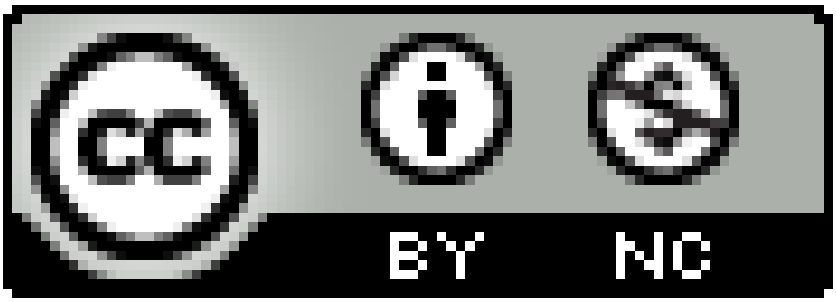
basic generalized linear models

Ben Bolker




```
library(ggplot2)
theme_set(theme_bw())
library(ggExtra)
library(cowplot)
library(dotwhisker)
```

Linear models

- foundation for (G)LM(M)s, other complex models
- flexible, robust, computationally efficient, standard
- includes (multiple) regression, ANOVA, ANCOVA, ...
- natural ways to express dependence, interactions

Linear models: assumptions

- response variables:
  - Gaussian (normally distributed)
  - independent
  - *conditionally* homoscedastic (equal variance)
  - univariate
- predictor variables
  - numeric or categorical (nominal)

Linear models: math

$$z = a + bx + cy + \epsilon$$

or (more predictor variables)

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \epsilon$$

or (more flexible distribution syntax)

$$y \sim \text{Normal}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots, \sigma^2)$$

or (more complex sets of predictors)

$$\boldsymbol{\mu} = \mathbf{Xfi}$$
$$y_i \sim \text{Normal}(\mu_i, \sigma^2)$$

what does "linear" mean?

- y is a linear function of the *parameters* ($\partial^2 y / \partial^2 \beta_i = 0$)
- e.g. polynomials: $y = a + bx + cx^2 + dx^3$
- or sinusoids: $y = a \sin(x) + b \cos(x)$
- but **not**: power-law (ax^b), exponential ($a \exp(-bx)$)

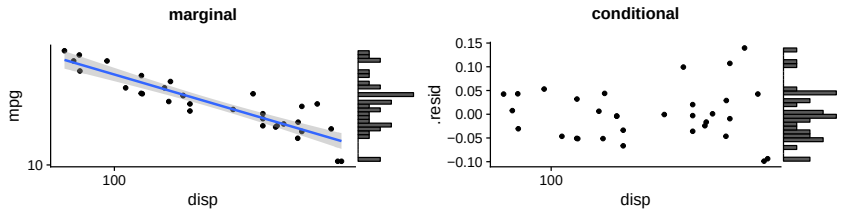
marginal vs. conditional distributions

- common mistake: worry about the overall distribution of the response,
  rather than the *conditional* distribution (i.e., residuals)
- if only categorical predictors, can mean-correct each group, then look at residuals
- otherwise have to fit the model first!

example

MPG vs displacement for cars

```
cars_lm <- lm(log10(mpg) ~ log10(disp), mtcars)
```



(We'll come back to how to judge this later)

categorical predictors

- how do categorical predictors fit into this scheme?
- *dummy variables*: convert to 0/1 values
- R does this automatically with formula syntax
- e.g. for two levels:

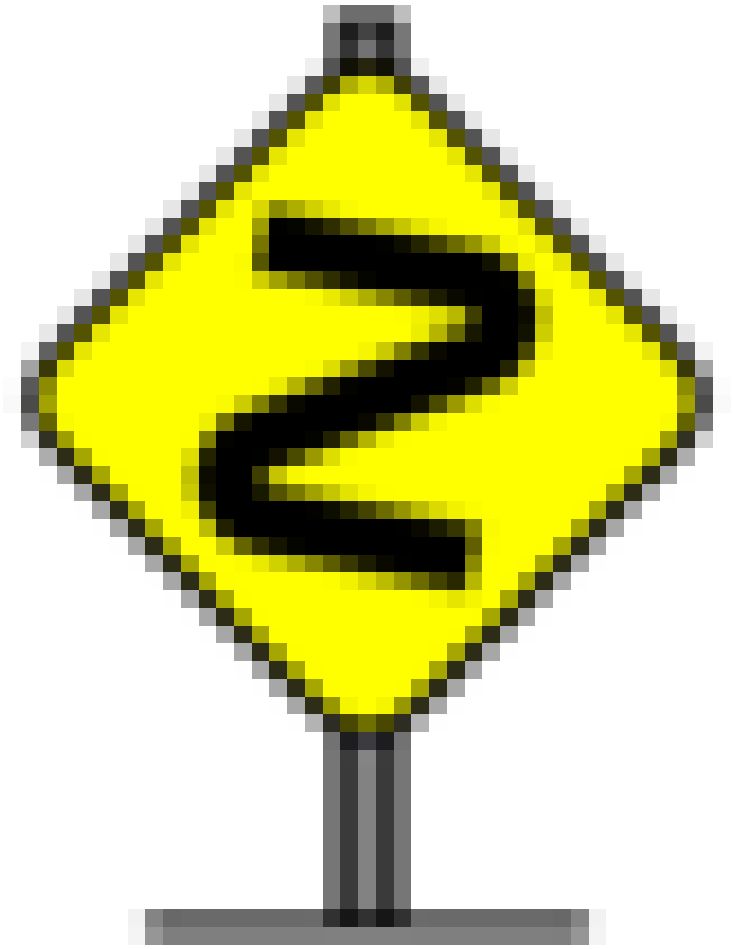
```
dd <- data.frame(flavour = rep(c("chocolate",
    "vanilla"), c(2, 3)))
print(dd)
```

```
##     flavour
## 1 chocolate
## 2 chocolate
## 3   vanilla
## 4   vanilla
## 5   vanilla
```

```
model.matrix(~flavour, dd)
```

```
##   (Intercept) flavourvanilla
## 1           1              0
## 2           1              0
## 3           1              1
## 4           1              1
## 5           1              1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$flavour
## [1] "contr.treatment"
```

