# The General Linear Model

## Standardization, Interactions, Effects Coding

Martin Corley

# Today

1. Standardisation
   - Recap of Last Week's Model
   - A Different Way of Scaling
2. Effects Coding
   - Introduction: Factors in Research
   - Dummy Coding
   - Orthogonal Coding
3. Interactions
   - An Example
   - Interaction is Multiplication
   - Interpreting Interactions

# Part I

# Standardisation

# Multiple Regression

## 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)
```

# Three Evaluations

## Is each model an improvement over the previous?

```
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
```

# Three Evaluations

## How much variance is explained?

```
summary(model2)
```

```
## ...
## 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
```

# Three Evaluations

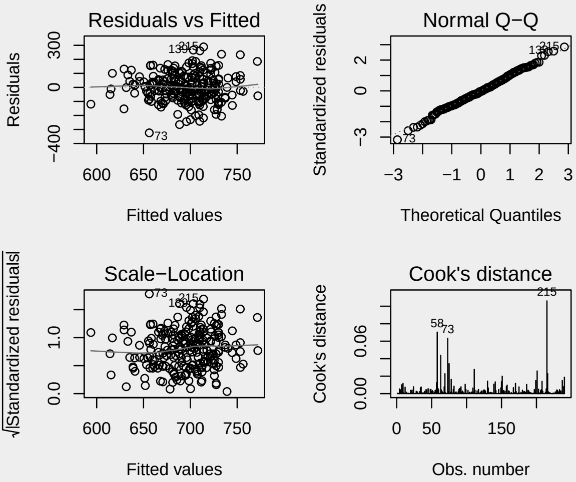
## What do the coefficients tell us?

```r
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
## ...
```

# ~~Three~~ Four Evaluations

## Do the model assumptions hold?



# Interpreting Coefficients

- we now know that:
  - for each additional unit of log frequency, naming time is 16.9 ms quicker
  - for each additional character length, naming time is 11.6 ms slower

- which is the 'more important' effect?
  - difficult to evaluate, because predictors are on different scales
  - how do we compare '1 unit log frequency' with '1 character'?

$\rightarrow$ **standardisation**

# Standardisation

- assuming that our predictors are roughly normal we can **scale** them using *z*-scores

$$z_i = \frac{x_i - \bar{x}}{\sigma_x}$$

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

# Scaled Model

- what is the $R^2$ of the new model?

```
summary(model2)$r.squared
## [1] 0.079
summary(model2.s)$r.squared
## [1] 0.079
```

- the new model has the *same fit* as the original
- coefficients are interpreted in terms of *1 sd change*

```
summary(model2.s)
```

```
## ...
## (Intercept)            9.68e-17   6.22e-02    0.00   1.0000
## scale(log(freq + 1)) -2.03e-01   6.46e-02   -3.13   0.0019 **
## scale(length)          1.49e-01   6.46e-02    2.31   0.0216 *
## ...
```

→ frequency has a bigger effect (but effects are small!)

# Lazy Scaling

- we can convert 'raw' coefficients $b$ to standardised coefficients $\beta$ without re-running the regression
- for predictor $x$ of outcome $y$:

$$\beta_x = b_x \times \frac{\sigma_x}{\sigma_y}$$

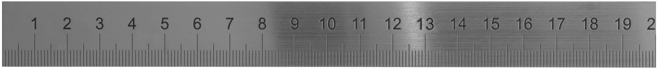
- or there's a function to do it for you:

```
require(lsr)

standardCoefs(model2)

##                  b  beta
## log(freq + 1) -17 -0.20
## length         12  0.15
```

# Part II

# Effects Coding

# Factors in Regression




- to date, we've been using continuous measures as predictors

→ **ratio**, **interval**, **ordinal** predictors

- in many analyses we want to compare *situations*

→ use **factors** → **nominal** predictors

# A Toy Example

## Which Apples are Tastiest?

