R for actuaries: Generalized Linear Models in R

Jean Rea
The views expressed in this presentation are those of the presenter(s) and not necessarily of the Society of Actuaries in Ireland.
Introduction

• Basic introduction to GLMs in R
• Not intended to be advanced
• Assumes some statistical knowledge and basic R knowledge
• Will work through a practical example based on the Titanic data from the kaggle competition
• Uses
Initial analysis

• Understanding of the topic / area
• Understand the problem / objectives of the analysis
• Express the problem in statistical terms
• Data quality
• Exploratory data analysis e.g. numerical and graphical summaries
Linear regression

• Linear regression refresh

• Linear relationship between x (explanatory variable) and y (dependent variable)

\[ y_i = \alpha + \beta \cdot x_i + \varepsilon_i \]

• Y is the value of the dependent variable, based on 2 components
  – Non random / structural component \( \alpha + \beta \cdot x_i \)
  – Random component / error term

• Parameter estimation is based on minimising the prediction error / residual sum of squares
GLM introduction

- GLMs are a flexible generalization of ordinary linear models.
- GLMs allow for response distributions other than Normal.
- It allows for non-linearity in the model structure by allowing the linear model to be related to the response variable via a link function.
- Applies to data from an exponential family distribution (Normal, Poisson, Gamma, Binomial...).
GLM introduction

• Response variable in a glm can have any distribution from an exponential family
• Form of the exponential family:

\[ f_\theta(y) = \exp \left[ \frac{y\theta - b(\theta)}{a(\phi)} \right] + c(y, \phi) \]

• a, b and c are arbitrary functions, \( \phi \) is a scale parameter
• Normal, Binomial, Poisson, Gamma...
Exponential family – binomial example

\[ P(Y = y) = \binom{n}{y} p^y (1-p)^{n-y} \]

\[ = \exp \left\{ \log \left[ \binom{n}{y} p^y (1-p)^{n-y} \right] \right\} \]

\[ = \exp \left\{ \log \left( \binom{n}{y} \right) + y \log p + (n-y) \log (1-p) \right\} \]

\[ = \exp \left\{ \log \left( \binom{n}{y} \right) + y \log p - y \log (1-p) + n \log (1-p) \right\} \]

\[ = \exp \left\{ \log \left( \binom{n}{y} \right) + y \log \frac{p}{1-p} + n \log (1-p) \right\} \]

\[ = \exp \left\{ y \log \frac{p}{1-p} + n \log (1-p) + \log \binom{n}{y} \right\} \]

\[ = \exp \left\{ \frac{y \log \frac{p}{1-p} - n \log \frac{1}{1-p}}{1} + \log \binom{n}{y} \right\} \]

\[ f(y) = \exp \left[ \frac{y \theta - b(\theta)}{a(\phi)} + c(y, \phi) \right] \]

- \( \theta = \log \frac{p}{1-p} = \log \frac{np}{n-np} = \log \frac{\mu}{n-\mu} = g(\mu) \)
- \( b(\theta) = n \log \frac{1}{1-p} = n \log (1 + \exp(\theta)) \)
- \( a(\phi) = 1 \)
- \( c(y, \phi) = \log \binom{n}{y} \)

