Generalized linear models

Regression Models

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Linear models

- Linear models are the most useful applied statistical technique. However, they are not without their limitations.
  - Additive response models don't make much sense if the response is discrete, or stricly positive.
  - Additive error models often don't make sense, for example if the outcome has to be positive.
  - Transformations are often hard to interpret.
  - There's value in modeling the data on the scale that it was collected.
  - Particularly interpetable transformations, natural logarithms in specific, aren't applicable for negative or zero values.

Generalized linear models

- Introduced in a 1972 RSSB paper by Nelder and Wedderburn.
- Involves three components
  - An *exponential family* model for the response.
  - A systematic component via a linear predictor.
  - A link function that connects the means of the response to the linear predictor.

Example, linear models

- Assume that $Y_i \sim N(\mu_i, \sigma^2)$ (the Gaussian distribution is an exponential family distribution.)
- Define the linear predictor to be $\eta_i = \sum_{k=1}^{p} X_{ik} \beta_k$.
- The link function as g so that $g(\mu) = \eta$.
  - For linear models $g(\mu) = \mu$ so that $\mu_i = \eta_i$
- This yields the same likelihood model as our additive error Gaussian linear model

$$Y_i = \sum_{k=1}^{p} X_{ik} \beta_k + \epsilon_i$$

where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$

Example, logistic regression

- Assume that $Y_i \sim \text{Bernoulli}(\mu_i)$ so that $E[Y_i] = \mu_i$ where $0 \leq \mu_i \leq 1$.
- Linear predictor $\eta_i = \sum_{k=1}^{p} X_{ik} \beta_k$
- Link function $g(\mu) = \eta = \log\left(\frac{\mu}{1-\mu}\right)$ g is the (natural) log odds, referred to as the **logit**.
- Note then we can invert the logit function as

$$\mu_i = \frac{\exp(\eta_i)}{1 + \exp(\eta_i)} \quad \text{and} \quad 1 - \mu_i = \frac{1}{1 + \exp(\eta_i)}$$

Thus the likelihood is

$$\prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i} = \exp\left(\sum_{i=1}^{n} y_i \eta_i\right) \prod_{i=1}^{n} (1 + \eta_i)^{-1}$$

Example, Poisson regression

- Assume that $Y_i \sim \text{Poisson}(\mu_i)$ so that $E[Y_i] = \mu_i$ where $0 \leq \mu_i$
- Linear predictor $\eta_i = \sum_{k=1}^{p} X_{ik} \beta_k$
- Link function $g(\mu) = \eta = \log(\mu)$
- Recall that e^x is the inverse of $\log(x)$ so that

$$\mu_i = e^{\eta_i}$$

Thus, the likelihood is

$$\prod_{i=1}^{n} (y_i!)^{-1} \mu_i^{y_i} e^{-\mu_i} \propto \exp\left(\sum_{i=1}^{n} y_i \eta_i - \sum_{i=1}^{n} \mu_i \right)$$

Some things to note

- In each case, the only way in which the likelihood depends on the data is through

$$\sum_{i=1}^{n} y_i \eta_i = \sum_{i=1}^{n} y_i \sum_{k=1}^{p} X_{ik} \beta_k = \sum_{k=1}^{p} \beta_k \sum_{i=1}^{n} X_{ik} y_i$$

  Thus if we don't need the full data, only $\sum_{i=1}^{n} X_{ik} y_i$. This simplification is a consequence of chosing so-called 'canonical' link functions.

- (This has to be derived). All models acheive their maximum at the root of the so called normal equations

$$0 = \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\mathrm{Var}(Y_i)} W_i$$

  where W_i are the derivative of the inverse of the link function.

About variances

$$0 = \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\mathrm{Var}(Y_i)} W_i$$

- For the linear model $\mathrm{Var}(Y_i) = \sigma^2$ is constant.
- For Bernoulli case $\mathrm{Var}(Y_i) = \mu_i(1 - \mu_i)$
- For the Poisson case $\mathrm{Var}(Y_i) = \mu_i$.
- In the latter cases, it is often relevant to have a more flexible variance model, even if it doesn't correspond to an actual likelihood

$$0 = \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\phi\mu_i(1 - \mu_i)} W_i \quad \text{and} \quad 0 = \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\phi\mu_i} W_i$$

- These are called 'quasi-likelihood' normal equations

Odds and ends

- The normal equations have to be solved iteratively. Resulting in $\hat{\beta}_k$ and, if included, $\hat{\phi}$.
- Predicted linear predictor responses can be obtained as $\hat{\eta} = \sum_{k=1}^{p} X_k \hat{\beta}_k$
- Predicted mean responses as $\hat{\mu} = g^{-1}(\hat{\eta})$
- Coefficients are interpretted as

$$g(E[Y|X_k = x_k + 1, X_{\sim k} = x_{\sim k}]) - g(E[Y|X_k = x_k, X_{\sim k} = x_{\sim k}]) = \beta_k$$

  or the change in the link function of the expected response per unit change in X_k holding other regressors constant.
- Variations on Newon/Raphson's algorithm are used to do it.
- Asymptotics are used for inference usually.
- Many of the ideas from linear models can be brought over to GLMs.