```
apples <- data.frame(colour = gl(2, 2, 8, labels = c("green", "red")),
    taste = runif(8, 1, 7))
apples
```

```
##   colour taste
## 1  green   4.3
## 2  green   3.3
## 3    red   3.9
## 4    red   2.7
## 5  green   6.4
## 6  green   7.0
## 7    red   3.4
## 8    red   5.9
```

- which apples are the tastiest?

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

$$\text{taste}_i = (\text{some function of colour}) + \epsilon_i$$

# A Toy Example

## Quantifying a Nominal Predictor

```
apples$redness <- ifelse(apples$colour == "red", 1, 0)
apples
```

```
##   colour taste redness
## 1  green   4.3       0
## 2  green   3.3       0
## 3    red   3.9       1
## 4    red   2.7       1
## 5  green   6.4       0
## 6  green   7.0       0
## 7    red   3.4       1
## 8    red   5.9       1
```

- this maps to a linear model

$$\text{taste}_i = b_0 + b_1 \cdot \text{redness} + \epsilon_i$$

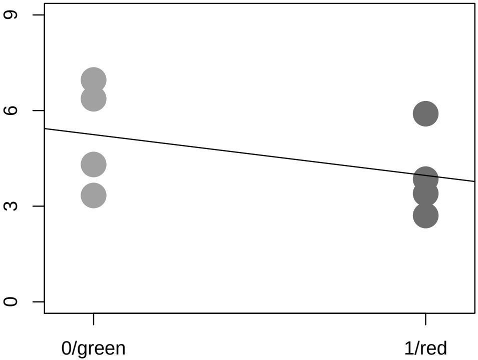
- $\overline{\text{taste}}$ for green apples = intercept
- 'change due to redness' = slope

# A Toy Example

## Of Little Practical Value

```
model <- lm(taste ~ redness, data = apples)
summary(model)
##
## Call:
## lm(formula = taste ~ redness, data = apples)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.907 -1.016 -0.341  1.274  1.938
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.245      0.774    6.77  0.00051 ***
## redness       -1.280      1.095   -1.17  0.28671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.6 on 6 degrees of freedom
## Multiple R-squared:  0.186,Adjusted R-squared:  0.0498
## F-statistic: 1.37 on 1 and 6 DF,  p-value: 0.287
```

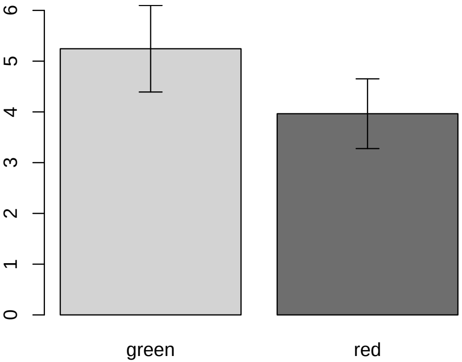
# A Toy Example




- there's no real difference between this and a linear model
- *except* that drawing a 'best fit line' is misleading

# A Toy Example

## A Better Presentation



# A Bigger Example

```
load(url("https://is.gd/refnet"))
# the next line doesn't have any statistical effect
naming <- naming[order(naming$RT), ]
head(naming)
```

```
##     length freq pos  RT
## 73       8  240   N 332
## 78       7   17   N 423
## 92       8   22   V 452
## 170      8    9   A 468
## 12       8   17   N 470
## 201      5 1239   A 474
```

- NB., we've reordered `naming` entirely so that we can see examples of all three levels of `pos` on the slide

# A Bigger Example

- we can discriminate `N`ouns from `A`djectives as before

```
naming$is.n <- ifelse(naming$pos == "N", 1, 0)
head(naming)
```

```
##     length freq pos  RT is.n
## 73       8  240   N 332    1
## 78       7   17   N 423    1
## 92       8   22   V 452    0
## 170      8    9   A 468    0
## 12       8   17   N 470    1
## 201      5 1239   A 474    0
```

- now, `is.n` is 1 for `N` and 0 for anything else
- but our predictor has *3* levels, so we need to repeat the trick

# A Bigger Example

## Dummy Coding for Three Levels

