MPP-E1180 Lecture 8: Statistical Modeling with R

Christopher Gandrud

6 November 2014

Objectives for the week

- Assignment 3
- Review
- Intro to the general syntax for statistical modelling in R.
- Specific examples using:
  - Normal linear regression
  - Logistic regression

Week 9

Class for Week 9 is on **10 October (Monday) from 14:00-16:00**.

Assignment 3

Purpose: Gather, clean, and analyse data

Deadline: 14 November 2015

You will submit a GitHub repo that:

- Gathers web-based data from at least **two sources**. Cleans and merges the data so that it is ready for statistical analyses.
- Conducts basic descriptive and inferential statistics with the data to address a relevant research question.
- Briefly describes the results including with dynamically generated tables and figures.
- Has a write up of 1,500 words maximum that describes the data gathering and analysis and uses literate programming.

Assignment 3

Note: I will be traveling/at a conference/not able to check my email much on the **13th** and **14th**.

So try to ask all of your **questions by the 12th (Wednesday)**.

I will have **normal office hours** on Wednesday.

Review

- What is **web scraping**? What are some of tools R has for web scraping?
- What are **regular expressions** (give at least two examples)? Why are they useful?
- What dplyr function can you use to create a **new variable** in a data frame by running a command on values from groups in that data frame?

Statistical Modelling in R

Caveat: We are **definitely not** going to cover anywhere near R's full capabilities for statistical modeling.

We are also **not going to cover** all of the **modeling concerns/diagnostics** you need to consider when using a given model.

You will need to **rely on your other stats courses** and **texts**.

What are we going to do?

- Discuss the basic syntax and capabilities in R for estimating normal linear and logistic regressions.
- Basic model checking in R.
- Discuss basic ways of interpreting results (we'll do more on this next week).

The basic model

Most statistical models you will estimate are from a general class that has **two parts**:

Stocastic Component that assumes the dependent variable Y_i is generated from as a random draw from the probability density function:

$$Y_i \sim f(\theta_i, \alpha)$$

- θ_i: parameter vector of the part of the function that **varies between observations**.
- α: matrix of **non-varying parameters**.

Sometimes referred to as the '**error structure**'.

The basic model

The **Systematic Component** indicating how θ_i varies across observations depending on values of the explanatory variables and (often) some constant:

$$\theta_i = g(X_i,\ \beta)$$

- X_i: a $1 \text{ x } k$ vector of **explanatory variables**.
- β: a $1 \text{ x } k$ vector of **parameters** (i.e. coefficients).
- $g(.,.)$: the **link function**, specifying how the explanatory variables and parameters are translated into θ_i.

Today

Today we will cover two variations of this general model:

- linear-normal regression (i.e. ordinary least squares)
- logit model

Linear-normal regression

For continuous dependent variables assume that Y_i is from the **normal distribution** $(N(.,.))$.

Set the main parameter vector θ_i to the **scalar mean** of: $\theta_i = E(y_i) = \mu_i$.

Assume the ancillary parameter matrix is the scalar homoskedastic variance: $\alpha = V(Y_i) = \sigma^2$.

- **Homoskedastic variance**: variance does not depend on the value of x. The standard deviation of the error terms is constant across values of x.

Set the systematic component to the linear form: $g(X_i, \beta) = X_i\beta = \beta_0 + X_{i1}\beta_1 + \dots$.

Linear-normal regression

So:

$$Y_i \sim N(\mu_i, \sigma^2), \quad \mu_i = X_i \beta$$

Logit regression

For binary data (e.g. 0, 1) we can assume that the stochastic component has a Bernoulli distribution.

The main parameter is $\pi_i = \Pr(Y_i = 1)$.

The systematic component is set to a logistic form: $\pi_i = \frac{1}{1+e^{-X_i\beta}}$.

