ST2304 Exercises Week 7: Multiple Regression

Bob O'Hara

19 February 2018

Problem 1: Fitting categorical variables in R

We can use the cake example to fit categorical variables with R. First, the data is in the lme4 package (so there should be no problems getting it into R!):

```
library(lme4) # use install.packages("lme4") if this doesn't work
```

```
## Loading required package: Matrix
```

```
data("cake")
```

The details of the experiment are here: http://bit.ly/2GNb4dA. Briefly, cakes were baked with different recipes and at different temperatures, and then tests for how well they held togetherby holding one end and lifting up the other until the cake broke. The angle at which the cake broke is the response. I assume that a lower angle is better (just think about a cake that can be bent by 63°). For practical reasons, cakes were baked on different days: these are referred to as replicates.

We will ignore the date effects for now, and look at the possible effects of temperature and recipe. There are three recipes: A is a normal one, B has warmed chocolate added, and C has more sugar. There are also 6 baking temeratures, from 175°C to 225°C. Here we will also treat these as categorical variables.

First, we can fit a model for the temperature, ignoring recipe. This is called a "one way model" because there is only one factor (a model with recipe and temperature is a "two-way model"). We will treat Temperature as a categorical variable, also known as a factor. The `temp` variable is continuous, so if we do this:

```
mod.wrong <- lm(angle~temp, data=cake)
coef(mod.wrong)
```

```
## (Intercept)        temp
##   0.5158730   0.1580317
```

we see we only get a single coefficient, i.e. the slope of a regression model. We need to make temp a factor, which can be done using the factor() function:

```
cake$tempF <-factor(cake$temp)
str(cake$tempF)
```

```
##  Factor w/ 6 levels "175","185","195",..: 1 2 3 4 5 6 1 2 3 4 ...
```

This creates a variable called tempF that is a factor. Internally, R codes it as integers (1 to the number of levels), and assigns each number to a level of the factor, so "175" is 1, "185" is 2 etc. (there is no guarantee that this order will be the same on another computer, but that is usually not a problem). By default, R will make the first value in tempF the first level of the factor, the next different value the second etc. We can see the levels:

```
levels(cake$tempF)
```

```
## [1] "175" "185" "195" "205" "215" "225"
```

And if we want to create our own factors, we can do that:

```
A_vector <- c(11:15, 11:15)
A_Factor <- factor(A_vector, levels=15:11)
A_Factor
```

```
##  [1] 11 12 13 14 15 11 12 13 14 15
## Levels: 15 14 13 12 11
```

(I put the levels in a different order just to be different)

When R uses tempF in a linear model, it converts the factor into a set of contrasts (i.e. it codes the factor levels as columns of 0s and 1s). We can see the contrasts that are used in the design matrix here:

```
contrasts(cake$tempF)
```

```
    185 195 205 215 225
175   0   0   0   0   0
185   1   0   0   0   0
195   0   1   0   0   0
205   0   0   1   0   0
215   0   0   0   1   0
225   0   0   0   0   1
```

175 is the baseline level (i.e. the intercept) and the other levels are contrasts, so the 185°C effect is difference between the predicted angle from a cake baked at 185°C and one baked at 175°C) to this. Most of the time we don't need to be aware of this: it is all internal to R.