Fitting GLMs in R

• `?glm`

**Fitting Generalized Linear Models**

**Description**

`glm` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

**Usage**

```r
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)
```

```r
glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)
```

```r
## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

**Basic formula** - `glm(formula, family=family(link=linkfunction), data=)`
Fitting GLMs in R

• **family**
  
  ```
  binomial(link = "logit")
  gaussian(link = "identity")
  Gamma(link = "inverse")
  inverse.gaussian(link = "1/mu^2")
  poisson(link = "log")
  quasi(link = "identity", variance = "constant")
  quasibinomial(link = "logit")
  quasipoisson(link = "log")
  ```

• **Binomial** – logistic (binary) regression / response number of successes from known number of trials

• **Gamma** – strictly positive real valued data

• **Poisson** – count data
Example: Titanic kaggle competition

- [https://www.kaggle.com/c/titanic](https://www.kaggle.com/c/titanic)

**Competition Description**

- The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

- One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

- In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.
# Example: Titanic kaggle competition

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Level (if applicable)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>survival</td>
<td>Survival</td>
<td>0 = No, 1 = Yes</td>
<td></td>
</tr>
<tr>
<td>pclass</td>
<td>Ticket class</td>
<td>1 = 1st, 2 = 2nd, 3 = 3rd</td>
<td>Proxy for socio-economic status 1\textsuperscript{st} = upper etc.</td>
</tr>
<tr>
<td>sex</td>
<td>Sex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>Age in years</td>
<td></td>
<td>Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5</td>
</tr>
<tr>
<td>sibsp</td>
<td># of siblings / spouses aboard the Titanic</td>
<td></td>
<td>Sibling = brother, sister, stepbrother, stepsister Spouse = husband, wife (mistresses and fiancés were ignored)</td>
</tr>
<tr>
<td>parch</td>
<td># of parents / children aboard the Titanic</td>
<td></td>
<td>Parent = mother, father Child = daughter, son, stepdaughter, stepson Some children travelled only with a nanny, therefore parch=0 for them</td>
</tr>
<tr>
<td>ticket</td>
<td>Ticket number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fare</td>
<td>Passenger fare</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cabin</td>
<td>Cabin number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>embarked</td>
<td>Port of Embarkation</td>
<td>C, Q, S</td>
<td></td>
</tr>
</tbody>
</table>
Example: Titanic kaggle competition

• Will broadly follow the analysis below
• This analysis is based on 7 steps
  1. Load and clean data
  2. Create data frame of variables
  3. Check for multicollinearity
  4. Build a logistic regression model
  5. Revise model
  6. Test accuracy of model on training data – not going to do this part
  7. Use model to predict survivability for test data
R Studio

```r
# Step 5: use model to predict survival for test data
predict(test) = predict(TitanicLog1, type = "response", newdata = Test)

# no preference over error t = 0.5
TestSurvived = as.numeric(predictTest >= 0.5)

table(TestSurvived)

> par(mfrow=c(2,2))
> plot(TitanicLog1)
> par(mfrow=c(1,1))

# Step 6: use model to predict survival for test data
predict(test) = predict(TitanicLog1, type = "response", newdata = Test)

# no preference over error t = 0.5
TestSurvived = as.numeric(predictTest >= 0.5)

table(TestSurvived)
```

```
> predict(test)
```

```
> table(TestSurvived)
```

```
> par(mfrow=c(2,2))
> plot(TitanicLog1)
> par(mfrow=c(1,1))
```
Step 1: Load and clean data

- Download csv files from website and save in your own directory
- Set working directory in R: Session ➔ Set working directory ➔ ...
- Then import the data into R, number of calls exist
  - `?read.csv`
  - `read.csv(file, header = TRUE, sep = ",", quote = "\\", dec = ".", fill = TRUE, comment.char = ",", ...)`
- File – filename.
- Header - a logical value indicating whether the file contains the names of the variables as its first line.
- Sep - the field separator character. Values on each line of the file are separated by this character.
Step 1: Load and clean data

What it looks like in Excel:

<table>
<thead>
<tr>
<th>Passenger</th>
<th>Survived</th>
<th>Pclass</th>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>SibSp</th>
<th>Parch</th>
<th>Ticket</th>
<th>Fare</th>
<th>Cabin</th>
<th>Embarked</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>Braund, M</td>
<td>male</td>
<td>22</td>
<td>1</td>
<td>0</td>
<td>A/5 21171</td>
<td>7.25</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Cumings, I</td>
<td>female</td>
<td>38</td>
<td>1</td>
<td>0</td>
<td>PC 17599</td>
<td>71.2833</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>Heikkinen, female</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>STON/O2.</td>
<td>7.925</td>
<td>S</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>Futrelle, M</td>
<td>female</td>
<td>35</td>
<td>1</td>
<td>0</td>
<td>113803</td>
<td>53.1</td>
<td>C123</td>
<td>S</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>3</td>
<td>Allen, Mr.</td>
<td>male</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>373450</td>
<td>8.05</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>3</td>
<td>Moran, M</td>
<td>male</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>330877</td>
<td>8.4583</td>
<td>Q</td>
<td>0</td>
</tr>
</tbody>
</table>

Importing into R:

```
Train=read.csv("train.csv",header=T,na.strings=c(""))
Test=read.csv("test.csv",header=T,na.strings=c(""))
str(Train)
str(Test)
```
Step 1: Load and clean data

- Data frame is a way of storing data – like a matrix except columns can have different data types
- Factors store categorical variables in R

