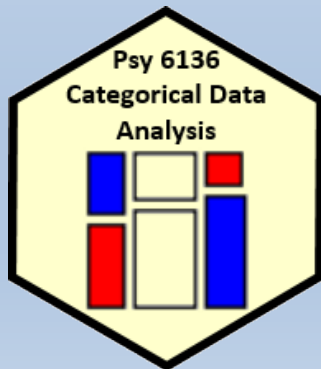
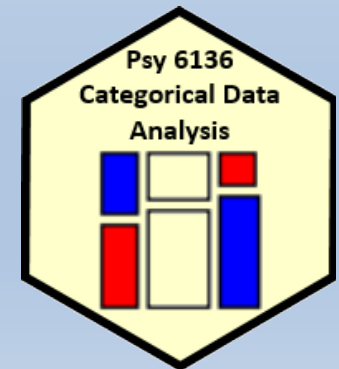


GLMs for Count Data



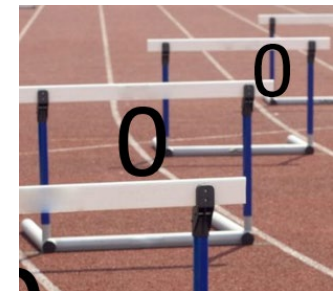
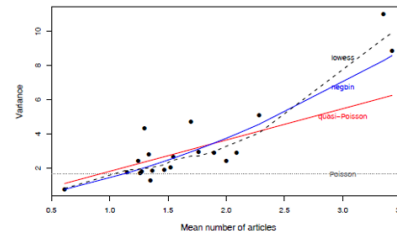
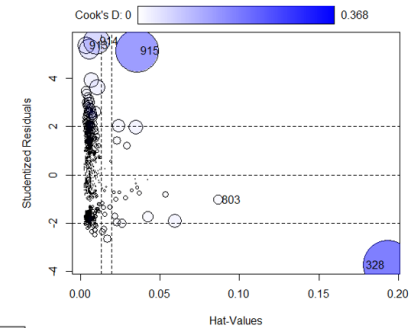
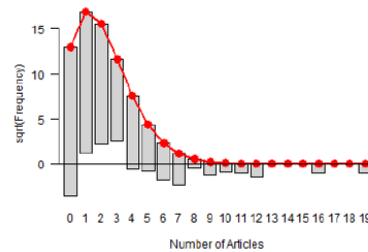
Michael Friendly
Psych 6136

<https://friendly.github.io/psy6136>



Topics

- Generalized linear models
- GLMs for count data
 - Example: PhD publications
- Model diagnostics
 - Interactions
 - Nonlinearity
 - Outliers, leverage & influence
- Overdispersion
 - Quasi-poisson models
 - Negative binomial models
- Excess zeros
 - Zero-inflated models
 - Hurdle models
- Counted proportions: Beta binomial model



Count data models: Overview

- **Count data** models arise when the basic observation is a frequency, $y = 0, 1, 2, \dots$ of some event and we have some predictors, x_1, x_2, \dots to help explain them.
 - Typically, these counts $\sim \text{Poisson}()$ → “poisson regression”
- **Examples:**
 - Number of articles published by PhD candidates
 - Predictors: Married?, Female?, Kids < 5?, pubs by mentor
 - Number of parasites in blood samples of Norwegian cod
 - Predictors: Catch area, Year, length of fish
 - Female horseshoe crabs: Number of “satellite” males
 - Predictors: Female weight, color, spine condition, shell width
- **Special circumstances**
 - **Overdispersion:** when the variance > mean
 - **Zero-counts:** When excess 0 counts require an extra model



Generalized linear models

We have used **generalized** linear models fit with `glm()` in two contexts so far

Loglinear models

- the outcome variable is the **vector of frequencies** \mathbf{y} in a table cross-classified by factors in a design matrix \mathbf{X}
- The model is expressed as a linear model for $\log \mathbf{y}$

$$\log(\mathbf{y}) = \mathbf{X}\beta$$

- The random (or unexplained) variation is expressed as a Poisson distribution for $\mathcal{E}(\mathbf{y} | \mathbf{X})$

Hmm. Isn't the problem with frequency data just that of **non-constant variance**?

Questions:

- Why not just transform $y \rightarrow \log(y)$ and use standard OLS regression?
- Why should I bother with Poisson anyway? He wasn't even **NORMAL !**

Generalized linear models

Logistic regression

- the outcome variable is a **categorical response** \mathbf{y} , with predictors \mathbf{X}
- The model is expressed as a linear model for the log odds that $y = 1$ vs. $y = 0$.

$$\text{logit}(\mathbf{y}) \equiv \log \left[\frac{\Pr(y = 1)}{\Pr(y = 0)} \right] = \mathbf{X}\boldsymbol{\beta}$$

- The random (or unexplained) variation is expressed as a Binomial distribution for $\mathcal{E}(\mathbf{y} | \mathbf{X})$

Hey, aren't these both very like the familiar, classical linear model,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \quad ?$$

Yes, for some transformation, $g(\mathbf{y})$, and with different distributions!

 $g(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\mathcal{E}} \quad \boldsymbol{\mathcal{E}} \sim \text{Bin}() \parallel \text{Pois}() \parallel \text{Nbin}() \parallel \dots$

Generalized linear models

Nelder & Wedderburn (1972) said, “Let there be light!”, a *generalized linear model*, encompassing them all, and many more. This has 3 components:



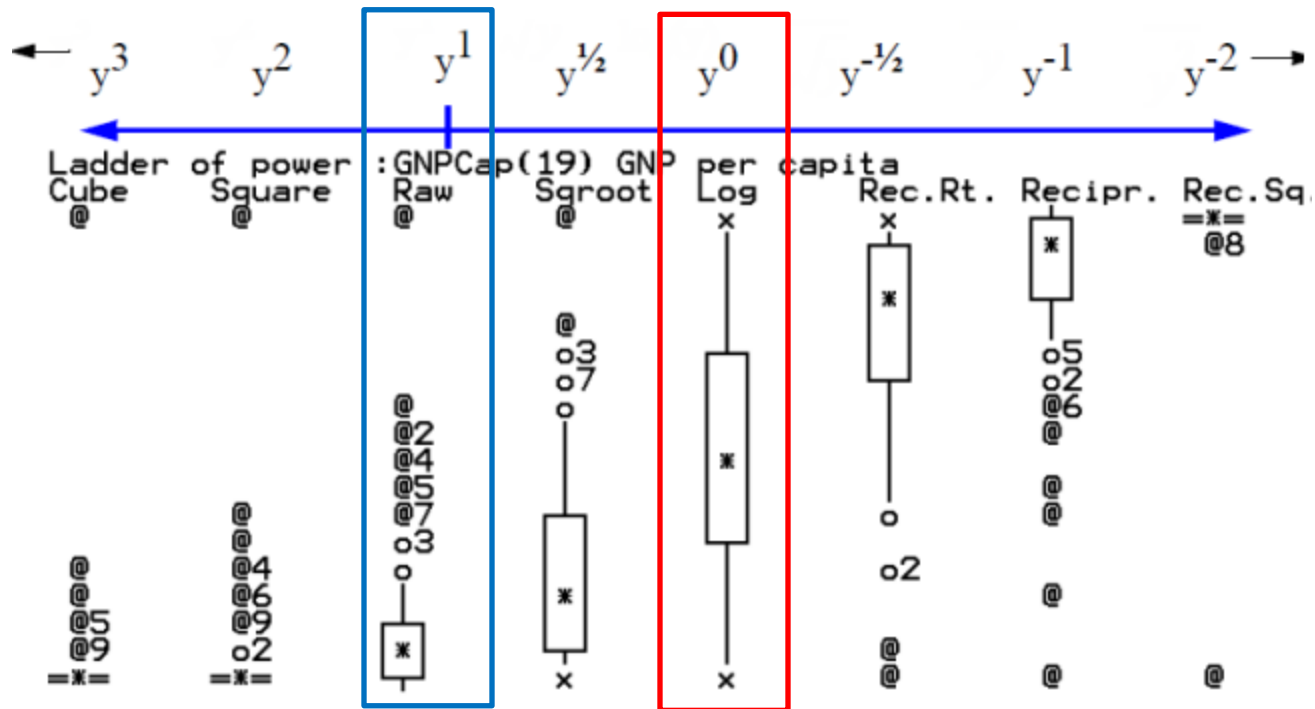
- A **random component**, specifying the conditional distribution of \mathbf{y} given the explanatory variables in \mathbf{X} , with mean $\mathcal{E}(y_i | \mathbf{x}_i) = \mu_i$
 - The normal (Gaussian), binomial, and Poisson are already familiar
 - But, these are all members of an **exponential family**
 - GLMs now include an even wider family: negative-binomial and others
- The **systematic component**, a linear function of the predictors called the **linear predictor**

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} \quad \text{or} \quad \eta_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip}$$

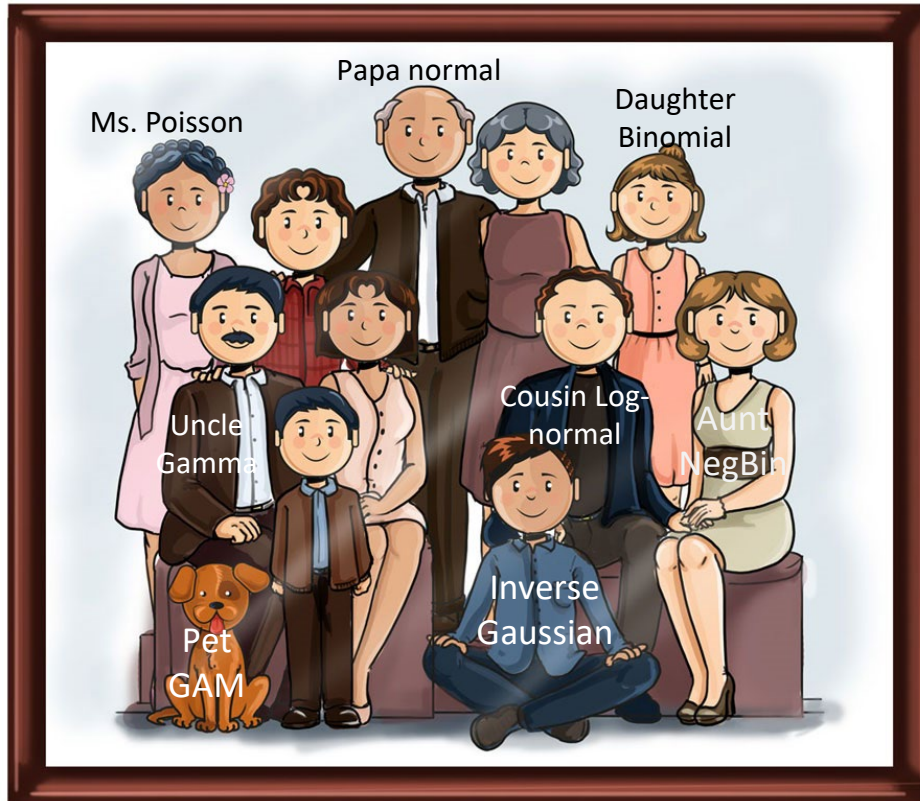
- An invertible **link function**, $g(\mu_i) = \eta_i = \mathbf{x}_i^T \boldsymbol{\beta}$ that transforms the expected value of the response to the linear predictor
 - The link function is invertible, so we can go back to the **mean function**
 $g^{-1}(\eta_i) = \mu_i$

GLMs: The light

- No need to consider all those special cases to transform y for homogeneity of variance
 - EDA approach: ladder of powers, transform to symmetry



GLMs: Families



All GLMs are members of a big happy family

They have different technical names, but all share common DNA – The Exponential Family includes direct descendants, uncles, cousins, ...

They all have a linear predictor,

$$\eta = g(\mu) = X \beta$$

They differ in their links: how to transform from $\mu \rightarrow g(\mu) = \eta$

They can get back to their roots with an inverse transformation,

$$g^{-1}(\eta) = \mu$$

Quasi-cousins: QuasiPoisson, QuasiBinomial
Pets: Allow flexible non-linearities

Link functions for the mean

Standard GLM link functions and their inverses:

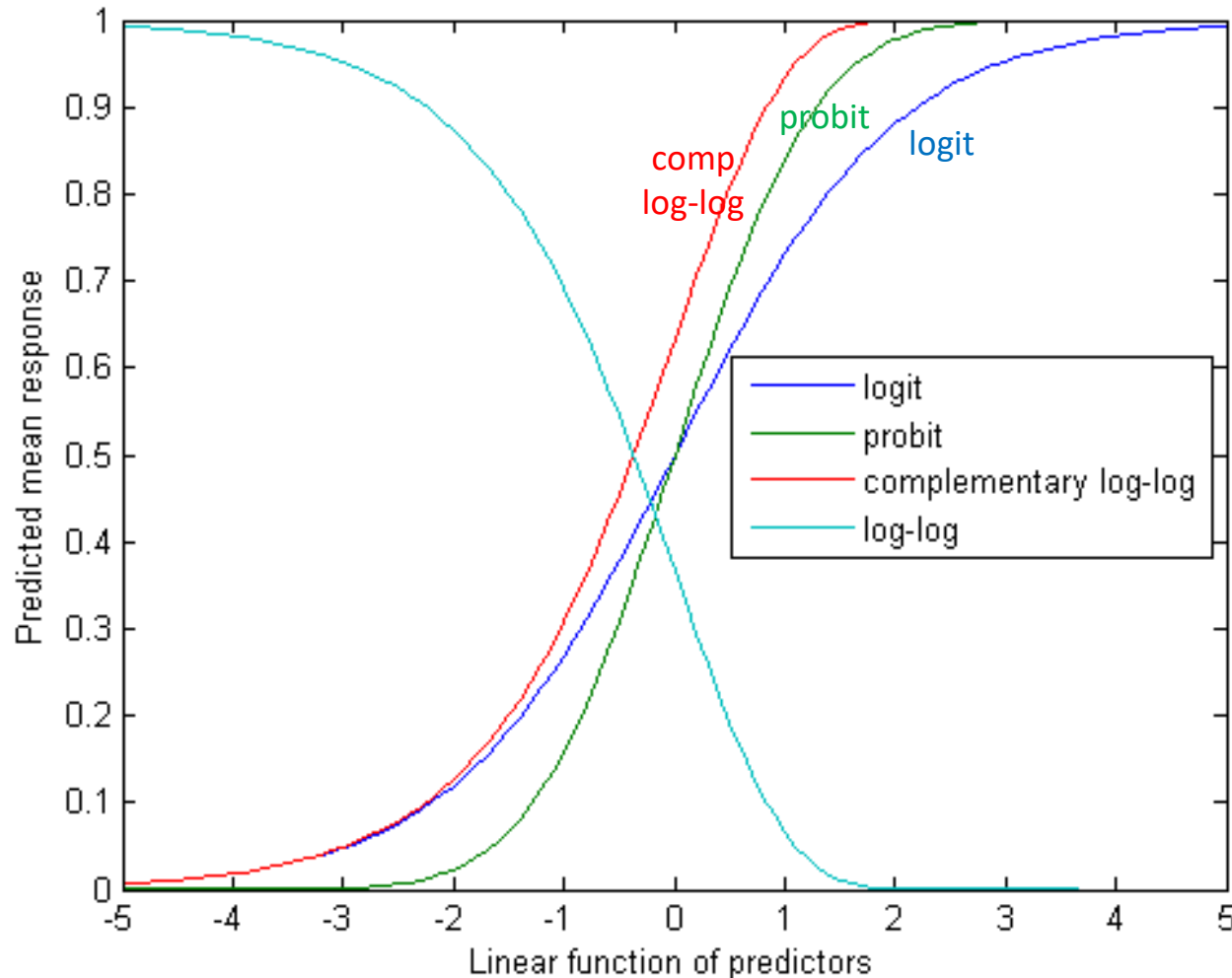
Table 11.1: Common link functions and their inverses used in generalized linear models

	Link name	Function: $\eta_i = g(\mu_i)$	Inverse: $\mu_i = g^{-1}(\eta_i)$
quantitative	identity	μ_i	η_i
	square-root	$\sqrt{\mu_i}$	η_i^2
	log	$\log_e(\mu_i)$	$\exp(\eta_i)$
	inverse	μ_i^{-1}	η_i^{-1}
	inverse-square	μ_i^{-2}	$\eta_i^{-1/2}$
binomial	logit	$\log_e \frac{\mu_i}{1-\mu_i}$	$\frac{1}{1+\exp(-\eta_i)}$
	probit	$\Phi^{-1}(\mu_i)$	$\Phi(\eta_i)$
	log-log	$-\log_e[-\log_e(\mu_i)]$	$\exp[-\exp(-\eta_i)]$
	comp. log-log	$\log_e[-\log_e(1-\mu_i)]$	$1 - \exp[-\exp(\eta_i)]$

The link function must be **invertible** e.g., $|\mu|$ is not

- The top section recognizes standard **transformations** of y_i often used with **classical** linear models to stabilize variance: y , $y^{1/2}$, $\log y$, y^{-1} , ...
- The bottom section is for **binomial** data, where y_i represents an observed count in n_i trials

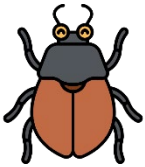
Link functions for binomial data



For binomial data, the logit, probit and c-log-log all have similar shapes

These take a linear predictor on $(-\infty, +\infty)$ to the range $(0,1)$ for probability

The logit is most widely used because of its' simple interpretation as log odds



Example: BeetleMortality

Mortality of adult flour beetle after five hours' exposure to gaseous carbon disulphide.

```
> data("BeetleMortality",
      package = "glmx")
> BeetleMortality
  dose died  n
1 1.6907   6 59
2 1.7242  13 60
3 1.7552  18 62
4 1.7842  28 56
5 1.8113  52 63
6 1.8369  53 59
7 1.8610  61 62
8 1.8839  60 60
```

