A version of the logarithmic transformation that accommodates zeros

Werner A. Stahel, ETH Zurich

October 15, 2025

Abstract

The log transformation plays an important role in displaying and analyzing statistical data from non-negative variables. In practice, such variables will often include zero values, either when this is a genuinly acceptable value or because of technical limitations measuring or documenting tiny values. Since the logarithm of zero is minus infinity, a modification is needed to determine a finite value for the transformed zero.

The package `plgraphics` includes the function `logst` that implements such a modification. A short documentation is given in its help documentation.

1 Rationale for `logst`

When asking for a log transform of data, the difficulty of transfoming zeros pops up. A popular suggestion ask for just adding a constant to all values, a rule that has been named "started logs" by John Tukey. The constant is often chosen as 1, regardless of the data. This recipe has serious drawbacks.

a. If the constant does not depend on the data, the unit of measurements has a unwarranted effect of the outcome: if the data consists of lengths, the effect of the started log transformation is different when they are measured in inches or millimeters—or even meters. Adding one meter to lengths that are shorter makes the started log transformation essentially linear and therefore ineffective for its usual purpose.

a*. Thus, the constant should depend on the data. The first idea is to take the minimum value of the data. This seems to be too random in most contexts. Therefore, the whole data sample should be used to determine the constant.

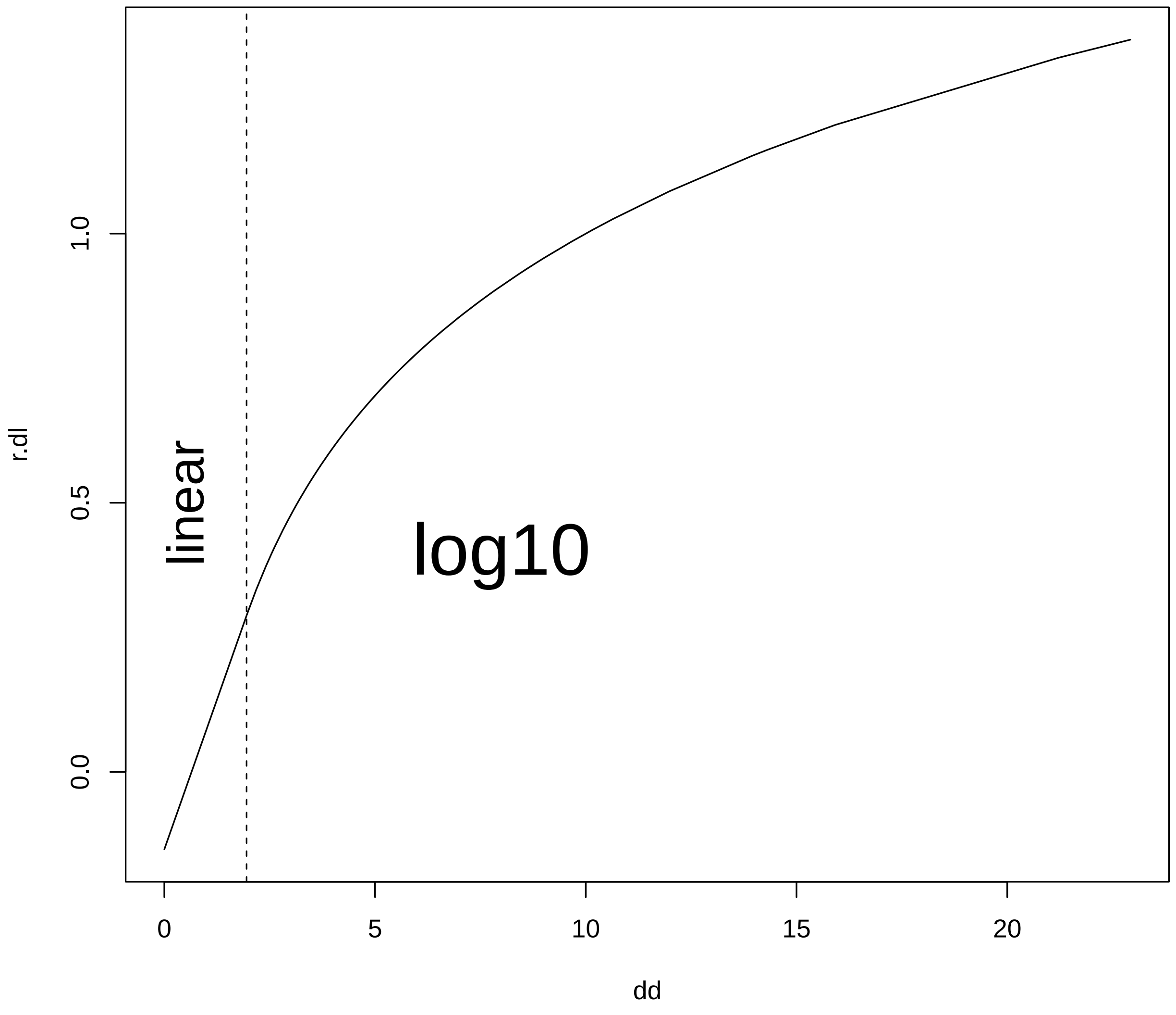
a**. The 'logst' function identifies, for lognormal data, 2 percent of the data as small (unless modified by setting the argument 'mult'). Thus the threshold is $c* = q1/(q3/q1)$, where $q1$ and $q3$ are the quartiles of the nonzero values in the sample.

