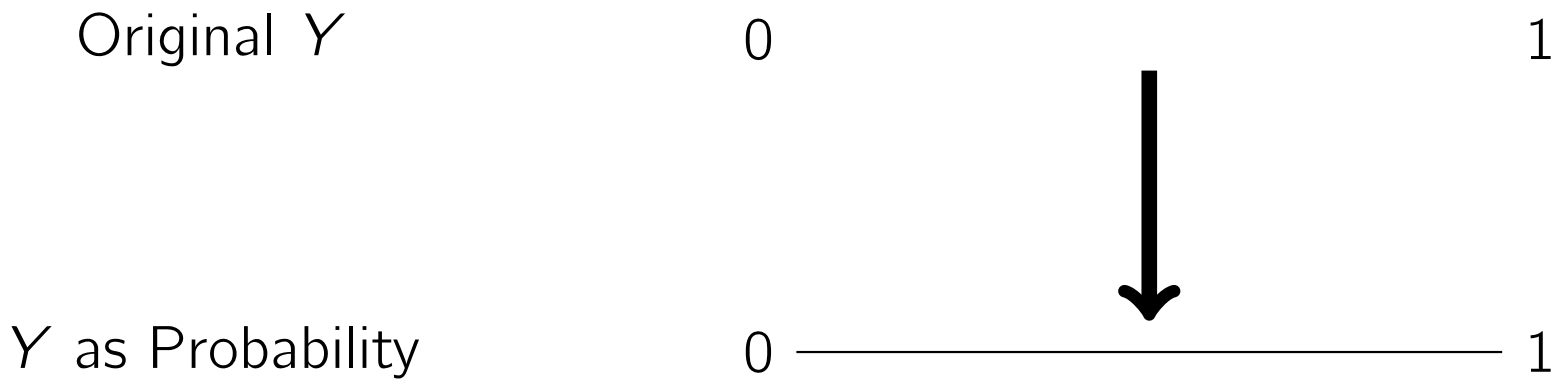
$F(\cdot)$ is called a link function.

So what link function maps Y to continuous $F(Y) \in [-\infty; \infty]$?

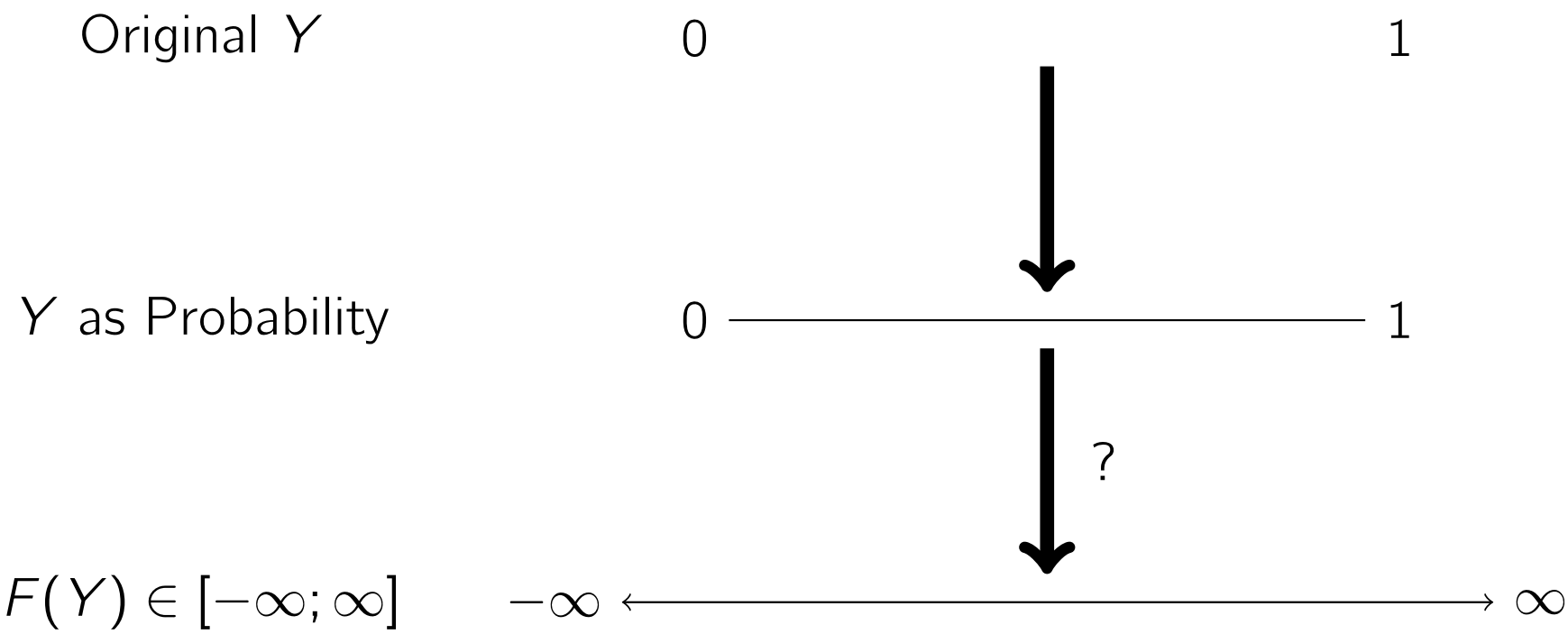
Mapping a Binary DV to a Continuous Scale

Original Y	0	1

Mapping a Binary DV to a Continuous Scale



Mapping a Binary DV to a Continuous Scale



The Probit Link Function

Actually this is rather about mapping the right-hand side of the equation onto a bounded probability scale.

The Probit Link Function

Actually this is rather about mapping the right-hand side of the equation onto a bounded probability scale.

The cumulative distribution function (cdf) of the standard normal distribution can map unbounded values onto a [0; 1] scale!

$$Y = \Phi(\mathbf{X}\beta + \epsilon)$$

The Probit Link Function

Actually this is rather about mapping the right-hand side of the equation onto a bounded probability scale.

The cumulative distribution function (cdf) of the standard normal distribution can map unbounded values onto a [0; 1] scale!

$$Y = \Phi(\mathbf{X}\beta + \epsilon)$$

Or conversely:

$$\Phi^{-1}(Y) = \mathbf{X}\beta + \epsilon$$

The Probit Link Function

Actually this is rather about mapping the right-hand side of the equation onto a bounded probability scale.

The cumulative distribution function (cdf) of the standard normal distribution can map unbounded values onto a [0; 1] scale!

$$Y = \Phi(\mathbf{X}\beta + \epsilon)$$

Or conversely:

$$\Phi^{-1}(Y) = \mathbf{X}\beta + \epsilon$$

$F(Y) = \Phi^{-1}(Y)$ is known as the probit link function.