In this example, the cloglog link fits best

The 'lapply()' trick: Apply a function w/ parameters → a list of fitted models

```
links <- c("logit", "probit", "cloglog")
m <- lapply(links, function(type)
  glm(cbind(died, n - died) ~ dose,
      data = BeetleMortality,
      family = binomial(link = type)))
names(m) <- links
```

Then, sapply() LRStats() to each model:

```
> t(sapply(m, vcdExtra::LRstats))
      AIC  BIC  LR Chisq Df Pr(>Chisq)
logit  41.4 41.6 11.2   6  0.0815
probit  40.3 40.5 10.1   6  0.12
cloglog 33.6 33.8  3.45  6  0.751
```

Visualize model fits

```
plot(I(died/n) ~ dose, data = BeetleMortality,  
     ylab = "Proportion died", cex.lab = 1.2, pch = 16)  
lines(fitted(m[[1]]) ~ dose, data = BeetleMortality, col = 2, lwd = 2)  
lines(fitted(m[[2]]) ~ dose, data = BeetleMortality, col = 3, lwd = 2)  
lines(fitted(m[[3]]) ~ dose, data = BeetleMortality, col = 4, lwd = 2)  
legend(1.81, 0.4,  
       title = "Link", legend = links,  
       col = 2:4, lty = 1, lwd=2)
```

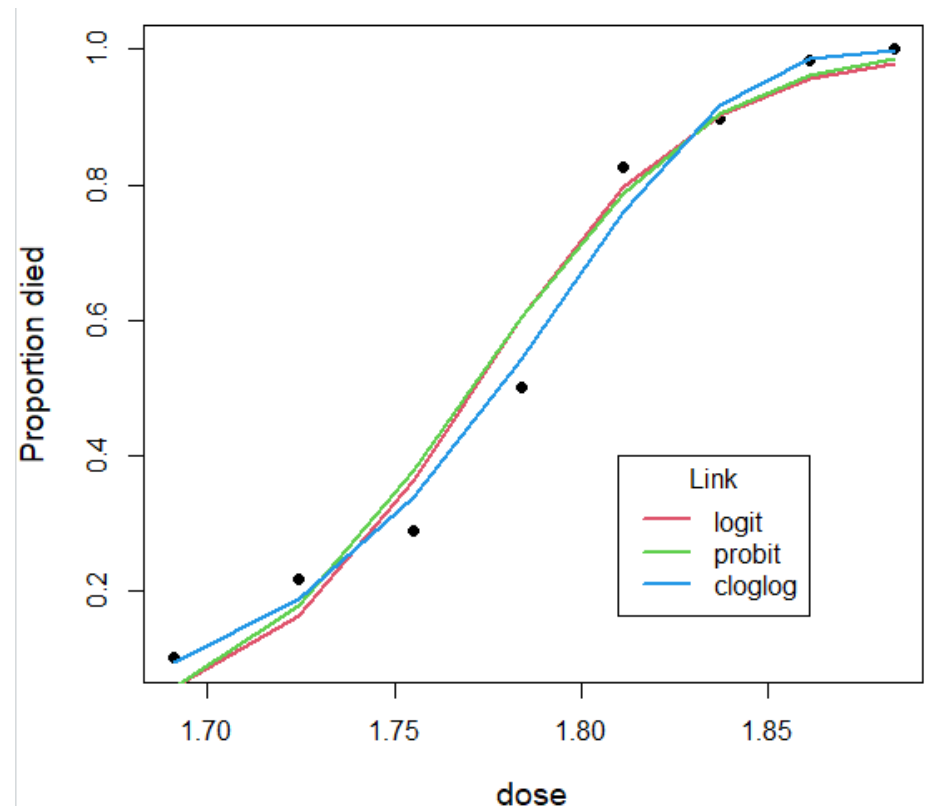
Plot data points

Add fitted lines

We can sort of see why the cloglog link fits best (subtle!)

But the coefficients in the model do not have as clear an interpretation as log odds in the logit model.

```
> t(sapply(m, coef))  
      (Intercept) dose  
logit      -60.7  34.3  
probit     -34.9  19.7  
cloglog    -39.6  22.0
```



Canonical links and variance functions

- For every distribution family, there is a default, **canonical link** function
- Each one also specifies the expected relation between the mean and **variance**

Table 11.2: Common distributions in the exponential family used with generalized linear models and their canonical link and variance functions

Family	Notation	Canonical link	Range of y	Variance function, $\mathcal{V}(\mu \eta)$
Gaussian	$N(\mu, \sigma^2)$	identity: μ	$(-\infty, +\infty)$	ϕ
Poisson	$\text{Pois}(\mu)$	$\log_e(\mu)$	$0, 1, \dots, \infty$	μ
Negative-Binomial	$\text{NBin}(\mu, \theta)$	$\log_e(\mu)$	$0, 1, \dots, \infty$	$\mu + \mu^2 / \theta$
Binomial	$\text{Bin}(n, \mu) / n$	$\text{logit}(\mu)$	$\{0, 1, \dots, n\} / n$	$\mu(1 - \mu) / n$
Gamma	$G(\mu, \nu)$	μ^{-1}	$(0, +\infty)$	$\phi\mu^2$
Inverse-Gaussian	$IG(\mu, \nu)$	μ^2	$(0, +\infty)$	$\phi\mu^3$

Choose a basic family:

- Get a default, canonical link, $g(\mu)$
- Also get a variance function for free!

Variance functions & overdispersion

- In the classical Gaussian linear model, the conditional variance is constant, $\phi = \sigma_\epsilon^2$.
- For binomial data, the variance function is $\mathcal{V}(\mu_i) = \mu_i(1 - \mu_i)/n_i$, with ϕ fixed at 1
- In the Poisson family, $\mathcal{V}(\mu_i) = \mu_i$ and the dispersion parameter is fixed at $\phi = 1$.
- In practice, it is common for count data to exhibit **overdispersion**, meaning that $\mathcal{V}(\mu_i) > \mu_i$.
- One way to correct for this is to allow the dispersion parameter to be estimated from the data, giving what is called the **quasi-Poisson** family, with $\mathcal{V}(\mu_i) = \hat{\phi}\mu_i$.

What is overdispersion?

Overdispersion often results from **failures of assumptions** of the model

- Supposedly independent observations may be **correlated**
- The probability of an event may not be **constant**, or
- It may vary with unmeasured or **unmodeled** variables

Don't fear overdispersion – embrace (& understand!) it

- For Poisson (freq) data, parameter estimates are **unchanged**; it affects only the **std. errors** (& z-tests)
- It tells you something interesting about your data or analysis
- Can lead to better understanding of your model: What did I leave out?

Maximum likelihood estimation

- GLMs are fit by the method of **maximum likelihood**
 - Likelihood (L) = Pr (data | model), as function of model parameters
- For the **Poisson** distribution with mean μ , the probability that the random variable Y takes the values $y = 0, 1, 2, \dots$ is

$$\Pr(Y = y) = \frac{e^{-\mu} \mu^y}{y!}$$

- In the GLM with a **log link**, the mean, μ , depends on the predictors through

$$\log_e(\mu_j) = \mathbf{x}_j^T \boldsymbol{\beta}$$

Maximum likelihood estimation

- The log-likelihood function is the probability of the data as a function of the parameters, β . It has the form (for Poisson)

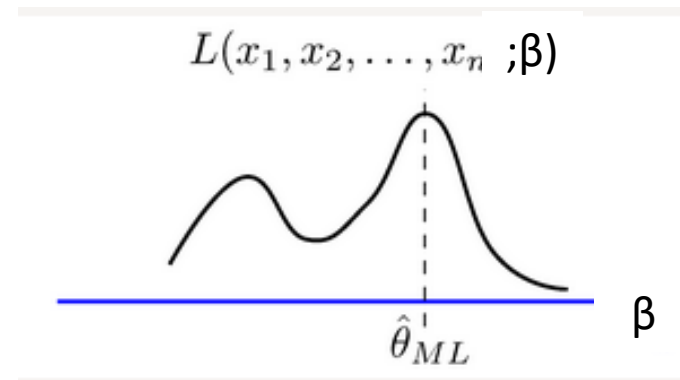
$$\log_e \mathcal{L}(\beta) = \sum^n \{y_i \log_e(\mu_i) - \mu_i\}$$

Why log \mathcal{L}

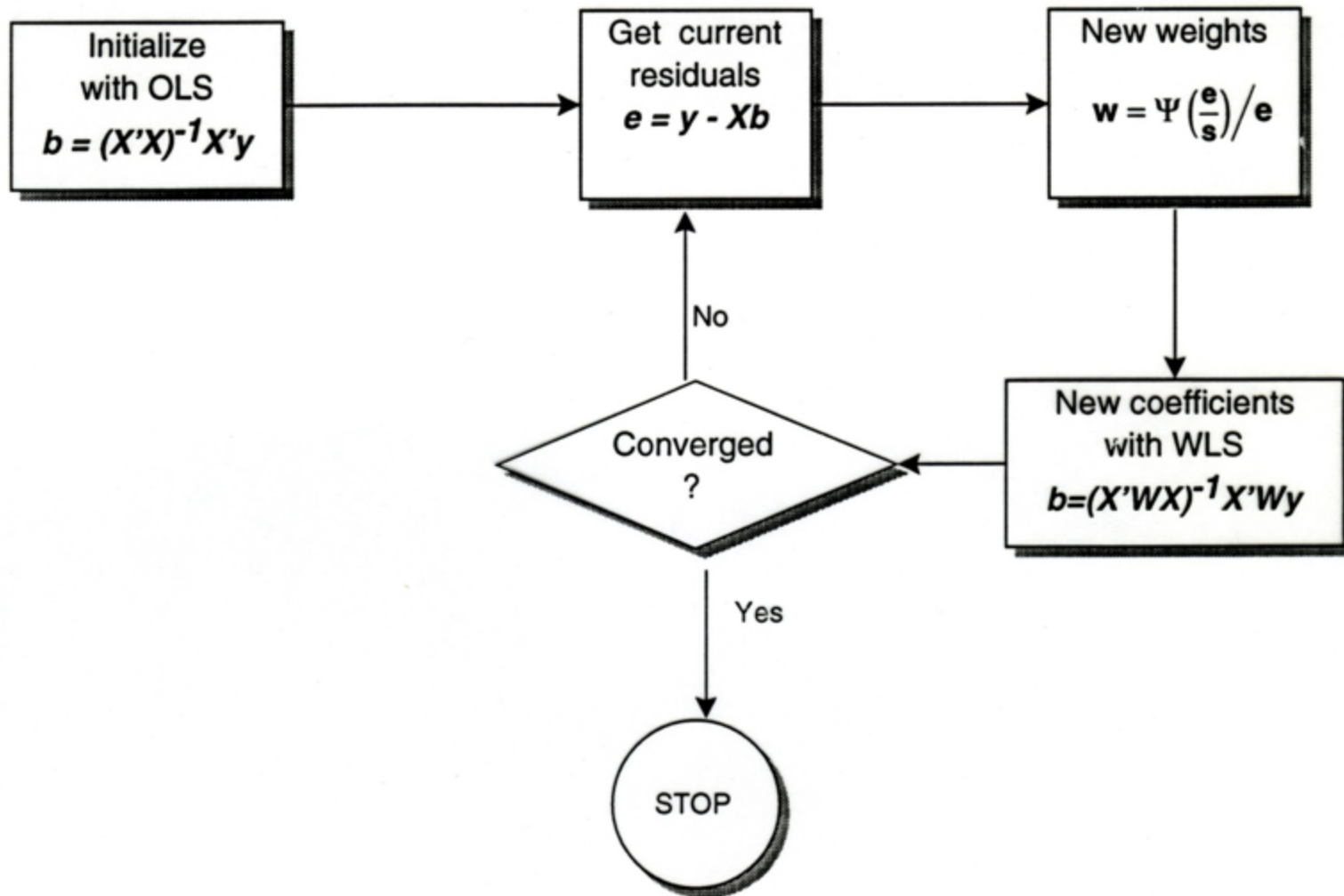
- Easier to work with
- Has the same max value

- Then, find the values of β that maximize log \mathcal{L}

Unlike OLS, where there is an exact solution, MLEs are found by **iteratively reweighted least squares**.



Iteratively reweighted least squares



Goodness of fit

- The **residual deviance** defined as twice the difference between the maximum log-likelihood for the *saturated model* that fits perfectly and maximized log-likelihood for the fitted model.

$$D(\mathbf{y}, \hat{\boldsymbol{\mu}}) \equiv 2[\log_e \mathcal{L}(\mathbf{y}; \mathbf{y}) - \log_e \mathcal{L}(\mathbf{y}; \hat{\boldsymbol{\mu}})] .$$

- For classical (Gaussian) linear models, this is just the **residual sum of squares**
- For Poisson models with a log link giving $\boldsymbol{\mu} = \exp(\mathbf{x}^T \boldsymbol{\beta})$, the deviance takes the form

$$D(\mathbf{y}, \hat{\boldsymbol{\mu}}) = 2 \sum_{i=1}^n \left[y_i \log_e \left(\frac{y_i}{\hat{\mu}_i} \right) - (y_i - \hat{\mu}_i) \right] .$$

- For a GLM with p parameters, both the Pearson and residual deviance statistics follow approximate χ_{n-p}^2 distributions with $n - p$ degrees of freedom.

GLMs for count data

- Typically, these are fit using

```
glm(y ~ x1 + x2 + ..., family=poisson, data=mydata)
```

- As in other linear models, the predictors, x_i , can be discrete factors, quantitative variables, interactions, etc.
- This fixes the dispersion parameter, ϕ to 1, assuming the count variable $y | x_1, x_2, \dots$ is Poisson distributed
- It is possible to relax this, and fit a quasi-Poisson model, allowing ϕ to be estimated from the data
 - Specify `family=quasipoisson`. This allows variance to be proportional to the mean

$$\mathcal{V}(y_i | \eta_i) = \phi \mu_i$$

- Another possibility is the `negative-binomial` model, which has

$$\mathcal{V}(y_i | \eta_i) = \mu_i + \mu_i^2 / \theta$$

Example: Publications of PhD candidates

Example 3.24 in DDAR gives data on the number of publications by PhD candidates in biochemistry in the last 3 years of study

```
> data("PhdPubs", package = "vcdExtra")  
> table(PhdPubs$articles)
```

0	1	2	3	4	5	6	7	8	9	10	11	12	16	19
275	246	178	84	67	27	17	12	1	2	1	1	2	1	1

Predictors are:

- gender, marital status
- number of young children < 5 yo
- prestige of the doctoral department
- number of publications by the student's mentor

Q: Which of these do you think would have strong effects on pubs?

Example: Publications of PhD candidates

Initially, ignore the predictors

This is equivalent to an intercept-only Poisson model

```
glm(articles ~ 1, family=poisson, data = PhdPubs)
```

As a check on the Poisson assumption, calculate the mean and variance

```
> with(PhdPubs, c(mean=mean(articles),  
                  var=var(articles),  
                  ratio=var(articles)/mean(articles)))
```

mean	var	ratio
1.69	3.71	2.19

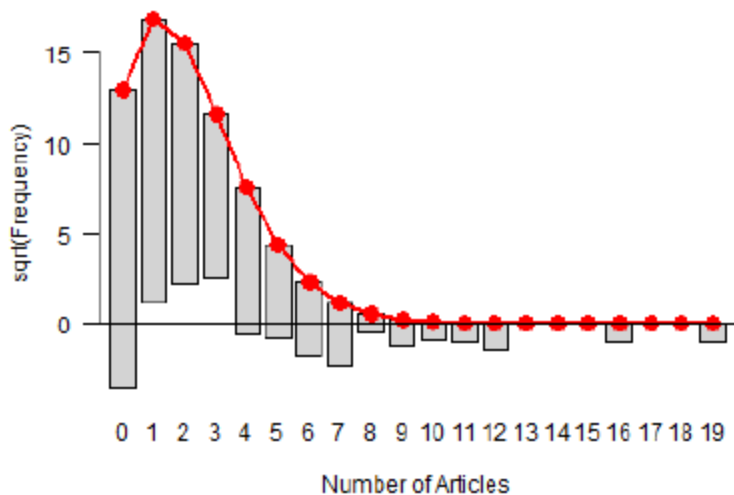
Oops! The assumption that mean = variance could be met when we add predictors

R code for this example: <https://friendly.github.io/psy6136/R/phdpubs.R>

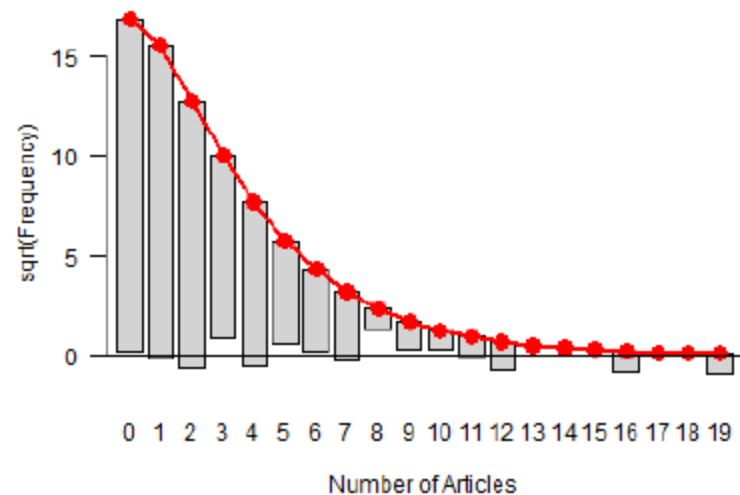
First, look at rootograms:

```
plot(goodfit(PhdPubs$articles), xlab = "Number of Articles",  
     main = "Poisson")  
plot(goodfit(PhdPubs$articles, type = "nbinomial"),  
     xlab = "Number of Articles", main = "Negative binomial")
```

Poisson



Negative binomial



One reason the Poisson doesn't fit: excess 0s (some never published?)

Q: What might some other reasons be?

