The General Linear Model

Multiple Regression and Model Criticism

Martin Corley

Today

Part I

A Linear Model

A Word-Naming Experiment

```
load(url('https://is.gd/refnet'))
ls()
## [1] "naming"
summary(naming)
##      length          freq        pos            RT
##  Min.   : 4   Min.   :   0   N:80   Min.   : 332
##  1st Qu.: 7   1st Qu.:   9   V:80   1st Qu.: 626
##  Median : 8   Median :  21   A:80   Median : 689
##  Mean   : 8   Mean   :  61          Mean   : 695
##  3rd Qu.: 9   3rd Qu.:  52          3rd Qu.: 770
##  Max.   :13   Max.   :1452          Max.   :1003
```

- `RT` = naming-aloud times (for 240 words)
- `length` in characters
- `freq` in wpm
- `pos`: `N`oun, `V`erb, or `A`djective

Several Equations To Start With

A General Model of Observed Data

$$\text{outcome}_i = (\text{model}) + \text{error}_i$$

$$\widehat{\text{outcomes}} = (\text{model})$$

Linear Model

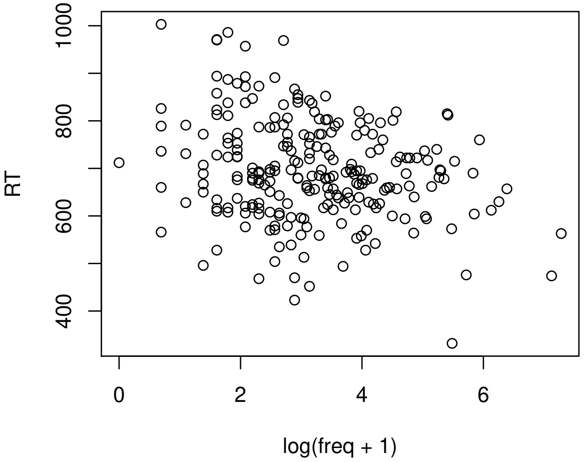
$$\hat{y}_i = b_0 \cdot 1 + b_1 \cdot x_i$$

```
y ~ 1 + x
RT ~ log(freq+1)
```

- we want estimates of b_0 (**intercept**) and b_1 (**slope**)

Begin By Inspecting the Data

```
with(naming, plot(RT ~ log(freq+1)))
```



A Simple Linear Model

```
model <- lm (RT ~ log(freq+1), data=naming)
summary(model)
```

```
## Call:
## lm(formula = RT ~ log(freq + 1), data = naming)
## ...
## Multiple R-squared:  0.0587,Adjusted R-squared:  0.0548
## F-statistic: 14.8 on 1 and 238 DF,  p-value: 0.00015
```

- R^2 and F are basic indicators of how ‘good’ a model is
- part of R’s output when summarising an `lm` object
- we’ll revisit adjusted R^2 later

A Simple Linear Model

```
summary(model)
```

```
## Call:
## lm(formula = RT ~ log(freq + 1), data = naming)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -316.9  -65.2   -6.1   70.4  263.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     759.87      18.04   42.13  < 2e-16 ***
## log(freq + 1)   -20.24       5.25   -3.85  0.00015 ***
## ...
```

- glancing at `Residuals` gives an indication of whether they are roughly symmetrically distributed
- the `Coefficients` give you the model
- the `Estimate` for `(intercept)` is b_0
- the `Estimate` for `log(freq + 1)` is b_1, the **slope**