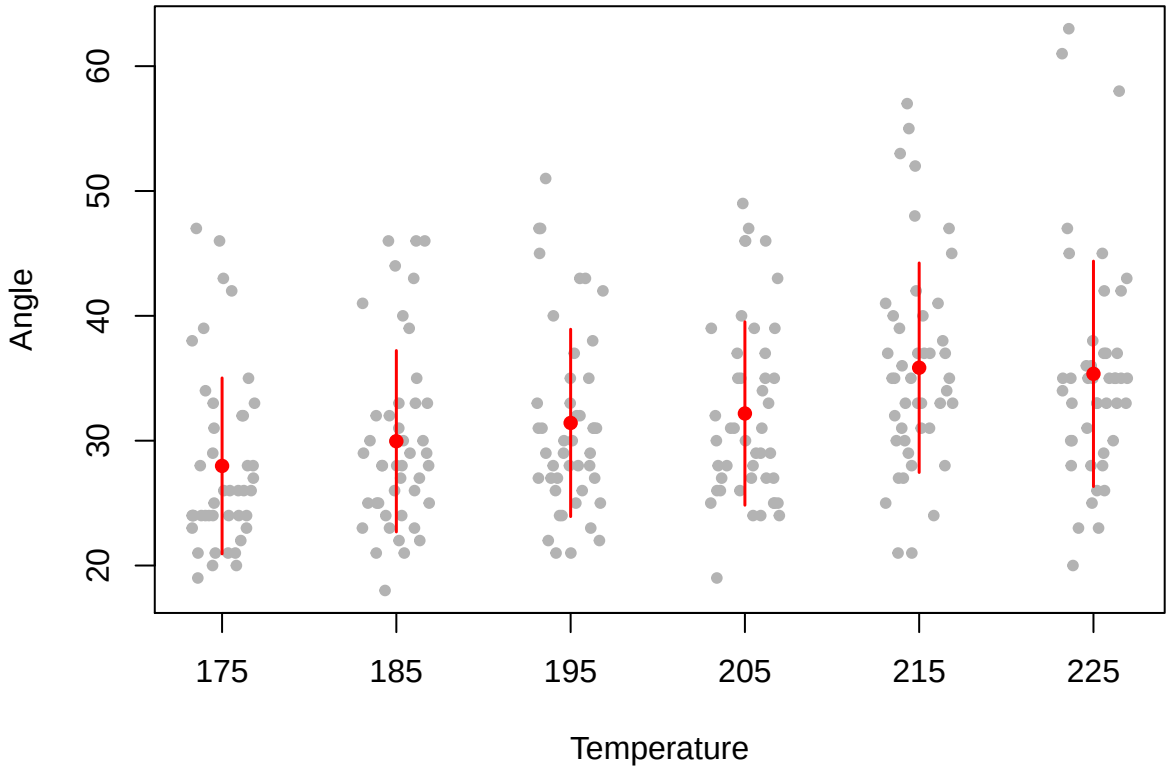
Now we can fit the model. The model formula looks the same as it did with regression: the difference is that tempF has been defined to be a factor, so R know it has to create the contrasts to put into the design matrix. This means we don't have to worry about the details of this, as long as we understand the output:

```
model.temp <- lm(angle~tempF, data=cake)
summary(model.temp)
```

```
Call:
lm(formula = angle ~ tempF, data = cake)

Residuals:
    Min      1Q  Median      3Q     Max
-15.356  -5.128  -1.633   3.044  27.644

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   27.978      1.162  24.076  < 2e-16 ***
tempF185       1.978      1.643   1.203   0.2299
tempF195       3.444      1.643   2.096   0.0370 *
tempF205       4.200      1.643   2.556   0.0112 *
tempF215       7.867      1.643   4.787 2.83e-06 ***
tempF225       7.378      1.643   4.489 1.07e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.795 on 264 degrees of freedom
Multiple R-squared:  0.1158,    Adjusted R-squared:  0.09902
F-statistic: 6.913 on 5 and 264 DF,  p-value: 4.391e-06
```

We can see that the (multiple) R^2 is 12%, which is low. Plotting the results of the model is awkward. We can plot the data reasonably well (a problem for you: work out what jitter() does, and why it is useful):

```
Mean <- tapply(cake$angle, cake$temperature, mean)
SD <-  tapply(cake$angle, cake$temperature, sd)
plot(jitter(cake$temp), cake$angle, col="grey70", pch=20,
     xaxt="n", xlab="Temperature", ylab="Angle")
axis(1, at=as.numeric(names(Mean)))
points(as.numeric(names(Mean)), Mean, col=2, pch=16)
segments(as.numeric(names(Mean)), Mean+SD,
         as.numeric(names(Mean)), Mean-SD, col=2, lwd=1.5)
```