```r
> str(Train)
'data.frame':  891 obs. of  13 variables:
$ PassengerId: int  1  2  3  4  5  6  7  8  9  10 ...  
$ Survived: int  0  1  1  1  0  0  0  0  1  1 ...  
$ Pclass: int  3  1  3  1  3  1  3  1  3  2 ...  
$ Name: Factor w/  891 levels "Abbing, Mr. Anthony",...:  109  191  358  277  16  559  520  629  41  7  581 ...  
$ Sex: Factor w/  2 levels "female","male":  2  1  1  1  2  2  2  2  1  1 ...  
$ Age: num  22  38  26  35  35 NA  54  2  27  14 ...  
$ SibSp: int  1  1  0  1  0  0  0  3  0  1 ...  
$ Parch: int  0  0  0  0  0  0  0  1  2  0 ...  
$ Ticket: Factor w/  681 levels "110152","110413",...:  524  597  670  50  473  276  86  396  345  133 ...  
$ Fare: num  7.25  71.28  7.92  53.1  8.05 ...  
$ Cabin: Factor w/  147 levels "A10","A14","A16",...: NA  82  NA  56  NA  NA  130  NA  NA NA ...  
$ Embarked: Factor w/  3 levels "C","Q","S":  3  1  3  3  3  2  3  3  3  1 ...  
$ CabinInd: int  0  1  0  1  0  0  0  1  0  0 ...  
```
apply(Train,2,function(x) sum(is.na(x)))

> apply(Train,2,function(x) sum(is.na(x)))

<table>
<thead>
<tr>
<th>PassengerId</th>
<th>Survived</th>
<th>Pclass</th>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>SibSp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>177</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>687</td>
<td>2</td>
</tr>
</tbody>
</table>

• Lots of missing ages and cabins, also 2 observations with missing Embarked

#lots of missing age values for each - replace with mean

Train$Age[is.na(Train$Age)] = mean(Train$Age,na.rm=T)

• Also deal with other missing observations – I have removed them
Step 1: Load and clean data

```r
summary(Train)
```

<table>
<thead>
<tr>
<th>PassengerId</th>
<th>Survived</th>
<th>Pclass</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 1</td>
<td>Min. : 0.0000</td>
<td>Min. : 1.0000</td>
<td>Abbing, Mr. Anthony : 1</td>
</tr>
<tr>
<td>1st Qu.: 224</td>
<td>1st Qu. : 0.0000</td>
<td>1st Qu. : 2.0000</td>
<td>Abbott, Mr. Rossmore Edward : 1</td>
</tr>
<tr>
<td>Median : 446</td>
<td>Median : 0.0000</td>
<td>Median : 3.0000</td>
<td>Abbott, Mrs. Stanton (Rosa Hunt) : 1</td>
</tr>
<tr>
<td>Mean : 446</td>
<td>Mean : 0.3825</td>
<td>Mean : 2.312</td>
<td>Abelson, Mr. Samuel : 1</td>
</tr>
<tr>
<td>3rd Qu.: 668</td>
<td>3rd Qu. : 1.0000</td>
<td>3rd Qu. : 3.0000</td>
<td>Abelson, Mrs. Samuel (Hannah Wizosky) : 1</td>
</tr>
<tr>
<td>Max. : 891</td>
<td>Max. : 1.0000</td>
<td>Max. : 3.0000</td>
<td>Adahl, Mr. Mauritz Nils Martin : 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>SibSp</th>
<th>Parch</th>
<th>Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>female: 312</td>
<td>Min. : 0.42</td>
<td>Min. : 0.0000</td>
<td>Min. : 0.0000</td>
<td>1601 : 7</td>
</tr>
<tr>
<td>male : 577</td>
<td>1st Qu. : 22.00</td>
<td>1st Qu. : 0.0000</td>
<td>1st Qu. : 0.0000</td>
<td>347082 : 7</td>
</tr>
<tr>
<td>Median : 29.70</td>
<td>Median : 0.0000</td>
<td>Median : 0.0000</td>
<td>CA. 2343 : 7</td>
<td></td>
</tr>
<tr>
<td>Mean : 29.65</td>
<td>Mean : 0.5242</td>
<td>Mean : 0.3825</td>
<td>3101295 : 6</td>
<td></td>
</tr>
<tr>
<td>3rd Qu. : 35.00</td>
<td>3rd Qu. : 1.0000</td>
<td>3rd Qu. : 0.0000</td>
<td>347088 : 6</td>
<td></td>
</tr>
<tr>
<td>Max. : 80.00</td>
<td>Max. : 8.0000</td>
<td>Max. : 6.0000</td>
<td>CA 2144 : 6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fare</th>
<th>Cabin</th>
<th>Embarked</th>
<th>CabinInd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 0.0000</td>
<td>B96, B98 : 4</td>
<td>C : 168</td>
<td>Min. : 0.0000</td>
</tr>
<tr>
<td>1st Qu. : 7.896</td>
<td>C23, C25, C27 : 4</td>
<td>Q : 77</td>
<td>1st Qu. : 0.0000</td>
</tr>
<tr>
<td>Median : 14.454</td>
<td>G6 : 4</td>
<td>S : 644</td>
<td>Median : 0.0000</td>
</tr>
<tr>
<td>Mean : 32.097</td>
<td>C22, C26 : 3</td>
<td>Mean : 0.2272</td>
<td></td>
</tr>
<tr>
<td>3rd Qu. : 31.000</td>
<td>D : 3</td>
<td>3rd Qu. : 0.0000</td>
<td></td>
</tr>
<tr>
<td>Max. : 512.329</td>
<td>(Other) : 184</td>
<td>Max. : 1.0000</td>
<td></td>
</tr>
</tbody>
</table>

NA's : 687
Step 1: Load and clean data

Histograms
Step 1: Load and clean data

Boxplots
Step 1: Load and clean data

Also consider categorical variables