```
naming$is.v <- ifelse(naming$pos == "V", 1, 0)
head(naming)
```

```
##     length freq pos  RT is.n is.v
## 73       8  240   N 332    1    0
## 78       7   17   N 423    1    0
## 92       8   22   V 452    0    1
## 170      8    9   A 468    0    0
## 12       8   17   N 470    1    0
## 201      5 1239   A 474    0    0
```

- this maps to a linear model

$$\text{RT}_i = b_0 + b_1 \cdot \text{is.n} + b_2 \cdot \text{is.v} + \epsilon_i$$

- *individual* coefficients ($b_1$, $b_2$) for `N` and `V` (compared to `A`)

# A Bigger Example

```
model <- lm(RT ~ is.n + is.v, data = naming)
summary(model)
```

```
##
## Call:
## lm(formula = RT ~ is.n + is.v, data = naming)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -351.1  -65.3   -5.0   68.9  320.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  682.862     11.866   57.55   <2e-16 ***
## is.n           0.188     16.781    0.01    0.991
## is.v          37.375     16.781    2.23    0.027 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 106 on 237 degrees of freedom
## Multiple R-squared:  0.027,Adjusted R-squared:  0.0188
## F-statistic: 3.29 on 2 and 237 DF,  p-value: 0.039
```

# Summary So Far

## Categorical (Factorial) Predictors

- are treated *exactly* the same as continuous predictors
- all we do is assign some numbers to the various categories

- good news: `R` can assign the numbers for you
- bad news: there are many different ways of assigning numbers
  - so far, I've shown you **dummy** (or 'effect') coding
  - this is the default in `R`
- let's see how `R` does it. . .

# Coding Re-Explained

## Specific Model for 3-Level Factor

$$y_i = b_0 + b_1 f_1 + b_2 f_2 + \epsilon_i$$

| level | (cat) | $f_1$ | $f_2$ |
|---|---|---|---|
| 1 | `A` | 0 | 0 |
| 2 | `N` | 1 | 0 |
| 3 | `V` | 0 | 1 |

- coefficients for other levels give *difference from first level*

# Coding of Factors is Built In

```
contrasts(naming$pos)
```

```
##   N V
## A 0 0
## N 1 0
## V 0 1
```

- so these models are equivalent...

```
model.a <- lm(RT ~ is.n + is.v, data = naming)
model.b <- lm(RT ~ pos, data = naming)
```

```
summary(model.a)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  682.862     11.866   57.55   <2e-16 ***
## is.n           0.188     16.781    0.01    0.991
## is.v          37.375     16.781    2.23    0.027 *
```

```
summary(model.b)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  682.862     11.866   57.55   <2e-16 ***
## posN           0.188     16.781    0.01    0.991
## posV          37.375     16.781    2.23    0.027 *
```

# Interpreting Dummy Coding

```
summary(model.b)
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  682.862     11.866   57.55   <2e-16 ***
## is.n           0.188     16.781    0.01    0.991
## is.v          37.375     16.781    2.23    0.027 *
```

- compared to `A`:
  - `N` take the same time to name
  - `V` take 37ms longer to name

- important to note that the model improves on chance

```
## ...
## F-statistic: 3.29 on 2 and 237 DF,  p-value: 0.039
```

# Different Intercept

- by default, comparisons are made in *alphabetical order* of factor name
- what if we don't want to use `A` as the intercept?

```
model <- lm(RT ~ relevel(pos, "V"), data = naming)
summary(model)
```

```
## ...
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)           720.2       11.9   60.70   <2e-16 ***
## relevel(pos, "V")A    -37.4       16.8   -2.23    0.027 *
## relevel(pos, "V")N    -37.2       16.8   -2.22    0.028 *
```

- goodness-of-fit is *not affected* by changes to the intercept

```
## ...
## F-statistic: 3.29 on 2 and 237 DF,  p-value: 0.039
```

# Other Contrast Codings

- there may be a different contrast coding which better suits our research question
- for a predictor with $g$ levels (or ‘groups’), there are $g - 1$ possible contrasts
- these can be anything you like (there are a few built in to R)
- usefulness depends on your research question
- contrasts act like ‘tests of differences of interest’ once the model has been fit
- model fit is not affected by the choice of contrasts[1]

[1] for type 1 sums of squares

# Orthogonal Contrasts

needs 3 or more levels

- orthogonal contrasts are contrasts for which
  - each column sums to zero
  - the row products sum to zero
- this guarantees that no comparison is made twice (the variance is equally distributed between contrasts)

- orthogonal contrasts can be ‘hand-built’ to test questions of theoretical interest

# Orthogonal Contrasts

- research question: do `V` take longer to name than other words?
  - subquestion: do `N` differ from `A`?

