These are the log lifetime data from Table 3.1. I’ve typed them in by group, that is, group 1 are together, group 2 are together, etc, but that is not necessary. They could be in any order at all.

Descriptive statistics are almost always a good place to start. Since we use all the data as an argument, we are getting summary statistics for all the data lumped together.

Multiple modes in the stem and leaf plot?

1 | 1 represents 0.11  Leaf digit unit = 0.01
Cmd> `stemleaf(llifetime,5)`

There is an optional second argument to `stemleaf()` that allows you to specify the number of stems to use if you don’t like the default choice. See `help(stemleaf)`.

```
 1 +0.|8
 (19) 1*|00001122222233334
17 1. |5556666777889999
 2 2*|00
```

1* | 1 represents 1.1 Leaf digit unit = 0.1

Cmd> `temp<-factor(`
   `1,1,1,1,1,1,1,1,`
   `2,2,2,2,2,2,2,2,`
   `3,3,3,3,3,3,3,3,`
   `4,4,4,4,4,4,4,4,`
   `5,5,5,5,5,5)``

The function `factor()` creates factors, which are vectors of numbers that indicate grouping, rather than quantity. Here, we use values 1, 2, ..., 5 that indicate which temperature treatment group each of our responses is in. These 1, 2, etc match up with the data in `llifetime`. What the `factor()` function does is set an internal flag that tells MacAnova that in this variable, numbers indicate groups, not quantities.

The input to `factor()` should be positive integers. However, be warned that if your integers are 3, 4, and 5, then MacAnova will think that you have 5 groups instead of 3. This affects some computations later (and is much more serious if your integers are 300, 400, and 500!). If you have “nonstandard” levels to make into a factor, try using `makefactor()` instead. See `help(makefactor)`.

Cmd> `temp<-factor(`
   `rep(run(5),`
   `vector(8,8,8,7,6)))``

Here is an alternate way to get the same variable `temp`. The function `run(5)` gives you the numbers 1 through 5. `rep()` repeats its first argument. `rep(x,5)` repeats x five times. In our case, the second argument to `rep()` is the same length as the first argument and gives repetition counts for each of the elements of the first argument. Thus 1 gets repeated 8 times, and 5 gets repeated 6 times. See `help(rep)` and `help(run)`.

Cmd> `tabs(llifetime,temp)`

`tabs()` computes mean, variance, and count of one variable by the levels of another variable. This allows us to get statistics separately for each treatment group. If we wanted just means or just variances, we could do `tabs(llifetime,temp,mean:T)` or `tabs(...,var:T)`. `tabs()` can also compute minima, maxima, and standard deviations.

```
component: mean
(1) 1.932 1.629 1.377 1.194
(5) 1.057

component: var
(1) 0.004021 0.01098 0.01148 0.002095
(5) 0.01915

component: count
(1) 8 8 8 8 7
(5) 6
```
This illustrates subscripting. Here we can get all the data that have temp equal to 1.

```
Cmd> llifetime[temp==1]
```

<table>
<thead>
<tr>
<th></th>
<th>2.04</th>
<th>1.91</th>
<th>2</th>
<th>1.92</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>1.85</td>
<td>1.96</td>
<td>1.88</td>
<td>1.9</td>
</tr>
</tbody>
</table>

```
Cmd> boxplot(llifetime[temp==1], llifetime[temp==2],
llifetime[temp==3], llifetime[temp==4],
llifetime[temp==5], vertical:T)
```

The function `boxplot()` makes boxplots. Here we will get a separate box for each input argument to `boxplot`. In this case, we are asking for a separate box for the five levels of temp. The `vertical:T` makes the boxes up and down, rather than side to side.

```
Cmd> split(llifetime, temp)
```

