Quantitative Understanding in Biology
2.3 Quantitative Comparison of Models and ANOVA

Jason Banfelder

October 8th, 2024

1 Fitting a Michaelis-Menten Model to Myoglobin Binding Data

A classic mathematical model for enzyme kinetics is the Michaelis-Menten equation:

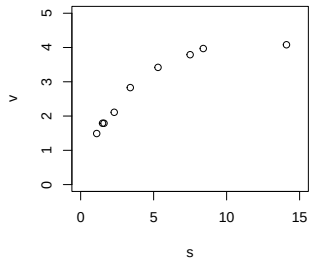
$$V = \frac{V_{max}[S]}{K_m + [S]} \tag{1}$$

Given data of V versus $[S]$ for a particular enzyme and substrate, we can determine the Michaelis-Menten parameters V_{max} and K_m using a regression procedure. Consider, for example, the following data for the association of myoglobin and oxygen.

P_{O_2} (torr)	1.1	1.5	1.6	2.3	3.4	5.3	7.5	8.4	14.1
$[O_2]$ (mL/dL)	1.49	1.79	1.79	2.11	2.83	3.42	3.79	3.97	4.08

We begin by entering the data into R and inspecting a basic plot.

```
myoglobin <- data.frame(s = c(1.1,  1.5,  1.6,  2.3,  3.4,  5.3,  7.5,  8.4, 14.1),
                        v = c(1.49, 1.79, 1.79, 2.11, 2.83, 3.42, 3.79, 3.97, 4.08))
plot(v ~ s, data = myoglobin, xlim = c(0, 15), ylim = c(0, 5))
```



Note that here we explicitly give limits for the ordinate and abscissa. In this case, allowing R to choose axis limits results in a deceiving plot.

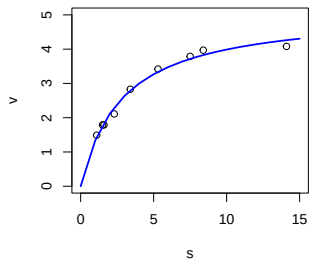
Next, we proceed with a non-linear regression in the normal fashion. Initial guesses for V_{max} and K_m are based on a quick glance at the plot. Recall that V_{max} is the maximal reaction rate, and K_m is the value of $[S]$ at which half of V_{max} is realized.

```
m0.myoglobin <- nls(v ~ Vmax * s / (Km + s),
                    start = list(Vmax = 4, Km = 2),
                    data = myoglobin)
summary(m0.myoglobin)
```

```
##
## Formula: v ~ Vmax * s/(Km + s)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Vmax   5.1171     0.1678   30.50 1.05e-08 ***
## Km     2.8282     0.2511   11.26 9.72e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1244 on 7 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 6.607e-06
```

A plot of the curve predicted by the model is consistent with the observed data:

```
x <- 0:15; lines(x, predict(m0.myoglobin, newdata = data.frame(s = x)),
                 col = "blue", lwd = 2)
```

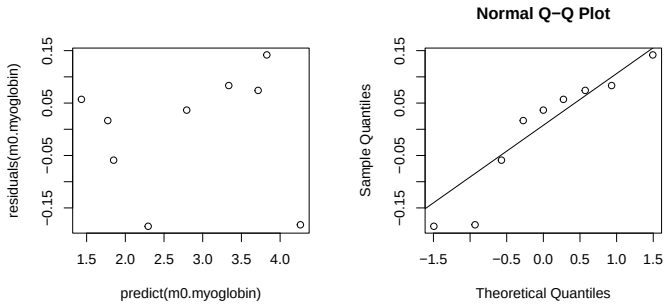


We proceed with quality control plots of residuals vs. fitted values, and a QQ-plot of the

residuals. It is difficult to evaluate these plots with so few data points, and the best that we can hope for is that we see nothing overtly wrong. For good measure, we also perform a Shapiro test for normality of the residuals.

```
plot(residuals(m0.myoglobin) ~ predict(m0.myoglobin))
qqnorm(residuals(m0.myoglobin))
qqline(residuals(m0.myoglobin))
shapiro.test(residuals(m0.myoglobin))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(m0.myoglobin)
## W = 0.8802, p-value = 0.1578
```



Finally, we check the confidence intervals of the model parameters to ensure that they include only physiologically feasible values, and that they are not too wide considering the small amount of data we have to work with.

```
confint(m0.myoglobin)
```

```
## Waiting for profiling to be done...
```

```
##         2.5%    97.5%
## Vmax 4.74741 5.535155
## Km   2.29631 3.476112
```

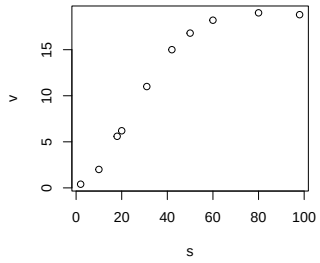
In this case, we are reasonably happy with our results. Ideally, we'd like a bit more data to rule out any systematic variation in the residuals and heteroscedasticity, but otherwise we are satisfied with our fit.

