

Linear models: not just straight lines!

Class 5 in MATH 615: Approaches to Quantitative Analysis in the Life Sciences

Refresher on data types — ANOVA as a linear model — General linear models — Curved linear models — Non-linear models — Generalized linear models

Setup

Data types

- **Class discussion**

Numerical real

Numerical integer (a.k.a. counts)

Ordinal (a.k.a. ranks)

Categorical (a.k.a. nominal, or ‘words’)

ANOVA

- **ANOVA-type data — predictor is categorical**

In ANOVA, data are organized into different groups, or *treatments*, corresponding to some non-quantitative variation called the *factor*. This means that the predictor is a nominal (aka. categorical) variable. An example would be comparing the heights of males and females. Sex is a category.

If there is just one factor, then we have a one-way ANOVA. The model can be written like this:

$$H_A : y_{g,i} = \mu_g + \epsilon_i$$

where g distinguishes the treatment, or group. There may be two, three, or many groups in a factor. This model assumes that the different groups have different means but the same variance, which is often reasonable. The null model is that there is *no* difference in the group means, and the observations can therefore be modeled as coming from a single group with a single mean:

$$H_0 : y_{g,i} = \mu + \epsilon_i$$

Here are some data from three groups ($k = 3$), "A", "B" and "C".

Out[482]/TableForm=

Treatment	y
A	2.97352
A	4.2354
A	1.88305
A	2.709
A	2.62321
A	1.4652
A	2.95513
A	1.92135
A	2.28301
A	1.94908
B	5.94963
B	2.8595
B	3.70635
B	2.59146
B	5.95486
B	2.91273
B	3.80929
B	2.43043
B	3.22364
B	2.43377
C	4.83946
C	5.12966
C	5.51918
C	5.85913
C	6.70794
C	5.73099
C	5.65897
C	4.98476
C	5.27563
C	5.75119

These are the means of the three groups.

```
Out[483]= {2.4998, 3.58717, 5.54569}
```

The question is: can we consider them as different (i.e., being drawn from more than one underlying population)?

■ ANOVA as another kind of linear model

To fit an ANOVA, we turn the categorical variables into an X matrix with the first column being the ‘intercept’. This is the same as in regression, except that the intercept will end up being the overall mean of the y_i . The second and third columns represent the difference between the overall mean and the group means of the "A" and "B" groups respectively. Thus, for group "A", we put a 1 in the second column (and a 0 in the third), for group "B" we put a 1 in the third column (and a 0 in the second). What about group "C"? Well, if we know the overall mean and two of the group means, then we can calculate the third group mean. So instead of having a fourth column, we put a "-1" in both columns two and three for group "C". This is called *sigma-restricted encoding*.

Out[485]/TableForm=

A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
A	1	1	0
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
B	1	0	1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1
C	1	-1	-1

These are the calculations that describe each observation. We call the overall mean μ , the difference between that and the mean of group "A" as μ_{DA} , and the difference between that and the mean of group "B" as μ_{DB} .

Out[430]/TableForm=

A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
A	$\mu + \mu_{DA}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
B	$\mu + \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$
C	$\mu - \mu_{DA} - \mu_{DB}$

We then apply the linear solution we first looked at with simple regression. Remember this?

$$\hat{\beta} = (X'X)^{-1} X'y$$

```
In[486]:= y = anovaData [ [All, 2] ] ;
designMatrix = Inverse [Transpose [xMatrix] . xMatrix] ;
betaHat = {mu, muDA, muDB} = designMatrix.Transpose [xMatrix] . y
```

```
Out[488]= {3.87755, -1.37776, -0.290385}
```

The values are estimates of the overall mean and the two differences. To get estimates of the actual group means, we do the calculations

```
In[489]:= estimatedMeans = {mu + muDA, mu + muDB, mu - muDA - muDB}
```

```
Out[489]= {2.4998, 3.58717, 5.54569}
```

Thus we have the following results

Out[492]/TableForm=

	y	\hat{y}	Residuals
A	2.97352	2.4998	0.473722
A	4.2354	2.4998	1.7356
A	1.88305	2.4998	-0.616748
A	2.709	2.4998	0.209207
A	2.62321	2.4998	0.12341
A	1.4652	2.4998	-1.03459
A	2.95513	2.4998	0.455337
A	1.92135	2.4998	-0.57844
A	2.28301	2.4998	-0.216782
A	1.94908	2.4998	-0.550714
B	5.94963	3.58717	2.36246
B	2.8595	3.58717	-0.727662
B	3.70635	3.58717	0.119183
B	2.59146	3.58717	-0.995711
B	5.95486	3.58717	2.36769
B	2.91273	3.58717	-0.674434
B	3.80929	3.58717	0.222125
B	2.43043	3.58717	-1.15673
B	3.22364	3.58717	-0.363522
B	2.43377	3.58717	-1.1534
C	4.83946	5.54569	-0.706231
C	5.12966	5.54569	-0.416034
C	5.51918	5.54569	-0.0265082
C	5.85913	5.54569	0.313442
C	6.70794	5.54569	1.16224
C	5.73099	5.54569	0.185299
C	5.65897	5.54569	0.113278
C	4.98476	5.54569	-0.560933
C	5.27563	5.54569	-0.270058
C	5.75119	5.54569	0.205502

And we can check the standard deviation of our residuals.

```
In[493]:=  $\sigma_{\text{Hat}} = \text{Sqrt}[\text{residuals.residuals} / \{\text{Length}[\text{residuals}] - 1 - 1\}][[1]]$ 
```

```
Out[493]= 0.930904
```

So we have fit our ANOVA model using the same cool trick as our regression model! The similarities extend further, because the ratios of mean squares also follow the F distribution, so we can use that for our test of statistical significance. The main difference is in the degrees of freedom. For the ANOVA *model*, the number of degrees of freedom is $k - 1$ (one less than the number of groups — the -1 is for the overall mean). The total degrees of freedom is $n - 1$ (as in the regression model — again, the -1 is for the overall mean), and so the error degrees of freedom is the difference between these: $(n - 1) - (k - 1)$.

