Parameter interpretation and inference

Ben Bolker

October 2, 2018

Interpreting parameters

- continuous: units: depends whether scaled or not (talk about **scaling parameters**)
- categorical: differences between groups: depends on contrasts
- depends on presence of interactions
- **scale of measurement**: *link scale*

  log proportional The argument here is that if $\mu_0 = \exp \beta_0$ and $\mu_1 = \exp \beta_0 + \beta_1 x$,

  $$\begin{aligned} \mu_1 &= \exp(\beta_0 + \beta_1 x) \\ &= \mu_0 \exp(\beta_1 x) \\ &\approx \mu_0 (1 + \beta_1 x) \qquad \text{if } \beta_1 x \ll 1 \end{aligned}$$

  so for continuous predictors β_1 is the proportional change in the mean per unit change in x (for categorical predictors it's the proportional change between categories).

  Predicted values are the expected *geometric* mean of the category.

  logit log-odds change.

  - for $\beta \Delta x$ small, as for log (proportional)
  - for intermediate values, linear change in probability with slope $\approx \beta/4$
  - for large values, as for $\log(1 - x)$

  complementary log-log change in the *log-hazard*

  - hazard is the additional probability of failure per unit exposure
  - probability of failure in time t = $1 - \exp(\exp(\eta) t) = 1 - \exp(\text{hazard} \cdot t)$
  - rather than hazard, log-hazard is used as the linear predictor so η can be any real value (like log-odds)
  - $\beta \equiv$ proportional change in hazard
  - sensible for survival problems, cumulative exposure

Inference

Single vs multi-parameter

Single-parameter *Wald* vs. *likelihood ratio* test (LRT): the former is easier (it's what you get from `summary()`), because Wald standard errors of the estimates ($\sigma_{\hat{\beta}}$) are automatically available from the Hessian of the fitted model, so we can get p-values via a Z test on $\hat{\beta}/\sigma_{\hat{\beta}}$ (this is what `summary` gives) and confidence intervals via Normal confidence intervals on $\hat{\beta}$.

The *Hauck-Donner effect* occurs in cases of extreme parameter estimates (e.g. in the case of complete or near-complete separation), when the quadratic approximation is extremely poor: the hallmark is large parameter estimates (e.g. $|\hat{\beta}| > 10$) and very large confidence intervals (leading to small Z statistics and large p values).

You can get LRTs via

- `drop1(.,test="Chisq")` (only on parameters that can be dropped while respecting marginality, unless you use `scope= .~.`)
- `anova()`, explicitly testing different models:

```
reduced_model <- update(full_model,.~.-foo)
anova(full_model,reduced_model,test="Chisq")
```

  where `foo` is the parameter you want to test.
- or by hand (having fitted these models)

```
pchisq(deviance(reduced_model)-deviance(full_model),
       df=df.residual(reduced_model)-df.residual(full_model),
       lower.tail=FALSE)
```

You can get *profile confidence intervals* via `MASS::confint.glm`.

Multi-parameter If you want to test a hypothesis that multiple $\hat{\beta}$ values are simultaneously zero (i.e. you want to see if the overall effect of a factor is significant), you *could* do a Wald test: e.g. to test $\hat{\beta}_1 = \hat{\beta}_2 = 0$, you would calculate the sums of squares ($\hat{\beta}_1^2 + \hat{\beta}_2^2 = 0$) and the variance; the scaled result should be χ^2 distributed.

```
contr <- c(1,1)
t(contr) %*% vcov(model) %*% contr
pchisq(...)
```

This is what `car::Anova()` does. It generally makes more sense to do model comparisons. Do this with `anova()` or `drop1()` (`anova(model)` gives *sequential* (forward/"type I") tests: `anova(model1,model2,model3)` compares a specific sequence of models); these use LRTs (if `test="Chisq"`) or F tests (if `test="F"`, which you should use when the dispersion parameter is estimated (Gaussian, Gamma, or quasi-likelihood models).

Interactions/marginality issues

You have to be very careful when testing main effects in the presence of interactions. `drop1()` generally respects marginality, although you can do `drop1(.~.)` to get `drop1` to test *all* the effects (i.e not respecting marginality). ([1] is a standard reference from one of the proponents of respecting marginality: see Section 5.)

[1] Venables, W. N. (1998). Exegeses on linear models. 1998 International S-PLUS User Conference, Washington, DC

Your options with respect to marginality are:

- don't test main effects at all in the presence of interactions
- test main effects, but be very careful/aware that the meaning of the main effects depends on the parameterization/contrasts used
- split the data set and run separate analyses for each category involved in the interaction

Finite-size issues

In general LRTs are better than Wald tests, but even they make a (weaker) asymptotic assumption (not that the log-likelihood surface is quadratic, but that the deviance is χ^2 distributed). People generally ignore this problem since the number of observations is usually sufficiently large that this is a reasonable approximation, but [rarely used!] *Bartlett corrections* [2] are one approach to dealing with this issue.