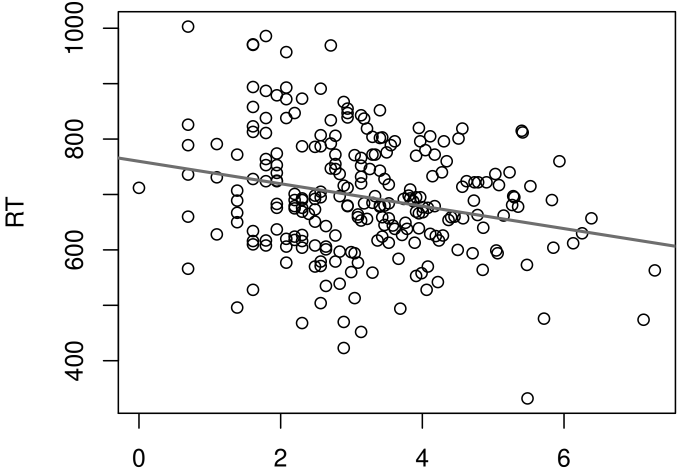
Coefficients

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        759.87      18.04   42.13  < 2e-16 ***
## log(freq + 1)      -20.24       5.25   -3.85  0.00015 ***
```

- *independently* of whether the model fit is ‘good’, coefficients can tell us about our data
- here, the `(Intercept)` b_0 isn't that useful
  - $\rightarrow$ it takes 760ms to name ‘zero-frequency words’
- but the slope b_1 of `log(freq + 1)` is quite informative
  - $\rightarrow$ words are named 20ms faster per unit increase
  - this is a significant finding
  - calculated from the estimated coefficient and its `Std. Error`, using the t distribution

Visualising the Model

```
with(naming,plot(RT ~ log(freq+1)))
abline(model,col='red',lwd=2)
```



Digression: R and Objects

- in `R`, *everything* is an **object** (a ‘thing’ with a name)
    - vectors, matrices, dataframes
    - functions, ...
- *functions* can take into account what kind of object they’re acting on

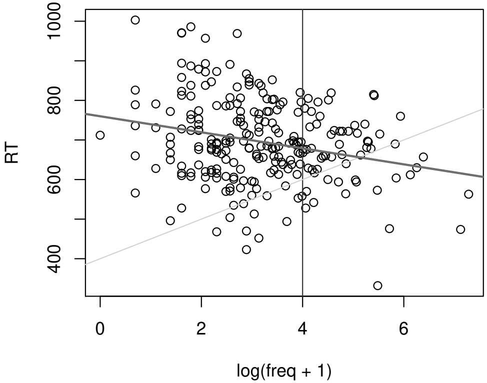
```
x <- 1:5 # numbers 1-5
y <- gl(5,1) # factor with 5 levels
summary(x)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       2       3       3       4       5

summary(y)

## 1 2 3 4 5
## 1 1 1 1 1
```

Digressing Further: abline()

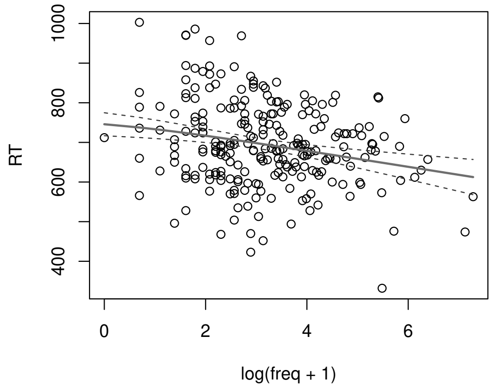
```
abline(v=4,col='blue')
```

```
abline(a=400,b=50,col='green') # intercept, slope
```

```
abline(model,col='red',lwd=2)
```

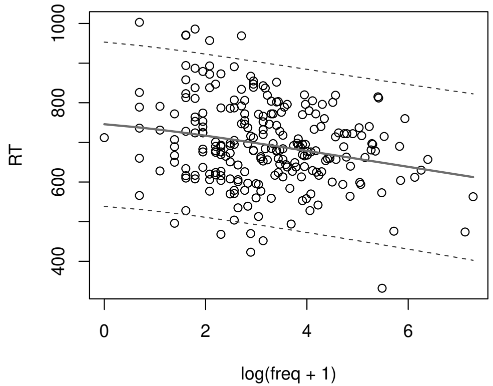


Visualisation (using `predict()`)



(confidence intervals for the *model*)

Visualisation (using `predict()`)



(confidence intervals for *predicted observations*)

Scaling of Predictors

- ‘words of zero frequency’ may not be very meaningful
- can **rescale** predictor to make interpretation more useful
- can also be used to ameliorate collinearity