Let's look at the output in the usual tabular form. We use the same function, `LinearModelFit`, that we used for regression. The only difference is that we do an ANOVA by specifying one or more of the predictor variables to be “nominal” (another word for “categorical”).

```
In[494]:= nom = LinearModelFit[anovaData, {Group}, {Group}, NominalVariables -> Group];
nom["ANOVA Table"]
```

```
Out[495]=
```

	DF	SS	MS	F-Statistic	P-Value
Group	2	47.6523	23.8262	26.5125	4.26452×10^{-7}
Error	27	24.2643	0.898678		
Total	29	71.9166			

■ Two-way ANOVA

In a two-way ANOVA we have two *factors*, and our data are split up into different combinations of the treatments in each factor. So, for example, we might test whether the addition of nitrogen fertilizer, phosphorus fertilizer, *or both* to soil increased the size of some crop. Factor one is nitrogen (present or not). Factor two is phosphorus (present or not). This is called a crossed design.

(0 Nitrogen only)
 (Phosphorus only Nitrogen and phosphorus)

Here are some example data:

Out[508]/TableForm=

	Factor 1 treatments	Factor 2 treatments	y
	0	0	1.6671
	0	0	-0.0714258
	0	0	3.75623
	0	0	1.25068
	0	0	2.71396
	0	0	3.10845
	0	0	3.90426
	0	0	3.82853
	0	0	0.581401
	0	0	2.86563
	N	0	4.50483
	N	0	4.65143
	N	0	2.06575
	N	0	3.11513
	N	0	3.61186
	N	0	2.05015
	N	0	3.1834
	N	0	5.58473
	N	0	2.23845
	N	0	4.98147
	0	P	2.74306
	0	P	2.41813
	0	P	3.18606
	0	P	2.24444
	0	P	3.59262
	0	P	-0.111464
	0	P	4.46018
	0	P	0.528763
	0	P	3.2957
	0	P	2.41067
	N	P	7.09679
	N	P	7.1291
	N	P	6.84481
	N	P	5.84092
	N	P	5.91319
	N	P	7.39984
	N	P	6.7996
	N	P	6.94027
	N	P	7.20345
	N	P	7.68765

Here is the model specification in *Mathematica*. Make sure you are comfortable interpreting it.

```
In[509]:= nom2 = LinearModelFit[anovaData2, {Nitrogen, Phosphorus},
  {Nitrogen, Phosphorus}, NominalVariables -> {Nitrogen, Phosphorus}];
nom2["ANOVATable"]
```

	DF	SS	MS	F-Statistic	P-Value
Nitrogen	1	79.7211	79.7211	37.6814	4.07751×10^{-7}
Phosphorus	1	28.954	28.954	13.6856	0.000698736
Error	37	78.2794	2.11566		
Total	39	186.954			

The analysis shows that both factors have a significant effect on the mean size (both p -values are less than 0.05). But we can go a bit further. Let's look at the means for each treatment combination. The combinations are

$$\begin{pmatrix} 0 & \text{Nitrogen only} \\ \text{Phosphorus only} & \text{Nitrogen and phosphorus} \end{pmatrix}$$

And the corresponding means are

```
Out[516]/MatrixForm=
```

$$\begin{pmatrix} 2.36048 & 3.59872 \\ 2.47681 & 6.88556 \end{pmatrix}$$

Notice that the mean for the double-fertilizer treatment is even higher than if you added the separate effects of nitrogen and phosphorus alone. This suggests an *interaction* between the factors — a greater-than-additive effect. We can include that as a possibility in our analysis — it is entered as the product of two factors.

```
In[511]:= nom3 = LinearModelFit[anovaData2, {Nitrogen, Phosphorus, Nitrogen * Phosphorus},
  {Nitrogen, Phosphorus}, NominalVariables -> {Nitrogen, Phosphorus}];
nom3["ANOVATable"]
```

	DF	SS	MS	F-Statistic	P-Value
Nitrogen	1	79.7211	79.7211	53.9983	1.15239×10^{-8}
Phosphorus	1	28.954	28.954	19.6117	0.0000848308
Nitrogen Phosphorus	1	25.1303	25.1303	17.0218	0.000208528
Error	36	53.149	1.47636		
Total	39	186.954			

This shows that the interaction term is significant as well. This what you would expect if, for example, nitrogen and phosphorus were both strongly limiting for the plant in question. Adding each one separately causes a small increase in plant size, but only adding both allow the plant to grow properly. Note that interaction effects do not need to have the same sign as the main effects — one can image two treatment factors that separately cause an increase in the response variable, but which, when together cause no increase, or even a decrease. In that case, neglecting the interaction term might lead you to conclude that neither factor has any impact. Here's an example:

Out[523]/TableForm=

	Factor 1 treatments	Factor 2 treatments	y
	0	0	0.875057
	0	0	1.19385
	0	0	1.78448
	0	0	1.00746
	0	0	1.78237
	0	0	1.77126
	0	0	0.964535
	0	0	3.26436
	0	0	0.373044
	0	0	2.19512
	N	0	5.20329
	N	0	2.82773
	N	0	1.57234
	N	0	3.85496
	N	0	3.73371
	N	0	1.93019
	N	0	2.94073
	N	0	2.7571
	N	0	5.62141
	N	0	1.82617
	0	P	2.43663
	0	P	0.51437
	0	P	2.70393
	0	P	1.89509
	0	P	4.20357
	0	P	3.5709
	0	P	1.88274
	0	P	4.36146
	0	P	3.58671
	0	P	1.25522
	N	P	-0.185828
	N	P	1.27158
	N	P	0.813978
	N	P	1.53987
	N	P	1.79986
	N	P	-0.106393
	N	P	2.25487
	N	P	1.07877
	N	P	1.43867
	N	P	2.92893

The treatment combination means are

Out[524]/MatrixForm=

$$\begin{pmatrix} 1.52115 & 3.22676 \\ 2.64106 & 1.28343 \end{pmatrix}$$

The main effects-only model concludes no relationships:

	DF	SS	MS	F-Statistic	P-Value
Nitrogen	1	0.302721	0.302721	0.160368	0.691121
Out[528]= Phosphorus	1	1.69507	1.69507	0.897972	0.349474
Error	37	69.8437	1.88767		
Total	39	71.8415			

The model with interaction effect reveals the truth:

	DF	SS	MS	F-Statistic	P-Value
Nitrogen	1	79.7211	79.7211	53.9983	1.15239×10^{-8}
Out[530]= Phosphorus	1	28.954	28.954	19.6117	0.0000848308
Nitrogen Phosphorus	1	25.1303	25.1303	17.0218	0.000208528
Error	36	53.149	1.47636		
Total	39	186.954			

More complicated *general* linear models

The matrix-based linear modeling framework can be extended to multiple predictors, mixing both categorical and continuous variables, and also to multiple response variables. Here is an example of having a categorical factor with four treatments and a continuous predictor, *temp*.

Out[532]/TableForm=

Factor	Temp	Response
treatment1	0.871185	0.90533
treatment3	5.79948	1.08313
treatment1	3.04928	0.297175
control	1.42131	0.903796
treatment2	6.38398	0.803086
treatment2	4.17104	0.422311
treatment2	1.05659	0.87167
treatment2	7.2511	1.13507
control	1.26086	0.587324
control	7.69884	0.206798
control	2.96393	0.424535
treatment2	2.73864	0.461841
treatment1	8.48694	0.0212778
treatment2	0.973383	1.02894
treatment1	7.27226	0.131134
treatment3	6.29295	1.23493
treatment2	5.01837	0.61219
treatment1	4.62513	0.292697
treatment2	9.5692	0.472294
control	2.27793	0.544152
control	3.40335	0.267596
treatment3	2.39345	1.19777
treatment2	3.69554	0.657017
treatment1	6.3153	0.661256
treatment2	5.51728	0.672724
control	4.27702	0.805626
treatment3	9.39699	0.848894
treatment3	2.9902	1.15378

treatment2	8.84694	0.838467
treatment2	9.16579	0.88199
treatment1	9.97725	0.128312
treatment1	6.8244	0.753592
control	7.02974	0.679205
treatment2	3.70718	0.940712
treatment3	4.17463	1.57228
control	9.39967	-0.0987066
treatment3	7.04658	0.898559
control	4.54829	0.664733
treatment2	8.38158	0.968559
treatment2	8.96418	0.815204
treatment1	2.81877	0.898084
treatment2	0.584505	1.29574
control	0.197115	0.680894
treatment2	6.85532	0.993281
treatment1	7.05062	0.0262493
treatment2	2.65673	0.626661
control	2.12846	0.593743
treatment2	4.41713	0.422252
control	7.28348	-0.090681
control	4.82941	0.287011
treatment1	0.762403	1.08177
control	3.93231	0.305683
treatment3	0.923614	2.0037
treatment3	4.4302	1.58289
control	4.96072	0.256077
treatment1	4.44488	0.58122
treatment1	5.71126	0.401015
treatment3	0.713504	1.42639
treatment1	3.067	0.343495
treatment2	5.13134	1.09071
treatment2	8.34382	0.786144
control	2.36346	0.441453
control	4.98357	0.118653
treatment3	7.32005	0.92454
treatment3	4.39472	1.45877
control	2.96662	0.511075
treatment3	8.05983	0.782384
treatment1	5.25221	0.649921
treatment3	4.44698	1.31483
treatment3	7.50826	1.20543
treatment1	8.05261	0.0826237
treatment2	3.23082	0.456945
treatment1	5.03789	1.0131
treatment2	8.55894	0.223281
treatment1	2.03496	0.635145
control	8.62395	0.195751
treatment2	2.75805	0.897567
treatment2	9.2381	1.06568

```

treatment3  3.64145    1.52099
treatment3  3.50656    1.62846
treatment1  4.48499    0.452506
treatment2  9.8605     0.574711
treatment1  5.1881     0.824716
treatment1  7.73021    0.920263
control    1.99135     0.437908
treatment3  9.69295     1.59431
treatment3  5.55909    0.922722
control    9.24427    -0.104914
treatment1  5.93182     0.450118
treatment1  6.28905     0.642432
treatment1  1.91825     0.316586
treatment3  1.37866     1.05731
treatment1  7.51016     0.0407864
treatment1  8.89308     0.0587054
control    1.44878     0.404608
treatment3  0.0463195    1.1546
treatment1  9.94677     0.636733
treatment3  0.0107488    1.49666
treatment2  3.10952     0.665753
treatment3  1.57934     1.32628

