Introductory Categorical Data Analysis in R

Clay Ford

Fall 2016

workshop outline

Topics on tap:

- Analysis of two-way contingency tables
- Analysis of three-way contingency tables
- Logistic regression

The focus will be on application, not statistical theory.

Categorical Data

In categorical data analysis, our response of interest (or dependent variable) is categorical.

The measurement scale of categorical data is a set of categories.

Categories can be *nominal* or *ordered*.

- Nominal example: Thai, Mexican, Italian, Chinese
- Ordinal example: mild, medium, hot, very hot

Whether your data are nominal or ordinal determines what kinds of statistical analyses you should use. Today we'll focus on nominal data with two categories.

Categorical data summaries and analyses

We typically tabulate category membership and calculate proportions.

Examples of analyses:

- compare proportions between two categories
- examine the ratio of proportions or odds between two categories
- look for an assocation between two categorical variables
- model category proportions as a function of other variables
- model category counts as a function of other variables

The big idea of categorical data analysis is to find out if the distribution of category membership is explained by or associated with other variables.

Categorical data analysis in R

Three steps:

1. get data into R
2. manipulate it (if necessary) so it's suitable for analysis and/or visualization
3. run the analysis and/or create some graphs

Example 1

Relationship between aspirin use and myocardial infarction (heart attacks) by the Physicians' Health Study Research Group at Harvard Medical School (*NEJM*, 318:262-264, 1988). Five year randomized study testing whether regular intake of aspirin reduces mortality from cardiovascular disease.

```
##           MI
## group      Yes    No
##   Placebo 189 10845
##   Aspirin 104 10933
```

Are these two categorical variables associated?

Some analysis options

1. Hypothesis test of independence between two variables (Chi-Square Test)
2. Compare proportion of MI for placebo group versus proportion of MI for aspirin group. (Difference of Proportions)
3. Examine ratio of MI proportions. (Risk Ratio or Relative Risk)
4. Examine ratio of the estimated odds of MI. (Odds Ratio)

Getting the data into R "by hand"

We can enter this manually if we like.

```
aspirin <- matrix(data = c(189, 104, 10845, 10933),
                  ncol = 2,
                  dimnames = list(group =
                                    c("Placebo","Aspirin"),
                                  MI =
                                    c("Yes","No")))
```

We now have a *contingency table* in the form of a matrix ready for analysis.

Our matrix

```
aspirin
```

```
##           MI
## group      Yes    No
##   Placebo 189 10845
##   Aspirin 104 10933
```

Notice the dimensions are labeled and the rows are in the order we specified.

Converting matrix to a table

We can convert our matrix to a `table` object, like so:

```
aspirinT <- as.table(aspirin)
aspirinT
```

```
##           MI
## group        Yes    No
##   Placebo    189 10845
##   Aspirin    104 10933
```

It doesn't make a difference for analysis purposes but does allow for some easy manipulation and visualization as we'll see in the R script.

Getting the data into R by import

Data is often stored in txt, csv or xlsx formats, or stored in another stats program. There are several ways to import such data into R.

- `read.table`
- `read.csv`
- `read_csv` in the `readr` package
- `read_excel` in the `readxl` package
- `read_spss`, `read_sas`, and `read_stata` in the `haven` package

Importing data in this manner creates a *data frame*. To form a contingency table from variables stored in a data frame, we can use the `table` or `xtabs` functions.

Example of `read.csv` for importing data

If your csv file has a header, and the csv file is in your working directory, then we simply need to give read.csv the name of the csv file.

```
aspirin.df <- read.csv("aspirin.csv")
nrow(aspirin.df) # number of rows
```

```
## [1] 22071
```

```
head(aspirin.df) # first 6 rows
```

```
##     group MI
## 1 Placebo No
## 2 Aspirin No
## 3 Placebo No
## 4 Aspirin No
## 5 Aspirin No
## 6 Aspirin No
```

A word about Factors

If we view the structure of `aspirin.df` we notice that each column is stored as a *Factor*.

```
str(aspirin.df)
```

```
## 'data.frame':    22071 obs. of  2 variables:
##  $ group: Factor w/ 2 levels "Aspirin","Placebo": 2 1 2
##  $ MI   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1
```

The `read.csv` function imports character data as Factors, which are integers with character levels. This is good for things like linear modeling (eg, logistic regression).

If we didn't want character data turned into Factors, we could have done this:
`read.csv("aspirin.csv", stringsAsFactors=FALSE)`.