- first alphabetical level (`chocolate`) used as default (use `relevel()` or `factor(...,levels=...)` to change default)



- *ordered factors* are handled differently

R formulas

- Wilkinson and Rogers (1973)
- `response ~ predictor1 + predictor2 + ...`
- numeric variables used "as is"
- categorical variables (factors) converted to dummy variables
- intercept added automatically (`1+ ...`)
- interaction: `:` *multiplies* relevant columns
- `a*b`: main effect plus interactions
- `model.matrix(formula, data)`

Formulas, continued

- `y~f`: 1-way ANOVA

- `y~f+g`: 2-way ANOVA (additive)
- `y~f*g`: 2-way ANOVA (with interaction)
- `y~x`: univariate regression
- `y~f+x`: ANCOVA (parallel slopes)
- `y~f*x`: ANCOVA (with interaction, non-parallel slopes)
- `y~x1+x2`: multivariate regression (additive)
- `y~x1*x2`: multiv. regression with interaction

If confused, (1) try to write out the equation; (2) `model.matrix()`

Contrasts

- Machinery for translating categorical variables to dummy (0/1) variables
- **treatment** contrasts (default):
  - β_1 = intercept = expected value of first level (by default, "aardvark")
  - β_i = difference between level $i+1$ and baseline
- **sum-to-zero** contrasts:
  - β_1 = intercept = unweighted mean of all levels
  - β_i = difference between level i and mean; last level not included (!)

too many ways to change contrasts (globally via `options()`; as attribute of factor; `contrasts` argument in `lm()`)

Example 1 (treatment contrasts)

Data on ant colonies from Gotelli and Ellison (2004):

```
ants <- data.frame(place = rep(c("field", "forest"),
    c(6, 4)), colonies = c(12, 9, 12, 10, 9, 6,
    4, 6, 7, 10))
aggregate(colonies ~ place, data = ants, FUN = mean)
```

```
##    place colonies
## 1  field 9.666667
## 2 forest 6.750000
```

```
pr <- function(m) printCoefmat(coef(summary(m)),
    digits = 3, signif.stars = FALSE)
pr(lm1 <- lm(colonies ~ place, data = ants))
```

```
##             Estimate Std. Error t value
## (Intercept)    9.667      0.958   10.09
```

```
## placeforest   -2.917      1.515   -1.92
##             Pr(>|t|)
## (Intercept)    8e-06
## placeforest     0.09
```

Ants: sum-to-zero contrasts

```
pr(lm2 <- update(lm1, contrasts = list(place = contr.sum)))
```

```
##             Estimate Std. Error t value
## (Intercept)    8.208      0.758   10.83
## place1         1.458      0.758    1.92
##             Pr(>|t|)
## (Intercept)  4.7e-06
## place1          0.09
```

```
data(lizards, package = "brglm")
```

Interactions: example

- Bear road-crossing
- Predictor variables: sex (categorical: M/F), road type (categorical: major/minor), road length (continuous)
- **Two-way interactions**
  - sex × road length: "are females more sensitive to amount of road than males?"
  - sex × road type: "do females prefer major over minor roads more than males?"
  - road type × road length: "does amount of road affect crossings differently for different road types?"
- **Three-way interaction**: does the difference of the effect of road length between road types differ between sexes?