The function `split()` takes its first argument and splits it up into groups according to the levels of the second argument. Since temp has values 1 through 5, we get as value a structure with five components.

```
component: temp1
(1) 2.04 1.91 2 1.92
(5) 1.85 1.96 1.88 1.9

component: temp2
(1) 1.66 1.71 1.42 1.76
(5) 1.66 1.61 1.55 1.66

component: temp3
(1) 1.53 1.54 1.38 1.31
(5) 1.35 1.27 1.26 1.38

component: temp4
(1) 1.15 1.22 1.17 1.16
(5) 1.21 1.28 1.17

component: temp5
(1) 1.26 0.83 1.08 1.02
(5) 1.09 1.06
Several MacAnova functions can take structures as arguments. For boxplot(), we get boxes for each of the components of the structure used in the argument. We could also do describe(split(llifetime,temp)) to get descriptive statistics separately for each level of temp.

![Box Plot]

This is the basic Analysis of Variance command. The argument to anova() is a character string giving a model to be fit. The response variable goes on the left hand side of the equals sign, and the explanatory variable(s) goes on the right. The default output is an ANOVA table giving source, DF, SS, and MS. Because we made temp a factor, anova() knows to treat temp as a grouping variable. If temp were not a factor, anova() would treat temp like a quantitative variable with numbers 1-5; that would essentially be a linear regression, and it would be inappropriate here.

Model used is llifetime=temp
WARNING: summaries are sequential

<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
</tr>
<tr>
<td>temp</td>
<td>4</td>
<td>3.538</td>
<td>0.8844</td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
</tr>
</tbody>
</table>

The anova() command has several options. One of them is fstats:T, which makes anova() print F-statistics and p-values. See help(anova). The treatment effect is extremely significant in these data, but that was fairly obvious from the boxplots.

Model used is llifetime=temp
WARNING: summaries are sequential

<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
<td>8653.9536</td>
<td>0</td>
</tr>
<tr>
<td>temp</td>
<td>4</td>
<td>3.538</td>
<td>0.8844</td>
<td>96.3630</td>
<td>0</td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cmd> 1-cumF(96.36,4,32)
   If you want to compute the p-value “by hand”, you can use cumF(). cumF() takes three arguments, the F value, numerator df, and denominator df. It gives the cumulative area under the F curve up to the F value. To get the p-value (area to the right), we need to take 1-cumF().
   (1) 0

Cmd> invF(.9999,4,32)
   Just for informational purposes, we compute the critical value for the .0001 significance level using invF().
   (1) 8.339

Cmd> coefs()
   After we have done an anova(), we can recover the estimated parameters (the \( \hat{\beta} \) and \( \hat{\alpha}_i \)s) with the coefs() command. The results are returned in a structure with a component for the CONSTANT (\( \hat{\beta} \)) and a component for temp (\( \hat{\alpha}_i \)s). MacAnova uses the restriction that the sum of the \( \hat{\alpha}_i \)s is zero, not the weighted sum. Thus you cannot use these effects in the “hand” formulae for sums of squares unless all the sample sizes are equal.

   component: CONSTANT
   (1) 1.438

   component: temp
   (1) 0.4946 0.1908 -0.06044 -0.2437
   (5) -0.3813

Cmd> coefs("temp")
   If we just want the coefficients for temp, we can get them by using "temp" as an argument to coefs().
   (1) 0.4946 0.1908 -0.06044 -0.2437
   (5) -0.3813

Cmd> coefs()$temp
   Or we can just select the component named temp from the structure that coefs() returns.
   (1) 0.4946 0.1908 -0.06044 -0.2437
   (5) -0.3813

Cmd> coefs(2)
   Or we can select coefficients for the second term in the model.
   (1) 0.4946 0.1908 -0.06044 -0.2437
   (5) -0.3813
We can also get standard errors for the estimated coefficients. An equivalent form is `coefs("temp", se:T)`. Roughly speaking, these standard errors can be used to test if a particular group has an average response equal to the overall average. They cannot be used directly to compare two groups, because the estimated coefficients are not independent of each other (and you therefore must deal with their correlation). Note that the standard errors are not all the same (in this case, they depend on sample sizes).

```
Cmd> secoefs("temp")
```

```
component: coefs
(1) 0.4946 0.1908 -0.06044 -0.2437
(5) -0.3813
```

```
component: se
(1) 0.03065 0.03065 0.03065 0.03222
(5) 0.03419
```

```
Cmd> predtable()
predtable() gives you the table of predicted values for the full model. For this model, it is the group means.

```
(1) 1.933 1.629 1.378 1.194
(5) 1.057
```

```
Cmd> coefs("CONSTANT")+coefs("temp")
Here is another way to get the predicted values or group means, as \( \hat{\mu} + \hat{\alpha}_i \).