2 Fitting a Michaelis-Menten Model to Hemoglobin Binding Data

We will now repeat this exercise for similar data taken for hemoglobin. The experimental observations are:

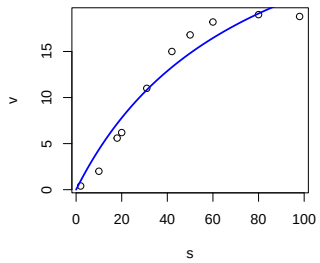
P_{O_2} (torr)	2	10	18	20	31	42	50	60	80	98
$[O_2]$ (mL/dL)	0.4	2.0	5.6	6.2	11.0	15.0	16.8	18.2	19.0	18.8

```
hemoglobin = data.frame(s = c(  2,  10,  18,  20,  31,  42,  50,  60,  80,  98),
                        v = c(0.4, 2.0, 5.6, 6.2, 11.0, 15.0, 16.8, 18.2, 19.0, 18.8))
plot(v ~ s, data = hemoglobin)
```



The plot of the raw data already suggests a sigmoidal shape that may not be consistent with our model. However, this could just be noise in the model, so we proceed objectively with a similar fit as before.

```
m0.hemoglobin <- nls(v ~ Vmax * s / (Km + s),
                     start = list(Vmax = 19, Km = 40),
                     data = hemoglobin)
x <- 0:100; lines(x, predict(m0.hemoglobin, newdata = data.frame(s = x)),
                  col = "blue", lwd = 2)
```



Here we see that the model curve does not fit the data too well. While we could proceed with our quality control plots, for current purposes we'll stop here and reconsider the model. It turns out that if you take into account that hemoglobin is a multimeric protein, and assume that affinity for binding at different sites is not independent, you get a more elaborate form of the Michaelis-Menten relationship, called the Hill model:

$$V = \frac{V_{max}[S]^n}{K_m^n + [S]^n} \tag{2}$$

The exponent, n, is called the Hill exponent, and is an indication of the degree of cooperativity the system exhibits. If $n > 1$, the system is said to exhibit positive cooperativity; if $n < 1$, the system exhibits negative cooperativity.

We also see that when $n = 1$, the model reduces to the Michaelis-Menten model. In other words, the Michaelis-Menten model is a special case of the Hill model. This relationship between the models is important, and has a specific term: the models are said to be 'nested'.

We proceed to fit the Hill model to our data:

```
m1.hemoglobin <- nls(v ~ Vmax * s ^ n / (Km ^ n + s ^ n),
                     start = list(Vmax = 19, Km = 40, n = 1),
                     data = hemoglobin)
```

```
## Error in numericDeriv(form[[3L]], names(ind), env, central = nDcentral):
Missing value or an infinity produced when evaluating the model
```

```
m1.hemoglobin <- nls(v ~ Vmax * s ^ n / (Km ^ n + s ^ n),
                     start = list(Vmax = 19, Km = 25, n = 1),
                     data = hemoglobin)
summary(m1.hemoglobin)
```

```
##
## Formula: v ~ Vmax * s^n/(Km^n + s^n)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Vmax  20.3000     0.5683   35.72 3.50e-09 ***
## Km    27.5289     1.0494   26.23 2.99e-08 ***
## n      2.4347     0.1789   13.61 2.72e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4781 on 7 degrees of freedom
```



```
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 1.174e-06
```

Note that on our first attempt, the iterative numerical procedure failed to converge. A more informed initial guess met with success. Getting such optimization to converge is sometimes more art than science, and occasionally it is not possible. Some tips to aid in convergence are:

1. Plot the curve predicted by the model at the initial guess, and adjust the parameters "by hand" to get a decent starting guess.
2. Try fitting the model with one or more of the parameters fixed. Then use the optimized values for the remaining parameters as starting points for a full optimization.
3. Make use of the `trace=TRUE` option in the `nls` function.
4. Try different algorithms; the `nls` function supports `algorithm = "plinear"` and `algorithm = "port"`.

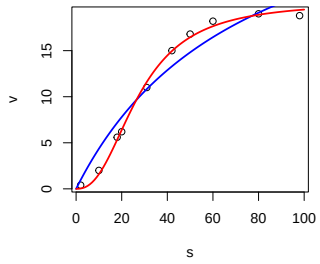
Also note that `nls` is designed to work with real data that contain some noise. If your data were generated from a function and all of the residuals were zero, `nls` would probably fail. This is counter-intuitive, as you would expect most optimizations to perform well when the error is zero. The reason for this is that R not only finds the best values for the parameters, but also makes estimates of their uncertainty; some non-zero residuals are necessary to avoid mathematical singularities.

We now plot the model-predicted curve, and inspect the confidence intervals of the parameters.

```
lines(x, predict(m1.hemoglobin, newdata = data.frame(s = x)),
      col = "red", lwd = 2)
confint(m1.hemoglobin)
```

```
## Waiting for profiling to be done...
```

```
##           2.5%     97.5%
## Vmax 19.145095 21.788067
## Km   25.382860 30.265201
## n     2.046159  2.879289
```

Everything seems reasonable so far. Additional checks on the model are left as an exercise.

We should note here that the physiologically accepted value of the Hill constant for hemoglobin is between 2.5 and 3.0.

3 A Return to Myoglobin

Given the success of our Hill model, and given that regular Michaelis-Menten kinetics are a special case of the Hill model, one might wonder why we don't just always use the Hill model. After all, if the system does not demonstrate cooperativity, the regression will tell us by reporting a Hill exponent of unity.