```
model.S <- lm(RT ~ I(log (freq+1) - mean(log(freq+1))), data=naming)
summary(model.S)
```

```
## ...
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   695.38       6.72  103.41  < 2e-16 ***
## I(lf)         -20.24       5.25   -3.85  0.00015 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104 on 238 degrees of freedom
## Multiple R-squared:  0.0587,Adjusted R-squared:  0.0548
## F-statistic: 14.8 on 1 and 238 DF,  p-value: 0.00015
```

- slope unchanged
- 695ms corresponds to words of mean log frequency

Scaling of Predictors

- *linear* scaling of predictors doesn't change model fit

```
summary(model)$r.squared
## [1] 0.059
summary(model.S)$r.squared
## [1] 0.059
summary(lm(RT ~ I(5 * log(freq + 1)), data=naming))$r.squared
## [1] 0.059
```

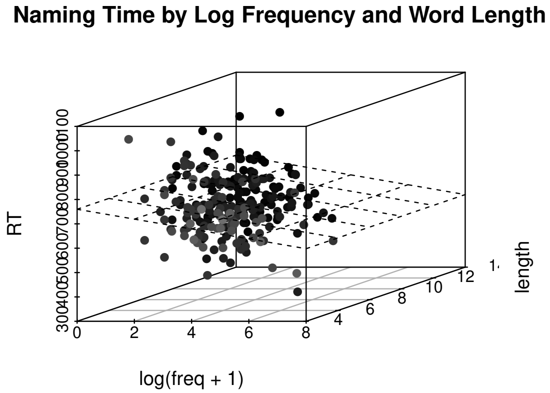
- *non-linear* scaling—like `log()` above—changes fit

```
summary(lm(RT ~ freq, data=naming))$r.squared
## [1] 0.044
```

Part II

Multiple Regression

Multiple Regression



- so far, have accounted for one predictor
- adding predictors increases the dimensionality of the model

Adding Predictors

- in multiple regression, R^2 measures the fit of the entire model
- sum of individual R^2s *if predictors not correlated*
- interpretation more tricky if predictors correlated

Specific Model for Multiple Regression

$$y_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + \ldots + b_n x_{ni} + \epsilon_i$$

- does word length have an effect on naming time (over and above frequency)?

```
model2 <- lm(RT ~ log(freq+1) + length,data=naming)
```

Comparing Models

- R^2 for `model` was .059
- R^2 for the new `model2` is .079 (from `summary(model2)`)
- does this mean that `model2` is better?

- *any* predictor will improve R^2 (chance associations guarantee this)

```
model3 <- lm(RT ~ log(freq+1) + runif(240),data = naming)
# add purely random predictor
summary(model3)
```

```
## ...
## Multiple R-squared:  0.0619,Adjusted R-squared:  0.054
## ...
```

- **adjusted** R^2 controls for additional predictors

Comparing Models

```
summary(model) # without length
```

```
## ...
## F-statistic: 14.8 on 1 and 238 DF,  p-value: 0.00015
## ...
```

```
summary(model2) # with length
```

```
## ...
## F-statistic: 10.2 on 2 and 237 DF,  p-value: 5.47e-05
## ...
```

- each model improves over *chance*, but do they successively improve over *each other*?

Comparing Models

```
anova(model2)
```

```
## Analysis of Variance Table
##
## Response: RT
##                 Df  Sum Sq Mean Sq F value  Pr(>F)
## log(freq + 1)    1  161118  161118   15.12 0.00013 ***
## length           1   56969   56969    5.35 0.02163 *
## Residuals      237 2525770   10657
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

NB *order* of predictors matters. . .

```
model2b <- lm(RT~length+log(freq+1),data=naming)
```

```
anova(model2b)
```

```
## ...
##                 Df  Sum Sq Mean Sq F value Pr(>F)
## length           1  113441  113441   10.64 0.0013 **
## log(freq + 1)    1  104646  104646    9.82 0.0019 **
## Residuals      237 2525770   10657
## ...
```