```
(1) 1.933 1.629 1.378 1.194
(5) 1.057
```

```
Cmd> contrast("temp", vector(1,-1,0,0,0))
The `contrast()` function computes contrasts. Its first argument is the name of the term for which you want to make a contrast. We can also use the number of the term in place of the name (here, 2 instead of "temp"). The second argument is the set of coefficients for the contrast. There must be one coefficient for every level of the term and the coefficients must add to 0. The value of `contrast()` is a structure giving the estimated contrast, the sum of squares for the contrast, and the standard error for the contrast.

In this example, we are comparing the first two levels of `temp`. The estimate 0.3038 is .4946 - 1.908, the difference of the first two treatment effects as found in `coefs()` above. The se for the difference is .0479; note that this is not equal to \( \sqrt{.03065^2 + .03065^2} = .0433 \) (se’s from `secoefs()` combined as if the estimated effects were independent).

```
component: estimate
(1) 0.3038
```

```
component: ss
(1) 0.3691
```

```
component: se
(1) 0.04790
```
Here is a contrast to compare the average of levels 2 and 4 with the average of levels 1, 3, and 5.

```
Cmd> contrast(\"temp\", vector(1/3,-1/2,1/3,-1/2,1/3))
```

component: estimate
(1) 0.04404

component: ss
(1) 0.01712

component: se
(1) 0.03224

This contrast also compares levels 2 and 4 with levels 1, 3, and 5, though the coefficients are -3 times those of the preceding contrast. The estimate changes by a factor of -3, the se changes by a factor of 3, and the ss stays the same.

```
Cmd> contrast(\"temp\", vector(-1,1.5,-1,1.5,-1))
```

component: estimate
(1) -0.1321

component: ss
(1) 0.01712

component: se
(1) 0.09672
The macro `pairwise` does pairwise comparisons. The arguments to `pairwise` are the factor in which the comparisons are made (here `temp`), the level at which the comparisons are made (here `.05`), and an indication of the method to use (here, `hsd`). For these data, groups 4 and 5 are not significantly different, but all other differences are significant.

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-0.381</td>
</tr>
<tr>
<td>4</td>
<td>-0.244</td>
</tr>
<tr>
<td>3</td>
<td>-0.0604</td>
</tr>
<tr>
<td>2</td>
<td>0.191</td>
</tr>
<tr>
<td>1</td>
<td>0.495</td>
</tr>
</tbody>
</table>
```

We get the same conclusions at the .01 level.

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-0.381</td>
</tr>
<tr>
<td>4</td>
<td>-0.244</td>
</tr>
<tr>
<td>3</td>
<td>-0.0604</td>
</tr>
<tr>
<td>2</td>
<td>0.191</td>
</tr>
<tr>
<td>1</td>
<td>0.495</td>
</tr>
</tbody>
</table>
```

At the .001 level, we also see that groups 4 and 3 are not significantly different.

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-0.381</td>
</tr>
<tr>
<td>4</td>
<td>-0.244</td>
</tr>
<tr>
<td>3</td>
<td>-0.0604</td>
</tr>
<tr>
<td>2</td>
<td>0.191</td>
</tr>
<tr>
<td>1</td>
<td>0.495</td>
</tr>
</tbody>
</table>
```

For this setup, `bsd` is giving us the same results as `hsd`.

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-0.381</td>
</tr>
<tr>
<td>4</td>
<td>-0.244</td>
</tr>
<tr>
<td>3</td>
<td>-0.0604</td>
</tr>
<tr>
<td>2</td>
<td>0.191</td>
</tr>
<tr>
<td>1</td>
<td>0.495</td>
</tr>
</tbody>
</table>
```

`LSD` finds more differences than `hsd`, as might be expected.
The groups in temp actually represent quantitative temperature levels in degrees. Let’s make a vector containing the actual temperatures. Since temp contains the group numbers, we only need to make a scratch vector containing the temperature for each group, and then subscript it with temp to get a vector of the desired length with the correct temperature for each group. This is a trick that can save you a lot of time.

Here it is.

