# Generalized Linear Models

September 19, 2014

## Motivation

It is common in biology to measure responses on different metric spaces:

- **Counts**:
  species abundances, # of mutations, etc.
- **Categories (bounded counts)**:
  counts of alleles, animal choices of prey items, etc.
- **Proportions**:
  percent cover of vegetation, allele frequencies, etc.
- **Continuous all-positive data**:
  mutation rates, survival times, etc.

All of these are on a bounded interval, ie. $[0, \infty)$

## Motivation

Type of metric often associated with several families of distributions:

- **Counts** $\Rightarrow$
  *Poisson*, negative binomial, geometric $\in (0, \infty)$
- **Bounded counts** $\Rightarrow$
  *binomial*, multinomial, hypergeometric $\in (0, \mathrm{N})$
- **Continous all-positive** $\Rightarrow$
  *Gamma*, Weibull, *inverse Gaussian*, exponential $\in [0, \infty)$
- **Proportions** $\Rightarrow$
  beta, logit-normal $\in [0, 1]$

Italicized families are canonical (can be fit with **glm()** in base R). Other families are implemented in various packages.

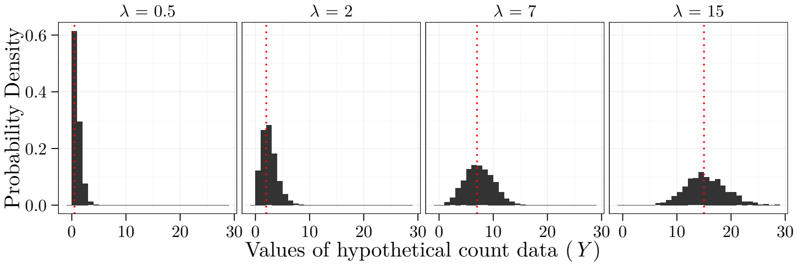
Poisson and binomial families are the most frequently used.

## Counts: Poisson distribution

The **Poisson** distribution models counts $Y$, and has a single parameter (the rate).

$$Y \sim \mathcal{P}o(\lambda)$$

The rate ($\lambda$) is the mean (or expected) count. **The variance equals the mean.**



## Bounded Counts: Binomial

The **binomial** distributions models number of successes $Y$ out of a given number of trials.

Parameters are the probability of success $\phi$ and the fixed # of trials $n$ (fixed = known beforehand).

$$Y \sim \mathcal{B}i(\phi, n)$$

For example, if we measure 3 viable seeds out of 30 seeds,

$$Y = 3 \sim \mathcal{B}i(\phi, 30)$$

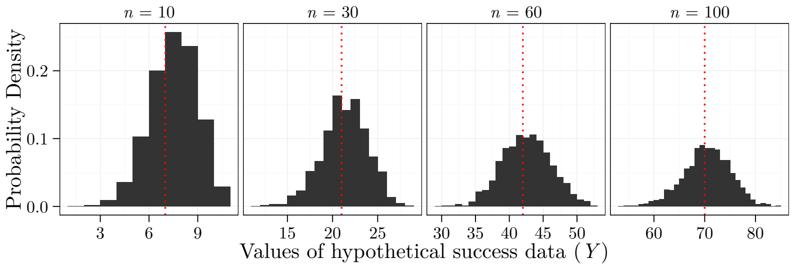
$\phi$ is the quantity of interest, and is bounded at [0, 1]. The mean is $\phi n$, and the variance is a function of the mean and # trials:

$$\phi n(1 - \phi)$$

When trial size ($n$) is 1, is called **Bernoulli** distribution and in regression **logistic regression**.

## Bounded Counts: Binomial

Example of binomial count with same $\phi$ but varying $n$:



As $n$ increases, our estimate of $\phi$ becomes more accurate.

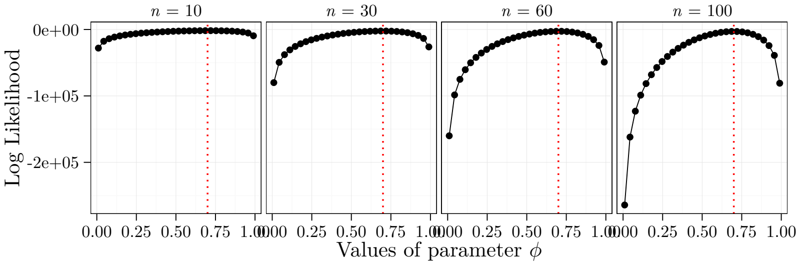
It may be tempting to transform counts into proportions: ie. $Y/n = Z$, then use $Z$ as response variable.