Let's try this approach with our myoglobin data. Our initial guess is informed by our previous run:

```
m1.myoglobin <- nls(v ~ Vmax * s ^ n / (Km ^ n + s ^ n),
                    start = list(Vmax = 5.1, Km = 2.8, n = 1),
                    data=myoglobin)
summary(m1.myoglobin)
```

```
##
## Formula: v ~ Vmax * s^n/(Km^n + s^n)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Vmax   4.7768     0.3532  13.523 1.01e-05 ***
## Km     2.4393     0.3860   6.319 0.000734 ***
## n      1.1398     0.1606   7.099 0.000392 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
##
## Residual standard error: 0.1252 on 6 degrees of freedom
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 5.015e-06
```

Here we see that the three parameter model fits the data almost as well as the two-parameter model. This can be quantified by looking the sum of the squares of the residuals (the objective function of the implicit optimizations we are performing whenever we run `nls`).

```
sum(residuals(m0.myoglobin) ^ 2)
```

```
## [1] 0.1083247
```

```
sum(residuals(m1.myoglobin) ^ 2)
```

```
## [1] 0.09411742
```

Inspecting the CIs for the parameters is informative as well:

```
confint(m1.myoglobin)
```

```
## Waiting for profiling to be done...
```

```
##           2.5%    97.5%
## Vmax 4.1971852 6.156505
## Km   1.8864886 4.542650
## n    0.7902735 1.532160
```

The first thing that you should notice is that the CI for the Hill exponent is quite wide; we could have reasonably significant positive or negative cooperativity. Comparing the CIs for the other parameters with those from the two-parameter model shows that the uncertainty in the three-parameter model is significantly larger. This alone is a reason for rejecting the three parameter model if we can; it will reduce the uncertainty in the parameter CIs. However, a more compelling argument is that of maximum parsimony, or Occam's razor. Given a choice between two models, if we don't have good evidence to support the more complex model (such as the cooperative Hill model), we should prefer the simpler one.

Another way of looking at the problem is to keep in mind here that we only have nine data points for myoglobin. A two parameter model therefore has seven degrees of freedom, while a three parameter model has six. This is not an insignificant change, and there is a real possibility that the Hill model represents an over-fit of the limited available data.

While choosing between models is often a judgment call that should integrate all available

scientific information, there are tools that help us in the decision. We will consider two, the F-test and an interesting, non-statistical approach called AIC.

4 The F-test for Model Comparison

Using the F-test to compare two models follows the classical framework for statistical testing. You state a null hypothesis, assume it is true, and then compute a p-value that gives the probability of observing your data (or something more extreme) under that assumption. If the probability is low enough, you reject the null hypothesis.

We know that the more complex model will always fit better because we have more parameters. If we start with the more complex model and remove a parameter, we expect that the SSQ will go up. In fact, we can quantify this expected change in SSQ: if the simpler model is the correct one, then we expect that the relative change in the SSQ should be about equal to the relative change in the degrees of freedom. If the complex model is correct, we expect the SSQ to change more than this amount.

The p-value computed by the F-test answers the question: assuming that the simpler model is the correct one, what is the probability that we see a change in SSQ at least large as the one we observed when we simplify the complex model. If this p-value is low, then we reject the null hypothesis and accept the more complex model.

From a purely statistical point of view, if the p-value is above our pre-determined cutoff, we could not make any conclusion. However, since either model is considered a viable candidate for explaining our data, we apply the principle of maximum parsimony, and accept the less complex model.

The F-test is intimately related with concepts from ANOVA. In fact, to perform an F-test for model comparison in R, simply use the `anova` function, passing it two models as parameters. We begin by comparing the classic Michaelis-Menten model with the Hill model for our myoglobin data.

```
anova(m0.myoglobin, m1.myoglobin)
```

```
## Analysis of Variance Table
##
## Model 1: v ~ Vmax * s/(Km + s)
## Model 2: v ~ Vmax * s^n/(Km^n + s^n)
##   Res.Df Res.Sum Sq Df   Sum Sq F value Pr(>F)
## 1      7   0.108325
## 2      6   0.094117  1 0.014207  0.9057  0.378
```

Note that in this case, the SSQ changed to 0.108 from 0.094, an increase of about 15%.

The degrees of freedom changed to 7 from 6, an increase of about 17%. The F-value is the ratio of these changes; since it is close to one, we don't expect these changes to be inconsistent with the null hypothesis. The magic (or, if you prefer, the mathematics) of the F-test is that it can convert this F-value into a p-value which tells us how surprised we are to see such an F-value assuming that the simpler model is correct. The reported p-value, 0.38, is not less than our standard cutoff of 0.05; we are not surprised. Therefore, we have no reason to reject the simpler Michaelis-Menten model. Invoking the principle of maximum parsimony, we therefore accept this simpler model as the better explanation for these data.