<table>
<thead>
<tr>
<th></th>
<th>temper</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>175</td>
</tr>
<tr>
<td>(2)</td>
<td>175</td>
</tr>
<tr>
<td>(3)</td>
<td>175</td>
</tr>
<tr>
<td>(4)</td>
<td>175</td>
</tr>
<tr>
<td>(5)</td>
<td>175</td>
</tr>
<tr>
<td>(6)</td>
<td>175</td>
</tr>
<tr>
<td>(7)</td>
<td>175</td>
</tr>
<tr>
<td>(8)</td>
<td>175</td>
</tr>
<tr>
<td>(9)</td>
<td>194</td>
</tr>
<tr>
<td>(10)</td>
<td>194</td>
</tr>
<tr>
<td>(11)</td>
<td>194</td>
</tr>
<tr>
<td>(12)</td>
<td>194</td>
</tr>
<tr>
<td>(13)</td>
<td>194</td>
</tr>
<tr>
<td>(14)</td>
<td>194</td>
</tr>
<tr>
<td>(15)</td>
<td>213</td>
</tr>
<tr>
<td>(16)</td>
<td>213</td>
</tr>
<tr>
<td>(17)</td>
<td>213</td>
</tr>
<tr>
<td>(18)</td>
<td>213</td>
</tr>
<tr>
<td>(19)</td>
<td>213</td>
</tr>
<tr>
<td>(20)</td>
<td>213</td>
</tr>
<tr>
<td>(21)</td>
<td>231</td>
</tr>
<tr>
<td>(22)</td>
<td>231</td>
</tr>
<tr>
<td>(23)</td>
<td>231</td>
</tr>
<tr>
<td>(24)</td>
<td>231</td>
</tr>
<tr>
<td>(25)</td>
<td>231</td>
</tr>
<tr>
<td>(26)</td>
<td>231</td>
</tr>
<tr>
<td>(27)</td>
<td>231</td>
</tr>
<tr>
<td>(28)</td>
<td>231</td>
</tr>
<tr>
<td>(29)</td>
<td>231</td>
</tr>
<tr>
<td>(30)</td>
<td>231</td>
</tr>
<tr>
<td>(31)</td>
<td>231</td>
</tr>
<tr>
<td>(32)</td>
<td>231</td>
</tr>
<tr>
<td>(33)</td>
<td>250</td>
</tr>
<tr>
<td>(34)</td>
<td>250</td>
</tr>
<tr>
<td>(35)</td>
<td>250</td>
</tr>
<tr>
<td>(36)</td>
<td>250</td>
</tr>
<tr>
<td>(37)</td>
<td>250</td>
</tr>
</tbody>
</table>

We make variables temper2, temper3, and temper4 containing the squares, cubes, and fourth powers of temper. This is in preparation for fitting a polynomial model.

Fit a model using temperature, its square, cube, and fourth power. We do this by putting these four variables on the right hand side of the equals sign with plus signs between. The SS for temper is the linear sum of squares for temperature. The SS for temper2 is the SS for quadratic adjusted for linear. Similarly the SS for temper3 is the SS for cubic in temperature adjusted for linear and quadratic, and so on. Note that the error DF and SS are the same as for the model using the 5 level factor temp. Also, if you add up the SS for the four temper variables you get the SS (with 4 df) for the temp variable.

Further note that neither temper3 nor temper4 seems to be significant.
Look at the coefficients. Since none of the terms in the model is a factor, this model is essentially a multiple regression and these are the regression coefficients.

```
component: CONSTANT
(1) 0.96995
component: temper
(1) 0.075733
component: temper2
(1) -0.00076488
component: temper3
(1) 2.6003e-06
component: temper4
(1) -2.9879e-09
```

Here is an alternative way to do the same ANOVA. You can specify transformations in the model by enclosing the term in curly brackets. This may be simpler for a single model, but will be more typing if we do several models using the same transformations.

```
Model used is llifetime=temper+{temper^2}+{temper^3}+
{temper^4}

WARNING: summaries are sequential

<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
</tr>
<tr>
<td>temper</td>
<td>1</td>
<td>3.459</td>
<td>3.459</td>
</tr>
<tr>
<td>{temper^2}</td>
<td>1</td>
<td>0.07834</td>
<td>0.07834</td>
</tr>
<tr>
<td>{temper^3}</td>
<td>1</td>
<td>1.857e-05</td>
<td>1.857e-05</td>
</tr>
<tr>
<td>{temper^4}</td>
<td>1</td>
<td>8.257e-06</td>
<td>8.257e-06</td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
</tr>
</tbody>
</table>
```