**Don't do this!** (You lose information about the trial size, and thus the innate variability, of the data).

## Bounded Counts: Binomial

Remember that the standard error of an estimate is related to the curvature of the likelihood surface. **The flatter, the less precise.**

Visualise the estimate of probability of success $\phi$ with increasing trial size $n$:



## Regression with Non-normal Distributions

In regression/ANOVA/etc., we want to
relate a set of continous/categorical covariates to the mean of the assumed distribution of our data.

$$Y \sim \text{Distribution}(\text{Parameters})$$
$$\mathbb{E}[y] = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$$

This involves solving for the regression coefficients $\beta$ ...

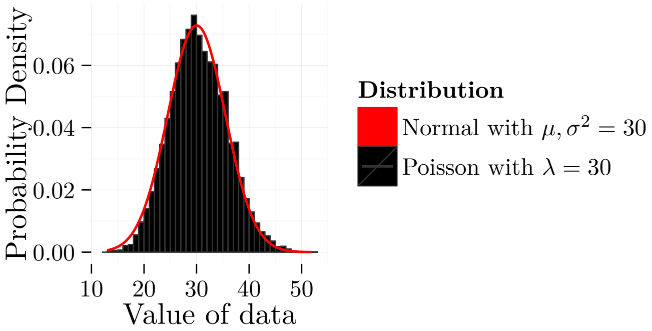
But hard to do multivariate optimization on a bounded interval!

Easy to get values of $\beta$ that lead to an $\mathbb{E}[Y]$ outside of the bounds.

## Normal approximation

Sometimes fitting a normal linear model to non-normal data, provides ‘reasonable’ answers (**normal approximation**).

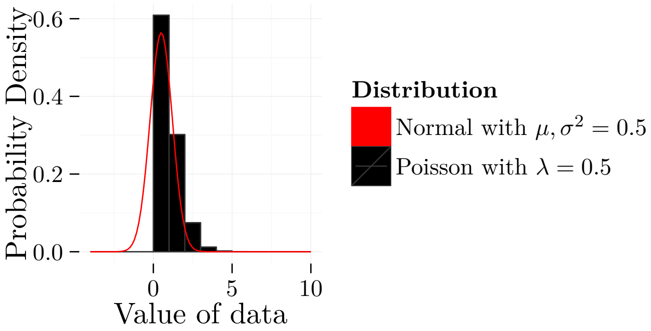
For example, when the rate parameter of a Poisson distribution is large,



# Normal approximation

But often the normal approximation is **rubbish**, especially for multi-parameter models (ie. multiple regression).

The normal PDF either fits the data poorly, or extends to impossible values (ie. negative counts)



## Introducing the GLM

**Generalized Linear Models** (GLMs) allow non-normal (bounded) data to be modelled as a linear function.

GLMs are a flexible extension of normal linear models... and include normal linear models as a special case.

Developed in 70s/80s, these have quickly become the standard mode of analysis for non-normal data, particularly counts.[1]

[1] McCullagh and Nelder *Generalized Linear Models* (1989) is the definitive reference

## Introducing the GLM

**How does a GLM work?**

- We assume the data follow a distribution.
- The GLM transforms the mean of the distribution to an unbounded space on $[-\infty, \infty]$.

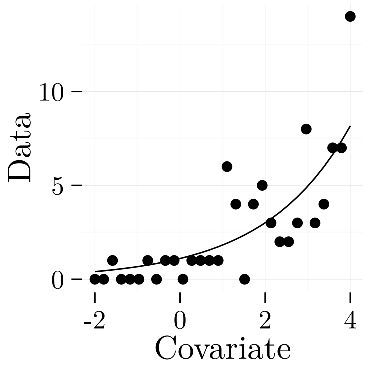
  **For example with a log-transformation:**

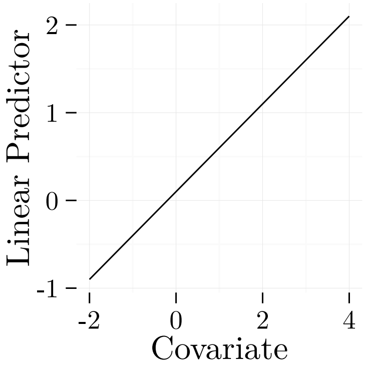
  - $\log \mathbb{E}[Y]$ goes from $[0, \infty]$ to $[-\infty, \infty]$.
  - If $0 < y < 1$ then $\log y < 0$,
    If $y > 1$ then $\log y > 0$.
- This transformation is termed a **link function** (*linking* the linear model with the distribution of the data).
- The transformed mean is called the **linear predictor**.
- **The raw data are not transformed.**

## Introducing the GLM

An intuition for the link function in a regression context:

Take a bounded, curvy line and **make it straight**:





GLM finds the slope and intercept of the straight line the best fit the data when back-transformed to a curvy line.

## Introducing the GLM

Basic recipe for a GLM:

1. Assume distribution for response (ie. Poisson)

$$Y \sim \mathcal{P}o(\lambda)$$

2. Choose link function appropriate for distribution
   (ie. log for Poisson)
3. Take the expected value of the response distribution ...
   transform ...
   and write as linear model.

$$\log \mathbb{E}[Y] = \log \lambda = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

4. Optimize $\beta$ to maximize likelihood of data.

R automates this except for the choice of distribution and link function.