The Logit Link Function

An alternative to this is the logit transformation.

The Logit Link Function

An alternative to this is the logit transformation.

Any probability can be expressed as an odds-ratio.

The Logit Link Function

An alternative to this is the logit transformation.

Any probability can be expressed as an odds-ratio.

For example, let $p = 0.16$ denote the probability of civil war. I. e., 16 out of 100 countries have a civil war. The odds of civil war are 16:84 or $\frac{16}{84} = 0.19$. The odds of peace are 84:16 or $\frac{84}{16} = 5.25$. These are called odds-ratios.

The Logit Link Function

An alternative to this is the logit transformation.

Any probability can be expressed as an odds-ratio.

For example, let $p = 0.16$ denote the probability of civil war. I. e., 16 out of 100 countries have a civil war. The odds of civil war are 16:84 or $\frac{16}{84} = 0.19$. The odds of peace are 84:16 or $\frac{84}{16} = 5.25$. These are called odds-ratios.

Values between 0 and 1 indicate ≤ 0.5 probability. Values above 1 indicate probabilities above 0.5, and there is no upper bound. To avoid this skewness of the odds-ratios, their logarithm can be computed in order to scale them symmetrically around 0.

The Logit Link Function

This is the well-known logit transformation:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right),$$

The Logit Link Function

This is the well-known logit transformation:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right),$$

which can be mapped back onto probability scale $[0; 1]$ with the inverse logit transformation:

$$\text{logit}^{-1}(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

The Logit Link Function

This is the well-known logit transformation:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right),$$

which can be mapped back onto probability scale $[0; 1]$ with the inverse logit transformation:

$$\text{logit}^{-1}(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

In practice, it does not matter if we use logit or probit link functions as long as we remember the inverse transformation to get back to the original scale.

The Logit Link Function

This is the well-known logit transformation:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right),$$

which can be mapped back onto probability scale $[0; 1]$ with the inverse logit transformation:

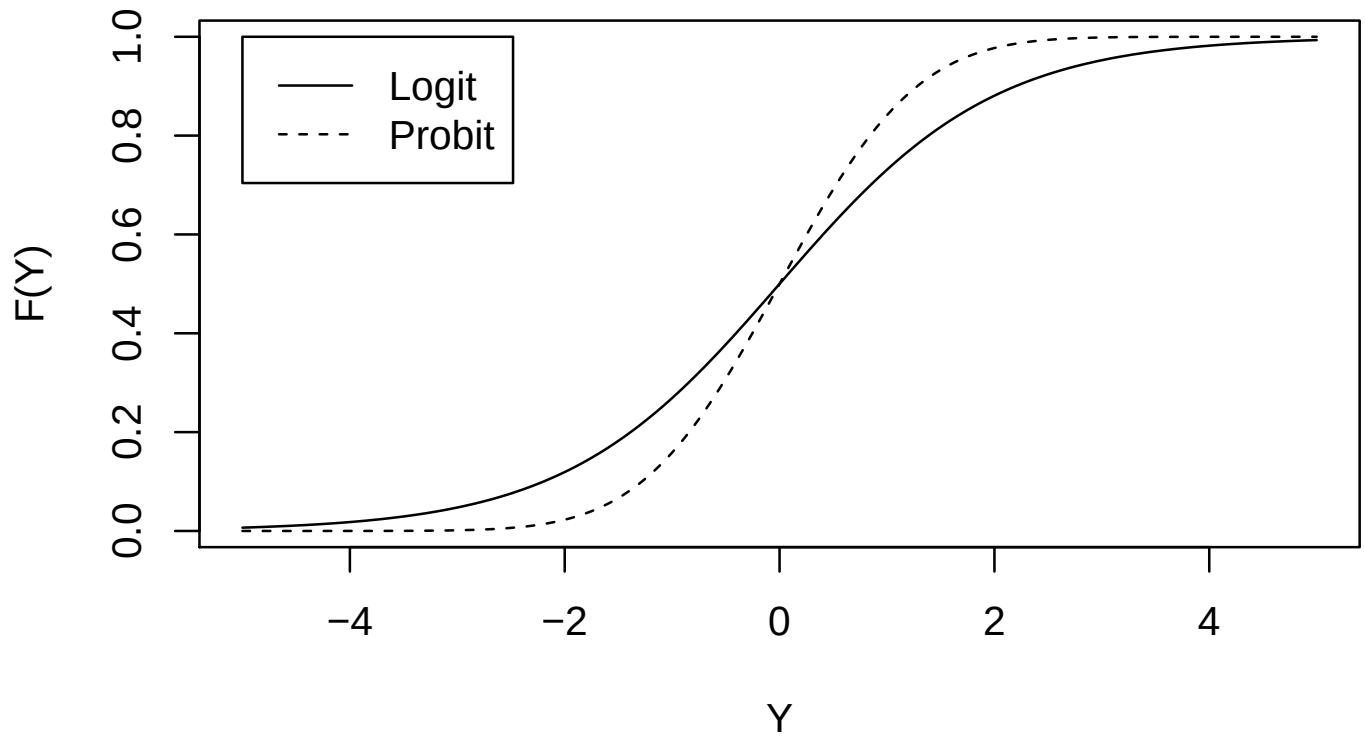
$$\text{logit}^{-1}(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

In practice, it does not matter if we use logit or probit link functions as long as we remember the inverse transformation to get back to the original scale.

Either transformation can be used to map a linear (= additive) model onto the probability scale.

Logit and Probit in Comparison

```
Y <- seq(-5, 5, 0.05)
plot(Y, 1 / (1 + exp(-Y)), type = "l", xlab = "Y",
     ylab = "F(Y)")
lines(Y, pnorm(Y), lty = "dashed")
legend(-5, 1, legend = c("Logit", "Probit"), lty = 1:2)
```



Both transformations indeed have very similar shapes.

The Logit Model

Binary outcomes can be modelled as Bernoulli trials (e. g., coin flip). Bernoulli trials are defined by a single parameter θ for the probability of success. Bernoulli trials are binomial experiments with $n = 1$:

$$P(Y_i = y_i) = \theta_i^{y_i}(1 - \theta_i)^{1-y_i}$$

with each $y_i \in \{0; 1\}$.