Here is yet another way to do it. PK(variable) says to make polynomial terms in variable up to order K. Here we get linear, quadratic, cubic, and quartic terms in temper.

```
Model used is llifetime=P4(temper)

WARNING: summaries are sequential

<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
</tr>
<tr>
<td>(temper)</td>
<td>1</td>
<td>3.459</td>
<td>3.459</td>
</tr>
<tr>
<td>(temper^2)</td>
<td>1</td>
<td>0.07834</td>
<td>0.07834</td>
</tr>
<tr>
<td>(temper^3)</td>
<td>1</td>
<td>1.857e-05</td>
<td>1.857e-05</td>
</tr>
<tr>
<td>(temper^4)</td>
<td>1</td>
<td>8.257e-06</td>
<td>8.257e-06</td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
</tr>
</tbody>
</table>
Cmd> `anova("llifetime=temper+temper2", fstats=T)`

Refit using just the linear and quadratic terms. Note that when you drop a couple of terms, their SS and DF get pooled into error.

Model used is llifetime=temper+temper2
WARNING: summaries are sequential

```
<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
<td>9193.9859</td>
<td>0</td>
</tr>
<tr>
<td>temper</td>
<td>1</td>
<td>3.459</td>
<td>3.459</td>
<td>400.4333</td>
<td>0</td>
</tr>
<tr>
<td>temper2</td>
<td>1</td>
<td>0.07834</td>
<td>0.07834</td>
<td>9.0688</td>
<td>0.0049</td>
</tr>
<tr>
<td>ERROR1</td>
<td>34</td>
<td>0.2937</td>
<td>0.008639</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Cmd> `coefs()`

When you drop these higher order terms, the coefficients for the other terms in the model will usually change. They did here. A lot.

```
component: CONSTANT
(1) 7.418
component: temper
(1) -0.045098
component: temper2
(1) 7.8604e-05
```

Cmd> `anova("llifetime=temper+temper2+temper3+temper4+temp")`

What happens if we combine both the factor temp and the quantitative variables? There are only 4 degrees of freedom between the 5 groups, so there is no way to estimate all 8 df this would entail; somebody comes home empty handed.

In this model, we first entered all four quantitative variables, and then the factor temp. There are no df left for temp, so it gets 0 df. Getting 0 df essentially means that any variability it could explain has been explained by earlier terms.

Model used is llifetime=temper+temper2+temper3+temper4+temp
WARNING: summaries are sequential

```
<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
</tr>
<tr>
<td>temper</td>
<td>1</td>
<td>3.459</td>
<td>3.459</td>
</tr>
<tr>
<td>temper2</td>
<td>1</td>
<td>0.07834</td>
<td>0.07834</td>
</tr>
<tr>
<td>temper3</td>
<td>1</td>
<td>1.857e-05</td>
<td>1.857e-05</td>
</tr>
<tr>
<td>temper4</td>
<td>1</td>
<td>8.257e-06</td>
<td>8.257e-06</td>
</tr>
<tr>
<td>temp</td>
<td>0</td>
<td>0 undefined</td>
<td></td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
</tr>
</tbody>
</table>
```

Cmd> `anova("llifetime=temp+temper+temper2+temper3+temper4")`

Here we put the factor in first and then the quantitative terms. The factor swallows all the df, leaving nothing for the quantitative variables.