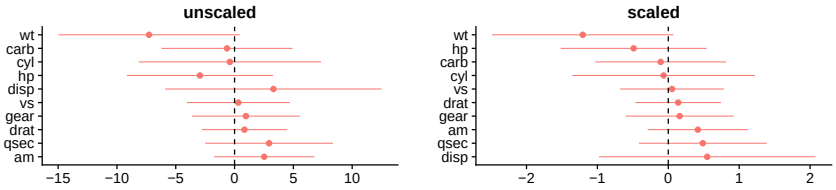
Centering (Schielzeth 2010)

- in interaction models, interpretation of main effects **depends on the center-point of the predictors**
- *centering* makes main effects much more interpretable
  - numeric predictors (subtracting the mean by default; other choices could be sensible)
  - categorical predictors: sum-to-zero (weighted or unweighted)
- e.g. if Gregorian year is a predictor, the intercept is at year 0 (!)
- also improves model stability, decorrelates coefficients

Scaling (Schielzeth 2010)

- scaling parameters improves interpretability
- standard deviation scaling:
  parameter magnitudes = importance

```
mtcars_big <- lm(mpg ~ ., data = mtcars)
mtcars_big_sc <- lm(mpg ~ ., data = as.data.frame(scale(mtcars)))
dwfun <- function(., title) {
    dwplot(., order_vars = names(sort(coef(.)))) +
        geom_vline(xintercept = 0, linetype = 2) +
        ggtitle(title)
}
plot_grid(dwfun(mtcars_big, "unscaled"), dwfun(mtcars_big_sc,
    "scaled"))
```

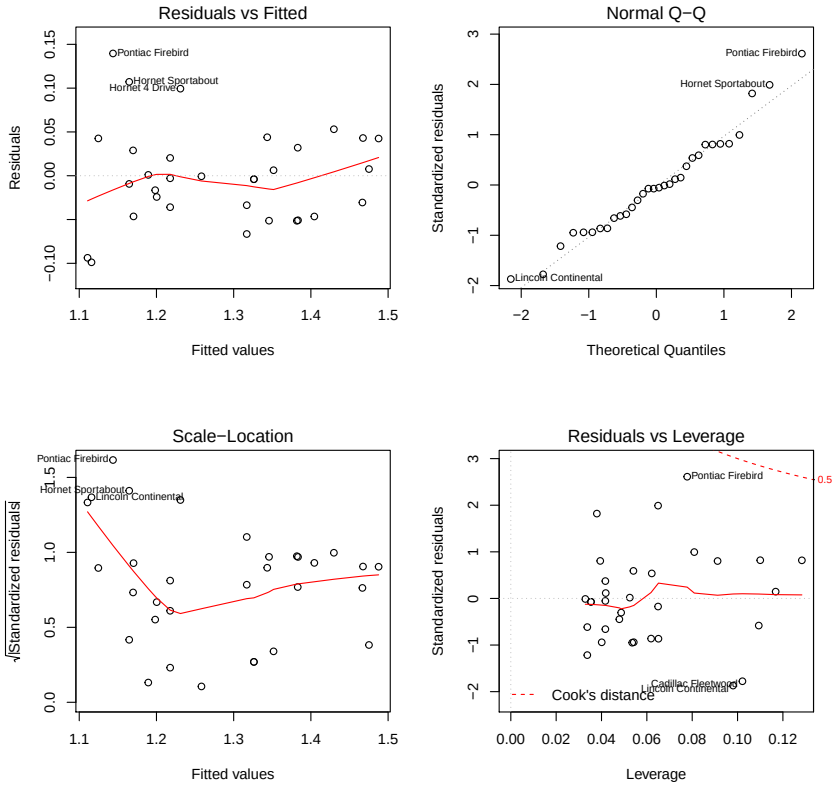


LM diagnostics

- fitted vs. residual: pattern in mean? (linearity)
- scale-location: pattern in variance? (homoscedasticity)
- Q-Q plot: Normality of **residuals**
- leverage/Cook's distance: influential points?
- independence is often hard to test
- Normality is the **least important** of these assumptions

LM diagnostics

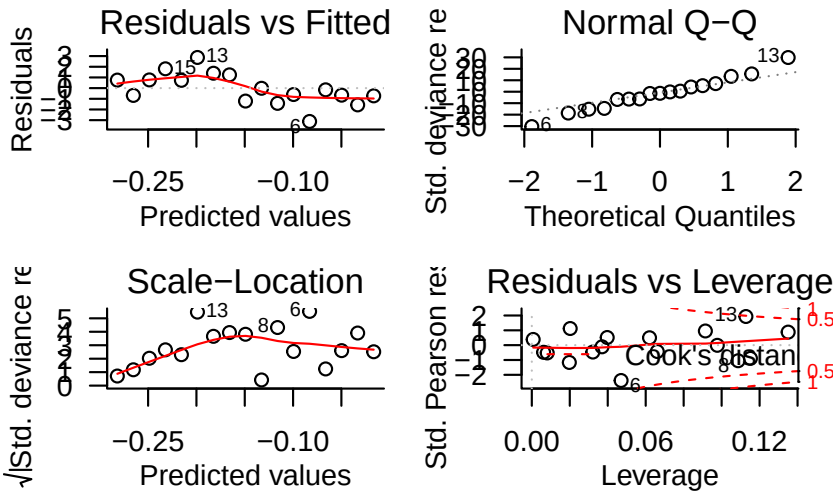
```
par(mfrow = c(2, 2))
plot(cars_lm)
```



