Classic model selection

Key

Tue May 10 15:28:16 2022

Classic model selection

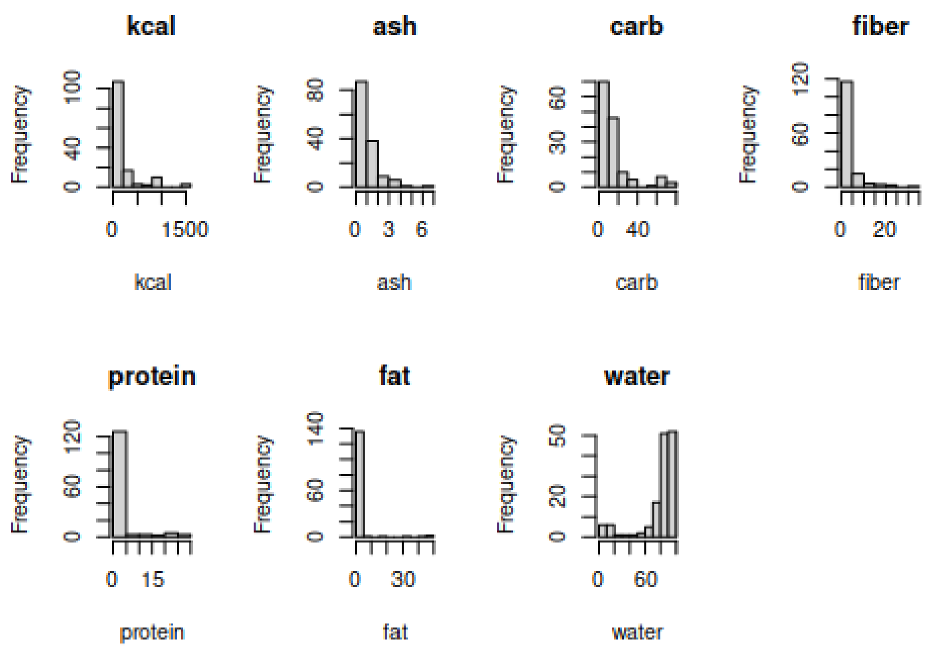
We will work with data from the USDA National Nutrient Database. Import the data set:

```
library(readxl)
data.frame(read_excel("macronutrients.xls")) -> food
```

Graphing the data

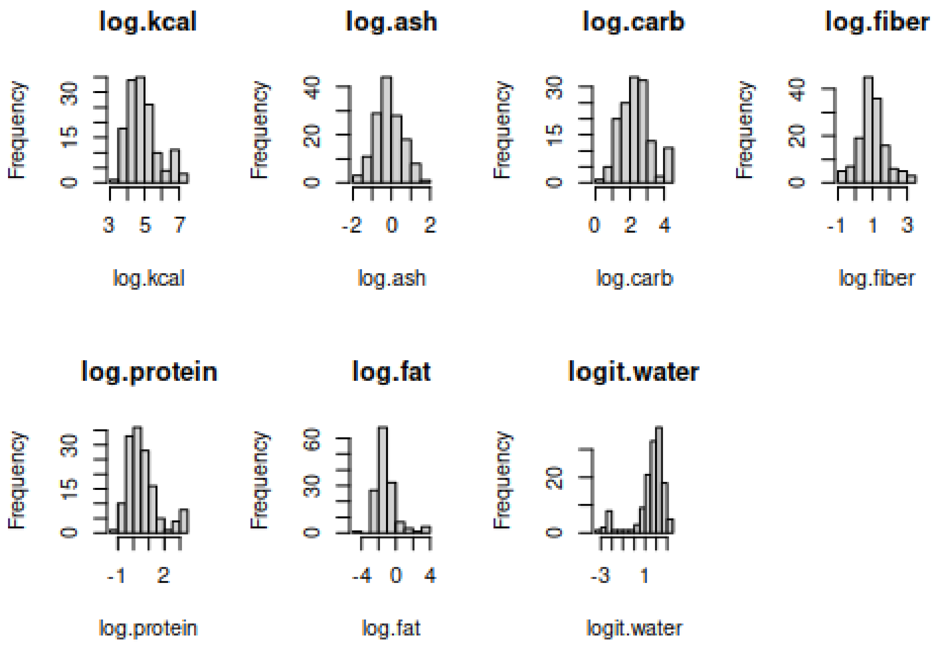
To get an idea of the distribution of each of the variables, we can plot a histogram for each. To save space, we'll plot them in a 4x4 array.

```
oldpar <- par()
par(oma = c(0,0,3,0), mfrow=c(2,4))

untrans.var <- c("kcal","ash","carb","fiber","protein","fat","water")

lapply(untrans.var, FUN = function(x) hist(food[ , x], main = x, xlab
= x))
```