```

We expect the response to vary with *temp* — on its own, this would be a regression. But each treatment might have a *different* relationship between the response and *temp*. This kind of set-up is called an analysis of covariance (ANCOVA) — *temp* is the covariate. The `LinearModelFit` function allows the situation to be described very simply.

```

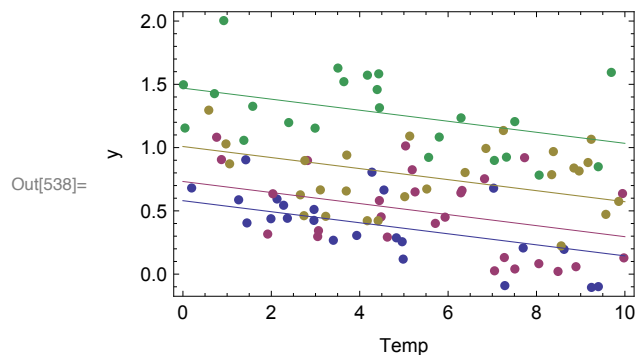
In[533]:= lm = LinearModelFit[data, {Treatment, Temp},
    {Treatment, Temp}, NominalVariables -> Treatment];

```

Obtain the ANOVA table:

	DF	SS	MS	F-Statistic	P-Value
Treatment	3	11.1035	3.70116	50.5715	1.27068×10^{-19}
Temp	1	1.4298	1.4298	19.5364	0.0000262094
Error	95	6.95274	0.0731867		
Total	99	19.486			

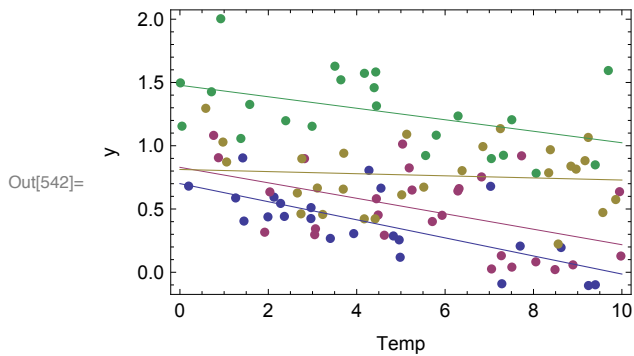
We can visualize the data by coloring according to treatment and plotting the relationship of each treatment to *temp*.



This model forces the slope of the relationship between *y* and *temp* to be the same — just the intercept is allowed to vary. We can allow the slopes to vary as well by introducing an interaction between treatment and temperature.

```
In[539]:= lm = LinearModelFit[data, {Treatment, Temp, Treatment * Temp},
  {Treatment, Temp}, NominalVariables -> Treatment];
```

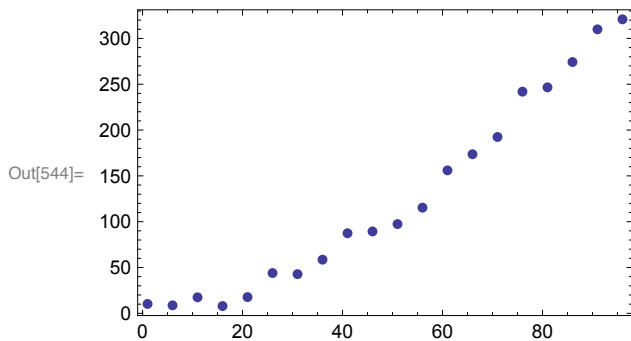
	DF	SS	MS	F-Statistic	P-Value
Treatment	3	11.1035	3.70116	52.4606	7.38827×10^{-20}
Temp	1	1.4298	1.4298	20.2661	0.0000196999
Temp Treatment	3	0.462021	0.154007	2.18291	0.0953425
Error	92	6.49072	0.0705513		
Total	99	19.486			



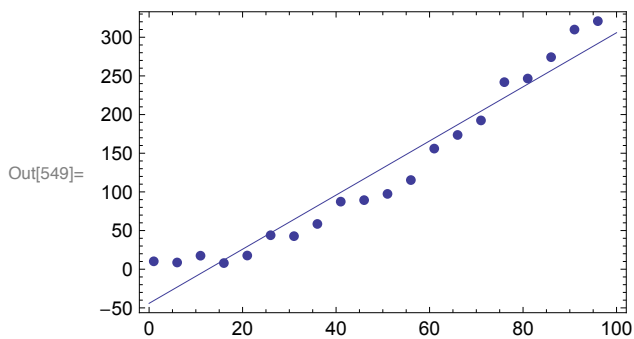
The slopes do vary, but the effect is not significant ($p \approx 0.095$).

Some linear models are curved!