So:

$$Y_i \sim \text{Bernoulli}(\pi_i), \quad \pi_i = \frac{1}{1 + e^{-X_i\beta}}$$

Error structure family and link function

Error Family	Canonical link
Normal	identity
binomial	logit
poisson	log

R syntax

The general syntax for estimating statistical models in R is:

```
response variable ~ explanatory variable(s)
```

Where '~' reads 'is modelled as a function of'.

Model functions

We use model functions to specify the model structure.

Basic model functions include:

- `lm`: fits a linear model where Y is assumed to be normally distributed and with homoskedastic variance.
- `glm`: allows the fitting of many Generalised Linear Models. Lets you specify the `family` and the `link` function.

Example of `lm`

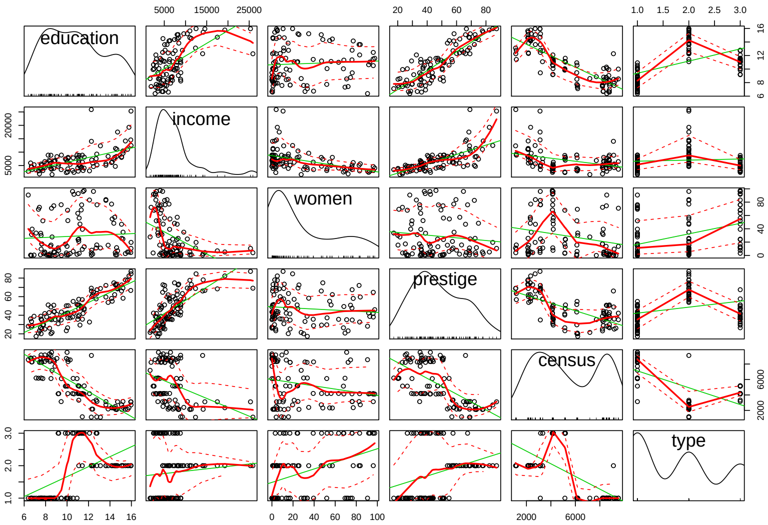
Example data *Prestige* (example based on
`http://www.princeton.edu/~otorres/Regression101R.pdf`).

The observations are **occupations** and the dependent variable is a score of each occupations' **prestige**.

```
library(car)
data(Prestige)
```

Examine correlation matrix

```
car::scatterplotMatrix(Prestige)
```



Example of `lm`

Estimate simple model:

```
M1 <- lm(prestige ~ education, data = Prestige)
```

```r
summary(M1)
```

```
##
## Call:
## lm(formula = prestige ~ education, data = Prestige)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.0397  -6.5228   0.6611   6.7430  18.1636
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -10.732      3.677  -2.919  0.00434 **
## education      5.361      0.332  16.148  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
##
## Residual standard error: 9.103 on 100 degrees of freedo
## Multiple R-squared:  0.7228, Adjusted R-squared:  0.72
```

Confidence intervals of parameter point estimates

Note: **Always prefer estimation intervals** over point estimates.

Deal with your **uncertainty**!

About **95%** of the time the population parameter will be within **about 2 standard errors** of the point estimate.

Using **Central Limit Theorem** (at least about 50 observations and the data is not extremely skewed):

$$CI\_95 = \text{point estimate} \pm 1.96 * SE$$

Confidence intervals of parameter point estimates

```r
confint(M1)
```

```
##                   2.5 %    97.5 %
## (Intercept) -18.027220 -3.436744
## education     4.702223  6.019533
```

Example of `lm`

Estimate model with categorical (factor) variable:

```
M2 <- lm(prestige ~ education + type,
         data = Prestige)
```

```r
summary(M2)
```

```
##
## Call:
## lm(formula = prestige ~ education + type, data = Prestig
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.410  -5.508   1.360   5.694  17.171
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.6982     5.7361  -0.470   0.6392
## education     4.5728     0.6716   6.809 9.16e-10 ***
## typeprof      6.1424     4.2590   1.442   0.1526
## typewc       -5.4585     2.6907  -2.029   0.0453 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
##
```