Generalized linear models, binary data

Regression models

Brian Caffo, Jeff Leek and Roger Peng
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Frequently we care about outcomes that have two values
  - Alive/dead
  - Win/loss
  - Success/Failure
  - etc
- Called binary, Bernoulli or 0/1 outcomes
- Collection of exchangeable binary outcomes for the same covariate data are called binomial outcomes.

Example Baltimore Ravens win/loss

Ravens Data

```
download.file("https://dl.dropboxusercontent.com/u/7710864/data/ravensData.rda"
              , destfile="./data/ravensData.rda",method="curl")
load("./data/ravensData.rda")
head(ravensData)
```

```
  ravenWinNum ravenWin ravenScore opponentScore
1           1        W         24             9
2           1        W         38            35
3           1        W         28            13
4           1        W         34            31
5           1        W         44            13
6           0        L         23            24
```

Linear regression

$$RW_i = b_0 + b_1 RS_i + e_i$$

RW_i - 1 if a Ravens win, 0 if not

RS_i - Number of points Ravens scored

b_0 - probability of a Ravens win if they score 0 points

b_1 - increase in probability of a Ravens win for each additional point

e_i - residual variation due

Linear regression in R

```
lmRavens <- lm(ravensData$ravenWinNum ~ ravensData$ravenScore)
summary(lmRavens)$coef
```

```
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           0.2850   0.256643   1.111  0.28135
ravensData$ravenScore 0.0159   0.009059   1.755  0.09625
```

Odds

Binary Outcome 0/1

$$RW_i$$

Probability (0,1)

$$\Pr(RW_i|RS_i, b_0, b_1)$$

Odds $(0, \infty)$

$$\frac{\Pr(RW_i|RS_i, b_0, b_1)}{1 - \Pr(RW_i|RS_i, b_0, b_1)}$$

Log odds $(-\infty, \infty)$

$$\log\left(\frac{\Pr(RW_i|RS_i, b_0, b_1)}{1 - \Pr(RW_i|RS_i, b_0, b_1)}\right)$$

Linear vs. logistic regression

Linear

$$RW_i = b_0 + b_1 RS_i + e_i$$

or

$$E[RW_i | RS_i, b_0, b_1] = b_0 + b_1 RS_i$$

Logistic

$$Pr(RW_i | RS_i, b_0, b_1) = \frac{\exp(b_0 + b_1 RS_i)}{1 + \exp(b_0 + b_1 RS_i)}$$

or

$$\log\left(\frac{Pr(RW_i | RS_i, b_0, b_1)}{1 - Pr(RW_i | RS_i, b_0, b_1)} \right) = b_0 + b_1 RS_i$$

Interpreting Logistic Regression

$$\log\left(\frac{\Pr(RW_i|RS_i, b_0, b_1)}{1-\Pr(RW_i|RS_i, b_0, b_1)}\right) = b_0 + b_1 RS_i$$

b_0 - Log odds of a Ravens win if they score zero points

b_1 - Log odds ratio of win probability for each point scored (compared to zero points)

$\exp(b_1)$ - Odds ratio of win probability for each point scored (compared to zero points)

Odds

- Imagine that you are playing a game where you flip a coin with success probability p.
- If it comes up heads, you win X. If it comes up tails, you lose Y.
- What should we set X and Y for the game to be fair?

$$E[\text{earnings}] = Xp - Y(1-p) = 0$$

- Implies

$$\frac{Y}{X} = \frac{p}{1-p}$$

- The odds can be said as "How much should you be willing to pay for a p probability of winning a dollar?"
  - (If $p > 0.5$ you have to pay more if you lose than you get if you win.)
  - (If $p < 0.5$ you have to pay less if you lose than you get if you win.)

Visualizing fitting logistic regression curves

```
x <- seq(-10, 10, length = 1000)
manipulate(
    plot(x, exp(beta0 + beta1 * x) / (1 + exp(beta0 + beta1 * x)),
         type = "l", lwd = 3, frame = FALSE),
    beta1 = slider(-2, 2, step = .1, initial = 2),
    beta0 = slider(-2, 2, step = .1, initial = 0)
    )
```

