University of Glasgow
Institute of Neuroscience & Psychology

# Generalized mixed effects models for the analysis of experimental data

Christoph Scheepers

christoph.scheepers@glasgow.ac.uk

# Course Overview

- **Session 1: Linear Regression**

  Relationship to t-test and ANOVA; predictor coding and interpretation; performing tests

- **Session 2: Generalized Linear Models**

  Adjusting assumptions about error distributions and the relationship between IVs and DVs; data types and model *families*; gamma regression; binary and ordinal logistic regression

- **Session 3: Generalized Linear Mixed Models**

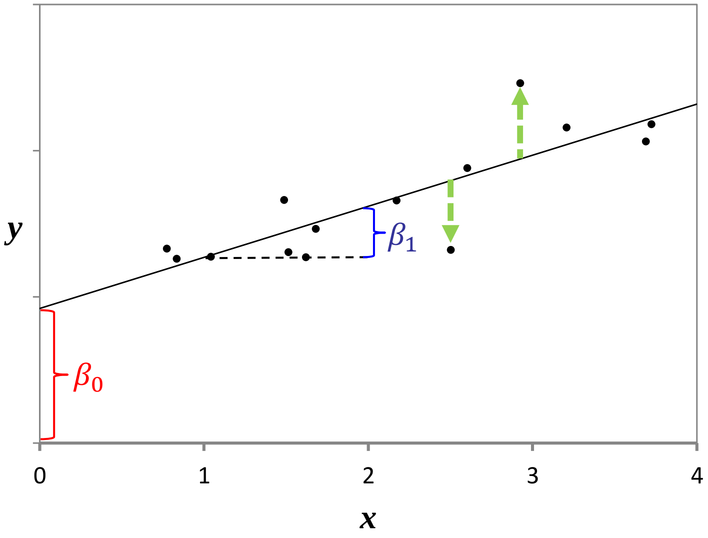
  Repeated-measures designs; random and fixed factors/effects; random intercepts and random slopes; ‘maximal’ GLMMs

- **Session 4: Control Predictors in Maximal GLMMs**

  Confound variables and how to treat them in a maximal GLMM

# Linear Regression

Christoph Scheepers

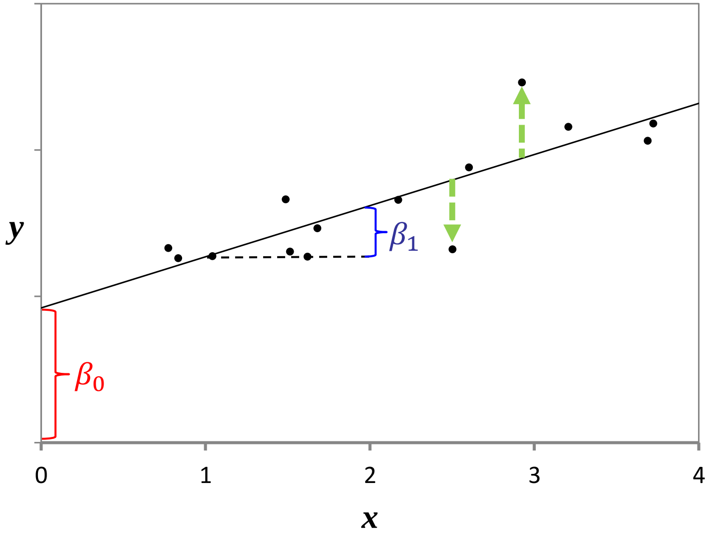


christoph.scheepers@glasgow.ac.uk

# Simple Linear Regression

- **Goal**: Predict a continuous DV (*y*) from a continuous IV (x), assuming a *linear relationship* between the two

$$\hat{y}_i = \beta_o + \beta_1 x_i \ ,$$
$$y_i = \hat{y}_i + e_i$$



where

$\hat{y}_i$ = predicted value of $y_i$

$x_i$ = value of the predictor variable

$\beta_0$ = the ***intercept*** (or *regression constant*): the value of $\hat{y}_i$ when $x$ = 0

$\beta_1$ = the ***slope*** (or *regression coefficient*): the difference in $\hat{y}_i$ associated with a one-unit increase in $x$

$e_i$ = prediction ***error*** (residuals)

# Simple Linear Regression

- The ‘best fitting’ line through an $x \cdot y$ data cloud is one that ***minimizes* the *residuals*** (prediction errors); formally:

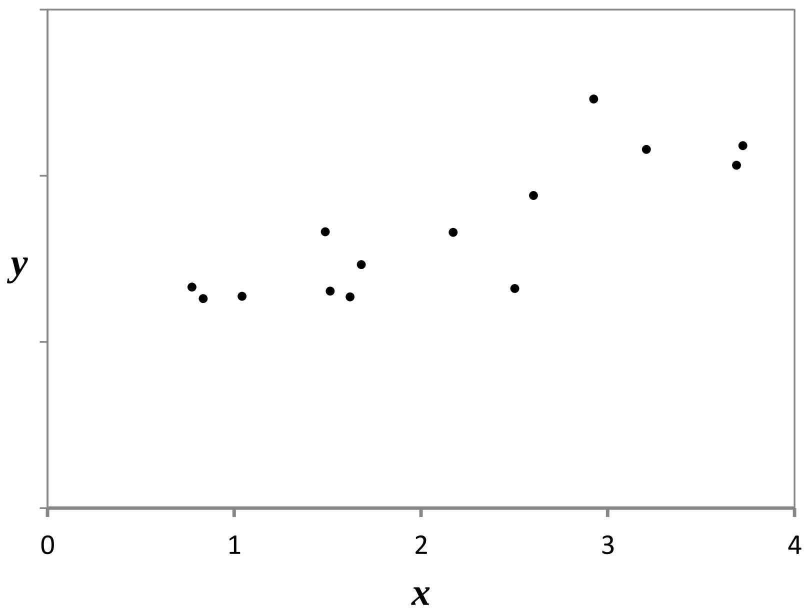
$$min(\sum(y - \hat{y})^2)$$

- This can be achieved by setting

$$\beta_1 = \frac{COV_{xy}}{s_x^2} = \boldsymbol{r} \cdot (\boldsymbol{s_y}/\boldsymbol{s_x}) \,,$$

$$\beta_0 = \bar{y} - \beta_1 \cdot \bar{x}$$

- Where $\boldsymbol{r}$ is indeed the good old Pearson correlation coefficient!



Original data

# Simple Linear Regression

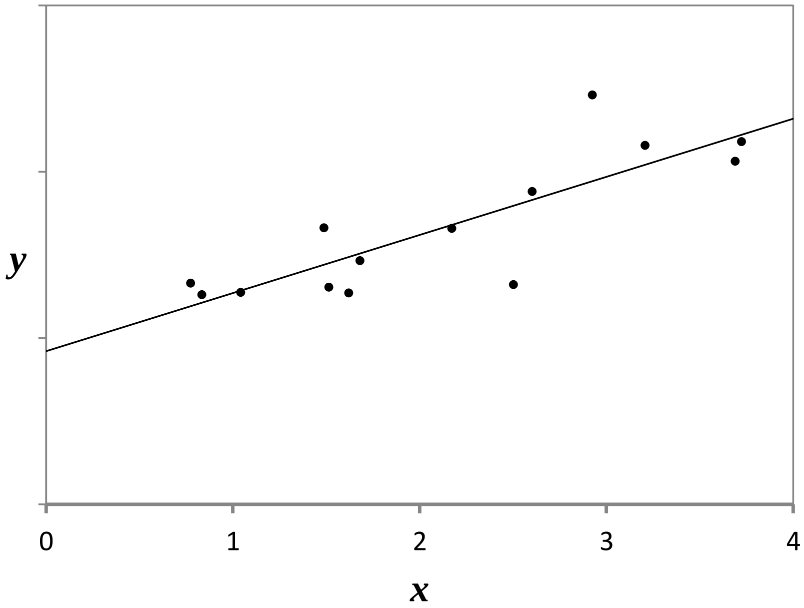
- The 'best fitting' line through an $x \cdot y$ data cloud is one that ***minimizes* the *residuals*** (prediction errors); formally:

$$min(\sum(y - \hat{y})^2)$$

- This can be achieved by setting

$$\beta_1 = \frac{COV_{xy}}{s_x^2} = \boldsymbol{r} \cdot (\boldsymbol{s_y}/\boldsymbol{s_x}) \text{ ,}$$

$$\beta_0 = \bar{y} - \beta_1 \cdot \bar{x}$$



Original data plus regression line such that $\sum(\boldsymbol{y} - \hat{\boldsymbol{y}})^2$ is minimized

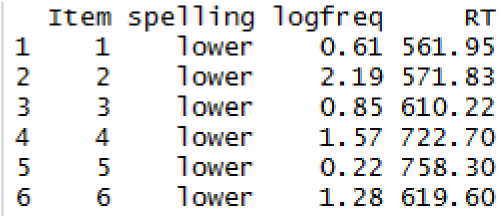
Once we've determined the values for $\boldsymbol{\beta_0}$ (intercept) and $\boldsymbol{\beta_1}$ (slope), we can more or less reliably **predict** what the most likely *y* would be at a given *x*, **e.g. for x = 5:** $\hat{\boldsymbol{y}} = \boldsymbol{\beta_0} + \boldsymbol{\beta_1} \cdot 5$

# Let’s do this in R

- **Example data:** Lexical decision experiment (real data)
  - 144 words (and plenty of non-words as ‘fillers’, which are not included)
  - Each word presented either in `UPPER` or `lower`case font (variable *spelling*)
  - Task: decide as quickly and accurately as possible (button press) whether a given stimulus is an actual word
  - Also recorded for each word: lexical frequency (log10 per million word counts)
  - 33 subjects, but data are aggregated up to item level (not trial-by-trial data!)
- **Questions:**
  - Is there a linear relationship between lexical frequency and RT?
  - How can we predict RT from lexical frequency?

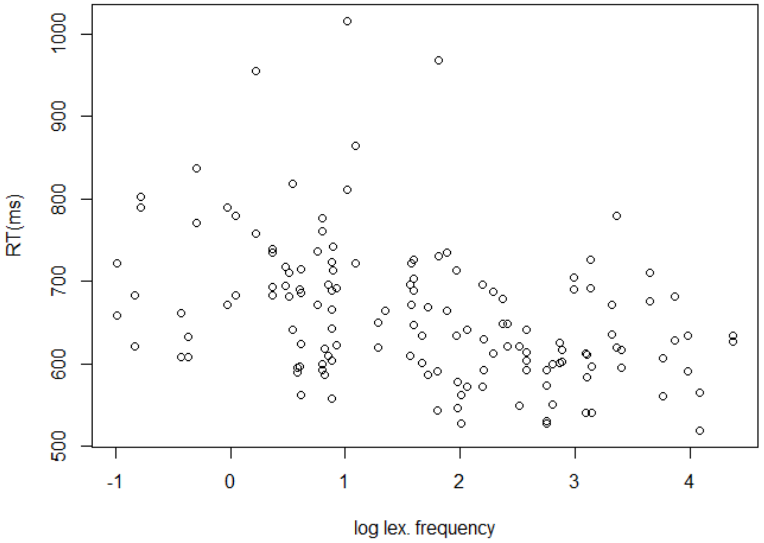
# Let’s do this in R

```
# Simple regression example data
RT.data <- read.csv("http://www.psy.gla.ac.uk/~christop/MScStats/2018/Regress/RTs.csv")
head(RT.data)
```

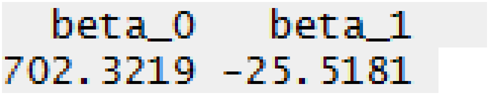


```
  Item spelling logfreq     RT
1    1    lower    0.61 561.95
2    2    lower    2.19 571.83
3    3    lower    0.85 610.22
4    4    lower    1.57 722.70
5    5    lower    0.22 758.30
6    6    lower    1.28 619.60
```

```
# Plot y against x
x <- RT.data$logfreq
y <- RT.data$RT
plot(x, y,
     xlab = "log lex. frequency",
     ylab = "RT(ms)")
```



```
# Calculate intercept (beta_0) and
# slope (beta_1) "on foot"
slope <- cor(x, y) * (sd(y) / sd(x))
intercept <- mean(y) - slope * mean(x)

# Look at what we’ve done
c("beta_0" = intercept, "beta_1" = slope)
```



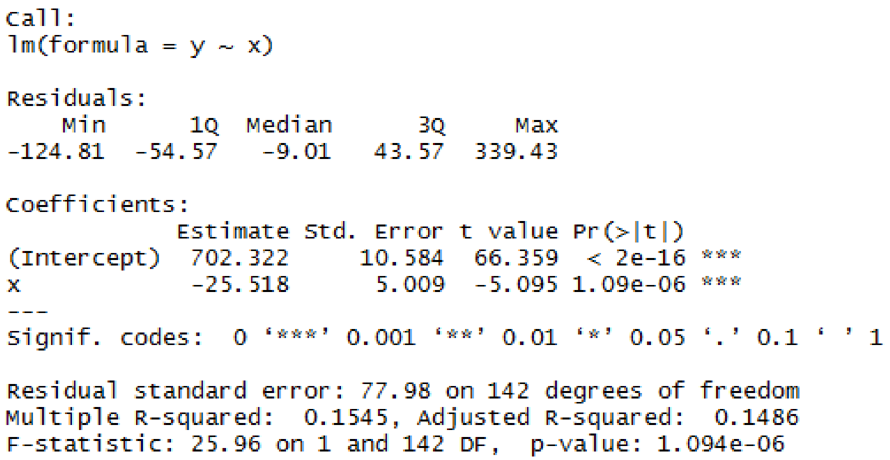
```
  beta_0    beta_1
702.3219 -25.5181
```

- **Thus, we can predict that with every 1-unit increase in *x* (*logfreq*) there is a 25.5 ms decrease in *y* (RT) :**

$$\widehat{RT} = 702\ ms\ - 25.5\ ms\ \cdot logfreq$$

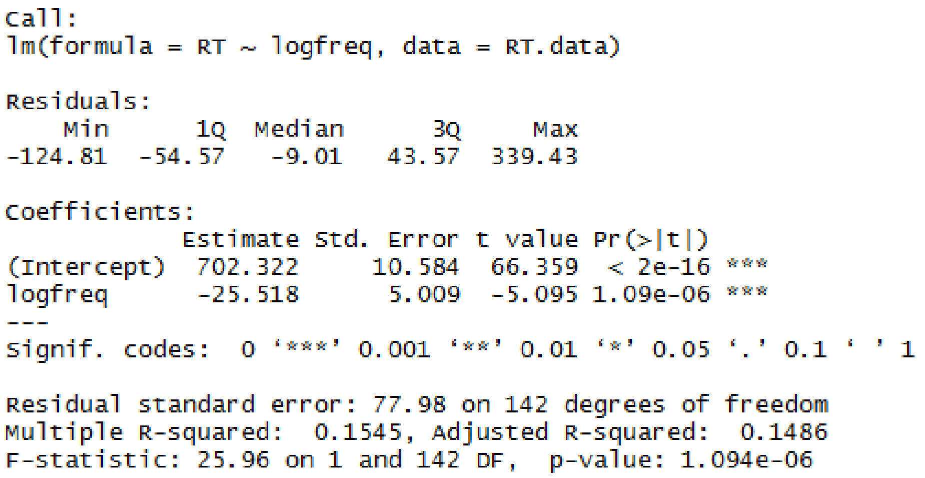
# Function `lm(…)`

```
# Using function lm()
LF1 <- lm(y ~ x)
summary(LF1)
```



```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-124.81  -54.57   -9.01   43.57  339.43 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  702.322     10.584  66.359  < 2e-16 ***
x            -25.518      5.009  -5.095 1.09e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.98 on 142 degrees of freedom
Multiple R-squared:  0.1545,	Adjusted R-squared:  0.1486 
F-statistic: 25.96 on 1 and 142 DF,  p-value: 1.094e-06
```

```
# Alternatively
LF2 <- lm(RT ~ logfreq, data=RT.data)
summary(LF2)
```



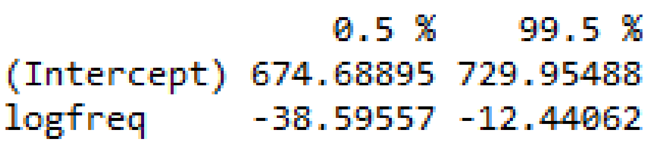
```
Call:
lm(formula = RT ~ logfreq, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-124.81  -54.57   -9.01   43.57  339.43 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  702.322     10.584  66.359  < 2e-16 ***
logfreq      -25.518      5.009  -5.095 1.09e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.98 on 142 degrees of freedom
Multiple R-squared:  0.1545,	Adjusted R-squared:  0.1486 
F-statistic: 25.96 on 1 and 142 DF,  p-value: 1.094e-06
```

- **Things to look out for:**
  - **Coefficients** (intercept and slope); one-sample t tests against zero
    - Use coefficients for prediction (model equation)
  - **R-squared**: goodness of fit
    - how much variance in the DV is explained by the model (here, containing only one continuous IV)

# Other useful queries

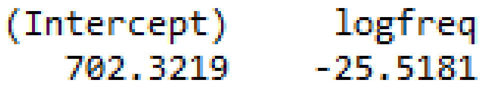
```
coefficients(LF2) # model coefficients (=> vector)
confint(LF2, level=0.95) # (95%) CIs for model parameters (=> matrix)
fitted(LF2); predict(LF2) # predicted values (=> vector)
residuals(LF2) # residuals (=> vector); observed - predicted values of DV
anova(LF2) # anova table (=> data frame)
vcov(LF2) # covariance matrix for model parameters (=> matrix)
influence(LF2) # regression diagnostics (=> vector)
```

```
# Example 1: 99% CIs on coefficients
> confint(LF2, level=0.99)
```



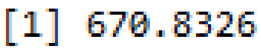
```
                0.5 %    99.5 %
(Intercept) 674.68895 729.95488
logfreq     -38.59557 -12.44062
```

```
# Example 2: Coefficients
> coefficients(LF2)
```



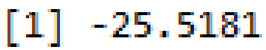
```
(Intercept)     logfreq 
   702.3219    -25.5181 
```

```
# Example 3: Predicted y for a hypothetical x, given a
# linear fit
Pred.y <- function(hypo.x, fit)
  unname(coefficients(fit)[1] +
           coefficients(fit)[2] * hypo.x)

> Pred.y(1.234, LF2)
```

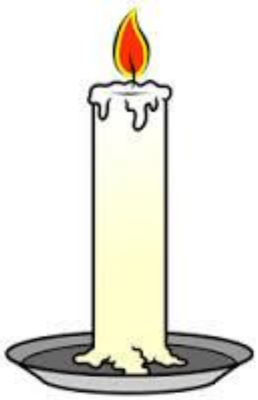


```
[1] 670.8326
```

```
> Pred.y(1, LF2) - Pred.y(0, LF2)
```



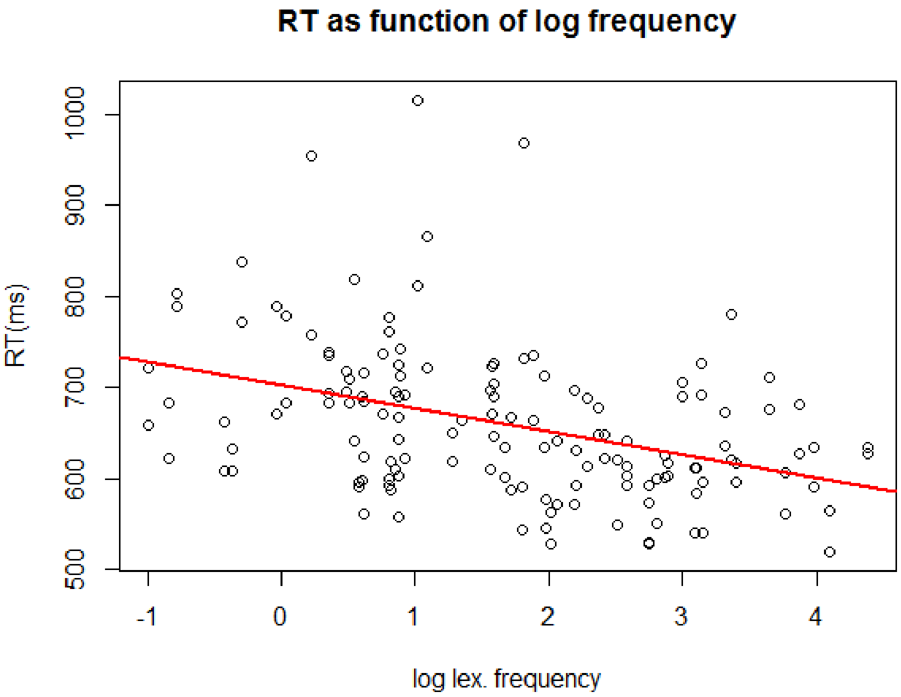
```
[1] -25.5181
```



Picture (makes this slide look nicer..)