The Logit Model

Binary outcomes can be modelled as Bernoulli trials (e. g., coin flip). Bernoulli trials are defined by a single parameter θ for the probability of success. Bernoulli trials are binomial experiments with $n = 1$:

$$P(Y_i = y_i) = \theta_i^{y_i}(1 - \theta_i)^{1-y_i}$$

with each $y_i \in \{0; 1\}$. The Bernoulli trials are like a binomial experiment, except there are different probabilities in each trial.

The Logit Model

Binary outcomes can be modelled as Bernoulli trials (e. g., coin flip). Bernoulli trials are defined by a single parameter θ for the probability of success. Bernoulli trials are binomial experiments with $n = 1$:

$$P(Y_i = y_i) = \theta_i^{y_i}(1 - \theta_i)^{1-y_i}$$

with each $y_i \in \{0; 1\}$. The Bernoulli trials are like a binomial experiment, except there are different probabilities in each trial.

Now we can plug the linear part into this model through the link function. This is where the different probabilities come from:

$$\theta_i = \text{logit}^{-1}\left(\mathbf{X}_i^\top \boldsymbol{\beta}\right) = \frac{1}{1 + e^{-\mathrm{X}_i^\top \boldsymbol{\beta}}}$$

The Logit Model

Binary outcomes can be modelled as Bernoulli trials (e. g., coin flip). Bernoulli trials are defined by a single parameter θ for the probability of success. Bernoulli trials are binomial experiments with $n = 1$:

$$P(Y_i = y_i) = \theta_i^{y_i}(1 - \theta_i)^{1-y_i}$$

with each $y_i \in \{0; 1\}$. The Bernoulli trials are like a binomial experiment, except there are different probabilities in each trial.

Now we can plug the linear part into this model through the link function. This is where the different probabilities come from:

$$\theta_i = \text{logit}^{-1}\left(\mathbf{X}_i^\top \boldsymbol{\beta}\right) = \frac{1}{1 + e^{-\mathrm{X}_i^\top \boldsymbol{\beta}}}$$

And voilà: we have a logistic model – a non-linear model for binary data! (And similar for the probit model. . .)

3. Estimation

The Likelihood Function of a Logit Model

We have defined the probability for a single observation, $P(Y_i = y_i)$. We can now define the joint probability for all observations:

$$P(\mathbf{Y}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \theta_i^{y_i}(1-\theta_i)^{1-y_i}$$

where each θ_i is the logit-transformed linear component.

The Likelihood Function of a Logit Model

We have defined the probability for a single observation, $P(Y_i = y_i)$. We can now define the joint probability for all observations:

$$P(\mathbf{Y}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \theta_i^{y_i}(1-\theta_i)^{1-y_i}$$

where each θ_i is the logit-transformed linear component.

The corresponding log likelihood (applying the laws of logarithms):

$$\log \mathcal{L}(\boldsymbol{\theta}|\mathbf{Y}) = \sum_{i=1}^{n} [y_i \log \theta_i + (1-y_i) \log(1-\theta_i)]$$

The Likelihood Function of a Logit Model

We have defined the probability for a single observation, $P(Y_i = y_i)$. We can now define the joint probability for all observations:

$$P(\mathbf{Y}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \theta_i^{y_i}(1-\theta_i)^{1-y_i}$$

where each θ_i is the logit-transformed linear component.

The corresponding log likelihood (applying the laws of logarithms):

$$\log \mathcal{L}(\boldsymbol{\theta}|\mathbf{Y}) = \sum_{i=1}^{n} [y_i \log \theta_i + (1-y_i) \log(1-\theta_i)]$$

This can be further simplified (see Ward and Ahlquist 2018) and then solved analytically. Or we can maximise computationally.

Likelihood Maximisation in R

Let's code up the (negative) log-likelihood function in R:

Likelihood Maximisation in R

Let's code up the (negative) log-likelihood function in R:

```r
negLL <- function(b, X, y) {
  theta <- 1 / (1 + exp(-X %*% b))
  ll <- sum(y * log(theta) + (1 - y) * log(1 - theta))
  return(-ll)
}
```

Likelihood Maximisation in R

Let's code up the (negative) log-likelihood function in R:

```
negLL <- function(b, X, y) {
  theta <- 1 / (1 + exp(-X %*% b))
  ll <- sum(y * log(theta) + (1 - y) * log(1 - theta))
  return(-ll)
}
```

Prepare the design matrix, DV, and $\boldsymbol{\theta}$ start values:

```
myX <- cbind(1, ANES$ft_dem, ANES$ft_rep, ANES$black,
             ANES$hispanic, ANES$other, ANES$income)
myY <- ANES$vote_obama
startval <- rep(0, ncol(myX))
```

Likelihood Maximisation in R

Let's code up the (negative) log-likelihood function in R:

```
negLL <- function(b, X, y) {
  theta <- 1 / (1 + exp(-X %*% b))
  ll <- sum(y * log(theta) + (1 - y) * log(1 - theta))
  return(-ll)
}
```

Prepare the design matrix, DV, and $\boldsymbol{\theta}$ start values:

```
myX <- cbind(1, ANES$ft_dem, ANES$ft_rep, ANES$black,
             ANES$hispanic, ANES$other, ANES$income)
myY <- ANES$vote_obama
startval <- rep(0, ncol(myX))
```

And now we are ready to optimise the function given the data:

```
results <- optim(startval, negLL, hessian = TRUE,
                 method = "BFGS", X = myX, y = myY)
```

Extract the Results

Extract parameters and SEs (via diagonal of VCOV matrix):

```
data.frame(coef = results$par,
           SE = sqrt(diag(solve(results$hessian))))
##          coef          SE
## 1 -0.22379114 0.253417386
## 2  0.09382491 0.004277321
## 3 -0.09086239 0.004182228
## 4  3.16617989 0.394914617
## 5  0.97007557 0.189397775
## 6  0.55869840 0.254419592
## 7 -0.02506850 0.008531221
```

Extract the Results

Extract parameters and SEs (via diagonal of VCOV matrix):

```
data.frame(coef = results$par,
           SE = sqrt(diag(solve(results$hessian))))
##          coef          SE
## 1 -0.22379114 0.253417386
## 2  0.09382491 0.004277321
## 3 -0.09086239 0.004182228
## 4  3.16617989 0.394914617
## 5  0.97007557 0.189397775
## 6  0.55869840 0.254419592
## 7 -0.02506850 0.008531221
```

Extract the log likelihood:

```
results$value
## [1] 733.8795
```

Extract the Results

Extract parameters and SEs (via diagonal of VCOV matrix):

```
data.frame(coef = results$par,
           SE = sqrt(diag(solve(results$hessian))))
##          coef          SE
## 1 -0.22379114 0.253417386
## 2  0.09382491 0.004277321
## 3 -0.09086239 0.004182228
## 4  3.16617989 0.394914617
## 5  0.97007557 0.189397775
## 6  0.55869840 0.254419592
## 7 -0.02506850 0.008531221
```