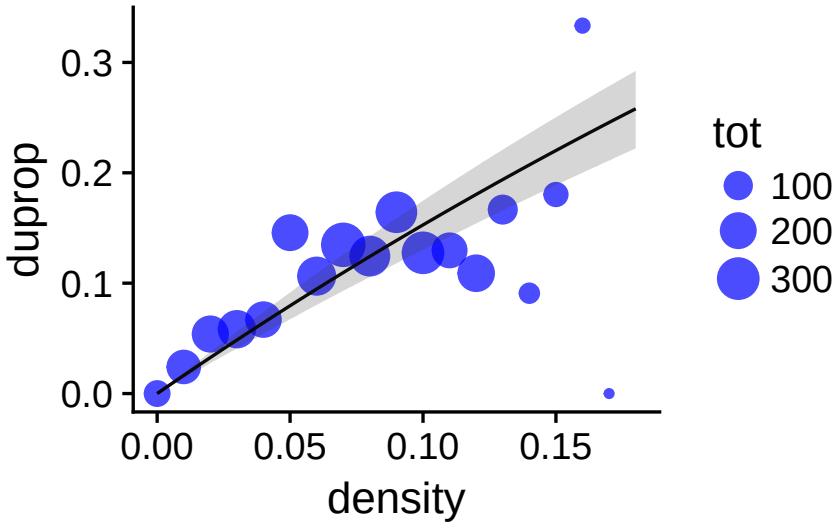
- problems are not independent
- deal with problems in order (location > scale > outliers > distribution)
- smooth lines help interpretation
- highlighted points are 3 most extreme (`id.n` argument)

a bad model (Tiwari et al. 2006)

- this is based on a GLM, but the ideas are the same



original data/fit . . .



Diagnostics

- statisticians: "don't use p-values to evaluate LM assumptions"
- everyone else: "so what should I do?"
- statisticians: "look at pictures"
- everyone else: "how do I decide whether to worry?"
- statisticians: ". . . "

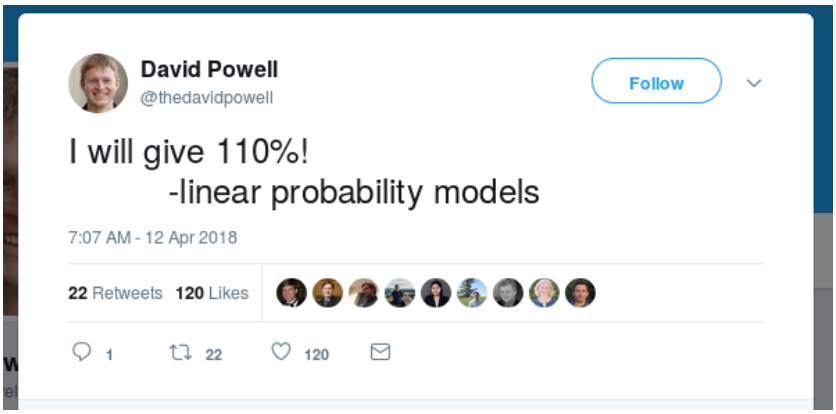
testing hypotheses and interpreting results

- parameter-by-parameter: `summary()` (t test)
- multi-parameter comparisons: `anova()`, `car::Anova()` (F test)
- order matters
- interactions/main effects matter

From LM to GLM

Why GLMs?

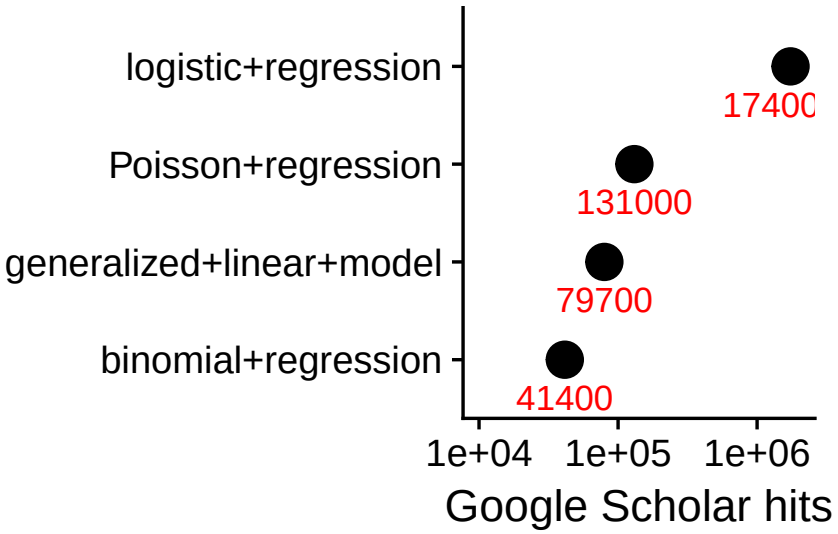
- assumptions of LMs do break down sometimes
- count data: discrete, non-negative
- proportion data: discrete counts, $0 \leq x \leq N$
- hard to transform to Normal
- linear model doesn't make sense



GLMs in action

- vast majority of GLMs
  - *logistic regression* (binary/Bernoulli data)
  - *Poisson regression* (count data)
- lots of GLM theory carries over from LMs
  - formulas
  - parameter interpretation (partly)
  - diagnostics (partly)

Most GLMs are logistic



Family

- family: what kind of data do I have?