Think back to assumptions: independent obs; constant probs; unmodelled vars

Fitting the Poisson model

Fit the model with **all main effects** (NB: `~ .` notation)

```
> phd.pois <- glm(articles ~ ., data=PhdPubs, family=poisson)
> Anova(phd.pois)
Analysis of Deviance Table (Type II tests)
```

Response: articles

	LR	Chisq	Df	Pr(>Chisq)	
female	17.1	1	3.6e-05	***	
married	6.6	1	0.01	*	
kid5	22.1	1	2.6e-06	***	
phdprestige	1.0	1	0.32		
mentor	126.8	1	< 2e-16	***	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Model fitting intelligence:

- Only **phdprestige** is NS; it does no harm to keep it, for now
- Maybe important to control / adjust for?
- Maybe it interacts w/ something else?

Interpreting coefficients

β_j is the **increment** in log (articles) for a 1 unit change in x_j ; $\exp(\beta_j)$ is the **multiple** of articles:

```
round(cbind(beta = coef(phd.pois),  
            expbeta = exp(coef(phd.pois)),  
            pct = 100 * (exp(coef(phd.pois)) - 1)), 3)
```

```
##          beta expbeta    pct  
## (Intercept)  0.266   1.304  30.425  
## female1     -0.224   0.799 -20.102  
## married1    0.157   1.170  17.037  
## kid5        -0.185   0.831 -16.882  
## phdprestige  0.025   1.026   2.570  
## mentor      0.025   1.026   2.555
```

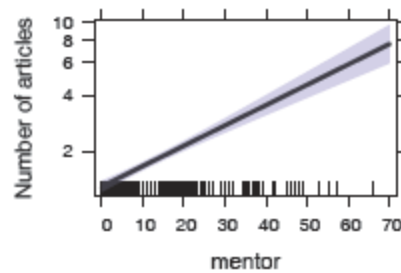
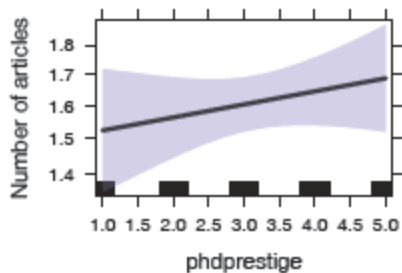
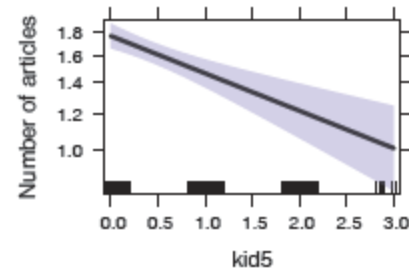
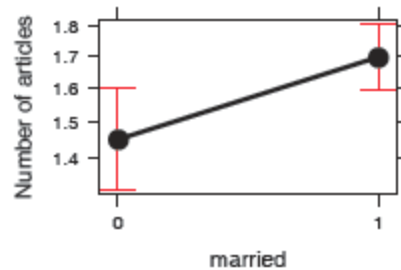
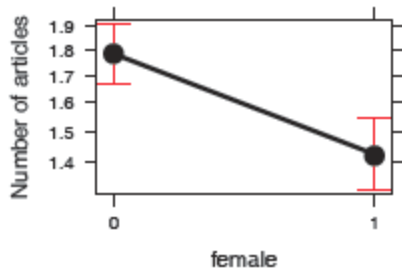
Thus:

- females publish -0.224 fewer log (articles), or $0.8 \times$ that of males
- married publish 0.157 more log (articles); or $1.17 \times$ unmarried (17% increase)
- each additional young child decreases this by 0.185; or $0.831 \times$ articles (16.9% decrease)
- each mentor pub multiplies student pub by 1.026, a 2.6% increase

Effect plots

As usual, we can understand the fitted model from predicted values for the model effects:

```
library(effects); plot(allEffects(phd.pois))
```



This just displays the fitted model

These are better visual summaries for a model than a table of coefficients.

Model diagnostics



Model diagnostics

Diagnostic methods for count data GLMs are similar to those used for classical linear models

- Test for presence of **interactions**
 - Fit model(s) with some or all two-way interactions
- **Non-linear** effects of quantitative predictors
 - Component-plus-residual plots– `car::crPlot()` is useful here
- **Outliers?** Influential observations?
 - `car::influencePlot()` is your friend
- For count data models we should also check for **overdispersion**
 - Similar to homogeneity of variance checks in `lm()`

Checking for interactions

As a **quick check** for interactions, fit a model with **all two-way terms**, `. ~ .^2`

```
> phd.pois1 <- update(phd.pois, . ~ .^2)
> Anova(phd.pois1)
Analysis of Deviance Table (Type II tests)
```

Response: articles

	LR	Chisq	Df	Pr(>Chisq)	
female		14.5	1	0.00014	***
married		6.2	1	0.01277	*
kid5		19.5	1	9.8e-06	***
phdprestige		1.0	1	0.32655	
mentor		128.1	1	< 2e-16	***
female:married		0.3	1	0.60995	
female:kid5		0.1	1	0.72929	
female:phdprestige		0.2	1	0.63574	
female:mentor		0.0	1	0.91260	
married:kid5			0		
married:phdprestige		1.7	1	0.19153	
married:mentor		1.2	1	0.28203	
kid5:phdprestige		0.2	1	0.68523	
kid5:mentor		2.8	1	0.09290	.
phdprestige:mentor		3.8	1	0.05094	.

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Q: why df=0 here?

interactions

Compare models

The all main effects and all two-way models are nested, so we can compare them with `anova()`

```
> anova(phd.pois, phd.pois1, test="Chisq")
Analysis of Deviance Table

Model 1: articles ~ female + married + kid5 + phdprestige + mentor
Model 2: articles ~ female + married + kid5 + phdprestige + mentor +
female:married +
  female:kid5 + female:phdprestige + female:mentor + married:kid5 +
  married:phdprestige + married:mentor + kid5:phdprestige +
  kid5:mentor + phdprestige:mentor
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1          909         1634
2          900         1618   9     15.2   0.086 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- No strong evidence that the two-way terms result in a significantly better model
- A more principled analysis would consider **which interactions** might be interesting / important, e.g., `phdprestige:mentor`
- `pscl::vuong()` allows testing non-nested models

Compare models

We can also compare using AIC/BIC with `vcdExtra::LRstats()`

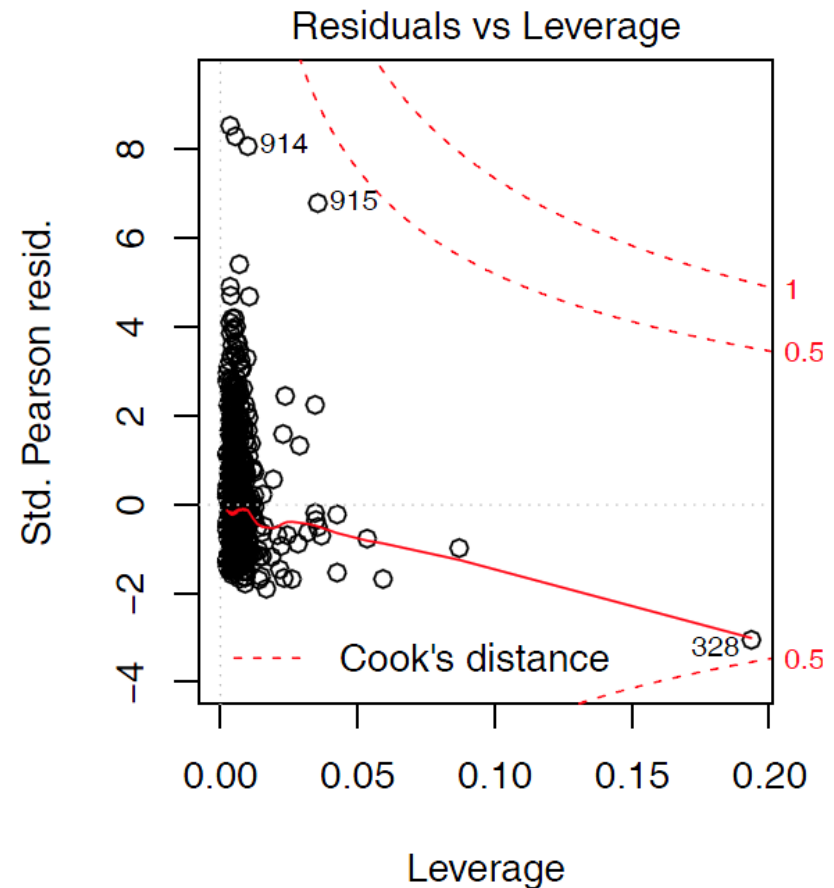
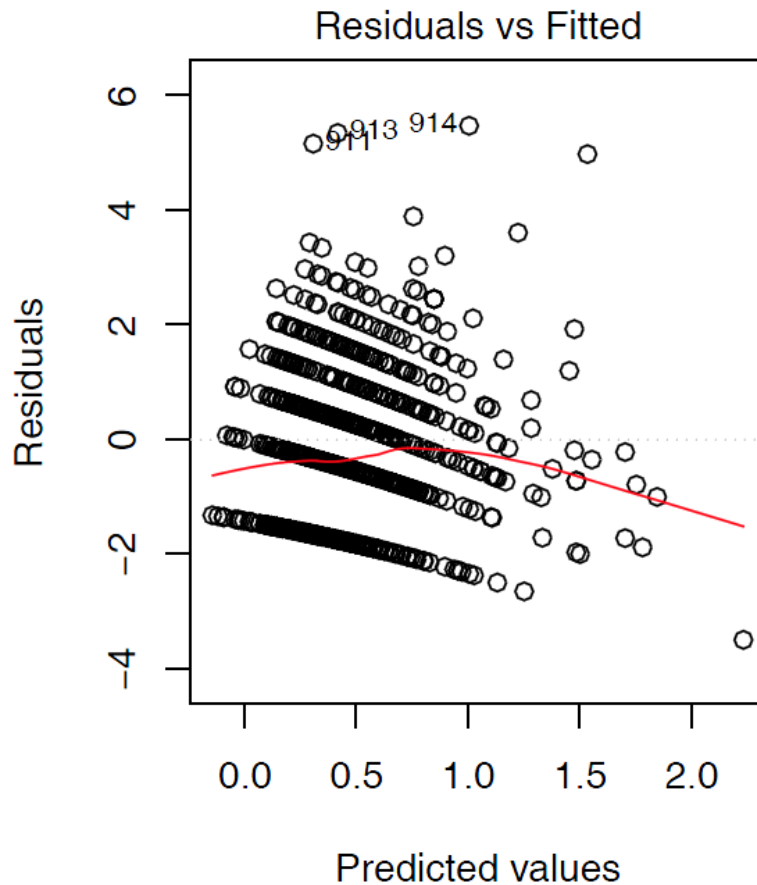
```
> LRstats(phd.pois, phd.pois1)
Likelihood summary table:
      AIC   BIC LR Chisq  Df Pr(>Chisq)
phd.pois 3313 3342    1634 909   <2e-16 ***
phd.pois1 3316 3388    1618 900   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- There seems to be no reason to include interactions in this model
 - All these interactions **increase** AIC & BIC
- We might want to revisit this, after examining other models for the basic count distribution (quasi-poisson, negative-binomial)
- We might want to consider some **specific** interaction(s) that seem substantively interesting or important to test.

Basic model plots

Only two of the standard model plots are informative for count data models

```
plot(phd.pois, which=c(1,5))
```



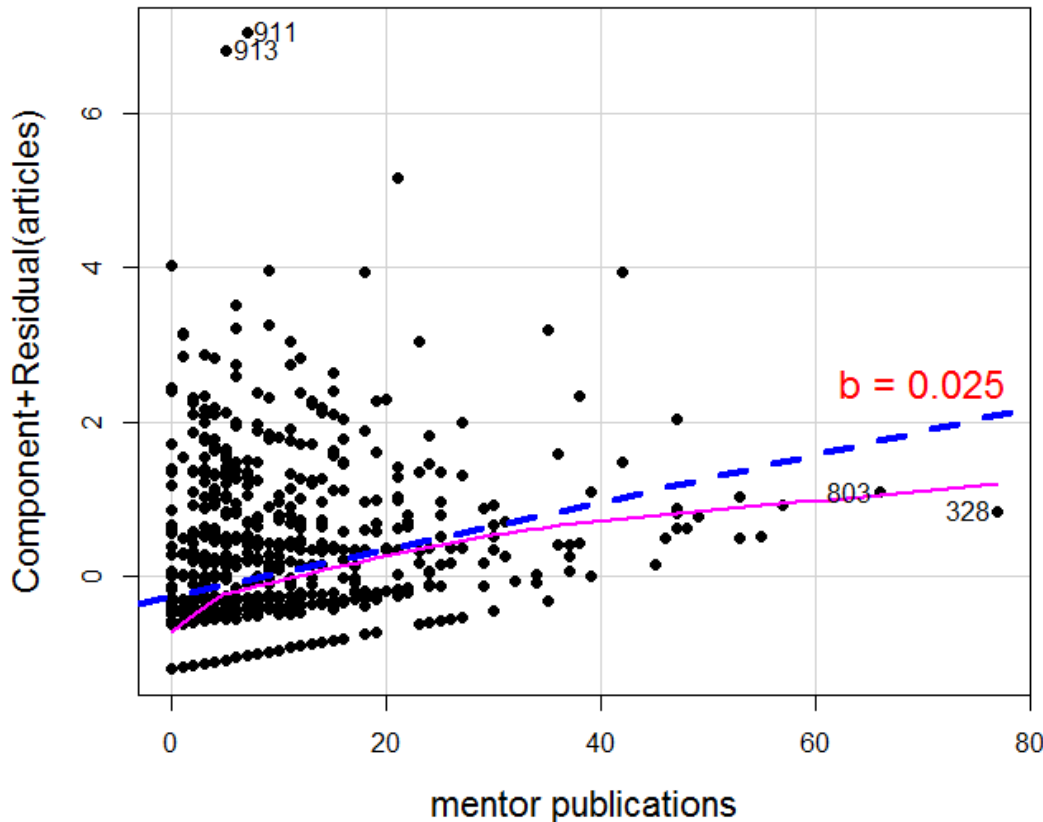
Nonlinearity diagnostics

- Nonlinear relations are difficult to assess in **marginal** plots, because they don't control (or adjust) for other predictors
- **Component-plus-residual** plots (also called: **partial residual** plots) can show **nonlinear** relations for numeric predictors
 - These graph the value of $\hat{\beta}_i x_i + \text{residual}_i$ vs. the predictor x_i
 - In this plot, the slope of the points is the coefficient $\hat{\beta}_i$ in the full model
 - The residual is $y_i - \hat{y}_i$ in the full model
- A non-parametric (e.g., `loess()`) smooth facilitates detecting nonlinearity

Nonlinearity diagnostics: crPlot()

Is the relation between article published by the student and by the mentor adequately represented as linear?

```
crPlot(phd.pois, "mentor", pch=16, lwd=4, id = list(n=2))
```



The smoothed curve doesn't differ much from the fitted line

A couple of points stand out: 328, 803, 911, 913

Residuals

Residuals contain all the information about how a model doesn't fit, and maybe why

For GLMs, there are several types, based on the Pearson and deviance goodness-of-fit statistics

- the **Pearson residual** is the case-wise contribution to Pearson χ^2

$$r_i^P = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{V}(y_i)}}$$

- the **deviance residual** is the signed square root of the contribution to the deviance G^2

$$r_i^D = \text{sign}(y_i - \hat{\mu}_i) \sqrt{d_i}$$

These are **raw** residuals, on the scale of the counts themselves

Residuals

- Both of these have **standardized** forms that correct for conditional variance and leverage, and have approx. $\mathcal{N}(0, 1)$ distributions.

$$\tilde{r}_i^P = \frac{r_i^P}{\sqrt{\hat{\phi}(1 - h_i)}}$$
$$\tilde{r}_i^D = \frac{r_i^D}{\sqrt{\hat{\phi}(1 - h_i)}}$$

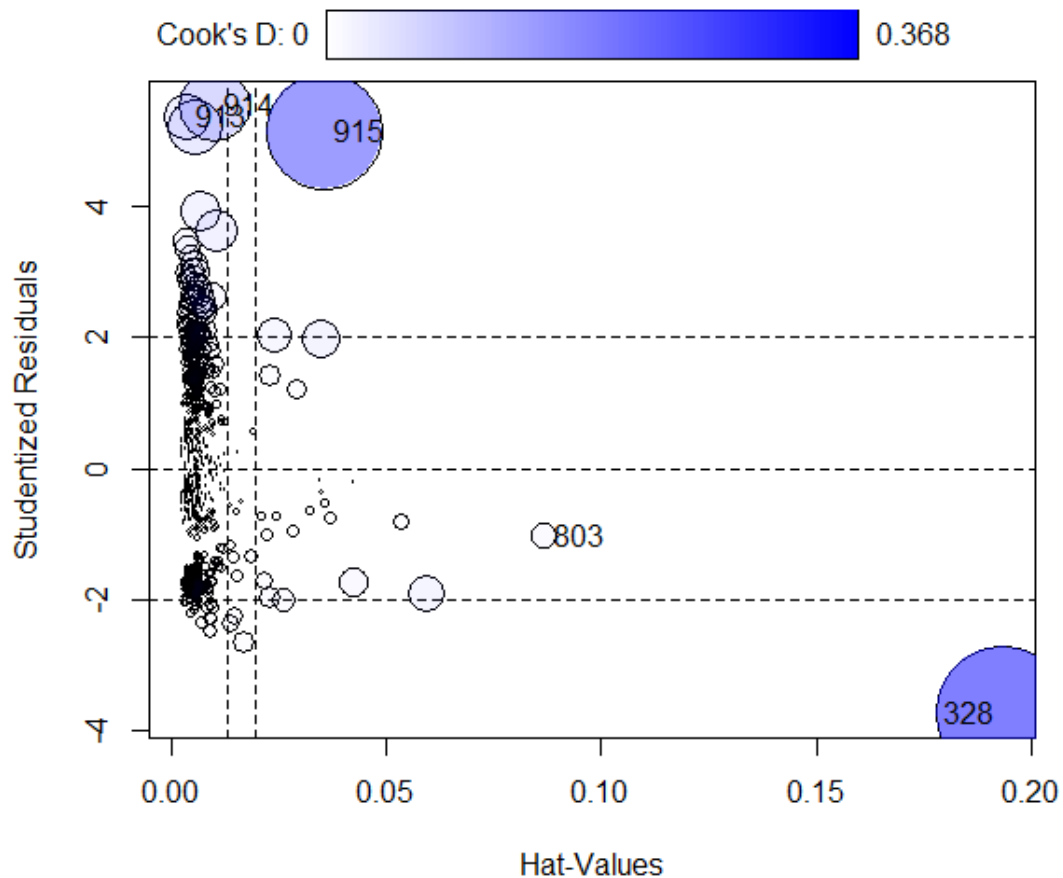
- The most useful is the **studentized residual** (or deletion residual), `rstudent()` in R. This estimates the standardized residual resulting from omitting each observation in turn. An approximation is:

$$\tilde{r}_i^S = \text{sign}(y_i - \hat{\mu}_i) \sqrt{(1 - h_i)(\tilde{r}_i^D)^2 + h_i(\tilde{r}_i^P)^2} .$$

Don't worry about the formulas, but do know the difference among **raw**, **standardized** and **studentized** residuals

Outliers, leverage & influence

```
influencePlot(phd.pois, id = list(n=2))
```



Influence (CookD) =
Leverage (Hat) x |Residual|

Several cases (913-915) stand out with large + residuals

One observation (328) has a large leverage

Why are they unusual? Do they affect conclusions?