Extract the log likelihood:

```
results$value
## [1] 733.8795
```

Compute AIC (recap: $\text{AIC} = -2 \log \hat{\mathcal{L}} + 2k$):

```
2 * results$value + 2 * ncol(myX) # (not -2 because negLL)
## [1] 1481.759
```

Let's Look at the `glm` Output Again for Comparison...

```
summary(model)
##
## Call:
## glm(formula = vote_obama ~ ft_dem + ft_rep + black + hispanic +
##     other + income, family = binomial, data = ANES)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2260  -0.1350   0.0140   0.1639   4.1982
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.223619   0.253391  -0.883   0.3775
## ft_dem       0.093786   0.004278  21.923  < 2e-16 ***
## ft_rep      -0.090827   0.004182 -21.716  < 2e-16 ***
## black        3.165810   0.394835   8.018 1.07e-15 ***
## hispanic     0.969995   0.189366   5.122 3.02e-07 ***
## other        0.558628   0.254384   2.196   0.0281 *
## income      -0.025063   0.008530  -2.938   0.0033 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5247.0  on 3869  degrees of freedom
## Residual deviance: 1467.8  on 3863  degrees of freedom
## AIC: 1481.8
##
## Number of Fisher Scoring iterations: 8
```

Estimation of Logit and Probit Models

Logit model again, now specifying logit link function explicitly:

```
model <- glm(vote_obama ~ ft_dem + ft_rep + black +
             hispanic + other + income,
             family = binomial(link = logit), data = ANES)
```

Similarly, we can specify the probit link function:

```
model2 <- glm(vote_obama ~ ft_dem + ft_rep + black +
              hispanic + other + income,
              family = binomial(link = probit), data = ANES)
```

Logit and Probit Comparison

```
library("texreg")
texreg(list(model, model2), single.row = TRUE, dcolumn = TRUE,
       booktabs = TRUE, use.packages = FALSE, table = FALSE,
       custom.model.names = c("Logit", "Probit"))
```

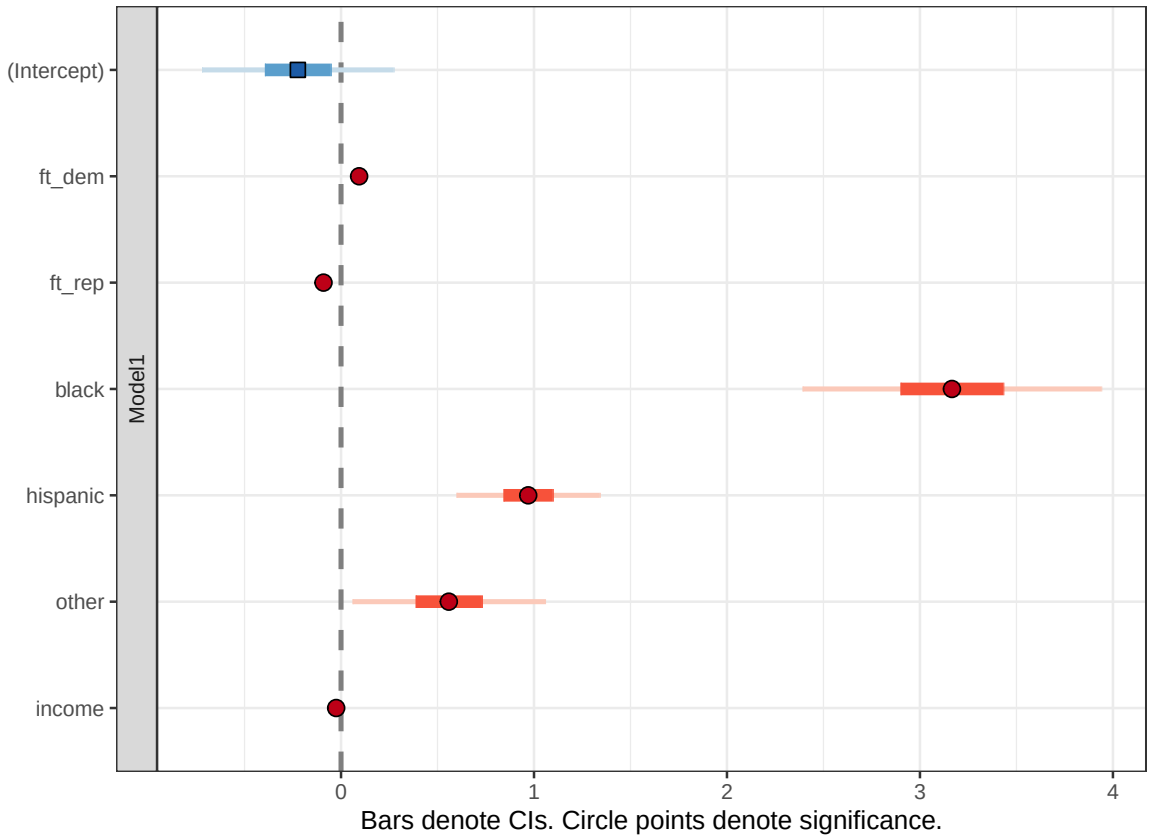
	Logit	Probit
(Intercept)	−0.22 (0.25)	−0.14 (0.14)
ft_dem	0.09 (0.00)***	0.05 (0.00)***
ft_rep	−0.09 (0.00)***	−0.05 (0.00)***
black	3.17 (0.39)***	1.66 (0.18)***
hispanic	0.97 (0.19)***	0.60 (0.10)***
other	0.56 (0.25)*	0.29 (0.14)*
income	−0.03 (0.01)**	−0.01 (0.00)**
AIC	1481.76	1542.69
BIC	1525.59	1586.52
Log Likelihood	−733.88	−764.35
Deviance	1467.76	1528.69
Num. obs.	3870	3870

***$p < 0.001$; **$p < 0.01$; *$p < 0.05$

4. Interpretation

Coefficient Plots in `texreg`

```
plotreg(model)
```



Interpretation of Logit Coefficients

Coefficients are log odds-ratios. These are hard to interpret. We can at least convert them into odds-ratios by exponentiating:

Interpretation of Logit Coefficients

Coefficients are log odds-ratios. These are hard to interpret. We can at least convert them into odds-ratios by exponentiating:

```
exp(coef(model))
## (Intercept)      ft_dem      ft_rep       black    hispanic
##   0.7996197   1.0983245   0.9131759  23.7079342   2.6379315
##       other      income
##   1.7482726   0.9752484
```

Interpretation of Logit Coefficients

Coefficients are log odds-ratios. These are hard to interpret. We can at least convert them into odds-ratios by exponentiating:

```
exp(coef(model))
## (Intercept)      ft_dem      ft_rep       black    hispanic
##   0.7996197   1.0983245   0.9131759  23.7079342   2.6379315
##       other      income
##   1.7482726   0.9752484
```

For example, one additional point on the feeling thermometer for Democrats increases the odds of voting for Obama by 9 per cent.

