Lecture 8: Categorical Variables

Bob O'Hara

bob.ohara@ntnu.no

Before we start...

Reference Group

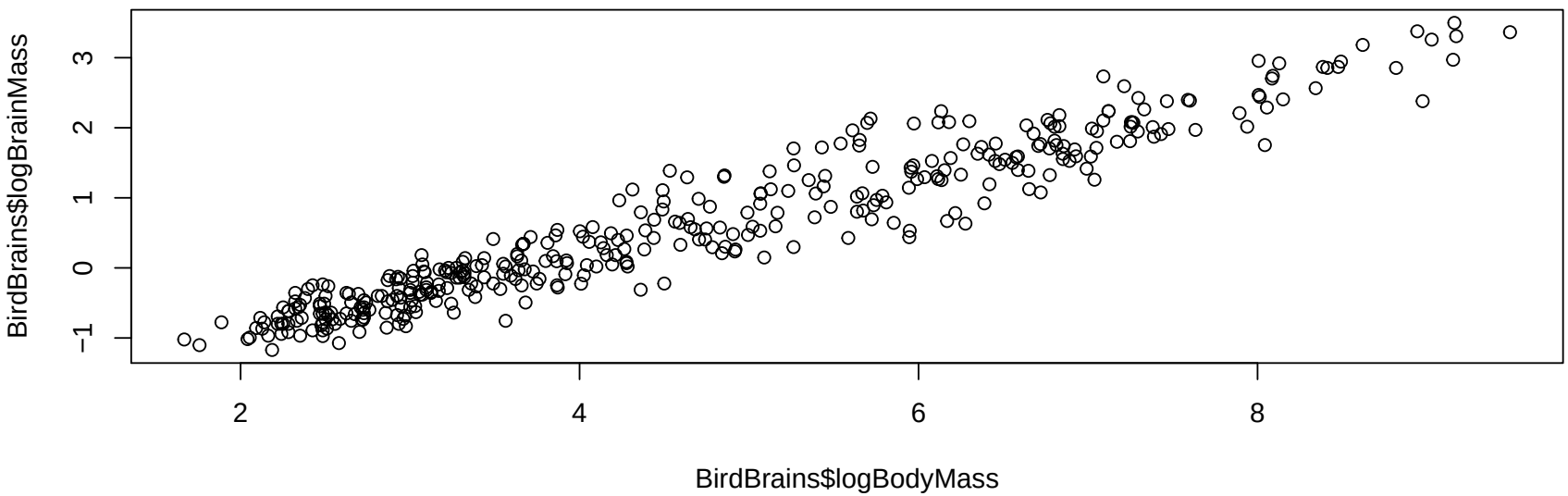
Exercise 1 and R

Problem 1: Bird Brains

We want to look at the relationship between the log of body size and log of brain size. Adapt the code above to answer the following questions:

- First, read in the data

```
BirdBrains <- read.csv("../Data/BirdBrains.csv")
plot(BirdBrains$logBodyMass, BirdBrains$logBrainMass)
```



Problem 1: Bird Brains

What is the relationship between log of body size and log of brain size? What is the effect of increasing body size? Does brain size increase proportionally (e.g. if you double body size, does brain size double?)?

- fit a model

```
brains.mod <- lm(logBrainMass~logBodyMass, data=BirdBrains)
# just show the coefficients
round(summary(brains.mod)$coefficients, 3)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.038      0.043 -47.267        0
## logBodyMass       0.563      0.009  65.293        0
```

Problem 1: Bird Brains

What is the relationship between log of body size and log of brain size? What is the effect of increasing body size? Does brain size increase proportionally (e.g. if you double body size, does brain size double?)?

- the relationship is positive: a larger body size means a larger brain size.
- if the brain size doubles, logBrainSize increases by $\log(2) = 0.69$.
- So body mass increases by $0.56 \times \log(2) = 0.39$.

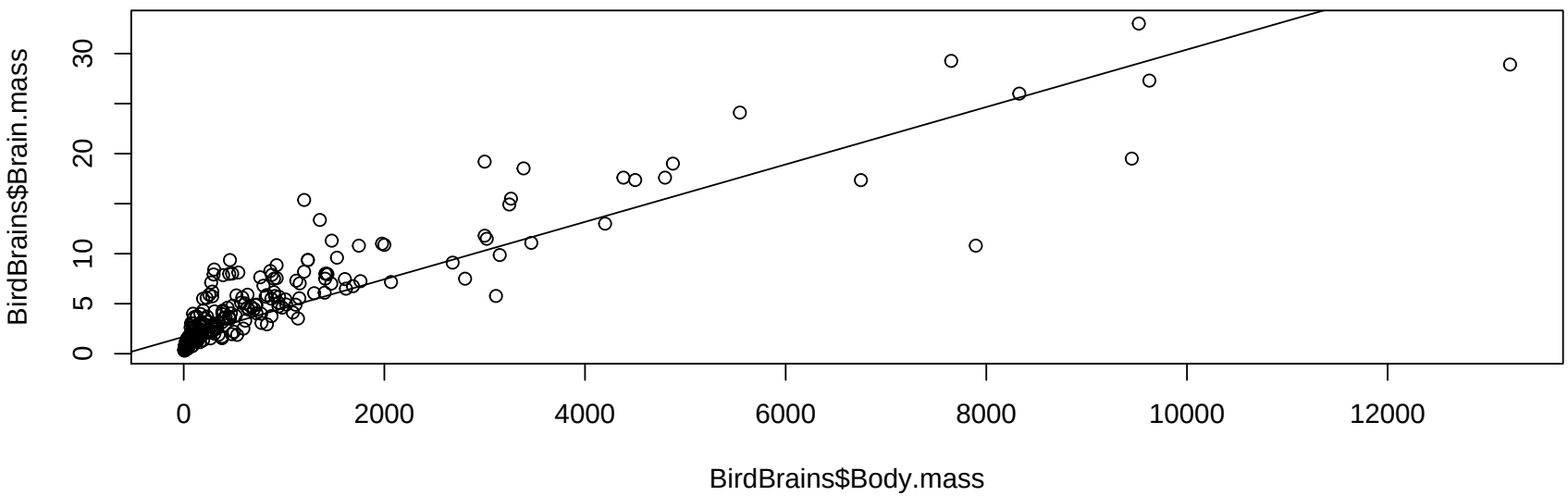
This is less than $\log(2)$, so it is less than proportional

Problem 1: Bird Brains

1. What is the relationship between log of body size and log of brain size? What is the effect of increasing body size? Does brain size increase proportionally (e.g. if you double body size, does brain size double?)?

- We can see that it is less than proportional by plotting the original data

```
plot(BirdBrains$Body.mass, BirdBrains$Brain.mass)
abline(lm(Brain.mass~Body.mass, data=BirdBrains))
```



Model Fit

How good is the model fit? Does it look like there any problems with the fit, e.g. outliers, curavture in the data?