Type I vs. Type 3 SS

- order matters because R, by default, uses **Type I** sums of squares
  - calculate the improvement to the model caused by each successive predictor *in turn*
- compare to **Type III** sums of squares
  - calculate the improvement to the model caused by each predictor *taking all other predictors into account*
  - default for, e.g., SPSS

- huge debate about which is 'better'
- good arguments for Type I
- (nobody likes Type II)
- most important: be aware of the consequence. . .
- predictors should be entered into models in a theoretically-motivated order

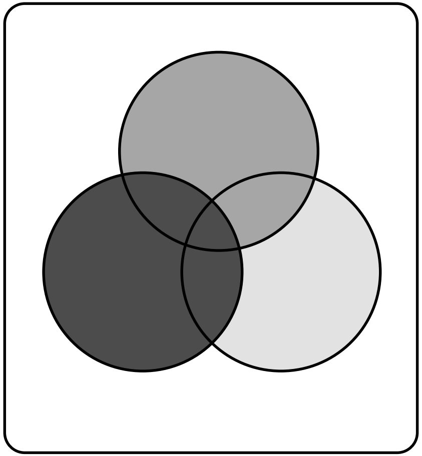
Type I vs. Type 3 SS

- order matters because R, by default, uses **Type I** sums of squares
  - calculate the improvement to the model caused by each successive predictor *in turn*
- compare to **Type III** sums of squares
  - calculate the improvement to the model caused by each predictor *taking all other predictors into account*
  - default for, e.g., SPSS

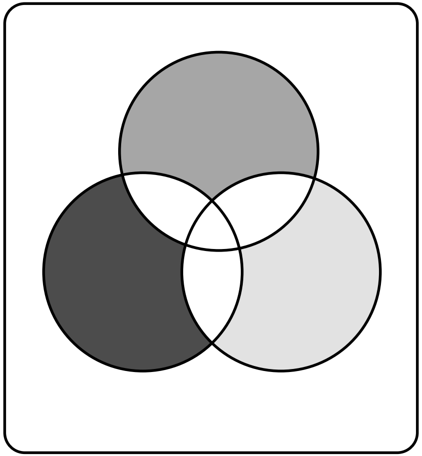
- huge debate about which is 'better'
- good arguments for Type I
- (nobody likes Type II)
- most important: be aware of the consequence. . .
- predictors should be entered into models in a theoretically-motivated order

Type 1 vs. Type 3 SS

Type I



Type III



Type III SS

- can easily get Type III-like output

```
drop1(model2,test='F')
```

```
## Single term deletions
##
## Model:
## RT ~ log(freq + 1) + length
##               Df Sum of Sq     RSS  AIC F value Pr(>F)
## <none>                     2525770 2229
## log(freq + 1)  1    104646 2630416 2236    9.82 0.0019 **
## length         1     56969 2582739 2232    5.35 0.0216 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Two-Predictor Model

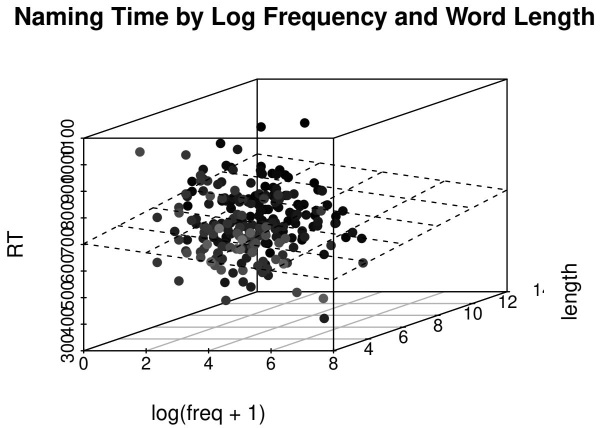
```
summary(model2)
```

```
## ...
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       656.5       48.2   13.63   <2e-16 ***
## log(freq + 1)    -16.9         5.4   -3.13   0.0019 **
## length            11.6         5.0    2.31   0.0216 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 103 on 237 degrees of freedom
## Multiple R-squared:  0.0795,Adjusted R-squared:  0.0717
## F-statistic: 10.2 on 2 and 237 DF,  p-value: 5.47e-05
```

- RT *decreases* by 17ms for every additional unit of log frequency
- RT *increases* by 12ms for every character of length
- model accounts for 8% of the variance

The Two-Predictor Model