Interpretation of Logit Coefficients

Coefficients are log odds-ratios. These are hard to interpret. We can at least convert them into odds-ratios by exponentiating:

```
exp(coef(model))
## (Intercept)      ft_dem      ft_rep       black    hispanic
##   0.7996197   1.0983245   0.9131759  23.7079342   2.6379315
##       other      income
##   1.7482726   0.9752484
```

For example, one additional point on the feeling thermometer for Democrats increases the odds of voting for Obama by 9 per cent.

Or being hispanic (as opposed to white) increases the odds of voting for Obama by $(2.64 - 1) \cdot 100 = 164$ per cent.

Interpretation of Logit Coefficients

Coefficients are log odds-ratios. These are hard to interpret. We can at least convert them into odds-ratios by exponentiating:

```
exp(coef(model))
## (Intercept)      ft_dem      ft_rep       black    hispanic
##   0.7996197   1.0983245   0.9131759  23.7079342   2.6379315
##       other      income
##   1.7482726   0.9752484
```

For example, one additional point on the feeling thermometer for Democrats increases the odds of voting for Obama by 9 per cent.

Or being hispanic (as opposed to white) increases the odds of voting for Obama by $(2.64 - 1) \cdot 100 = 164$ per cent.

These are relative statements. Do not confuse with probabilities. They depend on the values of all other variables. We need predicted probabilities to interpret the effects more clearly.

Predicted Probabilities for Logit Models

We could predict the probability of voting for Obama given a specific ethnicity and given mean values on all other variables.

Predicted Probabilities for Logit Models

We could predict the probability of voting for Obama given a specific ethnicity and given mean values on all other variables.

For this, we first need a data frame with this profile of values:

Predicted Probabilities for Logit Models

We could predict the probability of voting for Obama given a specific ethnicity and given mean values on all other variables.

For this, we first need a data frame with this profile of values:

```
newdata <- data.frame(ft_dem = mean(ANES$ft_dem, na.rm = TRUE),
                      ft_rep = mean(ANES$ft_rep, na.rm = TRUE),
                      black = 0,
                      hispanic = 0,
                      other = 0,
                      income = mean(ANES$income, na.rm = TRUE))
newdata
##     ft_dem   ft_rep black hispanic other   income
## 1 54.72016 42.04109     0        0     0 14.55995
```

Predicted Probabilities for Logit Models

We could predict the probability of voting for Obama given a specific ethnicity and given mean values on all other variables.

For this, we first need a data frame with this profile of values:

```
newdata <- data.frame(ft_dem = mean(ANES$ft_dem, na.rm = TRUE),
                      ft_rep = mean(ANES$ft_rep, na.rm = TRUE),
                      black = 0,
                      hispanic = 0,
                      other = 0,
                      income = mean(ANES$income, na.rm = TRUE))
newdata
##     ft_dem   ft_rep black hispanic other   income
## 1 54.72016 42.04109     0        0     0 14.55995
```

We basically create scenarios that we can compare to each other. `newdata` serves as our template. . .

Predicted Probabilities for Logit Models

Based on this template, we can create four scenarios for the different ethnicities:

Predicted Probabilities for Logit Models

Based on this template, we can create four scenarios for the different ethnicities:

```
newdata <- newdata[rep(1, 4), ]
newdata$black[2] <- 1
newdata$hispanic[3] <- 1
newdata$other[4] <- 1
rownames(newdata) <- c("white", "black", "hispanic", "other")
newdata
##             ft_dem   ft_rep black hispanic other   income
## white    54.72016 42.04109     0        0     0 14.55995
## black    54.72016 42.04109     1        0     0 14.55995
## hispanic 54.72016 42.04109     0        1     0 14.55995
## other    54.72016 42.04109     0        0     1 14.55995
```

Predicted Probabilities for Logit Models

Based on this template, we can create four scenarios for the different ethnicities:

```
newdata <- newdata[rep(1, 4), ]
newdata$black[2] <- 1
newdata$hispanic[3] <- 1
newdata$other[4] <- 1
rownames(newdata) <- c("white", "black", "hispanic", "other")
newdata
##             ft_dem   ft_rep black hispanic other   income
## white    54.72016 42.04109     0        0     0 14.55995
## black    54.72016 42.04109     1        0     0 14.55995
## hispanic 54.72016 42.04109     0        1     0 14.55995
## other    54.72016 42.04109     0        0     1 14.55995
```

Based on this, we can now predict the probabilities.

Predicted Probabilities for Logit Models

First, let's predict the log odds:

Predicted Probabilities for Logit Models

First, let's predict the log odds:

```
logodds <- predict(model, newdata = newdata)
logodds
##     white     black  hispanic     other
## 0.7249835 3.8907932 1.6949785 1.2836117
```

Predicted Probabilities for Logit Models

First, let's predict the log odds:

```
logodds <- predict(model, newdata = newdata)
logodds
##     white     black  hispanic     other
## 0.7249835 3.8907932 1.6949785 1.2836117
```

Next, let's predict probabilities:

Predicted Probabilities for Logit Models

First, let's predict the log odds:

```
logodds <- predict(model, newdata = newdata)
logodds
##     white     black  hispanic     other
## 0.7249835 3.8907932 1.6949785 1.2836117
```

Next, let's predict probabilities:

```
prob <- predict(model, newdata = newdata, type = "response")
prob
##     white     black  hispanic     other
## 0.6737035 0.9799799 0.8448778 0.7830639
```

Predicted Probabilities for Logit Models

First, let's predict the log odds:

```
logodds <- predict(model, newdata = newdata)
logodds
##     white     black  hispanic     other
## 0.7249835 3.8907932 1.6949785 1.2836117
```

Next, let's predict probabilities:

```
prob <- predict(model, newdata = newdata, type = "response")
prob
##     white     black  hispanic     other
## 0.6737035 0.9799799 0.8448778 0.7830639
```

We can also convert the log odds to probabilities manually via the logit transformation:

Predicted Probabilities for Logit Models

First, let's predict the log odds:

```
logodds <- predict(model, newdata = newdata)
logodds
##     white     black  hispanic     other
## 0.7249835 3.8907932 1.6949785 1.2836117
```

Next, let's predict probabilities:

```
prob <- predict(model, newdata = newdata, type = "response")
prob
##     white     black  hispanic     other
## 0.6737035 0.9799799 0.8448778 0.7830639
```

We can also convert the log odds to probabilities manually via the logit transformation:

```
exp(logodds) / (1 + exp(logodds))
##     white     black  hispanic     other
## 0.6737035 0.9799799 0.8448778 0.7830639
```

Predicted Probabilities for Logit Models

First, let's predict the log odds:

```
logodds <- predict(model, newdata = newdata)
logodds
##     white     black  hispanic     other
## 0.7249835 3.8907932 1.6949785 1.2836117
```

Next, let's predict probabilities:

```
prob <- predict(model, newdata = newdata, type = "response")
prob
##     white     black  hispanic     other
## 0.6737035 0.9799799 0.8448778 0.7830639
```

We can also convert the log odds to probabilities manually via the logit transformation:

```
exp(logodds) / (1 + exp(logodds))
##     white     black  hispanic     other
## 0.6737035 0.9799799 0.8448778 0.7830639
```

Black people have a 98 per cent probability of voting for Obama (at average income etc levels). 20 per cent higher than whites.

Confidence Intervals for the Predictions

```
pred <- predict(model, newdata = newdata, type = "response",
                se.fit = TRUE)
crit <- qnorm(1 - (0.05 / 2))
crit
## [1] 1.959964

data.frame(prediction = pred$fit,
           upper_ci = pred$fit + crit * pred$se.fit,
           lower_ci = pred$fit - crit * pred$se.fit)
##           prediction  upper_ci  lower_ci
## white      0.6737035 0.7106951 0.6367118
## black      0.9799799 0.9950438 0.9649159
## hispanic   0.8448778 0.8905384 0.7992171
## other      0.7830639 0.8649926 0.7011353
```

Confidence Intervals for the Predictions

```
pred <- predict(model, newdata = newdata, type = "response",
                se.fit = TRUE)
crit <- qnorm(1 - (0.05 / 2))
crit
## [1] 1.959964

data.frame(prediction = pred$fit,
           upper_ci = pred$fit + crit * pred$se.fit,
           lower_ci = pred$fit - crit * pred$se.fit)
##           prediction  upper_ci  lower_ci
## white      0.6737035 0.7106951 0.6367118
## black      0.9799799 0.9950438 0.9649159
## hispanic   0.8448778 0.8905384 0.7992171
## other      0.7830639 0.8649926 0.7011353
```

Note how these point predictions are useful for categorical independent variables. Next, we will look at the continuous case. . .

Predictions for the Democratic Feeling Thermometer

Create scenarios with varying thermometer levels and all other variables fixed at their means, then predict:

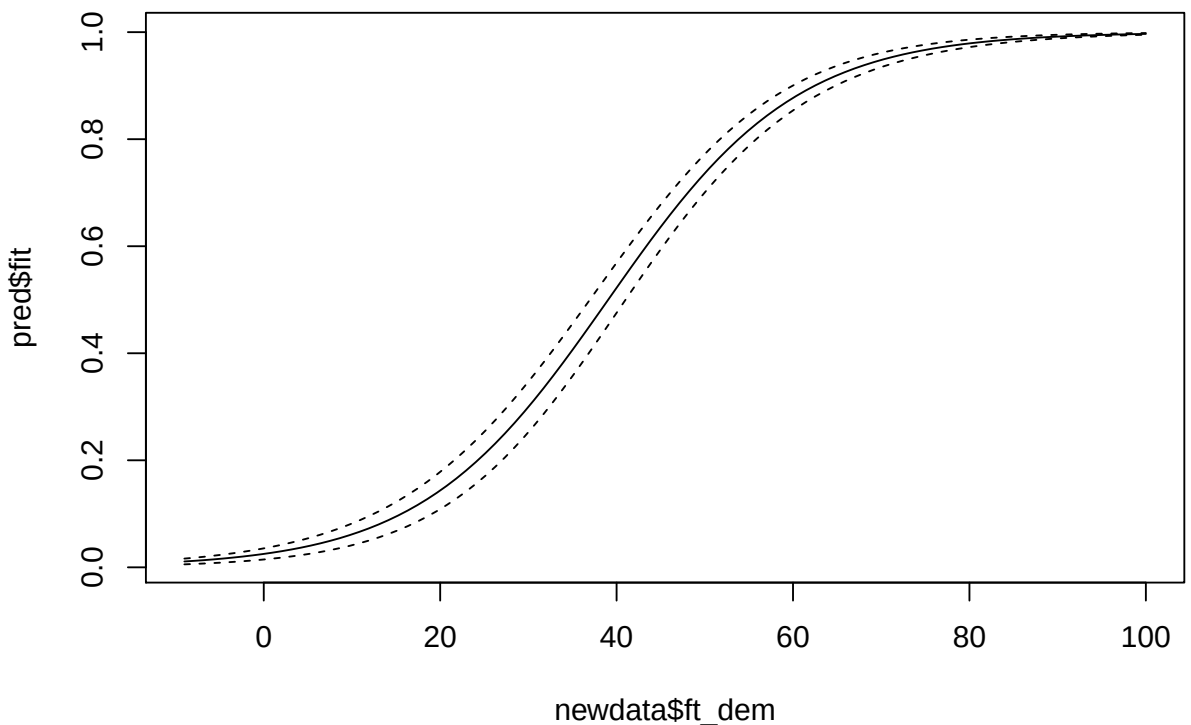
```
dtherm <- seq(from = min(ANES$ft_dem), to = max(ANES$ft_dem),
              length.out = 100)
newdata <- data.frame(ft_dem = dtherm,
                      ft_rep = mean(ANES$ft_rep, na.rm = TRUE),
                      black = mean(ANES$black, na.rm = TRUE),
                      hispanic = mean(ANES$hispanic, na.rm = TRUE),
                      other = mean(ANES$other, na.rm = TRUE),
                      income = mean(ANES$income, na.rm = TRUE))
tail(newdata)
##        ft_dem   ft_rep    black  hispanic      other   income
## 95   94.49495 42.04109 0.181137 0.1457364 0.05400517 14.55995
## 96   95.59596 42.04109 0.181137 0.1457364 0.05400517 14.55995
## 97   96.69697 42.04109 0.181137 0.1457364 0.05400517 14.55995
## 98   97.79798 42.04109 0.181137 0.1457364 0.05400517 14.55995
## 99   98.89899 42.04109 0.181137 0.1457364 0.05400517 14.55995
## 100 100.00000 42.04109 0.181137 0.1457364 0.05400517 14.55995

pred <- predict(model, newdata = newdata, type = "response",
                se.fit = TRUE)
```

Predictions for the Democratic Feeling Thermometer

Plot the predicted probabilities with a 95 per cent CI:

```
plot(newdata$ft_dem, pred$fit, type = "l")
lines(newdata$ft_dem, pred$fit + crit * pred$se.fit,
      lty = "dashed")
lines(newdata$ft_dem, pred$fit - crit * pred$se.fit,
      lty = "dashed")
```



We can see how the IV is non-linearly related to the probability of voting for Obama, all things being equal.