b. The rules for logarithms will no longer hold even for data that is far from zero and thus no problem for a plain log transformation. That is: $logmod(x * y)! = logmod(x) + logmod(y)$. It is preferable to transform the bulk of the nonzero data by the simple log transformation and only define a modification for zero (and tiny) values.

b*. Formally, the log of zero is minus infinity. Any type of 'starting' the log transformation will bring it near the transformed values of 'tiny' data values, as described by a** above.

b**. The `logst` function continues the curve representing the log transform below $c*$ linearly with its derivative at this point, see the graph of `logst` in the figure below. Explicitly, the values below $c*$ are transformed to $log10(c*) + (x - c*)/(c * *log(10))$, and those larger than the threshold, to $log10(x)$.

```
dd <- c(seq(0,1,0.1),5*10^rnorm(100,0,0.2))
dd <- sort(dd)
r.dl <- logst(dd)
attr(r.dl, "threshold")

## [1] 1.95

cbind(data=dd, logst=r.dl, log10=log10(dd))[1:20,]

##       data   logst   log10
##  [1,] 0.00 -0.1438    -Inf
##  [2,] 0.10 -0.1216 -1.0000
##  [3,] 0.20 -0.0993 -0.6990
##  [4,] 0.30 -0.0771 -0.5229
##  [5,] 0.40 -0.0548 -0.3979
##  [6,] 0.50 -0.0326 -0.3010
##  [7,] 0.60 -0.0103 -0.2218
##  [8,] 0.70  0.0119 -0.1549
##  [9,] 0.80  0.0342 -0.0969
## [10,] 0.90  0.0564 -0.0458
## [11,] 1.00  0.0787  0.0000
## [12,] 1.64  0.2203  0.2140
## [13,] 1.96  0.2917  0.2917
## [14,] 2.18  0.3377  0.3377
## [15,] 2.41  0.3812  0.3812
## [16,] 2.54  0.4046  0.4046
## [17,] 2.61  0.4170  0.4170
## [18,] 2.81  0.4492  0.4492
## [19,] 2.82  0.4508  0.4508
## [20,] 2.94  0.4679  0.4679

## logst and log10 coincide for data values
##   larger than the threshold 1.55
plot(dd, r.dl, type="l")
abline(v=attr(r.dl,"threshold"),lty=2)

plpoints(8,0.4, plab="log10", csize=2)
text(0.5,0.5, "linear", cex=2, srt=90)
```