[2] McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. Chapman and Hall, London; and Cordeiro, G. M. and Ferrari, S. L. P. (1998). A note on bartlett-type correction for the first few moments of test statistics. *Journal of Statistical Planning and Inference*, 71(1-2):261–269

If the dispersion parameter is estimated (rather than fixed, as it is for Poisson and binomial models), then we should use F tests ("quasi-LRT" for want of a better term) rather than χ^2, because the deviance differences are now scaled by the (χ^2-distributed) $\hat{\phi}$ (note that this does *not* address the issue of whether the deviance itself is really χ^2 distributed).

Bootstrapping

You can use bootstrap or parametric bootstrap samples to get p-values/confidence intervals that account for finite-size effects: for example, nonparametric bootstrapping resamples the data with replacement (using `sample(.,replace=TRUE)`).

Set up data and model:

```
data(lizards,package="brglm")
lizards <- transform(lizards,
                  gprop =grahami/(grahami+opalinus),
                  N= grahami+opalinus)
model1 <- glm(gprop~height+diameter+light+time,
              family=binomial, weights=N, data=lizards)
```

A function to take a bootstrap sample of the data, refit the model, and extract the coefficients:

```
bootFun <- function() {
    bootdat <- lizards[sample(nrow(lizards),replace=TRUE),]
    newmodel <- update(model1,data=bootdat)
    return(coef(newmodel))
}
```

Use a `for` loop to compute the samples:

```
nsamp <- 1000
set.seed(101)
bootParms <- matrix(NA,nrow=nsamp,ncol=length(coef(model1)))
for (i in 1:nsamp) {
    bootParms[i,] <- bootFun()
}
```

There are a variety of different approaches for computing bootstrap confidence intervals, but a simple one is to find the quantiles of the bootstrapped coefficients. Get 2.5% and 97.5% quantiles of each column (`MARGIN=2` specifies columns rather than rows), and transpose the results (because `apply` always returns its results column-wise):

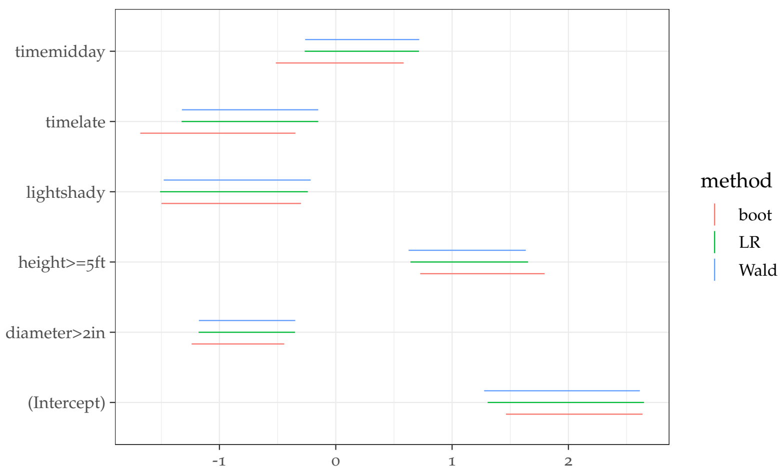
```
ptab <- t(apply(bootParms,MARGIN=2,quantile,c(0.025,0.975)))
rownames(ptab) <- names(coef(model1))  ## assign row names, for interpretability
print(ptab)

##                   2.5%      97.5%
## (Intercept)   1.4634553  2.6372131
## height>=5ft   0.7257110  1.7953832
## diameter>2in -1.2393941 -0.4427184
## lightshady   -1.4986304 -0.2987468
## timemidday   -0.5150444  0.5834759
## timelate     -1.6807495 -0.3471012
```

Compute two-sided p-values (twice the *smaller* of the two tails):

```
bootp <- apply(bootParms,
      MARGIN=2,
      function(x) 2*min(mean(x<0),mean(x>0)))
cbind(coef(summary(model1)),bootp)
```

```
##                Estimate Std. Error    z value     Pr(>|z|) bootp
## (Intercept)   1.9446882  0.3414768  5.6949348 1.234191e-08 0.000
## height>=5ft   1.1299913  0.2570898  4.3953169 1.106113e-05 0.000
## diameter>2in -0.7626343  0.2112694 -3.6097720 3.064662e-04 0.000
## lightshady   -0.8472755  0.3223825 -2.6281682 8.584606e-03 0.004
## timemidday    0.2271105  0.2501770  0.9077995 3.639842e-01 0.332
## timelate     -0.7368117  0.2990005 -2.4642486 1.373008e-02 0.006
```

Compare Wald, likelihood ratio, and bootstrap confidence intervals:



You can also use `car::Boot()` to do this more automatically:

```
bb <- car::Boot(model1)
confint(bb)
```

```
## Bootstrap bca confidence intervals
##
##                     2.5 %     97.5 %
## (Intercept)   1.2817166  2.4727391
## height>=5ft   0.6795117  1.6585993
## diameter>2in -1.2594149 -0.3021639
## lightshady   -1.4373456 -0.2311024
## timemidday   -0.4265333  0.6079867
## timelate     -1.5653378 -0.2834665
```

References

Cordeiro, G. M. and Ferrari, S. L. P. (1998). A note on bartlett-type correction for the first few moments of test statistics. *Journal of Statistical Planning and Inference*, 71(1-2):261–269.

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. Chapman and Hall, London.

Venables, W. N. (1998). Exegeses on linear models. 1998 International S-PLUS User Conference, Washington, DC.