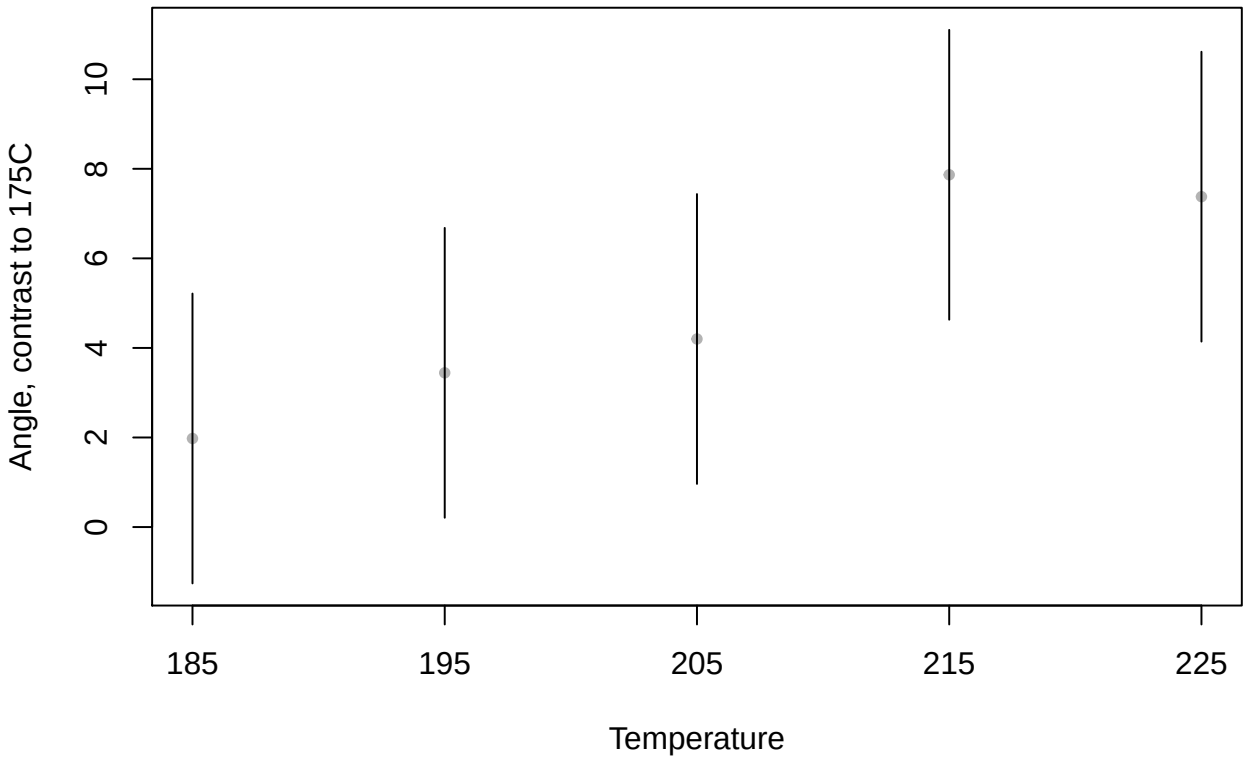
You won't have to know about tapply(), but for those who want to, it is a function that takes the first argument, and for each level of the second argument calculate a statistic, which is the third argument (this can also be a function). For example:

```
thing <- 1:50
lvls <- factor(rep(1:5, each=10))
tapply(thing, lvls, mean)
```

```
##    1    2    3    4    5
##  5.5 15.5 25.5 35.5 45.5
```