First, from the summary, we can see that the R^2 is 92%, which means that the model is really good: only 8% of the variance is unexplained

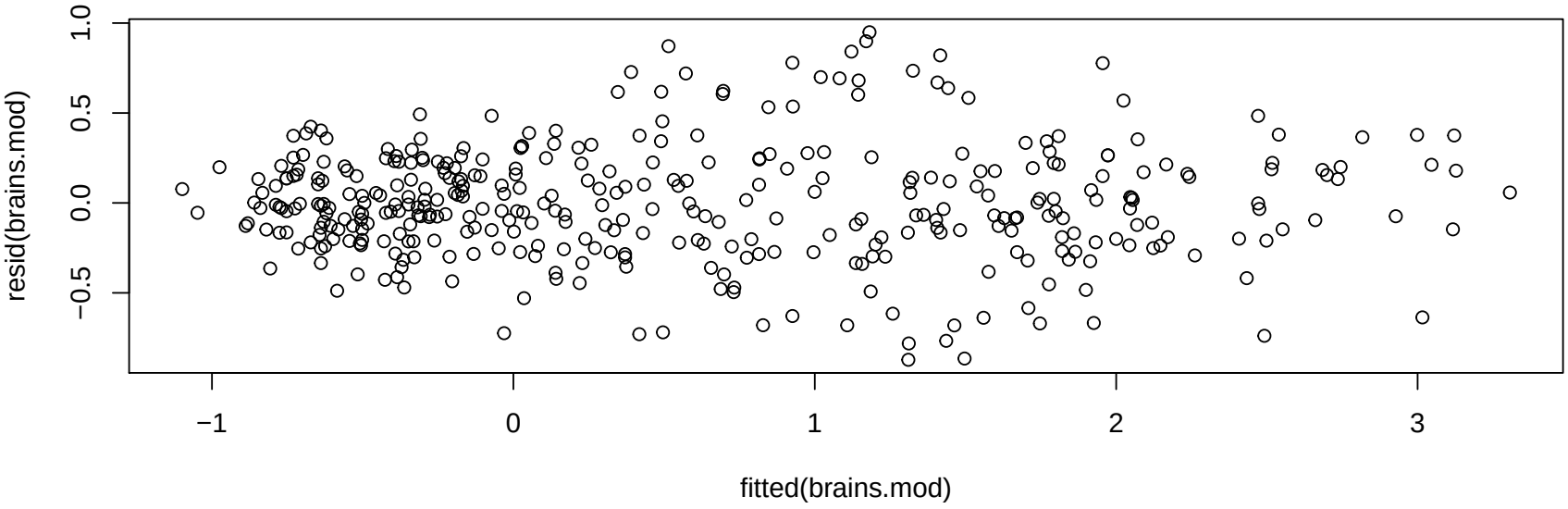
```
Residual standard error: 0.32 on 382 degrees of freedom
 Multiple R-squared:  0.92, Adjusted R-squared:  0.92
 F-statistic: 4.3e+03 on 1 and 382 DF,  p-value: <2e-16
```

Model Fit

How good is the model fit? Does it look like there any problems with the fit, e.g. outliers, curavture in the data?

Now some residual plots

```
plot(fitted(brains.mod), resid(brains.mod))
```



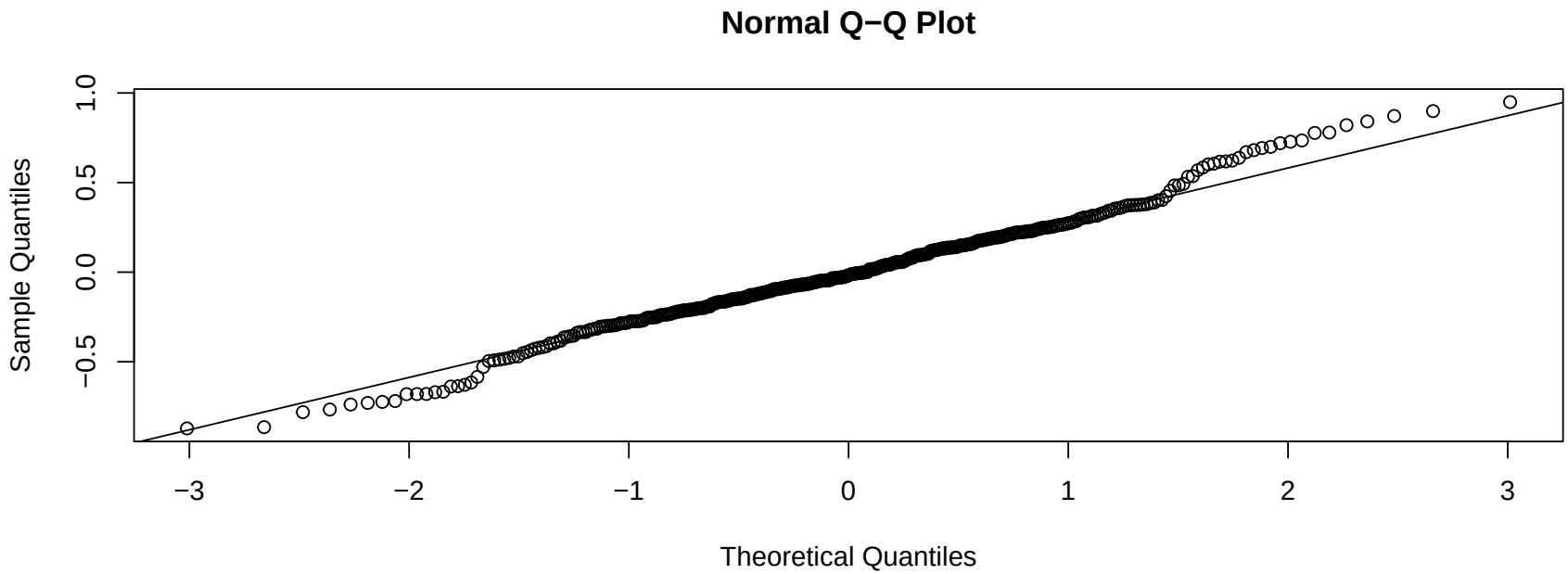
This looks OK: it's a blob. There doesn't seem to be any curvature.

Model Fit

How good is the model fit? Does it look like there any problems with the fit, e.g. outliers, curavture in the data?

Normal Probability plots

```
qqnorm(resid(brains.mod))
qqline(resid(brains.mod))
```



Looks largely OK, but tails might be a bit thick

Model Fit

For the parrots (Order Psitacciformes), predict what their brain size would be if they were normal birds. Compare the predicted and real values - does the distribution of their brain sizes look the same as other birds?