The `table` function

`table` forms a contingency table from vectors. The first vector becomes the rows and the second vector the columns. Notice the rows and columns are placed in alphabetical order:

```
table(aspirin.df$group, aspirin.df$MI)
```

```
##
##                  No   Yes
##    Aspirin 10933   104
##    Placebo 10845   189
```

The `xtabs` function

The `xtabs` function can create a table in two different ways:

1. from a data frame where there is one observation per record
2. from a data frame where each row represents total counts for each group combination

Syntax:

- `xtabs( ~ group + MI, data = aspirin.df)` - one obs per record
- `xtabs(Freq ~ group + MI, data = aspirin.df)` - each row has total counts stored in column called `Freq`

xtabs example 1

This returns the same thing as `table(aspirin.df$group, aspirin.df$MI)` except with rows and columns labeled:

```
xtabs(~ group + MI, data = aspirin.df)
```

```
##           MI
## group        No   Yes
##   Aspirin 10933   104
##   Placebo 10845   189
```

xtabs example 2

Say our data frame looks like this:

```
aspirin.df2
```

```
##     group  MI  Freq
## 1 Aspirin  No 10933
## 2 Placebo  No 10845
## 3 Aspirin Yes   104
## 4 Placebo Yes   189
```

xtabs example 2

We can use `xtabs` to create a table like so:

```
xtabs(Freq ~ group + MI, data = aspirin.df2)
```

```
##           MI
## group         No   Yes
##   Aspirin 10933   104
##   Placebo 10845   189
```

Don't forget to save tables if you want to analyze them

These just print tables to the R console:

```
table(aspirin.df$group, aspirin.df$MI)
xtabs(~ group + MI, data = aspirin.df)
xtabs(Freq ~ group + MI, data = aspirin.df2)
```

These save the tables for future use:

```
aspirin.tab1 <- table(aspirin.df$group, aspirin.df$MI)
aspirin.tab2 <- xtabs(~ group + MI, data=aspirin.df)
aspirin.tab3 <- xtabs(Freq ~ group + MI, data=aspirin.df2)
```

working with 2-way tables

We often want to work with tables to find proportions or marginal totals. Good functions to know include:

- `prop.table` (cell proportions)
- `margin.table` (table margins)
- `addmargins` (add table margins)

We can also use subsetting brackets `[]` to extract cells, rows or columns of tables.

prop.table example

prop.table calculates cell proportions.

Specify margin = 1 for row-wise proportions, margin = 2 for column-wise proportions. Not specifying margin returns cell proportions summing to 1.

```
prop.table(aspirinT, margin = 1)
```

```
##           MI
## group                Yes          No
##   Placebo   0.01712887 0.98287113
##   Aspirin   0.00942285 0.99057715
```

margin.table example

`margin.table` calculates the marginal totals.

Specify `margin = 1` for row-wise margins, `margin = 2` for column-wise margins. Not specifying `margin` sums all cell totals.

```
margin.table(aspirinT, margin = 1)
```

```
## group
## Placebo Aspirin
##   11034   11037
```

```
margin.table(aspirinT, margin = 2)
```

```
## MI
##   Yes    No
##   293 21778
```

addmargins example

The basic usage is to add row- and column-wise totals as well as the table total. Specifying `margin=1` or `margin=2` shows only row or column totals, respectively.

```
addmargins(aspirinT)
```

```
##           MI
## group        Yes    No   Sum
##   Placebo    189 10845 11034
##   Aspirin    104 10933 11037
##   Sum        293 21778 22071
```

See `?addmargins` for advanced usage.

Subsetting brackets

Specify row/column number, or row/column name, in brackets to extract parts of a table.

Extract the row 1, column 1 cell:

```
aspirinT[1,1]
```

```
## [1] 189
```

or:

```
aspirinT["Placebo","Yes"]
```

```
## [1] 189
```

Subsetting brackets (cont'd)

Extract row 1 (the Placebo group):

```
aspirinT[1,]
```

```
##    Yes     No
##    189 10845
```

Extract column 2 (all cases of MI = "No"):

```
aspirinT[,"No"]
```

```
## Placebo Aspirin
##   10845   10933
```

Analysis: Chi-square test

The chi-square test of independence tests the null hypothesis that two categorical variables are not associated with one another. We can use the `chisq.test` function for this.

```
chisq.test(aspirinT)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity corre
##
## data:  aspirinT
## X-squared = 24.429, df = 1, p-value = 7.71e-07
```

A low p-value rejects the null of no association. Notice it doesn't tell you the strength of association or the direction. A Chi-square test is rarely sufficient for answering all the questions you'll have about your data.

Analysis: comparing proportions

Use the `prop.test` function to compare proportions of "successes". It can take a 2 x 2 table (or matrix) with counts of successes and failures, respectively, in the columns.

```
prop.test(aspirinT)
```

```
##
##  2-sample test for equality of proportions with continu
##  correction
##
## data:  aspirinT
## X-squared = 24.429, df = 1, p-value = 7.71e-07
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.004597134 0.010814914
## sample estimates:
##     prop 1     prop 2
## 0.01712887 0.00942285
```

A drawback of difference in proportions

Using the absolute difference between proportions to compare two groups can be misleading. Example:

- The difference between 0.410 and 0.401 is 0.009.
- The difference between 0.010 and 0.001 is also 0.009.

