Transformation: log transformation of responses

Philip Dixon

9/10/2020

Transformation.r: computing log transformations

We will use the emmeans library for "after the ANOVA" activities. It includes useful options for transformed data.

The data set is hamburger.xlsx, which we will read using read_excel in the readxl library. There are other libraries with functions to read .xls or .xlsx files, but read_excel is fast and flexible.

```
library(emmeans)
library(readxl)
hamb <- read_excel('../data/hamburger.xlsx')
hamb$trt.f <- factor(hamb$treatment)
hamb
```

```
## # A tibble: 12 x 3
##    treatment   cfu trt.f
##    <chr>     <dbl> <fct>
##  1 control   0.17  control
##  2 control   0.35  control
##  3 control   0.22  control
##  4 control   2.5   control
##  5 control   0.53  control
##  6 control   0.570 control
##  7 active    0.11  active
##  8 active    0.19  active
##  9 active    0.03  active
## 10 active    0.18  active
## 11 active    0.02  active
## 12 active    0.11  active
```

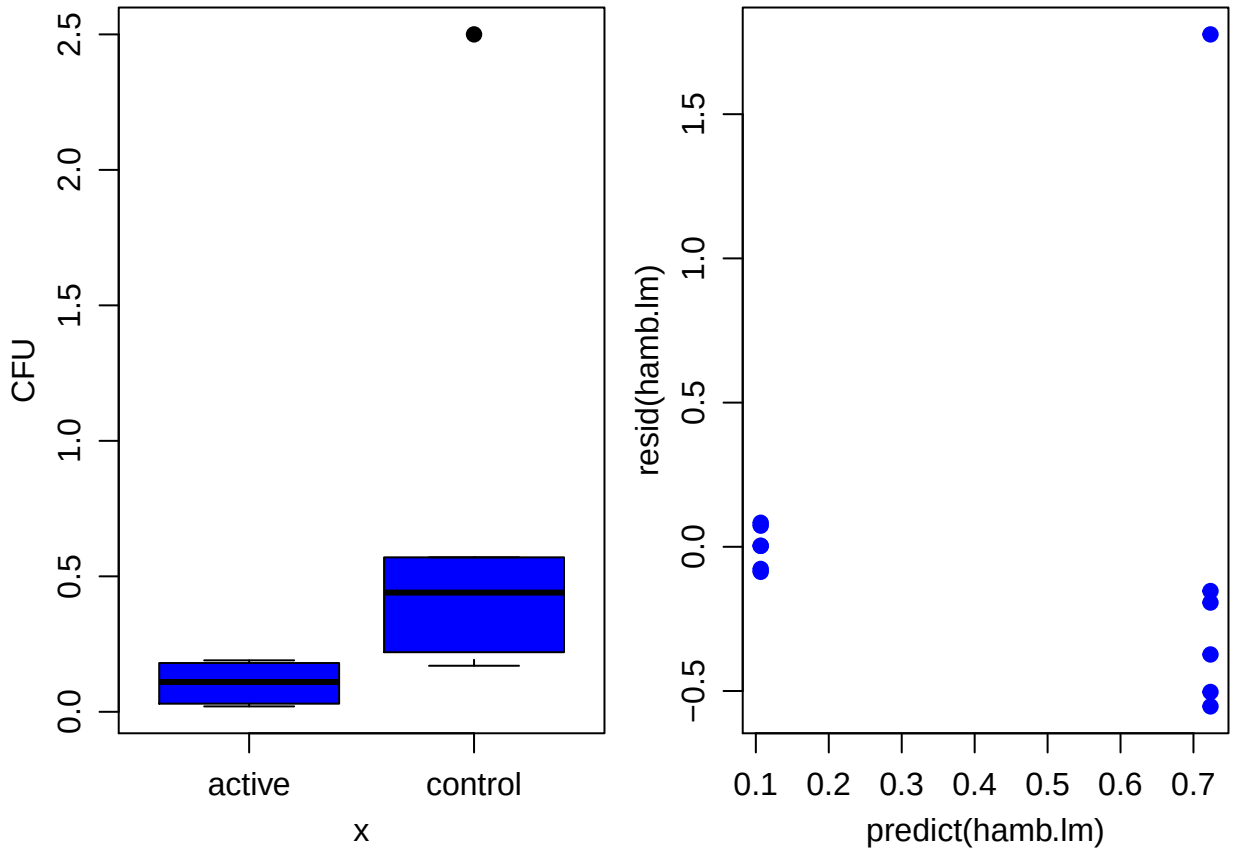
The result from read_excel is a tibble (tidy table), an extension of a data frame. Printing a tibble automatically just prints what fits on a page.