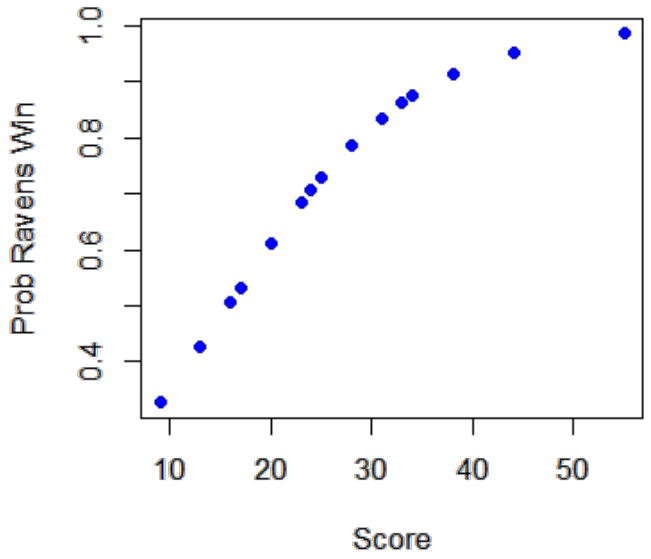
Ravens logistic regression

```
logRegRavens <- glm(ravensData$ravenWinNum ~ ravensData$ravenScore,family="binomial")
summary(logRegRavens)
```

```

Call:
glm(formula = ravensData$ravenWinNum ~ ravensData$ravenScore,
    family = "binomial")

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.758  -1.100   0.530   0.806   1.495

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           -1.6800     1.5541   -1.08     0.28
ravensData$ravenScore  0.1066     0.0667    1.60     0.11

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 24.435  on 19  degrees of freedom
Residual deviance: 20.895  on 18  degrees of freedom
AIC: 24.89
```

Ravens fitted values

```
plot(ravensData$ravenScore,logRegRavens$fitted,pch=19,col="blue",xlab="Score",ylab="Prob Ravens Win")
```



Odds ratios and confidence intervals

```
exp(logRegRavens$coeff)
```

```
          (Intercept) ravensData$ravenScore
               0.1864                1.1125
```

```
exp(confint(logRegRavens))
```

```
                        2.5 % 97.5 %
(Intercept)           0.005675  3.106
ravensData$ravenScore 0.996230  1.303
```

ANOVA for logistic regression

```
anova(logRegRavens,test="Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: ravensData$ravenWinNum

Terms added sequentially (first to last)

                     Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                                    19       24.4
ravensData$ravenScore  1     3.54        18       20.9     0.06 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpreting Odds Ratios

- Not probabilities
- Odds ratio of 1 = no difference in odds
- Log odds ratio of 0 = no difference in odds
- Odds ratio < 0.5 or > 2 commonly a "moderate effect"
- Relative risk $\frac{\Pr(RW_i|RS_i=10)}{\Pr(RW_i|RS_i=0)}$ often easier to interpret, harder to estimate
- For small probabilities RR ≈ OR but **they are not the same**!

Wikipedia on Odds Ratio

Further resources

- Wikipedia on Logistic Regression
- Logistic regression and glms in R
- Brian Caffo's lecture notes on: Simpson's paradox, Case-control studies
- Open Intro Chapter on Logistic Regression

Count outcomes, Poisson GLMs

Regression Models

Brian Caffo, Jeffrey Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Many data take the form of counts
  - Calls to a call center
  - Number of flu cases in an area
  - Number of cars that cross a bridge
- Data may also be in the form of rates
  - Percent of children passing a test
  - Percent of hits to a website from a country
- Linear regression with transformation is an option

Poisson distribution

- The Poisson distribution is a useful model for counts and rates
- Here a rate is count per some monitoring time
- Some examples uses of the Poisson distribution
  - Modeling web traffic hits
  - Incidence rates
  - Approximating binomial probabilities with small p and large n
  - Analyzing contigency table data

The Poisson mass function

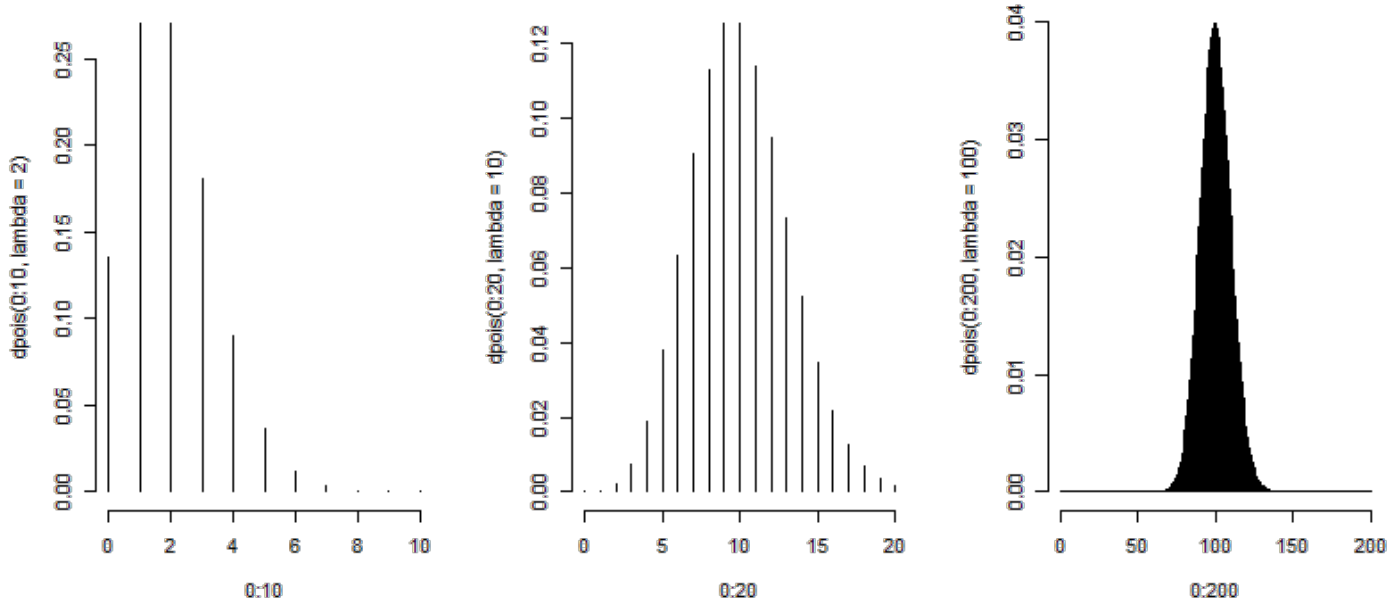
- $X \sim \text{Poisson}(t\lambda)$ if

$$P(X = x) = \frac{(t\lambda)^x e^{-t\lambda}}{x!}$$

  For $x = 0, 1, \ldots$.

- The mean of the Poisson is $E[X] = t\lambda$, thus $E[X/t] = \lambda$
- The variance of the Poisson is $\text{Var}(X) = t\lambda$.
- The Poisson tends to a normal as $t\lambda$ gets large.