Examine data & decide what to do

Who is influential & why?

At the very least, you should **examine** these flagged observations in the data

```
> PhdPubs[c(328, 803, 913:915),]
  articles female married kid5 phdprestige mentor
328      1      0      1      1           2      77
803      4      0      1      2           5      66
913     12      0      1      1           2       5
914     16      0      1      0           2      21
915     19      0      1      0           2      42
```

case 328: Mentor published 77 papers! Student, only 1

803: High prestige school, mentor published 66; published < than predicted. Two kids?

913-915: Wow! all published >> than predicted. Hot shots???

Outlier test

- A formal test for outliers can be based on the **studentized residuals**, `rstudent(model)`, using the standard normal distribution for p -values
- A Bonferroni correction should be applied, because interest focuses on the **largest n** absolute residuals.

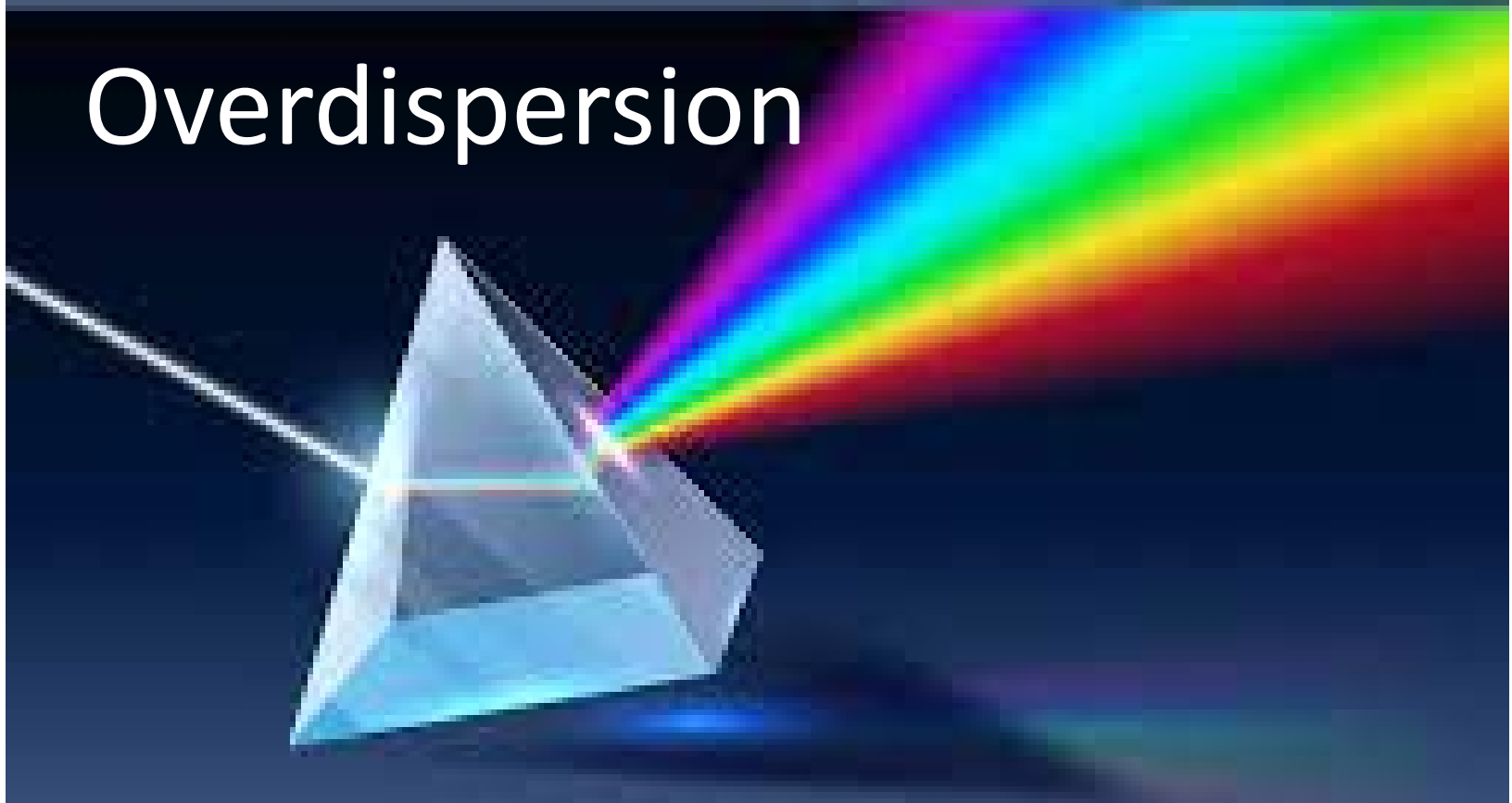
For this Poisson model, 4 observations are flagged as large + residuals

```
> car::outlierTest(phd.pois, cutoff = 0.001)
      rstudent unadjusted p-value Bonferroni p
914      5.54      2.99e-08      2.73e-05
913      5.38      7.36e-08      6.74e-05
911      5.21      1.92e-07      1.75e-04
915      5.15      2.60e-07      2.38e-04
```

What to do?

- Delete them & refit?
- Keep them, but report as unusual?
- Fit a better model, hope these will go away?

Overdispersion



Overdispersion

- The Poisson model for counts assumes $\mathcal{V}(\mu_i) = \mu_i$, i.e., the dispersion parameter $\phi = 1$
- But often, the counts exhibit greater variance than the Poisson distribution allows, $\mathcal{V}(\mu_i) > \mu_i$ or $\phi > 1$
 - The observations (counts) may not be independent (clustering)
 - The probability of an “event” may not be constant
 - There may be unmeasured influences, not accounted for in the model
 - These effects are sometimes called “unmodeled heterogeneity”
- The consequences are:
 - Standard errors of the coefficients, $se(\hat{\beta}_j)$ are optimistically small
 - Wald tests, $z_j = \hat{\beta}_j/se(\hat{\beta}_j)$, are too large, and thus overly liberal.

Testing overdispersion

- Statistical tests for overdispersion test $H_0: \text{Var}(y) = \mu$ vs. the alternative

$$H_1: \text{Var}(y) = \mu + \phi \times f(\mu)$$

- Implemented in **AER::dispersiontest()**
 - If significant, overdispersion should not be ignored
 - You can try fitting a more general model
 - Quasi-poisson
 - Negative-binomial

```
> AER::dispersiontest(phd.pois)
```

```
Overdispersion test
data:  phd.pois
z = 5.7347, p-value = 4.885e-09
alternative hypothesis: true dispersion
is greater than 1
sample estimates:
dispersion
 1.825903
```

Quasi-poisson models

- The quasi-poisson model allows the dispersion, ϕ , to be a free parameter, estimates with other coefficients
- The conditional variance is allowed to be a multiple of the mean

$$\text{Var}(y_i | \eta_i) = \phi \mu_i$$

- This model is fit with `glm()` using `family=quasipoisson`
 - The estimated coefficients $\hat{\beta}$ are **unchanged**
 - The standard errors are multiplied by $\phi^{1/2}$
 - Peace, order & good government is restored!

Quasi-poisson models

- A simple estimate of the dispersion parameter is the residual deviance divided by degrees of freedom: $\hat{\phi} = D(y, \hat{\mu}) / df$
- A Pearson χ^2 statistic has better statistical properties & is more commonly used

$$\hat{\phi} = \frac{\chi_p^2}{n - p} = \sum_{i=1}^n \frac{(y_i - \hat{\mu}_i)^2}{\hat{\mu}_i} / (n - p)$$

For the PhdPubs data, these estimates are quite similar: about 80% overdispersion

```
> with(phd.pois, deviance/df.residual)
[1] 1.8
```

```
> sum(residuals(phd.pois, type = "pearson")^2)/phd.pois$df.residual
[1] 1.83
```

Fitting the quasi-poisson model

You can fit the quasi-poisson model using `glm()`

```
> phd.qpois <- glm(articles ~ ., data = PhdPubs, family = quasipoisson)
```

The estimate of the dispersion parameter is calculated by the `summary()` method. You can get it as follows:

```
> (phi <- summary(phd.qpois)$dispersion)
[1] 1.83
```

This is much better than variance/mean ratio of 2.91 calculated for the marginal distribution ignoring the predictors.

Coefficients unchanged; std. errors multiplied by $\hat{\phi}^{1/2} = \sqrt{1.83} = 1.35$.

```
> summary(phd.qpois)
```

Call:

```
glm(formula = articles ~ ., family = quasipoisson, data = PhdPubs)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-3.488	-1.538	-0.365	0.577	5.483

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.26562	0.13478	1.97	0.04906	*
female1	-0.22442	0.07384	-3.04	0.00244	**
married1	0.15732	0.08287	1.90	0.05795	.
kid5	-0.18491	0.05427	-3.41	0.00069	***
phdprestige	0.02538	0.03419	0.74	0.45815	
mentor	0.02523	0.00275	9.19	< 2e-16	***

Consequently, t
stats are smaller

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 1.83)

Null deviance: 1817.4 on 914 degrees of freedom
Residual deviance: 1633.6 on 909 degrees of freedom
AIC: NA

The negative-binomial model

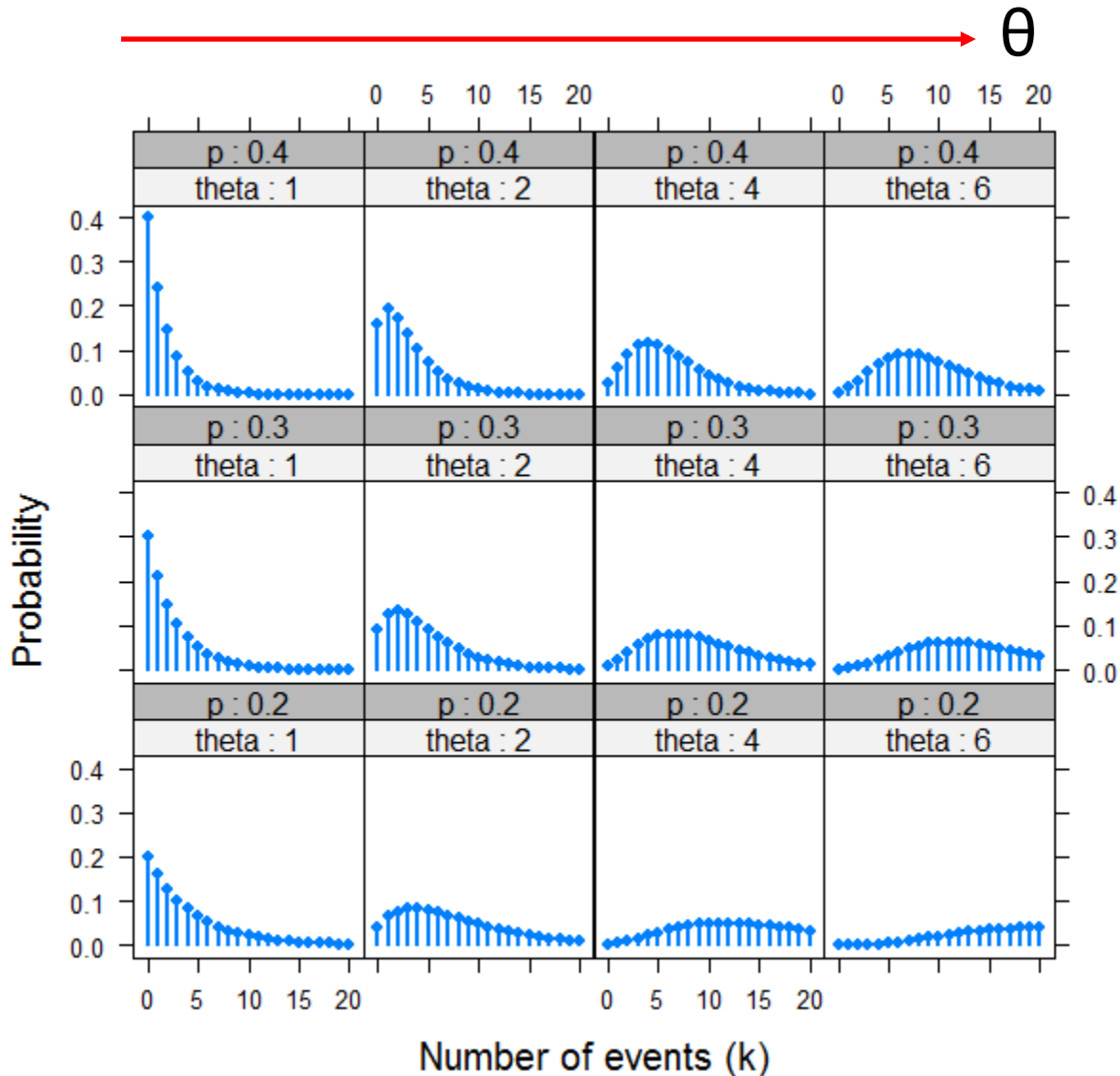
- The negative-binomial model is a different generalization of the Poisson that allows for over-dispersion
- Mathematically, it allows the mean $\mu | \mathbf{x}_i$ to vary across observations as a gamma distribution with a shape parameter θ .
- The variance function, $\mathcal{V}(y_i) = \mu_i + \mu_i^2/\theta$, allows the variance of y to increase more rapidly than the mean.
- Another parameterization uses $\alpha = 1/\theta$

$$\mathcal{V}(y_i) = \mu_i + \mu_i^2/\theta = \mu_i + \alpha\mu_i^2 ,$$

- As $\alpha \rightarrow 0$, $\mathcal{V}(y_i) \rightarrow \mu_i$ and the negative-binomial converges to the Poisson.

See: [Negative Binomial: Interpreting the overdispersion parameter](#)

The negative-binomial model



Negative-binomial distributions for varying p & θ

Overdispersion

↓ as θ increases

↑ as α increases

Fitting the negative-binomial

- For fixed θ , the negative-binomial is another special case of the GLM
- This is handled in the `MASS` package, with
`family=negative.binomial(theta)` `MASS::glm(family=)`
- But most often, θ is unknown, and must be estimated from the data
- This is implemented in `glm.nb()` in the `MASS` package.

