Anova Assumptions don't hold

If some of the assumptions of analysis of variance are not met, there may be nothing that can be done to remedy the problem. An example would be non-random assignment of experimental units to treatments in a randomized trial.
For other assumptions there are three possibilities:

- Ignore the problem (for example, anova is somewhat robust to deviations from normality and equal variances)
- Transform the variable to address non-normality (common) or unequal variances (uncommon)
- Use a non-parametric test
- model the non-normality explicitly

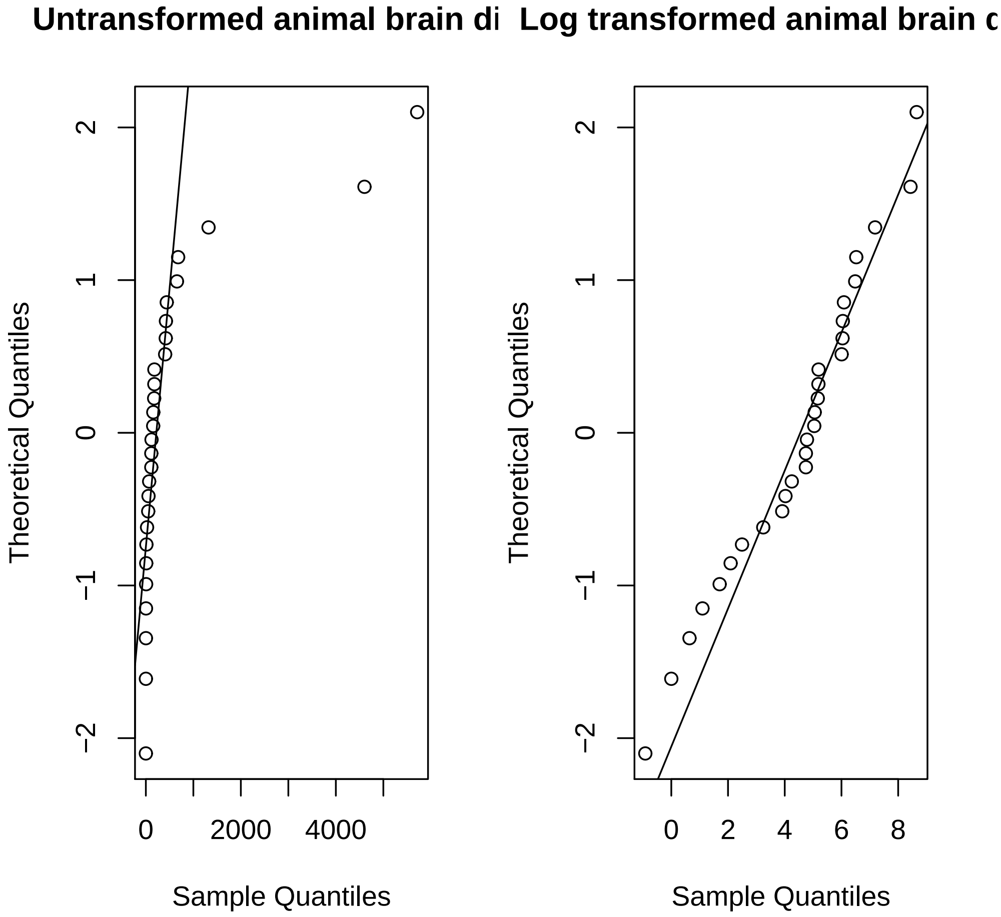
Transformations

- Log
- Log+1: this one and Log tend to lessen the right-skew that is common in many size/magnitude datasets
- arcsin-square root: this transformation is useful for making percentages or indices on $[0, 1]$ more normal.
- Square Root: Useful for count data that might be right-skewed

Although they can be useful, sometimes it is hard to interpret the results of inference performed on transformed variables.