Model used is llifetime=temp+temper+temper2+temper3+temper4
WARNING: summaries are sequential

```
<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
</tr>
<tr>
<td>temp</td>
<td>4</td>
<td>3.538</td>
<td>0.8844</td>
</tr>
<tr>
<td>temper</td>
<td>0</td>
<td>0 undefined</td>
<td></td>
</tr>
<tr>
<td>temper2</td>
<td>0</td>
<td>0 undefined</td>
<td></td>
</tr>
<tr>
<td>temper3</td>
<td>0</td>
<td>0 undefined</td>
<td></td>
</tr>
<tr>
<td>temper4</td>
<td>0</td>
<td>0 undefined</td>
<td></td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
</tr>
</tbody>
</table>
```
Here we only put in the first two quantitative variables, and then the factor temp. There are 2 df left for temp, so it swallows them. Note that the SS for temp equals the sum of those for temper3 and temper4. This is sometimes a useful approach when you have a couple of interesting explanatory terms, and then you just let the factor explain everything else.

```
Cmd> anova("llifetime=temper+temper2+temp")

Model used is llifetime=temper+temper2+temp
WARNING: summaries are sequential

<table>
<thead>
<tr>
<th></th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT</td>
<td>1</td>
<td>79.42</td>
<td>79.42</td>
</tr>
<tr>
<td>temper</td>
<td>1</td>
<td>3.459</td>
<td>3.459</td>
</tr>
<tr>
<td>temper2</td>
<td>1</td>
<td>0.07834</td>
<td>0.07834</td>
</tr>
<tr>
<td>temp</td>
<td>2</td>
<td>2.683e-05</td>
<td>1.341e-05</td>
</tr>
<tr>
<td>ERROR1</td>
<td>32</td>
<td>0.2937</td>
<td>0.009178</td>
</tr>
</tbody>
</table>
```

```
Cmd> coefs()

Here are the coefficients. When there are missing df in a factor (here temp only has 2 df instead of its natural 4), there is considerable arbitrariness in the coefficients. MacAnova has one algorithm for choosing coefficients, other packages have other algorithms (and there are infinitely many possibilities). The bottom line is that when there are missing df in factors, the coefficients are difficult to interpret and frequently bizarre.

WARNING: Missing df(s) in term temp
Missing effects set to zero
component: CONSTANT
  (1) 7.47
component: temper
  (1) -0.04559
component: temper2
  (1) 7.975e-05
component: temp
  (1) -0.001499 0.001791 0 0
  (5) -0.0002922
It is possible to compute orthogonal contrasts in the 5 levels of `temp` that give the linear, quadratic adjusted for linear, etc. sums of squares. When the quantitative levels are equally spaced and the group sizes are the same, the coefficients for these contrasts are rather simple and are tabulated in most design texts. The coefficients are incredibly messy otherwise, and it is almost always easier to get the polynomial SS by multiple regression, but the contrast coefficients do exist. Being the compulsive sort that I am, I’ll show you how to get them. Why this works is explained in Appendix A.

Begin by modelling `temper` (the linear term) with a model that just includes a constant. In MacAnova, this is done by using a 1 as the explanatory term on the right.

```
Cmd> anova("temper=1",silent:T)
```

```
It is possible to compute orthogonal contrasts in the 5 levels of `temp` that give the linear, quadratic adjusted for linear, etc. sums of squares. When the quantitative levels are equally spaced and the group sizes are the same, the coefficients for these contrasts are rather simple and are tabulated in most design texts. The coefficients are incredibly messy otherwise, and it is almost always easier to get the polynomial SS by multiple regression, but the contrast coefficients do exist. Being the compulsive sort that I am, I’ll show you how to get them. Why this works is explained in Appendix A.

Begin by modelling `temper` (the linear term) with a model that just includes a constant. This just fits the grand mean. In MacAnova, this is done by using a 1 as the explanatory term on the right.
```

```
Now we tabulate the residuals of fitting the linear term by the grand mean, getting group means and counts. We then multiply the group mean residual by the group count, getting the group total which we put in `c1`. `c1` is our linear contrast for the effect of temperature. Note that its values sum to zero, as is required for a contrast.
```

```
(1) -0.004668 0.0115 -0.006346 -0.003115
(5) 0.002628
```

```
Cmd> anova("temper2=temper",silent:T)
```

```
Now we want to get the contrast for quadratic adjusted for linear. Start by fitting the squared temperatures using a model including lower power terms, here the linear.
```

```
Cmd> ct2<-tabs(RESIDUALS,temp);
c2<-ct2$mean*ct2$count;c2
Now we repeat with these model residuals what we did for the linear residuals. Get group means and counts, and the take their product to get group total residuals, saving the result in `c2`. This is our quadratic adjusted for linear contrast.
```

```
For cubic, you would model `temper3` using `temper` plus `temper2`, and then get group total residuals, and so on.
```

```
(1) -0.004668 0.0115 -0.006346 -0.003115
(5) 0.002628
```

```
Cmd> anova("llifetime=temp",silent:T)
```

```
Reset the anova model so that we can do contrasts.
```

```
Cmd> contrast("temp",c1)$SS
Try out our set of contrast coefficients for linear in temperature. We get the same SS as for linear above.
```

```
(1) 3.4593
```

```
Cmd> contrast("temp",c2)$SS
Try out our quadratic adjusted for linear contrast. Again, we get the same SS as above.
```

```
(1) 0.078343
```

```
Cmd> sum(c1*c2/ct2$count)
Verify that these two contrasts are orthogonal. Their weighted sum of cross products is zero (up to round off error).
```

```
(1) -4.3656e-10
```