Predictions for the Democratic Feeling Thermometer

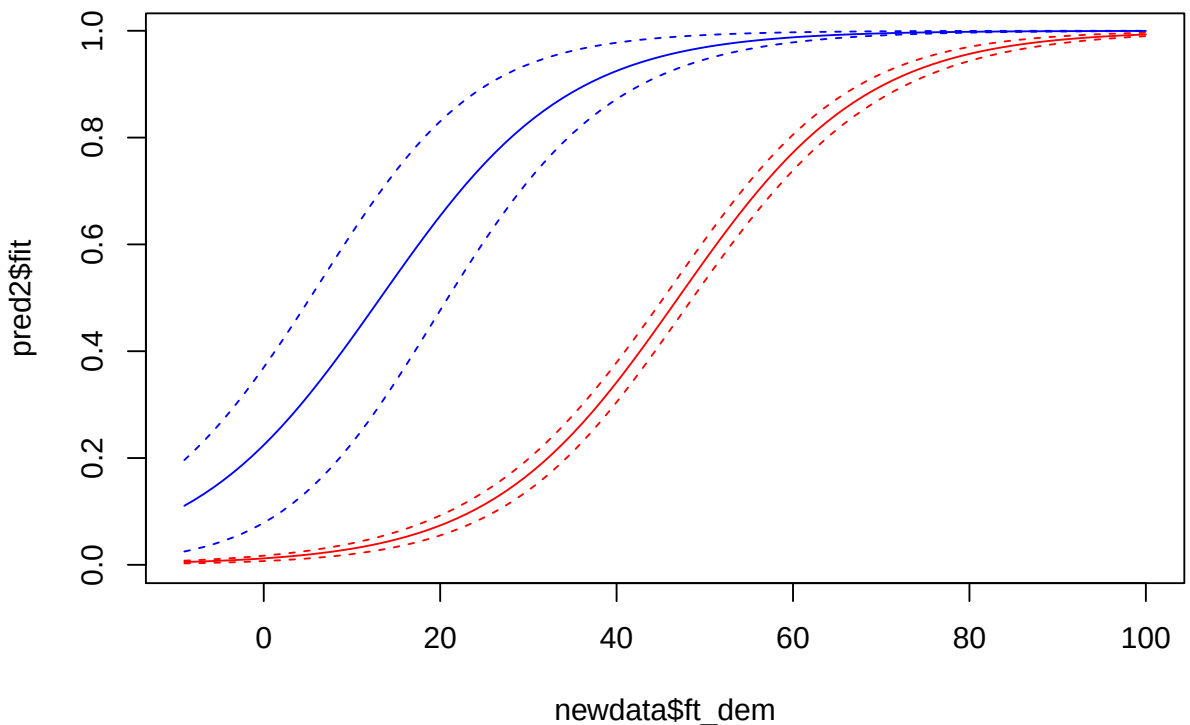
Create thermometer scenarios conditional on ethnicities:

```
newdata$black <- 0
newdata$hispanic <- 0
newdata$other <- 0
newdata$white <- 1
pred2 <- predict(model, newdata = newdata, type = "response",
                 se.fit = TRUE)

newdata$black <- 1
newdata$white <- 0
pred3 <- predict(model, newdata = newdata, type = "response",
                 se.fit = TRUE)
```

Predictions for the Democratic Feeling Thermometer

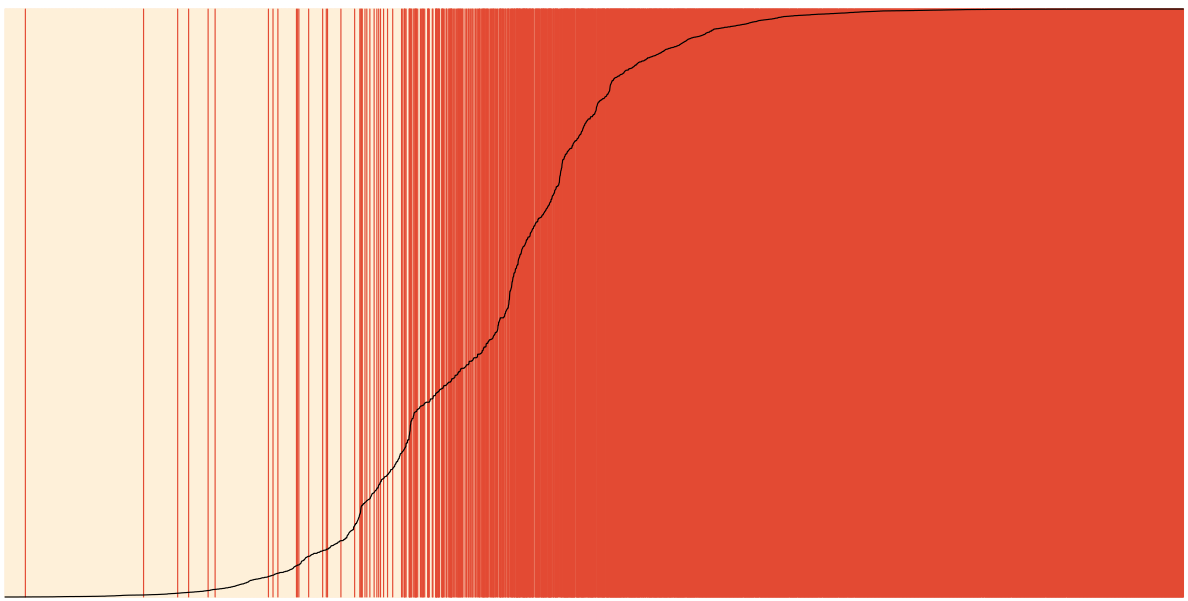
```
plot(newdata$ft_dem, pred2$fit, type = "l", col = "red")
lines(newdata$ft_dem, pred2$fit + crit * pred2$se.fit,
      lty = "dashed", col = "red")
lines(newdata$ft_dem, pred2$fit - crit * pred2$se.fit,
      lty = "dashed", col = "red")
lines(newdata$ft_dem, pred3$fit, type = "l", col = "blue")
lines(newdata$ft_dem, pred3$fit + crit * pred3$se.fit,
      lty = "dashed", col = "blue")
lines(newdata$ft_dem, pred3$fit - crit * pred3$se.fit,
      lty = "dashed", col = "blue")
```