## Introducing the GLM

The normal linear model is just a species case of the GLM.

The link function here is the ‘identity’ function (does nothing).

$$\begin{aligned} Y &\sim \mathcal{N}(\mu, \sigma) \\ \mathbb{E}[Y] = \mu &= \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \end{aligned}$$

Note that there are multiple potential link functions per response distribution.

For example, a normal model with a log-link is just a particular type of nonlinear model.

## Interpreting the Link Function

All the regression coefficients, standard errors, etc. are on the **scale of the linear predictor** (the transformed scale).

**Not** on the scale of the response (the bounded interval).

Therefore, the regression coefficients can no longer be interpreted as a unit change in covariate $\Rightarrow$ unit change in response.

Regression coefficients must be interpreted with reference to the link function.

This is a common source of confusion.

## Interpreting the Link Function: log-link

The most common link for the Poisson distribution is the log-link.

$$\log \mathbb{E}[\,Y] = \beta_1 x_1$$

Exponentiate to put on the scale of the data:

$$\mathbb{E}[\,Y] = e^{\beta_1 x_1} = e^{\beta_1 x_1} = (e^{\beta_1})^{x_1}$$

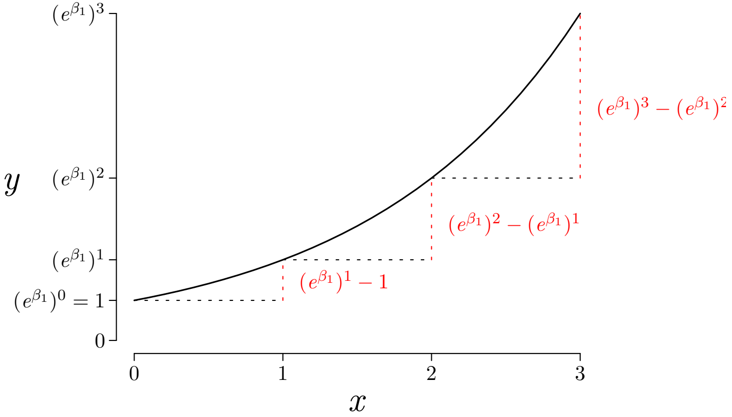
Therefore, for each unit change in $x_1$ there is a multiplicative change in $\mathbb{E}[\,Y]$ of $e^{\beta_1}$.

For example, if $x_1 = 3$,

$$\mathbb{E}[\,Y] = (e^{\beta_1})^3 = e^{\beta_1} \cdot e^{\beta_1} \cdot e^{\beta_1}$$

## Interpreting the Link Function: log-link

Let's visualise $\log \mathbb{E}[Y] = \beta_1 x_1$ on the untransformed scale.



If increasing the value of $x$, the net increase in $y$ depends on the current value of $y$.

## Interpreting the Link Function: logit-link

The default link for binomial models is the **logit link**.

$$\mathbb{E}[Y] \propto \phi$$
$$\text{logit}\phi = \log \frac{\phi}{1-\phi} = \beta_1 x_1$$

In binomial distribution $\phi$ is the probability of a binary event (let's say DYING).

Then $1 - \phi$ is the probability of LIVING.

The **odds** of DYING are $\frac{\phi}{1-\phi}$.

If the **odds** are 3, you/it/them/us are three times more likely to DIE than to LIVE.

## Interpreting the Link Function: logit-link

The logit transformation is the logged odds.

$$\text{logit } \phi = \log \frac{\phi}{1-\phi}$$

Thus the regression coefficient from a logit-link model is the multiplicative increase in the odds of the event occurring.

For example, in:

$$\frac{\phi}{1-\phi} = \exp\{\beta_1 x_1\} = (e^{\beta_1})^{x_1}$$

A one-unit change in $x_1$ causes a multiplicative increase of $\exp\{\beta_1\}$ in the odds of DYING.

## Distributions, Links, and R

Canonical families (all using **glm()** in base R)

| Family | Links | Data Type |
|---|---|---|
| Poisson | **log**, sqrt, identity | Counts |
| Binomial | **logit**, probit, cloglog | Successes/#Trials |
| Gamma | **inverse**, log, identity | Continuous positive |
| Inverse Gaussian | **inverse**$^2$, log, identity | Continuous positive |
| Quasipoisson | same as above | Overdispersed counts |
| Quasibinomial | same as above | Overdispersed Successes/#Trials |
| Gaussian | **identity**, log, sqrt | Normal data |

Non-canonical but very useful families

| Family | Links | Data Type | Function(Package) |
|---|---|---|---|
| Negative binomial | **log**, sqrt, identity | Counts | glm.nb(MASS) |
| Beta | **logit**, probit, cloglog | Proportions | betareg(betareg) |

And there are many more. Default links in **bold**.