Most functions accept tibbles just like data frames. If you run into an exception, just convert the tibble back to a data frame: hamb <- as.data.frame(hamb)

data and residual vs predicted value plots of cfu

```
par(mfrow=c(1,2), mar=c(3,3,0,0)+0.3, mgp=c(2,0.8,0))
with(hamb, plot(trt.f, cfu, pch=19, col=4, ylab='CFU') )

hamb.lm <- lm(cfu ~ trt.f, data=hamb)
plot(predict(hamb.lm), resid(hamb.lm), pch=19, col=4)
```



log transformations: three ways to do it

1) compute a new variable with the log value using base r

```
hamb$logcfu <- log(hamb$cfu)
```

2) use dplyr functions and pipes.

This code is easier to read, especially with multiple transformations.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
hamb <- hamb %>% mutate(logcfu = log(cfu))
```

In both 1) and 2), you would then use logcfu as the response variable.

3) create log(cfu) "on the fly"

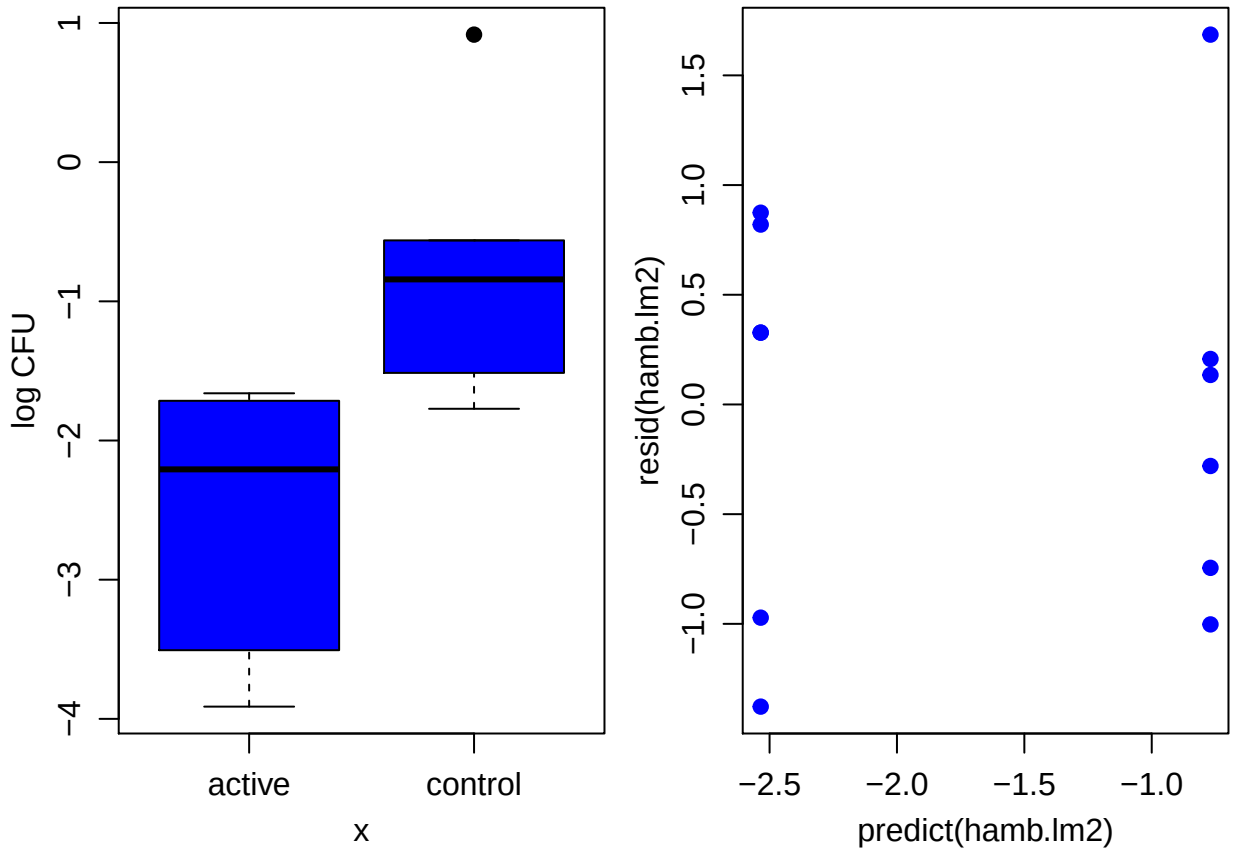
This is especially useful when you use emmeans, because then emmeans is aware that you have log transformed the response. It figures this out by look for log() in name of the response variable. If you use 1) or 2), emmeans just has an arbitrary variable name. You can tell it that it is log transformed (see the vignette on transformations if you want to do this), but I find it simpler to do the transformation inside the modeling function.

