4. Model fitting

Thomas Lumley
Ken Rice

Universities of Washington and Auckland

*Seattle, July 2012*
Regression commands

Two of the most important R commands;

• `lm()`: fits **Linear Models**

• `glm()`: fits **Generalized Linear Models**

(If you’ve used SAS, its `glm` is not the same as R’s)

‘Linear Regression’ and ‘Logistic Regression’ are special cases.

There’s a lot to learn here – entire graduate courses! – so the help files are huge. How are `lm()`, `glm()` used in genetics?
Linear regression, with SNPs

Many analyses fit the ‘additive model’

\[ y = \beta_0 + \beta \times \# \text{minor alleles} \]
Linear regression, with SNPs

An alternative is the ‘dominant model’;

\[ y = \beta_0 + \beta \times (G \neq AA) \]
Linear regression, with SNPs

or the ‘recessive model’;

\[ y = \beta_0 + \beta \times (G == AA) \]
Finally, the ‘two degrees of freedom model’;

$$y = \beta_0 + \beta_{Aa} \times (G == Aa) + \beta_{aa} \times (G == aa)$$
Use of `lm()` in genetics

The `lm()` command fits all of these, in the same way. Formally,

```
lm(outcome \sim \text{genetic.predictor}, [...] )
```

estimates the association between outcome and predictor.

The **optional** arguments [...] might be

- `data = \text{my.data} - \text{your dataset}
- `subset = \text{race=="CEPH" - use partial data
- `weights = - for advanced analyses
Use of `lm()` in genetics

How to make the `genetic.predictor` variable? Suppose you had genotypes stored as character strings ("AA"/"Aa"/"aa") in a vector `g`. You might use these commands:

<table>
<thead>
<tr>
<th>Chosen Model</th>
<th>Command to define variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additive</strong></td>
<td><code>genetic.predictor &lt;- (g==&quot;Aa&quot;) + 2*(g==&quot;aa&quot;)</code></td>
</tr>
<tr>
<td><strong>Dominant</strong></td>
<td>`genetic.predictor &lt;- (g==&quot;Aa&quot;)</td>
</tr>
<tr>
<td><strong>Recessive</strong></td>
<td><code>genetic.predictor &lt;- g==&quot;aa&quot;</code></td>
</tr>
<tr>
<td>2 degs of freedom</td>
<td><code>genetic.predictor &lt;- factor(g)</code></td>
</tr>
</tbody>
</table>

When R meets `FALSE` or `TRUE` in a ‘math’ setting, it will **coerce** them to be zero or one. So `1 + 2*TRUE` is 3, `TRUE + 2*FALSE` is 1, etc. Using `factor()` sets up several binary variables.

- There are *many* other ways to do this!
  - Use `table(g, genetic.predictor)` to check your method
- Often, genotypes may be stored as 0/1/2. This is easier to work with in R – but makes it harder to decide if A/C/G/T is the minor allele, or risk allele.
**lm(): Estimates, Intervals, p-values**

`lm()` produces **point estimates** for your model;

```r
> genetic.predictor <- (g=="Aa") + 2*(g=="aa") #using additive model
> my.lm <- lm( cholesterol ~ genetic.predictor )
> my.lm
Call:
  lm(formula = cholesterol ~ genetic.predictor)
Coefficients:
  (Intercept)  predictor
        0.2104      0.9507
```

– also available via `my.lm$coefficients` or `coef(my.lm)`.

The coefficients in the output tell you the **additive increase** in outcome associated with a **one-unit** difference in the genetic predictor.

The coefficient for `predictor` is in units of cholesterol per 'a' allele.
**lm(): Estimates, Intervals, p-values**

You will also want *confidence intervals*;

```r
> confint.default(my.lm)
     2.5 %    97.5 %
(Intercept) 0.08391672 0.3368275
predictor   0.85279147 1.0486953
```

Remember to **round these numbers** to an appropriate number of significant figures! (2 or 3 is usually enough)