First, extract the parrots and make the predictions

```
BirdBrains$IsParrot <- BirdBrains$Order=="Psitacciformes"
ParrotPred <-BirdBrains[BirdBrains$IsParrot,]
p.pred <- predict(brains.mod, newdata = ParrotPred,
                  interval = "prediction")# Why the interc



 # You do not need to worry about this next line!
rownames(p.pred) <- sapply(ParrotPred$Species.name.,
    function(str)gsub('^.* ', paste0(substr(str, 1, 1),

                                     ". "), str))
```

Compare Parrots

For the parrots (Order Psitacciformes), predict what their brain size would be if they were normal birds. Compare the predicted and real values - does the distribution of their brain sizes look the same as other birds?

We should compare the real & predicted values

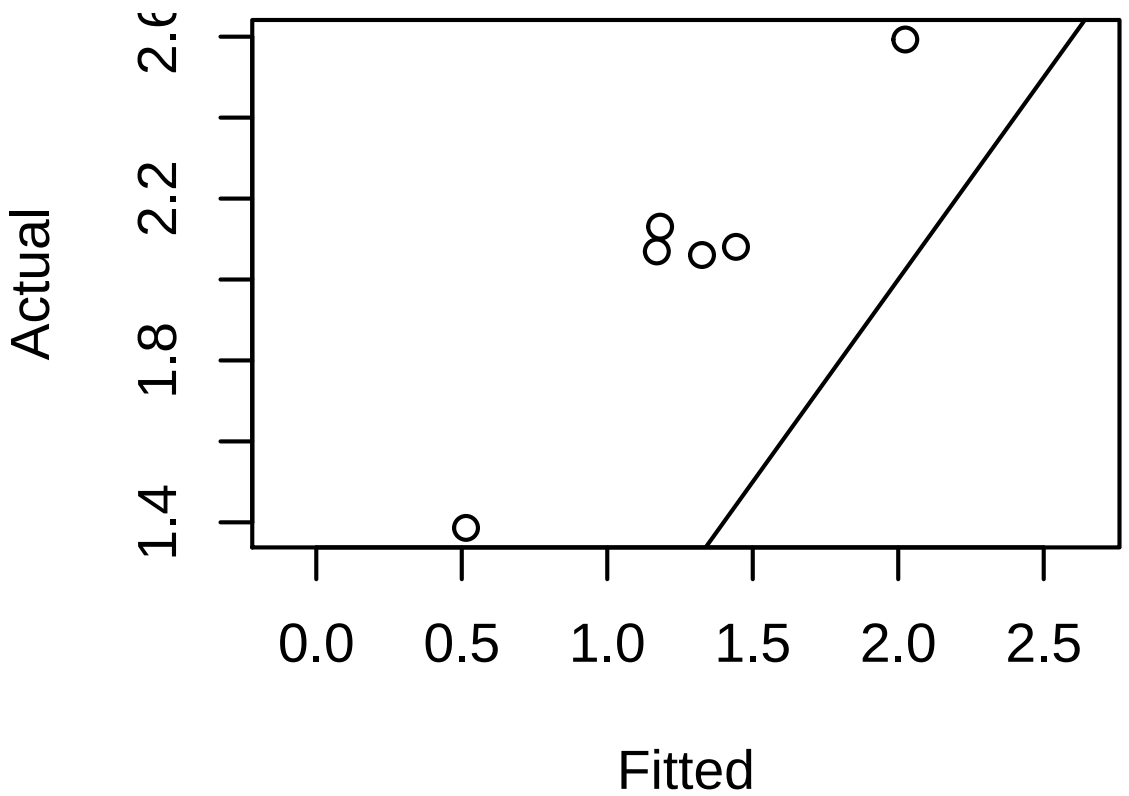
```
print(cbind(ParrotPred$logBrainMass, p.pred), digits=2)
```

```
                    fit   lwr upr
A. aestiva      2.1 1.33  0.70 2.0
A. amazonica    2.1 1.18  0.56 1.8
A. finschi      2.1 1.17  0.55 1.8
A. ochrocephala 2.1 1.44  0.82 2.1
A. ararauna     2.6 2.02  1.40 2.7
P. meyeri       1.4 0.51 -0.11 1.1
```

Plot Parrots

Better to plot. A few ways to do this

```
par(mar=c(4.1,4.1,0.1,1), cex=0.8)
plot(p.pred[,"fit"], ParrotPred$logBrainMass, xlim=range(p
     xlab="Fitted", ylab="Actual")
abline(0,1) # add 1:1 line
```



Parrot Differences

For the parrots (Order Psitacciformes), predict what their brain size would be if they were normal birds. Compare the predicted and real values - does the distribution of their brain sizes look the same as other birds?

Or calculate differences:

```
round(ParrotPred$logBrainMass[1:3] - p.pred[1:3,"fit"], 2)
```

```
  A. aestiva A. amazonica   A. finschi
        0.73         0.95         0.90
```

```
round(ParrotPred$logBrainMass[4:6] - p.pred[4:6,"fit"], 2)
```

```
A. ochrocephala     A. ararauna       P. meyeri
           0.64            0.57            0.87
```

Categorical Variables, aka Factors

So far we have dealt with continuous variables.

But not everything is continuous.

Design of Experiments

A lot of the theory was developed for designed experiments

- field trials in Rothamsted
- lab studies
- clinical trials

Examples

For historical reasons, this is also called "Analysis of Variance"

- Testing different crop varieties
- Test effects of drugs/poisons

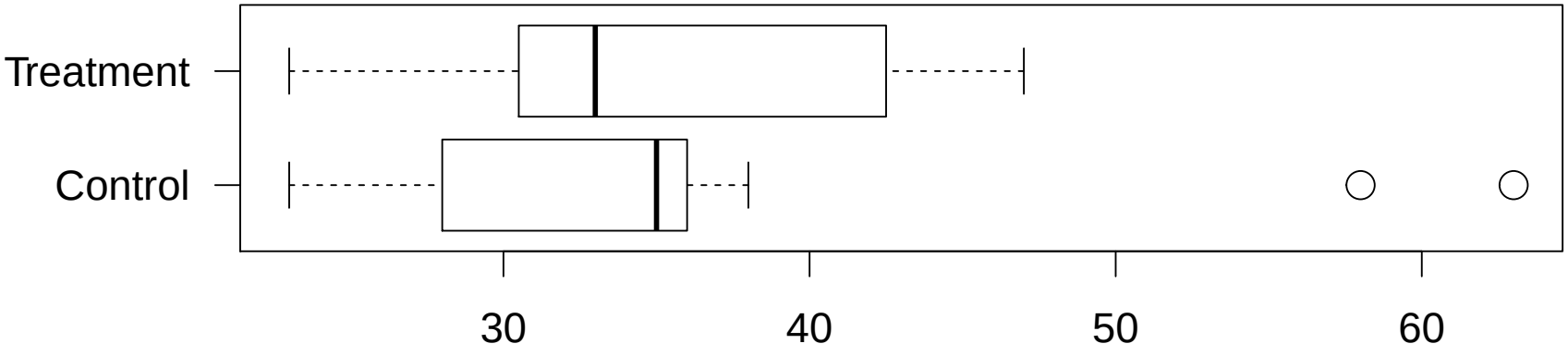
Real Data

Effects of sugar and chcolate temperature on breaking chocolate cake

- Bake a chocolate cake
- hold 1 end
- lift other end
- measure angle at which it breaks
- repeat until you get your Masters degree

One categorical Variable, 2 levels

Use data for 2 recipes: "normal" (control), and more sugar. For now only use 1 temperature



"Easy" example: control and treatment

- same as a t-test

Question: does the treatment (i.e. adding sugar) have an effect?