  - from **first principles**: family specifies the relationship between the mean and variance
  - binomial: proportions, out of a total number of counts; includes binary (Bernoulli) ("logistic regression")
  - Poisson (independent counts, no set maximum, or far from the maximum)
  - other (Normal (`"gaussian"`), Gamma)
- default family for `glm` is Gaussian

link functions

- transform *prediction*, not response
- e.g. rather than $\log(\mu) = \beta_0 + \beta_1 x$, use $\mu = \exp(\beta_0 + \beta_1 x)$
- in this case log is the **link function**, exp is the **inverse link** function
- extreme observations don't cause problems (usually)

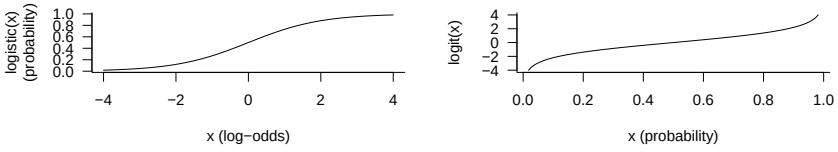
family definitions

- link function plus variance function
- typical defaults
  - Poisson: log (exponential)
  - binomial: logit/log-odds (logistic)

log link

- proportional scaling of effects
- small values of coefficients (< 0.1) $\approx$ proportionality
- otherwise change per unit is $\exp(\beta)$
- large parameter values (> 10) mean some kind of trouble

logit link/logistic function



- `qlogis()` function (`plogis()` is logistic/inverse-link)
- *log-odds* ($\log(p/(1-p))$)
- most natural scale for probability calculations
- interpretation depends on *base probability*
  - small probability: like log (proportional)
  - large probability: like log(1-p)
  - intermediate ($0.3 < p < 0.7$): effect $\approx \beta/4$

binomial models

- for Poisson, Bernoulli responses we only need one piece of information
- how do we specify denominator (N in k/N)?
- traditional R: response is two-column matrix `cbind(k,N-k)` **not** `cbind(k,N)`
- also allowed: response is proportion (k/N), also specify `weights=N`
- if equal for all cases and specified on the fly need to replicate: `glm(p~...,data,weights=rep(N,nrow(data)))`

diagnostics

- a little harder than linear models: `plot` is still somewhat useful
- binary data especially hard (e.g. `arm::binnedplot`)
- goodness of fit tests, R^2 etc. hard (can always compute `cor(observed,predict(model, type="response"))`)
- residuals are *Pearson residuals* by default ($(\text{obs} - \text{exp})/V(\text{exp})$); predicted values are on the effect scale (e.g. log/logit) by default (use `type="response"` to get data-scale predictions)
- also see `DHARMa` package

overdispersion

- too much variance
- more detail later
- should have residual df $\approx$ residual deviance

back-transformation

- confidence intervals are symmetric on link scale
- can back-transform estimates and CIs for log
- logit is hard (must pick a reference level)
- don't back-transform standard errors!

estimation

- iteratively re-weighted least-squares
- usually Just Works

inference

like LMs, but:

- one-parameter tests are usually Z rather than t
- CIs based on standard errors are approximate (Wald)
- `confint.glm()` computes *likelihood profile* CIs

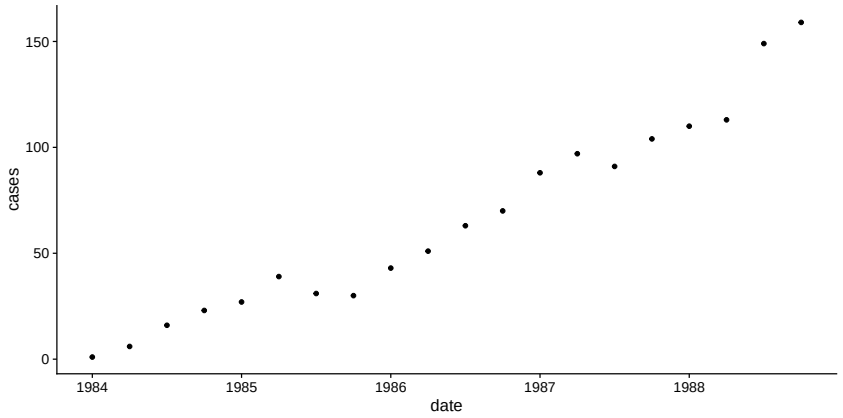
Common(est?) `glm()` problems

- binomial/Poisson models with non-integer data
- failing to specify `family` ($\rightarrow$ linear model); using `glm()` for linear models (unnecessary)
- predictions on effect scale
- using (k, N) rather than $(k, N-k)$ in binomial models
- back-transforming SEs rather than CIs
- neglecting overdispersion
- Poisson for *underdispersed* responses
- equating negative binomial with binomial rather than Poisson (
- worrying about overdispersion unnecessarily (binary/Gamma)
- ignoring random effects