You might be forgiven for thinking that linear models — at least regressions — are straight lines. Not so! Here are some data



A linear fit is clearly not ideal



A curved function, for example a polynomial that includes an x^2 term, might be better:

$$y_i = c + m x_i + n x_i^2 + \epsilon_i$$

This kind of a model can easily be implemented in the linear framework by simply generating a new predictor variable that is the existing variable x , but squared. So our predictor matrix now becomes

Out[550]/TableForm=

1	1	1
1	6	36
1	11	121
1	16	256
1	21	441
1	26	676
1	31	961
1	36	1296
1	41	1681
1	46	2116
1	51	2601
1	56	3136
1	61	3721
1	66	4356
1	71	5041
1	76	5776
1	81	6561
1	86	7396
1	91	8281
1	96	9216

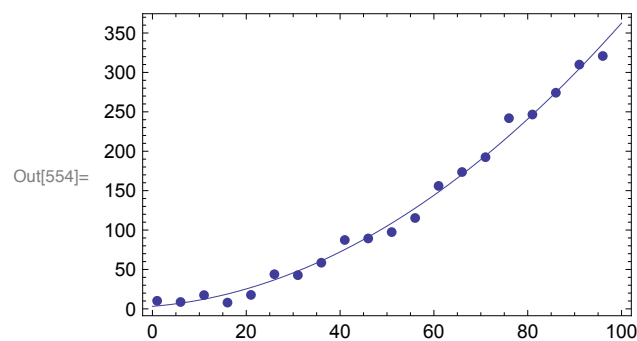
The usual fitting procedure is applied. Using a built-in routine

```
In[551]:= lm = LinearModelFit[data, {x, x^2}, {x}];
lm["ANOVATable"]
lm[x]
```

Out[552]=

	DF	SS	MS	F-Statistic	P-Value
x	1	203641.	203641.	2120.61	2.73086×10^{-19}
x ²	1	10634.9	10634.9	110.747	7.29019×10^{-9}
Error	17	1632.49	96.029		
Total	19	215908.			

```
Out[553]= 3.22771 + 0.480014 x + 0.0311325 x^2
```



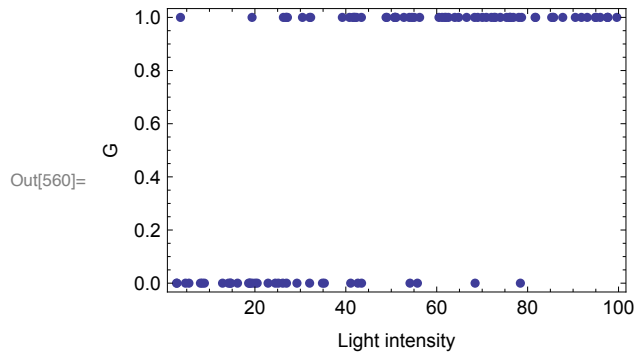
So, a linear model is one in which the response can be expressed as a linear combination of the predictors *or transformations thereof*. The practical outcome is that you can generate a new column in your predictor matrix, and treat it as a new, linear

predictor (as we did above). Another way to look at it is in a linear model, the parameters are simple multipliers of the predictors — never powers themselves. So, for example, $y = c + m x^2$ is a linear model (but curved), whereas $y = c + 2 x^m$ is non-linear.

A non-linear model — logistic regression

■ Some data

100 seeds, different light level for each seed. Does it germinate or not? (1 = yes, 0 = no.) Distribution is *Bernoulli* (Binomial with just one trial).



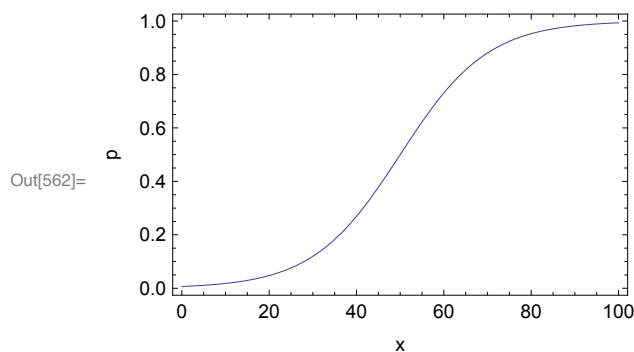
How can we model this? What is it that appears to be changing with light intensity?

■ The logit function and the logistic model

The logistic function has the following form

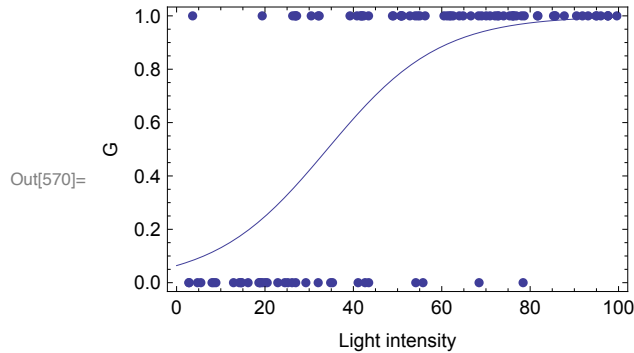
$$p = \frac{e^{c+mx}}{1 + e^{c+mx}} = \frac{1}{1 + e^{-(c+mx)}}$$

and here is what it looks like in general:



It is bounded between 0 and 1, and therefore good for modeling how *probability* changes with some predictor. It has two parameters, just like a linear regression. How can we fit it to the data? Linear, least squares methods do not apply, because a) the response is no longer Normally distributed, and b) the function cannot be expressed as a linear combination of any transforms of x . We therefore return to our more general method of maximizing likelihood by numerical search. The logistic function describes the p parameter of the Binomial distribution, the number of trials is 1, and our data are 1 (success) or 0 (failure).

