Modeling Proportions and Probabilities: The beta distribution is your friend

Paul Teetor
William Blair & Co.

ASA Conference on Statistical Practice
Jacksonville, FL
February 2017
This talk is all about modeling data on the unit interval

- Proportions (muggle: percentages)
- Probabilities
- Continuous ranking (0.0, ..., 1.0)
- Rates, concentrations, etc.
The binomial distribution is a weak model of unit data.
The beta distribution adds expressive power

- Beta(2, 2)
- Beta(5, 2)
- Beta(0.5, 0.5)
- Beta(1, 1)
The beta distribution has two parameters

- $\alpha$: weight on $x$ (for $x \in [0, 1]$)
- $\beta$: weight on $1 - x$
- Richer density function than one-parameter binomial model

$$Beta(x|\alpha, \beta) \sim \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1 - x)^{\beta-1}$$

where $B$ is the beta function, providing the normalizing constant
An alternative parameterization uses mean and precision

- **Precision** is reciprocal of variance \( = 1/\sigma^2 \)
- Simple transformation between \( \alpha \) and \( \beta \) versus mean \( (\mu) \) and precision \( (\phi) \)

\[
\begin{align*}
\mu &= \alpha/(\alpha + \beta) \\
\phi &= \alpha + \beta
\end{align*}
\]

- Alternative parameterization simplifies regression math
How do we estimate the parameters from sample data?

These are the method of moments estimators.

\[ \hat{\alpha} = \bar{x} \left( \frac{\bar{x}(1 - \bar{x})}{\bar{v}} - 1 \right) \]

\[ \hat{\beta} = (1 - \bar{x}) \left( \frac{\bar{x}(1 - \bar{x})}{\bar{v}} - 1 \right) \]
For data, let’s use the Chicago Cubs historical record

<table>
<thead>
<tr>
<th>Season</th>
<th>PrWin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1874</td>
</tr>
<tr>
<td>2</td>
<td>1875</td>
</tr>
<tr>
<td>3</td>
<td>1876</td>
</tr>
<tr>
<td>4</td>
<td>1877</td>
</tr>
</tbody>
</table>

The diagram shows a histogram and a density plot of the probability of winning (PrWin) for the Chicago Cubs.
Parameter estimation on Cubs data

\[
M = \text{mean}(\text{cubs}\$PrWin); \ V = \text{var}(\text{cubs}\$PrWin)
\]

\[
\alpha = M \times \frac{(M \times (1 - M))}{V - 1}
\]

\[
\beta = (1 - M) \times \frac{(M \times (1 - M))}{V - 1}
\]

\[
\text{print}(\alpha); \ \text{print}(\beta)
\]

## [1] 15.04555

## [1] 14.10103
The estimated parameters give a parametric density

The parametric density nicely echoes the non-parametric density shown earlier.
Beyond simple fit: Regressors

What if we had a predictor? How would that influence the distribution?
Beta regression models responses with beta distributions

- Response has beta distribution, not normal
- Transform responses from (0,1) to \((-\infty, +\infty)\)

\[ p_i' = \logit(p_i) \]

where \( \logit(x) = \log\left(\frac{x}{1-x}\right) \)

- Then regress on \( p_i' \)

\[ p_i' = \beta_0 + \beta_1 x_i + \varepsilon_i \]

- A form of generalized linear model (GLM), usually expressed like this

\[ \logit(p_i) = \beta_0 + \beta_1 x_i + \varepsilon_i \]
The \textit{logit} function mediates between predictors and dependent variable

\begin{itemize}
    \item Linear predictors range over \(( -\infty, +\infty )\)
    \item Dependent variable ranges over bounded \(( 0, 1 )\)
    \item Logit link function “expands” dependent into \(( -\infty, +\infty )\)
\end{itemize}
Could this year’s success predict next year’s success?
Add column to data: next year’s PrWin

<table>
<thead>
<tr>
<th>#</th>
<th>Season</th>
<th>PrWin</th>
<th>PrWinNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1874</td>
<td>0.475</td>
<td>0.448</td>
</tr>
<tr>
<td>2</td>
<td>1875</td>
<td>0.448</td>
<td>0.788</td>
</tr>
<tr>
<td>3</td>
<td>1876</td>
<td>0.788</td>
<td>0.441</td>
</tr>
<tr>
<td>4</td>
<td>1877</td>
<td>0.441</td>
<td>0.500</td>
</tr>
<tr>
<td>5</td>
<td>1878</td>
<td>0.500</td>
<td>0.582</td>
</tr>
</tbody>
</table>
Beta regression in R is pretty easy

```r
install.packages("betareg")  # One-time install

library(betareg)

This example takes the data from the cubs data frame.

m = betareg(PrWinNext ~ PrWin, data=cubs)

The returned model, m, includes $\beta_0$ and $\beta_1$. 
Beta regression: the Cubs data

```R
library(betareg)
m = betareg(PrWinNext ~ PrWin, data=cubs)
print(m)
```

```r
## Call:
betareg(formula = PrWinNext ~ PrWin, data = cubs)
##
## Coefficients (mean model with logit link):
## (Intercept)  PrWin
##    -1.059    2.191

## Phi coefficients (precision model with identity link):
##  (phi)
##   39.76
```
In typical R fashion, `summary` gives the details (1/2)