We now apply this test to the hemoglobin models:

```
anova(m0.hemoglobin, m1.hemoglobin)
```

```
## Analysis of Variance Table
##
## Model 1: v ~ Vmax * s/(Km + s)
## Model 2: v ~ Vmax * s^n/(Km^n + s^n)
##   Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
## 1      8    25.5516
## 2      7     1.6002  1 23.951  104.78 1.834e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, we see that the p-value is well below our pre-established cutoff. If the simpler model were correct, we'd be quite surprised to see as large a change in SSQ as we did. Note that SSQ went from 1.6 to 25.5, a nearly 1,500% increase, while the degrees of freedom changed from 7 to 8, a roughly 14% increase.

It is very, very important to know that the F-test is only applicable for nested models, and only when you are fitting them to the exact same data. You can't compare unrelated models with it (e.g., the power law and the asymptotic exponential models we investigated in the previous session). And you can't compare a transformed and non-transformed model with it (the data are not the same).

5 Using AIC to Compare Models

The derivation for Akaike's Information Criteria (AIC) is well beyond the scope of this course. It involves information theory, maximum likelihood theory, and entropy. We can get a rough feel for what the method is doing by looking at the resultant formula.

$$\mathrm{AIC} = n \cdot \ln\left(\frac{\mathrm{SSQ}}{n}\right) + 2\,(P+1) \tag{3}$$

Here, n is the number of observations, and P is the number of model parameters in the regression. We observe that the larger the SSQ, the higher the AIC will be. Also, the AIC increases as we add parameters to the model. Therefore, we can conclude that lower AICs are better. We can imagine that if we add a model parameter (increment P), the SSQ will go down. If the parameter was worth adding, the increase in the second term will be more than offset by a decrease in the first term.

By itself, the AIC is meaningless. The astute observer will realize that the SSQ has units of measure, and therefore there is an implicit standardization. We can therefore make the numerical value of the AIC whatever we like by altering the units of SSQ (or the standard value).

This ostensible shortcoming is overcome, however, when we look at the difference between the AICs of two models:

$$\Delta\mathrm{AIC} = n \cdot \ln\left(\frac{\mathrm{SSQ}_B}{\mathrm{SSQ}_A}\right) + 2\,(P_B - P_A) \tag{4}$$

The problem of the units of SSQ goes away. In practice, however, we can compute our AICs using consistent units, and select the model with the lower value.

A correction to AIC is necessary when n is not much greater than P. The corrected AIC equation is:

$$\mathrm{AIC}_C = \mathrm{AIC} + 2\frac{(P+1)\,(P+2)}{n-P} \tag{5}$$

We can use the AIC to compare any two models fitted to the same dataset. The models do not need to be nested; this makes the use of AICs a very powerful technique for comparing unrelated models.

R can compute AICs for us; unfortunately, it does not apply the above correction, so we need to do that ourselves. Applying this methodology to our myoglobin models again confirms that Michaelis-Menten kinetics is the preferred description.

```
n <- length(myoglobin$s)
p <- length(coefficients(m0.myoglobin))
AICC.m0.myoglobin <- AIC(m0.myoglobin) + 2 * (p + 1) * (p + 2) / (n - p)
p <- length(coefficients(m1.myoglobin))
AICC.m1.myoglobin <- AIC(m1.myoglobin) + 2 * (p + 1) * (p + 2) / (n - p)
c(michaelis.menten = AICC.m0.myoglobin, hill = AICC.m1.myoglobin)
```



```
## michaelis.menten             hill
##       -4.8091556       -0.8363698
```

Similarly, we see that the Hill model is preferred for the hemoglobin data.

```
n <- length(hemoglobin$s)
p <- length(coefficients(m0.hemoglobin))
AICC.m0.hemoglobin <- AIC(m0.hemoglobin) + 2 * (p + 1) * (p + 2) / (n - p)
p <- length(coefficients(m1.hemoglobin))
AICC.m1.hemoglobin <- AIC(m1.hemoglobin) + 2 * (p + 1) * (p + 2) / (n - p)
c(michaelis.menten = AICC.m0.hemoglobin, hill = AICC.m1.hemoglobin)
```

```
## michaelis.menten             hill
##         46.75993         23.76821
```

You are much more likely to see F-tests in the literature. Because these tests are based on the classical statistical framework, many people feel more comfortable with them. However, comparison of AICs can be more powerful, especially when dealing with non-nested models.

It turns out that the difference in AIC (or AIC_C) is related to the probability that one model is correct relative to another. A difference of 6 corresponds to a 95% chance that the lower scoring model is correct. Therefore, if a more complex model has a lower score than a simpler model, but the difference is less than 6, you may still want to stick with the simpler model, because the evidence in favor of the complex one is not overwhelming. Given two non-nested models (perhaps with the same number of parameters), you might simply choose the one with the lower AIC score, but appreciate that the difference between the models is not 'significant'. We were warned that some of this is more art than science.

6 Confidence Intervals with the F-Test [Optional]

An interesting application of the idea behind the F-test is that it can be used as an alternative means of estimating the uncertainty in model parameters. The basic idea is to use what we learned about F-tests to compare a model with zero parameters to our best fit model.

Consider the Michaelis-Menten myoglobin model. The SSQ for this model is 0.108325. We had $n = 9$ data points and $P = 2$ parameters, so we had $\texttt{DF} = 7$ degrees of freedom. Take this as model A.

Now consider a hypothetical model with no parameters; we would have $n = 9$, $P = 0$, and $\texttt{DF} = 9$. Take this as model B.

When we compare these two models, we compute the F statistic in the usual way:

$$F = \frac{\left(\frac{\text{SSQ}_B - \text{SSQ}_A}{\text{SSQ}_A}\right)}{\left(\frac{\text{DF}_B - \text{DF}_A}{\text{DF}_A}\right)} \tag{6}$$

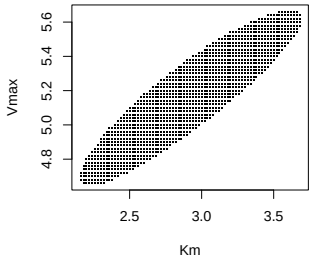
It turns out that we can compute the critical F-value for a 95% CI using R. This comes from the F-distribution, and is computed in R using the command: `qf(0.95, P, n - P)`. For our case $F_{crit} = 4.737$. This means that if the F-value computed for a comparison between models A and B is less than 4.737, we do not consider them to be significantly different.

Setting F to F_{crit} in the above equation allows us to compute SSQ_{crit}. For the myoglobin data, $\text{SSQ}_{crit} = 0.2549351$. This means that any model (with all parameters fixed) where $\text{SSQ} < 0.2549351$ is considered not significantly different from the best-fit model we have. Given any choice of parameters, we can compute SSQ and compare to this value.