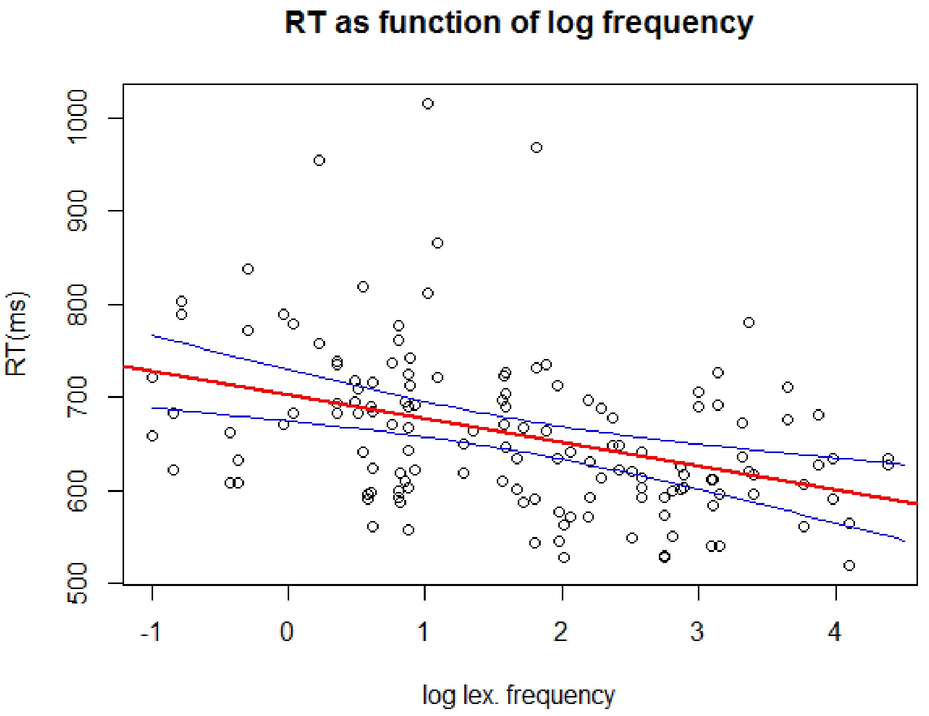
# Plot Regression Line

```
# Scatterplot with regression line,
# using previous linear fit LF2
# Plot y against x
x <- RT.data$logfreq
y <- RT.data$RT
plot (x, y,
      xlab = "log lex. frequency",
      ylab = "RT(ms)")
abline(LF2, col="red", lwd=2)
title("RT as function of log frequency")
```



# Plot Confidence Interval

```r
# Scatterplot with regression line,
# using previous linear fit LF2
# Plot y against x
x <- RT.data$logfreq
y <- RT.data$RT
plot (x, y,
      xlab = "log lex. frequency",
      ylab = "RT(ms)")
abline(LF2, col="red", lwd=2)
title("RT as function of log frequency")
```

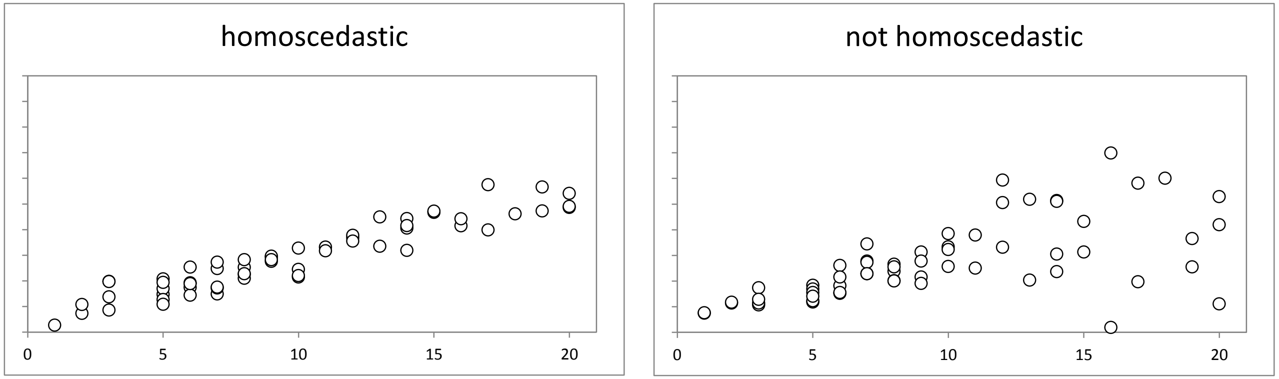


```r
# Generate new sequence of x-values
new.x <- seq(-1,4.5, by=0.05)

# Determine CI for each of the new x-values using predict()
# NB: Here, we are using 99% CIs
prd <- predict(LF2, newdata=data.frame(logfreq=new.x),
               interval="confidence", level=0.99)

# Add the lower and upper CI limits as lines to the plot
lines(new.x, prd[,2], col="blue", lty=1, lwd=1)
lines(new.x, prd[,3], col="blue", lty=1, lwd=1)
```

```
> head(prd)
       fit      lwr      upr
1 727.8400 689.0451 766.6350
2 726.5641 688.3561 764.7721
3 725.2882 687.6649 762.9115
4 724.0123 686.9715 761.0531
5 722.7364 686.2756 759.1972
6 721.4605 685.5771 757.3439
```

# Assumptions

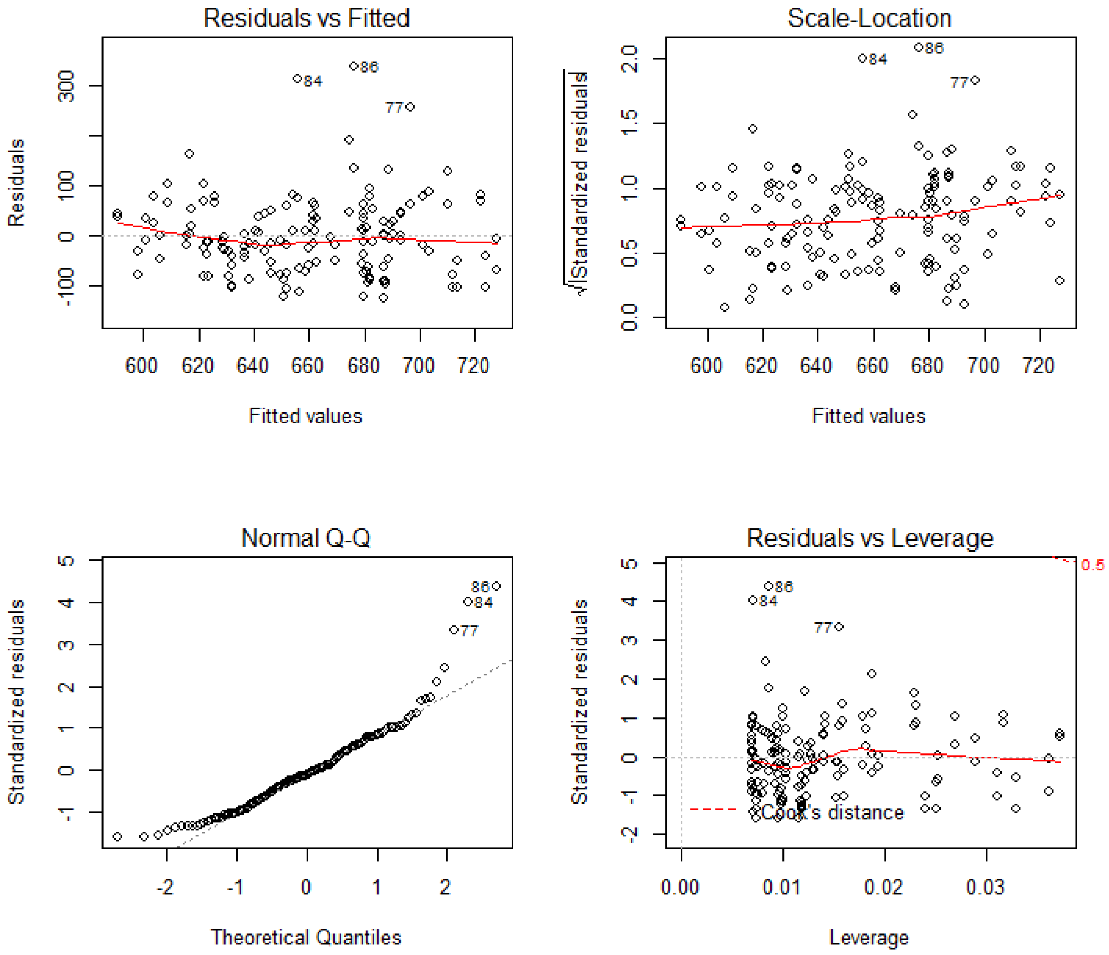
- Variables are measured on at least **interval scale** (continuous data)
  - Can theoretically range from −∞ to +∞
  - Exception: Categorical predictor variables (dummy-coding etc.)
- Linearity / additivity
- Homoscedasticity
  - Constant variance of residuals over the entire x-range, e.g.



- Normality of residuals
  - $e_i \sim N(0, \sigma)$

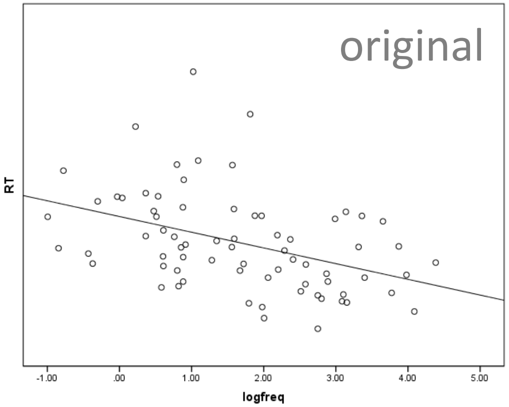
# Diagnostic Plots

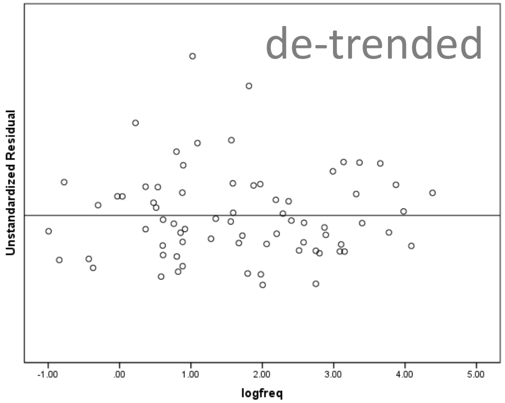
```
# diagnostic plots
# formatted as 2*2 matrix
layout(matrix(c(1,2,3,4),2,2))
plot(LF2)
```



# Summary

- Simple (bivariate) linear regression is a useful tool for prediction and ‘hypothetical forecasting’
  - E.g., what would be the most likely $\boldsymbol{y}$ for a very large $\boldsymbol{x}$ which I haven’t actually observed?
- Quality of prediction (→ confidence in predicted values) depends on how much variance is explained by the regression line
  - High $R^2$means good fit of the model to the data (reliable prediction)
- Other frequent use: *de-trending* of data (by subtracting $\hat{y}$s from the $y$s), e.g. to eliminate the influence of a “control variable”





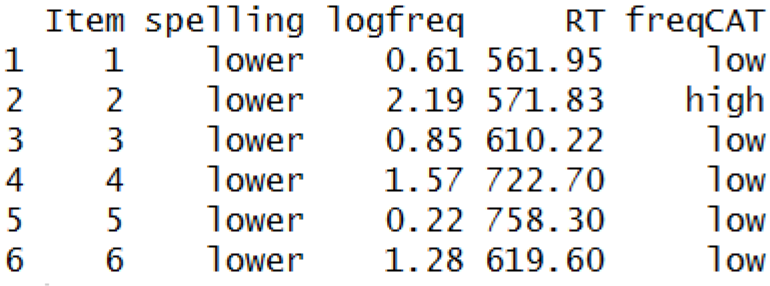
# What if our predictor *(x)* is categorical?

- That's actually no problem for regression
- Indeed, we shall see that (say) an independent measures **t-test is just *"regression in disguise"…***

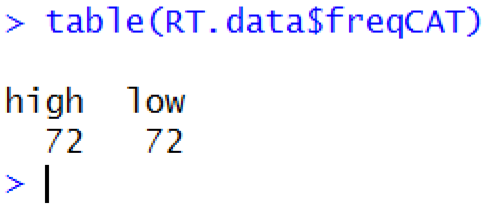
```r
# Simple regression example data
RT.data <- read.csv("http://www.psy.gla.ac.uk/~christop/MScStats/2018/Regress/RTs.csv")
```

- Let's **dichotomize** our *logfreq* variable by performing a median-split into *low-frequency* vs. *high-frequency* words

```r
# Median-split logfreq into two categories
RT.data$freqCAT <- ifelse(RT.data$logfreq > median(RT.data$logfreq),
                          "high", "low")
head(RT.data)
```



```
  Item spelling logfreq     RT freqCAT
1    1    lower    0.61 561.95     low
2    2    lower    2.19 571.83    high
3    3    lower    0.85 610.22     low
4    4    lower    1.57 722.70     low
5    5    lower    0.22 758.30     low
6    6    lower    1.28 619.60     low
```



```
> table(RT.data$freqCAT)

high  low 
  72   72 
> |
```

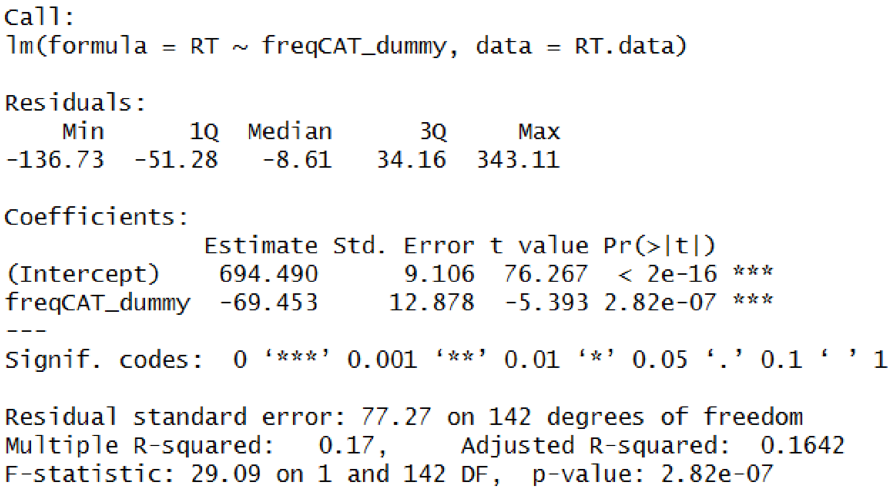
# Categorical Predictor

- Unfortunately, regression doesn’t actually work with character variables (“low” vs. “high”) as predictors
- Let’s use ***Dummy Coding*** instead (“low” is coded as **0** and “high” as **1**):

```
# "Dummy coding" of freqCAT
RT.data$freqCAT_dummy <- ifelse(RT.data$freqCAT == "low",0,1)
```

- And then perform a linear regression of RT as a function of *freqCAT_dummy*

```
# Perform linear regression (RT as a function of freqCAT_dummy)
catmod <- lm(RT ~ freqCAT_dummy, data = RT.data)
summary(catmod)
```



```
Call:
lm(formula = RT ~ freqCAT_dummy, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-136.73  -51.28   -8.61   34.16  343.11 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    694.490      9.106  76.267  < 2e-16 ***
freqCAT_dummy  -69.453     12.878  -5.393 2.82e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.27 on 142 degrees of freedom
Multiple R-squared:   0.17,	Adjusted R-squared:  0.1642 
F-statistic: 29.09 on 1 and 142 DF,  p-value: 2.82e-07
```

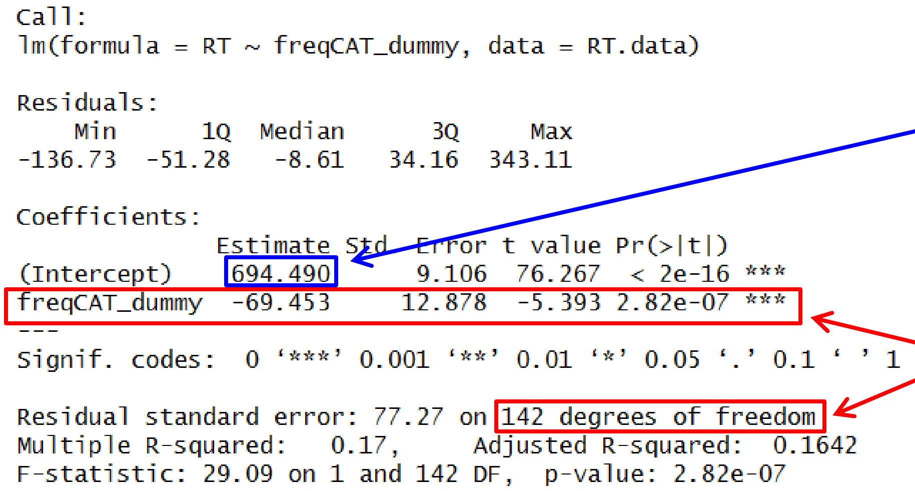
# Categorical Predictor

- Unfortunately, regression doesn’t actually work with character variables (“low” vs. “high”) as predictors
- Let’s use ***Dummy Coding*** instead (“low” is coded as **0** and “high” as **1**):

```
# "Dummy coding" of freqCAT
RT.data$freqCAT_dummy <- ifelse(RT.data$freqCAT == "low",0,1)
```

- And then perform a linear regression of *RT* as a function of *freqCAT_dummy*

```
# Perform linear regression (RT as a function of freqCAT_dummy)
catmod <- lm(RT ~ freqCAT_dummy, data = RT.data)
summary(catmod)
```



```
Call:
lm(formula = RT ~ freqCAT_dummy, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-136.73  -51.28   -8.61   34.16  343.11 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    694.490      9.106  76.267  < 2e-16 ***
freqCAT_dummy  -69.453     12.878  -5.393 2.82e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.27 on 142 degrees of freedom
Multiple R-squared:   0.17,	Adjusted R-squared:  0.1642 
F-statistic: 29.09 on 1 and 142 DF,  p-value: 2.82e-07
```

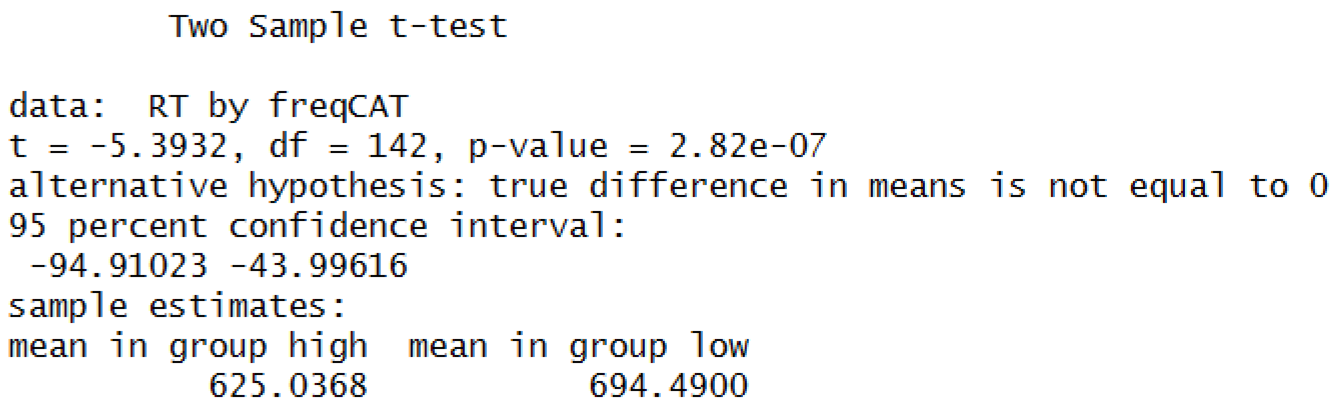
- The mean RT for “low frequency” words (x = 0) is 694.49 ms

- Overall, “high” frequency words (x = 1) are responded to 69.453 ms *faster* (negative slope) than “low” frequency words (x = 0)
- This effect is significant at ***t*(142) = -5.393**; ***p* = 0.0000000282**

# Categorical Predictor

- Alternatively, we could perform a t-test, and get the **exact same** results:

```
# The same as t-test (note: here we can use character variables)
t.test(RT ~ freqCAT, var.equal = TRUE, data = RT.data)
```



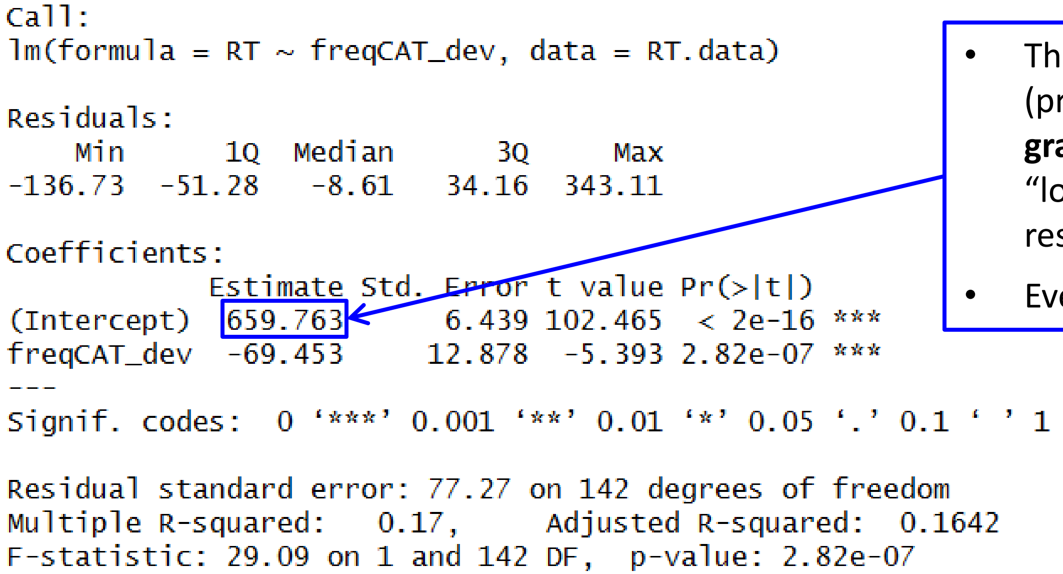
```
        Two Sample t-test

data:  RT by freqCAT
t = -5.3932, df = 142, p-value = 2.82e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -94.91023 -43.99616
sample estimates:
mean in group high  mean in group low 
          625.0368           694.4900 
```