```r
summary(m)
```

##
## Call:
## betareg(formula = PrWinNext ~ PrWin, data = cubs)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -2.8089 -0.6272 -0.1244  0.6003  4.3545
##
## Coefficients (mean model with logit link):
##            Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -1.0589     0.1546  -6.847   7.53e-12
## PrWin       2.1912     0.2964   7.393  1.43e-13
## PrWin ***
##
## Phi coefficients (precision model with identity link):
##
## Estimate Std. Error z value Pr(>|z|)  
## (phi) 39.758 4.661 8.531 <2e-16 ***
##
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 161.1 on 3 Df  
## Pseudo R-squared: 0.2792  
## Number of iterations: 15 (BFGS) + 3 (Fisher scoring)
Forecast from the model using predict

- Create a data frame, `pred`, containing the new predictor values.
- Use `predict` to feed new predictors into model.
- The `type` argument controls what's returned:
  ```r
  predict(m, newdata=pred, type="response")
  predict(m, newdata=pred, type="variance")
  predict(m, newdata=pred, type="quantile", at=c(0.025, 0.975))
  ```

- `predict` handles details of link transformation.
Let’s forecast 2017 performance using predict

```r
pred = data.frame(PrWin = 0.640)  # Record for 2016
predict(m, newdata=pred)          # Expectation for 2017

## 1
## 0.585039

sqrt(predict(m, newdata=pred, type="variance"))

## 1
## 0.07717715
```
Let’s forecast 2017 performance using predict (con’t)

```r
predict(m, newdata=pred, type="quantile",
at=c(0.025, 0.975))
```

```r
## q_0.025 q_0.975
## [1,] 0.4306028 0.731366
```
The regression yields a conditioned beta distribution
The `betareg` package provides other functions, too

```r
methods(class="betareg")
```

```r
## [1] coef        cooks.distance gleverage
## [4] hatvalues  logLik      model.frame
## [7] model.matrix plot       predict
## [10] print      residuals   summary
## [13] terms      update      vcov
## see '?methods' for accessing help and source code
```

Methods for `tidy`, `augment`, and `glance` available from `broom package`
Recall that precision is inverse of variance $(1/\sigma^2)$
Model the precision parameter ($\phi$) with a second GLM
Example: suppose $z_i$ predicts variance of $p_i$

\[
\text{logit}(\mu_i) = x_i^T \beta \\
\log(\phi_i) = z_i^T \gamma
\]

 Incorporates heteroskedasticity into your model
 Small downside: Residuals will be heteroskedastic, so use 
\textit{standardized weight residuals}
Typical case: Heteroskedasticity at extremes of the unit interval

- In my experience, variance can change from midpoint to end-points
- Suggests modeling $\phi_i$
- Select $z_i$ best able to model precision, $\phi_i$
Heteroskedasticity illustration

Score

Bucket

MAD
So, why doesn’t the *binomial* model work?

- Using a simple point estimate of $p$ in $\text{Binom}(N, p)$ is naive.
- Does not reflect the uncertainty of the estimate.
- Tails of distr. too small

For 2017: $\text{Binom}(N=162, p=0.64) / 162$
Improve the model by acknowledging the undercertainty

- Model $p$ as a random variable, imperfect estimate
- Still parameter of binomial distribution, but now stochastic
- The result is called a *Beta-Binomial* model

\[ PrWin \sim Binom(N, p) \]

where

\[ p \sim Beta(\alpha, \beta) \]
Q: What’s the distribution of $p$?

- Remember, $p$ has a beta distribution.
- Let’s estimate mean of $p$ from 2017 record (0.64)
- Estimate variance of $p$ from full history (0.008284)
- From mean and variance, calculate $\alpha$ and $\beta$ of beta distr.
Beta-binomial distribution for 2017 is more realistic

\[ N = 10000 \]

\[ \text{prob} = rbeta(N, \alpha, \beta) \]

\[ \text{sampWins} = rbinom(N, \text{size}=\text{SIZE}, \text{prob} = \text{prob}) \]

\[ \text{sampProbs} = \text{sampWins} / \text{SIZE} \]

For 2017: \( \text{Binom}(162, p) / 162 \) where \( p \sim \text{Beta}(17.16, 9.65) \)
The `gamlss` package can fit a beta-binomial model

```r
install.packages("gamlss")
```

Beta-binomial model with two predictors, `pred1` and `pred2`:

```r
library(gamlss)
m = gamlss(resp ~ pred1 + pred2,
            family = BB, data = yourData)
```

Add a variance predictor, `pred3` (for heteroskedasticity):

```r
m = gamlss(resp ~ pred1 + pred2, sigma.formula = ~pred3,
            family = BB, data = yourData)
```
Which is better?

Beta regression:

- Easy. *Hey, there’s an R package!*
- Resembles familiar linear regression
- Tools to compare models: log-likelihood, pseudo $R^2$

Beta-binomial:

- Bayesian, so you get explicit probability distributions, credible intervals
- *gamlss* package does the heavy lifting
- Straight-forward simulation or Monte Carlo
Links to resources

- *betareg* package: https://cran.r-project.org/package=betareg
- *betareg* vignette: https://cran.r-project.org/web/packages/betareg/vignettes/betareg.pdf
- *gamlss* package: https://cran.r-project.org/package=gamlss
- Dave Robinson’s excellent beta-binomial tutorial: http://varianceexplained.org/r/beta_binomial_baseball/