```r
par(mfrow = c(1, 3))
plot(0 : 10, dpois(0 : 10, lambda = 2), type = "h", frame = FALSE)
plot(0 : 20, dpois(0 : 20, lambda = 10), type = "h", frame = FALSE)
plot(0 : 200, dpois(0 : 200, lambda = 100), type = "h", frame = FALSE)
```



Poisson distribution

Sort of, showing that the mean and variance are equal

```
x <- 0 : 10000; lambda = 3
mu <- sum(x * dpois(x, lambda = lambda))
sigmasq <- sum((x - mu)^2 * dpois(x, lambda = lambda))
c(mu, sigmasq)
```

```
[1] 3 3
```

Example: Leek Group Website Traffic

- Consider the daily counts to Jeff Leek's web site

http://biostat.jhsph.edu/~jleek/

- Since the unit of time is always one day, set $t = 1$ and then the Poisson mean is interpretted as web hits per day. (If we set $t = 24$, it would be web hits per hour).

Website data

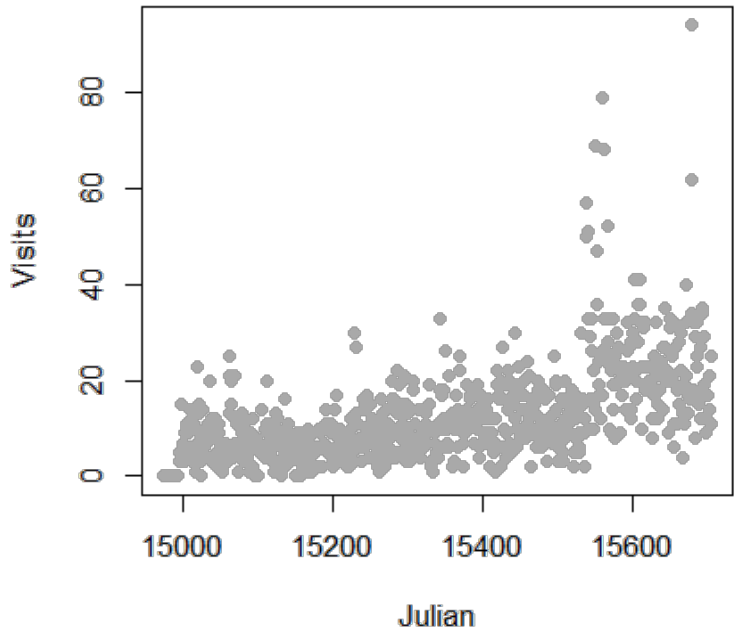
```
download.file("https://dl.dropboxusercontent.com/u/7710864/data/gaData.rda",destfile="./data/gaData.rda
load("./data/gaData.rda")
gaData$julian <- julian(gaData$date)
head(gaData)
```

```
        date visits simplystats julian
1 2011-01-01      0           0  14975
2 2011-01-02      0           0  14976
3 2011-01-03      0           0  14977
4 2011-01-04      0           0  14978
5 2011-01-05      0           0  14979
6 2011-01-06      0           0  14980
```

http://skardhamar.github.com/rga/

Plot data