- **The independent t-test (equal variance assumed) really is just regression in disguise!**

# Categorical Predictor

- Instead of *Dummy Coding* (0, 1), we could have used ***Deviation Coding* (-0.5, 0.5)** of our categorical predictor in `lm()`:

```
# "Deviation coding" of freqCAT
RT.data$freqCAT_dev <- ifelse(RT.data$freqCAT == "low",-0.5,0.5)

# Perform linear regression (RT as a function of freqCAT_dev)
catmod2 <- lm(RT ~ freqCAT_dev, data = RT.data)
summary(catmod2)
```



```
Call:
lm(formula = RT ~ freqCAT_dev, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-136.73  -51.28   -8.61   34.16  343.11 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  659.763      6.439 102.465  < 2e-16 ***
freqCAT_dev  -69.453     12.878  -5.393 2.82e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.27 on 142 degrees of freedom
Multiple R-squared:   0.17,	Adjusted R-squared:  0.1642 
F-statistic: 29.09 on 1 and 142 DF,  p-value: 2.82e-07
```

- The crucial difference is that the *intercept* (predicted RT at x = 0) now indexes the **grand average RT** rather the mean RT for “low” frequency words (cf. dummy coding results)!
- Everything else stays the same

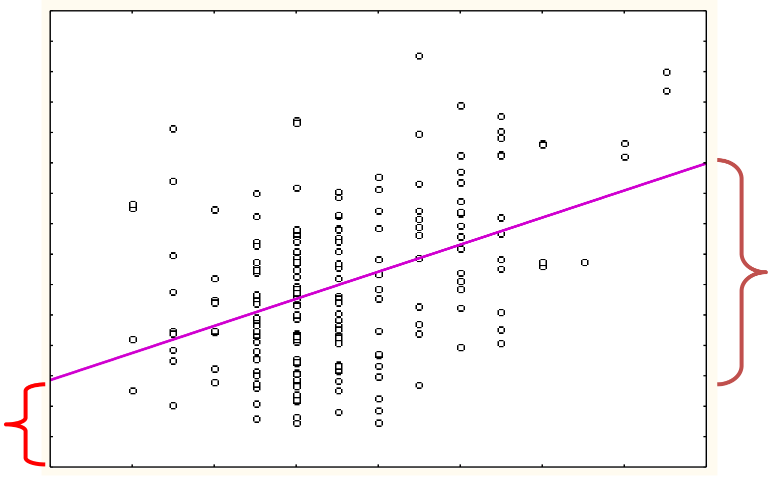
# Multiple Regression

- Now, we will extend the idea of linearly predicting a DV ($\boldsymbol{y}$) from a single IV ($\boldsymbol{x}$) to the case where we linearly predict $\boldsymbol{y}$ from a combination of *several* unrelated IVs ($\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_k}$); ***multiple linear regression***
- Various applications / goals / purposes:
  - Prediction
  - Confirmatory testing of individual predictors
    - E.g., amongst a range of theoretically relevant IVs, does a specific IV of interest make a significant contribution to the prediction of $y$?
    - Which IV is '*more important*' in predicting $y$?
  - Exploratory 'model selection'
    - Find a model that strikes an optimal compromise between number of IVs (the fewer the better – *Occam's razor*) and quality of prediction (high $R^2$) → $adjusted\ R^2$ for model comparison
    - Stepwise regression heuristics (*forward*, *backward,* etc.)

# Multiple Regression

### Simple (Bivariate) Linear Regression

one criterion
one predictor



"slope"($\beta_1$)

"intercept" or "constant" ($\beta_0$)

$$\hat{y} = \beta_0 + \beta_1 x \rightarrow y_i = \hat{y} + e_i \, , \, e_i \sim N(0, \sigma)$$

### Multiple Linear Regression

one criterion
*several* predictors

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

$$\rightarrow y_i = \hat{y} + e_i \, , \, e_i \sim N(0, \sigma)$$

# Multiple Regression

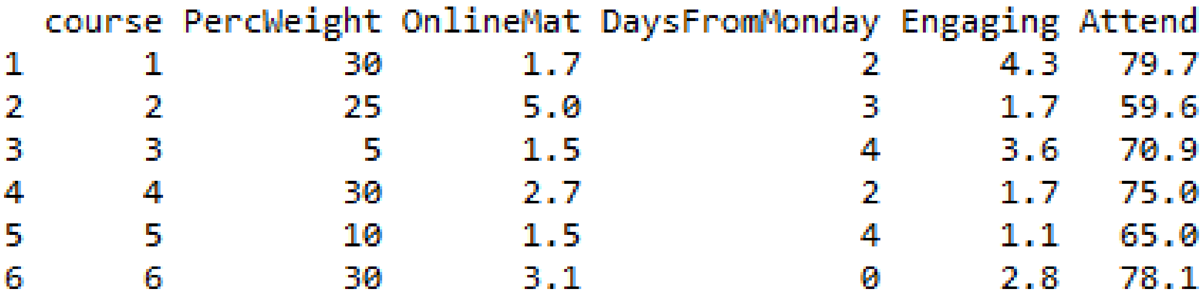
- The general purpose of multiple regression is to learn more about the linear relationship between several independent variables (predictors) and a single dependent variable (criterion).
- Multiple regression works much the same way as simple linear regression
- In the multivariate case (when there is more than one predictor), the regression line cannot be visualized in a two-dimensional space, but it can be computed just as easily (*a line in a k*+1-*dimensional space*, where k stands for the number of predictors)

# An Example

- **All imaginary!**
- A fictitious university is concerned about low class attendance by students. Based on available data (including student feedback etc.) from 40 courses held on campus last year, they try to determine which factors contribute to class attendance in what way.
- Class attendance is measured as the average percentage of students attending a given course in relation to the total number of students enrolled in that course.
- Of particular interest are four predictor variables:
  - How much the course contributes to the students' grade (variable *PercWeight*, ranging from 5% to 35%)
  - The quality of the online materials for the course, including lecture notes, podcasts etc. (variable *OnlineMat*, average student rating from 1 = "poor quality" to 5 = "excellent quality")
  - At what day of the week the course is held (variable *DaysFromMonday*: 0=Monday, 1=Tuesday, 2=Wednesday, 3=Thursday, 4=Friday)
  - How engaging the lecturer is (variable *Engaging*, average student rating from 1 = "very boring" to 5 = "very engaging")

# Let’s do some R

```
# Multiple regression example data
courses.data <- read.csv("http://www.psy.gla.ac.uk/~christop/MScStats/2018/Regress/Courses.csv")

> head(RT.data)
```

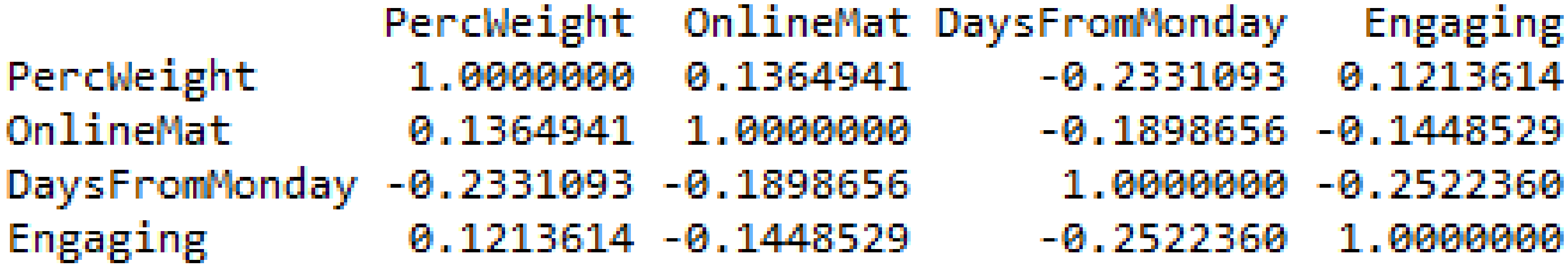


```
  course PercWeight OnlineMat DaysFromMonday Engaging Attend
1      1         30       1.7              2      4.3   79.7
2      2         25       5.0              3      1.7   59.6
3      3          5       1.5              4      3.6   70.9
4      4         30       2.7              2      1.7   75.0
5      5         10       1.5              4      1.1   65.0
6      6         30       3.1              0      2.8   78.1
```

# Multicollinearity

- Standard multiple regression actually assumes that predictor variables are independent from one another
- If predictors are strongly correlated, coefficient estimates become unreliable and difficult to interpret.
- A ‘quick and dirty’ way of testing this is by checking correlation matrices

```
# Correlation matrix considering only the predictors (variables 2-5 in original
# data-frame):
> cor(as.matrix(courses.data[2:5]))
```

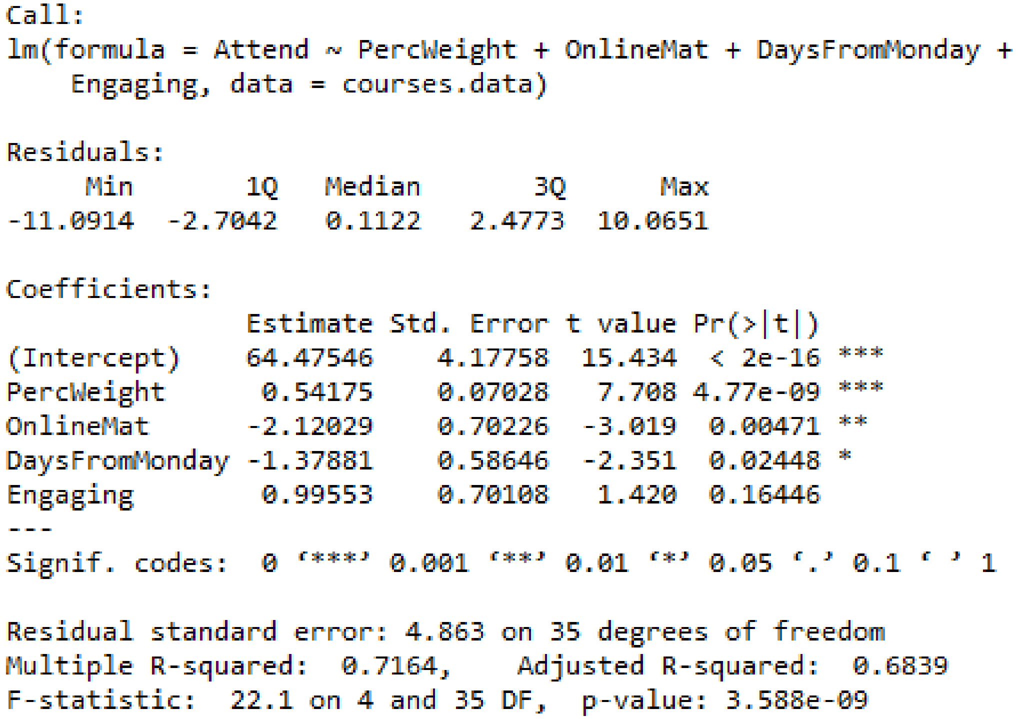


```
                PercWeight  OnlineMat DaysFromMonday   Engaging
PercWeight       1.0000000  0.1364941     -0.2331093  0.1213614
OnlineMat        0.1364941  1.0000000     -0.1898656 -0.1448529
DaysFromMonday  -0.2331093 -0.1898656      1.0000000 -0.2522360
Engaging         0.1213614 -0.1448529     -0.2522360  1.0000000
```

- No huge correlations – good!

# Output

```r
# Running multiple regression using lm()
courses.fit <- lm(Attend ~ PercWeight + OnlineMat + DaysFromMonday + Engaging,
                  data = courses.data)
summary(courses.fit)
```



```
Call:
lm(formula = Attend ~ PercWeight + OnlineMat + DaysFromMonday +
    Engaging, data = courses.data)

Residuals:
     Min       1Q   Median       3Q      Max 
-11.0914  -2.7042   0.1122   2.4773  10.0651 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    64.47546    4.17758  15.434  < 2e-16 ***
PercWeight      0.54175    0.07028   7.708 4.77e-09 ***
OnlineMat      -2.12029    0.70226  -3.019  0.00471 ** 
DaysFromMonday -1.37881    0.58646  -2.351  0.02448 *  
Engaging        0.99553    0.70108   1.420  0.16446    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.863 on 35 degrees of freedom
Multiple R-squared:  0.7164,	Adjusted R-squared:  0.6839 
F-statistic:  22.1 on 4 and 35 DF,  p-value: 3.588e-09
```

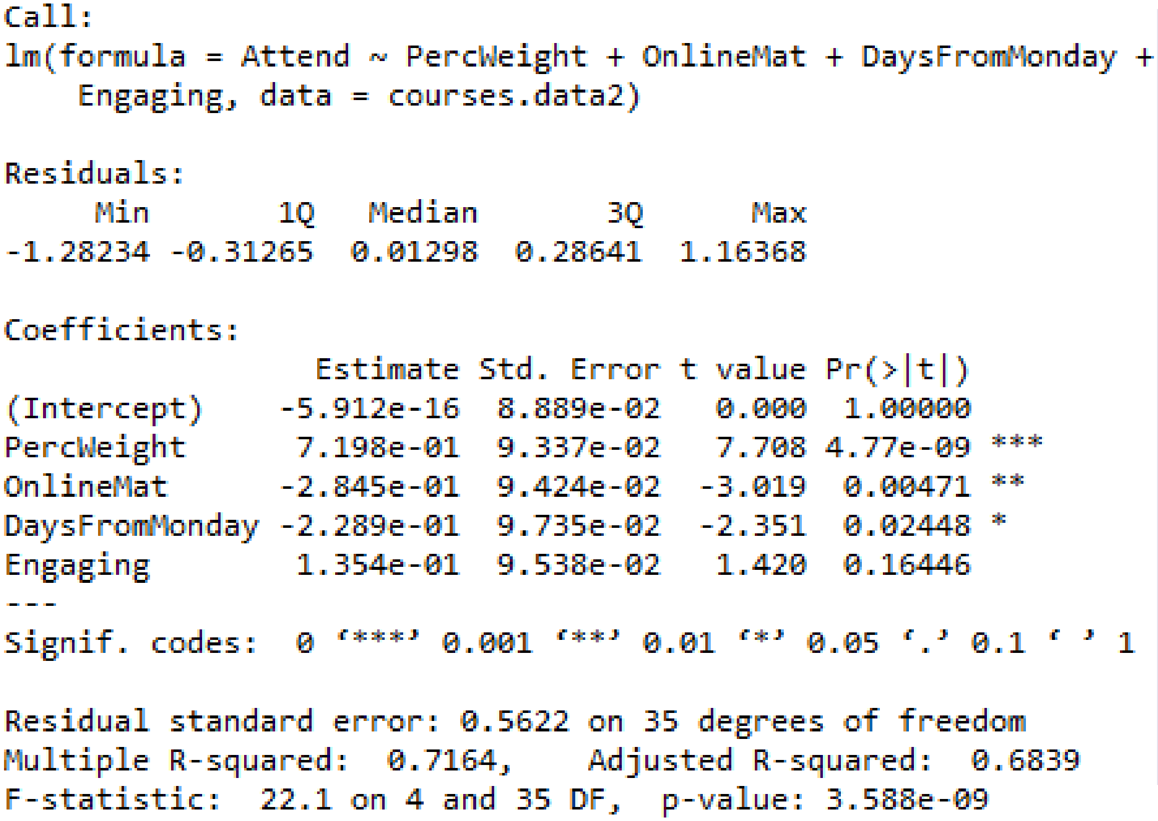
- **Thus:**
  - Attend = 64.5 + 0.54*PercWeight – 2.12*OnlineMat – 1.38*DaysFromMonday + 0.99*Engaging (+*e*)
  - Model explains (fake) data very well : R-squared = 0.72
  - The slope coefficient for Engaging is not significantly different from zero

# Standardized Coefficients (“betas”)

- Qualitatively, we can see that students attend more when
  - Courses contribute more to the final mark
  - Courses have poorer quality online materials
  - Courses take place closer to Monday than to Friday
  - Courses are given by more engaging lecturers (?)
- What about “relative importance” of predictors?
  - Perhaps easier to see when predictors and criterion are standardized (on the same scale; mean=0; SD = 1)
  - “Beta-coefficients”

# Standardized Coefficients (“betas”)

```
# Scale the data (by default, center=TRUE [subtract mean], and
# scale=TRUE [divide by SD]) and run lm() again
courses.data2 <- data.frame(scale(courses.data))
courses.fit2 <- lm(Attend ~ PercWeight + OnlineMat + DaysFromMonday + Engaging,
                   data = courses.data2)
summary(courses.fit2)
```



```
Call:
lm(formula = Attend ~ PercWeight + OnlineMat + DaysFromMonday +
    Engaging, data = courses.data2)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.28234 -0.31265  0.01298  0.28641  1.16368 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)    -5.912e-16  8.889e-02   0.000  1.00000    
PercWeight      7.198e-01  9.337e-02   7.708 4.77e-09 ***
OnlineMat      -2.845e-01  9.424e-02  -3.019  0.00471 ** 
DaysFromMonday -2.289e-01  9.735e-02  -2.351  0.02448 *  
Engaging        1.354e-01  9.538e-02   1.420  0.16446    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5622 on 35 degrees of freedom
Multiple R-squared:  0.7164,	Adjusted R-squared:  0.6839 
F-statistic:  22.1 on 4 and 35 DF,  p-value: 3.588e-09
```

- Note changes in parameter estimates & residuals; everything else (R-squared, t-tests, etc.) stays the same as before
- Use `coefficients(courses.fit2)` to extract only the parameters of interest
- Warning: Do not use these values for prediction (unless you want to predict everything on *SD*-unit scales)

# Standardized Coefficients (“betas”)

**Interpretation:**

- Attendance increases by 1 SD unit with

  a 0.72 *SD* increase in *PercWeight* +

  a 0.28 *SD* decrease in *OnlineMat* +

  a 0.23 *SD* decrease in *DaysFromMonday* +

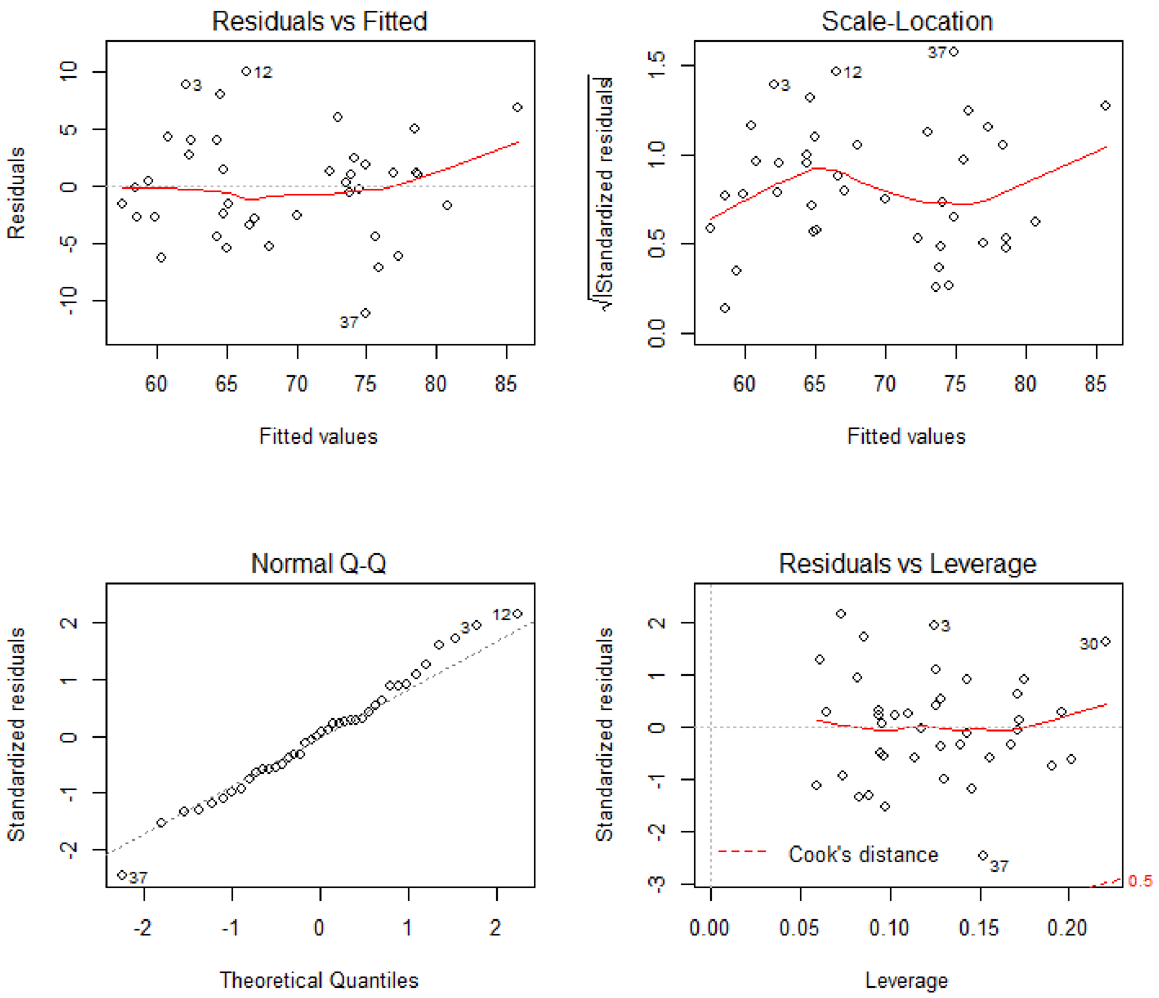
  a 0.14 *SD* decrease in *Engaging*

- *PercWeight* is clearly the most important predictor (well, that’s sort of evident from the t-statistics already…)
- **Caution:** Whether “betas” can be directly interpreted in terms of importance is debatable (better consider CIs for the betas).