Example

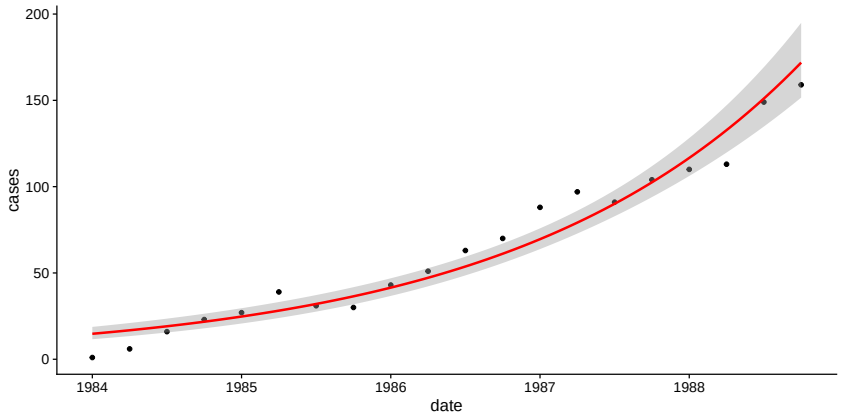
AIDS (Australia: Dobson & Barnett)

```
aids <- read.csv("../data/aids.csv")
aids <- transform(aids, date = year + (quarter -
    1)/4)
print(gg0 <- ggplot(aids, aes(date, cases)) +
    geom_point())
```



Easy GLMs with ggplot

```
print(gg1 <- gg0 + geom_smooth(method = "glm",
    colour = "red", method.args = list(family = "quasipoisson")))
```