Out[569]= { -38.6445, { b0 → -2.68076, b1 → 0.0786474 } }



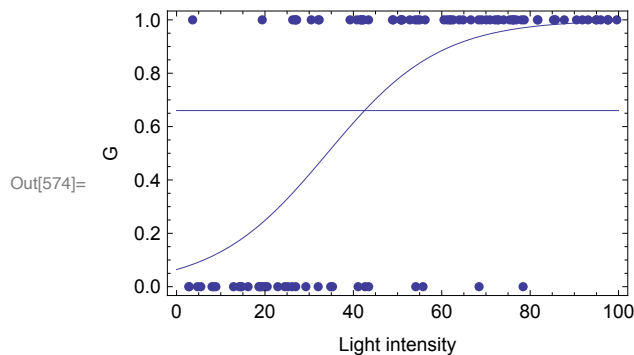
What about significance testing? An obvious candidate is to use a model-choice statistic like AIC. Here is AIC for the fitted model:

Out[571]= 81.2891

We also need AIC for the 'null' model, i.e., one with no slope term.

Out[573]= 130.207

Clearly the logistic model is much better (much lower AIC). Here are both models on one plot

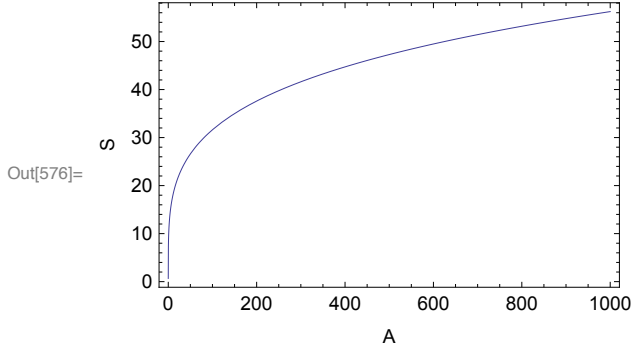


Making a non-linear model linear by transformation

Because linear models are easy to fit, investigators often try to make their data fit the assumptions of a linear model by transforming them. A classic example in ecology is data on the species-area relationship. Many studies have found that the following function describes the number of species on an island as a function of that island's area.

$$S = c A^z$$

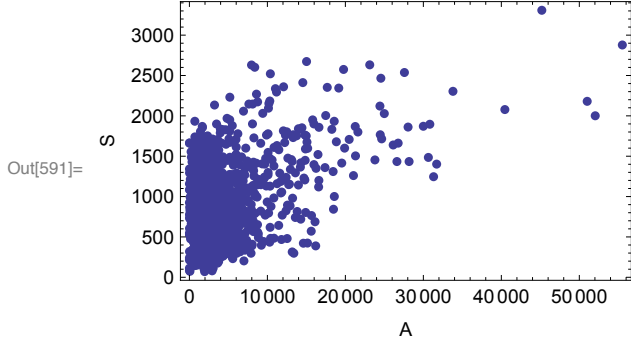
where c and z are constants. Furthermore, the value of the z parameter is often about 0.25, which makes the function look something like this



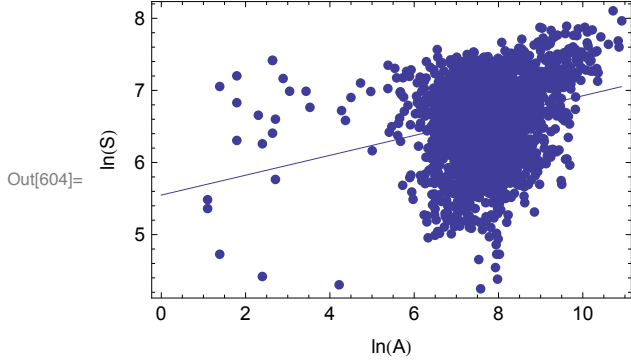
This is clearly a non-linear model, and it can't be treated as linear because one of the parameters (z) is a power of A . However, if we log-transform both sides of the model we get this

$$\log(S) = \log(c) + z \log(A)$$

which is a linear model. So if we transform our data (i.e., our species and area observations), we can fit a linear model and our fitted intercept will be an estimate of $\log(c)$ and our fitted slope will be an estimate of z . Here are some data on the area and number of native plant species in each county of the contiguous United States.



The log-transformed data, with a fitted linear function, look like this



The fitted linear model is

$$\text{Out}[615]= 5.54852 + 0.137731 \text{ area}$$

which corresponds to

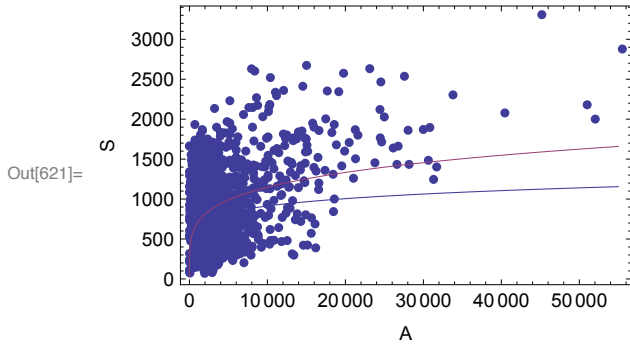
$$\text{Out}[620]= 256.858 \text{ area}^{0.137731}$$

(the estimate of c is $e^{5.54} = 256.858$). But is this equivalent to doing a non-linear fit on the untransformed data? Let's see:

```
In[596]:= nlm = Normal[NonlinearModelFit[allData[[All, {3, 5}]], c * area^z, {c, z}, area]
```

```
Out[596]= 157.132 area0.215904
```