However the latter difference seems more important if we compare ratios:

- $0.410/0.401 = 1.02244$
- $0.010/0.001 = 10$

0.010 is 10 times larger than 0.001. A difference in proportions may have greater importance when both proportions are near 0 or 1.

Analysis: risk ratios

The *risk ratio* (or *relative risk*) is simply the proportion of successes in one group divided the proportion of successes in the other group. A proportion of 1 implies no difference between the groups.

Risk ratio for aspirin data reveals the proportion of MI for placebo group was about 82% higher:

```
prop.table(aspirinT, margin = 1)[1,1] /
  prop.table(aspirinT, margin = 1)[2,1]
```

```
## [1] 1.817802
```

We'll use the `epitools` package for easier calculation of risk ratios and to obtain confidence intervals.

odds

We can also compare odds of success in one group versus odds of success in another group.

The odds are related to probability as follows:

$$odds = \frac{p}{1-p}$$

- When $p = 0.75$, the odds are $\frac{0.75}{0.25} = 3$. We expect to observe 3 successes for every 1 failure.
- When $p = 0.25$, the odds are $\frac{0.25}{0.75} = \frac{1}{3}$. We expect to observe about 1 success for every 3 failures.

odds ratios

When using odds to compare groups, we typically take the ratio of the two odds.

The ratio of two odds is called an *odds ratio*. An odds ratio of 1 implies there is no difference between the odds of success in the two groups.

A quick formula for calculating the odds ratio in a 2 x 2 table is

$$\hat{\theta} = \frac{n_{11} n_{22}}{n_{12} n_{21}}$$

For this reason, the odds ratio is also known as the *cross-product ratio*.

Analysis: odds ratios

For our aspirin data, the odds of MI in the placebo group is about 83% higher than the odds of MI in the aspirin group:

```
(aspirinT[1,1] * aspirinT[2,2]) /
  (aspirinT[1,2] * aspirinT[2,1])
```

```
## [1] 1.832054
```

Again, we'll use the `epitools` package for easier calculation of odds ratios and to obtain confidence intervals.

Let's go to R!

Example 2

Eight studies in China about smoking and lung cancer. (*Intern. J. Epidemiol.*, 21:197-201,1992) We investigate the relationship between smoking and lung cancer while controlling for city.

```
##                          lung.cancer yes  no
## city       smoking
## Beijing    smokers                   126 100
##            nonsmokers                 35  61
## Shanghai   smokers                   908 688
##            nonsmokers                497 807
## Shenyang   smokers                   913 747
##            nonsmokers                336 598
## Nanjing    smokers                   235 172
##            nonsmokers                 58 121
## Harbin     smokers                   402 308
##            nonsmokers                121 215
## Zhengzhou  smokers                   182 156
##            nonsmokers                 72  98
## Taiyuan    smokers                    60  99
```

Three-way contingency tables

The data on the previous slide have three categorical variables. The tabular layout is a *three-way contingency table*.

This layout is useful when we want to compare two categorical variables while controlling for a third categorical variable.

"Controlling for a third categorical variable" basically means holding it at a constant level while analyzing the other two categorical variables.

The two-way table formed when controlling for a third variable is called a *partial table*.

Some analysis options

1. Test the null hypothesis that smoking and lung cancer are conditionally independent given city, that is the odds ratio $= 1$ for each partial table (Cochran-Mantel-Haenszel Test)
2. Test the null hypothesis that the odds ratio between smoking and lung cancer is the same at each level of city. (Breslow-Day Test)
3. Estimate a *common odds ratio* if association seems stable across the partial tables.

Getting data into R

We demonstrate how to get this data into R in the R script that accompanies this workshop. The code doesn't fit comfortably onto slides.

The manual method involves the `array` function that allows us to specify a third dimension, or third layer, of data.

Once again we can use the `table` and `xtabs` functions to create our contingency tables if we import our data into a data frame.

The table function for 3-way tables

The first vector in the table function forms the rows, the second the columns, and each subsequent vector a layer.

```
table(lc.df$smoking, lc.df$lung.cancer, lc.df$city)
```

```
## , ,  = Beijing
##
##
##                no yes
##   nonsmokers   61  35
##   smokers     100 126
##
## , ,  = Harbin
##
##
##                no yes
##   nonsmokers  215 121
##   smokers     308 402
##
```

The xtabs function for 3-way tables

```
# one row per obs
xtabs( ~ smoking + lung.cancer + city, data = lc.df)
```

```
## , , city = Beijing
##
##                lung.cancer
## smoking          no yes
##   nonsmokers     61  35
##   smokers       100 126
##
## , , city = Harbin
##
##                lung.cancer
## smoking          no yes
##   nonsmokers    215 121
##   smokers       308 402
##
## , , city = Nanchang
```

