J. Bradley–Terry Model for Paired Preferences

1. Compare items pairwise
2. Try to compare them, using all results at the same time.
3. Use more information than just pairs
4. Model probability for item \( j \) beating \( k \) as
   \[
   \frac{\exp(\beta_j - \beta_k)}{1 + \exp(\beta_j - \beta_k)}
   \]
5. Adding a constant to each of the \( \beta \) keeps probabilities the same
   a. Hence one of the items must be taken as baseline
6. Note that \( P[j \text{ beats } k] = 1 - P[k \text{ beats } j] \), as it should.
7. \( \beta_k \) represents strength of item \( k \)
8. Can add intercept for “home team advantage”.
9. Model fit is same as for quasi-symmetry model
   A: 5.4.2–5.4.3

XI. Exact Methods

A. Contingency tables:

1. Model:
   a. \( X_{ij} \sim \mathcal{P}(\lambda_{ij}) \)
   b. \( \log(\lambda_{ij}) = \alpha_i + \beta_j + \gamma_{ij} \)
Lecture 11

c. \( \beta_j = 0, \gamma_{iJ} = \gamma_{IJ} = 0 \forall i, j \).

2. \( H_0: \gamma_{ij} = 0 \forall i, j \) vs. \( H_A: \gamma_{ij} \neq 0 \) for some \( i, j \).

3. Test statistics:
   a. Score statistic is Pearson \( \chi^2 \): \[ T = \sum_{i,j} \frac{(X_{ij} - \bar{X}_i \cdot \bar{X}_j)^2}{\bar{X}_i \cdot \bar{X}_j / \bar{X}_{..}} \]
   b. LR statistic
   c. Fisher’s statistic \( 1/P[X] \)

4. Remove effect of unknown parameters:
   a. Remove \( \alpha_i \) by conditioning on \( X_i \).
      i. Reduces \( I \times J \) independent Poisson variables to \( J \) independent multinomials, each with \( I \) bins.
      ii. \( X_i \) are exactly ancillary
      iii. Little loss due to discreteness
   b. Remove \( \beta_j \) by conditioning on \( X_{.j} \)
      i. Reduces \( J \) independent multinomials, each with \( I \) bins, to generalized geometric
      ii. Probabilities are \( \prod_{i=1}^I x_{i.}! \prod_{j=1}^J x_{.j}! / (x_{..}! \prod_{i=1}^I \prod_{j=1}^J x_{ij}!) \)
      iii. Violates conditionality principal: column totals are not ancillary
iv. Bigger problem: discreteness

5. Difference not on log scale
   a. $H_0 : \pi_2 = \pi_1 + \delta$ vs $H_A : \pi_2 > \pi_1 + \delta$
   b. Different from previous if $\delta \neq 0$
   c. Important practical problem: non-inferiority trial
   d. No parallel conditional analysis
      i. Not a canonical exponential family

\[
(1 - \pi_1)^{n_1 - x_1} (1 - \delta - \pi_1)^{n_2 - x_2} \pi_1^{x_1} (\delta + \pi_1)^{x_2} \Gamma(1 + n_1) \Gamma(1 + n_2) \\
\frac{\Gamma(1 + n_1 - x_1) \Gamma(1 + x_1) \Gamma(1 + n_2 - x_2) \Gamma(1 + x_2)}
\]

   ii. See the chapter we skipped

iii. Test statistic is

\[
Z = \frac{X_1/n_1 - X_2/n_2}{\sqrt{\tilde{\pi}_1 (1 - \tilde{\pi}_1)/n_1 + \tilde{\pi}_2 (2 - \tilde{\pi}_2)/n_2}}
\]

iv. Called $\delta$-projected $Z$ statistic

v. $\tilde{\pi}_1$ maximizing likelihood is given by a cubic equation

vi. Remove nuisance parameter effect using suprema
   - That is, note that for any common $\pi^0$ value,

\[
P_{\pi^0} \left[ \sup_{\pi \in [0,1]} P_{\pi} [Z \geq z] \right] \leq \sup_{\pi \in [0,1]} P_{\pi} [Z \geq z].
\]

vii. Convexity condition: If test rejects for $(x_1, x_2)$, then test rejects for more extreme $(x_1, x_2 + 1)$ and $(x_1 - 1, x_2)$.
Lecture 11

- Extend to \( p \)-values: \( p(x_1, x_2) \geq p(x_1, x_2 + 1) \) and \( p(x_1, x_2) \geq p(x_1 - 1, x_2) \)

viii. Implies that one need only look along boundary of alternative hypothesis for maximizer.

ix. That is, \( p \)-value is the same whether we test \( H_0 : \delta = \delta_0 \) vs \( H_A : \delta > \delta_0 \) or \( H_0 : \delta \leq \delta_0 \) vs \( H_A : \delta > \delta_0 \)

x. Heuristically, because rejection region probabilities become less

xi. Need to check for convexity: \( Z \) statistics, Fisher’s exact test all work.

xii. Can also phrase question in terms of relative risk \( \pi_2/\pi_1 \).