```
library(scatterplot3d)
s3d <- with(naming,scatterplot(log(freq +1),length,RT))
s3d$plane3d(model2)
```



Part III

Model Criticism

Assumptions of Linear Models

Required

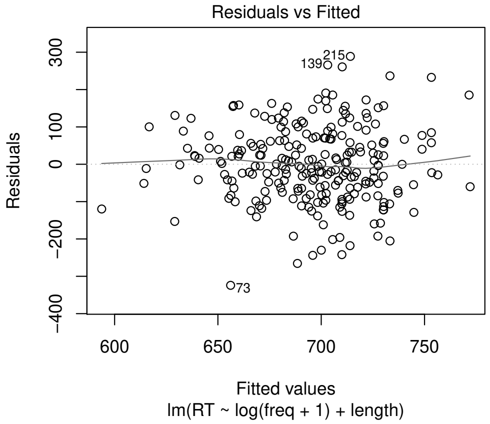
- **linearity** of relationships(!)
- for the *residuals*:
    - **normality**
    - **homogeneity of variance**
    - **independence**

Desirable

- uncorrelated predictors (no collinearity)
- no ‘bad’ (overly influential) observations

Linearity

```
plot(model2, which=1)
```

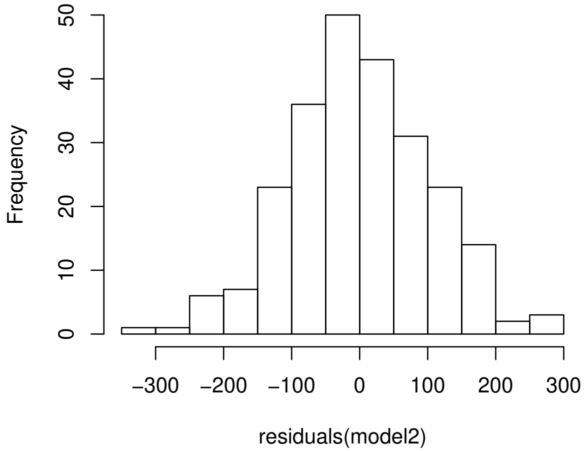


- plotting fitted values $\hat{y}_i$ against residuals ϵ_i
- the 'average residual' is roughly zero across $\hat{y}_i$, so relationship is likely to be linear

Normality of Residuals

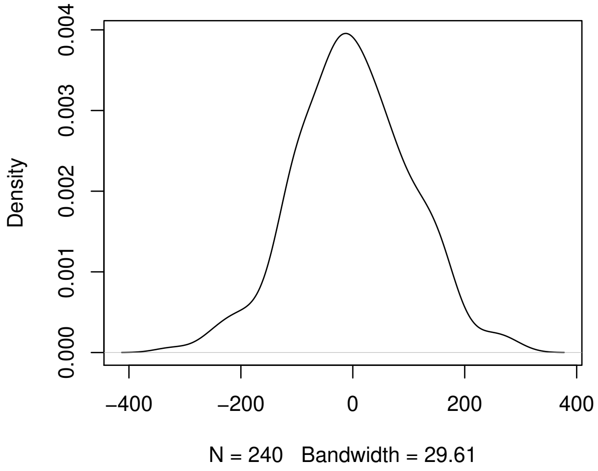
- simple assessments are often useful