If our data look like this. . .

```
head(lc.df2, n = 8)
```

```
##       smoking lung.cancer    city Freq
## 1 nonsmokers          no Beijing   61
## 2    smokers          no Beijing  100
## 3 nonsmokers         yes Beijing   35
## 4    smokers         yes Beijing  126
## 5 nonsmokers          no  Harbin  215
## 6    smokers          no  Harbin  308
## 7 nonsmokers         yes  Harbin  121
## 8    smokers         yes  Harbin  402
```

The xtabs function for 3-way tables

```
# one row per cell Freq
xtabs(Freq ~ smoking + lung.cancer + city, data = lc.df2)
```

```
## , , city = Beijing
##
##                lung.cancer
## smoking          no yes
##   nonsmokers     61  35
##   smokers       100 126
##
## , , city = Harbin
##
##                lung.cancer
## smoking          no yes
##   nonsmokers    215 121
##   smokers       308 402
##
## , , city = Nanchang
```

working with 3-way tables

Working with 3-way tables is a little trickier due to the third dimension. We can use the same functions, but we need to account for the third dimension.

- `prop.table` (cell proportions)
- `margin.table` (get table margins)
- `addmargins` (add table margins)
- `ftable` ("flatten" contingency tables into two dimensions)

We can also use subsetting brackets to extract parts of tables, but again need to specify third dimension.

prop.table example

Specify `margin = c(1,3)` for row-wise proportions *within each two-way table*; specify `margin = c(2,3)` for column-wise proportions *within each two-way table*. Not specifying `margin` returns cell proportions summing to 1.

```
prop.table(lung.cancer, margin = c(1,3))
```

```
## , , city = Beijing
##
##                lung.cancer
## smoking              yes        no
##   smokers     0.5575221 0.4424779
##   nonsmokers 0.3645833 0.6354167
##
## , , city = Shanghai
##
##                lung.cancer
## smoking              yes        no
##   smokers     0.5689223 0.4310777
```

margin.table example

Specify `margin = c(1,3)` for row-wise margins *within each two-way table*; specify `margin = c(2,3)` for column-wise margins *within each two-way table*. Not specifying `margin` sums all cell totals.

```
margin.table(lung.cancer, margin = c(1,3))
```

```
##                 city
## smoking       Beijing Shanghai Shenyang Nanjing Harbin Zh
##   smokers         226     1596     1660     407    710
##   nonsmokers       96     1304      934     179    336
##                 city
## smoking       Nanchang
##   smokers          193
##   nonsmokers        57
```

addmargins example

The basic usage is to add row- and column-wise totals as well as the table total. Specifying `margin=1` or `margin=2` shows only row or column totals, respectively.

```
addmargins(lung.cancer)
```

```
## , , city = Beijing
##
##                lung.cancer
## smoking         yes  no Sum
##   smokers       126 100 226
##   nonsmokers     35  61  96
##   Sum           161 161 322
##
## , , city = Shanghai
##
##                lung.cancer
## smoking         yes   no  Sum
##   smokers       908  688 1596
```

ftable example

By default, the third layer becomes the columns while the first and second layers are collapsed into the rows. You can specify which variables form the rows and columns using the `row.vars` and `col.vars` arguments.

```
ftable(lung.cancer)
```

```
##                                city Beijing Shanghai Shenyang Na
## smoking    lung.cancer
## smokers    yes                        126      908      913
##            no                         100      688      747
## nonsmokers yes                         35      497      336
##            no                          61      807      598
```

Subsetting brackets for 3-way tables

Just the partial table for Beijing:

```
lung.cancer[,,1]
```

```
##                lung.cancer
## smoking         yes  no
##   smokers       126 100
##   nonsmokers     35  61
```

Or:

```
lung.cancer[,,"Beijing"]
```

Subsetting brackets for 3-way tables (cont'd)

All smokers with lung cancer:

```
lung.cancer["smokers","yes",]
```

```
##    Beijing  Shanghai  Shenyang   Nanjing     Harbin Zhengz
##        126       908       913       235        402
##  Nanchang
##        104
```

All people with lung cancer:

```
lung.cancer[,"yes",]
```

```
##              city
## smoking       Beijing Shanghai Shenyang Nanjing Harbin Zh
##   smokers         126      908      913     235    402
##   nonsmokers       35      497      336      58    121
##              city
## smoking       Nanchang
##   smokers          104
```

Analysis: Cochran-Mantel-Haenszel (CMH) Test

Use the `mantelhaen.test` function to test the null hypothesis that smoking and lung cancer are conditionally independent given city (ie, odds ratio $= 1$ for each partial table)

```
##
##  Mantel-Haenszel chi-squared test with continuity correc
##
## data:  lung.cancer
## Mantel-Haenszel X-squared = 279.38, df = 1, p-value < 2.
## alternative hypothesis: true common odds ratio is not eq
## 95 percent confidence interval:
##  1.984002 2.383249
## sample estimates:
## common odds ratio
##          2.174482
```

Common odds ratio

It is more informative to estimate the strength of association than test a hypothesis about it.