# Diagnostics

**All fine** (more or less)

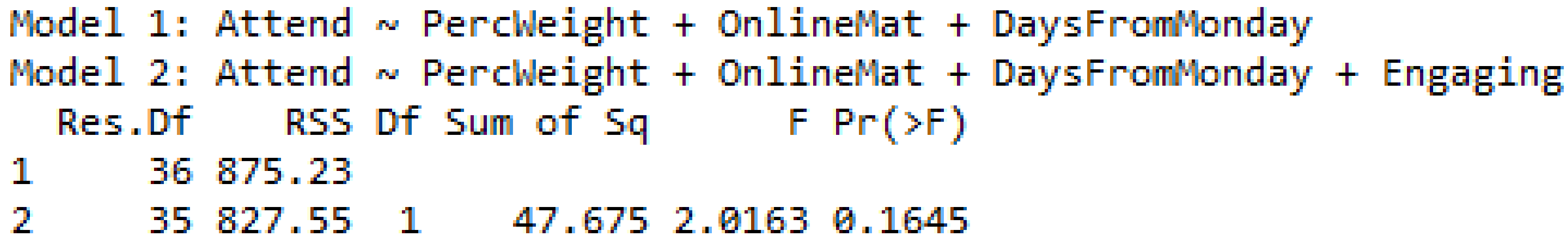
Fake data!



# Model Comparison

- E.g., do we really need to include *Engaging* to fit the current data accurately?

```
# Fit Model with and without “Engaging” and compare the models using anova()
full.fit <- lm(Attend ~ PercWeight + OnlineMat + DaysFromMonday + Engaging,
               data = courses.data)
noteng.fit <- lm(Attend ~ PercWeight + OnlineMat + DaysFromMonday,
                 data = courses.data)
anova(noteng.fit, full.fit)
```

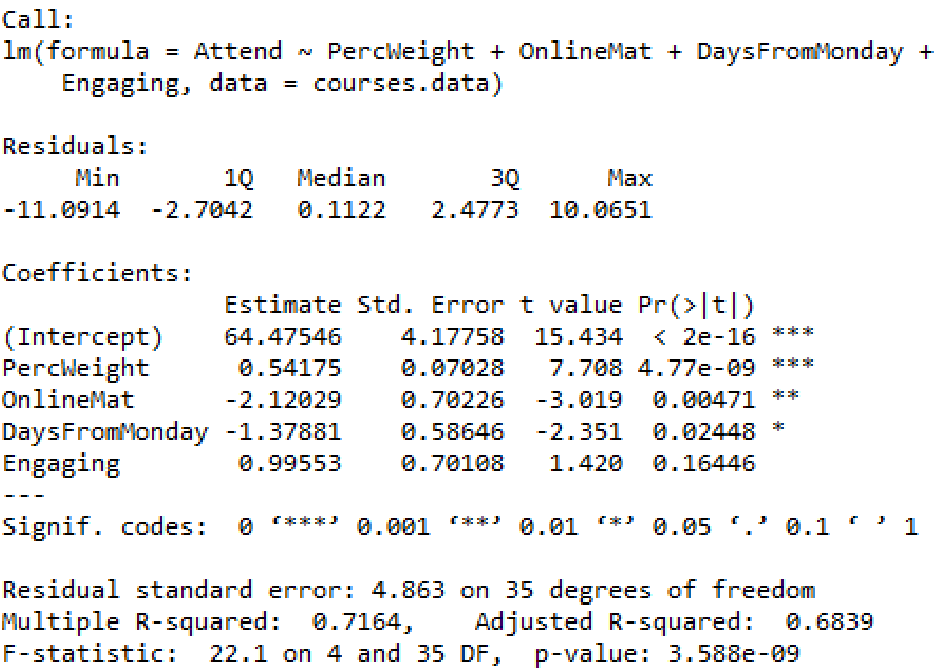


```
Model 1: Attend ~ PercWeight + OnlineMat + DaysFromMonday
Model 2: Attend ~ PercWeight + OnlineMat + DaysFromMonday + Engaging
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     36 875.23
2     35 827.55  1    47.675 2.0163 0.1645
```

- The full model does not seem to significantly improve on the model without *Engaging*

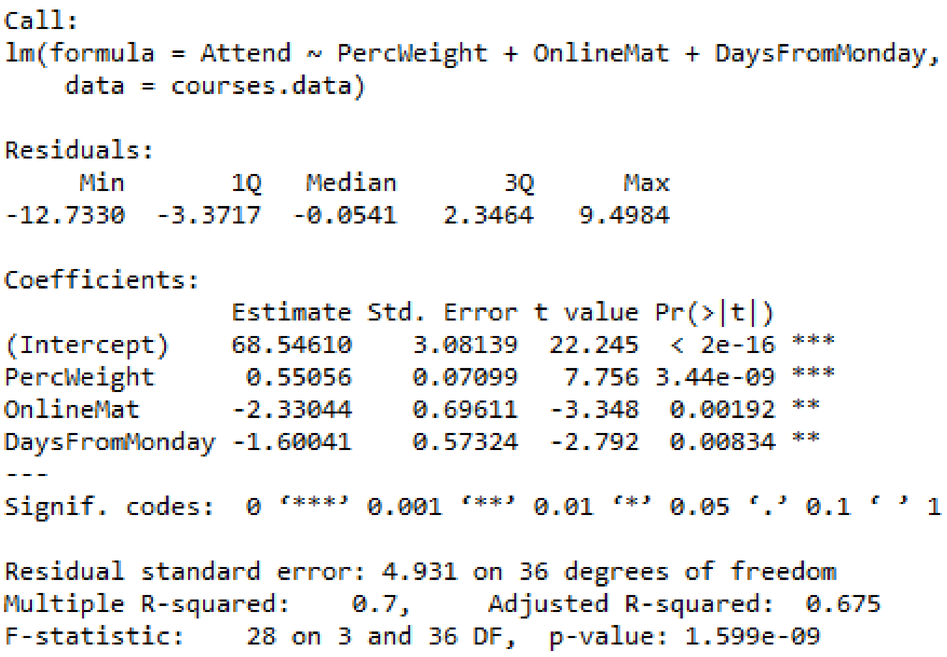
# Model Comparison (II)

```
> summary(full.fit)
```



```
Call:
lm(formula = Attend ~ PercWeight + OnlineMat + DaysFromMonday +
    Engaging, data = courses.data)

Residuals:
     Min       1Q   Median       3Q      Max
-11.0914  -2.7042   0.1122   2.4773  10.0651

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    64.47546    4.17758  15.434  < 2e-16 ***
PercWeight      0.54175    0.07028   7.708 4.77e-09 ***
OnlineMat      -2.12029    0.70226  -3.019  0.00471 **
DaysFromMonday -1.37881    0.58646  -2.351  0.02448 *
Engaging        0.99553    0.70108   1.420  0.16446
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.863 on 35 degrees of freedom
Multiple R-squared:  0.7164,	Adjusted R-squared:  0.6839
F-statistic:  22.1 on 4 and 35 DF,  p-value: 3.588e-09
```

```
> summary(noteng.fit)
```



```
Call:
lm(formula = Attend ~ PercWeight + OnlineMat + DaysFromMonday,
    data = courses.data)

Residuals:
     Min       1Q   Median       3Q      Max
-12.7330  -3.3717  -0.0541   2.3464   9.4984

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    68.54610    3.08139  22.245  < 2e-16 ***
PercWeight      0.55056    0.07099   7.756 3.44e-09 ***
OnlineMat      -2.33044    0.69611  -3.348  0.00192 **
DaysFromMonday -1.60041    0.57324  -2.792  0.00834 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.931 on 36 degrees of freedom
Multiple R-squared:    0.7,	Adjusted R-squared:  0.675
F-statistic:    28 on 3 and 36 DF,  p-value: 1.599e-09
```

- Things to note:
  - Coefficient estimates are not the same **(“context dependency” of estimates)**
  - If we use **Adjusted R-squared** as criterion (full: 0.684, noteng: 0.675), we’d probably better keep *Enagagement* in the model!
  - Adjusted R-squared = R-squared plus penalty for increasing number of model parameters:
    - See e.g. http://www.graphpad.com/guides/prism/6/curve-fitting/index.htm?reg_interpreting_the_adjusted_r2.htm

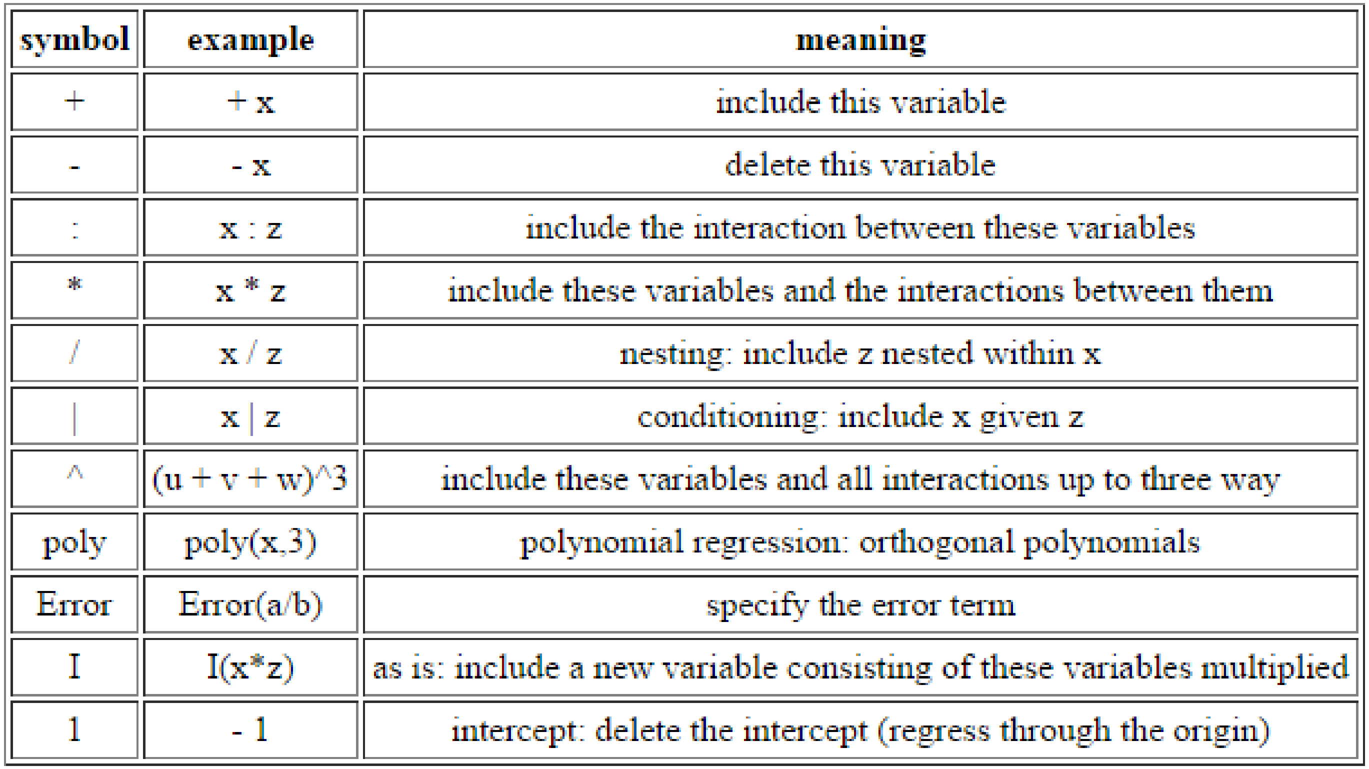
# Model Comparison: Discussion

- Sophisticated algorithms available to select the “best” model from a candidate set of predictors (“*stepwise*” *regression*)
- However, results are strongly dependent on the inclusion criteria used, direction of testing (forward/backward), ordering of effects etc.
- ***Hypothesis driven*** vs. ***data driven*** – which approach?
- Depends on your actual research goals
  - **Confirmatory (aim: generalisation)** If you wish to test hypotheses about a specific set of theoretically relevant predictors, test all predictors simultaneously (regardless of whether they ‘improve the fit’ or not)
  - **Exploratory (aim: hypothesis-generation)** If you have a lot of *potentially* relevant predictor variables for your current data, use a heuristic model selection approach to obtain a ‘parsimonious’ model of your data (aim: hypothesis-generation)
- Be clear about it!
  - Don’t try to ‘sell’ an exploratory analysis as confirmatory or vice versa

# More Complex Models

- The previous models only contained **continuous/categorical predictors as main effect terms**
- In fact, this is the typical use of multiple regression
- However, using the function `lm()` in R, you can actually specify and test more complex types of regression models (including interactions, polynomial relationships etc.)
    - All you need to do is to adjust the model formula in `lm()`, using appropriate syntax
- It is also possible to include categorical predictors
    - This requires **numerical coding** of the categorical predictor levels in a meaningful way

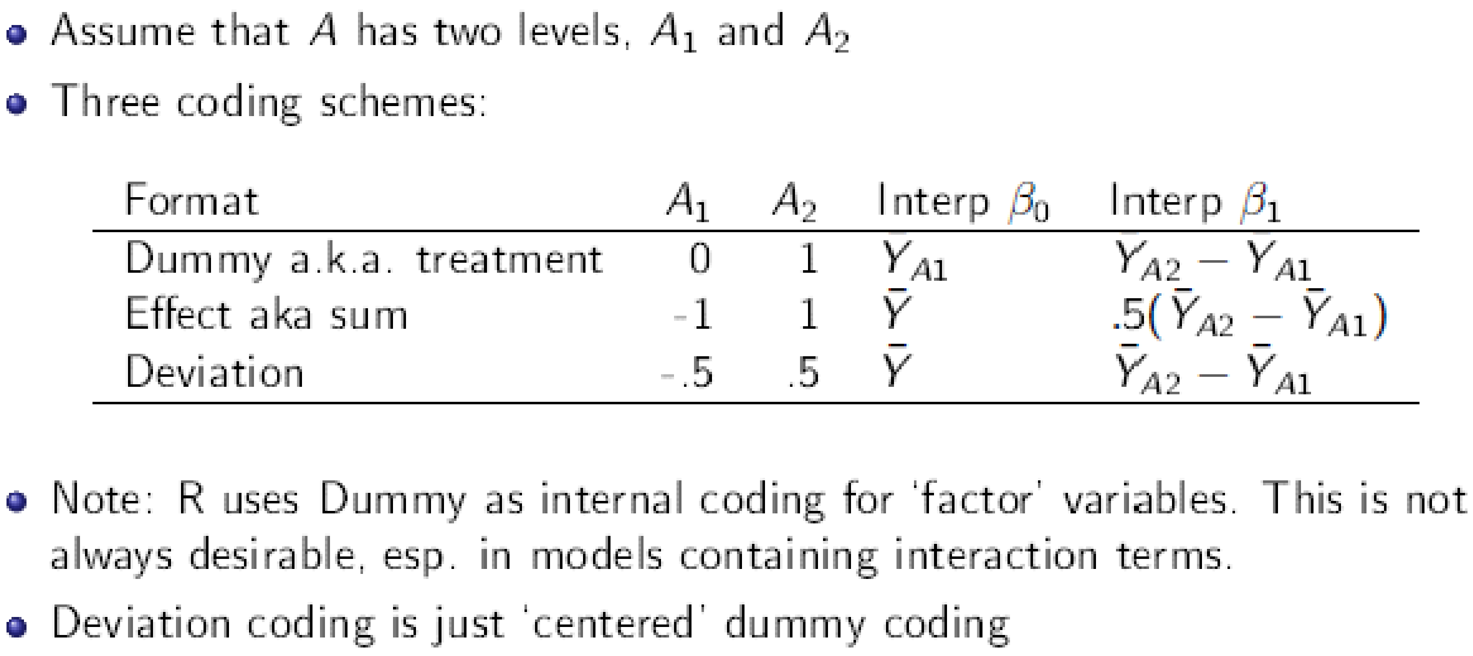
# Quick Tour: Formulae



| symbol | example | meaning |
|---|---|---|
| + | + x | include this variable |
| - | - x | delete this variable |
| : | x : z | include the interaction between these variables |
| * | x * z | include these variables and the interactions between them |
| / | x / z | nesting: include z nested within x |
| \| | x \| z | conditioning: include x given z |
| ^ | (u + v + w)^3 | include these variables and all interactions up to three way |
| poly | poly(x,3) | polynomial regression: orthogonal polynomials |
| Error | Error(a/b) | specify the error term |
| I | I(x*z) | as is: include a new variable consisting of these variables multiplied |
| 1 | - 1 | intercept: delete the intercept (regress through the origin) |

- Shamelessly stolen from [http://ww2.coastal.edu/kingw/statistics/R-tutorials/formulae.html](http://ww2.coastal.edu/kingw/statistics/R-tutorials/formulae.html)

# Quick Tour: Categorical Predictor Coding



- Assume that $A$ has two levels, $A_1$ and $A_2$
- Three coding schemes:

| Format | $A_1$ | $A_2$ | Interp $\beta_0$ | Interp $\beta_1$ |
|---|---|---|---|---|
| Dummy a.k.a. treatment | 0 | 1 | $\bar{Y}_{A1}$ | $\bar{Y}_{A2} - \bar{Y}_{A1}$ |
| Effect aka sum | -1 | 1 | $\bar{Y}$ | $.5(\bar{Y}_{A2} - \bar{Y}_{A1})$ |
| Deviation | -.5 | .5 | $\bar{Y}$ | $\bar{Y}_{A2} - \bar{Y}_{A1}$ |

- Note: R uses Dummy as internal coding for 'factor' variables. This is not always desirable, esp. in models containing interaction terms.
- Deviation coding is just 'centered' dummy coding

See also, e.g.:

https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/

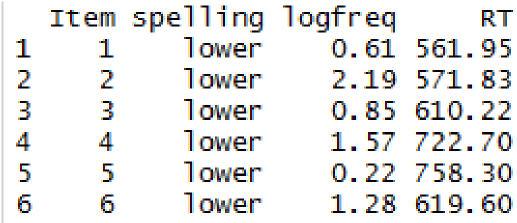
http://talklab.psy.gla.ac.uk/tvw/catpred/

# Back to our initial example…

- **Example data:** Lexical decision experiment (real data)
  - 144 words (and plenty of non-words as ‘fillers’, which are not included)
  - **Each word presented either in `UPPER` or `lower`case font (variable *spelling*)**
  - Task: decide as quickly and accurately as possible (button press) whether a given stimulus is an actual word
  - Also recorded for each word: lexical frequency (log10 per million word counts)
  - 33 subjects, but data are aggregated up to item level (not trial-by-trial data!)
- **New Questions:**
  - Does the spelling of the words (UPPER vs. lowercase) also have an influence on RT?
  - Do spelling and lexical frequency *interact* in producing different RTs?
    - Different slopes for lexical frequency dependent on levels of spelling

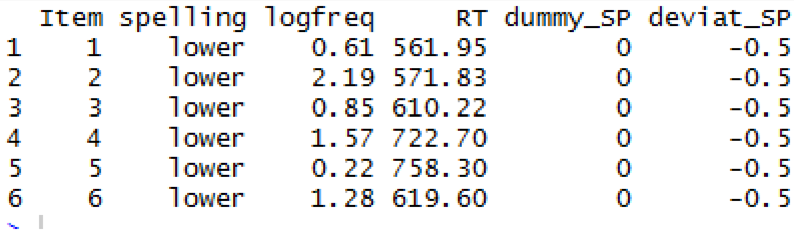
# Back to our initial example…

```r
# Initial example data
RT.data <- read.csv("http://www.psy.gla.ac.uk/~christop/MScStats/2018/Regress/RTs.csv")
head(RT.data)
```



```
  Item spelling logfreq     RT
1    1    lower    0.61 561.95
2    2    lower    2.19 571.83
3    3    lower    0.85 610.22
4    4    lower    1.57 722.70
5    5    lower    0.22 758.30
6    6    lower    1.28 619.60
```

```r
# Let's numerically code the variable 'spelling' first
RT.data$dummy_SP <- ifelse(RT.data$spelling=="lower",0,1) # dummy coding
RT.data$deviat_SP <- RT.data$dummy_SP - mean(RT.data$dummy_SP) # deviation coding
head(RT.data)
```