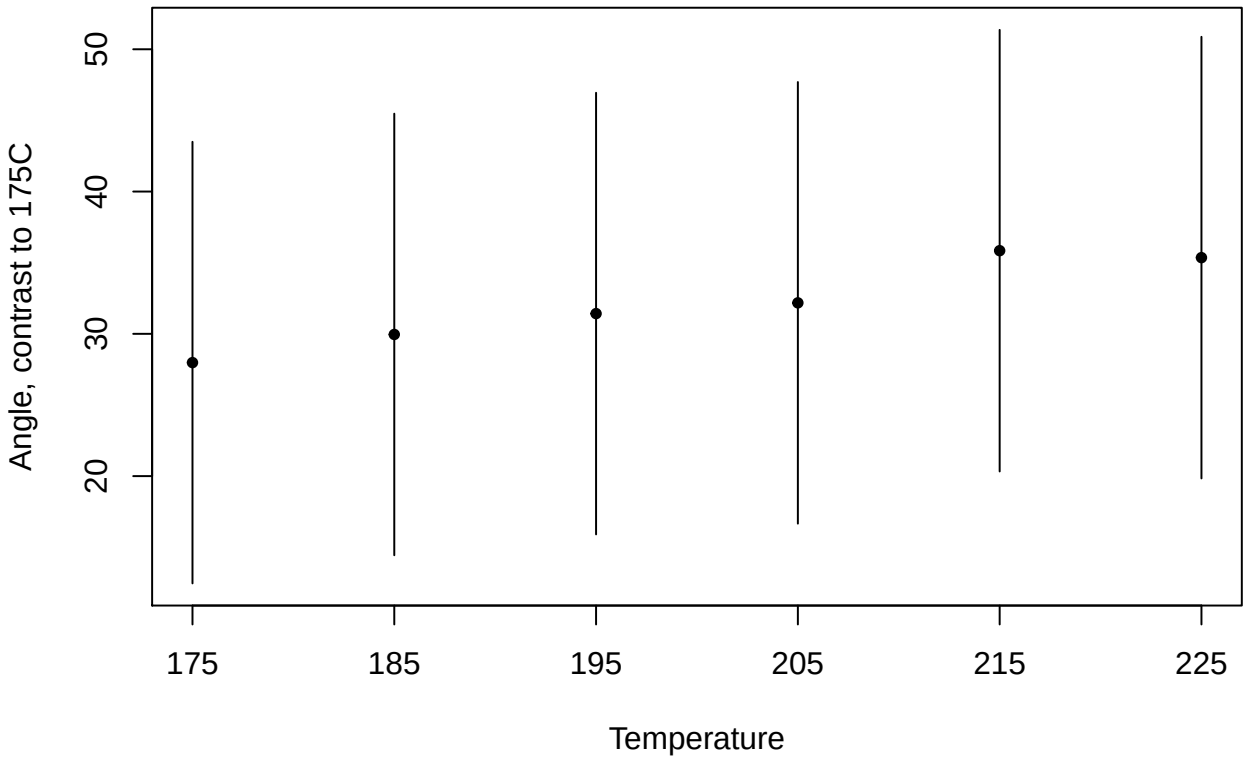
We can plot the parameter estimates from the model, but they are contrasts to the first level:

```
ConfInts <- confint(model.temp)
ConfInts <- ConfInts[rownames(ConfInts)!="(Intercept)",]
Coefs <- coef(model.temp)[names(coef(model.temp))!="(Intercept)"]
plot(1:length(Coefs), Coefs, col="grey70", pch=20, ylim=range(ConfInts),
     xaxt="n", xlab="Temperature", ylab="Angle, contrast to 175C")
segments(1:length(Coefs), ConfInts[,1],
         1:length(Coefs), ConfInts[,2])
axis(1, at=1:length(Coefs),
     labels=gsub("tempF", "", rownames(ConfInts)))
```



We can see that the angles are all larger than at the first temperature (which has a value of 0), and that it increases as temperature increases. We can also plot the predictions from the model. It is easier to do this for some new data, so we only predict each level once.

```
Pred.dat <- data.frame(tempF=levels(cake$tempF))
Pred.cake <- predict(model.temp, newdata = Pred.dat, interval="prediction")
Pred.cake <- cbind(Pred.dat, Pred.cake) # merge data into 1 dataframe

plot(1:nrow(Pred.cake), Pred.cake$fit, pch=20, ylim=range(Pred.cake[,-1]),
     xaxt="n", xlab="Temperature", ylab="Angle, contrast to 175C")
segments(1:nrow(Pred.cake), Pred.cake[,"lwr"],
         1:nrow(Pred.cake), Pred.cake[,"upr"])
axis(1, at=1:nrow(Pred.cake),
     labels=gsub("tempF", "", Pred.cake$tempF))
```



Notice that the confidence intervals are wider. This is because the intercept is also uncertain, so that uncertainty propagates into the predictions. This is one reason why using contrasts can be better: they isolate the comparisons that are important (e.g. between a control and treatments), and any uncertainty elsewhere is ignored.

Problem 1

- Using the cake data, adapt the code above to look at the effect of replicate (i.e. date) on the angle, treating this as a factor. How does the angle vary with date (the replicates are recorded in order, so higher replicates were baked later)?
- How well does date explain the variation in the angle? We are not really interested in date, so what does this suggest about the model we should be fitting?
- Does the model fit the data? Do the residuals look OK (are they normally distributed, are there outliers, is the variance constant etc.)?

Problem 2

For the cake data, we are interested in the effect of temperature and recipe, so we can fit a joint model.

- Fit a model with both temperature and recipe explaining the angle: the code to do this works the same way as the code for multiple regression. How much of the variation in the data is explained by these factors? What are the effects of temperature and recipe?
- Now add replicate (i.e. date) as an extra effect. Does this change the estimates of the effects of temperature and recipe? How about the uncertainty in these effects (i.e. the standard error)? Does this model explain more of the data?
- How good is the model fit (residuals etc)?
- Although we are not interested in the replicate effect, is it still worth including in the model? Can you comment more generally on what this means for designing (and analysing) your own experiments, and how you can deal with sources of variation that are difficult to control?