- does the cake break more easily?
- is it different from the control?

Experimental Design

We assume that individual units are assigned at random to Control and Treatment

(the full experiment has cakes baked on different days, which we will ignore, and temperature, which we will include later)

The Model

We assume that the response is control and treatment are normally distributed, and the only difference is their mean:

$$y_{i,control} = \mu_{control} + \varepsilon_i$$
$$y_{i,treatment} = \mu_{treatment} + \varepsilon_i$$
$$\epsilon_i \sim N(0, \sigma^2)$$

We can write this like this:

$$y_i = \alpha + \beta_j + \varepsilon_i$$
$$\epsilon_i \sim N(0, \sigma^2)$$

where β_j is the difference between control & treatment effects (we can set $\beta_1 = 0$), and α is an intercept.

From this, $E(y_i) = \alpha + \beta_j$

Regression

We could also use regression

- set $X = 0$ for the control
- set $X = 1$ for the treatment

(X is called an 'indicator variable')

The coefficient is a change is the response when X changes by 1, i.e. from control to treatment

The Intercept is the mean for the control

Regression

```
Sugar$X <- as.numeric(Sugar$treatment=="Treatment")
mod.reg <- lm(angle~X, data=Sugar)
print(summary(mod.reg)$coefficients, digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   35.733       2.45  14.560 1.37e-14
## X             -0.667       3.47  -0.192 8.49e-01
```

So the difference is -0.67, with a standard error of 3.47.

Regression

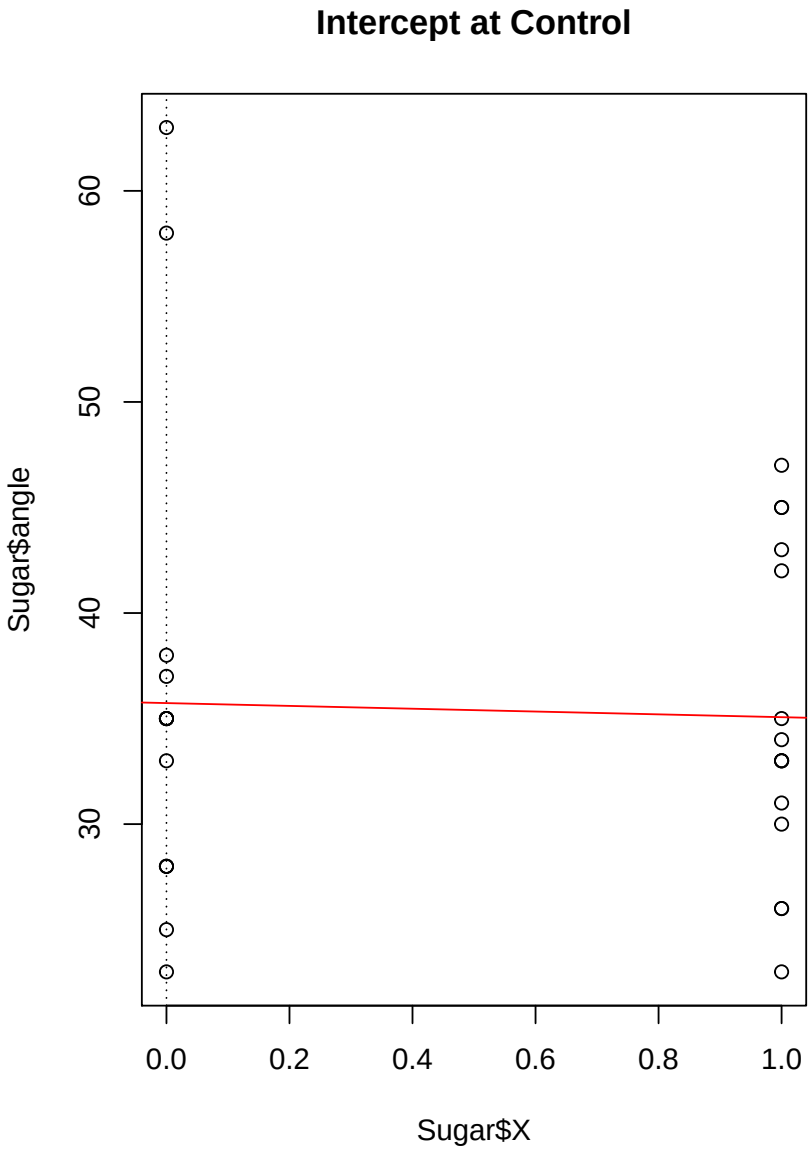
But why put the intercept on the control? Why not put it mid-way between the Control & Treatment, for example?