The `mantelhaen.test` also returns an estimate of the *common odds ratio* along with a 95% confidence interval. The common odds ratio is basically a weighted average of the partial table odds ratios.

For the lung cancer data, the commons odds ratio is estimated as $2.17(1.98, 2.38)$. The odds of lung cancer for smokers is about twice the odds of lung cancer for non-smokers.

The CMH Test and common odds ratio estimate is really only appropriate when association is similar in each table.

Analysis: Breslow-Day Test

We can use the `BreslowDayTest` function in the `DescTools` package to test the null hypothesis that the odds ratio between smoking and lung cancer is the same at each level of city.

As we'll see in the R script, the result of the Breslow-Day Test for our lung cancer data is not significant.

This means we're justified in summarizing the conditional association by a single odds ratio for all eight partial tables.

Let's go to R!

Modeling categorical data

Another way to analyze categorical data is to *model* it as a linear function of one or more explanatory variables.

By *linear function* we mean a weighted sum of variables. Say we model Y as a linear function of X and Z and obtain the following:

$$Y = 0.65X + 0.04Z$$

This says we approximate Y by adding 0.65 * X and 0.04 * Z.

Logistic regression

We model categorical data using *logistic regression*.

- Binary logistic regression is for modeling binary outcomes (a categorical variable with two levels)
- Multinomial logistic regression is for modeling multiple unordered outcomes (a nominal categorical variable with more than two levels)
- Ordered logistic regression is for modeling multiple ordered outcomes (an ordered categorical variable with more than two levels)

We will just cover binary logistic regression.