```
hamb.lm2 <- lm(logcfu ~ trt.f, data=hamb)
# or
hamb.lm2 <- lm(log(cfu) ~ trt.f, data=hamb)
```

Let's look at the data and residuals for a log scale analysis

```
par(mfrow=c(1,2), mar=c(3,3,0,0)+0.3, mgp=c(2,0.8,0))
with(hamb, plot(trt.f, log(cfu), pch=19, col=4, ylab='log CFU') )

plot(predict(hamb.lm2), resid(hamb.lm2), pch=19, col=4)
```



Estimating log-scale and backtransformed means

Usual use of emmeans returns estimates on the log scale. The output includes a note that these are on the log scale.

```
emmeans(hamb.lm2, 'trt.f')
```

```
##  trt.f    emmean    SE df lower.CL upper.CL
##  active  -2.535 0.388 10    -3.40  -1.6702
##  control -0.769 0.388 10    -1.63   0.0951
##
```

```
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
```

emmeans will do the backtransformation when you add type='response' either to emmeans or to summary. These are exp(log-scale means). But, this only works when emmeans knows that a transformation was used when fitting the model, i.e. when the response variable is written as log(y). These are estimates of the geometric mean (always) or median (when log(Y) is symmetric).

```
emmeans(hamb.lm2, 'trt.f', type='response')
```

```
##  trt.f   response     SE df lower.CL upper.CL
##  active    0.0793 0.0308 10   0.0334    0.188
##  control   0.4633 0.1798 10   0.1952    1.100
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```
summary(emmeans(hamb.lm2, 'trt.f'), type='response')
```

```
##  trt.f   response     SE df lower.CL upper.CL
##  active    0.0793 0.0308 10   0.0334    0.188
##  control   0.4633 0.1798 10   0.1952    1.100
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

Adding bias.adj=T requests exp(mean + variance/2), i.e. estimates the mean of Y when log(Y) is normally distributed. This is only an option to summary(). Doesn't work in emmeans().

```
summary(emmeans(hamb.lm2, 'trt.f'), type='response',
  bias.adj=T)
```

```
##  trt.f   response     SE df lower.CL upper.CL
##  active     0.115 0.0447 10   0.0485    0.273
##  control    0.673 0.2609 10   0.2833    1.596
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Bias adjustment applied based on sigma = 0.95042
```