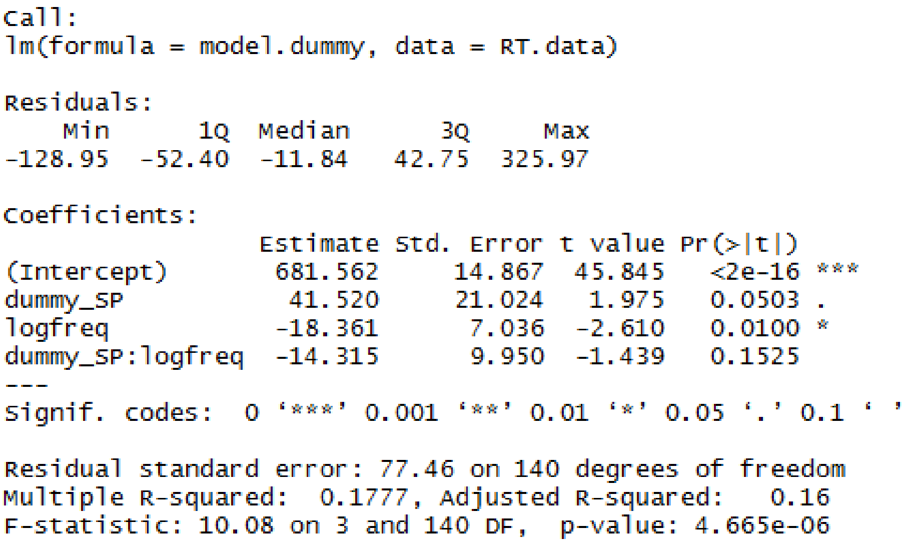
```
  Item spelling logfreq     RT dummy_SP deviat_SP
1    1    lower    0.61 561.95        0      -0.5
2    2    lower    2.19 571.83        0      -0.5
3    3    lower    0.85 610.22        0      -0.5
4    4    lower    1.57 722.70        0      -0.5
5    5    lower    0.22 758.30        0      -0.5
6    6    lower    1.28 619.60        0      -0.5
```

- `dummy_SP` takes the values 0 (for lowercase spelling) and 1 (for uppercase spelling)
- `deviat_SP` is coded as −0.5 (lowercase) and 0.5 (uppercase), respectively

# Let’s run `lm()` to address our new questions

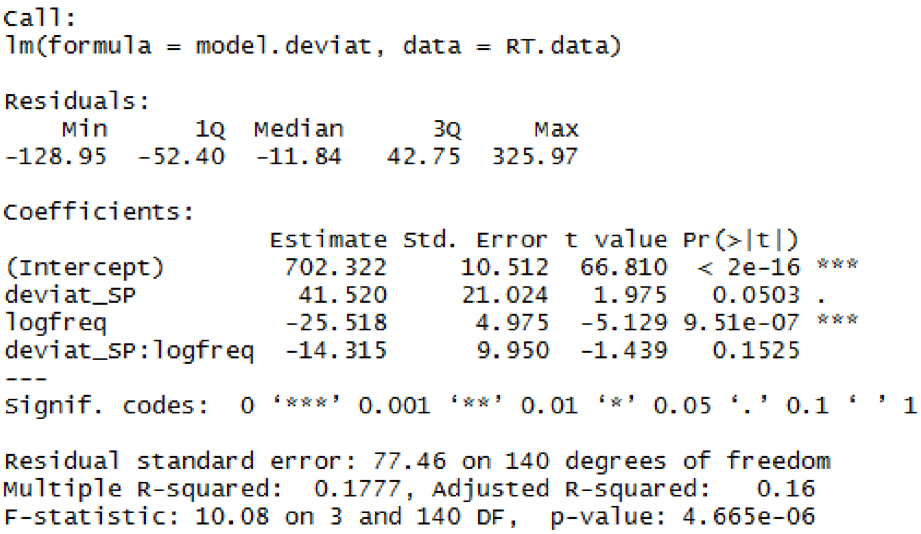
```
# “Outsourcing” model formulae:
model.dummy <- "RT ~ dummy_SP + logfreq + dummy_SP:logfreq"
model.deviat <- "RT ~ deviat_SP + logfreq + deviat_SP:logfreq"
# Fitting both models
fit.dummy <- lm(model.dummy, data=RT.data)
fit.deviat <- lm(model.deviat, data=RT.data)
```

```
> summary(fit.dummy)
```



```
Call:
lm(formula = model.dummy, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       681.562     14.867  45.845   <2e-16 ***
dummy_SP           41.520     21.024   1.975   0.0503 .  
logfreq           -18.361      7.036  -2.610   0.0100 *  
dummy_SP:logfreq  -14.315      9.950  -1.439   0.1525    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

```
> summary(fit.deviat)
```

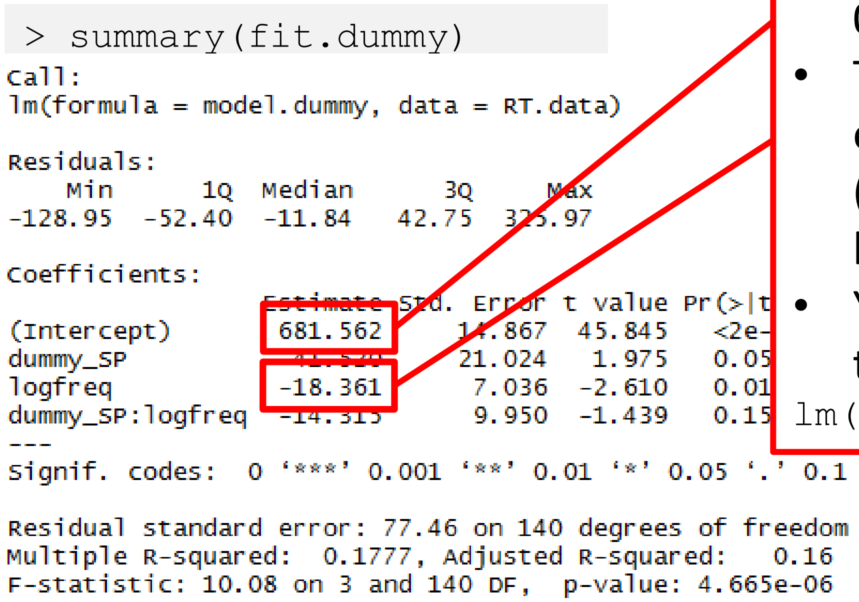


```
Call:
lm(formula = model.deviat, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        702.322     10.512  66.810  < 2e-16 ***
deviat_SP           41.520     21.024   1.975   0.0503 .  
logfreq            -25.518      4.975  -5.129 9.51e-07 ***
deviat_SP:logfreq  -14.315      9.950  -1.439   0.1525    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

- Effect of spelling (at *logfreq* = 0) is significant(*ish*): 42 ms higher RTs with UPPER than lowercase spelling
- Interaction is not significant (although estimate indicates that *logfreq* slope is 14 ms more negative with UPPER than lowercase spelling)
- **Importantly, notice differences in estimates for *intercept* and *logfreq* main effect!**

# Let's run `lm()` to address our new questions

```
# "Outsourcing" model formulae:
model.dummy <- "RT ~ dummy_SP + logfreq + dummy_SP:logfreq"
model.deviat <- "RT ~ deviat_SP + logfreq + deviat_SP:logfreq"
# Fitting both models
fit.dummy <- lm(model.dummy, data
fit.deviat <- lm(model.deviat, da
```

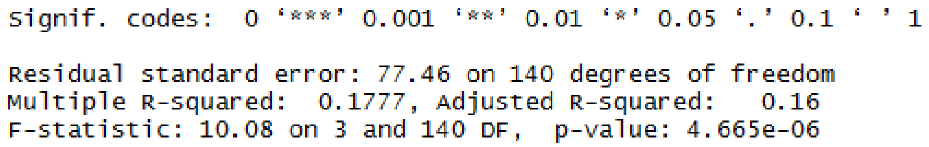


```
> summary(fit.dummy)
Call:
lm(formula = model.dummy, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                 Estimate Std. Error t value Pr(>|t
(Intercept)       681.562     14.867  45.845   <2e-
dummy_SP           41.530     21.024   1.975   0.05
logfreq           -18.361      7.036  -2.610   0.01
dummy_SP:logfreq  -14.315      9.950  -1.439   0.15
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' 

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

When the categorical predictor is dummy-coded (0,1):

- The *intercept* indexes the mean RT for *dummy_SP* = 0 (lowercase spelling) **and** *logfreq* = 0
- The *logfreq* "main effect" is actually not a main effect – it's the *logfreq* slope at dummy_SP = 0 (conceptually: the "simple effect" of *logfreq*, given lowercase spelling)
- You would obtain the same estimate (but not SE) for this coefficient if you ran

`lm(RT~logfreq, data=subset(RT.data, dummy_SP==0)`



```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

- Effect of spelling (at *logfreq* = 0) is significant(*ish*): 42 ms higher RTs with UPPER than lowercase spelling
- Interaction is not significant (although estimate indicates that *logfreq* slope is 14 ms more negative with UPPER than lowercase spelling)
- **Importantly, notice differences in estimates for *intercept* and *logfreq* main effect!**

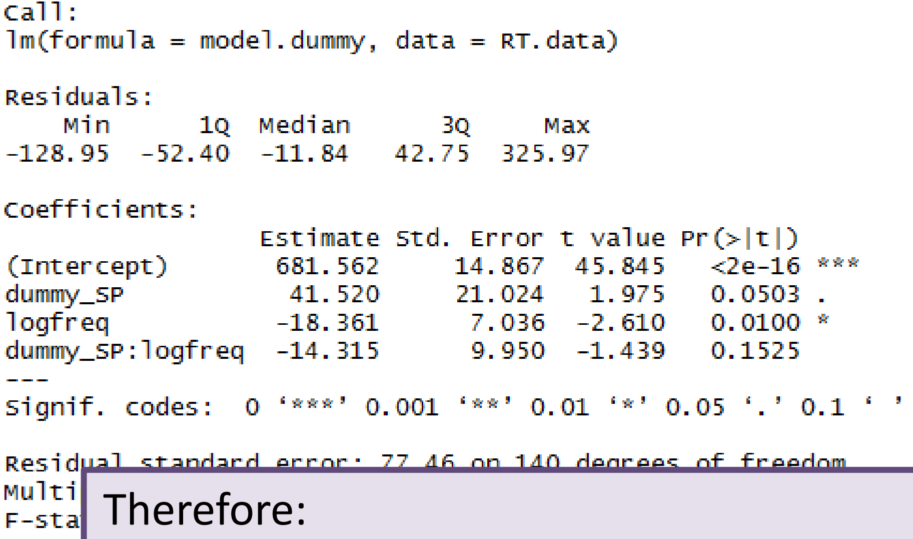
# Let’s run `lm()` to address our new questions

```
# “Outsourcing” model formulae:
_SP:logfreq"
iat_SP:logfreq"
```

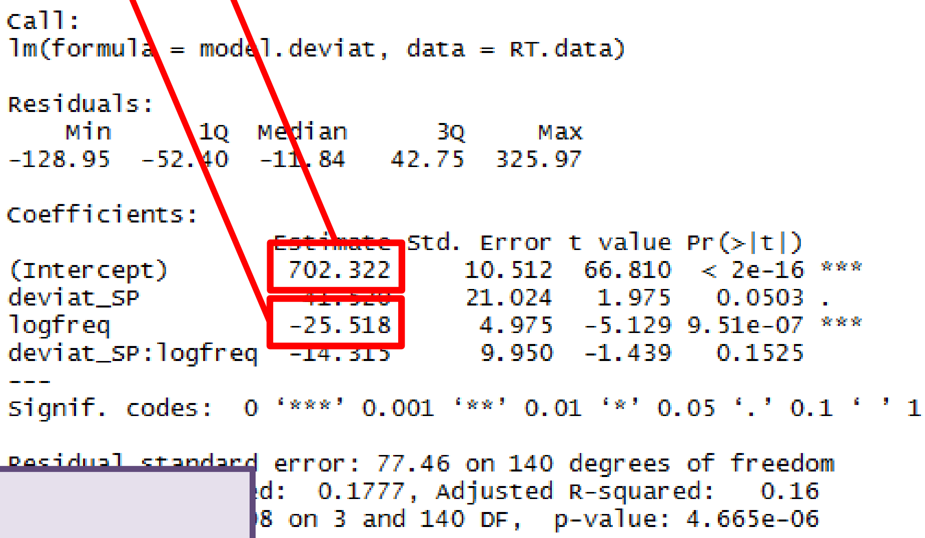
When the categorical predictor is deviation-coded (−0.5, 0.5):

- The *intercept* indexes the mean RT at *logfreq* = 0
- The *logfreq* effect is a proper main effect (overall slope associated with *logfreq*)

```
mary(fit.deviat)
```



```
Call:
lm(formula = model.dummy, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       681.562     14.867  45.845   <2e-16 ***
dummy_SP           41.520     21.024   1.975   0.0503 .  
logfreq           -18.361      7.036  -2.610   0.0100 *  
dummy_SP:logfreq  -14.315      9.950  -1.439   0.1525    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 77.46 on 140 degrees of freedom
Multi
F-sta
```



```
Call:
lm(formula = model.deviat, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        702.322     10.512  66.810  < 2e-16 ***
deviat_SP           41.520     21.024   1.975   0.0503 .  
logfreq            -25.518      4.975  -5.129 9.51e-07 ***
deviat_SP:logfreq  -14.315      9.950  -1.439   0.1525    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
d:  0.1777,	Adjusted R-squared:   0.16 
8 on 3 and 140 DF,  p-value: 4.665e-06
```

Therefore:

In designs including interactions:

- Use deviation coding to assess **main effects**
- Use dummy coding to assess **simple effects**

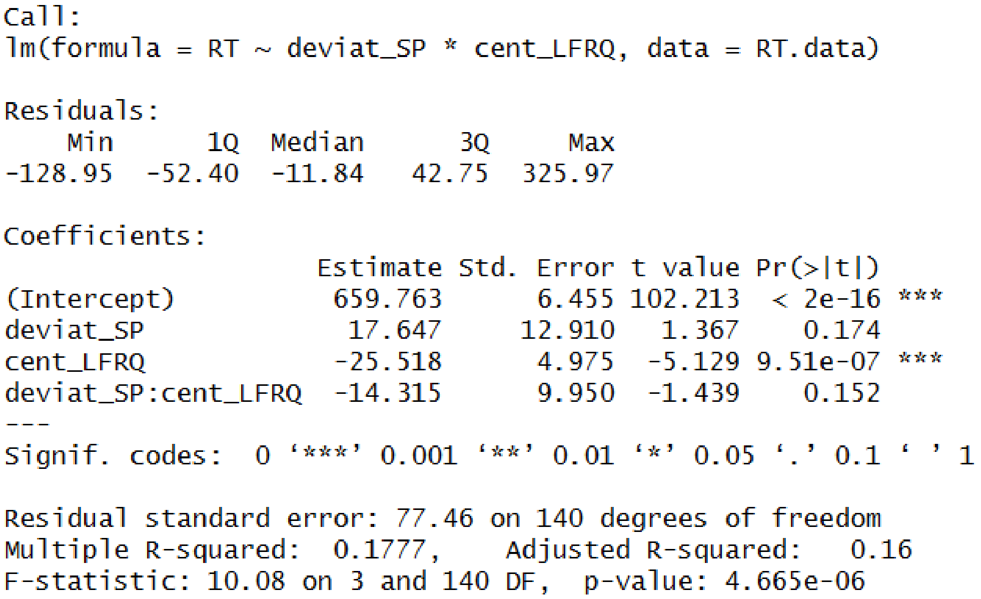
# Centring of Predictors

- Deviation coding of a categorical predictor is in fact “mean-centred” dummy coding of that predictor
- In most (if not all) applications, it makes sense to mean-centre *continuous* predictors as well!
- If all predictors are mean-centred, then
  - The model intercept indexes the grand average of the DV
  - “Main effect parameters” truly index main effects
- Dummy-coding is, however, useful to perform follow-up ‘simple effects’ analyses (see further down…)

# Lets do this again…

- **Mean-centre <u>all</u> the predictors** and run `lm()` again:

```
# "deviation coding" (mean-centred dummy coding) of spelling - as before
RT.data$deviat_SP <- scale(ifelse(RT.data$spelling=="lower",0,1), scale=FALSE)

# mean-centring of the continuous logfreq variable as well
RT.data$cent_LFRQ <- scale(RT.data$logfreq, scale = FALSE)

# Perform linear regression again
# Note: A*B in formula is just shorthand for A+B+A:B
centmod <- lm(RT ~ deviat_SP*cent_LFRQ, data = RT.data)
summary(centmod)
```



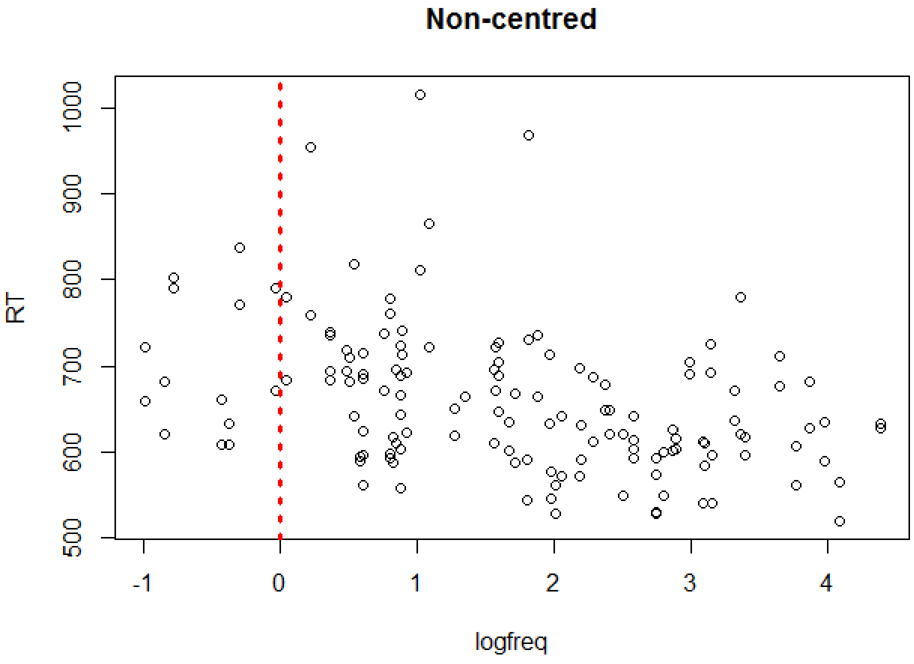
```
Call:
lm(formula = RT ~ deviat_SP * cent_LFRQ, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          659.763      6.455 102.213  < 2e-16 ***
deviat_SP             17.647     12.910   1.367    0.174    
cent_LFRQ            -25.518      4.975  -5.129 9.51e-07 ***
deviat_SP:cent_LFRQ  -14.315      9.950  -1.439    0.152    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

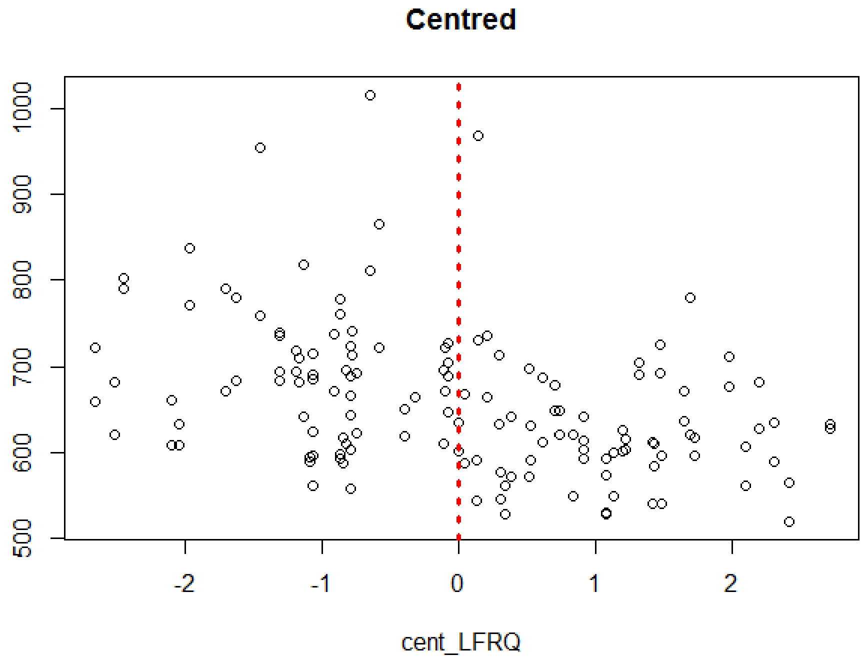
- Now the *intercept* is 659.763 ms (= grand average RT)!
- The effect of deviat_SP is a true **main effect** (estimated given *average word frequency*)
- The effect of cent_LFRQ is a true **main effect** (estimated given *average spelling*)

# Centring of *LogFreq*

```r
# Plot RT as a function of logfreq
plot(RT.data$RT ~ RT.data$logfreq,
     xlab = "logfreq", ylab = "RT",
     main = "Non-centred")
abline(v = 0, col="red", lwd=3, lty=3)
```

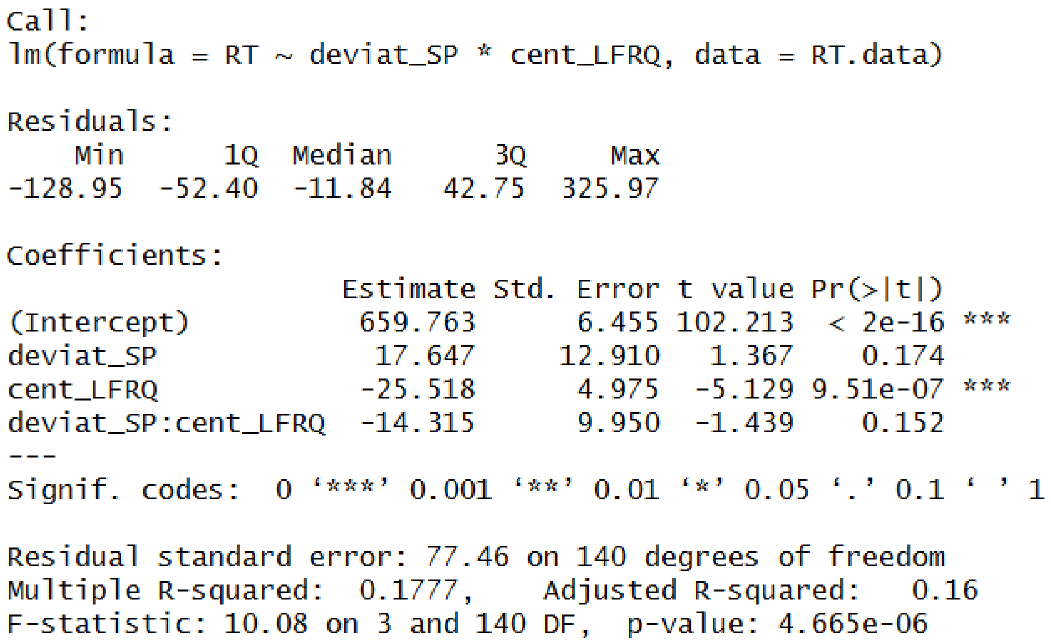
```r
# Plot RT as a function of cent_LFRQ
plot(RT.data$RT ~ RT.data$cent_LFRQ,
     xlab = "logfreq", ylab = "RT",
     main = “Centred")
abline(v = 0, col="red", lwd=3, lty=3)
```





# Prediction

- Given the output from the previous omnibus analysis (with mean-centred predictors), what would be the predicted mean RT for a word with, say,
  - a *cent_LFRQ* value of 2.0 and *lowercase* spelling ?