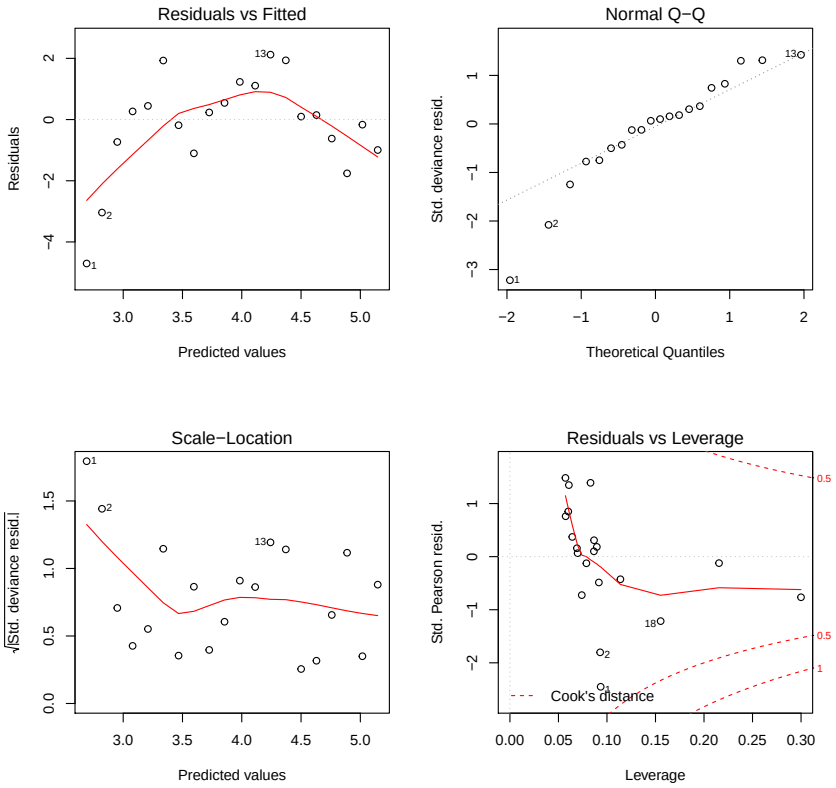


Equivalent code

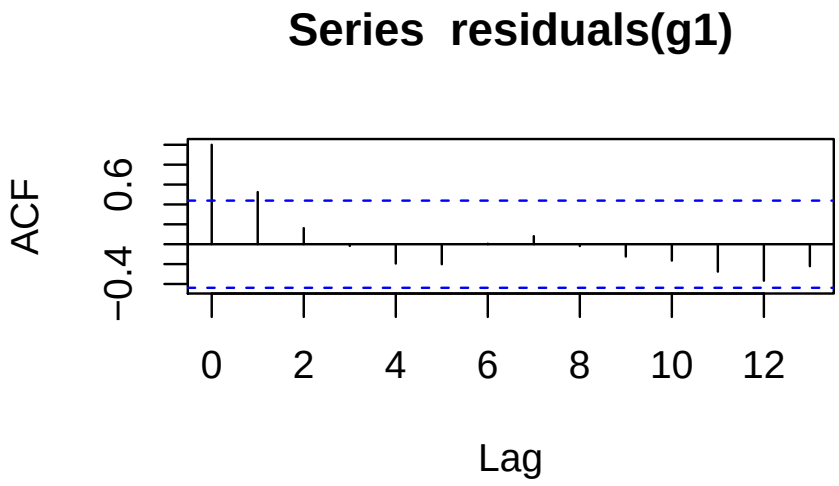
```
g1 <- glm(cases ~ date, aids, family = quasipoisson(link = "log"))
summary(g1)
```

```
##
## Call:
## glm(formula = cases ~ date, family = quasipoisson(link = "log"),
##     data = aids)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7046  -0.7978   0.1218   0.6849   2.1217
##
## Coefficients:
##               Estimate Std. Error t value
## (Intercept) -1.023e+03  6.806e+01  -15.03
## date         5.168e-01  3.425e-02   15.09
##             Pr(>|t|)
## (Intercept) 1.25e-11 ***
## date        1.16e-11 ***
## ---
## Signif. codes:
##   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1  ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 2.354647)
##
##     Null deviance: 677.26  on 19  degrees of freedom
## Residual deviance:  53.02  on 18  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

Diagnostics (`plot(g1)`)

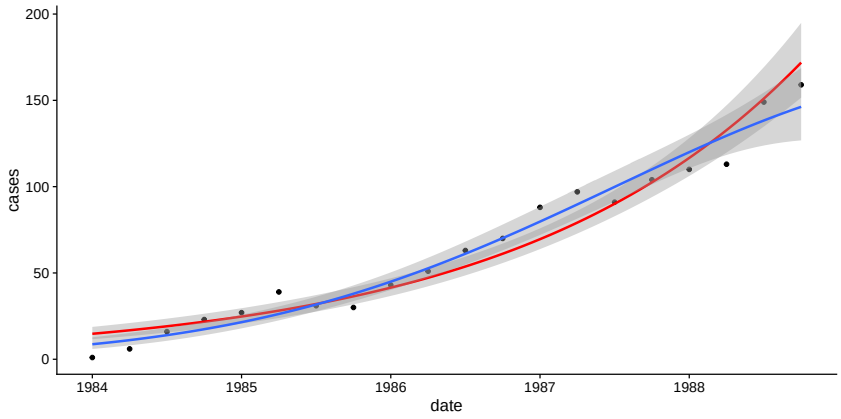



```
acf(residuals(g1))  ## check autocorrelation
```



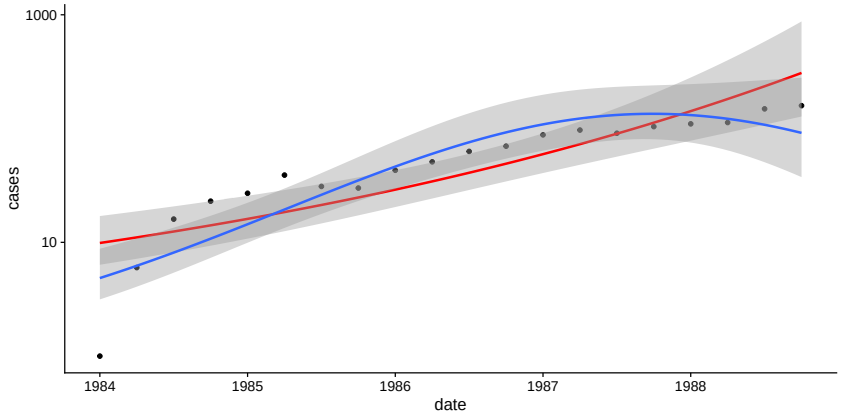
ggplot: check out quadratic model

```
print(gg2 <- gg1 + geom_smooth(method = "glm",
    formula = y ~ poly(x, 2), method.args = list(family = "quasipoisson")))
```



on log scale