## Goodness of Fit

The extension of sums of squares to maximum likelihood framework is called **deviance**.

Deviance is used for optimization (fitting), assessing goodness-of-fit, hypothesis testing. Defined as:

$$-2 \cdot (\text{log likelihood of model}) + 2 \cdot (\text{log likelihood of saturated model})$$

The **saturated** model is a model where the data is perfectly explained–completely overfit–and gives a baseline likelihood.

Hence **lower deviance is better.**

## Example data

Size-selectivity in black bears feeding on salmon (Cunningham et al. *American Naturalist* 2013).

- Count variable: total number of salmon deaths
- Binomial variable: salmon deaths by bear or other?
- Covariates: Salmon size, sex, and year

## Example data

```r
kill_count <- read.csv("counts.csv")
str(kill_count, max.level = 1, give.attr = F,
    width = 40)

## 'data.frame': 526 obs. of  4 variables:
##  $ year         : int  1997 1997 1997 1997 1997 1997 1997 1997 1997 1997 ...
##  $ sex          : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
##  $ salmon_length: int  390 400 410 420 430 440 450 460 470 480 ...
##  $ deaths       : int  0 0 2 0 0 0 2 1 0 0 ...

kill_type <- read.csv("kills.csv")
str(kill_type, max.level = 1, give.attr = F,
    width = 40)

## 'data.frame': 38 obs. of  3 variables:
##  $ death_by_other: int  0 1 2 12 23 86 144 265 213 185 ...
##  $ death_by_bear : int  1 0 0 1 1 7 14 15 19 19 ...
##  $ salmon_length : int  230 240 250 260 270 280 290 300 310 320 ...
```

## Fitting GLMs in R

For Poisson model use:
**glm(<formula>, <family>(<link>), <data>)**

```r
count_mod <- glm(deaths ~ year + sex + salmon_length,
    poisson(log), kill_count)
```

Aside from the family-link argument, the syntax is identical to **lm()**. The response must be counts (positive integers and zeros).

```r
head(kill_count$deaths)

## [1] 0 0 2 0 0 0
```

## Fitting GLMs in R

For a binomial model,

- If multiple counts in observation, formula must be: **cbind(<success counts>, <failure counts>) ∼ covariates**
- If binary response (trial size is 1), formula must be: **<event> ∼ covariates**
  In this case **<event>** is binary (0/1).

```
death_mod <- glm(cbind(death_by_bear, death_by_other) ~
    salmon_length, binomial(logit), kill_type)
## structure of response
with(kill_type, cbind(death_by_bear, death_by_other))[1:3,
    ]

##      death_by_bear death_by_other
## [1,]             1              0
## [2,]             0              1
## [3,]             0              2
```

## Output from a GLM

Most summary/extractor functions from LMs work for GLMs.

```
summary(death_mod)


##
## Call:
## glm(formula = cbind(death_by_bear, death_by_other) ~ salmon_length,
##     family = binomial(logit), data = kill_type)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -7.319  -2.466  -0.437   1.533   4.740
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.606579   0.100539   -45.8   <2e-16 ***
## salmon_length  0.009777   0.000232    42.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2225.45  on 37  degrees of freedom
## Residual deviance:  334.49  on 36  degrees of freedom
## AIC: 539
##
## Number of Fisher Scoring iterations: 4
```

## Output from a GLM

Extractor functions ...

**formula(<glm object>)** : returns formula
**resid(<glm object>, <type>)** : returns residuals
**fitted(<glm object>, <type>)** : returns fitted values
**coef(<glm object>)** : returns estimated coefficients
**vcov(<glm object>)** : variance-covariance of coefficients
**model.matrix(<glm object>)** : matrix of predictors
**logLik(<glm object>)** : returns log-likelihood of model

**<type>** is used to specify the scale.

```
head(predict(count_mod, type = "response"))


##     1     2     3     4     5     6
## 6.242 6.155 6.069 5.984 5.901 5.818
```

## Asymptopia

Output from GLMs (residuals, estimates, hypothesis tests, etc.) behaves like normal linear models **asymptotically**.

**Asymptotically** means as the # of data points approaches $\infty$.

Realistically, at least 50-100 data points (depending on # of zeros in data).

With small sample sizes, **don't expect GLMs to behave like LMs**.

With small sample sizes, we can use simulation to perform diagnostics.

## Diagnostics: Residuals

Residuals are the main tool for diagnostics.

Because the link function connects the model to the data, we have many different types of residuals. The two most useful:

- **Deviance residuals**: the contribution of each data point to the model's deviance. Used for checking heteroskedasticity, goodness of fit.
- **Working residuals**: the residuals used in the fitting process, used for calculating partial residuals.