Transformations: Log

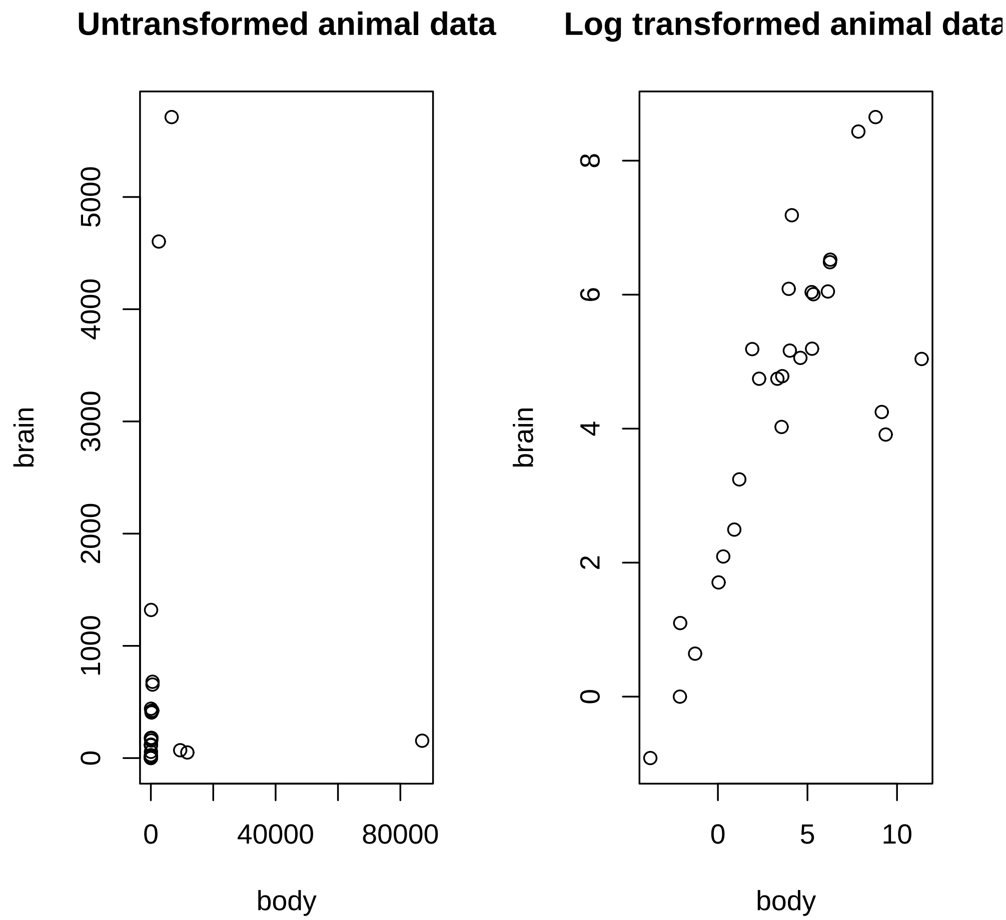
```
> library(MASS)
> data(Animals)
> par(mfrow = c(1, 2))
> qqnorm(Animals$brain, main = "Untransformed animal brain dist",
+     datax = T)
> qqline(Animals$brain, datax = T)
> qqnorm(log(Animals$brain), main = "Log transformed animal brain dist",
+     datax = T)
> qqline(log(Animals$brain), datax = T)
```



Transformations: Log

Sometimes the transformation allows a pattern to unfold

```
> par(mfrow = c(1, 2))
> plot(Animals, main = "Untransformed animal data")
> plot(log(Animals), main = "Log transformed animal data")
```



Transformations: arcsin square root

$$x_{\text{transformed}} = \sin^{-1}(\sqrt(x))$$

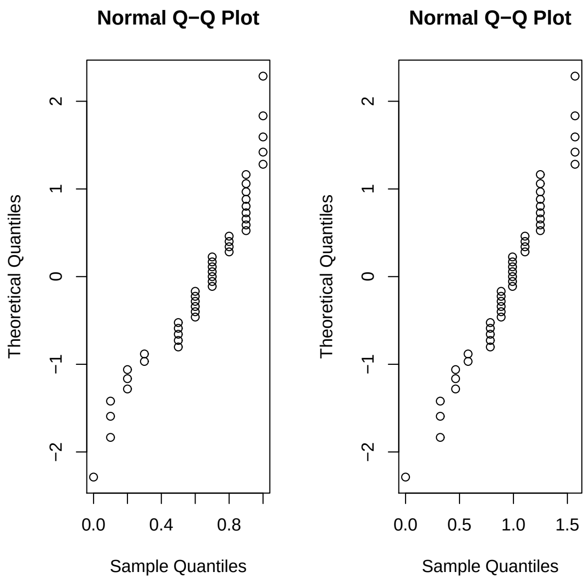
Transformations: arcsin square root

```
> pd

 [1] 0.0 0.1 0.1 0.1 0.2 0.2 0.2 0.3 0.3 0.5 0.5 0.5 0.5 0.5 0.6 0.6 0.6 0.6 0.6
[20] 0.6 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.8 0.8 0.8 0.8 0.9 0.9 0.9 0.9 0.9 0.9 0.9
[39] 0.9 0.9 1.0 1.0 1.0 1.0 1.0
```