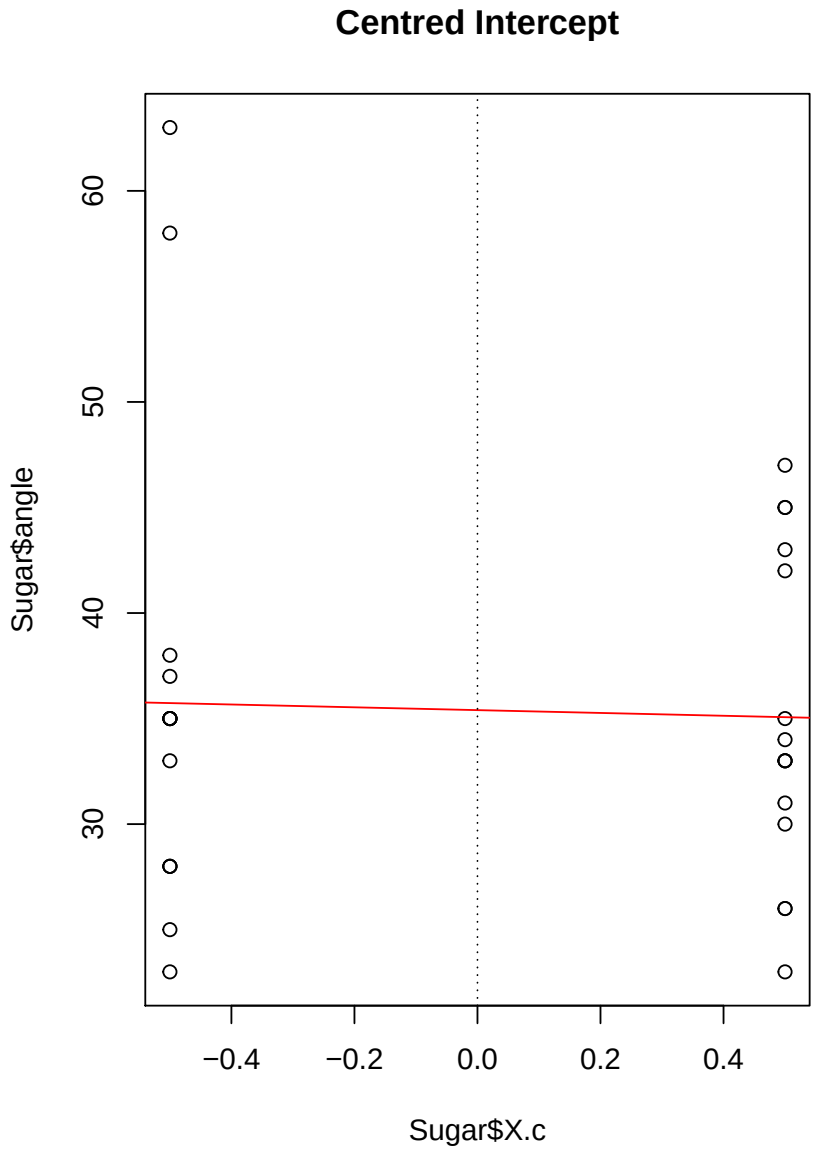
```
Sugar$X.c <- as.numeric(Sugar$treatment=="Treatment")-0.5
mod.reg2 <- lm(angle~X.c, data=Sugar)
print(summary(mod.reg2)$coefficients, digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   35.400       1.74  20.398 2.42e-18
## X.c           -0.667       3.47  -0.192 8.49e-01
```

The difference is the same, but the intercept is now at 35.4

Regression

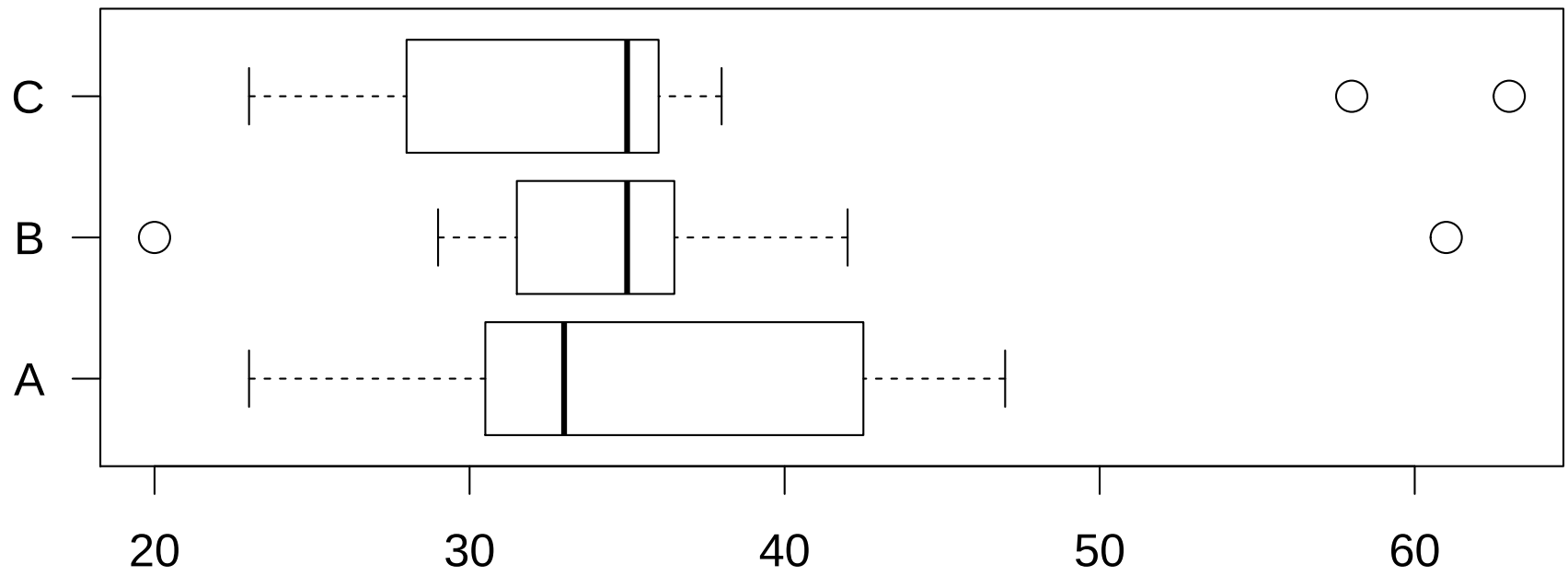

Intercept at Control
Sugar$angle
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
Sugar$X


Centred Intercept
Sugar$angle
30
40
50
60
−0.4
−0.2
0.0
0.2
0.4
Sugar$X.c

More than 2 levels

With Control & Treatment we have 2 levels, but what if we have more?

```
Recipes <- cake[cake$temp=="225",] # use temp. of 225C
par(mar=c(4.1,6,1,1), cex=1.5)
boxplot(Recipes$angle~Recipes$recipe,
        horizontal=TRUE, las=1)
```