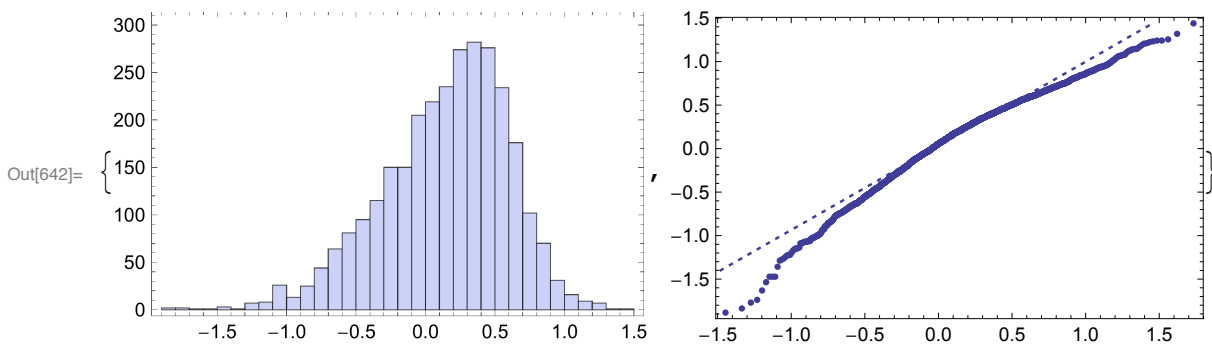
These parameter estimates seem very different. Let's plot both on the original data.



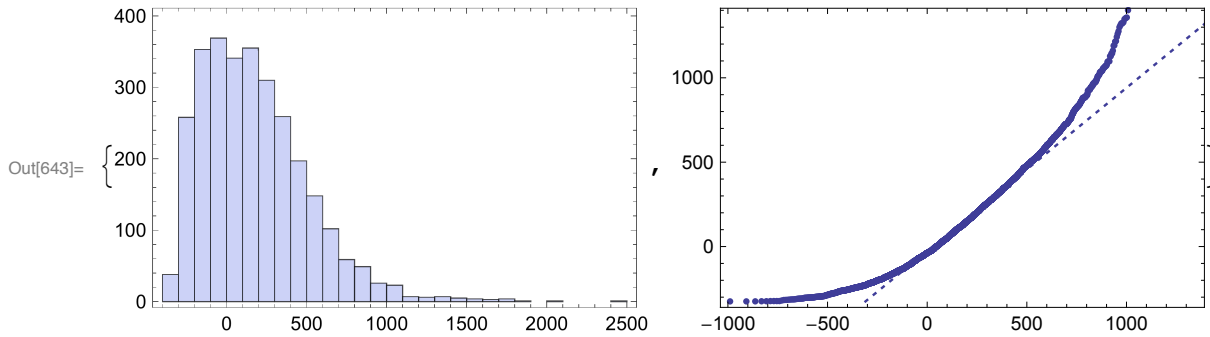
The blue line is the linear fit to the transformed data, and the purple line is the non-linear fit to the untransformed data. Neither seems to do a great job predicting the species richness of the largest counties, but the non-linear model does best.

Why *are* the parameter estimates different? The reason is that transformations don't affect just the shape of the function, but also the distribution of both predictor and response. In the case above, most counties have an area in the range 500–5000 ha, but there are a small number of counties with somewhat smaller area and a small number with much larger area (up to 55,000 ha). Raw areas cannot be smaller than 0, so the small areas can't be much smaller than the mean area, but the same is not true of the large areas. In the log-transformed data, however, the zero bound is removed and the small areas are stretched out, whereas the large areas are compressed. This means that the fitted model is influenced more by small areas, and less by large areas, in the log-transformed model than in the untransformed model.

So, to transform or not to transform? The availability of modern methods for fitting non-linear models removes one of the drivers behind transformations — we do *not* need to do it just to make it easier to fit. There are other reasons, though, generally to do with error distributions. A good model has homoscedastic errors (meaning that the errors are unbiased and of constant variance over the range of the predictors). In the case above, the errors are clearly heteroscedastic in both models in that the large areas typically have positive errors. Also, both fits assumed normally-distributed errors, so we can (and should) do a quantile-quantile plot of the residuals to check. For the log-transformed data:



And for the original data:



Clearly, log-transformation made the errors closer to our assumption of a Normal distribution. So out of these two models, it might be preferred. But of course, maybe the original data could be better modeled by something other than the Normal. Species richness measures are basically integer counts, so perhaps we should be using the Poisson, or a derivative of that.

Finally, data transforms are not so much of a problem if you are trying to reject a null hypothesis, but can cause problems in interpreting the alternative hypothesis that you have therefore accepted, because that hypothesis is properly only expressed in terms of the transformed variables. In the case of the log-transformed species-area fit, we should only say that ‘the log of the number of species increases with the log of the area of an island’ and that the slope of that increase is 0.138. But what are ‘log species’ in the real world? In practice, most investigators break this rule and say that the relationship is $S = cA^z$ with $z = 0.138$. But our non-linear fit shows us that isn’t exactly true...

■ Take-home message

Use a data transform mainly if it improves your error distribution *a lot*. Don’t use one just to make a non-linear shape linear, or to make errors normal — fit the non-linear shape and non-normal data instead. (See the next section.)