```
ssq.crit <- 0.2549351
ssq.myoglobin <- function(Vmax, Km) {
  sum((Vmax * myoglobin$s / (Km + myoglobin$s) - myoglobin$v) ^ 2)
}
Vmax <- data.frame(Vmax = seq(2, 10, 0.02))
Km <- data.frame(Km = seq(1, 5, 0.02))
cases <- merge(Vmax, Km)
head(cases) # no pun intended here
```

```
##   Vmax Km
## 1 2.00  1
## 2 2.02  1
## 3 2.04  1
## 4 2.06  1
## 5 2.08  1
## 6 2.10  1
```

```
cases$ssq <- mapply(ssq.myoglobin, cases$Vmax, cases$Km)
plot(Vmax ~ Km, data = subset(cases, ssq < ssq.crit), pch = ".")
```

The resultant plot gives an envelope of values that are consistent with the best-fit model. Compare this to the asymptotic CIs reported by `confint`:

```
coefficients(m0.myoglobin)
```

```
##     Vmax       Km
## 5.117063 2.828153
```

```
confint(m0.myoglobin)
```

```
## Waiting for profiling to be done...
```

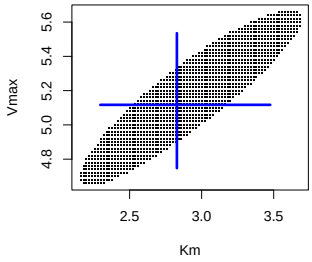
```
##         2.5%    97.5%
## Vmax 4.74741 5.535155
## Km   2.29631 3.476112
```

```
lines(rep(coefficients(m0.myoglobin)[["Km"]], 2),
      confint(m0.myoglobin, 'Vmax'),
      col = "blue", lwd = 3)
```

```
## Waiting for profiling to be done...
```

```
lines(confint(m0.myoglobin, 'Km'),
      rep(coefficients(m0.myoglobin)[["Vmax"]], 2),
      col = "blue", lwd = 3)
```

```
## Waiting for profiling to be done...
```

If you think about it, this plot makes sense. The combination of parameters V_{max}/K_m is the initial slope of the Michaelis-Menten curve. Based on the data we have (look back at the original plot), we have a pretty good idea of what that should be. However, determining the maximum (plateau) of the curve is quite difficult from our data (this is notoriously difficult experimentally; you need to go to very high values of $[S]$). To get K_m, which is the half-maximal concentration, we need a decent idea of V_{max}. So while there is significant uncertainty in both K_m and V_{max}, we do expect that they have a relationship (i.e., they are not independent).

7 Fitting with Categorical Variables

In all of the model-fitting examples that we've looked at so far, we have only considered cases where the explanatory variables were continuous. But in many cases, we work with systems where some (or all) of the explanatory variables are categorical. For example, we may want to investigate if any of a number of drugs have an effect on cognitive ability, as measured by how long it takes to solve a puzzle. To explore this, we begin by generating some simulated data.

```
control <- data.frame(    t = rnorm(10, mean = 7,  sd = 0.6),
                      group = factor("ctrl"))
drug_a  <- data.frame(    t = rnorm( 8, mean = 9,  sd = 0.6),
                      group = factor("drgA"))
drug_b  <- data.frame(    t = rnorm( 9, mean = 7,  sd = 0.6),
                      group = factor("drgB"))
drug_c  <- data.frame(    t = rnorm( 7, mean = 7,  sd = 0.6),
                      group = factor("drgC"))
drug_d  <- data.frame(    t = rnorm( 8, mean = 11, sd = 0.6),
                      group = factor("drgD"))

d <- rbind(control, drug_a, drug_b, drug_c, drug_d)
```



```
d[sample(nrow(d), 6), ] # sample 6 rows from the dataframe
```

```
##            t group
## 22  5.966859  drgB
## 10  6.955673  ctrl
## 9   8.062409  ctrl
## 41 11.139252  drgD
## 1   7.157122  ctrl
## 39 11.606003  drgD
```

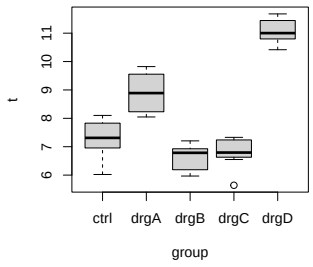
```
str(d)
```

```
## 'data.frame': 42 obs. of  2 variables:
##  $ t    : num  7.16 6.85 8.1 7.4 7.22 ...
##  $ group: Factor w/ 5 levels "ctrl","drgA",..: 1 1 1 1 1 1 1 1 1 1 ...
```

As you can see, we've synthesized a dataset where drug A has a small effect, and drug D has a large effect, while drugs B and C have no real effect at all. You'll also note that the number of subjects in each group isn't the same, but the SDs are. Also note that the group variable is a factor in R; this is critical to everything else we're going to do, so always check this before doing your analyses.

If we had just collected this data in the lab, the first thing we'd want to do is plot it. Note that when R knows that a variable is categorical, this is trivial.

```
plot(t ~ group, data = d)
```



One way to think about this experiment is that there is a model that can predict the time to solve the puzzle as a function of the group that your subject was in. Using R's model formula syntax for linear models, this is expressed the same way as you would for a numerical (continuous) predictor:


```
m.drug <- lm(t ~ group, data = d)
```

But the result is very different looking, because the dependent variable (time, in this case) is a function of a categorical variable.