More than 2 levels

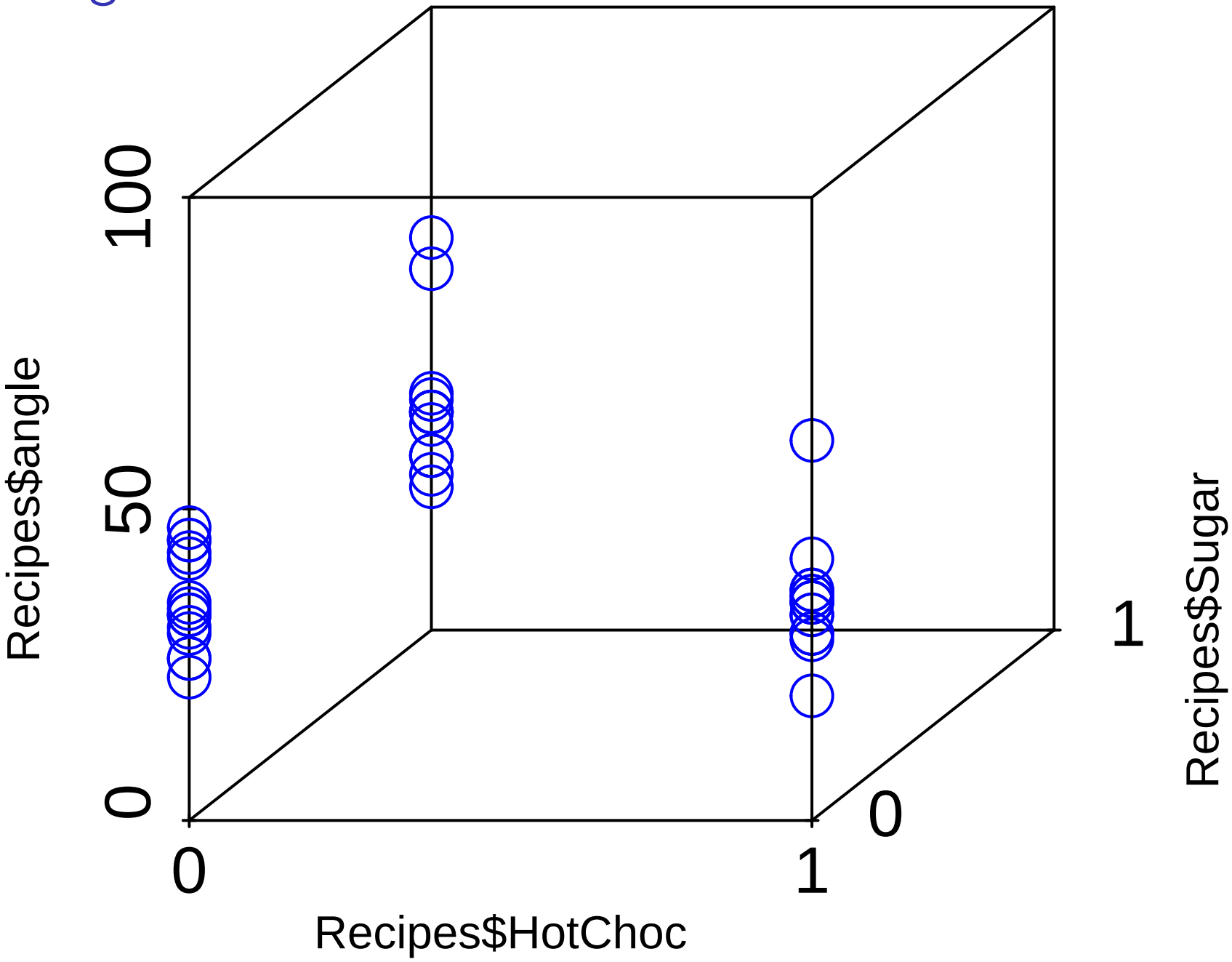
We can use the same regression trick

```
Recipes$HotChoc <- as.numeric(Recipes$recipe=="B") # hotte
Recipes$Sugar <- as.numeric(Recipes$recipe=="C") # More su

head(Recipes[,c("recipe", "HotChoc", "Sugar", "angle")])
```

```
   recipe HotChoc Sugar angle
6       A       0     0    42
12      B       1     0    42
18      C       0     1    63
24      A       0     0    45
30      B       1     0    61
36      C       0     1    58
```

Drawing 3 levels



More than 2 levels

Fit the model...

```
mod.3lvl <- lm(angle ~ HotChoc + Sugar,data=Recipes)
print(summary(mod.3lvl)$coefficients, digits=3)
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    35.067       2.38 14.7046 3.64e-18
HotChoc         0.200       3.37  0.0593 9.53e-01
Sugar           0.667       3.37  0.1977 8.44e-01
```

We now have effects for the difference between hotter chocolate and the control & more sugar and the control

Changing the Intercept

Make Recipe B the intercept

```
Recipes$ColdChoc <- as.numeric(Recipes$recipe=="A") # colde
Recipes$Sugar <- as.numeric(Recipes$recipe=="C") # More sug

mod.3lvl.1 <- lm(angle ~ ColdChoc + Sugar, data=Recipes)
print(summary(mod.3lvl.1)$coefficients, digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   35.267       2.38 14.7885 2.97e-18
## ColdChoc      -0.200       3.37 -0.0593 9.53e-01
## Sugar          0.467       3.37  0.1384 8.91e-01
```

The estimate for chocolate temperature was 0.2, now it is -0.2

The estimate for Sugar before was 0.67, now we have 0.47

- now it is a contrast between "more sugar and cold chocolate" and "hot chocolate"

The Design Matrix Reloaded

Last time I introducted multiple regression as

$$\mathbf{Y} = X\beta + \varepsilon$$

where X is the design matrix

Today I have been building the design matrix with 0's and 1's

	recipe	HotChoc	Sugar
6	A	0	0
12	B	1	0
18	C	0	1
24	A	0	0
30	B	1	0
36	C	0	1

The Design Matrix Reloaded

Building the design matrix by hand is a pain, so we get the computer to do it. In the data, recipe is coded as a factor (i.e. categorical):

```
str(Recipes$recipe)
```

```
 Factor w/ 3 levels "A","B","C": 1 2 3 1 2 3 1 2 3 1 ...
```

with different levels

```
levels(Recipes$recipe)
```

```
[1] "A" "B" "C"
```

The Design Matrix Reloaded

All this means the computer knows what to do

```
DesignMatrix <- model.matrix(~recipe, data=Recipes)
head(DesignMatrix)
```

```
   (Intercept) recipeB recipeC
6            1       0       0
12           1       1       0
18           1       0       1
24           1       0       0
30           1       1       0
36           1       0       1
```

The Design Matrix Reloaded

We can thus just fit the model without worrying

```
mod.an <- lm(angle~recipe, data=Recipes)
print(summary(mod.an)$coefficients, digits=3)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   35.067       2.38 14.7046 3.64e-18
recipeB        0.200       3.37  0.0593 9.53e-01
recipeC        0.667       3.37  0.1977 8.44e-01
```

. . . except about how to interpret the parameter estimates

The Design Matrix Contrasted

The coefficients are the same as before:

```
coef(mod.an)
```

```
(Intercept)     recipeB     recipeC
 35.0666667   0.2000000   0.6666667
```

```
coef(mod.3lvl)
```

```
(Intercept)     HotChoc       Sugar
 35.0666667   0.2000000   0.6666667
```

Contrasts

By default, R sets the first level of a factor to the intercept, and the other levels contrasted to the first level

- makes sense here, but not if (for example) we are comparing 6 varieties of barley