```
> library(MASS)
> phd.nbin <- glm.nb(articles ~ ., data=PhdPubs)
> # extract theta & std.err
```

```
> unlist(summary(phd.nbin)[c("theta", "SE.theta")])
  theta SE.theta
 2.267    0.272
```

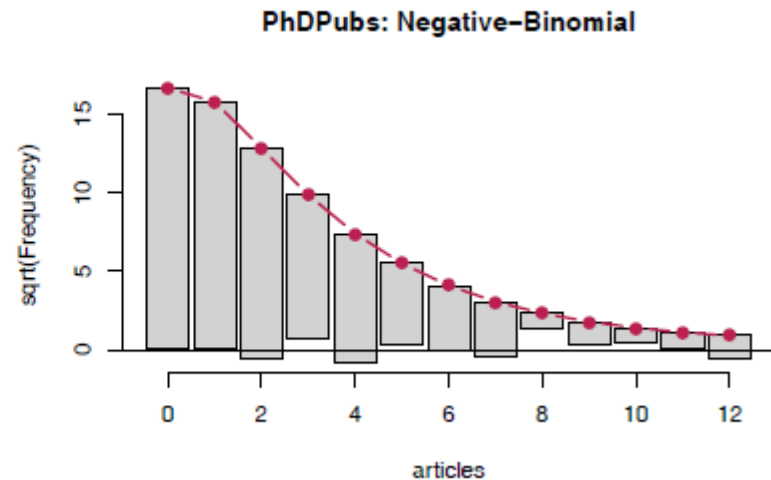
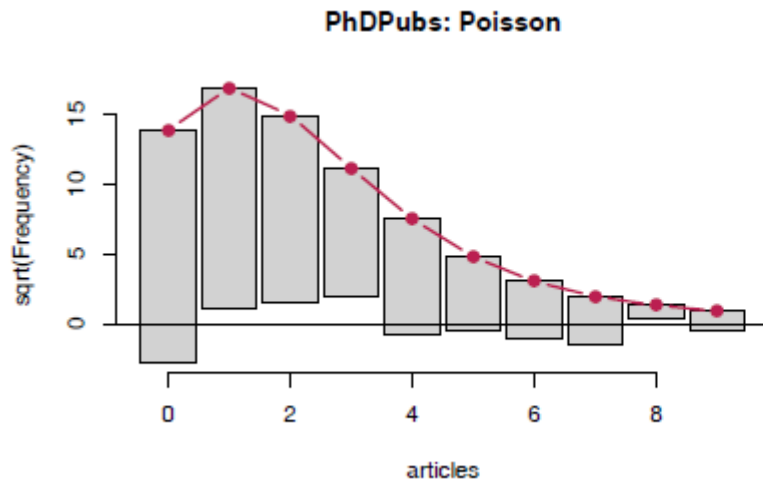
Equivalently: $\alpha = 1/\theta = 0.44$

Individual pub rates vary by a factor with
 $SD = \sqrt{\alpha} = 0.66$ around predicted rates

Visualizing goodness-of-fit

The countreg package extends rootogram() to work with fitted models:

```
countreg::rootogram(phd.pois, main="PhDPubs: Poisson")
countreg::rootogram(phd.nbin, main="PhDPubs: Negative-Binomial")
```




The Poisson model shows a systematic, wave-like pattern with excess zeros, too few observed frequencies for counts of 1--3.

Comparing models: What difference does it make?


The NB is certainly a better fit than the Poisson; the QP cannot be distinguished by standard LR tests

```
> LRstats(phd.pois, phd.qpois, phd.nbin)
Likelihood summary table:
      AIC  BIC LR Chisq  Df Pr(>Chisq)
phd.pois 3313 3342      1634 909    <2e-16 ***
phd.qpois          909
phd.nbin  3135 3169      1004 909    0.015 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can also compare **coefficients** and their **standard errors** for these models



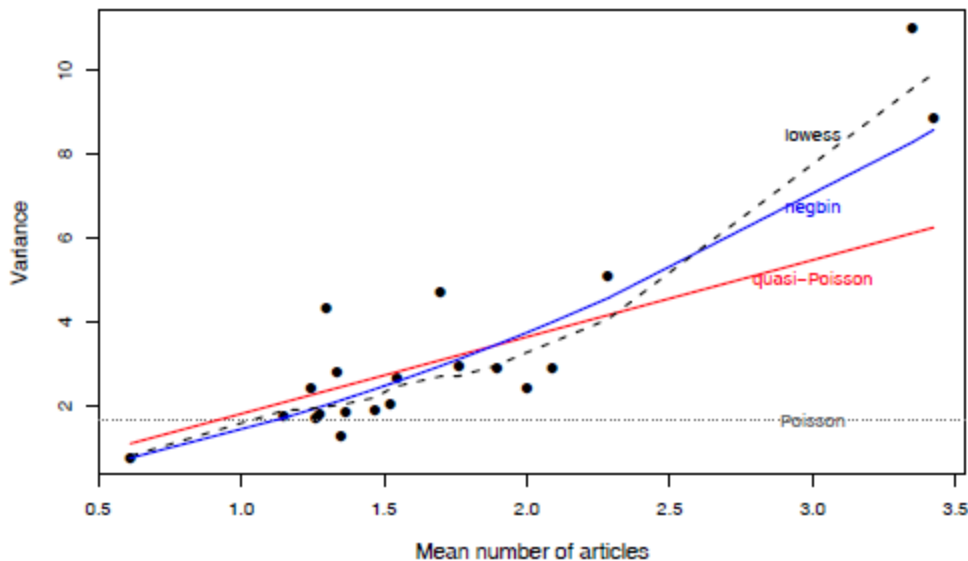
	pois	qpois	nbin
(Intercept)	0.266	0.266	0.213
female1	-0.224	-0.224	-0.216
married1	0.157	0.157	0.153
kid5	-0.185	-0.185	-0.176
phdprestige	0.025	0.025	0.029
mentor	0.025	0.025	0.029



	pois	qpois	nbin
(Intercept)	0.0996	0.1348	0.1327
female1	0.0546	0.0738	0.0726
married1	0.0613	0.0829	0.0819
kid5	0.0401	0.0543	0.0528
phdprestige	0.0253	0.0342	0.0343
mentor	0.0020	0.0027	0.0032

Visualizing the mean-variance relation

One way to see the difference among models is to plot the variance vs. mean for **grouped** values of the fitted linear predictor.



- The smoothed (loess) curve gives the **empirical mean–variance** relationship
- Also plot the theoretical mean–variance from different models
- For PhdPubs, the data is most similar to the negative-binomial
- The models differ most for those with > 3 articles

I call this a “**model sensitivity plot**” – how much effect do different assumptions make?

What have we learned so far?

A summary to this point should use the result of the negative-binomial model

```
> lmtest::coeftest(phd.nbin)
z test of coefficients:

```

	Estimate	Std. Error	z value	Pr(> z)		
(Intercept)	0.21295	0.13274	1.60	0.10866		
female1	-0.21625	0.07259	-2.98	0.00289	**	✓
married1	0.15279	0.08194	1.86	0.06224	.	?
kid5	-0.17634	0.05279	-3.34	0.00084	***	✓
phdprestige	0.02934	0.03427	0.86	0.39192		X
mentor	0.02868	0.00324	8.86	< 2e-16	***	✓ ✓

For interpretation, examine the coefficients, β , e^β and % change for each pub

```
> round(cbind(beta = coef(phd.nbin),
              expbeta = exp(coef(phd.nbin)),
              pct = 100 * (exp(coef(phd.nbin)) - 1)), 3)

```

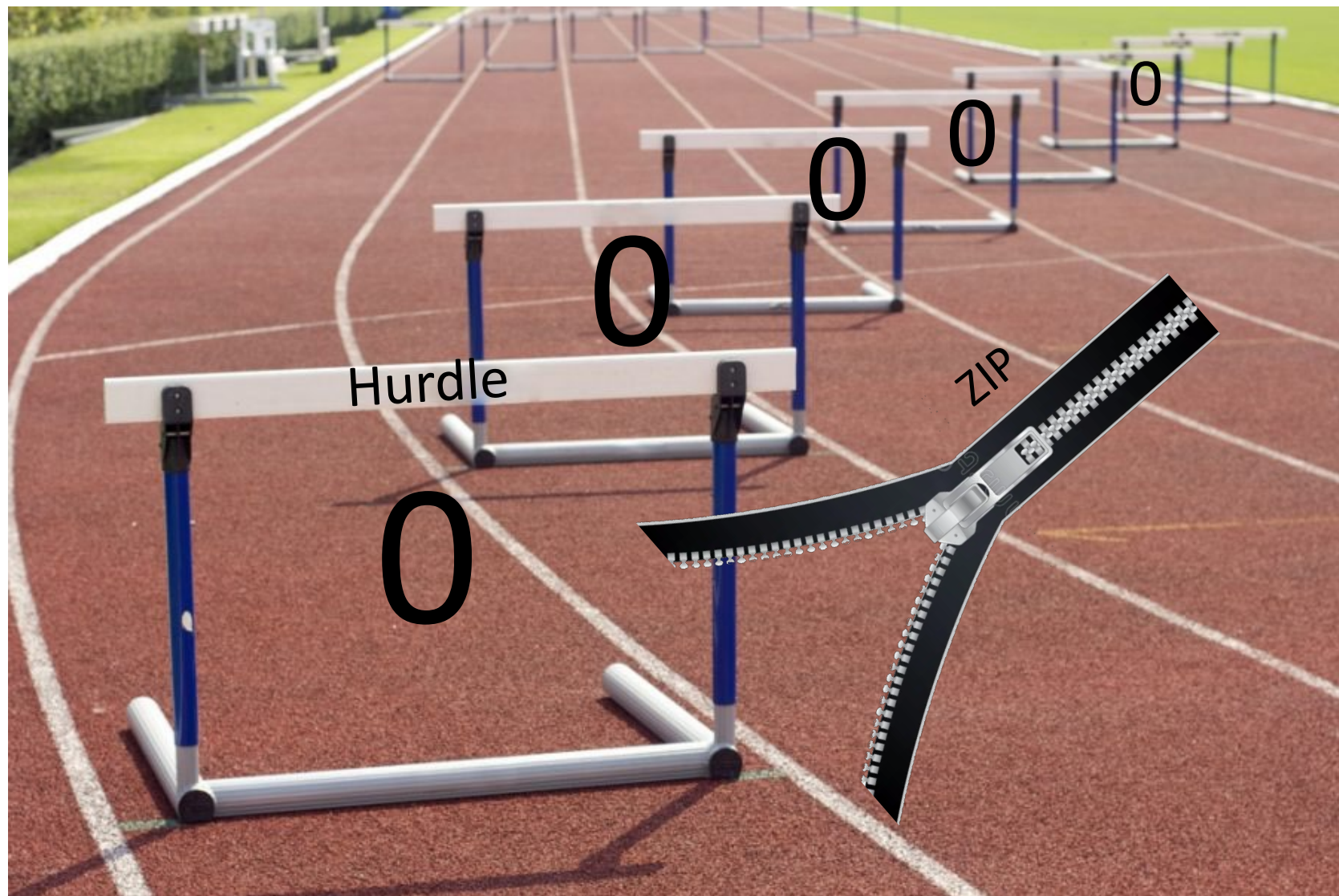
	beta	expbeta	pct
(Intercept)	0.213	1.237	23.73
female1	-0.216	0.806	-19.45
married1	0.153	1.165	16.51
kid5	-0.176	0.838	-16.17
phdprestige	0.029	1.030	2.98
mentor	0.029	1.029	2.91

What have we learned so far?

The number of articles published by PhD candidates:

- Most strongly predicted by **mentor** pubs, but with a modest effect. On average, each mentor pub increases PhD articles by 2.9%
- Next, increasing young children (**kids5**) results in fewer publications. On average, each additional kid reduces PhD articles by 16%
- Being **married** is marginally NS, but intriguing. Our estimate shows married candidates publish 16.5% more articles than non-married.
- Perhaps surprisingly, the prestige of the PhD institution has no significant effect in this purely main-effect model. Yet, a unit change in **phdprestige** is estimated as a 3% increase in PhD articles
- Yet, we still have doubts:
 - Several cases (328, 913-915) appeared unusual in diagnostic plots. Should we refit w/o them to see if conclusions change?
 - The NB model might not be the best way to account for the zero counts – students who never published
 - Is there a better way?

Excess zeros



Excess zero counts

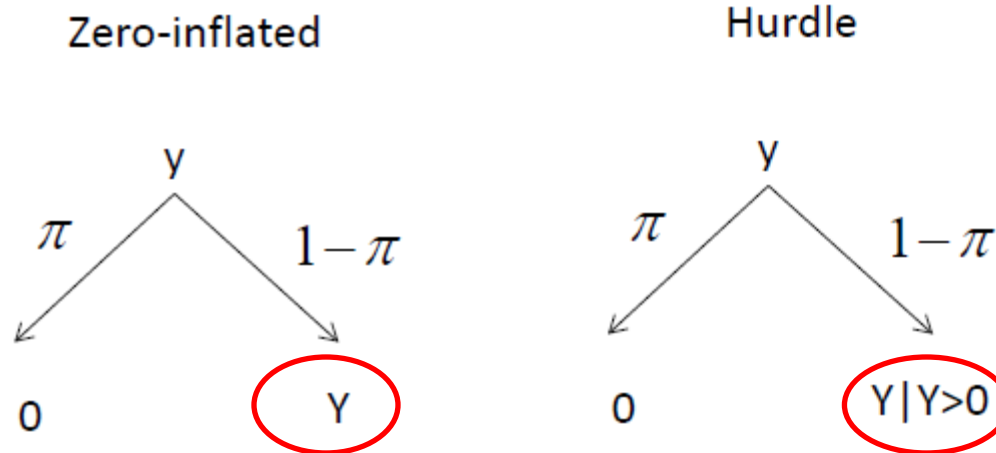
- A common problem in count data models is that many sets of data have more observed zero counts than the (quasi) Poisson or NB models can handle.
 - In the PhdPubs data, 275 of 915 (30%) candidates published zilch, bupkis
 - The expected count of 0 articles in the Poisson model is only 191 (21%)
- Maybe there are two types of students giving zero counts:
 - Those who never intend to publish (non-academic career path?)
 - The rest, who do intend to publish, but have not yet done so
 - This suggests the idea of **zero inflation**
- An alternative idea is that there is some **hurdle** to overcome before attaining a positive count, e.g., external pressure from the mentor.

Beyond simply identifying this as a problem of lack-of-fit, understanding the **reasons** for excess zero counts can contribute to a more complete explanation of the phenomenon of interest.

Models for excess zeros

Two types of models, with different mechanisms for zero counts

- **zero-inflated models:** The responses with $y_i = 0$ arise from a mixture of **structural**, always 0 values, with $\Pr(y_i = 0) = \pi_i$ and the rest, which are **random** 0s, with $\Pr(y_i = 0) = 1 - \pi_i$
- **hurdle models:** One process determines whether $y_i = 0$ with $\Pr(y_i = 0) = \pi_i$. A second process determines the distribution of values of positive counts, $\Pr(y_i | y_i > 0)$



Zero-inflated models

The **zero-inflated Poisson** (ZIP) model has two components:

- A logistic regression model for membership in the unobserved (latent) class of those for whom y_i is necessarily zero

$$\text{logit}(\pi_i) = \mathbf{z}_i^T \boldsymbol{\gamma} = \gamma_0 + \gamma_1 z_{i1} + \gamma_2 z_{i2} + \cdots + \gamma_q z_{iq} .$$

- A Poisson model for the other class (e.g., “publishers”), for whom y_i may be 0 or positive.

$$\log_e \mu(y_i | \mathbf{x}_i) = \mathbf{x}_i^T \boldsymbol{\beta} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_q x_{ip} .$$

In application, the same predictors can be (and often are) used in both models ($\mathbf{x} = \mathbf{z}$). But not necessarily!

Zero-inflated models: ZIP & ZINB

In the ZIP model, the probabilities of observing counts of $y_i = 0$ and $y_i > 0$ are:

zero model $\Pr(y_i = 0 | \mathbf{x}, \mathbf{z}) = \pi_i \times (1 - \pi_i) e^{-\mu_i}$

count model $\Pr(y_i | \mathbf{x}, \mathbf{z}) = (1 - \pi_i) \times \left[\frac{\mu_i^{y_i} e^{-\mu_i}}{y_i!} \right], \quad y_i \geq 0 .$

The conditional expectation and variance of y_i then are:

$$\begin{aligned} \mathcal{E}(y_i) &= (1 - \pi_i) \mu_i \\ \mathcal{V}(y_i) &= (1 - \pi_i) \mu_i (1 + \mu_i \pi_i) . \end{aligned}$$

When $\pi_i > 0$, the mean of y is always less than μ_i ; the variance of y is greater than its mean by a dispersion factor of $(1 + \mu_i \pi_i)$.

The model for the count variable could also be negative-binomial, giving a *zero-inflated negative-binomial* (ZINB) model using $\text{NBin}(\mu, \theta)$

Exploring zero-inflated data

A little insight can be gained by generating random data from Poisson & zero-inflated analog. The example uses `VGAM::rzipois()`

$\text{Pois}(\mu=3) = \text{ZIP}(\mu=3, \pi=0)$

vs. $\text{ZIP}(\mu=3, \pi=.3)$

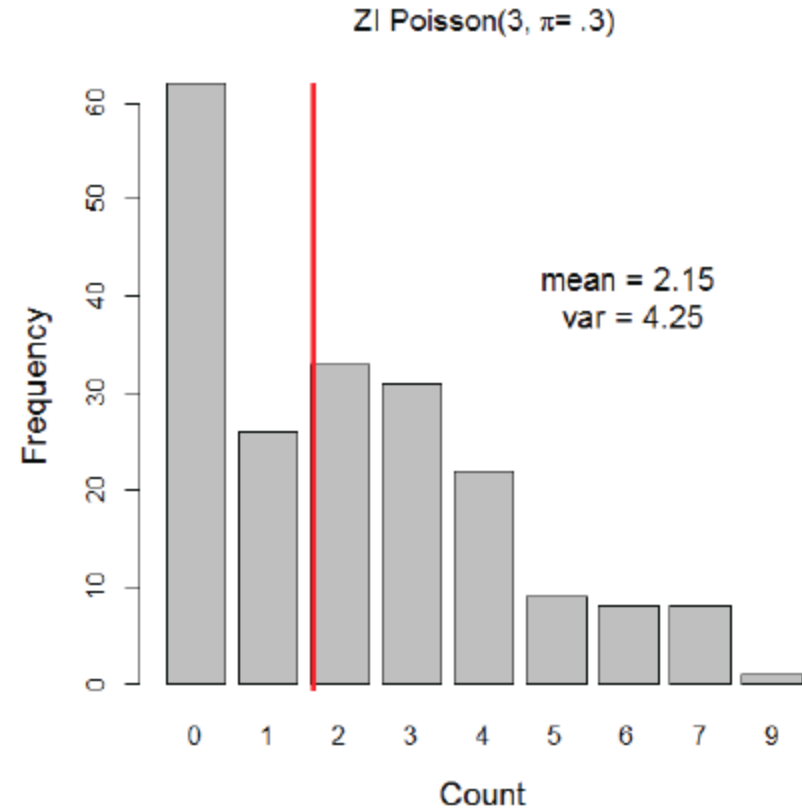
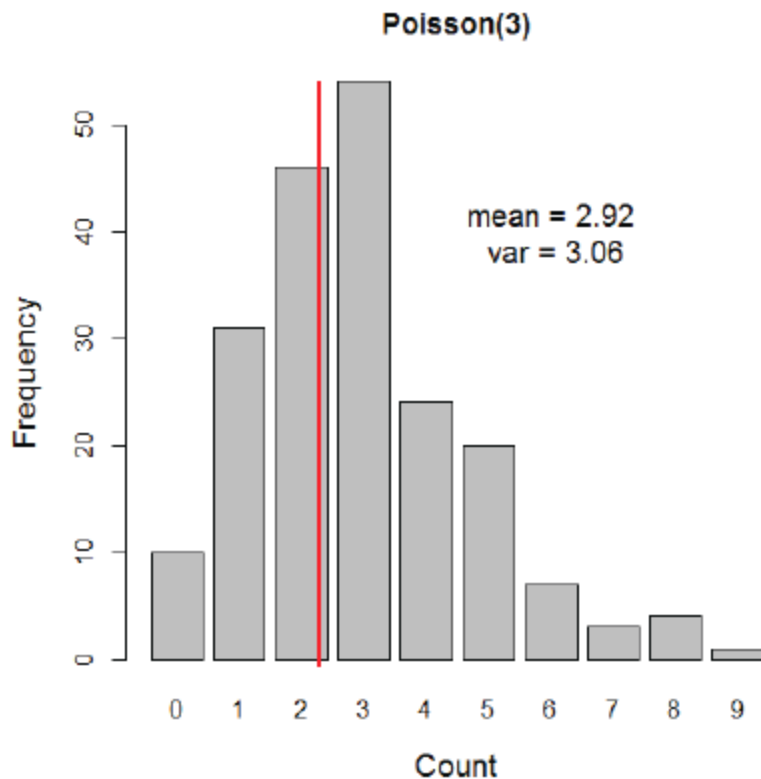
```
> set.seed(1234)
> data1 <- VGAM::rzipois(200, 3, 0)
> data2 <- VGAM::rzipois(200, 3, .3)
```

The tables of counts show far more zeros in data2

```
> table(data1)
data1
 0  1  2  3  4  5  6  7  8  9
10 31 46 54 24 20  7  3  4  1
> table(data2)
data2
 0  1  2  3  4  5  6  7  9
62 26 33 31 22  9  8  8  1
```

Exploring zero-inflated data

Bar plots of the counts:



The 30% extra zeros decrease the mean and inflate the variance

Hurdle models

The **Hurdle** model also has two components:

- A logistic regression model, for the probability that $y_i = 0$ vs. $y_i > 0$

$$\text{logit} \left[\frac{\Pr(y_i = 0)}{\Pr(y_i > 0)} \right] = \mathbf{z}_i^T \boldsymbol{\gamma} = \gamma_0 + \gamma_1 z_{i1} + \gamma_2 z_{i2} + \cdots + \gamma_q z_{iq} .$$

- A model for the **positive** counts, taken as a **left-truncated** Poisson or negative-binomial, excluding the zero counts
- Comparing the ZIP and Hurdle models:
 - In ZIP models, the first (latent) process generates **extra** zeros (with probability π_i).
 - In Hurdle models, $y_i = 0$ and $y_i > 0$ are fully observed. The first process generates all the zeros.