6. Computation

a. Either enumerate all tables, and calculate probabilities
   straight-forwardly, or

b. (Pagano and Halvorsen, 1981) calculate recursively

i. \[ P \left[ X_{11} = x_{11} | X_{1..}, X_{.1}, X_{..} \right] = \frac{x_{11}!x_{.1}!(x_{..} - x_{11})!(x_{..} - x_{1})!}{x_{1.}!x_{.1}!(x_{1} - x_{11})!(x_{1.} - x_{11})!(x_{..} - x_{1.} - x_{1} + x_{11})!} \]

ii. Probabilities do not depend on other aspects of conditioning event.

iii. Hence same expression holds for \( P \left[ X_{11} = x_{11} | X_{i..}, X_{.j} \forall i, j \right] \)
iv. \[ P \left[ X_{21} = x_{21} \mid X_{11} = x_{11}, X_{1}, X_{2}, X_{11}, X_{..} \right] = x_{2}! (x_{..} - x_{1} - x_{2.})! (x_{1} - x_{11})! (x_{..} - x_{2} - x_{1} + x_{11})! / \]
\[ (x_{..} - x_{1}!) x_{21}! (x_{1} - x_{11} - x_{21})! (x_{2} - x_{21})! \times (x_{..} - x_{1} - x_{1} - x_{2} + x_{11} + x_{21})! \]

v. More generally, split table into 9 bits, some possibly empty:
\[
\begin{pmatrix}
  \ast & \ast & \dagger \\
  \ast & X_{i,j} & \dagger \\
  \ast & \dagger & \dagger
\end{pmatrix}
\]

- Condition on all marginals of collapsed table, and on totals marked with \( \ast \).
- \( \dagger \) cell fixed by conditioning event as well
- Result is hypergeometric distribution from \( 2 \times 2 \) table
- Hence \( P \left[ X = x \mid X_{.j} = x_{.j} \forall j, X_{.i} = x_{.i} \forall i \right] = \prod_{i,j} p_{ij} \) for hypergeometric probabilities \( p_{ij} \).

A: 5.4.4–5.4.5

B. Logistic Regression (Hirji, Mehta, and Patel, 1987)

c. \( X_{j} \sim \text{Bin} \left( \exp(z_{j} \theta) / (1 + \exp(z_{j} \theta)), n_{j} \right) \)

i. Suppose \( n_{i} = 1 \)

d. \( T = Z^\top X \)

e. Probabilities \( \exp(t^\top \theta - \sum_{j} n_{j} \log(1 + \exp(z_{j} \theta))) c(t) \), for \( c(t) \) the number of \( x \) vectors with \( Z^\top x = t \).
Lecture 11

f. Conditional probabilities

i. \( T = (U, V) \), \( \theta = (\omega, \tau) \)

ii. \( P[V = v | U = u] = c(u, bv) \exp(v\tau) / \sum_v c(u, bv) \exp(v\tau) \)

g. So we need an algorithm to generate a list of \( v \) consistent with \( u \),
and to calculate \( c(u, v) \) for these \( u \)

h. Let

i. \( \Omega_i \) be sample space using observations \( 1, \ldots, i \), satisfying
conditioning statements

ii. \( c_i \) be counts of \( X_1, \ldots, X_i \) ensembles giving \( t \)

i. Note that

i. \( \Omega_1 = \{0, z_1\} \)

ii. \( c_1(0) = c_1(z_1) = 1 \).

iii. \( \Omega_i = \Omega_{i-1} \cup (z_i + \Omega_{i-1}) \)

- After removing duplicates

iv. \( c_i(t) = c_{i-1}(t) + c_{i-1}(t - z_i) \)

j. Collects all possible \( t \)

i. Excessive: we only need vectors consistent with conditioning event.

ii. Algorithm more efficient if we can eliminate from \( \Omega_i \) many
entries that can never satisfy conditioning event

iii. Easiest condition to implement: dump those if component gets too large or small.