5. Model Fit

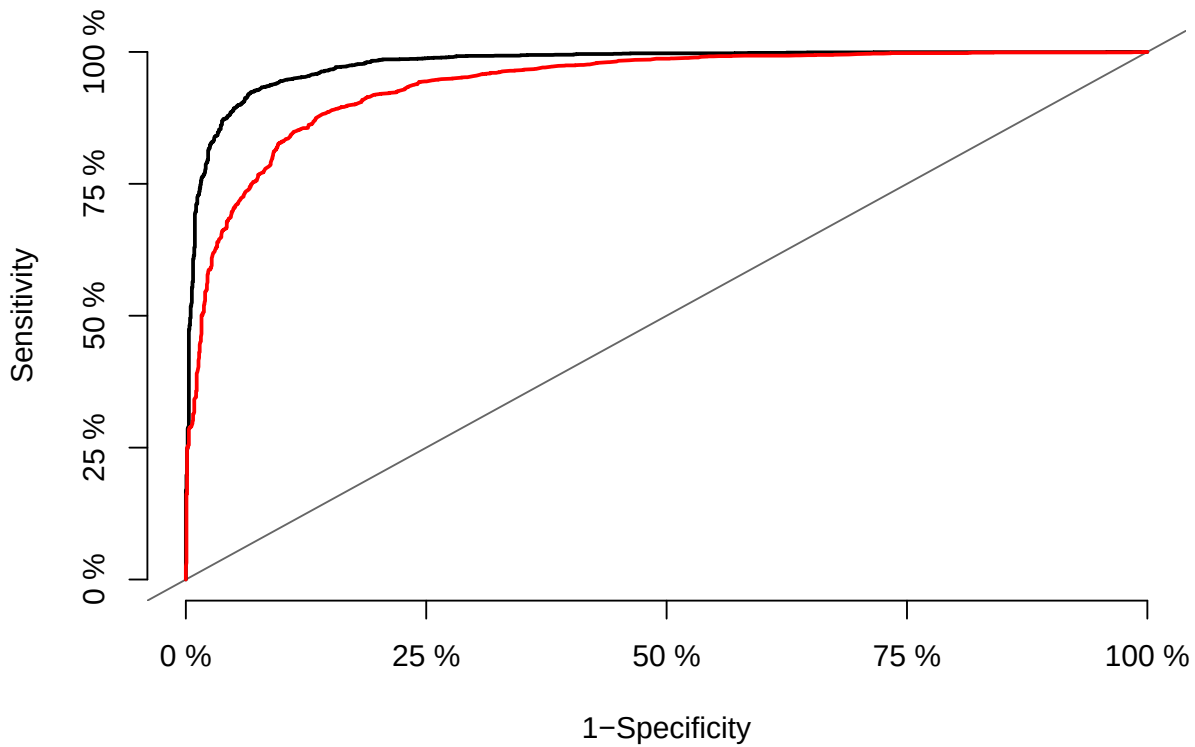
Separation Plots

```
library("separationplot")
separationplot(pred = model$fitted.values,
               actual = model$y,
               line = TRUE,
               newplot = FALSE)
```



Receiver Operating Characteristics (ROC)

```
library("ModelGood")
model3 <- glm(vote_obama ~ ft_rep + black + hispanic +
                other + income, family = binomial,
                data = ANES)  # leaving out ft_dem
plot(Roc(model))
lines(Roc(model3), col = "red")
```



ROC Interpretation

ROC Interpretation

- Sensitivity: True positive rate. How well do we predict actual ones as ones?

ROC Interpretation

- Sensitivity: True positive rate. How well do we predict actual ones as ones?
- 1– Specificity: False positive rate. How many zeroes do we predict as ones?

ROC Interpretation

- Sensitivity: True positive rate. How well do we predict actual ones as ones?
- 1– Specificity: False positive rate. How many zeroes do we predict as ones?
- We clearly want to be in the upper left corner of the plot with our model.

ROC Interpretation

- Sensitivity: True positive rate. How well do we predict actual ones as ones?
- 1− Specificity: False positive rate. How many zeroes do we predict as ones?
- We clearly want to be in the upper left corner of the plot with our model.
- The ROC curve shows the trade-off between the two criteria at each level, for many simulations from the estimated model.

ROC Interpretation

- Sensitivity: True positive rate. How well do we predict actual ones as ones?
- 1− Specificity: False positive rate. How many zeroes do we predict as ones?
- We clearly want to be in the upper left corner of the plot with our model.
- The ROC curve shows the trade-off between the two criteria at each level, for many simulations from the estimated model.
- Simulations were generated by incorporating random fluctuation from the VCOV matrix.

ROC Interpretation

- Sensitivity: True positive rate. How well do we predict actual ones as ones?
- 1− Specificity: False positive rate. How many zeroes do we predict as ones?
- We clearly want to be in the upper left corner of the plot with our model.
- The ROC curve shows the trade-off between the two criteria at each level, for many simulations from the estimated model.
- Simulations were generated by incorporating random fluctuation from the VCOV matrix.
- ROC can be used for model comparison.

Area Under the Curve (AUC)

The area under the curve is an overall measure of model fit.

```
Roc(model)
## Receiver operating characteristic
## Sample size:  3870
##
## Response:  '0' (n=1598) '1' (n=2272)
##
## Area under the ROC curve (AUC, higher better):
## full data
##     97.76
##
## Brier score (Brier, lower better):
## full data
##      5.43
```

Area Under the Curve (AUC)

The area under the curve is an overall measure of model fit.

```
Roc(model)
## Receiver operating characteristic
## Sample size:  3870
##
## Response:  '0' (n=1598) '1' (n=2272)
##
## Area under the ROC curve (AUC, higher better):
## full data
##     97.76
##
## Brier score (Brier, lower better):
## full data
##      5.43
```

We can also use this to compare models:

```
Roc(model)$Auc$glm$Auc - Roc(model3)$Auc$glm$Auc
## [1] 0.03816528
```