Use **<type>** in **resid(<glm object>, <type>)** to specify working or deviance.

```r
resid(count_mod, type = "deviance")
resid(count_mod, type = "working")
```

## Diagnostics: Residuals

Deviance residuals are asymptotically normally distributed. Use to check heteroskedasticity, normality, and linearity.

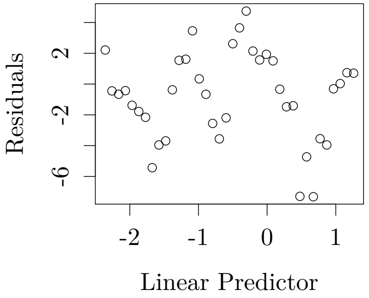
Use **predict(<glm object>, type = "link")** to get fitted values on scale of link function.

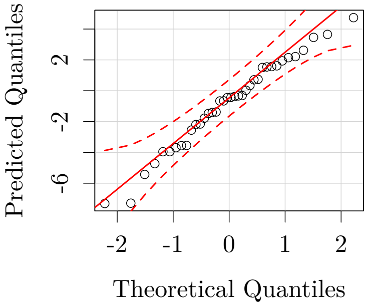
```
head(predict(count_mod, type = "link"))

##     1     2     3     4     5     6
## 1.831 1.817 1.803 1.789 1.775 1.761
```

# Diagnostics: Residuals

```r
library(car)
par(mfrow = c(1, 2))
plot(predict(death_mod, type = "link"), resid(death_mod,
    type = "deviance"), xlab = "Linear Predictor",
    ylab = "Residuals")
qqPlot(resid(death_mod, type = "deviance"),
    ylab = "Predicted Quantiles", xlab = "Theoretical Quantiles")
```

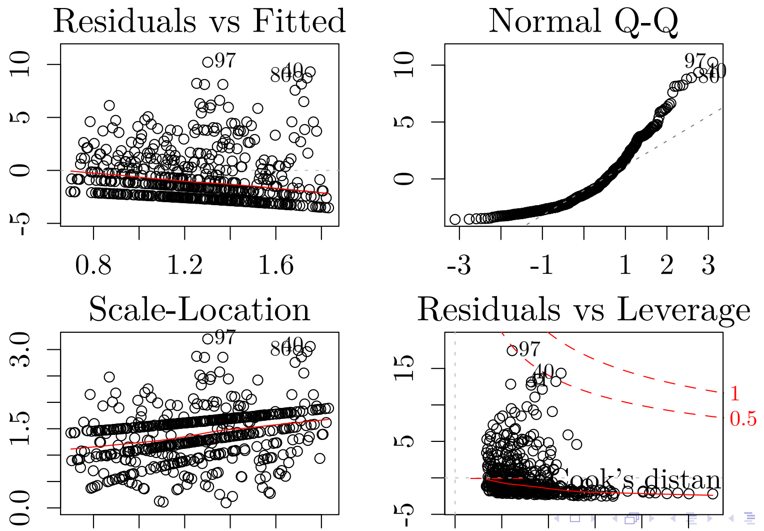




## Diagnostics: Residuals

Default plotting function for glm objects makes these plots for you.

```r
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
plot(count_mod)
```



## Diagnostics: Overdispersion

**Overdispersion** is the biggest problem with Poisson/Binomial GLMs.

Both Poisson/Binomial have a **fixed** variance: the variance is completely determined by the mean (also # trials for binomial).

**This is a restrictive assumption**. Most of the time, biological data have extra-Poisson and extra-Binomial variation.

Unmeasured variables cause more variation than we would expect under a 'pure' process.

## Diagnostics: Overdispersion

Why/how is this a problem?

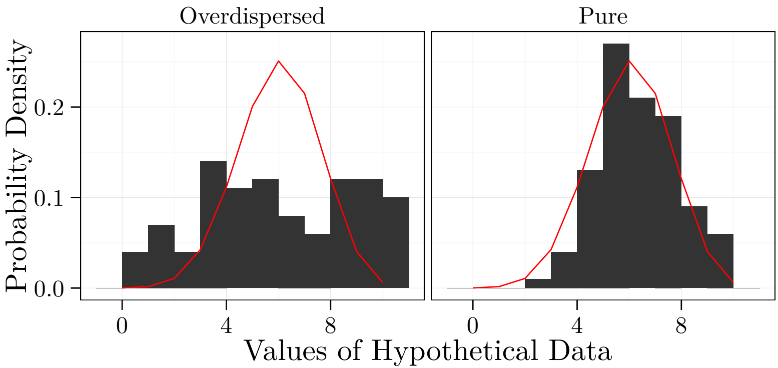
If we assume there is less variation than there actually is, we are overconfident in the accuracy our estimates.

In other words, the model is **anti-conservative**: it underestimates standard errors $\Rightarrow$

- P-values shrink,
- Type-I error inflates.

## Diagnostics: Overdispersion

Visualize this with simulated data:



Both are from binomial distributions with the same parameters, but left is overdispersed. Intuitively: we should be less confident in the mean estimate from the overdispersed data.