Fitting ZIP & Hurdle models

In R, these models can be fit using the `pscl` and `countreg` packages.*

`countreg` is ~~more mature, but is~~ only available on R-Forge, not on CRAN. Use:

```
install.packages("countreg", repos="http://R-Forge.R-project.org")
```

The functions have the following arguments:

```
zeroinfl(formula, data, subset, na.action, weights, offset,  
          dist = c("poisson", "negbin", "geometric", "binomial"),  
          ...)
```

```
hurdle(formula, data, subset, na.action, weights, offset,  
        dist = c("poisson", "negbin", "geometric", "binomial"),  
        ...)
```

The formula, $y \sim x_1 + x_2 + \dots$ uses the same predictors for both models.

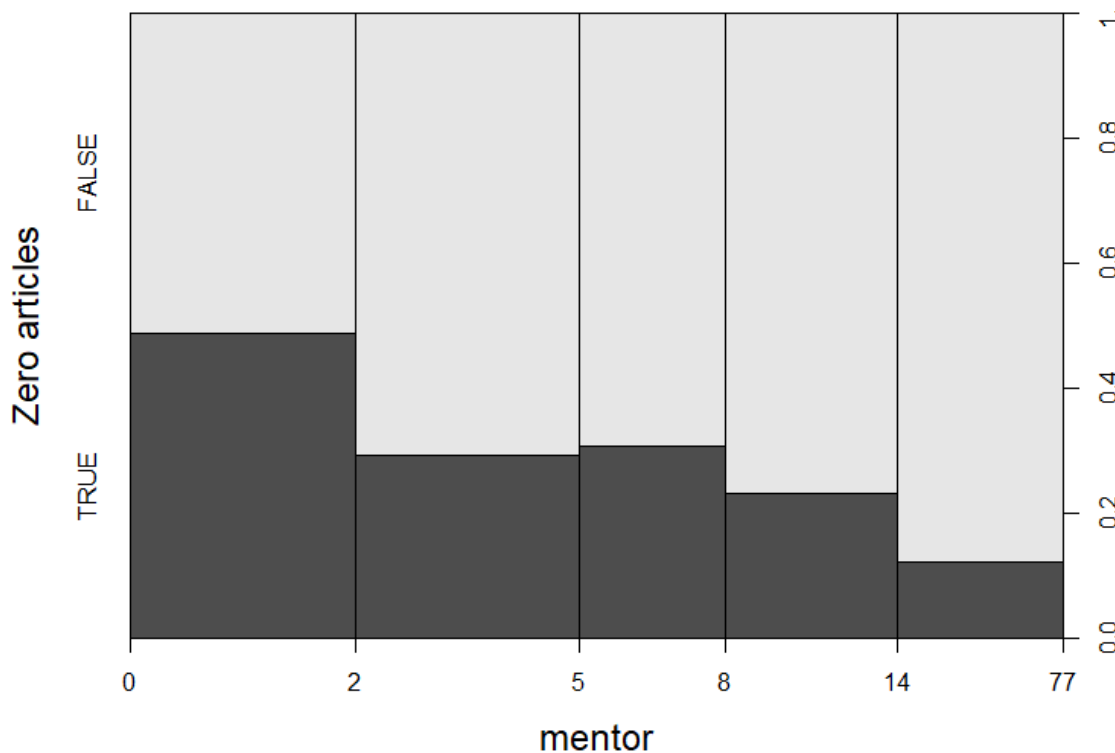
Using $y \sim x_1 + x_2 + \dots | z_1 + z_2 + \dots$ allows separate predictors for the 0 submodel.

* NB: These analyses use `countreg`. Not everything may work the same with `pscl`

Visualizing zero counts

It is often useful to plot the data for the binary distinction between $y_i = 0$ vs. $y_i > 0$ as in logistic regression models.

```
plot(factor(articles==0) ~ mentor, data=PhdPubs,  
      ylevels=1:2, ylab="Zero articles",  
      breaks=quantile(mentor, probs=seq(0,1,.2)))
```



As expected, zero counts decrease with mentor pubs

Could do this for other predictors

NB: This gives a spineplot

Fitting models

To illustrate, I fit all four models, the combinations of (ZI, hurdle) × (poisson, nbin) to the phdpubs data.

For simplicity, I use [all predictors](#) for both the zero model and the non-zero model.

```
phd.zip <- zeroinfl(articles ~ ., data=PhdPubs, dist="poisson")
phd.znb <- zeroinfl(articles ~ ., data=PhdPubs, dist="negbin")

phd.hp <- hurdle(articles ~ ., data=PhdPubs, dist="poisson")
phd.hnb <- hurdle(articles ~ ., data=PhdPubs, dist="negbin")
```

Model formula notation: formula = response ~ pred_count + ... | pred_zero + ...

Intuition: For hurdle model, the count model is conditional (“|”) on $Y > 0$

```
y ~ x1 + x2           # same regressors for count and zero models
y ~ x1 + x2 | x1 + x2 # equivalent
y ~ x1 + x2 | z1 + z2 # different regressors in both models
y ~ X1 + x2 | 1       # all zero counts have same probability
```

Testing the hurdle assumption

For `pscl::hurdle()`, there is also a `hurdletest()` function.

It tests the hypothesis that **all** coefficients in the count model are all equal to their values in the zero model

```
> hurdltest(phd.hp)
Wald test for hurdle models
```

Restrictions:

```
count_(Intercept) - zero_(Intercept) = 0
count_female - zero_female = 0
count_married - zero_married = 0
count_kid5 - zero_kid5 = 0
count_phdprestige - zero_phdprestige = 0
count_mentor - zero_mentor = 0
```

Model 1: restricted model

Model 2: articles ~ .

```
   Res.Df Df  Chisq Pr(>Chisq)
1     909
2     903  6 85.824 2.228e-16 ***
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TODO:

- Anova() method to test individually
- Extend to zip, negbin, ...
- Plot methods: ???

Interp: different variables determine if a count is zero, or it's non-zero value

Comparing models

Compare the models, sorting by BIC

```
> LRstats(phd.pois, phd.nbin, phd.zip, phd.znb, phd.hp, phd.hnb,
           sortby="BIC")
Likelihood summary table:
      AIC      BIC LR Chisq  Df Pr(>Chisq)
phd.pois 3313.3 3342.3  3301.3 909 < 2.2e-16 ***
phd.hp   3234.5 3292.4  3210.5 903 < 2.2e-16 ***
phd.zip  3233.5 3291.3  3209.5 903 < 2.2e-16 ***
phd.hnb  3130.9 3193.5  3104.9 902 < 2.2e-16 ***
phd.znb  3125.8 3188.4  3099.8 902 < 2.2e-16 ***
phd.nbin 3135.4 3169.1  3121.4 909 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The standard negative binomial model looks best by BIC.
Why do you think this is? (Hint: look at the residual df)

Nevertheless, it is useful to examine the coefficients in the ZIP model

```
> lmtest::coefstest(phd.zip)
```

```
t test of coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)		
count_(Intercept)	0.59918	0.11861	5.05	5.3e-07	***	} counts > 0
count_female1	-0.20879	0.06353	-3.29	0.0011	**	
count_married1	0.10623	0.07097	1.50	0.1348		
count_kid5	-0.14271	0.04744	-3.01	0.0027	**	
count_phdprestige	0.00700	0.02981	0.23	0.8145		
count_mentor	0.01785	0.00233	7.65	5.3e-14	***	} counts = 0
zero_(Intercept)	-0.56332	0.49405	-1.14	0.2545		
zero_female1	0.10816	0.28173	0.38	0.7011		
zero_married1	-0.35558	0.31796	-1.12	0.2637		
zero_kid5	0.21974	0.19658	1.12	0.2639		
zero_phdprestige	-0.00537	0.14118	-0.04	0.9697		
zero_mentor	-0.13313	0.04643	-2.87	0.0042	**	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Only mentor is significant in the ZIP model, simplifying interpretation.

Let's refit the ZIP and ZNB models using only **mentor** for the zero models

```
phd.zip1 <- zeroinfl(articles ~ .| mentor, data=PhdPubs, dist="poisson")
phd.znb1 <- zeroinfl(articles ~ .| mentor, data=PhdPubs, dist="negbin")
```

Compare models again

```
> LRstats(phd.pois, phd.nbin, phd.zip, phd.znb, phd.hp, phd.hnb,
+         phd.zip1, phd.znb1, sortby="BIC")
```

Likelihood summary table:

	AIC	BIC	LR	Chisq	Df	Pr(>Chisq)	
phd.pois	3313	3342		3301	909	<2e-16	***
phd.hp	3235	3292		3211	903	<2e-16	***
phd.zip	3234	3291		3210	903	<2e-16	***
phd.zip1	3227	3266		3211	907	<2e-16	***
phd.hnb	3131	3194		3105	902	<2e-16	***
phd.znb	3126	3188		3100	902	<2e-16	***
phd.nbin	3135	3169		3121	909	<2e-16	***
phd.znb1	3124	3168		3106	906	<2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Now, the `phd.znb1` model looks best by BIC. Let's stick with this.

Model interpretation: Coefficients

Ignoring the NS coefficients in the revised ZNB model (phd.znb1):

```
> coef(phd.znb1)[c(1,2,4,6,7,8)]
count_(Intercept)      count_female1      count_kid5      count_mentor
          0.3572          -0.2116          -0.1675          0.0241
zero_(Intercept)      zero_mentor
          -0.8169          -0.6080
```

- Count model:

$$\log(\text{articles}) = 0.357 - 0.21 \text{ female} - 0.17 \text{ kids5} + 0.024 \text{ mentor}$$

- Zero model:

$$\text{logit}(\text{articles} = 0) = -0.817 - 0.608 \text{ mentor}$$

Can you describe these in words?

Model interpretation: Coefficients

Often easier to interpret $\exp(\beta)$

```
> exp(coef(phd.znb1)[c(1,2,4,6,7,8)])
count_(Intercept)      count_female1      count_kid5      count_mentor
          1.429          0.809          0.846          1.024
zero_(Intercept)      zero_mentor
          0.442          0.544
```

Female: Women publish .21 fewer log articles, .81 times that of men (20% decrease)

Kids5: Each additional kid<5 → .17 fewer log articles, a 15% decrease

Mentor: Each additional mentor article → .024 more PhD log pubs (2.4% increase)

Count model: Each additional mentor article decreases log odds PhDpubs = 0 by 0.608, a 45% decrease

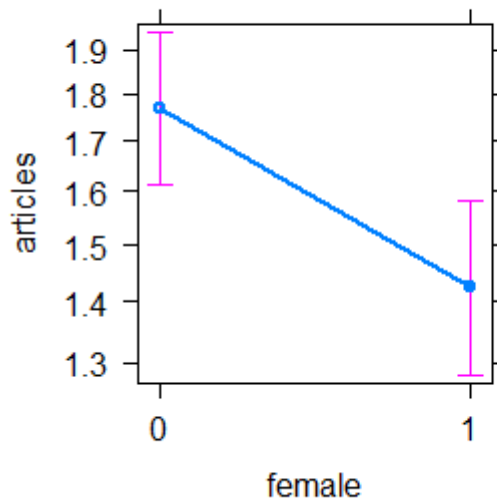
Get your mentor to publish!!

Model interpretation: Effect plots

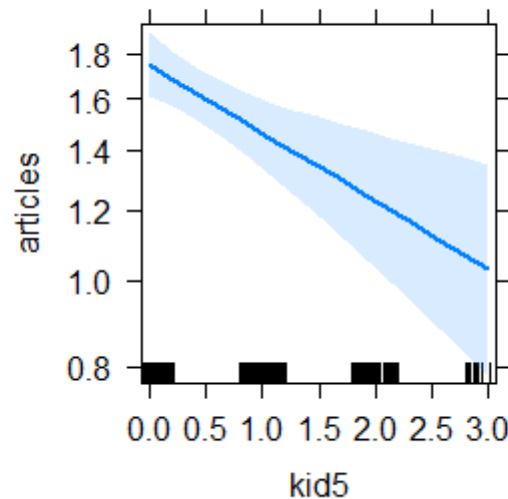
- The `effects` package cannot yet handle zero-inflated or hurdle models.
- But the fitted values don't differ very much among these models
- Here, I use the `phd.nbin` model, and just show the effects for the important terms

```
plot(allEffects(phd.nbin)[c(1,3,5)], rows=1, cols=3)
```

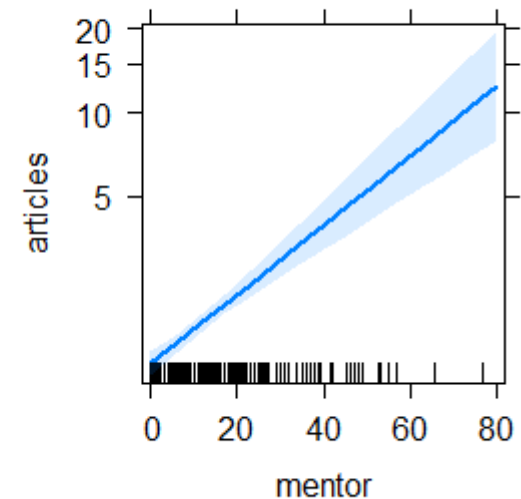
female effect plot



kid5 effect plot



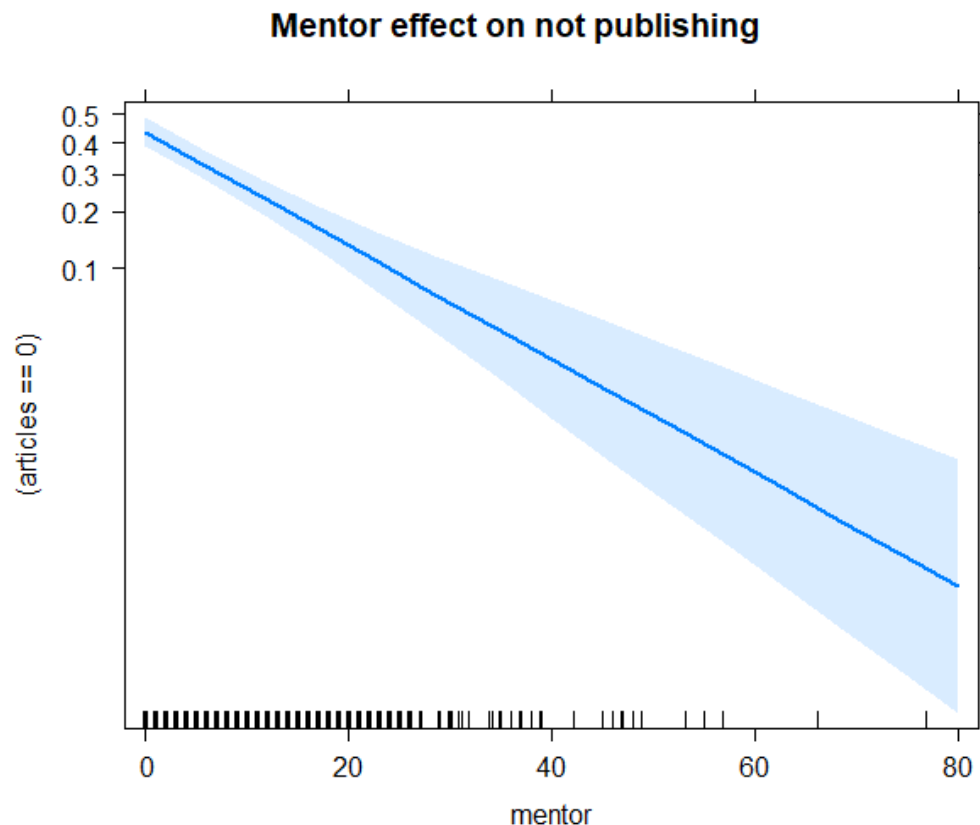
mentor effect plot



The ZIP sub-model for the zero counts (“did not publish”) can also be interpreted visually

- As an approximation, fit a separate logistic model for `articles==0`
- The effect plot for that gives an interpretation of the zero model.