```r
> table(Train$Sex, Train$Survived)

    0  1
female 81  231
male   468 109

> prop.table(table(Train$Sex, Train$Survived), 1)

     0      1
female 0.2596154 0.7403846
male   0.8110919 0.1889081

> prop.table(table(Train$Sex, Train$Survived), 2)

       0      1
female 0.1475410 0.6794118
male   0.8524590 0.3205882
```
Step 2: Create data frame of variables

# Step 2: Create DF of independent/dependent variables

nonvars = c("PassengerId","Name","Ticket","Embarked","Cabin","CabinInd")
Train = Train[,!(names(Train) %in% nonvars)]

str(Train)

> str(Train)
'data.frame': 889 obs. of 7 variables:
$ Survived: int 0 1 1 1 0 0 0 0 1 1 ...
$ Pclass : int 3 1 3 1 3 3 1 3 3 2 ...
$ Sex   : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
$ Age   : num 22 38 26 35 35 ...
$ SibSp : int 1 0 1 0 0 0 3 0 1 ...
$ Parch : int 0 0 0 0 0 0 0 1 2 0 ...
$ Fare  : num 7.25 71.28 7.92 53.1 8.05 ...

Train$Pclass = as.factor(Train$Pclass)
Step 3: Check for MultiCollinearity
Step 4: Build a logistic regression model

# Step 4: Build a Logistic Regression Model

TitanicLog1 = glm(Survived~., data = Train, family = binomimal(link=logit))

summary(TitanicLog1)
Step 4: Build a logistic regression model

Call:
glm(formula = survived ~ ., family = binomial(link = logit),
data = Train)

Deviance Residuals:
       Min          1Q        Median          3Q         Max
-2.7/055 -0.6098 -0.42/1  0.61/4  2.41/88

Coefficients:
       Estimate Std. Error  z value Pr(>|z|)
(Intercept) 3.836254   0.446497   8.592   < 2e-16 ***
Pclass2 -1.017868   0.293867  -3.464  0.000533 ***
Pclass3 -2.144843   0.289561  -7.407  1.29e-13 ***
Sexmale -2.753512   0.199454 -13.805  < 2e-16 ***
Age -0.039687   0.007858  -5.051  4.40e-07 ***
SibSp -0.349212   0.109498  -3.189  0.001427 **
Parch -0.111842   0.117598  -0.951   0.341579
Fare  0.002969   0.002441   1.216   0.223854
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1182.82  on 888  degrees of freedom
  Residual deviance:  788.16  on 881  degrees of freedom
AIC: 804.16

Number of Fisher Scoring iterations: 5

. means to include first order terms of all variables

Other options:
0 to exclude the intercept
- to exclude terms
: to include interactions

Alternative is to explicitly state them in the formula:

```R
glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare, family = binomial, data = Train)
```
Step 4: Build a logistic regression model

Call: glm(formula = survived ~ ., family = binomial(link = logit),
       data = Train)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-2.7055  -0.6098  -0.4271  0.6147  2.4188

Coefficients:                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                      3.836254   0.446497   8.592  < 2e-16 ***
Pclass2                           -1.017868   0.293867  -3.464  0.000533 ***
Pclass3                           -2.144843   0.289561  -7.301  1.29e-13 ***
Sexmale                           -2.753512   0.199454 -13.805  < 2e-16 ***
Age                              -0.039687   0.007858  -5.051  4.40e-07 ***
SibSp                            -0.349212   0.109498  -3.189  0.001427 **
Parch                            -0.111842   0.117598  -0.951  0.341579
Fare                             0.002969   0.002441   1.216  0.223854
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1182.82  on 888 degrees of freedom
Residual deviance: 788.16  on 881 degrees of freedom
AIC: 804.16

Number of Fisher Scoring iterations: 5
Step 4: Build a logistic regression model

Call:
glm(formula = survived ~ ., family = binomial(link = logit),
     data = Train)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-2.7055  -0.6098  -0.4271   0.6147   2.4188

Coefficients:  Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.836254   0.446497  8.592  < 2e-16 ***
Pclass2       -1.017868   0.293867  -3.464  0.000533 ***
Pclass3       -2.144843   0.289561  -7.407  1.29e-13 ***
Sexmale       -2.753512   0.199454 -13.805  < 2e-16 ***
Age           -0.039687   0.007858  -5.051  4.40e-07 ***
SibSp         -0.349210   0.109498  -3.189  0.001427 **
Parch         -0.111842   0.117598  -0.951  0.341579
Fare          0.002969   0.002441  1.216   0.223854
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1182.82 on 888 degrees of freedom
Residual deviance:  788.16 on 881 degrees of freedom
AIC: 804.16

Number of Fisher Scoring iterations: 5

Note factors / categorical variables are relative to a baseline
**Step 4: Build a logistic regression model**

```r
call: glm(formula = survived ~ ., family = binomial(link = logit), data = Train)
```

<table>
<thead>
<tr>
<th>Deviance Residuals:</th>
<th>Min 1Q Median 3Q Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-2.7055 -0.6098 -0.4271 0.6147 2.4188</td>
</tr>
</tbody>
</table>

| Coefficients: | Estimate  | Std. Error | z value | Pr(>|z|) |
|---------------|-----------|------------|---------|----------|
| (Intercept)   | 3.836254  | 0.446497   | 8.592   | < 2e-16 *** |
| Pclass2       | -1.017868 | 0.293867   | -3.464  | 0.000533 *** |
| Pclass3       | -2.144843 | 0.289561   | -7.407  | 1.29e-13 *** |
| Sexmale       | -2.753512 | 0.199454   | -13.805 | < 2e-16 *** |
| Age           | -0.039687 | 0.007858   | -5.051  | 4.40e-07 *** |
| SibSp         | -0.349212 | 0.109498   | -3.189  | 0.001427 ** |
| Parch         | -0.111842 | 0.117598   | -0.951  | 0.341579 |
| Fare          | 0.002969  | 0.002441   | 1.216   | 0.223854 |

**Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1**

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1182.82 on 888 degrees of freedom
Residual deviance: 788.16 on 881 degrees of freedom

AIC: 804.16

Number of Fisher Scoring iterations: 5

Hypothesis testing individual coefficients.

Null hypothesis is that parameter is zero.

Test statistic $z$-statistic is the estimate divided by the standard error e.g. $Pclass2 \sim \frac{-1.02}{0.29} = -3.51$

Note here scale parameter is known.

$z$-statistic is asymptotically standard normal when $H_0$ is true and the sample size is fairly large.
Step 4: Build a logistic regression model

Call: glm(formula = survived ~ ., family = binomial(link = logit), data = Train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max
  -2.7055 -0.6098 -0.4271  0.6147  2.4188

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 3.836254  0.446497  8.592  < 2e-16 ***
Pclass2     -1.017868  0.293867 -3.464  0.000533 ***
Pclass3     -2.144843  0.289561 -7.407 1.29e-13 ***
Sexmale     -2.753512  0.199454 -13.805  < 2e-16 ***
Age         -0.039687  0.007858 -5.051 4.40e-07 ***
SibSp       -0.349212  0.109498 -3.189  0.001427 **
Parch       -0.111842  0.117598 -0.951  0.341579
Fare        0.002969  0.002441  1.216  0.223854
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1182.8  on 888  degrees of freedom
Residual deviance: 788.16 on 881  degrees of freedom
AIC: 804.16

Number of Fisher Scoring iterations: 5
Step 4: Build a logistic regression model

Check on overall model fit / appropriateness – Probability of a Chi-Squared 881 random variable being as large as 788. The probability is approximately 99% which is high and supports that it does follow a Chi-Squared distribution with 881 df.

The Null deviance is the deviance for a model with just a constant term.

Residual deviance is the deviance of the fitted model.

These can be combined to give the proportion deviance explained, a generalization of R Squared, as follows:

\[
> \frac{(1182.82 - 788.16)}{1182.82} \quad [1] \\
0.3336602
\]
Step 5: Revise model

-Parch
 Means to remove Parch
Step 5: Revise model

We can do this in an automated way:

```
step(TitanicLog1, test="LRT")
```
Step 5: Revise model

- Check model diagnostics – not very informative for logistic regression

plot(TitanicLog3)
Step 7: Use model to predict survival on test data

• # Step 7: Use Model to predict survivability for Test Data
• predictTest = predict(TitanicLog3, type = "response", newdata = Test)
  • Setting type to response means that you get the predicted probabilities otherwise the default for a binomial are predictions on the logit / log odds scale
• If we had the survival indicator for the test data we could also calculate a misclassification rate
Interpretation

Logodds, odds and probabilities are variations of each other.

The estimates are on the scale of the linear predictor (logodds for binomial), can convert.

Factors need to be interpreted relative to the baseline.

For continuous it relates to a one unit increase e.g. a one unit increase in age changes the log odds of surviving by an estimated -0.04.
Other considerations

• Over dispersion
• Use of offset
  – Paper “Applications of the Offset in Property-Casualty Predictive Modeling”
Other considerations - overdispersion

• Illustrative glm call for quasibinomial - not saying it is present here

Call:
glm(formula = Survived ~ . - Parch - Fare, family = quasibinomial, data = Train)

Deviance Residuals:
    Min      1Q     Median      3Q     Max
-2.6877  -0.6028   -0.4219   0.6116   2.4523

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.021856   0.407074  9.880  < 2e-16 ***
 PClass2    -1.183245   0.266645  -4.438  1.02e-05 ***
PClass3     -2.341213   0.247292  -9.467  < 2e-16 ***
Sexmale     -2.732937   0.197860 -13.813  < 2e-16 ***
   Age      -0.040059   0.007953  -5.037  5.73e-07 ***
   SibSp    -0.357112   0.105977  -3.370   0.000785 ***

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 1.036169)

Null deviance: 1182.8 on 888 degrees of freedom
Residual deviance: 790.3 on 883 degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
Other datasets

Third party motor insurance claims in Sweden in 1977

Description
In Sweden all motor insurance companies apply identical risk arguments to classify customers, and thus their portfolios and their claims statistics can be combined. The data were compiled by a Swedish Committee on the Analysis of Risk Premium in Motor Insurance. The Committee was asked to look into the problem of analyzing the real influence on claims of the risk arguments and to compare this structure with the actual tariff.

Usage
data(motorins)

Format
A data frame with 1797 observations on the following 8 variables.

Kilometres
an ordered factor representing kilometers per year with levels 1: < 1000, 2: 1000-15000, 3: 15000-20000, 4: 20000-25000, 5: > 25000

Zone
a factor representing geographical area with levels 1: Stockholm, Goteborg, Malmo with surroundings 2: Other large cities with surroundings 3: Smaller cities with surroundings in southern Sweden 4: Rural areas in southern Sweden 5: Smaller cities with surroundings in northern Sweden 6: Rural areas in northern Sweden 7: Gotland

Bonus
No claims bonus. Equal to the number of years, plus one, since last claim

Make
A factor representing eight different common car models. All other models are combined in class 9

Intered
Number of insured in policy-years

Claims
Number of claims

Payment
Total value of payments in Skr

payment per claim
install.packages("faraway")
library(faraway)
data(motorins)
str(motorins)

> str(motorins)
'data.frame': 1797 obs. of 8 variables:
$ Kilometres: Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<...: 1 1 1 1 1 1 1 1 1 1 ...
$ Zone: Factor w/ 7 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
$ Bonus: int 1 1 1 1 1 1 1 1 1 2 ...
$ Make: Factor w/ 9 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 1 ...
$ Insured: num 455.1 69.2 72.9 1292.4 191 ...
$ Claims: int 108 19 13 124 40 57 23 14 1704 45 ...
$ Payment: int 392491 46221 15694 422201 119373 170913 56940 77487 6805992 214011 ...
$ perd: num 3634 2433 1207 3405 2984 ...
Uses

- Non Life pricing
- Non Life reserving – see paper “STOCHASTIC LOSS RESERVING USING GENERALIZED LINEAR MODELS” by Greg Taylor and Gráinne McGuire
- Life modelling
R sessions – coming soon
Questions

Thank you