Create categorical variable from continuous variable

Use the cut function to create a categorical (factor) variable from a continuous variable.

```
Prestige$income_cat <- cut(Prestige$income,
                breaks = c(0, 4999, 9999, 14999, 30000),
                labels = c('< 5,000', '< 10,000', '< 15,00
                          '>= 15,000'))
summary(Prestige$income_cat)
```

```
##    < 5,000  < 10,000  < 15,000 >= 15,000
##         38        51         9         4
```

Note: cut excludes the left value and includes the right value, e.g. (0, 4999].

Example of lm

```r
M3 <- lm(prestige ~ education + income_cat,
         data = Prestige)
confint(M3)
```

```
##                              2.5 %    97.5 %
## (Intercept)             -10.914031  3.182500
## education                 3.350201  4.778444
## income_cat< 10,000        6.085375 13.030546
## income_cat< 15,000        9.584532 23.391854
## income_cat>= 15,000      12.040936 29.902733
```

Example of `lm`

Estimate models with polynomial transformations:

```
# Cubic polynomial transformation
M4  <- lm(prestige ~ education + poly(income, 2),
          data = Prestige)
confint(M4)
```

```
##                        2.5 %    97.5 %
## (Intercept)        -1.470988  13.33552
## education           3.132827   4.48515
## poly(income, 2)1   45.174019  81.39445
## poly(income, 2)2  -43.150740 -12.87994
```

Example of `lm`

Estimate models with (natural) logarithmic transformations:

```
# Cubic polynomial transformation
M5  <- lm(prestige ~ education + log(income),
          data = Prestige)
```

```
summary(M5)
```

```
##
## Call:
## lm(formula = prestige ~ education + log(income), data =
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.0346  -4.5657  -0.1857   4.0577  18.1270
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -95.1940    10.9979  -8.656 9.27e-14 ***
## education     4.0020     0.3115  12.846  < 2e-16 ***
## log(income)  11.4375     1.4371   7.959 2.94e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
##
## Residual standard error: 7.145 on 99 degrees of freedom
```

Example of `lm`

Estimate model with interactions:

```
M6 <- lm(prestige ~ education * type,
         data = Prestige)
```

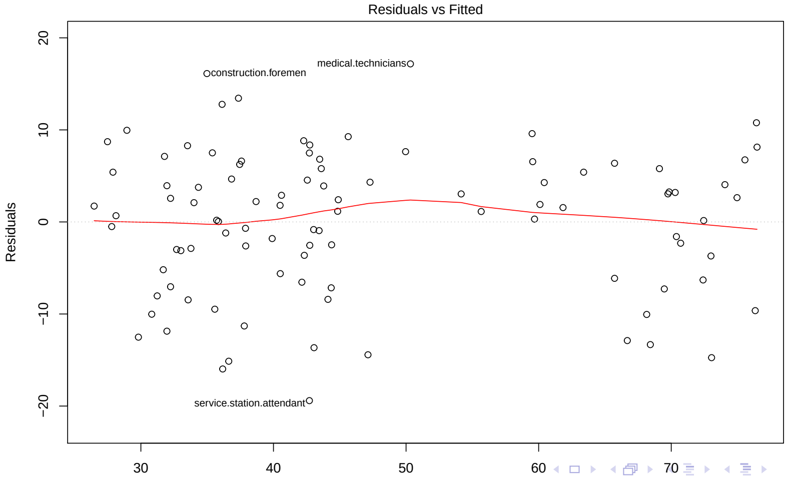
```r
summary(M6)
```

```
##
## Call:
## lm(formula = prestige ~ education * type, data = Prestig
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.7095  -5.3938   0.8125   5.3968  16.1411
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -4.2936     8.6470  -0.497    0.621
## education             4.7637     1.0247   4.649 1.11e-05
## typeprof             18.8637    16.8881   1.117    0.267
## typewc              -24.3833    21.7777  -1.120    0.266
## education:typeprof   -0.9808     1.4495  -0.677    0.500
## education:typewc      1.6709     2.0777   0.804    0.423
## ---
```

Diagnose heteroscedasticity

Use `plot` on a model object to run visual diagnostics.