We can see the contrasts R uses:

```
contrasts(Recipes$recipe)
```

```
  B C
A 0 0
B 1 0
C 0 1
```

Each contrast is a column

- for contrast B, B gets a value of 1
- for contrast C, C gets a value of 1

Sum to Zero Contrasts

We can also use different contrasts:

```
Recipes$recipeS <- C(Recipes$recipe, contr = contr.sum)
contrasts(Recipes$recipeS)
```

```
  [,1] [,2]
A    1    0
B    0    1
C   -1   -1
```

- harder to interpret
- but can be more flexible if you have specific hypotheses

Sum to Zero Contrasts

```
mod.anS <- lm(angle~recipeS, data=Recipes)
print(summary(mod.anS)$coefficients, digits=3)
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   35.3556       1.38 25.6790 2.65e-27
recipeS1      -0.2889       1.95 -0.1484 8.83e-01
recipeS2      -0.0889       1.95 -0.0457 9.64e-01
```

No intercept

We can also "cheat" by fitting a model with no intercept, by putting "-1" on the right hand side of the model:

```
mod.NoI <- lm(angle~recipe - 1, data=Recipes)
print(summary(mod.NoI)$coefficients, digits=3)
```

```
        Estimate Std. Error t value Pr(>|t|)
recipeA     35.1       2.38    14.7 3.64e-18
recipeB     35.3       2.38    14.8 2.97e-18
recipeC     35.7       2.38    15.0 1.87e-18
```

Two categorical Variables

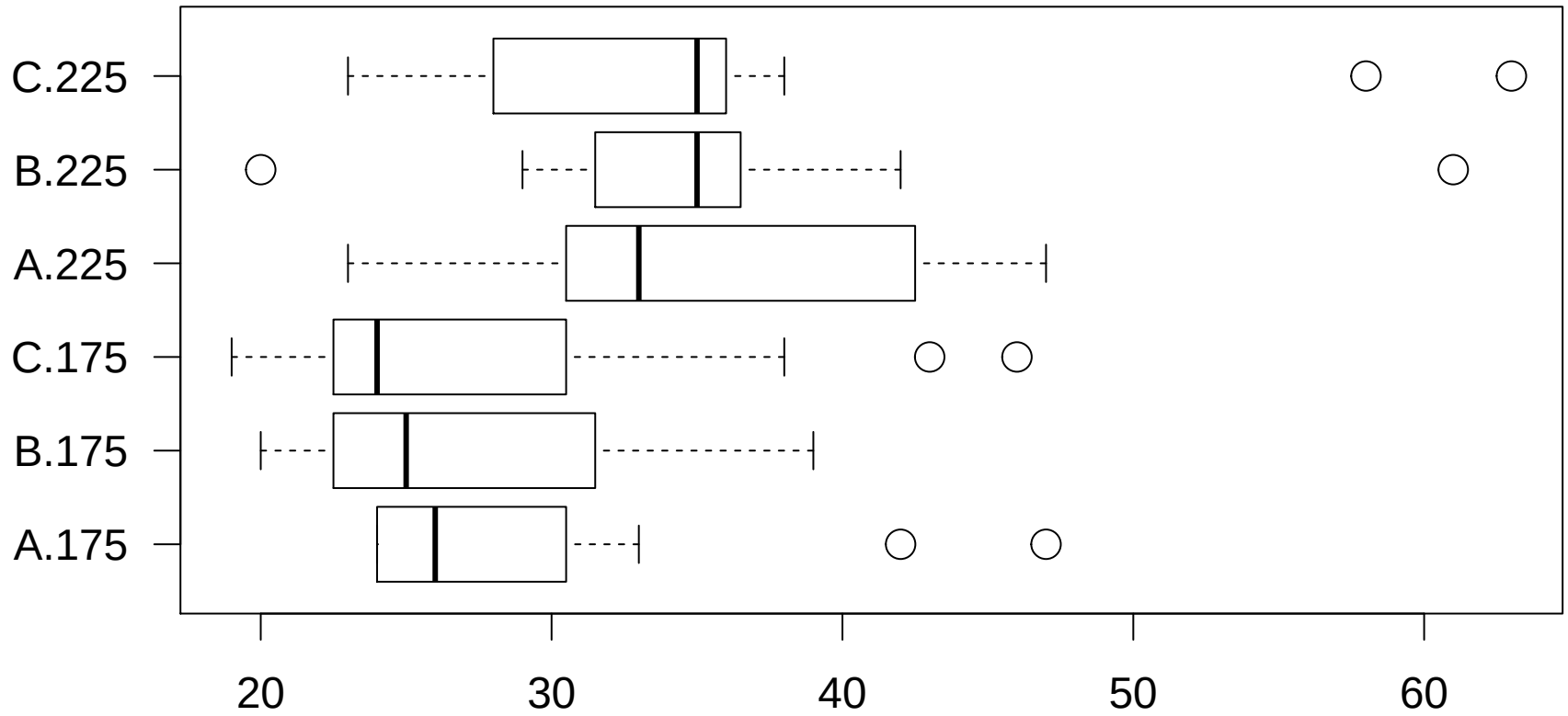
We do not have to only consider one variable: just like multiple regression, we can consider several.

e.g. with the cakes, temperature was also a factor that was controlled

```
 # use temperatures of 175C & 225C
Cakes2 <- cake[cake$temp%in%c("175", "225") ,]
```

Two categorical Variables

```
par(mar=c(4.1,6,1,1), cex=1.5)
boxplot(Cakes2$angle~Cakes2$recipe + Cakes2$temp,
        horizontal=TRUE, las=1)
```



Two categorical Variables

The contrasts for the temperature can be written in the same way

```
Cakes2$tempF <- factor(Cakes2$temp)
contrasts(Cakes2$tempF)
```

```
    225
175   0
225   1
```

Two categorical Variables

The design matrix is now like this:

```
head(model.matrix(~recipe + tempF, data=Cakes2))
```

```
   (Intercept) recipeB recipeC tempF225
1            1       0       0        0
6            1       0       0        1
7            1       1       0        0
12           1       1       0        1
13           1       0       1        0
18           1       0       1        1
```

We now just have an extra column for temperature

Two categorical Variables: Fitting the model

We can fit the model, just like last time!

```
mod.2way <- lm(angle~recipe + tempF, data=Cakes2)
print(summary(mod.2way)$coefficients, digits=2)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    28.41        1.7   16.48  2.4e-28
recipeB        -1.03        2.1   -0.49  6.3e-01
recipeC        -0.27        2.1   -0.13  9.0e-01
tempF225        7.38        1.7    4.28  4.9e-05
```

Little effect of recipe, big effect of temperature

- higher temperature means cake bends more before breaking

Why the intercept?