## Diagnostics: Overdispersion

How to diagnose overdispersion?

In **summary(<glm object>)**, look for ratio of residual deviance to degrees freedom. **The ideal ratio is 1**.

```
summary(count_mod)


##
## Call:
## glm(formula = deaths ~ year + sex + salmon_length, family = poisson(log),
##     data = kill_count)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.533  -2.469  -1.378   0.448  10.211
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   78.910631  10.266234    7.69  1.5e-14 ***
## year          -0.038323   0.005111   -7.50  6.4e-14 ***
## sexm          -0.383196   0.045926   -8.34  < 2e-16 ***
## salmon_length -0.001405   0.000383   -3.67  0.00024 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 3671.5  on 525  degrees of freedom
## Residual deviance: 3537.4  on 522  degrees of freedom
```

## Diagnostics: Overdispersion

Can formally test as a $\chi^2$ statistic, against null hypothesis of no overdispersion:

```
pchisq(count_mod$deviance, count_mod$df.residual,
    lower = F)
```

```
## [1] 0
```

Also notice that the estimated std. errors and p-values are extremely low.

```
summary(count_mod)$coef
```

```
##                 Estimate Std. Error z value  Pr(>|z|)
## (Intercept)    78.910631  10.266234   7.686 1.513e-14
## year           -0.038323   0.005111  -7.499 6.448e-14
## sexm           -0.383196   0.045926  -8.344 7.191e-17
## salmon_length -0.001405   0.000383  -3.668 2.445e-04
```

## Modelling overdispersion

Two common approaches: **quasi-likelihood** and **mixtures**.

The basic idea in both approaches is that we have two sources of variance:

- **Variance of the base distribution**,
  for example a Poisson distribution has variance equal to the mean.
- **And extra variance**,
  which we model with a separate parameter.

Functionally, approaches differ in how the variance increases with the mean: linearly (quasi) or asymptotically/linearly/exponentially (depending on mixture)

## Modelling overdispersion: quasi-families

A quasi-family GLM has a variance that is a linear function of the mean:

$$\mathrm{Var}[Y] = \theta \cdot \mathbb{E}[Y]$$

$\theta$ is the **dispersion parameter**.

So for a Poisson distribution, the variance still increases with the mean but is not constrained to a 1-to-1 relationship.

- Pros: can be implemented directly in **glm()**
- Cons: likelihood becomes quasi-likelihood, AIC becomes quasi-AIC, not implemented for more complex models (ie. GLMMs)

## Modelling overdispersion: quasi-families

Use **family=quasi<whatever>** to fit quasi-model.

```
count_mod_quasi <- glm(deaths ~ year + sex +
    salmon_length, quasipoisson, kill_count)
summary(count_mod_quasi)


##
## Call:
## glm(formula = deaths ~ year + sex + salmon_length, family = quasipoisson,
##     data = kill_count)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.533  -2.469  -1.378   0.448  10.211
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   78.91063   30.70191    2.57   0.0104 *
## year          -0.03832    0.01528   -2.51   0.0125 *
## sexm          -0.38320    0.13734   -2.79   0.0055 **
## salmon_length -0.00140    0.00115   -1.23   0.2206
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 8.944)
##
##     Null deviance: 3671.5  on 525  degrees of freedom
## Residual deviance: 3537.4  on 522  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

Note that std. errors, p-values are much more reasonable.

## Modelling overdispersion: mixtures

In a mixture, add a ‘residual’ (data-point specific random component) to the linear predictor.

$$\begin{aligned} Y &\sim \mathcal{P}o(\lambda) \\ \lambda &\sim \text{Distribution(Parameters)} \end{aligned}$$

Most familiar is the negative binomial for count data:

$$\begin{aligned} Y &\sim \mathcal{P}o(\lambda) \\ \mathbb{E}[Y] = \lambda &\sim \mathcal{G}amma(\frac{\mu^2}{\theta}, \frac{\theta}{\mu}) \end{aligned}$$

A mixture of Poisson and Gamma distributions. The data-point specific random component is a Gamma variate with mean $\mu$ and variance $\theta$.

$$\log \mu = \beta_1 x_1 + \beta_2 x_2 + ...$$

## Modelling overdispersion: mixtures

In R, use **glm.nb()** from MASS package. Works just like **glm()** function, but don't specify the family.