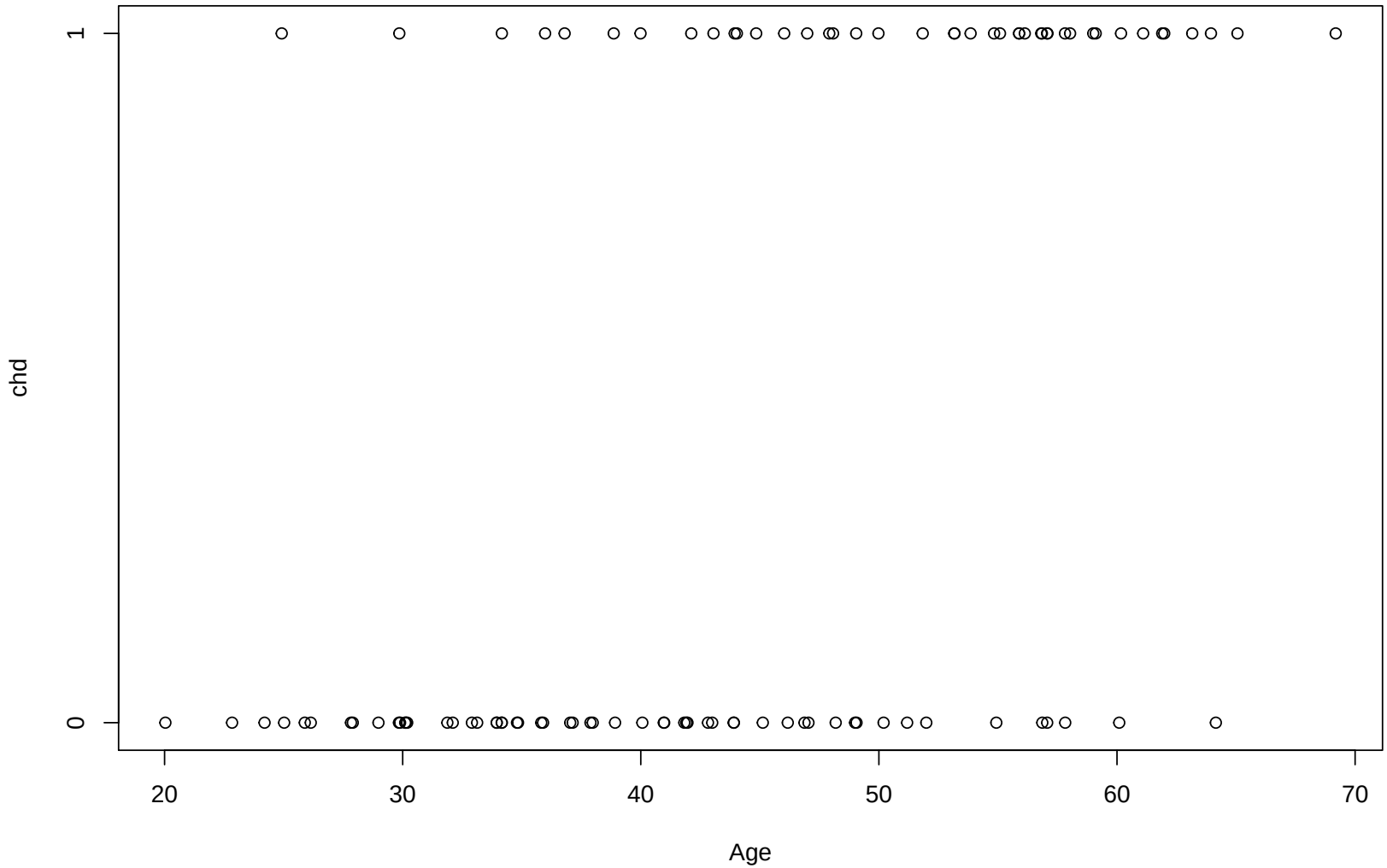
Example 3

Explore the relationship between age and presence/absence of coronary heart disease (CHD) in a study population. (Table 1.1, *Applied Logistic Regression*, 2nd Ed.)

```
##    id age chd
## 1   1  20   0
## 2   2  23   0
## 3   3  24   0
## 4   5  25   1
## 5   4  25   0
## 6   7  26   0
```

Scatterplot of CHD by age

Difficult to describe this relationship based on this graph.

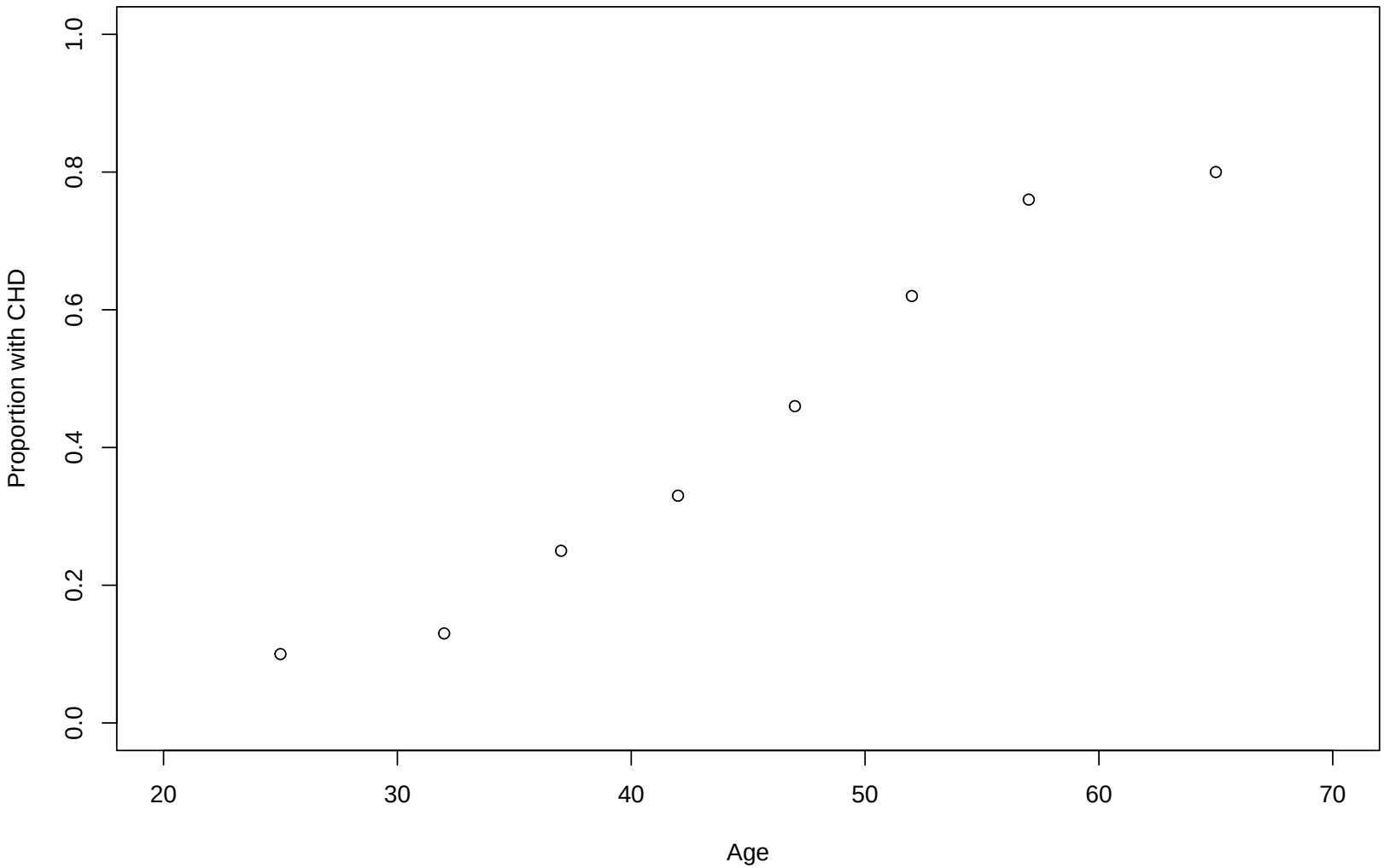


Transform the data to better understand relationship

Calculate proportion of CHD within age groups.

```
##    agrp  n absent present proportion
## 1   <29 10      9       1       0.10
## 2 30-34 15     13       2       0.13
## 3 35-39 12      9       3       0.25
## 4 40-44 15     10       5       0.33
## 5 45-49 13      7       6       0.46
## 6 50-54  8      3       5       0.62
## 7 55-59 17      4      13       0.76
## 8   >60 10      2       8       0.80
```