```
plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")
```



Linear regression

$$NH_i = b_0 + b_1 JD_i + e_i$$

NH_i - number of hits to the website

JD_i - day of the year (Julian day)

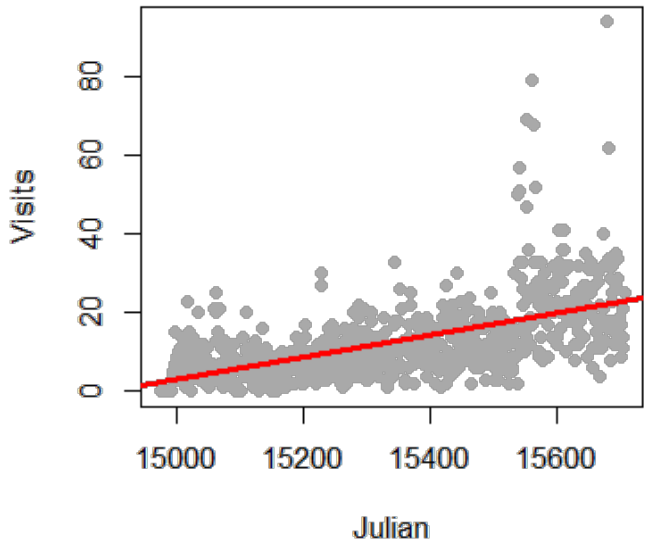
b_0 - number of hits on Julian day 0 (1970-01-01)

b_1 - increase in number of hits per unit day

e_i - variation due to everything we didn't measure

Linear regression line

```
plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")
lm1 <- lm(gaData$visits ~ gaData$julian)
abline(lm1,col="red",lwd=3)
```



Aside, taking the log of the outcome

- Taking the natural log of the outcome has a specific interpretation.
- Consider the model

$$\log(NH_i) = b_0 + b_1 JD_i + e_i$$

NH_i - number of hits to the website

JD_i - day of the year (Julian day)

b_0 - log number of hits on Julian day 0 (1970-01-01)

b_1 - increase in log number of hits per unit day

e_i - variation due to everything we didn't measure

Exponentiating coefficients

- $e^{E[\log(Y)]}$ geometric mean of Y.
  - With no covariates, this is estimated by $e^{\frac{1}{n}\sum_{i=1}^{n}\log(y_i)} = (\prod_{i=1}^{n} y_i)^{1/n}$
- When you take the natural log of outcomes and fit a regression model, your exponentiated coefficients estimate things about geometric means.
- e^{β_0} estimated geometric mean hits on day 0
- e^{β_1} estimated relative increase or decrease in geometric mean hits per day
- There's a problem with logs with you have zero counts, adding a constant works

```
round(exp(coef(lm(I(log(gaData$visits + 1)) ~ gaData$julian))), 5)
```

```
  (Intercept) gaData$julian
        0.000         1.002
```

Linear vs. Poisson regression

Linear

$$NH_i = b_0 + b_1 JD_i + e_i$$

or

$$E[NH_i | JD_i, b_0, b_1] = b_0 + b_1 JD_i$$

Poisson/log-linear

$$\log\left(E[NH_i | JD_i, b_0, b_1]\right) = b_0 + b_1 JD_i$$

or

$$E[NH_i | JD_i, b_0, b_1] = \exp\left(b_0 + b_1 JD_i\right)$$

Multiplicative differences

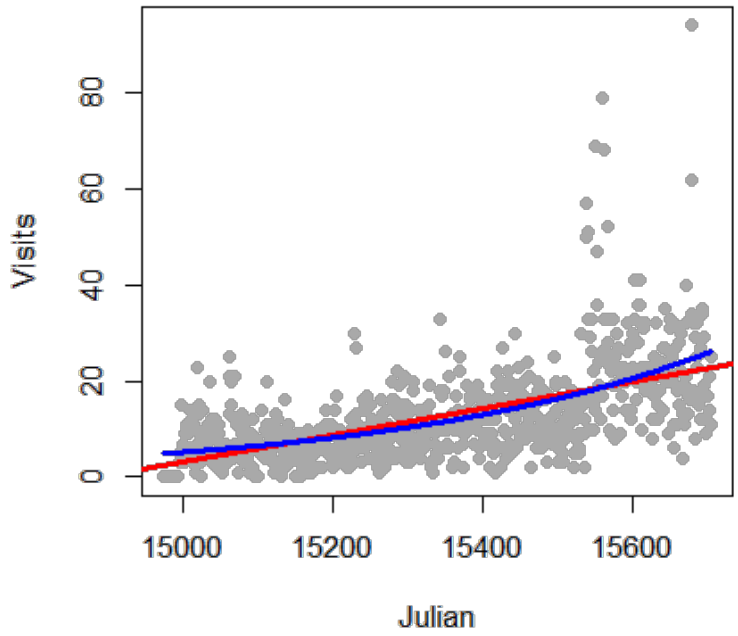
$$E[NH_i|JD_i, b_0, b_1] = \exp\left(b_0 + b_1 JD_i\right)$$

$$E[NH_i|JD_i, b_0, b_1] = \exp\left(b_0\right) \exp\left(b_1 JD_i\right)$$

If JD_i is increased by one unit, $E[NH_i|JD_i, b_0, b_1]$ is multiplied by $\exp\left(b_1\right)$

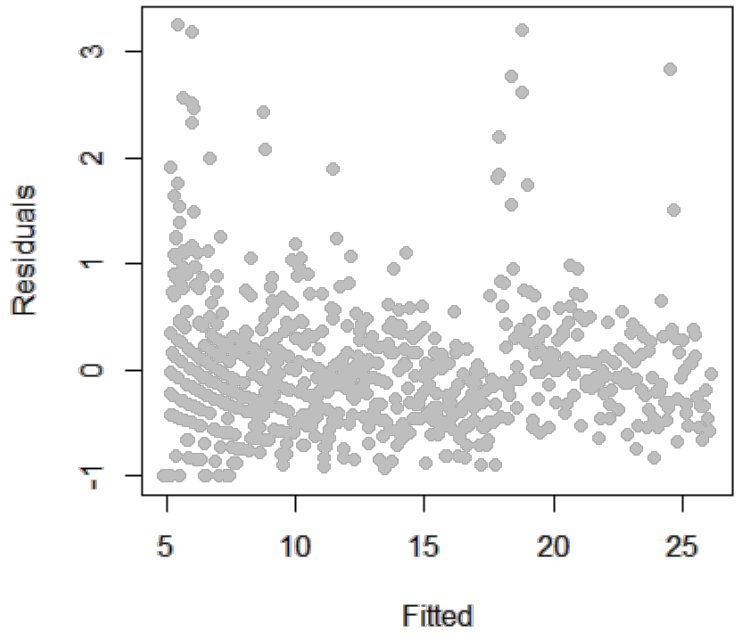
Poisson regression in R