```
summary(m.drug)
```

```
##
## Call:
## lm(formula = t ~ group, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31494 -0.43685  0.03644  0.48370  0.91999
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.3367     0.1843  39.804  < 2e-16 ***
## groupdrgA     1.5658     0.2765   5.664 1.79e-06 ***
## groupdrgB    -0.6923     0.2678  -2.585   0.0138 *
## groupdrgC    -0.5506     0.2872  -1.917   0.0630 .
## groupdrgD     3.7364     0.2765  13.514 6.97e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5829 on 37 degrees of freedom
## Multiple R-squared:  0.8998,Adjusted R-squared:  0.8889
## F-statistic: 83.04 on 4 and 37 DF,  p-value: < 2.2e-16
```

When you fit a linear model like this, under the hood R creates a set of dummy variables, one for each level of the factor; we'll write these as xs. In this case the set of variables might be written as x_{ctrl}, x_A, x_B, x_C, and x_D. For each datapoint, the x corresponding to the group's level is set to one, and all of the others are set to zero. It is as if the data looked something like this

```
##            t group x_ctrl x_A x_B x_C x_D
## 10  6.955673  ctrl      1   0   0   0   0
## 42 11.285075  drgD      0   0   0   0   1
## 23  6.927736  drgB      0   0   1   0   0
## 34  6.711829  drgC      0   0   0   1   0
## 25  6.906656  drgB      0   0   1   0   0
## 8   6.021768  ctrl      1   0   0   0   0
```



```
## 15  9.486211  drgA      0   1   0   0   0
## 7   7.762736  ctrl      1   0   0   0   0
```

and then R fitted this to a linear model with the equation:

$$y = \beta + \beta_A x_A + \beta_B x_B + \beta_C x_C + \beta_D x_D + \epsilon \tag{7}$$

In this case, β is the mean of the control group, and β_i is the difference in the means of the group treated with drug i and the control group (i.e., it is an estimate of the effect of drug i relative to the control).

8 Introduction to ANOVA as Method of Model Comparison

In the above section, we developed a model that allows the mean response for each group to take on a different value. Such a model will have as many parameters as there are levels in the `group` factor. We've seen that when models have lots of parameters, there is a danger of overfitting, so we should ask ourselves if the data might be better explained by a model where a single mean suffices to explain the data from all of the groups. Biologically speaking, this would be a case where none of the drugs have an observable effect. Such a model would have a trivially simple form of...

$$y = \beta + \epsilon \tag{8}$$

... and be computed with the command:

```
m.null <- lm(t ~ 1, data = d)
summary(m.null)
```

```
##
## Call:
## lm(formula = t ~ 1, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4647 -1.2396 -0.7408  1.3418  3.5722
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```



```
## (Intercept)   8.1065     0.2699   30.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.749 on 41 degrees of freedom
```

For this model, the optimal value of β is unsurprisingly the mean of all of the measurements (or, alternatively, the mean of the one and only group for this simple model).

We can then use the `anova()` function to compare these two models. This answers the question: Are the data better explained by a model where each drug might have an effect (i.e. has its own mean), or one where the drugs are assumed to have no effect (all measurements are sampled from a distribution with the same mean)? Of course the "drug" model will have a lower SSQ than the "null" model, but the F-test performed by the `anova()` function tells us if the reduction in the sum of squares is worth the added complexity of the additional four parameters in the "drug" model.

```
anova(m.null, m.drug)
```

```
## Analysis of Variance Table
##
## Model 1: t ~ 1
## Model 2: t ~ group
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     41 125.41
## 2     37  12.57  4    112.84 83.038 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the strong p-value here we can see that the null hypothesis that all group means are the same has been squarely rejected, leading us to conclude that at least some of the true means differ. This kind of test is often called an omnibus test, because it tests the relationship between many means in a single test.

You should be aware that there is another way to get to the same result. This uses the `aov()` function, which is a bit more direct, but doesn't show you (or even compute) the group means.

```
a <- aov(t ~ group, data = d)
summary(a)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## group        4 112.84   28.21   83.04 <2e-16 ***
## Residuals   37  12.57    0.34
```



```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In either case, if you run an omnibus test and get a result that indicates that the more complex model is preferred, your next questions will likely be along the lines of "OK, so now that I am convinced that not all of the means are the same, can you tell me which groups differ, and by how much?" You probably want to know which drugs have a (statistically) significant effect relative to the control, and possibly which drugs are (statistically) significantly better than others. Answering such questions is in the realm of what are called post-tests (because you perform them after the omnibus test). There are many procedures for post-tests, and the details depend on which comparisons you want to make. As you hopefully appreciate by now, post-test procedures will account for multiple hypothesis testing concerns.

One common post-testing procedure is known as Tukey's Honest Significant Differences. This computes 95% CIs for the differences between all groups. Tukey's HSD procedure is similar to a series of pairwise t-tests followed up by a multiple hypothesis testing correction, but has the benefit of using all of the pooled variance information, making it a bit more powerful.