Now plot the transformed histograms:

```
par(oma = c(0,0,3,0), mfrow=c(2,4))

trans.var <-
c("log.kcal","log.ash","log.carb","log.fiber","log.protein","log.fat",
"logit.water")

lapply(trans.var, FUN = function(x) hist(food[ , x], main = x, xlab =
x))
```

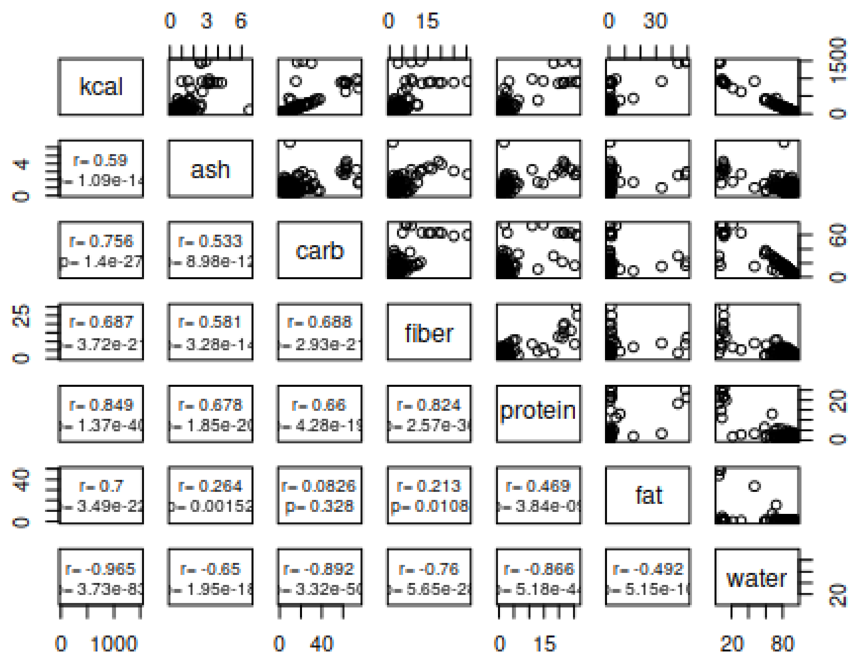


Run this chunk to add a function, panel.cor(), that will put the correlation coefficient and p-value into the lower panel of the correlation matrix plot.

```
panel.cor <- panel.cor <- function(x, y, cex.cor = 0.8, method =
"pearson", ...) {
options(warn = -1)                          # Turn off warnings
(e.g. tied ranks)
usr <- par("usr"); on.exit(par(usr))        # Saves current "usr"
and resets on exit
par(usr = c(0, 1, 0, 1))                    # Set plot size to 1 x
1
r <- cor(x, y, method = method, use = "pair")   # correlation coef
p <- cor.test(x, y, method = method)$p.val      # p-value
txt <- format(r, digits = 3)                # Format r-value
txt1 <- format(p, digits = 3)               # Format p-value
txt2 <- paste0("r= ", txt, '\n', "p= ", txt1)   # Make panel text
text(0.5, 0.5, txt2, cex = cex.cor, ...)    # Place panel text
options(warn = 0)                           # Reset warning
}
```