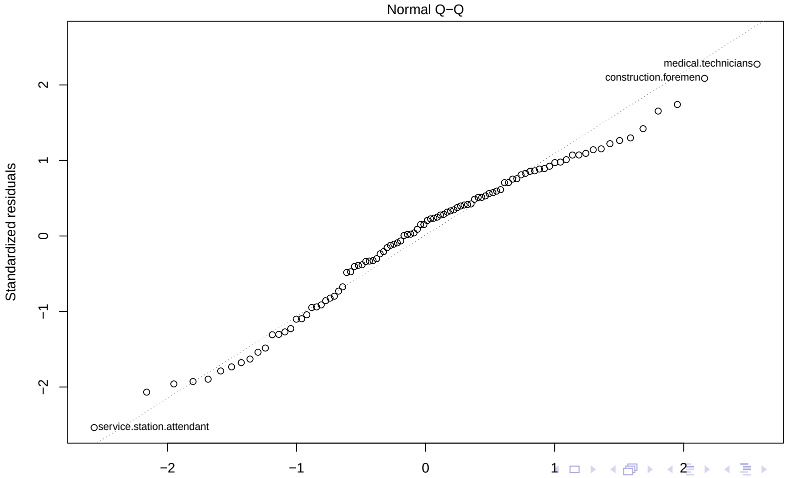
```r
plot(M2, which = 1)
```



Diagnose non-normality of errors

`plot` to see if a model's errors are normally distributed.

```
plot(M2, which = 2)
```



Example of logistic regression with `glm`

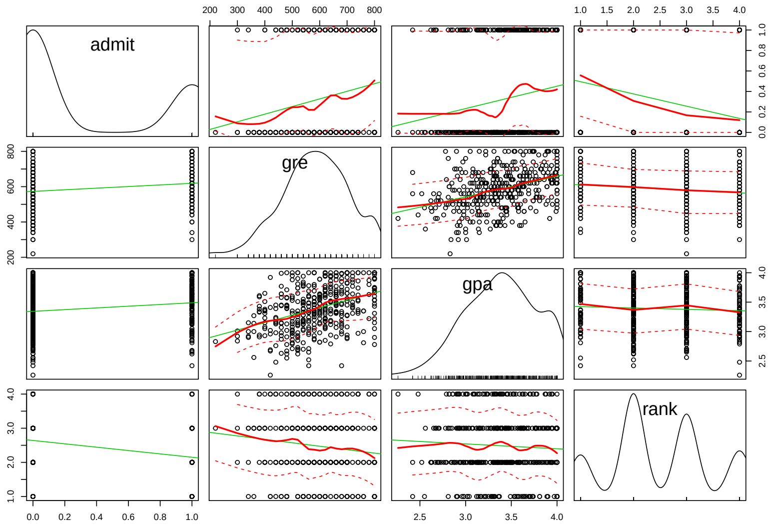
Example from UCLA IDRE.

Simulated data of admission to grad school.

```
# Load data
URL <- 'http://www.ats.ucla.edu/stat/data/binary.csv'
Admission <- read.csv(URL)
```

Example of logistic regression with `glm`

```
car::scatterplotMatrix(Admission)
```



Contingency table for school rank and admission

```
xtabs(~admit + rank, data = Admission)
```

```
##       rank
## admit  1  2  3  4
##     0 28 97 93 55
##     1 33 54 28 12
```

```
summary(xtabs(~admit + rank, data = Admission))
```

```
## Call: xtabs(formula = ~admit + rank, data = Admission)
## Number of cases in table: 400
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 25.242, df = 3, p-value = 1.374e-05
```

Example of logistic regression with `glm`

```
Logit1 <- glm(admit ~ gre + gpa + as.factor(rank),
              data = Admission, family = 'binomial')
```

Note: Link function is assumed to be logit if `family = 'binomial'`.