```
phd.zero <- glm((articles==0) ~ mentor, data=PhdPubs, family=binomial)
plot(allEffects(phd.zero), main="Mentor effect on not publishing")
```



What have we learned?

- The simple Poisson regression model fits very badly
 - Standard errors do not reflect overdispersion
 - Inference about model effects is compromised by *overly liberal* tests
- The quasi-poisson model corrects for overdispersion.
 - But doesn't account for excess 0s
- The negative-binomial model provides valid tests and fits the 0 counts well.
 - But it doesn't provide any insight into why there are so many 0s
- The ZIP and ZNB models fit well, and account for the 0s.
 - But they lose here on BIC (and AIC) measures, because they have $2\times$ the number of parameters.
 - For simplicity, I have slighted the analogous hurdle models

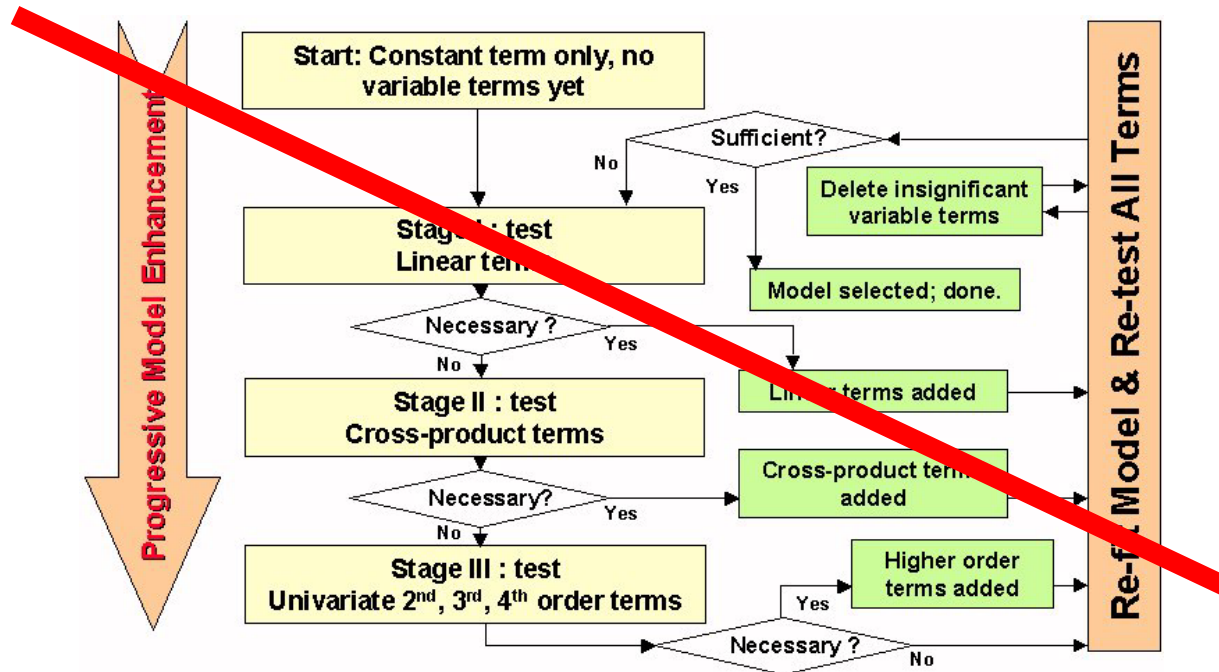
What have we learned?

- The revised ZNB model ([phd.znb1](#)), with only `mentor` predicting 0s, wins on parsimony, and has a simple interpretation.
 - The log odds that a student **does not publish** decrease by 0.61 for every article published by the mentor
 - Each mentor pub increases student publications by about 2.5%
 - \Rightarrow Encourage or help your supervisor to publish!
 - (Or, choose a high publishing one.)
- For this data set, the main substantive interpretation and predicted effects are similar across models. But **details matter!**
- In data sets where there are substantive reasons for excess 0s, the ZI and hurdle models provide **different** explanations.
 - It is not always just a matter of model fit!
 - Hurdle models make the distinction between 0 and > 0 more explicit
 - In ZI models, the interpretation of the mean count is clearer.

What have we forgotten?

“All models are wrong, but some are useful” --- GEP Box

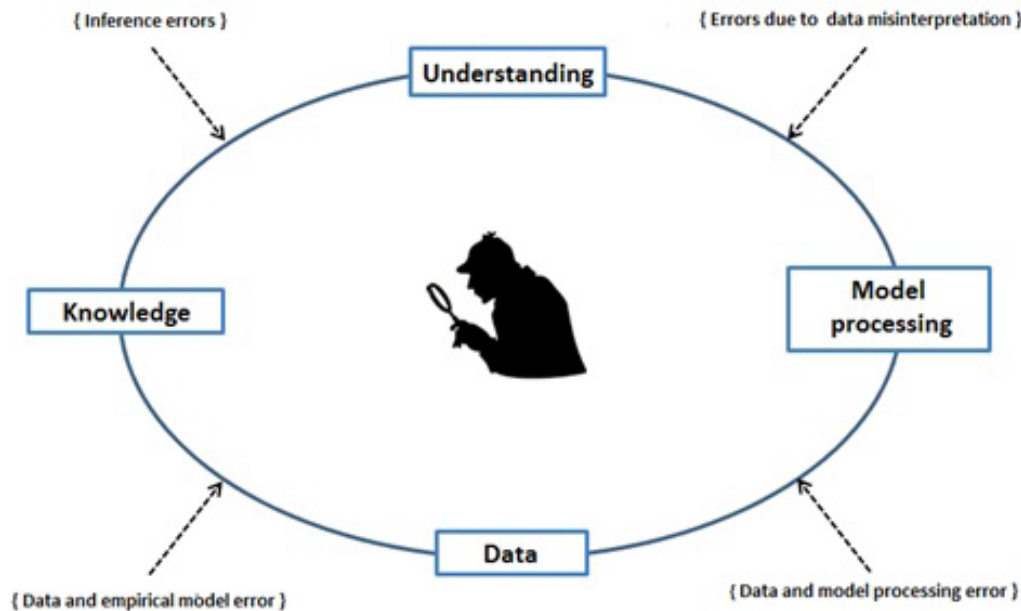
- Model **building** and model **criticism** go hand in hand
- But they don't form a linear series of steps you can put into a flow chart



What have we forgotten?

- Sometimes, you have to go back and revisit decisions made earlier:

Fit → Re-think → Re-fit → Re-interpret



What I missed

- In the initial model, **phdprestige** was NS. I decided to keep it
- In the check for two-way interactions, the interaction **phdprestige:mentor** was borderline ($p = 0.051$)
 - I did a global test for all interactions together
 - This was NS ($p = 0.08$), so I decided to dismiss them all
 - (I wanted to keep the model simple, to go on to other topics: overdispersion, models for excess zeros)

Back to square TWO

- A question in a former class made me reconsider the `phdprestige:mentor` interaction
- Perhaps, the effect of mentor varied with `phdprestige`?

Try this, starting with the negative-binomial, `phd.nbin` (`update()` is your friend)

```
> phd.nbin2 <- update(phd.nbin, . ~ . + phdprestige:mentor)
> Anova(phd.nbin2)
Analysis of Deviance Table (Type II tests)
```

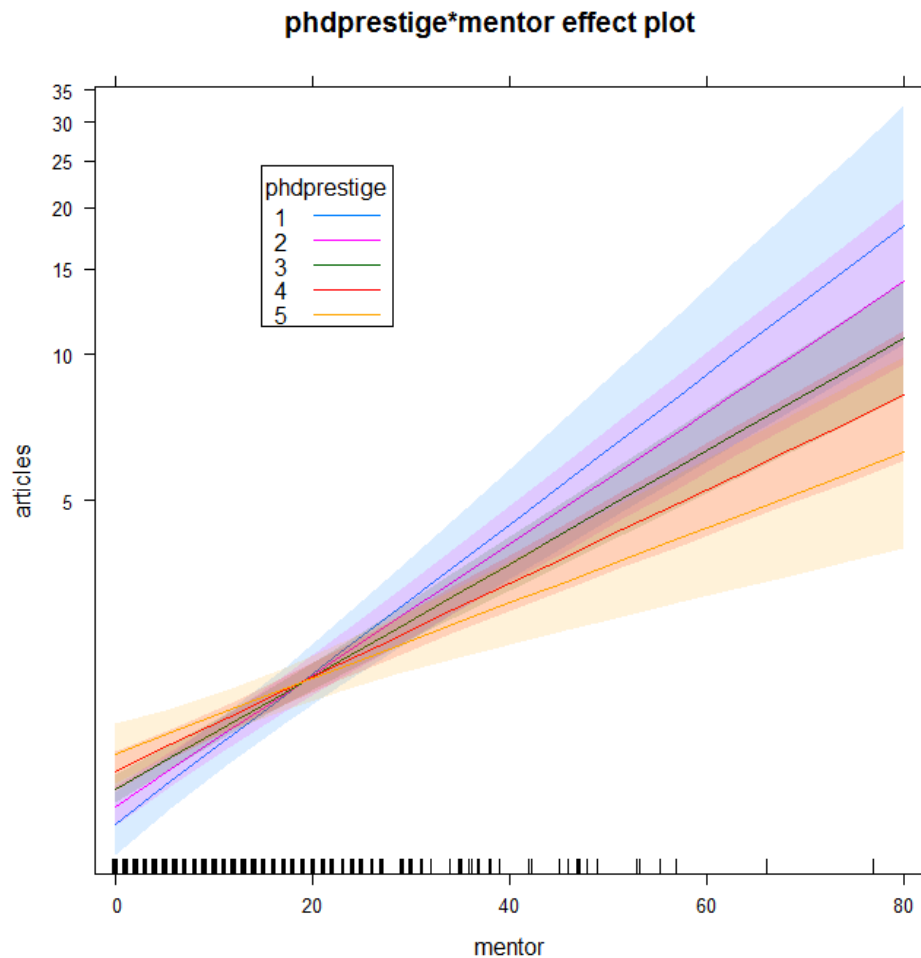
Response: articles

	LR	Chisq	Df	Pr(>Chisq)	
female		9.1	1	0.0026	**
married		3.1	1	0.0762	.
kid5		10.7	1	0.0011	**
phdprestige		0.7	1	0.3921	
mentor		72.8	1	<2e-16	***
phdprestige:mentor		5.6	1	0.0179	*

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Visualize the interaction

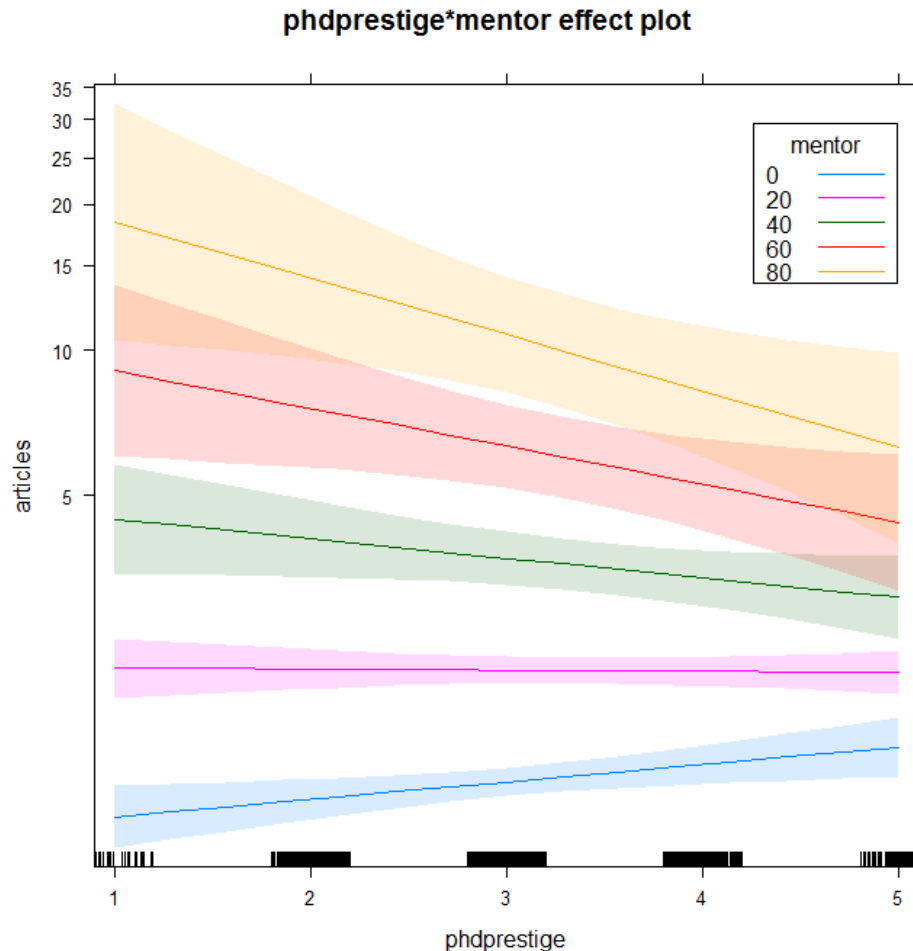
```
phd.effnb2 <- allEffects(phd.nbin2)
plot(phd.effnb2[4], x.var="mentor", multiline=TRUE, ci.style="bands", ...)
```



- An effect plot for `phdprestige*mentor` shows the average over other predictors
- This plot, with `mentor` on the X-axis shows that the slope for `mentor` increases with higher prestige of the student's university

Visualize the interaction— The other way

```
phd.effnb2 <- allEffects(phd.nbin2)
plot(phd.effnb2[4], multiline=TRUE, ci.style="bands", ...)
```



- This plot, with `phdprestige` on the X-axis shows that the slopes change sign depending on the value of `mentor`.
- It explains why the `main` effect of `phdprestige` is near 0.
- The widths of the confidence bands indicate `model uncertainty`— they get wider as `mentor` pubs increase, and `phdprestige` differs from average.

Back to square ONE

Aren't we done yet?

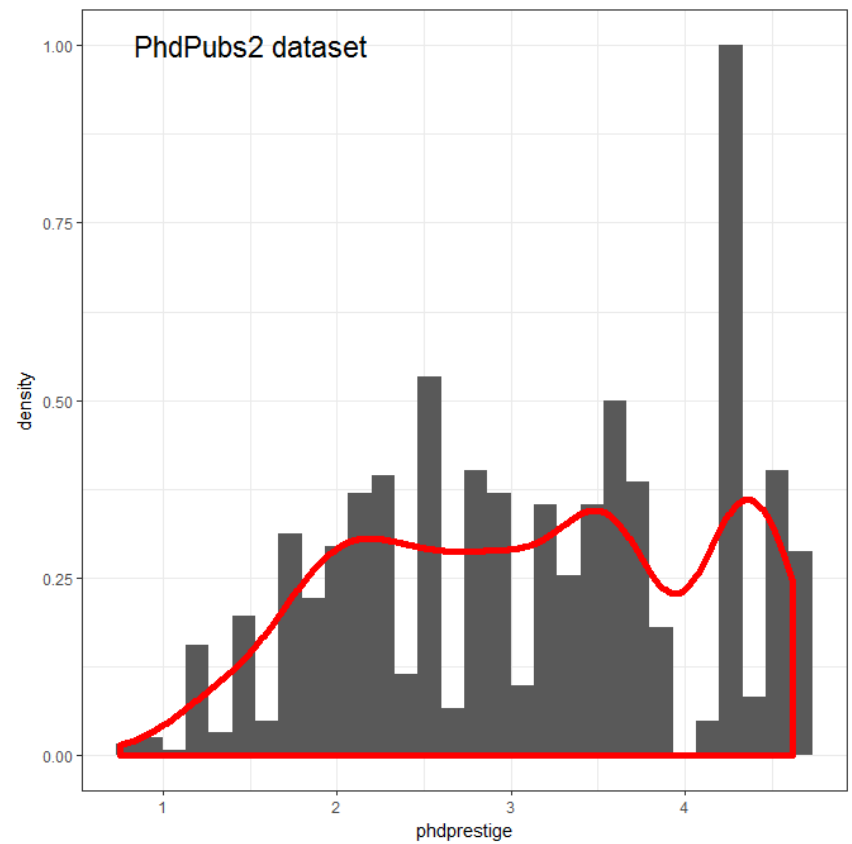
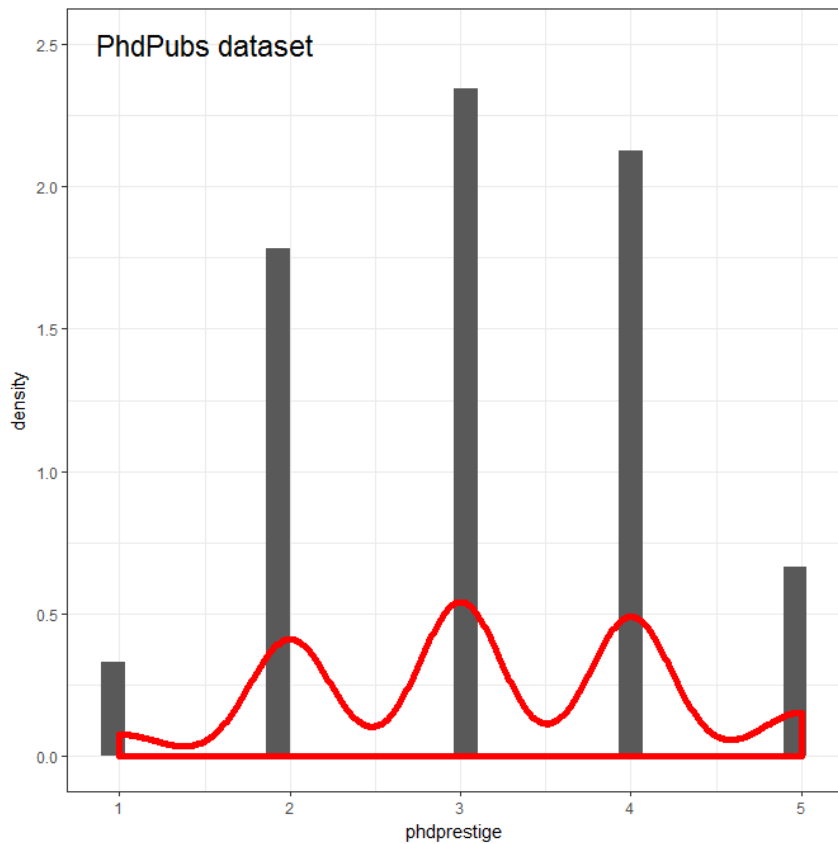
“All data are wrong, but some are useful” – Sitsofe Tsagbey et al.
TAS, 2017

- A nagging doubt: what is the coding for phdprestige?
 - Email from Scott Long: “the higher the number, the more prestigious the program”
 - “PS: The data I used did not categorize the continuous phd scale into discrete categories”
- Found the original Stata data set:

```
library(foreign)
PhdPubs2 <-
  read.dta("http://www.stata-press.com/data/1f2/couart2.dta")
```

Compare distributions

Histograms with smoothed density estimate of the two versions of `phdprestige`
They are very different!



What to do?

Re-run the analysis with the new data set, PhdPubs2

- This could be called a **sensitivity analysis** – does the new data alter conclusions?
- Q: Are the results of the `phd.nbin2` and `phd.znb2` models about the same. A: **YES!**
- Q: Is the interaction of **`phdprestige:mentor`** about the same. A: **YES!**
- Q: Does the effect plot look about the same? A: **YES!**

What else is there?

The PhdPubs example was rather simple

- There were only a few predictors
 - Model selection methods could be based on simple `Anova()`, `coefstest()`, `LRstats()`
 - No need for more complex model selection methods or cross-validation
- Of the quantitative predictors, only `mentor` & `kid5` had important effects
 - The effects of these were sufficiently linear
 - No need to try non-linear effects (`poly(mentor,2)`, `ns(mentor,2)`)
- There turned out to be one important interaction
 - In Psychology, these are called “moderator” effects
 - Interpretation often based on post-hoc tests of simple slopes
 - Interpretation here was simplified in effect plots

Overdispersion in proportions and counts

- So far: count data, $Y \sim \text{Poisson}(\mu)$ — handled overdispersion with `NegBin()`
- What about **proportions** (0,1) or **binomial data** (success/n)?
 - $Y = \text{number of "successes" out of } n \text{ trials: } Y \sim \text{Binomial}(n, \pi)$
 - Logistic / binomial regression assumes $\text{Var}(Y) = n\pi(1-\pi)$
 - But real data often show **more** variation than this
- When/why does overdispersion arise here???
 - π varies across individuals beyond what predictors explain
 - Clustering: cases w/in groups are more similar than between groups
 - Unobserved heterogeneity in the "true" probability

Diagnosing binomial overdispersion

- Residual deviance \gg residual df in `glm(..., family=binomial)`
 - Pearson $\chi^2/\text{df} \gg 1$
- What about: Overdispersion parameter $\hat{\phi} = \chi^2_p / (N - p) > 1$

```
m.bin <- glm(cbind(y, n - y) ~ x1 + x2, data = dat, family = binomial)
summary(m.bin) # check: residual deviance / df
```

- Quick fix: the QuasiBinomial Family
 - inflates SEs for overdispersion, but doesn't model the heterogeneity

```
m.bin <- glm(cbind(y, n - y) ~ x1 + x2, data = dat, family = quasibinomial)
# or for [0,1] proportions
dat$prop <- dat$y / dat$n
m.bin <- glm(prop ~ x1 + x2, data = dat, family = quasibinomial)
```

Mixing probabilities: The Beta distribution

Core idea: instead of a single fixed π , let π itself vary across observations or clusters

$$\pi \sim \text{Beta}(\alpha, \beta), \quad \alpha > 0, \beta > 0$$

Properties of $\text{Beta}(\alpha, \beta)$

- Support: $\pi \in (0, 1)$ — natural for probabilities
- Mean: $E(\pi) = \mu = \alpha / (\alpha + \beta)$
- Variance: $\text{Var}(\pi) = \mu(1-\mu) / (\alpha + \beta + 1)$

Re-parameterize with mean μ and **concentration** $\phi = \alpha + \beta$:

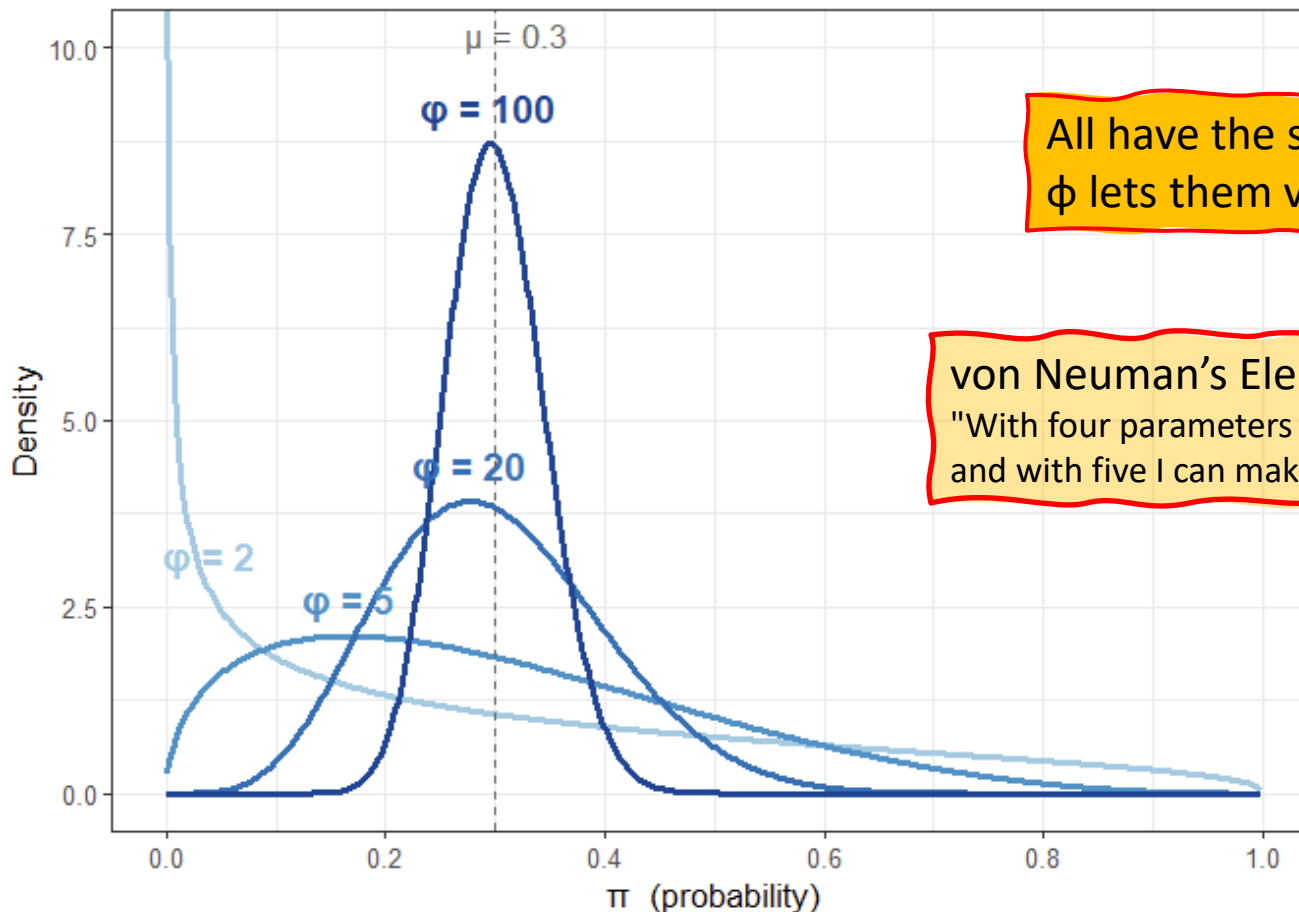
- $\alpha = \mu\phi$, $\beta = (1-\mu)\phi$
- Large ϕ : π **tightly concentrated** around μ (less heterogeneity)
- Small ϕ : π **spread widely** over $(0, 1)$ (more heterogeneity)

Concentration

Concentration, ϕ , gives a natural way to parameterize heterogeneity

Beta distribution: varying concentration ϕ

$\mu = 0.3$ fixed; $\alpha = \mu\phi$, $\beta = (1-\mu)\phi \rightarrow \phi = c(2, 5, 20, 100)$



All have the same mean, $\mu=0.3$
 ϕ lets them vary widely

von Neuman's Elephant:
"With four parameters I can fit an elephant,
and with five I can make him wiggle his trunk."

Fitting Beta-Binomial models in R

Several R packages support beta-binomial regression:

```
library(aod)
m.bb1 <- betabin(cbind(y, n - y) ~ x1 + x2, ~ 1, data = dat)

library(VGAM)
m.bb2 <- vglm(cbind(y, n - y) ~ x1 + x2, family = betabinomial, data = dat)

# allows mixed model effects
library(glmmTMB)
m.bb3 <- glmmTMB(y/n ~ x1 + x2, weights = n, family = betabinomial, data = dat)
```

Interpretation:

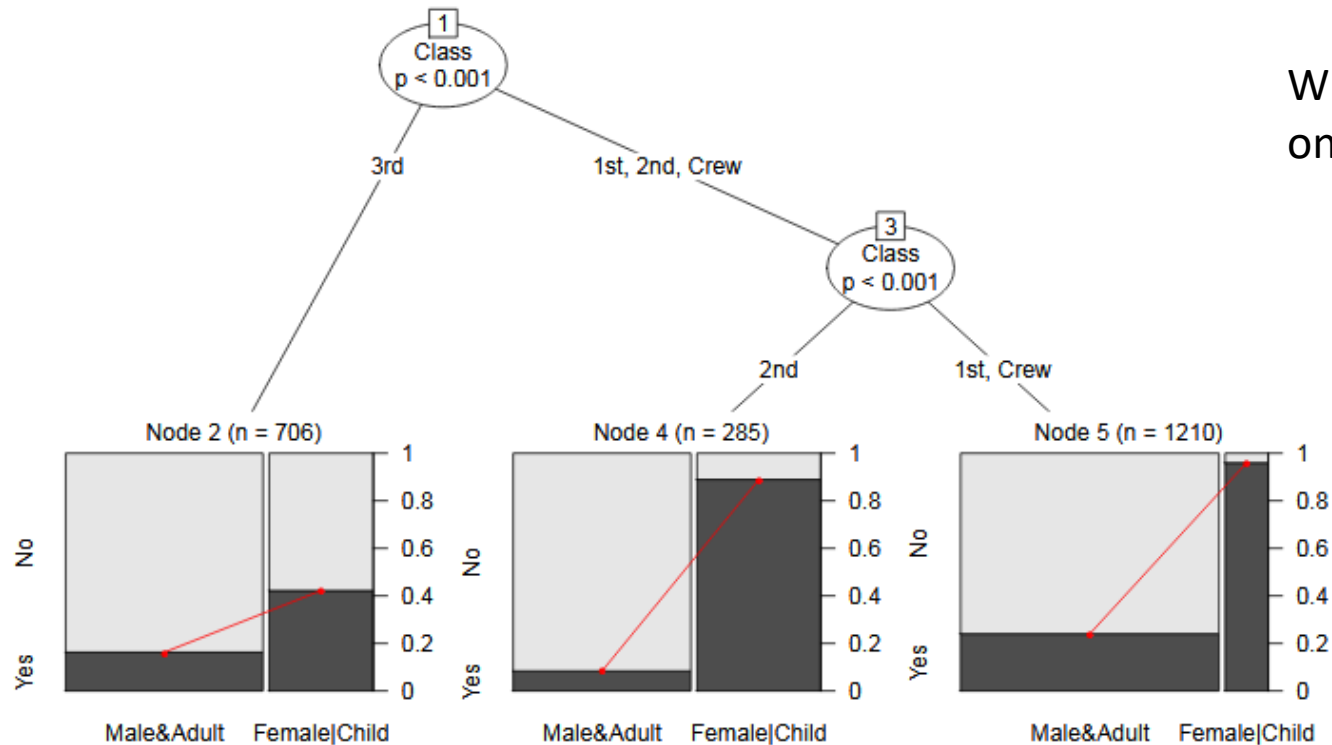
- β : log-odds coefficients — same as logistic regression
- ϕ (or $\hat{\rho}$): estimated concentration / ICC
- $H_0: \phi \rightarrow \infty$ (i.e., $\rho = 0$) — test with LRT vs. binomial

Model comparison: Is overdispersion significant?

```
anova(m.bin, m.bb, test = "LRT")
LRstats(m.bin, m.bb, ...)
```

Other methods: Recursive partitioning

- Recursive partitioning, or **regression trees** are often an attractive alternative to linear models
 - Interactions are handled by partitioning the ranges of variables
 - Or, models can be fit to **subsets** of the data defined by recursive partitioning

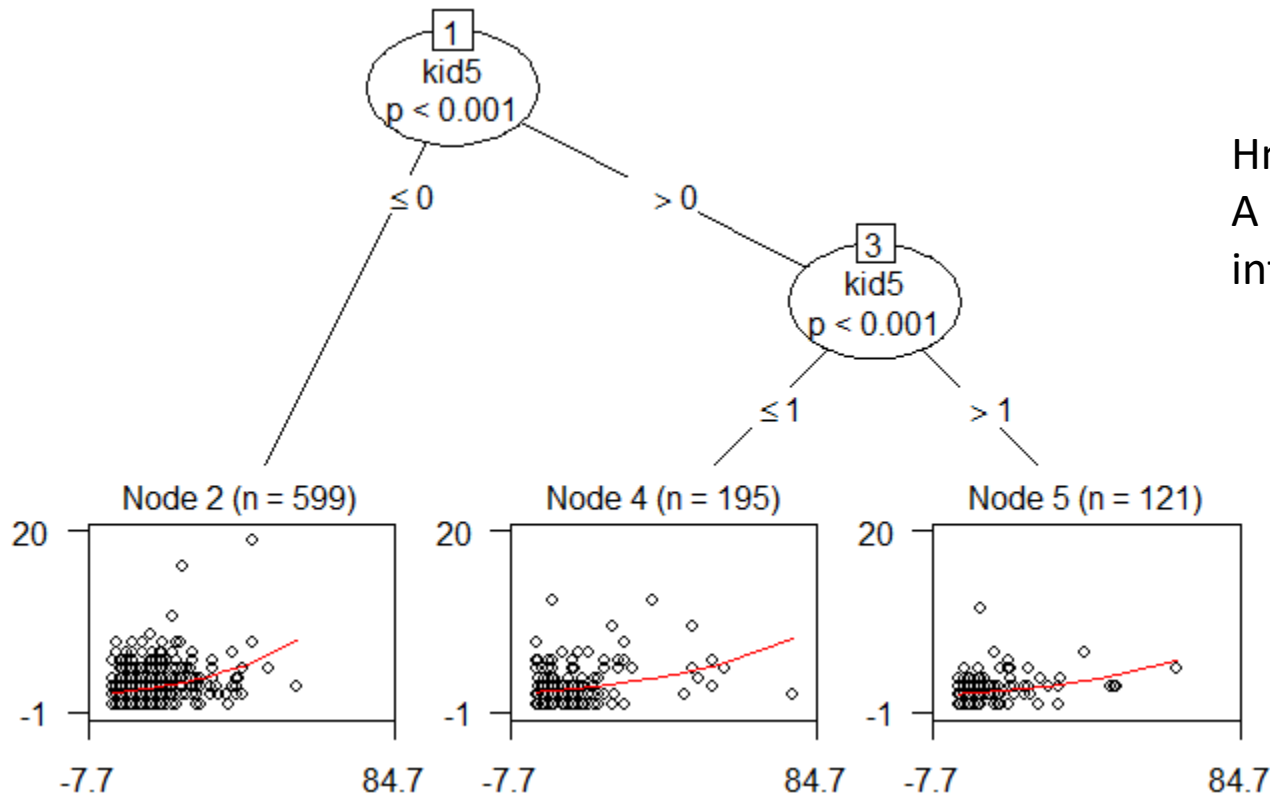


Logistic regression tree fit to the Titanic data with `partykit::glmtree()`

Other methods: Recursive partitioning

Could there be a simpler or different model for the PhdPubs data?

```
library(partykit)
phd.tree <- glmtree(articles ~ mentor| female+married+kid5+phdprestige,
                    data=PhdPubs, family=poisson)
plot(phd.tree)
```



Hmm?
A kid5:mentor
interaction?

Summary

- GLMs provide a unified framework for linear models
 - Different families, all estimated in the same way
 - →link function and associated variance function
- For count data, starting from $\log(\mu) = \mathbf{X} \beta$, $\mu | \mathbf{X} \sim$ Poisson:
 - Overdispersion → quasi-poisson, negative binomial
 - Standard tools for assessing model fit
- Excess zero counts introduce new ideas & methods
 - ZIP model: structural model for the 0s
 - Hurdle model: random model for 0s, 2nd model for $Y > 0$
- In all this, we rely on data & model **plots** for understanding