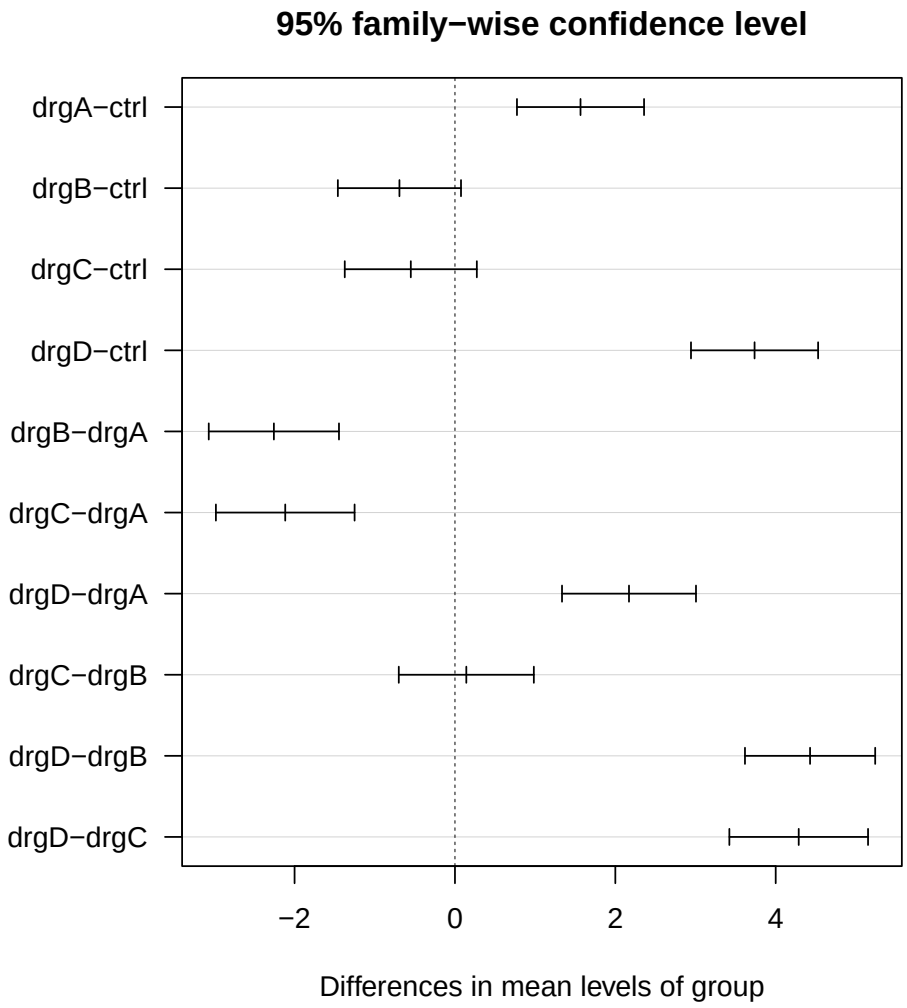
R makes this particular procedure almost too easy if you've used the `aov()` function to perform your ANOVA:

```
(tHSD <- TukeyHSD(a)) # remember, outer ()s to assign and print in one step
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = t ~ group, data = d)
##
## $group
##                  diff        lwr        upr     p adj
## drgA-ctrl  1.5658368  0.7732179  2.3584557 0.0000171
## drgB-ctrl -0.6923259 -1.4600917  0.0754398 0.0940236
## drgC-ctrl -0.5505516 -1.3740232  0.2729200 0.3266882
## drgD-ctrl  3.7363877  2.9437687  4.5290066 0.0000000
## drgB-drgA -2.2581627 -3.0701167 -1.4462088 0.0000000
## drgC-drgA -2.1163884 -2.9812066 -1.2515703 0.0000003
## drgD-drgA  2.1705509  1.3350571  3.0060446 0.0000001
## drgC-drgB  0.1417743 -0.7003242  0.9838728 0.9884934
## drgD-drgB  4.4287136  3.6167597  5.2406676 0.0000000
## drgD-drgC  4.2869393  3.4221211  5.1517575 0.0000000
```

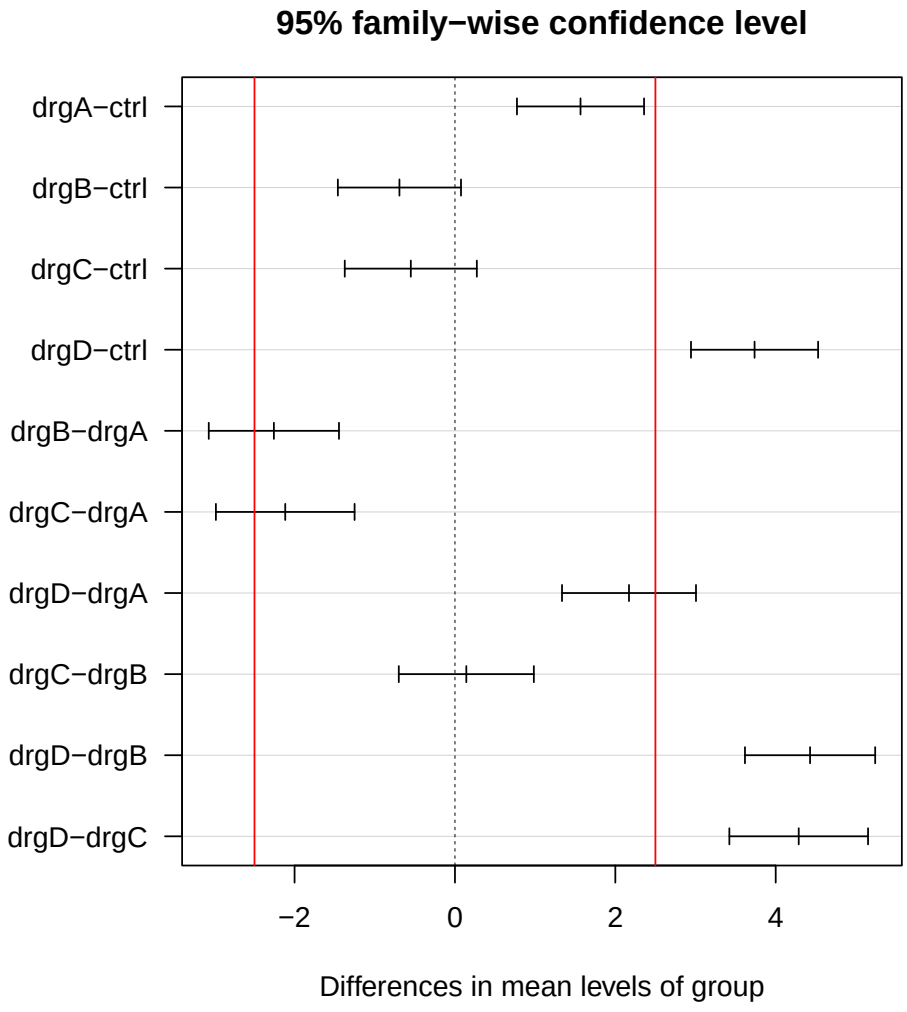


```
par(mar=c(5,8,2,1)) ; plot(tHSD, las=1) # horizontal x-axis labels
```