```
hist( residuals(model2), main='', breaks=20)
```



Normality of Residuals

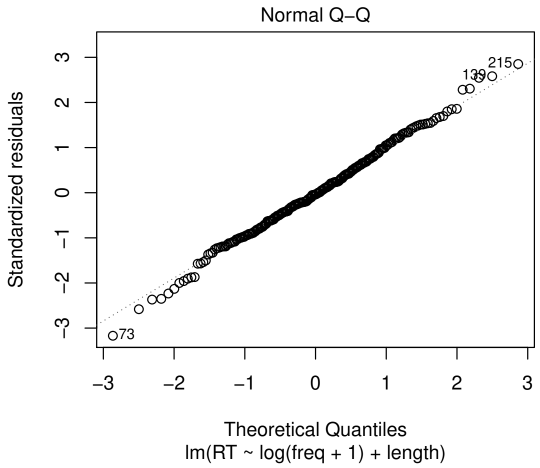
```
plot(density(residuals(model2)),main='')
```



Checking Assumptions
normality of residuals

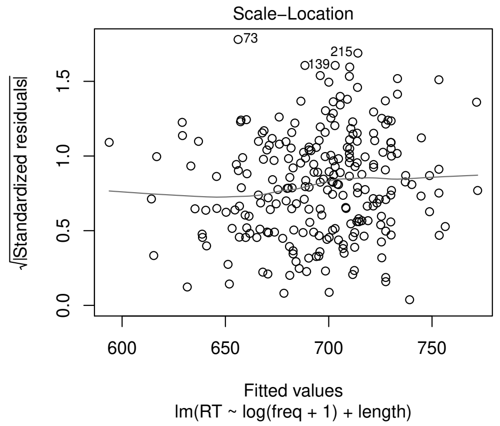
- a useful way to check *any* distribution is a QQ plot

```
plot(model2, which =2)
```



Homogeneity of Variance

```
plot(model2, which=3)
```



- shows $\sqrt{|\epsilon_i|}$ as a function of $\hat{y}_i$
- horizontal line suggests that variance is matched across $\hat{y}_i$

Independence

- no easy way to check **independence** of residuals
- in part, because it depends on the *source* of the observations

- one determinant might be a *person* observed multiple times
- e.g., my naming times might tend to be slower than yours

$\rightarrow$ repeated measures $\rightarrow \ldots \rightarrow$ mixed models

but meanwhile. . .

⏩ skip collinearity

Desirables

collinearity

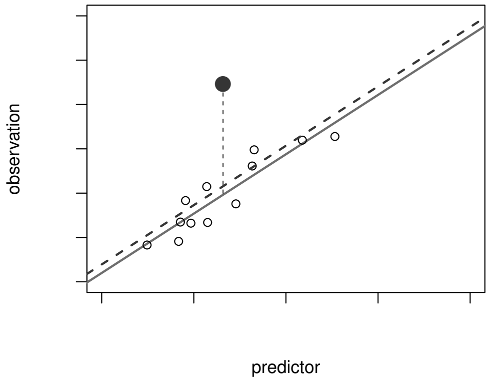
- correlated predictors widen the confidence interval (i.e., raise the SE of the coefficient)
- we can estimate how much using a calculation of **variance inflation factor (VIF)**
- calculated from R^2s of models using predictors to predict each other