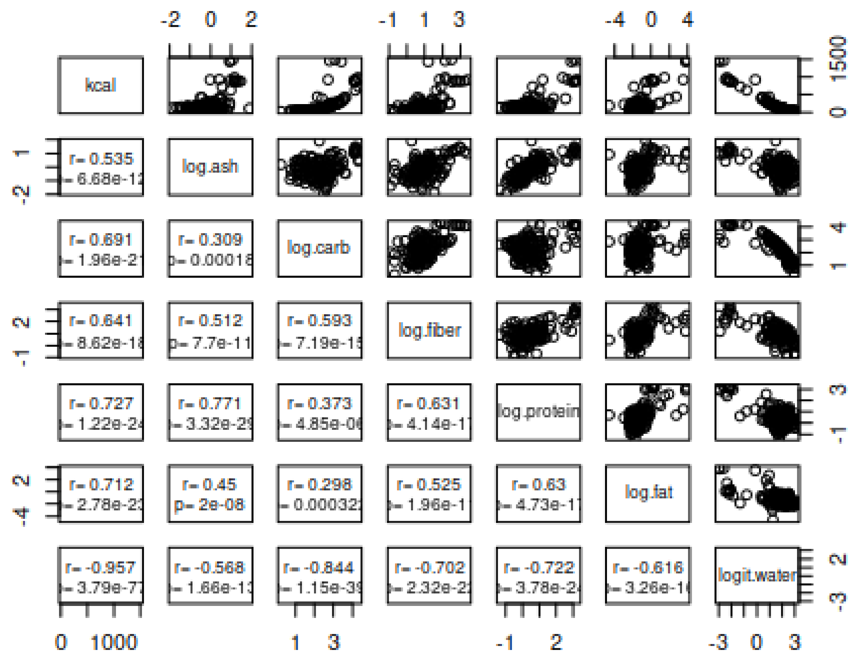
Next we should see how the variables relate to one another. First, we will make a scatterplot matrix using kcal, along with the un-transformed versions of the predictors:

```
pairs(food[,untrans.var], lower.panel = panel.cor)

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



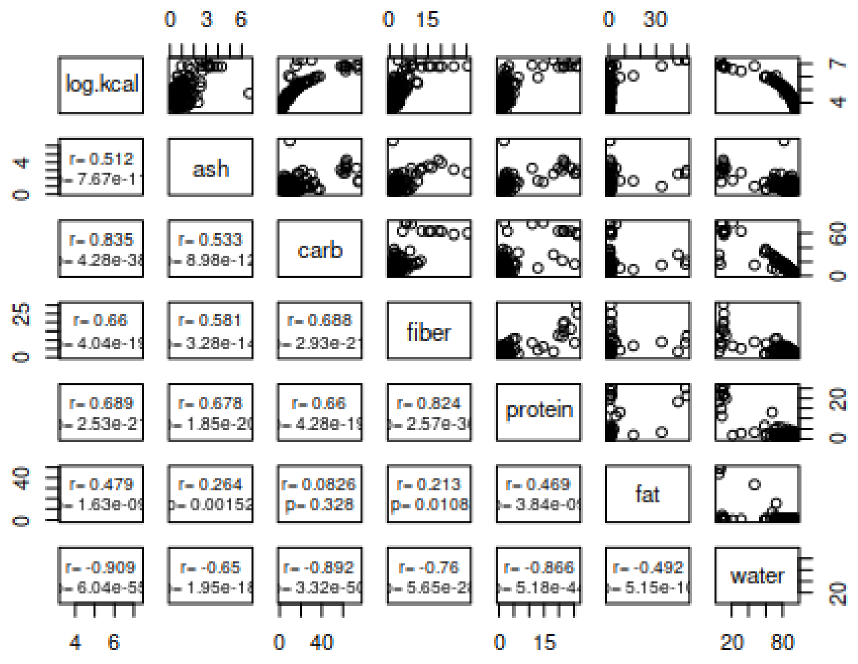
Now make one for kcal against the log-transformed versions of the predictors:

```
pairs(food[,c("kcal",trans.var[-1])], lower.panel = panel.cor)
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



Using the log of kcal might help - plot it against the un-transformed predictors:

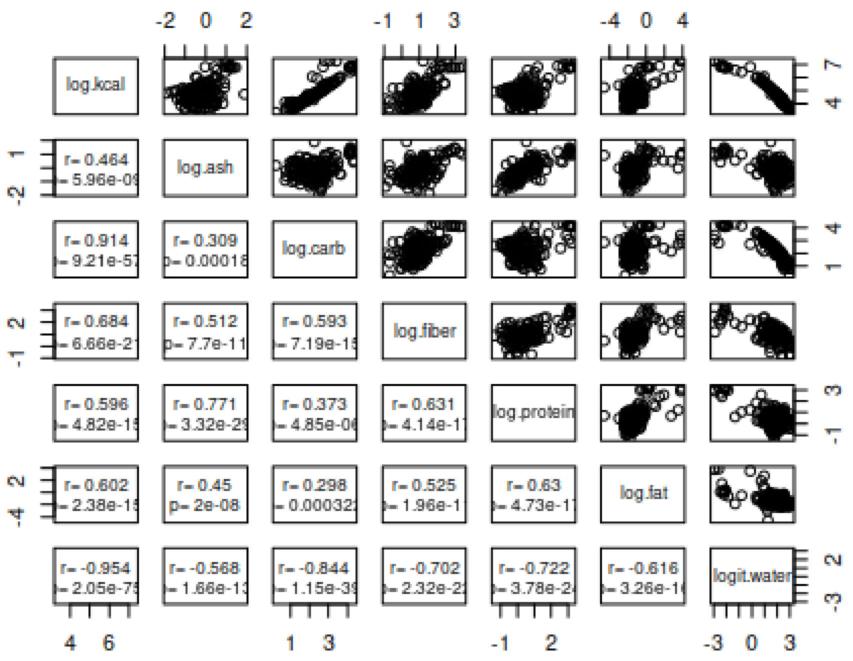
```
pairs(food[,c("log.kcal","ash","carb","fiber","protein","fat","water")
], lower.panel = panel.cor)

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



Finally, plot the transformed log.kcal against the transformed predictors:

```
pairs(food[,trans.var], lower.panel = panel.cor)
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



Question: why is it better to use the transformed versions of the variables instead of the raw variables, based on the scatterplots?

The scatterplots are more linear using the transformed variables.

Fitting models

Make an empty list to hold the models:

```
models.list <- list()
```

Add all of the single predictor models to the list:

```
models.list$log.ash <- lm(log.kcal ~ log.ash, data = food)
models.list$log.carb <- lm(log.kcal ~ log.carb, data = food)
models.list$log.fiber <- lm(log.kcal ~ log.fiber, data = food)
models.list$log.protein <- lm(log.kcal ~ log.protein, data = food)
models.list$log.fat <- lm(log.kcal ~ log.fat, data = food)
models.list$logit.water <- lm(log.kcal ~ logit.water, data = food)
```

Get the r.squared values for each model:

```
sapply(models.list, FUN = function(x) summary(x)$r.squared) ->
r.squared
```

Get the adj.r.squared values:

```
sapply(models.list, FUN = function(x) summary(x)$adj.r.squared) ->
adj.r.squared
```

Combine the r.squared and adj.r.squared into a data frame to make them easier to see:

```
data.frame(r.squared, adj.r.squared)

##             r.squared adj.r.squared
## log.ash     0.2154293     0.2098252
## log.carb    0.8356839     0.8345102
## log.fiber   0.4676565     0.4638540
## log.protein 0.3557148     0.3511128
## log.fat     0.3620739     0.3575173
## logit.water 0.9109813     0.9103455
```

Question: which single predictor variable model has the highest adjusted r-squared?