```
Call:
lm(formula = RT ~ deviat_SP * cent_LFRQ, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          659.763      6.455 102.213  < 2e-16 ***
deviat_SP             17.647     12.910   1.367    0.174    
cent_LFRQ            -25.518      4.975  -5.129 9.51e-07 ***
deviat_SP:cent_LFRQ  -14.315      9.950  -1.439    0.152    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

**Answer:**

659.763 (grand average) +

17.647 × (−.5) (lowercase spelling) +

−25.518 × 2.0 (cent_LFRQ = 2) +

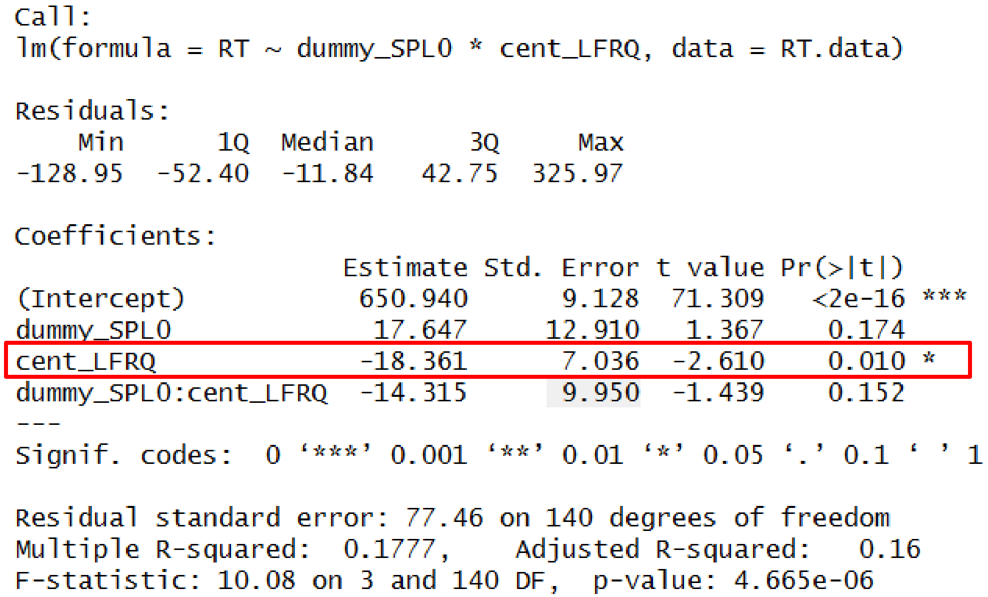
−14.315 × (−.5 × 2.0) (interaction)

= **650.219 ms**

# Simple effects (1)

- Let’s pretend the interaction (*deviat_SP:cent_LFRQ*) were significant, and we were interested in the ‘simple effect’ of *word frequency* at each level of *spelling*
- The way to find out would be to use **dummy coding** of the *spelling* predictor

```
# "dummy coding" of spelling - "lower" = 0
RT.data$dummy_SPL0 <- ifelse(RT.data$spelling=="lower",0,1)

# Perform linear regression again
SPL_mod <- lm(RT ~ dummy_SPL0*cent_LFRQ, data = RT.data)
summary(SPL_mod)
```

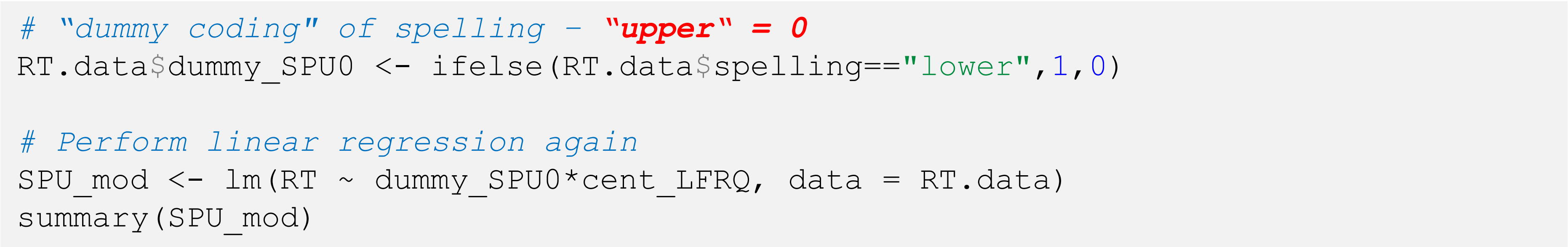


```
Call:
lm(formula = RT ~ dummy_SPL0 * cent_LFRQ, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)           650.940      9.128  71.309   <2e-16 ***
dummy_SPL0             17.647     12.910   1.367    0.174    
cent_LFRQ             -18.361      7.036  -2.610    0.010 *  
dummy_SPL0:cent_LFRQ  -14.315      9.950  -1.439    0.152    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

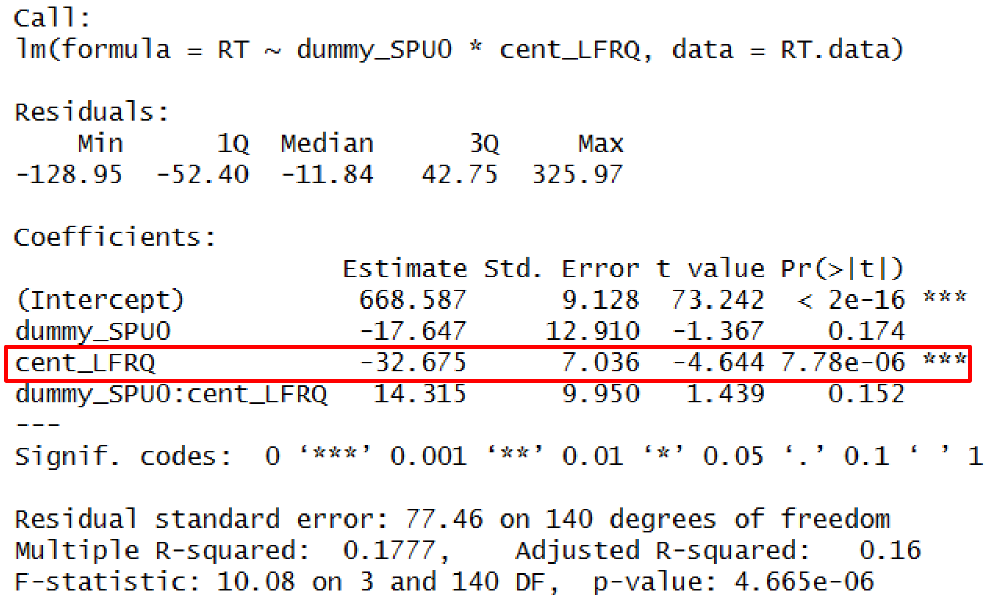
- With **lower case** spelling, every one-unit increase in *word frequency* leads to 18.361 ms faster RTs
- This simple effect is significant

# Simple effects (2)

- Let’s pretend the interaction (*deviat_SP:cent_LFRQ*) were significant, and we were interested in the ‘simple effect’ of *word frequency* at each level of *spelling*
- The way to find out would be to use **dummy coding** of the *spelling* predictor



```
# “dummy coding” of spelling – “upper“ = 0
RT.data$dummy_SPU0 <- ifelse(RT.data$spelling=="lower",1,0)

# Perform linear regression again
SPU_mod <- lm(RT ~ dummy_SPU0*cent_LFRQ, data = RT.data)
summary(SPU_mod)
```



```
Call:
lm(formula = RT ~ dummy_SPU0 * cent_LFRQ, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)           668.587      9.128  73.242  < 2e-16 ***
dummy_SPU0            -17.647     12.910  -1.367    0.174    
cent_LFRQ             -32.675      7.036  -4.644 7.78e-06 ***
dummy_SPU0:cent_LFRQ   14.315      9.950   1.439    0.152    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

- With **upper case** spelling, every one-unit increase in *word frequency* leads to 32.675 ms faster RTs
- This simple effect is also significant (even stronger than with lowercase spelling, but not reliably stronger)

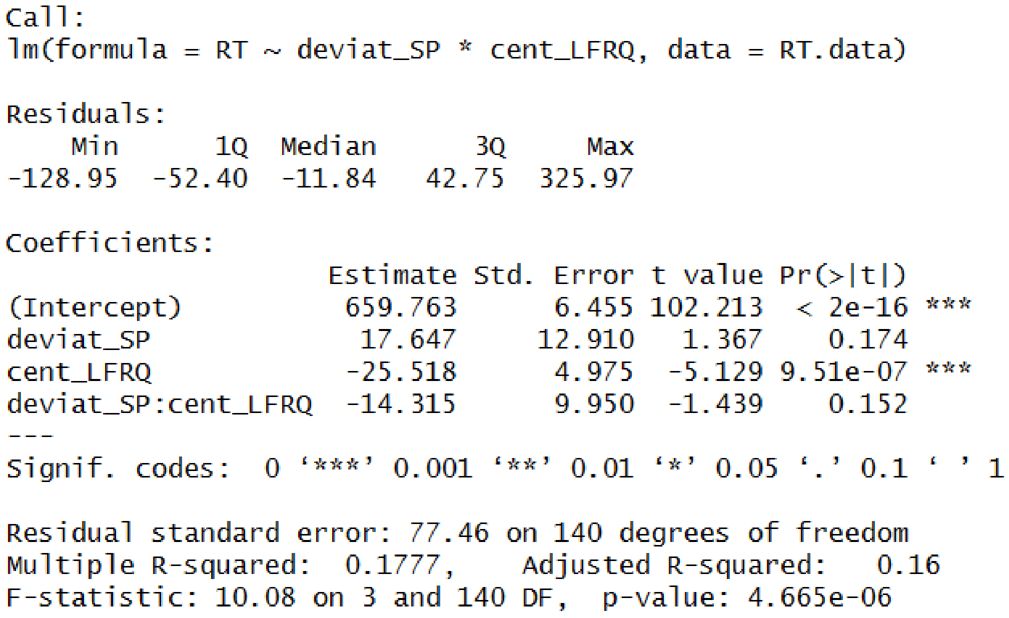
# Factorial designs: Good practise

- For **omnibus analysis**, *mean-centre your predictor variables* to establish main effects and interactions (2-way, 3-way etc.)
  - use deviation or sum coding for categorical predictors
  - mean-centre your continuous predictors as well!
- For **follow-up tests**, use *dummy-coding* (0,1) of a ‘conditioning predictor’ to decompose higher-order effects into simpler ones, e.g.
  - 3-way interactions into simple 2-way interactions
  - 2-way interactions into simple main effects
  - Etc.

# *F*-values instead of *t*-values

- We have seen that `lm()` can perform pretty much the same job as AN(C)OVA, including
  - **Categorical and continuous IVs**
  - **Main effects of, and interactions between IVs**
- However, the summary output only provides **t-tests on parameter estimates**, e.g.

```
# Fitting an lm() on our RT data (mean-centred predictors)
centmod <- lm(RT ~ deviat_SP * cent_LFRQ, data = RT.data)
summary(centmod)
```



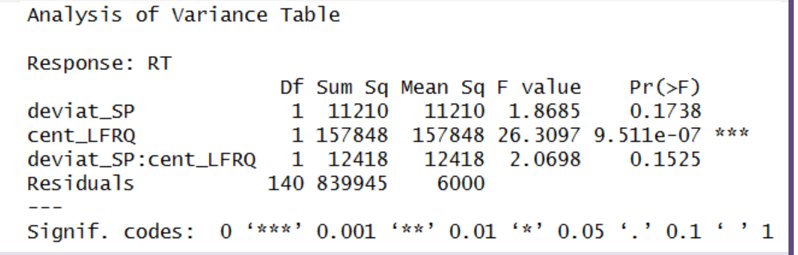
```
Call:
lm(formula = RT ~ deviat_SP * cent_LFRQ, data = RT.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-128.95  -52.40  -11.84   42.75  325.97 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          659.763      6.455 102.213  < 2e-16 ***
deviat_SP             17.647     12.910   1.367    0.174    
cent_LFRQ            -25.518      4.975  -5.129 9.51e-07 ***
deviat_SP:cent_LFRQ  -14.315      9.950  -1.439    0.152    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77.46 on 140 degrees of freedom
Multiple R-squared:  0.1777,	Adjusted R-squared:   0.16 
F-statistic: 10.08 on 3 and 140 DF,  p-value: 4.665e-06
```

Question:

**How can we get *F*-values for reporting?**

# `anova()` and `Anova()`

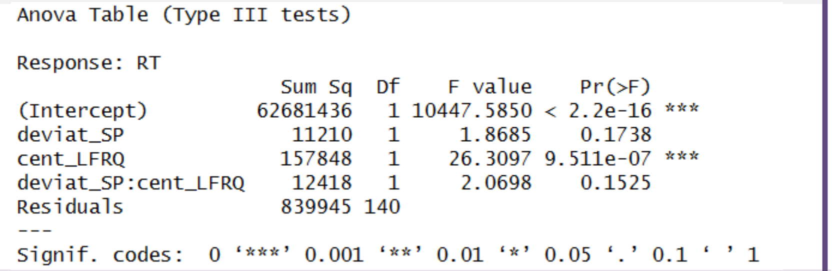
```r
# Using the base function anova()
anova(centmod)
```



```
Analysis of Variance Table

Response: RT
                   Df Sum Sq Mean Sq F value    Pr(>F)    
deviat_SP           1  11210   11210  1.8685    0.1738    
cent_LFRQ           1 157848  157848 26.3097 9.511e-07 ***
deviat_SP:cent_LFRQ   1  12418   12418  2.0698    0.1525    
Residuals         140 839945    6000                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **`anova()`** uses “sequential” **Type-I variance decomposition**
- Cumulative addition of effects (in the order specified by the model formula):
  - ***deviat_SP*** is tested on top of a model containing only intercept term ($\beta_0$)
  - ***cent_LFRQ*** is tested on top of $\beta_0 + \beta_1 \mathit{deviat\_SP}$
  - ***deviat_SP:cent_LRFQ*** is tested on top of $\beta_0 + \beta_1 \mathit{deviat\_SP} + \beta_2 \mathit{cent\_LFRQ}$
- No strictly simultaneous testing of effects!

```r
# Using the function Anova()
# (part of 'car' package)
library(car)
Anova(centmod, type="III")
```

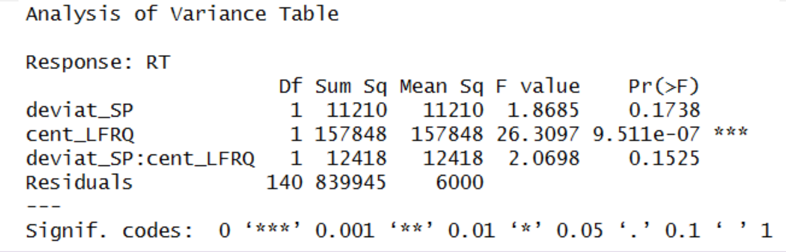


```
Anova Table (Type III tests)

Response: RT
                      Sum Sq  Df    F value    Pr(>F)    
(Intercept)         62681436   1 10447.5850 < 2.2e-16 ***
deviat_SP              11210   1     1.8685    0.1738    
cent_LFRQ             157848   1    26.3097 9.511e-07 ***
deviat_SP:cent_LFRQ    12418   1     2.0698    0.1525    
Residuals             839945 140                         
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **`Anova()`** is more flexible due to *type* argument
- Here: **Type-III variance decomposition**
- Simultaneous testing of effects:
  - ***deviat_SP*** is tested on top of $\beta_0 + \beta_1 \mathit{cent\_LFRQ} + \beta_2 \mathit{deviat\_SP}{:}\mathit{cent\_LFRQ}$
  - ***cent_LFRQ*** is tested on top of $\beta_0 + \beta_1 \mathit{deviat\_SP} + \beta_2 \mathit{deviat\_SP}{:}\mathit{cent\_LFRQ}$
  - ***deviat_SP:cent_LFRQ*** is tested on top of $\beta_0 + \beta_1 \mathit{deviat\_SP} + \beta_2 \mathit{cent\_LFRQ}$

# `anova()` and `Anova()`

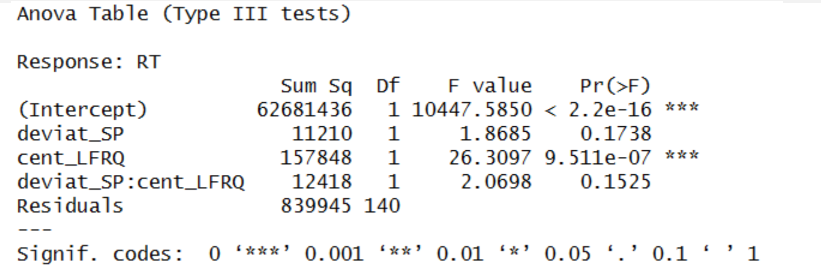
```r
# Using the base function anova()
anova(centmod)
```



```
Analysis of Variance Table

Response: RT
                    Df Sum Sq Mean Sq F value    Pr(>F)    
deviat_SP            1  11210   11210  1.8685    0.1738    
cent_LFRQ            1 157848  157848 26.3097 9.511e-07 ***
deviat_SP:cent_LFRQ  1  12418   12418  2.0698    0.1525    
Residuals          140 839945    6000                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **`anova()`** uses “sequential” Type-I variance decomposition

```r
# Using the function Anova()
# (part of 'car' package)
library(car)
Anova(centmod, type="III")
```



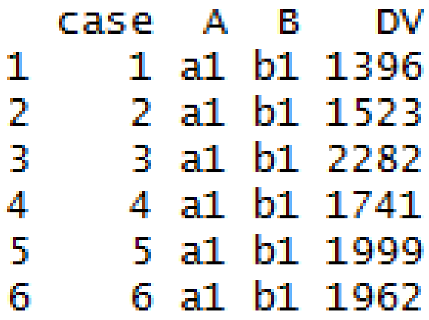
```
Anova Table (Type III tests)

Response: RT
                      Sum Sq  Df    F value    Pr(>F)    
(Intercept)         62681436   1 10447.5850 < 2.2e-16 ***
deviat_SP              11210   1     1.8685    0.1738    
cent_LFRQ             157848   1    26.3097 9.511e-07 ***
deviat_SP:cent_LFRQ    12418   1     2.0698    0.1525    
Residuals             839945 140                         
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Here, the two approaches make no difference (**balanced data!**)
- For unbalanced data, it could make a difference what kind of variance decomposition you are using
- For more info, see [https://www.r-bloggers.com/anova-%E2%80%93-type-iiiiii-ss-explained/](https://www.r-bloggers.com/anova-%E2%80%93-type-iiiiii-ss-explained/)
- **Type-III** variance decomposition is perhaps the most generalizable option (it’s also the default in stats packages such as SPSS, STATISTICA, SAS, etc.)