Scatterplot of proportion with CHD by age group



A function to model the relationship

Modeling the data in the previous graph essentially means finding a smooth mathematical function that describes the relationship between age and proportion with CHD.

The function would allow us to estimate the probability of CHD in the study population given a certain age.

The function would also allow us to summarize how the probability of CHD changes as age increases.

The function needs to work such that plugging in age produces a number between 0 and 1. *This is basically what logistic regression ensures.*

Performing logistic regression in R

We use the `glm` function with the `family` argument set to `binomial` to perform logistic regression in `R`.

For the CHD, data this would be done as follows:

```
mod1 <- glm(chd ~ age, data = chd, family = binomial)
```

This performs binary logistic regression and saves the result to an object we called `mod1`.

`chd ~ age` means "model chd as a function of age".

To see the results:

```
summary(mod1)
```

logistic regression results

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9718  -0.8456  -0.4576   0.8253   2.2859

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.30945    1.13365  -4.683 2.82e-06 ***
age          0.11092    0.02406   4.610 4.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 136.66  on 99  degrees of freedom
Residual deviance: 107.35  on 98  degrees of freedom
AIC: 111.35
```

Reviewing the results

The "formula" for our functional model is in the "Coefficients" section. For the model we fit:

$$P(\text{chd}) = -5.31 + 0.11 * \text{age}$$

Both are significant, which is to say they are significantly different from 0. (The intercept is almost always "significant".)

The Null deviance summarizes goodness of fit for a model with no predictors. The Residual deviance summarizes goodness of fit for the model we fit. Ideally the residual deviance will be much smaller than the null deviance.

The logit transformation

Notice if we plug in an age $= 60$, we get chd $= 1.29$. That's because the response is on the *logit*, or *log-odds*, scale!

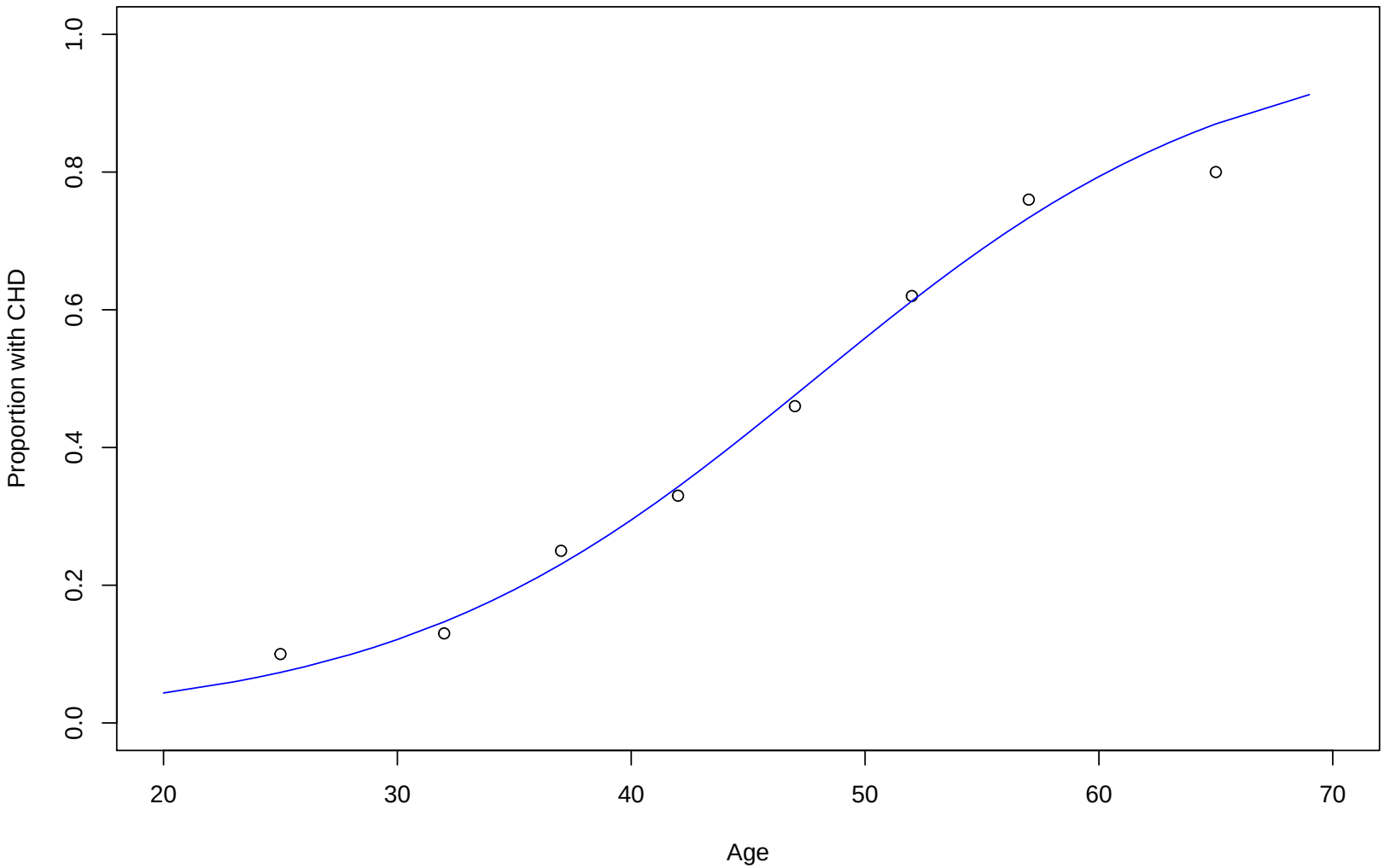
The logit transformation transforms values ranging from 0 to 1, to numbers ranging from $-\infty$ to $+\infty$.

To make our model return a number between 0 and 1, we need to take the *inverse logit*. Fortunately R will do this for us. Here's how:

```
predict(mod1, type="response",
        newdata = data.frame(age = 60))
```

```
##         1
## 0.7934446
```

Our functional model with observed data



Interpreting coefficients

If we exponentiate a variable's coefficient in a logistic regression model we get an odds ratio estimate of that variable's effect on the response.

```
exp(0.11) = 1.116
```

This says the odds of CHD increase by about 12% for every one year increase in age. To see how the odds change for every 10 years we could do this:

```
exp(0.11 * 10) = 3.004
```

The odds of CHD for a 50 year old are about 3 times that of a 40 year old.

Multiple logistic regression

The power of logistic regression is being able to include multiple independent variables in your model, categorical or continuous. This allows us to "control" for other variables.

The odds ratio interpetation of the coefficients is the same, but now it is in the context of the additional variables being held constant.

Simply add additional variables to your model with +. For example:

```
mod2 <- glm(chd ~ age + smoking + bp, data = chd,
family = binomial)
```

Let's go to R!

logistic regression and analysis of contigency tables

It turns out we can do most of the same analyses with logistic regression that we did with contingency tables.

Recall the estimated odds ratio of MI for those on Placebo vs. Aspirin in our two-way contingency table: 1.83

Recall the common odds ratio of lung cancer for smokers vs. nonsmokers in our 3-way contingency table: 2.17

The aspirin data analyzed with logistic regression

```
mod.asp <- glm(MI ~ group, data = aspirin.df,
               family = binomial)
coef(mod.asp) # the model coefficients
```

```
##   (Intercept) groupPlacebo
##    -4.6551501    0.6054377
```

```
# odds ratio estimate
exp(coef(mod.asp)[2])
```

```
## groupPlacebo
##     1.832054
```

The lung cancer data analyzed with logistic regression

Exponentiating the smoking coefficient provides the same common odds ratio estimate as the Cochran-Mantel-Haenszel (CMH) Test.

```
mod.lc <- glm(lung.cancer ~ smoking + city, data = lc.df,
              family = binomial)

# common odds ratio estimate
exp(coef(mod.lc)[2])
```

```
## smokingsmokers
##       2.175072
```

The lung cancer data analyzed with logistic regression

Including an interaction in a logistic regression with two categorical predictors allows us to carry out a Breslow-Day test. The `anova` function called on a modeling object tests the significance of the interaction, which is the same as a Breslow-Day test.

```
mod.lc <- glm(lung.cancer ~ smoking + city
                smoking:city, data = lc.df,
                 family = binomial)

# Breslow-Day test of homogeneity
# (ie, test if interaction is significant)
anova(mod.lc)
```

Let's go to R!

References

Agresti, A. *An Introduction to Categorical Data Analysis*. (1996)

Hosmer, D. and Lemeshow, S. *Applied Logistic Regression*, 2nd Ed.. (2000)

Faraway, J. *Extending the Linear Model with R*. (2006)

Thanks for coming today!

For help and advice with your statistical analysis:
`statlab@virginia.edu`

Sign up for more workshops or see past workshops:
`http://data.library.virginia.edu/training/`

Register for the Research Data Services newsletter to stay up-to-date on RDS events and resources:
`http://data.library.virginia.edu/newsletters/`