Generalized Linear Models

Previously, we made a distinction between linear models and non-linear models. However, that line has been blurred by a relatively new set of techniques that are collectively called **Generalized Linear Modelling**. These recognise that non-linear functions like the logistic can be transformed into linear functions. The function that does the transformation is called the **link function**. For the logistic function it is $\frac{\ln(p)}{\ln(1-p)}$, which is called the *logit* function. In other words, if we take the logistic function

$$p = \frac{e^{c+mx}}{1 + e^{c+mx}}$$

and apply the transformation to both sides, we get

$$\text{logit}(p) = \frac{\ln(p)}{\ln(1-p)} = c + mx$$

So the GLM framework breaks up functions into their linear component and link function. (If they are linear anyway, then the link function is called the ‘identity’ function — a function that doesn’t do anything.)

The other thing that is different about a model like the logistic is the **distribution function**, which is Binomial instead of Normal. The GLM framework can be applied to any model in which the distribution function comes from the Exponential *family*. Luckily, this is almost every common function, including Normal, Poisson, Exponential (obviously) and Binomial. The reason for the constraint is that for these distributions, the best-fitting model can be found not by direct numerical maximization of the likelihood (as we did for the logistic regression above) but by a much more efficient procedure called *iteratively reweighted least squares*. I have told you before that least squares estimation applies only to Normal errors, but in fact that is true only for

unweighted least squares. Weighted least squares allows for finding the maximum likelihood estimators for non-Normal distribution functions (as long as they are from the Exponential family). The reason this is efficient is that although it is a numerical procedure, at each step candidate parameters are found by the solving the linear model (with weights), and then updating the *weights* (rather than the parameters themselves). The end result is that the best-fit model can be found in many fewer steps than simply searching across all parameters, especially for models with many predictors. This is what makes generalized linear models so powerful.

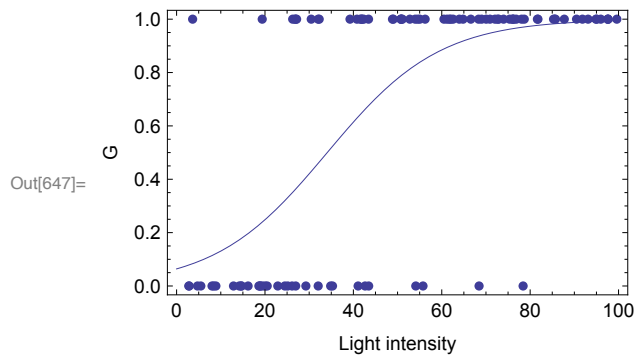
Let's see it in action!

```
In[644]:= glm = GeneralizedLinearModelFit[
      germinationData, m, m, ExponentialFamily -> "Binomial"];
```

```
In[645]:= Normal[glm]
```

$$\text{Out[645]= } \frac{1}{1 + e^{2.6813 - 0.078662 m}}$$

See how these parameters are the same as our MLE estimators in the previous section?



This framework allows us to calculate AIC as before, but also things like *p* values for the individual parameters.

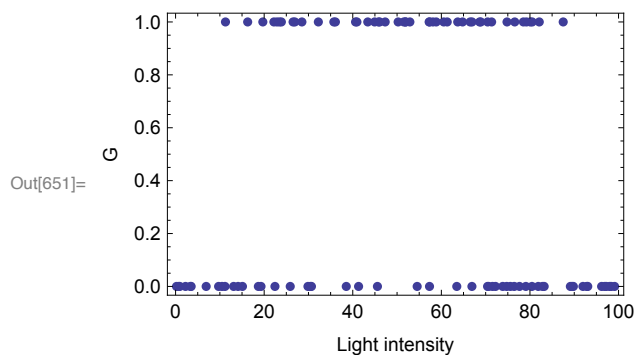
```
In[648]:= glm["AIC"]
      glm["ParameterPValues"]
```

```
Out[648]= 81.2891
```

```
Out[649]= {0.0000313032, 3.83921 × 10-7}
```

Here is an example of a more complex model: Binomial data with a quadratic term in the linear component

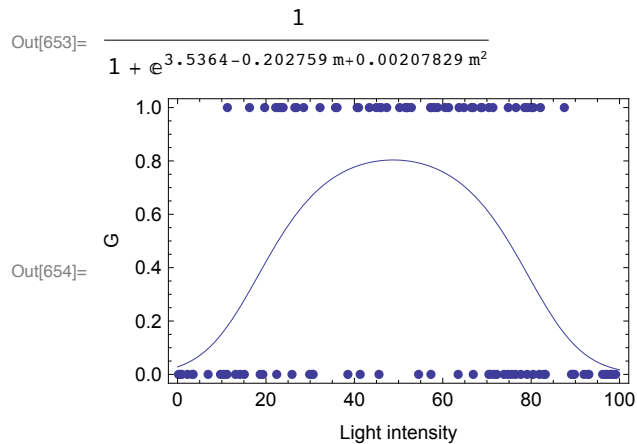
$$p = \frac{e^{c+mx+nx^2}}{1 + e^{c+mx+nx^2}} = \frac{1}{1 + e^{-(c+mx+nx^2)}}$$



It is easily fit.

```
In[652]:= glm2 = GeneralizedLinearModelFit[
  germinationData2, {m, m^2}, m, ExponentialFamily -> "Binomial"];
```

```
In[653]:= Normal[glm2]
```



```
In[655]:= glm2["AIC"]
glm2["ParameterPValues"]
```

Out[655]= 110.742

Out[656]= {0.000410816, 0.000013949, 7.62862×10^{-6} }

A note about notation

Sometimes the acronym GLM refers to general linear modelling, and sometimes to generalized linear modelling. You'll usually know the difference based on whether the response variable is Normal or not. If not, it's *generalized* linear modelling.