```
print(gg2 + scale_y_log10())
```



improved model

```
g2 <- update(g1, . ~ poly(date, 2))
summary(g2)
```

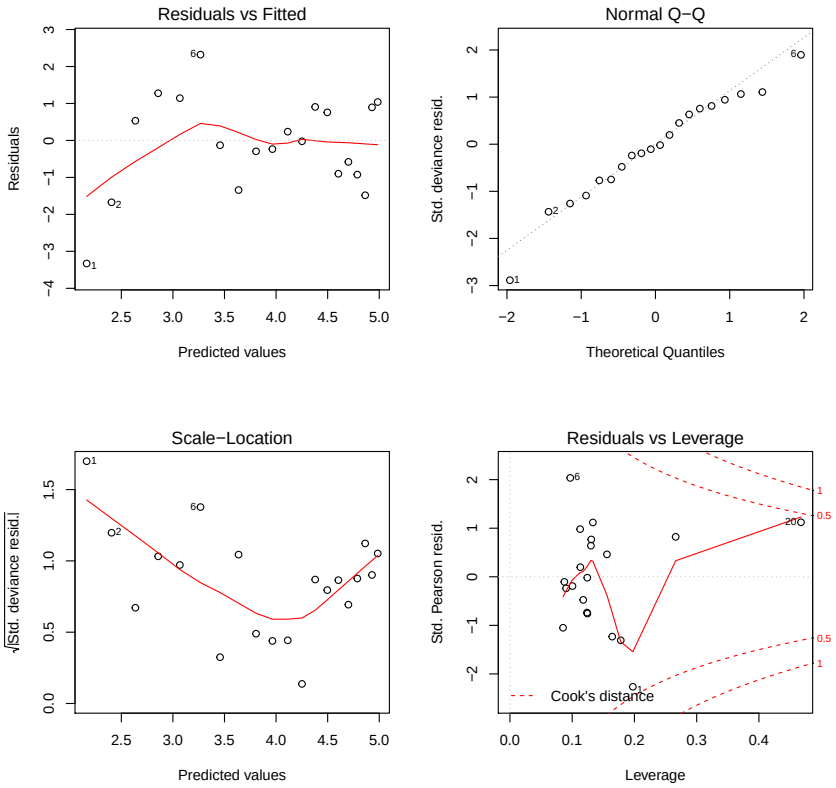
```
##
## Call:
## glm(formula = cases ~ poly(date, 2), family = quasipoisson(link = "log"),
##     data = aids)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3290  -0.9071  -0.0761   0.8985   2.3209
##
## Coefficients:
##                Estimate Std. Error t value
## (Intercept)     3.86859    0.05004  77.311
```

```
## poly(date, 2)1  3.82934    0.25162  15.219
## poly(date, 2)2 -0.68335    0.19716  -3.466
##                Pr(>|t|)
## (Intercept)     < 2e-16 ***
## poly(date, 2)1 2.46e-11 ***
## poly(date, 2)2  0.00295 **
## ---
## Signif. codes:
##   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1  ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.657309)
##
##     Null deviance: 677.264  on 19  degrees of freedom
## Residual deviance:  31.992  on 17  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

```
anova(g1, g2, test = "F")  ## for quasi-models specifically
```

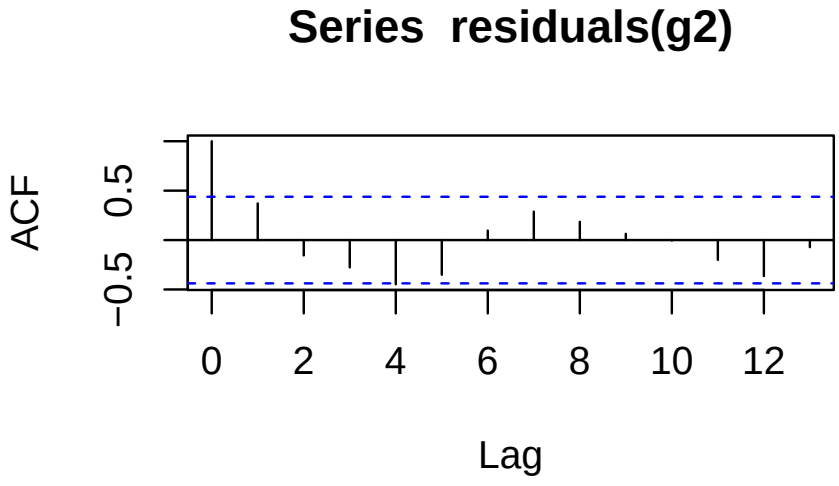
```
## Analysis of Deviance Table
##
## Model 1: cases ~ date
## Model 2: cases ~ poly(date, 2)
##   Resid. Df Resid. Dev Df Deviance      F
## 1        18     53.020
## 2        17     31.992  1   21.028 12.688
##     Pr(>F)
## 1
## 2 0.002399 **
## ---
## Signif. codes:
##   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1  ' ' 1
```

new diagnostics



autocorrelation function

```
acf(residuals(g2))  ## check autocorrelation
```



References

Gotelli, Nicholas J., and Aaron M. Ellison. 2004. *A Primer of Ecological Statistics*. Sunderland, MA: Sinauer.

Schielzeth, Holger. 2010. "Simple Means to Improve the Interpretability of Regression Coefficients." *Methods in Ecology and Evolution* 1: 103–13. doi:10.1111/j.2041-210X.2010.00012.x.

Tiwari, Manjula, Karen A. Bjorndal, Alan B. Bolten, and Benjamin M. Bolker. 2006. "Evaluation of Density-Dependent Processes and Green Turtle *Chelonia Mydas* Hatchling Production at Tortuguero, Costa Rica." *Marine Ecology Progress Series* 326: 283–93.

Wilkinson, G. N., and C. E. Rogers. 1973. "Symbolic Description of Factorial Models for Analysis of Variance." *Applied Statistics* 22 (3): 392–99. doi:10.2307/2346786.