# Categorical predictors with ***more than two levels***

- Can be handled using a regression approach (e.g., `lm()`) as well, but it's a bit more tricky than with 2-level predictors
- **Let's try an example with**
  - Two categorical predictors: A={a1, a2, a3}; B={b1, b2}; **a 3×2 design**
  - Continuous DV (ranging from 344 to 2934)
  - 90 cases

```
# The data:
dat <- read.csv("http://www.psy.gla.ac.uk/~christop/MScStats/2018/Regress/threeleveldata.csv")
head(dat)
```



```
  case  A  B   DV
1    1 a1 b1 1396
2    2 a1 b1 1523
3    3 a1 b1 2282
4    4 a1 b1 1741
5    5 a1 b1 1999
6    6 a1 b1 1962
```

# Categorical predictors with more than two levels

- **General rule for coding**: given *k* levels of a categorical predictor, you'll need *k*−1 coding variables for that predictor in the regression model; here:
  - 3−1 = 2 coding variables for predictor A
  - 2−1 = 1 coding variable for predictor B

```
# Deviation coding of both predictors:
dat$a1a2 <- scale(ifelse(dat$A=="a2",1,0), center=TRUE, scale=FALSE)
dat$a1a3 <- scale(ifelse(dat$A=="a3",1,0), center=TRUE, scale=FALSE)
dat$Bdv <- scale(ifelse(dat$B=="b2",1,0), center=TRUE, scale=FALSE)
```

Resulting coding:

| *A* | *a1a2* | *a1a3* | Σ |
|---|---|---|---|
| *a1* | −0.33 | −0.33 | −0.66 |
| *a2* | 0.66 | −0.33 | 0.33 |
| *a3* | −0.33 | 0.66 | 0.33 |

| *B* | *Bdv* |
|---|---|
| *b1* | −0.5 |
| *b2* | 0.5 |

# Categorical predictors with more than two levels

- **General rule for coding**: given *k* levels of a categorical predictor, you'll need *k*−1 coding variables for that predictor in the regression model; here:
    - 3−1 = 2 coding variables for predictor A
    - 2−1 = 1 coding variable for predictor B

```r
# Deviation coding of both predictors:
dat$a1a2 <- scale(ifelse(dat$A=="a2",1,0), center=TRUE, scale=FALSE)
dat$a1a3 <- scale(ifelse(dat$A=="a3",1,0), center=TRUE, scale=FALSE)
dat$Bdv <- scale(ifelse(dat$B=="b2",1,0), center=TRUE, scale=FALSE)
```

Resulting coding:

| *A* | *a1a2* | *a1a3* | *Σ* | |
|---|---|---|---|---|
| *a1* | −0.33 | −0.33 | −0.66 | **Reference category** |
| *a2* | 0.66 | −0.33 | 0.33 | |
| *a3* | −0.33 | 0.66 | 0.33 | |

| *B* | *Bdv* |
|---|---|
| *b1* | −0.5 |
| *b2* | 0.5 |

# Categorical predictors with more than two levels

- **General rule for coding**: given *k* levels of a categorical predictor, you'll need *k*−1 coding variables for that predictor in the regression model; here:
  - 3−1 = 2 coding variables for predictor A
  - 2−1 = 1 coding variable for predictor B

```
# Deviation coding of both predictors:
dat$a1a2 <- scale(ifelse(dat$A=="a2",1,0), center=TRUE, scale=FALSE)
dat$a1a3 <- scale(ifelse(dat$A=="a3",1,0), center=TRUE, scale=FALSE)
dat$Bdv <- scale(ifelse(dat$B=="b2",1,0), center=TRUE, scale=FALSE)
```

Resulting coding:

| *A* | *a1a2* | *a1a3* | Σ |
|---|---|---|---|
| *a1* | −0.33 | −0.33 | −0.66 |
| *a2* | 0.66 | −0.33 | 0.33 |
| *a3* | −0.33 | 0.66 | 0.33 |

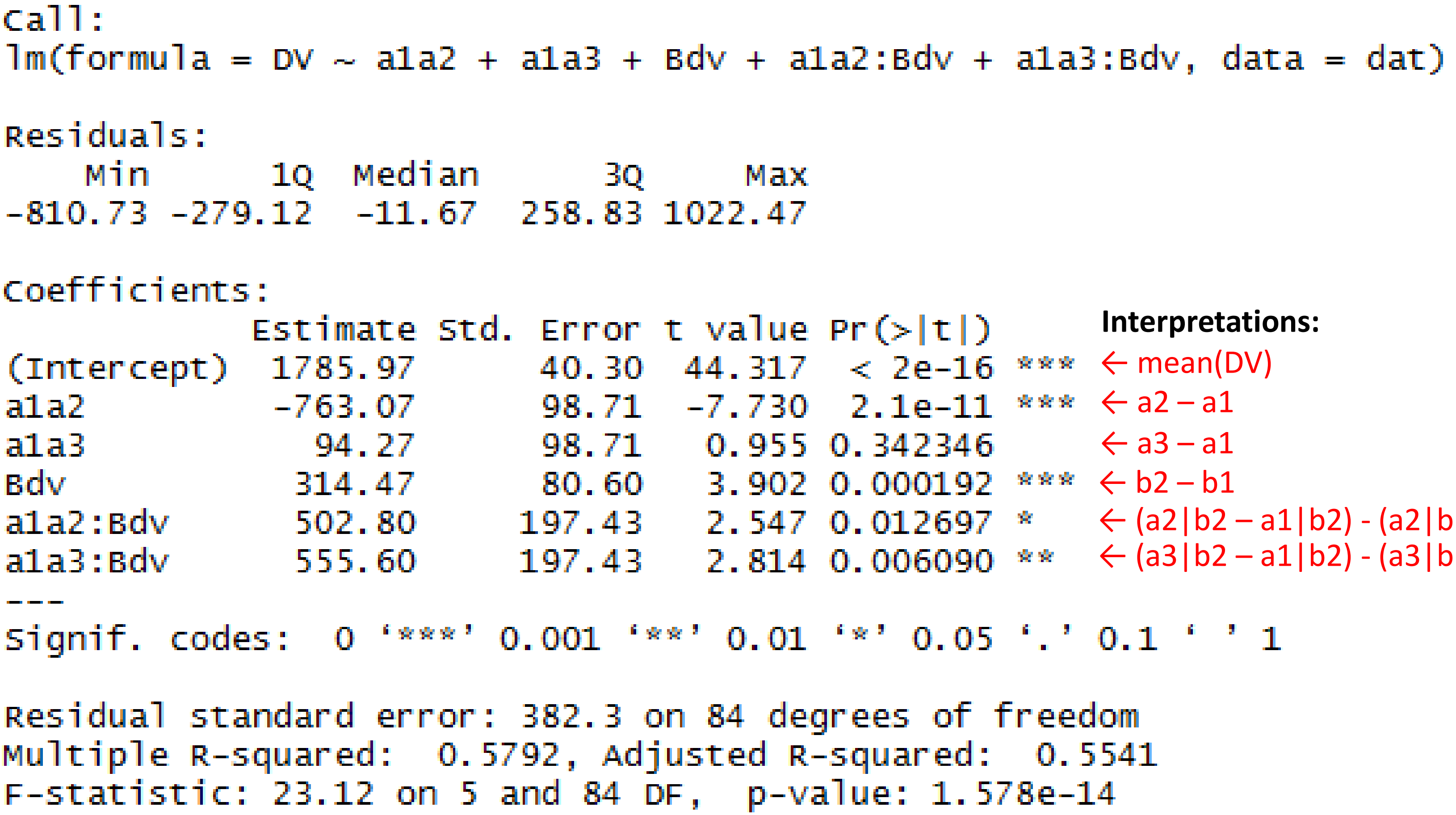
Reference category

$\overline{a2} - \overline{a1}$ $\overline{a3} - \overline{a1}$

| *B* | *Bdv* |
|---|---|
| *b1* | −0.5 |
| *b2* | 0.5 |

# Categorical predictors with more than two levels

```
# Determine the linear fit (including all main effects and interactions):
fit <- lm(DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv, data=dat)
summary(fit)
```



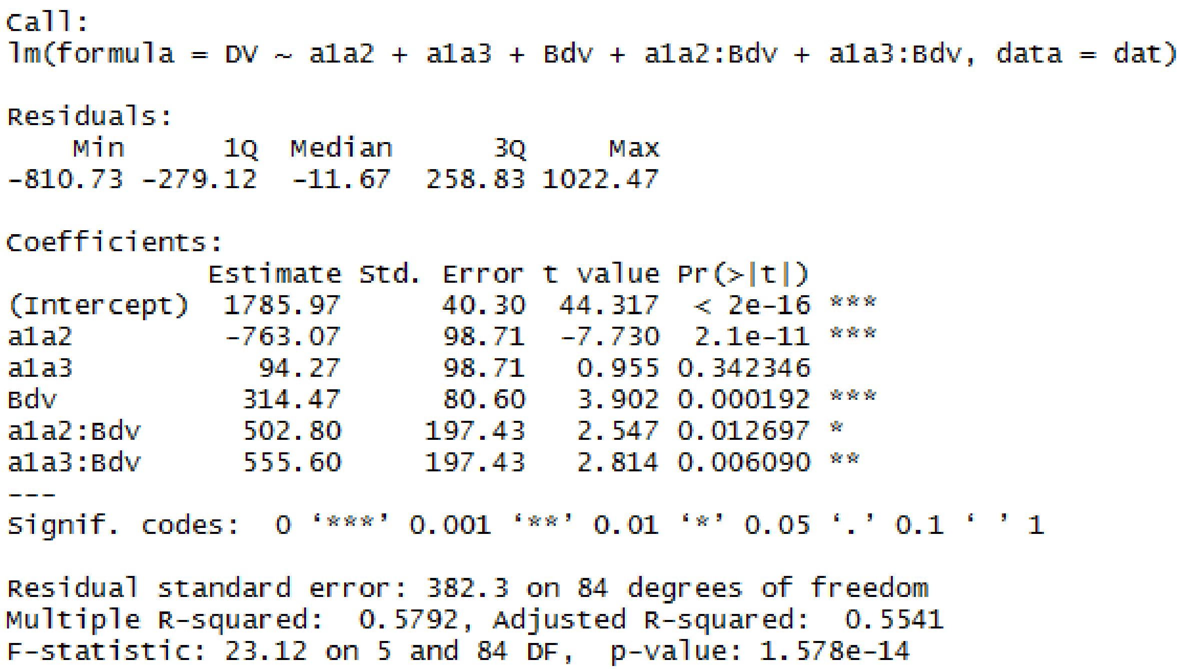
```
Call:
lm(formula = DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-810.73 -279.12  -11.67  258.83 1022.47 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1785.97      40.30  44.317  < 2e-16 ***
a1a2         -763.07      98.71  -7.730  2.1e-11 ***
a1a3           94.27      98.71   0.955 0.342346    
Bdv           314.47      80.60   3.902 0.000192 ***
a1a2:Bdv      502.80     197.43   2.547 0.012697 *  
a1a3:Bdv      555.60     197.43   2.814 0.006090 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 382.3 on 84 degrees of freedom
Multiple R-squared:  0.5792,	Adjusted R-squared:  0.5541 
F-statistic: 23.12 on 5 and 84 DF,  p-value: 1.578e-14
```

**Interpretations:**

← mean(DV)
← a2 – a1
← a3 – a1
← b2 – b1
← (a2|b2 – a1|b2) - (a2|b1 – a1|b1)
← (a3|b2 – a1|b2) - (a3|b1 – a1|b1)

# Categorical predictors with more than two levels

```
# Determine the linear fit (including all main effects and interactions):
fit <- lm(DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv, data=dat)
summary(fit)
```



```
Call:
lm(formula = DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-810.73 -279.12  -11.67  258.83 1022.47 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1785.97      40.30  44.317  < 2e-16 ***
a1a2        -763.07      98.71  -7.730  2.1e-11 ***
a1a3          94.27      98.71   0.955 0.342346    
Bdv          314.47      80.60   3.902 0.000192 ***
a1a2:Bdv     502.80     197.43   2.547 0.012697 *  
a1a3:Bdv     555.60     197.43   2.814 0.006090 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 382.3 on 84 degrees of freedom
Multiple R-squared:  0.5792,	Adjusted R-squared:  0.5541 
F-statistic: 23.12 on 5 and 84 DF,  p-value: 1.578e-14
```

**Not so nice:**

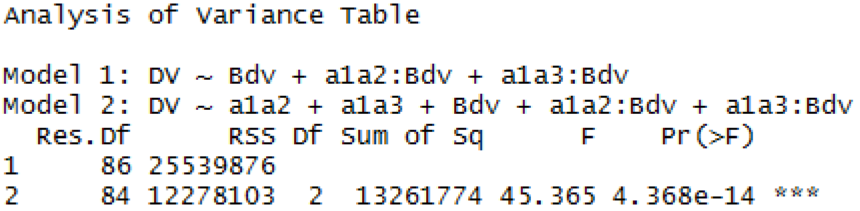
- We now have 2 coefficients for the main effect of A, and 2 coefficients for the A×B interaction
- *t*-values instead of *F*-values

# Categorical predictors with more than two levels

- How can we get ***F*-values** for
  - the overall main effect of **A** (3 levels => 2 degrees of freedom)
  - the overall main effect of **B** (2 levels => 1 degree of freedom)
  - the **A × B** interaction (=> (3-1) × (2-1) = 2 degrees of freedom)
- **The trick is to use `anova()` model comparisons**, testing a *model that excludes the parameters for a given effect of interest* against the *full model* (previously stored as "`fit`")

# Categorical predictors with more than two levels

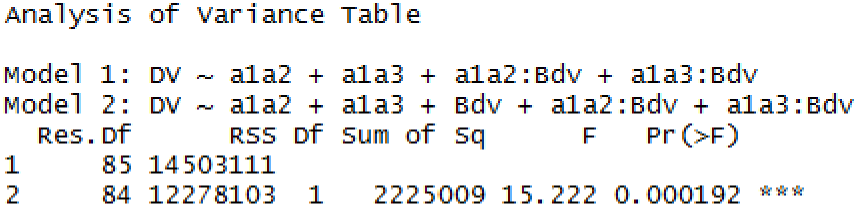
```r
# Determine the linear fit including all effects except those related to main
# effect of A
fit_no_A <- lm(DV ~ Bdv + a1a2:Bdv + a1a3:Bdv, data=dat)
# and compare with previous fit (including all effects)
anova(fit_no_A, fit)
```



```
Analysis of Variance Table

Model 1: DV ~ Bdv + a1a2:Bdv + a1a3:Bdv
Model 2: DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv
  Res.Df      RSS Df Sum of Sq      F    Pr(>F)    
1     86 25539876                                  
2     84 12278103  2  13261774 45.365 4.368e-14 ***
```

Main effect of **A** is significant at:
$F(2, 84) = 45.365$; $p < .001$

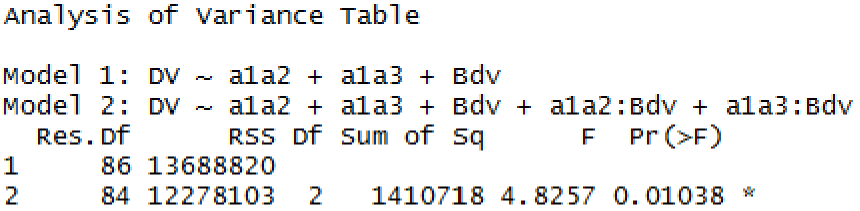
```r
# Do the same with main effect of B
fit_no_B <- lm(DV ~ a1a2 + a1a3 + a1a2:Bdv + a1a3:Bdv, data=dat)
anova(fit_no_B, fit)
```



```
Analysis of Variance Table

Model 1: DV ~ a1a2 + a1a3 + a1a2:Bdv + a1a3:Bdv
Model 2: DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv
  Res.Df      RSS Df Sum of Sq      F   Pr(>F)    
1     85 14503111                                 
2     84 12278103  1   2225009 15.222 0.000192 ***
```

Main effect of **B** is significant at:
$F(1, 84) = 15.222$; $p < .001$

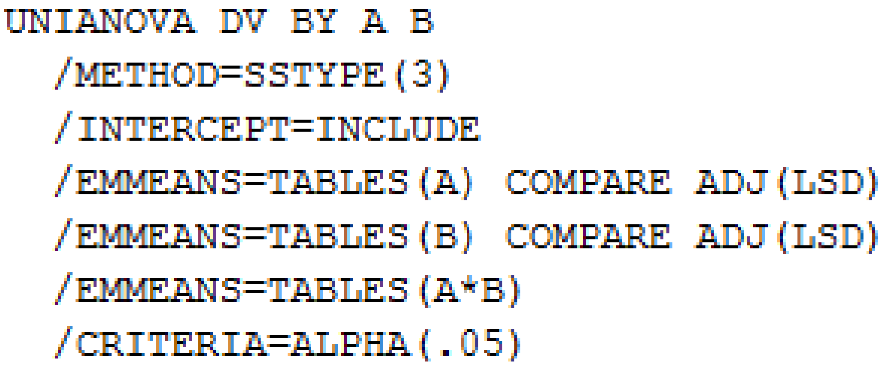
```r
# And finally, the interaction:
fit_no_AB <- lm(DV ~ a1a2 + a1a3 + Bdv, data=dat)
anova(fit_no_AB, fit)
```



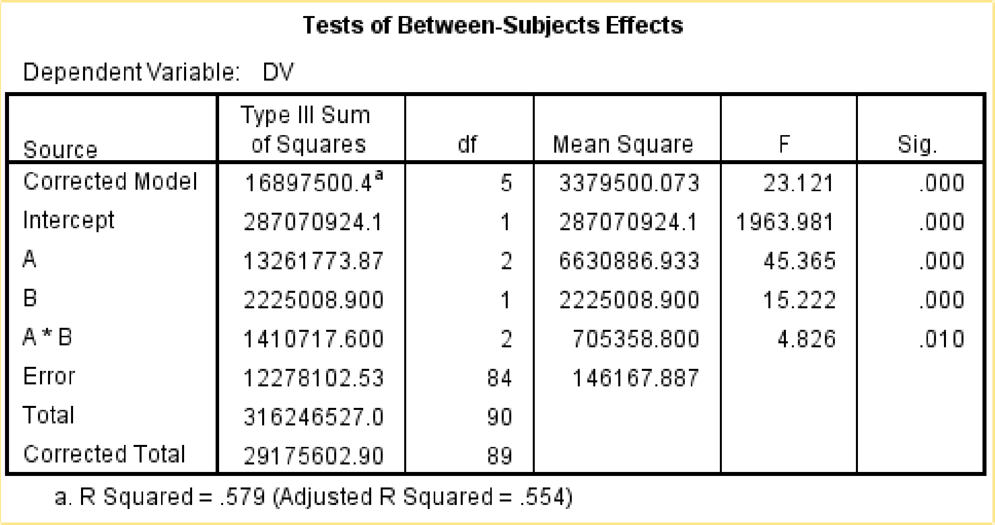
```
Analysis of Variance Table

Model 1: DV ~ a1a2 + a1a3 + Bdv
Model 2: DV ~ a1a2 + a1a3 + Bdv + a1a2:Bdv + a1a3:Bdv
  Res.Df      RSS Df Sum of Sq      F  Pr(>F)  
1     86 13688820                              
2     84 12278103  2   1410718 4.8257 0.01038 *
```

The **A×B** interaction is significant at:
$F(2, 84) = 4.826$; $p = .01$

# Just to confirm: The same in SPSS (yuk!)…



```
UNIANOVA DV BY A B
  /METHOD=SSTYPE(3)
  /INTERCEPT=INCLUDE
  /EMMEANS=TABLES(A) COMPARE ADJ(LSD)
  /EMMEANS=TABLES(B) COMPARE ADJ(LSD)
  /EMMEANS=TABLES(A*B)
  /CRITERIA=ALPHA(.05)
```

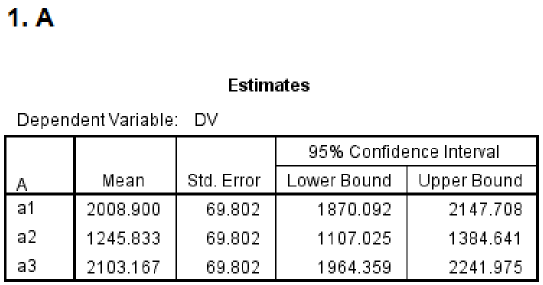


**Tests of Between-Subjects Effects**

Dependent Variable: DV

| Source | Type III Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|
| Corrected Model | 16897500.4[a] | 5 | 3379500.073 | 23.121 | .000 |
| Intercept | 287070924.1 | 1 | 287070924.1 | 1963.981 | .000 |
| A | 13261773.87 | 2 | 6630886.933 | 45.365 | .000 |
| B | 2225008.900 | 1 | 2225008.900 | 15.222 | .000 |
| A * B | 1410717.600 | 2 | 705358.800 | 4.826 | .010 |
| Error | 12278102.53 | 84 | 146167.887 | | |
| Total | 316246527.0 | 90 | | | |
| Corrected Total | 29175602.90 | 89 | | | |

a. R Squared = .579 (Adjusted R Squared = .554)

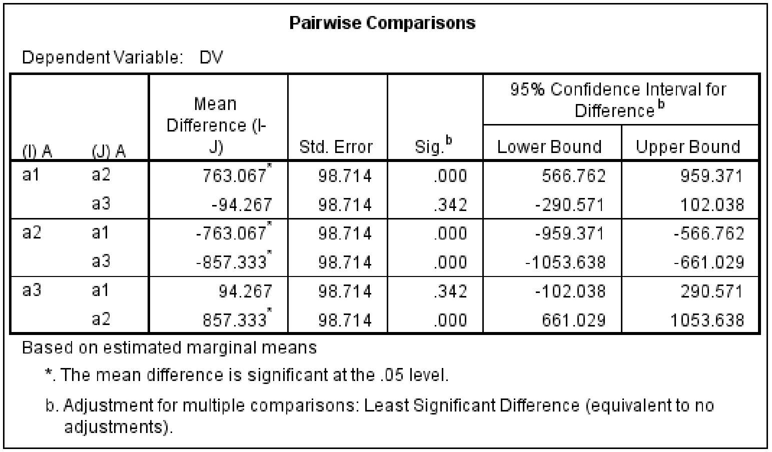


**1. A**

**Estimates**

Dependent Variable: DV

| A | Mean | Std. Error | 95% Confidence Interval Lower Bound | 95% Confidence Interval Upper Bound |
|---|---|---|---|---|
| a1 | 2008.900 | 69.802 | 1870.092 | 2147.708 |
| a2 | 1245.833 | 69.802 | 1107.025 | 1384.641 |
| a3 | 2103.167 | 69.802 | 1964.359 | 2241.975 |



**Pairwise Comparisons**

Dependent Variable: DV

| (I) A | (J) A | Mean Difference (I-J) | Std. Error | Sig.[b] | 95% Confidence Interval for Difference[b] Lower Bound | 95% Confidence Interval for Difference[b] Upper Bound |
|---|---|---|---|---|---|---|
| a1 | a2 | 763.067* | 98.714 | .000 | 566.762 | 959.371 |
| | a3 | -94.267 | 98.714 | .342 | -290.571 | 102.038 |
| a2 | a1 | -763.067* | 98.714 | .000 | -959.371 | -566.762 |
| | a3 | -857.333* | 98.714 | .000 | -1053.638 | -661.029 |
| a3 | a1 | 94.267 | 98.714 | .342 | -102.038 | 290.571 |
| | a2 | 857.333* | 98.714 | .000 | 661.029 | 1053.638 |