What happens if we remove the intercept?

```
mod.2way <- lm(angle~recipe + tempF-1, data=Cakes2)
print(summary(mod.2way)$coefficients, digits=2)
```

```
          Estimate Std. Error t value Pr(>|t|)
recipeA       28.4        1.7    16.5  2.4e-28
recipeB       27.4        1.7    15.9  2.7e-27
recipeC       28.1        1.7    16.3  4.5e-28
tempF225       7.4        1.7     4.3  4.9e-05
```

We get 3 levels of recipe, but still only one of tempF

Why the intercept?

The problem is that we can only remove the intercept for one factor, otherwise there are too many moving parts.

```
head(model.matrix(~recipe + tempF-1, data=Cakes2))
```

```
   recipeA recipeB recipeC tempF225
1        1       0       0        0
6        1       0       0        1
7        0       1       0        0
12       0       1       0        1
13       0       0       1        0
18       0       0       1        1
```

If we had a 175°C effect, we could subtract that from all of the recipe effects and add it to the 225°C effect & get the same model

Next Week

More complicated model: interactions

- when the recipe effect depends on temperature

Problem 2

We can simulate some bad data and look at what happens. To simulate some "good" data, we can do this:

```
# define the parameters
alpha <- 5; beta <- 5; sigma <- 1 # standard deviation

# 100 samples from a uniform distribution between 0 and 1
x <- runif(100, 0, 1) # random uniform distribution
# calculate E(y) using the defined values of alpha & beta,
#   and the simulated values of x
mu <- alpha + beta*x
# simulate y, with mean mu and standard deviation sigma
y <- rnorm(length(mu), mu, sigma)
```

Problem 2

Simulate the 'bad' data

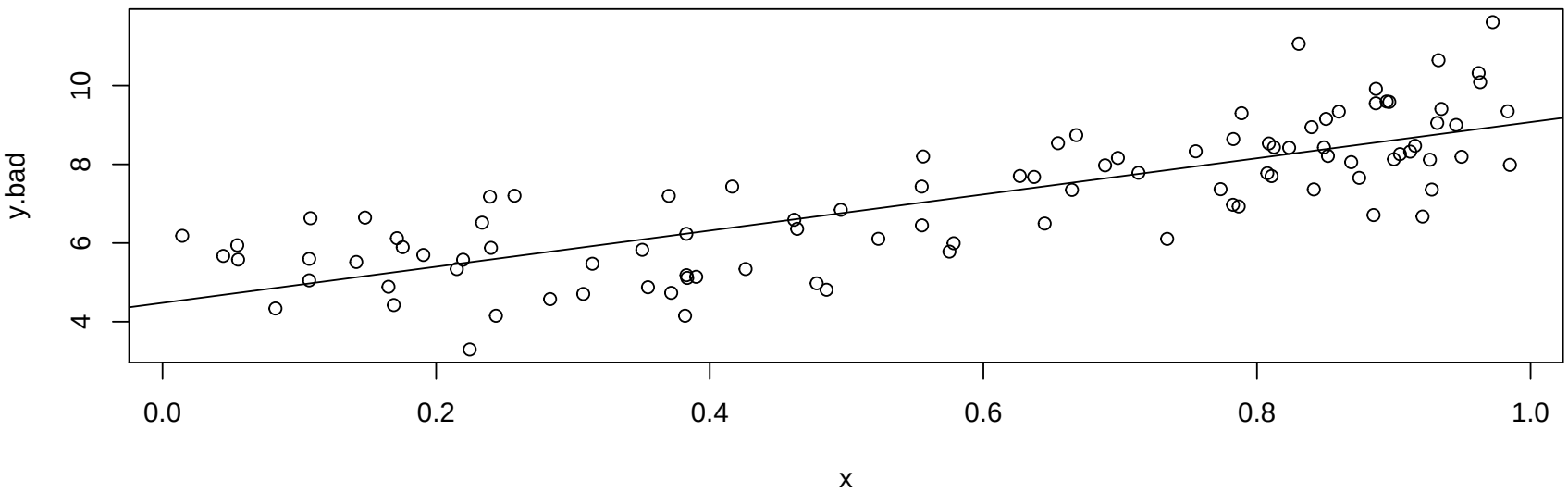
```
mu.bad <- alpha + beta*x^2
y.bad <- rnorm(length(mu.bad), mu.bad, sigma)
```

Problem 2

Fit the model to the bad data

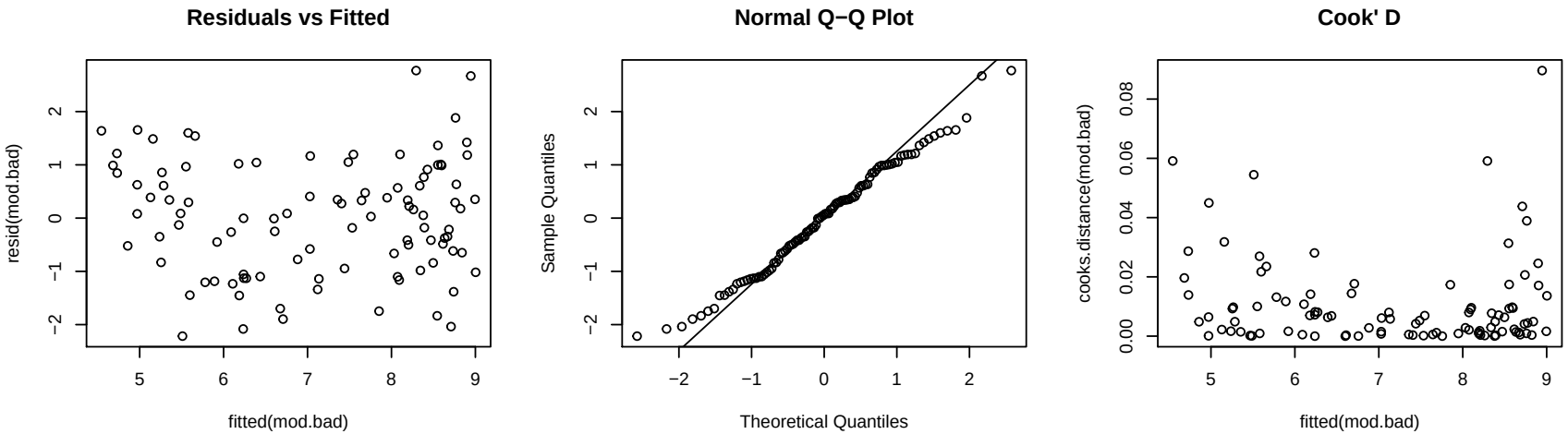
```
mod.bad <- lm(y.bad~x)

plot(x, y.bad)
abline(mod.bad) # add the fitted line
```



This doesn't look too bad.

Problem 2: Residuals



The residuals look curved. THe QQ-plot looks fine: Cook's D is OK.