```
library(MASS)
count_mod_nb <- glm.nb(deaths ~ year + sex +
    salmon_length, kill_count)
summary(count_mod_nb)
```

```
##
## Call:
## glm.nb(formula = deaths ~ year + sex + salmon_length, data = kill_count,
##     init.theta = 0.4742159769, link = log)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.584  -1.370  -0.561   0.163   2.453
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   69.34445   30.16887    2.30   0.0215 *
## year          -0.03337    0.01501   -2.22   0.0262 *
## sexm          -0.35303    0.13647   -2.59   0.0097 **
## salmon_length -0.00226    0.00111   -2.03   0.0424 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.4742) family taken to be 1)
##
##     Null deviance: 563.47  on 525  degrees of freedom
## Residual deviance: 548.53  on 522  degrees of freedom
## AIC: 2428
##
## Number of Fisher Scoring iterations: 1
##
```

## Modelling overdispersion: mixtures

A normal mixture is one where we add a normally distributed 'residual' to the linear predictor:

$$\begin{aligned} Y &\sim \mathcal{P}o(\lambda) \\ \log \mathbb{E}[Y] = \log \lambda &= \beta_1 x_1 + \beta_2 x_2 + \epsilon \\ \epsilon &\sim \mathcal{N}(0, \sigma) \end{aligned}$$

Here $\epsilon$ is the 'residual' and effectively accounts for variance additonal to that of the Poisson.

This is called a **Poisson log-normal mixture**.

## Modelling overdispersion: mixtures

Fit binomial-logit-normal and Poisson-log-normal mixtures as a mixed-effects model (using **glmer()** in lme4)

```
library(lme4)
kill_count$id <- 1:nrow(kill_count)
glmer(deaths ~ scale(year) + sex + scale(salmon_length) +
    (1 | id), family = poisson, data = kill_count)


## Warning:  Model failed to converge with max|grad| = 0.00585849 (tol = 0.001)

## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: deaths ~ scale(year) + sex + scale(salmon_length) + (1 | id)
##    Data: kill_count
##      AIC      BIC   logLik deviance df.resid
##     2437     2458    -1214     2427      521
## Random effects:
##  Groups Name        Std.Dev.
##  id     (Intercept) 1.46
## Number of obs: 526, groups: id, 526
## Fixed Effects:
##          (Intercept)           scale(year)                  sexm
##               0.4829                0.0411               -0.3429
## scale(salmon_length)
##               0.0329
```

This syntax is new, and particular to mixed-effects models (more in later weeks).

## Diagnostics: simulation

The most flexible and powerful way to diagnose a model is through simulation.

The basic idea:

1. We fit a model to our data.
2. We are unsure if the model fits our data, in some regard.
3. So we simulate datasets from the model fit.
4. These simulated datasets are by definition consistant with the model.
5. We then compare our observed data to the simulated data (using some statistic of interest).
6. If observation and simulation are consistant, then we conclude that our data are consistant with the model.

## Diagnostics: simulation

To simulate new datasets, use **simulate(<model>, <number of simulations>)**.

```
head(simulate(count_mod, 5))

##   sim_1 sim_2 sim_3 sim_4 sim_5
## 1     6     8     5     4     1
## 2     8     3     5     6     4
## 3     3     3     3     9     6
## 4     9     4     2    10     5
## 5     7     3     1     7     5
## 6    10     4     6     8     5
```

Rows are observations corresponding to original data, columns are new datasets.

## Diagnostics: simulation

An example: is the number of zeros in our data consistant with a Poisson model?

The steps:

1. Count number of zeros in data
2. Fit Poisson model to data.
3. Simulate data from model many times.
4. For each simulated dataset, calculate number of zeros.
5. Build distribution of expected number of zeros from simulations.
6. See if observed number of zeros is consistant with simulated distribution.

## Diagnostics: simulation

Example application:

- Count number of zeros in data

```r
obs_zeros <- sum(kill_count$deaths == 0)
```

- Fit Poisson model to data:

```r
count_mod <- glm(deaths ~ year + sex + salmon_length,
    poisson(log), kill_count)
```

- Simulate data from model many times

```r
new_dat <- simulate(count_mod, 1000)  ## 1000 new datasets
```

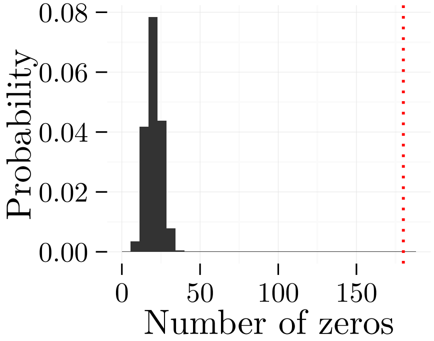
- For each simulated dataset, calculate number of zeros.

```r
sim_zeros <- apply(new_dat, 2, function(x) sum(x ==
    0))
```

## Diagnostics: simulation

- Build empirical distribution of # zeros expected under model, and
- Compare observed # zeros (red) to expected # zeros (hist)

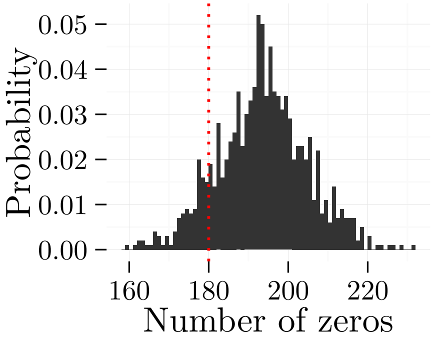
```r
ggplot(data.frame(x = sim_zeros), aes(x = x)) +
    geom_histogram(aes(y = ..density..)) +
    geom_vline(xintercept = obs_zeros, col = "red",
        size = 1, lty = 3) + theme_minimal() +
    xlab("Number of zeros") + ylab("Probability")
```



## Diagnostics: simulation

How about with the negative binomial model?