The logit.water model had an adjusted R^2 of 0.91.

The two predictors log.ash and log.protein are strongly correlated. Fit a model with both included and get a summary here:

```
models.list$log.ash.log.protein <- lm(log.kcal ~ log.ash +
log.protein, data = food)
sapply(models.list, FUN = function(x) summary(x)$r.squared) ->
r.squared
sapply(models.list, FUN = function(x) summary(x)$adj.r.squared) ->
adj.r.squared
data.frame(r.squared, adj.r.squared)
```

```
##                     r.squared adj.r.squared
## log.ash             0.2154293     0.2098252
## log.carb            0.8356839     0.8345102
## log.fiber           0.4676565     0.4638540
## log.protein         0.3557148     0.3511128
## log.fat             0.3620739     0.3575173
## logit.water         0.9109813     0.9103455
## log.ash.log.protein 0.3557625     0.3464929
```

Question: did the model with both log.ash and log.protein have a higher r.squared than than either of them by themselves? Did it have a higher adj.r.sqaured than either by themselved?

> The model with log.ash and log.protein had an R^2 of 0.35576, which is slightly higher than log.protein alone (0.35571), and higher than log.ash alone (0.21543).

Fit a model with all of the predictors included:

```
models.list$all.var <- lm(log.kcal ~ log.ash + log.carb + log.fiber +
log.protein + log.fat + logit.water, data = food)
sapply(models.list, FUN = function(x) summary(x)$r.squared) ->
r.squared
sapply(models.list, FUN = function(x) summary(x)$adj.r.squared) ->
adj.r.squared
data.frame(r.squared, adj.r.squared)
```

```
##                     r.squared adj.r.squared
## log.ash             0.2154293     0.2098252
## log.carb            0.8356839     0.8345102
## log.fiber           0.4676565     0.4638540
## log.protein         0.3557148     0.3511128
## log.fat             0.3620739     0.3575173
## logit.water         0.9109813     0.9103455
## log.ash.log.protein 0.3557625     0.3464929
## all.var             0.9734324     0.9722516
```

Question: did the model with all predictors included have a higher adj.r.squared than all the others so far?

> Yes, the adjusted R^2 for the model with all of the predictors is 0.9734, which is higher than any of the other models.

Use a Type II SS ANOVA table to decide whether there are variables that are not contirbuting statistically significant amounts of explained variation to the model:

```
library(car)

## Loading required package: carData

Anova(models.list$all.var)

## Anova Table (Type II tests)
##
## Response: log.kcal
##             Sum Sq  Df  F value    Pr(>F)
## log.ash     0.0037   1   0.1640    0.6862
## log.carb    4.3001   1 190.7779 < 2.2e-16 ***
## log.fiber   0.0061   1   0.2698    0.6043
## log.protein 0.0001   1   0.0037    0.9514
## log.fat     2.3441   1 103.9970 < 2.2e-16 ***
## logit.water 1.1617   1  51.5380 4.247e-11 ***
## Residuals   3.0429 135
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question: which variables are statistically significant?

> log.carb, log.fat, and logit.water are all statistically significant.

Fit a model with only the statistically significant predictors from the ANOVA:

```
models.list$signif.pred <- lm(log.kcal ~ log.carb + log.fat +
logit.water, data = food)
data.frame(r.squared, adj.r.squared)

##                     r.squared adj.r.squared
## log.ash             0.2154293     0.2098252
## log.carb            0.8356839     0.8345102
## log.fiber           0.4676565     0.4638540
## log.protein         0.3557148     0.3511128
## log.fat             0.3620739     0.3575173
## logit.water         0.9109813     0.9103455
## log.ash.log.protein 0.3557625     0.3464929
## all.var             0.9734324     0.9722516
```

Question: did the model with only the significant predictors have a better adj.r.squared than all the others?

> Yes, by a tiny amount, 0.9727 instead of 0.9722.

Question: log.protein did not make it into the best model. Does this mean that protein has no calories? Explain.

> No, protein has caleries. The correlation between log.protein and the other predictors is large enough that log.protein is no longer significant when it's included with the others.

Double check that the three significant predictors that you dropped are not just statistically redundant with one another by testing the model without them against the model that included them:

```
anova(models.list$signif.pred, models.list$all.var)

## Analysis of Variance Table
##
## Model 1: log.kcal ~ log.carb + log.fat + logit.water
## Model 2: log.kcal ~ log.ash + log.carb + log.fiber + log.protein +
log.fat +
##     logit.water
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    138 3.0595
## 2    135 3.0429  3  0.016554 0.2448 0.8649
```

Question: how did your test of the reduced model against the full model show you that log.ash, log.protein, and log.fiber are not just statistically redundant with one another?

> Dropping log.protein, log.ash, and log.fiber didn't reduce the fit of the model to the data significantly, because the p-value was 0.86. This means that the variables were not contributing to the fit, rather than that they were important predictors that were so correlated with one another that their coefficients were non-significant.

Question: look at the p-values for each of the models. Are any of the models non-significant? If you had just looked at a single model, would you have considered it to be a poor model based on the p-value alone for any of these? What does that tell you about the importance of fitting multiple models and comparing their adjusted R^2?

> All of the models were statistically significant, so none of them would have appeared to be poorly supported by the data. Fitting multiple models allowed us to see that a combination of log.carb, log.fat, and logit.water was the best representation for the data, and was much better than any of the single-predictor models.

To make sure we aren't selecting a "best" model that is in reality terrible, we should make sure that the model is statistically significant. Get the summary() of signif.pred:

```
summary(models.list$signif.pred)

##
## Call:
## lm(formula = log.kcal ~ log.carb + log.fat + logit.water, data =
food)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41775 -0.08656 -0.02313  0.05090  0.59965
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.10820    0.09895  41.519   <2e-16 ***
## log.carb     0.56459    0.03154  17.899   <2e-16 ***
## log.fat      0.15609    0.01491  10.468   <2e-16 ***
## logit.water -0.23408    0.02402  -9.744   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1489 on 138 degrees of freedom
## Multiple R-squared:  0.9733, Adjusted R-squared:  0.9727
## F-statistic:  1676 on 3 and 138 DF,  p-value: < 2.2e-16
```

Question: is the model with only the statistically significant predictors significant overall? How do you know (what part of the output tells you this)?

Yes, it has a p-value of 2.2e-16.

Models as hypotheses

Once you have your own hypotheses created in the Excel spreadsheet provided, import it as a dataset called food.hypotheses:

```
hypotheses <- read_excel("macronutrients_nopredictors.xls")
```

To start, make an empty list for the models you will use (my H.example, and your two hypotheses), called food.hypotheses.list:

```
food.hypotheses.list <- list()
```

Fit a model with my hypothesis (H.example) as a predictor, and assign it to the food.hypotheses.list:

```
food.hypotheses.list$H.example <- lm(log.kcal ~ H.example, data =
hypotheses)
summary(food.hypotheses.list$H.example)

##
## Call:
## lm(formula = log.kcal ~ H.example, data = hypotheses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.34854 -0.42817 -0.04348  0.31313  1.92313
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.3224     0.3041  14.214  < 2e-16 ***
## H.exampleFruit   0.5378     0.3141   1.712   0.0891 .
## H.exampleLeaf   -0.1136     0.3231  -0.352   0.7256
## H.exampleRoot    0.7397     0.3346   2.211   0.0287 *
## H.exampleSeed    2.3341     0.3422   6.820 2.72e-10 ***
## H.exampleShoot   0.2721     0.3477   0.782   0.4354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6082 on 136 degrees of freedom
## Multiple R-squared:  0.5608, Adjusted R-squared:  0.5446
## F-statistic: 34.73 on 5 and 136 DF,  p-value: < 2.2e-16
```

Code chunks for your own models should follow.