Transformations: arcsin square root

```
> par(mfrow = c(1, 2))
> qqnorm(pd, datax = T)
> qqnorm(asin(sqrt(pd)), datax = T)
```



Transformations: arcsin square root

```
> library(ctest)
> shapiro.test(pd)

        Shapiro-Wilk normality test

data:  pd
W = 0.91, p-value = 0.001971

> shapiro.test(asin(sqrt(pd)))

        Shapiro-Wilk normality test

data:  asin(sqrt(pd))
W = 0.9554, p-value = 0.08148
```

Transformations, conclusion

Should you transform always?

- Transformations are great if you want to test the hypothesis that there is significant differences among treatments

- Unfortunately, transformation may
    - create non-symmetric confidence interval
    - cause biological interpretation to be difficult

Non-Parametric Tests

Every inferential method that we have looked at so far operates by contructing a test statistic based on estimates of population parameters and comparing to a known sampling distribution.
These are called parametric tests and make restrictive assumptions, especially with respect to the distribution of the data.
Non-parametric tests do not make assumptions about the distribution of data. As a result, they might best be known as *distribution-free* inferential statistics

Two samples

If you have two samples of randomly collected data that meet t-test assumptions other than the assumptions associated with the distribution of the data one can employ a test based upon the ranks of the observations. In general, all of the samples are pooled togther and ranked. Subsequent calculations are performed on the ranks

Two samples: Wilcoxon Rank Sum Test

One means to compare the location of two distributions is the Wilcoxon Two Sampled Rank Sum Test.
Assumes:

1. independent, random sampling of groups
2. continuous data
3. two groups must be similarly distributed

Wilcoxon test cont.

To perform:

1. First rank all data irrespective of group. If samples are tied, use average rank for tied individuals
2. Size of group 1: n_1 size of group 2: n_2
3. Sum ranks in the smaller group, call it ΣR.

4. Test statistic

$$C = n_1 n_2 + \frac{n_2(n_2+1)}{2} - \Sigma R$$

$$U = \begin{cases} \text{if} n_1 n_2 - C > C & n_1 n_2 - C \\ \text{else} & C \end{cases}$$

Wilcoxon test cont.

Use the `qwilcox()` function to calculate the critical value of U to reject the null at a particular α

Wilcoxon test example

```
> X <- c(104, 109, 112, 114, 116, 118, 118, 119, 121, 123, 125,
+     126, 126, 128, 128, 128)
> Y <- c(100, 105, 107, 107, 108, 111, 116, 120, 121, 123)
> X.Y <- c(X, Y)
> ranks <- rank(X.Y)
> X.ranks <- ranks[1:length(X)]
> Y.ranks <- ranks[(length(X) + 1):(length(X) + length(Y))]
> sum.R <- sum(Y.ranks)
> X.ranks
```

```
 [1]  2.0  7.0  9.0 10.0 11.5 13.5 13.5 15.0 17.5 19.5 21.0 22.5 22.5 25.0 25.0
[16] 25.0
```

```
> Y.ranks
```

```
 [1]  1.0  3.0  4.5  4.5  6.0  8.0 11.5 16.0 17.5 19.5
```

```
> C <- length(X) * length(Y) + (length(Y) * (length(Y) + 1))/2 -
+     sum.R
> C
```

```
[1] 123.5
```

Wilcoxon example cont

```
> if ((length(X) * length(Y) - C) < C) {
+     U <- C
+ } else {
+     U <- length(X) * length(Y) - C
+ }
> print(paste("ties: non-exact p-val", (1 - pwilcox(U, length(X),
+     length(Y)))))
```

```
[1] "ties: non-exact p-val 0.00990015503408959"
```

R-builtin

```
> wilcox.test(X, Y)

        Wilcoxon rank sum test with continuity correction

data:  X and Y
W = 123.5, p-value = 0.02320
alternative hypothesis: true location shift is not equal to 0
```