```
contrasts(naming$pos) <- cbind('ANvV'=c(1/3,1/3,-2/3),
# compare A+N to V
                               'AvN'=c(1/2,-1/2,0))
# compare A to N
contrasts(naming$pos)
```

```
##    ANvV  AvN
## A  0.33  0.5
## N  0.33 -0.5
## V -0.67  0.0
```

- these contrasts are orthogonal

```
# columns sum to zero
colSums(contrasts(naming$pos))
```

```
## ANvV  AvN
##    0    0
```

```
# sum of row products is zero
sum(apply(contrasts(naming$pos), 1, prod))
```

```
## [1] 0
```

# Orthogonal Contrasts

```
# NB., the contrasts for pos have changed
model <- lm(RT ~ pos, data = naming)
summary(model)
```

```
## ...
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  695.383      6.851  101.50   <2e-16 ***
## posANvV      -37.281     14.533   -2.57    0.011 *
## posAvN        -0.187     16.781   -0.01    0.991
```

- model fit remains the same
- but coefficients have different interpretations

- `V` take, on average, 37ms longer to name than `N` and `A` combined
- no difference between `N` and `A`

# More Contrasts

- a whole bunch of contrasts specialised for different types of question
- `R` has functions to help build the more tricky ones
- `contr.helmert()`, `contr.poly()`, `contr.sum()`, etc.
- default is `contr.treatment()` (= 'dummy', = 'effects')

- using `contrasts()` is *always* equivalent to specifically setting up columns
- many seasoned `R` users do these interchangeably, depending on the situation

# Part III

# Interactions

# Forget Words

```
summary(coffee)
```

```
##        sc              fl           drink          SAT
##  Min.   : 69   Min.   : 10   coffee:50   Min.   :  0
##  1st Qu.: 84   1st Qu.: 37   tea   :50   1st Qu.: 17
##  Median : 98   Median : 60               Median : 24
##  Mean   : 98   Mean   : 59               Mean   : 37
##  3rd Qu.:108   3rd Qu.: 82               3rd Qu.: 56
##  Max.   :137   Max.   :100               Max.   :100
```

## A Coffee-Rating Study

- `sc` : self-confidence
- `fl` : strength of flavour
- `drink` : whether the judge is a habitual coffee or tea drinker
- `SAT` : satisfaction ratings for the coffee

# A Simple Model First

- let's assume that strength of flavour affects ratings
- . . . and also that self-confident people rate differently

```
model <- lm(SAT ~ fl + sc, data = coffee)
summary(model)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  67.8076    14.3886    4.71  8.2e-06 ***
## fl            0.4785     0.0769    6.22  1.2e-08 ***
## sc           -0.6074     0.1357   -4.48  2.1e-05 ***
```

- it appears that flavour improves satisfaction
- . . . but more self-confident people tend to be less satisfied
- what if that's not the whole story?

# Interactions

- self-confidence and flavour might *interact*
- that is, they might not be independent
  - for example, the more self-confident a person is, the more (or less) they might be affected by flavour

- **interaction terms** in linear models can be used to express these relationships
- interaction terms are, simply, **coefficients for products**

$$y_i = b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2 + \epsilon_i$$

# Our Example

$$\mathrm{SAT}_i = b_0 + b_1 \cdot \mathrm{fl}_i + b_2 \cdot \mathrm{sc} + b_3 \cdot (\mathrm{fl} \cdot \mathrm{sc}) + \epsilon_i$$