```
plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")
glm1 <- glm(gaData$visits ~ gaData$julian,family="poisson")
abline(lm1,col="red",lwd=3); lines(gaData$julian,glm1$fitted,col="blue",lwd=3)
```



Mean-variance relationship?

```
plot(glm1$fitted,glm1$residuals,pch=19,col="grey",ylab="Residuals",xlab="Fitted")
```



Model agnostic standard errors

```
library(sandwich)
confint.agnostic <- function (object, parm, level = 0.95, ...)
{
    cf <- coef(object); pnames <- names(cf)
    if (missing(parm))
        parm <- pnames
    else if (is.numeric(parm))
        parm <- pnames[parm]
    a <- (1 - level)/2; a <- c(a, 1 - a)
    pct <- stats:::format.perc(a, 3)
    fac <- qnorm(a)
    ci <- array(NA, dim = c(length(parm), 2L), dimnames = list(parm,
                                                               pct))
    ses <- sqrt(diag(sandwich::vcovHC(object)))[parm]
    ci[] <- cf[parm] + ses %o% fac
    ci
}
```

http://stackoverflow.com/questions/3817182/vcovhc-and-confidence-interval

Estimating confidence intervals

```
confint(glm1)
```

```
                  2.5 %     97.5 %
(Intercept)   -34.34658 -31.159716
gaData$julian   0.00219   0.002396
```

```
confint.agnostic(glm1)
```

```
                  2.5 %     97.5 %
(Intercept)   -36.362675 -29.136997
gaData$julian   0.002058   0.002528
```

Rates

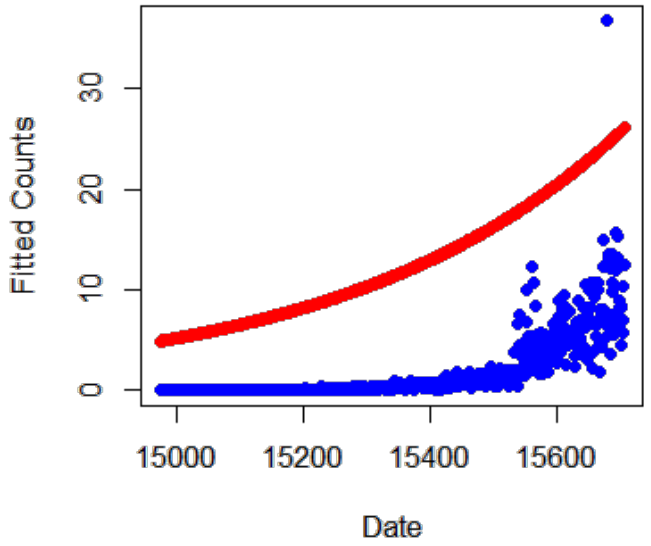
$$E[NHSS_i|JD_i, b_0, b_1]/NH_i = \exp\left(b_0 + b_1 JD_i\right)$$

$$\log\left(E[NHSS_i|JD_i, b_0, b_1]\right) - \log(NH_i) = b_0 + b_1 JD_i$$

$$\log\left(E[NHSS_i|JD_i, b_0, b_1]\right) = \log(NH_i) + b_0 + b_1 JD_i$$

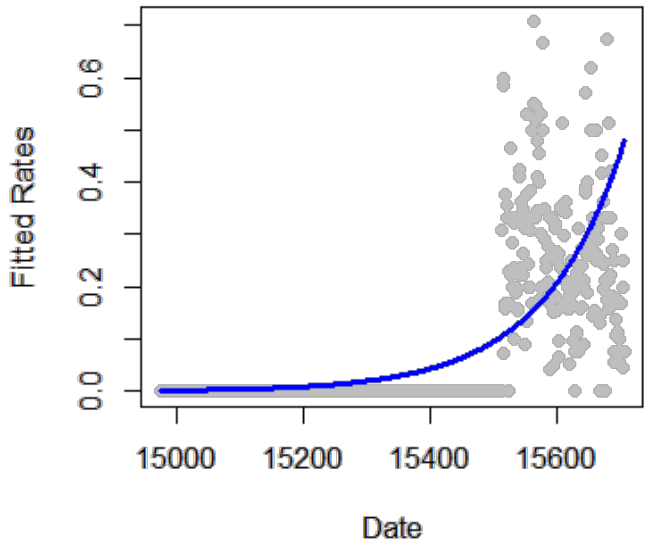
Fitting rates in R