```
new_dat_nb <- simulate(count_mod_nb, 1000)
sim_zeros_nb <- apply(new_dat_nb, 2, function(x) sum(x ==
    0))
ggplot(data.frame(x = sim_zeros_nb), aes(x = x)) +
    geom_histogram(aes(y = ..density..),
        binwidth = 1) + geom_vline(xintercept = obs_zeros,
    col = "red", size = 1, lty = 3) + theme_minimal() +
    xlab("Number of zeros") + ylab("Probability")
```



## Likelihood Ratio Tests

A common tool for model selection/hypothesis testing in GLMs (and many other models) is the **likelihood ratio test**.

Basic idea:

- improvement in fit = difference in deviance between models $= \Delta D$.
- increase in complexity = difference in number of parameters between models $= \Delta P$.

We ask: is improvement in fit greater than we would expect at random, given increase in complexity?

Turns out the null distribution for $\Delta D$ is $\chi^2$ with degrees of freedom $\Delta P$.

## Likelihood Ratio Tests: Application

In R, calculate likelihood ratio test with
**anova(<model 1>, <model 2>)**

```
full_model <- count_mod_nb
reduced_model <- update(full_model, . ~ . -
    sex)
anova(full_model, reduced_model)
```

```
## Likelihood ratio tests of Negative Binomial Models
##
## Response: deaths
##                         Model  theta Resid. df    2 x log-lik.   Test    df
## 1       year + salmon_length 0.4658       523           -2424
## 2 year + sex + salmon_length 0.4742       522           -2418 1 vs 2     1
##   LR stat. Pr(Chi)
## 1
## 2    6.292 0.01213
```

Models should be nested (differing in single covariate).

## Likelihood Ratio Tests: Application

Two approaches:

- **Sequential**: add parameters from null model
- **Marginal**: remove parameters from full model

Analogous to Type I and Type III sums of squares. Easy way to do all marginal tests is with **drop1()**:

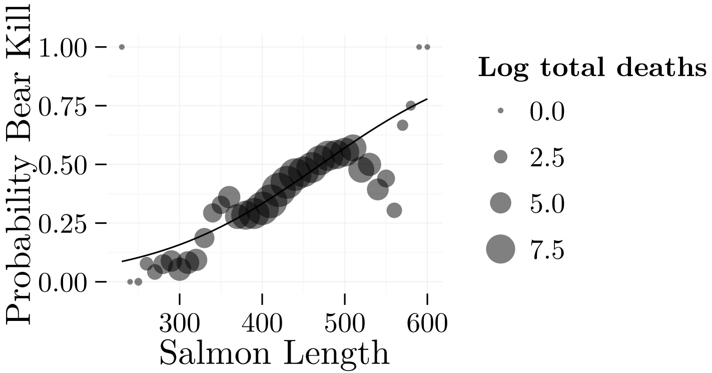
```
drop1(full_model, test = "Chisq")  ## gives all marginal likelihood ratio tests
```

```
## Single term deletions
##
## Model:
## deaths ~ year + sex + salmon_length
##              Df Deviance  AIC  LRT Pr(>Chi)
## <none>                549 2426
## year          1       554 2430 5.77    0.016 *
## sex           1       555 2430 6.35    0.012 *
## salmon_length 1       550 2426 1.64    0.200
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Plotting binomial fits

GLM returns the predicted probabilities of success, not counts of successes. To convey the sample size for each proportion, I often use the size of the point.

```r
preds <- predict(death_mod, type = "response")  ## these are proportions! Not counts!
ggplot(data.frame(preds, kill_type), aes(x = salmon_length,
    y = death_by_bear/(death_by_bear + death_by_other))) +
    geom_point(aes(size = log(death_by_bear +
        death_by_other)), alpha = 0.5) +
    geom_line(aes(y = preds)) + theme_minimal() +
    xlab("Salmon Length") + ylab("Probability Bear Kill") +
    scale_size_continuous(name = "Log total deaths")
```



## Additional topics I recommend you look up:

- complete separation: when a covariate completely predicts response, hell ensues
- zero-inflation: some additional process generates zeros in our data, more than would be expected given distribution
- non-canonical families: beta (proportions), Weibull (survival models), Pareto (truncated continuous data), t (robust normal models), etc.
- multivariate extensions of GLMs: dirichlet, multinomial, negative multinomial, etc.