```
model <- lm(SAT ~ fl + sc + fl:sc, data = coffee)
anova(model)
```

```
## Analysis of Variance Table
##
## Response: SAT
##           Df Sum Sq Mean Sq F value  Pr(>F)
## fl         1  18012   18012    48.4 4.3e-10 ***
## sc         1   8566    8566    23.0 5.9e-06 ***
## fl:sc      1   5718    5718    15.3 0.00017 ***
## Residuals 96  35757     372
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Our Example

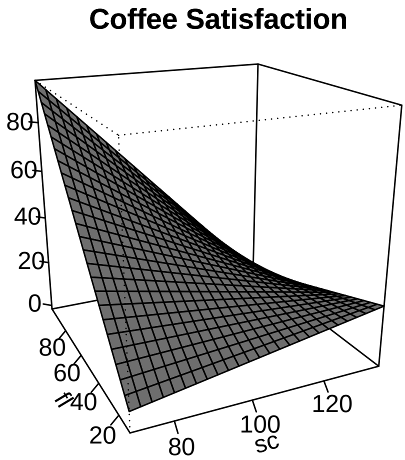
$$\mathrm{SAT}_i = b_0 + b_1 \cdot \mathrm{fl}_i + b_2 \cdot \mathrm{sc} + b_3 \cdot (\mathrm{fl} \cdot \mathrm{sc}) + \epsilon_i$$

```
summary(model)
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -34.26414   29.30875   -1.17  0.24527
## fl            2.32891    0.47769    4.88  4.3e-06 ***
## sc            0.44107    0.29607    1.49  0.13957
## fl:sc        -0.01908    0.00487   -3.92  0.00017 ***
```

- nutty intercept! We might rescale in real life...
- in general, ratings improve 2.3 units per unit increase in flavour
- there is no general effect of self-confidence
- the more self-confident a participant, the *less* they're influenced by flavour

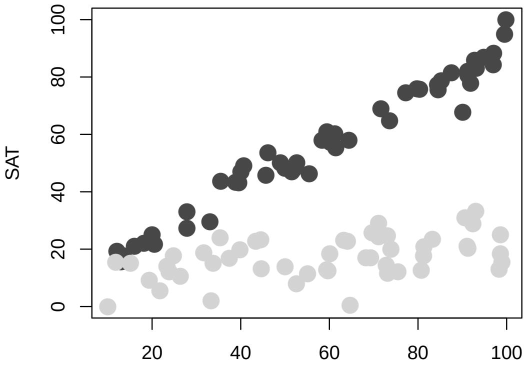
# What Does The Model Say?



# Forget Self-Confidence

- let's just plot the effects on flavour on satisfaction
- ... and colour the points by `drink`

```
with(coffee, plot(SAT ~ fl, pch = 16, cex = 2, col = ifelse(drink == "tea",
    "green", "brown")))
```



# Clearly, Tea Lovers Aren't Impressed

- the continuous variable `fc` *interacts with* the categorical variable `drink`
  - → you can't tell what influence flavour will have *unless* you know what the preferred drink is

```
model <- lm(SAT ~ fl + drink + fl:drink, data = coffee)
# or you could say 'model <- lm(SAT ~ fl*drink, data=coffee)'
anova(model)
```

```
## Analysis of Variance Table
##
## Response: SAT
##           Df Sum Sq Mean Sq F value Pr(>F)
## fl         1  18012   18012     598 <2e-16 ***
## drink      1  38609   38609    1283 <2e-16 ***
## fl:drink   1   8542    8542     284 <2e-16 ***
## Residuals 96   2890      30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Interpreting the Model

```
summary(model)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.0159     1.8162    4.41  2.7e-05 ***
## fl            0.8213     0.0279   29.42  < 2e-16 ***
## drinktea      1.1990     2.6428    0.45     0.65
## fl:drinktea  -0.6877     0.0408  -16.84  < 2e-16 ***
```

- if people drink *coffee*, the satisfaction increase per unit flavour is $0.8$
- if people drink *tea*, the satisfaction increase per unit flavour is 0.8 - 0.7 $= 0.1$

# The Model