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

# Other categorical predictor coding schemes

- There are numerous other schemes for coding categorical predictors in regression analysis, see e.g. https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/
- Different coding schemes may be used to **test different linear hypotheses** on the data
- While we cannot go through all of them here, the example of *backward difference coding* will be explained in more detail

# Example: Backward difference coding

- Suppose the three levels of the categorical predictor **A** in our previous example data set had an *ordinal interpretation*: a1 < a2 < a3
- If so, it would make sense to use a coding scheme whereby ‘successive’ levels are incrementally compared with one another, i.e.
  - one contrast parameter encodes **a2 – a1**
  - the other contrast parameter encodes **a3 – a2**
- => *backward difference coding*

# Example: Backward difference coding

```
# Mean-centred backward difference coding of A:
dat$a1a2 <- scale(ifelse(dat$A=="a2" | dat$A=="a3",1,0), scale=FALSE)
dat$a2a3 <- scale(ifelse(dat$A=="a3",1,0), scale=FALSE)
# deviation coding of B, as before
dat$Bdv <- scale(ifelse(dat$B=="b2",1,0), scale=FALSE)
```

Resulting coding:

| *A* | *a1a2* | *a2a3* | Σ |
|---|---|---|---|
| *a1* | −0.66 | −0.33 | −0.99 |
| *a2* | 0.33 | −0.33 | 0.00 |
| *a3* | 0.33 | 0.66 | 0.99 |

| *B* | *Bdv* |
|---|---|
| *b1* | −0.5 |
| *b2* | 0.5 |

# Example: Backward difference coding

```
# Mean-centred backward difference coding of A:
dat$a1a2 <- scale(ifelse(dat$A=="a2" | dat$A=="a3",1,0), scale=FALSE)
dat$a2a3 <- scale(ifelse(dat$A=="a3",1,0), scale=FALSE)
# deviation coding of B, as before
dat$Bdv <- scale(ifelse(dat$B=="b2",1,0), scale=FALSE)
```

Resulting coding:

| *A* | *a1a2* | *a2a3* | Σ |
|---|---|---|---|
| *a1* | −0.66 | −0.33 | −0.99 |
| *a2* | 0.33 | −0.33 | 0.00 |
| *a3* | 0.33 | 0.66 | 0.99 |

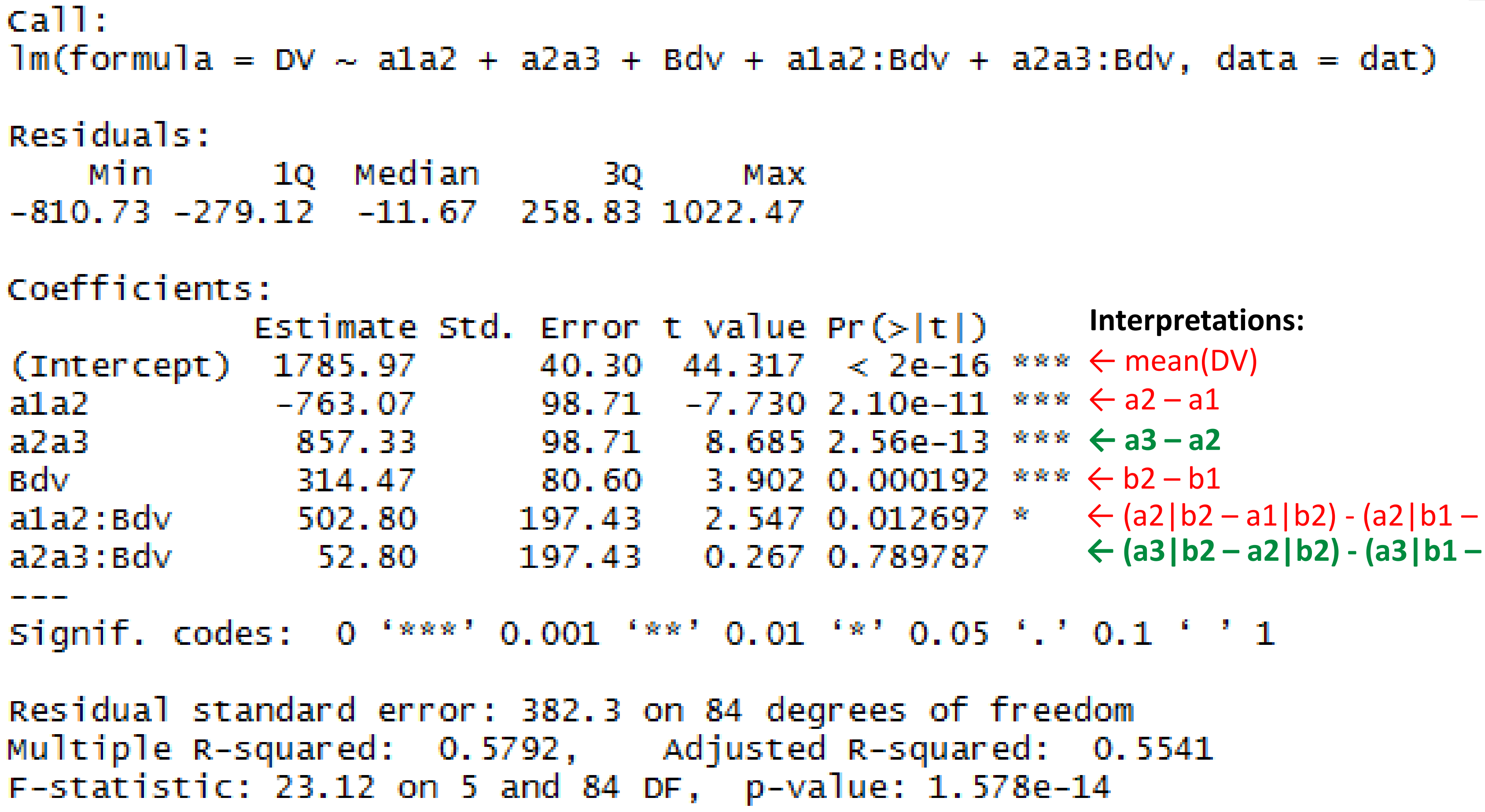
**Reference category**

$\overline{a2}-\overline{a1}$ $\overline{a3}-\overline{a2}$

| *B* | *Bdv* |
|---|---|
| *b1* | −0.5 |
| *b2* | 0.5 |

# Example: Backward difference coding

```
# Determine the linear fit (including all main effects and interactions):
fit2 <- lm(DV ~ a1a2 + a2a3 + Bdv + a1a2:Bdv + a2a3:Bdv, data=dat)
summary(fit2)
```

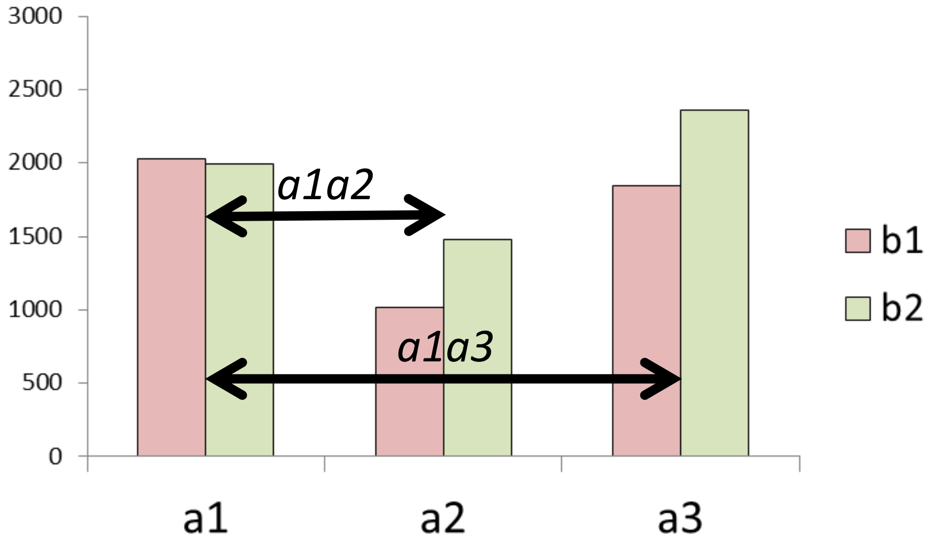


```
Call:
lm(formula = DV ~ a1a2 + a2a3 + Bdv + a1a2:Bdv + a2a3:Bdv, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-810.73 -279.12  -11.67  258.83 1022.47 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1785.97      40.30  44.317  < 2e-16 ***
a1a2         -763.07      98.71  -7.730 2.10e-11 ***
a2a3          857.33      98.71   8.685 2.56e-13 ***
Bdv           314.47      80.60   3.902 0.000192 ***
a1a2:Bdv      502.80     197.43   2.547 0.012697 *  
a2a3:Bdv       52.80     197.43   0.267 0.789787    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 382.3 on 84 degrees of freedom
Multiple R-squared:  0.5792,	Adjusted R-squared:  0.5541 
F-statistic: 23.12 on 5 and 84 DF,  p-value: 1.578e-14
```

**Interpretations:**

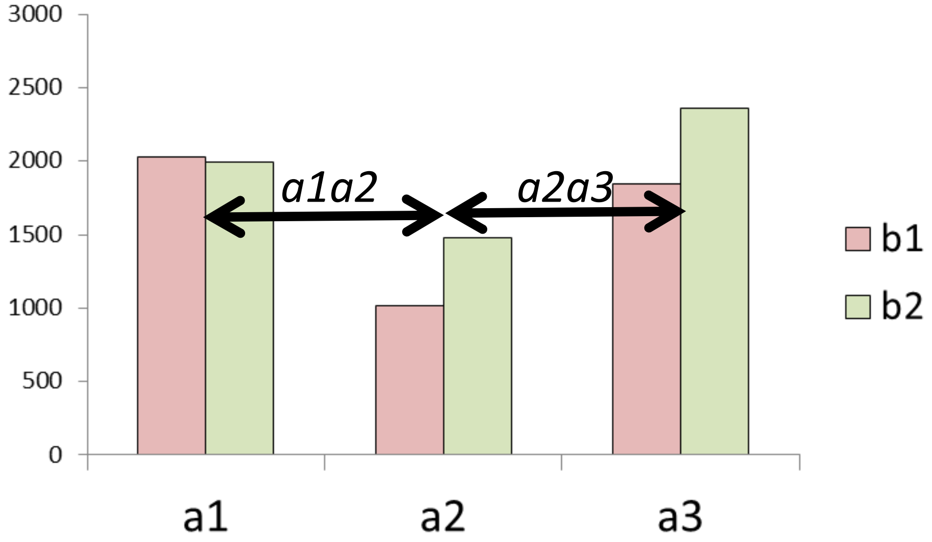
← mean(DV)
← a2 – a1
**← a3 – a2**
← b2 – b1
← (a2|b2 – a1|b2) - (a2|b1 – a1|b1)
**← (a3|b2 – a2|b2) - (a3|b1 – a2|b1)**

# What has changed here?...



**Previous coding of A:**

- *a2* is compared to *a1*
- *a3* is compared to *a1*

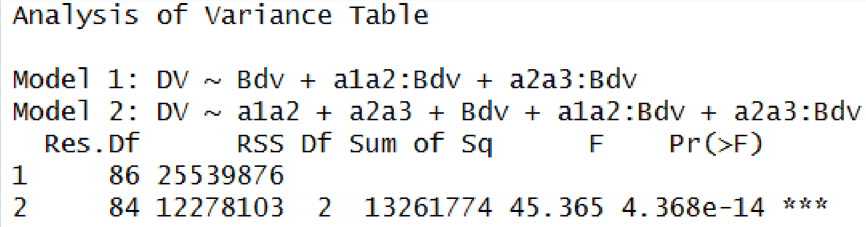


**Backward difference coding of A:**

- *a2* is compared to *a1*
- *a3* is compared to *a2*

# Does backward difference coding affect *overall* results? => Absolutely NOT!

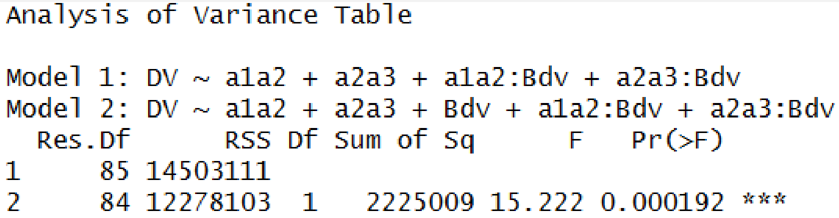
```
# Determine the linear fit including all effects except those related to main
# effect of A
fit_no_Ax <- lm(DV ~ Bdv + a1a2:Bdv + a2a3:Bdv, data=dat)
# and compare with previous fit2 (including all effects)
anova(fit_no_Ax, fit2)
```



```
Analysis of Variance Table

Model 1: DV ~ Bdv + a1a2:Bdv + a2a3:Bdv
Model 2: DV ~ a1a2 + a2a3 + Bdv + a1a2:Bdv + a2a3:Bdv
  Res.Df      RSS Df Sum of Sq      F    Pr(>F)    
1     86 25539876                                  
2     84 12278103  2  13261774 45.365 4.368e-14 ***
```

Main effect of **A** is significant at:
*F*(2, 84) = 45.365; *p* < .001

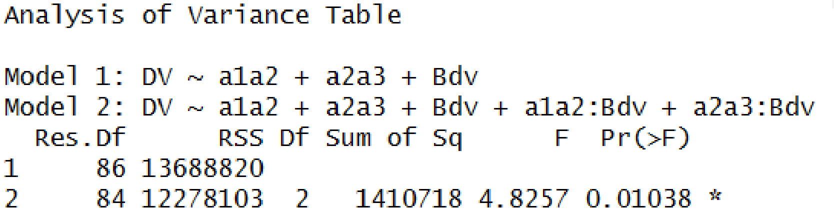
```
# Do the same with main effect of B
fit_no_Bx <- lm(DV ~ a1a2 + a2a3 + a1a2:Bdv + a2a3:Bdv, data=dat)
anova(fit_no_Bx, fit2)
```



```
Analysis of Variance Table

Model 1: DV ~ a1a2 + a2a3 + a1a2:Bdv + a2a3:Bdv
Model 2: DV ~ a1a2 + a2a3 + Bdv + a1a2:Bdv + a2a3:Bdv
  Res.Df      RSS Df Sum of Sq      F   Pr(>F)    
1     85 14503111                                 
2     84 12278103  1   2225009 15.222 0.000192 ***
```

Main effect of **B** is significant at:
*F*(1, 84) = 15.222; *p* < .001

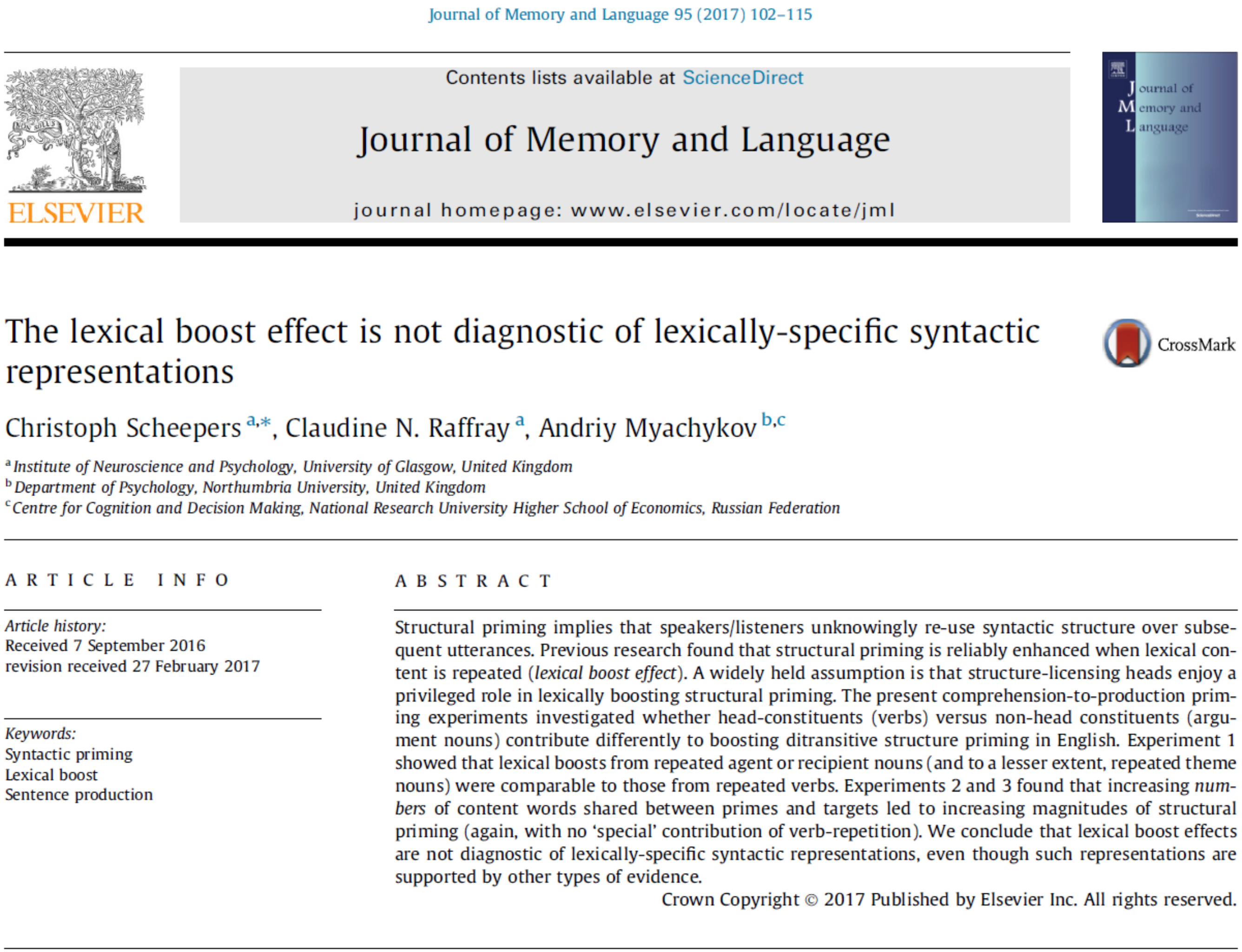
```
# And finally, the interaction:
fit_no_ABx <- lm(DV ~ a1a2 + a2a3 + Bdv, data=dat)
anova(fit_no_ABx, fit2)
```



```
Analysis of Variance Table

Model 1: DV ~ a1a2 + a2a3 + Bdv
Model 2: DV ~ a1a2 + a2a3 + Bdv + a1a2:Bdv + a2a3:Bdv
  Res.Df      RSS Df Sum of Sq      F  Pr(>F)  
1     86 13688820                              
2     84 12278103  2   1410718 4.8257 0.01038 *
```

The **A×B** interaction is significant at:
*F*(2, 84) = 4.826; *p* = .01

# Backward difference coding “in action”



Journal of Memory and Language 95 (2017) 102–115

Contents lists available at ScienceDirect

Journal of Memory and Language

journal homepage: www.elsevier.com/locate/jml

## The lexical boost effect is not diagnostic of lexically-specific syntactic representations

Christoph Scheepers [a,*], Claudine N. Raffray [a], Andriy Myachykov [b,c]

[a] *Institute of Neuroscience and Psychology, University of Glasgow, United Kingdom*
[b] *Department of Psychology, Northumbria University, United Kingdom*
[c] *Centre for Cognition and Decision Making, National Research University Higher School of Economics, Russian Federation*

ARTICLE INFO

*Article history:*
Received 7 September 2016
revision received 27 February 2017

*Keywords:*
Syntactic priming
Lexical boost
Sentence production

ABSTRACT

Structural priming implies that speakers/listeners unknowingly re-use syntactic structure over subsequent utterances. Previous research found that structural priming is reliably enhanced when lexical content is repeated (*lexical boost effect*). A widely held assumption is that structure-licensing heads enjoy a privileged role in lexically boosting structural priming. The present comprehension-to-production priming experiments investigated whether head-constituents (verbs) versus non-head constituents (argument nouns) contribute differently to boosting ditransitive structure priming in English. Experiment 1 showed that lexical boosts from repeated agent or recipient nouns (and to a lesser extent, repeated theme nouns) were comparable to those from repeated verbs. Experiments 2 and 3 found that increasing *numbers* of content words shared between primes and targets led to increasing magnitudes of structural priming (again, with no ‘special’ contribution of verb-repetition). We conclude that lexical boost effects are not diagnostic of lexically-specific syntactic representations, even though such representations are supported by other types of evidence.

- 3 Experiments, each with 5 × 2 design
- Binary logistic mixed effects models
- **Experiments 2 and 3 employ backward difference coding for (ordinal) 5-level predictor**
- Data and R-scripts available here: http://www.psy.gla.ac.uk/~christop/LexOverlap.zip

# You now (hopefully) understand

- The basics of linear regression
- How to perform linear regression using `lm()` in R
- How to specify designs with main effects and interactions
- How predictor coding affects parameter interpretation
- The importance of mean-centred predictor coding for omnibus analyses (=> main effects, interactions…)
- How to perform follow-up tests (simple effect analyses)
- How to derive *F*-statistics using `anova()` respectively `Anova()`

- All you've learnt here will be useful when moving on to *Generalized Linear Models* and *Generalized Linear Mixed Effects Models* in the following sessions