```
glm2 <- glm(gaData$simplystats ~ julian(gaData$date),offset=log(visits+1),
            family="poisson",data=gaData)
plot(julian(gaData$date),glm2$fitted,col="blue",pch=19,xlab="Date",ylab="Fitted Counts")
points(julian(gaData$date),glm1$fitted,col="red",pch=19)
```



Fitting rates in R

```
glm2 <- glm(gaData$simplystats ~ julian(gaData$date),offset=log(visits+1),
            family="poisson",data=gaData)
plot(julian(gaData$date),gaData$simplystats/(gaData$visits+1),col="grey",xlab="Date",
     ylab="Fitted Rates",pch=19)
lines(julian(gaData$date),glm2$fitted/(gaData$visits+1),col="blue",lwd=3)
```



More information

- Log-linear models and multiway tables
- Wikipedia on Poisson regression, Wikipedia on overdispersion
- Regression models for count data in R
- pscl package - the function *zeroinfl* fits zero inflated models.

Hodgepodge

Regression models

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

How to fit functions using linear models

- Consider a model $Y_i = f(X_i) + \epsilon$.
- How can we fit such a model using linear models (called scatterplot smoothing)
- Consider the model

$$Y_i = \beta_0 + \beta_1 X_i + \sum_{k=1}^{d} (x_i - \xi_k)_+ \gamma_k + \epsilon_i$$

where $(a)_+ = a$ if $a > 0$ and 0 otherwise and $\xi_1 \leq \ldots \leq \xi_d$ are known knot points.

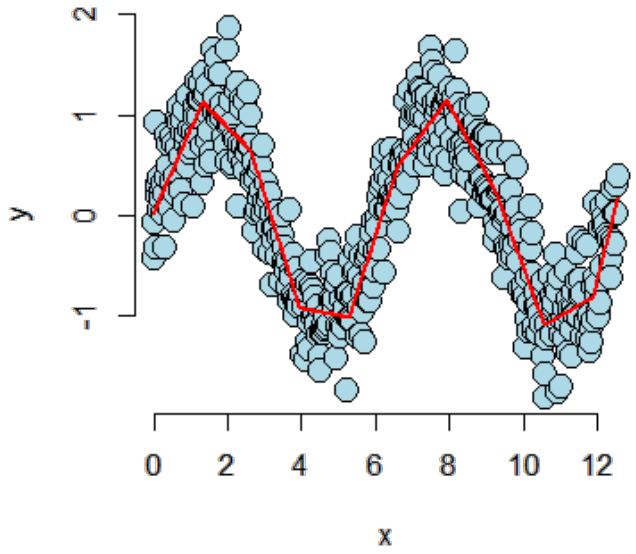
- Prove to yourelf that the mean function

$$\beta_0 + \beta_1 X_i + \sum_{k=1}^{d} (x_i - \xi_k)_+ \gamma_k$$

is continuous at the knot points.

Simulated example