```
library(car)
vif(model2)
```

```
## log(freq + 1)        length
##           1.1           1.1
```

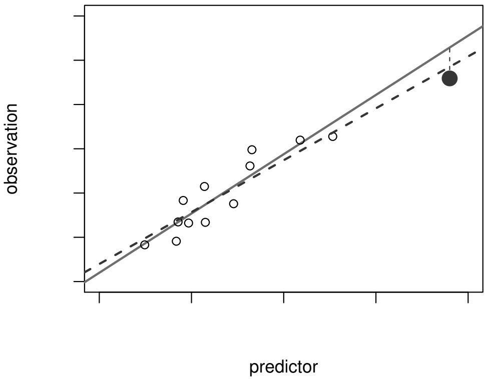
- $\sqrt{\text{VIF}}$ tells you how much the SE has been inflated
- $\sqrt{1.1} = 1.0$: no problem here!

Identifying 'Bad' Observations



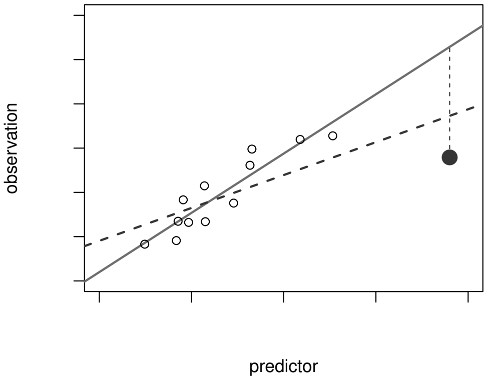
- **outliers** affect the intercept only
- the **studentised residual** is the difference between the observation and the regression without that observation

Identifying 'Bad' Observations



- observations with high **leverage** are inconsistent with other data, but may not be distorting the model

Identifying ‘Bad’ Observations



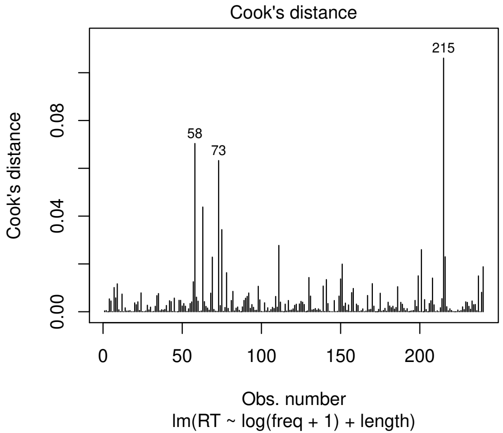
- what we care about most are observations with high **influence** (outliers with high leverage)

Desirables

identifying ‘bad’ observations

- one way of identifying observations with high influence is using **Cook’s distance**
- Cook’s distances over 1 are worth looking at

```
plot(model2, which=4)
```



Assumptions Violated

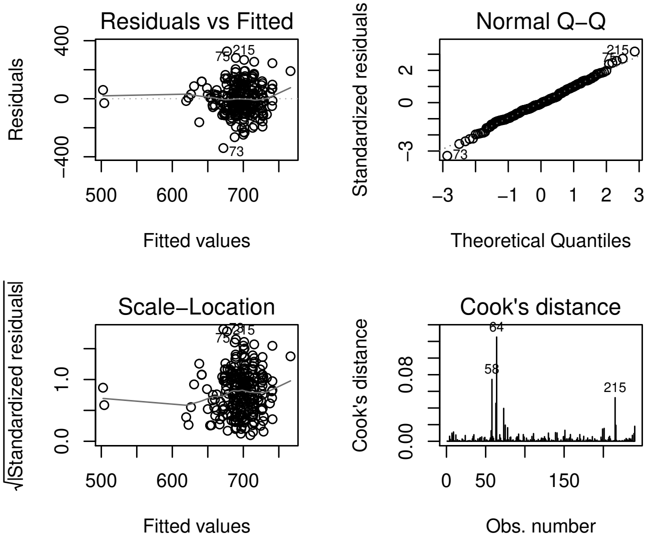
if we didn't that reaction times scaled with *log* frequency. . .

```
model2.B <- lm(RT ~ freq + length,data=naming)
summary(model2.B)
```

```
## ...
## Residuals:
##    Min     1Q Median     3Q    Max
## -339.9  -66.3   -6.0   66.2  326.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 599.7618    41.0356   14.62   <2e-16 ***
## freq         -0.1292     0.0475   -2.72   0.0070 **
## length       12.8954     4.9538    2.60   0.0098 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104 on 237 degrees of freedom
## Multiple R-squared:  0.0704,Adjusted R-squared:  0.0625
## F-statistic: 8.97 on 2 and 237 DF,  p-value: 0.000175
```

Assumptions Violated

```
par(mfrow=c(2,2))
plot(model2.B,which=c(1:4))
```


Residuals vs Fitted
Residuals
400
0
−400
500
600
700
Fitted values
215
75
73
Normal Q−Q
Standardized residuals
2
0
−3
−3
−1
0
1
2
3
Theoretical Quantiles
215
75
73
Scale−Location
√|Standardized residuals|
1.0
0.0
500
600
700
Fitted values
73
215
75
Cook's distance
Cook's distance
0.08
0.00
0
50
150
Obs. number
64
58
215