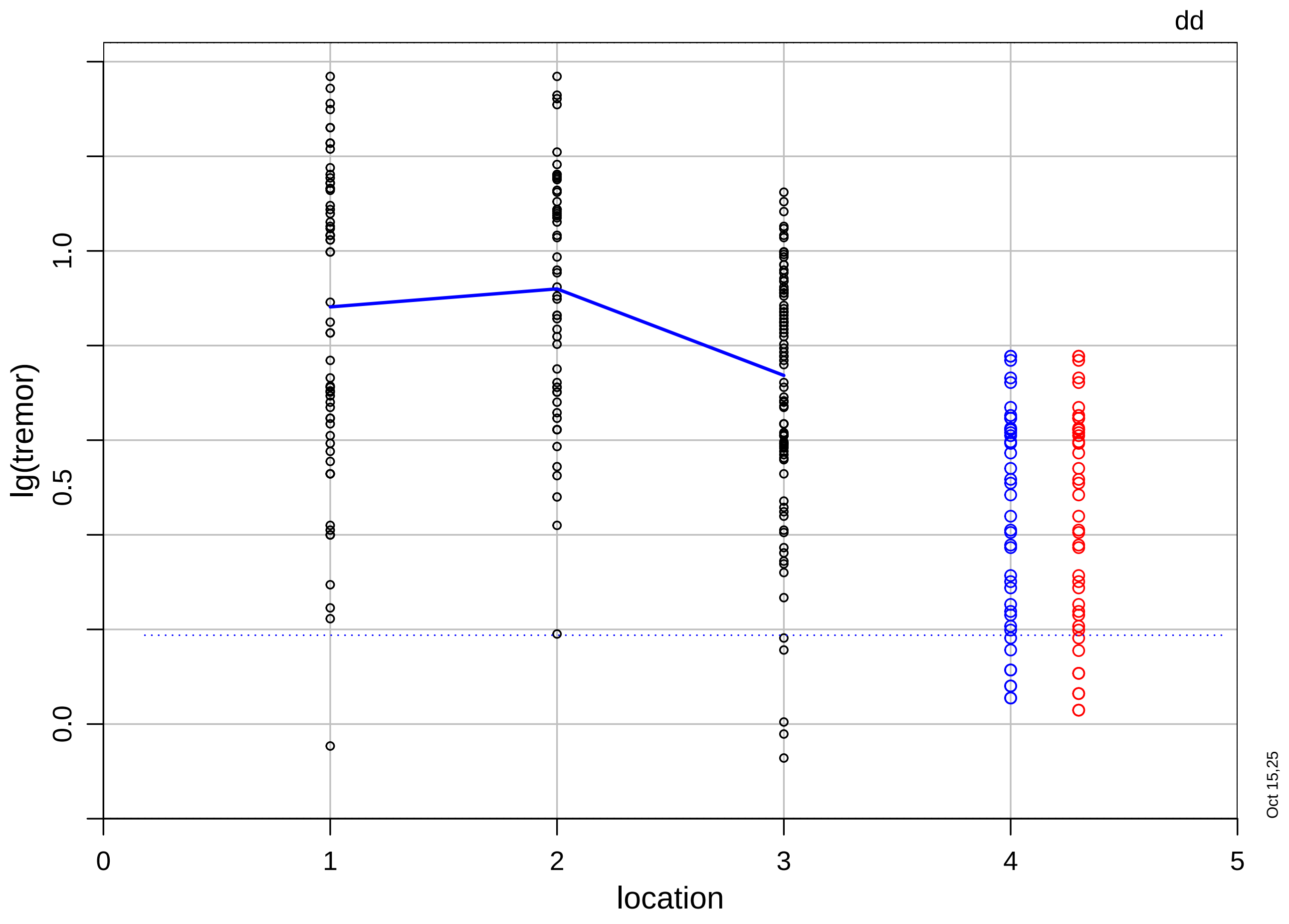
Why `log10`? The function uses log10 rather than log because this makes transformations and back-transformations in the mind much easier: If I learn the the transformed value is between 2 and 3, then

I know that the original value was in the hundreds. A more precise value is easily given by the first decimal of the log value, since $log10(2) \approx 0.3$, and therefore, $log10(5) = log(10/2) = 1 - 0.3 = 0.7$, and so on. This leads to concluding that the original value to a transformed value of 1.7 is 50, since $log10(50) = log10(10) + log10(5) = 1.7$.

Keeping the threshold constant. As we have argued, the threshold should depend upon the data. This would lead to a disadvantage when several samples are examined, since they would be treated with different transformations. In such a case, it may be appropriate to determine a common threshold and apply it to all the samples.

In the following example, we use the data of the target variable `tremor` in the first 3 `location`s to apply `logst` and obtain their threshold, and then get the data for the forth location, for which this same threshold should be applied.

```
##  example showing the effect of fixing the threshold
data(d.blast)

##  first 3 locations
dd <- d.blast[as.numeric(d.blast$location)<=3,]
y13 <- logst(dd$tremor)
plyx(y13~as.numeric(dd$location), data=dd,
  xlim=c(0,5), ylim=c(-0.2,NA), xlab="location", ylab="lg(tremor)")

## forth location
tremor4 <- d.blast$tremor[d.blast$location=="loc4"]

## transform tremor of location 4 alone
y4raw <- logst(tremor4)
plpoints(rep(4.3,47), y4raw, col="red")
abline(h=log10(attr(y4raw, "threshold")), lty=3, col="red")
## transform tremor of location 4 using threhold from locations 1-3
y4 <- logst(tremor4, threshold=attr(y13, "threshold"))
plpoints(rep(4,47), y4, col="blue")
abline(h=log10(attr(y13, "threshold")), lty=3, col="blue")
```



The red points result for applying `logst` without setting the threshold. The points below the dotted line remain the `log10` values even though they fall into the linear part of the appropriate transformation. It is easy to construct more drastic effects with other (artificial) datasets.