We are *seldom* interested in the Intercept
lm(): Estimates, Intervals, p-values

Two-sided **p-values** are also available;

```r
> summary(my.lm)
Coefficients:
            Estimate Std. Error  t value Pr(>|t|)  
(Intercept) 0.21037    0.06426   3.274   0.00119 ** 
predictor   0.95074    0.04977  19.101   < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- In this data, we have **strong evidence** of an **additive effect** of the minor allele on cholesterol

- summary(my.lm) gives **many** other details – ignore for now

- Confidence intervals are just Estimate ± 2×Std.Error
Use of `glm()` in genetics

**Logistic regression** is the ‘default’ analysis for **binary outcomes**

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Type</th>
<th>Regression</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesterol</td>
<td>Continuous</td>
<td>Linear</td>
<td>Difference in Outcome</td>
</tr>
<tr>
<td>Blood Pressure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Death</td>
<td>Binary</td>
<td>Logistic</td>
<td>Ratio of odds</td>
</tr>
<tr>
<td>Stroke</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMI &gt; 30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What are **odds**? Really just **probability**...
Use of \texttt{glm()} in genetics

Odds are a [gambling-friendly] measure of chance;

<table>
<thead>
<tr>
<th>Probability</th>
<th>AA</th>
<th>Aa</th>
<th>aa</th>
</tr>
</thead>
<tbody>
<tr>
<td>66%</td>
<td>2</td>
<td>1</td>
<td>33%</td>
</tr>
<tr>
<td>50%</td>
<td>1</td>
<td>1</td>
<td>50%</td>
</tr>
<tr>
<td>90%</td>
<td>9</td>
<td>1</td>
<td>10%</td>
</tr>
</tbody>
</table>
Use of \texttt{glm()} in genetics

Odds are a [gambling-friendly] measure of chance;

\begin{align*}
\text{Prob(death)} & \quad 90\% \quad 9 \\
\text{Prob of survival, 10 yrs} & \quad 10\% \quad 1 \\
& \quad 33\% \quad 1 \\
& \quad 66\% \quad 2 \\
& \quad 50\% \quad 1 \\
& \quad 50\% \quad 1 \\
& \quad 10\% \quad 1 \\
\end{align*}
Use of \texttt{glm()} in genetics

Odds are a [gambling-friendly] measure of chance;

\begin{itemize}
  \item AA: 66%
  \item Aa: 33%
  \item aa: 50%
\end{itemize}

Prob of survival, 10 yrs

\begin{itemize}
  \item 0.0: 90%
  \item 0.2: 50%
  \item 0.4: 50%
  \item 0.6: 10%
  \item 0.8: 2
  \item 1.0: 1
\end{itemize}

Odds(death)

\begin{itemize}
  \item 0.0: 1
  \item 1.0: 9
\end{itemize}

Prob(death)

- so what are odds ratios?
Use of `glm()` in genetics

Using the data from slide 4.12:

```r
> genpred2 <- factor(g) # the 2df model
> glm1 <- glm( dead10yrs ~ genpred2, family=binomial)
> coef(glm1)

   pred2Aa  pred2aa
0.6931   2.8904
```

- These are log odds ratio estimates; to transform to OR, use $e^{0.6931} = 2, e^{2.8904} = 18$

- They are given relative to the baseline group – ‘AA’ in this case

- Don’t forget the `family=binomial` argument!
Use of `glm()` in genetics

Confidence intervals, p-values as with `lm()`, for the log odds ratios;

```r
> confint.default(glm1)
            2.5 %      97.5 %
genpred2Aa  0.1201986  1.2660957
ngenpred2aa 2.1148912  3.6658523
```

```r
> summary(glm1)

    Estimate Std. Error   z value  Pr(>|z|)   
genpred2Aa 0.6931  0.2923   2.371     0.01773 *
genpred2aa 2.8904  0.3957   7.305   2.77e-13 ***
```

Use `exp()` to get odds ratio estimates, intervals; p-values are scale-independent
The formula syntax

We saw `lm(y \sim \text{genetic.predictor})` and `glm(y \sim \text{genpred2})`. To see how phenotype depends on several covariates, we specify e.g.

\[ y \sim \text{genotype.pred} + \text{age} + \text{sex} \]

– formally, this gives multivariate regression; the genotype.pred coefficients reflect the genotype effects adjusted for age and sex

- Separate covariates with ‘+’. This is not addition!
- For now, make predictor variables first, then do regression. It’s possible to do everything in one step, but use of e.g. ‘+’ will confuse R – unless you’re careful.
- For keen people; in the formula syntax, * indicates that interactions should be fitted, I() insulates mathematical operations, -1 removes the intercept... see ?formula
- For very keen people; `vcovHC()` in the sandwich package provides ‘robust’ standard errors; `coeftest()` in the lmtest package can use them to give ‘robust’ tests.
More extractor functions

We saw that the point estimates can be extracted using either;

- `my.lm$coefficients` or `my.lm$coeff`, i.e. the coefficients attribute of the `my.lm` object
- `coef(my.lm)` or `coefficients(my.lm)`

Many statisticians are familiar with `lm` and `glm` objects, so prefer the first version. But using ‘extractor functions’ makes the code easier to read, more portable, and more robust to internal changes. More are below; see also `?influence.measures`

- `predict()`; predicted values at given covariates
- `fitted.values()`; fitted values for original data
- `residuals()`; residuals for original data
- `confint.default()`; see earlier slides
- `vcov()`; variance-covariance matrix for the point estimates
- `vcovHC()`; robust version – in the `sandwich` package
- `AIC()`, `BIC()`; An Information Criterion (and another one)