```
n <- 500; x <- seq(0, 4 * pi, length = n); y <- sin(x) + rnorm(n, sd = .3)
knots <- seq(0, 8 * pi, length = 20);
splineTerms <- sapply(knots, function(knot) (x > knot) * (x - knot))
xMat <- cbind(1, x, splineTerms)
yhat <- predict(lm(y ~ xMat - 1))
plot(x, y, frame = FALSE, pch = 21, bg = "lightblue", cex = 2)
lines(x, yhat, col = "red", lwd = 2)
```

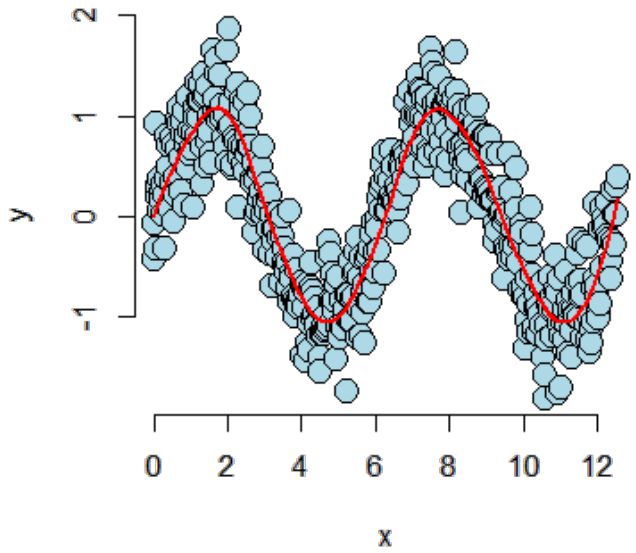


Adding squared terms

- Adding squared terms makes it continuously differentiable at the knot points.
- Adding cubic terms makes it twice continuously differentiable at the knot points; etcetera.

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \sum_{k=1}^{d} (x_i - \xi_k)_+^2 \gamma_k + \epsilon_i$$

```r
splineTerms <- sapply(knots, function(knot) (x > knot) * (x - knot)^2)
xMat <- cbind(1, x, x^2, splineTerms)
yhat <- predict(lm(y ~ xMat - 1))
plot(x, y, frame = FALSE, pch = 21, bg = "lightblue", cex = 2)
lines(x, yhat, col = "red", lwd = 2)
```

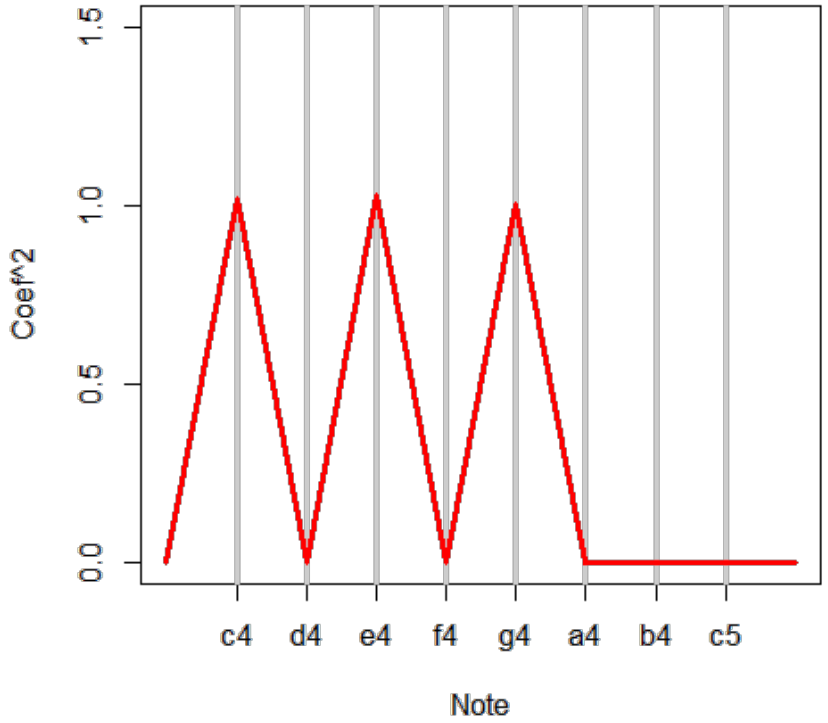


Notes

- The collection of regressors is called a basis.
  - People have spent **a lot** of time thinking about bases for this kind of problem. So, consider this as just a teaser.
- Single knot point terms can fit hockey stick like processes.
- These bases can be used in GLMs as well.
- An issue with these approaches is the large number of parameters introduced.
  - Requires some method of so called regularization.

Harmonics using linear models

```
##Chord finder, playing the white keys on a piano from octave c4 - c5
notes4 <- c(261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88, 523.25)
t <- seq(0, 2, by = .001); n <- length(t)
c4 <- sin(2 * pi * notes4[1] * t); e4 <- sin(2 * pi * notes4[3] * t);
g4 <- sin(2 * pi * notes4[5] * t)
chord <- c4 + e4 + g4 + rnorm(n, 0, 0.3)
x <- sapply(notes4, function(freq) sin(2 * pi * freq * t))
fit <- lm(chord ~ x - 1)
```


Coef^2
0.0
0.5
1.0
1.5
c4
d4
e4
f4
g4
a4
b4
c5
Note

```
##(How you would really do it)
a <- fft(chord); plot(Re(a)^2, type = "l")
```