Example of logistic regression with `glm`

```r
confint(Logit1)
```

```
##                          2.5 %       97.5 %
## (Intercept)      -6.2716202334 -1.792547080
## gre               0.0001375921  0.004435874
## gpa               0.1602959439  1.464142727
## as.factor(rank)2 -1.3008888002 -0.056745722
## as.factor(rank)3 -2.0276713127 -0.670372346
## as.factor(rank)4 -2.4000265384 -0.753542605
```

Interpreting logistic regression results

β's in logistic regression are interpretable as **log odds**. These are weird.

If we exponentiate log odds we get **odds ratios**.

```
exp(cbind(OddsRatio = coef(Logit1), confint(Logit1)))
```

```
##                    OddsRatio       2.5 %    97.5 %
## (Intercept)        0.0185001 0.001889165 0.1665354
## gre                1.0022670 1.000137602 1.0044457
## gpa                2.2345448 1.173858216 4.3238349
## as.factor(rank)2 0.5089310 0.272289674 0.9448343
## as.factor(rank)3 0.2617923 0.131641717 0.5115181
## as.factor(rank)4 0.2119375 0.090715546 0.4706961
```

These are **also weird**.

Interpreting logistic regression results

What we really want are **predicted probabilities**

First create a data frame of fitted values:

```
fitted <- with(Admission,
               data.frame(gre = mean(gre),
                          gpa = mean(gpa),
                          rank = factor(1:4)))
fitted
```

```
##     gre    gpa rank
## 1 587.7 3.3899    1
## 2 587.7 3.3899    2
## 3 587.7 3.3899    3
## 4 587.7 3.3899    4
```

Interpreting logistic regression results

Second predict probability point estimates for each fitted value.

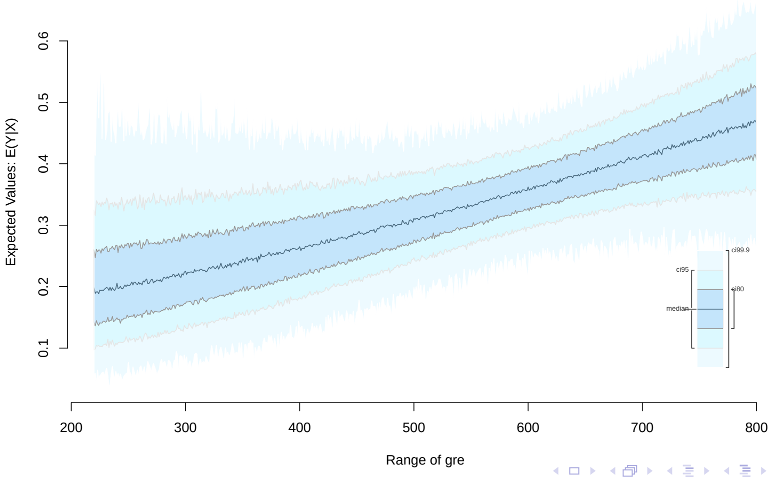
```
fitted$predicted <- predict(Logit1, newdata = fitted,
                            type = 'response')

fitted
```

```
##      gre    gpa rank predicted
## 1 587.7 3.3899    1 0.5166016
## 2 587.7 3.3899    2 0.3522846
## 3 587.7 3.3899    3 0.2186120
## 4 587.7 3.3899    4 0.1846684
```

More interpretation

Next week we will explore other methods of interpreting results from regression models.



Seminar: modeling

Begin working on the statistical models for **your project**.

and/or

Out of Lecture Challenge: Estimate a normal regression model and **plot predicted values** with uncertainty across a range of fitted values.

Functions you might find useful:

- `coef` and `confint`