It is important to remember that, so far, these plots and analyses only address *statistical* significance. If, for example, you believe that differences in time to complete the cognitive task of less than 2.5 are not biologically significant, then you might adorn the plot as follows:

```
par(mar=c(5,8,2,1)) ; plot(tHSD, las=1) # horizontal x-axis labels
abline(v = c(-2.5, 2.5), col="red")
```

Your interpretation might then be that although drug A has a statistically significant effect relative to the control condition, it is not plausible that the difference is biologically significant. Drug D, however, clearly has biologically significant activity relative to control. Similarly, while we can see that while drug D has statistically significantly different activity than drug A, we can't tell if the difference is large enough to be considered biologically significant.

There are other post-test procedures to choose from; for example if you're only interested in comparing each drug to the control condition, but not to each other, you might look at Dunnett's procedure. Often, in R, you'll need to pull in addtional packages (and figure out how to use them!) to get access to some of the more complex alternatives.

8.1 Some cautions around ANOVA

There are three assumptions that go into an ANOVA analysis:

1. The data within each group is sampled from a normal distribution
2. The SDs of the groups are the same
3. The data are independent

Like many statistical tests we've covered, ANOVA is pretty robust to the first assumption. As we've done before, unless your data are pathologically non-normal, you're probably OK on the first point.

Data independence is more a function of study design and sampling methods than of statistical analysis. Other than to re-iterate that it is important to ensure that data collected are indeed independent, we won't discuss this further here.

The second assumption, that of equal SDs, is important because ANOVA is not particularly robust to this assumption. As a rule of thumb, if the SD of one group is three or more times that of another, you very well might have something to worry about. The effects are magnified when the groups have substantially different ns, and especially so when the ns is small (say less than 5 or 6 per group). The effect is manifested as a substantial departure of the true Type I error rate from the α cutoff used. Depending on the particulars, this could be an increase or a decrease in the deviation.

You can use the `leveneTest()` function in the `car` package to assess this.

```
library(car)

## Loading required package:  carData

leveneTest(t ~ group, data = d)

## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  4  1.4267 0.2444
##       37
```

The null hypothesis for the Levene test is that the variances of all the groups are the same. Here we see that the p-value for the Levene test is above 0.05. Thus, we cannot reject H_0; it is plausible that the assumptions of equal variances across groups is true, and we probably don't need to worry about ANOVA problems due to unequal variances. In practice, one usually performs a Levene test before running `anova()` or `aov()`.

8.2 Taking ANOVA further

In this example, we've looked at the simplest case of an ANOVA: a so-called one-way between subjects ANOVA. If you have two categorical explanatory variables, you can perform a two-way ANOVA by using a model formula that looks something like `y ~ a + b` if you want a model where the effects of each factor are independent and additive, or `y ~ a * b` if you want a model where the factors interact.

Working with complex experiment designs in R can be challenging for the beginner, in large part because it is difficult to find consistent, systematic guidance that isn't replete with technical language that you may not be familiar with. A good resource for learning about complex designs is the software package Prism, published by GraphPad. Prism does a good job of asking you how your data is structured, and guiding you to appropriate tests and procedures based on your responses. It also does a excellent job of explaining what different options mean in real-world, jargon-free terms, and offers sound but opinionated advice. Performing a guided analysis in Prism, and then reproducing the results in R, is a good way of both learning priniciples and building confidence that you're using the tools correctly, while still maintaining the benefits of reproducibility, automatability, scalability, and the breadth of methods available in R. Plus, Prism is expensive, so you may not want to assume that everyone you collaborate with will have access to it.

Finally, we point you to the excellent text "Designing Experiments and Analyzing Data," by Maxwell and Delaney, for a very complete yet readable review of the theory of ANOVA. At over 1,000 pages, this book may appear intimidating, but is remarkably accessible. In particular, this book develops ANOVA theory primarily based on a model comparison approach, unlike most texts which emphasize almost exclusively a partioning of variance approach. To my mind, the model comparison approach is more intuitive and generalizable to complex designs. Be sure to pick up the third edition of this book; the second edition (published in 2004) works examples in two other statistical programs (SAS and SPSS), while the third edition includes examples in R.

